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
«___8___» ___08___________2023 г.

Безопасность систем баз данных

Методические указания по выполнению курсового проекта для студентов направления подготовки 10.03.01«Информационная безопасность»

Курск 2023

УДК 004.773.5

Составители: А. Л. Марухленко

Рецензент

Кандидат технических наук, доцент кафедры
вычислительной техники А.В. Киселев

Безопасность систем баз данных: Методические указания по выполнению курсового проекта / Юго-Зап. гос. ун-т; сост.: А.Л. Марухленко. – Курск, 2023. – 164 с.: Библиогр.: с. 163.

Содержат сведения по вопросам проектирования баз данных. Указывается порядок выполнения курсового проекта и его содержание.

Предназначены для студентов направления подготовки 10.03.01«Информационная безопасность».

Текст печатается в авторской редакции
Подписано в печать . Формат 60x84 1/16.
Усл. печ.л. . Уч. –изд.л. .Тираж 50 экз. Заказ .
Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

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

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.

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 Ф.И.О. адрес	38 04 124674 Газаев С.Е.
Счет	тип номер баланс	расходный 367 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	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_	Пос	VARCHAR

	other	ле 9 не звонить.	(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 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

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 Сидирома.	VARCHAR (200)

			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	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

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.

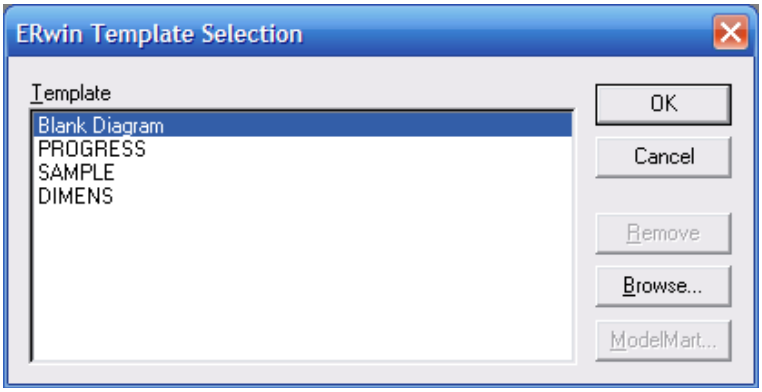


Рис. 1 - Окно создания новой диаграммы

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).

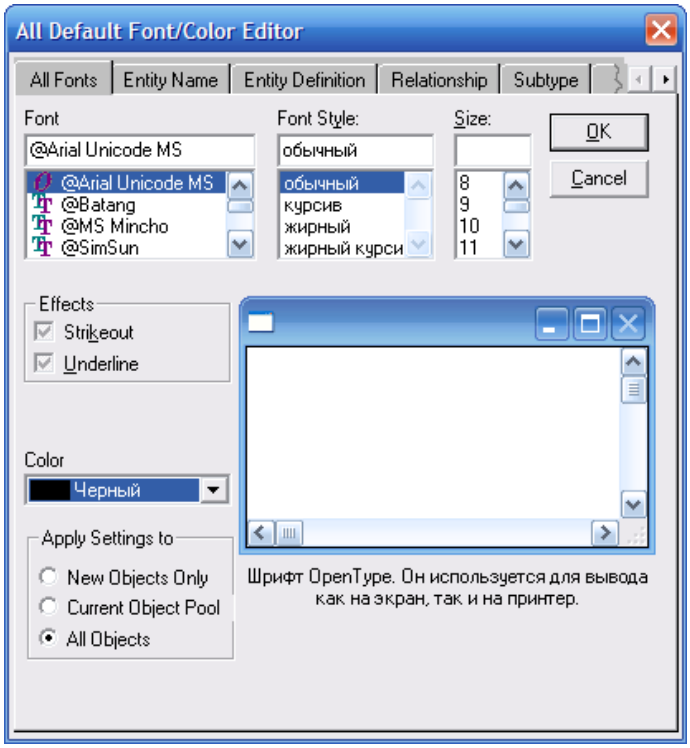


Рис. 2 - Окно настройки шрифта

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

В итоге получаем примерно такую физическую модель базы данных.

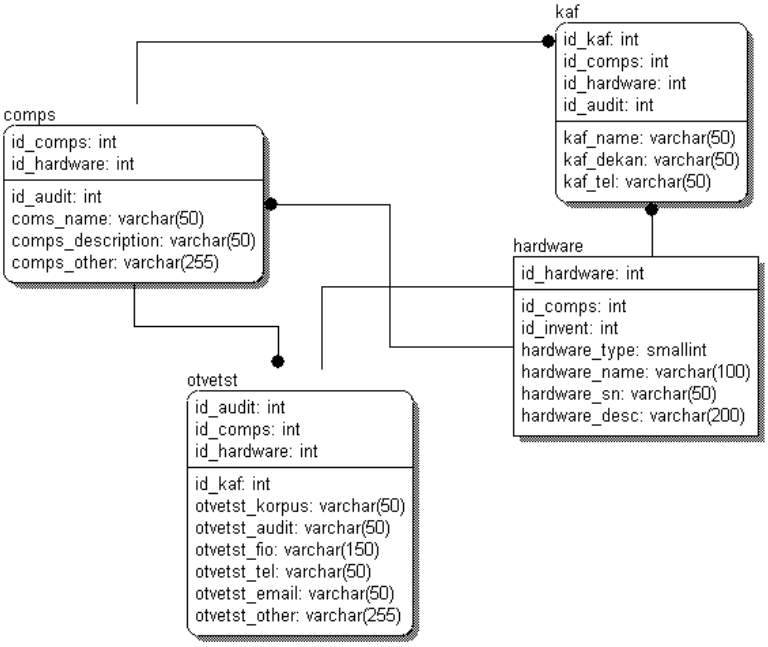


Рис. 3 - Физическая модель БД

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

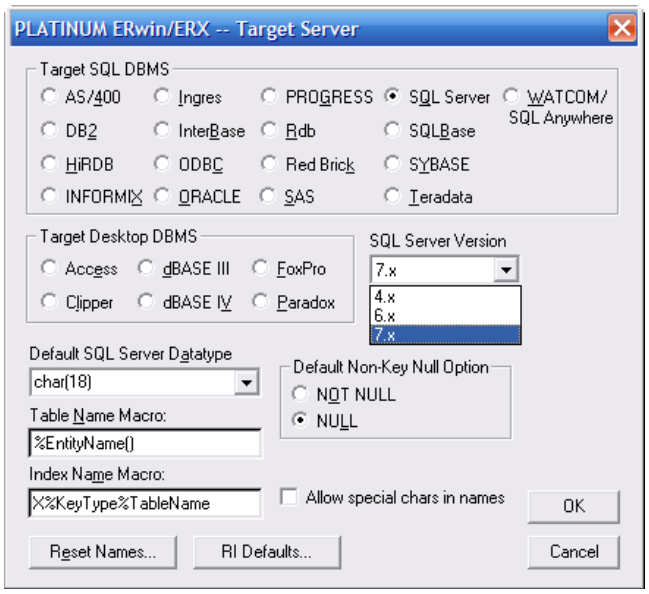
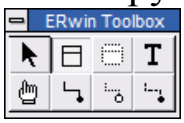


Рис. 4 - Окно выбора типа SQL Server

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.

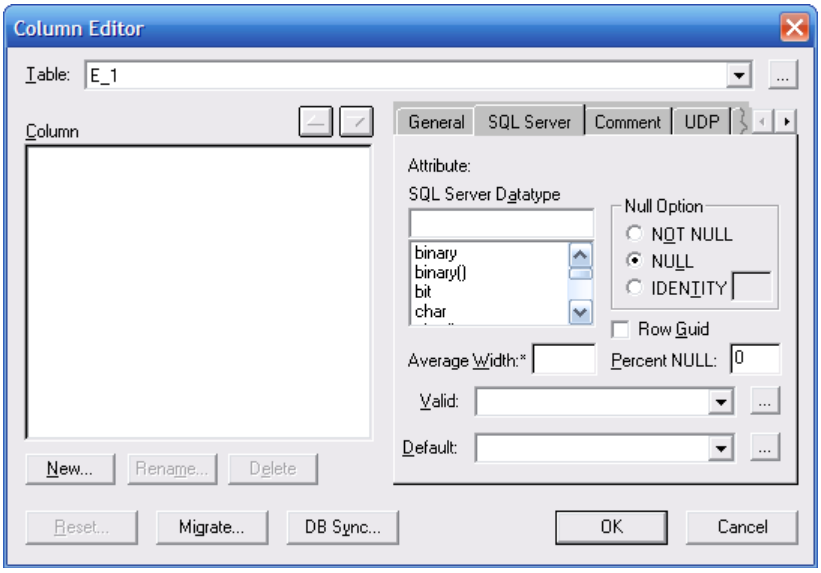


Рис. 5 - Окно добавления атрибута

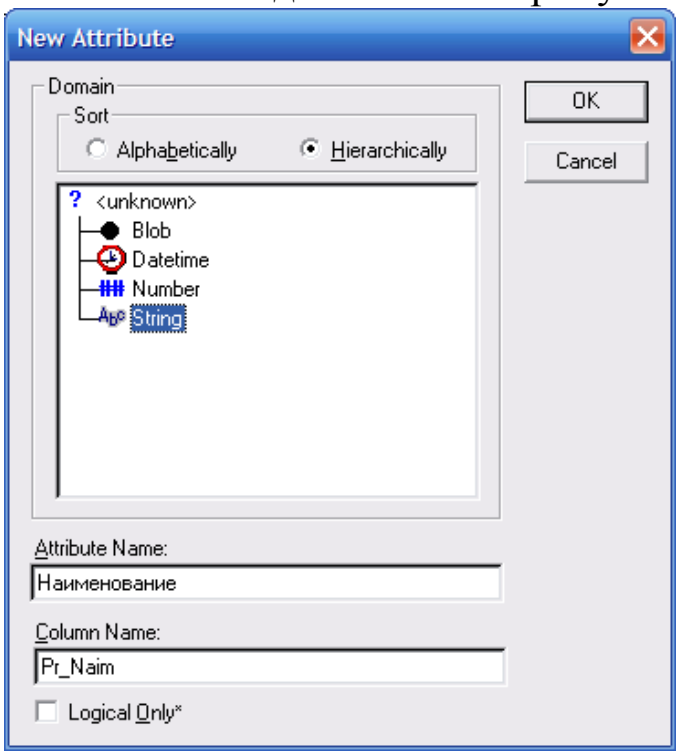


Рис. 6 - Окно свойств атрибута

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)

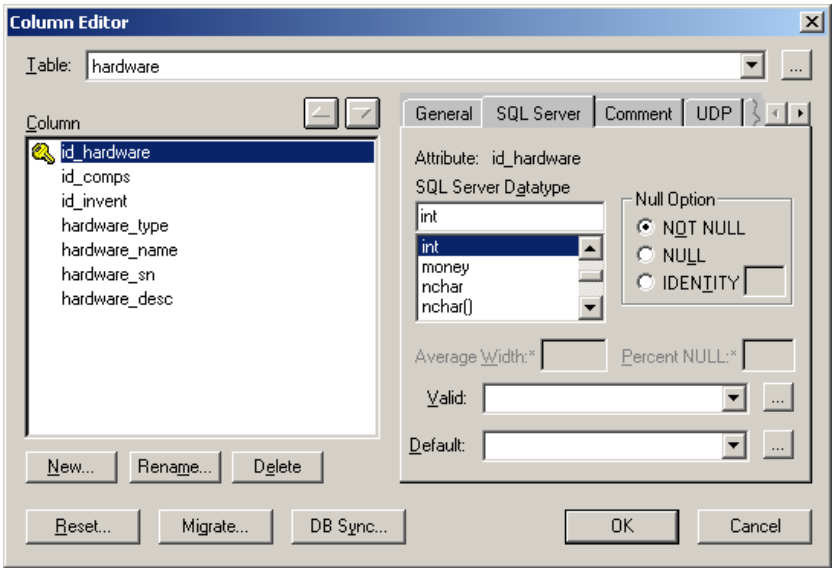


Рис. 7 - Окно переопределения типов

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

Default – Значение по умолчанию.

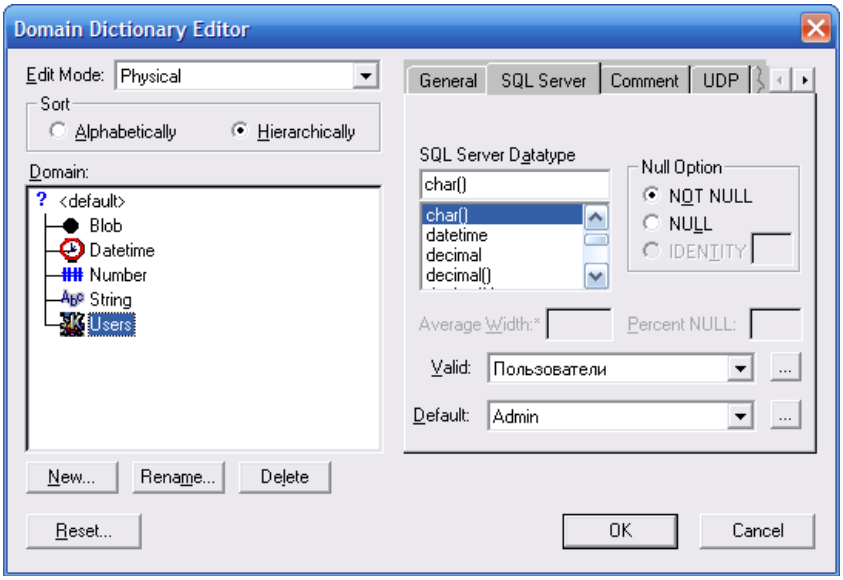


Рис. 8 - Выбор типа данных

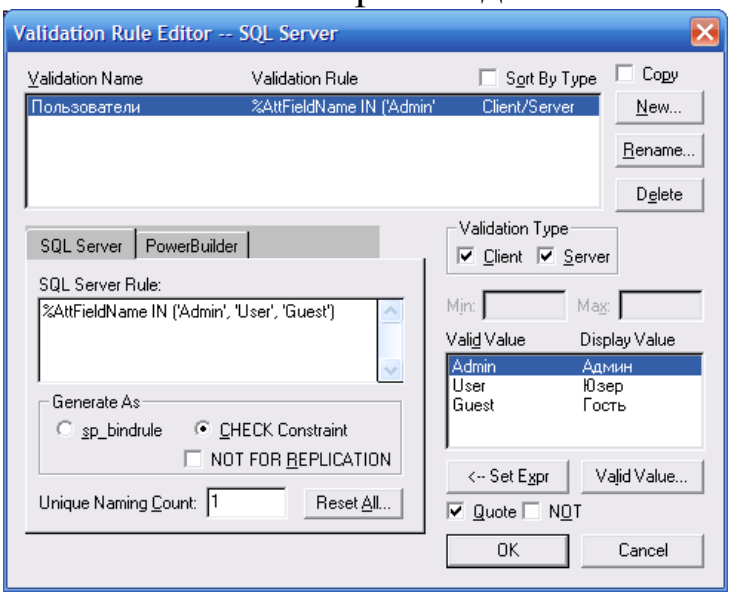


Рис. 9 - Назначение пользователей

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)

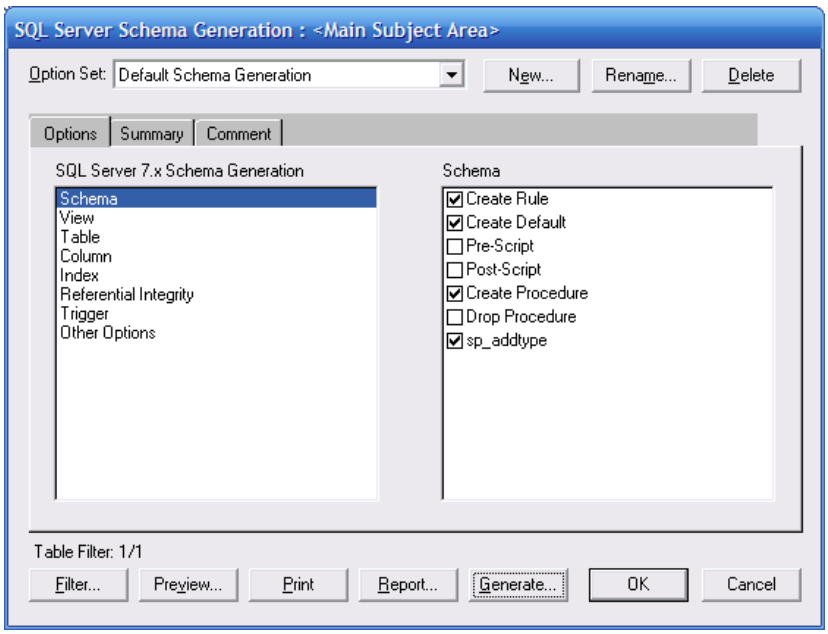


Рис. 10 - Окно настройки схемы.

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)

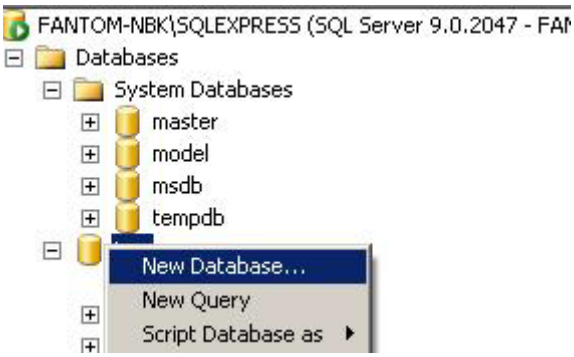


Рис. 11 - Обозреватель объектов sql

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)

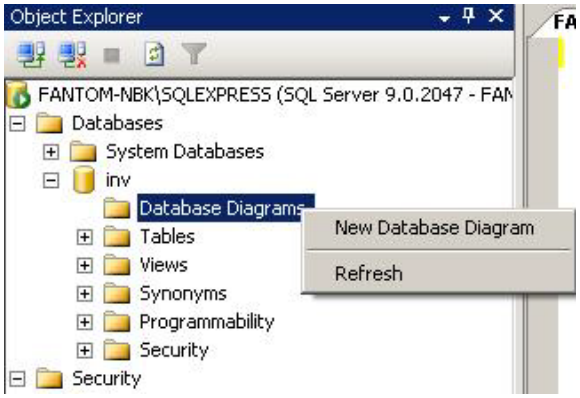


Рис. 12 - Создание диаграммы БД

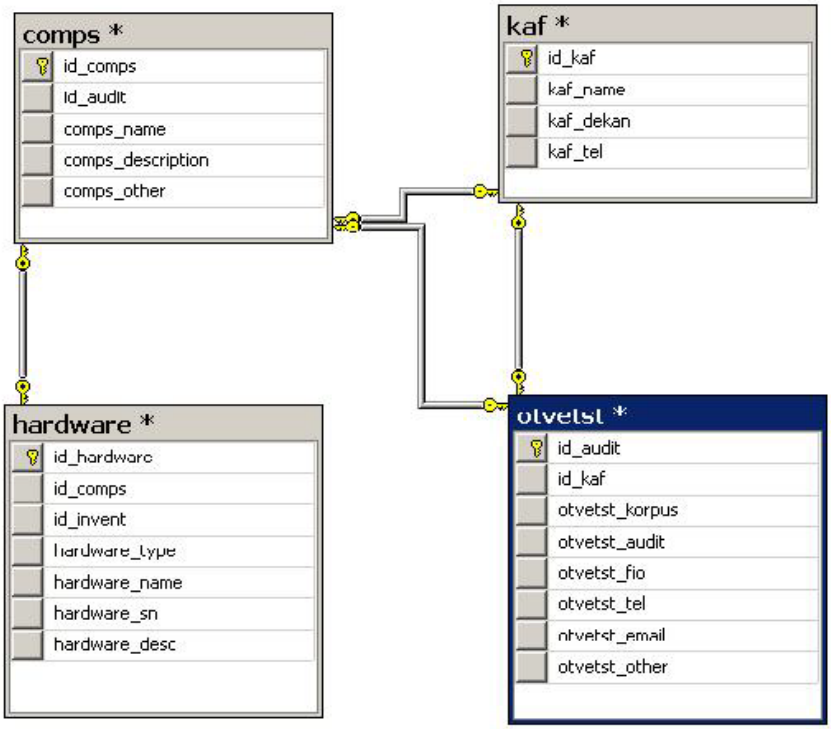


Рис. 13 - Диаграмма полученной БД

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)

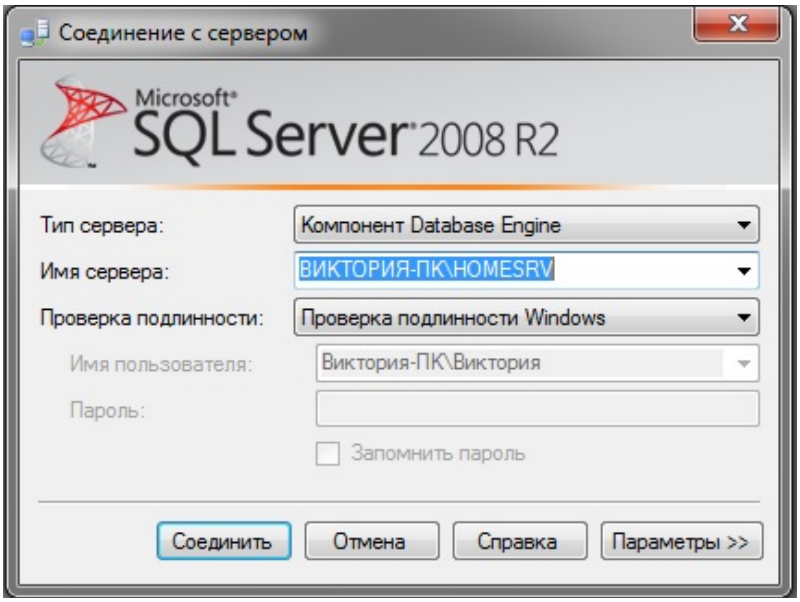


Рис. 14 - Окно подключения к «Среда SQL Server Management Studio»

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)

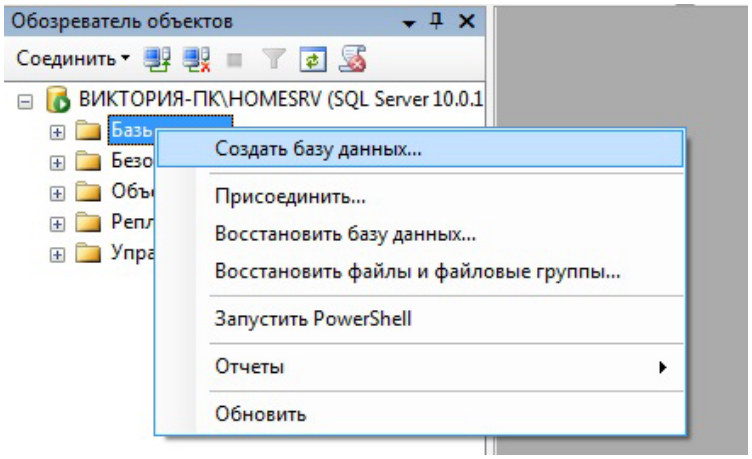


Рис. 15 - Создание новой БД

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)

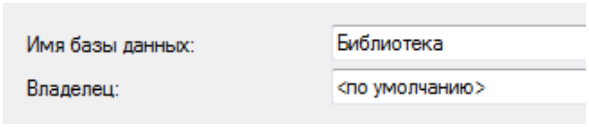


Рис. 16 - Окно свойств БД

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

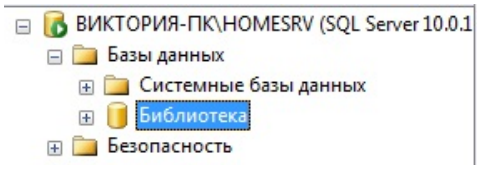


Рис. 17 - БД в обозревателе компонентов

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)

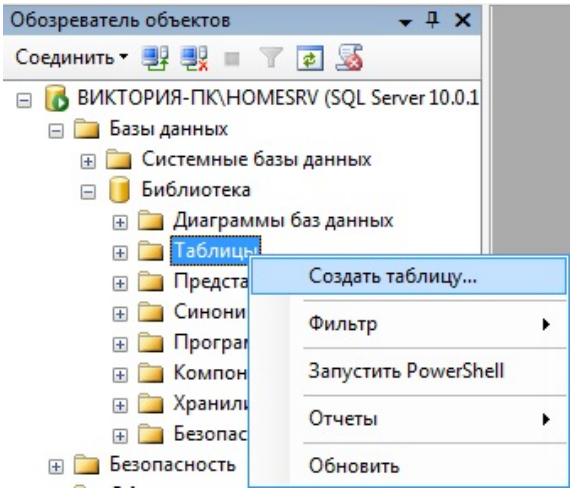


Рис. 18 - Порядок создание таблиц

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

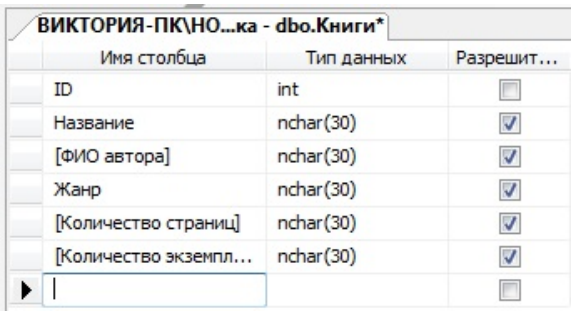


Рис. 19. Создание столбцов и их свойства

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)

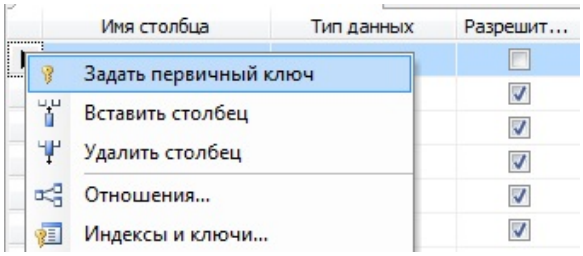


Рис. 20 - Задание первичного ключа

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

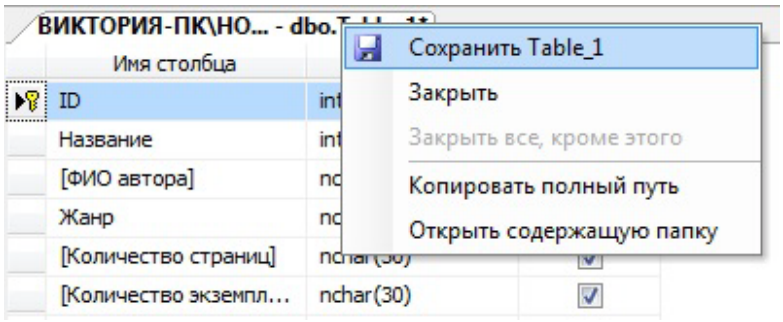


Рис. 21 - Сохранение таблицы

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

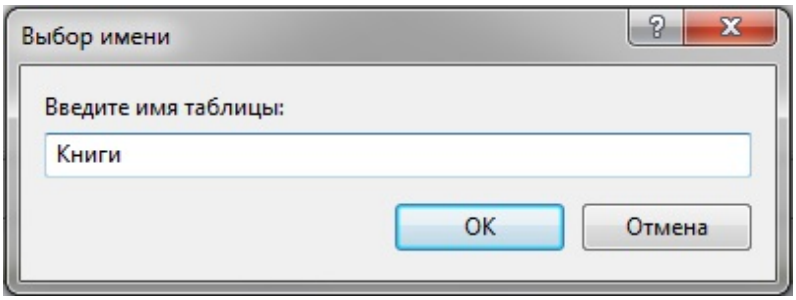


Рис. 22 - Окно Выбор имени таблицы

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)

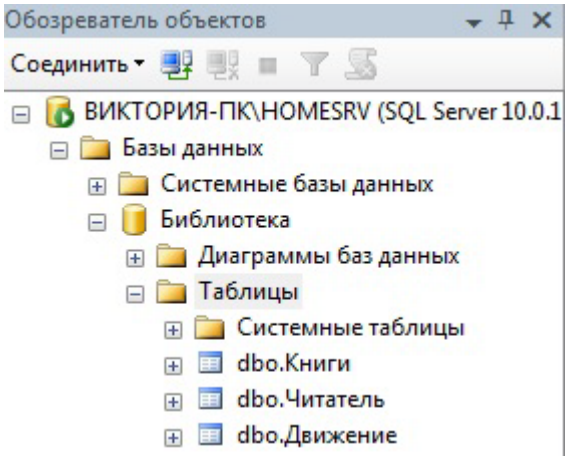


Рис. 23 - Отображение таблиц в обозревателе компонентов

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)

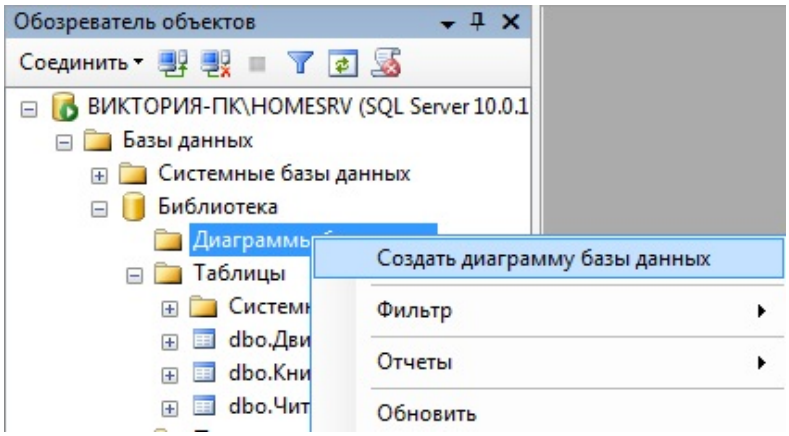


Рис. 24 - Порядок создания диаграммы

2. В диалоговом окне выберите нужные таблицы и нажмите OK.
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 OK (Рис. 25)

Важно, чтобы тип данных связанных столбцов совпадал.

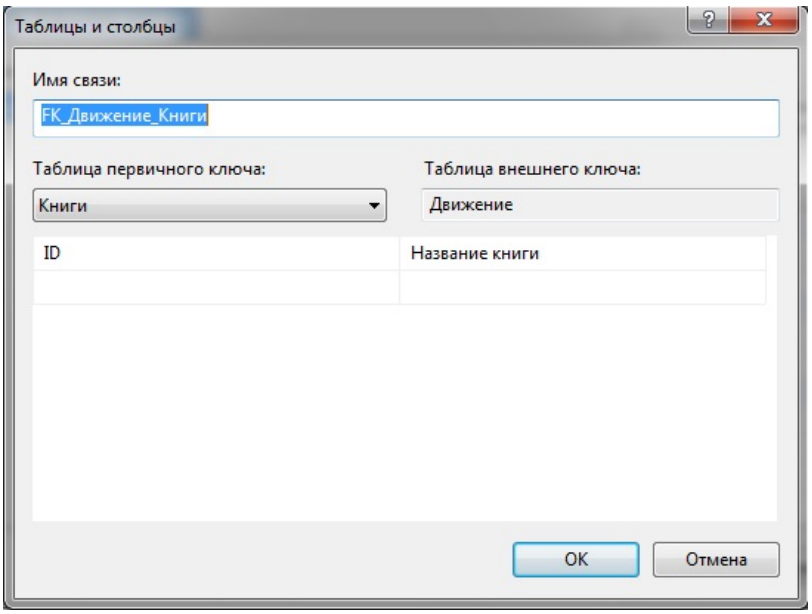


Рис. 25 - Связь между таблицами

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

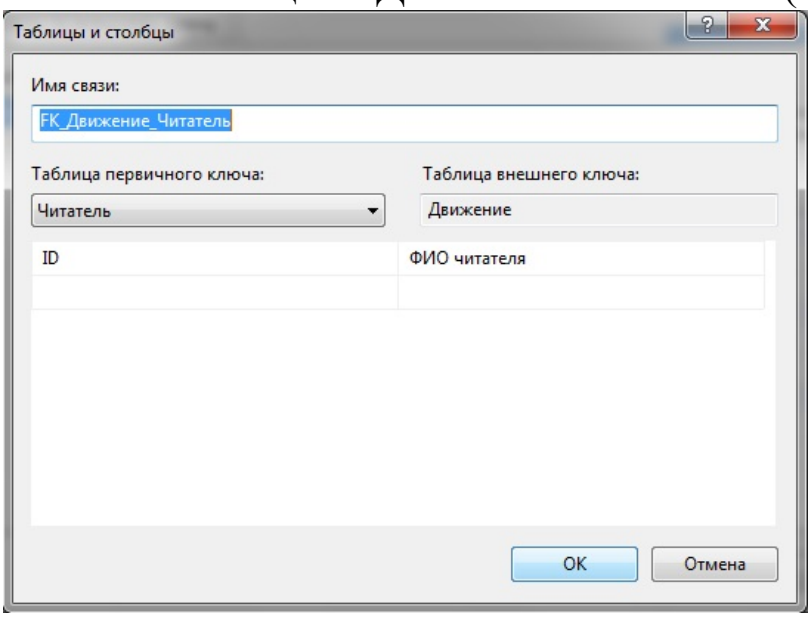


Рис. 26 - Связь между таблицами

Таким образом диаграмма примет следующий вид (Рис. 27)

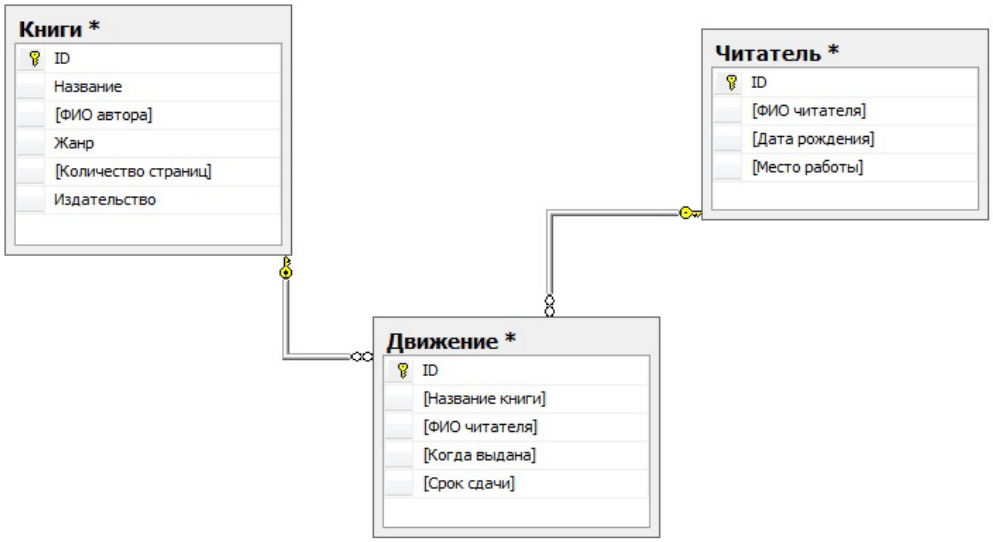


Рис. 27 - Диаграмма базы данных

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

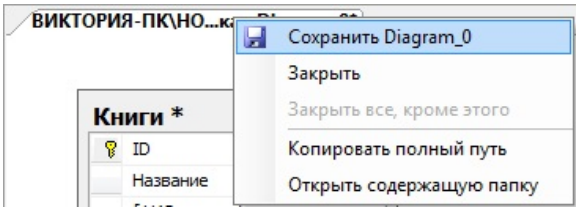


Рис. 28 - Окно сохранения диаграммы БД

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

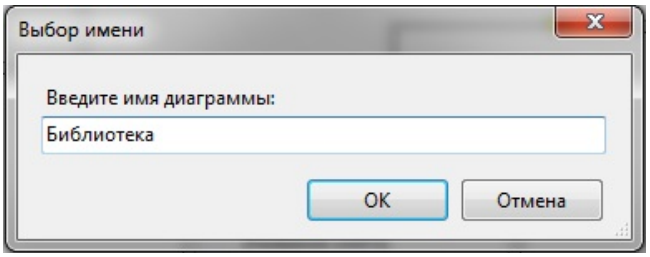


Рис. 29 - Окно «Выбор имени диаграммы»

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

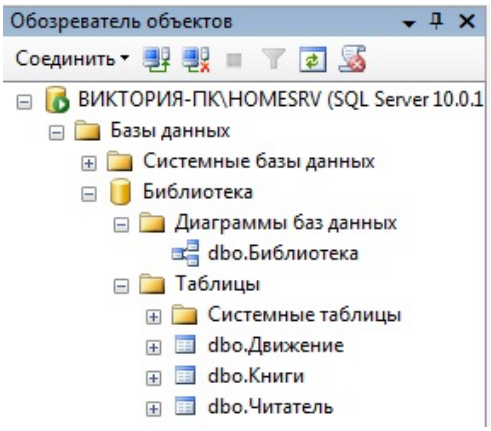


Рис. 30. Отображение диаграммы в обозревателе компонентов

Переходим к созданию базы данных с помощью Transact-SQL.

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

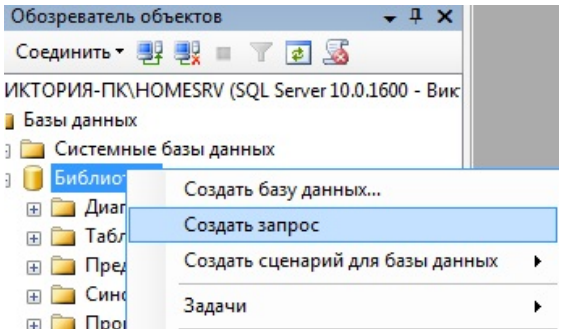


Рис. 31 - Порядок создания запроса

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)

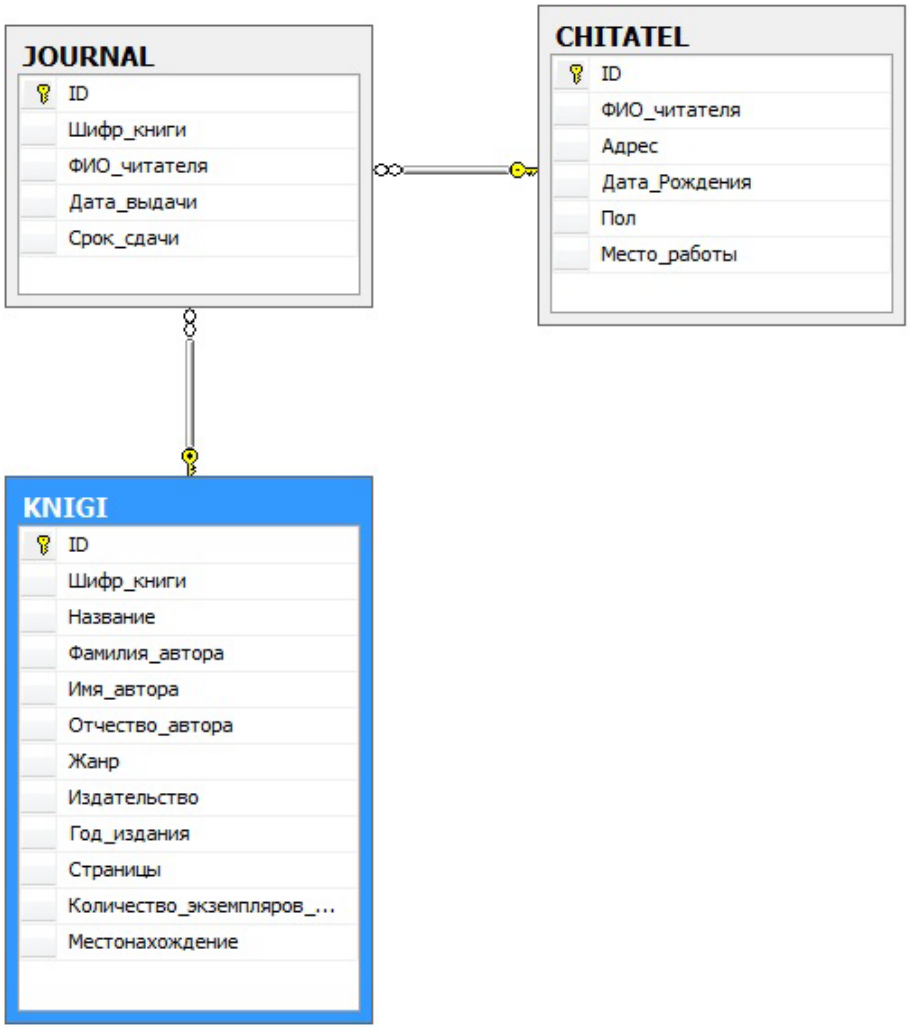


Рис. 32 - Диаграмма БД

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

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

ЦЕЛЬ РАБОТЫ

Заполнить полученную базу данных в SQL Management Studio и сформировать SQL запросы на выборку данных из базы данных.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Язык SQL – это непроцедурный язык, поэтому в нем необходимо указывать, какая информация должна быть получена, а не как ее можно получить. Иначе говоря, язык SQL не требует указания методов доступа к данным.

Структура команд задается набором ключевых слов, представляющих собой обычные слова английского языка, такие как CREATE TABLE (Создать таблицу), INSERT (Вставить), SELECT (Выбрать), UPDATE (Обновить), Delete (удалить).

В настоящее время для языка SQL существуют международные стандарты, формально определяющие его как стандартный язык создания и манипулирования реляционными базами данных, каковым он фактически и является.

Для определения формата операторов SQL мы будем применять следующую расширенную форму системы обозначений BNF (Backus Naur Form—форма Бэкуса-Наура)

2. Прописные буквы будут использоваться для записи зарезервированных слов и должны указываться в операторах точно так же, как это будет показано.
3. Строчные буквы будут использоваться для записи слов, определяемых пользователем.
4. Вертикальная черта (|) указывает на необходимость выбора одного из нескольких приведенных значений, например a | b | c.
5. Фигурные скобки определяют обязательный элемент, например {a}.
6. Квадратные скобки определяют необязательный элемент, например [a].

Структура SQL

В отличие от реляционной алгебры, где были представлены только операции запросов к БД, SQL является полным языком, в нем присутствуют не только операции запросов, но и операторы, соответствующие Data Definition Language (DDL) — языку описания

данных. Кроме того, язык содержит операторы, предназначенные для управления (администрирования) БД.

Таблица1 - Операторы определения данных DDL

Оператор	Действие
CREATETABLE	Создает новую таблицу в БД
DROPTABLE	Удаляет таблицу из БД
ALTERTABLE	Изменяет структуру существующей таблицы или ограничения целостности, задаваемые для данной таблицы
CREATEVIEW	Создает виртуальную таблицу, соответствующую некоторому SQL-запросу
ALTERVIEW	Изменяет ранее созданное представление
DROP VIEW	Удаляет ранее созданное представление
CREATEINDEX	Создает индекс для некоторой таблицы для обеспечения быстрого доступа по атрибутам, входящим в индекс
DROPINDEX	Удаляет ранее созданный индекс
DELETE	Удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы. Применение оператора согласуется с принципами поддержки целостности, поэтому этот оператор не всегда может быть выполнен корректно, даже если синтаксически он записан правильно
INSERT	Вставляет одну строку в базовую таблицу. Допустимы модификации оператора, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу
UPDATE	Обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации
SELECT	Оператор, заменяющий все операторы реляционной алгебры и позволяющий сформировать результирующее отношение, соответствующее запросу

Таблица2 - Средства администрирования данных

Оператор	Смысл	Действие
ALTERDATABASE	Изменить БД	Изменить набор основных объектов в базе данных, ограничений, касающихся всей базы данных
ALTERDBAREA	Изменить область хранения БД	Изменить ранее созданную область хранения
ALTERPASSWORD	Изменить пароль	Изменить пароль для всей базы данных
CREATEDATABASE	Создать БД	Создать новую базу данных, определив основные параметры для нее
CREATEDBAREA	Создать область хранения	Создать новую область хранения и сделать ее доступной для размещения данных
DROPDATABASE	Удалить БД	Удалить существующую базу данных (только в том случае, когда вы имеете право выполнить это действие)
DROP DBAREA	Удалить область хранения БД	Удалить существующую область хранения (если в ней на настоящий момент не располагаются активные данные)
GRANT	Предоставить права	Предоставить права доступа на ряд действий над некоторым объектом БД
REVOKE	Лишить прав	Лишить прав доступа к некоторому объекту или некоторым действиям над объектом

Таблица3 - Средства управления транзакциями

Оператор	Смысл	Действие
COMMIT	Завершить транзакцию	Завершить комплексную взаимосвязанную обработку информации, объединенную в транзакцию
ROLLBACK	Откатить транзакцию	Отменить изменения, проведенные в ходе выполнения транзакции
SAVEPOINT	Сохранить промежуточную точку выполнения транзакции	Сохранить промежуточное состояние БД, пометить его для того, чтобы можно было в дальнейшем к нему вернуться

Таблица4 - Программный SQL

Оператор	Смысл	Действие
DECLARE	Определяет курсор для запроса	Задает некоторое имя и определяет связанный с ним запрос к БД
OPEN	Открыть курсор	Формирует виртуальный набор данных, соответствующий описанию указанного курсора и текущему состоянию БД
FETCH	Считать строку из множества строк, определенных курсором	Считывает очередную строку, заданную параметром команды из виртуального набора данных, соответствующего открытому курсору
CLOSE	Закрыть курсор	Прекращает доступ к виртуальному набору данных, соответствующему указанному курсору
PREPA	Подготовит	Сгенерировать план выполнения

RE	ь оператор SQL к динамическому выполнению	запроса, соответствующего заданному оператору SQL

Обработка элементов оператора SELECT выполняется в следующей последовательности.

1. FROM. Определяются имена используемой таблицы или нескольких таблиц.
2. WHERE. Выполняется фильтрация строк объекта в соответствии с заданными условиями.
3. GROUP BY. Образуются группы строк, имеющих одно и то же значение в указанном столбце.
4. HAVING. Фильтруются группы строк объекта в соответствии с указанным условием.
5. SELECT. Устанавливается, какие столбцы должны присутствовать в выходных данных.
6. ORDER BY. Определяется упорядоченность результатов выполнения оператора.

Порядок конструкций в операторе SELECT не может быть изменен. Только две конструкции оператора — SELECT и FROM — являются обязательными, все остальные конструкции могут быть опущены

Оператор Insert предназначен для добавления новых данных в таблицу. Он имеет следующий формат:

INSERT INTO TableName [(columnList)] VALUES (dataValueList);

Здесь параметр TableName (Имя таблицы) может представлять имя таблицы базы данных. Параметр columnList (Список столбцов) представляет собой список, состоящий из имен одного или более столбцов, разделенных запятыми. Параметр columnList является необязательным. Если он опущен, то предполагается использование списка из имен всех столбцов таблицы, указанных в том порядке, в котором они были описаны в операторе CREATE TABLE.

Оператор Update позволяет изменять содержимое уже существующих строк указанной таблицы. Этот оператор имеет следующий формат:

UPDATE tableName

SET columnNamel = dataValuel [, columnName2 = dataValue2 …] [WHERE searchCondition];

Здесь параметр tableName представляет имя таблицы базы данных. В конструкции SET указываются имена одного или более столбцов, данные в которых необходимо изменить. Конструкция WHERE является необязательной. Если она опущена, значения указанных столбцов будут изменены во всех строках таблицы. Если конструкция WHERE присутствует, то обновлены будут только те строки, которые удовлетворяют условию поиска, заданному в параметре searchCondition.

Оператор Delete позволяет удалять строки данных из указанной таблицы. Этот оператор имеет следующий формат:

DELETE FROM tableName [WHERE searchCondition];

Как и в случае операторов INSERT и UPDATE, параметр TableName представляет собой таблицы базы данных. Параметр searchCondition является необязательным — если он опущен, из таблицы будут удалены все существующие в ней строки. Однако сама по себе таблица удалена не будет. Если необходимо удалить не только содержимое таблицы, но и ее определение, следует использовать оператор DROP TABLE. Если конструкция WHERE присутствует, из таблицы будут удалены только те строки, которые удовлетворяют условию отбора, заданному параметром searchCondition.

ВЫПОЛНЕНИЕ РАБОТЫ

Приступим к заполнению строк в наших таблицах. Для этого щелкаем правой кнопкой мыши по нужной нам таблице и выбираем Изменить первые 200 строк. (Рис. 1)

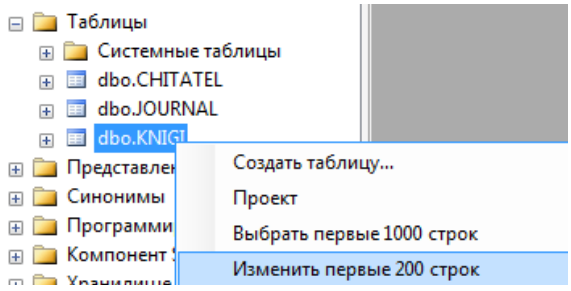


Рис.1 – Вызов окна заполнения таблиц

В открывшемся окне заполняем строки нужными данными. В нашем случае столбец ID – счётчик и заполняется автоматически.

ВИКТОРИЯ-ПК\НО...ка - dbo.KNIGI

ID	Шифр_книги	Название	ФИО_автора	Жанр	Издательство	Год_издания
1	111	Евгений Онегин	Александр Сер...	Классика	Перо	2002
2	222	Преступление ...	Федор Михайло...	Классика	Москва	2006
3	333	Горе от ума	Александр Сер...	Классика	Москва	2008

Рис.2 – Заполненная таблица KNIGI

Таким же образом заполняем остальные таблицы.

ID	ФИО_читателя	Адрес	Дата_рождения	Пол	Место_работы
1	Иванов Иван И...	Курск 50 лет О...	1998-12-11	муж	Такси
2	Кривцова Анна ...	Курск улица Ле...	1993-07-14	жен	ТРЦ Европа 40

Рис.3 – Заполненная таблица CHITATEL

ID	Шифр_книги	ФИО_читателя	Дата_выдачи	Срок_сдачи
1	1	2	2014-03-23	2014-08-23
2	3	1	2014-11-12	2014-12-12

Рис.4 – Заполненная таблица JOURNAL

Важно иметь ввиду что столбцы Шифр_книги и Фио_читателя таблицы JOURNAL заполняются только внесенными данными в столбец ID KNIGI и ID CHITATEL соответственно. В противном случае SQL выдаст ошибку (Рис. 5)

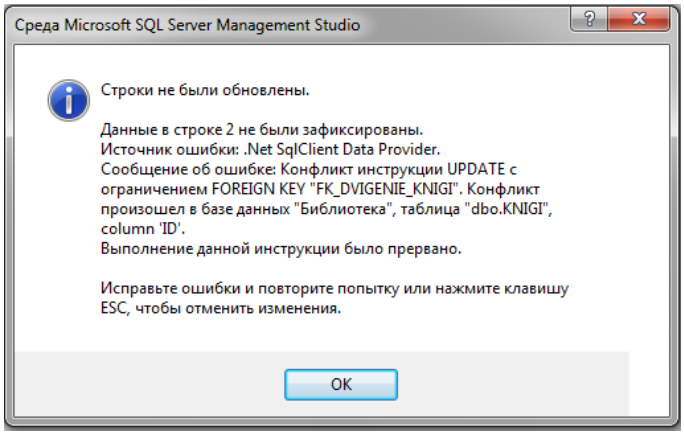


Рис.5 – Окно ошибки «Строки не были обновлены»

Теперь приступим к выборке данных с помощью запросов. Для начала рассмотрим 4 основных запроса SELECT, INSERT, UPDATE, DELETE. Будем работать с таблицей KNIGI.

SELECT – наиболее часто используемый SQL оператор. Он предназначен для выборки информации из таблиц. Чтобы при помощи оператора SELECT извлечь данные из таблицы, нужно указать как минимум две вещи — что вы хотите выбрать и откуда.

Щелкаем правой кнопкой мыши по нужной таблице, выбираем Создать сценарий для таблицы Используя SELECT Новое окно редактора запросов. (Рис. 6)

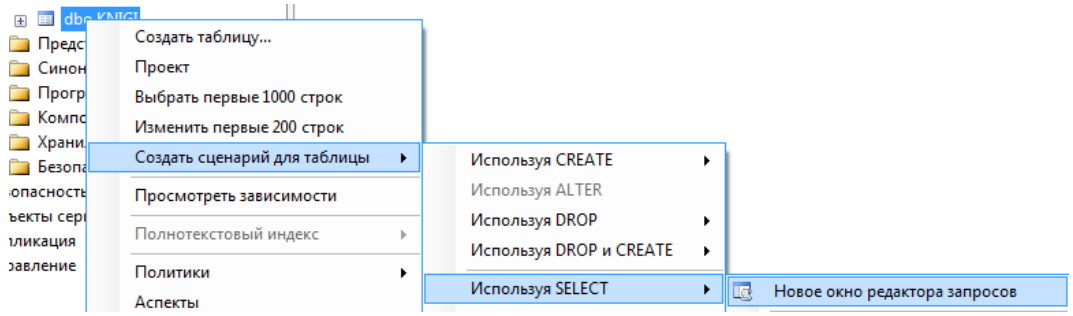


Рис.6 – Вызов окна редактора запросов

Сделаем выборку отдельных столбцов:

```
SELECT [Имя столбца]
FROM Имя таблицы
```

Для создания и тестирования данного запроса в Management Studio выполните следующие шаги:

1. В открывшемся окне создания нового запроса введите представленные выше инструкции SQL.
2. Для запуска запроса на выполнение щелкните кнопку Выполнить на панели инструментов или нажмите клавишу F5. В нижней части экрана должны появиться результаты.

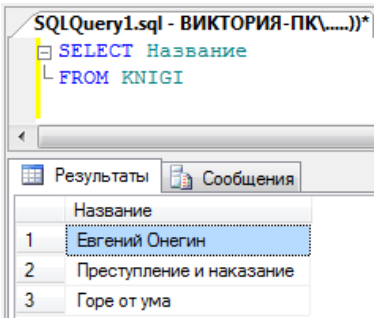


Рис.7 – Результат выполнения запроса

Сделаем выборку нескольких столбцов:

Для выборки из таблицы нескольких столбцов используется тот же оператор SELECT. Отличие состоит в том, что после ключевого слова SELECT необходимо через запятую указать несколько имен столбцов. (Рис. 8)

SELECT Имя столбца1, Имя столбца2

FROM Имя таблицы

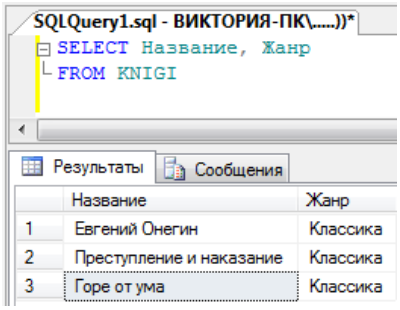


Рис.8 – Результат выполнения запроса

Выборка всех столбцов:

Помимо возможности осуществлять выборку определенных столбцов (одного или нескольких), при помощи оператора SELECT можно запросить все столбцы, не перечисляя каждый из них. Для этого вместо имен столбцов вставляется групповой символ “звездочка” (*). Это делается следующим образом. (Рис. 9)

SELECT * FROM Имя таблиц

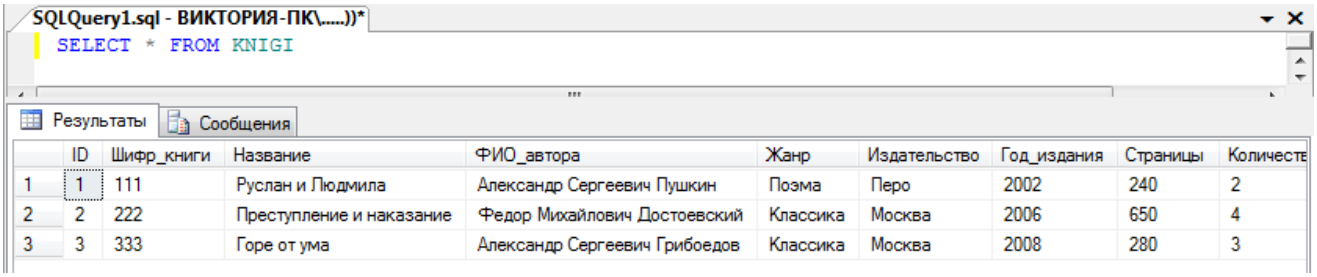
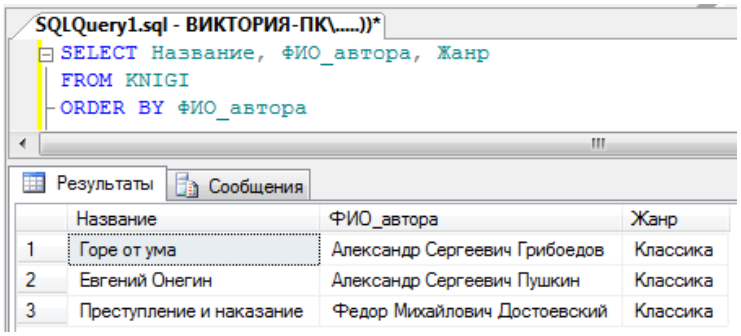


Рис.9 – Результат выполнения запроса

Сортировка данных:

В результате выполнения запроса на выборку данные выводятся в том порядке, в котором они находятся в таблице. Для точной сортировки выбранных при помощи оператора SELECT данных используется предложение ORDER BY. В этом предложении указывается имя одного или нескольких столбцов, по которым необходимо отсортировать результаты. (Рис. 10). Взгляните на следующий пример

```
SELECT Имя столбца1, Имя столбца2, Имя столбца3
FROM Название
ORDER BY Имя столбца3
```



	Название	ФИО_автора	Жанр
1	Горе от ума	Александр Сергеевич Грибоедов	Классика
2	Евгений Онегин	Александр Сергеевич Пушкин	Классика
3	Преступление и наказание	Федор Михайлович Достоевский	Классика

Рис.10 – Результат выполнения запроса

Указание направления сортировки:

В предложении ORDER BY можно также использовать порядок сортировки по убыванию. Для этого необходимо указать ключевое слово DESC. (Рис. 11)

```
SELECT Имя столбца1, Имя столбца2, Имя столбца3
FROM Название
ORDER BY Имя столбца3 DESC
```

Ключевое слово DESC применяется только к тому столбцу, после которого оно указано. Таким образом, столбец Имя столбца3 отсортирован в порядке убывания.

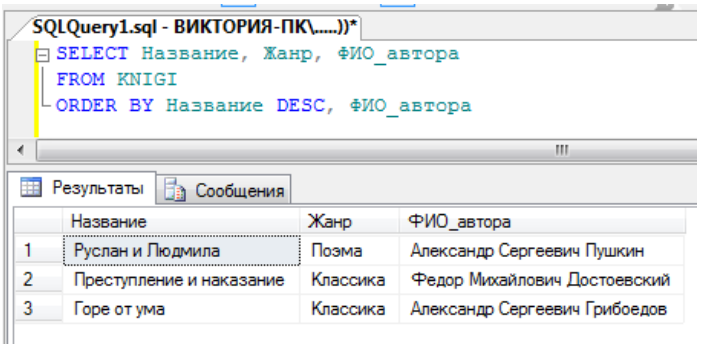


Рис.11 – Результат выполнения запроса

Фильтрация данных:

В таблицах баз данных обычно содержится много информации и довольно редко возникает необходимость выбирать все строки таблицы. Гораздо чаще бывает нужно извлечь какую-то часть данных таблицы для каких-либо действий или отчетов. Выборка только необходимых данных включает в себя критерий поиска, также известный под названием предложение фильтрации. В операторе SELECT данные фильтруются путем указания критерия поиска в предложении WHERE. Предложение WHERE указывается сразу после названия таблицы (предложения FROM) следующим образом (Рис. 12):

SELECT Имя столбца1, Имя столбца2, Имя столбца3
FROM Название таблицы
WHERE Имя столбца= ‘значение'

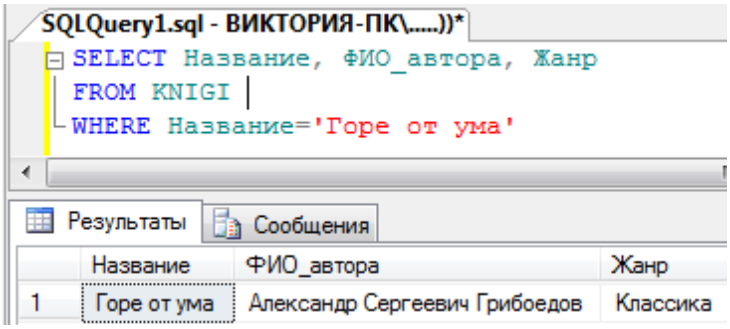


Рис.12 – Результат выполнения запроса

При совместном использовании предложений ORDER BY и WHERE, предложение ORDER BY должно следовать после WHERE.

SQL поддерживает весь спектр условных (логических) операций, которые приведены в следующей таблице.

Таблица5 – Условные операторы SQL

	Описание

Операция	
=	Равенство
<>	Неравенство
!=	Неравенство
<	Меньше
<=	Меньше или равно
!<	Не меньше
>	Больше
>=	Больше или равно
!>	Не больше
BETWEEN	Между двумя указанными значениями
IS NULL	Значение NULL

Для фильтрации данных по критерию соответствия определенной символьной строки заданному шаблону используется оператор LIKE. Шаблон может включать обычные символы и символы-шаблоны. Во время сравнения с шаблоном необходимо, чтобы его обычные символы в точности совпадали с символами, указанными в строке. Символы-шаблоны могут совпадать с произвольными элементами символьной строки. Использование символов-шаблонов с оператором LIKE предоставляет больше возможностей, чем использование обычных операторов сравнения. Шаблон может включать в себя следующие символы-шаблоны.

Таблица6 – Символы-шаблоны оператора LIKE

Символ-шаблон	Описание
%	Любое количество символов
_	Любой одиночный символ
[]	Любой символ, указанный в квадратных скобках
[^]	Любой символ, кроме перечисленных в квадратных скобках

Отбираются значения, соответствующие образцу (Рис. 13)

SELECT * FROM Название таблицы WHERE Имя столбца LIKE 'знач%'

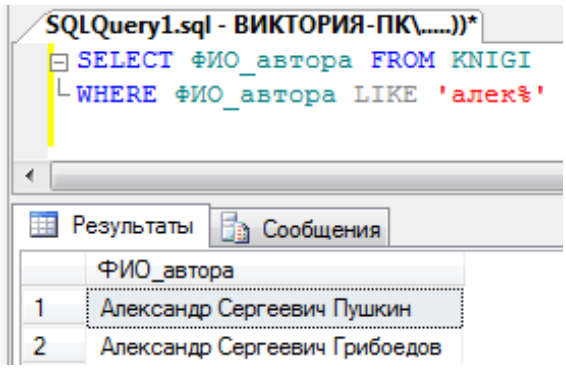


Рис.13 – Результат выполнения запроса

Исключение дублирующих записей:

Для исключения из результата выборки повторяющихся строк используется ключевое слово DISTINCT, которое указывается сразу после SELECT.

SELECT DISTINCT Столбец1
FROM Таблица1

Использование агрегатных функций:

В SQL определено множество встроенных функций различных категорий, среди которых особое место занимают агрегатные функции, оперирующие значениями столбцов множества строк и возвращающие одно значение. Аргументами агрегатных функций могут быть как столбцы таблиц, так и результаты выражений над ними. В следующей таблице приведены наиболее часто используемые стандартные унарные агрегатные функции.

Таблица7 - Стандартные унарные агрегатные функции

Функция	Возвращаемое значение
COUNT	Количество значений в столбце или строк в таблице
SUM	Сумма
AVG	Среднее
MIN	Минимум
MAX	Максимум

Общий формат унарной агрегатной функции следующий:

имя_функции ([ALL | DISTINCT] выражение)

где DISTINCT указывает, что функция должна рассматривать только различные значения аргумента, а ALL — все значения, включая повторяющиеся (этот вариант используется по умолчанию). Например, функция AVG с ключевым словом DISTINCT для строк столбца со значениями 1, 1, 1 и 3 вернет 2, а при наличии ключевого слова ALL вернет 1,5.

Агрегатные функции применяются во фразах SELECT и HAVING. Здесь мы рассмотрим их использование во фразе SELECT. В этом случае выражение в аргументе функции применяется ко всем строкам входной таблицы фразы SELECT.

Функция COUNT имеет два формата. В первом случае возвращается количество строк входной таблицы, во втором случае — количество значений аргумента во входной таблице:

- COUNT(*)
- COUNT([DISTINCT | ALL] выражение)

Простейший способ использования этой функции — подсчет количества строк в таблице (всех или удовлетворяющих указанному условию). Для этого используется первый вариант синтаксиса. (Рис. 14)

```
SELECT COUNT(*) AS ‘ИМЯ НОВОЙ ТАБЛИЦЫ’
FROM Название таблицы
```

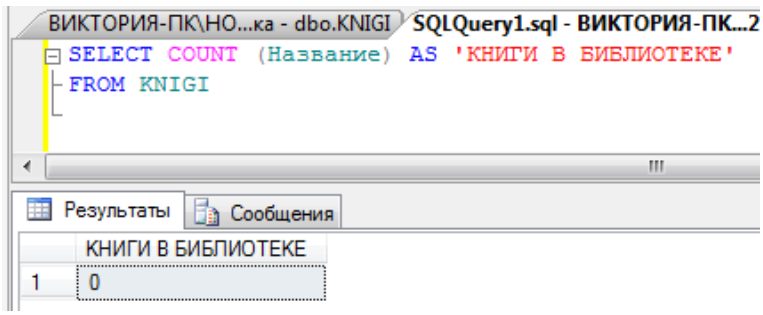


Рис.14 – Результат выполнения запроса

Использование остальных унарных агрегатных функции аналогично COUNT за тем исключением, что для функций MIN и MAX использование ключевых слов DISTINCT и ALL не имеет смысла. С функциями COUNT, MAX и MIN кроме числовых могут использоваться и символьные поля. Если аргумент агрегатной функции не содержит значений, функция COUNT возвращает 0, а все остальные - значение NULL. (Рис. 15)

SELECT MAX(ИМЯ СТОЛБЦА)
FROM [ТАБЛИЦА]
WHERE ИМЯ СТОЛБЦА<'1.09.2010'

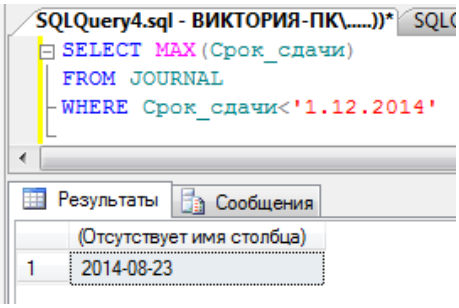


Рис.15 – Результат выполнения запроса

Инструкция INSERT добавляет в таблицу одну или несколько новых строк. В простейшем виде инструкция INSERT имеет следующий вид:

INSERT [INTO] table_or_view [(column_list)] data_values

Инструкция INSERT выполняет вставку в указанную таблицу или представление значений data_values в виде одной или нескольких строк. Параметр column_list представляет собой разделяемый запятыми список имен столбцов, для которых представляются данные. Если аргумент column_list не задается, данные получают все столбцы таблицы или представления.

Если в аргументе column_list указаны не все столбцы таблицы или представления, то пропущенные столбцы вставляется либо значение по умолчанию (если для столбца оно задано), или значение NULL. Для всех пропущенных столбцов должно быть либо определено значение по умолчанию, либо допускаться значение NULL.

Щелкаем правой кнопкой мыши по нужной таблице, выбираем Создать сценарий для таблицы Используя INSERT Новое окно редактора запросов. (Рис. 16)

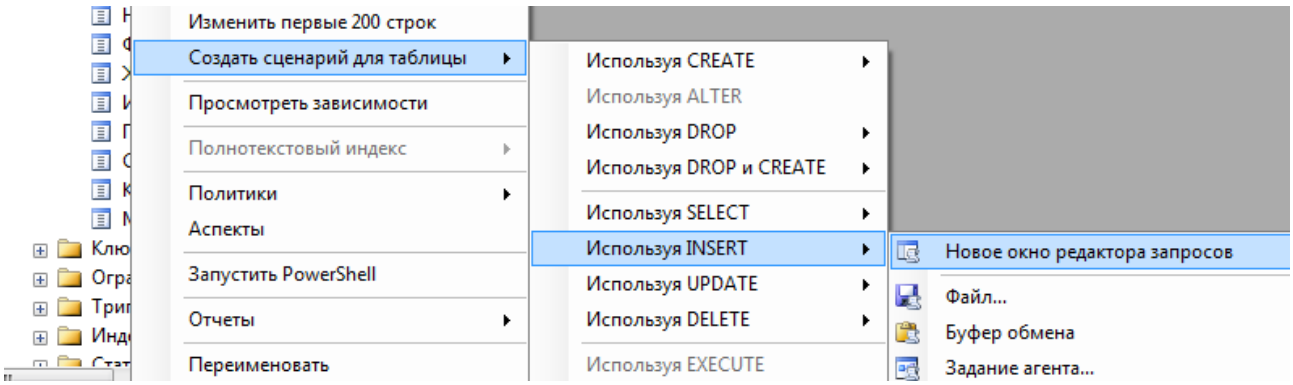


Рис.16 – Вызов окна редактора запросов

INSERT INTO table_name (column1, column2, column3)
VALUES (‘data1’, ‘data2’, ‘data3’)

Как видно из SQL запроса, сначала перечисляются столбцы с которые будет производиться запись, затем параметром VALUES вводится информация в соответствии со столбцами. Если запись производиться во все столбцы и по порядку, то перечислять их не обязательно, но желательно. (Рис 17, 18)

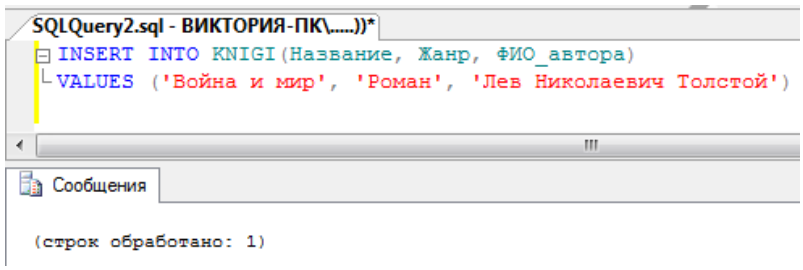
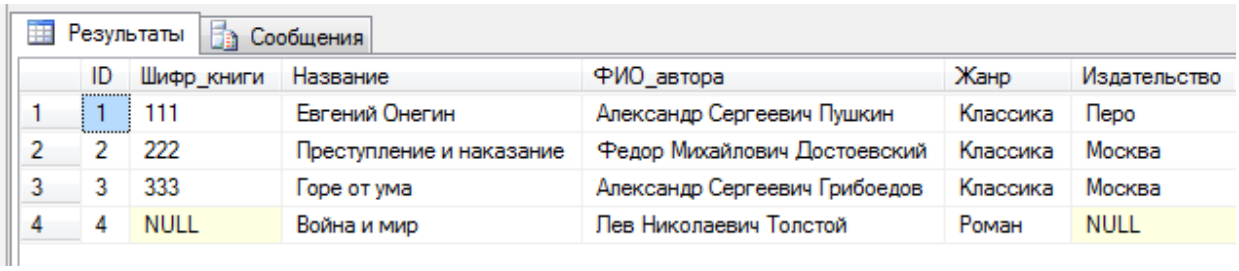


Рис.17 – Выполнение запроса



	ID	Шифр_книги	Название	ФИО_автора	Жанр	Издательство
1	1	111	Евгений Онегин	Александр Сергеевич Пушкин	Классика	Перо
2	2	222	Преступление и наказание	Федор Михайлович Достоевский	Классика	Москва
3	3	333	Горе от ума	Александр Сергеевич Грибоедов	Классика	Москва
4	4	NULL	Война и мир	Лев Николаевич Толстой	Роман	NULL

Рис.18 – Результат выполнения запроса

Инструкция UPDATE – SQL запрос обновления существующих данных. Щелкаем правой кнопкой мыши по нужной таблице, выбираем Создать сценарий для таблицы Используя UPDATE Новое окно редактора запросов. (Рис. 19)

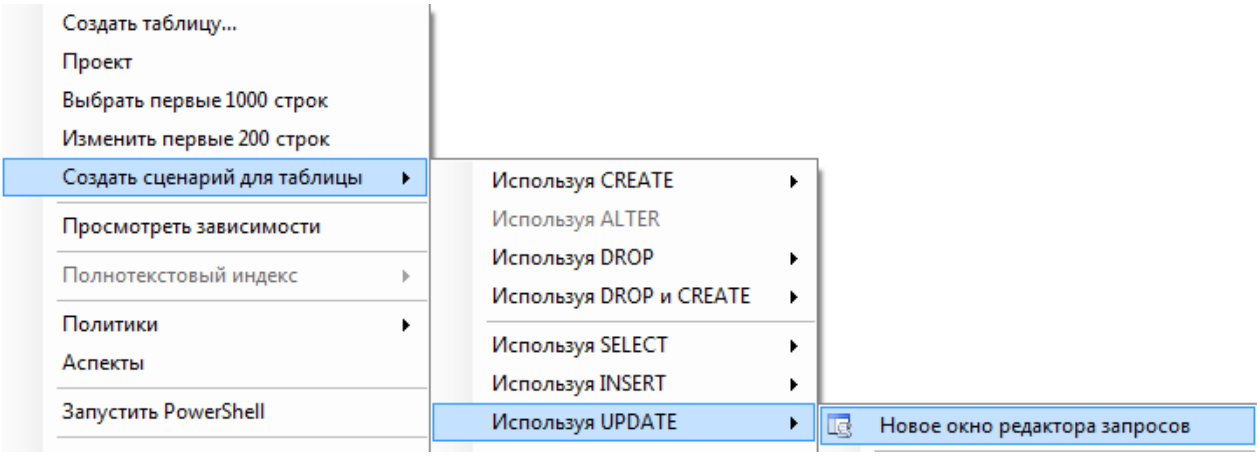


Рис.19 – Вызов окна редактора запросов

Запрос UPDATE используется для изменения существующих данных на новые значения. Чаще всего данный запрос применяется с условием WHERE и выглядит так:

UPDATE table_name SET column1 = ‘data1’, column2 = ‘data2’
WHERE column3 = ‘data3’

Данный запрос обновляет строки в столбцах column1, column2, в которых выполняется условие column3 = ‘data3’. (Рис. 20, 21)

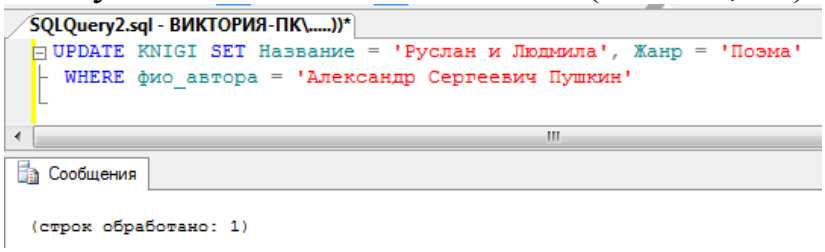
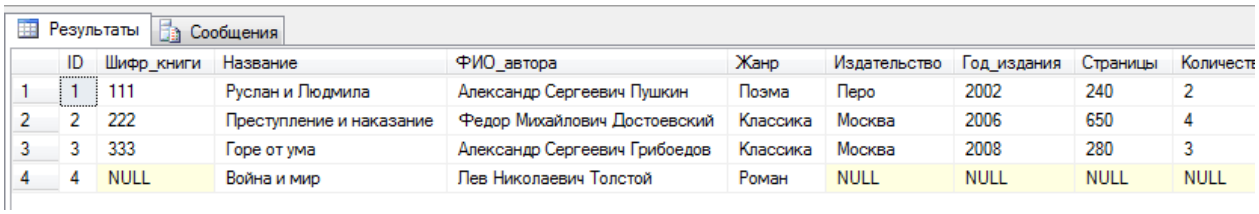


Рис.20 – Выполнение запроса



	ID	Шифр_книги	Название	ФИО_автора	Жанр	Издательство	Год_издания	Страницы	Количеств
1	1	111	Руслан и Людмила	Александр Сергеевич Пушкин	Поэма	Перо	2002	240	2
2	2	222	Преступление и наказание	Федор Михайлович Достоевский	Классика	Москва	2006	650	4
3	3	333	Горе от ума	Александр Сергеевич Грибоедов	Классика	Москва	2008	280	3
4	4	NULL	Война и мир	Лев Николаевич Толстой	Роман	NULL	NULL	NULL	NULL

Рис.21 – Результат выполнения запроса

Инструкция DELETE – SQL запрос удаления записи. Щелкаем правой кнопкой мыши по нужной таблице, выбираем Создать сценарий для таблицы Используя DELETE Новое окно редактора запросов. Запрос DELETE удаляет ненужные вам строки (Рис. 22, 23). Будьте внимательны запрос удаляет все строку целиком:

DELETE FROM table_name WHERE column1 = ‘data1’

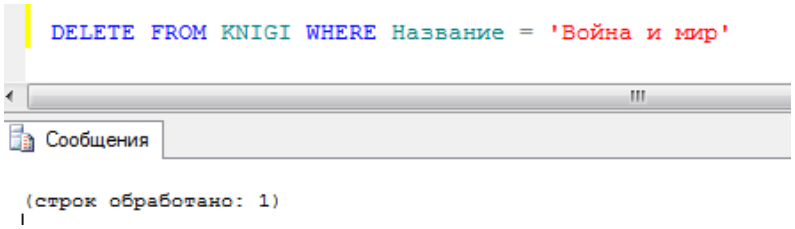


Рис.22 – Выполнение запроса

Результаты | Сообщения

	ID	Шифр_книги	Название	ФИО_автора	Жанр
1	1	111	Руслан и Людмила	Александр Сергеевич Пушкин	Поэма
2	2	222	Преступление и наказание	Федор Михайлович Достоевский	Классика
3	3	333	Горе от ума	Александр Сергеевич Грибоедов	Классика

Рис.23 – Результат выполнения запроса

Этот SQL запрос удаляет все строки где выполняется условие column1 = ‘data1’. Помните после выполнения запроса, удаленная строка не подлежит восстановлению.

Оператор INNER JOIN возвращает все записи из таблиц table_01 и table_02, связанные посредством primary/foreign ключей, и соответствующие условию WHERE для таблицы table_01. Если в какой-либо из таблиц отсутствует запись, соответствующая соседней, то в выдачу такая пара включена не будет. Иными словами, выдадутся только те записи, которые есть и в первой, и во второй таблице. То есть выборка идет фактически по связи (ключу), выдадутся только те записи, которые связаны между собой. «Одинокие» записи, для которых нет пары в связи, выданы не будут. (Рис. 24)

```
SELECT * FROM table_01
INNER JOIN table_02
ON table_01.primary_key = table_02.foreign_key
```

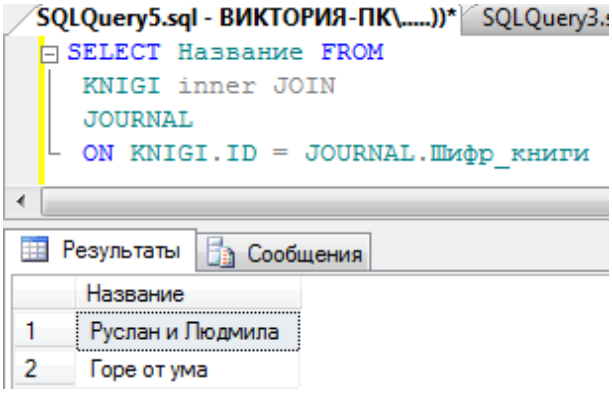


Рис.24 – Результат выполнения запроса

Оператор LEFT JOIN возвращает все данные из «левой» таблицы, даже если не найдено соответствий в «правой» таблице («левая» таблица в SQL-запросе стоит левее знака равно, «правая» — правее, то есть обычная логика правой и левой руки). Иными словами, если мы присоединяем к «левой» таблице «правую», то выберутся все записи в соответствии с условиями WHERE для левой таблицы. Если в «правой» таблице не было соответствий по ключам, они будут возвращены как NULL. Таким образом, здесь главной выступает «левая» таблица, и относительно нее идет выдача. В условии ON «левая» таблица прописывается первой по порядку (table_01), а «правая» – второй (table_02), (Рис. 25, 26):

```
SELECT * FROM table_01
LEFT JOIN table_02
ON table_01.primary_key = table_02.foreign_key
```

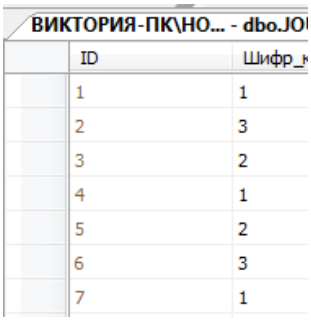


Рис.25 – Таблица Journal

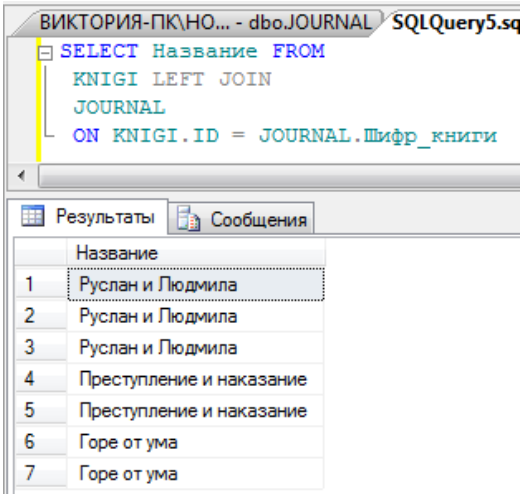


Рис.26 – Результат выполнения запроса

Оператор RIGHT JOIN возвращает все данные из «правой» таблицы, даже если не найдено соответствий в «левой» таблице. То есть примерно также, как и в LEFT JOIN, только NULL вернется для полей «левой» таблицы. Грубо говоря, эта выборка ставит во главу угла правую «таблицу», относительно нее идет выдача. Обратите внимание на WHERE в следующем примере, условие выборки затрагивает «правую» таблицу (Рис. 27):

```
SELECT * FROM table_01
RIGHT JOIN table_02
ON table_01.primary_key = table_02.foreign_key
WHERE table_02.column_01 = ‘value’
```

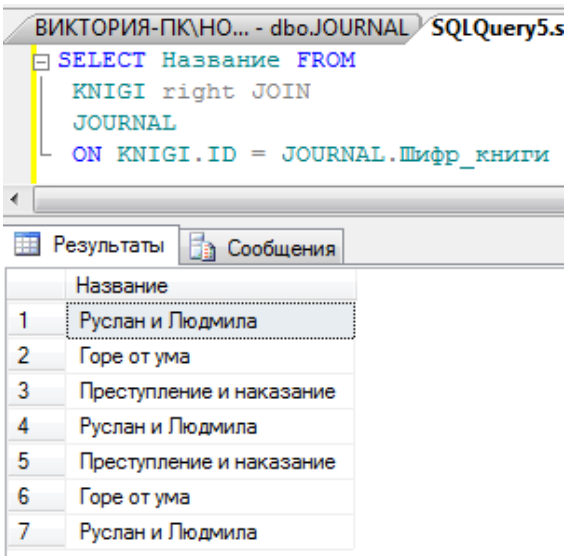


Рис.27 – Результат выполнения запроса

Запросы, отделенные круглыми скобками и входящие в состав конструкций HAVING, WHERE, FROM, SELECT и WITH внешнего запроса SELECT или каких-либо других перечисленных запросов, заключающих в себе эти конструкции, называются вложенными запросами.

Вложенные подзапросы могут быть простыми и связанными. Если результат вложенного запроса не зависит от результата внешнего, то такой вложенный запрос называется простым.

К примеру, данные из таблиц, перечисленных после ключевого слова FROM внешнего и вложенного запроса, находятся независимо, что не требует вводить полный адрес столбцов.

В одном запросе может быть несколько подзапросов, синтаксис у такого запроса следующий (Рис. 28)

SELECT имя_столбца FROM имя_таблицы
WHERE часть условия IN
(SELECT имя_столбца FROM имя_таблицы
WHERE часть условия IN
(SELECT имя_столбца FROM имя_таблицы WHERE условие))

Обратите внимание, что подзапросы могут выбирать только один столбец, значения которого они будут возвращать внешнему запросу. Попытка выбрать несколько столбцов приведет к ошибке.

Не рекомендуется создавать запросы со степенью вложения больше трех. Это приводит к увеличению времени выполнения и к сложности восприятия кода.

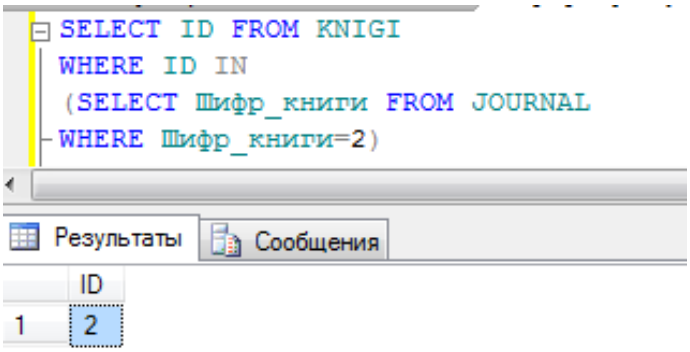


Рис.28 – Результат выполнения запроса

3. АДМИНИСТРИРОВАНИЕ БАЗЫ ДАННЫХ

ЦЕЛЬ РАБОТЫ

Научиться добавлять пользователей в SQLServer и наделять их различными правами на управление базой данных (admin и user). Научиться создавать план обслуживания базы данных.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

При определении прав пользователя SQL Server 2008 использует два уровня зашиты.

Первый уровень — проверка подлинности пользователя. Во время проверки определяется, имеется ли у пользователя право на подключения к данному SQL Server 2008.

Второй уровень системы безопасности — авторизация, часто называемая также проверкой прав доступа. При этом определяется, какие действия пользователь сможет выполнять с БД, после того как он пройдет проверку подлинности SQL Server 2008.

Чтобы установить соединение с SQL Server 2008, пользователь должен указать правильный идентификатор учетной записи (login identifier) пользователя, который определяет права доступа к SQL Server

2008. Каждый SQL Server из числа имеющихся в системе проверяет, действительно ли идентификатор учетной записи пользователя, введенный при установке соединения, обеспечивает пользователю право подключаться к этому SQL Server. Проверка идентификатора учетной записи пользователя называется проверкой подлинности (authentication). В SQL Server 2008 предусмотрены два вида проверки подлинности: средствами Windows и средствами SQL Server. Подключаясь к SQL Server 2008, пользователь указывает вид проверки подлинности для данного соединения.

Администратор БД может предоставить пользователям и группам пользователей Windows NT 4.0/2000 право устанавливать соединение с данным SQL Server 2008. Если при подключении указывается, что проверка подлинности выполняется средствами Windows, то для проверки того, зарегистрирован ли данный пользователь в этой сети, используются средства Windows. SQL Server идентифицирует пользователя по имени его учетной записи, существующей в сети, и разрешает или запрещает этому пользователю устанавливать соединение, не требуя его отдельной регистрации как пользователя SQL Server 2008. Такой тип соединения называется доверенным (trusted).

При проверке подлинности средствами Windows используются механизмы защиты Windows NT 4.0/2000, в том числе такие средства, как защищенный режим проверки прав пользователя в сети и шифрование пароля, аудит, ограничение срока действия пароля, ограничение на минимальную длину пароля и блокирование учетной записи пользователя после нескольких неудачных попыток регистрации.

Администратор БД может создавать учетные записи пользователей SQL Server, вводя имя пользователя и назначая ему соответствующий пароль. Эти учетные записи никак не связаны с учетными записями пользователей и групп Windows NT 4.0/2000.

Если при подключении к серверу выбран режим проверки подлинности средствами SQL Server, то SQL Server 2008 сам проверяет подлинность пользователя, уточняя, имеется ли такая учетная запись с указанным именем и паролем на SQL Server 2008.

В SQL Server 2008 есть два режима проверки подлинности. По умолчанию используется режим проверки подлинности средствами Windows (Windows Authentication Mode).

При этом устанавливать соединение с SQL Server разрешается только зарегистрированным пользователям Windows NT 4.0/2000, прошедшим проверку подлинности Windows. SQL Server 2008 также

может работать в смешанном режиме (Mixed Mode), при этом пользователь может подключиться к SQL Server 2008, если он прошел проверку подлинности Windows NT 4.0/2000 или указал правильное имя и пароль пользователя SQL Server.

После того как SQL Server 2008 проверит подлинность пользователя, SQL Server 2008 определяет права данного пользователя выполнять различные действия в размещенных на сервере БД. Сам по себе идентификатор учетной записи пользователя не дает зарегистрированному пользователю прав доступа к различным объектам БД. Он лишь позволяет перейти к следующему этапу — авторизации, или проверке прав пользователя. Такой механизм защиты гарантирует, что зарегистрированный пользователь не получит автоматически доступ ко всем БД на SQL Server 2008, с которым он установил соединение.

Как правило, администратор БД должен сопоставить идентификатор учетной записи пользователя идентификатору пользователя в БД, прежде чем пользователь, подключившийся с использованием этого идентификатора учетной записи, получит доступ или сможет выполнить какие-либо действия в этой БД. Администратор БД определяет права доступа к объектам (таким как таблицы, представления и хранимые процедуры) в БД для всех учетных записей пользователей.

Если учетная запись пользователя на сервере, позволяющая пользователю подключиться к SQL Server 2008, не связана ни с одной учетной записью пользователя в БД, она автоматически связывается с идентификатором учетной записи пользователя guest в этой БД (если такой идентификатор существует). Если в БД присутствует учетная запись пользователя guest, права подключающегося пользователя ограничиваются правами пользователя guest. Если в БД отсутствует учетная запись пользователя guest, подключающийся пользователь не получит доступ к БД до тех пор, пока его учетная запись на сервере не будет связана с учетной записью БД. По умолчанию во всех вновь созданных пользовательских БД отсутствует учетная запись пользователя guest.

Роли позволяют администратору БД объединять пользователей в группы, для которых задаются определенные права.. Роли в SQL Server 2008 во многом аналогичны группам пользователей в Windows NT 4.0/2000. В SQL Server 2008 имеются встроенные роли, определенные на уровне сервера, и роли, определенные на уровне БД. Для этих ролей заранее установлены права на уровне всего сервера и на уровне БД. Кроме того, администратор БД может создавать новые роли на уровне

БД. Каждый пользователь БД является участником роли БД public и, следовательно, обладает всеми правами, предоставленными роли public, если для него особо не определены какие-либо специальные права. Дополнительные права следует предоставлять пользователю или группе, к которой он принадлежит, в явном виде.

Фиксированные роли базы данных задаются на уровне базы данных и предусмотрены в каждой базе данных. Элементы ролей базы данных db_owner и db_securityadmin могут управлять членством в фиксированных ролях базы данных, однако только члены роли базы данных db_owner могут добавлять членов в фиксированную роль базы данных db_owner.

Имеются следующие фиксированные роли базы данных:

- db_accessadmin
- db_backupoperator
- db_datareader
- db_datawriter
- db_ddladmin
- db_denydatareader
- db_denydatawriter
- db_owner
- db_securityadmin

Каждый пользователь базы данных принадлежит к роли базы данных public. Если для пользователя не были заданы особые разрешения на защищаемый объект, то он наследует разрешения роли public на этот объект.

Члены фиксированной роли базы данных db_accessadmin могут добавлять или удалять права удаленного доступа для имен входа и групп Windows, а также имен входа SQL Server.

Члены фиксированной роли базы данных db_backupoperator могут выполнять ее резервирование.

Элементы фиксированной роли базы данных db_datareader могут считывать все данные из всех пользовательских таблиц.

Члены фиксированной роли базы данных db_datawriter могут добавлять, удалять или изменять данные во всех пользовательских таблицах.

Члены фиксированной роли базы данных db_ddladmin могут выполнять любые команды языка определения данных (DDL) в базе данных.

Члены фиксированной роли базы данных db_denydatareader не могут считывать данные из пользовательских таблиц базы данных.

Члены фиксированной роли базы данных db_denydatawriter не могут добавлять, изменять или удалять данные в пользовательских таблицах базы данных.

Члены фиксированной роли базы данных db_owner могут выполнять все действия по конфигурации и обслуживанию базы данных, а также удаление базы данных.

Элементы фиксированной роли базы данных db_securityadmin могут изменять членство в роли и управлять разрешениями.

Чтобы пользователь мог выполнять какие-либо действия в системе SQL Server 2008, он должен подключиться к SQL Server 2008, используя учетную запись, и к БД, используя регистрационную запись.

Шифрование - процесс кодирования конфиденциальных данных с использованием ключа или пароля. Шифрование надежно защищает данные и сокращает вероятность несанкционированного раскрытия конфиденциальной информации, так как без соответствующего ключа или пароля данные бесполезны. SQL Server располагает многими режимами шифрования, в том числе на уровне ячеек, базы данных, файлов через Windows и шифрования на транспортном уровне.

Шифрование SQL Server не решает проблему доступности инфраструктуры и баз данных SQL Server, но повышает защищенность данных на уровнях базы данных и операционной системы, даже если нарушена конфиденциальность инфраструктуры или баз данных SQL Server.

Модель шифрования SQL Server в основном предоставляет функции управления ключами шифрования, соответствующие стандарту ANSI X9.17. В этом стандарте определены несколько уровней ключей шифрования, использующихся для шифрования других ключей, которые в свою очередь применяются для шифрования собственно данных.

В таблице №1 перечислены уровни ключей шифрования SQL Server и ANSI X9.17.

Таблица 1 - уровни ключей шифрования SQL Server и ANSI X9.17

Таблица	Уровень шифрования ключей SQL Server и ANSI X9.17	
Уровень SQL Server	Уровень ANSI X9.17	Описание
SMK	Главный ключ	SMK – ключ верхнего уровня, используемый для шифрования DMK. SMK шифруется с применением DPAPI.
DMK	Ключ шифрования ключей	DMK – симметричный ключ, используемый для шифрования симметричного ключа, ассиметричного ключа и сертификата. Для каждой базы данных может быть определен только один DMK.
Симметричные ключи, ассиметричные ключи и сертификаты	Ключ данных	Симметричные ключи, ассиметричные ключи и сертификаты используются для шифрования данных.

Главный ключ службы Service master key(SMK) - ключ верхнего уровня и предок всех ключей в SQL Server. SMK - асимметричный ключ, шифруемый с использованием Windows Data Protection API (DPAPI). SMK автоматически создается, когда шифруется какой-нибудь объект, и привязан к учетной записи службы SQL Server. SMK используется для шифрования главного ключа базы данных Database master key (DMK).

Второй уровень иерархии ключей шифрования - DMK. С его помощью шифруются симметричные ключи, асимметричные ключи и сертификаты. Каждая база данных располагает лишь одним DMK.

Следующий уровень содержит симметричные ключи, асимметричные ключи и сертификаты. Симметричные ключи - основное средство шифрования в базе данных. Microsoft рекомендует шифровать данные только с помощью симметричных ключей. Кроме того, в SQL Server 2008 и более новых версиях есть сертификаты уровня сервера и ключи шифрования базы данных для прозрачного шифрования данных. На рисунке №1 показана иерархия ключей шифрования для SQL Server 2008 и более новых версий.

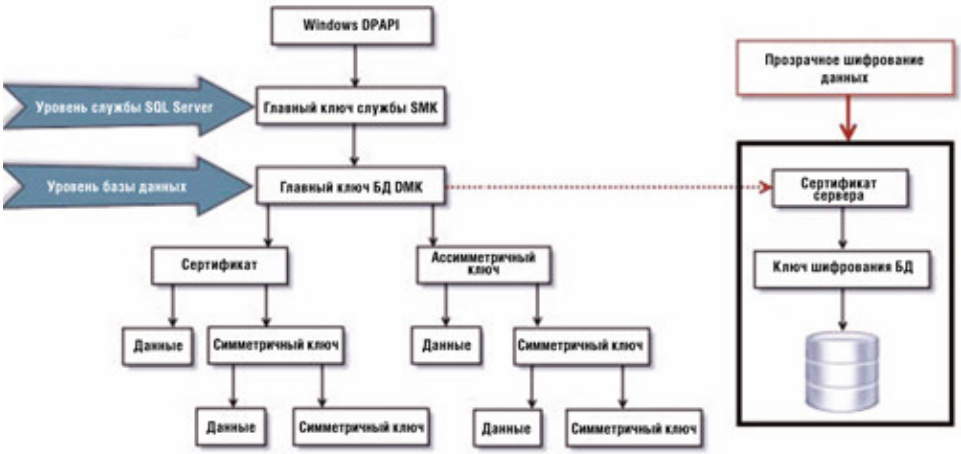


Рис. 1 - Иерархия ключей шифрования для SQL Server 2008.

После знакомства с иерархией ключей шифрования SQL Server мы рассмотрим способы реализации шифрования, доступные в SQL Server.

Начиная с SQL Server 2005, можно шифровать или расшифровывать данные на сервере. Делать это можно различными способами. Например, можно шифровать данные в базах данных одним из следующих методов.

- Пароль. Это наименее надежный способ, так как для шифрования и расшифровки данных используется одна и та же парольная фраза. Если хранимые процедуры и функции не зашифрованы, то доступ к парольной фразе возможен через метаданные.
- Сертификат. Этот способ обеспечивает надежную защиту и высокое быстродействие. Сертификат можно связать с пользователем; подписать его необходимо с помощью DMK.
- Симметричный ключ. Достаточно надежен, удовлетворяет большинству требований к безопасности данных и обеспечивает достаточное быстродействие. Для шифрования и расшифровки данных используется один ключ.

- Асимметричный ключ. Обеспечивает надежную защиту, так как применяются различные ключи для шифрования и расшифровки данных. Однако это негативно влияет на быстродействие. Специалисты Microsoft не рекомендуют использовать его для шифрования крупных значений. Асимметричный ключ может быть подписан с использованием DMK или создан с помощью пароля.

SQL Server располагает встроенными функциями для шифрования и расшифровки на уровне ячеек. Функции шифрования:

* ENCRYPTBYKEY, использует симметричный ключ для шифрования данных;
* ENCRYPTBYCERT, использует открытый ключ сертификата для шифрования данных;
* ENCRYPTBYPASSPHRASE, использует парольную фразу для шифрования данных;
* ENCRYPTBYASYMKEY, использует асимметричный ключ для шифрования данных.

Функции расшифровки:

- DECRYPTBYKEY, использует симметричный ключ для расшифровки данных;
- DECRYPTBYCERT, использует открытый ключ сертификата для расшифровки данных;
- DECRYPTBYPASSPHRASE, использует парольную фразу для расшифровки данных;
- DECRYPTBYASYMKEY, использует асимметричный ключ для расшифровки данных;
- DECRYPTBYKEYAUTOASYMKEY, использует асимметричный ключ, который автоматически расшифровывает сертификат.

SQL Server располагает двумя системными представлениями, с помощью которых можно получить метаданные для всех симметричных и асимметричных ключей, существующих в экземпляре SQL Server. Как видно из названий, sys.symmetric_keys возвращает метаданные для симметричных, а sys.asymmetric_keys - для асимметричных ключей. Еще одно полезное представление - sys.openkeys. В этом представлении каталога содержится информация о ключах шифрования, открытых в текущем сеансе.

В SQL Server 2008 появилась возможность зашифровать всю базу данных с использованием прозрачного шифрования. При таком шифровании можно защитить базы данных без изменения существующих

приложений, структур баз данных или процессов. Это лучший вариант для выполнения требований нормативных актов и правил корпоративной безопасности, поскольку шифруется вся база данных на жестком диске.

Прозрачное шифрование данных шифрует базы данных в реальном времени, по мере внесения записей в файлы (*.mdf) базы данных SQL Server и файлы (*.ldf) журнала транзакций. Записи также шифруются в реальном времени во время резервного копирования базы данных, а затем формируются моментальные снимки. Данные шифруются перед записью на диск и расшифровываются перед извлечением. Процесс полностью прозрачен для пользователя или приложения, поскольку выполняется на уровне SQL Server Service.

При прозрачном методе SQL Server шифрует базу данных с помощью ключа шифрования базы данных. Этот асимметричный ключ хранится в загрузочной записи базы данных и потому всегда доступен при восстановлении.

Как показано на рисунке 1, ключ шифрования базы данных шифруется с использованием сертификата сервера, который шифруется с помощью DMK базы данных master. DMK базы данных master шифруется с применением SMK. SMK - асимметричный ключ, зашифрованный с помощью Windows DPAPI. Ключ SMK автоматически формируется при первом шифровании любого объекта и привязан к учетной записи SQL Server Service.

Приступим к резервному копированию (Backup). Начнем с того, что резервное копирование (англ. backup) — процесс создания копии данных на носителе (жёстком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном месте их расположения в случае их повреждения или разрушения, соответствующими программами — резервными дубликаторами данных.

Полный бэкап — это фундамент, на котором будет держаться вся ваша система резервирования. По большому счету, это копия базы данных, которая включает в себя вообще все — таблицы, данные, индексы, триггеры, статистику, хранимые процедуры и еще много разного добра. Более того — если вы выбираете вариант динамического резервного копирования (то есть во время резервирования данных пользователи могут полноценно работать с сохраняемой БД), то все изменения, которые пользователи сделают за то время, пока создается бэкап, тоже будут сохранены. Вы можете легко вернуть базу данных к тому состоянию, в котором она была в какой-то определенный момент, если у вас есть полный бэкап, сделанный в это время.

Бэкап лога отличается от полного бэкапа тем, что в него входят исключительно изменения базы данных (то есть операции INSERT, UPDATE и DELETE) с момента последнего бэкапа, будь то полный бэкап, дифференцированный или предыдущий бэкап лога. Поскольку объем сохраняемых данных крайне мал, такой тип резервирования намного быстрее, требует меньше ресурсов и занимает меньше места на диске. Недостатков, увы, тоже хватает. В первую очередь, бэкап лога бесполезен, если у вас нет хотя бы одного полного бэкапа. Объясняется это тем, что в таком логе не сохраняется никакая информация о таблицах, индексах, хранимых процедурах и так далее. Вторым существенным недостатком является то, что если с момента последнего полного бэкапа вы успели сделать сотню бэкапов лога, а потом случилась беда, то прежде, чем вы восстановите заветный сотый, вам потребуется восстановить не только полный бэкап, но и предыдущие девяносто девять бэкапов лога, да к тому же в правильном порядке. Согласитесь, приятного в такой перспективе не очень много. Еще одна немаловажная особенность заключается в том, что бэкап лога доступен только для тех баз данных, у которых указан FULL или BULK LOGGED режим восстановления.

Дифференцированный бэкап — это что-то среднее между полным бэкапом и бэкапом лога. В нем сохраняются изменения, сделанные с момента последнего полного бэкапа по настоящее время. Главным его преимуществом по сравнению с бэкапом лога является то, что для полного восстановления базы данных вам достаточно восстановить только лишь полный и последний дифференцированный бэкапы. Недостатком же является то, что вы не можете восстановить промежуточное состояние базы данных на какой-то момент времени — история изменения БД в дифференцированном бэкапе не сохраняется. Как и бэкап лога, такой бэкап бесполезен, если у вас нет полной копии базы данных.

Важно:

•Используйте DBCC CHECKDB для проверки каждой базы данных перед копированием, это своевременно предупредит вас о надвигающихся проблемах.

•Используйте опцию BACKUP WITH CHECKSUM, чтобы убедиться, что все прошло хорошо. Недостатком такого решения является то, что для больших баз данных проверка контрольной суммы может серьезно загрузить систему.

•Если у вас проблемы с дисковым пространством, доступным для хранения бэкапов, используйте опцию BACKUP WITH COMPRESSION.

Сжатие позволяет уменьшить итоговый размер файла с бэкапом в несколько раз, обычно в 3-5. Как и в случае с проверкой контрольной суммы, такая операция очень серьезно влияет на производительность, особенно для больших баз данных.

•Создавайте отдельный файл для каждого бэкапа, не храните все бэкапы в одном файле. В таком случае, если с этим файлом что-то случится, вы потеряете один бэкап, а не все.

•Естественно, не храните бэкап на том же диске, на котором хранится ваша БД.

•Если существует угроза, что кто-то посторонний будет иметь доступ к вашим бэкапам, используйте парольную защиту или шифрование.

•Автоматизируйте процесс создания бэкапов. Это просто, и вы будете уверены, что у вас всегда есть свежая копия базы данных.

•Бэкапы бесполезны, если вы не можете их использовать. Поэтому регулярно тренируйтесь в восстановлении баз данных, чтобы при необходимости вы могли в стрессовой ситуации сделать все быстро и безошибочно.

ВЫПОЛНЕНИЕ РАБОТЫ

Открываем "Безопасность" => Нажимаем правой кнопкой мыши по "Имена входа" => Выбираем "Создать имя входа...". (Рис. 2)

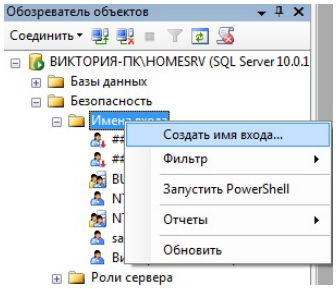


Рис. 2 – Порядок открытия окна «Создать имя входа»

Заполняем строку "Имя входа:". Выбираем "Проверка подлинности SQL Server". Заполняем строки "Пароль" и "Подтверждение пароля. Убираем галочку с "Требовать использование политики паролей". Выбираем нужную базу данных. Выбираем "Язык по умолчанию:" - "Russian". (Рис. 3)

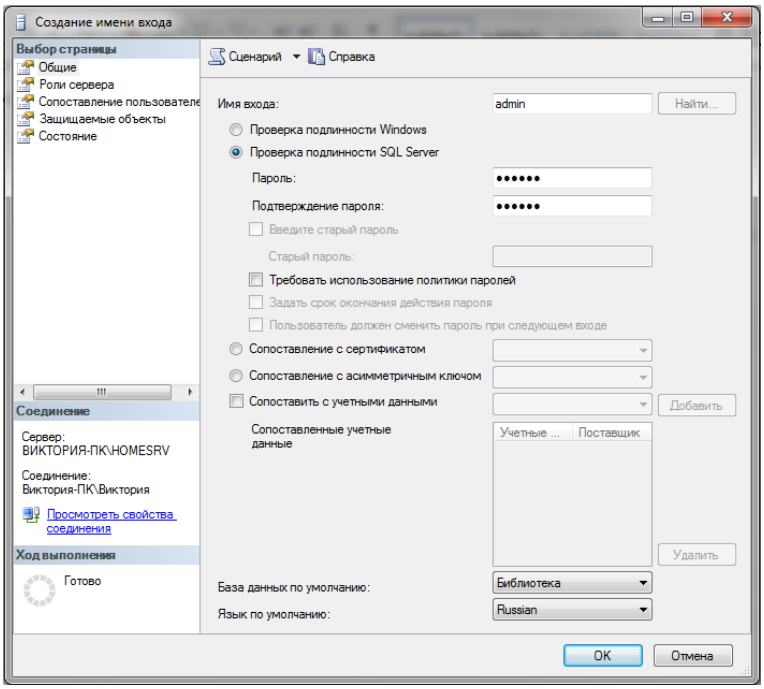


Рис. 3 – Общие параметры «Имя входа» - admin

Переходим во вкладку "Роли сервера:". Выделяем всё галочками. (Рис. 4)

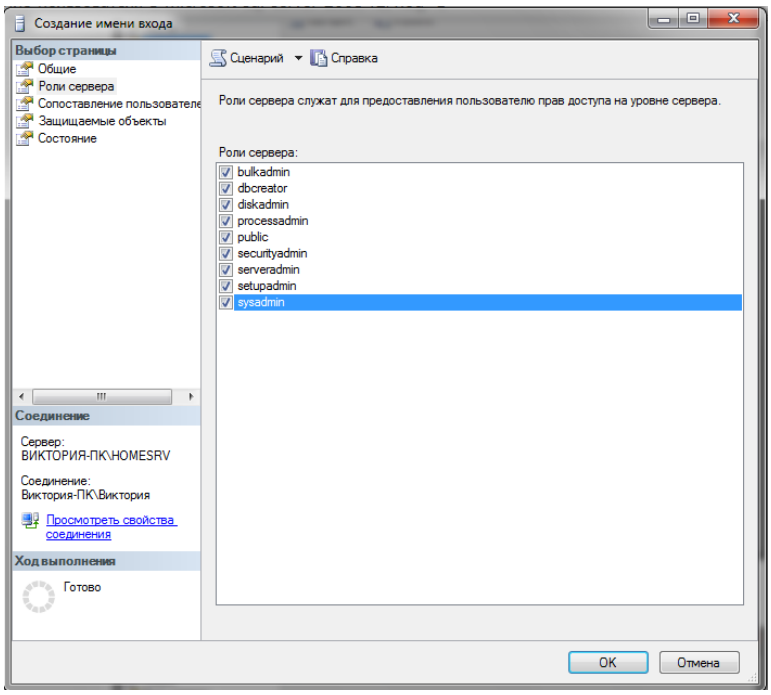


Рис. 4 – Вкладка «Роли сервера» - admin

Переходим во вкладку "Сопоставление пользователей". Выбираем БД `Библиотека` и назначаем роли сервера пользователю admin. (Рис. 5)

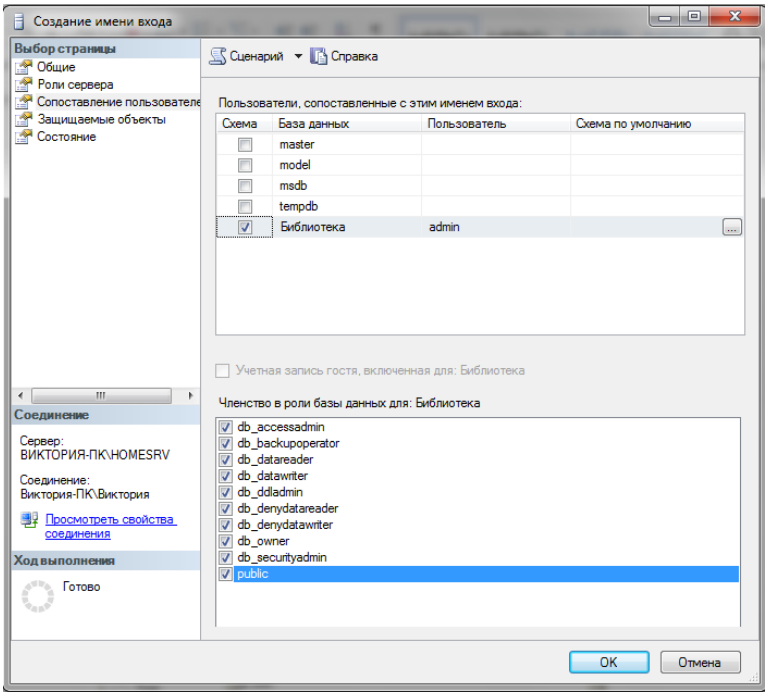


Рис. 5 – Вкладка «Сопоставление пользователей» - admin

Сохраняем изменения нажав на кнопку ОК, и переходим к созданию пользователя «user», аналогичным способом. (Рис. 6, 7)

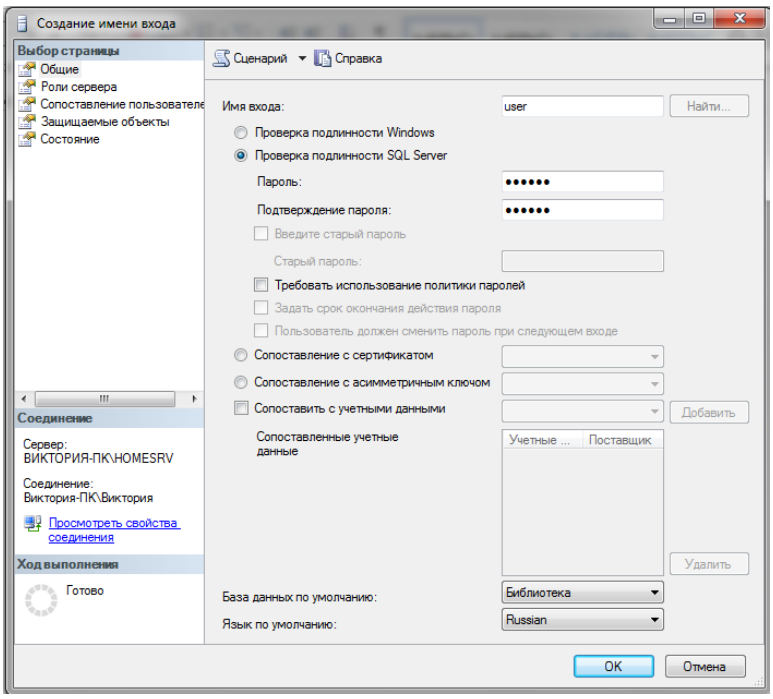


Рис. 6 – Вкладка «Общие» - user

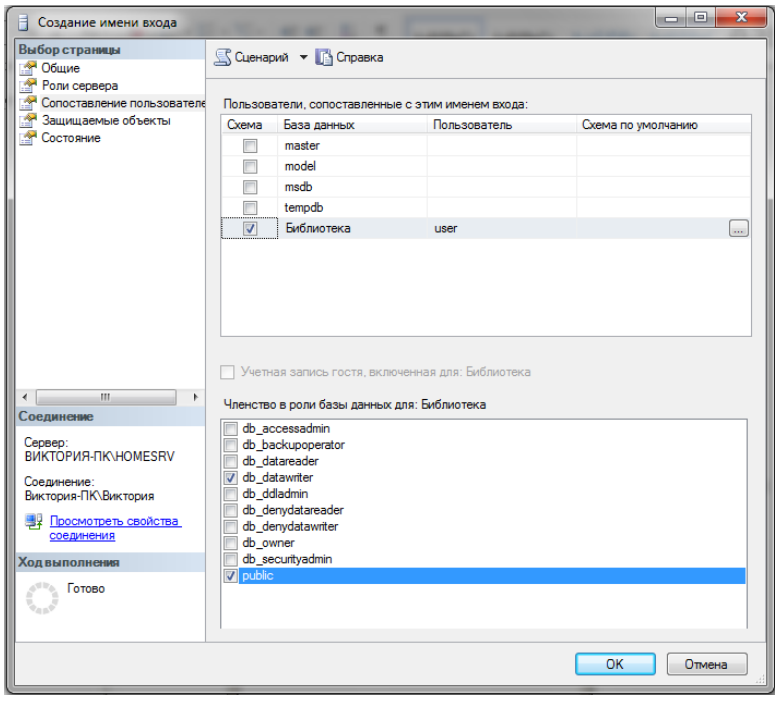


Рис. 7 – Вкладка «Сопоставление пользователей» - user

Пользователи user и admin успешно добавились в обозреватель объектов. (Рис. 8)

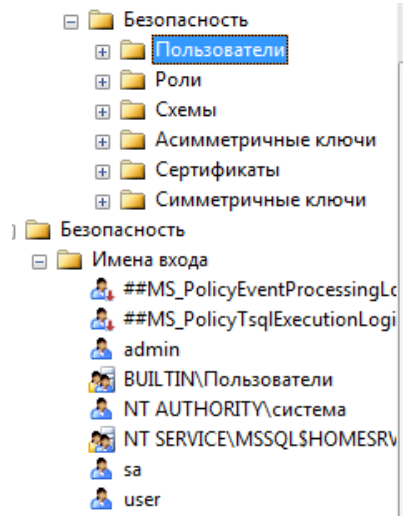


Рис. 8 – Отображение user и admin в обозревателе объектов

Теперь приступим к созданию BackUp для нашей базы данных. Первое что нам необходимо сделать, это убедиться, что Агент SQL Server установлен и работает. Для этого запустим оснастку «Службы» («Пуск» (Start) — «Администрирование» (Administrative Tools) — «Службы» (Services)) и в списке служб найдем службу «Агент SQL сервер». (Рис. 9)

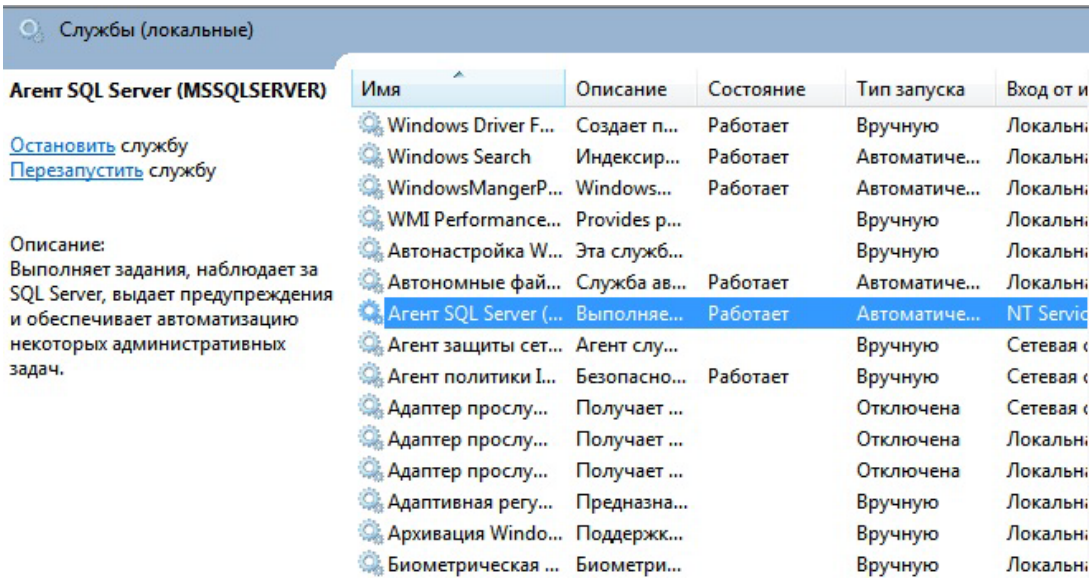


Рис. 9 – Окно «Службы»

Откроем свойства этой службы (кликнув по ней 2 раза) и убедимся, что:

1. Тип запуска стоит «Автоматически» (Startup type: Automatic);
2. Состояние «Работает» (Service status: Started. (Рис. 10)

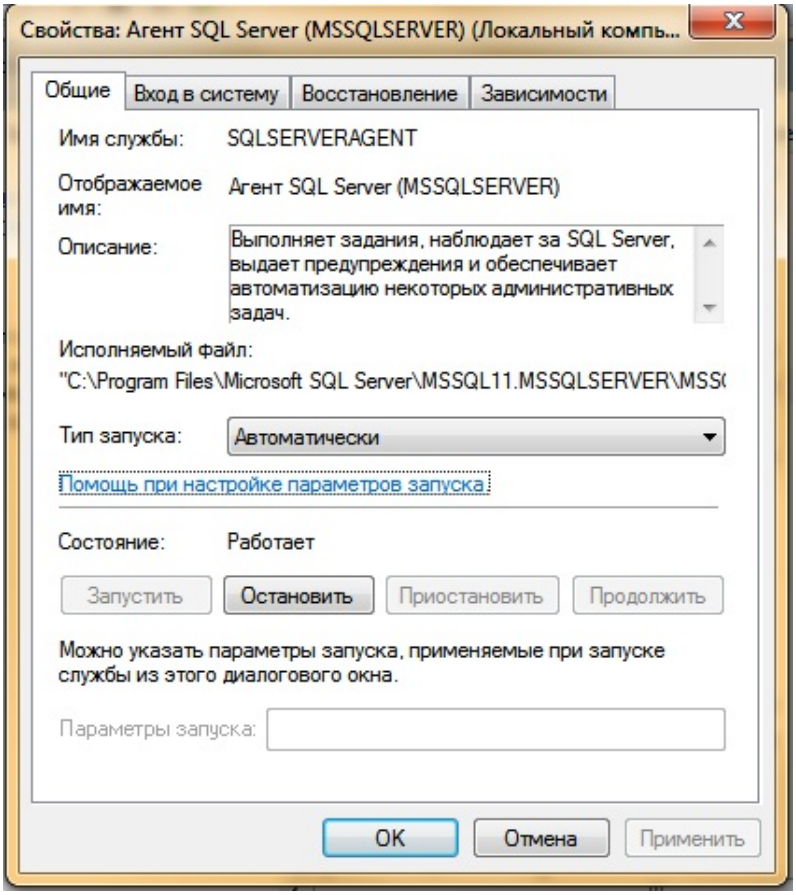


Рис. 10 – Выбор свойств Агент SQL Server

Теперь запустим SQL Sever Management Studio («Пуск» — «Все программы» — «Microsoft SQL Server 2008 R2» — «Средства SQL Server 2008 R2«) и введем данные для авторизации.

После чего, еще раз убедимся, что Агент SQL Server работает (в обозревателе объектов должна быть вкладка «Агент SQL Server» (SQL Server Agent) с зеленой иконкой слева. (Рис. 11)

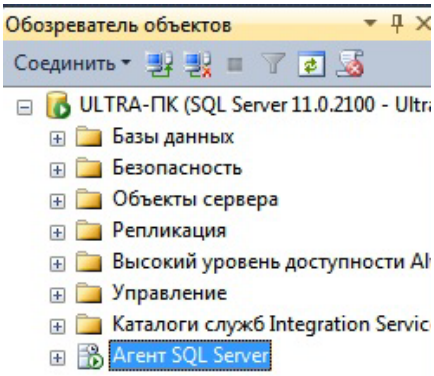


Рис. 11 – Отображение «Агент SQL Server» в обозревателе объектов

Теперь перейдем непосредственно к созданию плана обслуживания. В обозревателе объектов (Object Explorer) раскроем вкладку «Управление» (Management), кликнем правой кнопкой мыши по вкладке «Планы обслуживания» (Maintenance Plans) и в контекстном меню выберем «Мастер планов обслуживания» (Maintenance Plan Wizard). (Рис. 12)

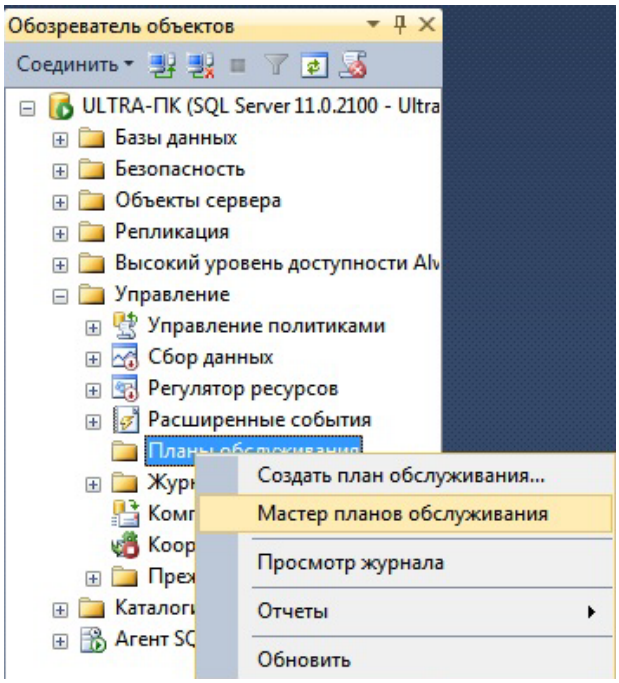


Рис. 12 – Порядок вызова «Мастер планов обслуживания»

В запустившемся мастере планов обслуживания на странице приветствия нажимаем «Далее» (Next) и в следующем окне вводим имя и описание нового плана. Затем необходимо определиться с расписанием, по которому будет выполняться данный план обслуживания. Для этого установим переключатель на «Единое расписание для всего плана или без расписания» (Single schedule for the entire plan ore no schedule) и нажмем «Изменить…» (Change…) для назначения расписания. (Рис. 13)

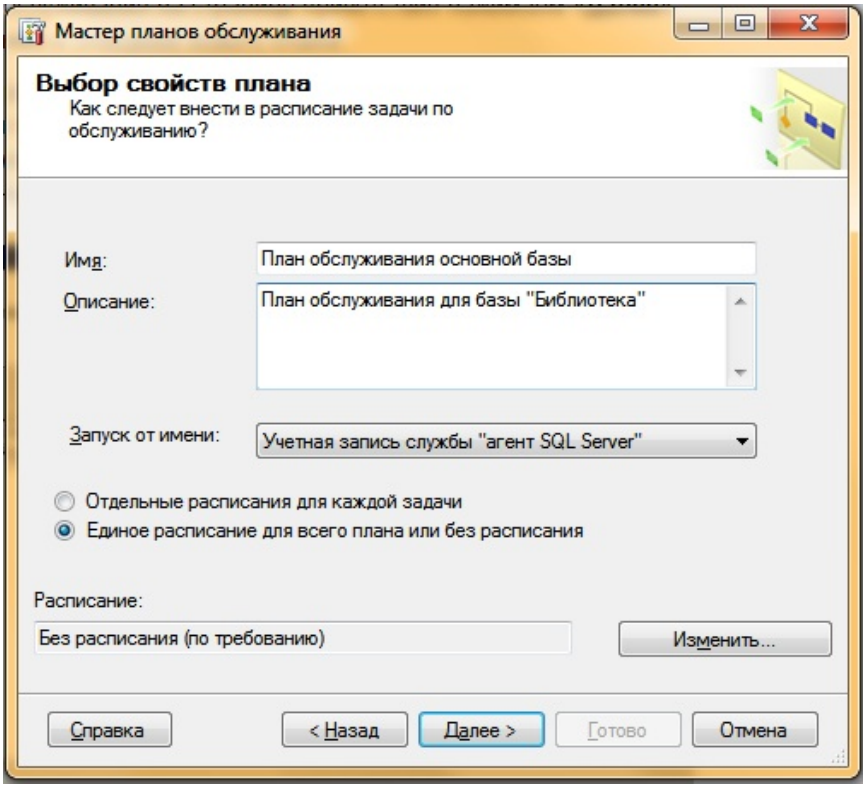


Рис. 13 – Окно «Мастер планов обслуживания»

Откроется окно «Создание расписания задания». Здесь зададим те параметры, согласно которым должен выполняться план обслуживания и нажмем «ОК». (Рис. 14)

В данном примере это:

1. Выполняется — «Еженедельно» (Occurs — Weekly);
2. Повторяется каждые — «1 нед.» в «Воскресенье» (Recurs every: 1 week(s) on Sunday);
3. Выполняться один раз в день в: — «2:00:00» (Occurs once at: «2:00:00»)

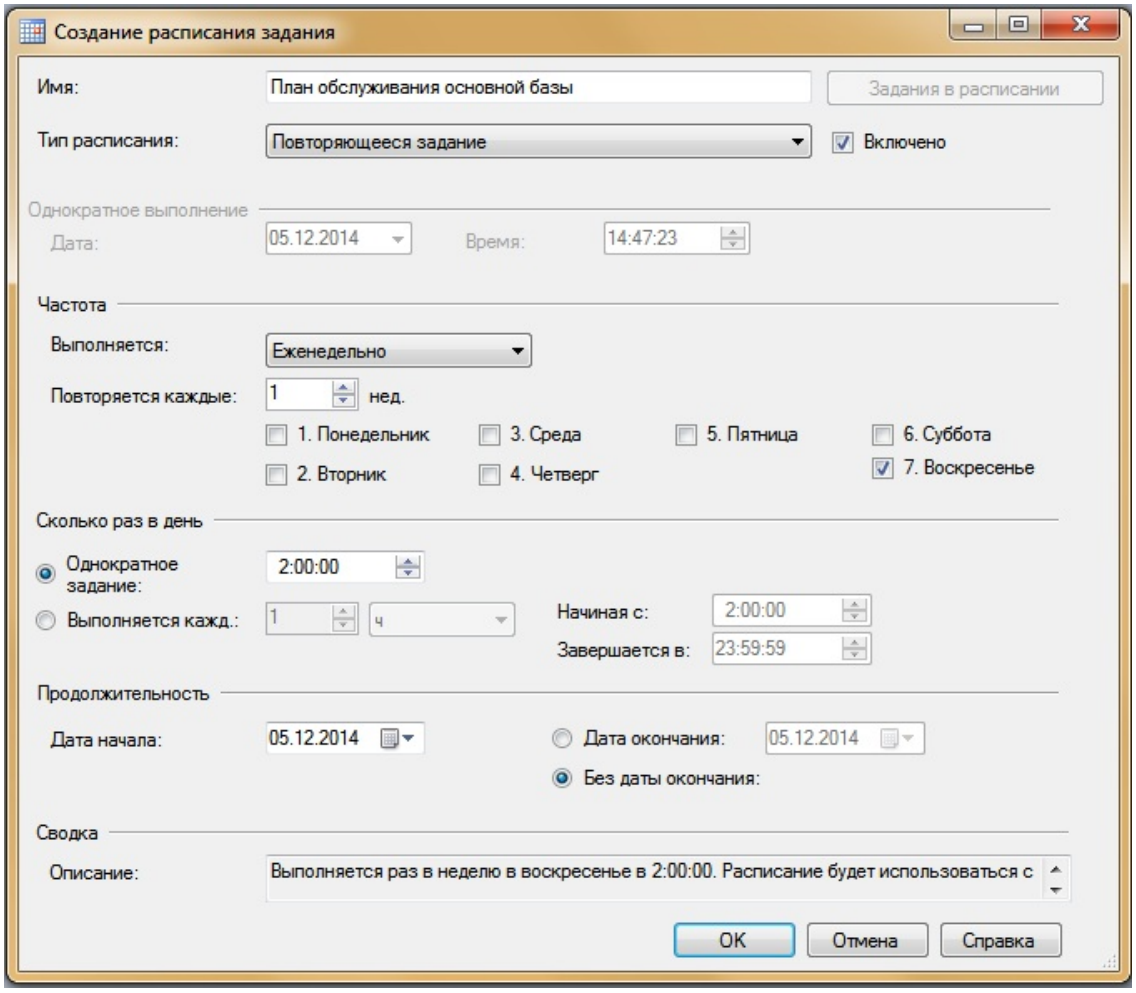


Рис. 14 – Окно «Создание расписания задания»

Еще раз убедимся, что расписание задано верно, и нажмем «Далее» (Next).

В следующем окне (Рис. 15) выберем те задачи, которые будет выполнять наш план обслуживания. В данном примере это:

1. Проверка целостности базы данных (Check Database Integrity);
2. Резервное копирование базы данных (полное) (The Back Up Database (Full));

Заметьте, что для каждой задачи приводится ее краткое описание в поле снизу. Выбрав необходимые задачи, жмем «Далее» (Next).

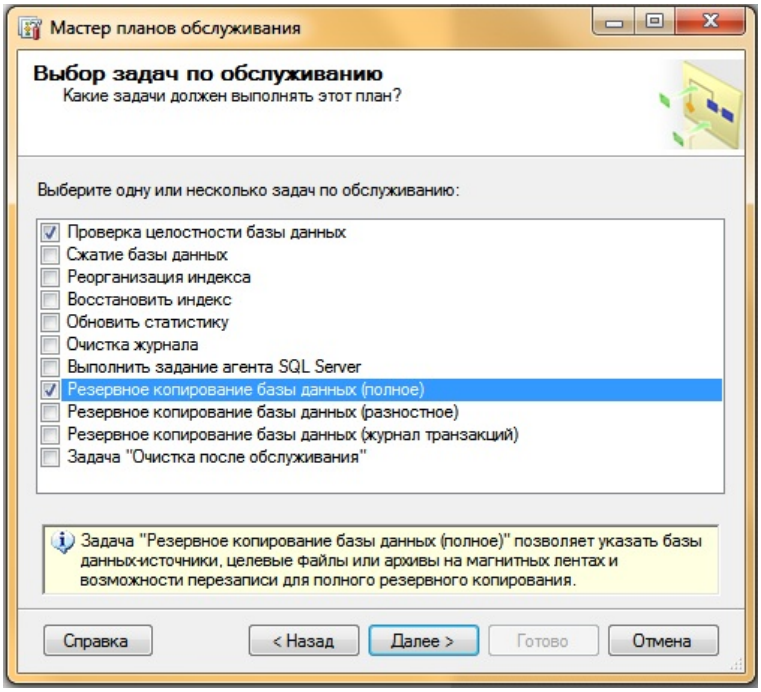


Рис. 15 – Выбор задач плана обслуживания

Теперь необходимо задать порядок выполнения задач, используя кнопки «Вверх…» (Move Up) и «Вниз…» (Move Down). Установив порядок, жмем «Далее» (Next). (Рис. 16)

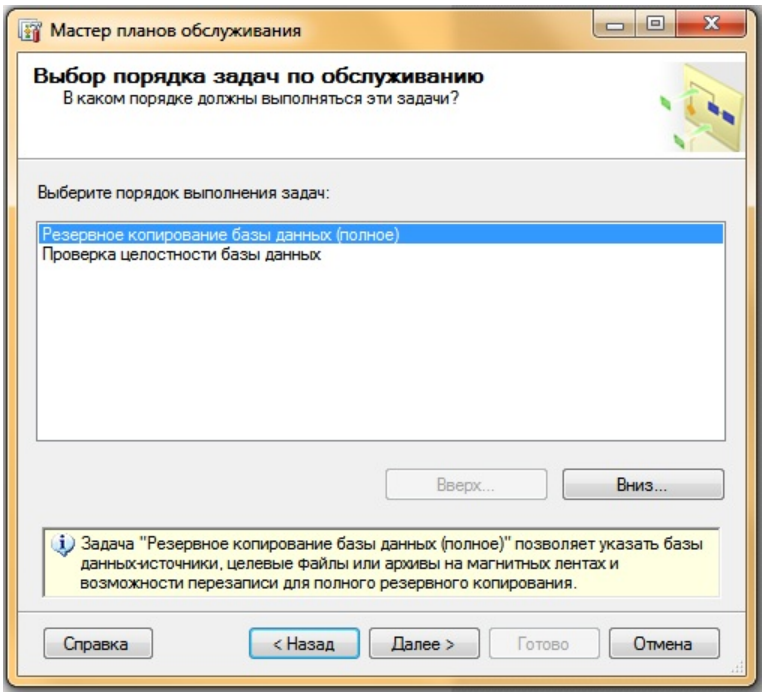


Рис. 16 – Задание порядка выполнения задач

В открывшемся окне требуется задать параметры для каждой задачи в плане. Первая задача в нашем списке — это «Копирование БД (полное)» (Back Up Database (Full)).

Прежде всего необходимо выбрать базы данных для резервного копирования, нажав на кнопку выбора списка «Определенные базы данных» (Select one ore more). Выбрав необходимые для резервного копирования базы данных, нажимаем «ОК». (Рис. 17)

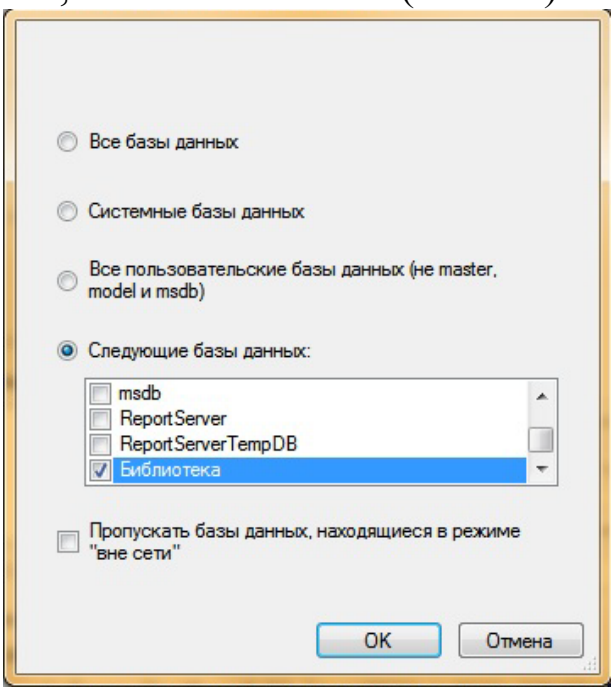


Рис. 17 – Выбор базы данных для резервного копирования

Ниже зададим размещение и срок хранения резервных копий, а также установим дополнительные параметры:

1. Если установить переключатель «Создать файл резервной копии для каждой базы данных» (Create a backup file for every database), то при выполнении задания в выбранной директории будет создаваться несколько файлов резервных копий с именами, соответствующими названиям баз данных. Ну а установка флага «Создавать вложенный каталог для каждой базы данных» (Create a sub-directory for each database) разложит файлы по отдельным папкам. Обратите внимание, что необходимо оставить заполненным расширение файла резервной копии. (Рис. 18)
2. Установка флага «Срок действия резервного набора данных истекает» (Backup set will expire) указывает SQL-серверу, когда этот набор может быть перезаписан без явного пропуска проверки на истечение срока.

3. Для наибольшей надежности, можно установить флаг «Проверять целостность резервной копии» (Verify backup integrity).
4. Также рекомендую выбрать режим «Сжимать резервные копии» (Compress backup) для экономии дискового пространства.

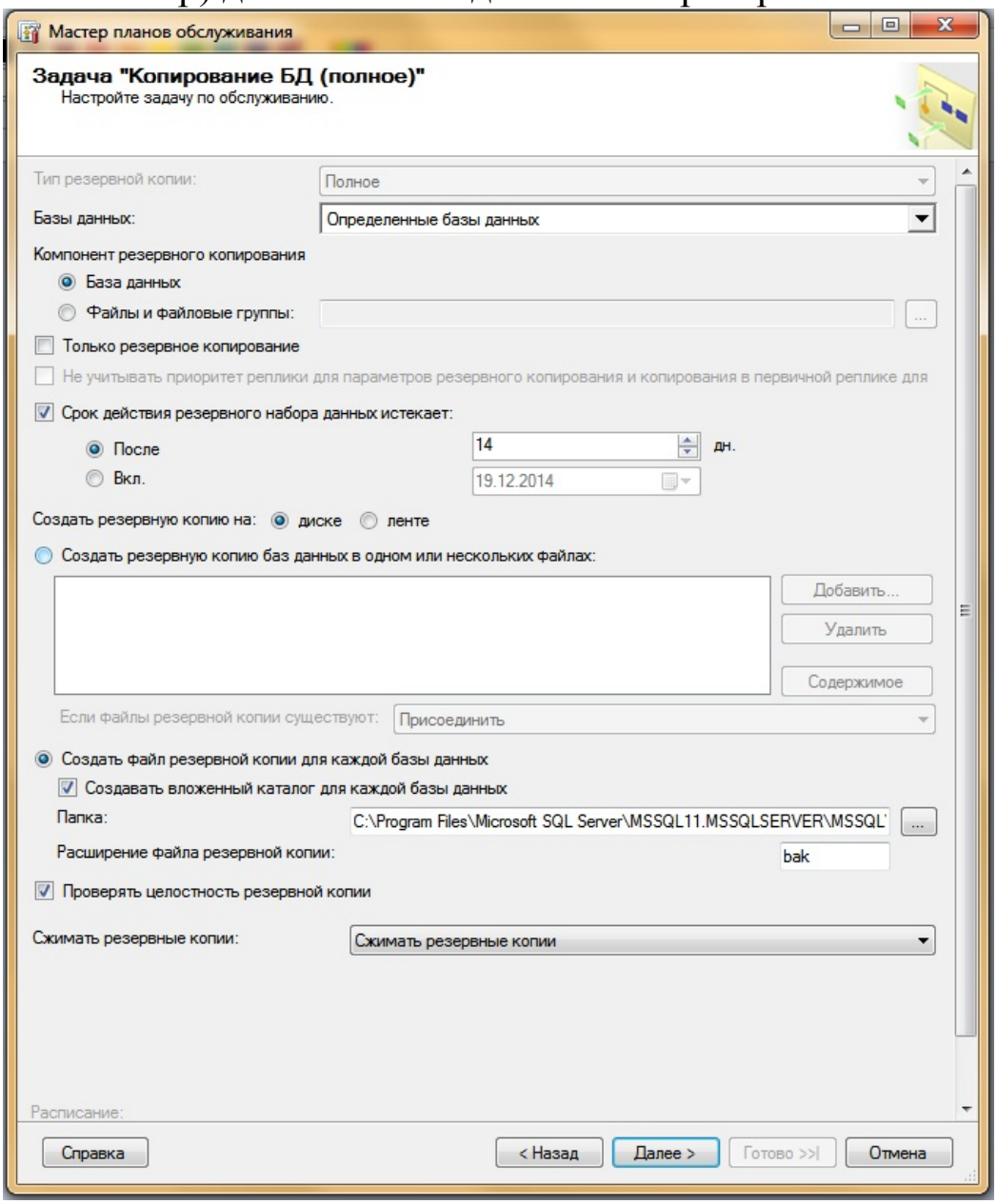


Рис. 18 – Установка параметров резервных копий

Если дисковое пространство ограничено, можно также выбрать один файл для хранения резервной копии, который будет перезаписываться после каждого выполнения плана обслуживания. Для этого установим соответствующий переключатель на «Создать резервную копию баз данных в одном или нескольких файлах» и укажем соответствующее имя файла (будьте внимательны, файл резервной копии следует задавать с расширением. bak), а также выберем режим «Перезаписать» в случае,

если файлы резервной копии существуют. Определившись с настройками жмем «Далее» (Next). (Рис. 19)

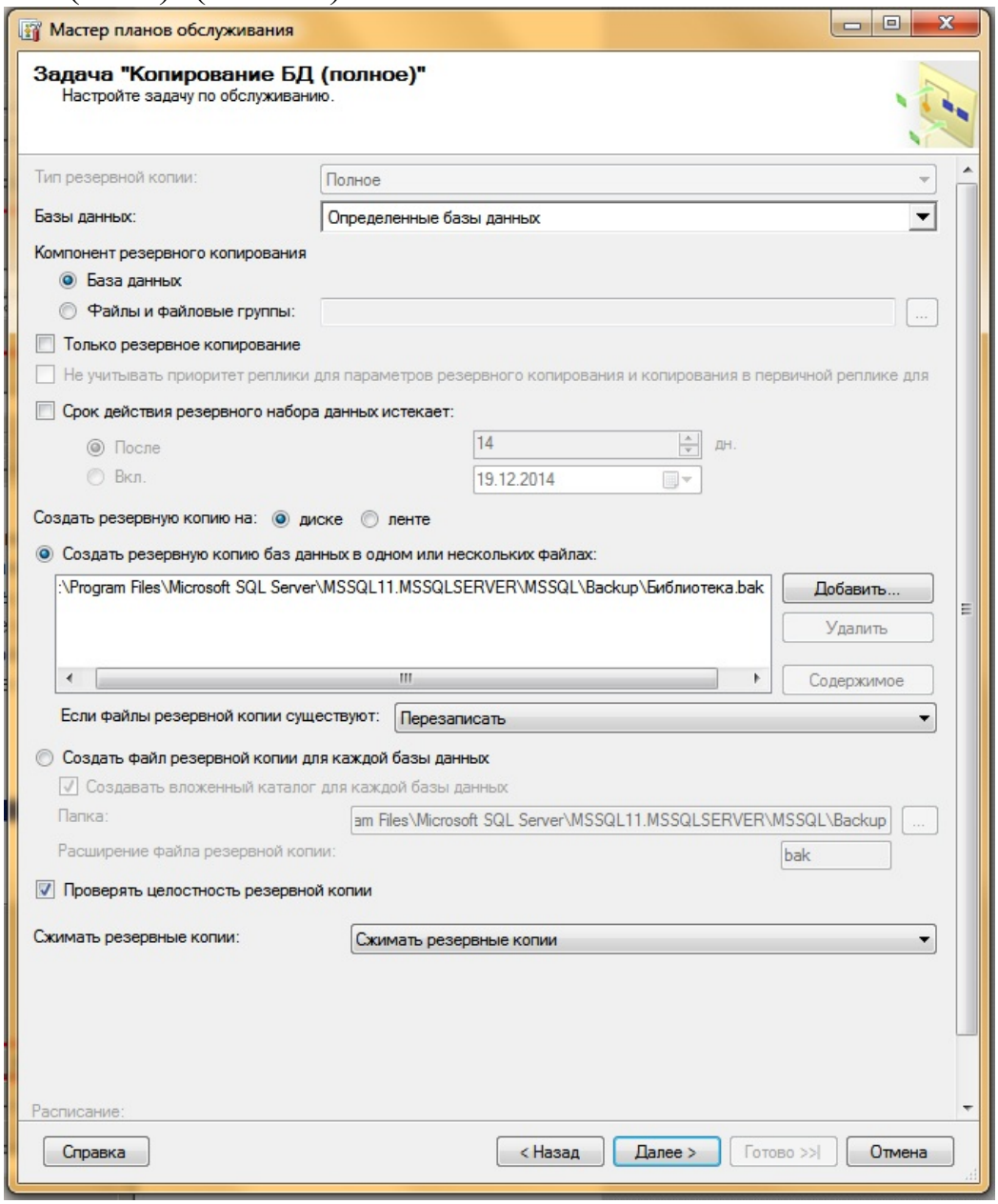


Рис. 19 – Установка параметров резервных копий

Теперь очередь задачи «Проверка целостности базы данных» (Database Check Integrity). Для нее всего лишь необходимо выбрать базу данных. В нашем примере это все та же база данных, что и на предыдущем шаге. Определившись с базами, жмем «Далее» (Next). (Рис. 20)

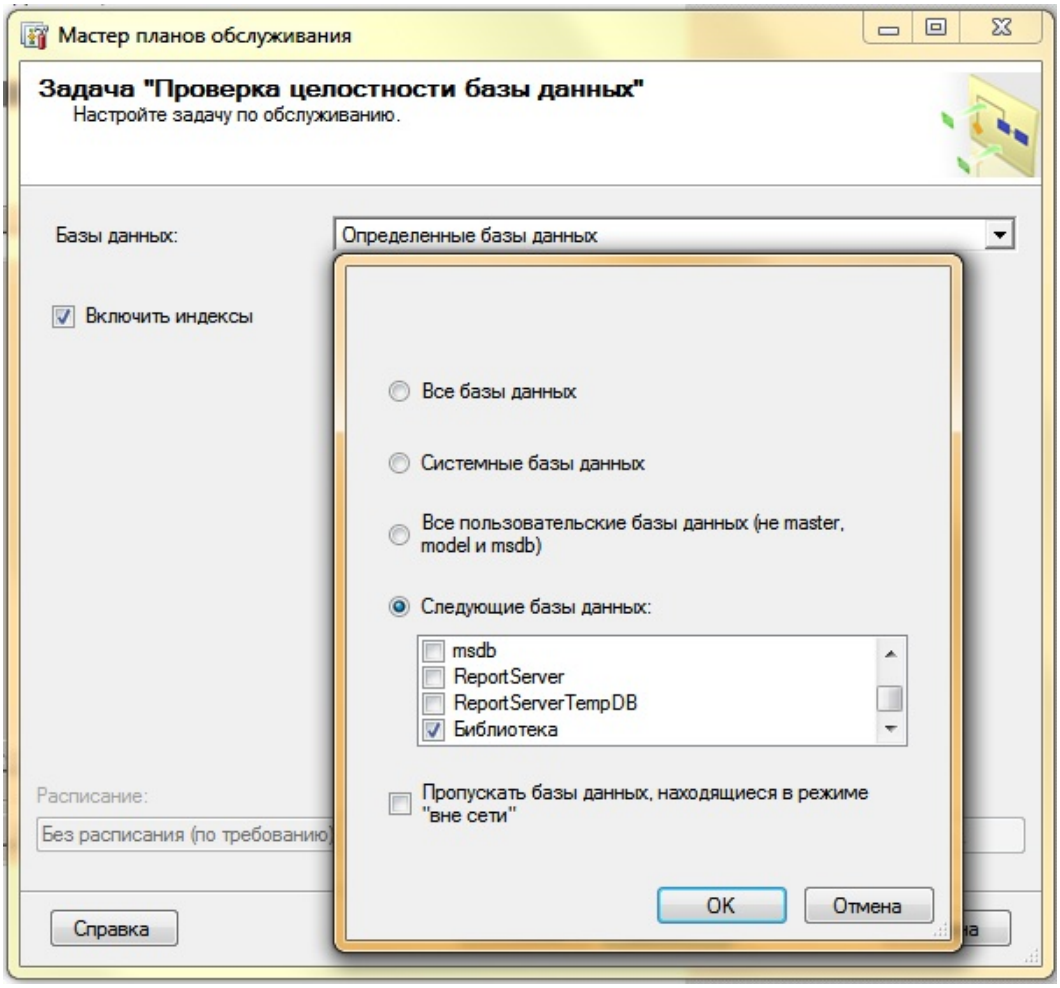


Рис. 20 – Проверка целостности базы данных

В следующем окне возможно выбрать директорию, куда будет сохраняться лог выполнения задания, а также указать оператора SQL Server для отправки отчета по электронной почте. Задав параметры, снова жмем «Далее». (Рис. 21)

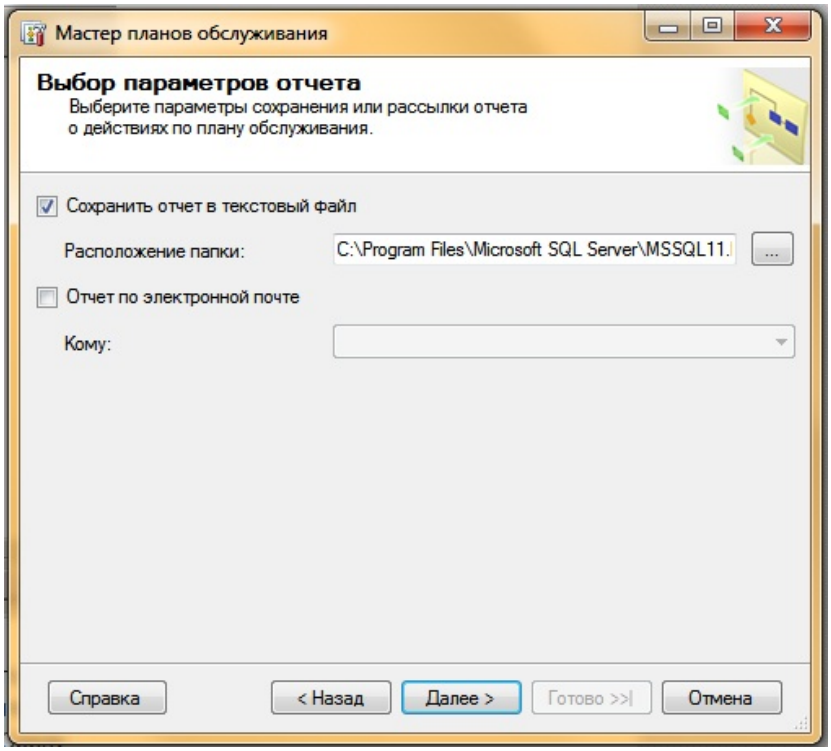


Рис. 21 – Возможность отправки отчета по электронной почте

Проверим еще раз все настройки плана обслуживания, и если все верно, нажимаем «Готово» (Finish). (Рис. 22)

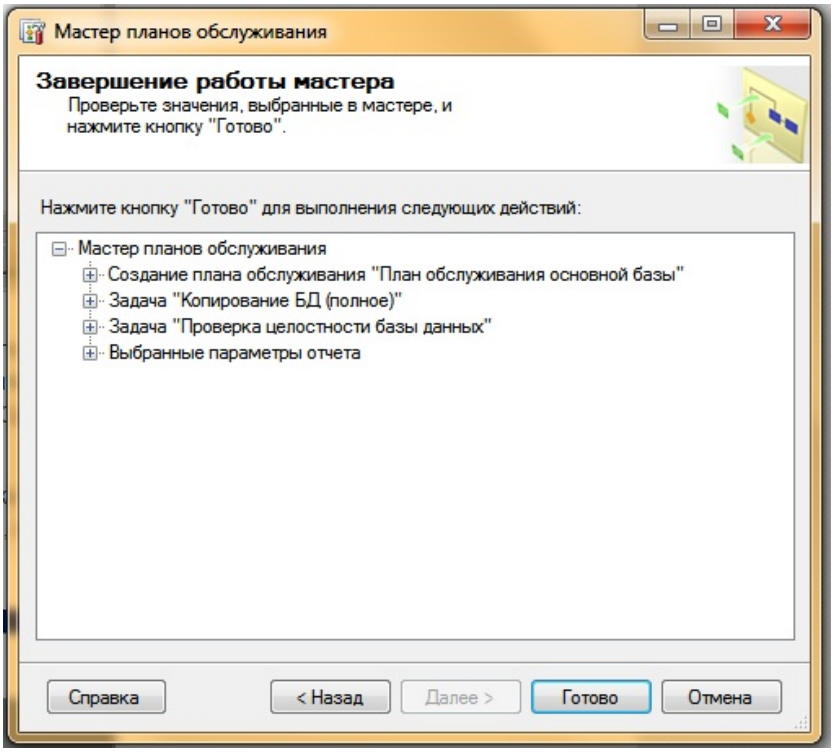


Рис. 22 – Проверка настроек плана обслуживания

Мастер начнет построение плана обслуживания. Если мастер не обнаружит ошибок, то увидим сообщение об успешном построении

плана. В противном случае необходимо устранить ошибки и повторить процедуру снова. Закроем окно, нажав «Закрыть» (Close) и перейдем в Среду Microsoft SQL Server Management Studio.

Здесь, раскрыв вкладку «Планы обслуживания» (Maintenance Plans) увидим наш только что созданный план. Чтобы проверить его работу, кликнем по нему правой кнопкой мыши, и в контекстном меню выберем пункт «Выполнить» (Execute). (Рис. 23)

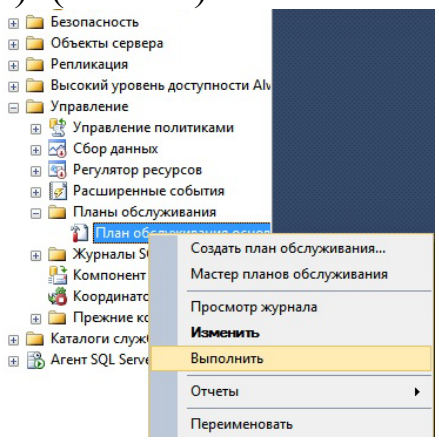


Рис. 23 – Отображение плана обслуживания в обозревателе объектов

После чего запустится окно выполнения плана обслуживания, в котором, спустя необходимое количество времени, должно появиться сообщение об успешном выполнении. (Рис. 24)

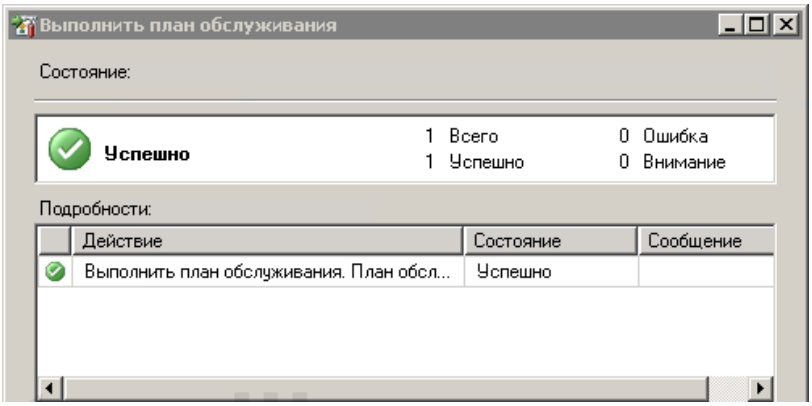


Рис. 24 – Успешное выполнение плана обслуживания.

А в соответствующих директориях должны появиться файл резервной копии и файл лога выполнения плана. (Рис. 25)

Рис. 25 – Файл резервной копии и файл лога

Открыв, файл лога, вы должны увидеть примерно следующее:

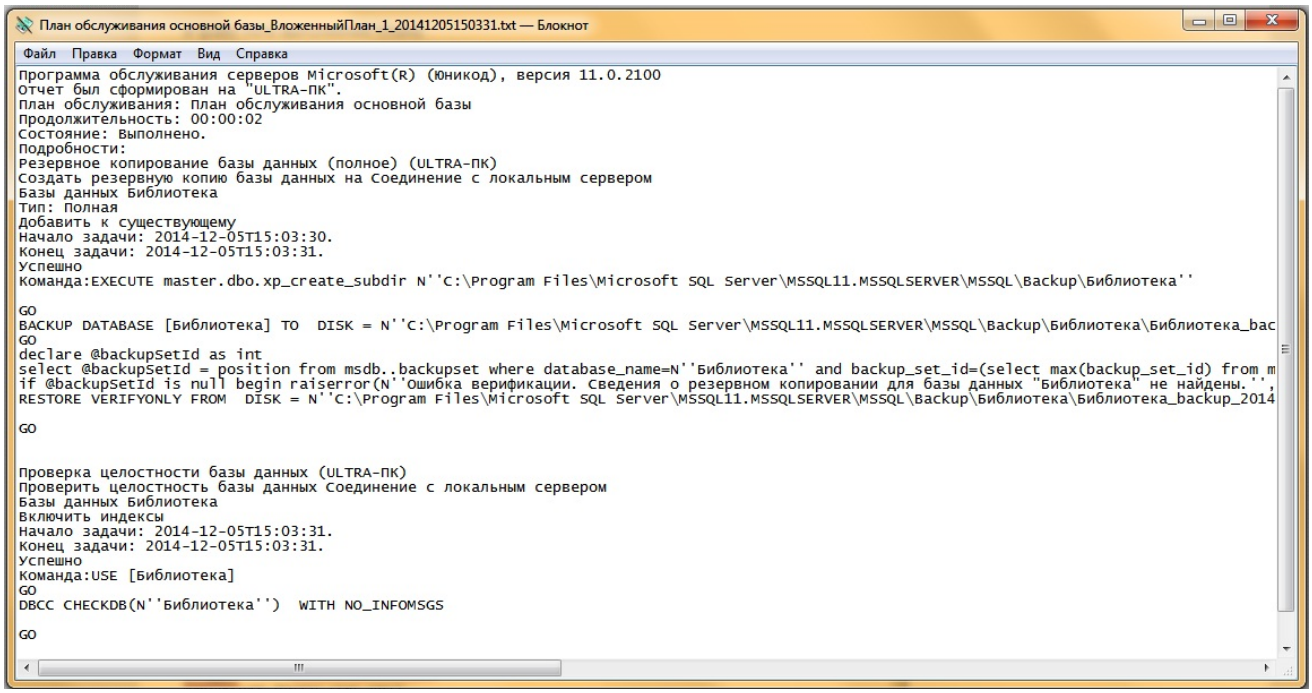


Рис. 26 – План обслуживания базы данных «Библиотека»

4. РАЗРАБОТКА КЛИЕНТСКОГО ИНТЕРФЕЙСА ДЛЯ БД И СОЗДАНИЕ ОТЧЕТОВ В КЛИЕНТСКОМ ПРИЛОЖЕНИИ

ЦЕЛЬ РАБОТЫ

Получить доступ к данным СУБД, разработать клиентское приложение с удобным пользовательским интерфейсом, позволяющим осуществлять просмотр, выборку и изменение содержимого БД. Формирование отчетов средствами клиентского приложения.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Компоненты Delphi для работы с базами данных были созданы в расчете на работу с SQL и архитектурой клиент/сервер. При работе с ними вы можете воспользоваться характеристиками расширенной поддержки удаленных серверов. Delphi осуществляет эту поддержку двумя способами. Во-первых, непосредственные команды из Delphi позволяют разработчику управлять таблицами, устанавливать пределы, удалять, вставлять и редактировать существующие записи. Второй способ заключается в использовании запросов на языке SQL, где строка запроса передается на сервер для ее разбора, оптимизации, выполнения и передачи обратно результатов.

Для доступа к данным SQL Server из клиентских приложений, написанных на языке Delphi для платформы Win32 можно применять хорошо известные универсальные технологии Borland Database Engine (BDE) и DBExpress, разработанные фирмой Borland. Однако рекомендуемый подход – использование технологии Microsoft ActiveX Data Objects (ADO), оптимизированной для SQL Server.

Технология ADO основана на возможностях COM, а именно интерфейсов OLE DB. Базовый набор интерфейсов OLE DB предустановлен во всех версиях Microsoft Windows, поэтому при переносе приложения на другой компьютер для его работоспособности достаточно лишь правильно настроить провайдер OLE DB. Провайдер OLE DB представляет собой COM-сервер, предоставляющий набор интерфейсов для доступа к данным, и «скрывающий» особенности конкретных источников данных. Провайдеры OLE DB разработаны для большинства СУБД (Oracle, Interbase и др.) и многих других источников данных. Часть провайдеров (например, OLE DB Provider for SQL Server) уже установлена в системе, другие доступны для скачивания в Internet.

Технология ADO – надстройка над интерфейсом OLE DB, облегчающая его использование прикладными программистами.

Технология ADO и интерфейсы OLE DB предоставляют приложениям единый способ доступа к источникам данных различных типов. Приложение, использующее ADO, может однотипно работать с данными, хранящимися на сервере SQL, с электронным таблицами и локальными СУБД. Согласно терминологии ADO, любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым при помощи провайдера взаимодействует приложение.

В результате приложение обращается не напрямую к источнику данных, а к объекту OLE DB, который представляет данные в виде таблицы БД или результата выполнения запроса SQL. Такая архитектура позволяет сделать набор объектов и интерфейсов открытым и расширяемым. Набор объектов и соответствующий провайдер могут быть созданы для любого хранилища данных без изменения исходной структуры ADO. При этом существенно расширяется само понятие данных. Можно разработать набор объектов и интерфейсов и для не табличных данных, например, графических данных, древовидных структур, данных CASE-инструментов и др.

В Delphi на странице ADO палитры компонентов расположены компоненты доступа к данным, инкапсулирующие технологию ADO. Общая методика их использования построена по тем же принципам, что и у остальных компонентов доступа к данным (BDE, IBExpress, DBExpress и др.), однако внутренняя организация совсем другая. Это удобно, так как программист может с успехом использовать ранее имевшиеся навыки и опыт работы с другими СУБД. Например, компоненты ADO поддерживают навигацию, работу с наборами данных, кэшируемые изменения (здесь они называются пакетными обновлениями), управление транзакциями. Наиболее серьезное препятствие здесь – научиться мыслить категориями архитектуры клиент-сервер, и не пытаться переносить методы и приемы создания персональных баз, данных в многопользовательскую клиент серверную среду.

С другой стороны, с помощью свойств и методов упомянутых компонентов при необходимости легко обратиться к дополнительным возможностям ADO для более «тонкой» настройки приложения.

Если в ваших приложениях вы собираетесь использовать SQL, то вам непременно придется познакомиться с компонентом TQuery. Компоненты TQuery и TTable наследуются от TDataset. TDataset

обеспечивает необходимую функциональность для получения доступа к базам данных. Как таковые, компоненты TQuery и TTable имеют много общих признаков. Для подготовки данных для показа в визуальных компонентах используется все тот же TDatasource. Также, для определения к какому серверу и базе данных необходимо получить доступ, необходимо задать имя псевдонима. Это должно выполняться установкой свойства aliasName объекта TQuery.

Все же TQuery имеет некоторую уникальную функциональность. Например, у TQuery имеется свойство с именем SQL. Свойство SQL используется для хранения SQL-запроса.

Компонент TDatabase обеспечивает функциональность, которой не хватает TQuery и TStoredProc. В частности, TDatabase позволяет создавать локальные псевдонимы BDE, так что приложению не потребуются псевдонимы, содержащиеся в конфигурационном файле BDE. Этим локальным псевдонимом в приложении могут воспользоваться все имеющиеся TTable, TQuery и TStoredProc. TDatabase также позволяет разработчику настраивать процесс подключения, подавляя диалог ввода имени и пароля пользователя, или заполняя необходимые параметры. И, наконец, самое главное, TDatabase может обеспечивать единственную связь с базой данных, суммируя все операции с базой данных через один компонент. Это позволяет элементам управления для работы с БД иметь возможность управления транзакциями.

ВЫПОЛНЕНИЕ РАБОТЫ

Итак, у нас есть готовая база данных на сервере MSSQL Server 2008 и можно приступить непосредственно к разработке клиентского интерфейса.

1. Открываем Delphi и создаем новый проект. Для удобства, чтобы не создавать много форм в проекте, используется компонента PageControl из вкладки Win32. Таблиц в нашей базе всего 3, поэтому страниц на PageControl у нас тоже 3. Добавление новой страницы достигается при помощи контекстного меню. (Рис. 3)

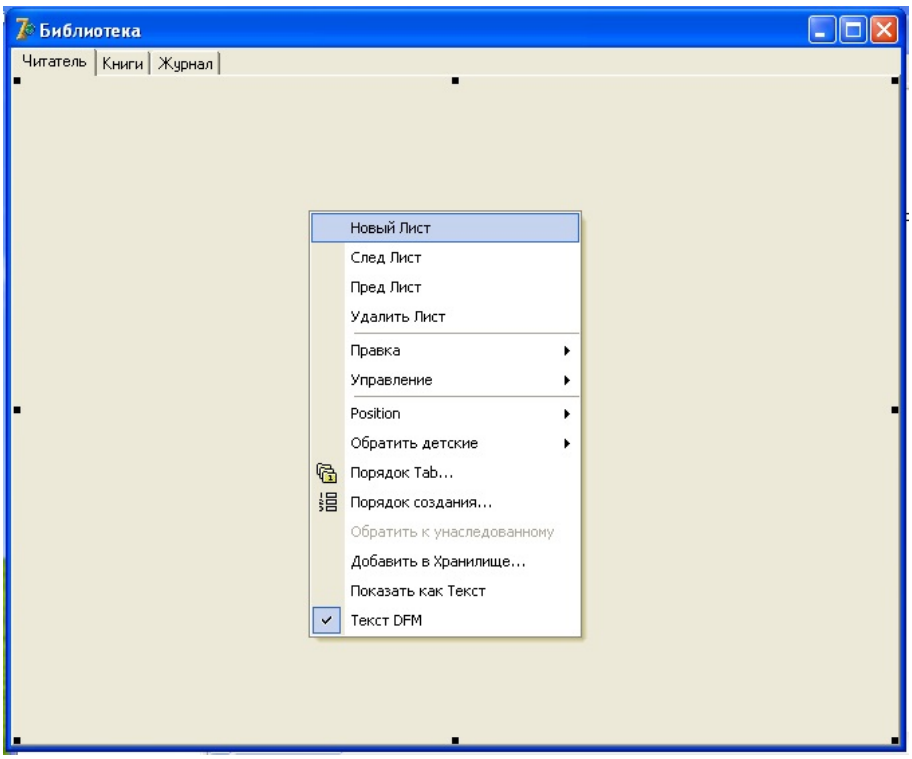
5. 

Рис. 3 - Добавление новой страницы

2. Создаём модуль данных (Data Module): Файл=> Новый=> Модуль данных (Рис. 4)
3. На наш Модуль Данных «ставим» следующие компоненты:

- TDataSource
- TADOConnection
- TADOQuery

TDataSource находится на вкладке Data Access, предназначен для связи нашей сетки отображения данных, с самой БД.

TADOConnection находится на вкладке ADO, предназначен для подключения нашей БД по определенному провайдеру.

TADOQuery находится также на вкладке ADO, предназначен для получение нужных результатов из нашей БД. (Рис. 5). Компоненты можно переименовать с помощью инспектора объектов изменяя поле Name.

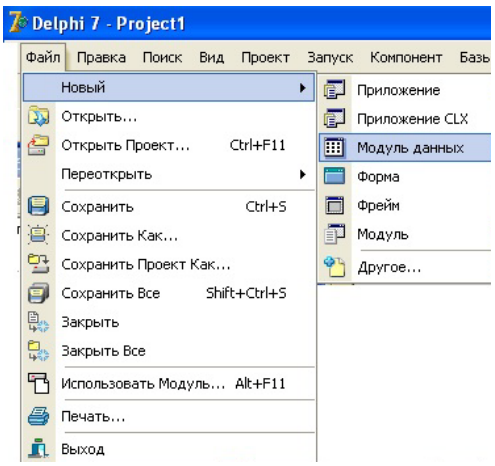
6. 

Рис. 4 - Создание модуля данных

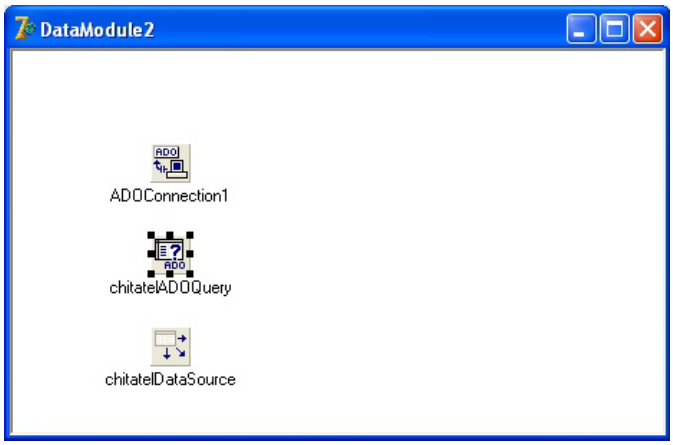


Рис. 5 - Добавление компонентов на модуль Данных

4. Связываем TADOQuery в свойстве Connection с TADOConnection. Из выпадающего списка выбираем имя данного компонента.
5. Связываем TDataSource в свойстве DataSet с TADOQuery из выпадающего списка выбираем имя данного компонента.
6. Выделяем компонент TADOConnection в свойстве ConnectionString нажимаем на кнопку с «…», появится окно следующего вида (Рис. 6). В данном окне нажимаем на кнопку «Build…», появится окно Свойство связи с данными (Рис. 7). В данном окне мы выбираем провайдера, а именно Microsoft OLE DB Provider for SQL Server и нажимаем кнопку «Далее».

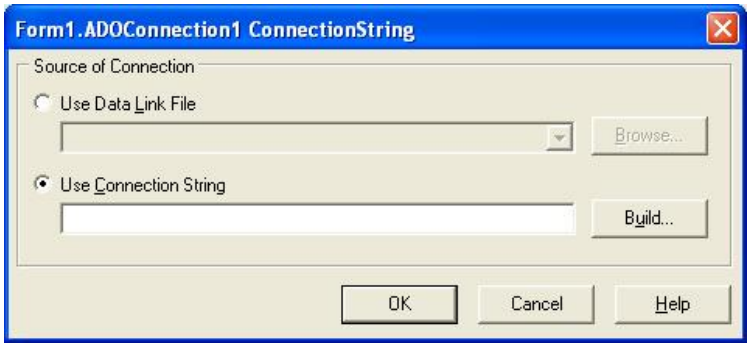
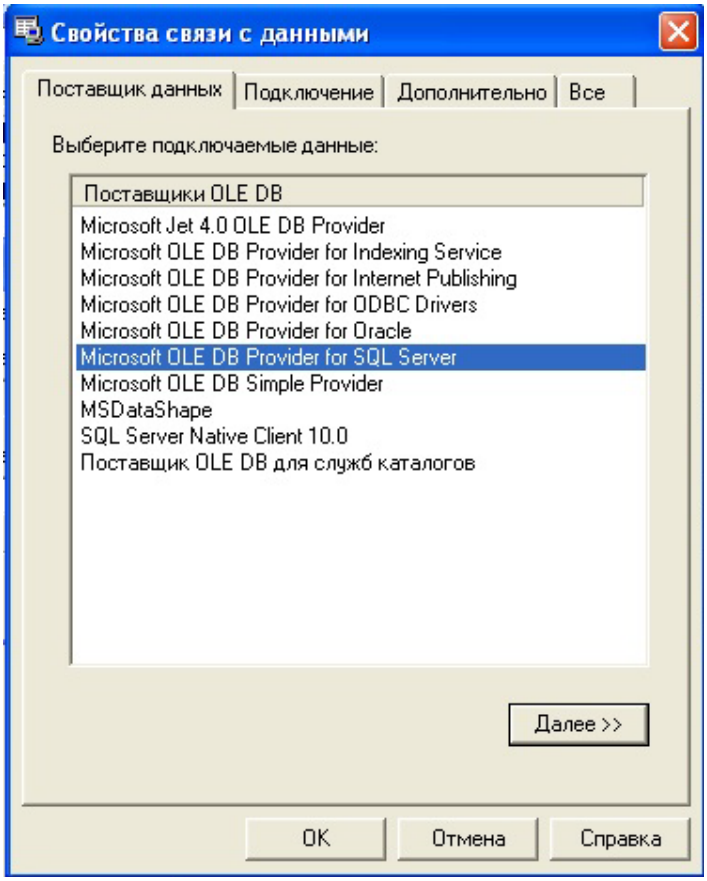


Рис. 6 - Окно компонента TADOConnection- ConnectionString



7.

Рис. 7 - Окно Свойства связи с данными (Поставщик данных)

7. Во вкладке Подключение указываем имя сервера; имя пользователя; пароль (который мы создавали в sql server management studio для нашей базы данных) и выбираем нашу базу данных (Рис. 8). Нажимаем проверить подключение, если всё введено правильно, выскакивает окно, что проверка подключения выполнена, нажимаем ОК. (Рис. 9)

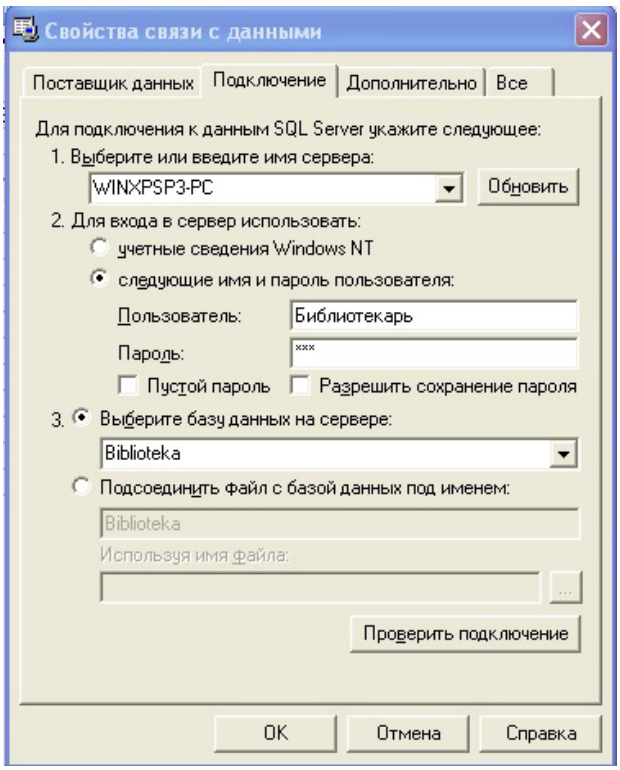


Рис. 8 - Окно Свойства связи с данными (Подключение)

8.

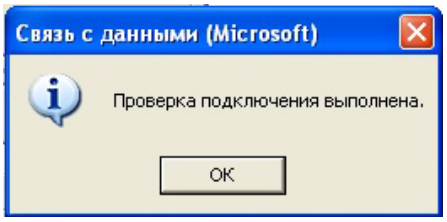


Рис. 9 - Проверка подключения

8. В свойстве ACTIVE TADOConnection ставим True. Заходим в свойства TADOQuery выбираем свойство SQL нажимаем на кнопку с «…» и появляется окно следующего вида (Рис. 10). Пишем в этом окне запрос на выбор всех столбцов из таблицы chitatel нашей базы данных (select * from chitatel) и нажимаем ОК. Ставим свойство ACTIVE в положение TRUE.

9.

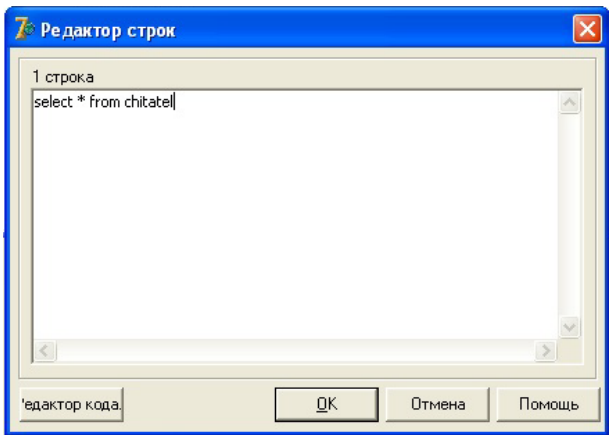


Рис. 10 - Запрос на выбор всех столбцов из таблиц

Итак, у нас есть контакт с сервером, переходим к созданию интерфейса

1. Поместим на форму (на активную страницу компонента PageControl) одну компоненту GroupBox из вкладки Standard и одну компоненту PageControl из вкладки Win32. В Object Inspector в GroupBox в поле Caption напишем “Поиск”. В поле Align соответственно выбираем “alTop” и “alBottom”. Между компонентами GroupBox и PageControl вставим из вкладки EhLib компоненту DBGridEh. В компоненте DBGridEh в поле Align выбираем “alClient” (Рис. 11).

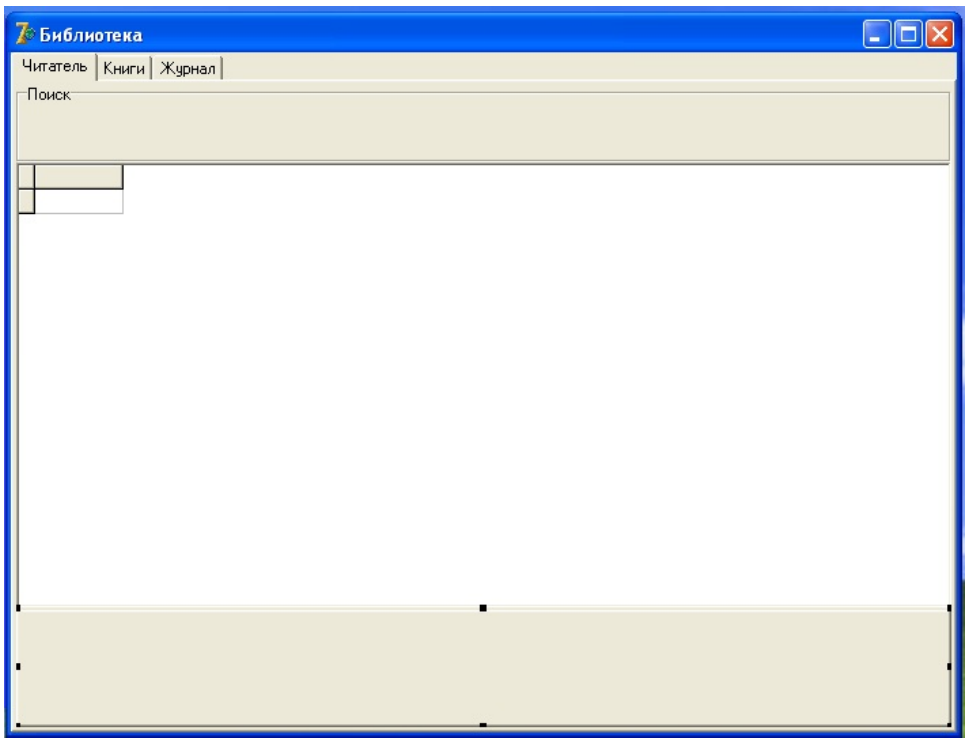


Рис. 11 - Расстановка компонентов

Параметры поля Align мы меняли для того, чтобы при развертывании окна приложения, компоненты также меняли свои размеры.

2. Заполним GroupBox “Поиск”, поместим на него компоненты ComboBox, Edit и Button (все компоненты с вкладки Standard); кнопку назовем “Искать”, текст компоненты Edit очистим, а компоненту ComboBox трогать пока не будем (она понадобится нам для поиска по отдельным столбцам таблицы). Добавляем новые листы на компоненте PageControl (Рис. 12)

10.

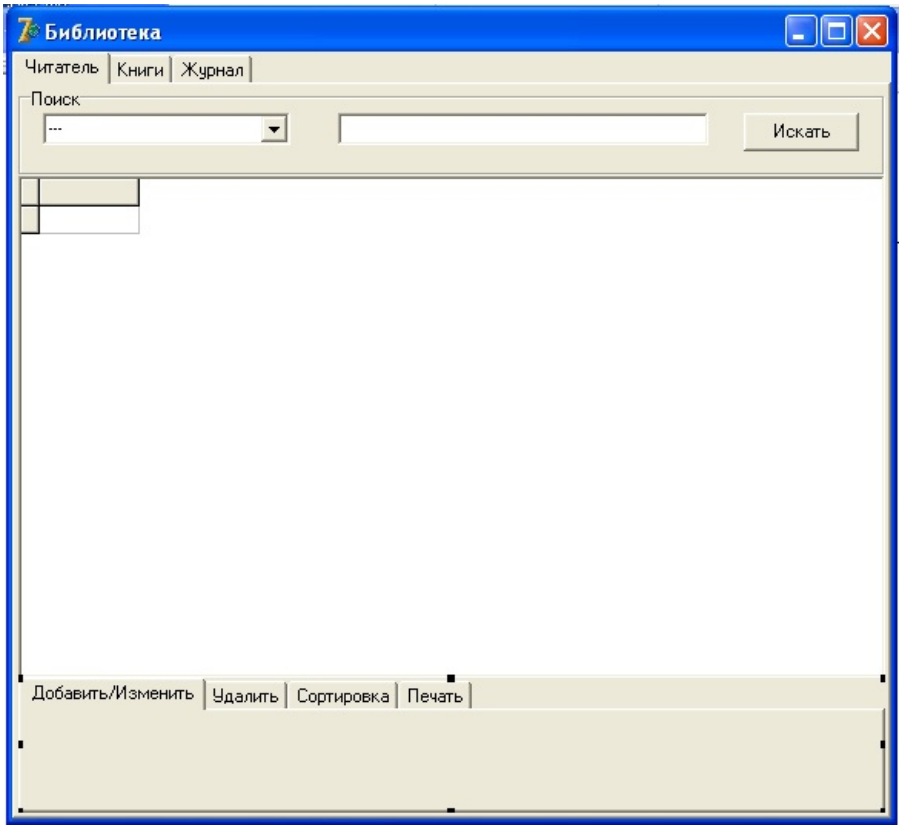


Рис. 12 - Добавление компонентов

Интерфейс для одной из таблиц базы данных создан. Данная последовательность шагов повторяется для всех таблиц, которые необходимо отобразить в клиентском приложении.

Из вкладки Ehlib помещаем на любую область формы компонент:

1. PrintDBGridEh – компонента для печати содержимого DBGridEh.
2. Компонент PrintDBGridEh в параметре DBGridEh выберем нужный нам DBGridEh.

Связываем нашу базу данных с DBGridEh: в параметре DataSource компоненты DBGridEh выбираем нужный нам DataSource – в конкретном примере “chitatelDataSource” (Рис. 14), но перед этим в Unit1.pas в uses подключаем (прописываем) Unit2 (Рис. 13)

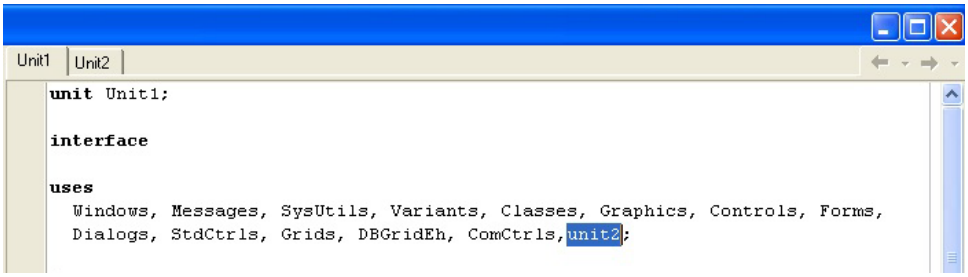


Рис. 13 - Подключение Unit2

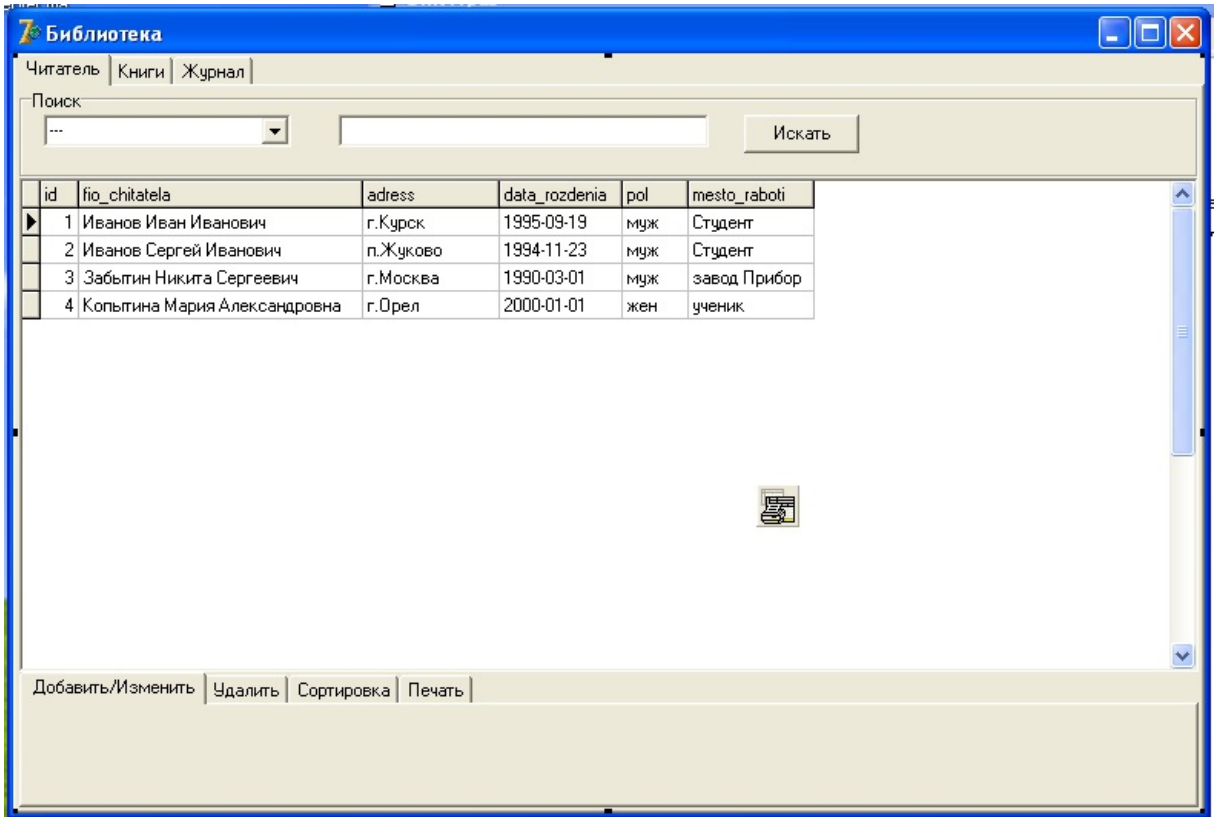


Рис. 14 - Связь базы данных с DBGridEh

Таким образом, мы видим отображение таблицы базы данных в нашем клиентском приложении (Рис. 14), но требуется немного отредактировать это отображение. Выполним двойной клик на компоненте DBGridEh, появится окно Editing DBGridEh1.Columns в нем выполним левый клик и выберем опцию “Add all fields” (если опция “Add all fields” неактивна, значит TADOQuery не выставлена

в состояние true).

11.

Рис. 15 - Настройка полей таблицы

В окне Editing DBGridEh1.Columns (Рис. 15) мы видим поля таблицы, ими мы можем управлять в инспекторе объектов. Нас интересует изменение полей Title.Caption (заголовок поля в DBGridEh), Visible (показывать/не показывать поле), Width (ширина поля). Изменив параметры этих полей приведем DBGridEh в оформленный вид (Рис. 16)

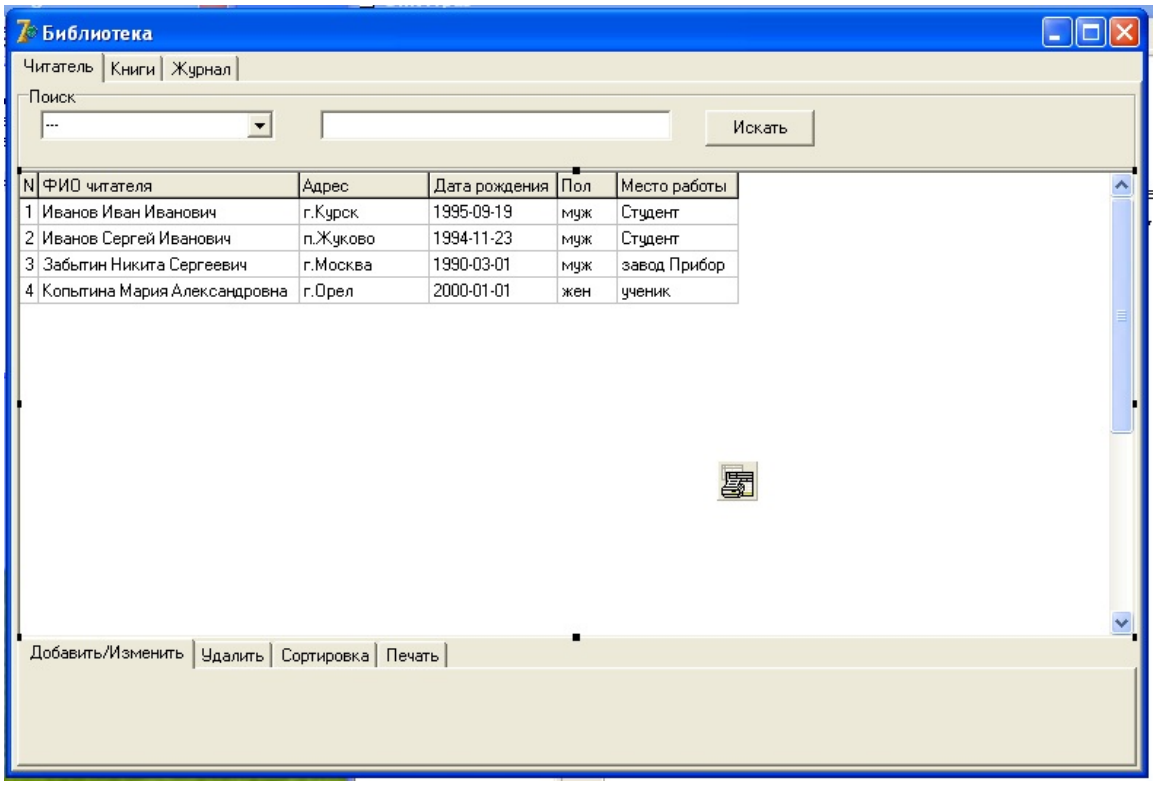


Рис. 16 - Полученный интерфейс

Итак, мы получили отображение в приложении одной таблицы из базы данных, проделываем тоже самое с другими таблицами.

Теперь сделаем поиск по таблице базе данных, которая отображается у нас в DBGridEh. Поиск у нас будет проходить как по каждому полю таблицы, так и по всем ее полям, для начала добавим в ComboBox1.Items нужные нам поля (Рис. 17)

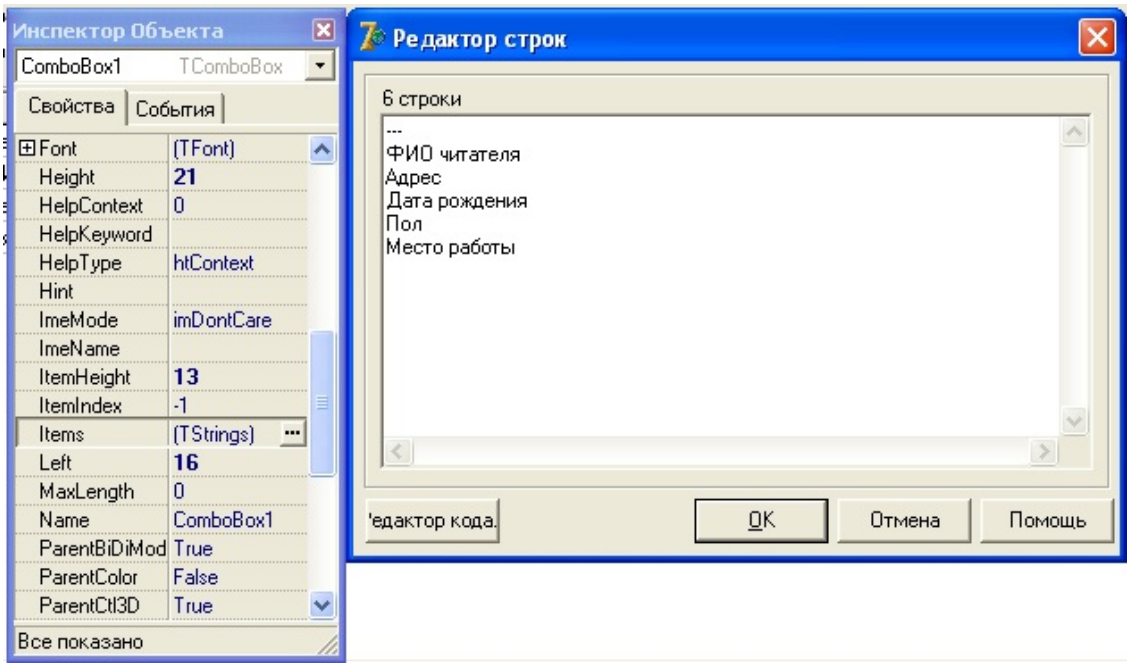


Рис. 17 - Добавление в ComboBox1.Items нужные нам поля

Сделаем двойной клик на кнопке “Искать” и вставим в процедуру обработки нажатия кнопки следующий код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if edit1.Text='' then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel';
DataModule2.chitatelADOQuery.Active:=true;
end else
begin
If ComboBox1.ItemIndex=1 then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where fio_chitatela like '"+'%'+edit1.text+'%'+"'";
DataModule2.chitatelADOQuery.Active:=true;
end else
If ComboBox1.ItemIndex=2 then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where adress like '"+'%'+edit1.text+'%'+"'";
```

```
DataModule2.chitatelADOQuery.Active:=true;
end else
If ComboBox1.ItemIndex=3 then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where data_rozdenia like '"+'%'+edit1.text+'%'+"'";
DataModule2.chitatelADOQuery.Active:=true;
end else
If ComboBox1.ItemIndex=4 then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where pol like '"+'%'+edit1.text+'%'+"'";
DataModule2.chitatelADOQuery.Active:=true;
end else
If ComboBox1.ItemIndex=5 then
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where mesto_raboti like '"+'%'+edit1.text+'%'+"'";
DataModule2.chitatelADOQuery.Active:=true;
end else
begin
DataModule2.chitatelADOQuery.Active:=false;
DataModule2.chitatelADOQuery.SQL.Text:='select * from chitatel where (fio_chetatela like '"+'%'+edit1.text+'%'+"') or (adress like '"+'%'+edit1.text+'%'+"') or (data_rozdenia like '"+'%'+edit1.text+'%'+"') or (pol like '"+'%'+edit1.text+'%'+"') or (mesto_raboti like '"+'%'+edit1.text+'%'+"')';
DataModule2.chitatelADOQuery.Active:=true;
end;
end;
end;
```

Поиск по всей таблице и по каждому полю готов и описывается вышеописанным кодом. Аналогичным образом данный шаг повторяется для выполнения поиска для других таблиц.

Перейдем к процедуре добавления в базу данных. Для этого добавим на форму на компоненту PageControl (лист Добавить/Изменить) 5

DBEditEh из вкладки Ehlib и 5 Label из вкладки Standart и одну кнопку Button (Рис. 18)

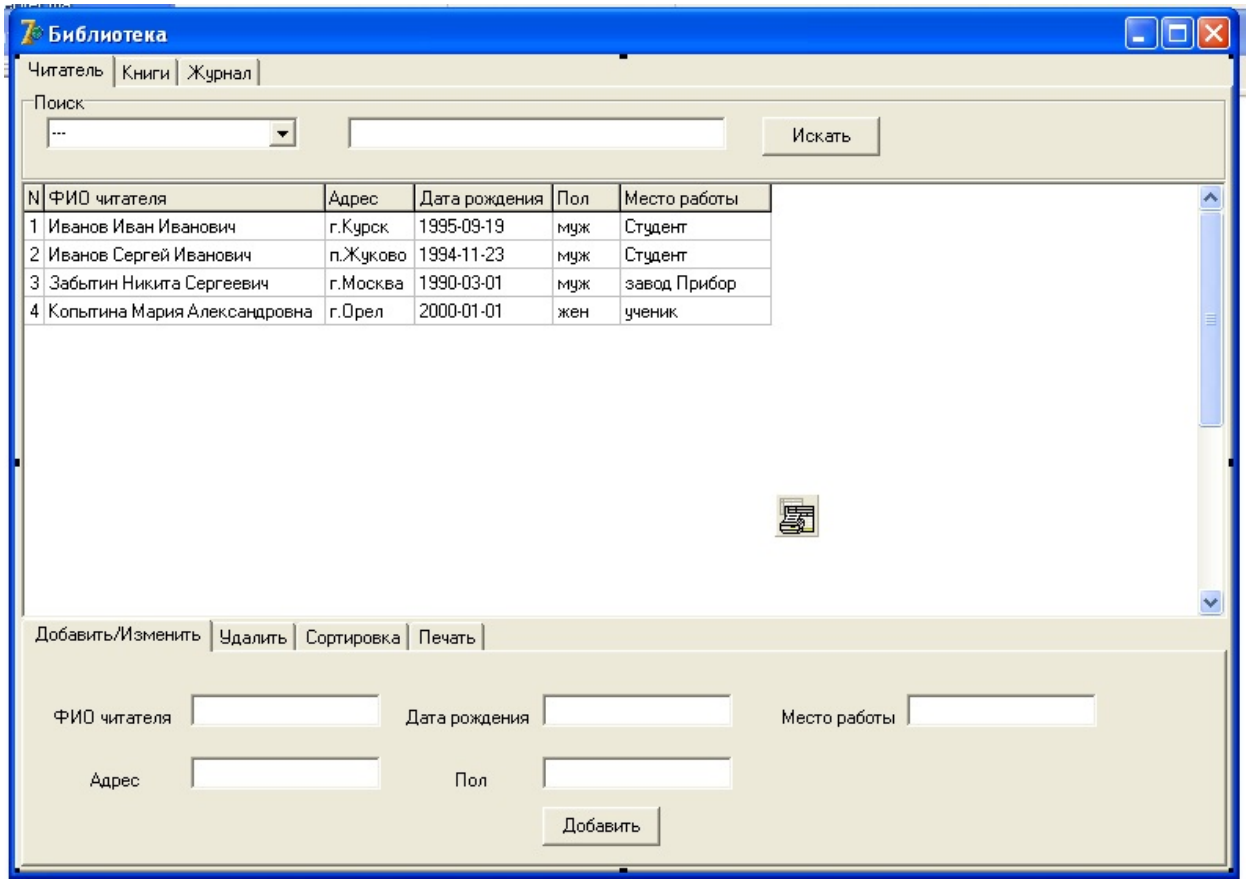


Рис. 18 - Добавление компонентов на лист “Добавить/Изменить”

Далее в свойствах DBEditEh1: DataSource выставляем DataModule2.chitatelDataSource, а в DataField - fio_chitatela.(Рис. 19)

Проделываем эту же операцию для оставшихся DBEditEh, с тем же значением поля DataSource, ставя соответствующими значениями поля DataField:

DBEditEh1=>DataField=>fio_chitatela(название столбца в таблице)

DBEditEh2=>DataField=>adress

DBEditEh3=>DataField=>data_rozdenia

DBEditEh4=>DataField=>pol

DBEditEh5=>DataField=>mesto_raboti

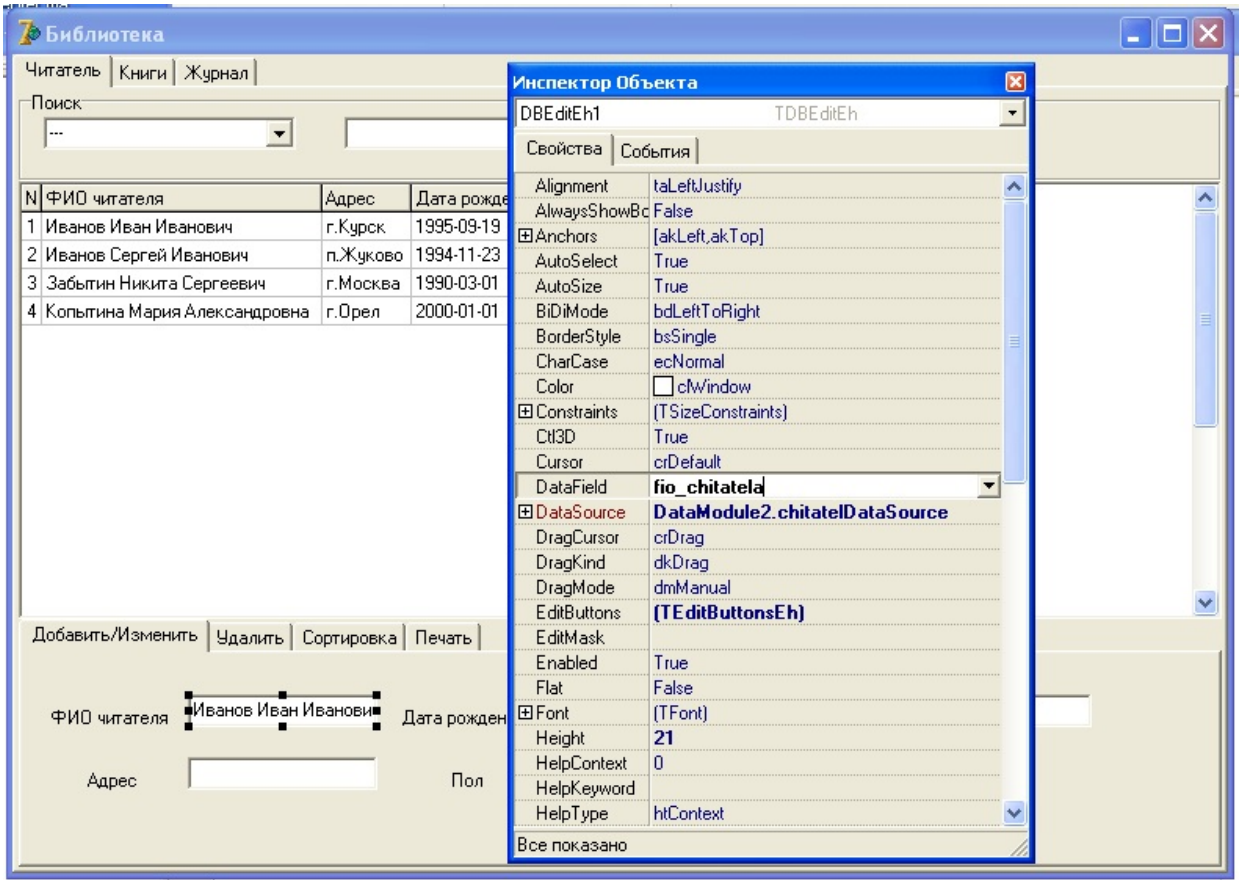


Рис. 19 - Свойства DBEditEh1

Далее заходим в SQL Server Management Studio и нажимаем создать запрос, в открывшемся окне создаём процедуру. Процедуру мы создаём для упрощения кода в Delphi:

```
create procedure new_chit (
@fio_chitatela        NVARCHAR(50),
@adress               NVARCHAR(50),
@data_rozdenia         date,
@pol                   NVARCHAR(50),
@mesto_raboti         NVARCHAR(50))
as
begin
insert into chitatel values ((select ISNULL(MAX(id)+1,1) from chitatel),
@fio_chitatela,@adress,
@data_rozdenia, @pol,
@mesto_raboti); End
```

При успешном выполнении данной процедуры нам выведет сообщение «Выполнение команд успешно завершено» (Рис. 20)

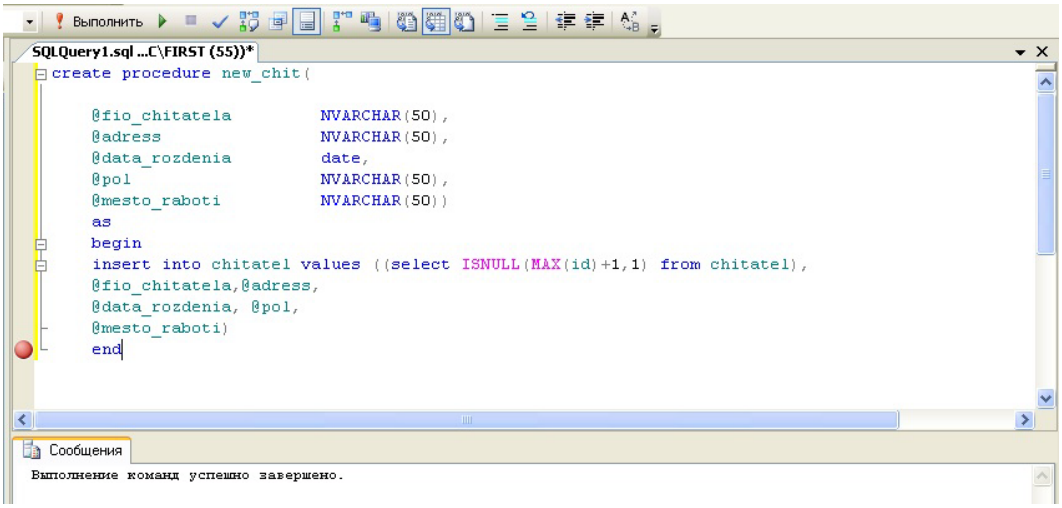


Рис. 20 - Успешное выполнение команд

Процедура создана. Закрываем sql server management studio. Переходим к Delphi. Открываем DataModule: Вид=> Формы=> DataModule2. Жмём ОК (Рис. 21)

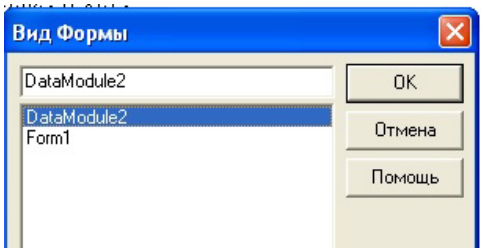


Рис. 21 - Открытие DataModule2

Добавляем на форму DataModule ещё один компонент TADOQuery сразу переименовываем его в HelpQuery (будет вспомогательным ADOQuery). В свойстве Connection выбираем ADOConnection1. Выбираем свойство SQL нажимаем на кнопку с «…». Пишем в этом окне любой запрос, т.к. это вспомогательной компонент (Например: select * from chitatel) и нажимаем ОК (Рис. 22). Ставим свойство ACTIVE в положение TRUE

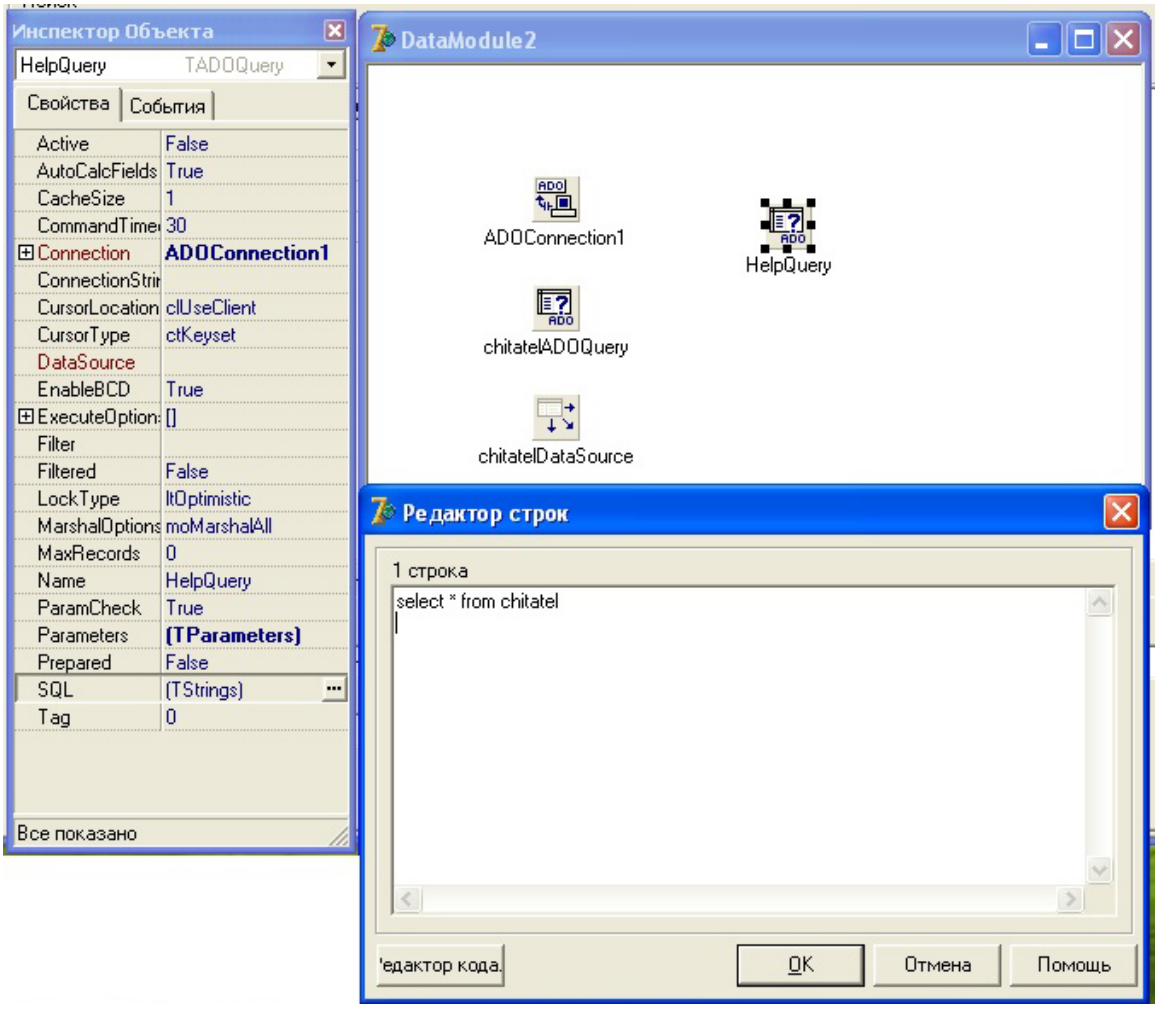


Рис. 22 - Свойства HelpQuery

Сделаем двойной клик на кнопке “Добавить” и вставим в процедуру обработки нажатия кнопки следующий код:

```
procedure TForm1.Button2Click(Sender: TObject);
var
fio_chitatela,adress,data_rozdenia,pol,mesto_raboti:string;
begin
 fio_chitatela:=quotedstr(unit1.Form1.DBEditEh1.Text);
 adress:=quotedstr(unit1.Form1.DBEditEh2.Text);
 data_rozdenia:=quotedstr(unit1.Form1.DBEditEh3.Text);
 pol:=quotedstr(unit1.Form1.DBEditEh4.Text);
 mesto_raboti:=quotedstr(unit1.Form1.DBEditEh5.Text);
 with unit2.DataModule2.HelpQuery do
 begin close;
 SQL.Clear;
 SQL.ADD
 ('exec new_chit'+fio_chitatela+','+adress+',
```

```
'+data_rozdenia+','+pol+','+mesto_raboti);
ExecSQL;
Unit2.DataModule2.chitatelADOQuery.Active:=False;
Unit2.DataModule2.chitatelADOQuery.Active:=True;
end; end;
```

Для изменения записей таблицы достаточно просто выбрать нужную строку таблицы, значения столбцов выбранной строки отобразятся в DBEditEh записываем другие значения, после нажатия на любую часть формы значения поменяются, также значения можно менять простым нажатием на нужное поле в DBGridEh.

Приступаем к созданию процедуры удаления. Нажимаем на лист Удалить компоненты PageControl. Выставляем 2 элемента: кнопку Button из вкладки Standart и DBComboBoxEh из вкладки ADO.

В свойствах DBComboBoxEh DataSource выбираем DataModule2.chitatelDataSource DataField – fio_chitatela. (Рис. 24)

Для удаления нужно создать дополнительную форму для подтверждения удаления. Форма может выглядеть примерно вот так (Рис. 23)

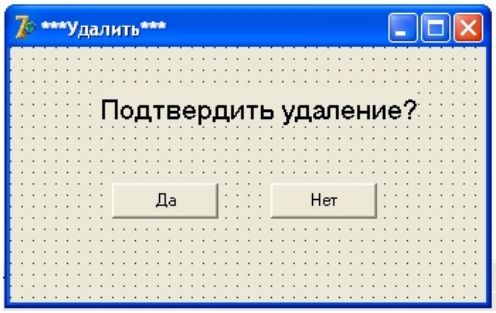


Рис. 23 - Форма подтверждения удаления

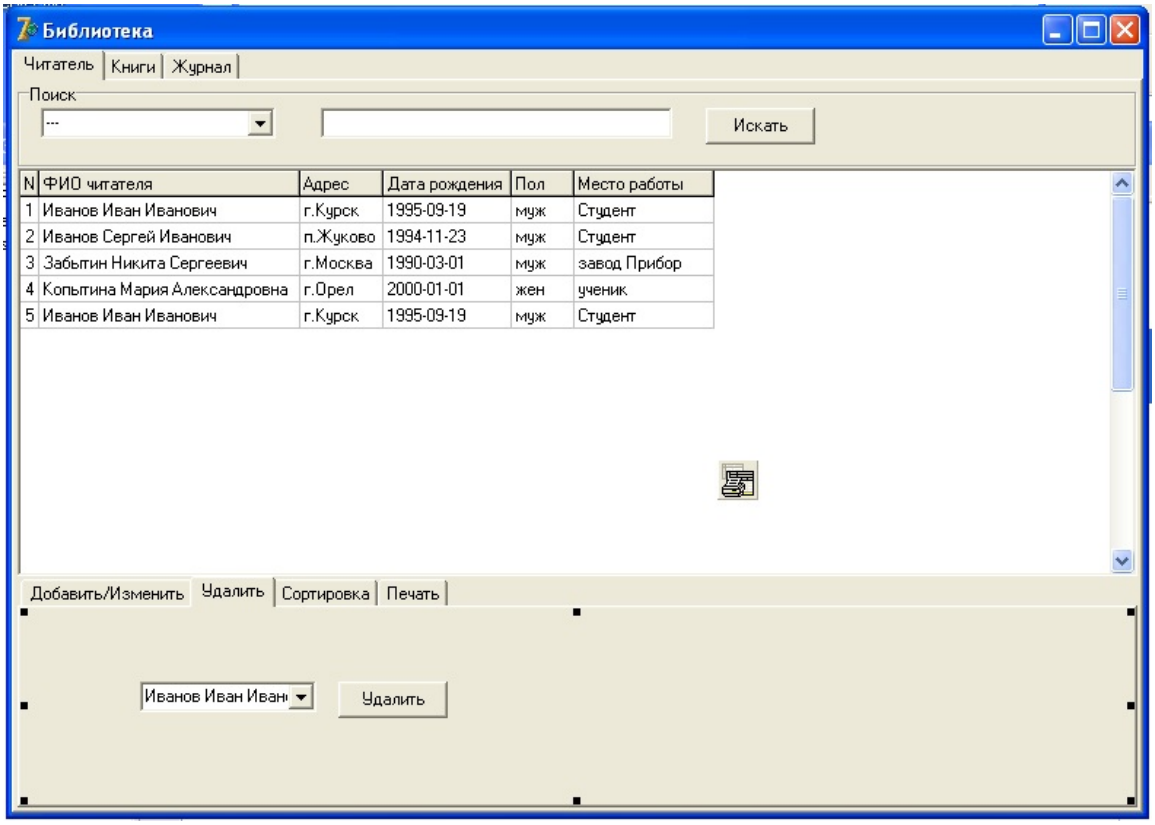


Рис. 24 - Добавление компонентов на лист “Удалить”

Кнопка “Нет” закрывает форму и имеет следующий код:

```
procedure TForm3.Button2Click(Sender: TObject);
begin
close;
end;
```

Кнопка “Да” имеет следующий код:

```
procedure TForm3.Button1Click(Sender: TObject);
begin
DataModule2.chitatelADOQuery.Delete;
close;
end;
```

Сделаем двойной клик на кнопке “Удалить” и вставим в процедуру обработки нажатия кнопки следующий код:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
Form3.Showmodal;
end;
```

Перейдем к процедуре сортировки. В качестве примера сделаем сортировку по ФИО читателя, на основе примера можно будет сделать

все сортировки какие необходимы. Итак, на Форме компонента PageControl (лист Сортировка) помещаем компонент из вкладки Ehlib DBCheckBoxEh и в его свойствах DataSource и DataField выбираем соответствующие значения DataModule2.chitatelDataSource и fio_chitatela. Также меняем Caption (Рис. 25)

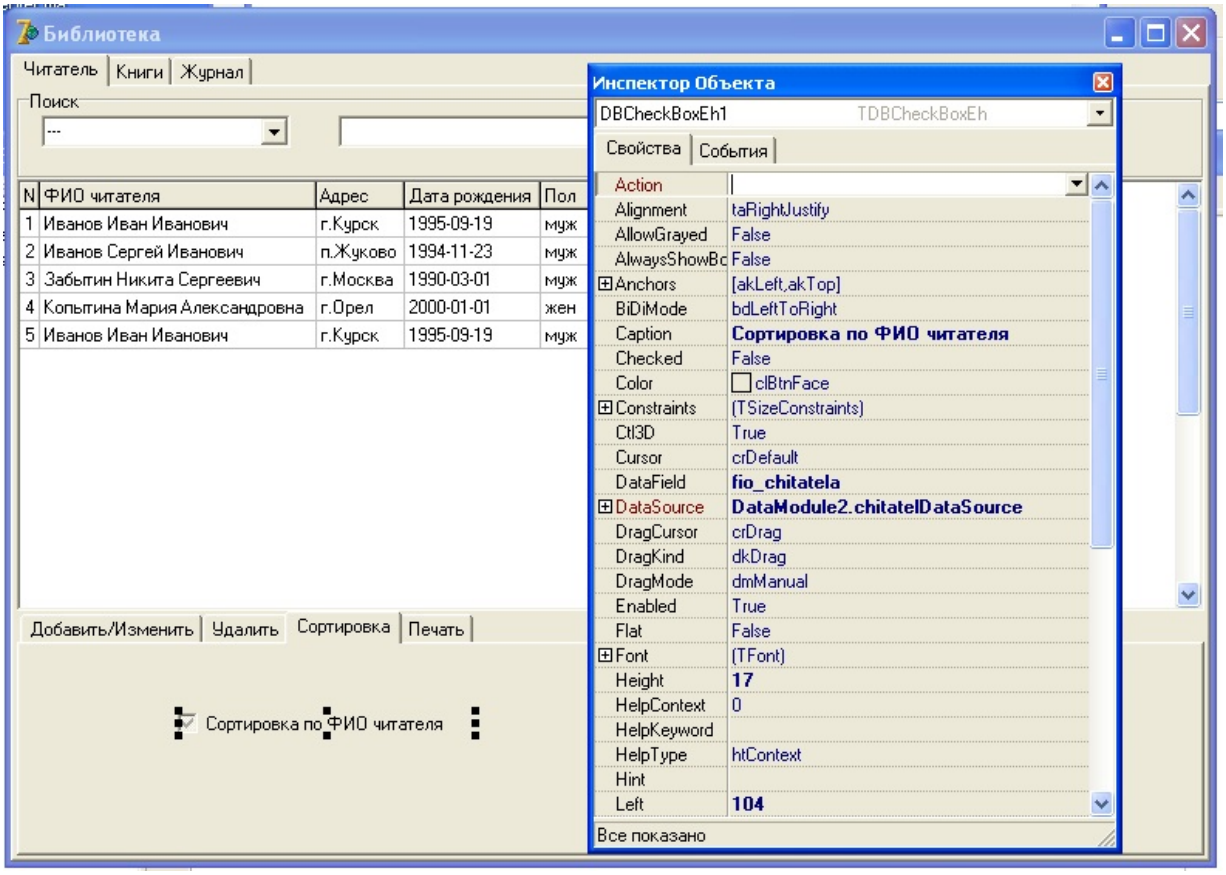


Рис. 25 - DBCheckBoxEh1 и его свойства

Сделаем двойной клик на DBCheckBoxEh “Сортировать по ФИО читателя” и вставим в процедуру обработки нажатия кнопки следующий код:

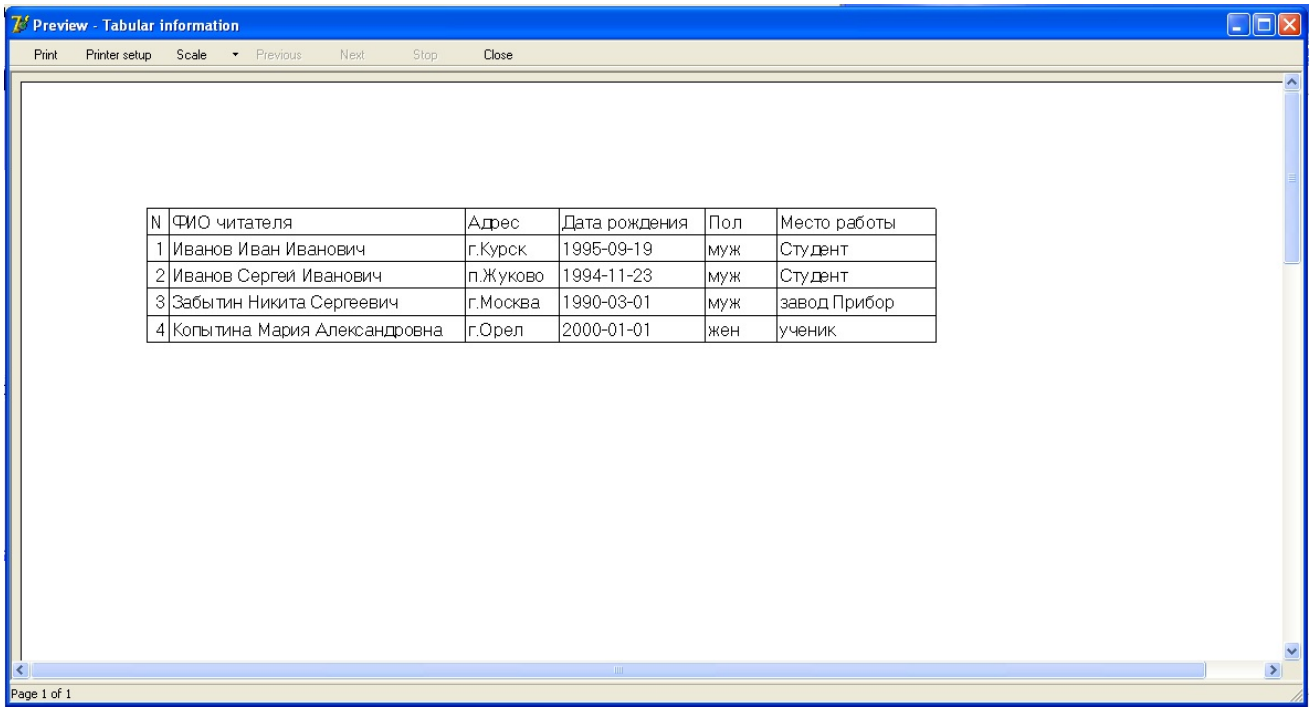
```
procedure TForm1.DBCheckBoxEh1Click(Sender: TObject);
begin
if unit1.Form1.DBCheckBoxEh1.Checked=true then
begin
with unit2.DataModule2.chitatelADOQuery do
begin
close;
SQL.Clear;
SQL.ADD('select * from chitatel order by fio_chitatela');
open;
```

```
end;
end;
end;
```

При помощи компонента PrintDBGridEh опишем процесс вывода результата запроса из базы данных на принтер. В параметре DBGridEh у нас уже стоит нужный нам DBGridEh. Просто создадим процедуру для кнопки “РАСПЕЧАТАТЬ”, вставив туда нижеследующий код:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
PrintDBGridEh1.SetSubstitutes(['%(Today)',DateToStr(Now)]);
PrintDBGridEh1.Preview;
end;
```

Результатом работы кода является появление окна с preview-видом таблицы, где можно произвести необходимые настройки принтера (Рис. 26)



N	ФИО читателя	Адрес	Дата рождения	Пол	Место работы
1	Иванов Иван Иванович	г.Курск	1995-09-19	муж	Студент
2	Иванов Сергей Иванович	п.Жуково	1994-11-23	муж	Студент
3	Забытин Никита Сергеевич	г.Москва	1990-03-01	муж	завод Прибор
4	Копытина Мария Александровна	г.Орел	2000-01-01	жен	ученик

Рис. 26 - Результат работы кода

Данный шаг нужно повторить необходимое число раз для необходимого числа компонент DBGridEh и создать необходимое число компонент PrintDBGridEh.

Следующая процедура - создание произвольных запросов. Для этого открываем DataModule2 и помещаем 2 вспомогательных компонента ADOQuery1(в свойстве Connection выбираем ADOConnection1 в свойстве

SQL пишем любой запрос, ставим значение ACTIVE- True) и DataSource1(выставляем в свойстве DataSet: ADOQuery1). (Рис. 27)

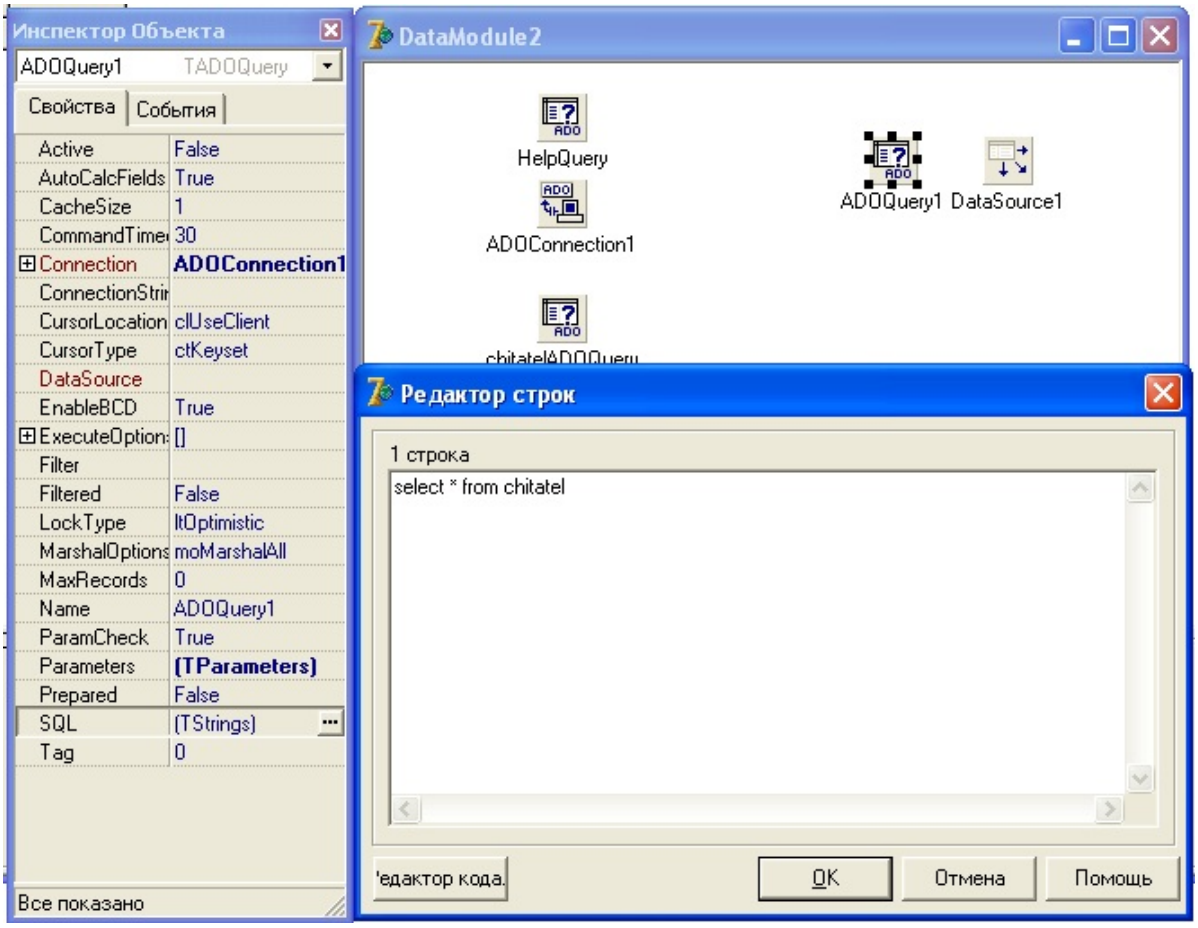


Рис. 27 - Добавление и настройка компонентов ADOQuery1 и DataSource1

Выделим главный PageConrol из пункта и создадим новый лист (Произвольная таблица). Ставим на форму GroupBox из вкладки Standart и меняем Caption “Произвольный запрос”. Помещаем на компонент GroupBox, компонент Memo из вкладки Standart и в свойстве Memo ScrollBars выставляем значение: ssVertical. Ставим 2 BUTTON из вкладки Standart и меняем Caption.

Кнопки будут “OK” и “РАСПЕЧАТАТЬ” выставляем на форму DBGridEh из вкладки Ehlib (в свойствах ставим DataSOurce: DataModule2.DataSource1, Align: alClient).

Из вкладки Ehlib помещаем на любую область формы компонент: PrintDBGridEh в параметре DBGridEh выберем нужный нам DBGridEh. (Рис. 28)

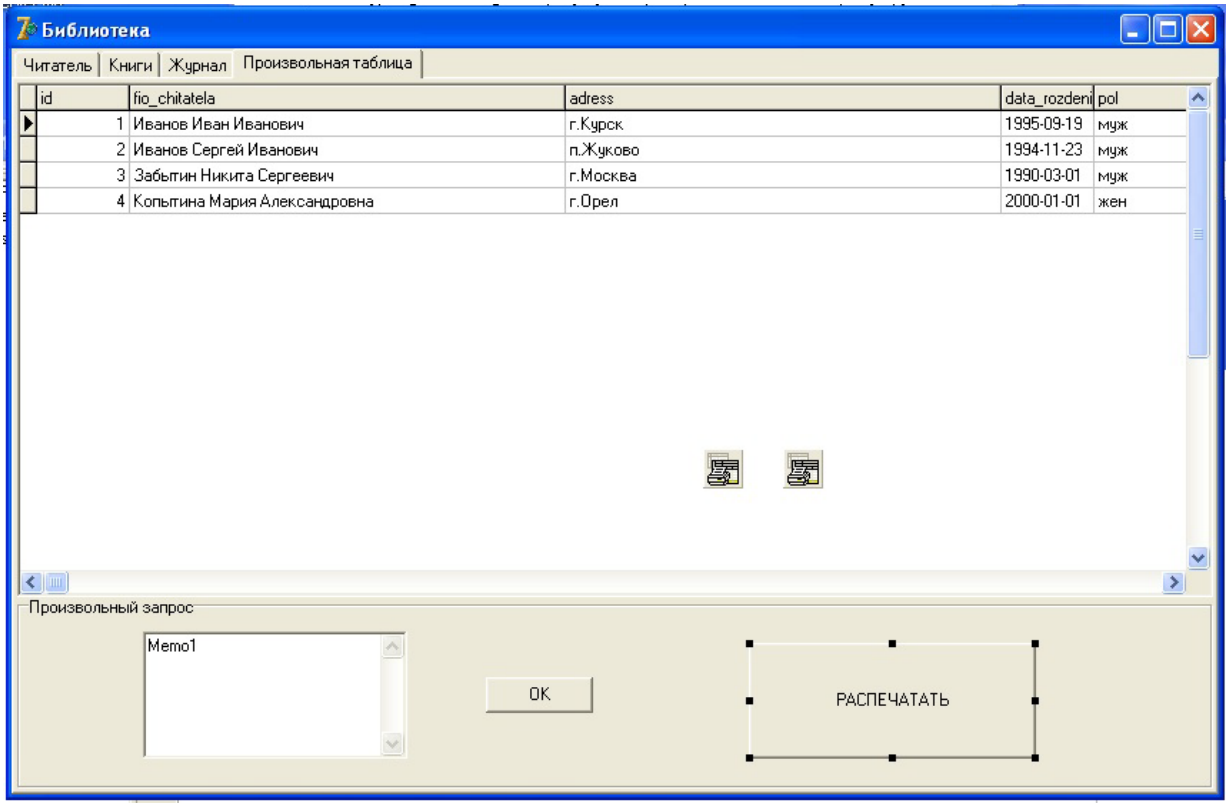


Рис. 28 - Добавление компонентов на лист “Произвольная таблица”

Сделаем двойной клик на кнопку “РАСПЕЧАТАТЬ” и вставим в процедуру обработки нажатия кнопки следующий код:

```
procedure TForm1.Button6Click(Sender: TObject);
begin
PrintDBGridEh2.SetSubstitutes(['%(Today)',DateToStr(Now)]);
PrintDBGridEh2.Preview;
end;
```

Сделаем двойной клик на кнопку “OK” и вставим в процедуру обработки нажатия кнопки следующий код:

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  DataModule2.ADOQuery1.close;
  DataModule2.ADOQuery1.SQL.Clear;
  DataModule2.ADOQuery1.Active:=false;
  DataModule2.ADOQuery1.SQL.ADD("+Memo1.text+");
  DataModule2.ADOQuery1.Active:=true; end;
```

Данная процедура позволит нам вводить любой запрос в Memo результат которого будет отображаться в PageControl (лист Произвольная таблица).

Таким образом мы соединили базу данных sql server management studio с Delphi 7. Связали базу данных с делфи при помощи компонентов ADO.

Для того чтобы в автоматическом режиме открывалось окно Свойство связи с данными (где мы выбираем провайдера, а именно Microsoft OLE DB Provider for SQL Server и указываем имя сервера; имя пользователя и пароль (который мы создавали в sql server management studio для нашей базы данных), выбираем нашу базу данных при запуске базы данных с помощью Delphi. (Рис. 29)

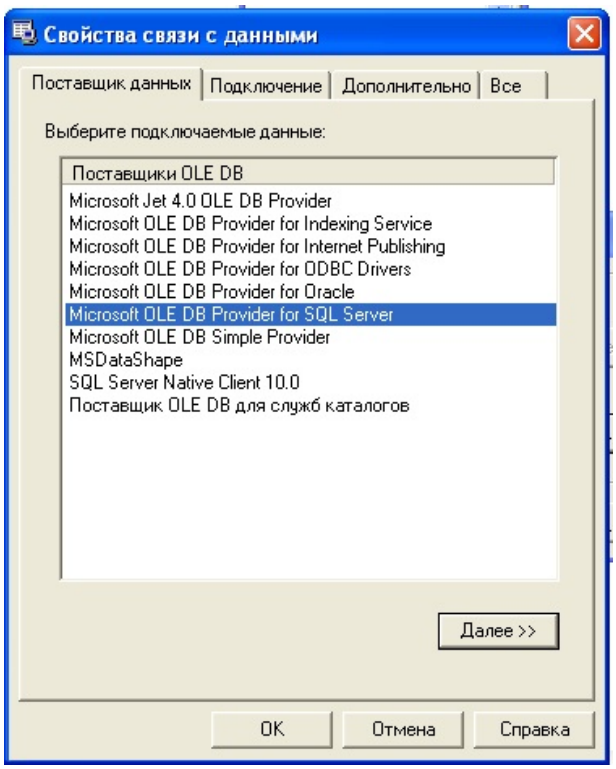


Рис. 29 - Настройка соединения с базой данных

Открываем DataModule2 выбираем компонент ADOConnection1=> События=> OnWillConnect жмём дважды по пустому полю и записываем в процедуру обработки нажатия следующий код:

```
procedure TDataModule2.ADOConnection1WillConnect(
Connection: TADOConnection;
var ConnectionString, UserID,
Password: WideString;
var ConnectOptions: TConnectOption;
var EventStatus: TEventStatus);
begin
```

```
ADODB.PromptDataSource(0,");
end;
```

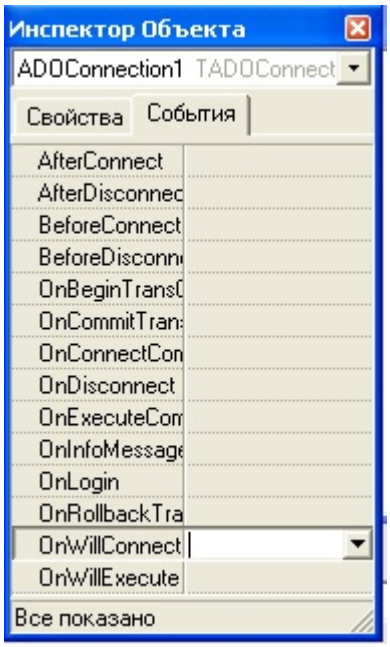


Рис.30 – События компонента ADOConnection1

5. ШИФРОВАНИЕ SQL SERVER

ЦЕЛЬ РАБОТЫ

Изучить теоретическую часть. Рассмотреть основные методы шифрования. Зашифровать базу данных в среде Microsoft SQL Server 2008 R2 любым из представленных способом.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Шифрование - процесс кодирования конфиденциальных данных с использованием ключа или пароля. Шифрование надежно защищает данные и сокращает вероятность несанкционированного раскрытия конфиденциальной информации, так как без соответствующего ключа или пароля данные бесполезны. SQL Server располагает многими режимами шифрования, в том числе на уровне ячеек, базы данных, файлов через Windows и шифрования на транспортном уровне.

Шифрование SQL Server не решает проблему доступности инфраструктуры и баз данных SQL Server, но повышает защищенность данных на уровнях базы данных и операционной системы, даже если нарушена конфиденциальность инфраструктуры или баз данных SQL Server.

Несмотря на то, что шифрование является полезным средством обеспечения безопасности, его не следует применять ко всем данным или соединениям. При решении о внедрении шифрования необходимо проанализировать, как пользователи получают доступ к данным.

Если пользователи получают доступ к данным через открытую сеть, то шифрование может потребоваться для повышения безопасности. Однако если весь доступ осуществляется по безопасной внутренней сети, то шифрование не требуется. Использование шифрования включает политику управления паролями, ключами и сертификатами.

Модель шифрования SQL Server в основном предоставляет функции управления ключами шифрования, соответствующие стандарту ANSI X9.17. В этом стандарте определены несколько уровней ключей шифрования, использующихся для шифрования других ключей, которые в свою очередь применяются для шифрования собственно данных.

В таблице №1 перечислены уровни ключей шифрования SQL Server и ANSI X9.17.

Таблица 1 - уровни ключей шифрования SQL Server и ANSI X9.17

Таблица	Уровень шифрования ключей SQL Server и ANSI X9.17	
Уровень SQL Server	Уровень ANSI X9.17	Описание
SMK	Главный ключ	SMK – ключ верхнего уровня, используемый для шифрования DMK. SMK шифруется с применением DPAPI.
DMK	Ключ шифрования ключей	DMK – симметричный ключ, используемый для шифрования симметричного ключа, ассиметричного ключа и сертификата. Для каждой базы данных может быть определен только один DMK.
Симметричные ключи, ассиметричные ключи и сертификаты	Ключ данных	Симметричные ключи, ассиметричные ключи и сертификаты используются для шифрования данных.

Главный ключ службы Service master key(SMK) - ключ верхнего уровня и предок всех ключей в SQL Server. SMK - асимметричный ключ, шифруемый с использованием Windows Data Protection API (DPAPI). SMK автоматически создается, когда шифруется какой-нибудь объект, и привязан к учетной записи службы SQL Server. SMK используется для шифрования главного ключа базы данных Database master key (DMK).

Второй уровень иерархии ключей шифрования - DMK. С его помощью шифруются симметричные ключи, асимметричные ключи и сертификаты. Каждая база данных располагает лишь одним DMK.

Следующий уровень содержит симметричные ключи, асимметричные ключи и сертификаты. Симметричные ключи - основное средство шифрования в базе данных. Microsoft рекомендует шифровать данные только с помощью симметричных ключей. Кроме того, в SQL

Server 2008 и более новых версиях есть сертификаты уровня сервера и ключи шифрования базы данных для прозрачного шифрования данных.

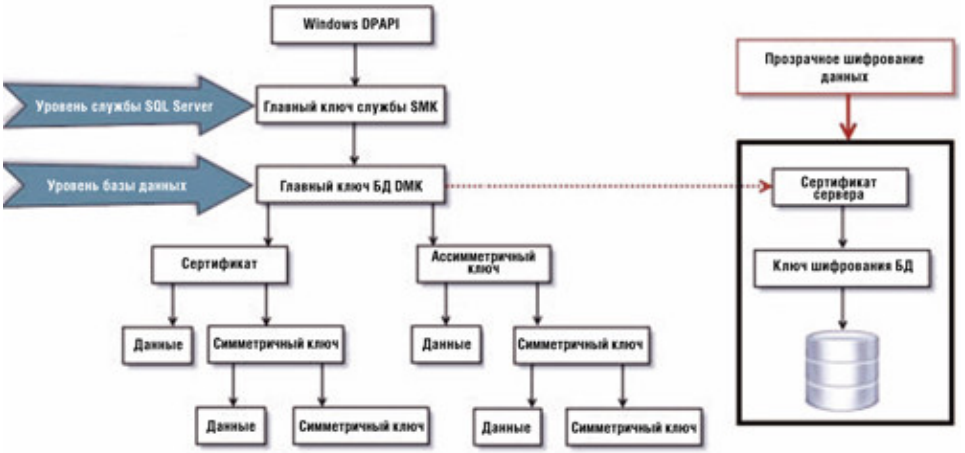


Рис. 1 - Иерархия ключей шифрования для SQL Server 2008

Следует учитывать следующие основные понятия.

- Для лучшей производительности данные следует шифровать с помощью симметричных ключей, а не с помощью сертификатов и асимметричных ключей.
- Главные ключи базы данных защищены главным ключом службы. Главный ключ службы создается при установке SQL Server и шифруется API-интерфейсом защиты данных Windows (DPAPI).
- Возможны другие иерархии шифрования с дополнительными уровнями.
- Модуль расширенного управления ключами хранит симметричные или асимметричные ключи вне SQL Server.
- Прозрачное шифрование данных (TDE) должно использовать симметричный ключ, который называется ключом шифрования базы данных, защищенный сертификатом, который, в свою очередь защищается главным ключом базы данных master или асимметричным ключом, хранящимся в модуле расширенного управления ключами.
- Главный ключ службы и все главные ключи базы данных являются симметричными ключами.

SQL Server поддерживает следующие механизмы шифрования:

- функции Transact-SQL;
- асимметричные ключи;
- симметричные ключи;
- сертификаты

- прозрачное шифрование данных

Функции Transact-SQL - отдельные элементы можно шифровать по мере того, как они вставляются или обновляются, с помощью функций Transact-SQL.

Сертификат открытого ключа, или просто сертификат, представляет собой подписанную цифровой подписью инструкцию, которая связывает значение открытого ключа с идентификатором пользователя, устройства или службы, имеющей соответствующий закрытый ключ. Сертификаты поставляются и подписываются центром сертификации (certification authority, CA). Сущность, получающая сертификат от центра сертификации, является субъектом этого сертификата. Как правило, сертификаты содержат следующие сведения.

- Открытый ключ субъекта.
- Идентификационные данные субъекта, например имя и адрес электронной почты.
- Срок действия, то есть интервал времени, на протяжении которого сертификат будет считаться действительным.

Сертификат действителен только в течение указанного в нем периода, который задается в каждом сертификате при помощи дат (Valid From и Valid To), определяющих начало и окончание срока действия. По истечении срока действия сертификата его субъект должен запросить новый сертификат.

- Идентификационные данные поставщика сертификата.
- Цифровая подпись поставщика.

Эта подпись подтверждает действительность связи между открытым ключом и идентификационными данными субъекта. (В процессе создания цифровой подписи данные, вместе с некоторыми секретными данными отправителя, преобразуются в тег, называемый подписью.)

Главное преимущество сертификатов в том, что они позволяют не хранить на узлах совокупность паролей отдельных субъектов. Вместо этого узел просто устанавливает доверительные отношения с поставщиком сертификата; после этого поставщик может подписать неограниченное количество сертификатов.

Когда узел (например, защищенный веб-сервер) указывает, что конкретный поставщик сертификата является доверенным корневым центром сертификации, он неявно выражает доверие к политикам, которые поставщик использовал при определении связей для изданных им сертификатов. Таким образом, узел выражает уверенность в том, что

поставщик проверил идентификационные данные субъекта сертификата. Узел делает поставщика доверенным корневым центром сертификации, помещая самостоятельно подписанный сертификат поставщика, содержащий открытый ключ поставщика, в хранилище сертификатов доверенного корневого центра сертификации на главном компьютере. Промежуточные или подчиненные центры сертификации являются доверенными только в том случае, если к ним ведет правильный путь от доверенного корневого центра сертификации.

Поставщик может отозвать сертификат до истечения срока его действия. При этом отменяется связь открытого ключа с идентификационными данными, указанными в сертификате. Каждый поставщик ведет список отозванных сертификатов, который можно использовать для проверки конкретного сертификата.

Самостоятельно подписанные сертификаты, созданные SQL Server, соответствуют стандарту X.509 и поддерживают поля X.509 v1.

Асимметричный ключ состоит из закрытого ключа и соответствующего открытого ключа. Каждый из этих ключей позволяет дешифровать данные, зашифрованные другим ключом. На выполнение асимметричных операций шифрования и дешифрования требуется сравнительно много ресурсов, но они обеспечивают более надежную защиту, чем симметричное шифрование. Асимметричный ключ можно использовать для шифрования симметричного ключа перед его сохранением в базе данных.

Симметричный ключ — это ключ, используемый и для шифрования, и для дешифрования данных. Данные при использовании симметричного ключа шифруются и дешифруются быстро, и он вполне подходит для повседневной защиты конфиденциальных данных, хранящихся в базе данных.

Прозрачное шифрование данных (TDE) является особым случаем шифрования с использованием симметричного ключа. TDE шифрует всю базу данных, используя симметричный ключ, который называется ключом шифрования базы данных. Функция прозрачного шифрования данных (TDE) выполняет в реальном времени шифрование и дешифрование файлов данных и журналов в операциях ввода-вывода. При шифровании используется ключ шифрования базы данных (DEK), который хранится в загрузочной записи базы данных для доступности при восстановлении.

Ключ шифрования базы данных является симметричным ключом, защищенным сертификатом, который хранится в базе данных master на сервере, или асимметричным ключом, защищенным модулем расширенного управления ключами. Функция прозрачного шифрования данных защищает «неактивные» данные, то есть файлы данных и журналов.

Основное правило прозрачного шифрования, это то, что шифрование файла базы данных проводится на уровне страниц. Страницы в зашифрованной базе данных шифруются до записи на диск и дешифруются при чтении в память. Прозрачное шифрование данных не увеличивает размер зашифрованной базы данных. При этом не стоит забывать, что прозрачное шифрование не шифрует данные при передаче через каналы связи, для этого к примеру, можно включить шифрование соединения. Так же надо учитывать, что шифрование\дешифрование данных это нагрузка на CPU, следовательно, при проектировании необходимо учесть запас мощностей процессоров.

При включении функции прозрачного шифрования данных необходимо немедленно создать резервную копию сертификата и закрытого ключа, связанного с этим сертификатом. В случае если сертификат окажется недоступен или понадобится восстановить базу данных на другом сервере либо присоединить ее к другому серверу, необходимо иметь копии сертификата и закрытого ключа. В противном случае будет невозможно открыть базу данных. Сертификат шифрования следует сохранить, даже если функция прозрачного шифрования в базе данных будет отключена. Даже если база данных не зашифрована, части журнала транзакций могут по-прежнему оставаться защищенными, а для некоторых операций будет требоваться сертификат до выполнения полного резервного копирования базы данных. Сертификат, который превысил свой срок хранения, все еще может быть использован для шифрования и расшифровки данных с помощью прозрачного шифрования данных.

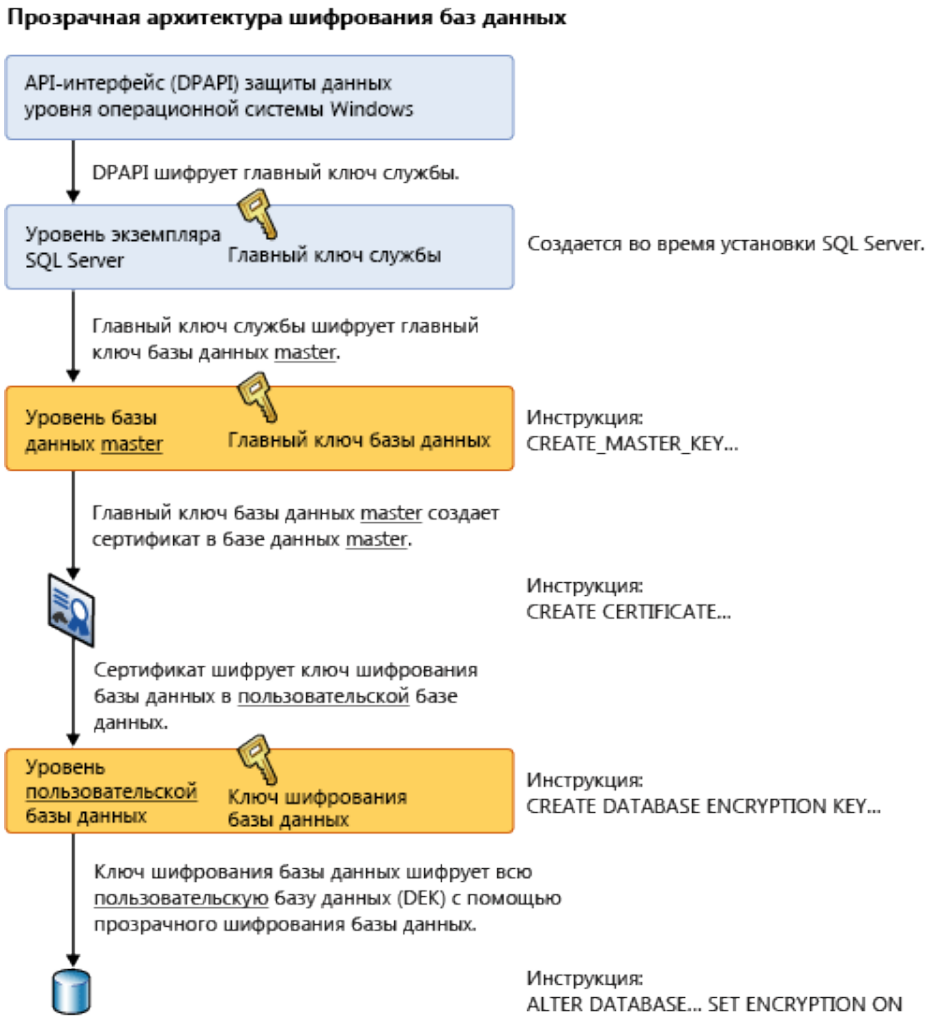


Рис.2 - Архитектура прозрачного шифрования данных

12.

13. Выбор алгоритма шифрования

14. **Алгоритмы шифрования определяют преобразования данных, исключающие возможность легкого восстановления исходного текста неавторизированными пользователями. SQL Server позволяет администраторам и разработчикам выбирать из нескольких алгоритмов, в том числе DES, Triple DES, TRIPLE_DES_3KEY, RC2, RC4, RC4 с 128-разрядным ключом, DESX, AES с 128-разрядным ключом, AES с 192-разрядным ключом и AES с 256-разрядным ключом.**

15. **Не существует одного алгоритма, идеально подходящего для всех случаев. Информация по качеству каждого из них лежит за пределами электронной документации по SQL Server. Однако можно руководствоваться следующими общими принципами:**

- Надежные алгоритмы шифрования обычно потребляют больше ресурсов ЦП, чем менее надежные средства шифрования.

- Использование длинных ключей, как правило, дает более надежные результаты, чем шифрование с помощью коротких ключей.
- Асимметричное шифрование слабее симметричного шифрования с использованием ключа той же длины, но является относительно медленным.
- Блочные шифры с длинными ключами надежнее поточных шифров.
- Длинные сложные пароли надежнее, чем короткие пароли.
- При необходимости шифровать большие объемы данных, шифруйте данные с помощью симметричного ключа, а симметричный ключ — с помощью асимметричного ключа.
- Зашифрованные данные не поддаются сжатию, но сжатые данные могут быть зашифрованы. При использовании сжатия данных, выполняйте эту операцию до их шифрования.

Алгоритм RC4 поддерживается только в целях обратной совместимости. Когда база данных имеет уровень совместимости 90 или 100, новые материалы могут шифроваться только с помощью алгоритмов RC4 или RC4_128. (Не рекомендуется.) Вместо этого используйте более новые алгоритмы, например AES. В SQL Server 2012 и более поздних версиях материалы, зашифрованные с помощью алгоритмов RC4 или RC4_128, могут быть расшифрованы на любом уровне совместимости.

Многократное использование одного и того же KEY_GUID алгоритма RC4 или RC4_128 для разных блоков данных приведет к формированию одинакового ключа RC4, так как SQL Server не предоставляет помеху автоматически. Повторное использование ключа RC4 является типичной ошибкой, снижающей надежность шифра. Таким образом, ключевые слова RC4 и RC4_128 являются устаревшими.

ВЫПОЛНЕНИЕ РАБОТЫ

Рассмотрим шифрование с помощью ассиметричного ключа. Основные функции для управления ассиметричными ключами:

- CREATE ASYMMETRIC KEY.

Создает асимметричный ключ в базе данных. Асимметричный ключ является защищаемой сущностью на уровне базы данных.

В его форме по умолчанию эта сущность содержит как открытый, так и закрытый ключ. CREATE ASYMMETRIC KEY при выполнении без предложения FROM формирует новую пару ключей. CREATE

ASYMMETRIC KEY при выполнении с предложением FROM импортирует пару ключей из файла или открытый ключ из сборки.

По умолчанию закрытый ключ защищается с помощью главного ключа базы данных. Для защиты закрытого ключа необходим пароль, если не был создан главный ключ базы данных. Если главный ключ базы данных существует, пароль необязателен. Закрытый ключ может быть длинной 512, 1024 или 2048 бит.

- ALTER ASYMMETRIC KEY.

Изменяет свойства асимметричного ключа:

- Изменение пароля закрытого ключа. В следующем примере изменяется пароль, используемый для защиты закрытого ключа асимметричного ключа PacificSales09. Новым паролем будет <enterStrongPasswordHere>.

```
ALTER ASYMMETRIC KEY PacificSales09
WITH PRIVATE KEY (
DECRYPTION BY PASSWORD = '<oldPassword>',
ENCRYPTION BY PASSWORD = '<enterStrongPasswordHere>');
GO
```

- Удаление закрытого ключа из асимметричного ключа. **В следующем примере закрытый ключ удаляется из ассиметричного ключа** PacificSales19**, в котором остается только открытый ключ.**

```
ALTER ASYMMETRIC KEY PacificSales19
REMOVE PRIVATE KEY;
GO
```

- Снятие парольной защиты с закрытого ключа. В следующем примере снимается парольная защита закрытого ключа и устанавливается его защита главным ключом базы данных.

```
OPEN MASTER KEY;
ALTER ASYMMETRIC KEY PacificSales09
WITH PRIVATE KEY (
DECRYPTION BY PASSWORD = '<enterStrongPasswordHere>' );
GO
```

- DROP ASYMMETRIC KEY.

Удаляет асимметричный ключ из текущей базы данных. Нельзя удалять асимметричный ключ, которым в базе данных зашифрован симметричный ключ или с которым сопоставлено имя входа.

Прежде чем удалять такой ключ, следует сначала удалить пользователя или имя входа, с которым этот ключ сопоставлен, либо удалить или перешифровать симметричный ключ, который зашифрован данным асимметричным ключом. Для удаления шифрования, выполненного асимметричным ключом, предназначен параметр DROP ENCRYPTION инструкции ALTER SYMMETRIC KEY.

Метаданные асимметричных ключей доступны через представление каталога sys.asymmetric_keys. Сами ключи для просмотра из базы данных недоступны.

Если асимметричный ключ сопоставлен ключу поставщика расширенного управления ключами на устройстве поставщика, а параметр REMOVE PROVIDER KEY не указан, то ключ будет удален из базы данных, но не с устройства. Будет выдано предупреждение.

- ENCRYPTBYASYMKEY.

Шифрует данные при помощи асимметричного ключа. Шифрование/дешифрование с помощью асимметричного ключа является очень дорогостоящим по сравнению с шифрованием/дешифрованием с помощью симметричного ключа. Рекомендуется не шифровать асимметричным ключом большие наборы данных, например таблицы с пользовательскими данными. Вместо этого данные следует шифровать с помощью сильного симметричного ключа, а симметричный ключ шифровать с помощью асимметричного.

EncryptByAsymKey возвращает значение NULL, если ввод превышает определенное число байтов, в зависимости от алгоритма. Ограничения: ключ RSA на 512 бит может шифровать до 53 байт, ключ на 1024 бита может шифровать до 117 байт, ключ на 2048 бит может шифровать до 245 байт. (Обратите внимание, что в SQL Server и сертификаты, и асимметричные ключи служат оболочками для ключей RSA.)

- DECRYPTBYASYMKEY.

Дешифрует данные асимметричным ключом. Шифрование и дешифрование асимметричным ключом являются очень дорогостоящими по сравнению с шифрованием и дешифрованием симметричным

ключом. Рекомендуется использовать асимметричный ключ при работе с большими наборами данных, например, с данными пользователей в таблицах.

Переходим непосредственно к шифрованию нашей базы данных. Рассмотрим пример шифрования с помощью асимметричного ключа:

1. Войдём в нашу базу данных (“Библиотека”) и создадим для неё новый запрос. Чтобы создать асимметричный ключ, напишем следующий код:

```
CREATE ASYMMETRIC KEY SampleAsymKey
WITH ALGORITHM = RSA_2048
ENCRYPTION BY PASSWORD = '<Password>'
```

Если всё введено верно вы увидите сообщение об успешном выполнении команд. Во вкладке Безопасность => Асимметричные ключи нашей базы данных появится созданный нами ассиметричный ключ (Рис 3).

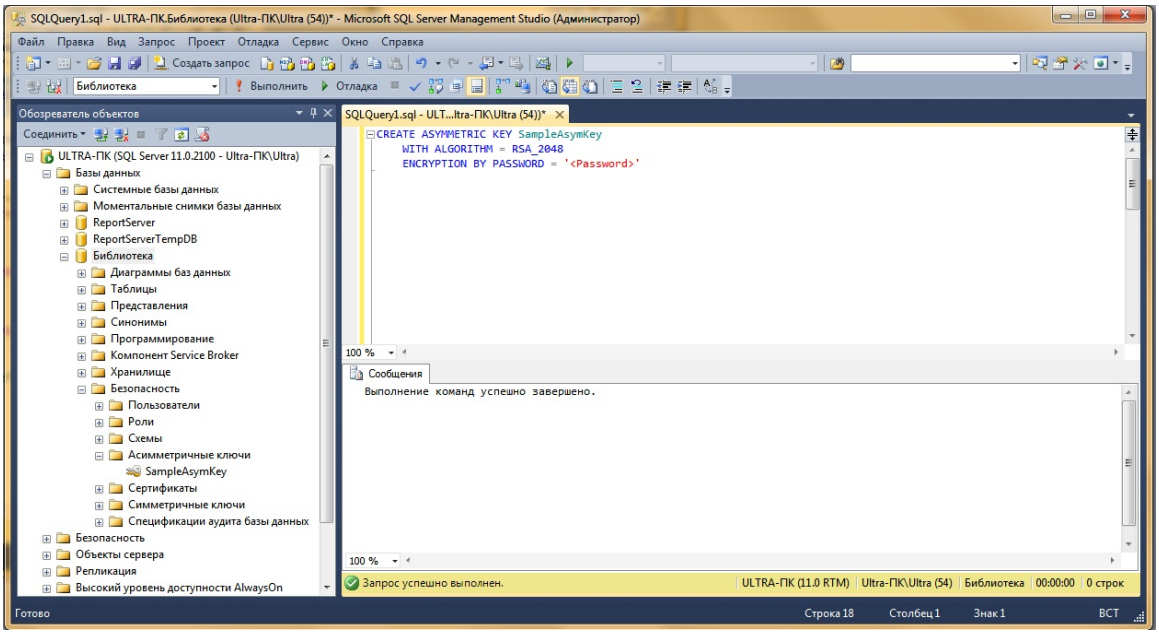


Рис. 3 – Создание асимметричного ключа

2. Следующий шаг – объявление переменных. Объявим две переменные (текстовая и бинарная). В текстовой записываем некий текст и выведем его на экран. Зашифруем его с помощью функции ENCRYPTBYASYMKEY и расшифруем с помощью функции DECRYPTBYASYMKEY (Рис. 4):

```
declare @plaintext nvarchar(60)
```

```
declare @ciphertext varbinary(256)

--Инициализировать открытый текст
set @plaintext='Это простой текст'
print @plaintext

--Шифруем
set @ciphertext=ENCRYPTBYASYMKEY
(AsymKey_ID('SampleAsymKey'),@plaintext)
print @ciphertext

--Расшифровка
set @ciphertext=DECRYPTBYASYMKEY
(AsymKey_ID('SampleAsymKey'),@ciphertext,N'<Password>')
print cast (@plaintext as nvarchar(max))
```

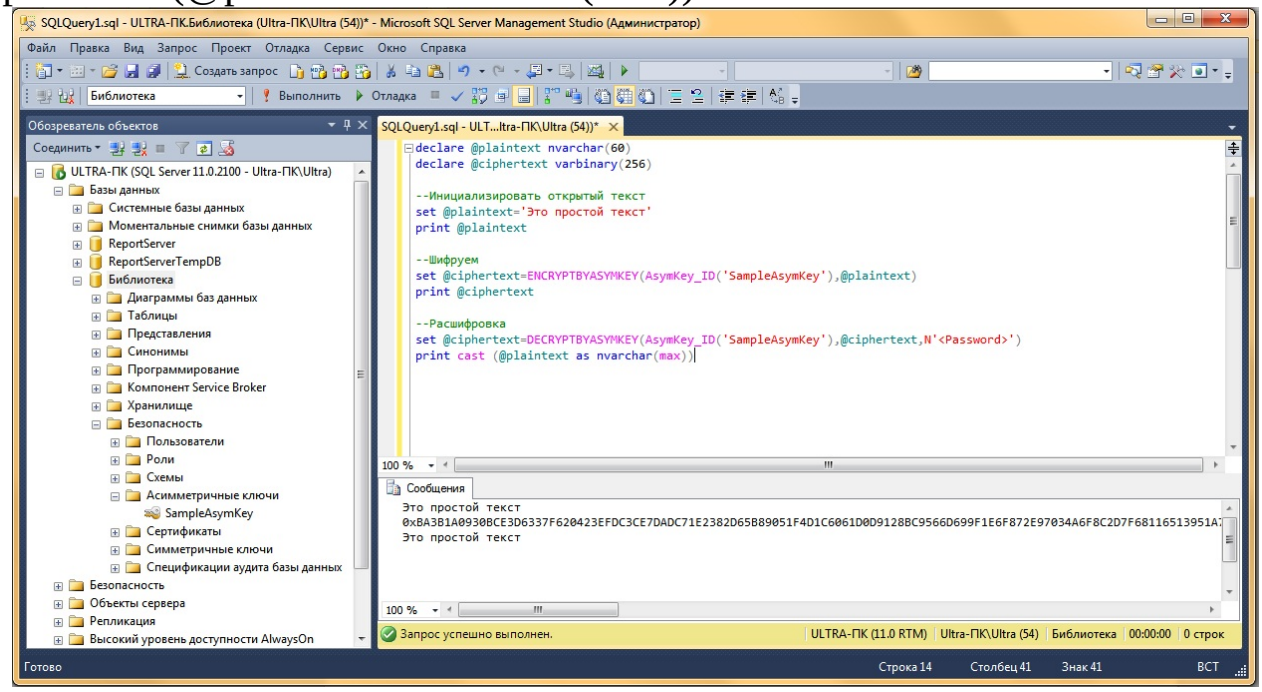


Рис. 4 – Шифрование и дешифрование текста

3. Чтобы удалить асимметричный ключ введите следующий запрос (Рис. 5):

```
DROP ASYMMETRIC KEY SampleAsymKey
```

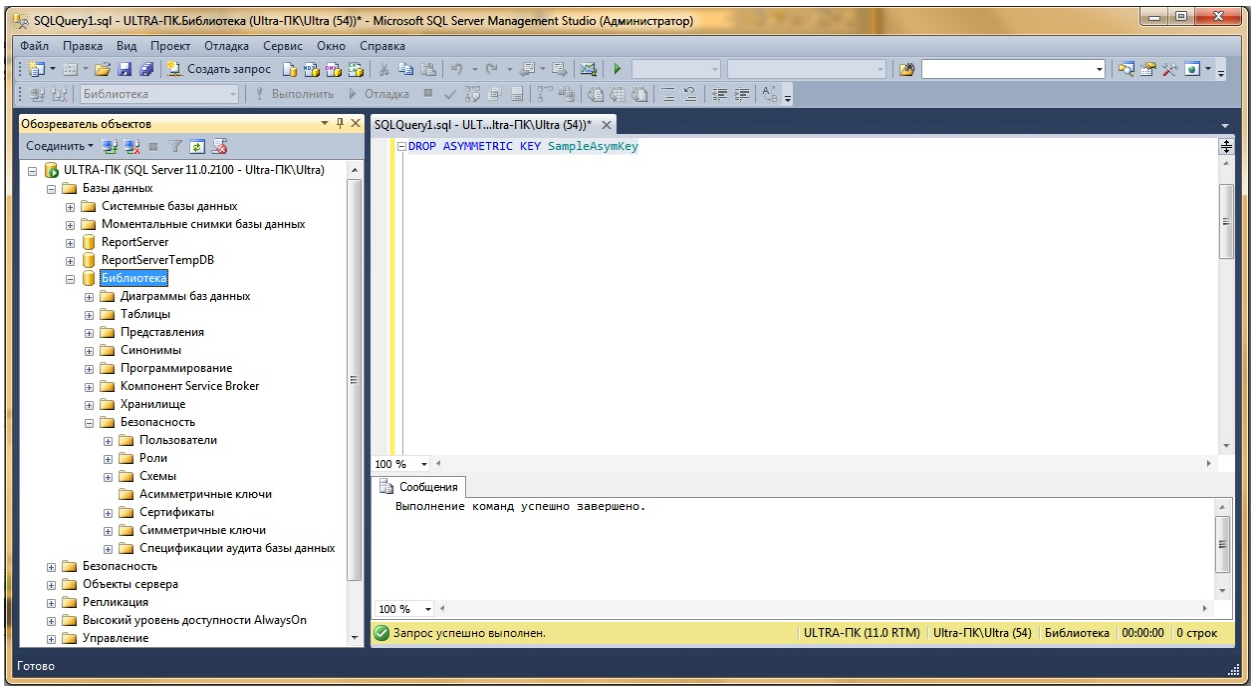


Рис. 5 – Удаление асимметричного ключа

Рассмотрим шифрование с помощью сертификата. Функции для управления сертификатами:

1. CREATE CERTIFICATE - добавляет сертификат в базу данных SQL Server. Сертификат — это защищаемый объект уровня базы данных, соответствующий стандарту X.509 и поддерживающий поля X.509 V1. Инструкция CREATE CERTIFICATE может загрузить сертификат из файла или сборки. Она также может создать пару ключей и самостоятельно подписанный сертификат.

Закрытые ключи, создаваемые SQL Server имеют в длину 1024 бит. Закрытые ключи, импортированные из внешнего источника, имеют минимальную длину в 384 бит и максимальную длину в 4096 бит. Длина импортируемого закрытого ключа должна быть кратной 64 бит. Закрытый ключ должен соответствовать открытому ключу, указанному в аргументе certificate_name.

При создании сертификата из контейнера загрузка закрытого ключа необязательна. Однако в случае, когда SQL Server создает самостоятельно подписанный сертификат, закрытый ключ создается всегда. По умолчанию закрытый ключ шифруется главным ключом базы данных. Если главного ключа базы данных не существует, а пароль не указан, произойдет ошибка при выполнении инструкции.

Если закрытый ключ зашифрован главным ключом базы данных, указывать пароль расшифровки не требуется.

2. ALTER CERTIFICATE - изменяет закрытый ключ, используемый для шифрования сертификата, или добавляет закрытый ключ, если он не существует. Изменяет доступность сертификата для компонента Компонент Service Broker. Закрытый ключ должен соответствовать открытому ключу, заданному в аргументе certificate_name. Предложение DECRYPTION BY PASSWORD может быть опущено, если пароль в файле защищен паролем, равным NULL.

Если закрытый ключ сертификата, уже существующий в базе данных, импортируется из файла, закрытый ключ будет автоматически защищен главным ключом базы данных. Чтобы защитить закрытый ключ паролем, используйте предложение ENCRYPTION BY PASSWORD.

Параметр REMOVE PRIVATE KEY удалит закрытый ключ сертификата из базы данных. Это можно сделать, если сертификат будет использоваться для проверки подписей или в сценариях компонента Компонент Service Broker, не требующих закрытого ключа. Не удаляйте закрытый ключ сертификата, который защищает симметричный ключ.

Указывать пароль расшифровки не нужно, если закрытый ключ зашифрован с помощью главного ключа базы данных.

- Изменение пароля сертификата:

```
ALTER CERTIFICATE Shipping04
WITH PRIVATE KEY
(DECRYPTION BY PASSWORD = 'pGF$5DGvbd2439587y',
ENCRYPTION BY PASSWORD = '4-329578thlkajdshglXCSgf');
GO
```

- Изменение пароля, используемого для шифрования закрытого ключа:

```
ALTER CERTIFICATE Shipping11
WITH PRIVATE KEY
(ENCRYPTION BY PASSWORD = '34958tosdgfkh##38',
DECRYPTION BY PASSWORD = '95hkjdskghFDGGG4%');
GO
```

- Изменение защиты закрытого ключа паролем на защиту главным ключом базы данных:

```
ALTER CERTIFICATE Shipping15
WITH PRIVATE KEY
(DECRYPTION BY PASSWORD = '95hk000eEnvjkjy#F%');
GO
```

3. DROP CERTIFICATE - удаляет сертификат из базы данных. Резервная копия сертификата, используемого для шифрования базы данных, должна быть сохранена, даже если она больше не включена в базе данных. Даже если база данных больше не зашифрована, части журнала транзакций могут по-прежнему оставаться защищенными, а для некоторых операций будет требоваться сертификат до выполнения полного резервного копирования базы данных.

Сертификат также потребуется для восстановления из резервных копий, созданных в то время, когда база данных была зашифрована. Сертификаты можно удалять только при условии, что с ними не связано никаких сущностей.

4. BACKUP CERTIFICATE - экспортирует сертификат в файл. Если закрытый ключ зашифрован с паролем в базе данных, необходимо указать пароль для дешифрования.

При создании резервной копии закрытого ключа в файле шифрование является необходимым. Пароль, используемый для защиты резервной копии сертификата, не является тем же ключом, который применялся для шифрования закрытого ключа сертификата.

Чтобы восстановить сертификат из резервной копии, используйте инструкцию CREATE CERTIFICATE.

5. ENCRYPTBYCERT - шифрует данные на открытом ключе сертификата.

Эта функция шифрует данные при помощи открытого ключа сертификата. Код может быть расшифрован только с помощью соответствующего закрытого ключа. Ассиметричные преобразования гораздо более накладны, чем шифрование и дешифрование с использованием симметричного ключа. Поэтому использование ассиметричного шифрования не рекомендуется при работе с большими объемами данных, например, таблицами пользовательских данных.

6. DECRYPTBYCERT - расшифровывает данные при помощи закрытого ключа сертификата.

Эта функция расшифровывает данные при помощи закрытого ключа сертификата. Криптографические преобразования с использованием асимметричных ключей требуют значительных ресурсов. Поэтому функции EncryptByCert и DecryptByCert не подходят для операций шифрования пользовательских данных.

Рассмотрим пример использования функций для управления сертификатами:

1. Вначале создадим папку (Cerf), где будут храниться сертификаты (Рис. 6)

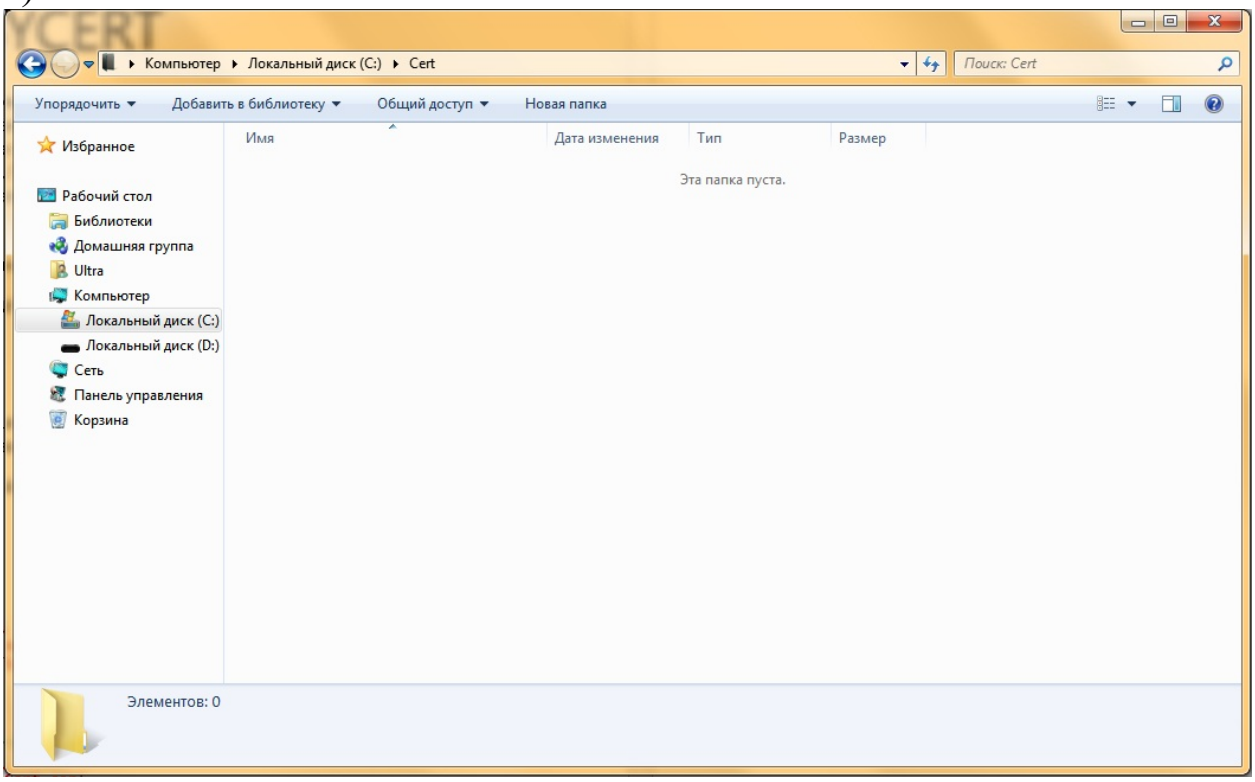


Рис. 6 – Папка для хранения сертификатов

2. Создаём сертификат в нашей базе данных, который шифруется с помощью пароля, указываем тему сертификата и время действия сертификата. Экспортируем его в файловую систему и указываем приватный ключ. Задаём пароль для шифрации и дешифрации (Рис.7):

```
USE Библиотека go
--Создаём сертификат
CREATE CERTIFICATE SampleCert
ENCRYPTION BY PASSWORD = 'Password'
WITH SUBJECT = 'Sample certificate',
EXPIRY_DATE = '2026-10-10';

--Сохраняем сертификат в виде файла
BACKUP CERTIFICATE SampleCert
TO FILE='C:\Cert\BackupSampleCert.cer'
WITH PRIVATE KEY ( FILE='C:\Cert\BackupSampleCert.pvk',
ENCRYPTION BY PASSWORD='p@ssword',
DECRYPTION BY PASSWORD='Password')
```

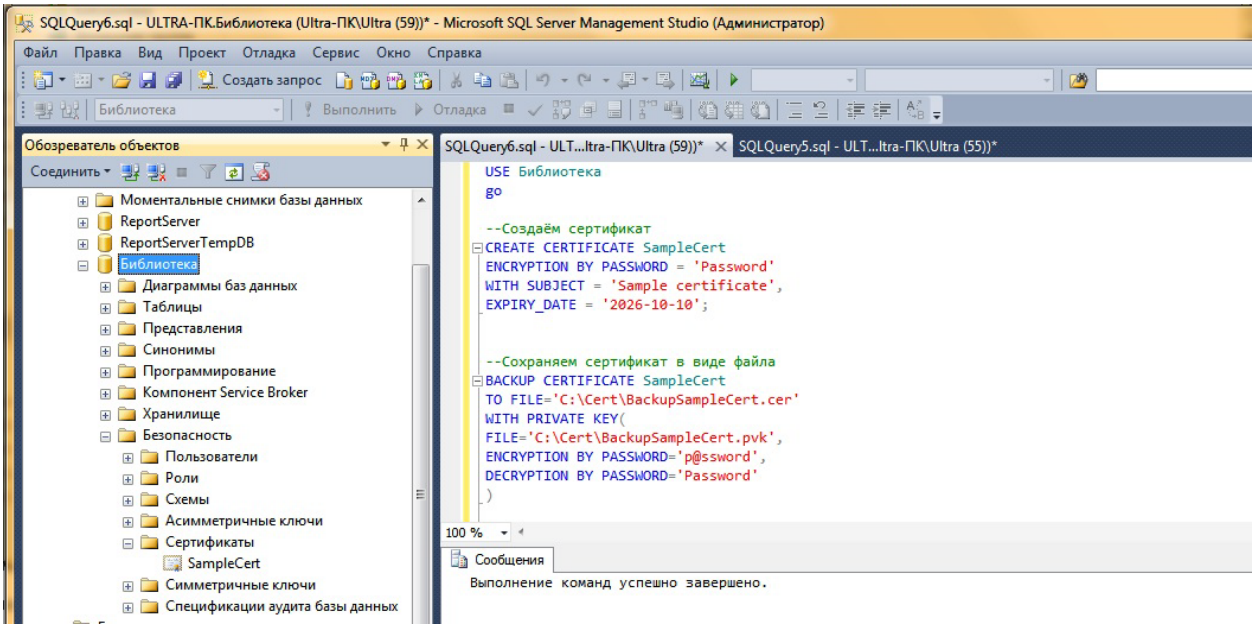


Рис. 7 – Создание сертификата

4. Чтобы удалить сертификат из sql server введите следующий запрос (Рис. 8):

```
DROP CERTIFICATE SampleCert
```

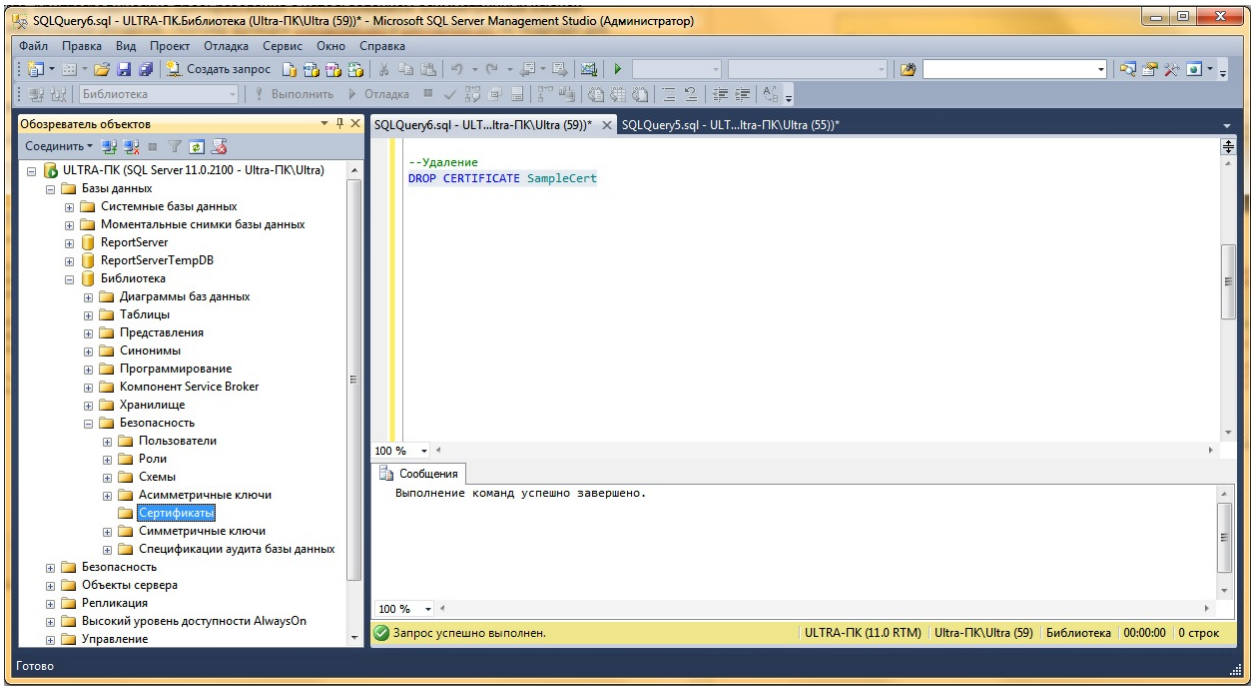


Рис. 8 – Удаление сертификата

5. Чтобы воссоздать из файла введите следующий вопрос (Рис. 9):

```
CREATE CERTIFICATE SampleCert
FROM FILE='C:\Cert\BackupSampleCert.cer'
WITH PRIVATE KEY(
```

FILE='C:\Cert\BackupSampleCert.pvk',
ENCRYPTION BY PASSWORD='p@ssword',
DECRYPTION BY PASSWORD='Password')

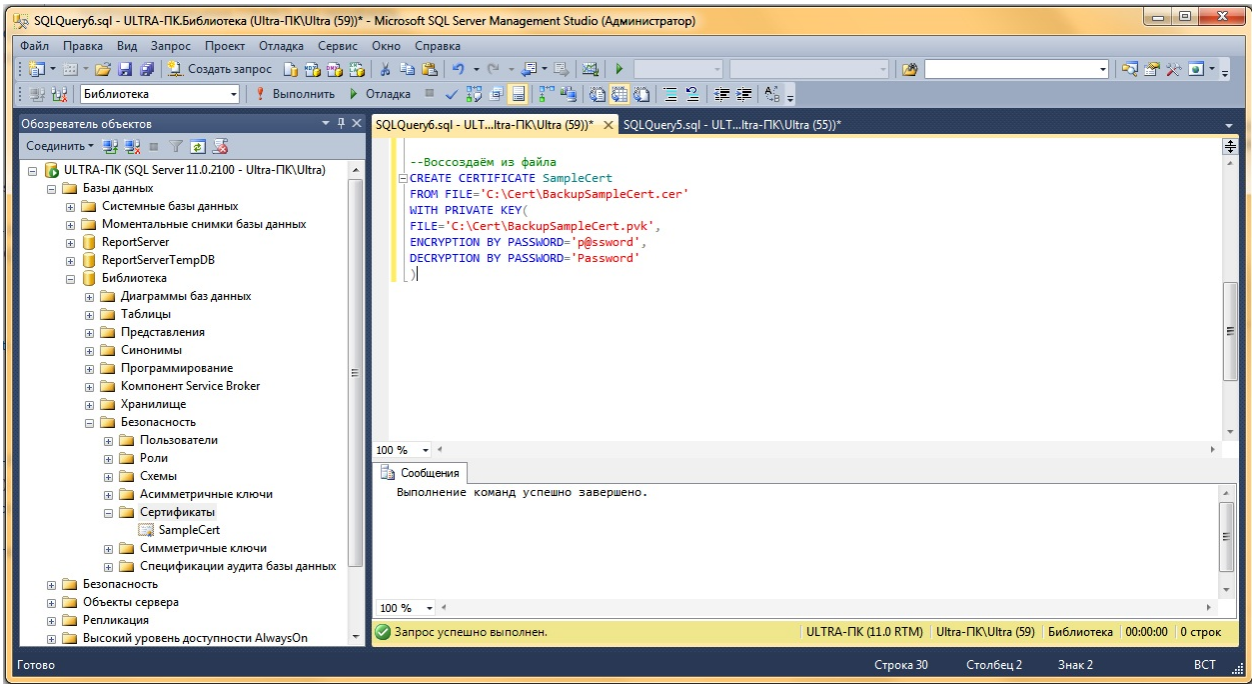


Рис. 9 – Воссоздаём сертификат из созданного файла

Рассмотрим симметричные ключи. Функции для управления симметричными ключами:

- CREATE SYMMETRIC KEY - создает симметричный ключ и указывает его свойства в SQL Server. После создания симметричный ключ должен быть зашифрован с помощью по крайней мере одного из следующих средств: сертификат, пароль, симметричный ключ, асимметричный ключ или PROVIDER. Ключ может быть зашифрован более чем один раз для каждого типа шифрования. Другими словами, один симметричный ключ может быть зашифрован с использованием нескольких сертификатов, симметричных ключей и асимметричных ключей одновременно.

Если симметричный ключ шифруется с использованием пароля вместо открытого ключа главного ключа базы данных, то используется алгоритм шифрования TRIPLE DES. Поэтому ключи, созданные с помощью сильных алгоритмов шифрования, таких как AES, защищены с помощью более слабого алгоритма.

Необязательный пароль можно использовать для шифрования симметричного ключа перед распространением ключа нескольким пользователям.

Владельцем временных ключей является пользователь, который создает их. Временные ключи действительны только в текущем сеансе.

Аргумент IDENTITY_VALUE формирует идентификатор GUID, с помощью которого маркируются данные, зашифрованные новым симметричным ключом. Маркирование может быть использовано для сопоставления ключей шифрованным данным.Идентификатор GUID, формируемый указанной фразой, будет всегда одним и тем же. Фраза, использованная для создания идентификатора GUID, не может быть повторно использована до тех пор, пока активен хотя бы один сеанс, использующий эту фразу.Предложение IDENTITY_VALUE является необязательным, но рекомендуется использовать его для хранения данных, зашифрованных с применением временного ключа.

Не существует алгоритма шифрования по умолчанию.

Защищать конфиденциальные данные с помощью потоковых шифров RC4 и RC4_128 не рекомендуется

Сведения о симметричных ключах доступны в представлении каталога sys.symmetric_keys.

Симметричные ключи не могут быть зашифрованы с помощью симметричных ключей, созданных поставщиком шифрования.

Пояснение к алгоритмам DES:

- DESX был именован неправильно. Симметричные ключи, созданные с параметром ALGORITHM = DESX, в действительности используют шифр TRIPLE DES с 192-битным ключом. Алгоритм DESX не предоставляется. В будущей версии Microsoft SQL Server этот компонент будет удален. Избегайте использования этого компонента в новых разработках и запланируйте изменение существующих приложений, в которых он применяется.
- Симметричные ключи, созданные с параметром ALGORITHM = TRIPLE_DES_3KEY, используют шифр TRIPLE DES с 192-битным ключом.
- Симметричные ключи, созданные с параметром ALGORITHM = TRIPLE_DES, используют шифр TRIPLE DES с 128-битным ключом.

Многократное использование одного и того же RC4 или RC4_128 KEY_GUID для различных блоков данных приведет к одному и тому же ключу RC4, так как SQL Server не предоставляет рассеивание автоматически. Повторное использование одного и того же ключа RC4 является типичной ошибкой, становящейся причиной очень слабого шифрования. Таким образом, ключевые слова RC4 и RC4_128 являются

устаревшими. В будущей версии Microsoft SQL Server этот компонент будет удален. Не используйте его при работе над новыми приложениями и как можно быстрее измените приложения, в которых он в настоящее время используется.

Алгоритм RC4 поддерживается только в целях обратной совместимости. Когда база данных имеет уровень совместимости 90 или 100, новые материалы могут шифроваться только с помощью алгоритмов RC4 или RC4_128. (Не рекомендуется.) Используйте вместо этого более новые алгоритмы, например AES. В SQL Server 2014 материалы, зашифрованные с помощью алгоритмов RC4 или RC4_128, могут быть расшифрованы на любом уровне совместимости.

2. ALTER SYMMETRIC KEY - изменяет свойства симметричного ключа. Для изменения шифрования симметричного ключа используйте предложения ADD ENCRYPTION и DROP ENCRYPTION. Невозможно, чтобы ключу не соответствовал никакой способ шифрования. Поэтому оптимальным способом изменения метода шифрования является добавление новой формы шифрования и затем удаление старой.

Для изменения владельца симметричного ключа выполните инструкцию ALTER AUTHORIZATION.

3. DROP SYMMETRIC KEY - удаляет симметричный ключ из текущей базы данных. Если ключ является открытым в текущем сеансе, то инструкция завершится с ошибками.

Если асимметричный ключ был сопоставлен с ключом расширенного управления ключами на устройстве расширенного управления ключами, а параметр REMOVE PROVIDER KEY не был указан, то ключ будет удален из базы данных, но не с устройства; также будет выдано предупреждение.

4. OPEN SYMMETRIC KEY - расшифровывает симметричный ключ и делает его доступным для использования. Открытые симметричные ключи привязаны к сеансу, а не к контексту безопасности. Открытый ключ останется доступным, пока не будет явно закрыт или сеанс не будет прерван. Если открыт симметричный ключ, после чего произошло переключение контекста, ключ останется открытым и будет доступным в олицетворенном контексте.

Если симметричный ключ был зашифрован другим ключом, сначала необходимо открыть этот ключ. Если симметричный ключ уже открыт, то

запрос является запросом NO_OP. Если пароль, сертификат или ключ для расшифровки симметричного ключа неверен, запрос завершается сбоем.

Симметричные ключи, созданные из поставщиков шифрования, не могут быть открыты. Операции шифрования и расшифровки, для которых используется симметричный ключ этого типа, выполняются успешно без инструкции OPEN, поскольку открытие и закрытие ключа осуществляется поставщиком шифрования.

5. CLOSE SYMMETRIC KEY - закрывает симметричный ключ или все симметричные ключи, открытые в текущем сеансе. Открытые симметричные ключи привязаны к сеансу, а не к контексту безопасности. Открытый ключ останется доступным, пока не будет явно закрыт или сеанс не будет прерван. Инструкция CLOSE ALL SYMMETRIC KEYS закрывает любой главный ключ базы данных, который был открыт в текущем сеансе при помощи инструкции OPEN MASTER KEY.

6. ENCRYPTBYKEY - производит шифрование данных при использовании симметричного ключа. Функция EncryptByKey использует симметричный ключ. Этот ключ должен быть открыт. Если симметричный ключ уже открыт в текущем сеансе, не нужно открывать его снова в контексте запроса.

Структура проверки подлинности позволяет исключить прямую подмену зашифрованных полей.

Если при шифровании данных указана структура проверки подлинности, то для дешифровки данных функцией DecryptByKey потребуются эти же данная структура проверки подлинности. При шифровании хэш данных проверки подлинности шифруется вместе с данными. При дешифровке эта же структура проверки подлинности должны быть переданы функции DecryptByKey. Если эти данные не совпадают, дешифровка завершится ошибкой. Это будет означать, что значение перемещено с другого места со времени шифрования. В качестве структуры проверки подлинности рекомендуется использовать столбец, содержащий уникальное и неизменное значение. Если значение структуры проверки подлинности изменится, можно утратить доступ к данным.

Симметричное шифрование и расшифровка осуществляются относительно быстро и подходит для работы с большими объемами данных.

7.DECRYPTBYKEY - расшифровывает данные с помощью симметричного ключа. В функции DecryptByKey используется симметричный ключ. Этот симметричный ключ должен быть открыт уже в базе данных. Одновременно могут быть открыты несколько ключей. Открывать ключ непосредственно перед раскодированием необязательно.

Симметричное кодирование и декодирование осуществляется относительно быстро и подходит для работы с большими объемами данных.

Рассмотрим пример шифрования с помощью симметричного ключа. Будем создавать два ключа первый будет шифровать данные, а второй будет шифровать первый ключ.

1. Сначала создаём второй ключ, а потом создаём ключ, который шифрует данные (Рис. 10):

```
USE Библиотека
go
--Создаём симметричный ключ для шифрования другого симметричного ключа
CREATE SYMMETRIC KEY SymKey
WITH ALGORITHM =Triple_DES
ENCRYPTION BY PASSWORD='password'
```

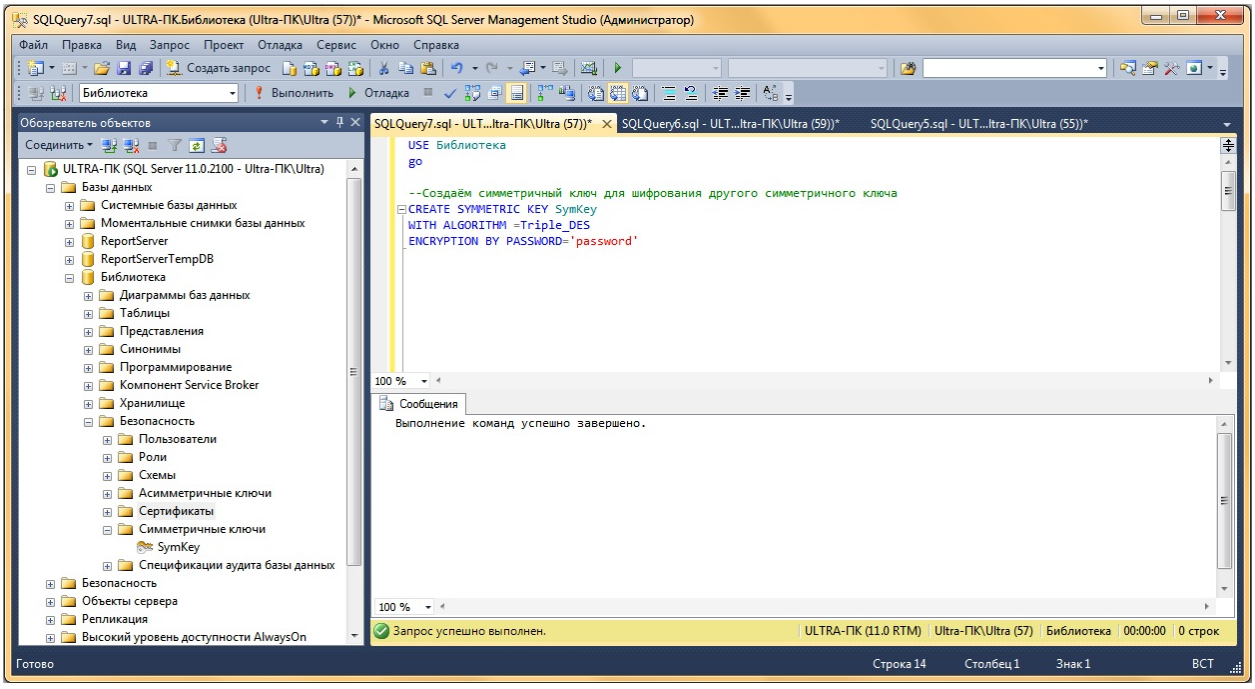


Рис. 10 – Создание ключа для шифрования другого ключа

2. Открываем этот ключ:

```
OPEN SYMMETRIC KEY SymKey
```

DECRYPTION BY PASSWORD='password'

3. Создаём и открываем ключ, который шифрует данные (Рис.11):

```
--Создаём симметричный ключ для шифрования данных
CREATE SYMMETRIC KEY SymData
WITH ALGORITHM=Triple_DES
ENCRYPTION BY SYMMETRIC KEY SymKey
--Открываем ключ для шифрования данных
OPEN SYMMETRIC KEY SymData
DECRYPTION BY SYMMETRIC KEY SymKey
```

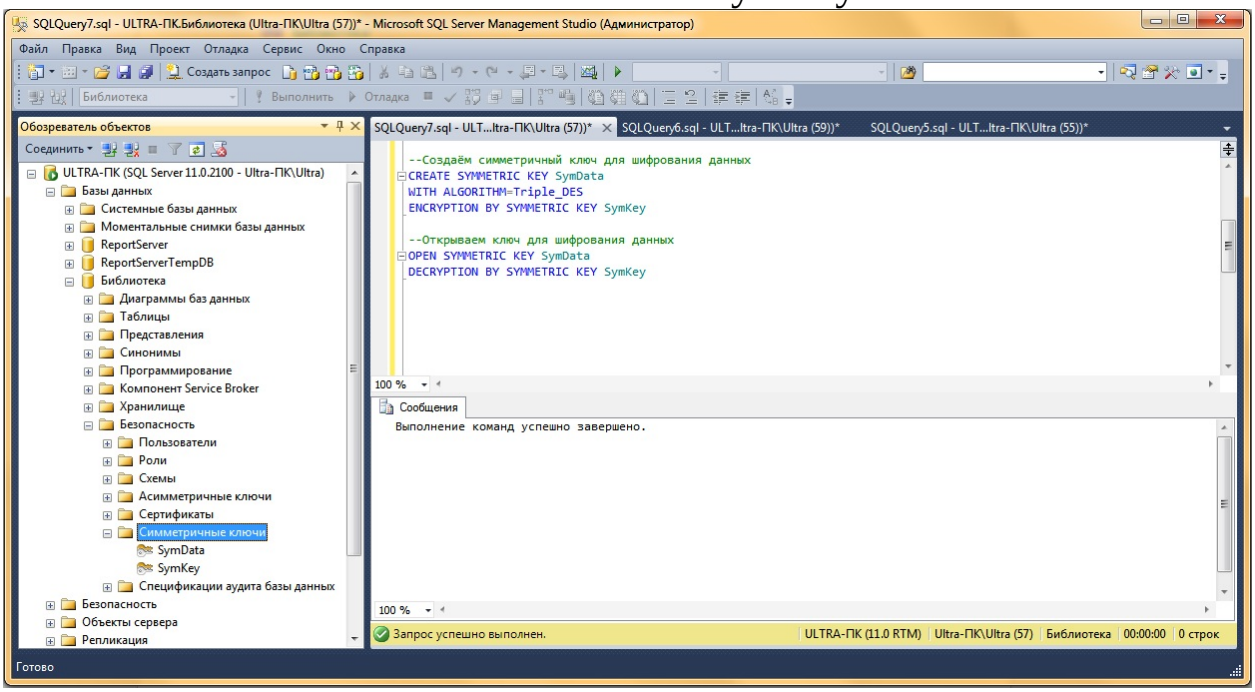


Рис. 11 – Создание ключ для шифрования данных

4. Инициализируем текст, для шифрования, шифруем его и сразу же расшифровываем (Рис. 12):

```
--Инициализируем открытый текст
DECLARE @plaintext nvarchar(512)
SET @plaintext='Добрый день'
print @plaintext

--Шифруем данные
DECLARE @ciphertext varbinary(1024)
SET @ciphertext=ENCRYPTBYKEY
(KEY_GUID('SymData'),@plaintext)
print @ciphertext
```

--Расшифровываем данные
SET @plaintext=CAST
(DECRYPTBYKEY(@ciphertext) as nvarchar(512))
print @plaintext

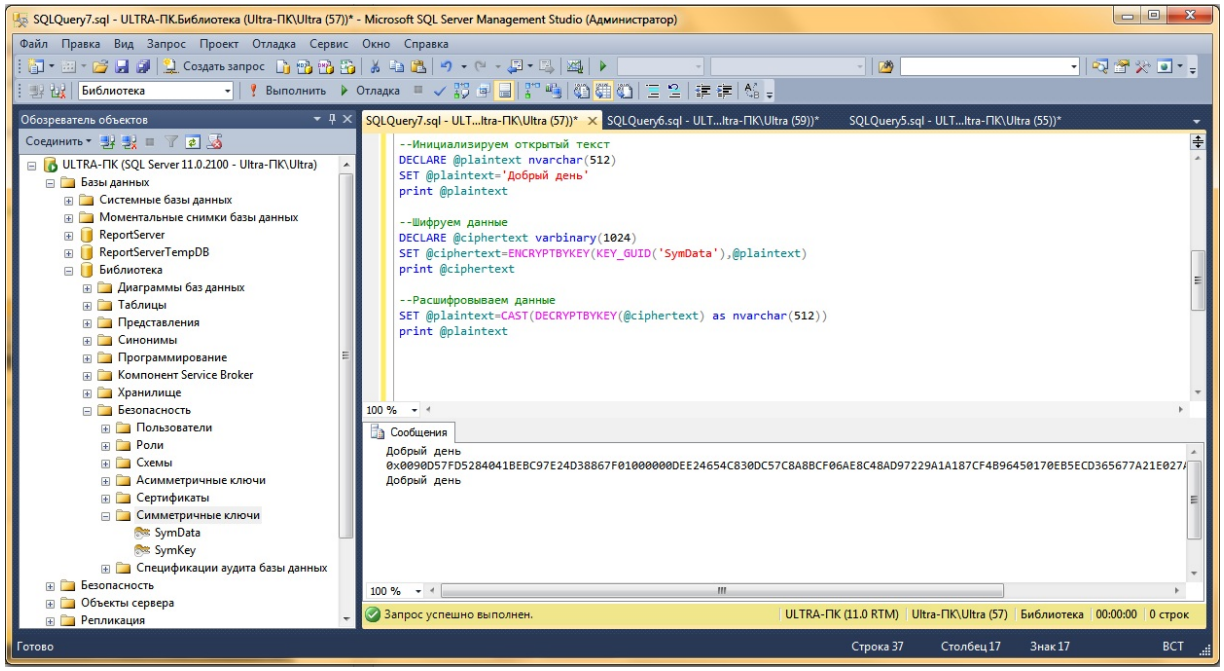


Рис. 12 – Шифрование и дешифрование текста

5. Закрываем ключ шифрования данных и ключ шифрования ключа (Рис. 13):

--Закрываем ключ шифрования данных
CLOSE SYMMETRIC KEY SymData
--Закрываем ключ шифрования ключа
CLOSE SYMMETRIC KEY SymKey

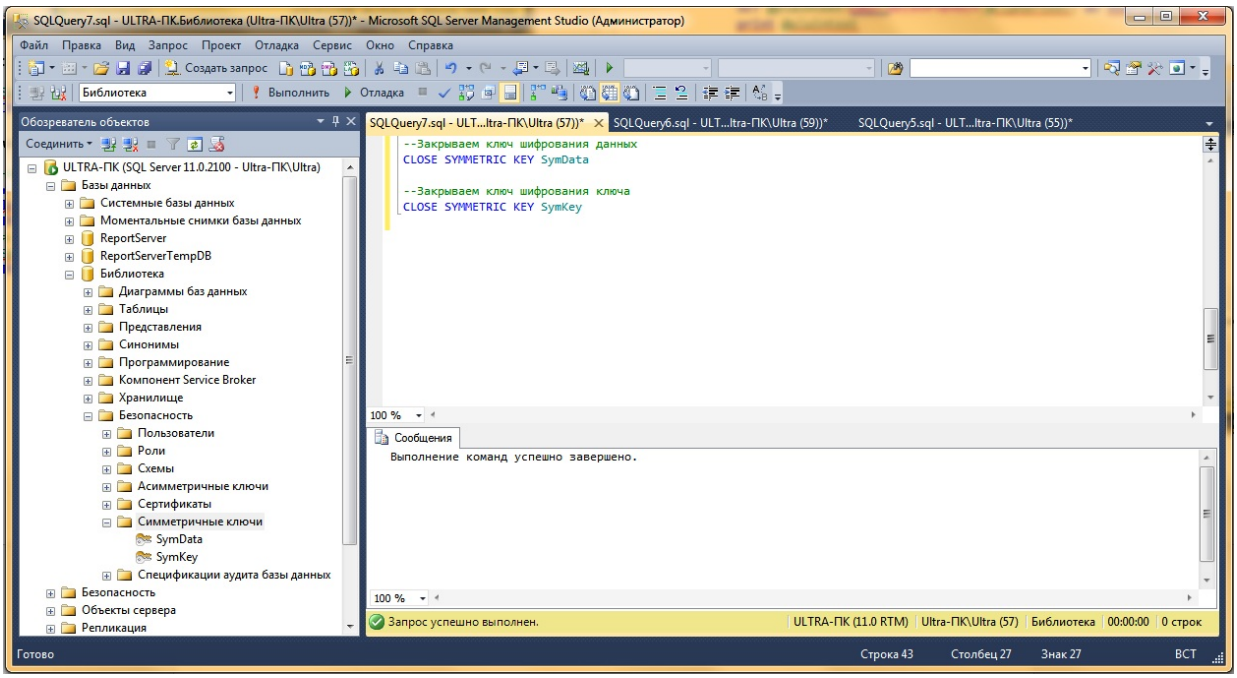


Рис. 13 – Закрываем симметричные ключи

SQL SERVER 2008 позволяет также шифровать данные с использование ключевых фраз (PassPhase). Ключевая фраза - это строка или двоичное значение от которого sql server может наследовать симметричный ключ для шифрования данных.

16.Функция ENCRYPTBYPASSPHRASE - шифрование данных с помощью парольной фразы с использованием алгоритма TRIPLE DES и 128-битного ключа.

DECRYPTBYPASSPHRASE - расшифровывает данные, зашифрованные с помощью парольной фразы.

Пример (Рис. 14):

```
Declare @plaintext nvarchar(1000), @enc_text varbinary(2000)
SET @plaintext='Я помню чудное мгновенье'
SET @enc_text=ENCRYPTBYPASSPHRASE('LOM',@plaintext)
SELECT 'Оригинальный текст: ',@plaintext
SELECT 'Зашифрованный текст: ',@enc_text
SELECT 'Расшифровка:', CAST (DECRYPTBYPASSPHRASE
('LOM',@enc_text) as nvarchar(1000))
```

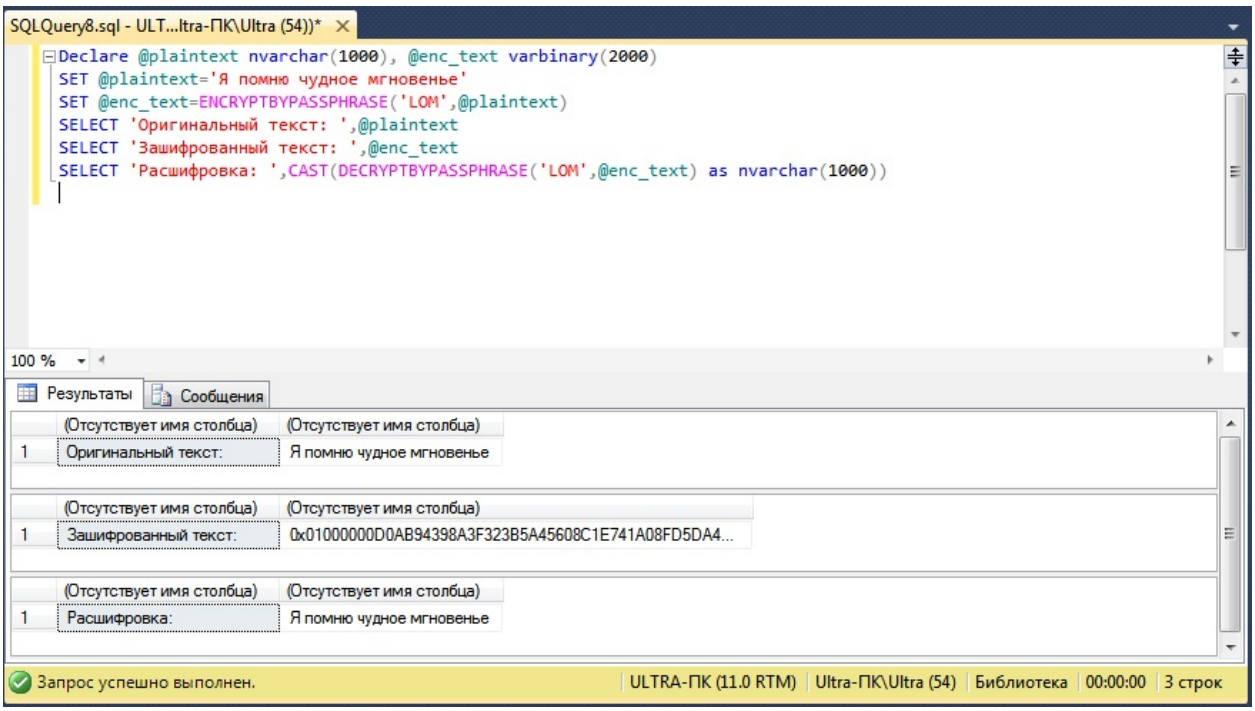


Рис. 14 – Шифрование с помощью ключевой фразы

Создадим базу данных, данные в которой шифруются:

1. с использованием симметричного ключа, который будет зашифрован с помощью асимметричного ключа.
2. с использованием симметричного ключа, но на этот раз симметричный ключ шифруется сертификатом.

Создание базы данных:

```
USE [master]
GO
CREATE DATABASE [DB]
GO
```

Запустите код T-SQL для создания таблицы с именем TelephonNumber в базе данных DB:

```
USE [DB]
GO
CREATE TABLE [dbo].[ TelephonNumber]
([PersonID] [int] PRIMARY KEY,
[TelephonNumber] [varbinary](max))
GO
```

Эта таблица будет содержать ложную информацию о телефонных номерах. Номера телефонов будут сохранены в столбце двоичных переменных, потому что они будут шифроваться.

Используйте следующий программный код для создания главного ключа DMK базы данных DB, шифруемого с помощью парольной фразы $tr0nGPa$$w0rd:

```
USE [DB]
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '$tr0nGPa$$w0rd'
GO
```

Необходимо создать асимметричный ключ, зашифровать его парольной фразой $tr0nGPa$$w0rd, создать симметричный ключ и зашифровать симметричный ключ с помощью только что созданного асимметричного ключа. Выполнить эти задачи можно, запустив программный код:

```
USE [DB]
GO
--Создание асимметричного ключа, зашифрованного парольной фразой StrongPa$$w0rd!
CREATE ASYMMETRIC KEY MyAsymmetricKey
WITH ALGORITHM = RSA_2048
ENCRYPTION BY PASSWORD = 'StrongPa$$w0rd!'
GO
--Создание симметричного ключа, зашифрованного асимметричным ключом
CREATE SYMMETRIC KEY MySymmetricKey
WITH ALGORITHM = AES_256
ENCRYPTION BY ASYMMETRIC KEY MyAsymmetricKey
GO
```

1. Теперь мы можем приступить к шифрованию данных. Для этого необходимо сначала открыть симметричный ключ, только что созданный с помощью команды OPEN SYMMETRIC KEY, за которой следует имя симметричного ключа. Затем указать, что нужно расшифровать его с использованием заданного асимметричного ключа. Программный код выглядит следующим образом:

```
USE [DB]
GO
OPEN SYMMETRIC KEY MySymmetricKey
DECRYPTION BY ASYMMETRIC KEY MyAsymmetricKey
WITH PASSWORD = 'StrongPa$$w0rd!'
```

GO

После выполнения этого кода направьте запрос в представление sys.openkeys, чтобы убедиться, что ключ открыт (Рис.15):

```
USE [DB]
GO
SELECT * FROM [sys].[openkeys]
```

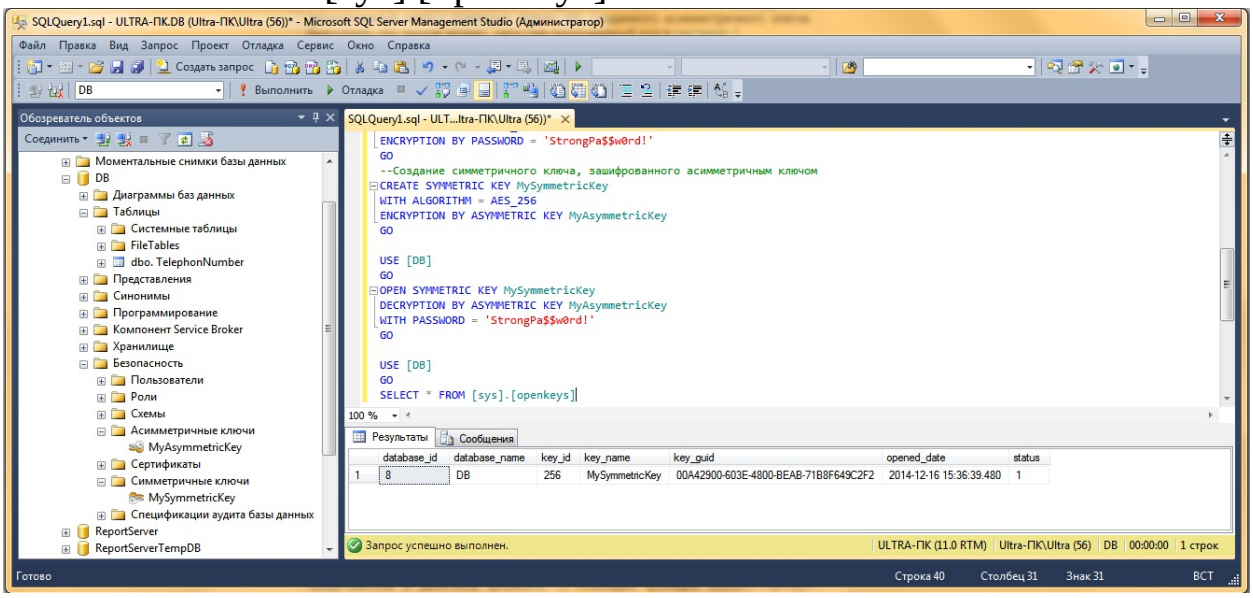


Рис. 15 – Открытие симметричного ключа

Введем несколько телефонных номеров в таблицу TelephoneNumber, запустив код:

```
USE [DB]
GO
DECLARE @SymmetricKeyGUID AS [uniqueidentifier]
SET @SymmetricKeyGUID = KEY_GUID('MySymmetricKey')
IF (@SymmetricKeyGUID IS NOT NULL)
BEGIN
INSERT INTO [dbo].[ TelephonNumber]
VALUES (01, ENCRYPTBYKEY(@SymmetricKeyGUID,
N'8-761-123-87-63'))
INSERT INTO [dbo].[ TelephonNumber]
VALUES (02, ENCRYPTBYKEY(@SymmetricKeyGUID,
N'8-768-765-87-65'))
INSERT INTO [dbo].[ TelephonNumber]
VALUES (03, ENCRYPTBYKEY(@SymmetricKeyGUID,
N'8-761-234-11-11'))
```

```
END
TRUNCATE TABLE [dbo].[TelephonNumber]
```

Выведем получившуюся таблицу на экран (Рис. 16):

```
USE [DB]
GO
SELECT * FROM [dbo].[ TelephonNumber]
```

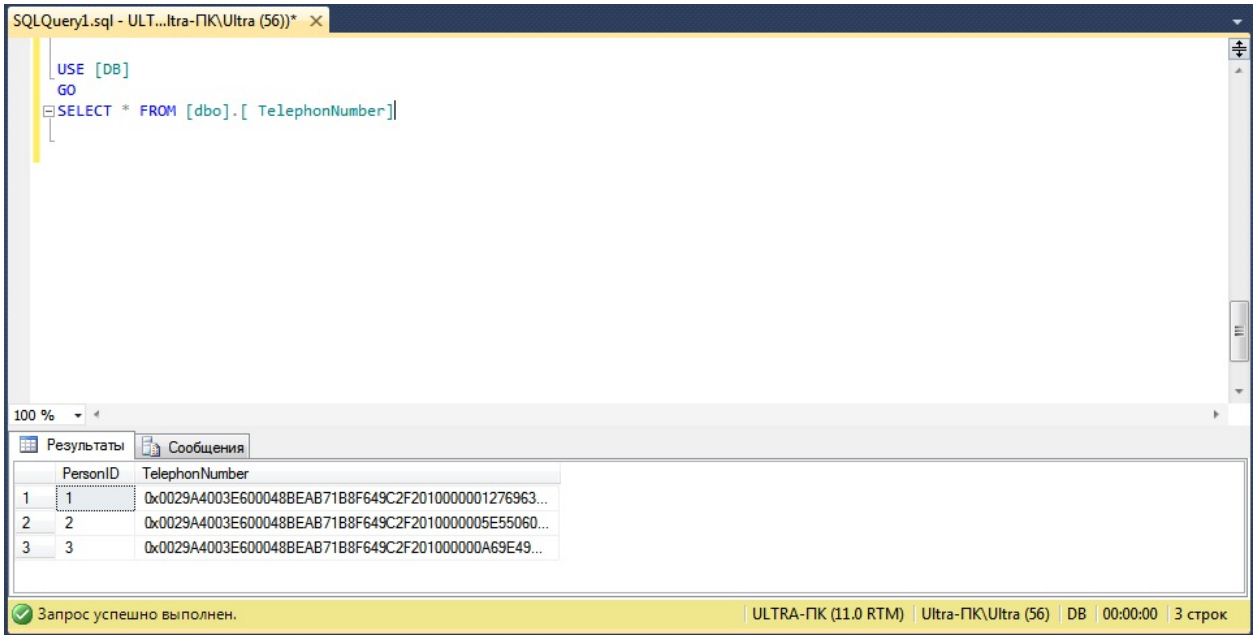


Рис. 16 – Зашифрованные данные из таблицы TelephoneNumber

Все данные в столбце TelephoneNumber представлены в двоичном формате. С помощью функции DECRYPTBYKEY можно просмотреть зашифрованные данные (Рис. 17):

```
USE [DB]
GO
SELECT [PersonID],
CONVERT([nvarchar](32),
DECRYPTBYKEY(TelephonNumber))
AS [TelephonNumber]
FROM [dbo].[ TelephonNumber]
GO
```

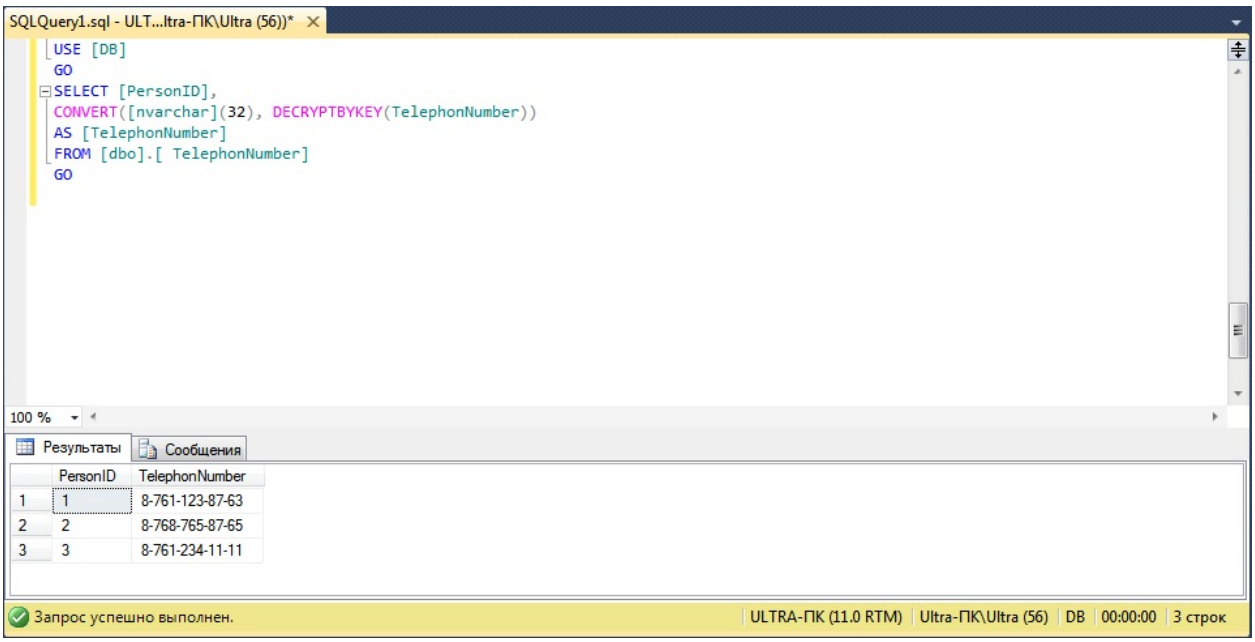


Рис. 17– Просмотр зашифрованных данных

2. Создадим сертификат с помощью инструкции CREATE CERTIFICATE. Затем создается симметричный ключ, шифруемый сертификатом. Наконец, открыв симметричный ключ, код вставляет три строки в таблицу TelephonNumber:

```
USE [DB]
GO
--Создание сертификата
CREATE CERTIFICATE
[CertToEncryptSymmetricKey]
WITH SUBJECT ='Самозаверяющий сертификат
для шифрования симметричного ключа.'
--Создание симметричного ключа, зашифрованного сертификатом
CREATE SYMMETRIC KEY
[SymmetricKeyEncryptedWithCert]
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE
[CertToEncryptSymmetricKey]
--Открытие симметричного ключа
OPEN SYMMETRIC KEY
[SymmetricKeyEncryptedWithCert]
DECRYPTION BY CERTIFICATE
[CertToEncryptSymmetricKey]
--Усечение таблицы TelephonNumber
```

```
TRUNCATE TABLE
[dbo].[ TelephonNumber]
--Вставка данных в таблицу
DECLARE @SymmetricKeyGUID
AS [uniqueidentifier]
SET @SymmetricKeyGUID =
KEY_GUID
('SymmetricKeyEncryptedWithCert')
IF (@SymmetricKeyGUID IS NOT NULL)
BEGIN
INSERT INTO [dbo].[ TelephonNumber]
VALUES (01, ENCRYPTBYKEY
(@SymmetricKeyGUID,
N'8-861-123-87-63'))
INSERT INTO [dbo].[ TelephonNumber]
VALUES (02, ENCRYPTBYKEY
(@SymmetricKeyGUID,
N'8-868-765-87-65'))
INSERT INTO [dbo].[ TelephonNumber]
VALUES (03, ENCRYPTBYKEY
(@SymmetricKeyGUID,
N'8-861-234-11-11'))
END
```

Просмотр зашифрованных данных (Рис. 18):

```
USE [DB]
GO
SELECT * FROM
[dbo].[ TelephonNumber]
```

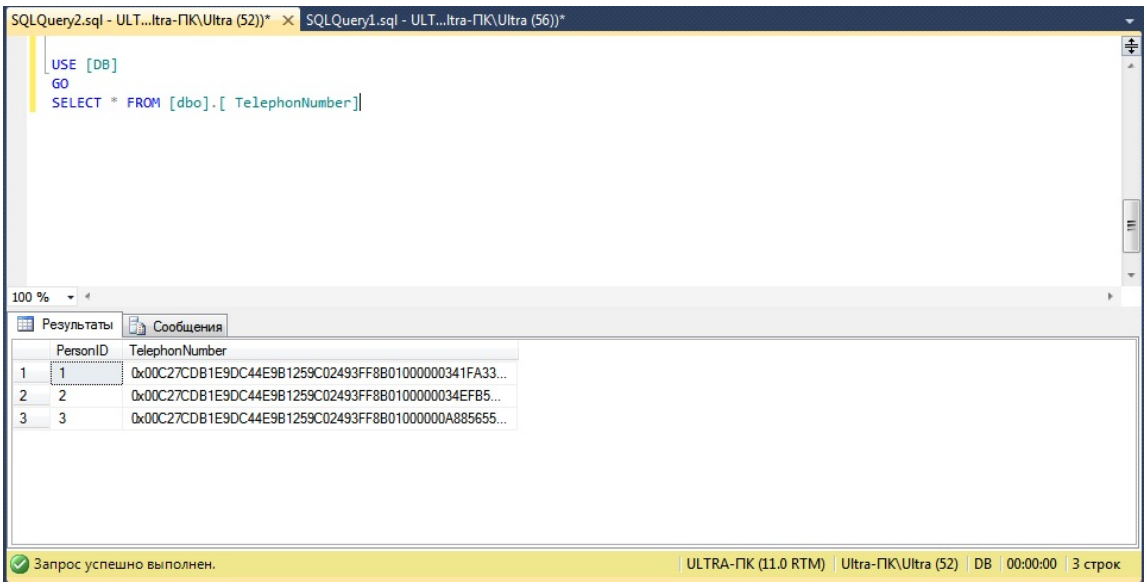


Рис. 18 – Вывод данных таблицы “TelephonNumber”

Расшифровка (Рис. 19):

```
USE [DB]
GO
SELECT [PersonID],
CONVERT([nvarchar](32), DECRYPTBYKEY(TelephonNumber))
AS [TelephonNumber]
FROM [dbo].[ TelephonNumber]
GO
```

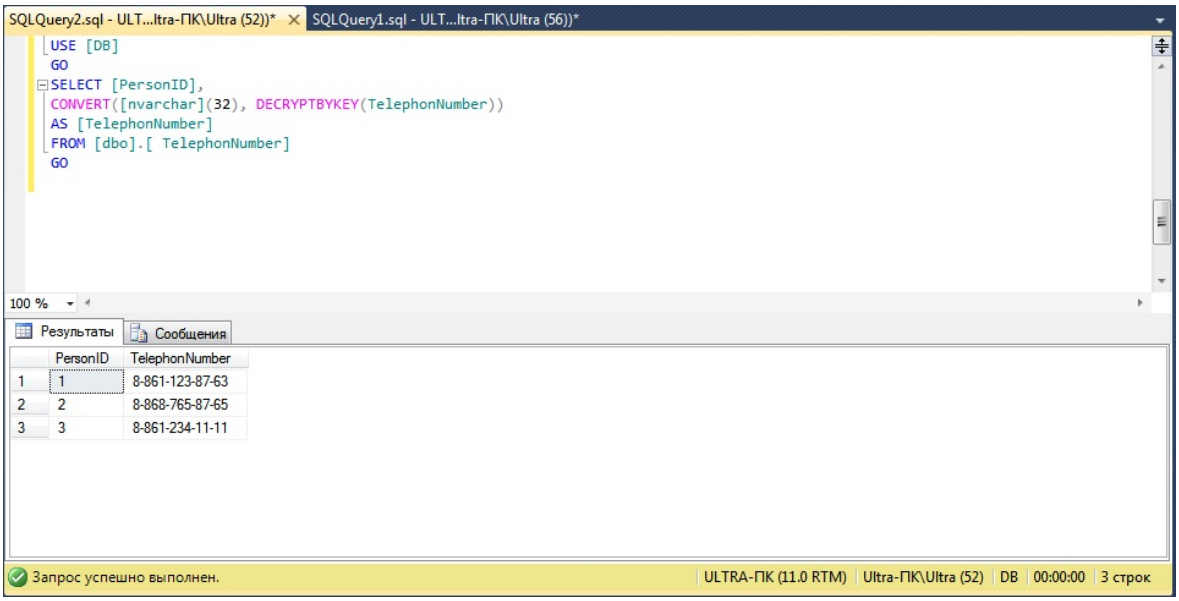


Рис. 19 – Дешифровка данных таблицы “TelephonNumber”

Рассмотрим пример прозрачного шифрования данных (Рис. 20)

1. Создаем главный ключ шифрования

```
CREATE MASTER KEY ENCRYPTION
```

BY PASSWORD = 'StrongPassword#1';

Созданный наш ключ можно увидеть в view:
select * from sys.key_encryptions

2. Чтобы удалить ключ введите следующий запрос:
drop master key

3. После того как создали главный ключ, необходимо сделать его резервную копию и поместить резервную копию в надежное место:

BACKUP MASTER KEY TO FILE = 'C:\Cert\MasterBackup.bak'
ENCRYPTION BY PASSWORD = 'Password1'

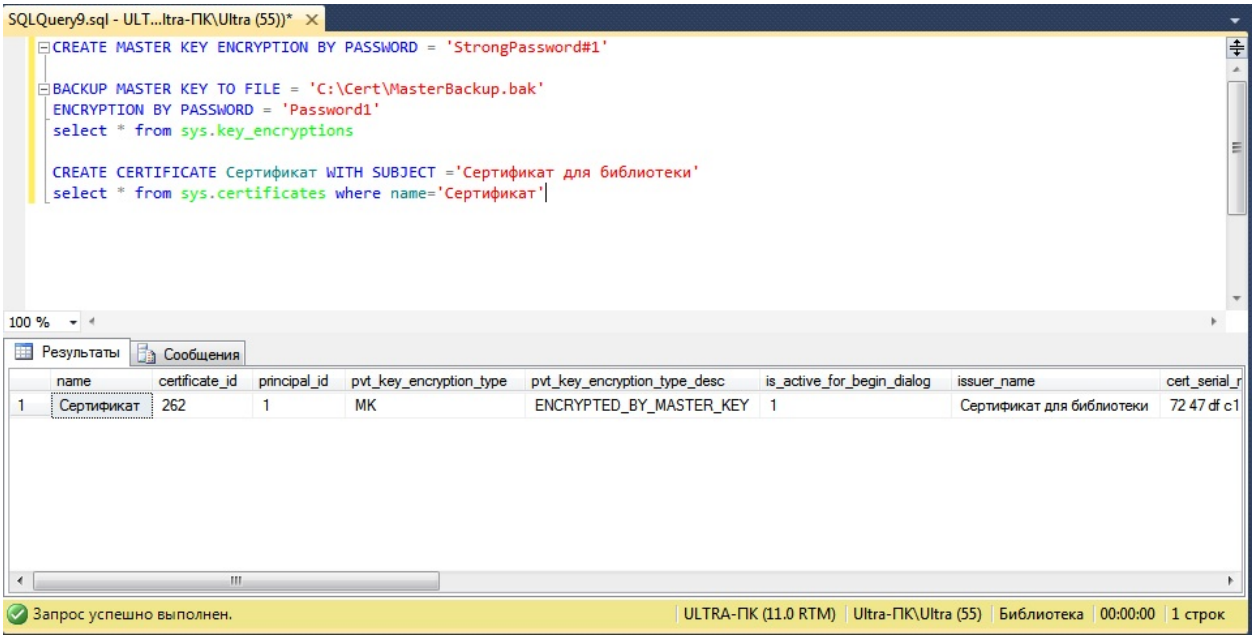


Рис. 20 – Создание главного ключа и сертификата

4. Создание сертификата осуществляется следующим запросом:

CREATE CERTIFICATE Сертификат
WITH SUBJECT ='Сертификат для библиотеки'

Проверка наличия созданного сертификата:
select * from sys.certificates where name='Сертификат'

5. Создание резервной копии сертификата с закрытым ключом (Рис. 21):

```
BACKUP CERTIFICATE Сертификат
TO FILE = 'C:\Cert\Certif'
WITH PRIVATE KEY
(FILE = 'C:\Cert\PrivateCertif',
 ENCRYPTION BY PASSWORD = 'Password#3');
```

Выполнив всё вышеизложенное приступим к созданию ключа шифрования в нашей базе данных с использование нашего сертификата:

```
USE Библиотека
go
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE Сертификат;
```

Включаем шифрование для нашей базы данных:

```
ALTER DATABASE Библиотека
SET ENCRYPTION ON;
```

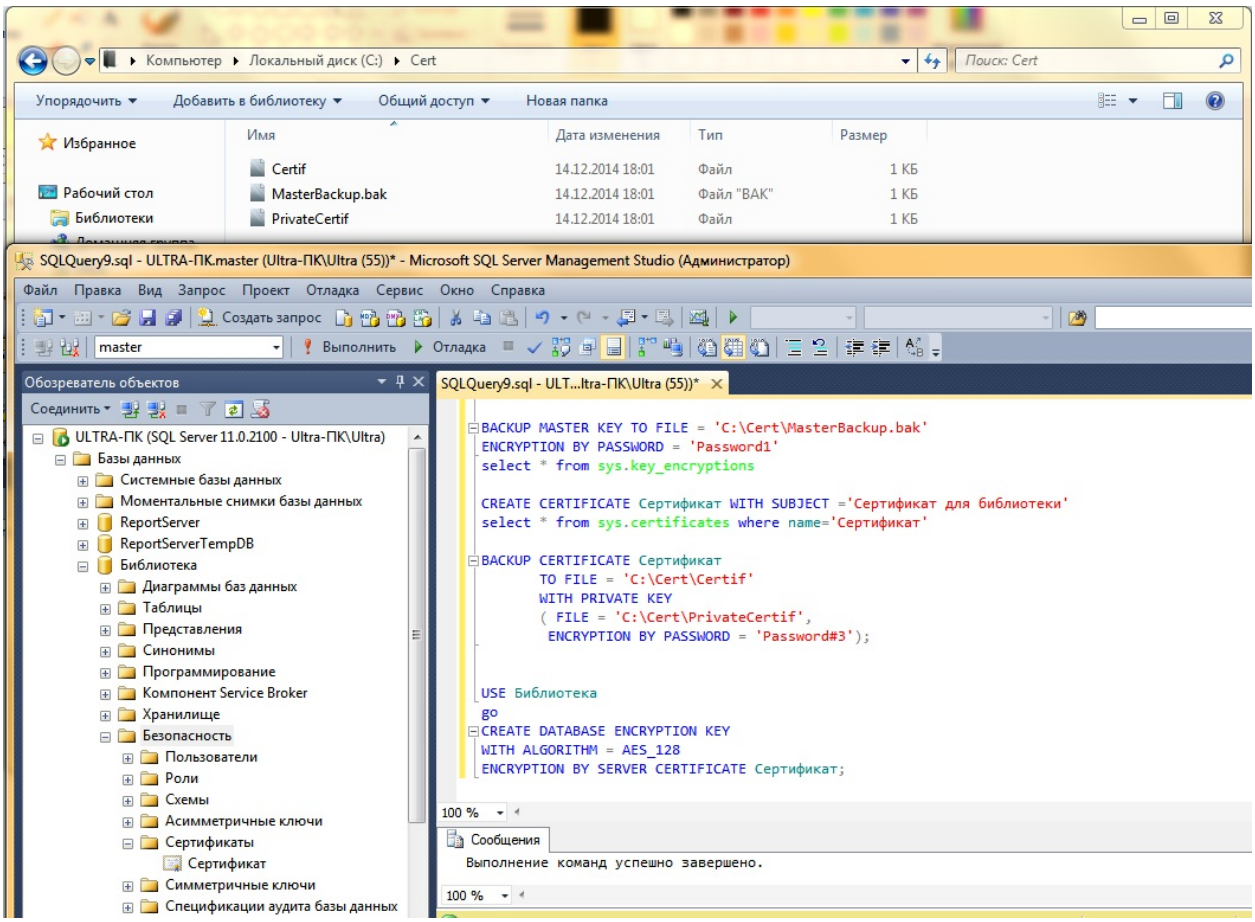


Рис. 21 – Резервные копии сертификата и главного ключа. Создание ключа шифрования в нашей базе данных

Чтобы посмотреть зашифрованные базы данных, нужно ввести следующий запрос (Рис. 22):

```
USE [master]
GO SELECT db.[name]
, db.[is_encrypted]
, dm.[encryption_state]
, dm.[percent_complete]
, dm.[key_algorithm]
, dm.[key_length]
FROM [sys].[databases] db
LEFT OUTER JOIN [sys].[dm_database_encryption_keys] dm
ON db.[database_id] = dm.[database_id];
 GO
```

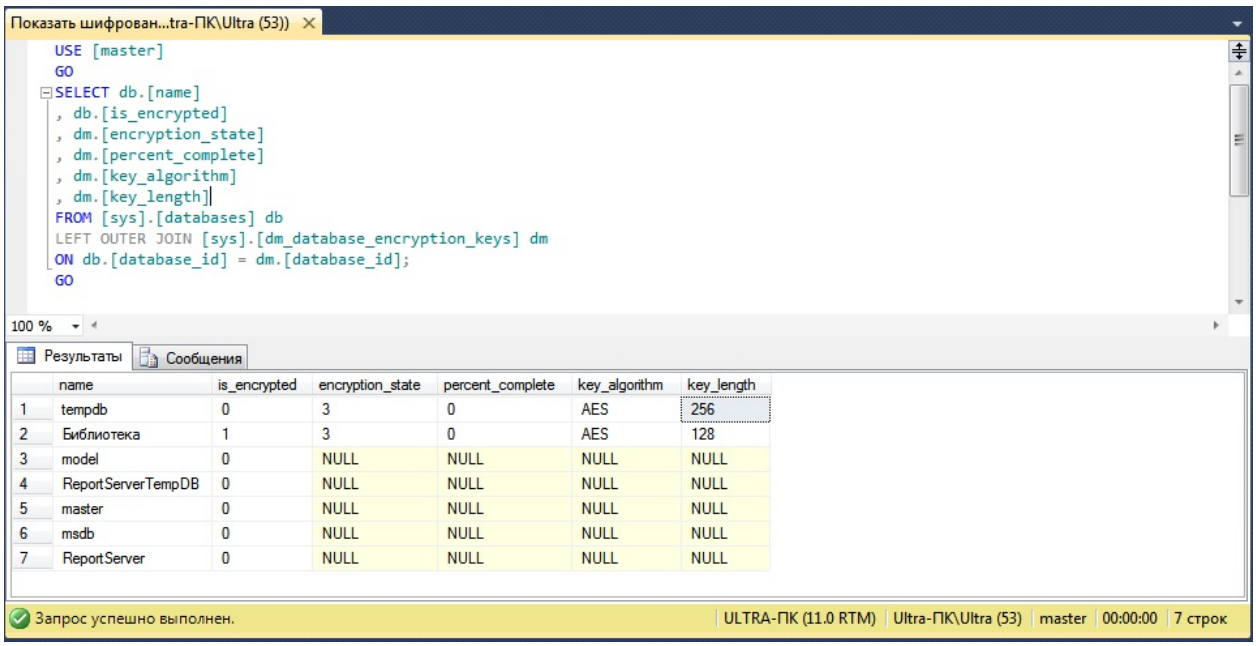


Рис. 22 – Вывод списка баз данных

Несколько моментов в работе с базой данных с включенным шифрованием:

Если создать резервную копию и попытаться восстановить на другом сервере, то получим ошибку:

Msg 33111, Level 16, State 3, Line 2
Cannot find server certificate with thumbprint '0x5B139FF1F2C5ED9EB3D503E78A63DEF3DD1FD96F'.
Msg 3013, Level 16, State 1, Line 2
RESTORE DATABASE is terminating abnormally.

Такую же ошибку получим и при попытки присоединения файлов базы данных. Порядок восстановления базы данных с прозрачным шифрованием на другом экземпляре MS SQL Server:
1. Создать мастер главный ключ шифрования на сервере MS SQL Server.

2. Восстановить из резервной копии сертификат с закрытым ключом:

CREATE CERTIFICATE Сертификат
FROM FILE = 'c:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\Certif'
WITH PRIVATE KEY (FILE = 'c:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\PrivateCertif',
DECRYPTION BY PASSWORD = 'Password#3');

3. Восстановить шифрованную базу данных или присоединить файлы базы данных с включенным шифрованием. БД готова для работы.

ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ТЕМ КУРСОВЫХ РАБОТ

ВАРИАНТЫ ЗАДАНИЙ

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
**	**	**	*	***	***	**	**	**	**	***	*	**	**	**	**	***	***

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	
**	**	**	***	***	**	**	**	***	**	***	**	**	*	*	***	**	

1. Коллекция видеофильмов (**)

Коллекция видеофильмов MPEG4 разбросана по компьютерам, соединенным в сеть. Написать базу, в. которой хранятся названия фильмов, жанр (комедия, боевик и др.), режиссер, основные артисты, краткая аннотация, сведения о наградах, год выпуска фильма, продолжительность и путь к файлу (файлам). База предназначена для поиска фильмов по названию, автору, режиссерам, наградам, жанрам, артистам, ключевому слову. Предусмотреть возможность просмотра выбранного фильма по сети. Клиентская программа может быть запущена с любого компьютера.

2. Электронный архив (**)

База предназначена для хранения систематизированной информации о документах электронного архива. Архив распределенный, располагается на нескольких компьютерах сети. Архив включает в себя текстовые документы, графические файлы (растровые и векторные), звуко- и видеозаписи. Документы располагаются по темам проектов. Информация о теме: шифр, наименование, руководитель, дата начала и окончания. Информация о документах: архивный номер, тема проекта, наименование документа, номер версии, дата помещения в архив, тип документа, отдел-разрабочик, ФИО исполнителя.

Клиентская часть может быть запущена с любого компьютера сети, и предназначена для поиска документа(-ов) по всем перечисленным признакам, просмотра и/или копирования выбранных документов на компьютер пользователя. Все обращения к документам регистрируются в базе данных.

3. Прокат автомобилей (**)

Предприятие занимается прокатом автомобилей гражданам и организациям. База предназначена для хранения сведений об автомобилях, клиентах и договорах проката. Пользователями базы являются владелец проката и клиенты. Владелец хранит информацию об имеющихся автомобилях, обращавшихся к нему клиентах и историю проката машин. В базе зарегистрированы имеющиеся автомобили: модель, цвет, номер, фотография автомобиля, год выпуска, номер двигателя, номер кузова, дата приобретения, дата прохождения последнего техосмотра, сведения о ремонтах и др. При оформлении договоров проката фиксируется: если клиентом является физическое лицо – Ф.И.О. клиента, паспортные данные, контактный телефон,

адрес; если клиентом является юридическое лицо – название организации, ИНН, контактный телефон, юридический адрес; а также: номер договора, дата заключения договора, период проката, номер автомобиля, стоимость услуг.

Клиенты с помощью базы могут выбрать понравившуюся модель, просмотрев характеристики и фотографии имеющихся автомашин, найти среди них те, которые в данный момент свободны и не находятся в ремонте, для остальных определить срок их возврата в пункт проката.

4. Должники (*)

Персональная база для регистрации своих и чужих долгов. В базе хранится информация о личных вещах, отданных в пользование знакомым, и чужих вещей, которые надо вернуть. Вещи группируются в базе по типам (книги, диски и т.д.). Для каждой вещи хранится информацию, кому и когда она была отдана (или от кого взята) и на какой срок.

При добавлении и удалении записей необходимо проверять корректность данных, чтобы не получилось так, что по ошибке одна вещь одновременно отдана двум разным людям.

5. Автовокзал (***)

Через автовокзал проходят пригородные и междугородние, в том числе транзитные маршруты. Автовокзал осуществляет продажу билетов на рейсы в ближайшие три дня.

Разработать базу данных автовокзала. База хранит расписание рейсов, сведения о проданных местах, и служит для оперативного выбора рейса и заказа билетов. Пользователями базы являются: администратор (редактирует расписание и маршруты, анализирует статистику рейсов); кассиры (продают билеты на рейсы на ближайшие три дня) и пассажиры (просматривают расписание, через справочную систему выбирают оптимальный рейс до нужного им населенного пункта, смотрят количество свободных мест).

В базе данных также хранится информация обо всех автобусах парка: марка, сведения о водителях, количество посадочных мест, пригоден ли автобус в данный момент к эксплуатации или находится на ремонте. Расписание составляется в двух вариантах: 1) расписание общее (пункт назначения, время и номер маршрута); 2) расписание работы каждого водителя на ближайшие три дня.

Предусмотреть статистическую обработку сведений о проданных билетах с целью определения наиболее и наименее востребованных рейсов по часам, дням недели и месяцам.

6. База данных микросхем (***)

Разработать базу для поиска микросхем и её поставщиков. В базе хранится информация об отечественных и импортных микросхемах, их аналогах, и предприятиях-изготовителях или фирмах-поставщиках этих микросхем. Каждая микросхема в зависимости от предприятия-изготовителя может выпускаться в разном конструктивном исполнении (отличаться корпусом, временными характеристиками, температурным режимом и т.д.). Для каждой микросхемы в базе

хранится: маркировка, краткая характеристика, тип корпуса, рисунок с цоколёвкой, перечень аналогов и список адресов предприятий-изготовителей или поставщиков данной микросхемы с ценами. Микросхемы группируются по типам (усилители, регистры. ПЛИС, процессоры и пр.).

Предусмотреть возможность просмотра всех микросхем заданного типа; поиск аналогов микросхемы; вывод всех предприятий, у которых можно приобрести указанную микросхему; для заданного набора микросхем определить предприятия, у которых наиболее выгодно их приобретать (критерием является не только стоимость, но и возможность приобретения как можно большего числа видов микросхем у одного предприятия).

7. Ботанический сад (**)

База предназначена для ознакомления граждан с коллекцией растений ботанического сада и заказа полюбившихся растений. В базе хранятся сведения о растениях по группам (плодовые, хвойные и декоративные деревья, плодовые и декоративные кустарники, овощи, цветы, вьющиеся растения и пр.). Для каждого растения имеется: название, краткое описание (когда цветет, где применяется и пр.), одна или несколько фотографий, для растений, которые можно приобрести, дополнительно указывается, как оно продается (семенами и/или саженцами), количество, цена и сроки пересылки.

В базе также регистрируются заказы граждан: ФИО получателя, почтовый адрес, электронный адрес, телефон, список заказанных растений с указанием количества и цены, форма оплаты (предоплата или наложенный платеж). Покупатели, приобретающие растения регулярно в течение трех лет, имеют скидку 20%.

Предусмотреть возможность поиска растения по названию, или всех растений заданной группы.

С целью предсказания спроса граждан на растения в будущем посевном сезоне, предусмотреть статистическую обработку заказов: список растений, пользующихся спросом, с указанием количества; наиболее активный регион РФ, из которого сделано больше всего заказов.

8. Прокат видеокассет, CD, DVD (**)

Разработать многопользовательскую базу для пункта проката. Пользователями базы являются операторы проката, регистрирующие выдачу и возврат кассет и дисков; и клиенты, производящие поиск интересующих фильмов и музыки.

Клиенты могут взять напрокат ресурсы двумя способами: 1) оставляя в залог документ или деньги, получают номерок, по которому регистрируется заказ; 2) оформляются по паспорту, получая карточку постоянного клиента, и далее заказ регистрируется на карточку. При прокате в базу заносится: номерок или номер карточки постоянного клиента, список взятых кассет и дисков, срок начала и окончания проката.

Клиентам предоставляется возможность поиска по базе интересующих их фильмов и музыки: название кассет и дисков, наличие на текущий момент (если отсутствуют, то предполагаемый срок возврата), инвентарные номера. Музыка

ищется по: исполнителю, автору, жанру, году издания альбома.. Видеофильмы ищутся по названию, жанру, фамилии режиссера, фамилиям актеров. Предусмотреть возможность поиска по шаблону и по типу носителя (кассета, CD, DVD). Для каждого фильма, кроме всего прочего, в базе хранится краткая аннотация.

С целью предсказания спроса предусмотреть возможность статистического анализа: какие жанры музыки и фильмов пользуются наибольшим спросом, вывести десять наиболее популярных фильмов и исполнителей музыки.

9. Супермаркет (**)

Разработать базу данных для супермаркета. Торговый зал магазина и склад товаров связаны локальной сетью. В торговом зале установлены кассовые аппараты на базе ПК. Каждый аппарат оснащен устройством чтения штрих-кодов. Цена товара выбирается по штрих-коду из базы, и может быть изменена в течение дня администратором базы данных. С помощью базы формируется кассовый чек. В магазине действует система скидок. Покупатели, имеющие карточку постоянного клиента, берут товары со скидкой *N*% (величину скидки определяет администратор базы).

По запросу оператора склада формируется отчет о товарах, количество которых в торговом зале близко к нулю, или они полностью раскуплены. Также формируется статистика раскупаемости товаров по дням недели и типам товаров.

10. Расписание тренировок спортклуба (**)

Разработать базу данных спортивного клуба, в которой будет хранится информация о тренерах, спортсменах, соревнованиях, наградах спортсменов, расписании тренировок и командировок. О тренерах в базе хранится: ФИО, квалификация, имеет ли право на судейскую деятельность; список спортсменов, которых он тренирует, контактный телефон и адрес. О спортсменах хранится: ФИО, разряд, дата рождения, телефон, адрес, список наград, расписание тренировок и участия в соревнованиях, и пр.

База используется:

- для составления личного расписания тренеров (тренировки, судейство, выезд в командировки со своими подопечными);
- для составления личного расписания спортсменов;
- для вывода биографии и послужного списка спортсмена;
- для поиска судей, которые находятся на момент проведения соревнований в городе, и в соревнованиях не участвуют их подопечные.

11. Больница (***)

Разработать базу данных для учета больных, находящихся на излечении. Больница состоит из нескольких отделений. Каждое отделение включает в себя: больничные палаты, процедурные кабинеты и кабинеты врачей. Также имеется ряд общебольничных процедурных кабинетов, не входящих в состав ни одного из отделений. Поступающие больные регистрируются (паспортные данные, номер мед. полиса) и размещаются в отделение, соответствующее основному типу заболевания.

К каждому больному прикрепляется лечащий врач, и, если есть заболевания иного профиля, врачи-консультанты из других отделений.

В базе данных должны учитываться сведения о поступлении, переводах и выписках больных; истории болезни, назначенные процедуры, лекарства и диеты. Пользователями базы данных являются: приемное отделение (регистрация, переводы, выписка больных, поиск больных по фамилии), столовая (ежедневно получает заказ на приготовление диет.питания), врачи (ведение истории болезни, назначение лекарств и диет).

12. Электронная библиотека (*)

На сервере хранятся файлы с документацией различного характера. Документация группируется по категориям. Для каждого документа хранится название, авторы, язык, краткое содержание, список ключевых слов. Разработать базу данных для поиска документации, клиентская часть базы может быть запущена с любого компьютера локальной сети. Поиск может вестись по автору, названию, категории, ключевым словам, дате создания документа. По запросу выбранный файл копируется на машину пользователя. Предусмотреть возможность просмотра списка из 20 самых популярных документов.

13. База данных УВД (**)

Разработать базу данных УВД, в которой хранится информация о гражданах, состоящих на учете в милиции и история их преступлений. Для каждого человека хранится: ФИО, дата рождения, краткая биография, список псевдонимов, фотографии по годам, отпечатки пальцев, информация о судимостях, номера уголовных дел, ссылки на электронные документы. Доступ к некоторым документам может ограничиваться паролем. Также в базе хранится список правонарушений, совершенных в контролируемом районе.

Предусмотреть возможность поиска людей по имени, псевдониму, номерам уголовных дел и статьям УК РФ.

Выходные документы:

- список не раскрытых дел;
- список раскрытых дел;
- информация о лицах, проходящих по делу №*X*.

14. Банк-магазин (**)

Банк и магазины связаны договором, согласно которому клиенты банка могут расплачиваться за покупки с помощью кредитной карты. Каждый такой магазин имеет в банке счет. Деньги, снятые с кредитной карточки клиента, автоматически переводятся на счет магазина. Гражданин может открыть в банке несколько счетов "до востребования" и получить кредитную карту. Когда карта ему выдается, то он отмечает, с каких счетов будут сниматься деньги по этой карте. Если в процессе оплаты деньги с одного счета сняты полностью, далее они снимаются со следующего. Когда все счета исчерпаны, покупка в магазине по карте невозможна.

По требованию клиента (например, в случае потери кредитной карты), расплата кредитной картой блокируется.

Разработать базу данных для проведения банковских операций начисления денег на счет, снятия денег со счета, выдачи и блокировки кредитной карты, оплаты в магазинах кредитной картой.

Выходные документы:

- ежедневная распечатка доходов для каждого магазина: время, сумма, номер кредитной карты;
- распечатка для клиента: в каких магазинах, сколько он потратил, в какое время, с какого счета сняты средства;
- для конкретного счета: распечатка о дате, времени и сумме начислений и расходов;
- список помесячных доходов магазинов.

15. Товары – почтой (**)

База данных установлена в фирме, которая производит поставку товаров почтой в разные регионы страны. В фирме работает несколько сотрудников. Каждый сотрудник курирует несколько заказов. Заказ включает в себя перечень товаров с указанием их количества. Заказы могут оплачиваться наложенным платежом после получения заказа, или предварительно банковским переводом. Если в течение 14 дней со дня регистрации заказа банковский перевод не пришел, заказ считается аннулированным. Сумма пересылки рассчитывается в процентах от стоимости заказа, и зависит от региона и формы оплаты.

База сетевая. Сотрудники вносят в базу новые заказы; отмечают, когда пришел денежный перевод, или заказчик внес наложенный платеж, и отмечают, если посылка отправлена, и заказ выполнен. Сотрудники не имеют права просматривать и редактировать заказы, которые обслуживаются их коллеги. При регистрации заказов в базу вносятся следующие сведения о заказчике: ФИО или название организации, почтовый адрес и телефон, наличие e-mail. Заказчик, который приобрел товаров на сумму более 5 тысяч руб., переходит в разряд постоянных клиентов и имеет 10%-ную скидку от стоимости.

Заказы наложенным платежом не принимаются от заказчиков, которые трижды аннулировали свой заказ.

16. Каталог статей в периодических изданиях (**)

Электронный каталог предназначен для учета и поиска статей в периодических изданиях, которые выписывает библиотека. Учет проводится по нескольким категориям областей знаний. Одна статья может относиться к нескольким категориям. Для каждой статьи хранятся выходные данные (автор, название, название журнала/газеты, год, номер, начальная страница), список ключевых слов и, по возможности, ее краткое содержание.

Поиск статей производят посетители библиотеки. Поиск может вестись по категориям, автору, названию и по ключевым словам. Поисковая система в результате формирует список статей с полными выходными данными и кратким содержанием. Второй тип поиска: вывести список журналов и газет, которые выписывала библиотека в заданный период времени. В списке указываются номера и годы выпуска. Третий тип поиска: для выбранного журнала и газеты (название может быть задано по маске) вывести список номеров, имеющихся в библиотеке.

17. Хлебозавод (***)

Хлебозавод включает в себя: склад сырья и три цеха по выпечке хлебобулочных изделий: хлеб и батоны, сдобная продукция, торты и пирожные. У каждого цеха свой ассортимент. На каждое изделие установлен план выпуска (штук в день) и рецептура, в которой указан расход ингредиентов в килограммах на одно изделие. Ингредиенты хранятся на складе. Хлебозавод принимает заказы на производство нестандартной продукции по рецепту заказчика.

В начале рабочего дня планируется выпуск изделий по цехам. Для каждого цеха определяется перечень изделий и план выпуска, согласно которому на склад направляется план отгрузки сырья по цехам. Если оставшегося на складе сырья недостаточно для бесперебойной работы завода в течение трех дней (с текущим планом загрузки цехов), то на склад направляется требование на закупку сырья с указанием количества, достаточного на 10 дней работы завода.

База данных предназначена для координации работы цехов и отдела поставок сырья. В соответствии с планом выпуска база формирует выходные документы:

- для каждого цеха по выпечке план выпуска изделий. Если изделие по рецепту заказчика, то к плану прикладывается рецептура;
- для склада – план отгрузки по цехам;
- для склада – план закупок сырья.

18. Театральная касса (***)

Организация занимается распространением билетов на представления в цирке, театрах и концертных залах города. Стоимость билетов зависит от «посадочной зоны» конкретного зала. Продажа билетов производится в нескольких кассах (филиалах). Кассы и офис связаны сетью.

Разработать базу данных для продажи билетов. Поисковая система базы определяет наличие свободных мест на представления, и исключает места согласно купленным билетам. Офис может получить статистику раскупаемости билетов через филиалы.

19. Проектная организация (**)

В проектной организации работают инженеры-конструкторы, инженеры-схемотехники и инженеры-программисты. Руководители проектов планируют нагрузку работников с использованием базы данных. Руководитель может одновременно вести несколько проектов. Инженеры-исполнители в каждый момент времени участвуют не более чем в одном проекте. При открытии проекта руководитель планирует квалификацию и число исполнителей и определяет каждому работнику сроки начала и окончания работ. Для выполнения работ руководитель может привлекать инженеров, которые на заданный период времени не участвуют в других проектах.

Выходные документы:

- каждый инженер может просмотреть график своего участия в различных проектах;

- руководитель получает план работ с указанием исполнителей и сроков их участия в проекте;

- руководитель может просмотреть все свои проекты.

- поиск инженеров, не участвующих ни в одном проекте в указанный промежуток времени.

20. Расписание занятий (**)

База данных учебного отдела содержит расписание занятий по аудиториям: время, группа, дисциплина, аудитория, преподаватель. Для каждой аудитории хранится число посадочных мест, назначение (лекционный зал, практические занятия, лаборатория, дисплейный класс, наличие доски –признаки могут комбинироваться). В расписании учитываются верхние и нижние недели.

База дополнена поисковой системой. Она применяется для поиска свободных аудиторий. Запрашивается тип аудитории, количество человек, день недели и номер пары.

Выходные документы:

- для группы вывести ее расписание на неделю;

- для преподавателя вывести его личное расписание;

- для указанной аудитории вывести перечень групп, которые там занимаются.

21. Отдел кадров (**)

Разработать базу данных для отдела кадров. О сотрудниках хранится следующая информация: ФИО, фотография, дата рождения, паспортные данные, семейное положение, сведения о детях, образование (среднее, средне-специальное, высшее) профессия (по диплому), номер и дата выдачи диплома; "история" занимаемых на этом предприятии должностей: должность, ставка, номер приказа, дата поступления и окончания; сведения о наградах.

Выходные документы:

- список руководящих работников предприятия с указанием отдела и занимаемой должности;

- список всех сотрудников указанного отдела с должностями и ставками;

- список всех сотрудников пенсионного возраста (55 лет для женщин и 60 для мужчин);

- список сотрудников, имеющих детей до 18 лет;

- для указанного сотрудника вывести послужной список и перечень наград.

22. Цех (***)

Для сборки изделий на заводе применяется ряд унифицированных деталей. Детали изготавливаются в механическом цехе, затем направляются в цех лакокрасочных и гальванических покрытий, а после покрытия поступают на сборочный конвейер. Диспетчер производства цеха покрытий каждый день должен запускать в покрытие нужное количество каждой детали, чтобы сборочный конвейер не простаивал. План выпуска изделия каждого типа устанавливается ежедневно.

Разработать базу данных, связывающую механический цех, цех покрытий и сборочный цех. На основании плана выпуска изделий сформировать отчеты:

1) для механического цеха – план выпуска деталей;

2) для цеха покрытий – перечень деталей с указанием типа покрытия (хромирование, никелирование, лак, краска) и количества;

3) для сборочного цеха – план сборки. В плане сборки указывается число изделий, для каждого типа изделий – перечень деталей.

В базе должен хранится план выпуска изделий за последние 30 дней.

23. Компьютерная фирма (***)

Компьютерная фирма состоит из нескольких филиалов. Каждый филиал оснащен складом, на котором хранятся комплектующие и собранные их ПК. Имеется несколько "стандартных" моделей ПК и возможна сборка под заказ. Филиалы связаны сетью. База данных предназначена для учета товаров на складах, выбора покупателями комплектующих и оборудования. Для каждого вида комплектующих хранится его характеристика, цена и размещение по складам филиалов. Через базу идет учет покупок и продаж комплектующих и готовых моделей.

Выходные документы:

- прайс-листы по типам комплектующих;

- состав "стандартных" моделей ПК;

- для указанного вида товара список, на каком складе сколько единиц товара находится;

- пользователь определяет состав устройств ПК и предел цены: вывести все возможные варианты сборки ПК из комплектующих, имеющихся на складах.

Предусмотреть возможность просмотра статистики комплектующих и моделей. пользующихся повышенным спросом.

24. Результаты сессии (**)

В деканате учитываются результаты сессий за все семестры. Для каждой специальности хранится учебный план (список дисциплин по семестрам, количество часов, формы контроля: экзамен, зачет, курсовой проект/работа). Учебные планы отличаются на 4-5 курсах дисциплинами специализации.

В базу заносятся все факты пересдачи по допускам. Итоговой считается последняя оценка. Она идет в выходные документы: выписку из зачетной ведомости; список студентов и дисциплин, которые им необходимо досдать к началу текущей сессии (в том числе имеющие неудовлетворительные оценки); список студентов, окончивших текущую сессию на "4" и "5" – для начисления стипендии, и только на "5" – для начисления повышенной стипендии; список студентов 5 курса (проучившихся 9 семестров) – кандидатов на красный диплом (оценки только "4" и "5", оценок "4" не более 25%).

25. Междугороднее сообщение (**)

База данных представляет собой справочную систему, используемую для выбора оптимального способа добраться до заданного пункта назначения. База хранит информация обо всех железнодорожных, автобусных и авиационных рейсах, которыми можно добраться из нашего города в другие населенные пункты. Для каждого рейса хранится: № рейса, вид транспорта, пункт назначения, время отбытия, время прибытия. Поиск может вестись по нескольким признакам (по отдельности или смешанно): пункт назначения, вид транспорта, время отбытия, время прибытия, день недели (дата).

База многопользовательская. Пользователями являются операторы, осуществляющие ввод и редактирование данных, и граждане, только просматривающие результаты поиска.

26. Биржа труда (**)

База данных биржи труда предназначена для регистрации заявок с предприятий и анкет граждан, ставших на учёт. Пользователями базы являются работники биржи и безработные граждане. Работники биржи регистрируют заявки от предприятий и граждан, результаты устройства на работу (удачное или неудачное). Граждане используют базу для поиска подходящих вакансий и просмотра статистики востребованных профессий.

В заявке от предприятия указывается: наименование предприятия, требуемая квалификация работника, заработная плата и срок, в течение которого будут

рассматриваться кандидаты. Заявка удаляется, если истёк срок ее действия, или предприятие сообщило о приеме на работу.

Анкета работника, ставшего на учёт, включает: ФИО, образование (неполное среднее, среднее, среднее специальное, высшее) и список профессий, которыми он владеет (список всевозможных профессий стандартизован). Гражданин, поступивший на работу, снимается с учёта. Также снимаются с учёта граждане, которым по одной и той же профессии предлагались вакансии три раза, но эти предложения их не устроили; впоследствии такие граждане имеют право стать на учет не ранее чем через три месяца по той же профессии, или немедленно по другой профессии.

База данных должна автоматически формировать направления для устройства на работу и список вакансий с заработной платой не менее указанного минимума (перечень профессий задается).

27. Учёт коммунальных платежей (***)

Разработать базу данных для учета коммунальных платежей граждан и расчета пени. В базе хранятся адреса, информация о квартиросъемщиках и членах семьи, прописанных на этой площади (ФИО, паспортные данные), информация о льготах квартиросъемщиков, сумма помесячной оплаты (начисленная и фактически уплаченная), пени. База применяется для выписки счетов-квитанций на оплату. В счет-квитанцию входят следующие статьи:

- ремонт и содержание жилья (= общая площадь × стоимость по тарифу за $м^2$);
- отопление (= общая площадь × стоимость по тарифу за $м^2$; в частных домах и квартирах, оснащенных водогрейными котлами, не начисляется);
- горячая вода (= число проживающих × стоимость по тарифу на 1 чел.; в частных домах и квартирах, оснащенных водогрейными котлами, не начисляется);
- холодная вода (= число проживающих × стоимость по тарифу на 1 чел.);
- газ (= число проживающих × стоимость по тарифу на 1 чел.; для частного сектора и квартир с водогрейными котлами дополнительно прибавляется: общая площадь × стоимость по тарифу за $м^2$);
- прочие отчисления (вывоз мусора, антенна, лифт) – ставка фиксированная в расчете на 1чел., эти статьи начисляются в зависимости от места проживания.

Расчет платежей на проживающих в квартире и имеющих льготы, производится со скидкой 50%, на остальных – 100%-ные начисления.

Оплата за месяц должна производиться не позднее 10 числа следующего месяца включительно. За каждый просроченный день начисляется пени в размере 0,5% от суммы оплаты.

Предусмотреть автоматизированное составление следующих выходных документов:

1. счета-квитанции по квартирам;
2. список квартир, имеющих задолженности:
   2.1. с суммой долга, превышающей заданный предел, руб.;
   2.2. с задолженностью по оплате 3 и более месяцев;

2.3. всех задолжников.

28. Библиотека (**)

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.

29. Зоопарк (***)

База данных предназначена для координации работы отдела поставок, склада продуктов и кухни, а также для хранения сведений о животных, содержащихся в зоопарке и историй болезни. Соответственно пользоваться базой будут четыре отдела: отдел поставок, склад продуктов, кухня, ветеринарный отдел. Состав продуктов, входящих в рацион конкретного животного, определяет ветеринарный отдел в зависимости от вида животного, возраста и состояния здоровья. В рацион включаются овощи, фрукты, каши, мясные и молочные продукты. Все продукты хранятся на складе. Нормативный срок использования мясных и молочных продуктов – 1-3 дня, овощей – 10-30 дней, фруктов – 5-20 дней, круп – 120 дней. Ежедневно в отдел поставок направляется заказ на закупку продуктов, которые скоро закончатся, и заказ на лекарства. База данных предназначена для автоматизированного составления запросов в отдел поставок и выписки рецептур для кухни на приготовление рационов животных. Выходные документы:

- перечень продуктов и лекарств, требующих закупки (по категориям);
- перечень продуктов для выдачи со склада на кухню, с указанием количества;
- среднегодовой расход продуктов по категориям.

30. Автопарк (**)

Автотранспортное хозяйство осуществляет перевозки людей и грузов на заказ. Имеющийся транспорт: легковые автомобили, грузовики и автобусы. Каждый из водителей (категорий C,D,E) закреплен за конкретным транспортным средством, но может замещать отсутствующего коллегу в случае, если собственный транспорт находится на ремонте. Все перевозки регистрируются в базе данных: заказчик, время и дата заказа (с точностью до половины дня), тип перевозок, пункт назначения, транспортное средство, водитель. Сроки ремонтов транспорта также протоколируются. Выходные документы:

- полная информация о заказах, выполненных в заданный промежуток времени;
- "история" перевозок транспортного средства в заданный промежуток времени: куда ездило, кто был водитель, кто заказчик, и пр.;
- список транспортных средств определенного типа (например, автобусов с числом посадочных мест не менее 20), свободных на заданный интервал времени;
- списки водителей с указанием категории, которые на текущий день остались без работы по причине ремонта их транспорта.

31. Издательство (**)

База данных издательства хранит информацию о статьях, опубликованных журналистами в журнале, персональных сведениях о журналистах, а также реальное содержимое журнала. О журналистах хранится: ФИО, контактный телефон, адрес постоянного проживания, образование, стаж, штатный/нештатный корреспондент. О статьях: название; год, номер и страницы журнала; число знаков текста, авторы (может быть несколько), жанр (аналитический обзор, интервью, исторический очерк, новости и т.д.), рейтинг (числовая мера качества статьи). Файл с текстом журнала (в формате издательской системы) также помещается в базу данных. База данных применяется для поиска статей по автору и жанру; содержащиеся в ней сведения также могут использоваться поиска областей, в которых наиболее полно раскроется творческий потенциал работника. Выходные документы:

- для заданного журналиста вывести перечень его статей в журналах;

- для каждого работника определить три жанра, статьи в которых обладают наивысшим рейтингом;

- перечень журналистов в порядке убывания их среднего рейтинга.

32. Склад товаров (*)

Однопользовательская база данных оптового склада для учета прихода/расхода товаров. В базе хранятся приходные ордерá (номер ордера, дата, список товаров с указанием упаковки, количества, закупочной цены, адрес и название фирмы-поставщика) и расходные ордера (содержание аналогично). База должна автоматически формировать следующие документы:

- прайс-лист с указанием цены товара и минимального объема закупок;
- расходный ордер;
- месячная прибыль по каждому типу товаров.

Название фирм-поставщиков и отображать на экране по убыванию суммы закупленного у нее товара.

33. Спрос и предложение (*)

Составить базу данных для учета спроса и предложения товаров и услуг, и помощи в поиске партнеров. В базу заносится информация о гражданах, предприятиях и организациях: ФИО (название организации), контактная информация (телефон и/или адрес), вид услуг, спрос или предложение, срок, прочая информация (минимальный объем, цена, условия реализации и т.п.).

Выходные документы:

- для заданного типа спроса вывести все имеющиеся предложения, действительные на текущий момент;
- для заданного типа предложения вывести все заявки на спрос;
- найти и вывести пары: спрос-предложение.

34. Экологические технологии (***)

База данных предназначена для поиска технологий обезвреживания и использования отходов, вырабатываемых промышленными предприятиями страны. В базе хранится информация о предприятиях: название, адрес, контактные телефоны, ФИО директора или иных руководителей; вырабатываемой продукции, побочных продуктах производства, полуфабрикатах, отходах производства, исходном сырье и ингредиентах (с указанием количества). Всё вышеперечисленное определяется применяемой технологией. Список технологий, применяемых на каждом предприятии, также хранится в базе данных.

База предназначена для выявления таких сырьевых связей между предприятиями, в которых побочные продукты и отходы производства одного предприятия могут служить сырьем для другого предприятия. Желательно, чтобы объемы производства и потребления были соизмеримы.

Выходные документы:

- для заданной технологии вывести перечень предприятий, на которых данная технология применяется (выводить адреса предприятий и прочую контактную информацию);
- для заданного ингредиента вывести список предприятий, на которых он используется как сырье, и на которых он является продуктом/отходом производства (с указанием годового объема);

- вывести список предприятий, которые могут быть вовлечены в цепочку производитель-потребитель, с указанием вида продукта.

35. База данных ГИБДД (**)

В базе данных хранится информация об автомобилях и водительских удостоверениях граждан. Для автомобиля хранится: модель, цвет, текущие номера, дата постановки на учет, дата снятия с учета, номера двигателя и кузова, владелец, срок последнего техосмотра. Также имеется список автомобилей, находящихся в розыске. При постановке и снятии с учета проверяется, чтобы автомобиль не был в розыске.

При выдаче водительского удостоверения в базу заносится: номер удостоверения, дата выдачи, категория, ФИО, дата рождения, паспортные данные, прописка, фотография, вождение в очках. Также в базе хранится информация обо всех дорожных авариях, вовлеченных в них автомобилях и наложенных штрафах.

Выходные документы:

- для указанного владельца машины вывести все дорожные происшествия, произошедшие с его машинами (машиной) за последние N месяцев;
- список автомобилей с просроченной датой техосмотра.

Перечень литературы

1. Технологии обеспечения безопасности информационных систем : учебное пособие / А. Л. Марухленко, Л. О. Марухленко, М. А. Ефремов [и др.]. – Москва ; Берлин : Директ-Медиа, 2021. – 210 с. – URL: https://biblioclub.ru/index.php?page=book&id=598988 (дата обращения: 22.05.2023). – Режим доступа: по подписке. – Текст : электронный.

2. Марухленко, А. Л. Разработка защищённых интерфейсов Web-приложений : учебное пособие / А. Л. Марухленко, Л. О. Марухленко, М. А. Ефремов. – Москва ; Берлин : Директ-Медиа, 2021. – 175 с. – URL: https://biblioclub.ru/index.php?page=book&id=599050 (дата обращения: 22.05.2023). – Режим доступа: по подписке. – Текст : электронный.

3. Кобылянский, В. Г. Операционные системы, среды и оболочки : учебное пособие / В. Г. Кобылянский. – Новосибирск : Новосибирский государственный технический университет, 2018. – 80 с. – URL: https://biblioclub.ru/index.php?page=book&id=576354 (дата обращения: 22.05.2023). – Режим доступа: по подписке. – Текст : электронный.

4. Основы администрирования информационных систем : учебное пособие / Д. О. Бобынцев, А. Л. Марухленко, Л. О. Марухленко [и др.]. – Москва ; Берлин : Директ-Медиа, 2021. – 202 с. – URL: https://biblioclub.ru/index.php?page=book&id=598955 (дата обращения: 22.05.2023). – Режим доступа: по подписке. – Текст : электронный.

5. Ищейнов, В. Я. Информационная безопасность и защита информации : теория и практика : учебное пособие / В. Я. Ищейнов. – Москва ; Берлин : Директ-Медиа, 2020. – 271 с. – URL: https://biblioclub.ru/index.php?page=book&id=571485 (дата обращения: 22.05.2023). – Режим доступа: по подписке. – Текст : электронный.