МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ЧЕБОКСАРСКИЙ ИНСТИТУТ (ФИЛИАЛ) МОСКОВСКОГО ПОЛИТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Кафедра Информационных технологий, электроэнергетики и систем управления

УТВЕРЖДАЮ
Директор филиала
_____ А.В. Агафонов
«29» марта 2023 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению Курсовой Работы по
Базам Данных

Направление подготовки	**09.03.01 Информатика и вычислительная техника**
	(код и наименование направления подготовки)
Направленность подготовки	**Программное обеспечение вычислительной техники и автоматизированных систем**
	(наименование профиля подготовки)
Квалификация выпускника	**Бакалавр**
Форма обучения	**очная и заочная**

Чебоксары, 2023

Методические указания разработаны в соответствии с требованиями ФГОС ВОпо направлению подготовки **09.03.01 Информатика и вычислительная техника**

Автор Скипина Людмила Николаевна, кандидат технических наук, доцент кафедры информационных технологий и систем управления

(указать ФИО, ученую степень, ученое звание или должность)

Методические рекомендации одобрены на заседании кафедры Информационных технологий и систем управления (протокол № 6 от 04.03.2023 г.).

1 Общие сведения о базах данных

1.1 Модели данных

В основе структуры баз данных (БД) лежит понятие модели данных.

Определение. Под ***моделью данных*** будем понимать набор средств, описывающих типы данных, отношения между данными, семантику данных и ограничения на данные.

Применительно к БД все модели можно условно разбить на 3 группы:

- физические модели данных;
- логические модели, основанные на понятии записи;
- логические модели, основанные на понятии объекта.

Физические модели данных.

Физические модели являются низкоуровневым представлением данных и определяют способы их размещения, методы доступа к данным, технику индексирования. Физическая модель описывает данные средствами конкретной системы управления базами данных (СУБД).

Модели данных, основанные на понятии записи.

Эти модели описывают данные на концептуальном уровне и уровне представления и определяют не только архитектуру БД, но и дают общее описание ее реализации. Основными типами моделей этого класса являются:

- Реляционная;
- Сетевая;
- Иерархическая.

Модели данных, основанные на понятии объекта.

Модели этого типа применяются для описания данных на концептуальном уровне и уровне представления пользователя. Они позволяют определять структуру БД и ограничения на целостность данных. Существует много видов моделей этого класса, наиболее часто используемые из них - это:

- модели «сущность-отношение»;
- объектно-ориентированные модели (ООМ).

Модель «сущность-отношение»

Базовыми понятиями в этом типе моделей являются понятия сущности и отношения.

Определение. Сущностью называется объект, характеризующийся (отличающийся от других сущностей) набором специфических признаков, называемых атрибутами.

Пример. Признаки (атрибуты) номер счета и остаток по счету характеризуют (определяют/отличают) сущность – банковский счет.

Определение. ***Отношением*** будем называть ассоциацию (связь) между различными сущностями.

Пример. Отношение КлиентСчет (CustAcc) связывает клиента банка и счет, которым он обладает.

Определение. Множество всех сущностей и отношений одного типа называется множеством сущностей и множеством отношений соответственно.

Одним из важных ограничений на данные в этой модели является мощность отображения, которая определяется количеством сущностей, связанных с другой сущностью посредством множества отношений. При этом можно с помощью мощности указать, сколько экземпляров родительской сущности связано с сущностью-потомком.

Пример.

Сущность «клиент» связана с сущностью «счет» отношением «КлиентСчет». Его мощность – «один ко многим».

Модель «сущность-отношение» и, тем самым, логическая структура БД может быть представлена графически с помощью *E-R диаграммы*. Основными элементами диаграммы являются:

- прямоугольники, представляющие множество сущностей;
- эллипсы, представляющие атрибуты;
- ромбы, представляющие отношения между множествами сущностей;
- линии, которые связывают сущность с ее атрибутами и множество сущностей с отношениями.

Мощность отношения тоже может быть выражена графически посредством указания точки на конце линии, относящейся к сущности-потомку, и явного написания мощности отношения.

Внутри каждой из этих фигур записывается имя, соответствующее сущности, атрибуту или отношению. Эти диаграммы еще называют *диаграммами Чена*, по имени их автора.

Объектно-ориентированная модель.

Так же как и *E-R- модель* объектно-ориентированная модель основана на наборе объектов. Объектно-ориентированный подход к проектированию БД предлагает широкий набор типов. Существуют базовые типы данных: простые числа, действительные числа, булевы выражения, строки символов. Кроме того, с помощью *конструктора типов* можно из базовых типов построить новые типы:

- структуры записей (структуры);
- типы множеств (массивы, списки и множества);
- типы ссылок (указатель).

Объект содержит значения данного типа (*неизменяемые объекты*) или переменные, значения которых относятся к этому же типу (*изменяемые объекты*), а также функции или процедуры, оперирующие с объектами (*методы*).

Объекты, содержащие одинаковые типы значений и, может быть, одинаковые методы, группируются в *классы*.

Метод для класса C должен иметь по крайней мере один аргумент, являющийся объектом класса C, и может иметь другие аргументы любого класса.

Пример.

Рассмотрим класс «Множество целых чисел». Для него можно определить метод объединения двух множеств или метод возвращения логического значения, показывающего, является ли данное множество пустым.

В объектно-ориентированной модели предполагается, что все объекты имеют значение, называемое *идентичностью объекта* (OID). Никакие два объекта не могут иметь одно и то же OID, и ни один объект не может иметь два различных OID. Т.о., OID – это значение, которое имеет ссылка на конкретный объект.

Преимущества объектно-ориентированного подхода к проектированию БД.

1. С помощью богатой системы типов можно оперировать данными в формах более естественных, нежели отношения в других моделях данных.
2. Совместное или повторное использование прикладных программ и схем БД с помощью классов и их иерархии проще, чем в других моделях.
3. Абстрактные типы данных помогают предотвратить неправильное использование данных, если разрешить доступ к ним только посредством точно спроектированных функций.

Абстрактные типы данных.

Классы являются *абстрактными типами данных*. Они *инкапсулируют* или ограничивают доступ к объектам данного класса так, что только методы, определенные для этого класса, могут прямым образом изменять его объекты.

Иерархии классов.

Один класс A может являться *подклассом* другого класса B. Тогда класс A *наследует* все свойства класса B, включая тип данных и методы. Однако A может иметь и дополнительные свойства.

Пример.

Рассмотрим класс, объектами которого являются банковские счета.

```
CLASS Account = {accountNo:integer;
                 Balance:real;
                 Owner:REF Customer;
                 }
```

Можно определить методы для этого класса, например,
Deposit (a:Account, m:real),
увеличивающий баланс для объекта a класса Account на величину m.
Класс Account может иметь множество подклассов, например TimeDeposit, для которого можно определить дополнительный метод:
Penalty (a: TimeDeposit),

приписывающий счет а подклассу TimeDeposit и начисляющий штраф за преждевременное снятие денег со счета как функцию от поля DueDate объекта а и текущей системной даты.

1.2 Реляционная модель данных

В последние годы в большинстве ***БД*** используются ***реляционные модели данных***, и практически все современные ***СУБД*** ориентированы именно на такое представление информации. ***Реляционную модель*** можно представить как особый метод рассмотрения данных, который включает как собственно данные (в виде таблиц), так и способы работы и манипуляции с ними (в виде связей). Другими словами, в ***реляционной БД*** используется несколько таблиц, между которыми устанавливаются связи. Таким образом, информация, введенная в одну таблицу, может быть связана с одной или несколькими записями из другой таблицы.

Двумерная таблица в реляционной теории называется ***отношением***. Наименованиями столбцов в таблице служат имена ***атрибутов***.

Совокупность схем отношений, используемых для представления модели данных, называется ***схемой реляционной базы данных*** (реляционной моделью данных). Текущие значения соответствующих отношений называются ***реляционной базой данных***.

Между записями двух таблиц (например, таблиц A и B) могут существовать следующие основные виды связей:

- "***один к одному***" (каждой записи из таблицы A соответствует одна определенная запись из таблицы B, например, работник получает зарплату на предприятии, и только одну);
- "***один ко многим***" (каждой записи из таблицы A соответствует несколько записей из таблицы B, например, в доме проживает много жильцов)
- "***многие к одному***" (множеству записей из таблицы A соответствует одна определенная запись из таблицы B, например, несколько студентов учатся в одной группе);
- "***многие ко многим***" (множеству записей из таблицы A соответствует множество записей из таблицы B, например, у нескольких студентов занятия ведут разные преподаватели).

1.3 Манипулирование данными в реляционной модели

Для манипулирования данными в реляционной модели используются два формальных аппарата:

- ***реляционная алгебра***, основанная на теории множеств;
- ***реляционное исчисление***, базирующееся на исчислении предикатов первого порядка.

Операции, реализуемые с помощью средств указанных аппаратов, обладают важным свойством: они замкнуты на множестве отношений. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления

определяются над отношениями реляционных БД и результатом вычисления также являются отношения. В результате любое выражение или формула могут интерпретироваться как отношение, что позволяет использовать их в других выражениях или формулах.

Реляционная алгебра обладает большой выразительной мощностью, очень сложные запросы к базе данных могут быть выражены с помощью одного выражения реляционной алгебры.

Выражения реляционной алгебры строятся на основе алгебраических операций (высокого уровня), и подобно тому, как интерпретируются арифметические и логические выражения, выражения реляционной алгебры имеют процедурную интерпретацию.

Операции реляционной алгебры определены на множестве отношений и являются замкнутыми относительно этого множества (образуют алгебру). Любой произвольный запрос к БД можно представить в виде последовательности, составленной из пяти базовых операций реляционной алгебры.

1. ***Выбор*** (селекция). Обозначается σ_P **(R),** где R-отношение, а P – предикат (условие выбора). В качестве условия выбора может использоваться любое арифметическое или логическое выражение. Результатом этой операции является множество строк (кортежей) отношения, компоненты которых удовлетворяют условию P.
2. ***Проекция.*** Обозначается $\Pi_{A1,A2,\ldots,Am}$ **(R)**, где R-отношение, A_1, A_2, ..., A_m – имена столбцов. Результатом операции является множество строк (кортежей), полученное из всех строк отношения R выбором столбцов с именами A_1, A_2, ..., A_m. Другими словами, это операция построения «вертикального» подмножества, получаемого путем выбора определенных атрибутов и исключения остальных. Повторяющиеся строки исключаются.
3. ***Декартово произведение.*** Обозначается **R x S**, где R – k_1-арное отношение, а S – k_2-арное отношение. Результатом этой операции является множество кортежей длины $k_1 + k_2$, где первые k_1 компонентов которых образуют кортежи, принадлежащие R, а последние k_2 – кортежи, принадлежащие S.
4. ***Объединение.*** Обозначается $\mathbf{R} \cup \mathbf{S}$, где R и S – отношения. Результатом операции является множество кортежей, которые принадлежат или R, или S, или им обоим. Для операции объединения требуется одинаковая арность отношений.
5. ***Разность.*** Обозначается **R – S**, где R и S – отношения. Результатом операции является множество кортежей, принадлежащих или R и не принадлежащих S. Для этой операции требуется одинаковая арность отношений.

Наряду с базовыми операциями используются дополнительные операции:

6. ***Пересечение.*** Обозначается $\mathbf{R} \cap \mathbf{S}$, где R и S – отношения. Результатом операции является множество кортежей, принадлежащих как R, так и S. Пересечение может быть выражено через уже введенные операции
$$R \cap S = R - (R - S).$$
7. ***Соединение.*** Обозначается $\mathbf{R} \otimes \mathbf{S}$, где $R = (A_1, A_2, \ldots, A_k, B_1, \ldots, B_n)$ и $S = (A_1, A_2, \ldots, A_k, C_1, \ldots, C_m)$, отношения, имеющие одинаковые атрибуты. Результатом операции является множество кортежей в декартовом произведении R и S, удовлетворяющих выражению
$$R \otimes S = \Pi_{B1,\ldots,Bn,A1,\ldots,Ak,C1,\ldots,Cm}(\sigma_{R.A1=S.A1\&R.A2=S.A2\&\ldots.R.Ak=S.Ak}(RxS)).$$

1.4 Нормализация отношений

Важным понятием в теории реляционных ***БД*** является нормализация отношений. Так как реляционная модель предполагает различные варианты выбора схем отношений, то от правильного выбора схемы в значительной степени будет зависеть эффективность функционирования базы данных.

Основными проблемами при проектировании логической модели БД являются:

- Дублирование информации (избыточность).
- Потенциальная противоречивость (аномалии обновления).
- Потенциальная возможность потери сведений (аномалии удаления).
- Потенциальная возможность невключения информации в базу данных (аномалии включения).

В теории реляционных баз данных существуют формальные методы построения реляционной модели баз данных, в которой отсутствуют избыточность и аномалии обновления, удаления и включения.

Построение рационального варианта схем отношений (обладающего лучшими свойствами при операциях включения, модификации и удаления данных, чем все остальные наборы схем) осуществляется путем нормализации схем отношений, принципы которой можно сформулировать в виде следующих основных правил, используемых при разработке структуры ***БД***

1. В каждом поле таблицы должен находиться уникальный вид информации, т.е. в одной и той же таблице не должны находиться повторяющиеся поля

2. В каждой таблице должен быть первичный ключ или уникальный идентификатор, который однозначно определяет данную запись среди множества записей таблицы.

3. Каждому значению первичного ключа должна соответствовать исчерпывающая информация об объекте таблицы.

4. Изменение значения любого поля таблицы, не входящего в состав первичного ключа, не должно влиять на информацию в других ее полях.

Нормализация производится в несколько этапов. На начальном этапе схема отношений должна находиться в первой нормальной форме (1 НФ). Далее отношение, представленное в первой нормальной форме, последовательно преобразуется во вторую и третью нормальную формы. При некоторых предположениях о данных третья нормальная форма является искомым наилучшим вариантом.

Если эти предположения не выполняются, то процесс нормализации продолжается дальше, и отношения преобразуются в четвертую и пятую нормальные формы.

2 Работа в MS SQL Server

2.1 Средства управления базами данных SQL Server

Службы SQL Server

MS SQL Server реализован в виде самостоятельных служб, каждая из которых отвечает за выполнение определенного круга задач:

- MSSQLServer - процессор базы данных SQLServer;
- SQLSeverAgent - автоматическое выполнение задач администрирования;
- MSDTS - управление распределенными транзакциями;
- MSSearch - полнотекстовый поиск.

Процессор базы данных MSSQLServer

Процессор баз данных MSSQLServer реализован как служба операционной системы, которая осуществляет большинство основных функций SQL Server:

- выполнение хранимых процедур,
- управление файлами баз данных и журналами транзакций,
- проверку учетных записей пользователей, распределение ресурсов между ними, выполнение запросов и команд Transact-SQL.

MSSQLServer эффективно распределяет ресурсы операционной системы между множеством пользователей, оптимизируя и распараллеливая их запросы для повышения производительности.

MSSQLServer является основной службой СУБД: если он не запущен, то невозможны доступ к базам данных и выполнение задач администрирования.

3 Создание новой базы данных

Создание новой базы данных удобнее всего осуществить посредством ***SQL Server Enterprise Manager***. Для этого выбирается ***Databases*** в дереве объектов окна ***SQL Server Enterprise Manager*** (рис. 3.1). После нажатия правой кнопки мыши на экране появится выпадающее меню. В данном меню выбираем ***New Database***.

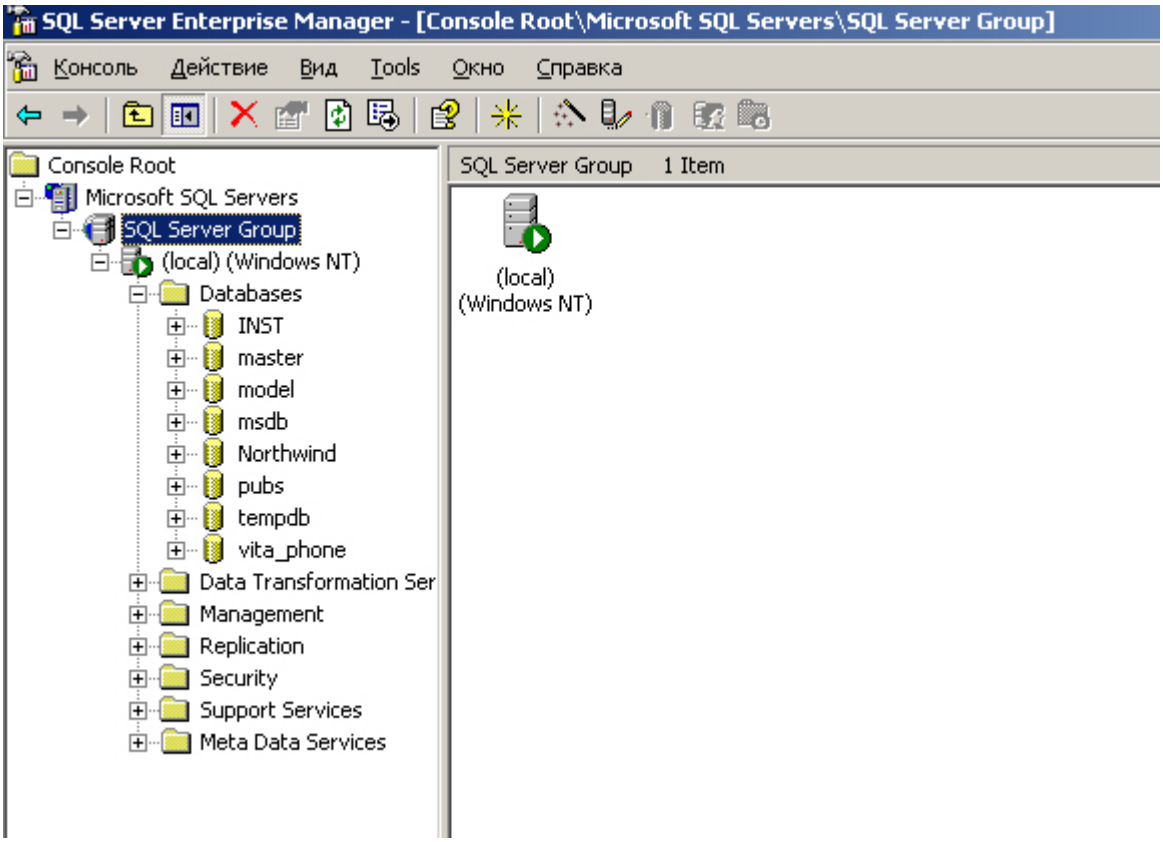

Рис 3.1. MS SQL Server Enterprise Manager.

Появляется новое диалоговое окно, в котором необходимо определить некоторые свойства новой базы данных.

Главная вкладка ***(General)*** (рис. 3.2) предназначена для определения имени новой базы данных и ограничений (при необходимости) на размеры файлов базы. После ввода логического имени новой базы данных в поле ***Name*** в окне ***Database files*** появляются предлагаемые по умолчанию параметры: имя созданного файла, путь к файлу базы данных, размер файла базы и принадлежность к группе.

Все эти параметры могут быть изменены. Необходимость использования нескольких файлов базы под одним логическим именем, введенном в поле **Name,** может потребовать добавления в окне ***Database files*** еще файлов.

> **Примечание**. Такая схема размещения файлов может быть полезна в случае сопровождения большой базы данных на нескольких жестких дисках или других носителях ограниченного объема или для удобства группировки различных объектов, хранимых в базе в собственных файлах.

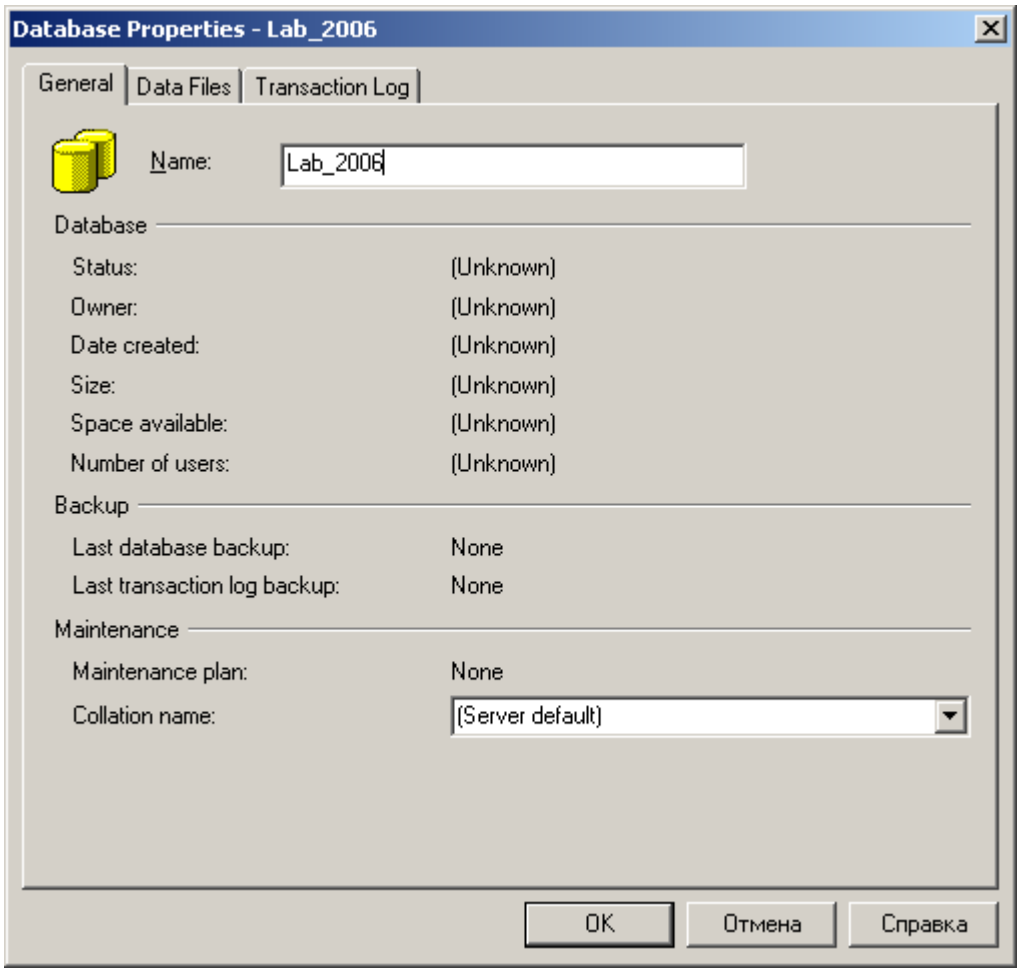

Рис 3.2. Вкладка General

Для каждого определенного в окне файла базы в нижней части вкладки ***Data Files*** окна ***Database Properties*** (рис. 3.3) автоматически задаются типичные значения некоторых параметров, установленных в ***MS SQL Server,*** для файла базы данных. Эти параметры задают начальный размер и методику увеличения размера файла по мере заполнения базы данными.

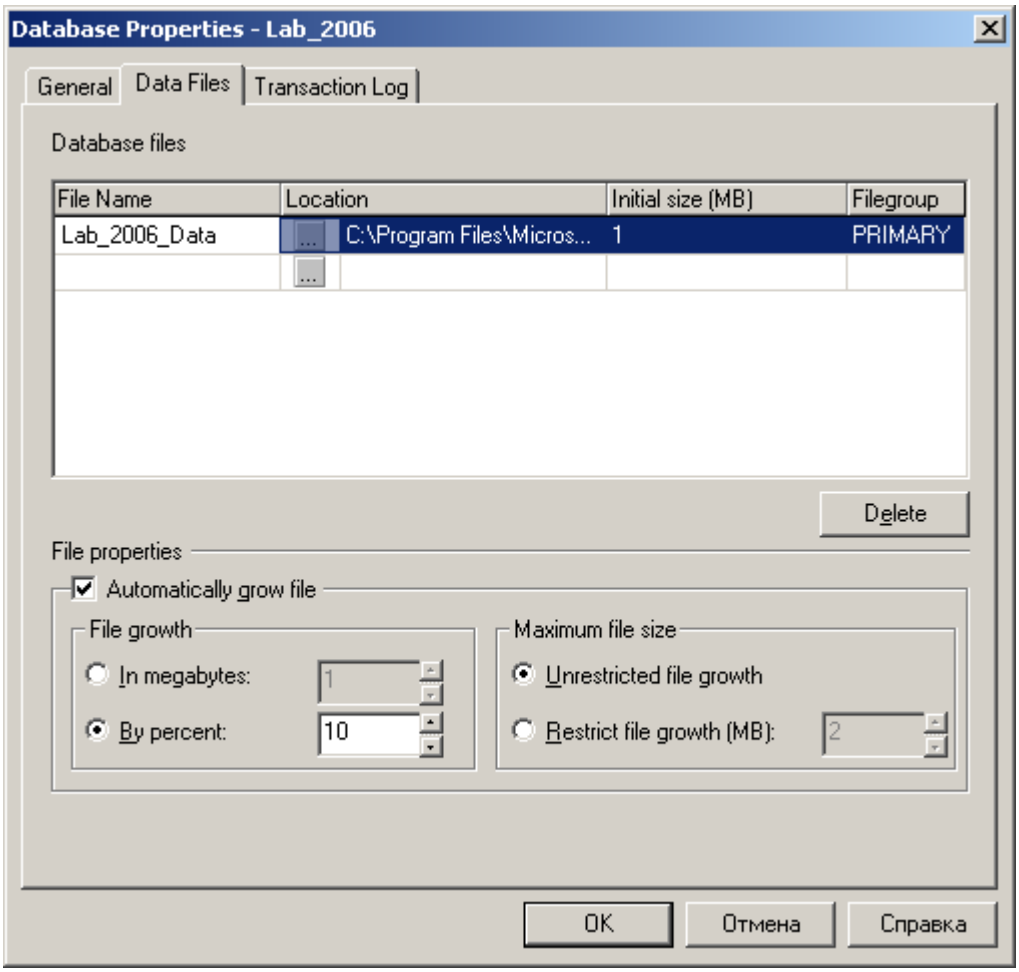

Рис 3.3. Вкладка Data Files

Установив флажок ***Automatically grow file***, можно задать автоматическое увеличение размера файла в соответствии с ограничениями, которые определяются здесь же. Может быть задан неограниченный рост файла - ***Unrestricted filegrowth*** или рост до определенного предела - ***Restricted filegrowth***. Во втором случае необходимо задать максимальный размер файла в мегабайтах. Кроме этого может быть установлен размер автоматического приращения файла при наступлении каждой необходимости его увеличения. Это приращение может быть определено либо в абсолютных значениях (мегабайтах), либо в относительных (процентах).

На следующей вкладке ***Transaction Log*** в окне ***File name*** (рис 3.4) определяются имя, размещение и размер обязательного файла, входящего в состав базы данных - журнала транзакций. Подробно суть и механизмы поддержки режима транзакций ***MS SQL Server*** будут рассмотрены позднее. Структура файлов, входящих в журнал, ограничения на размер файлов журнала транзакций и параметры их приращения при необходимости могут быть установлены аналогично параметрам файла базы данных.

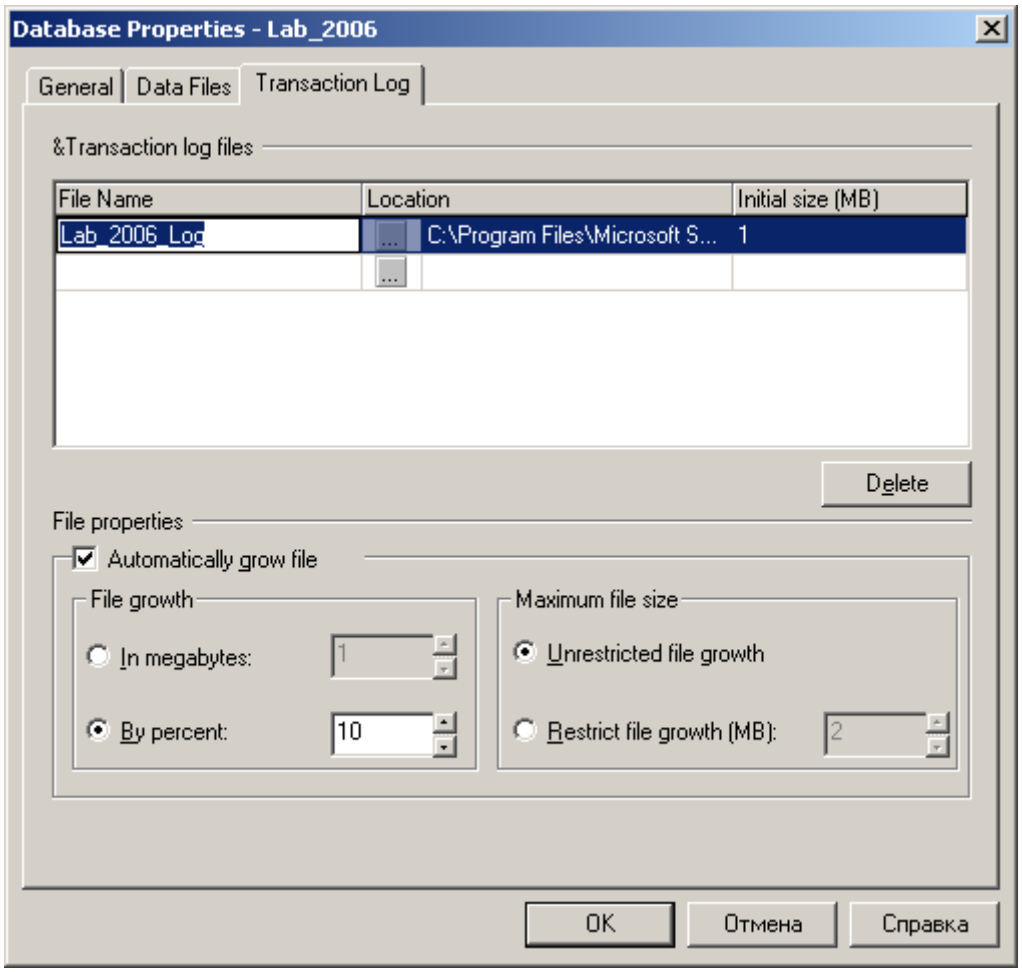

Рис 3.4. Вкладка Transaction_log

Последняя вкладка (рис 3.5) - ***Options*** (дополнительные параметры) позволяет определить следующие параметры (в некоторых версиях MS SQL Server данная вкладка доступна после создания базы при изменении ее свойств):

- ***Db_owner, dbcreator or sysadmin*** - база данных используется только ее владельцем, создателем или системным администратором.
- ***Single User*** - база данных используется только в монопольном режиме доступа одним пользователем. Одновременная работа с данными нескольких пользователей запрещена.
- ***Read Only*** - база данных может быть использована только для чтения данных. Модификация данных запрещена.
- ***ANSI NULL Default*** - разрешение заполнять поля автоматически значением NULL.
- ***Recursive Triggers*** - разрешение рекурсивного вызова триггеров.
- ***Auto close*** - автоматическая остановка сервера базы данных при окончании сеанса работы с ней последнего пользователя.

·***Torn Page Detection*** - проверка целостности операций ввода/вывода после краха сервера базы данных. Восстановление данных при необходимости из резервной копии.

Если определены все необходимые параметры новой базы данных, необходимо нажать кнопку ***OK*** и новая база данных будет создана на устройствах хранения файлов.

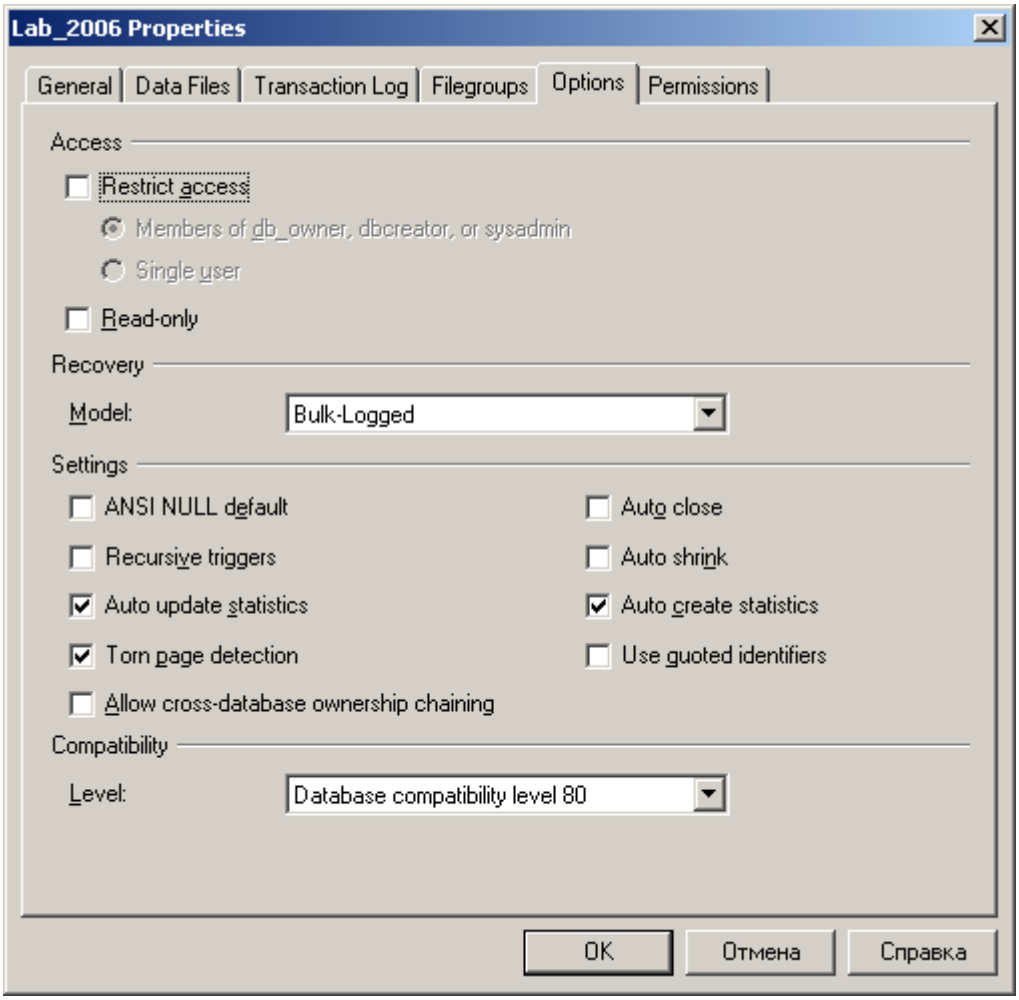

Рис 3.5. Вкладка OPTIONS

Необходимо помнить, что создание новой базы данных всегда является копированием модели базы (логическое имя - ***model***, пустая база в каталогах ***MS SQL Server***) в рабочую область и установкой ее параметров, необходимых разработчику, из образца (***template***). В пустой базе данных изначально содержатся системные таблицы, предназначенные для регистрации и использования сервером информации о вновь создаваемых пользовательских объектах и организации доступа к ним (имена пользователей, таблиц, полей и др.).

3.1 Объекты в базе данных

После создания новой базы данных в ней могут быть созданы определения (шаблоны) записей в нее информации. Все записи или хранимые данные в базе **SQL Server** организованы в виде различных объектов (различных типов информации). К ним относятся:

- **Пользователи базы данных** *(Users)* - учетные данные пользователей, которым разрешен доступ к данным.
- **Роли базы данных** *(Roles)* - поименованные группы прав доступа к данным. Пользователи ***SQL Server*** могут принадлежать одной или нескольким группам, что определяет их права доступа или роли.
- **Таблицы** *(Tables)* - определения для двумерных (таблицы/поля) структур хранения данных. Таблицы являются основными объектами реляционных баз данных.
- **Представления** *(Views)* - логические (виртуальные) двумерные структуры данных, определяемые как набор полей из других таблиц. В базе данных представления хранятся в виде операторов ***SQL***, обеспечивающих заданную выборку.
- **Хранимые процедуры** *(Stored Procedures)* - процедуры на языке ***Transact SQL***, которые хранятся в базе данных и выполняются на узле сети, выполняющем роль сервера.
- **Правила** *(Rules)* - ограничения на ввод данных в определенное поле. Правила хранятся в виде логических выражений.
- **Значения по умолчанию** *(Defaults)* - значения полей которые присваиваются им в момент создания записи и до наполнения данным присваиванием.
- **Типы данных, определенные пользователем** (*User Defined Datatypes*) - тип данных, который пользователь конструирует на основе имеющихся типов, задает ему имя и назначает впоследствии под этим именем.
- **Диаграммы базы данных** (*Diagrams*) - функциональные модели структуры базы и информационных связей между объектами.
- **Индексы** *(Indexes)* - поименованные наборы полей в таблице, по которым предполагается осуществлять поиск записи. Выборка записей по значениям этих полей со ссылкой на название индекса ускоряет доступ данных в несколько раз. Объявление индекса предполагает хранение в базе информации для доступа к записи не перебором записей, а непосредственным доступом к нужной зоне записей (странице) и перебором в ней.
- **Ключи** *(Keys)* - индекс, значение которого для каждой записи уникально и предназначено для выборки одной единственной записи. В отличие от простого уникального индекса ключ - это уникальный индекс, по которому осуществляется поиск записи по умолчанию, т.е. ключ - уникальный первичный индекс.

- **Триггеры** (Triggers) - хранимые процедуры, выполнение которых осуществляется каждый раз при совершении определенного для триггера события. Перечень событий обрабатываемых языком ограничен.

Для создания таблицы в базе данных при помощи ***SQL Server Enterprise Manager*** в дереве баз данных и их объектов необходимо выделить подраздел ***Tables*** нужной базы данных и либо щелкнуть на нем правой кнопкой мышки, а затем в выпадающем меню задать ***New Table***, либо нажать кнопку ***New*** на панели инструментов.

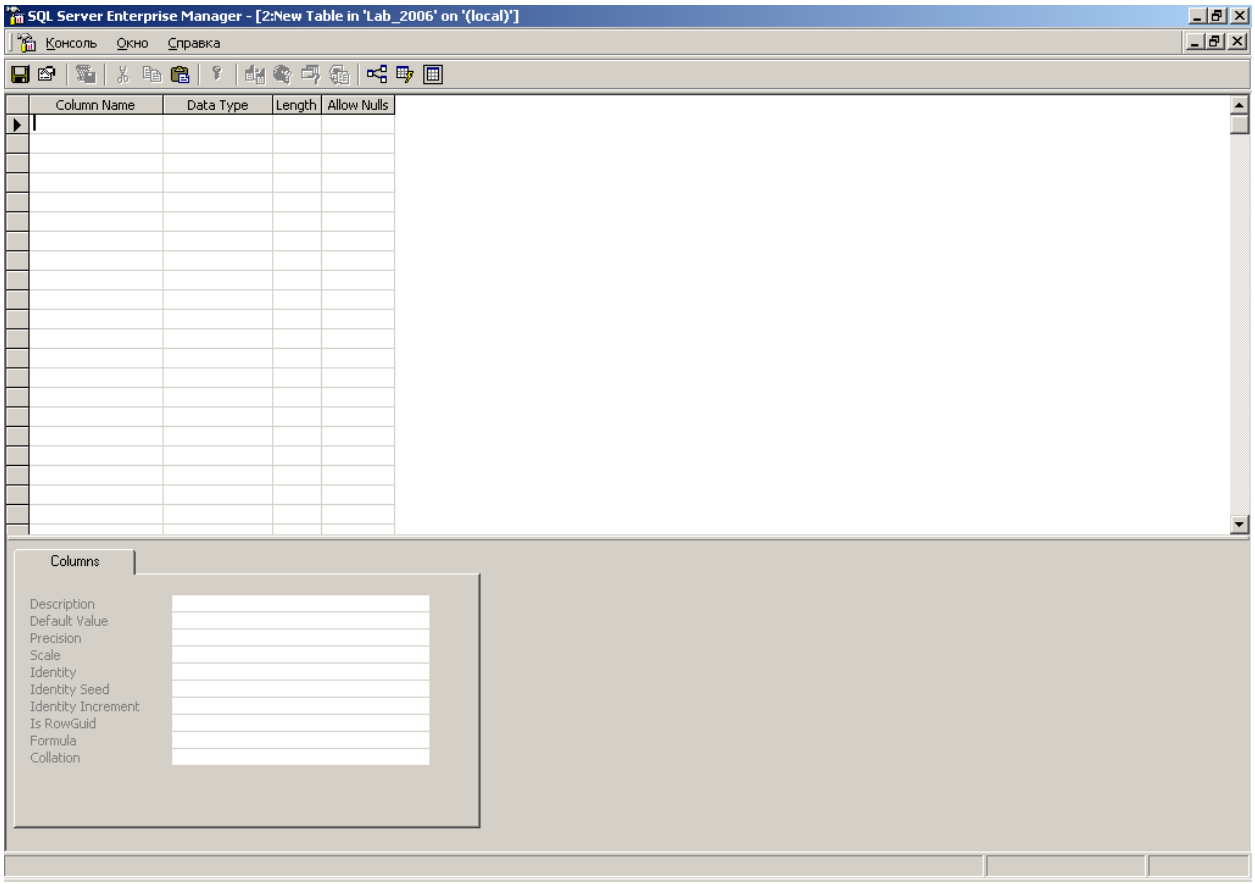

Рис 3.6. Окно создания новой таблицы

После ввода в появившемся окне имени новой таблицы (рис 3.6) может быть получен доступ к окну ***Design Table*** (рис. 3.7) определения полей (строк) данной таблицы.

В этом окне могут быть добавлены новые поля (строки) и определены их атрибуты или изменены атрибуты уже имеющихся полей. Для каждого поля (строки) должны быть обязательно определены имя и тип хранимых в нем данных. Остальные параметры ***MS SQL Server Enterprise Manager*** предложит по умолчанию. Значения по умолчанию могут быть изменены пользователем. К вторичным параметрам относятся:

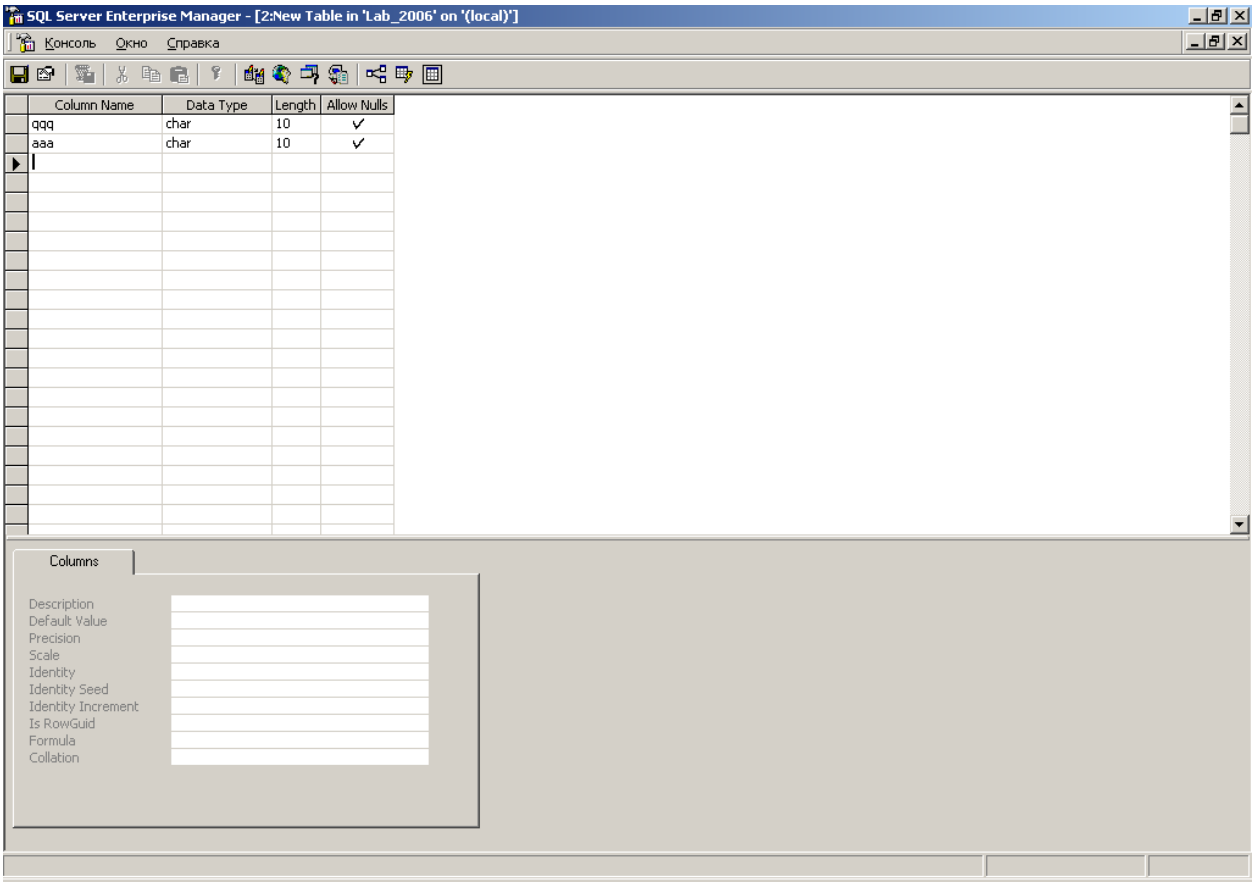

Рис 3.7. Структура полей таблицы

- Параметры управления форматом:

- **Length** - длина.
- **Precision** – точность
- **Scale** - масштаб.

- Другие параметры:

- **Allow Nulls** - разрешение не заполнять данное поле (поддерживать его пустым). В противном случае не может быть создана запись в базе, если данное поле не получило значения.
- **Default Value** - значение, которое будет размещено во вновь созданной записи по умолчанию при ее создании.
- **Identity** - означает, что данное поле будет использоваться как идентификатор записи и его значение будет вычисляться автоматически при создании записи посредством, определенным способом описанных правил в последующих столбцах ***Identity Increment*** (приращение) и ***Identity Seed*** (начальное значение). Разумеется, этот флажок работает только для различных форматов целочисленных данных.

· **Is Row Guid -** поле будет использоваться как глобальный уникальный идентификатор (т.е. использоваться системой для внутренней сортировки и уникального выбора записи по умолчанию).

Таблица в базе данных может быть создана также посредством оператора языка ***Transact SQL***. Синтаксис оператора следующий:

CREATE TABLE <таблица>
({ <поле> | <имя поля> AS <выражение> | <условие на значение таблицы> }
[, ...n])
[ON {<группа> | DEFAULT}]
[TEXTIMAGE_ON {<группа> | DEFAULT}]

где:

<таблица> - <имя таблицы> <тип данных> [**NULL** | **NOT NULL**]
[**IDENTITY** [(<начало>, <приращение>)
[**NOT FOR REPLICATION**]]]
[**ROWGUIDCOL**]

<условие на значение таблицы> - [CONSTRAINT <имя условия на значения>]
({ **PRIMARY KEY** | **UNIQUE**}
[**CLUSTERED** | **NONCLUSTERED**]
[**WITH** [**FILLFACTOR** = <фактор заполнения>]]
[**ON** { <группа> | **DEFAULT**}] | [**FOREIGN KEY**]
REFERENCES <ссылочная таблица>
[<ссылочное поле>]
[**NOT FOR REPLICATION**] | **DEFAULT** <константа> | **CHECK**
[**NOT FOR REPLICATION**]
(логическое выражение) }] [, ...n]

или:

<условие на значение таблицы> - [**CONSTRAINT** <имя условия на значения>]
({ **PRIMARY KEY** | **UNIQUE**}
[**CLUSTERED** | **NONCLUSTERED**]
{ (<поле> [, ...n]) }
[**WITH** [**FILLFACTOR** = <фактор заполнения>]]
[**ON** { <группа> | **DEFAULT**}] | [**FOREIGN KEY**]
{ (<поле> [, ...n]) }
REFERENCES <ссылочная таблица>
[<ссылочное поле>]
[**NOT FOR REPLICATION**] |

CHECK [**NOT FOR REPLICATION**]
(логическое выражение) }] [, ...n]

- <**имя таблицы**> - наименование таблицы, не более 128 символов (для временных - не более 116).
- **ON** <группа> - группа файлов в базе данных, где будет размещена таблица. DEFAULT - разместить в группе по умолчанию.
- **TEXTIMAGE_ON** <группа> - определить отдельную группу для таблиц с типами text, ntext, image.
- **NULL | NOT NULL -** разрешено или нет хранить значение NULL в поле.
- **IDENTITY** - значение в поле вносится автоматически внутренним счетчиком SQL Server при создании записи (например, поле используется как идентификатор записи).
- **ROWGUIDCOL** - глобальный уникальный идентификатор.
- **<поле> AS <выражение> -** определение виртуального поля (вычисляемого через значения других полей). В качестве <выражения> могут выступать константы, имена других полей, арифметические выражения или функция.

ПРИМЕР СОЗДАНИЯ ТАБЛИЦЫ:

```
CREATE TABLE [dbo].cif (
        id CHAR (10) NOT NULL ,
        lname CHAR (15) NOT NULL ,
        fname CHAR (15) NOT NULL ,
        mname CHAR (15) NOT NULL ,
        adress VARCHAR (40) NOT NULL ,
        region VARCHAR (30) NOT NULL ,
        CONSTRAINT icif PRIMARY KEY NONCLUSTERED
        (id, lname ) ON [PRIMARY]
        ) ON [PRIMARY]
```

Пример создания таблицы с использованием оператора языка T-SQL в SQL Query Analyzer представлен на рис. 3.8.

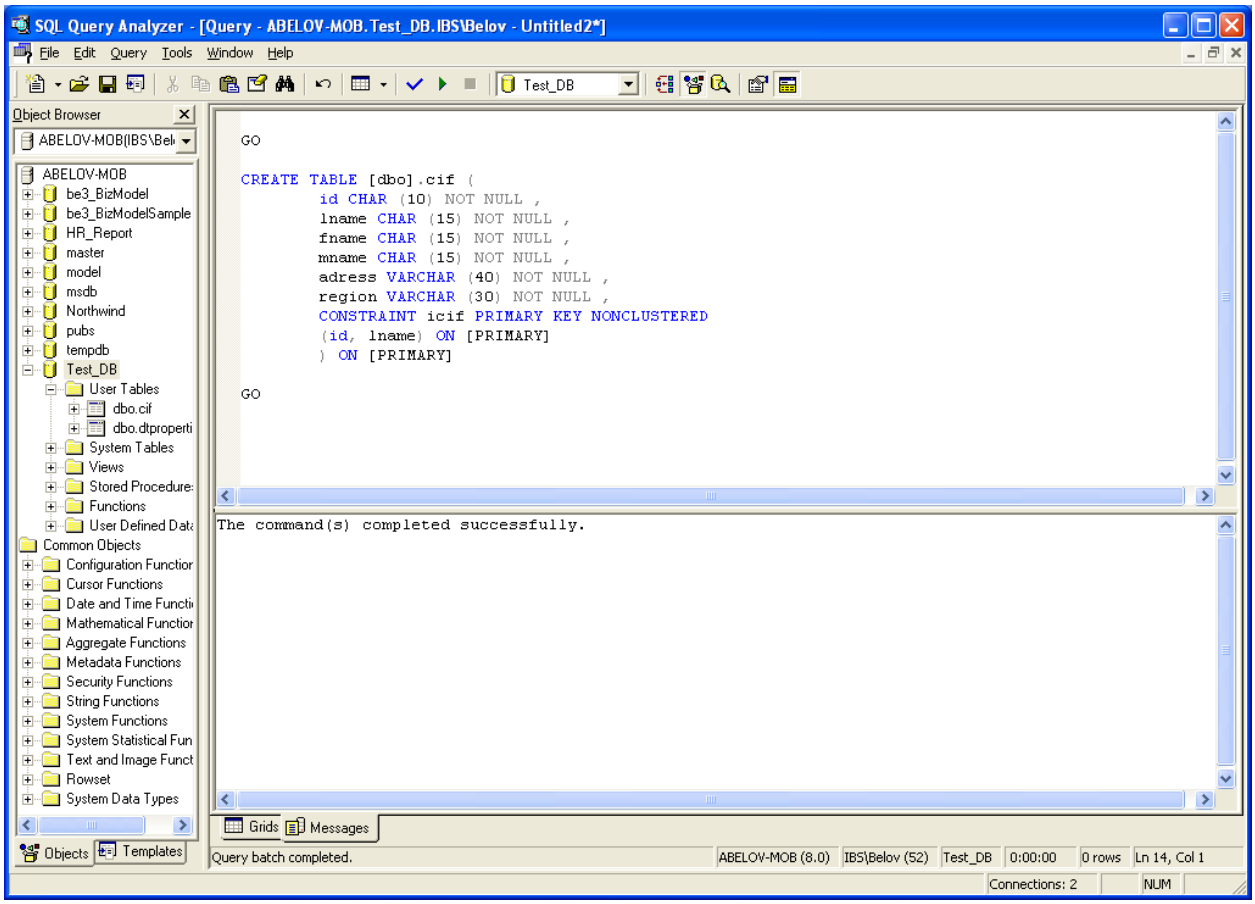

Рис. 3.8. Создание таблицы средствами SQL Query Analyzer

3.2 Добавление пользователей в системе

Для подключения к серверу пользователь должен ввести имя и пароль. Возможны два варианта определения прав на доступ:

- ***Use Windows NT authentication*** - подключение под именем, с которым пользователь регистрировался в Windows. В этом случае серверу автоматически передается имя, группа и пароль, под которым регистрировался пользователь и, если они корректны для сервера базы данных, осуществляется подключение.
- ***Use SQL Server authentication*** - для подключения к серверу пользователь должен явным образом объявить имя и пароль.

Подключение осуществляется к тому серверу базы данных, который выбран пользователем в верхнем поле диалогового окна ***Connect SQL Server***.

Имя пользователя, с которым он подключился к серверу базы данных, определяет его полномочия на просмотр и редактирование информации в тех или иных объектах базы данных. ***SQL Server*** предполагает два уровня проверки полномочий:

- проверка полномочий на этапе подключения к серверу (контроль осуществляется посредством таблицы ***syslogins*** главной базы данных ***master***).

- проверка полномочий на доступ к объектам конкретной базы данных во время сеанса работы с сервером (контроль осуществляется на основе информации, размещенной в таблице ***users*** и связанных с ней таблицах).

Права каждого пользователя определяются в соответствии с правами группы пользователей, к которой принадлежит данный пользователь (его идентификатор), и индивидуальными правами пользователя. Каждый пользователь при объявлении его на сервере или в базе данных и назначении его в одну или несколько групп наследует, таким образом, все права, которые установлены для данной группы или составлены как сумма прав всех групп. Права, наследуемые пользователем для доступа к каждому объекту базы данных в соответствии с правами группы (или ролью группы), могут быть в дальнейшем изменены, т. е. они могут быть уточнены для каждого пользователя отдельно по каждому объекту базы.

В модели базы данных, поставляемой вместе с дистрибутивом и являющейся прообразом любой новой базы данных, имеются, как правило, два стандартных пользователя ***SA*** и ***BUILTIN/Administrators***.

- ***SA*** - это пользователь из группы системных администраторов ***SQL Server*** с неограниченными правами по доступу к объектам.
- ***BUILTIN/Administrators*** - пользователь, который зарегистрировался в ***Windows NT*** в группе администраторов. Все они имеют доступ к серверу базы данных с правами системного администратора. Если это не предполагает характер доступа к операционной системе, права пользователей ***BUILTIN/Administrators*** можно минимизировать снятием с них роли ***sysadmin***.

Добавление входов на сервер базы данных может быть осуществлено с помощью оболочки ***MS SQL Server Enterprise Manager***. Для этого нужно выделить ***New Database User*** на сервере, входы на который необходимо отредактировать, и щелкнуть на нем правой кнопкой мыши.

В выпадающем контекстном меню, предложенном после этого, выбрать ***New Login***. Появится диалоговое окно ***SQL Server Login Properties*** с вкладками ***General, Server roles, Database Access***.

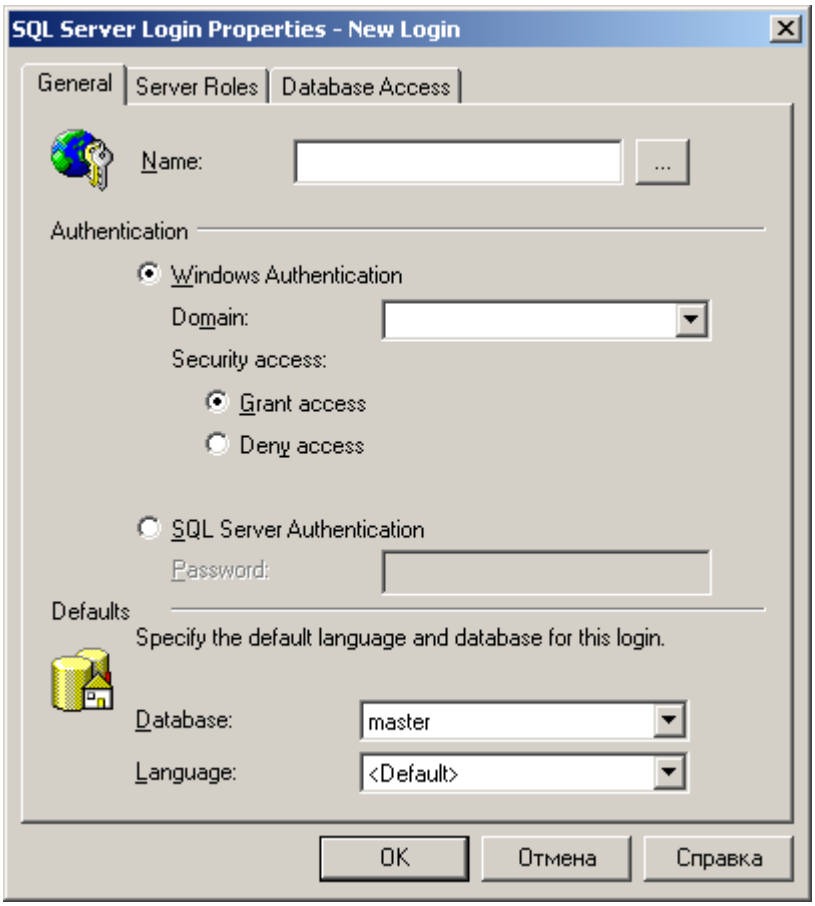

Рис 3.9. Вкладка General.

На вкладке ***General*** (рис 3.9) необходимо ввести или выбрать:

- новый идентификатор пользователя в поле ***Name***.
- базу данных данного сервера, с которой будет работать пользователь в поле ***Database***.
- язык (таблицу перекодировки для данного пользователя) в поле ***Language***.
- метод контроля за входом на сервер (***Authentication***).

Возможны два метода контроля прав доступа - ***Windows NT Authentication*** (разрешить доступ пользователей ***Windows NT*** под учетными данными операционной системы без дополнительного диалога, если они являются разрешенными пользователями ***SQL Server*** с такими же учетными данными) и ***SQL Server Authentication*** (запрашивать имя и пароль пользователя при входе на сервер).

В первом случае необходимо указать в поле ***Domain*** наименование домена, на котором хранятся учетные данные пользователей, а также установить тип доступа - ***security access***, который может быть разрешен (grant access) или запрещен (deny access).

Во втором случае необходимо установить пароль данного пользователя, которым он будет пользоваться при входе на сервер.

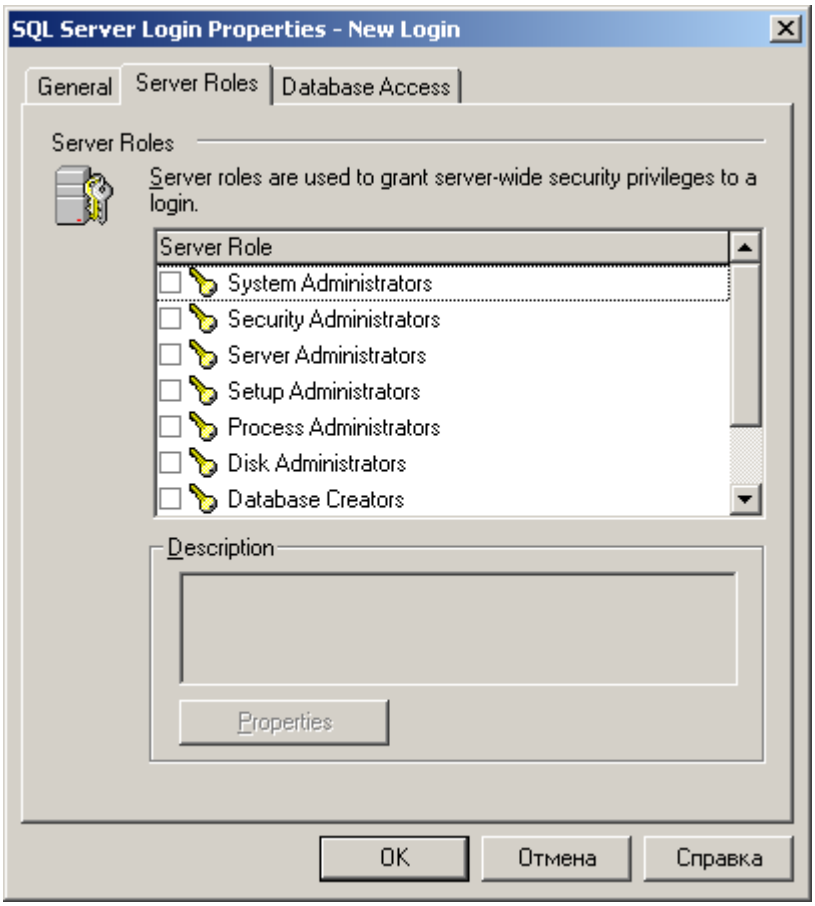

Рис 3.10. Вкладка Server Roles

На вкладке ***Server Roles*** (рис. 3.10) можно пометить галочкой одну или несколько ролей в соответствующем списке установленных ролей на данном сервере (групп пользователей). Права помеченных ролей в сумме должны обеспечить необходимый для данного пользователя уровень доступа. В нижней части окна в поле ***Description*** при перемещении маркера на каждую роль появляется подсказка в виде описания прав данной группы (роли).

В дистрибутивной версии предполагаются следующие стандартные роли для сервера:

- ***sysadmin*** - действия без ограничения прав.
- ***securityadmin*** - неограниченный доступ к учетным данным пользователей, создание баз данных и чтение журналов.
- ***serveradmin*** - доступ к параметрам конфигурации сервера базы данных.
- ***setupadmin*** - создание объекта репликации и управление расширенными процедурами.
- ***processadmin*** - управление процессами, порожденными SQL Server.
- ***diskadmin*** - управление файлами, обслуживаемыми данным сервером.
- ***dbcreator*** - создание и модификация баз данных.

На вкладке ***Database Access*** (рис. 3.11) необходимо определить роли пользователя для конкретных баз данных, входящих в состав сервера баз данных. Для этого в верхней таблице вкладки под названием ***Specify wich***

databases can be accessed by this login помечаются галочкой те базы, к которым пользователь может иметь доступ. В нижней таблице под названием ***Database Roles*** помечаются галочкой роли пользователя для каждой из разрешенных ему баз.

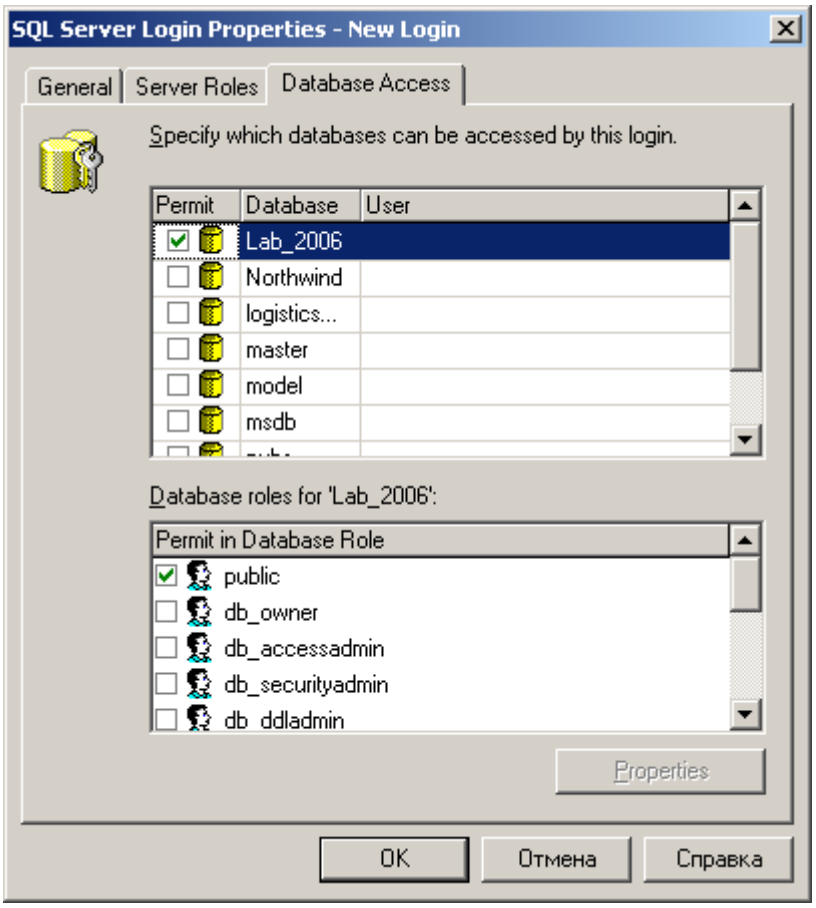

Рис 3.11. Вкладка Database Access

В дистрибутивной версии предполагаются следующие роли для базы данных:

- ***db_owner*** - неограниченные возможности, как у владельца базы данных.
- ***db_accessadmin*** - доступ к таблице пользователей и их прав для данной базы данных.
- ***db_datareader*** - чтение информации из любой таблицы.
- ***db_datawriter*** - изменение записей в любой пользовательской (!!!) таблице.
- ***db_ddladmin*** - добавление, редактирование и удаление объектов.
- ***db_securityadmin*** - доступ к таблицам ролей базы данных.
- ***db_backupoprrator*** - выполнять операции резервирования базы данных.
- ***db_denydatareader*** - модификация схемы базы данных.
- ***db_denydatawriter*** - запрещение модификации схемы базы данных.

Если все необходимые и достаточные для данного пользователя параметры введены, нажатие кнопки **OK** приведет к их вводу в системные таблицы баз данных.

4 Элементы языка запросов к базе данных

Для того чтобы клиентское приложение и серверная СУБД взаимодействовали друг с другом, они должны использовать общий язык. Это язык называется Structured Query Language (язык структурированных запросов), или просто SQL.

SQL появился в IBM в семидесятые годы как часть проекта System R. С этого времени его используют в качестве языка коммерческих реляционных СУБД. Он доступен и на универсальных ЭВМ, и на микрокомпьютерах.

Язык SQL относится к языкам четвертого поколения (4GL), что обеспечивает его независимость от аппаратного обеспечения системы, т.е. сам обеспечивает все функции ввода/вывода и хранения информации. Поэтому применение SQL упрощает управления базами данных по сравнению с процедурными языками программирования третьего поколения.

В настоящее время имеются два стандарта SQL: ANSI SQL-89 и ANSI SQL-92, которые были представлены American National Standards Institute или ANSI (американским национальным институтом стандартов) в 1998 и 1992 годах соответственно.

Хотя название SQL предполагает, что это язык запросов, он включает в себя помимо средств запросов определение таблиц, обновление базы данных, определение представлений данных и привилегий доступа, другие действия обработки информации, хранящейся в базе данных.

Существует несколько различных версий SQL. Microsoft SQL Server использует версию, называемую Transact-SQL (T-SQL). T-SQL соответствует всем требованиям ANSI SQL-89 и ANSI SQL-92, а также обеспечивает дополнительные возможности программирования, добавляя к нему функции, управляющие процессом выполнения программы, такие как if и while, локальные переменные, а также возможности, которые позволяют составлять сложные запросы, хранимые процедуры, триггеры, индексы и другие объекты.

4.1 Команды SQL

Команды SQL обозначаются ключевыми или зарезервированными словами (keywords), имеющими специальное значение в Microsoft SQL Server.

Команды SQL подразделяются на три типа:

- Data Definition Language (DDL) (язык определения данных),
- Data Manipulation Language (DML) (язык манипулирования данными),
- Data Control Language (DCL) (язык управления данными),

которые используются для создания баз данных, для добавления к ним информации и для работы с базами данных.

Различия между командами определения, манипулирования и управления такие же, как и различия между организацией, построением и регулированием. С помощью команд DDL системный администратор создает

группу таблиц, которые будут находиться на сервере. После этого конечный пользователь, применяя клиентскую программу, созданную с помощью DML и/или программного кода, манипулирует данными внутри структуры базы данных. Системный администратор или владелец базы данных может воспользоваться командами DCL, чтобы ограничить доступ к объектам базы данных и информации, содержащейся в ней.

Data Definition Language (DDL) (язык определения данных) - та часть T-SQL, которая применяется для построения, организации и удаления структур данных, создаваемых в базе данных. Для того чтобы использовать эти команды, необходимо получить полномочия на создание или редактирование объекта, который нужно определить или переопределить (это касается как самого объекта, так и его характеристики). Примеры команд DDL:

CREATE DATABASE
CREATE TABLE
DROP TABLE
CREATE VIEW
DROP VIEW

Data Manipulation Language (DML) (язык манипулирования данными) содержит команды T-SQL, которыми придется пользоваться чаще всего. В состав DML входят все команды, используемые для манипулирования объектами базы данных, а также информацией, содержащейся в базе данных. Для применения этих команд необходимо получить полномочия на внесение изменений в информацию, содержащуюся в таблицах, у владельца базы данных. Примеры команд DML:

INSERT
SELECT
UPDATE
DELETE
EXECUTE

Data Control Language (DCL) (язык управления данными) включает в себя те команды, которые управляют доступом к объектам базы данных. Сразу после загрузки системы полномочия на использование команд получает системный администратор (регистрационное имя sa (system administrator)), который предоставляет полномочия на использование этих команд владельцу базы данных (регистрационное имя dbo (database owner)). Системный администратор может применять наивысший уровень DCL, дающий право на управление любыми объектами базы данных, в то время как владелец базы данных имеет возможность использовать команды DCL, предоставляющие доступ только к тем объектам, которыми он владеет. Примеры команд DCL:

GRANT
REVOKE

4.2 SQL Query Analyzer

Команды T-SQL удобнее всего исполнять в графической утилите ***SQL Query Analyzer,*** которую, например, можно запустить из программного меню MS SQL Server. Затем необходимо подключиться к серверу, набрав в поле SQL Server имя сервера, откроется окно ***SQL Query Analyzer*** (рис. 4.1).

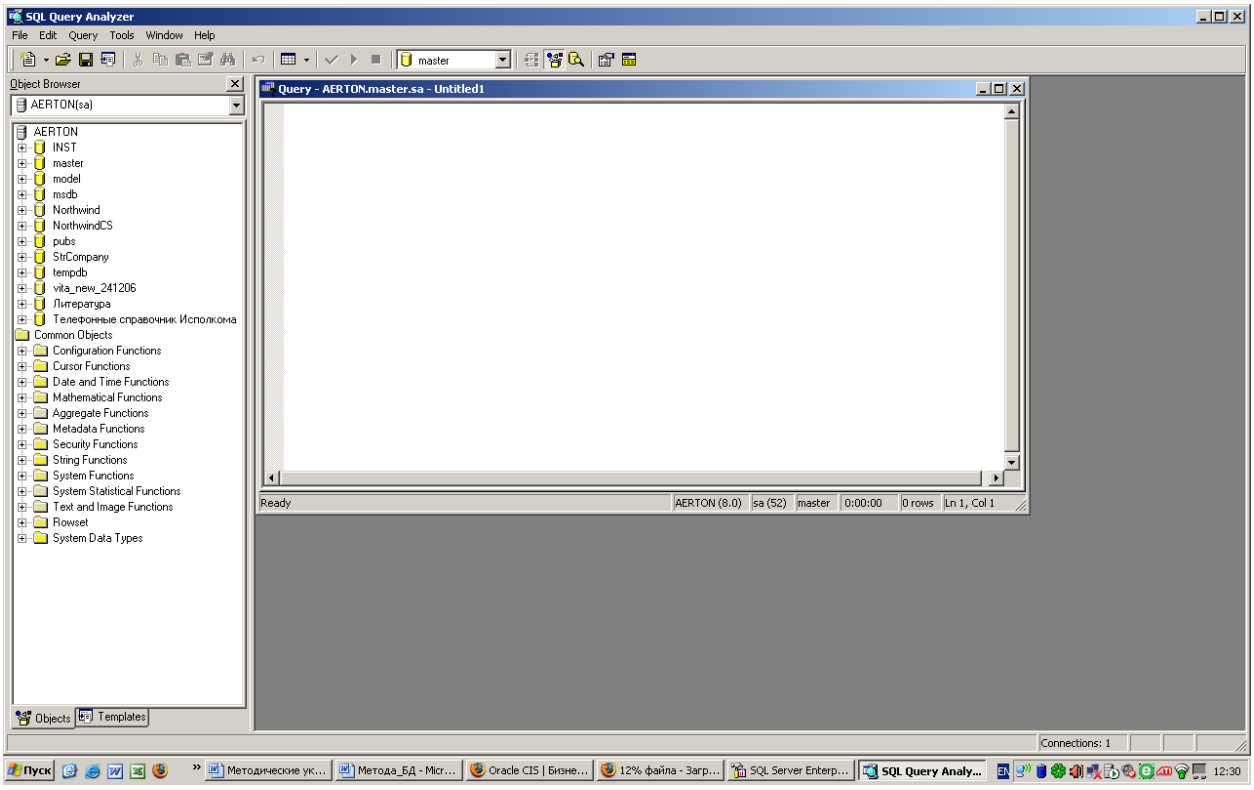

Рис. 4.1. Окно ***SQL Query Analyzer***

4.3 Описание базы данных *Pubs*

База данных ***Pubs (Publisher)*** является учебной базой данных MS SQL Server. Ее основное назначение - обеспечить наибольший набор интересных данных для выполнения основных команд T-SQL.

Диаграмма базы данных ***Pubs*** представлена на рис 4.2.

Каждый прямоугольник на диаграмме представляет таблицу и содержит название таблицы и названия полей. Линии между прямоугольниками представляют связи между таблицами.

База данных содержит информацию о деятельности трех филиалов издательской компании.

Таблица *Publishers* содержит информацию о трех издательствах: их идентификационные номера, названия и адреса.

Информация о каждом авторе, имеющем контракт с издателем, содержится в таблице *Abthors*: с номером карточки социального страхования, именем, фамилией и адресными данными.

Аналогичную информацию о каждом редакторе содержит таблица *Editors.* Кроме того, в ней имеется дополнительный столбец, описывающий

вид выполняемой редактором работы (подбор информации или управление всем проектом).

По вышедшим и готовящимся к печати книгам таблица *Titles* содержит следующую информацию: идентификационный номер, название, тема, идентификационный номер издательства, цена, расходы, количество проданных экземпляров, состояние контракта, дополнительные заметки и дата выхода. Числа в столбце *ytd_sales* должны изменяться по мере увеличения количества проданных книг одним из следующих способов:

- с помощью команд модификации данных,
- из приложений, которые будут автоматически изменять значения в столбце *ytd_sales*, как только будут вводиться новые данные в таблице *SalesDetails*,
- с помощью инструкций SQL для определения триггеров, выполняющих автоматическое обновление.

Книги и авторы представлены в разных таблицах, но могут быть связаны с помощью третьей таблицы - *TitleAuthors*. Для каждой книги таблица *TitleAuthors* содержит строку с значениями идентификатора книги, идентификатора автора, позиции автора в списке авторов книги и информацию по разделению гонорара.

Таблица *TitlEditors* аналогично связывает книги с их редакторами. Кроме того, она описывает порядок редактирования, т.е. можно узнать, кто был первым или последним редактором.

Таблица *Royshed* описывает зависимость между количеством проданных книг и размером авторского гонорара. Гонорар составляет определенную часть суммы, полученной за проданные книги.

Таблица *Sales* содержит общую информацию о заказах, полученных от книжных магазинов, номер квитанции на продажу, полученный от издателя, идентификатор магазина, номер заказа на покупку, получаемый от книжного магазина, и дату выполнения заказа.

Таблица *SalesDetails* содержит информацию о каждом пункте заказа на покупку, предполагая, что сразу может быть заказано несколько книг: название, количество заказанных книг, количество отправленных книг и дата отправления.

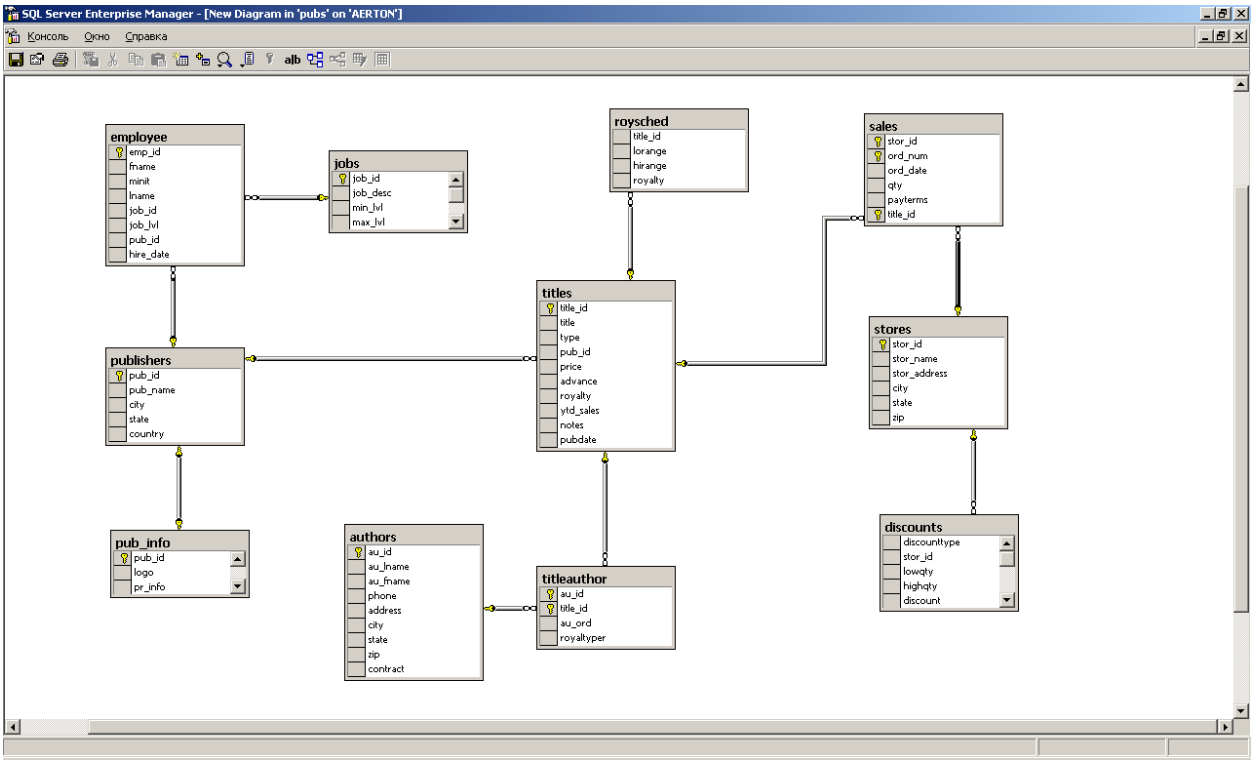

Рис. 4.2. Диаграмма базы данных ***Pubs***

4.4 Манипулирование данными: операции выборки данных

В этом разделе рассматривается команда или инструкция **SELECT**, без преувеличения являющаяся сердцем SQL, ее синтаксис и варианты использования. Все примеры основаны на базе данных ***Pubs,*** описание которой рассмотрено выше. Некоторые примеры могут показаться несколько сложными. Но из этого впечатления не следует делать вывод, что сложен сам язык SQL. Дело скорее в том, что не просто проиллюстрировать полную мощность языка SQL в локальных примерах.

При работе с таблицами основная цель всегда заключается в том, чтобы получить только ту информацию, которая требуется пользователям. Запрос к базе данных представляет собой операцию выборки, которая сужает диапазон считываемой информации и возвращает только определенный набор столбцов или строк, удовлетворяющих заданным критериям, из таблиц вашей базы данных.

Выбранные строки и столбцы базы данных всегда собираются во временную таблицу. В большинстве случаев эта таблица существует ровно столько времени, сколько нужно, чтобы данные были переданы запрашивающему клиенту.

4.4.1 Синтаксис инструкции SELECT

SELECT [ALL | DISTINCT] список__выбора
[**INTO** [имя_новой_таблицы]]
FROM {имя_таблицы_1 | имя_представления_1} [(режим оптимизатора)]

[[,{имя_таблицы_2 | имя_представления_2} [(режим оптимизатора)]
[...,{имя_таблицы_N | имя_представления_N} [(режим оптимизатора)]]
[**WHERE** предложение]
[**GROUP BY** предложение]
[**HAVING** предложение]
[**ORDER BY** предложение]
[**COMPUTE** предложение]
[**FOR BROUSE**]

Аргументы инструкции:

- ***список_ выбора.*** Список названий столбцов (или полей, или атрибутов) таблицы или таблиц базы данных, которые войдут в результирующую таблицу запроса.

Пример 1. Простая выборка. Выбрать (просмотреть) имена и фамилии авторов, которые указаны в таблице *Authors.* Порядок перечисления имен столбцов может быть произвольным:

SELECT *au_fname, au_iname*
FROM *Authors*

Данная инструкция определит для клиента все записи из таблицы *Authors,* содержащие только два поля *au_fname, au_iname.*

Особую роль в списке выбора играет символ “звездочка” (*), который означает выбор всех имен столбцов таблицы или таблиц базы данных, указанных в *списке_таблиц*, стоящем после зарезервированного слова **FROM**:

SELECT *
FROM *список_таблиц*

При этом, столбцы отображаются в том порядке, в котором они были определены в инструкции (или инструкциях) **CREATE TABLE**.

Пример 2. Простая выборка

Выбрать (просмотреть) все данные, которые находятся в таблице *Authors*:

SELECT *
FROM *Authors*

Результат выполнения запроса представлен на рис. 4.3.

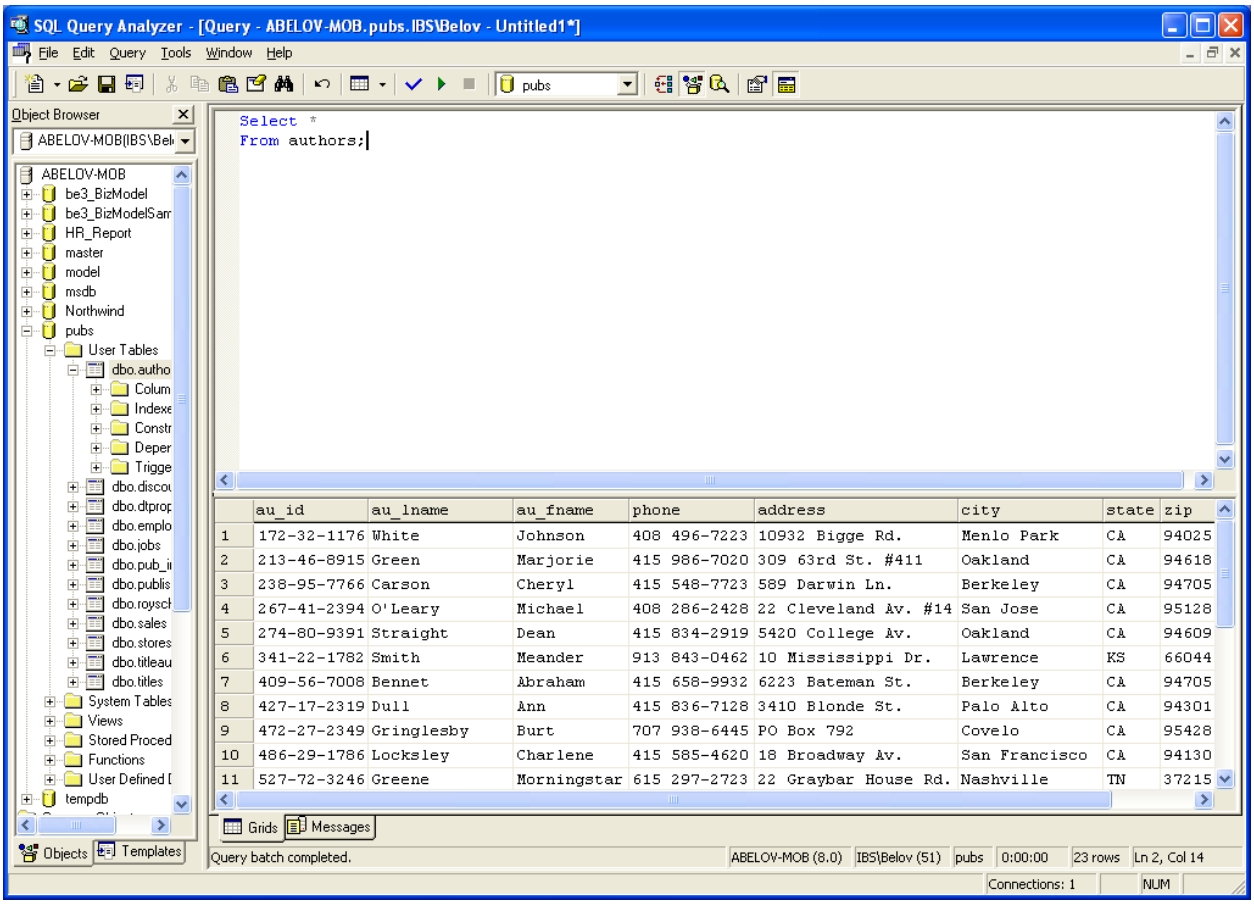

Рис. 4.3. Результат выполнения запроса

- ***список_таблиц.*** *Список имен таблиц, имена столбцов которых указаны в аргументе список_выбора.*

Пример 3. Простое эквисоединение
Просмотреть данные двух таблиц:

SELECT *
FROM *Authors, Titles*

Или

SELECT *Authors.*, Titles.**
FROM *Authors, Titles*

Возможно также выбрать таблицы из различных баз данных. В этом случае список таблиц состоит из элементов вида:

имя_базы_данных.имя_владельца_базы_данных.имя_таблицы

4.4.2 Примеры использования инструкции SELECT

Далее будут рассмотрены примеры, в которых последовательно продемонстрировано использование в запросах ключевых слов инструкции **SELECT**.

Пример 4. Ограниченная выборка (ограничение строк)

Определить имя, фамилию, номер телефона каждого автора, проживающего в штате Калифорния.

SELECT *au_fname, au_lname, state, phone*
FROM *Authos*
WHERE *state='ca'*

Данная инструкция будет посылать запрос к серверу и выводить на экран поля (имя, фамилию, штат проживания и номер телефона), указанные в строке SELECT, из таблицы, имя которой указано в строке FROM. При этом выбранные записи должны удовлетворять условию (авторы, проживающие в штате Калифорния), указанному в строке WHERE.

В условии ключевого слова WHERE допустимо использовать другие операции сравнения и логические операторы, которые приведены ниже соответственно в таблице 1 и таблице 2.

Таблица 1. Операторы сравнения

Символ	***Значение***
=	Равно
!=	Не равно
<>	Не равно
<	Меньше чем
>	Больше чем
<=	Меньше или равно
>=	Больше или равно
LIKE	Равно фрагменту значения

Таблица 2. Логические операторы

Название	***Значение***
OR	Логическое ИЛИ
END	Логическое И
NOT	Логическое отрицание (аналогичен оператору сравнения != (или <>))
BETWEEN	Выбор из диапазона значений

IN Задание списка значений

В Transact-SQL логические операторы позволяют сформировать несколько критериев считывания строк.

Пример 5. Выборка строк, удовлетворяющих заданным условиям

Выбрать авторов, имена которых начинаются с латинской буквы “B” и которые живут в штате Калифорния

SELECT *
FROM *Abthors*
WHERE *au_lname* **LIKE** *‘B%’*
AND *state=‘ca’*

Пример 6. Использование оператора **BETWEEN**

Определить названия книг, значение цены которых находится в заданном диапазоне.

SELECT *
FROM *Titles*
WHERE *price* **BETWEEN** *15* **AND** *21*

Пример 7. Использование оператора **IN**

Выбрать названия книг с заданной ценой (из указанного списка, стоящего после оператора IN).

SELECT *
FROM *Titles*
WHERE *price* **IN** (31.0, 20, 55.0)

Для представления неизвестной информации в полях таблиц базы данных используется значение **NULL**. В T-SQL имеются специальные операторы для выполнения сравнений со значением **NULL**:

WHERE *имя_столбца* **IS [NOT] NULL**

Пример 8. Выборка значений **NULL**

Получить информацию о книгах, для которых авансовые выплаты еще не определены.

SELECT *title, advance*
FROM *Titles*
WHERE *advance* **IS NULL**

Для удобства чтения полученной информации в результате произведенной выборки и более удобного ее анализа используется возможность представления этой информации в отсортированном виде:

ORDER BY *имя_столбца* **ASC [DESC]**

Здесь
ASC - возрастающий порядок сортировки,
DESC - убывающий порядок сортировки.

Пример 9. Сортировка результирующего набора
Выбрать названия книг, их цены и издательства, в которых они вышли. Результирующий список строк представить отсортированным по возрастанию цены.

SELECT *title, price, pub_in*
FROM *Title*
ORDER BY *price* **ASC**

В таблице в одном столбце могут храниться повторяющиеся значения. При выборке таких столбцов в результирующей таблице часто требуется просмотреть только уникальные значения. Это можно сделать с помощью ключевого слова **DISTINCT**.

Пример 10. Выборка уникальных значений
Получить список издательств.

SELECT DISTINCT *pub_in*
FROM *Titles*

SQL Server позволяет не только выбирать данные из таблиц, но и в *списке_выбора* выполнять математические действия над числовыми данными и константами, использовать специальные функции и символьные строки.

Таблица 3. Арифметические операторы Transact-SQL

Символ	***Действие***
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Модуль (остаток от деления целых значений)

Таблица 4. Базовые (агрегатные) функции Transact-SQL

Функция	*Действие*
AVG	Подсчет среднего (в столбце)
SUM	Суммирование (по столбцу)
MIN	Минимальное значение (в столбце)
MAX	Максимальное значение (в столбце)
COUNT	Подсчет числа выбранных строк

Пример 11. Переопределение имен столбцов при выводе и применение арифметических выражений.

Увеличить цену всех книг на 10%.

```
SELECT title ‘Название’, price ‘Цена’,
        price+price*0.1 AS ‘Новая цена’
FROM Titles
```

В результирующей таблице исходные имена столбцов получат новые названия и появится новый столбец с именем “Новая цена”.

Пример 12. Использование базовой функции **MAX** (определение максимального значения в столбце).

Определить самую дорогую книгу

```
SELECT MAX(price)
FROM Books
```

Функция возвратит результат в виде столбца (с одним значением цены самой дорогой книги).

Для выполнения объединения данных по некоторому критерию используется предложение **GROUP BY**, которое делит таблицу на группы строк с одним и тем же значением в заданном столбце.

Пример 13. Использование предложения **GROUP BY**

Определить количество книг, выпущенных каждым издательством.

```
SELECT pub_id, “Число книг”=COUNT(*)
FROM Titles
GROUP BY pub_id
```

При выполнении данного запроса строки с одинаковыми идентификаторами издательств объединяются в группы, для каждой группы вычисляется количество строк с помощью базовой функции **COUNT**. Результирующая таблица будет содержать столбец с идентификаторами

издательств (по одной строке на каждую группу) и столбец с числом книг, выпущенных в каждом из издательств.

Если выделенная группа удовлетворяет значению, указанному в предложении **HAVING**, то можно выбрать только определенные группы. Другими словами, предложение **HAVING** применяется к группам и работает аналогично предложению **WHERE** (но только для групп).

Пример 14. Использование предложения **HAVING**

Выбрать список идентификаторов издательств, отсортированный по возрастанию, сумму авансовых выплат и среднюю цену книг, не учитывая книг, цена которых меньше или равна 5, при этом значение идентификаторов должно быть больше ‘0800’, авансовые выплаты - больше 15000 и средняя цена книги - меньше 20.

```
SELECT pub_id, SUM(advance), AVG(price)
FROM Titles
WHERE price > 5
GROUP BY pub_id
HAVING SUM(advance) > 15000
    AND AVG(price) < 20
    AND pub_id > ‘0800’
ORDER BY pub_id
```

При выполнении запроса сначала будут выбраны строки, в которых цены книг больше 5 (**WHERE** price > 5), затем выбранные строки будут собраны в группы по значению идентификатора издательства (**GROUP BY** pub_id). В полученных группах будет подсчитана сумма выплаченных авансов и средняя цена книги. Промежуточная таблица будет содержать три столбца: идентификатор издательства, сумма выплаченных авансов и средняя цена книги в этом издательстве. В итоговую таблицу будет включен список строк из промежуточной таблицы, которые удовлетворяют условиям предложения **HAVING**.

4.5 Манипулирование данными: загрузка (импорт) данных в базу данных

Для загрузки и добавления информации в базу данных или таблицу используется инструкция **INSERT**. С помощью этой инструкции можно определять значения, которые будут вставлены в одну строку, или значения, которые будут вставлены сразу в несколько строк, используя для получения значений инструкцию **SELECT**.

Синтаксис инструкции INSERT:

INSERT [INTO] {<*таблица*> | <*представление*>}

{
{ [(*список_столбцов*)]
VALUES
(
{ **DEFAULT** | *константное_выражение* }[,...n])
| *предложэение* **SELECT**
| *предложение* **EXECUTE**
}
| **DEFAULT VALUES** }

Далее на примерах рассмотрены возможности использования инструкции **INSERT**.

Пример 1. Добавление одной строки

Вариант 1. Без указания параметра *список_столбцов*

INSERT *Authors*
VALUES (
‘257-41-2394’
‘Russel’
‘Chalie’
‘286-2484’
‘San Francisco‘
‘CA’
‘195311’)

Вариант 2. С использованием параметра *список_столбцов*

INSERT *Authors* (*au_id*, *au_iname*, *au_fname*, *phone*, *address*, *stste*,*zip*)
VALUES (
‘257-41-2394’
‘Russel’
‘Chalie’
‘286-2484’
‘San Francisco‘
‘CA’
‘195311’)

Данная инструкция добавляет строку в таблицу *Authors*, причем порядок указания значений полей в строке при вводе должен соответствовать порядку размещения столбцов при создании таблицы.

Пример 2. Добавление одной строки с заданием значений конкретных столбцов

INSERT *Authors* (*au_id, au_iname, au_fname*)
VALUES (
‘257-41-2394’
‘Russel’
‘Chalie’

Рассмотренная инструкция позволяет вставить значения в первые три столбца и проигнорировать оставшиеся.

Если в инструкции **INSERT** указывается столбец, но никакая информация в него не добавляется и, кроме того, в столбце не допускаются значения **NULL**, то весь процесс записи информации завершится неудачно. Эту проблему можно обойти, если с помощью опции **DEFAULT** задать значение, которое используется по умолчанию, если в инструкции **INSERT** значение не указано.

Пример 3. Добавление одной строки. Использование ключевого слова **DEFAULT** перед словом **VALUES**

INSERT *Authors*
DEFAULT VALUES

Пример 4. Добавление одной строки. Использование ключевого слова **DEFAULT** перед словом **VALUES**

INSERT *Authors*
DEFAULT VALUES

В примере внесено значение по умолчанию в каждый столбец таблицы.

Возможно использование ключевого слова **DEFAULT** и после слова **VALUES**, в этом случае значение по умолчанию будет внесено только в тот столбец, названию которого соответствует слово **DEFAULT**.

Возможно использование инструкции **INSERT** вместе с инструкцией **SELECT** для вставки сразу нескольких строк в таблицу. В этом случае инструкция **SELECT** помогает перенести данные из одной таблицы в другую.

Пример 5. Добавление нескольких строк

INSERT INTO *Addresses*
(*name, phone, address*)
VALUES (
SELECT *name, phone, address*
FROM *Pred_addresses)*

В результате выполнения приведенного выше кода для каждой строки таблицы *Pred_addresses* в таблице *Addresses* будет создана новая строка.

Следует помнить, что порядок следования столбцов, хотя сам по себе не важен, в обеих таблицах должен совпадать.

4.6 Манипулирование данными: изменение существующей информации

Изменение или обновление существующей информации - часто применяемая операция, для выполнения которой используется инструкция **UPDATE**. Этой инструкцией удобно пользоваться для изменения адресов, номеров телефонов, цен и других сведений, содержащихся в столбцах, т.е. для изменения существующих значений в столбцах таблицы.

Для удаления строк в одной таблице используется инструкция **DELETE**.

Синтаксис инструкции UPDATE
(сокращенная форма, достаточная для большинства случаев)

UPDATE {<*таблица*> | <*представление*>}
SET { *имя_столбца* = {выражение | **DEFAULT** | **NULL**}}[,...n]
[**WHERE** <*условие_поиска* >]

Обновляемые столбцы указываются в предложении **SET**, где простым присваиванием столбцу приписывается новое значение. Если обновляется несколько столбцов, то все они перечисляются через запятую.

Предложение **WHERE** используется для задания строк, которые должны быть изменены.

Пример 1. Использование арифметического выражения в предложении **SET**

Уменьшить стоимость всех книг на 2.

UPDATE *Titles*
SET *price=price*-2

Пример 2. Использование предложения **WHERE**
Изменить телефон у одного из авторов с заданным идентификатором.

UPDATE *Authors*
SET *phone*=‘234-7898’
WHERE *au_id*=‘0800’

Если изменилось также и место жительства этого автора, то через запятую необходимо определить новые значения для соответствующих столбцов в предложении **SET**.

4.7 Манипулирование данными: удаление существующей информации

Синтаксис предложения DELETE

(упрощенная форма, достаточная для большинства случаев)

DELETE
[FROM] <*имя_таблицы*>
WHERE <*условие_поиска*>

Инструкцию **DELETE** следует применять с большой осторожностью. Можно непреднамеренно удалить все строки таблицы и срочно искать последнюю резервную копию базы данных.

Пример 1. Удаление строк
Удалить информацию обо всех книгах данного издательства.

DELETE FROM *Titles*
WHERE *pub_id*=‘1389’

Из таблицы *Titles* удаляются все строки, относящиеся к издательству с идентификатором, равным 1389.

6. Порядок выполнения лабораторных работ

Лабораторные работы проводятся в два этапа.

6.1. Этап 1. Проектирование информационной системы и ее реализация средствами СУБД MS SQL Server

Задание 1. **Построить E-R модель (сформировать сущности, отношения и связи между ними). Представить модель в виде диаграммы Чена.**

Задание 2. **Реализовать для соответствующих моделей запросы на языке реляционной алгебры.**

К модели 1.:

- получить полную информацию обо всех сданных экзаменах и зачетах в выбранной группе;
- получить список всех студентов, не сдавших зачет по выбранной дисциплине;
- получить список студентов, не сдавших зачет более чем по 2-м предметам;
- получить список студентов, не получивших стипендию;
- получить список студентов, получивших ту же оценку на экзамене по выбранному предмету, что и выбранный студент;

- получить список предметов, по которым получено одинаковое количество зачетов;
- получить список групп, в которых студенты получили заданное количество оценок «пять» по выбранному предмету.

К модели 2.:

- получить список клиентов с заданным объектом страховки;
- получить список клиентов с заданным объектом страховки и одинаковой страховой суммой;
- получить список всех договоров, по которым произошел заданный страховой случай;
- получить список всех страховых случаев, происшедших с даты по дату;
- получить список клиентов, у которых тот же объект страховки, что и у заданного клиента;
- для заданного объекта страховки получить пары (имя клиента и номер договора), по которым произошел страховой случай.

К модели 3.:

- получить список всех пациентов с заданным диагнозом;
- получить список пациентов, которые обслуживаются тем же доктором, что и заданный пациент;
- получить список докторов- однофамильцев с заданным пациентом;
- получить список докторов, у которых лечатся пациенты не меньше заданного количества с заданным диагнозом;
- получить список пациентов, имеющих те же симптомы болезни, что и у заданного пациента;
- для заданного доктора «А» получить список его пациентов, имеющих тот же диагноз, что и выбранный пациент, лечащийся у доктора «В».

К модели 4.:

- получить список всех сотрудников, работающих в заданном подразделении;
- получить список сотрудников, работающих под руководством того же начальника, что и у выбранного сотрудника;
- получить список всех однофамильцев заданного начальника подразделения;
- для заданного подразделения получить пары (Ф.И.О. сотрудника, № военного билета), сотрудников, работающих в заданной должности;
- получить список всех подразделений, в которых работают сотрудники с ученой степенью (на основании диплома);
- получить список сотрудников, состоящий из троек (Ф.И.О. сотрудника, № паспорта, должность), пенсионного возраста. (Обратить внимание на то, что у мужчин и женщин пенсионный возраст разный).

К модели 5.:

- получить список всех актеров, снимающихся на заданной киностудии;
- получить список кинофильмов, в которых снимаются те же актеры, что и в заданном кинофильме;

- получить пары (Ф.И.О. актера, № контракта), занятых в фильмах, выпущенных на заданной киностудии;
- получить список киностудий, в которых были сняты фильмы по заданной тематике и в заданном году;
- получить список актеров, у которых имеется контракт с киностудией, расположенной в том же городе, в каком проживают эти актеры;
- получить список кинофильмов, в которых заданный актер не снялся ни разу.

К модели 6.:
- получить список всех матерей, у которых родились девочки;
- получить список матерей – однофамилиц, с заданным врачом;
- выбрать докторов, у которых пациентами являются дети, родившиеся в тот же день, что и у выбранной матери;
- получить список докторов, у которых пациентами являются близнецы;
- получить список матерей, у которых родились мальчики, на определенную дату;
- для заданного доктора получить пары (мать, ребенок), у которых пол совпадает.

К модели 7.:
- получить список всех пассажиров заданного рейса;
- получить список пассажиров, вылетающих рейсами до заданного города и имеющих билет на заданное место;
- получить список рейсов, на которых вылетали пассажиры с заданными Ф.И.О. и датой вылета;
- получить список, состоящий из пар (№ рейса, № места, на которые не проданы билеты);
- получить список пассажиров, вылетавших до заданного города в заданном интервале дат;
- получить список типов самолетов, летающих по тому же маршруту, что и самолет, на котором летел пассажир с заданным номером билета.

К модели 8.:
- получить список всех товаров, находящихся на всех складах;
- получить список кладовщиков, принимавших заданный товар;
- получить пары (название товара, № акта), отпущенные с заданного склада в заданный день товаров;
- получить список складов, на которые был осуществлен прием тех же товаров, что и на заданный склад;
- получить номера всех накладных на прием тех же товаров, что и принимал заданный кладовщик;
- получить список (наименование товара, номер склада, Ф.И.О. кладовщика), принятых по заданной накладной.

К модели 9.:
- получить список поставщиков заданной детали;

- получить список всех изделий, в которое входит заданная деталь от заданного поставщика;
- получить список потребителей изделия, в которые входит та же деталь, что и производится заданным поставщиком;
- получить список изделий, у которых поставщик деталей находится в том же городе, что и потребитель;
- получить список потребителей изделия, в которое входит одно и то же количество деталей, производимых разными поставщиками;
- получить все номера деталей, которые поставляются только одним поставщиком.

К модели 10.:

- получить список всех изданий, выпускаемых заданным издательством;
- получить список авторов, которые публикуют заданный вид рукописи в заданном издательстве;
- получить список изданий, выпущенных в том же издательстве, в котором публиковался автор, проживающий в том же городе, что и автор с заданной фамилией;
- получить список издательств и авторов, расположенных в одном и том же городе;
- получить список авторов, опубликовавших одну и ту же рукопись в разных издательствах;
- получить пары (автор, издательство) для рукописей, изданных за заданный период времени.

К модели 11.:

- получить список всех мат.ценностей (МЦ), требуемых заданному подразделению;
- получить список всех подразделений, которым требуется та же МЦ, что и заданному подразделению;
- получить список пар (МЦ, поставщик) таких, у которых стоимость МЦ не больше заданной;
- получить список пар (МЦ, количество), требуемых подразделениям организации, находящимся в том же городе, что и заданное подразделение;
- получить номера заявок от подразделений, которым требуется заданная МЦ;
- получить список поставщиков МЦ, находящихся в той же стране, что и подразделение организации, которой эта МЦ требуется.

К модели 12.:

- получить список всех исходящих документов за период;
- получить список всех корреспондентов, которым пришли письма в тот же день, что и заданному корреспонденту;
- получить номера входящих документов от корреспондентов, проживающих в том же городе, что и адресат;

- получить пары (номер входящего документа, номер исходящего), у которых темы одинаковы;
- получить список всех адресатов, которым были направлены письма по той же теме, что и заданному адресату;
- получить список (корреспонденты, номер входящего письма), ответы которым должны быть подготовлены к заданной дате.

К модели 13.:
- получить список всех налогоплательщиков, начисляющих заданный вид налога;
- получить список налогоплательщиков, переплативших по заданному виду налога;
- получить список налогов, по которым произошла недоплата;
- получить суммы начисленных налогов налогоплательщиками, которые перечислили деньги в тот же день, что и заданный налогоплательщик;
- получить список налогоплательщиков, не перечисливших деньги к заданной дате;
- получить список налогоплательщиков, у которых начисленная сумма по заданному виду налога лежит в заданном диапазоне.

К модели 14.:
- получить список всех УФПС, которым необходим заданный вид филат.продукции;
- получить список УФПС, которые заказали ту же фил. продукцию, что и заданное УФПС;
- получить номера заявок от УФПС, которым требуется заданное количество заданной фил.продукции;
- получить пары (вид фил. продукции, количество) для УФПС, расположенного в заданном регионе.

К модели 15.:
- получить все номера деталей, поставляемых поставщиком из заданного города;
- получить номера деталей, поставляемых для всех проектов, обеспечиваемых поставщиком из того же города, где размещен проект;
- получить список всех поставщиков, поставляющих одну и ту же деталь для всех проектов;
- получить список проектов, использующих по крайней мере одну деталь, производимую заданным поставщиком;
- получить список троек (номер детали, количество, поставщик), аналогичных деталям, поставляемым заданным поставщиком;
- получить список проектов, которые обеспечиваются деталями от одного поставщика.

К модели 16.:
- получить список ведущих артистов всех театров;
- получить список спектаклей, в которых занят заданный артист;
- получить список театров, в которых играют однофамильцы;

- получить пары (название театра, город), в котором идут те же спектакли, что и спектакль, в котором занят заданный артист;
- получить список спектаклей, в которых участвуют артисты, живущие в том же городе, что и театр;
- получить пары (название театра, спектакль), в которых играют артисты с той же фамилией, что и автор пьесы.

К модели 17.:
- получить список собак заданной породы;
- получить адреса хозяев собаки с заданными родителями;
- получить список хозяев собак заданной породы и заданного окраса;
- получить пары (имя хозяев, телефон) всех собак женского пола, у которых «папа» был тем же самым, что и у заданной собаки;
- получить список пород собак, хозяева которых проживают на одной улице;
- получить список имен собак заданной породы, у которых родители были чемпионами в заданном году.

К модели 18.:
- получить список всех поставщиков, производящих заданную ткань;
- получить список моделей одежды, у которых поставщики ткани и фурнитуры расположены в одном городе;
- получить пары(номер модели, номер изделия фурнитуры), входящих в модель одежды, сшитую из заданной ткани;
- получить список всех поставщиков, поставляющих ткань и фурнитуру для моделей заданного сезона;
- получить пары (наименование фурнитуры, количество), входящих в модели заданного сезона и заданной ткани;
- получить список номеров моделей одежды, сшитых из ткани заданного цвета.

К модели 19.:
- получить список всех товаров, продаваемых в заданном магазине;
- получить список товаров, поставщик которых находится в том же городе, что и магазин, их продающий;
- получить список всех потребителей, купивших товары стоимостью не больше заданной;
- получить список поставщиков, которые поставили заданный товар в магазин с заданным адресом;
- получить пары (номер товара, название поставщика), продаваемых через магазины, в которых обслуживается заданный потребитель;
- получить список пар (магазин, адрес), в которых продается заданный товар от заданного поставщика.

К модели 20.:
- получить список всех автомобилей, зарегистрированных в заданном году;
- получить список владельцев автомобиля заданной марки;

- получить список номеров свидетельств о регистрации АМТС той же марки и того же цвета, что и автомобиль, принадлежащий заданному владельцу;
- получить список пар (марка автомобиля, гос.номер), принадлежащих заданному владельцу и зарегистрированному не позднее заданного года;
- получить список троек (Ф.И.О. владельца, марка автомобиля, гос.номер) автомашин, выпущенных после заданного года и зарегистрированных в заданном отделении ГАИ;
- получить список адресов владельцев автомобилей, зарегистрированных в том же отделениии ГАИ и в том же году, что и автомобиль заданного владельца.

К модели 21.:

- получить список членов политклуба по определенной форме;
- получить список членов клуба, участвующих в заданном исполнительном органе государственной власти;
- получить список членов клуба, имеющих ту же награду, что и заданный член клуба;
- определить общую сумму взносов за заданный период времени;
- получить список невыполненных поручений и ответственных за них.

Задание 3. **Реализация информационной системы в MS SQL Server**

Создание новой базы данных

Перед созданием базы данных проверьте установки сервера базы данных: язык (кодовая страница), методика доступа и др. Во всех вариантах рабочий журнал транзакций располагать в том же каталоге, где будет размещена база данных, с тем же названием.

Формирование запросов к БД

Сформировать запросы к построенной базе данных информационной системы в соответствии с выбранной моделью и заданием 2 Этапа 1 выполнения лабораторной работы.

Список вариантов моделей приведен в разделе 7 настоящего документа.

7. Список заданий для выполнения работы

Для выполнения работы необходимо выбрать модель информационной системы из списка и осуществить последовательность действий в соответствии с приведенной в разделе 6 настоящего документа.

1. Модель «Деканат» должна содержать информацию о списках групп, начисленных стипендиях, зачетах, оценках, полученных на экзаменах.
2. Модель «Страховая компания» должна содержать информацию о клиентах, договорах страховки, объектах страховки, страховых случаях.
3. Модель «Поликлиника» должна содержать информацию о пациентах, докторах, диагнозах.

4. Модель «Отдел кадров» должна содержать информацию о сотрудниках, подразделениях организации, должностях, официальных документах (паспорт, диплом, военный билет и т.п.)

5. Модель «Фильмотека» должна содержать информацию о кинофильмах, актерах, киностудиях и контрактах между актерами и киностудиями.

6. Модель «Роддом» должна содержать информацию о родившихся детях, их матерях, докторах, курирующих ребенка.

7. Модель «Заказ билетов» должна содержать информацию о пассажире, билете, рейсе самолета.

8. Модель «Склад» должна содержать информацию о складе, товаре, кладовщике, документе (накладная на прием товара, акт на отпуск товара).

9. Модель «Предприятие» должна содержать информацию о поставщиках, производимых изделиях (составе изделия), потребителях продукции.

10. Модель «Издательство» должна содержать информацию об авторах, изданиях, видах рукописей.

11. Модель «Планирование материально-технического обеспечения организации» должна содержать информацию о материальных ценностях, подразделениях организации, заявках, поставщиках материальных ценностей.

12. Модель «Канцелярия» должна содержать информацию о типах документов (входящие, исходящие), собственно о документе, адресате и корреспонденте.

13. Модель «Налоговая инспекция» должна содержать информацию о зарегистрированных налогоплательщиках, видах налогов, начисленных и перечисленных суммах налогов.

14. Модель «Филателистическое агентство» должна содержать информацию о видах филателистической продукции, управлениях федеральной почтовой связи (УФПС), собственно о филателистической продукции, заявках от УФПС.

15. Модель «Завод» должна содержать информацию о поставщиках, деталях и проектах.

16. Модель «Театр» должна содержать информацию о театре, спектакле, ведущих артистах в спектакле.

17. Модель «Собачий каталог» должна содержать информацию о собаке, ее родителях, ее хозяевах.

18. Модель «Швейная компания» должна содержать информацию о моделях одежды, об используемых тканях, об используемой фурнитуре, о поставщиках ткани и фурнитуры.

19. Модель «Торговая компания» должна содержать информацию о товарах, магазинах, поставщиках, покупателях.

20. Модель «ГАИ» должна содержать информацию о зарегистрированных автомобилях, их владельцах, выданных документах на автомобиль.

21. Модель «Политклуб» должна содержать информацию о членах клуба (Ф.И.О.; гражданство; число, месяц, год рождения; место рождения; пол; паспорт (серия, номер, кем и когда выдан); образование (что и когда закончил); работа (наименование организации, должность); членство в общественных организациях (наименование); участие в представительных органах гос.власти и местного самоуправления (какие, в какое время, в качестве кого), правительств. награды; контактные телефоны; адрес проживания.), о мероприятиях клуба, выполнении персональных поручений, членских взносах, пожертвованиях.

Результаты выполненной работы оформляются в виде отчета.

8. Литература

1. Шустова, Л. И. Базы данных : учебник / Л.И. Шустова, О.В. Тараканов. — Москва : ИНФРА-М, 2021. — 304 с. + Доп. материалы [Электронный ресурс]. — (Среднее профессиональное образование). - ISBN 978-5-16-014161-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1189322 (дата обращения: 07.06.2022). – Режим доступа: по подписке

2. Базы данных : учебно-методическое пособие / Г. И. Ревунков, Н. А. Ковалева, Е. Ю. Силантьева [и др.]. — Москва : МГТУ им. Н.Э. Баумана, 2020. — 28 с. — ISBN 978-5-7038-5381-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/205187 (дата обращения: 07.06.2022). — Режим доступа: для авториз. пользователей.

Приложения
ЧЕБОКСАРСКИЙ ИНСТИТУТ (ФИЛИАЛ)
МОСКОВСКОГО ПОЛИТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Кафедра информационных технологий,
электроэнергетики и систем управления

КУРСОВАЯ РАБОТА

по дисциплине: Базы данных
на тему: «_____»

Выполнил:
студент группы:184-Ч091
Иванов Иван Иванович
учебный шифр:1242254

Проверила:
доцент Никитин А.В.

Чебоксары 2022

Заведующему кафедрой «Информационных технологий, электроэнергетики и систем управления»
Чебоксарского института (филиала Московского политехнического университета)
ФИО.
студента(ки) _____
_____
группы _____
тел. _____

заявление.

Прошу утвердить тему курсовой работы

_____
_____
(название темы)

_____
___
по дисциплине «Базы данных» и назначить руководителем ФИО.

Студент _____/ _____ / _____
(подпись) (ФИО студента) (дата)

Руководитель_____ / _____/ _____
(подпись) (ФИО руководителя) (дата)

Заведующий кафедрой _____ / _____ / _____
(подпись) (ФИО зав. кафедрой) (дата)