1 Лабораторная работа №1. Создание базы данных и определение ее структуры

Цель работы: освоение процедуры создания, удаления, резервирования и восстановления базы данных в среде MS SQL Server 20xx; освоение языка определения данных.

1.1 Знакомство с MS SQL Server 20xx

1.1.1 Логические компоненты базы данных

MS SQL Server является системой управления реляционными базами данных. *Реляционная база данных* – это база данных, разделенная на логически цельные сегменты, называемые *таблицами*, и внутри базы данных эти таблицы связаны между собой посредством *ключевых полей*. Таблицы являются основной формой хранения данных в базе данных. Реляционная база данных позволяет разделить данные на логически более мелкие и более управляемые сегменты, что обеспечивает оптимальное представление данных и возможность организации нескольких уровней доступа к данным. Вследствие наличия у таблиц общих *ключей* оказывается возможным объединить данные из нескольких таблиц в одно результирующее множество. Это является одним из основных достоинств реляционных баз данных. Набор действий, выполняемых по отношению к базе данных и рассматриваемый как единое целое, называется *транзакцией*. Транзакция представляет собой внесение в базу данных некоторых изменений. Каждая база данных имеет соответствующий ей *журнал транзакций* – место, куда SQL Server записывает все выполняемые транзакции перед тем, как записать их в базу данных.

Данные в SQL Server организованы в нескольких различных объектах, которые пользователь видит при создании базы данных. К ним относятся:

- пользователи базы данных Database Users);
- роли базы данных (Database Roles);
- таблицы (Tables);
- представления (SQL Server Views);
- хранимые процедуры (Stored Procedures);
- правила (Rules);
- значения по умолчанию (Defaults);
- типы данных, определенные пользователем (User Defined Datatypes);
- диаграммы базы данных (Database Diagrams).

Помимо этих видимых объектов в каждой базе данных имеются еще некоторые:

- условия на значения (Constraints);
- индексы (Indexes);
- ключи (Keys);
- триггеры (Triggers).

1.1.2 Типы команд SQL

Основные категории команд, реализующих в SQL выполнение различных функций:

- DDL (Data Definition Language — язык определения данных);
- DML (Data Manipulation Language — язык манипуляций данными);
- DQL (Data Query Language — язык запросов к данным);
- DCL (Data Control Language — язык управления данными);
- команды администрирования данных;
- команды управления транзакциями.

Среди таких функций — построение объектов базы данных, управление объектами, пополнение таблиц базы данных новыми данными, обновление данных, уже имеющихся в таблицах, выполнение запросов, управление доступом пользователей к базе данных, а также осуществление общего администрирования базы данных.

1.1.3 Основные типы данных

Типы данных позволяют хранить в базе данных различные по своей природе данные от любых символов до десятичных чисел, значений дат и времени. Подход к разделению данных на типы во всех языках одинаков - и при работе с переменными в языках третьего поколения типа C, и при работе с реляционными базами данных с помощью SQL. Хотя в каждой реализации SQL для стандартных типов данных используются разные имена, работают они практически одинаково. И при краткосрочном планировании, и с точки зрения перспективы нужно с особой тщательностью выбирать типы данных, их длину, масштаб и точность. При этом нужно принять во внимание и сложившиеся правила соответствующего бизнеса, и то, каким образом должны предоставляться данные конечному пользователю. Для этого вы должны понимать природу самих данных и то, как эти данные связаны внутри базы данных. Типы данных являются характеристиками самих данных, чьи атрибуты размещаются прямо в соответствующих полях таблицы. Например, можно указать, что некоторое поле должно содержать только числовые значения, и это не позволит вводить буквенно-числовые значения, когда, например, вы не хотите, чтобы последние появлялись в поле, предназначенном для хранения денежных значений.

Самыми общими типами в SQL, как и в большинстве других языков, являются: символьные строки; числовые строки; значения даты и времени.

Символьные (character) типы данных позволяют хранить буквенные, числовые и специальные (например, ? или >) символы. При загрузке в область хранения (такую, как столбец таблицы) символьные данные вводятся в одинарных или двойных кавычках.

Тип *char(n)*. При хранении данных этого типа для каждого символа используется один байт. Число n определяет размер области хранения максимального количества символов данного столбца. Если вводится значение, меньшее n, SQL Server добавит пробелы после последнего символа, чтобы общая длина равнялась n.

Для экономии дискового пространства, когда хранящиеся в столбце значения имеют разную длину, можно использовать тип *varchar(n).* В отличие от

предыдущего типа данных, размер области хранения для данных этого типа меняется в соответствии с фактическим количеством символов, хранящихся в каждом столбце таблицы, пробелы в конце введенного значения не добавляются.

Тип *text***.** Используется для хранения больших объемов текстовой информации. Для вставки данных в столбец, определенный для данных этого типа, они должны быть заключены в одинарные кавычки.

Числовые (numeric) типы данных. Стандартными для SQL являются следующие типы: *integer, smallint* – для хранения целых чисел; *real* – для хранения положительных или отрицательных дробей с точностью до семи цифр; *float(n)* - для хранения положительных или отрицательных дробей с точностью до пятнадцати цифр.

Типы данных *datetime и smalldatetime*. Они используются для хранения даты и времени. Гораздо удобнее хранить дату и время в формате одного из предназначенных для этого типов данных, а не в виде строки символов. В этом случае дата и время выводятся на экран в привычном формате. Тип datetime позволяет определить дату и время, начиная с 1/1/1753 и заканчивая 12/31/9999; а тип *smalldatetime* – с 1/1/1900 по 6/6/2079.

Перечисленные типы данных позволяют хранить до 90% информации. Кроме этих типов, Transact SQL содержит набор специальных типов данных. Можно определить собственный тип данных - *пользовательский*, который затем будет использоваться для сохраняемых структур.

При назначении типа данных столбцу таблицы возможно использование ограничений *NULL/NOT NULL*, которые позволяют указать, какие из столбцов должны обязательно иметь значения во всех строках таблицы.

1.2 Работа с базой данных

1.2.1 Создание базы данных

Для создания любого объекта SQL Server и, в частности, базы данных существует несколько способов, базирующихся на выполнении определенной команды.

Работа начинается с создания базы данных. Команда создания базы данных *Create Database* имеет следующий синтаксис:

CREATE DATABASE *имя_базы_данных*
ON [PRIMARY]
(NAME = *имя_базы_данных_data*,
FILENAME='*…\имя_базы_данных _data.mdf* ', size = *размер*,
maxsize = *максимальный размер*, filegrowth = *приращение*)
LOG ON
(NAME = *имя_базы_данных _log*,
FILENAME=*'…\имя_базы_данных _log.ldf* ', size = *размер*,
maxsize = *максимальный размер*, filegrowth = *приращение*)

Здесь и далее при описании общего вида команды, размещение опции в квадратных скобках означает, что этот параметр не всегда обязателен. Например, в данном случае параметр PRIMARY определяет файл, содержащий логическое начало базы данных и системных таблиц. В базе данных может быть только один

первичный (PRIMARY) файл. Если этот параметр пропущен, то первичным считается первый файл в списке. По умолчанию файлам типа *primary* присваивается расширение *.mdf*. Опции разделены вертикальной чертой – это означает возможность выбора из двух альтернативных вариантов. Многоточие означает путь.

1.2.2 Удаление базы данных

Удаление базы данных приводит к освобождению всего занимаемого ею пространства во всех файлах, где эта база данных находилась, а также к удалению всех содержащихся в ней объектов:

а) удаление базы данных в графическом режиме предполагает выполнение следующих действий:

- щелкните мышью имя базы данных, которую хотите удалить;
- выберите в контекстном меню команду Delete; в появившемся окне сообщений подтвердите необходимость удаления базы данных.

б) для удаления базы данных с помощью Transact-SQL достаточно выполнить команду

DROP DATABASE *имя_базы_данных*

1.2.3 Создание резервной копии базы данных

Для сохранения копии базы данных, переноса ее на съемный носитель, вначале необходимо создать резервную копию базы данных посредством специальных возможностей MS SQL Server 20xx:

- выберите свою базу данных из списка *Databases* и в контекстном меню строку *Tasks-Back Up*;
- в появившемся окне создания резервной копии вначале очистите список названий баз данных, готовых к копированию;
- кнопкой «Обзор» откройте следующее окно для выбора места на диске, введите имя резервной копии базы данных, подтвердите операцию копирования.

Система должна дать сообщение об успешном копировании.

Надо отметить, что под резервной копией понимается сохранение *данных* базы данных. Если же появилась необходимость переноса *всей*базы данных полностью, то это можно выполнить, скопировав *.mdf* и *.ldf* файлы, которые располагаются по адресу *C:\Program Files\...\Ms SQL\Data\имя_файла*.

1.2.4 Восстановление базы данных

Если появилась необходимость перенести сохраненную резервную копию на компьютер, это тоже выполняется посредством специальных возможностей MS SQL Server 20xx:

- выберите базу данных для проведения операции восстановления;
- в контекстном меню выберите строку *Tasks-Restore;*
- появится страница с общими параметрами восстановления; укажите источник и местоположение резервной копии;

- выберите страницу «Параметры», отметьте флажок «Перезаписать существующую базу данных» и первый переключатель в разделе «Состояние восстановления»;

- подтвердите выбранные операции.

Система сообщит об успешном восстановлении. Если появится сообщение о невозможности восстановления базы данных и система предложит изменить скрипт, необходимо выполнить это требование (в отдельном окне со скриптом).

1.3 Определение структуры базы данных

1.3.1 Описание базы данных, используемой в лабораторной работе

В дальнейшем для освоения работы в среде SQL Server будет рассматриваться база данных компании, занимающейся продажами. Структура базы данных определяется на этапе проектирования. Вопросы проектирования базы данных подробно рассматриваются на лекционных занятиях. Назовем эту базу *TradeCompany*.

База данных *TradeCompany* должна содержать данные о деятельности некоторой торговой фирмы. Фирма продает разнообразные виды товаров. На фирме имеется информация о клиентах фирмы (юридические лица). Для каждой сделки по продажам выписывается счет, в котором отражаются: номер счета, данные о клиенте, перечень и количество проданных товаров, дата продажи.

Предположим, что вся эта информация хранится в таблицах Товары, Клиенты и Счета. Любая таблица имеет *структуру* и хранимые в таблице *данные*. Структура таблицы определяется ее столбцами: их количеством, именем каждого столбца, типом данных, которые хранятся в столбце, и шириной столбца.

Структура таблиц базы данных *TradeCompany* приведена ниже.

Таблица 1.1 - Товары

Код товара	Описание товара	Цена товара

Таблица 1.2 - Клиенты

Код клиента	Название клиента	Адрес клиента	Телефон клиента

Таблица 1.3 - Счета

Номер счета	Код клиента	Код товара	Количество	Дата счета

Данные определяются содержимым строк таблицы. Столбцы в таблице называют *полями*, а строки - *записями*.

Ключевое поле (primary key) – это столбец, данные в котором однозначно идентифицируют каждую строку данных в таблице реляционной базы данных. Задачей ключевого поля является обеспечение уникальности каждой записи. Обычно ключ задается одним столбцом в таблице, но можно задать и сложный ключ на основе комбинации значений нескольких столбцов. Ключ таблице назначается при ее создании. *Внешний ключ(foreign key)* – это столбец в дочерней таблице, ссылающийся на ключ родительской таблицы. Столбец, назначенный внешним ключом, используется для ссылок на столбец, определенный как ключ в другой таблице.

В соответствии с требованиями используемого программного обеспечения для объектов рассматриваемой базы данных будем использовать названия полей на английском языке (сами *данные* можно вводить в кириллице). Что касается назначения имен таблицам, следует отметить следующее. В базе данных, кроме таблиц, может содержаться и множество других объектов. Поэтому за стандарт принято наличие суффикса _TBLв именах таблиц (а, например, суффикс _IDX используется для индексов таблиц). Желательно не только следовать правилам назначения имен, но и правилам, принятым внутри соответствующей области деятельности, чтобы имена носили описательный характер и соответствовали тем данным, на которые эти имена указывают. Также в именах столбцов, состоящих из нескольких слов, в качестве разделителя обычно используется символ подчеркивания. Использование суффиксов при назначении имен объектам базы данных не является обязательным. Выберем названия для таблиц создаваемой базы данных:
CUSTOMER_TBL, PRODUCTS_TBL, ORDERS_TBL.

С учетом изложенного, схему связей между таблицами рассматриваемой базы данных можно представить в следующем виде (рисунок 1.1).

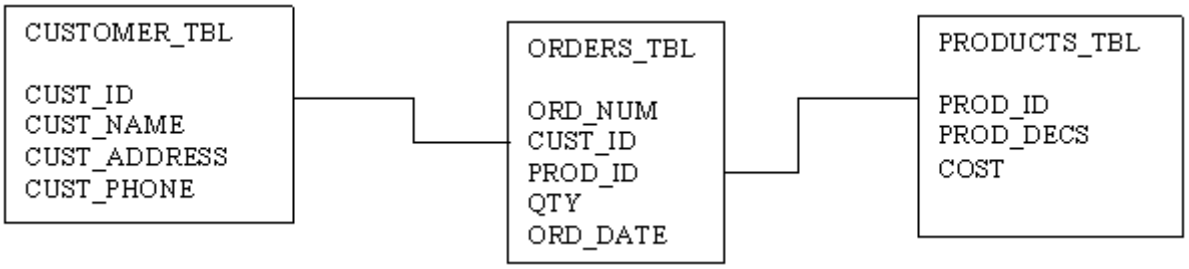

Рисунок 1.1 - Связи между таблицами базы данных *TradeCompany*

Линии, связывающие таблицы, указывают на связи таблиц посредством общего ключевого поля. В представленном варианте в таблице ORDERS_TBL не определен первичный ключ, так как в счете с одним номером может быть несколько товаров (номер счета используется в бухгалтерских документах).

1.3.2 Определение структур базы данных (DDL)

Язык определения данных (DDL) является частью SQL, дающей пользователю возможность создавать различные объекты базы данных и переопределять их структуру, например, создавать или удалять таблицы.

Рассмотрим следующие команды DDL: CREATE TABLE, ALTER TABLE, DROP TABLE.

Чтобы создавать таблицы, используется оператор CREATE TABLE.

Синтаксис оператора для создания таблиц:

CREATE TABLE *имя_таблицы*
(ПОЛЕ1 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ2 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ3 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ4 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ5 ТИП ДАННЫХ [NOT NULL])

Ключ таблице назначается при ее создании с помощью опции PRIMARY KEY одному или нескольким полям и является по своей сути ограничивающим условием:

CREATE TABLE *имя_таблицы* PRIMARY KEY
(ПОЛЕ1 ТИП ДАННЫХ [NOT NULL], …)

Можно создать ключ и непосредственно как ограничивающее условие через запятую после определения всех столбцов таблицы, причем, если ключ является составным, перечисляются все его компоненты:

CREATE TABLE *имя_таблицы*
(ПОЛЕ1 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ2 ТИП ДАННЫХ [NOT NULL], …
PRIMARY KEY (ПОЛЕ1, ПОЛЕ2))

Внешний ключ создается с помощью опции FOREIGN KEY. Внешний ключ создается после определения всех столбцов таблицы следующим образом:

CREATE TABLE *имя_таблицы_1*
(ПОЛЕ1_1 ТИП ДАННЫХ [NOT NULL],
ПОЛЕ1_2 ТИП ДАННЫХ [NOT NULL],
…,
ПОЛЕ2_1 ТИП ДАННЫХ [NOT NULL],
CONSTRAINT ПОЛЕ2_1_FK FOREIGN KEY (ПОЛЕ2_1)
REFERENCES *имя_таблицы_2 (*ПОЛЕ2_1))

Столбец ПОЛЕ2_1 здесь назначается внешним ключом таблицы *имя_таблицы_1*. Этот внешний ключ ссылается на столбец ПОЛЕ2_1 таблицы *имя_таблицы_2*.

1.4 Модификация таблицы

После создания таблицу можно модифицировать и с помощью команды ALTER TABLE. С помощью этой команды можно добавлять и удалять столбцы, менять определения столбцов, добавлять и удалять ограничения.

Стандартный синтаксис команды ALTER TABLE следующий:

ALTER TABLE *имя_таблицы* [MODIFY] [COLUMN *имя_столбца*]
[ТИП ДАННЫХ|NULL NOT NULL] [RESTRICT|CASCADE]
[DROP] [CONSTRAINT *имя_ограничения*]
[ADD] [COLUMN] *определение столбца*

а) *модификация элементов таблицы.* Атрибуты столбца задают правила представления данных в столбце. С помощью команды ALTER TABLE можно менять атрибуты столбца. Под атрибутами здесь понимается следующее: тип данных в столбце; длина, точность или масштаб данных в столбце; разрешение или запрет иметь в столбце значение NULL.

При добавлении столбца в уже существующую таблицу с имеющимися в ней данными новому столбцу нельзя назначить атрибут NOT NULL. NOT NULL означает, что столбец должен содержать значения для каждой строки в таблице, так что, если добавляемый столбец получит атрибут NOT NULL, вы сразу же получите противоречие с этим ограничением, поскольку имеющиеся в таблице столбцы не имеют значений для нового столбца. И все же имеется

возможность добавить столбец, требующий обязательного ввода данных, следующим образом:

- добавьте столбец, задав ему атрибут NULL (это значит, что в столбце не обязательно должны присутствовать данные);
- введите данные в каждую строку нового столбца таблицы;
- убедившись, что столбец содержит значение в каждой из строк таблицы, можно изменить атрибут столбца на NOT NULL;

б) *изменение столбцов*. При изменении столбцов таблиц нужно учитывать целый ряд моментов. Общие правила следующие:

- ширина столбца может быть увеличена до максимальной длины, разрешенной для соответствующего типа данных;
- ширину столбца можно уменьшить только до наибольшей длины имеющихся в этом столбце значений;
- для столбцов с числовыми данными ширину всегда можно увеличить;
- для столбцов с числовыми данными ширину можно уменьшить только тогда, когда нового числа знаков будет достаточно для размещения любого из имеющихся в столбце значений;
- для числовых данных можно увеличивать или уменьшать число десятичных знаков;
- тип данных в столбце обычно можно изменить.

В некоторых реализациях использование определенных опций оператора ALTER TABLE может быть запрещено. Например, вам могут не позволить удалять столбцы из таблиц. Вместо этого вам нужно будет удалить таблицу и создать новую с нужным числом столбцов. Могут возникнуть проблемы с удалением столбцов из таблицы, зависящей от столбца из другой таблицы, или с удалением столбца, на который ссылается другая таблица. По этому поводу внимательно просмотрите документацию, предлагаемую той реализацией SQL, с которой вы работаете;

в) *добавление ограничений.* Эта ситуация может возникнуть, например, в случае, когда при создании таблицы не были определены ключевые поля:

ALTER TABLE *имя_таблицы*
ADD CONSTRAINT *имя_таблицы*_PK
PRIMARY KEY (имя_поля1, имя_поля2)

г) *внешние ключи* можно назначить таблице следующим образом:

ALTER TABLE *имя_таблицы*
ADD CONSTRAINT ID_FK FOREIGN REY (*имя_поля*)
REFERENCES *имя_таблицы (имя_поля)*

д) *удаление таблиц* является, пожалуй, самым простым делом. Синтаксис оператора, используемого для удаления таблиц, следующий:

DROP TABLE *имя_таблицы* [RESTRICT|CASCADE]

Если используется опция RESTRICT, либо на таблицу ссылается представление или ограничение, используемый для удаления оператор DROP возвратит ошибку. При использовании опции CASCADE будет выполнено удаление не только самой таблицы, но и всех ссылающиеся на таблицу представлений и ограничений.

1.5 Задание на лабораторную работу

Прежде чем приступить к выполнению задания, следует ознакомиться с соответствующими разделами виртуальной обучающей системы*SQL_Education*.

1.5.1 Создание базы данных. *TradeCompany*:

а) создание базы данных в графическом режиме.

При этом способе выполняются следующие действия:

- запустите программу MS SQL Server 20xx посредством основного меню: *Пуск-Все программы- MS SQL Server 20xx-SQL Server Management Studio;*

- в появившемся диалоговом окне предлагается ввести имя сервера, оно установлено по умолчанию, щелкните на *Connect*;

- в следующем окне *Object Explorer* на строке *Databases* из контекстного меню выберите *New Database*;

- появится окно, в котором надо ввести имя создаваемой базы данных и ОК; в нижней части этого окна в таблице располагается информация о файлах базы данных (разверните окно, чтобы видеть все колонки таблицы); здесь кнопка *Add* предназначена для случая, если база данных располагается не в одном файле;

- закройте окно; убедитесь, что ваша база появилась в списке *Database*;

- удалите базу данных (команда *Delete* из контекстного меню);

б) создание базы данных посредством команды *Transact-SQL*.

В окне *Object Explorer* щелкните на кнопку *New Query*. В правой части экрана появится окно редактора запросов. Введите в него команду на создание базы данных. Если команда выполнилась успешно, в нижней части окна должно появиться сообщение: *Command (s) completed successfully*.

В отличие от графического режима, чтобы увидеть в списке баз данных только что созданную базу, необходимо на строке *Database* из контекстного меню выбрать *Refresh*.

1.5.2 Создайте резервную копию базы данных. Запомните путь, где сохранилась резервная копия.

1.5.3 Выполните восстановление базы данных.

1.5.4 Обоснованно выберите типы данных для полей таблиц рассматриваемой базы данных *TradeCompany*. Определите первичные и внешние ключи таблиц.

1.5.5 Активизируйте MS SQL Server 2005; сделайте текущей базу данных *TradeCompany*.

1.5.6 Создание и модификация таблиц базы данных *TradeCompany*.

Таблицы тоже можно создавать несколькими способами:

а) для создания таблицы в графическом режиме необходимо выполнить следующее (создайте таким способом таблицу CUSTOMER_TBL.):

- откройте папку своей базы данных; отметьте строку *Tables* и в контекстном меню выберите *New Table*;

- в появившемся окне введите наименования столбцов таблицы, выберите тип данных, установите (или уберите) флажок *Allows Null*; внизу окна установите свойства столбцов;

- закройте текущее окно, подтвердите сохранение изменений, введите имя таблицы;

б) нажмите в окне *Object Explorer* кнопку *New Query*, откроется окно редактора команд, вводя в это окно команды создания таблицы (создайте таким способом таблицы PRODUCTS_TBL, ORDERS_TBL), нажимая кнопку *Execute* (выполнение команды), создайте еще одну таблицу;

в) чтобы изменить структуру таблицы, выбираем из контекстного меню *Modify Table*;

г) для быстрого изучения команд SQL, выделите таблицу базы данных, выберите в контекстном меню *Script Table As/ … /New Query Editor Window* (вместо многоточия выберите требуемую команду), появится окно редактора команд, в котором можно просмотреть заготовки изучаемых скриптов на создание, удаление, обновление и т.д.

1.5.7 Просмотрите диаграмму связей созданной базы данных (раздел *Diagrams*).

1.6 Требования к отчету

Отчет по работе выполняется на бумажном носителе и должен содержать:

- описание вариантов создания базы данных;
- обоснование выбора типов данных для полей таблиц;
- определение первичных и внешних ключей таблиц;
- команды создания структур таблиц.

1.7 Контрольные вопросы

1.7.1 Что означает аббревиатура SQL?

1.7.2 Перечислите типы команд SQL.

1.7.3 Какие способы создания базы данных вы знаете, в чем их отличия?

1.7.4 Сколько файлов используется для базы данных?

1.7.5 Для чего предназначен журнал транзакций?

1.7.6 Объясните создание резервной копии и восстановление базы данных.

1.7.7 Объясните способы удаления базы данных.

1.7.8 Дайте определение реляционной базы данных.

1.7.9 В каких объектах хранится исходная информация базы данных?

1.7.10 Что называется полем таблицы, записью таблицы?

1.7.11 Какая команда определяет структуру таблицы?

1.7.12 Дайте определение первичного ключа таблицы.

1.7.13 Объясните назначение внешних ключей.

1.7.14 Какие основные типы данных вам известны?

1.7.15 С помощью какой команды можно модифицировать структуру таблицы?

1.7.16 Как можно удалить таблицу?