**Лабораторная работа 5**

**Язык Transact-SQL. Процедуры и функции.**

**Цель работы**

Создать базу данных из несколько таблиц используя графический интерфейс MS SQL SERVER. Использовать ограничения целостности и внешние ключи.

**Содержание работы и методические указания к ее выполнению**

*Замечание*. Для получения справки по командам Transact-SQL и хранимым процедурам можно воспользоваться утилитой *SQL Server Management Studio*. Для этого необходимо выделить имя оператора и нажать клавишу **F1**.

1. **Создайте собственную базу данных**, размер основного файла которой 5 Мб, размер журнала – 2 Мб. Основной файл может увеличиваться до 10 Мб c шагом 20%. Объем файла журнала увеличивается до 5 Мб с шагом 1 Мб. Расположение файлов на диске – ***D:\MSSQL\DATA*.** Написать код команды. Убедитесь, что база данных создана должным образом.
2. **Переименуйте созданную Вами базу данных.** Имя БД **A\_ХХ\_ГГ\_YY** , (где XX-номер группы, ГГ-год группы, YY-номер в журнале).
3. **Создайте собственную файловую группу**.
4. **Создать пользовательские типы данных для кодов курсов CourseNumber (на основе** базового типа данных varchar, с максимальной длиной равной 6 символов) и денежных столбцов **CashValue** (на основе базового типа данных decimal, с точностью равной 10 и масштабом равным 2).
5. **Создать таблицы Teachers, Cources, Students**
6. Информация о преподавателях будет храниться в таблице **Teachers**.

Таблица должна иметь следующие столбцы **id,** **Фамилия, Имя, Отчество** в которых должны храниться символьные строки переменной длины в формате Юникод, длина строк должна быть не более 50 символов.

1. Информация о курсах организации будет храниться в таблице **Cources.**

Таблица должна иметь следующие столбцы **id, Код (Code) типа CourseNumber , Название (name) и Краткое описание (shotDiscr)** - символьные строки переменной длины в формате Юникод, длиной не более 256 и 2000 символов соответственно. Создать ограничение целостности первичный ключ для таблицы Cources

1. ***З*аполните обе таблицы**. В конце фамилии добавляем свой номер и номер группы (Пример: Иванов1307, где 13-номер в журнале, 07 – номер группы).
2. **Получите список сотрудников с указанием фамилии и первого символа имени, а также его идентификационного номера.** Список должен быть упорядочен по фамилии и имени сотрудника. Первый столбец переименовать, так как в нем необходимо объединить данные из двух столбцов.
3. **Добавить таблицу для хранения информации о слушателях курсов с именем Students.**

Данная таблица должна иметь следующие столбцы:

Id – целочисленный 4 байтовый первичный ключ.

FirstName – символьная строка переменной длины в формате Юникод, длина строки должна быть не более 50 символов.Значения NULL не допускаются.

LastName – символьная строка переменной длины в формате Юникод, длина строки должна быть не более 50 символов.Значения NULL не допускаются.

MiddleName – символьная строка переменной длины в формате Юникод, длина строки должна быть не более 50 символов.Допускаются значения NULL.

BirthDate – датарождения.Значения NULL не допускаются.При вставке и изменении данных, **должна происходит проверка, что слушатель курсов старше 12 лет**.

RegistrationDate – дата регистрации слушателя. Значения NULL не допускаются. В случае, если дата регистрации не задана, **должна использоваться текущая дата**.

1. Создать таблицу **Contracts** для хранения информации о заключенных договорах.

Таблица содержит поля **– ContractId, StudentId, CourceId, дату заключения договора и размер оплаты типа CashValue**.

**Добавить ограничения целостности внешний ключ для таблицы Contracts:**

**StudentId** ссылается на столбец **Id** таблицы **Students**. При обновлении и удалении записей из таблицы **Students** не должно происходит никаких изменений связанных записей таблицы **Contracts.**

**Проверить, что при произойдет при удалении студента, для которого есть контракт.**

**CourceId** ссылается на столбец **Id** таблицы **Cources**. При обновлении и удалении записей из таблицы **Cources** не должно происходит никаких изменений связанных записей таблицы **Contracts**

1. **Заполните таблицы из текстового файла с помощью xp\_cmdshell и bcp (**EXEC master..xp\_cmdshell @cmd;**).**
2. **Написать курсор для удаления записей из таблицы Teachers с четным значением id.**
3. **Включить в 1 транзакцию:**

**операцию удаления первой записи из таблицы Teachers.**

**операцию обновление второй записи из таблицы Courses.**

**операцию вставки в третьей таблице Students.**

15. Добавить конструкцию BEGIN TRY...END TRY и BEGIN CATCH...END CATCH в транзакцию из пункта 14 и предусмотреть ее откат если возникает ошибка.

**Методические указания к работе 5**

**Ограничение CHECK**

Ограничение целостности check является наиболее часто используемым видом ограничения. **Оно позволяет задать для определённой колонки выражение, которое будет осуществлять проверку, помещаемого в эту колонку значения.** Если значение удовлетворяет, заданному ограничению, то выражение должно возвращать Логическое значение (истина).

Определение ограничения следует после указания типа данных, как и определение значения по умолчанию. Значения по умолчанию и ограничения могут следовать в произвольном порядке. Ограничение check состоит из ключевого слова CHECK, за которым следует выражение в круглых скобках. **Выражение в ограничении check должно использовать ту колонку, на которую оно накладывается**, в противном случае это ограничение не будет иметь смысла.

Пример 1,

Требуется, чтобы в колонке были только положительные цены товаров. Можно использовать:

CREATE TABLE products

( product\_no integer,

name text,

price numeric **CHECK (price > 0)** );

Вы можете дать ограничению конкретное имя. Это сделает более понятными сообщения об ошибках и позволит вам ссылаться на это ограничение, когда вам понадобится его изменить. Синтаксис будет такой:

Пример 2

CREATE TABLE products

( product\_no integer,

name text,

price numeric **CONSTRAINT positive\_price** CHECK (price > 0) );

Таким образом, для создания именованного ограничения, используйте ключевое слово **CONSTRAINT,** за которым следует индентификатор (имя), после которого следует, собственно, определение ограничения. (Если при использовании этого синтаксиса вы не укажете имя ограничения, СУБД выберет имя за вас.)

**Ограничение check может также накладываться на несколько колонок.**

Пример 3:

Допустим, что вы храните обычную цену и цену со скидкой и хотите иметь уверенность в том, что цена со скидкой меньше, чем обычная цена.

CREATE TABLE products (

product\_no integer,

name text,

price numeric CHECK (price > 0),

discounted\_price numeric CHECK (discounted\_price > 0),

**CHECK (price > discounted\_price)** );

Первые два ограничения нам уже знакомы. Третье использует новый синтаксис. Оно не привязано к определённой колонке, а появляется как отдельный элемент в списке колонок, разделённых запятой. Определения колонок и такие определения ограничений могут следовать в смешанном порядке.

**Пользовательские типы данных.**

Пользовательские типы данных, также именуемые как псевдонимы типов данных, позволяют **расширить базовые типы данных SQL** Server (например, **varcha**r) содержательным именем и форматом, который можно приспособить для конкретного использования.

**CREATE TYPE**:

CREATE TYPE type\_name

{ FROM base\_type

[ ( precision [ , scale ] ) ]

[ NULL | NOT NULL ] }

**type\_name *-* и**мя определяемого пользователем типа данных.

**base\_type *-*** тип данных SQL Server, на основе которого создается.

**Использование cmdshell**

чтобы можно было запустить команды в оболочке Windows необходимо выполнить реконфигурацию сервера SQL

-- To allow advanced options to be changed.

EXEC sp\_configure 'show advanced options', 1;

GO

-- To update the currently configured value for advanced options.

RECONFIGURE;

GO

-- To enable the feature.

EXEC sp\_configure 'xp\_cmdshell', 1;

GO

-- To update the currently configured value for this feature.

RECONFIGURE;

GO

**Пример написания курсора**

DECLARE @firm VARCHAR(50),

@fam VARCHAR(50),

@message VARCHAR(80)

PRINT ' Список клиентов'

DECLARE klient\_cursor CURSOR LOCAL FOR

SELECT Фирма, Фамилия

FROM Клиент

WHERE Город='Москва'

ORDER BY Фирма, Фамилия

OPEN klient\_cursor

FETCH NEXT FROM klient\_cursor INTO @firm, @fam

WHILE @@FETCH\_STATUS=0

BEGIN

SELECT @message='Клиент '+@fam+

' Фирма '+ @firm

PRINT @message

-- переход к следующему клиенту--

FETCH NEXT FROM klient\_cursor

INTO @firm, @fam

END

CLOSE klient\_cursor

DEALLOCATE klient\_cursor

**Контрольные вопросы**

1. Может ли база данных существовать без файла журнала транзакций?
2. Можно ли создать дополнительные файлы журналов транзакций?
3. Из каких логических единиц состоят файлы журнала транзакций?
4. Можно ли сжать базу данных до размера меньше первоначального?
5. Что такое именованное ограничение?
6. Какие типы данных используются в MS SQL SERVER&
7. Как создать пользовательский тип данных?