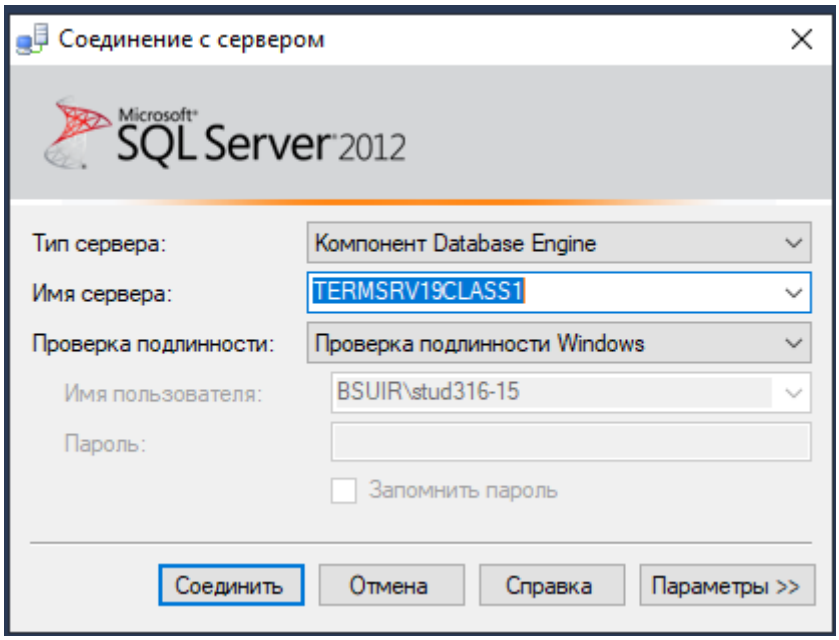
Лабораторная работа № 8.

СОЗДАНИЕ ТАБЛИЦ И ИНДЕКСОВ СРЕДСТВАМИ ЯЗЫКА ОПРЕДЕЛЕНИЯ ДАННЫХ - DDL

Для начала работы запустите среду Microsoft SQL Server Management Studio известным Вам способом и установите соединение с Вашим сервером СУБД. Например:



Изученным ранее способом создайте новую БД с названием «Номер_группы_Фамилия_ЛР_8».

Прочитайте теоретический материал и выполните задания для самостоятельной работы (в конце) с оформлением отчета и обязательными скриншотами результатов.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ ДЛЯ ИЗУЧЕНИЯ

Создание таблиц

Таблицы создаются командой CREATE TABLE.
Ввести данные в нее можно при помощи команды INSERT. Синтаксис команды CREATE TABLE:

CREATE TABLE
[имя_*базы_данных*.[*владелец*] .] *имя_таблицы*
(<имя_столбца> <тип_данных>

[[DEFAULT <выражение_константа>]

| [IDENTITY [(*начальное_значение*, *приращение)* [NOT FOR REPLICATION]]]]

[ROWGUIDCOL]

[COLLATE < способ_сравнения >]
[NULL | NOT NULL]

[< ограничения_столбца>]

| [*имя_столбца* AS выражение_для_вычисляемого_столбца] | < табличные_ограничения>] [, ...*n]*)

DEFAULT – значение по умолчанию.

IDENTITY – автоматическое присваивание значений (начальное + приращение).
NOT FOR REPLICATION – нужно ли при вставке строк в новую базу данных (путем репликации) присваивать строке новое identity-значение или использовать существующее.

ROWGUIDCOL – применяется для уникальной идентификации строк в таблице. Отличие от identity-значений состоит в длине идентификатора. Вместо использования цифрового счетчика SQL прибегает к помощи GUID (Global Unique Identifier – глобально уникальный идентификатор), уникального во времени и в пространстве.

COLLATE – управление порядком сортировки и правилами сравнения для отдельных столбцов.

[NULL | NOT NULL] – часто необходимо предохранить поле от ввода в него значений NULL. С этой целью используют ограничение NOT NULL.

[< ограничения_столбца>]: [CONSTRAINT *имя_ограничения*]
{ [NULL | NOT NULL]
| [{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
[WITH FILLFACTOR = *fillfactor*]
[ON {*filegroup* | DEFAULT}]]
]
| [[FOREIGN KEY]
REFERENCES *имя_таблицы*[**(***имя_столбца* **)**]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]
]
| CHECK [NOT FOR REPLICATION]

(условие)
}

Поля, значения которых требуют уникальности, но не являются первичными ключами, называют *ключами – кандидатами.* В SQL можно определить такие поля при помощи команды – ограничения UNIQUE. Обычно определяется группа из таких полей. Например, в таблице USP пара значений: номер студенческого билета и код предмета – не должны повторяться. Причем на эти поля необходимо накладывать ограничение NOT NULL.
С помощью ***PRIMARY KEY*** определяется ключевое поле. Ограничение ***NOT NULL*** обязательно.
Ограничение ***CHECK*** позволяет поставить условие, в соответствие с которым проверяется вводимое в таблицу значение.

Приведем пример создания таблицы primer, с полями P1 – integer; P2 – integer; P3- char(20) . Причем, поле P1 является первичным ключом, для поля P2 должно соблюдаться условие уникальности и необходимо предусмотреть ограничение на ввод значений 2,3,4,5; значение по умолчанию 5.

```
CREATE TABLE primer (
p1 INTEGER NOT NULL PRIMARY KEY,
p2 INTEGER  CHECK (p2 IN(2,3,4,5))  DEFAULT (5),
 p3 CHAR(20) NOT NULL UNIQUE)
```

Ограничение внешнего ключа используется как для обеспечения целостности, так и для задания отношений между таблицами. Столбцы, на которые производится ссылка, должны иметь ограничение первичного ключа, либо ограничение уникальности.
Каскадное обновление и удаление можно разрешить или запретить.
ON UPDATE NO ACTION – для ситуации с обновлением записей родительской таблицы воздержаться от каскадного обновления записей.
ON DELETE CASCADE – каскадное удаление.

После того как таблица была создана, ее можно изменить при помощи команды ALTER TABLE. Синтаксис этой команды следующий:

```
 ALTER TABLE <table name>
ADD <column name> <data type> [(size)];
```

Новое поле станет последним в таблице. Можно добавить сразу несколько полей, отделив их запятыми. Изменение структуры таблицы в момент ее использования чревато потерей информации.
При помощи этой же команды можно удалить созданный ранее столбец:

```
ALTER TABLE <table name>
DROP COLUMN <column name>;
```

Синтаксис команды для удаления таблицы:

DROP TABLE < table name>;

Создадим первичный ключ для существующей таблицы primer (предварительно отказавшись от ранее созданного в качестве первичного ключа p1):

ALTER TABLE primer
ADD CONSTRAINT PK_primerID
PRIMARY KEY (p3)

Добавим внешний ключ в таблицу Orders, чтобы ограничить значение поля EmployeeID (в котором храниться идентификатор служащего, заполнившего заказ) допустимыми значениями из таблицы Employees. Для реализации этой задачи необходимо уникально идентифицировать целевые записи в таблице, на которую будет производиться ссылка. Это осуществимо путем задания ссылки на первичный ключ либо на столбец с ограничением уникальности (воспользуемся существующим первичным ключом EmployeeID из таблицы Employees).

ALTER TABLE Orders
ADD CONSTRAINT FK_EmployeeCreatesOrder
FOREIGN KEY (Employee) REFERENCES Employees (Employee)

При добавлении в таблицу внешнего ключа, вы создаете зависимость между таблицей, для которой вы определили внешний ключ и таблицей, на которую ссылается *внешний ключ.* После создания внешнего ключа, любая запись, добавляемая в ссылочную таблицу, должна иметь соответствующую родительскую запись либо значения для столбцов внешнего ключа должны быть установлены в NULL.

Запустив хранимую процедуру *sp_helpconstraint* командой:
EXEC *sp_helpconstrain t <имя таблицы>*
можно получить сведения об именах, статусе и критериях всех ограничений данной таблицы.

Создание ограничений уникальности (альтернативных ключей) для существующих таблиц

ALTER TABLE primer
ADD CONSTRAINT AK_primerp3
UNIQUE (p3)

Ограничения проверки. Их применение ограничивается отдельными столбцами.

```
ALTER TABLE Customers
ADD CONSTRAINT CN_CustomerDate
CHECK
(DATE <=GETDATE ())
```
Невозможность ввода даты позже сегодняшней

Все записи в SQL вводятся с использованием команды модификации INSERT. В общем случае синтаксис команды следующий:

```
INSERT INTO <table name> (<column name>, <column name>…)
VALUES (<value>, <value>,….);
```

Если требуется внести NULL значение, то вместо значения поля пишется NULL. Можно также использовать команду INSERT для того, чтобы выбирать значения из одной таблицы и помещать их в другую (предложение VALUE заменяется на соответствующий запрос). Например, в таблицу primer1 необходимо вставить данные из таблицы primer, в которой значение поля p2=3:

```
INSERT INTO primer1
SELECT *
FROM primer
WHERE P2=3;
```

Таблица prime1 предварительно должна быть создана командой CREATE TABLE и иметь одинаковое количество столбцов с таблицей primer, которые совпадают по типу.

Удаление строк из таблицы осуществляется командой DELETE. В этой команде разрешается использовать предикат, выбирающий группу строк. Например:

```
DELETE FROM primer1
WHERE p2=4;
```

Изменение существующих данных производится командой UPDATE.
Например, чтобы изменить значение поля P2 на 3 необходимо выполнить команду:

```
UPDATE primer1
SET p2=3;
```
Можно также модифицировать данные из нескольких полей, указанных через запятую и использовать предикат для отбора.

Пользовательские типы данных создаются при помощи системной процедуры:
sp_addtype user_datatype_name, system_datatype, null|not null
user_datatype – имя типа;
system_datatype – стандартный тип данных, берется в ' '.

```
USE master
EXEC sp_addtype birthday, 'datetime', 'NULL'
```

Удаление пользовательского типа данных

sp_droptype [@typename =] 'type'

```
USE master
EXEC sp_droptype 'birthday'
```

1.7 Создание индексов

Синтаксис команды для создания индекса следующий:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX имя
   ON { имя_таблицы | имя_представления } ( имя_столбца [ ASC | DESC ] [ ,...n ] )
[ WITH

[PAD_INDEX]

[ [,] FILLFACTOR = <коэффициент_заполнения>]
[ [,] IGNORE_DUP_KEY ]

[ [,] DROP_EXISTING ]

[ [,] STATISTICS_NORECOMPUTE ]
[ [,] SORT_IN_TEMPDB ] ]

[ ON <группа файлов> ]
```

PAD_INDEX – Уровень заполнения листов дерева (в процентах) при первом создании индекса. Используется совместно с FILLFACTOR.

IGNORE_DUP_KEY – Не вызывает отката транзакций, нарушающих требование уникальности, при работе с ограничениями уникальности.

DROP_EXISTING – В случае существования индекса с таким же именем, существующий будет удален перед началом создания нового.

STATISTICS_NORECOMPUTE – Отключение автоматического обновления статистики по индексу (не рекомендуется).

SORT_IN_TEMPDB – Применяют, если БД tempdb хранится на отдельном физическом носителе. Если не использовать, то временные страницы будут записываться на тот же физический носитель, что и сама БД.

ON <группа файлов> – Позволяет хранить индексы отдельно от основных данных.

Например,

```
IF EXISTS (SELECT name FROM sysindexes
      WHERE name = 'au_id_ind')
   DROP INDEX authors.au_id_ind
GO
USE pubs
CREATE INDEX au_id_ind
   ON authors (au_id)
```

Создадим индекс для поля P3:

```
CREATE INDEX IN_P3 ON primer (p3);
```

Для создания уникальных индексов используют ключевое слово UNIQUE:

```
CREATE UNIQUE INDEX IN_P3 ON primer (p3);
```

Для удаления индекса существует команда DROP:

```
DROP INDEX <имя_индекса>;
```

Задания для выполнения:

Создание таблиц при помощи SQL – DDL.

Все задания необходимо выполнить средствами языка SQL – DDL. Запросы сохранить и распечатать в отчет.

⇒ **В базе данных, созданной в ЛР-2**, создайте структуру таблицы **STUDENTS_N** (N – Ваш номер по журналу) с полями SNUM, SFAM, SIMA, SOTCH, STIP, соответствующими ключевыми полями и подстановкой значения по умолчанию для поля STIP.

⇒ Измените структуру данной таблицы, добавив поля: COURS INT, TELEPHONE CHAR (15).

⇒ Удалите поля COURS INT, TELEPHONE

⇒ Создайте пользовательский тип данных TELEPHONE.

Пользовательский тип данных TELEPHONE должен представлять собой текстовое поле переменной длинны, максимально содержащее 20 символов, обязательное для заполнения.

⇒ Создайте индекс для поля SFAM.

⇒ Создайте остальные таблицы, соответствующие БД, разработанной в ЛР-2, но с именами по примеру **STUDENTS_N,** с необходимыми ключевыми полями и подстановками (для оценки в таблице успеваемости). Ключевые поля определить после создания таблиц командой ALTER.

⇒ С помощью оператора INSERT и вложенного запроса заполните таблицу STUDENTS_N данными только для отличников из таблицы STUDENTS из ЛР-2, (признаком является наличие **только** отличных оценок в таблице USP).

⇒ Заполните данными остальные созданные таблицы (с именами *_N) из соответствующих таблиц БД из ЛР-2, добавьте неотличников в STUDENTS_N.

⇒ Выполните добавление внешних ключей для обеспечения связей между таблицами. Предусмотреть возможность каскадного обновления и удаления.

⇒ Добавьте ограничение проверки для поля UDATE таблицы USP_N для проверки корректности вводимой даты (нельзя ввести будущую дату);

⇒ В таблице PREDMET_N для поля PNAME добавьте ограничение проверки для ограничения списка только этими предметами:

Физика
Химия
Математика
Философия
Экономика

⇒ Оформите отчет и представьте преподавателю.