**ПРАКТИЧЕСКАЯ РАБОТА №7**

**Тема: Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами**

**Цель:** формирование практических знаний и умений создания ключевых полей, задания индексов, установления и удаления связей между таблицами.

**Задание 1. Создание ключевых полей. Задание индексов.**

* 1. Изучить теоретический материал.
  2. Создать БД internetstore\_ФИО (вместо ФИО подставить свою фамилию), используя **T-SQL.**
  3. Создать таблицу Customers (Id, Age, FirstName, LastName, Email, Phone) и Orders (Id, CustomerId, CreatedAt). В каждой таблице создать ключевое поле, используя T-SQL.

Определены таблицы Customers и Orders. Customers является главной и представляет клиента. Orders является зависимой и представляет заказ, сделанный клиентом. Эта таблица через столбец CustomerId связана с таблицей Customers и ее столбцом Id. То есть столбец CustomerId является внешним ключом, который указывает на столбец Id из таблицы Customers.

**Задание 2. Установление и удаление связей между таблицами**

* 1. Установить связь таблицы Customers и Orders. Определить внешний ключ на уровне таблицы.
  2. Удалить связь между таблицами.
  3. Установить связь между таблицами с помощью оператора CONSTRAINT задать имя для ограничения внешнего ключа FK\_Orders\_To\_Customers

**Задание 3. Изменение таблицы**

* 1. Добавить новый столбец Address в Customers..
  2. Удалить столбец Address из таблицы Customers.
  3. Изменить в таблице Customers тип данных у столбца FirstName на NVARCHAR(200)
  4. С помощью ALTER TABLE добавить внешний ключ и первичный ключ
  5. Как удалить ограничения?

**Контрольные вопросы:**

1. Как создать первичный ключ?
2. Что такое внешний ключ? Как создать внешний ключ? Как определить имя ограничения для внешнего ключа?
3. Какие действия можно установить с помощью ON DELETE и ON UPDATE? Перечислить опции и их назначение.
4. Общий формальный синтаксис команды ALTER TABLE. Как с помощью ALTER TABLE добавить внешний ключ и первичный ключ?
5. Смысл разделения SQL-выражений на пакеты? Команда GO.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Внешние ключи применяются для установки связи между таблицами. Внешний ключ устанавливается для столбцов из зависимой, подчиненной таблицы, и указывает на один из столбцов из главной таблицы. Хотя, как правило, внешний ключ указывает на первичный ключ из связанной главной таблицы, но это необязательно должно быть непременным условием. Внешний ключ также может указывать на какой-то другой столбец, который имеет уникальное значение.

Общий синтаксис установки внешнего ключа на уровне столбца:

|  |  |
| --- | --- |
|  | [FOREIGN KEY] REFERENCES главная\_таблица (столбец\_главной\_таблицы)  [ON DELETE {CASCADE|NO ACTION}]  [ON UPDATE {CASCADE|NO ACTION}] |

Для создания ограничения внешнего ключа на уровне столбца после ключевого слова **REFERENCES** указывается имя связанной таблицы и в круглых скобках имя связанного столбца, на который будет указывать внешний ключ. Также обычно добавляются ключевые слова **FOREIGN KEY**, но в принципе их необязательно указывать. После выражения REFERENCES идет выражение **ON DELETE** и **ON UPDATE**.

Общий синтаксис установки внешнего ключа на уровне таблицы:

|  |  |
| --- | --- |
|  | FOREIGN KEY (стобец1, столбец2, ... столбецN)  REFERENCES главная\_таблица (столбец\_главной\_таблицы1, столбец\_главной\_таблицы2, ... столбец\_главной\_таблицыN)  [ON DELETE {CASCADE|NO ACTION}]  [ON UPDATE {CASCADE|NO ACTION}] |

Например, определим две таблицы и свяжем их посредством внешнего ключа:

|  |  |
| --- | --- |
|  | CREATE TABLE Customers  (      Id INT PRIMARY KEY IDENTITY,      Age INT DEFAULT 18,      FirstName NVARCHAR(20) NOT NULL,      LastName NVARCHAR(20) NOT NULL,      Email VARCHAR(30) UNIQUE,      Phone VARCHAR(20) UNIQUE  );  CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT REFERENCES Customers (Id),      CreatedAt Date  ); |

Здесь определены таблицы Customers и Orders. Customers является главной и представляет клиента. Orders является зависимой и представляет заказ, сделанный клиентом. Эта таблица через столбец CustomerId связана с таблицей Customers и ее столбцом Id. То есть столбец CustomerId является внешним ключом, который указывает на столбец Id из таблицы Customers.

Определение внешнего ключа на уровне таблицы выглядело бы следующим образом:

|  |  |
| --- | --- |
|  | CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt Date,      FOREIGN KEY (CustomerId)  REFERENCES Customers (Id)  ); |

С помощью оператора **CONSTRAINT** можно задать имя для ограничения внешнего ключа. Обычно это имя начинается с префикса "FK\_":

|  |  |
| --- | --- |
|  | CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt Date,      CONSTRAINT FK\_Orders\_To\_Customers FOREIGN KEY (CustomerId)  REFERENCES Customers (Id)  ); |

В данном случае ограничение внешнего ключа CustomerId называется "FK\_Orders\_To\_Customers".

### ON DELETE и ON UPDATE

С помощью выражений **ON DELETE** и **ON UPDATE** можно установить действия, которые выполняться соответственно при удалении и изменении связанной строки из главной таблицы. И для определения действия мы можем использовать следующие опции:

* **CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
* **NO ACTION**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.
* **SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL.
* **SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение NULL.

### Каскадное удаление

По умолчанию, если на строку из главной таблицы по внешнему ключу ссылается какая-либо строка из зависимой таблицы, то мы не сможем удалить эту строку из главной таблицы. Вначале нам необходимо будет удалить все связанные строки из зависимой таблицы. И если при удалении строки из главной таблицы необходимо, чтобы были удалены все связанные строки из зависимой таблицы, то применяется каскадное удаление, то есть опция **CASCADE**:

|  |  |
| --- | --- |
|  | CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt Date,      FOREIGN KEY (CustomerId) REFERENCES Customers (Id) ON DELETE CASCADE  ) |

Аналогично работает выражение **ON UPDATE CASCADE**. При изменении значения первичного ключа автоматически изменится значение связанного с ним внешнего ключа. Но так как первичные ключи, как правило, изменяются очень редко, да и с принципе не рекомендуется использовать в качестве первичных ключей столбцы с изменяемыми значениями, то на практике выражение **ON UPDATE** используется редко.

### Установка NULL

При установки для внешнего ключа опции **SET NULL** необходимо, чтобы столбец внешнего ключа допускал значение NULL:

|  |  |
| --- | --- |
|  | CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt Date,      FOREIGN KEY (CustomerId) REFERENCES Customers (Id) ON DELETE SET NULL  ); |

### Установка значения по умолчанию

|  |  |
| --- | --- |
|  | CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt Date,      FOREIGN KEY (CustomerId) REFERENCES Customers (Id) ON DELETE SET DEFAULT  ) |

### Изменение таблицы

Возможно, в какой-то момент мы захотим изменить уже имеющуюся таблицу. Например, добавить или удалить столбцы, изменить тип столбцов, добавить или удалить ограничения. То есть потребуется изменить определение таблицы. Для изменения таблиц используется выражение **ALTER TABLE**.

Общий формальный синтаксис команды выглядит следующим образом:

|  |  |
| --- | --- |
|  | ALTER TABLE название\_таблицы [WITH CHECK | WITH NOCHECK]  { ADD название\_столбца тип\_данных\_столбца [атрибуты\_столбца] |  DROP COLUMN название\_столбца |  ALTER COLUMN название\_столбца тип\_данных\_столбца [NULL|NOT NULL] |  ADD [CONSTRAINT] определение\_ограничения |  DROP [CONSTRAINT] имя\_ограничения} |

Таким образом, с помощью ALTER TABLE мы можем провернуть самые различные сценарии изменения таблицы. Рассмотрим некоторые из них.

### Добавление нового столбца

Добавим в таблицу Customers новый столбец Address:

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  ADD Address NVARCHAR(50) NULL; |

В данном случае столбец Address имеет тип NVARCHAR и для него определен атрибут NULL. Но что если нам надо добавить столбец, который не должен принимать значения NULL? Если в таблице есть данные, то следующая команда не будет выполнена:

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  ADD Address NVARCHAR(50) NOT NULL; |

Поэтому в данном случае решение состоит в установке значения по умолчанию через атрибут DEFAULT:

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  ADD Address NVARCHAR(50) NOT NULL DEFAULT 'Неизвестно'; |

В этом случае, если в таблице уже есть данные, то для них для столбца Address будет добавлено значение "Неизвестно".

### Удаление столбца

Удалим столбец Address из таблицы Customers:

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  DROP COLUMN Address; |

### Изменение типа столбца

Изменим в таблице Customers тип данных у столбца FirstName на NVARCHAR(200):

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  ALTER COLUMN FirstName NVARCHAR(200); |

### Добавление ограничения CHECK

При добавлении ограничений SQL Server автоматически проверяет имеющиеся данные на соответствие добавляемым ограничениям. Если данные не соответствуют ограничениям, то такие ограничения не будут добавлены. Например, установим для столбца Age в таблице Customers ограничение Age > 21.

|  |  |
| --- | --- |
|  | ALTER TABLE Customers  ADD CHECK (Age > 21); |

Если в таблице есть строки, в которых в столбце Age есть значения, несоответствующие этому ограничению, то sql-команда завершится с ошибкой. Чтобы избежать подобной проверки на соответствие и все таки добавить ограничение, несмотря на наличие несоответствующих ему данных, используется выражение **WITH NOCHECK**:

|  |  |
| --- | --- |
|  | ALTER TABLE Customers WITH NOCHECK  ADD CHECK (Age > 21); |

По умолчанию используется значение **WITH CHECK**, которое проверяет на соответствие ограничениям.

### Добавление внешнего ключа

Пусть изначально в базе данных будут добавлены две таблицы, никак не связанные:

|  |  |
| --- | --- |
|  | CREATE TABLE Customers  (      Id INT PRIMARY KEY IDENTITY,      Age INT DEFAULT 18,      FirstName NVARCHAR(20) NOT NULL,      LastName NVARCHAR(20) NOT NULL,      Email VARCHAR(30) UNIQUE,      Phone VARCHAR(20) UNIQUE  );  CREATE TABLE Orders  (      Id INT IDENTITY,      CustomerId INT,      CreatedAt Date  ); |

Добавим ограничение внешнего ключа к столбцу CustomerId таблицы Orders:

|  |  |
| --- | --- |
|  | ALTER TABLE Orders  ADD FOREIGN KEY(CustomerId) REFERENCES Customers(Id); |

### Добавление первичного ключа

Используя выше определенную таблицу Orders, добавим к ней первичный ключ для столбца Id:

|  |  |
| --- | --- |
|  | ALTER TABLE Orders  ADD PRIMARY KEY (Id); |

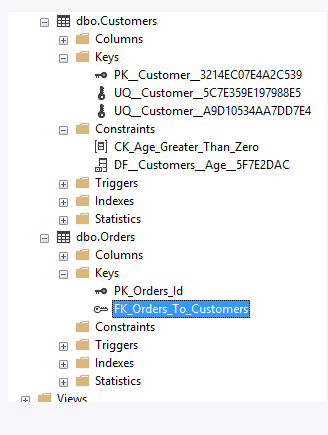
### Добавление ограничений с именами

При добавлении ограничений мы можем указать для них имя, используя оператор **CONSTRAINT**, после которого указывается имя ограничения:

|  |  |
| --- | --- |
|  | ALTER TABLE Orders  ADD CONSTRAINT PK\_Orders\_Id PRIMARY KEY (Id),  CONSTRAINT FK\_Orders\_To\_Customers FOREIGN KEY(CustomerId) REFERENCES Customers(Id);    ALTER TABLE Customers  ADD CONSTRAINT CK\_Age\_Greater\_Than\_Zero CHECK (Age > 0); |

### Удаление ограничений

Для удаления ограничений необходимо знать их имя. Если мы точно не знаем имя ограничения, то его можно узнать через SQL Server Management Studio:



Раскрыв узел таблиц в подузле Keys можно увидеть названия ограничений первичного и внешних ключей. Названия ограничений внешних ключей начинаются с "FK". А в подузле Constraints можно найти все ограничения CHECK и DEFAULT. Названия ограничений CHECK начинаются с "CK", а ограничений DEFAULT - с "DF".

Например, как видно на скриншоте в моем случае имя ограничения внешнего ключа в таблице Orders называется "FK\_Orders\_To\_Customers". Поэтому для удаления внешнего ключа я могу использовать следующее выражение:

|  |  |
| --- | --- |
|  | ALTER TABLE Orders  DROP FK\_Orders\_To\_Customers; |

### Пакеты. Команда GO

В предыдущих случаях сначала создавалась база данных, а затем в эту БД добавлялась таблица с помощью отдельных команд SQL. Но можно сразу совместить в одном скрипте несколько команд. В этом случае отдельные наборы команд называются **пакетами** (batch).

Каждый пакет состоит из одного или нескольких SQL-выражений, которые выполняются как оно целое. В качестве сигнала завершения пакета и выполнения его выражений служит команда **GO**.

Смысл разделения SQL-выражений на пакеты состоит в том, что одни выражения должны успешно выполниться до запуска других выражений. Например, при добавлении таблиц мы должны бы уверены, что была создана база данных, в которой мы собираемся создать таблицы.

Например, определим следующий скрипт:

|  |  |
| --- | --- |
|  | CREATE DATABASE internetstore;  GO    USE internetstore;    CREATE TABLE Customers  (      Id INT PRIMARY KEY IDENTITY,      Age INT DEFAULT 18,      FirstName NVARCHAR(20) NOT NULL,      LastName NVARCHAR(20) NOT NULL,      Email VARCHAR(30) UNIQUE,      Phone VARCHAR(20) UNIQUE  );    CREATE TABLE Orders  (      Id INT PRIMARY KEY IDENTITY,      CustomerId INT,      CreatedAt DATE,      FOREIGN KEY (CustomerId) REFERENCES Customers (Id) ON DELETE CASCADE  ); |

Вначале создается бд internetstore. Затем идет команда GO, которая сигнализирует, что можно выполнять следующий пакет выражений. И далее выполняется второй пакет, который добавляет в нее две таблицы - Customers и Orders.