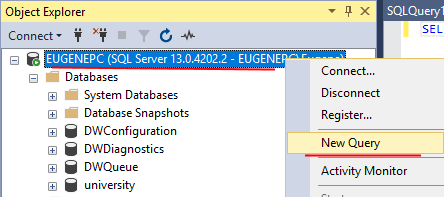
**Лабораторна робота № 1. Створення бази даних**

**УВАГА!!!**

|  |
| --- |
| **1. Зверніть увагу на назви вже наявних баз, щоб не створювати таку ж. Додавайте в кінці назви свої ініціали**  **2. В роботі показані різні варіанти створення таблиць і зв'язків між ними. Перед кожним новим варіантом створення, перевірте на наявність вже наявної таблиці і в разі її наявності виконайте команду DROP.**  **3. Бажано. Зберігати всі створені запити в окремий текстовий документ з поясненнями, що вони означають.** |

Для створення бази даних використовується команда CREATE DATABASE.

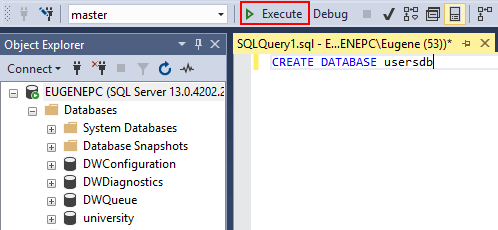
Щоб створити нову базу даних відкриємо SQL Server Management Studio. Натиснемо на призначення сервера у вікні Object Explorer і в меню виберемо пункт New Query.



У центральне поле для введення виразів sql введемо наступний код:

|  |
| --- |
| CREATE DATABASE usersdb |

Тим самим ми створюємо базу даних, яка буде називатися "usersdb":



Для виконання команди натиснемо на панелі інструментів на кнопку Execute або на клавішу F5. І на сервері з'явиться нова база даних.

Після створення бази даних, ми можемо встановити її в якості поточної за допомогою команди USE:

|  |
| --- |
| USE usersdb; |

**Видалення бази даних**

Для видалення бази даних застосовується команда DROP DATABASE, яка має наступний синтаксис:

|  |
| --- |
| DROP DATABASE database\_name1 [, database\_name2] ... |

Після команди через кому ми можемо перерахувати всі бази даних, які хочемо видалити. Наприклад, видалення бази даних contactsdb:

|  |
| --- |
| DROP DATABASE contactsdb |

## Створення та видалення таблиць

Для створення таблиць застосовується команда CREATE TABLE. З цією командою можна використовувати ряд операторів, які визначають стовпці таблиці і їх атрибути. І крім того, можна використовувати ряд операторів, які визначають властивості таблиці в цілому. Одна база даних може містити до 2 мільярдів таблиць.

Загальний синтаксис створення таблиці виглядає наступним чином:

|  |
| --- |
| CREATE TABLE назва\_таблиці  (Назва\_стовбця1 тип\_даних атрибути\_стовбця1,  назва\_стовбця2 тип\_даних атрибути\_стовбця2,  ................................................  назва\_стовбцяN тип\_даних атрибути\_стовбцяN,  атрибути\_таблиці  ) |

Після команди CREATE TABLE йде назва повинна містити таблиця. Ім'я таблиці виконує роль її ідентифікатора в базі даних, тому воно повинно бути унікальним. Ім'я повинно мати довжину трохи більше 128 символів. Ім'я може складатися з алфавітно-цифрових символів, а також символів $ і знака підкреслення. Причому першим символом повинна бути буква або знак підкреслення.

Ім'я об'єкта не може включати пробіли і не може бути одним із ключових слів мови Transact-SQL. Якщо ідентифікатор все ж містить пробільні символи, то його слід брати в лапки. Якщо необхідно в якості імені використовувати ключові слова, то ці слова поміщаються в квадратні дужки.

Приклади коректних ідентифікаторів:

|  |
| --- |
| Users  tags $ 345  users\_accounts  "Users accounts"  [Table] |

Після імені таблиці в дужках вказуються параметри всіх стовпців і в самому кінці атрибути, які відносяться до всієї таблиці. Атрибути стовпців і атрибути таблиці є необов'язковими компонентами, і їх можна не вказувати.

У самому просто вигляді команда CREATE TABLE повинна містити як мінімум ім'я таблиці, імена і типи стовпців. Таблиця може містити від 1 до 1024 стовпців. Кожен стовпець повинен мати унікальне в рамках поточної таблиці ім'я, і йому має бути призначено тип даних.

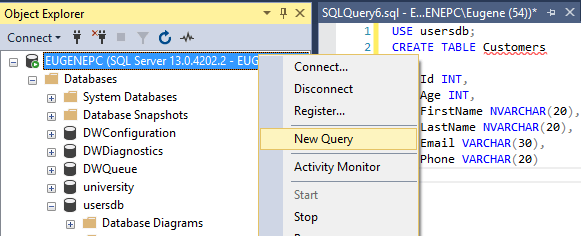
Наприклад, визначення найпростішої таблиці Customers:

|  |
| --- |
| CREATE TABLE Customers  (  Id INT,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20)  ) |

В даному випадку в таблиці Customers визначаються шість стовпців: Id, FirstName, LastName, Age, Email, Phone. Перші два стовпці представляють ідентифікатор клієнта і його вік і мають тип INT, тобто будуть зберігати числові значення. Наступні два стовпці представляють ім'я і прізвище клієнта і мають тип NVARCHAR (20), тобто представляють рядок UNICODE довжиною не більше 20 символів. Останні два стовпці Email і Phone представляють адреса електронної пошти та телефон клієнта і мають тип VARCHAR (30/20) - вони також зберігають рядок, але не в кодуванні UNICODE.

**Створення таблиці в SQL Management Studio**

Створимо просту таблицю на сервері. Для цього відкриємо SQL Server Management Studio та натиснемо правою кнопкою миші на назву сервера. У контекстному меню виберемо пункт New Query.

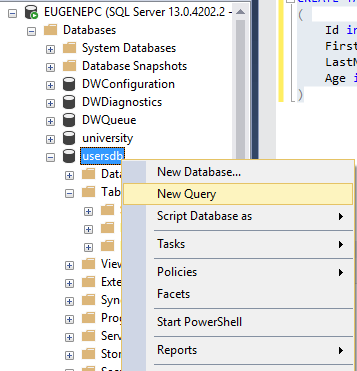


Таблиця створюється в рамках поточної бази даних. Якщо ми запускаємо вікно редактора SQL як це зроблено вище - з під назви сервера, то база даних за замовчуванням не встановлена. І для її установки необхідно застосувати команду USE, після якої вказується ім'я бази даних. Тому введемо в поле редактора SQL-команд такі вирази:

|  |
| --- |
| USE usersdb;  CREATE TABLE Customers  (  Id INT,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20)  ); |

Тобто в базу даних додається таблиця Customers, яка була розглянута раніше.

Також можна відкрити редактор з під бази даних, також натиснувши на неї правою кнопкою миші і вибравши New Query:



У цьому випадку в якості поточної буде розглядатися та база даних, з під якої був відкритий редактор, і додатково її встановлювати за допомогою команди USE не буде потрібно.

**Видалення таблиць**

Для видалення таблиць використовується команда DROP TABLE, яка має наступний синтаксис:

|  |
| --- |
| DROP TABLE table1 [, table2, ...] |

Наприклад, видалення таблиці Customers:

|  |
| --- |
| DROP TABLE Customers |

**Перейменування таблиці**

Для перейменування таблиць застосовується системна збережена процедура "sp\_rename". Наприклад, перейменування таблиці Users в UserAccounts в базі даних usersdb:

|  |
| --- |
| USE usersdb;  EXEC sp\_rename 'Users', 'UserAccounts'; |

## Атрибути і обмеження стовпців і таблиць

**УВАГА!!!**

|  |
| --- |
| **1. Не вводьте багато даних в таблиці, а тільки для перевірки роботи описаних параметрів**  **2. Призначені атрибути перевіряються у властивостях таблиці** |

При створенні стовпців в T-SQL ми можемо використовувати ряд атрибутів, ряд яких є обмеженнями. Розглянемо ці атрибути.

**PRIMARY KEY**

За допомогою виразу PRIMARY KEY стовпець можна зробити первинним ключем.

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20)  ) |

Первинний ключ унікально ідентифікує рядок в таблиці. В якості первинного ключа необов'язково повинні виступати стовпці з типом int, вони можуть представляти будь-який інший тип.

Установка первинного ключа на рівні таблиці:

|  |
| --- |
| CREATE TABLE Customers  (  Id INT,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20),  PRIMARY KEY (Id)  ) |

Первинний ключ може бути складовим (compound key). Такий ключ може знадобитися, якщо у нас відразу два стовпці повинні унікально ідентифікувати рядок в таблиці. наприклад:

|  |
| --- |
| CREATE TABLE OrderLines  (  OrderId INT,  ProductId INT,  Quantity INT,  Price MONEY,  PRIMARY KEY (OrderId, ProductId)  ) |

Тут поля OrderId і ProductId разом виступають як складової первинний ключ. Тобто в таблиці OrderLines не може бути двох рядків, де для обох з цих полів одночасно були б одні і ті ж значення.

**IDENTITY**

Атрибут IDENTITY дозволяє зробити стовпець ідентифікатором. Цей атрибут може призначатися для стовпців числових типів INT, SMALLINT, BIGINT, TYNIINT, DECIMAL і NUMERIC. При додаванні нових даних в таблицю SQL Server буде інкрементіровать на одиницю значення цього стовпця у останнього запису. Як правило, в ролі ідентифікатора виступає той же стовпець, який є первинним ключем, хоча в принципі це необов'язково.

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY IDENTITY,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20)  ) |

Також можна використовувати повну форму атрибута:

|  |
| --- |
| IDENTITY (seed, increment) |

Тут параметр seed вказує на початкове значення, з якого буде починатися відлік. А параметр increment визначає, наскільки буде збільшуватися в такому значенні. За замовчуванням атрибут використовує наступні значення:

|  |
| --- |
| IDENTITY (1, 1) |

Тобто відлік починається з 1. А наступні значення збільшуються на одиницю. Але ми можемо це поведінка перевизначити. наприклад:

|  |
| --- |
| Id INT IDENTITY (2, 3) |

В даному випадку відлік почнеться з 2, а значення кожної наступної записи буде збільшуватися на 3. Тобто перший рядок буде мати значення 2, друга - 5, третя - 8 і т.д.

Також слід враховувати, що в таблиці тільки один стовпець повинен мати такий атрибут.

**UNIQUE**

Якщо ми хочемо, щоб стовпець мав тільки унікальні значення, то для нього можна визначити атрибут UNIQUE.

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY IDENTITY,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30) UNIQUE,  Phone VARCHAR (20) UNIQUE  ) |

В даному випадку стовпці, які представляють електронну адресу і телефон, матимуть унікальні значення. І ми не зможемо додати в таблицю два рядки, у яких значення для цих стовпців буде збігатися.

Також ми можемо визначити цей атрибут на рівні таблиці:

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY IDENTITY,  Age INT,  FirstName NVARCHAR (20),  LastName NVARCHAR (20),  Email VARCHAR (30),  Phone VARCHAR (20),  UNIQUE (Email, Phone)  ) |

**NULL і NOT NULL**

Щоб вказати, чи може стовпець приймати значення NULL, при визначенні стовпця йому можна задати атрибут NULL або NOT NULL. Якщо цей атрибут явно не буде використаний, то за замовчуванням стовпець буде допускати значення NULL. Винятком є той випадок, коли стовпчик виступає в ролі первинного ключа - в цьому випадку за замовчуванням стовпець має значення NOT NULL.

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY IDENTITY,  Age INT,  FirstName NVARCHAR (20) NOT NULL,  LastName NVARCHAR (20) NOT NULL,  Email VARCHAR (30) UNIQUE,  Phone VARCHAR (20) UNIQUE  ) |

**DEFAULT**

Атрибут DEFAULT визначає значення за замовчуванням для стовпця. Якщо при додаванні даних для стовпця нічого очікувати передбачено значення, то для нього буде використовуватися значення за замовчуванням.

|  |
| --- |
| 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  ); |

Тут для стовпця Age передбачено значення за замовчуванням 18.

**CHECK**

Ключове слово CHECK задає обмеження для діапазону значень, які можуть зберігатися в стовпці. Для цього після слова CHECK вказується в дужках умова, якому повинен відповідати стовпець або декілька стовпців. Наприклад, вік клієнтів не може бути менше 0 або більше 100:

|  |
| --- |
| CREATE TABLE Customers  (  Id INT PRIMARY KEY IDENTITY,  Age INT DEFAULT 18 CHECK (Age> 0 AND Age <100),  FirstName NVARCHAR (20) NOT NULL,  LastName NVARCHAR (20) DEFAULT '' NOT NULL,  Email VARCHAR (30) UNIQUE CHECK (Email! = ''),  Phone VARCHAR (20) UNIQUE CHECK (Phone! = '')  ); |

Тут також вказується, що стовпці Email і Phone не можуть мати порожній рядок як значення (порожній рядок не є еквівалентною значенням NULL).

Для з'єднання умов використовується ключове слово AND. Умови можна задати у вигляді операцій порівняння більше (>), менше (<), не дорівнює (!=).

Також за допомогою CHECK можна створити обмеження в цілому для таблиці:

|  |
| --- |
| 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,  CHECK ((Age> 0 AND Age <100) AND (Email! = '') AND (Phone! = ''))  ) |

**Оператор CONSTRAINT. Установка імені обмежень.**

За допомогою ключового слова CONSTRAINT можна задати ім'я для обмежень. В якості обмежень можуть використовуватися PRIMARY KEY, UNIQUE, DEFAULT, CHECK.

Імена обмежень можна задати на рівні стовпців. Вони вказуються після CONSTRAINT перед атрибутами:

|  |
| --- |
| CREATE TABLE Customers  (Id INT CONSTRAINT PK\_Customer\_Id PRIMARY KEY IDENTITY,  Age INT  CONSTRAINT DF\_Customer\_Age DEFAULT 18  CONSTRAINT CK\_Customer\_Age CHECK (Age> 0 AND Age <100),  FirstName NVARCHAR (20) NOT NULL,  LastName NVARCHAR (20) NOT NULL,  Email VARCHAR (30) CONSTRAINT UQ\_Customer\_Email UNIQUE,  Phone VARCHAR (20) CONSTRAINT UQ\_Customer\_Phone UNIQUE  ) |

Обмеження можуть носити довільні назви, але, як правило, для застосовуються такі префікси:

* "PK\_" - для PRIMARY KEY
* "FK\_" - для FOREIGN KEY
* "CK\_" - для CHECK
* "UQ\_" - для UNIQUE
* "DF\_" - для DEFAULT

В принципі необов'язково ставити імена обмежень, при установці відповідних атрибутів SQL Server автоматично визначає їх імена. Але, знаючи ім'я обмеження, ми можемо до нього звертатися, наприклад, для його видалення.

І також можна задати всі імена обмежень через атрибути таблиці:

|  |
| --- |
| CREATE TABLE Customers  (  Id INT IDENTITY,  Age INT CONSTRAINT DF\_Customer\_Age DEFAULT 18,  FirstName NVARCHAR (20) NOT NULL,  LastName NVARCHAR (20) NOT NULL,  Email VARCHAR (30),  Phone VARCHAR (20),  CONSTRAINT PK\_Customer\_Id PRIMARY KEY (Id),  CONSTRAINT CK\_Customer\_Age CHECK (Age> 0 AND Age <100),  CONSTRAINT UQ\_Customer\_Email UNIQUE (Email),  CONSTRAINT UQ\_Customer\_Phone UNIQUE (Phone)  ) |

## Зовнішні ключі

Зовнішні ключі застосовуються для установки зв'язку між таблицями. Зовнішній ключ встановлюється для стовпців з залежною, підлеглої таблиці, і вказує на один із стовпців з головної таблиці. Хоча, як правило, зовнішній ключ вказує на первинний ключ з пов'язаної головною таблиці, але це не обов'язково має бути неодмінною умовою. Зовнішній ключ також може вказувати на якийсь інший стовпець, який має унікальне значення.

Загальний синтаксис установки зовнішнього ключа на рівні стовпця:

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

Для створення обмеження зовнішнього ключа на рівні стовпця після ключового слова REFERENCES вказується ім'я пов'язаної таблиці і в круглих дужках ім'я пов'язаного стовпця, на який буде вказувати зовнішній ключ. Також зазвичай додаються ключові слова FOREIGN KEY, але в принципі їх необов'язково вказувати. Після висловлення REFERENCES йде вираз ON DELETE і ON UPDATE.

Загальний синтаксис установки зовнішнього ключа на рівні таблиці:

|  |
| --- |
| FOREIGN KEY (Стобец1, столбец2, ... столбецN)  REFERENCES головна\_таблиця (стовбець\_головної\_таблиці\_1, стовбець\_головної\_таблиці2, ... стовбець\_головної\_таблиціN)  [ON DELETE {CASCADE | NO ACTION}]  [ON UPDATE {CASCADE | NO ACTION}] |

Наприклад, визначимо дві таблиці і зв'яжемо їх за допомогою зовнішнього 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  ); |

Тут визначені таблиці Customers і Orders. Customers є головною і представляє клієнта. Orders є залежною і являє замовлення, зроблене клієнтом. Ця таблиця через стовпець CustomerId пов'язана з таблицею Customers і її стовпцем Id. Тобто стовпець CustomerId є зовнішнім ключем, який вказує на стовпець Id з таблиці Customers.

Визначення зовнішнього ключа на рівні таблиці виглядало б так:

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

За допомогою оператора CONSTRAINT можна задати ім'я для обмеження зовнішнього ключа. Зазвичай це ім'я починається з префікса "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 даному випадку обмеження зовнішнього ключа CustomerId називається "FK\_Orders\_To\_Customers".

**ON DELETE і ON UPDATE**

За допомогою виразів ON DELETE і ON UPDATE можна встановити дії, які виконуватимуться відповідно при видаленні і зміні пов'язаної рядки з головної таблиці. І для визначення дії ми можемо використовувати такі опції:

* **CASCADE**: Автоматично видаляє або змінює рядки з залежною таблиці при видаленні або зміні пов'язаних рядків в головній таблиці.
* **NO ACTION**: Запобігає будь-які дії в залежною таблиці при видаленні або зміні пов'язаних рядків в головній таблиці. Тобто фактично будь-які дії відсутні.
* **SET NULL**: При видаленні пов'язаної рядки з головної таблиці встановлює для стовпця зовнішнього ключа значення NULL.
* **SET DEFAULT**: При видаленні пов'язаної рядки з головної таблиці встановлює для стовпця зовнішнього ключа значення за замовчуванням, яке задається за допомогою атрибути DEFAULT. Якщо для стовпця не задано значення за замовчуванням, то в якості нього застосовується значення NULL.

**Каскадне видалення**

За замовчуванням, якщо на рядок з головної таблиці по зовнішньому ключу посилається будь-який рядок з залежною таблиці, то ми не зможемо видалити цей рядок з головної таблиці. Спочатку нам необхідно буде видалити всі пов'язані рядки з залежною таблиці. І якщо при видаленні рядка з головної таблиці необхідно, щоб були вилучені всі пов'язані рядки з залежною таблиці, то застосовується каскадне видалення, тобто опція CASCADE:

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

Аналогічно працює вираз ON UPDATE CASCADE. При зміні значення первинного ключа автоматично зміниться значення пов'язаного з ним зовнішнього ключа. Але так як первинні ключі, як правило, змінюються дуже рідко, та й з принципі не рекомендується використовувати в якості первинних ключів стовпці із змінними значеннями, то на практиці вираз ON UPDATE використовується рідко.

**Установка NULL**

При установки для зовнішнього ключа опції SET NULL необхідно, щоб стовпець зовнішнього ключа допускав значення NULL:

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

**Установка значення за замовчуванням**

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

## Внесення змін в таблицю

Можливо, в якийсь момент ми захочемо змінити вже наявну таблицю. Наприклад, додати або видалити стовпці, змінити тип стовпців, додати або видалити обмеження. Тобто потрібно змінити визначення таблиці. Для зміни таблиць використовується вираз ALTER TABLE.

Загальний формальний синтаксис команди виглядає наступним чином:

|  |
| --- |
| ALTER TABLE назва\_таблиці [WITH CHECK | WITH NOCHECK]  {ADD назва\_стовбця тип\_данних\_стовбця [атрибути\_стовбця] |  DROP COLUMN назва\_стовбця |  ALTER COLUMN назва\_стовбця тип\_данних\_стовбця [NULL | NOT NULL] |  ADD [CONSTRAINT] визначення\_обмеження |  DROP [CONSTRAINT] ім’я\_обмеження} |

Таким чином, за допомогою ALTER TABLEми можемо провернути найрізноманітніші сценарії зміни таблиці. Розглянемо деякі з них.

**Додавання нового стовпця**

Додамо в таблицю Customers новий стовпець Address:

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

В даному випадку стовпець Address має тип NVARCHAR і для нього визначений атрибут NULL. Але що якщо нам треба додати стовпець, який не повинен приймати значення NULL? Якщо в таблиці є дані, то наступна команда не буде виконана:

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

Тому в даному випадку рішення полягає в установці значення за замовчуванням через атрибут DEFAULT:

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

В цьому випадку, якщо в таблиці вже є дані, то для них для стовпця Address буде додано значення "Невідомо".

**Видалення стовпця**

Видалимо стовпець Address з таблиці Customers:

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

**Зміна типу стовпця**

Змінимо в таблиці Customers тип даних у стовпці FirstName на NVARCHAR (200):

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

**Додавання обмеження CHECK**

При додаванні обмежень SQL Server автоматично перевіряє наявні дані на відповідність додаються обмеженням. Якщо дані не відповідають обмеженням, то такі обмеження не будуть додані. Наприклад, встановимо для стовпця Age в таблиці Customers обмеження Age> 21.

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Customers  ADD CHECK (Age> 21); |

Якщо в таблиці є рядки, в яких в стовпці Age є значення, невідповідні цього обмеження, то sql-команда завершиться з помилкою. Щоб уникнути подібної перевірки на відповідність і все таки додати обмеження, незважаючи на наявність невідповідних йому даних, використовується вираз WITH NOCHECK:

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

За замовчуванням використовується значення WITH CHECK, яке перевіряє на відповідність обмеженням.

**Додавання зовнішнього ключа**

Нехай спочатку в базі даних будуть додані дві таблиці, ніяк не пов'язані:

|  |
| --- |
| 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  ); |

Додамо обмеження зовнішнього ключа до колонку CustomerId таблиці Orders:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Orders  ADD FOREIGN KEY (CustomerId) REFERENCES Customers (Id); |

**Додавання первинного ключа**

Використовуючи вище певну таблицю Orders, додамо до неї первинний ключ для стовпця Id:

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

**Додавання обмежень з іменами**

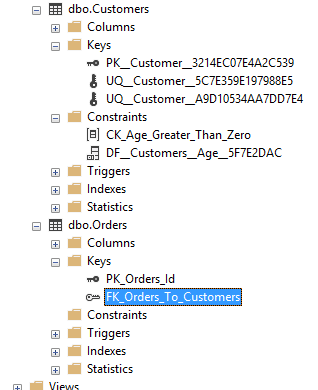
При додаванні обмежень ми можемо вказати для них ім'я, використовуючи оператор CONSTRAINT, після якого вказується ім'я обмеження:

|  |
| --- |
| 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); |

**Видалення обмежень**

Для видалення обмежень необхідно знати їх ім'я. Якщо ми точно не знаємо ім'я обмеження, то його можна дізнатися через SQL Server Management Studio.

Розкривши вузол таблиць в Підвузли Keys можна побачити назви обмежень первинного і зовнішніх ключів. Назви обмежень зовнішніх ключів починаються з "FK". А в Підвузли Constraints можна знайти все обмеження CHECK і DEFAULT. Назви обмежень CHECK починаються з "CK", а обмежень DEFAULT - з "DF".



Наприклад, як видно на скріншоті в моєму випадку ім'я обмеження зовнішнього ключа в таблиці Orders називається "FK\_Orders\_To\_Customers". Тому для видалення зовнішнього ключа я можу використовувати такий вираз:

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

**Завдання:**

1. Перевірити роботу всіх запитів
2. Створити БД з 2 таблиць Customers та Orders, які містять наступні поля:

**Customers:**

id – код користувача унікальний, ключовий, цілий, автоматичне зростання на 1

Age – вік користувача, ціле, за умовчанням 18, обмеження на вік від 18 до 100

FirstName – ім’я користувача, ненульове текстове поле на 20 символів

LastName – прізвище користувача, ненульове текстове поле на 20 символів,

Email – e-mail користувача, унікальне текстове поле,

Phone – телефон користувача, унікальне текстове поле на 13 символів

**Orders:**

Id – код замовлення унікальний, ключовий, цілий, автоматичне зростання на 1

CustomerId – код користувача, що придбав замовлення, зв’язок з полем id таблиці Customers, тип оновлення при видаленні – ставити код користувача з id 1

CreatedAt – дата створення замовлення

1. Наповнити таблиці значеннями (3-4 записи в Customers, 5-10 записів в Orders)
2. Перевірити роботу БД при видаленні записів з Customers