**Huy Bui**
Information Technology
T5616SN

DATABASE
**Referential integrity**
30 January 2018

# 1 Referential integrity

## 1.1 Introduction

A Referential integrity is a database concept that is used to build and maintain logical relationships between tables to avoid logical corruption of data. Usually, referential integrity is made up of the combination of a primary key and a foreign key.
The main concept of Referential integrity is that it does not allow to add any record in a table that contains the foreign key unless the reference table containing a corresponding primary key. If any record in referenced table (i.e. the table who contain primary key) is deleted, all the corresponding records in the referencing table will be deleted for the referential integrity.

## 1.2 Cascade

By using cascading referential integrity constraints, you can define the actions that the SQL Server takes when a user tries to delete or update a key to which existing foreign keys point. You can specify this kind of constraint using the REFERENCES clauses of the CREATE TABLE or ALTER TABLE statements, which support the ON DELETE and ON UPDATE clauses along with Cascading actions. The syntax is shown as below:

```
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
```

NO ACTION is the default if ON DELETE or ON UPDATE is not specified. ON DELETE NO ACTION specifies that if an attempt is made to delete a row with a key referenced by foreign keys in existing rows in other tables, an error is raised and the DELETE statement is rolled back. On the other hand, ON DELETE CASCADE specifies that if an attempt is made to delete a row with a key referenced by foreign keys in existing rows in other tables, all rows that contain those foreign keys are also deleted.

# 2 Practical part

## 2.1 Create foreign keys

At first, we create three tables (Client, Product, and Orders) using the following commands.

```
CREATE TABLE Client(
  Id int IDENTITY(1,1) NOT NULL PRIMARY KEY,
  FirstName nvarchar(50) NOT NULL,
  FamilyName nvarchar(50) NOT NULL,
);
```

```
CREATE TABLE Product (
  Id int IDENTITY(1,1) NOT NULL PRIMARY KEY,
  ProductName nvarchar(50) NOT NULL
);
```

```
CREATE TABLE Orders (
  Id int IDENTITY(1,1) NOT NULL PRIMARY KEY,
  OrderNumber uniqueidentifier NOT NULL UNIQUE NONCLUSTERED,
  Amount int NOT NULL,
  ClientId int NOT NULL,
  ProductId int NOT NULL
);
```

Next, we create two foreign keys in Orders table. The first one links the attribute ClientId in Orders table with the Id primary key in Client table.

```
ALTER TABLE Orders
ADD CONSTRAINT FK_Client_Id FOREIGN KEY (ClientId)
    REFERENCES Client(Id)
```

The second one links the attribute ProductId in Orders table with the Id primary key in Product table.

```sql
ALTER TABLE Orders
ADD CONSTRAINT FK_Orders_Product FOREIGN KEY ProductId
    REFERENCES Product(Id)
```

You can also do this using GUI software such as SQL Server Management Studio:

1. In Object Explorer, right-click the table that will be on the foreign-key side of the relationship and click **Design**. The table opens in **Table Designer**.
2. From the **Table Designer** menu, click **Relationships**.
3. In the **Foreign-key Relationships** dialog box, click **Add**.
4. Click the relationship in the **Selected Relationship** list.
5. Click **Tables and Columns Specification** in the grid to the right and click the ellipses (...) to the right of the property.
6. In the Tables and Columns dialog box, in the **Primary Key** drop-down list, choose the table that will be on the primary-key side of the relationship.
7. In the grid beneath, choose the columns contributing to the table's primary key. In the adjacent grid cell to the left of each column, choose the corresponding foreign-key column of the foreign-key table.
8. Choose **OK** to create the relationship.

## 2.2 Inserting data

To see how foreign keys work in action, we need first to insert data in each table using queries on Moodle page (InsertInClient.sql, InsertInOrders.sql, InsertInProduct.sql). At this point, please note that the order of inserting data does matter. Try to execute InsertInOrders.sql before the others, you will then see error messages. The reason is that data ClientId and ProductId attributes are references to that in Client and Product table, so it makes no sense to refer to non-existing data. Therefore, data must be inserted into Client and Product table first.

## 2.3 Delete Client

Try to delete a row in Client using the following command:

```sql
DELETE FROM Client
WHERE FirstName LIKE '%Dai%'
```

An error message appears and denies the DELETE statement. The reason is that ON DELETE NO ACTION is enabled by default and any attempts to delete any rows in Client table is prohibited. To be enable to remove a row from Client table, you have to enable cascading action. First, remove the existing constraint. Then, create the new one with cascading action:

```sql
ALTER TABLE Orders
DROP CONSTRAINT FK_Client_Id
ADD CONSTRAINT FK_Client_Id FOREIGN KEY (ClientId)
    REFERENCES Client(Id) ON DELETE CASCADE
```

As for people who prefer GUI application, the following instruction shows how to modify Foreign Key in SQL Server Management Studio:

1. In **Object Explorer**, expand the table with the foreign key and then expand **Keys**.
2. Right-click the foreign key to be modified and select **Modify**.
3. In the **Foreign Key Relationships** dialog box, you can make modifications. In our case, on **Delete Rule** drop-down list, choose **Cascade**.
4. Close the window and Save the table.
5. A dialog box will appear to ask you for confirmation to make changes on Client and Orders table. Choose Yes.