# Database Lab

## CSE 3104

**Lab-03**

# 6 SQL Constraints

Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

Here are the most important constraints:

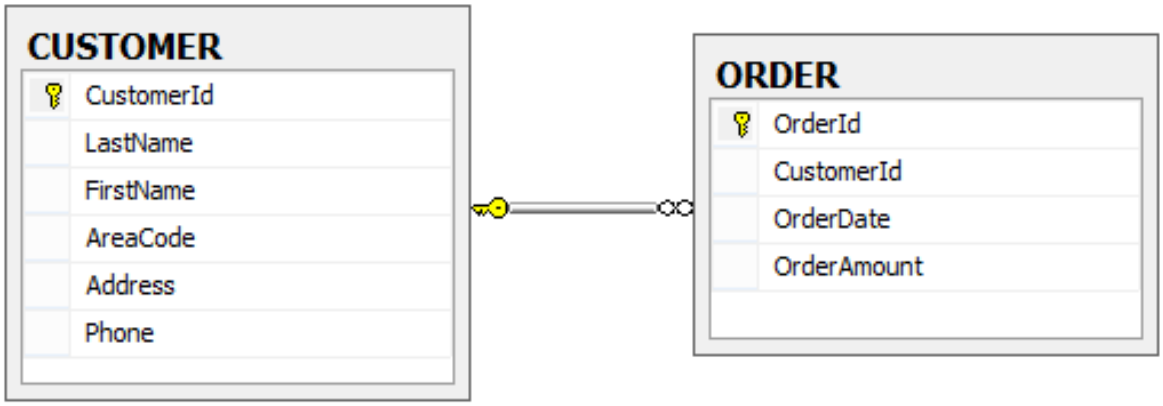
- PRIMARY KEY
- NOT NULL
- UNIQUE
- FOREIGN KEY
- CHECK
- DEFAULT
- IDENTITY

In the sections below we will explain some of these in detail.

# 6.3 FOREIGN KEY

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

**Example:**



At First create a table called ORDER using the following Query.

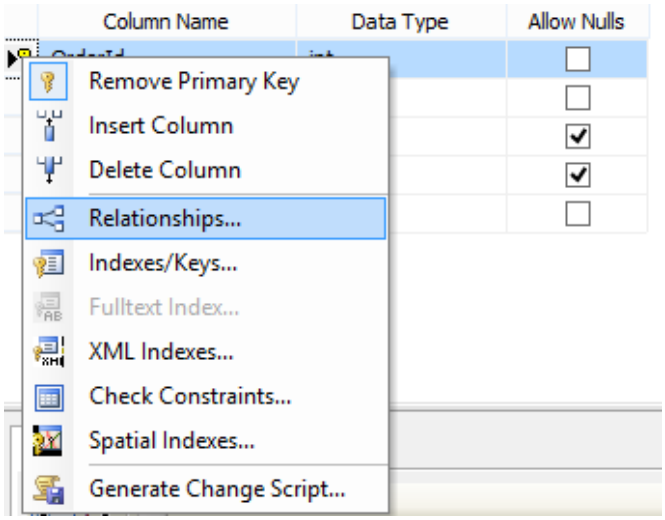
```
CREATE TABLE ORDER

(
OrderId int IDENTITY (1, 1) PRIMARY KEY,

CustomerId int NOT NULL FOREIGN KEY REFERENCES CUSTOMER (CustomerId),

OrderDate date NULL,

OrderAmount money NULL,

)
```

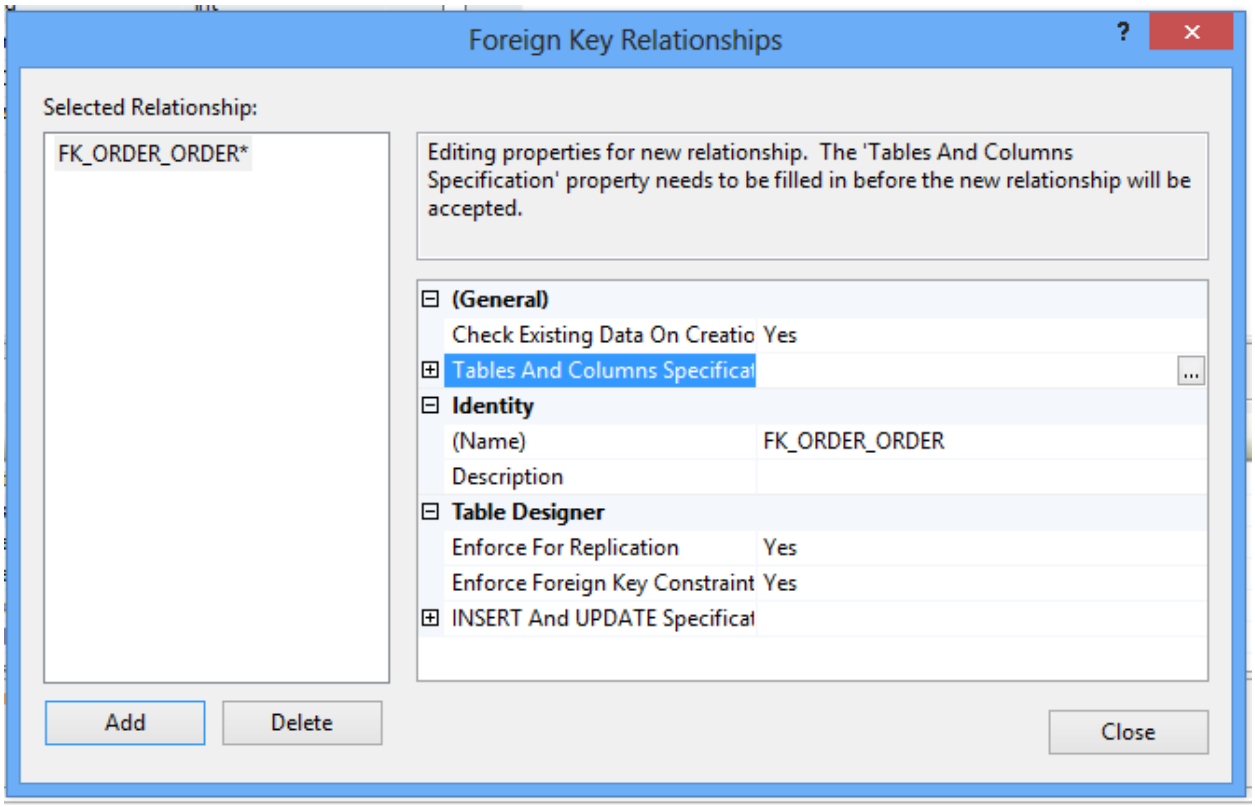
The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

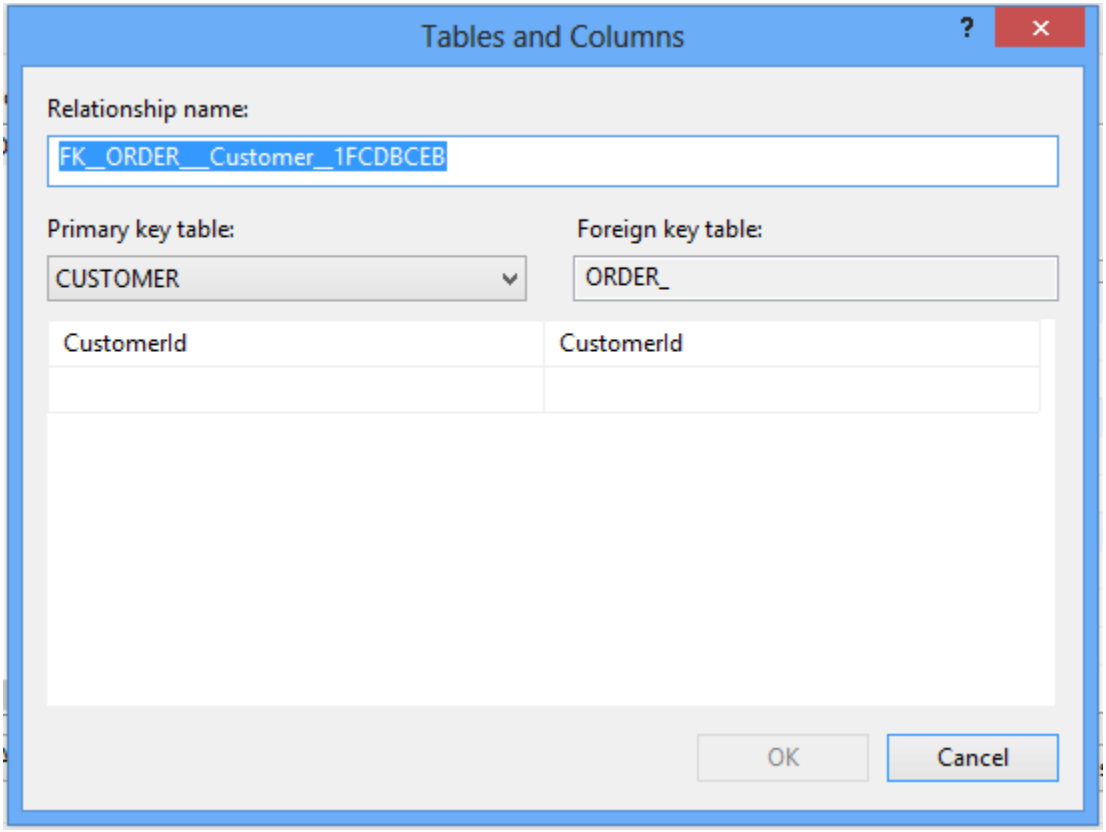
The FOREIGN KEY constraint also prevents that invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

**Setting Foreign Keys in the Designer Tools:**

If you want to use the designer, right-click on the column that you want to be the Foreign Key and select "**Relationships…**":




Foreign Key Relationships
Selected Relationship:
FK_ORDER_ORDER*
Editing properties for new relationship. The 'Tables And Columns Specification' property needs to be filled in before the new relationship will be accepted.
(General)
Check Existing Data On Creatio Yes
Tables And Columns Specificat
Identity
(Name) FK_ORDER_ORDER
Description
Table Designer
Enforce For Replication Yes
Enforce Foreign Key Constraint Yes
INSERT And UPDATE Specificat
Add
Delete
Close


Tables and Columns
Relationship name:
FK__ORDER__Customer__1FCDBCEB
Primary key table:
CUSTOMER
Foreign key table:
ORDER_
CustomerId
CustomerId
OK
Cancel

# 6.4 UNIQUE

The **UNIQUE** constraint uniquely identifies each record in a database table. The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.

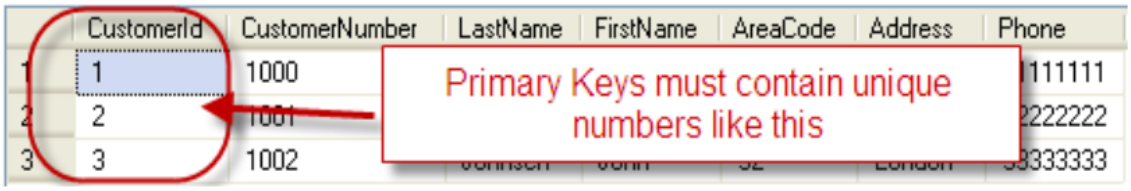
A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.

**Note!** You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

If we take a closer look at the CUSTOMER table created earlier:

```
CREATE TABLE CUSTOMER
(
        CustomerId int IDENTITY(1,1) PRIMARY KEY,
        CustomerNumber int NOT NULL UNIQUE ,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        ------------------------------
)
```

As you see we use the “Primary Key” keyword to specify that a column should be the Primary Key.



# 6.5 CHECK

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

```
CREATE TABLE CUSTOMER
(
        CustomerId int IDENTITY(1,1) PRIMARY KEY,
        CustomerNumber int NOT NULL UNIQUE  CHECK(CustomerNumber>1000),
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        ------------------------------
)
```

In this case, when we try to insert a Customer Number less than zero we will get an error message.

# 6.5 DEFAULT

The DEFAULT constraint is used to insert a default value into a column.

The default value will be added to all new records, if no other value is specified.

Example:

```
CREATE TABLE CUSTOMER
(
        CustomerId int IDENTITY(1,1) PRIMARY KEY,
        CustomerNumber int NOT NULL UNIQUE  CHECK(CustomerNumber>1000),
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        AreaCode int NULL,
        Address varchar(200) NULL  DEFAULT 'Dhaka',
        Phone varchar(11) NULL,
)
```

# 7 UPDATE

The UPDATE statement is used to update existing records in a table.

The syntax is as follows:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

**Note!** Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Example:

```
UPDATE CUSTOMER set AreaCode=46 where CustomerId=2
```

**Note:** If you don't include the WHERE clause then result becomes updated to all records. So make sure to include the WHERE clause when usingthe UPDATE command!

# 8 DELETE

The DELETE statement is used to delete rows in a table.

Syntax:

```
DELETE FROM table_name
WHERE some_column=some_value
```

**Note!** Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Example:

```
delete from CUSTOMER where CustomerId=2
```

### Delete All Rows:

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name
```

**Note!** Make sure to do this only when you really mean it! You cannot UNDO this statement!

# 9 SELECT

The SELECT statement is probably the most used SQL command. The SELECT statement is used for retrieving rows from the database and enables the selection of one or many rows or columns from one or many tables in the database.

We will use the CUSTOMER table as an example.

Example:

```
select * from CUSTOMER
```

This simple example gets all the data in the table CUSTOMER. The symbol "*" is used when you want to get all the columns in the table.

If you only want a few columns, you may specify the names of the columns you want to retrieve, example:

```
select CustomerId, LastName, FirstName from CUSTOMER
```

So in the simplest form we can use the SELECT statement as follows:

```
select <column_names> from <table_names>
```

If we want all columns, we use the symbol "*"

**Note!** SQL is not case sensitive. SELECT is the same as select.

The full syntax of the SELECT statement is complex, but the main clauses can be summarized as:

```
SELECT
[ ALL | DISTINCT ]
   [TOP ( expression ) [PERCENT] [ WITH TIES ] ]
select_list [ INTO new_table ]
[ FROM table_source ] [ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

It seems complex, but we will take the different parts step by step in the next sections.

# 10 The ORDER BY Keyword

If you want the data to appear in a specific order you need to use the "order by" keyword.

Example:

```
select * from CUSTOMER order by LastName
```

You may also sort by several columns, e.g. like this:

```
select * from CUSTOMER order by Address, LastName
```

If you use the “order by” keyword, the default order is ascending (“asc”). If you want the order to be opposite, i.e., descending, then you need to use the “desc” keyword.

```
select * from CUSTOMER order by LastName desc
```

# 11 SELECT DISTINCT

In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table.

The DISTINCT keyword can be used to return only distinct (different) values.

The syntax is as follows:

```
select distinct <column_names> from <table_names>
```

Example:

```
select distinct FirstName from CUSTOMER
```

# 12 The WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

The syntax is as follows:

```
select <column_names>
from <table_name>
where <column_name> operator value
```

Example:

```
select * from CUSTOMER where CustomerNumber='1001'
```

**Note!** SQL uses single quotes around text values, as shown in the example above.

# 13 Operators

With the WHERE clause, the following operators can be used:

| Operator | Description |
|---|---|
| = | Equal |
| <> | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| BETWEEN | Between an inclusive range |
| LIKE | Search for a pattern |
| IN | If you know the exact value you want to return for at least one of the columns |

Examples:

```
select * from CUSTOMER where AreaCode>30
```