

Introduction to SQL

SQL (Structured Query Language) is a database computer language designed for managing data in relational database management systems (RDBMS).

SQL, is a standardized computer language that was originally developed by IBM for querying, altering and defining relational databases, using declarative statements.

What can SQL do?

- ☞ SQL can execute queries against a database
- ☞ SQL can retrieve data from a database
- ☞ SQL can insert records in a database
- ☞ SQL can update records in a database
- ☞ SQL can delete records from a database
- ☞ SQL can create new databases
- ☞ SQL can create new tables in a database
- ☞ SQL can create stored procedures in a database
- ☞ SQL can create views in a database
- ☞ SQL can set permissions on tables, procedures, and views

There are lots of different database systems, or DBMS – Database Management Systems, such as:

- ☞ **Microsoft SQL Server**
- ☞ **Oracle**
- ☞ **MySQL** (Oracle, previously Sun Microsystems)
- ☞ **Microsoft Access**
- ☞ **IBM DB2**
- ☞ **Sybase**

Data Definition Language (DDL):

The **Data Definition Language (DDL)** manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME and DROP statements:

- ☐ **CREATE** creates an object (a table, for example) in the database.
- ☐ **DROP** deletes an object in the database, usually irretrievably.
- ☐ **ALTER** modifies the structure an existing object in various ways—for example, adding a column to an existing table.

Data Manipulation Language (DML):

The **Data Manipulation Language (DML)** is the subset of SQL used to add, update and delete data.

The acronym **CRUD** refers to all of the major functions that need to be implemented in a

relational database application to consider it complete.

Each letter in the acronym can be mapped to a standard SQL statement:

Operation SQL Description

Create INSERT INTO inserts new data into a database

Read (Retrieve) SELECT extracts data from a database

Update UPDATE updates data in a database

Delete (Destroy) DELETE deletes data from a database

Introduction to SQL Server

SQL Server consists of a **Database Engine** and a **Management Studio** (and lots of other stuff

which we will not mention here). The Database engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server). The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).

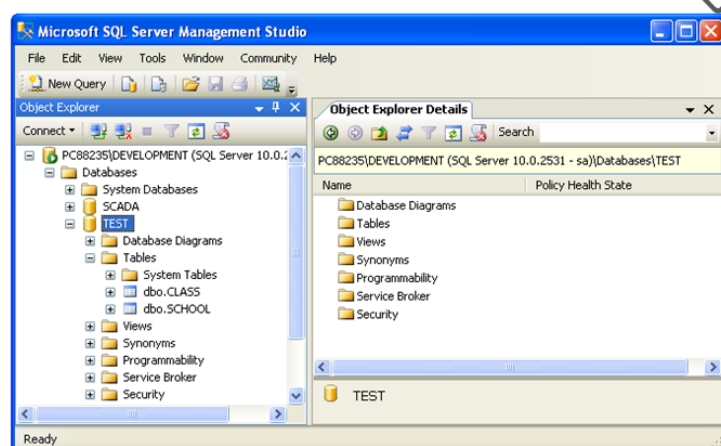
Database Engine



A Service running on the computer in the background



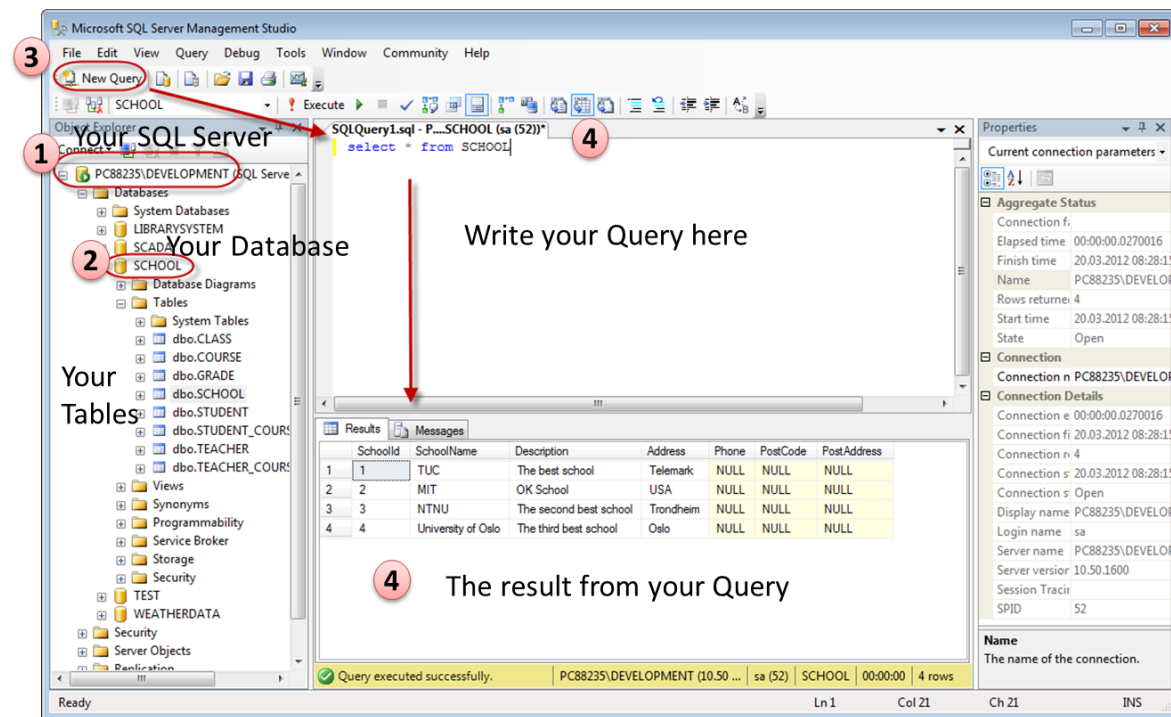
Management Studio



A Graphical User Interface to the database used for configuration and management of the database

SQL Server Management Studio

SQL Server Management Studio is a GUI tool included with SQL Server for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. As mentioned earlier, version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express.

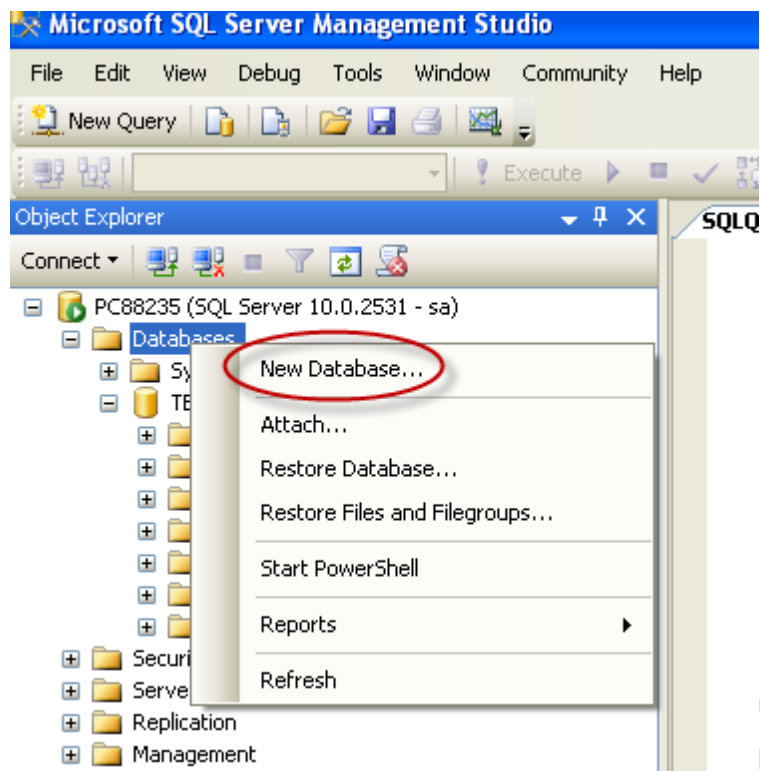


When creating SQL commands and queries, the “Query Editor” (select “New Query” from the Toolbar) is used (shown in the figure above).

With SQL and the “Query Editor” we can do almost everything with code, but sometimes it is also a good idea to use the different Designer tools in SQL to help us do the work without coding (so much).

Create a new Database

It is quite simple to create a new database in Microsoft SQL Server. Just right-click on the “Databases” node and select “New Database...”



New Database

Select a page

General

Options

Filegroups

Script

Help

Database name:

Owner:

<default>

...

☐ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
	Rows Data	PRIMARY	3	By 1 MB, unrestricted growth
_log	Log	Not Applicable	1	By 10 percent, unrestricted gr

Progress

Ready

Add

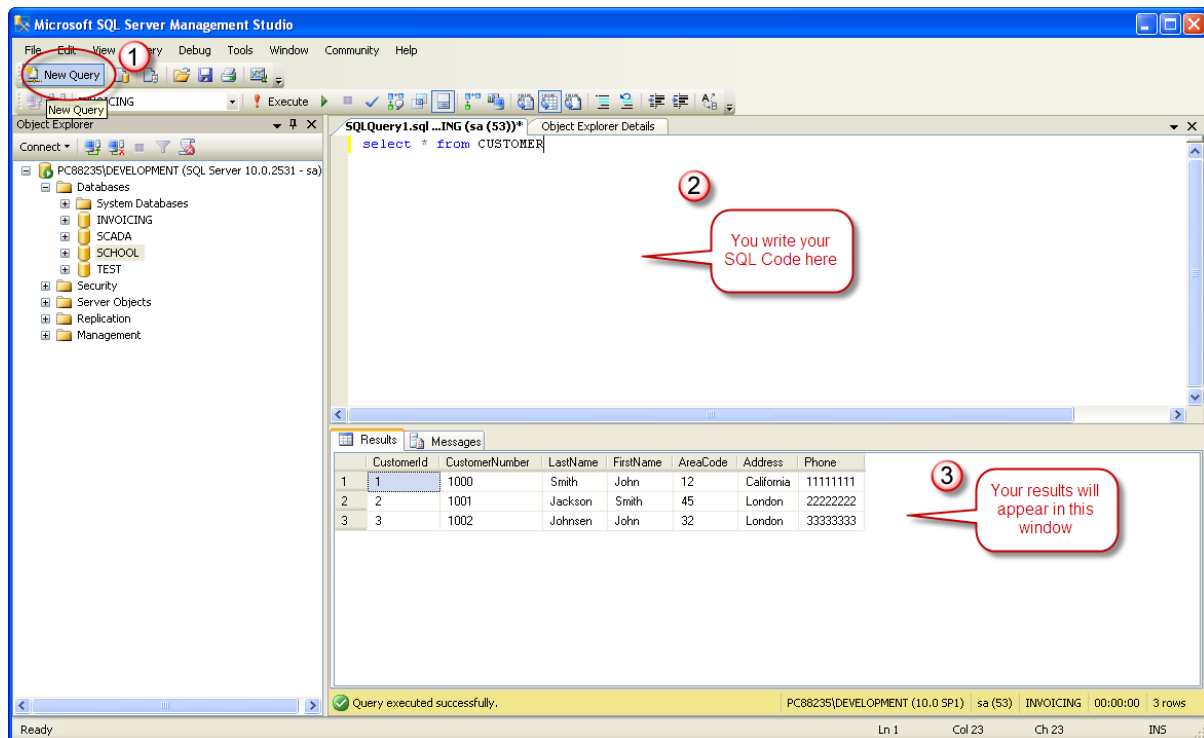
Remove

OK

Cancel

Queries:

In order to make a new SQL query, select the “New Query” button from the Toolbar.



The **CREATE TABLE** statement is used to create a table in a database.

Syntax:

```
CREATE TABLE table_name  
(  
  column_name1 data_type,  
  column_name2 data_type,  
  column_name3 data_type,  
  ....  
)
```

The data type specifies what type of data the column can hold.

You have special data types for numbers, text dates, etc.

Examples:

- ☐ Numbers: **int**, **float**
- ☐ Text/Stings: **varchar(X)** – where X is the length of the string
- ☐ Dates: **datetime**

Example:

We want to create a table called “CUSTOMER” which has the following columns and data types:

	Column Name	Data Type	Allow Nulls
▶?	CustomerId	int	<input type="checkbox"/>
	CustomerNumber	int	<input type="checkbox"/>
	LastName	varchar(50)	<input type="checkbox"/>
	FirstName	varchar(50)	<input type="checkbox"/>
	AreaCode	int	<input checked="" type="checkbox"/>
	Address	varchar(50)	<input checked="" type="checkbox"/>
	Phone	varchar(20)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Best practice:

When creating tables you should consider following these guidelines:

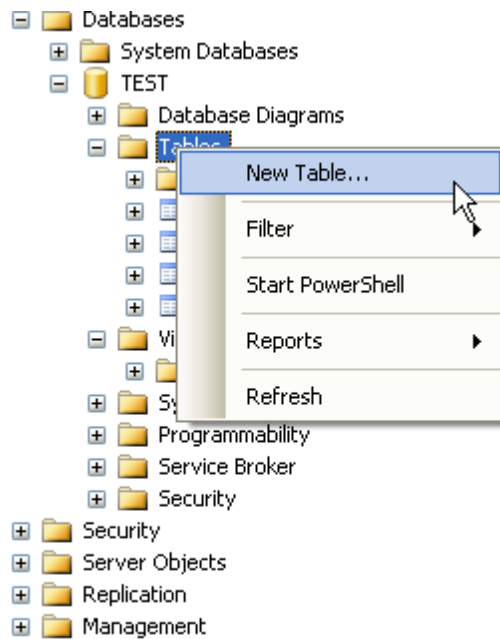
- Tables: Use upper case and singular form in table names – not plural, e.g., “STUDENT” (not students)
- Columns: Use Pascal notation, e.g., “StudentId”
- Primary Key:
If the table name is “COURSE”, name the Primary Key column “CourseId”, etc.
- “Always” use Integer and Identity(1,1) for Primary Keys. Use UNIQUE constraint for other columns that needs to be unique, e.g. RoomNumber
- Specify Required Columns (NOT NULL) – i.e., which columns that need to have data or not
- Standardize on few/these Data Types: int, float, varchar(x), datetime, bit
- Use English for table and column names
- Avoid abbreviations! (Use RoomNumber – not RoomNo, RoomNr, ...)

Create Tables using the Designer Tools

Even if you can do “everything” using the SQL language, it is sometimes easier to do it in the designer tools in the Management Studio in SQL Server.

Instead of creating a script you may as well easily use the designer for creating tables.

Step1: Select “New Table ...”:



Step2: Next, the table designer pops up where you can add columns, data types, etc.

	Column Name	Data Type	Allow Nulls
▶	CustomerId	int	<input type="checkbox"/>
	CustomerNumber	int	<input type="checkbox"/>
	LastName	varchar(50)	<input type="checkbox"/>
	FirstName	varchar(50)	<input type="checkbox"/>
	AreaCode	int	<input checked="" type="checkbox"/>
	Address	varchar(50)	<input checked="" type="checkbox"/>
	Phone	varchar(20)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

In this designer we may also specify Column Names, Data Types, etc.

Step 3: Save the table by clicking the Save button.

SQL Constraints

Constraints are used to limit the type of data that can go into a table.

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.

PRIMARY KEY

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain unique values. It is normal to just use running numbers, like 1, 2, 3, 4, 5, ... as values in Primary Key column. It is a good idea to let the system handle this for you by specifying that the Primary Key should be set to **identity(1,1)**. IDENTITY(1,1) means the first value will be 1 and then it will increment by 1.

Each table should have a primary key, and each table can have only ONE primary key.

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,
  AreaCode int NULL,
  Address varchar(50) NULL,
  Phone varchar(50) NULL,
)
GO
```

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

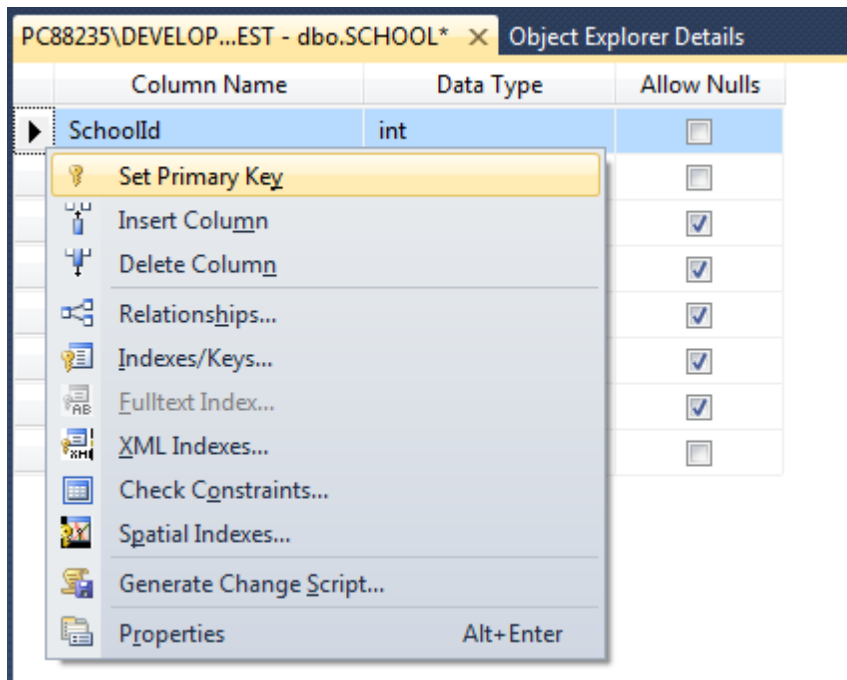
	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000					111111
2	2	1001					222222
3	3	1002	Johnson	John	52	London	33333333

Primary Keys must contain unique numbers like this

Setting Primary Keys in the Designer Tools:

If you use the Designer tools in SQL Server, you can easily set the primary Key in a table just by right-click and select “Set primary Key”.

The

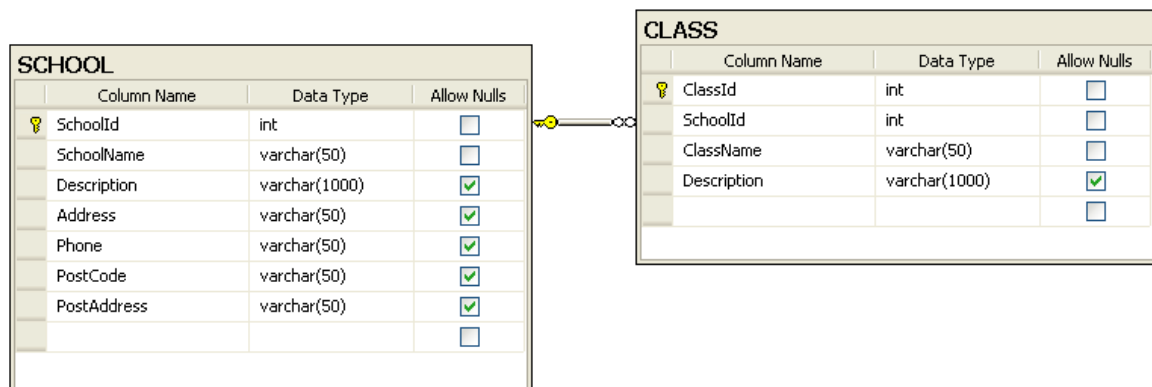


The primary Key column will then have a small key in front to illustrate that this column is a Primary Key.

FOREIGN KEY

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

Example:

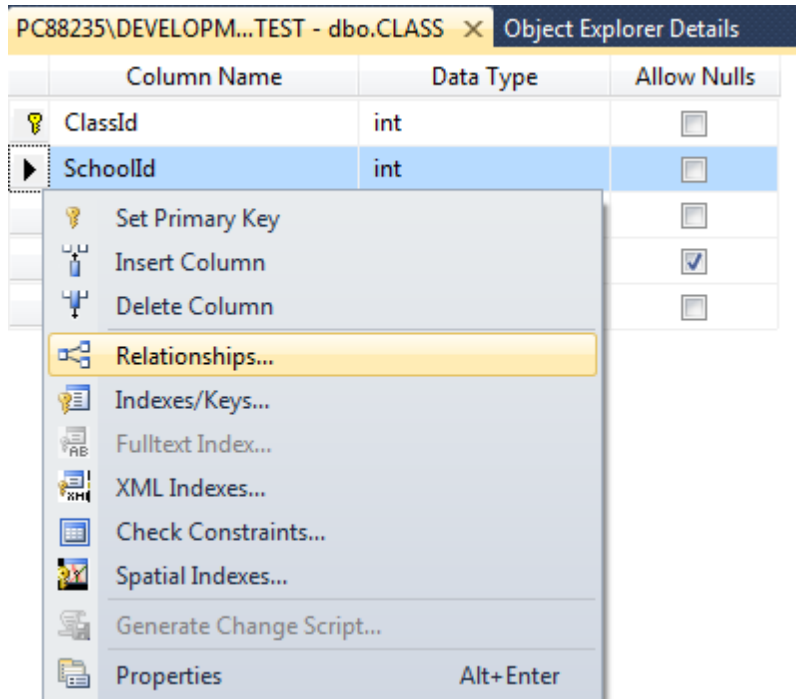


The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

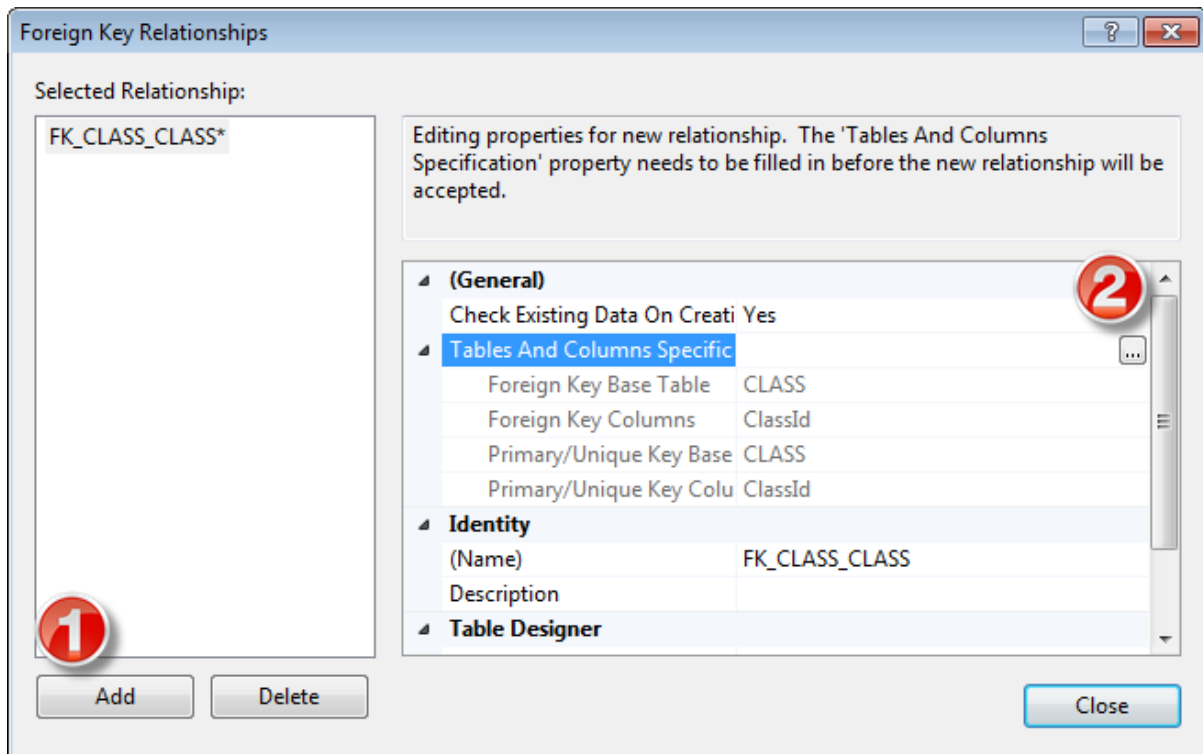
The FOREIGN KEY constraint also prevents that invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Setting Foreign Keys in the Designer Tools:

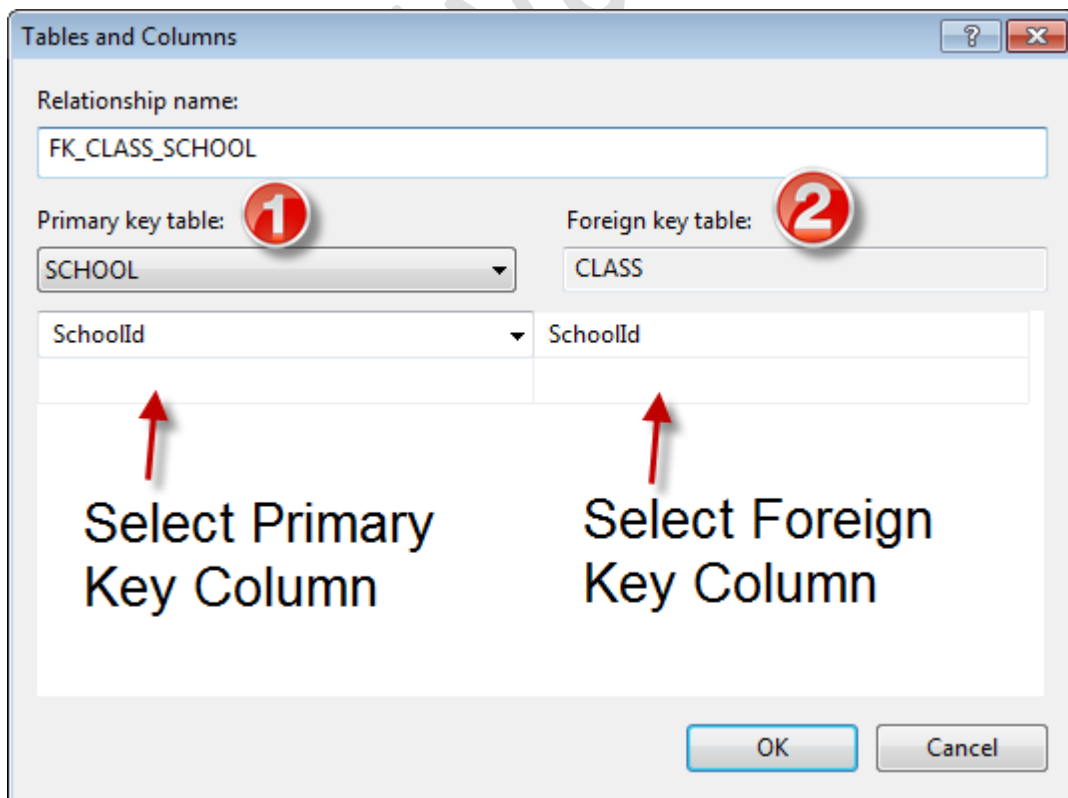
If you want to use the designer, right-click on the column that you want to be the Foreign Key and select “**Relationships...**”:



The following window pops up (Foreign Key Relationships):
Click



Click on the "Add" button and then click on the small "...". Then the following window pops up (Tables and Columns):



Here you specify the primary Key Column in the Primary Key table and the Foreign Key Column in the Foreign Key table.

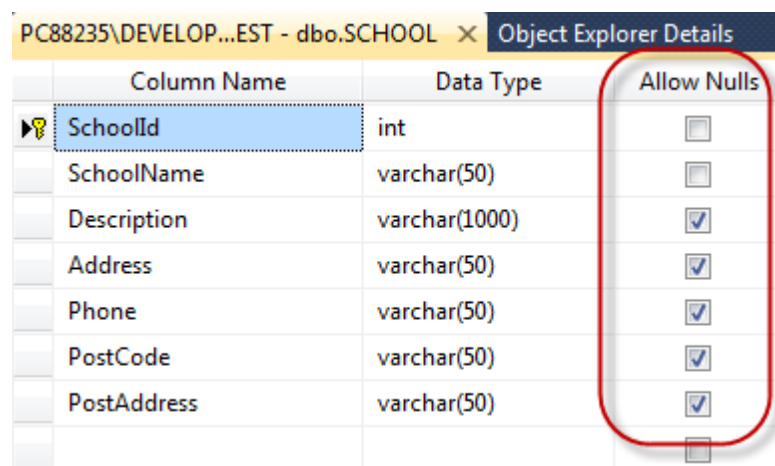
NOT NULL / Required Columns

The NOT NULL constraint enforces a column to NOT accept NULL values.

The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

Setting NULL/NOT NULL in the Designer Tools:

In the Table Designer you can easily set which columns that should allow NULL or not



Column Name	Data Type	Allow Nulls
SchoolId	int	<input type="checkbox"/>
SchoolName	varchar(50)	<input type="checkbox"/>
Description	varchar(1000)	<input checked="" type="checkbox"/>
Address	varchar(50)	<input checked="" type="checkbox"/>
Phone	varchar(50)	<input checked="" type="checkbox"/>
PostCode	varchar(50)	<input checked="" type="checkbox"/>
PostAddress	varchar(50)	<input checked="" type="checkbox"/>

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.

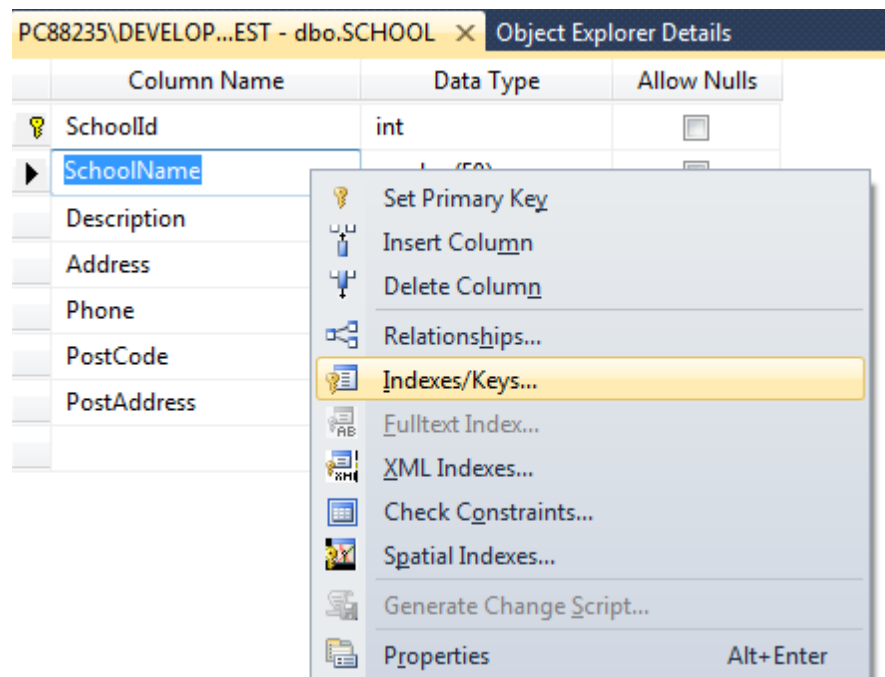
Note! You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

We see that the "CustomerNumber" is set to UNIQUE, meaning each customer must have a unique Customer Number. Example:

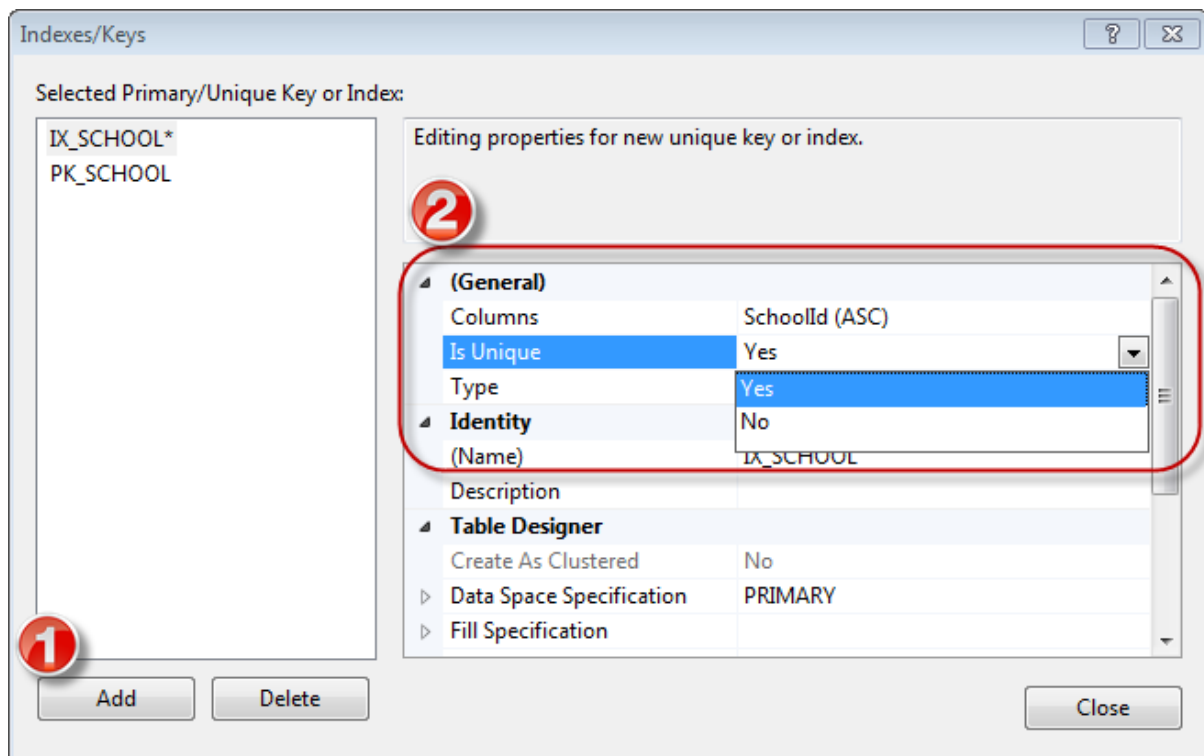
	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	2	1001	Jackson	Smith	45	London	22222222
3	3	1002	Johnsen	John	32	London	33333333

Setting UNIQUE in the Designer Tools:

If you want to use the designer, right-click on the column that you want to be UNIQUE and select “Indexes/Keys...”:



Then click “Add” and then set the “Is Unique” property to “Yes”:



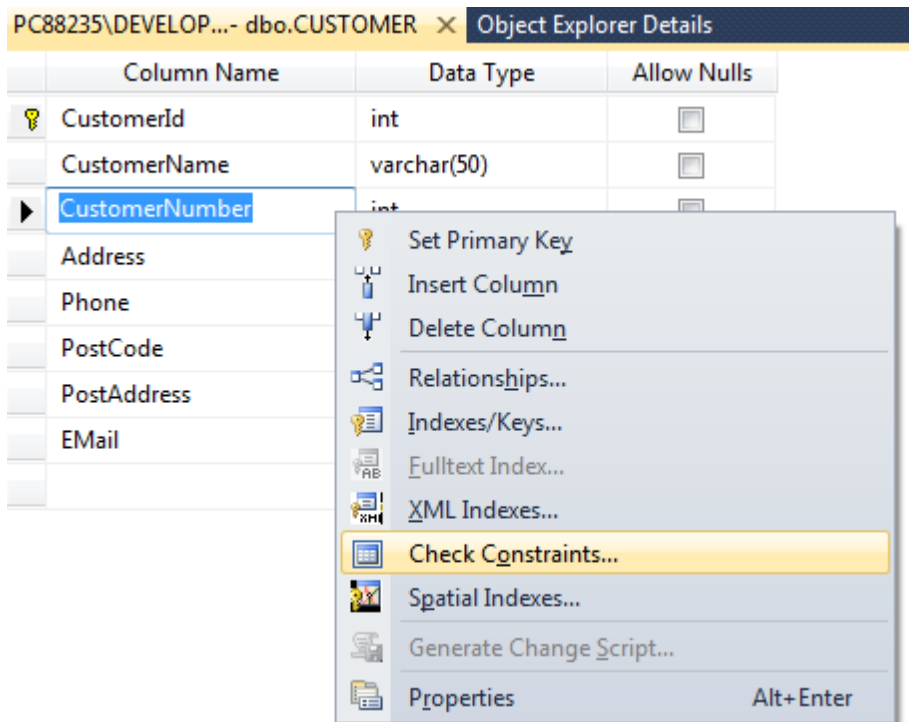
CHECK

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.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

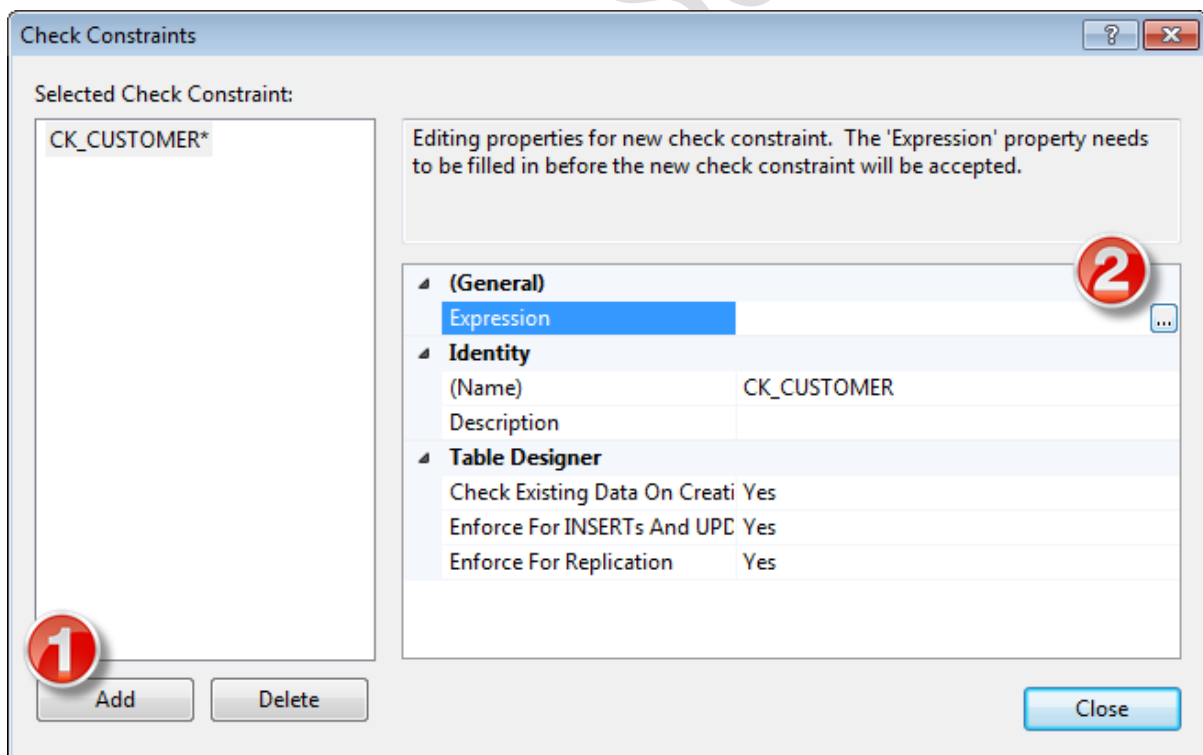
Setting CHECK constraints in the Designer Tools:

If you want to use the designer, right-click on the column where you want to set the constraints and select "**Check Constraints...**":

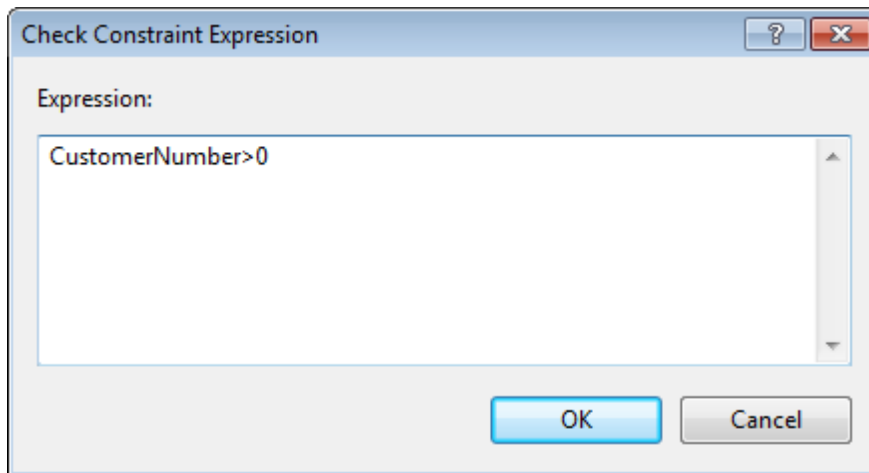


Structured Query Language (SQL)

Then click "Add" and then click "..." in order to open the Expression window:



In the Expression window you can type in the expression you want to use:

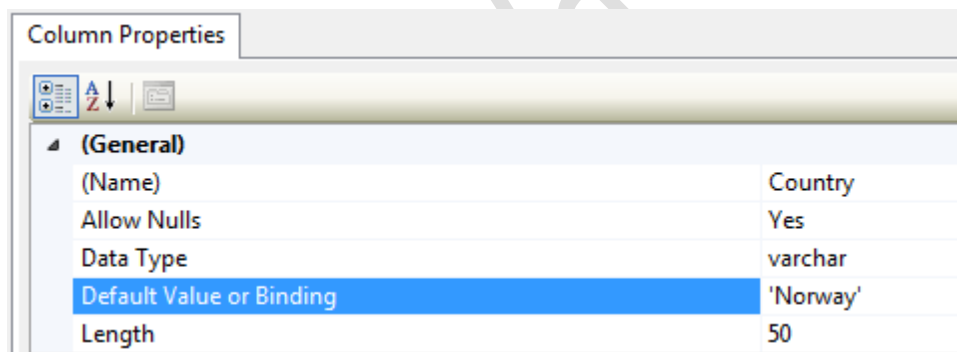


DEFAULT

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.

Setting DEFAULT values in the Designer Tools:

Select the column and go into the "Column Properties":



AUTO INCREMENT or IDENTITY

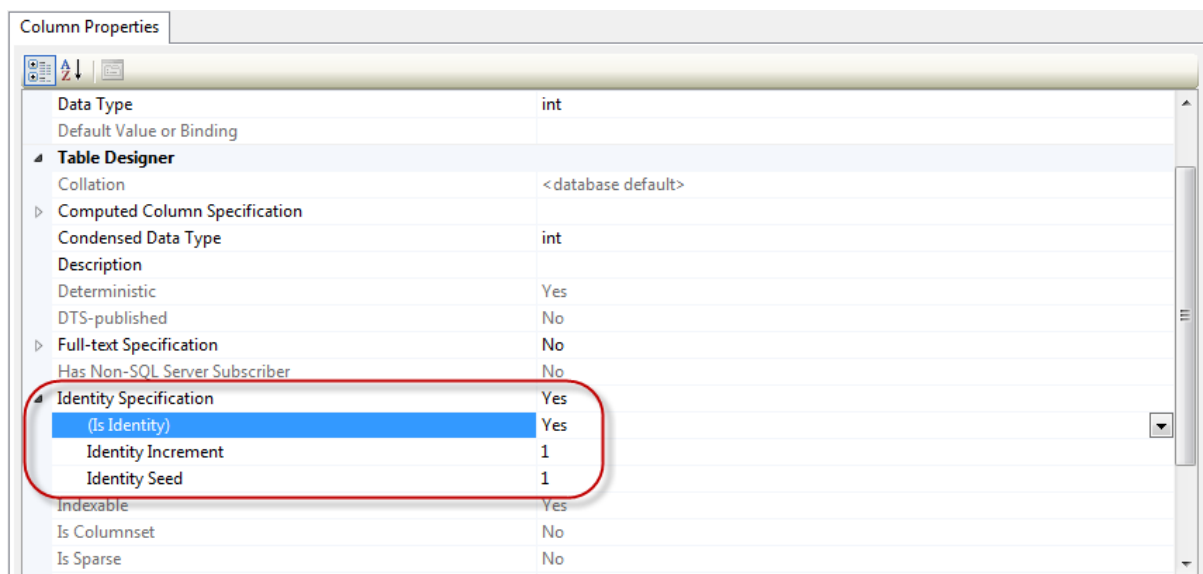
Very often we would like the value of the primary key field to be created automatically every time a new record is inserted.

As shown below, we use the IDENTITY() for this. IDENTITY(1,1) means the first value will be 1 and then it will increment by 1.

Setting identity(1,1) in the Designer Tools:

We can use the designer tools to specify that a Primary Key should be an identity column that is automatically generated by the system when we insert data in to the table.

Click on the column in the designer and go into the Column Properties window:



ALTER TABLE

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

INSERT INTO

The INSERT INTO statement is used to insert a new row in a table.

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

Example:

```
INSERT INTO CUSTOMER VALUES ('1000', 'Smith', 'John', 12,
'California', '11111111')
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
```

VALUES (value1, value2, value3,...)

Insert Data Only in Specified Columns:

It is also possible to only add data in specific columns.

Example:

```
INSERT INTO CUSTOMER (CustomerNumber, LastName, FirstName)
VALUES ('1000', 'Smith', 'John')
```

Note! You need at least to include all columns that cannot be NULL.

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
```

Before update:

	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	2	1001	Jackson	Smith	45	London	22222222
3	3	1002	Johnsen	John	32	London	33333333

After update:

	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	2	1001	Jackson	Smith	46	London	22222222
3	3	1002	Johnsen	John	32	London	33333333

If you don't include the WHERE clause the result becomes:

	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	46	California	11111111
2	2	1001	Jackson	Smith	46	London	22222222
3	3	1002	Johnsen	John	46	London	33333333

→ So make sure to include the WHERE clause when using the UPDATE command!

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
```

Before delete:

	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	2	1001	Jackson	Smith	45	London	22222222
3	3	1002	Johnsen	John	32	London	33333333

After delete:

	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	3	1002	Johnsen	John	32	London	33333333

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!