**SQL Introduction – 1**

**DDL (Data Definition Language)**

The SQL CREATE DATABASE Statement

The CREATE DATABASE statement is used to create a database.

SQL CREATE DATABASE Syntax

CREATE DATABASE *dbname*;

SQL CREATE DATABASE Example

The following SQL statement creates a database called "my\_db":

CREATE DATABASE my\_db;

Database tables can be added with the CREATE TABLE statement.

The SQL CREATE TABLE Statement

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

Tables are organized into rows and columns; and each table must have a name.

SQL CREATE TABLE Syntax

CREATE TABLE *table\_name*
(
*column\_name1 data\_type*(*size*),
*column\_name2 data\_type*(*size*),
*column\_name3 data\_type*(*size*),
....
);

The column\_name parameters specify the names of the columns of the table.

The data\_type parameter specifies what type of data the column can hold (e.g. varchar, integer, decimal, date, etc.).

The size parameter specifies the maximum length of the column of the table.

SQL CREATE TABLE Example

Now we want to create a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City.

We use the following CREATE TABLE statement:

Example

CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);

The PersonID column is of type int and will hold an integer.

The LastName, FirstName, Address, and City columns are of type varchar and will hold characters, and the maximum length for these fields is 255 characters.

The empty "Persons" table will now look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PersonID** | **LastName** | **FirstName** | **Address** | **City** |
|   |   |   |   |   |

**Tip:** The empty table can be filled with data with the INSERT INTO statement.

## SQL Constraints

SQL constraints are used to specify rules for the data in a table.

 If there is any violation between the constraint and the data action, the action is aborted by the constraint.

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

### SQL CREATE TABLE + CONSTRAINT Syntax

CREATE TABLE table\_name
(
column\_name1 data\_type(size) constraint\_name,
column\_name2 data\_type(size) constraint\_name,
column\_name3 data\_type(size) constraint\_name,
....
);

 In SQL, we have the following constraints:

* **NOT NULL** - Indicates that a column cannot store NULL value
* **UNIQUE** - Ensures that each row for a column must have a unique value
* **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have an unique identity which helps to find a particular record in a table more easily and quickly
* **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table
* **CHECK** - Ensures that the value in a column meets a specific condition
* **DEFAULT** - Specifies a default value when specified none for this column

## SQL NOT NULL Constraint

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.

The following SQL enforces the "P\_Id" column and the "LastName" column to not accept NULL values:

## Example

CREATE TABLE PersonsNotNull
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)

## SQL UNIQUE Constraint

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 that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

## SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "P\_Id" column when the "Persons" table is created:

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)

**MySQL:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
UNIQUE (P\_Id)
)

To allow naming of a UNIQUE constraint, and for defining a UNIQUE constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName)
)

## SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "P\_Id" column when the table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD UNIQUE (P\_Id)

To allow naming of a UNIQUE constraint, and for defining a UNIQUE constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName)

## To DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

**MySQL:**

ALTER TABLE Persons
DROP INDEX uc\_PersonID

**SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
DROP CONSTRAINT uc\_PersonID

## SQL PRIMARY KEY Constraint

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

Primary keys must contain unique values.

A primary key column cannot contain NULL values.

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

## SQL PRIMARY KEY Constraint on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "P\_Id" column when the "Persons" table is created:

**MySQL:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P\_Id)
)

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT pk\_PersonID PRIMARY KEY (P\_Id,LastName)
)

**Note:** In the example above there is only ONE PRIMARY KEY (pk\_PersonID). However, the value of the pk\_PersonID is made up of two columns (P\_Id and LastName).

## SQL PRIMARY KEY Constraint on ALTER TABLE

To create a PRIMARY KEY constraint on the "P\_Id" column when the table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD PRIMARY KEY (P\_Id)

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD CONSTRAINT pk\_PersonID PRIMARY KEY (P\_Id,LastName)

**Note:** If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

## To DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

**MySQL:**

ALTER TABLE Persons
DROP PRIMARY KEY

**SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
DROP CONSTRAINT pk\_PersonID

## SQL FOREIGN KEY Constraint

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

Let's illustrate the foreign key with an example. Look at the following two tables:

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

The "Orders" table:

|  |  |  |
| --- | --- | --- |
| **O\_Id** | **OrderNo** | **P\_Id** |
| 1 | 77895 | 3 |
| 2 | 44678 | 3 |
| 3 | 22456 | 2 |
| 4 | 24562 | 1 |

Note that the "P\_Id" column in the "Orders" table points to the "P\_Id" column in the "Persons" table.

The "P\_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "P\_Id" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

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

The FOREIGN KEY constraint also prevents 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.

## SQL FOREIGN KEY Constraint on CREATE TABLE

The following SQL creates a FOREIGN KEY on the "P\_Id" column when the "Orders" table is created:

**MySQL:**

CREATE TABLE Orders
(
O\_Id int NOT NULL,
OrderNo int NOT NULL,
P\_Id int,
PRIMARY KEY (O\_Id),
FOREIGN KEY (P\_Id) REFERENCES Persons(P\_Id)
)

**SQL Server / Oracle / MS Access:**

CREATE TABLE Orders
(
O\_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P\_Id int FOREIGN KEY REFERENCES Persons(P\_Id)
)

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Orders
(
O\_Id int NOT NULL,
OrderNo int NOT NULL,
P\_Id int,
PRIMARY KEY (O\_Id),
CONSTRAINT fk\_PerOrders FOREIGN KEY (P\_Id)
REFERENCES Persons(P\_Id)
)

## SQL FOREIGN KEY Constraint on ALTER TABLE

To create a FOREIGN KEY constraint on the "P\_Id" column when the "Orders" table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Orders
ADD FOREIGN KEY (P\_Id)
REFERENCES Persons(P\_Id)

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Orders
ADD CONSTRAINT fk\_PerOrders
FOREIGN KEY (P\_Id)
REFERENCES Persons(P\_Id)

## To DROP a FOREIGN KEY Constraint

To drop a FOREIGN KEY constraint, use the following SQL:

**MySQL:**

ALTER TABLE Orders
DROP FOREIGN KEY fk\_PerOrders

**SQL Server / Oracle / MS Access:**

ALTER TABLE Orders
DROP CONSTRAINT fk\_PerOrders

## SQL CHECK Constraint

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.

## SQL CHECK Constraint on CREATE TABLE

The following SQL creates a CHECK constraint on the "P\_Id" column when the "Persons" table is created. The CHECK constraint specifies that the column "P\_Id" must only include integers greater than 0.

**MySQL:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P\_Id>0)
)

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL CHECK (P\_Id>0),
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT chk\_Person CHECK (P\_Id>0 AND City='Sandnes')
)

## SQL CHECK Constraint on ALTER TABLE

To create a CHECK constraint on the "P\_Id" column when the table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD CHECK (P\_Id>0)

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ADD CONSTRAINT chk\_Person CHECK (P\_Id>0 AND City='Sandnes')

## To DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

**SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
DROP CONSTRAINT chk\_Person

**MySQL:**

ALTER TABLE Persons
DROP CHECK chk\_Person

## SQL DEFAULT Constraint

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.

## SQL DEFAULT Constraint on CREATE TABLE

The following SQL creates a DEFAULT constraint on the "City" column when the "Persons" table is created:

**My SQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons
(
P\_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)

## SQL DEFAULT Constraint on ALTER TABLE

To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:

**MySQL:**

ALTER TABLE Persons
ALTER City SET DEFAULT 'SANDNES'

**SQL Server / MS Access:**

ALTER TABLE Persons
ALTER COLUMN City SET DEFAULT 'SANDNES'

**Oracle:**

ALTER TABLE Persons
MODIFY City DEFAULT 'SANDNES'

## To DROP a DEFAULT Constraint

To drop a DEFAULT constraint, use the following SQL:

**MySQL:**

ALTER TABLE Persons
ALTER City DROP DEFAULT

**SQL Server / Oracle / MS Access:**

ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT