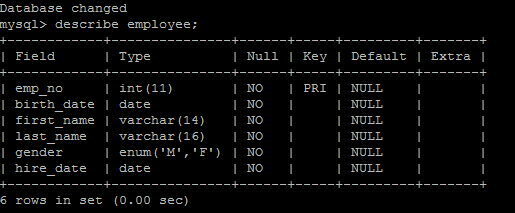
Week 3 SQL Tutorial

## Data Types

Each attribute in a table has a *data type* that is assigned at the time the table is created. The data type indicates what type of data the attribute is going to store – is it text, numbers, dates, etc. We can see the data types assigned to attributes in a table by describing that table:



W3schools.com has a great [basic MySQL data type overview](http://www.w3schools.com/sql/sql_datatypes.asp) that you should review carefully. Be sure to look for the table for **MySQL** data types, because they include information on other DBMSs, and data types can vary significantly from one DBMS to the next.

## Creating a Database

Databases in MySQL can be thought of as very similar to directories (i.e. folders) in the operating system. In the OS, files hold the data and directories keep the files organized. In MySQL, tables hold the data and databases keep related tables together and separate from other, unrelated tables.

To create a database is very simple. The only parameter is the database name:

CREATE DATABASE MyNewDatabase;

## Creating a Table

CREATE TABLE dept\_emp (

emp\_no INT(11) PRIMARY KEY,

dept\_no CHAR(4) NOT NULL,

from\_date DATE,

to\_date DATE

);

Note that NOT NULL designation indicates that the dept\_no attribute may not be omitted. Further, the PRIMARY KEY designation will (unsurprisingly) assign primary key responsibilities to the ‘emp\_no’ attribute. Note further that if you need to define a concatenated (multi-attribute) primary key – like you would for a weak entity or for a M:N relationship table – you must add the PK at the end of the create statement as detailed in the next example.

## Creating a Table with a Concatenated Primary Key

CREATE TABLE dept\_emp (

emp\_no INT(11),

dept\_no CHAR(4) NOT NULL,

from\_date DATE,

to\_date DATE,

PRIMARY KEY (EMP\_NO,DEPT\_NO)

);

While a primary key may be made up of multiple attributes, each table may have only ONE primary key. If you have multiple attributes that you want the DBMS to enforce as unique (and thus multiple candidates for the job of primary key), pick one as the PK and enforce uniqueness on the others with the UNIQUE constraint:

CREATE TABLE dept\_emp (

emp\_no INT(11) PRIMARY KEY,

license\_number INT(12) UNIQUE,

dept\_no CHAR(4) NOT NULL,

from\_date DATE,

to\_date DATE

);

## Letting the System Manage PK Values for You Automatically Using Auto-Increment

The job of a primary key is to provide a unique value for every record in a table. Normally we don’t care what that specific value is for any given record, but rather only want to be certain every value is unique. If you assign PK values manually, you need to check all of the existing PK values in the table to make sure you are not trying to assign a duplicate. This can be a pain. Luckily, we can have the system assign the values automatically:

MYSQL> CREATE TABLE PERSON (

-> ID INT(9) AUTO\_INCREMENT PRIMARY KEY,

-> NAME VARCHAR(20)

-> );

QUERY OK, 0 ROWS AFFECTED (0.00 SEC)

MYSQL> INSERT INTO PERSON (NAME) VALUES("MICHELLE");

QUERY OK, 1 ROW AFFECTED (0.00 SEC)

MYSQL> INSERT INTO PERSON (NAME) VALUES("ABDUL");

QUERY OK, 1 ROW AFFECTED (0.00 SEC)

MYSQL> SELECT \* FROM PERSON;

+----+----------+

| ID | NAME |

+----+----------+

| 1 | MICHELLE |

| 2 | ABDUL |

+----+----------+

2 ROWS IN SET (0.00 SEC)

As you can see, the system manages the values assigned for ID, taking the next available value automatically.

(You can set the number that the system starts at and how many it increments by from one record to the next using the system variables auto\_increment\_offset and auto\_increment\_increment.)

## Removing or Dropping a Table

drop table dept\_emp;

Dropping removes all of the data from the table as well as the definition of the table itself.

## Altering a Table

Altering a table involves changing the *definition* of that table (i.e. the table’s metadata). Changing a table’s actual values, on the other hand, is not done via alter but rather with an *UPDATE* command (described below)

### Renaming a Table

RENAME TABLE dept\_emp TO department\_employee;

### Adding an Attribute to an Existing Table

ALTER TABLE department\_employee ADD name VARCHAR(25);

Here we add a name attribute with a varchar data type that holds 25 characters max.

### Deleting an Attribute

ALTER TABLE department\_employee DROP name;

Here we drop the ‘name’ attribute we created in the previous step.

### Changing the Data Type of an Attribute

ALTER TABLE department\_employee MODIFY emp\_no CHAR(11);

Here we are changing the ‘emp\_no’ attribute from a numeric INT data type to a CHAR data type. Given that employee numbers, while numeric of course, are not the sort of numbers that you would ervery manipulate arithmetically, this could be a sensible change to make.

## Inserting Data into an existing Table

Switch over to the ‘employees’ database:

USE employees;  
  
Now insert a new value into the ‘department’ table:

INSERT INTO department VALUES ('d011', 'Information Technology');

This is the simplest form of insert statement. We must include data for ALL attributes, in the order in which they were originally defined. This order can be difficult to remember, plus often we want to leave attributes blank, so we can specify which attributes we want to insert and in what order, like this:

INSERT INTO department (dept\_name, dept\_no) VALUES ('d012', 'Accounts Payable');

We can also insert multiple records with a single insert statement, as follows:

INSERT INTO department (dept\_name, dept\_no) VALUES ('d013', 'distribution'), ('d014', 'purchasing'), ('d015', 'facilities');

## Deleting Data from a Table

To delete \*all\* data from a table (something that one very rarely actually wants to do!):

DELETE FROM department;

Far more often, we want to remove data that meets a certain condition, in much the same way that we specified “SELECT” statement query conditions previously:

DELETE FROM department WHERE dept\_name = 'Accounts Payable';

## Changing (Updating) Data Already in a Table

Suppose that the company has decided to rename the IT Department to the ‘Information Services” Department and we want the database to reflect that fact:

UPDATE department SET dept\_name = 'Information Services' WHERE dept\_name = 'Information Technology';

The “WHERE” conditions for an UPDATE statement can be as diverse and as complicated as they are in the SELECT statements we have seen previously, so be sure to go back to the Week 1 and 2 SQL tutorials and review those sections there, especially before attempting to complete the Week 3 SQL homework.

## Adding a primary key to an existing table defined without one

ALTER TABLE department ADD PRIMARY KEY(id);

## Adding a concatenated (multi-attribute) primary key to an existing table without a PK

ALTER TABLE department ADD PRIMARY KEY (id, name);

## Drop a primary key from an existing table

ALTER TABLE department DROP PRIMARY KEY;

Note that because there can be only one PK in a given table, you do NOT need to specify what the primary key consists of (and indeed, you’ll get a syntax error if you do).