Data Definition Language (DDL)

Overview

Regardless of database size and complexity, each database is comprised of tables. One of the first steps in creating a database is to create the tables that will store an organization's data. You will learn how to create a small database called company. This lesson describes the statements for creating, updating, and deleting tables.

Objectives

- Describe the data that can be stored in any of the character, numeric, date/time, and large object data types.
- Identify the table name and structure.
- Create a new table using the CREATE TABLE command.
- Add a column to an existing table.
- View a list of tables.
- Modify the definition of a column in an existing table.
- Delete a column from an existing table.
- Rename a table.
- Drop a table.

1. The Company Database

The company database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the mini world.

- The company is organized into departments. Each department has a unique name, number, and employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, number, and a single location.
- The database will store each employee's name, Social Security number, address, salary, sex (male or female), and birthdate. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- A Social Security number, or SSN, is a unique nine-digit identifier assigned to every U.S. citizen to keep track of their employment, benefits, and taxes. Other countries may have similar identification schemes, such as personal identification card numbers.
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birthdate, and relationship to the employee.

2. Storage Engines

A **storage engine** is a handler that manages a certain kind of table. The storage engine affects the way that the data is stored in the tables and whether or not the table supports a variety of features. Table 1 describes some storage engines. Each engine has distinct features, but only InnoDB enforces referential integrity.

InnoDB is the default engine for MySQL 5.7 for all the *MySQL* and *information_schema* databases. These two databases store and manage data using the MyISAM database engine. MyISAM was the default engine for earlier versions of the MySQL database.

Type	Description
ISAM	Legacy engine
MYISAM	Revision of ISAM engine with support for dynamic-length fields
INNODB	ACID-compliant transactional engine with support for foreign keys
MEMORY	Memory-based engine with support for hash indexes
CSV	Text-based engine for CSV recordsets
ARCHIVE	Archival storage (no modification of rows after insertion)
NDB	The engine for MySQL Cluster

Table 1 MySQL Storage Engines

It is also important to realize that there was no warning about the foreign key constraint with the MyISAM table. This is a MySQL feature. It can easily become an issue if you are not aware of this feature.

To see which storage engines the server knows about, use the SHOW ENGINES statement:

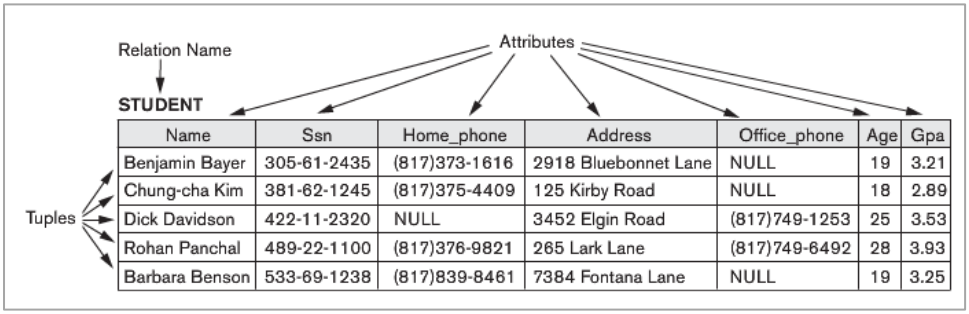
```
mysql> SHOW ENGINES;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.01 sec)
```

The engines table in the *Information_Schema* database provides the same information as SHOW ENGINES.

```
mysql> SELECT ENGINE
       FROM information_schema.engines;
+--------------------+
| ENGINE             |
+--------------------+
| InnoDB             |
| MRG_MYISAM         |
| MEMORY             |
| BLACKHOLE          |
| MyISAM             |
| CSV                |
| ARCHIVE            |
| PERFORMANCE_SCHEMA |
| FEDERATED          |
+--------------------+
9 rows in set (0.01 sec)
```

3. Table Creation

In relational database systems (DBS) data are represented using tables (relations). A query issued against the DBS also results in a table. Every database is composed of one or more tables. A table has the following structure:



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Example of a student Table (Relation)

A **table** is uniquely identified by its name and consists of **rows** that contain stored information. Each row contains exactly one **tuple** (or record). A table can have one or more columns. A **column** is made up of a column name and a data type, and it describes an attribute of the tuples. The structure of a table, also called **relation schema**, thus is defined by its attributes.

The type of information stored in a table is defined by the data types of the attributes at table creation time.

MySQL enables you to create tables, drop (remove) them, and change their structure with the CREATE TABLE, DROP TABLE, and ALTER TABLE statements.

To create a table, four pieces of information must be determined:

1. the table name,
2. the column (field) names
3. column datatypes
4. column sizes

Table Creation

All data in a relational database is stored in tables. When creating a new table, use the following rules for table names and column names:

- Must begin with a letter
- Must be 1 to 30 characters long
- Must contain only A -Z, a -z, 0 -9, _ (underscore), $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an MySQL Server reserved word

3.1. Naming Tables and Columns

Table and column names should be meaningful and reflect the nature of the data that is to be stored. If you are storing data about products, then the table should probably be named *product.* Similarly, if products are identified by a string of eight characters, then the column that stores product identification data should be named *product_number, product_id*, or *product_code*.

- Two tables belonging to the same database may not have the same name.
- Two columns in a table may not have the same name.
- The length of a table or column name is restricted to 64 characters.
- A name may consist of only letters, digits, and the special symbols: _ and $.
- Each name must begin with a letter or digit.
- Table and column names may not be reserved words.

3.2. Data Types Overview

MySQL supports several general categories of data values. These include numbers, string values, temporal values such as dates and times, spatial values, and the NULL value. The datatype you choose for a column determines the nature of the data that can be stored in the column. This is termed the **domain** of valid column values.

Identifying the type of data helps you verify that you input the correct data and allows you to manipulate data in ways specific to that data type. The main data types include VARCHAR, CHAR, INT, and DATE.

Each data type has several characteristics:

- What kind of values it can represent?
- How much storage space do values require?
- Are values fixed-length (all values of the type take the same amount of space) or variable-length (the amount of space depends on the particular value being stored)?
- How does MySQL compare and sort values of this type?
- Can the type be indexed? If so, how?

Literals

A literal is a fixed or unchanging value. Literals are used, for example, in conditions for selecting rows in SELECT statements and for specifying the values for a new row in INSERT statements.

```
SELECT first_name
FROM employee
WHERE ssn = 111111100;
```

Each literal has a particular data type, just like a column in a table. The names of the different types of literals are derived from the names of their respective data types as we use them in the CREATE TABLE statement.

There are several basic data types available for data to be stored in a table or to be used in expressions. The basic types are:

- numeric types
- String types
- date and time types

Some of the commonly used types are shown in Table 2.

Character strings

Strings are values such as ‘Madison, Wisconsin', 'patient shows improvement', or even '12345' (which looks like a number but isn’t). Usually, you can use either single or double quotes to surround a String value. String types have two general types:

- CHAR(99)
- VARCHAR (99)

CHAR is used for fixed length strings. We use CHAR (2) to store state abbreviations since they all have a 2 letter. If you are storing data where all of the data has the same number of characters- such as SSN, ISBN, some product codes, then CHAR is appropriate.

VARCHAR is used for variable length strings. We could use VARCHAR (25) to store customer last names assuming we won't need to store a name longer than 25 characters.

For both types—if you are defining a table column with CHAR or VARCHAR and try to insert a value longer than the defined length, you will get an error. CHAR defaults to a length of 1 if you do not state a length; meanwhile, VARCHAR requires a length.

Numeric Values

Numbers are values such as 48, 193.62, or -2.378E12. MySQL understands numbers specified as integers (which have no fractional part), fixed-point or floating-point values (which may have a fractional part), and bit-field values. When you specify a number, do not include commas as a separator. For example, 12345678.90 is legal, but 12,345,678.90 is not.

Numeric types are divided into two categories: exact-value and approximate-value numbers

- **Exact-value numbers** are used exactly as specified when possible.
  - Numbers that store integers like 15, 0, or -35.
  - Fixed point numbers which store numbers with a set number of digits such as a number with a total of 6 digits and two of those digits are after the decimal. So that type could store a value such as 1234.56 or 0.3 but not a value such as 34.5678.

  Approximate numbers are floating point numbers that store values such as 5.876E2.

Integers

We usually use INT which will store a number between approximately -2,000,000,000 and +2,000,000,000. There are other integer types which have different ranges: from the smallest to the largest tinyint (-128 to +127); smallint; mediumint, int; bigint.

Fixed precision

For this data type, you can specify how many digits you can have in front of and after the decimal point. For example, in DECIMAL(10,2), the first number (1) represents the precision and the second number (2) represents the scale. This means that columns with this data type can have a maximum of eight digits in front of the decimal point (scale minus precision) and two after it (the precision), or the range of this data type is -99,999,999.99 up to and including (≤) 99,999,999.99.

The Float Data Types

MySQL has two float data types: single precision and double precision. They differ in the amount of storage space that is reserved for any value. Because of this, they differ in range.

The range of the single-precision float data type is:
between -3.402823466E38 and -1.175494351E-38, and
between 1.175494351E-38 and 3.402823466E38.

The range of the double precision is bigger:
from -1.7976931348623157E308 to -2.2250738585072014E-308, and
from 2.2250738585072014E-308 to 1.7976931348623157E308.

The length that can be specified in a float data type determines the type of the float data type. It is single precision if the length is between 0 and 24, and double precision if the length is between 25 and 53.

Temporal (Date and Time) Values

Temporal values contain date or time values such as '2012-06-17' or '12:30:43'.

MySQL also understands combined date/time values, such as '2012-06-17 12:30:43'. MySQL displays dates in year-month-day order, and input values must be given in that order. For combined date and time values, it is permitted to specify a 'T' character rather than space between the date and time. For example, '2008-12-31T12:00:00.'

The syntax for time or combined date and time values permits fractional seconds part following the time, consisting of a decimal point and up to 6 digits (microseconds) precision. For example, '12:30:15.000045' or '2008-06-15 10:30:12.5'.

Boolean Values

In expressions, zero is considered false. For any nonzero, the non-NULL value is considered true. The special constants TRUE and FALSE evaluate to 1 and 0, respectively. They are not case-sensitive. There are other special purpose data types which we will look at over the semester.

The NULL Value

NULL represents the values of attributes that may be unknown or may not apply to a record.

Meanings for NULL values:

- Value unknown.
- Value exists but is not available.
- Attribute does not apply to this record (also known as value undefined).

You can insert NULL values into tables, retrieve them from tables, and test whether a value is NULL. However, you cannot perform arithmetic on NULL values. If you try, the result is NULL.

The keyword NULL is written without quotes and is not case-sensitive. MySQL also treats a standalone \N (case sensitive) as NULL:

```
mysql> SELECT \N, ISNULL(\N);
+------+------------+
| NULL | ISNULL(\N) |
+------+------------+
| NULL |          1 |
+------+------------+
1 row in set, 2 warnings (0.01 sec)
```

Type	Example	Description
CHAR(length)	CHAR(10)	CHAR stores a fixed-length string. Fixed-length strings up to 255 characters
VARCHAR(size)	VARCHAR(100)	Variable-length strings up to 255 characters For example, VARCHAR (20) can be used to store a string of up to 20 characters in length.
DATE	DATE	DATE stores a date in 'CCYY-MM-DD' format.
DATETIME	DATETIME	DATETIME is a date and time value in 'CCYY-MM-DD hh:mm:ss' format.
INT	INT(5)	INT stores an integer. An integer is a whole number such as 1, 10, and 115. Signed values: −2147483648 to 2147483647 (−231 to 231−1); Unsigned values: 0 to 4294967295 (0 to 232−1)
DECIMAL(size,d)	DECIMAL(10,2)	The DECIMAL data type is a fixed-point type and calculations are exact.
TEXT	TEXT	TEXT is a small string.
ENUM		ENUM is an enumeration; each column value is assigned one enumeration member.
SET		SET is a set; each column value is assigned zero or more set members.

Table 2 Common MySQL data types Data

4. Working with Sequences

Many applications must generate unique numbers for identification purposes: member-ship numbers, customer IDs, and so forth. MySQL's mechanism for providing unique numbers is the AUTO_INCREMENT column attribute, which enables you to generate sequential numbers automatically.

General AUTO_INCREMENT Properties

AUTO_INCREMENT columns must be defined according to the following conditions:

- There can be only one column per table with the AUTO_INCREMENT attribute. This attribute should have an integer data type.
- The column must have a NOT NULL constraint. MySQL makes the column NOT NULL, even if you do not explicitly declare it that way.

Once created, an AUTO_INCREMENT column behaves like this:

- Inserting NULL into an AUTO_INCREMENT column causes MySQL to generate the next sequence number and insert it into the column.
- AUTO_INCREMENT sequences normally begin at 1 and increase monotonically, so successive rows inserted into a table get sequence values of 1, 2, 3, and so forth.
- Depending on the storage engine, it may be possible to set or reset the next sequence number explicitly or to reuse values deleted from the top end of the sequence.

5. Table Creation

The CREATE TABLE statement is used to add a new table to the database. This statement requires the table name, column names, and columns datatypes. The CREATE TABLE statement is used to create a table object. To create a new table, use the following syntax details:

- table is the name of the table
- column is the name of the column
- Data type is the column's data type and length
- DEFAULT expression specifies a default value if a value is omitted in the INSERT statement

CREATE TABLE syntax:

```
CREATE TABLE tablename (
columnname datatype [ column_constraint ] ,
columnname datatype [ column_constraint ] ,
…….
[ table_constraint ] ,
[ table_constraint ] ,…….);
```

6. Creating the Company Database

6.1. Creating the EMPLOYEE table

With a create table statement, several properties are defined, including the name of the table, the columns of the table, and the primary key.

- The name of the table is specified first: CREATE TABLE employee.
- The columns of a table are listed between brackets.
- For each column name, a data type is specified, as in CHAR, VARCHAR, INT, or DATE. The data type defines the type of value that may be entered into the specific column.
- A primary key of a table is a column (or combination of columns) in which every value can appear only once.

After entering the table name, you define the columns to be included in the table. The CREATE TABLE syntax requires enclosing the column list in parentheses. If the table contains more than one column, the name, data type, and width (if applicable) are listed for the first column before the next column is defined. Commas separate columns in the list.

Example 1*:* Enter the following SQL command to create the EMPLOYEE table in the company database.

```
mysql> USE COMPANY;
CREATE TABLE employee (
   fname VARCHAR(15) NOT NULL,
   minit VARCHAR(1),
   lname VARCHAR(15) NOT NULL,
   Ssn CHAR(9),
   bdate DATETIME,
   address VARCHAR(50),
   sex CHAR(10),
   Salary DECIMAL(10,2),
   superssn CHAR(9),
   dno INT(4),
   PRIMARY KEY (ssn),
   FOREIGN KEY (superssn) REFERENCES employee(ssn)
)ENGINE = INNODB;
```

Notice that when you create the EMPLOYEE table structures you set the stage for the enforcement of entity integrity rules by using: PRIMARY KEY(ssn).

6.2. Creating the DEPARTMENT Table

The department table will store information about departments within our company. Each department has a unique, four-digit, department number. Each department also has a department name, manager identifying the number, and date the manager was assigned to supervise the department.

Example 2: Enter the following command to create the DEPARTMENT table in the company database.

```
CREATE TABLE department (
  dname VARCHAR(25) NOT NULL,
  dnumber INT(4),
  mgrssn CHAR(9) NOT NULL,
  mgrstartdate DATETIME,
  PRIMARY KEY (dnumber),
  UNIQUE (dname),
  FOREIGN KEY (mgrssn) REFERENCES employee(ssn)
)ENGINE = INNODB ;
```

6.3. Creating the DEPT_LOCATIONS Table

Example 3: Enter the following SQL command to create the DEPT_LOCATIONS table in the company database.

```
CREATE TABLE dept_locations (
  dnumber   INT(4),
  dlocation VARCHAR(15),
  PRIMARY KEY (dnumber,dlocation),
  FOREIGN KEY (dnumber) REFERENCES department(dnumber)
)ENGINE = INNODB;
```

6.4. Creating the PROJECT Table

Example 4: Enter the following SQL command to create the PROJECT table in the company database.

```
CREATE TABLE project (
  pname VARCHAR(25) NOT NULL,
  pnumber INT(4),
  plocation VARCHAR(15),
  dnum INT(4) NOT NULL,
  PRIMARY KEY (pnumber),
  UNIQUE (pname),
  FOREIGN KEY (dnum) REFERENCES department(dnumber)
)ENGINE = INNODB;
```

As you create this structure, also notice that the NOT NULL constraint is used to ensure that the columns dept_number does not accept nulls.

6.5. Creating the WORKS_ON Table

Example 5: Enter the following SQL command to create the WORKS_ON table in the company database.

```
CREATE TABLE works_on (
  essn CHAR(9),
  pno INT(4),
  hours DECIMAL(4,1),
  PRIMARY KEY (essn,pno),
  FOREIGN KEY (essn) REFERENCES employee(ssn),
  FOREIGN KEY (pno) REFERENCES project(pnumber)
)ENGINE = INNODB;
```

6.6. Creating the DEPENDENT Table

Example 6*:* Enter the following SQL command to create the DEPENDENT table in the company database.

```
CREATE TABLE dependent (
      essn CHAR(9),
      dependent_name VARCHAR(15),
      sex CHAR,
      bdate DATE,
      relationship VARCHAR(8),
      PRIMARY KEY (essn , dependent_name),
      FOREIGN KEY (essn)
      REFERENCES employee (ssn)
)  ENGINE=INNODB;
```

7. Deleting a Table

The DROP TABLE command is used to remove a table from a database. The DROP TABLE command permanently deletes a table from the database schema. When you write a script file to create a database schema, it is useful to add DROP TABLE commands at the start of the file. **Primary and foreign key constraints** control the order in which you drop the tables – generally

DROP TABLE syntax:

```
DROP TABLE [IF EXISTS] table_name;
```

You drop in the reverse order of creation. The DROP commands for the company database are:

```
DROP TABLE IF EXISTS dependent;
DROP TABLE IF EXISTS dept_locations;
DROP TABLE IF EXISTS works_on;
DROP TABLE IF EXISTS project;
DROP TABLE IF EXISTS department;
DROP TABLE IF EXISTS employee;
```

What if the table already exists?

If we process a CREATE TABLE statement with a table name that already exists, MySQL returns an error message. Adding IF NOT EXISTS represses this error message.

8. Viewing Table Structures: DESCRIBE

To determine whether the table structure was created correctly, you can use the SQL command **DESCRIBE** table name to display the table's structure. To see the structure of the EMPLOYEE table, enter the command:

DESCRIBE TABLE syntax:

```
{DESCRIBE | DESC} table_name;
```

The output from DESCRIBE has three columns that show the structure of the table. These columns are as follows:

- **Field lists** the names of the columns contained in the table.
- **Null** indicates whether the column can store null values. If set to NOT NULL, the column cannot store a null value. If blank, the column can store a null value. In the preceding example, you can see that the SSN column cannot store null value.
- **Type** indicates the type of the column. In the preceding example, you can see that the type of the dno column is INT(4) and that the type of the fname column is VARCHAR(15).

Example 7*:* Enter the following SQL command to describe the EMPLOYEE table.

```
mysql> DESCRIBE employee;
```

```
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| fname    | varchar(15)   | NO   |     | NULL    |       |
| minit    | varchar(1)    | YES  |     | NULL    |       |
| lname    | varchar(15)   | NO   |     | NULL    |       |
| ssn      | char(9)       | NO   | PRI | NULL    |       |
| bdate    | datetime      | YES  |     | NULL    |       |
| address  | varchar(50)   | YES  |     | NULL    |       |
| sex      | char(10)      | YES  |     | NULL    |       |
| salary   | decimal(10,2) | YES  |     | NULL    |       |
| superssn | char(9)       | YES  | MUL | NULL    |       |
| dno      | int(4)        | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
10 rows in set (0.01 sec)
```

9. Modifying Existing Tables

All changes in the table structure are made by using the ALTER TABLE command, followed by a keyword that produces the specific change you want to make. Be aware that if data rows already exist in a table, then adding a new column will result in the column containing a null value for the existing rows.

Syntax of the ALTER TABLE command:

```
ALTER TABLE tablename
[ ADD ( columnname | table_constraint ,
[ columnname | table_constraint ,……. ] ) ]
[ MODIFY columnname, [columnname, ……. ] ]
[ DROP column | PRIMARY KEY | FOREIGN KEY | INDEX ]
[ RENAME [TO | AS] newtablename ]
[CHANGE [COLUMN] old_column_name new_column_name column_definition];
```

The ALTER TABLE statement is used to:

- add new fields and/or constraints to a table
- modify the definition of a field
- drop columns, constraints or indexes from a table
- rename a table

9.1. Alter Table … Add Command

Using an ADD clause with the ALTER TABLE command allows a user to add a new column to a table.

Example 8*:* Add a new column named city to the EMPLOYEE table*.*

```
mysql> ALTER TABLE employee
       ADD city VARCHAR(15);
```

```
mysql> DESC employee;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| fname    | varchar(15)   | NO   |     | NULL    |       |
| minit    | varchar(1)    | YES  |     | NULL    |       |
| lname    | varchar(15)   | NO   |     | NULL    |       |
| ssn      | char(9)       | NO   | PRI | NULL    |       |
| bdate    | datetime      | YES  |     | NULL    |       |
| address  | varchar(50)   | YES  |     | NULL    |       |
| sex      | char(10)      | YES  |     | NULL    |       |
| salary   | decimal(10,2) | YES  |     | NULL    |       |
| superssn | char(9)       | YES  | MUL | NULL    |       |
| dno      | int(4)        | YES  |     | NULL    |       |
| city     | varchar(15)   | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

Example 9: Add a primary key constraint named dependent_pk to the DEPENDENT table.

```
mysql > ALTER TABLE dependent
        ADD CONSTRAINT dependent_pk
        PRIMARY KEY(essn,dependent_name);
```

9.2. Alter Table … Modify Command

To change an existing column's definition, you can use a MODIFY clause with the ALTER TABLE command. The following list shows some of the column aspects you can modify using ALTER TABLE:

- change the size of a column (if the data type is one whose length may be changed, such as CHAR or VARCHAR)
- change the precision of a numeric column
- change the data type of a column
- change the default value of a column

Example 10: Change the data type of the column city in the EMPLOYEE table from varchar(15) to char (20).

```
mysql> ALTER TABLE employee
       MODIFY city CHAR(20);
mysql> DESC employee;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| fname    | varchar(15)   | NO   |     | NULL    |       |
| minit    | varchar(1)    | YES  |     | NULL    |       |
| lname    | varchar(15)   | NO   |     | NULL    |       |
| ssn      | char(9)       | NO   | PRI | NULL    |       |
| bdate    | datetime      | YES  |     | NULL    |       |
| address  | varchar(50)   | YES  |     | NULL    |       |
| sex      | char(10)      | YES  |     | NULL    |       |
| salary   | decimal(10,2) | YES  |     | NULL    |       |
| superssn | char(9)       | YES  | MUL | NULL    |       |
| dno      | int(4)        | YES  |     | NULL    |       |
| city     | char(20)      | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

9.3. Alter Table … Drop Column Command

To delete an existing column from a table, you can use the DROP COLUMN clause with the ALTER TABLE command. This clause deletes both the column and its contents, so it should be used with extreme caution.

Example 11: Drop the column city from the EMPLOYEE table.

```
mysql> ALTER TABLE employee
       DROP COLUMN city;
mysql> DESC employee;
```

Field	Type	Null	Key	Default	Extra
fname	varchar(15)	NO		NULL	
minit	varchar(1)	YES		NULL	
lname	varchar(15)	NO		NULL	
ssn	char(9)	NO	PRI	NULL	
bdate	datetime	YES		NULL	
address	varchar(50)	YES		NULL	
sex	char(10)	YES		NULL	
salary	decimal(10,2)	YES		NULL	
superssn	char(9)	YES	MUL	NULL	
dno	int(4)	YES		NULL	

```
10 rows in set (0.00 sec)
```

You should keep the following rules in mind when using the DROP COLUMN clause:

- Unlike using ALTER TABLE with the ADD or MODIFY clauses, a DROP COLUMN clause can reference only one column.
- A primary key column cannot be dropped from a table.

9.4. Renaming a Table

MySQL allows changing the name of any table you own by using the RENAME . . . TO command.

Example 12: Change the name of the EMPLOYEE table to dept_employee.

```
mysql> RENAME TABLE employee
       TO dept_employee;
or
mysql> ALTER TABLE dept_employee
       RENAME TO employee;
```

9.5. Altering Field Names and Properties

A CHANGE clause can be used to alter a field's name, type, and properties, simply by using a new field definition instead of the original one.

Example 13: Change the field named fname defined as varchar(15) to a field named first_name with definition varchar(20).

```
mysql> ALTER TABLE employee
       CHANGE fname first_name varchar(20);
Query OK, 0 rows affected (1.14 sec)
```

10. Truncating a Table

With the TRUNCATE TABLE command, only the data is removed, whereas the DROP TABLE command deletes the data and removes the table as well.

Example 14: Truncate the DEPENDENT table.

```
mysql> TRUNCATE TABLE dependent;
Query OK, 0 rows affected (0.15 sec)
```

Example 15: Delete the DEPENDENT table.

```
mysql> DROP TABLE dependent;
```

Note: If what you actually intended was to empty the table of all records, use the TRUNCATE TABLE statement instead, which drops the table and then re-creates it.

11. CREATE TABLE ... LIKE Syntax

Use CREATE TABLE ... LIKE to create an empty table based on the definition of another table, including any column attributes and indexes defined in the original table:

```
CREATE TABLE new_tbl LIKE orig_tbl;
```

The copy is created using the same version of the table storage format as the original table.

```
mysql> CREATE TABLE employee_backup LIKE employee;
```

SQL Keywords

Keywords are words that have a predefined meaning in SQL. Keywords must be spelled as shown. Uppercase letters are used above to depict keywords. However, in practice, keywords may be entered in upper or lowercase letters; however, most information technology professionals follow the practice of always entering keywords in uppercase.

In some cases, keywords can be abbreviated. The allowed abbreviation is shown in uppercase letters with the remainder shown in lowercase, which means you can use either the full word or only the uppercase part.

`DESC employee;` can be entered as either DESC or DESCRIBE.

SQL syntax guidelines

- Do not forget to call USE command to select the particular database you want to work with.
- If you forget the name of your database, you can query for database names using `SHOW DATABASES;`
- If you forget the name of a table, you can query for the names of all tables in the active database using `SHOW TABLES;`
- SQL commands are case-insensitive. Database and table names may or may not be case-sensitive depending on which platform you are using (e.g., Windows vs. Linux). Column names are case-insensitive.
- SQL statements begin with a keyword such as SELECT and are composed of one or more clauses which begin with keywords such as FROM or WHERE clause. You should avoid using the SQL keywords as table or column names.
- Use commas to separate lists of items such as the columns to be displayed.
- In MySQL, text comparisons are case-insensitive.
- If you use a constant (a literal) in an SQL statement, it may need to be delimited.
  - Numbers do not use delimiters.
  - Do not use commas or currency symbols in numbers
  - Text literals are enclosed in single quotes (').
  - In MySQL, dates should be enclosed in single quotes; the default syntax for date follows the pattern 2009-06-29. We can use other date formats.
- You can use comments in your SQL statement. There are two forms of comments:
  1. The multi-line comment is delimited by /* comment */ . Start with a slash, an asterisk, and space.
  2. The single line comment is indicated by two hyphens followed by a space. This is the ANSI standard comment.

Viewing Database, Table, and Field Information

- The SHOW DATABASES statement displays a list of databases on the server.
- The SHOW TABLES statement displays a list of tables in a database.
- The DESCRIBE statement displays the structure of a table.
- The SHOW CREATE TABLE statement retrieves the SQL statements originally used to create the table.
- The SHOW ENGINES statement retrieves a list of available storage engines.

Key Terms

1. **ALTER** – a command used to alter the structure of a database object such as a table.
2. **CHAR** – a type of column data used to store fixed-length character data.
3. **Composite primary key** - when primary key consists of two or more columns in order to identify each row uniquely.
4. **CREATE TABLE** – a statement used to create a table in a database.
5. **DATE** – a type of column data that can only store valid dates.
6. **Domain** – describes the nature of the data that can be stored in a column; a domain of valid column values.
7. **DROP** – a command used to drop a database object such as tables.
8. **Foreign keys (FKs)** - columns in one table that reference primary key (PK) values in another table or in the same table.
9. **INT** – a type of column data used to store numeric data values.
10. **Not NULL** – a constraint meaning every data row must have a value for the column.
11. **Primary key** - a column or set of columns that uniquely identifies rows of data that are stored in a table.
12. **RENAME** – a command used to rename a database object such as a table.
13. **Table** – an object that stores organizational data.
14. **UPDATE** – a command used to alter data values for rows that already exist within a table.
15. **VARCHAR** – a type of column data used to store variable-length, character data up to 255 characters per column entry.