What is Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

So nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

A **Relational DataBase Management System (RDBMS)** is a software that:

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology:

Before we proceed to explain MySQL database system, let's revise few definitions related to database.

- **Database:** A database is a collection of tables, with related data.
- **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column:** One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy:** Storing data twice, redundantly to make the system faster.
- **Primary Key:** A primary key is unique. A key value can not occur twice in one table. With a key, you can find at most one row.
- **Foreign Key:** A foreign key is the linking pin between two tables.
- **Compound Key:** A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index:** An index in a database resembles an index at the back of a book.
- **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

mysql> SHOW DATABASES;

1) Creating a database

mysql> **CREATE database 134a;**

Query OK, 1 row affected (0.00 sec)

2) Deleting a database

mysql> **DROP database 134a;**

Query OK, 0 rows affected (0.00 sec)

3) After we have created the database we use the USE statement to

change the current database;

mysql> **USE 134a;**

Database changed

MySQL uses many different data types broken into three categories: numeric, date and time, and string types.

Numeric Data Types:

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions:

- **INT** - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not

required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

- **DOUBLE(M,D)** - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String Types:

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

4) Creating a table in the database is achieved with the CREATE table

statement

mysql> **CREATE TABLE president (**

-> last_name varchar(15) not null,

-> first_name varchar(15) not null,

-> state varchar(2) not null,

-> city varchar(20) not null,

-> birth date not null default '0000-00-00',

-> death date null

->);

Query OK, 0 rows affected (0.00 sec)

5) To see what tables are present in the database use the SHOW tables:

mysql> **SHOW tables;**

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

| Tables_in_134a |

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

| president |

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

1 row in set (0.00 sec)

6) The command DESCRIBE can be used to view the structure of a table

mysql> **DESCRIBE president;**

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

| Field | Type | Null | Key | Default | Extra | Privileges |

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

| last_name | varchar(15) | | | | | select,insert,update,references |

| first_name | varchar(15) | | | | | select,insert,update,references |

| state | char(2) | | | | | select,insert,update,references |

| city | varchar(20) | | | | | select,insert,update,references |

| birth | date | | | 0000-00-00 | | select,insert,update,references |

| death | date | YES | | NULL | | select,insert,update,references |

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

6 rows in set (0.00 sec)

7) To insert new rows into an existing table use the INSERT command:

mysql> **INSERT INTO president values ('Washington',**

'George',

'VA',

'Westmoreland County',

'17320212',

'17991214');

Query OK, 1 row affected (0.00 sec)

8) With the SELECT command we can retrieve previously inserted rows:

mysql> **SELECT * FROM president**;

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

| last_name | first_name | state | city | birth | death |

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

| Washington | George | VA | Westmoreland County | 1732-02-12 | 1799-12-14 |

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

1 row in set (0.00 sec)9) Selecting rows by using the WHERE clause in the SELECT command

mysql> **SELECT * FROM president WHERE state="VA";**

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

| last_name | first_name | state | city | birth | death |

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

| Washington | George | VA | Westmoreland County | 1732-02-12 | 1799-12-14 |

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

1 row in set (0.00 sec)

10) Selecting specific columns by listing their names

mysql> **SELECT state, first_name, last_name FROM president;**

```
+-------+------------+------------+
| state | first_name | last_name |
+-------+------------+------------+
| VA | George | Washington |
+-------+------------+------------+
```

1 row in set (0.00 sec)

11) Deleting selected rows from a table using the DELETE command

mysql> **DELETE FROM president WHERE first_name="George";**

Query OK, 1 row affected (0.00 sec)

12) To modify or update entries in the table use the UPDATE command

mysql> **UPDATE president SET state="CA" WHERE first_name="George";**

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

A general form of SELECT is:

SELECT ***what to select***

FROM ***table(s)***

WHERE ***condition that the data must satisfy*****;**

Comparison operators are: < ; <= ; = ; != or <> ; >= ; >

Logical operators are: AND ; OR ; NOT

Comparison operator for special value NULL: IS

14) The following MySQL query will return all the fields for the

presidents whose state field is "NY";

mysql> **SELECT * FROM president WHERE state="NY";**

```
+-----------+-------------+-------+---------------+------------+------------+
| last_name | first_name | state | city | birth | death |
+-----------+-------------+-------+---------------+------------+------------+
| Van Buren | Martin | NY | Kinderhook | 1782-12-05 | 1862-07-24 |
| Fillmore | Millard | NY | Cayuga County | 1800-01-07 | 1874-03-08 |
| Roosevelt | Theodore | NY | New York | 1858-10-27 | 1919-01-06 |
| Roosevelt | Franklin D. | NY | Hyde Park | 1882-01-30 | 1945-04-12 |
+-----------+-------------+-------+---------------+------------+------------+
```

4 rows in set (0.00 sec)

15) We can limit the values of the returned fields as it is shown bellow:

mysql> **SELECT last_name, first_name FROM president WHERE state="NY";**

```
+-----------+-------------+
| last_name | first_name |
+-----------+-------------+
```

| Van Buren | Martin |

| Fillmore | Millard |

| Roosevelt | Theodore |

| Roosevelt | Franklin D. |

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

4 rows in set (0.01 sec)

16) The following entry SELECT will return the last name and

birth date of presidents who are still alive

Note: The comparison operator will not work in this case:

mysql> **SELECT * FROM president WHERE death = NULL;**

Empty set (0.00 sec)

mysql> **SELECT last_name, birth FROM president WHERE death is NULL;**

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

| last_name | birth |

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

| Ford | 1913-07-14 |

| Carter | 1924-10-01 |

| Reagan | 1911-02-06 |

| Bush | 1924-06-12 |

| Clinton | 1946-08-19 |

| Bush | 1946-07-06 |

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

6 rows in set (0.00 sec)

17) This command will select the presidents who were born in the

18th century

mysql> **SELECT last_name, birth FROM president WHERE birth<"1800-01-01";**

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

| last_name | birth |

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

| Washington | 1732-02-12 |

| Adams | 1735-10-30 |

| Jefferson | 1735-04-13 |

| Madison | 1751-03-16 |

| Monroe | 1758-04-28 |

| Adams | 1767-07-11 |

| Jackson | 1767-03-15 |

| Van Buren | 1782-12-05 |

| Harrison | 1773-02-09 |

| Tyler | 1790-03-29 |

| Polk | 1795-11-02 |

| Taylor | 1784-11-24 |

| Buchanan | 1791-04-23 |

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

13 rows in set (0.00 sec)

18) The following command will select the president who was born first

mysql> **SELECT last_name, birth from president ORDER BY birth ASC LIMIT 1;**

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

| last_name | birth |

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

| Washington | 1732-02-12 |

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

1 row in set (0.00 sec)

19) The following query will return the names of fist 5 states (in

descending order) in which the greatest number of presidents have been

born

mysql> **SELECT state, count(*) AS times FROM president GROUP BY state**

-> ORDER BY times DESC LIMIT 5;

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

| state | times |

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

| VA | 8 |

| OH | 7 |

| MA | 4 |

| NY | 4 |

| NC | 2 |

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

5 rows in set (0.00 sec)

20) The following query will select presidents who have been born

in the last 60 years

mysql> **SELECT * FROM president WHERE(YEAR(now())- YEAR(birth)) < 60;**

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

| last_name | first_name | state | city | birth | death |

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

| Clinton | Bill | AR | Hope | 1946-08-19 | NULL |

| Bush | George W. | CT | New Haven | 1946-07-06 | NULL |

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

2 rows in set (0.00 sec)

Useful function to retrieve parts of dates are: **YEAR(), MONTH(), DAYOFMONTH(),**

TO_DAY().

21) The following query will sort presidents who have died by their

age and list the first 10 in descending order.

mysql> **SELECT last_name, birth, death, FLOOR((TO_DAYS(death) - TO_DAYS(birth))/365) AS age**

-> FROM president

-> WHERE death is not NULL ORDER BY age DESC LIMIT 10;

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

| last_name | birth | death | age |

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

| Jefferson | 1735-04-13 | 1826-07-04 | 91 |

| Adams | 1735-10-30 | 1826-07-04 | 90 |

| Hoover | 1874-08-10 | 1964-10-20 | 90 |

| Truman | 1884-05-08 | 1972-12-26 | 88 |

| Madison | 1751-03-16 | 1836-06-28 | 85 |

| Nixon | 1913-01-09 | 1994-04-22 | 81 |

| Adams | 1767-07-11 | 1848-02-23 | 80 |

| Van Buren | 1782-12-05 | 1862-07-24 | 79 |

| Jackson | 1767-03-15 | 1845-06-08 | 78 |

| Eisenhower | 1890-10-14 | 1969-03-28 | 78 |

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

SQL Join Types:

There are different types of joins available in SQL:

- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.

SELECT t.last_name, t.address, s.test_date, s.score

FROM test as t, student as s

WHERE t.ssn = s.ssn;

```
CREATE TABLE a (
id int NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
content varchar(20) NOT NULL,
done date NOT NULL
)
```

SELECT DISTINCT City FROM Customers;

SELECT TOP 3 * FROM CUSTOMERS;

SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS

GROUP BY NAME;

SELECT FROM table_name

WHERE column LIKE 'XXXX%'

or

SELECT FROM table_name

WHERE column LIKE '%XXXX%'

or

SELECT FROM table_name

WHERE column LIKE 'XXXX_'

or

SELECT FROM table_name

WHERE column LIKE '_XXXX'

or

SELECT FROM table_name

WHERE column LIKE '_XXXX_'

SQL > TRUNCATE TABLE CUSTOMERS;

Alter

The basic syntax of **ALTER TABLE** to add a new column in an existing table is as follows:

ALTER TABLE table_name ADD column_name datatype;

The basic syntax of ALTER TABLE to **DROP COLUMN** in an existing table is as follows:

ALTER TABLE table_name DROP COLUMN column_name;

The basic syntax of ALTER TABLE to change the **DATA TYPE** of a column in a table is as follows:

ALTER TABLE table_name MODIFY COLUMN column_name datatype;

The basic syntax of ALTER TABLE to add a **NOT NULL** constraint to a column in a table is as follows:

ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

Constraints

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table.

- NOT NULL Constraint: Ensures that a column cannot have NULL value.
- CREATE TABLE CUSTOMERS(
- ID INT NOT NULL,
- NAME VARCHAR (20) NOT NULL,
- AGE INT NOT NULL,
- ADDRESS CHAR (25) ,
- SALARY DECIMAL (18, 2),
- PRIMARY KEY (ID)
-);

Alter

ALTER TABLE CUSTOMERS

MODIFY SALARY DECIMAL (18, 2) NOT NULL;

- DEFAULT Constraint: Provides a default value for a column when none is specified.
- CREATE TABLE CUSTOMERS(
- ID INT NOT NULL,
- NAME VARCHAR (20) NOT NULL,
- AGE INT NOT NULL,
- ADDRESS CHAR (25) ,
- SALARY DECIMAL (18, 2) DEFAULT 5000.00,
- PRIMARY KEY (ID)
-);

Alter

ALTER TABLE CUSTOMERS

MODIFY SALARY DECIMAL (18, 2) DEFAULT 5000.00;

Drop Default Constraint:

ALTER TABLE CUSTOMERS

ALTER COLUMN SALARY DROP DEFAULT;

- UNIQUE Constraint: Ensures that all values in a column are different.
- CREATE TABLE CUSTOMERS(
- ID INT NOT NULL,
- NAME VARCHAR (20) NOT NULL,
- AGE INT NOT NULL UNIQUE,
- ADDRESS CHAR (25) ,
- SALARY DECIMAL (18, 2),
- PRIMARY KEY (ID)
-);

Alter

ALTER TABLE CUSTOMERS

MODIFY AGE INT NOT NULL UNIQUE;

ALTER TABLE CUSTOMERS

ADD CONSTRAINT myUniqueConstraint UNIQUE(AGE, SALARY);

Drop Constraint:

ALTER TABLE CUSTOMERS

DROP CONSTRAINT myUniqueConstraint;

- PRIMARY Key: Uniquely identified each rows/records in a database table.
- CREATE TABLE CUSTOMERS(
- ID INT NOT NULL,
- NAME VARCHAR (20) NOT NULL,
- AGE INT NOT NULL,
- ADDRESS CHAR (25) ,
- SALARY DECIMAL (18, 2),
- PRIMARY KEY (ID)
-);

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID, NAME)

);

Alter

ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);

ALTER TABLE CUSTOMERS

ADD CONSTRAINT PK_CUSTID PRIMARY KEY (ID, NAME);

Drop

ALTER TABLE CUSTOMERS DROP PRIMARY KEY ;

ALTER TABLE CUSTOMERS

DROP CONSTRAINT PK_CUSTID;

- FOREIGN Key: Uniquely identified a rows/records in any another database table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

CUSTOMERS table:

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

ORDERS table:

CREATE TABLE ORDERS (

ID INT NOT NULL,

DATE DATETIME,

CUSTOMER_ID INT references CUSTOMERS(ID),

AMOUNT double,

PRIMARY KEY (ID)

);

Alter

ALTER TABLE ORDERS

ADD FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (ID);

Drop

ALTER TABLE ORDERS

DROP FOREIGN KEY;

- CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- CREATE TABLE CUSTOMERS(
- ID INT NOT NULL,
- NAME VARCHAR (20) NOT NULL,
- AGE INT NOT NULL CHECK (AGE >= 18),
- ADDRESS CHAR (25) ,
- SALARY DECIMAL (18, 2),
- PRIMARY KEY (ID)
-);

Alter

ALTER TABLE CUSTOMERS

MODIFY AGE INT NOT NULL CHECK (AGE >= 18);

ALTER TABLE CUSTOMERS

ADD CONSTRAINT myCheckConstraint CHECK(AGE >= 18);

Drop

ALTER TABLE CUSTOMERS

DROP CONSTRAINT myCheckConstraint;

- INDEX: Use to create and retrieve data from the database very quickly.

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

CREATE INDEX index_name

ON table_name (column1, column2.....);

CREATE INDEX idx_age

ON CUSTOMERS (AGE);

Drop

ALTER TABLE CUSTOMERS

DROP INDEX idx_age;

Creating Views:

CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];

SQL > CREATE VIEW CUSTOMERS_VIEW AS

SELECT name, age

FROM CUSTOMERS;

SQL > SELECT * FROM CUSTOMERS_VIEW;

Dropping Views:

DROP VIEW view_name;

DROP VIEW CUSTOMERS_VIEW;

HAVING clause

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

SELECT column1, column2

FROM table1, table2

WHERE [conditions]

GROUP BY column1, column2

HAVING [conditions]

ORDER BY column1, column2

```
SQL > SELECT *

FROM CUSTOMERS

GROUP BY age

HAVING CO

.UNT(age) >= 2;

SQL> SELECT CONCAT(fname,' ',lname)as name from login;
```