|  |  |
| --- | --- |
|  | MySQL has played a significant role in the evolution of data storage and management. Its performance, scalability, and versatility have made it a popular choice for a wide range of applications. With its open-source nature and active community support, MySQL continues to evolve and thrive, addressing the evolving needs of modern data-driven systems.  *Prepared By*  **PARANOIA TECHNOLOGIES** |

MYSQL

* **What is Data & Database?**

Data refers to a collection of information, facts, or statistics that are represented in a structured or unstructured form. It can be in the form of numbers, text, images, audio, video, or any other type of digital representation. Data is typically used as input for various processes, analysis, and decision-making.

A database is a systematic collection of data in an electronic system that allows data to be stored, easily accessed, manipulated and updated.

* **Different Databases**
* Relational database
* Hierarchical database
* Network database
* Object-oriented database
* NoSQL database
* **What is Relational database?**

A relational database is a type of database management system (DBMS) that organizes data into tables consisting of rows(records) and columns(fields). It follows the relational model, which establishes relationships between tables using common attributes or keys.

* **Relational database management system**

It is a software system that enables the creation, management, and administration of relational databases. Most RDBMS use the SQL language to access the database.

* **SQL commands**
* DDL (Data Definition Language) - Create, Alter, Drop, Truncate, Rename
* DML (Data Manipulation Language) - Insert, Update, Delete
* DRL/DQL (Data Retrieval Language/Data Query Language) - Select
* TCL (Transaction Control Language) - Commit, Rollback, Save point
* DCL (Data Control Language) – Grant, Revoke
* **SQL**
* SQL stands for "Structured Query Language."
* It is a domain-specific programming language designed for managing and manipulating relational databases.
* SQL is used to interact with relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and many others.
* SQL provides a standardized way to perform tasks such as querying data, inserting, updating, and deleting records, creating and modifying database structures (tables, indexes, views, etc.), and managing user permissions.
* **Table**
* A table is a database object that serves as a container for storing related data.
* Tables are organized into rows and columns, creating a grid-like structure where each row represents a distinct record, and each column represents a specific attribute or field.
* Tables are defined with a name and a schema that specifies the structure of the table, including the names and data types of its columns.
* **Records (Rows)**
* In the context of a table, "records" refer to individual data entries or rows.
* Each row in a table represents a single, unique record or data point.

**For example:** if you have a table for storing information about customers, each row in the table might represent a specific customer, with each column containing details about that customer, such as their name, address, and contact information.

* **Fields**
* "Fields," also known as "columns" or "attributes," represent the individual pieces of data or characteristics that make up a record.
* Each column in a table is associated with a specific type of data, such as text, numbers, dates, or other data types.
* Fields define the structure and properties of the data stored in the table.
* **Data Integrity**
  + Data integrity refers to the accuracy, consistency, and reliability of data in a database or information system.
  + It is a fundamental aspect of data quality and ensures that data is both trustworthy and error-free.
  + Data integrity is crucial because it ensures that data is maintained in a state that accurately represents the real-world information it is intended to model or store.
* **Types**

|  |  |
| --- | --- |
| **Types** | **Description** |
| Entity Integrity | Ensures the uniqueness of primary keys in tables. |
| Domain Integrity | Enforces data type, format, and range constraints for column values. |
| Referential Integrity | Ensures the consistency of relationships between tables using foreign keys. |
| User Defined Integrity | Allows you to define custom rules and constraints for data. |

**Difference between DBMS and RDBMS**

| **Aspect** | **DBMS** | **RDBMS** |
| --- | --- | --- |
| Data Model | Various data models (e.g., hierarchical, network, flat file) | Relational data model (tables, rows, columns) |
| Data Integrity | May not enforce strong data integrity constraints | Enforces data integrity using constraints, foreign keys, and triggers |
| Query Language | Varies depending on the system and data model | Typically uses SQL (Structured Query Language) |
| Schema | Flexible schema, depending on the data model | Structured schema with predefined tables and columns |
| Data Flexibility | Can store data in various formats (text, binary, XML, etc.) | Stores data in structured tabular format |
| Performance | May not be as optimized for complex queries and joins | Optimized for complex queries, joins, and aggregations |
| Examples | MongoDB, CouchDB, Redis | MySQL, PostgreSQL, Oracle Database, SQL Server |

* **CREATE DATABASE**

1. CREATE DATABASE demo;
2. CREATE DATABASE IF NOT EXISTS demo;

* **DELETE DATABASE**

DROP DATABASE demo;

* **Data Types in MySQL**

1. VARCHAR (size) – variable length string (letter, number, special characters)
2. INT – Integer
3. FLOAT (size, d)
4. BOOLEAN – True (Non-zero) or False (Zero)
5. DATE – (yyyy-mm-dd)

* **CREATE TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| STUDENTS INFORMATION | | | |
| Student ID | Student Name | Age | Place |
| S1 | Akhil | 20 | Trivandrum |
| S2 | Manju | 18 | Ernakulam |
| S3 | Shareef | 13 | Palakkad |
| S4 | Praveen | 32 | Thrissur |
| S5 | John | 20 | Trivandrum |
| S6 | Akhil | 18 | Ernakulam |
| S7 | Aneesh | 14 | Kottayam |
| S8 | Lakshmi | 20 | Kozhikode |
| S9 | Cyril | 12 | Ernakulam |
| S10 | Santhosh | 28 | Malappuram |

1. USE demo;
2. CREATE TABLE student (student\_id VARCHAR (10) NOT NULL,

student\_name VARCHAR (100) NOT NULL, age INT NOT NULL,

place VARCHAR (100) NOT NULL, PRIMARY KEY (student\_id));

* **DELETE TABLE**

DROP TABLE student

* **SELECT TABLE**

SELECT \* FROM student;

* **INSERT DATA in a TABLE**
* INSERT INTO student (student\_id, student\_name, age, place)

VALUES (‘S1’, ‘Akhil’, 20, ‘Trivandrum’)

Or

* INSERT INTO student (student\_name, age, student\_id, place)

VALUES (‘Akhil’, 20, ‘S1’, ‘Trivandrum’);

* **UPDATE RECORDS**

UPDATE student SET place = ‘Kannur’ WHERE student\_id = ‘S1’;

* **COPY TABLE**

CREATE TABLE student1 LIKE student;

INSERT student1 select \* FROM student;

* **DELETE RECORDS**

DELETE FROM student1 where student\_id = ‘S2’;

* **SELECT STATEMENT**

SELECT student\_id, student\_name FROM student;

* **WHERE CLAUSE**

SELECT \* FROM student WHERE place = ‘Trivandrum’;

* **DISTINCT CLAUSE**

SELECT DISTINCT place FROM student;

* **ORDER BY**

1. **For ascending order**

SELECT \* FROM student ORDER BY student\_name ASC;

1. **For descending order**

SELECT \* FROM student ORDER BY student\_name DESC;

* **ADD OR MODIFY COLUMNS**

ALTER TABLE student1 ADD Contact INT (10) NOT NULL;

**NB:** Now we get a new column contact with datatype INT, now we want to change the datatype

ALTER TABLE student1 MODIFY contact VARCHAR (10) NOT NULL;

* **RENAME and DELETE COLUMNS**

ALTER TABLE student1 RENAME COLUMN contact TO student\_contact;

ALTER TABLE student1 drop column student\_contact;

* **TRUNCATE**

**NB:** To delete all records –

TRUNCATE student1

* **ACCESS FIRST and LAST RECORDS**

SELECT \* FROM student ORDER BY student\_id ASC LIMIT 1;

SELECT \* FROM student ORDER BY student\_id DESC LIMIT 1;

* **RANDOM RECORDS**

SELECT \* FROM student ORDER BY RAND () LIMIT 1;

* **SELECT AS ( Temporary altering field name)**

SELECT student\_name AS ‘first\_name’, age, place FROM student

* **ARITHMETIC OPERATORS**

SELECT 10 + 20 AS result;

SELECT 20 - 10 AS result;

SELECT 20 \* 10 AS result;

SELECT 10 / 20 AS result;

SELECT 20 % 3 AS result;

* **COMPARISON OPERATORS**

SELECT 10 = 10 AS result;

[ <, >, <=, >=, <> ]

NB: if True; ans 🡪 1

if False; ans 🡪 0

* USE demo;
* SELECT \* FROM student WHERE age = 20
* **LOGICAL OPERATORS (AND, OR, NOT)**
* SELECT \* FROM student WHERE age = 18 AND place = ‘Ernakulam’;
* SELECT \* FROM student WHERE age > 18 OR place = ‘Trivandrum’;
* SELECT \* FROM student WHERE age <18 NOT place = ‘Ernakulam’;
* SELECT \* FROM student WHERE age BETWEEN 20 AND 30;
* **EXISTS OPERATOR**

SELECT \* FROM student WHERE EXISTS(SELECT age FROM student WHERE age > 25);

* **IN OPERATOR**

SELECT \* FROM student WHERE place IN (‘Trivandrum’, ‘Ernakulam’)

* **LIKE OPERATOR**
* SELECT \* FROM student WHERE student\_name LIKE ‘Akhil’;
* SELECT \* FROM student WHERE student\_name LIKE ‘a%’;
* SELECT \* FROM student WHERE student\_name LIKE ‘%a’;
* SELECT \* FROM student WHERE student\_name LIKE ‘%a%’;
* **NOT LIKE OPERATOR**
* SELECT \* FROM student WHERE student\_name NOT LIKE ‘Akhil’;

**BUILT-IN FUNCTIONS**

* **STRING FUNCTIONS**

1. **char\_length ()**

SELECT place, char\_length(place) AS length FROM student;

1. **concat ()**

SELECT concat (student\_name, “ ‘‘, place) AS new\_string FROM student;

1. **format ()**

SELECT format (250500.5634,2) AS new\_number;

1. **insert ()**

SELECT insert (‘google’, 1, 3, ‘fff’);

1. **upper () & lower ()**

* SELECT upper (‘hello world’) AS new\_string;
* SELECT lower (‘HELLO WORLD’) AS new\_string;

1. **reverse ()**

SELECT reverse (‘HELLO’) AS new\_string;

1. **repeat ()**

SELECT repeat (‘hello’, 3) AS new\_string;

1. **left () & right ()**

* SELECT left (‘hello’, 4) AS new\_string;
* SELECT right (‘hello’, 4) AS new\_string;

1. **length ()**

SELECT length (‘hello’) AS new\_string;

* **MATHEMATICAL FUNCTIONS**

1. **abs ()**

SELECT abs (-10) AS new\_number;

1. **avg ()**

SELECT avg (age) as new\_number FROM student;

1. **ceiling () & floor ()**

* SELECT ceiling (25.3) AS new\_number;
* SELECT floor (25.3) AS new\_number;

1. **round ()**

SELECT round (25.5) AS new\_number;

1. **count ()**

SELECT count (student\_id) AS total\_student from student;

1. **max () & min ()**

* SELECT max (age) as max\_age FROM student;
* SELECT min (age) as min\_age FROM student

1. **sqrt ()**

SELECT sqrt (25) as new\_number;

1. **sum ()**

SELECT sum (age) as new\_number FROM student;

* **DATE FUNCTIONS**

1. SELECT current\_timestamp () AS time\_stamp;
2. SELECT dayname (‘2023/7/24’) AS today\_dayname;

* **TABLE WITH FOREIGN KEY**

|  |  |  |  |
| --- | --- | --- | --- |
| STUDENTS INFORMATION | | | |
| Student ID | Student Name | Age | Place |
| S1 | Akhil | 20 | Trivandrum |
| S2 | Manju | 18 | Ernakulam |
| S3 | Shareef | 13 | Palakkad |
| S4 | Praveen | 32 | Thrissur |
| S5 | John | 20 | Trivandrum |
| S6 | Akhil | 18 | Ernakulam |
| S7 | Aneesh | 14 | Kottayam |
| S8 | Lakshmi | 20 | Kozhikode |
| S9 | Cyril | 12 | Ernakulam |
| S10 | Santhosh | 28 | Malappuram |

|  |  |
| --- | --- |
| COURSES | |
| Course ID | Course Name |
| C1 | Computer Hardware |
| C2 | Networking |
| C3 | Web Designing |
| C4 | Graphic Designing |
| C5 | Ms Office |
| C6 | C++ |
| C7 | Python |

**USE** demo;

**CREATE TABLE** courses (course\_id VARCHAR (5) NOT NULL,

course\_name VARCHAR (50) NOT NULL,

PRIMARY KEY (course\_id));

**INSERT INTO** courses **VALUES**(‘C1’, ‘Computer Hardware’), (‘C2’, ‘Networking’),(‘C3’, ‘Web Designing’),(‘C4’, ‘Graphic Designing’),(‘C5’, ‘Ms Office’),(‘C6’, ‘C++’),(‘C7’, ‘Python’)

|  |  |  |
| --- | --- | --- |
| ENROLMENT | | |
| Enrolment ID | Student ID | Course ID |
| E1 | S1 | C1 |
| E2 | S7 | C4 |
| E3 | S6 | C2 |
| E4 | S5 | C1 |
| E5 | S4 | C5 |
| E6 | S1 | C4 |
| E7 | S2 | C6 |
| E8 | S3 | C7 |
| E9 | S2 | C4 |
| E10 | S4 | C2 |

**USE** demo;

**CREATE TABLE** enrolment (Enrolment\_id VARCHAR (100) NOT NULL,Student\_id VARCHAR (10),Course\_id VARCHAR (10),PRIMARY KEY (enrolment\_id), FOREIGN KEY (student\_id) REFERENCES student (student\_id), FOREIGN KEY (course\_id) REFRENCES courses (course\_id) );

**INSERT INTO** enrolment **VALUES**(‘E1’, ‘S1’, ‘C1’),(‘E2’, ‘S7’, ‘C4’),(‘E3’, ‘S6’, ‘C2’),(‘E4’, ‘S5’, ‘C1’),(‘E5’, ‘S4’, ‘C5’),(‘E6’, ‘S1’, ‘C4’),(‘E7’, ‘S2’, ‘C6’),(‘E8’, ‘S3’, ‘C7’),(‘E9’, ‘S2’, ‘C4’),(‘E10’, ‘S4’, ‘C3’),

**SQL JOINS**

* SQL joins are operations used to combine rows from two or more tables based on a related column between them.
* Joins are essential for retrieving data from multiple tables in a database and establishing relationships between them.
  1. **INNER JOIN**
  2. **LEFT (OUTER) JOIN**
  3. **RIGHT (OUTER) JOIN**
  4. **FULL (OUTER) JOIN**

**CREATE DATABASE** demo2;

**CREATE TABLE** country (country\_code VARCHAR (10) NOT NULL,country\_name VARCHAR (100) NOT NULL,PRIMARY KEY (country\_code) );

**INSERT INTO** country **VALUES**(‘IN’, ‘India’),(‘SL’, ‘Sri Lanka’),(‘PK’, ‘Pakistan’),(‘BN’, ‘Bangladesh),(‘NP’, ‘Nepal’);

**CREATE TABLE** capital (capital\_id VARCHAR (10) NOT NULL,country\_code VARCHAR (100),capital\_name VARCHAR (100),PRIMARY KEY (capital\_id),FOREIGN KEY (country\_code) REFERENCES country (country\_code));

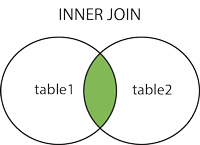
**INSERT INTO** capital **VALUES**(‘C1’, ‘IN’, ‘New Delhi’),(‘C2’, ‘PK’, ‘Islamabad’),(‘C3’, ‘NP’, ‘Kathmandu’);

|  |  |
| --- | --- |
| country\_code | country\_name |
| IN | India |
| SL | Sri Lanka |
| PK | Pakistan |
| BN | Bangladesh |
| NP | Nepal |

|  |  |  |
| --- | --- | --- |
| capital\_id | country\_code | capital\_name |
| C1 | IN | New Delhi |
| C2 | PK | Islamabad |
| C3 | NP | Kathmandu |

**INNER JOIN**

* Returns only the matching rows from both tables. The result set contains only the records where the join condition is satisfied.



**SELECT \* FROM country INNER JOIN capital ON country.country\_code = capital.country\_code;**

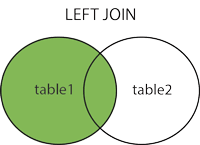
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| country\_code | country\_name | capital\_id | country\_code | capital\_name |
| IN | India | C1 | IN | New Delhi |
| PK | Pakistan | C2 | PK | Islamabad |
| NP | Nepal | C3 | NP | Kathmandu |

NB: To view only the required data

**SELECT country.country\_name, capital.capital\_name FROM country INNER JOIN capital ON country.country\_code = capital.country\_code**

**LEFT JOIN**

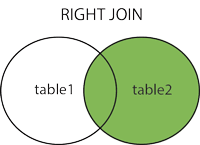
* Returns all the rows from the left table and the matching rows from the right table. If no match is found, NULL values are returned for the right table columns.



**SELECT \* FROM country LEFT JOIN capital ON country.country\_code = capital.country\_code**

**RIGHT JOIN**

* Returns all the rows from the right table and the matching rows from the left table. If no match is found, NULL values are returned for the left table columns.

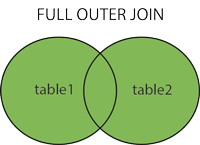


**SELECT \* FROM country RIGHT JOIN capital ON country.country\_code = capital.country\_code**

**FULL JOIN**

* Returns all the rows from both tables, regardless of whether they have a match or not which means combination of left & right joins.

**NB:** MYSQL does not supports full join, we can achieve it by doing union of left and right joins



**SELECT \* FROM country LEFT JOIN capital ON country.country\_code = capital.country\_code UNION SELECT \* FROM country RIGHT JOIN capital ON country.country\_code = capital.country\_code**