Table of Contents

[MySQL Notes 3](#_Toc172389833)

[Intro 3](#_Toc172389834)

[What is SQL? 3](#_Toc172389835)

[What can SQL do? 3](#_Toc172389836)

[What is My SQL? 4](#_Toc172389837)

[What is a Schema? 4](#_Toc172389838)

[Relationships in SQL Schema 5](#_Toc172389839)

[1. One-to-One Relationship 5](#_Toc172389840)

[2. One-to-Many (and Many-to-One) Relationship 5](#_Toc172389841)

[3. Many-to-Many Relationship 5](#_Toc172389842)

[Constraints 6](#_Toc172389843)

[SQL Primary Key Constraint 6](#_Toc172389844)

[SQL Foreign Key Constraint 6](#_Toc172389845)

[Star Schema 6](#_Toc172389846)

[Characteristics of Star Schema: 6](#_Toc172389847)

[Drawbacks of Star Schema: 7](#_Toc172389848)

[Snowflake Schema 7](#_Toc172389849)

[Characteristics of snowflake schema: 7](#_Toc172389850)

[Drawbacks of Snowflake Schema: 7](#_Toc172389851)

[Data manipulation language (DML) 8](#_Toc172389852)

[Data definition language (DDL) 8](#_Toc172389853)

[Data Control Language (DCL) 9](#_Toc172389854)

[Transaction Control Language (TCL) 9](#_Toc172389855)

[SQL Data types 9](#_Toc172389856)

[SQL Statements 10](#_Toc172389857)

[SELECT and FROM 10](#_Toc172389858)

[SELECT DISTINCT 10](#_Toc172389859)

[CREATE 10](#_Toc172389860)

[INSERT INTO 10](#_Toc172389861)

[DELETE 10](#_Toc172389862)

[ALTER TABLE 11](#_Toc172389863)

[LIMIT 11](#_Toc172389864)

[DROP 11](#_Toc172389865)

[ORDER BY 11](#_Toc172389866)

[FIELD 11](#_Toc172389867)

[WHERE 12](#_Toc172389868)

[AND 13](#_Toc172389869)

[OR 13](#_Toc172389870)

[BETWEEN 13](#_Toc172389871)

[LIKE 13](#_Toc172389872)

[IN 14](#_Toc172389873)

[IS NULL 14](#_Toc172389874)

[JOINS 14](#_Toc172389875)

[INNER JOIN 15](#_Toc172389876)

[LEFT JOIN 15](#_Toc172389877)

[RIGHT JOIN 15](#_Toc172389878)

[CROSS/FULL OUTER JOIN 15](#_Toc172389879)

[UNION 16](#_Toc172389880)

# MySQL Notes

## Intro

A database(DB) is an organised collection of data, typically stored in electronic format. It allows you to input data, organise the data and retrieve the data quickly. Traditional databases are organised by fields, records and files.

Most users do not access databases directly. Instead, users use a database management system(DBMS) to access the databases indirectly.

DBMS is a collection of programs that enable you to enter, organise and select data in a database.

A diagram of a database management system

Description automatically generated

## What is SQL?

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases

## What can SQL do?

|  |  |
| --- | --- |
| * Execute queries against a DB | * Create new databases |
| * Retrieve data from a DB | * Create new tables in a database |
| * Insert records in a DB | * Create stored procedures in a database |
| * Update records in a DB | * Create views in a database |
| * Delete records from a DB | * Set permissions on tables, procedures, and views |

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

 However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

**Note:** Most of the SQL database programs also have their own **proprietary extensions** in addition to the SQL standard!

## What is My SQL?

|  |  |
| --- | --- |
| * Relational database management system | * Cross platform |
| * Open-source | * Compliant with ANSI SQL standard |
| * Fast, reliable, scalable and easy to use | * Free |
| * Ideal for both small and large applications |  |
|  |  |

## What is a Schema?

A schema refers to the logical structure or blueprint that outlines the organisation of data in a database. It defines how data is organized and how relationships among data are structured. A database schema can include information about tables, fields, relationships, and constraints.

Key components of a database schema:

1. **Tables:** Describes the tables present in the database, including the names of tables, the fields they contain, and the data types of those fields.
2. **Fields (Columns):** Specifies the attributes or characteristics of the data stored in a table. Each field has a defined data type, such as text, number, date, etc.
3. **Relationships:** Defines how different tables in the database are related to each other. This includes specifying primary keys and foreign keys to establish connections between tables.
4. **Constraints:** Enforces rules and restrictions on the data to maintain data integrity. Examples include unique constraints, check constraints, and default values.
5. **Views:** Optional but often included, views provide a virtual representation of the data in one or more tables. They can simplify complex queries and provide a more user-friendly interface.

## Relationships in SQL Schema

Data model relationships are the connections that exist between different entities in a data model. They enable us to arrange related data together and make it easier to handle and analyse the information.

Cardinality in data modelling refers to the relationship between two entities in terms of their counts. It is crucial for understanding how individual entities relate to each other within a database structure. The principal types of cardinality are One-to-One, One-to-Many (or Many-to-One, from the opposite perspective), and Many-to-Many

### 1. One-to-One Relationship

A one-to-one relationship occurs when a single record in a table is related to only one record in another table, and vice versa.

• Example: Consider a database of a small company where each employee has a unique employee ID and a corresponding parking spot number. Here, one employee is related to one parking spot, and one parking spot is related to one employee.

• Advantages: This relationship is useful for dividing a table with many columns into smaller tables to organise data better and improve database performance.

### 2. One-to-Many (and Many-to-One) Relationship

A one-to-many relationship occurs when a single record in one table can be related to one or more records in another table. Conversely, many-to-one describes the same relationship from the opposite direction.

• Example: In a library database, one author can write multiple books, but each book has a single author. Here, one author relates to many books (one-to-many), and each book relates to one author (many-to-one).

• Advantages: This relationship type is common and reflects real-world data relationships well, allowing for efficient data organisation and retrieval.

### 3. Many-to-Many Relationship

A many-to-many relationship occurs when multiple records in one table are related to multiple records in another table.

• Example: In a school database, a student can enrol in many courses, and each course can have many students enrolled in it. This requires a third table (often called a junction or join table), such as Enrolments, which contains records linking students to courses.

• Advantages: This relationship type enables the representation of complex real-world relationships between entities, providing flexibility in data organisation and retrieval. It allows for the efficient association and disassociation of entities.

## Constraints

Constraints are limitations or rules placed on a field or column to ensure that data that is considered invalid is not entered.

### SQL Primary Key Constraint

* The **primary key** constraint uniquely identifies each record in the table
* Primary keys must contain **unique** values and cannot contain **null** values
* A table can have only **one** primary key; and in the table, this primary key can consist of single or multiple columns(fields)

### SQL Foreign Key Constraint

* The **foreign key** constraint is used to prevent actions that would destroy links between tables
* A **foreign key** is a field(or collection of fields) in one table, that refers to the **primary key** in another table
* The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

## Star Schema

### Characteristics of Star Schema:

* Simplified and fast queries. Fewer **JOIN** operations due to denormalization making information more readily available.
* Simple relationships. The schema works great with one-to-one or one-to-many relationships
* Singular dimensionality. One table describes each dimension.
* OLAP(Online Analytical Processing) systems widely use star schema to design data cubes

A diagram of a purchase

Description automatically generated

Fact table

Dimension table

### Drawbacks of Star Schema:

* **Redundancy**. The dimensional tables are one-dimensional, and data redundancy is present
* **Low integrity**. Due to denormalization, updating information is a complex task.
* **Limited queries**. The set of questions is limited, which also narrows down the analytical power.

## Snowflake Schema

### Characteristics of snowflake schema:

* **Small storage**. The snowflake schema does not require as much storage space.
* **High granularity**. Dividing tables into subdimensions allows analysis at various depths of interest. Adding new subdimensions is a simple process as well.
* **Integrity**. Due to normalisation, the schema has a higher level of data integrity and low redundancies

A diagram of a purchase

Description automatically generated

Fact table

Dimension table

### Drawbacks of Snowflake Schema:

* **Complexity**. The database model is complex, and so are the executed queries. Multiple multidimensional tables make the design complicated to work with overall.
* **Slow processing**. Many lookup tables require multiple JOIN operations, which slows down information retrieval
* **Hard to maintain**. A high level of granularity makes the schema hard to manage and maintain.

A diagram of a server

Description automatically generated

## Data manipulation language (DML)

A language that is used to insert data in, update, and query a DB. DMLs are often capable of performing mathematical and statistical calculations that facilitate generating reports.

DML commands –

INSERT: insert data into a table  
UPDATE: update existing data within a table  
DELETE: delete records from a database table  
LOCK: table control concurrency  
CALL: call a PL/SQL or JAVA subprogram  
EXPLAIN PLAN: describes the access path to data

## Data definition language (DDL)

A language that defines all attributes and properties of a DB, especially record layouts, field definitions, key fields, file locations, and storage strategy.

DDL commands –

CREATE: create the DB or objects(tables, index, function, views, store procedure and triggers)  
DROP: delete objects from the DB  
ALTER: alter structure of the DB  
TRUNCATE: remove all records from a table, including space allocated for them  
COMMENT: add comments to the data dictionary  
RENAME: rename an object in the DB

## Data Control Language (DCL)

Deals with rights, permissions, and other controls of the DB system.

DCL commands –

GRANT: gives users access privileges to the DB  
REVOKE: withdraws the user’s access privileges given by GRANT

## Transaction Control Language (TCL)

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fails. Therefore, a transaction has only two results: success or failure.

TCL commands -   
COMMIT: commits a transaction  
ROLLBACK: rollbacks a transaction in case of any error occurs  
SAVEPOINT: sets a save point within a transaction  
SET TRANSACTION: specifies characteristics for the transaction

## SQL Data types

Here are the most basic data types used in SQL.

|  |  |
| --- | --- |
| Data Type | Description |
| CHARACTER(n) | Character string. Fixed-length n |
| VARCHAR(n) | Character string. Variable length. Maximum length n |
| BINARY(n) | Binary string. Fixed-length n |
| BOOLEAN | Stores TRUE or FALSE values |
| INTEGER(p) | Integer numerical (no decimal). Precision p |
| INTEGER | Integer numerical (no decimal). Precision 10 |
| FLOAT | Approximate numerical |
| DATE | Stores year, month and day values |
| TIME | Stores hour, minute and second values |
| TIMESTAMP | Stores year, month, day, hour, minute and second values |
| INTERVAL | Composed of a number of integer fields, representing a period of time, depending on the type of interval. |

## SQL Statements

### SELECT and FROM

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

SELECT *column1*,*column2, ...*  
FROM *table\_name*;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT  \*  FROM *table\_name*;

### SELECT DISTINCT

The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT *column1*,*column2, ...*  
FROM *table\_name*;

### CREATE

This creates either a DB or a table within a DB.

CREATE DATABASE *database\_name;*

and

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
);

### INSERT INTO

This is used to add records in a table.

INSERT INTO *table\_name (column1, column2, column3, ...)*  
VALUES *(value1, value2, value3, ...);*

### DELETE

The DELETE statement is used to delete existing records in a table.

DELETE FROM *table\_name*WHERE *condition*;

### ALTER TABLE

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE *table\_name*  
ADD *column\_name datatype*;

### LIMIT

The LIMIT statement restricts the number of rows that are retrieved from the DB.

SELECT \* FROM column  
LIMIT number\_of\_rows;

### DROP

The DROP DATABASE or DROP TABLE statement(s) is used to drop an existing SQL DB or table.

DROP DATABASE *database\_name*;  
DROP TABLE *table\_name*;

### ORDER BY

The ORDER BY clause is used to sort the result set of a query by one or more columns. By default, it sorts the result set in ascending order. To sort the result set in descending order, you use the DESC keyword.

SELECT *column1*,*column2, ...*  
FROM *table\_name*  
ORDER BY *column1, column2, ...*ASC|DESC;

In MySQL, null values comes before non-null values. Therefore, when you the   
ORDER BY clause with the ASC option, nulls appear first in the result set.

Multi level sort - The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CustomerName" column

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

### FIELD

The FIELD() function returns the index (position) of a value within a list of values. If the value is not found in the list, the FIELD() function returns 0.

FIELD(value\_wanted, value1, value2, ...)

Example:   
There is a column in orders table called status. Suppose that you want to sort the sales orders based on their statuses in the following order:  
In Process On Hold Cancelled Resolved Disputed Shipped

To do this, you can use the FIELD() function to map each order status to a number and sort the result by the result of the FIELD() function:

SELECT orderNumber, status   
FROM orders   
ORDER BY   
 FIELD(  
 status,   
'In Process', 'On Hold', 'Cancelled', 'Resolved', 'Disputed', 'Shipped'  
 );

### WHERE

The WHERE clause is used to filter records. It is used to extract only those records that fulfil a specified condition.

SELECT *column1*,*column2, ...*  
FROM *table\_name*  
WHERE *condition*;

Example: Select all customers from Mexico. SQL requires single quotes around text values. However, numeric fields should not be enclosed in quotes

|  |  |
| --- | --- |
| SELECT \* FROM Customers WHERE Country='Mexico'; | SELECT \* FROM Customers WHERE CustomerID = 1; |

The specified condition is a combination of one or more expressions using the logical operator. The following operators can be used in the WHERE clause:

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Operator | Description |
| = | Equal | <> | Not equal |
| > | Greater than | BETWEEN | Between a range |
| < | Less than | LIKE | Search for a pattern |
| >= | Greater than or equal | IN | Specify multiple possible values for a column |
| <= | Less than or equal |

### AND

The AND operator is used to filter records based on more than one condition.

Example: Select all customers from Spain that starts with the letter 'G'

SELECT \*  
FROM Customers  
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';

### OR

The OR operator is used to filter records based on more than one condition

Example: Select all customers from Germany or Spain

SELECT \*  
FROM Customers  
WHERE Country = 'Germany' OR Country = 'Spain';

BETWEEN   
The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

Example: Selects all products with a price between 10 and 20

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

### LIKE

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. There are 5 wildcards often used in conjunction with the LIKE operator:

* percent sign % represents zero, one, or multiple characters
* underscore sign \_ represents one, single character
* Square Brackets [ ] specify a set or range of characters to match
* Caret ^ used within square brackets to indicate negation
* Escape \ is used when you need to include a literal percent sign (%) or underscore (\_) in the pattern

1. Example(%): Select all customers that starts with the letter "a":

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

1. Example(\_): Select all strings that starts with 'a' and is followed by exactly one character

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a\_';

1. Example([]): Select all strings that starts with a vowel

SELECT \* FROM Customers  
WHERE CustomerName LIKE '[aeiou]%';

1. Example(^): Select all strings that start with any character except a vowel

SELECT \* FROM Customers  
WHERE CustomerName LIKE '[^aeiou]%;

1. Example(\): Select all strings that contain '50% off'

SELECT \* FROM Customers  
WHERE percentage LIKE '50\% off’

### IN

The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Return all customers from 'Germany', 'France', or 'UK'

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

### IS NULL

To check if a value is [NULL](https://www.mysqltutorial.org/mysql-null/) or not, you use the [IS NULL](https://www.mysqltutorial.org/mysql-is-null/) operator, not the equal operator (=). The IS NULL operator returns TRUE if a value is NULL. In the database world, NULL is a marker that indicates that a value is missing or unknown. NULL is not equivalent to the number 0 or an empty string.

## JOINS

A JOIN clause is used to combine rows from two or more tables, based on a related column between them. The ON keyword is used to specify the condition that defines how the tables should be joined. It sets the rule for how rows from each table are matched.

MySQL supports the following types of joins:

|  |  |  |  |
| --- | --- | --- | --- |
| INNER | LEFT | RIGHT | CROSS |

### INNER JOIN

A diagram of two circles

Description automatically generatedReturns records that have matching values in both tables

SELECT *column\_name(s)*  
FROM *table1*  
INNER JOIN *table2*ON *table1.collum\_name*=*table2.column\_name*;

JOIN and INNER JOIN will return the same result. INNER is the default join type for JOIN, so when you write JOIN the parser actually writes INNER JOIN

### LEFT JOIN

A green and white circles with black text

Description automatically generatedReturns all records from the left table, and the matched records from the right table

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

### A diagram of a table Description automatically generatedRIGHT JOIN

Returns all records from the right table, and the matched records from the left table

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

### A green circles with black text Description automatically generatedCROSS/FULL OUTER JOIN

Returns all records when there is a match in either left or right table

The cross join makes a [Cartesian product](https://en.wikipedia.org/wiki/Cartesian_product) of rows from the joined tables. The cross join combines each row from the first table with every row from the right table to make the result set. Suppose the first table has n rows and the second table has m rows. The cross-join that joins the tables will return n x m rows.

SELECT *column\_name(s)*  
FROM *table1*  
CROSS JOIN *table2*ON *table1.column\_name*=*table2.column\_name*WHERE *condition*;

## UNION

The UNION operator is used to combine the result-set of two or more SELECT statements. This function basically stacks the columns on top of each other. Union only selects distinct values. Use UNION ALL to also select duplicate values

* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order