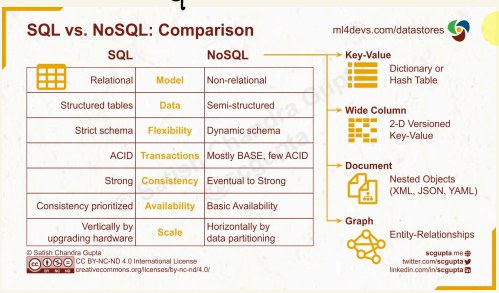
General Information on Databases / Storage Methods



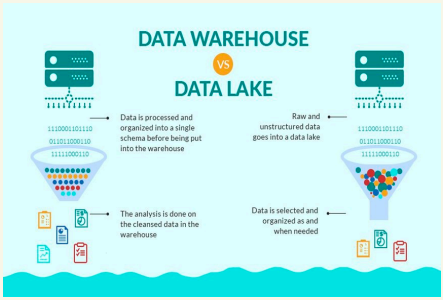
Data Storage:

① Database

Relational — SQL

NoSQL

② Data Warehouse vs Data Lakes

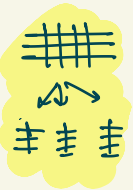


NO SQL

1]. Object Oriented Database

is stored in objects with attributes and methods Ex: db4O, ObjectDB

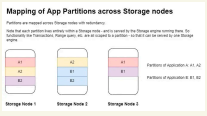
2). Columnar database:

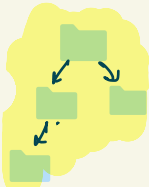


Data organised into column families / handling large data

Ex: Apache, Cassandra, HBase

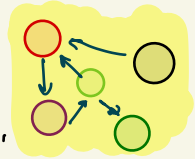
3] Document Database:





- Stored in flexible document such as (JSON, XML)
- Allows retrival and of data Ex: MongoDB, Couch DB, Elastic Search

4) Graph Databases:



Nodes + Edge

Ex: Neo4j, Allegro graph, Amazon Neptune

5] Time Series database

timestamp + data

Ex: Influx DB, Prometheus

6]. Spatial Database

Geographic Data

Ex: PostGIS, GeoTSON, GeoTiff

Note!:-

① ACID

A- Atomicity

C- Consistency

I - Isolation

D- Durability

② BASE

B, A → Basically Available

S → Soft state

E → Eventual consistency

Once why is clear How is half done

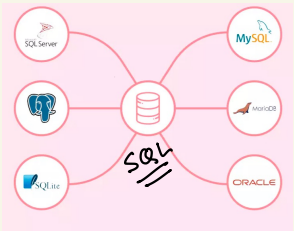
Why SQL?

Data Management

Data Retrival

Data Modification

Data Analysis



SQL Command List

- **SQL- commands**
  - **Data Definition Language (DDL)**
    - Create
    - Alter
    - Drop
    - Truncate
      - delete data from table.
    - Rename
  - **Data Manipulation Language (DML)**
    - Select
    - Insert
    - Update
    - Delete
  - **Data Control Language (DCL)**
    - Grant
    - Revoke
  - **Transaction Control Language (TCL)**
    - Commit
    - Rollback
    - Savepoint

https://www.geeksforgeeks.org/sql-transactions/

- **Constraints**
  - Primary key
  - Foreign key
  - Check
  - Unique
  - Default
  - Not Null

etc

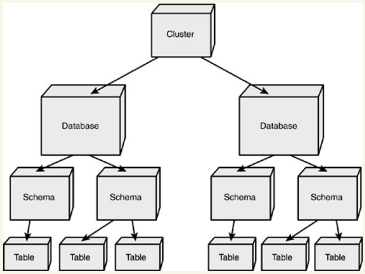
Definitions

Database: collection of structured tables/data

Schema: logical and visual configuration of RDBMS

Database State: What database contains at particular state

Schema
- conceptual database schema
- logical database schema
- physical database schema

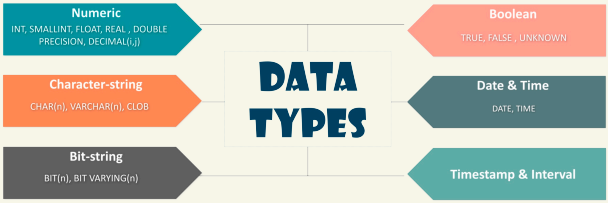


Note:- In MySQL, database ↔ schema ⇒ Don't confuse here!

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Copyright © mysqltutorial.org. All rights reserved.



Structured Query Language (SQL)

— Ayush Manojkumar Lodha

ALIAS

[]. AS : Temporary column name or table name

Syntax: SELECT col-name as alias-name FROM table_name;

SELECT col_name (S) FROM table-name AS alias_name

SELECT

selecting columns/data from database

Syntax: SELECT col1, col2, ... FROM table-name;

Select * FROM table-name; → select all columns

Select DISTINCT col-name → only distinct FROM table-name values

WHERE

Used to filter rows

Syntax: SELECT col1, col2, ... FROM table-name WHERE condition

Operator

=	Equal	>=	Greater Than Equal to
>	Greater Than	<=	Less than Equal to
<	Less Than	LIKE	→ Search for pattern
<> or !=	Not Equal to		
BETWEEN	Between certain Range		
IN	→ To specific multiple possible values for column		

ORDER BY

Sorting according to the cols used in ascending or descending

Syntax: SELECT column1, column2, FROM table-name ORDER BY col1 ASC, col2 DESC ...

ascending, descending → Ordering direction

GROUP BY

* column having same values finding number of cities in State

Commonly Used with agg. functions:

COUNT(), MAX() ,MIN(), SUM(), AVG() to group the result set by columns

Syntax: SELECT col-name(s) FROM table_name WHERE cond^n GROUP BY col-name(s)

HAVING: → since where cannot be used during agg. f^n's

Syntax: SELECT col-name(s) FROM table_name GROUP BY col-name HAVING condition

Ex: SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) >5

SQL CASE WHEN

conditional value statements.

Syntax:

```
CASE
    WHEN cond1 THEN result1
    WHEN cond2 THEN result2
    WHEN cond3 THEN result3
END;
```

AND, ALL

AND: means that any conditions would be true

ALL: means that the condition will be true only if operation is true for all values

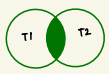
JOINS

INNER JOIN

Having matching value in both tables

SYNTAX:

```
SELECT col_name(s)
FROM table1
INNER JOIN table2
ON table1.col-name
= table2.col_name
```

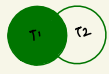


LEFT JOIN

All records from left table and matching from right table

SYNTAX:

```
SELECT col_name(s)
FROM table1
LEFT JOIN table2
ON table1.col-name
= table2.col_name
```

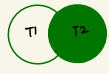


RIGHT JOIN

All records from right table and matching from left table

SYNTAX:

```
SELECT col_name(s)
FROM table1
RIGHT JOIN table2
ON table1.col-name
= table2.col_name
```



FULL OUTER JOIN

Return all records in left or right when there is match

SYNTAX:

```
SELECT col_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.col-name
= table2.col_name
```

SQL Commands

LIKE → Specified pattern in a column

% → represents zero, one or multiple

_ → One, Single character

Syntax: SELECT col1, col2, ... FROM table-name WHERE column LIKE pattern

Ex1: SELECT * FROM Customers WHERE City LIKE 'L_nd_';

Ex2: Contains the Letter 'L'

SELECT * FROM Customers WHERE City LIKE '%L%'

Ex3: Starts with specific letter

SELECT * FROM Customers WHERE CustomerName LIKE 'La%'

IS NULL

IS NULL operator is used to test for empty values

Syntax: SELECT col-name FROM table-name WHERE col-name IS NULL;

IS NOT NULL operator is used to test for non empty.

Syntax: SELECT col-names FROM table_name WHERE col-name IS NOT NULL;

WINDOW FUNCTIONS

Syntax: select col-name 1, window function (col-name 2) OVER([partition BY col-name3] [order by col-name3]), FROM table-name;

window function is applied on column 2

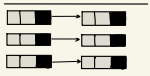
partition is done according to column3 and also ordering

Remember use case: highest Salary department Wise

AS new_column

Group By Logic

Window Functions Logic



Aggregate Functions

1). sum()
2). count()
3). average()
4). Max()/Min()
5) NTile (* of buckets *)
NTile (4) → Quartile
NTile (5) → Quintile
NTile (100) → Percentile

Ranking Functions:

1). row_number(): unique number for each row within partition with different numbers for tied values

2). rank(): ranking with partition, with gaps and same ranking for tied values

3). dense_rank(): ranking with partition, with no gaps() and same ranking for tied values

UNION

Union operator is used to combine the result-set of two or more SELECT statements

Syntax: SELECT col(s) FROM table_1 UNION SELECT col(s) FROM table_2 } only distinct values

UNION ALL: allow duplicate values

Syntax: SELECT col(s) FROM table_1 UNION ALL SELECT col(s) FROM table_2

Value Functions

- Lag() → returns value of previous row before current row in a partition. Null if previous does not exists.
- Lead() → returns value for the row after the current row in a partition. If no row exists, null is returned.
- first_value → returns the value of specified expression with respect to the first row in the window frame.
- last_value → returns the last value

COMMON TABLE EXPRESSION (CTE)

Syntax: WITH my_cte AS (Select a,b,c FROM T1) SELECT a,c FROM my_cte WHERE ...

Note: Just add a comma as you can add multiple such tables here

RECURSION IN SQL

Syntax:

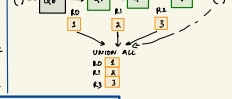
```
WITH R as ( <base query>
    UNION ALL
    <recursive-query involving R>)
<Query involving R>
```

base member (anchor member)

recursive member: ref. R

Ex:

```
WITH cnt_table AS
( SELECT 1 as n
UNION ALL
SELECT n+1 FROM cnt_table
WHERE n < 3)
SELECT * FROM cnt_table
```



STRING SPECIFIC FUNCTIONS

* SUBSTR → Extracts substring from string

Syntax: SUBSTR(string, start, length)

* LENGTH → length of string

Syntax: LENGTH (col-name)

* LEFT → beginning portion of string

Syntax: LEFT(string, no_of_chars)

* RIGHT → ending portion of string

Syntax: RIGHT (string, no_of_chars)

* REPLACE → replaces all occurences of substring with string

Syntax: REPLACE (string, sub-str, sub-str2)

* UPPER → upper case of string

* LOWER → lowercase of string