SQL and Database Notes

Database Overview

- Database is a collection of data that can be easily accessed and managed.
- A software application is used to manage databases.

Types of Databases:

- Relational (stored in tables) - Examples: MySQL, Oracle, PostgreSQL
- Non-relational (Not stored in tables) - Examples: MongoDB

SQL - Structured Query Language

- Used to interact with relational databases.
- CRUD Operations: Create, Read, Update, Delete
- Originally called SEQUEL by IBM.

Row – individual data
Column – Represent Schema

Creating a Database and Tables

- CREATE DATABASE db_name; // To create Database
- DROP DATABASE db_name; // To Delete Database
- USE db_name; // to define all the further operation would be performed on the above database
- CREATE TABLE table_name (column_name datatype constraint, ...); // to create table

SQL Data Types

- CHAR // it stores String Value (0-255)
- VARCHAR(size) // it store string but only store and that much space of the data provided (0-255)
- BLOB // Stores large binary object (0-65535)
- INT // Stores Integer Value
- TINYINT // Stores Integer (-128 to 127)
- BIGINT // Stores Integer (-9223.372 to 9223.372)
- FLOAT // Stores Decimal Values
- DOUBLE // Stores large Decimal
- BOOLEAN // 0 or 1
- DATE // (YYYY-MM-DD)

- YEAR // (4-digit format)(1902-2155)
- SIGNED and UNSIGNED // used to increase the range if only positive/negative data are present

UNSIGNED - positive data
SIGNED – Negative Data

SQL Command Types

- DDL - Data Definition Language (CREATE, ALTER, DROP, etc.)
- DQL - Data Query Language (SELECT)
- DML - Data Manipulation Language (INSERT, UPDATE, DELETE)
- DCL - Data Control Language (GRANT, REVOKE)
- TCL - Transaction Control Language (COMMIT, ROLLBACK, etc.)

Database Queries

- CREATE DATABASE IF NOT EXISTS db_name; // to create Database if doesn't exist
- DROP DATABASE IF EXISTS db_name; // to delete database if exist
- SHOW DATABASES; // to show Databases
- SHOW TABLES; // to Show tables

Table Queries

- DROP TABLE table_name; // Delete Table
- SELECT * FROM table_name; // Select all from table
- INSERT INTO table_name (columns) VALUES (...); // To insert data in table

Constraints

To Specify Rule for Data in table

- NOT NULL
- UNIQUE
- PRIMARY KEY (NOT NULL + UNIQUE)
- FOREIGN KEY // prevent action that would destroy link between table
- DEFAULT // store Default Value if null
- CHECK // it can limit the value allowded

SELECT Statements

It is used to select any data in database

- SELECT column1, column2 FROM table_name;

- SELECT DISTINCT column FROM table_name; // Unique values

WHERE clause in Select

Select * from student where marks>80;

- Operators:
  1. Arithmetic (+,-,*,/,%)
  2. Comparison (=, !=, <, >),
  3. Logical (AND, OR, NOT,IN,BETWEEN, LIKE, ANY)
  4. Bitwise &(and) or | (or)
- And // both the conditions should met
- OR // one of the conditions should be true
- BETWEEN // Used for Range
- NOT // To Negate (to reverse conditions)
- LIMIT // To set a limit on the output

Aggregate Functions

Perform a calculation on a Set of Value and return Single Value

- COUNT()
- MAX()
- MIN()
- AVG()
- SUM()

SELECT COUNT(Rollno) FROM Student;

ORDER BY Clause

To Sort in ascending or descending Order

SELECT * FROM Student
ORDER BY City ASc;

GROUP BY Clause

- Group row that have same value

SELECT City ,COUNT(name) FROM Student;
GROUP BY City;

- We use group by some aggregate Functions

SELECT City , AVG(Marks)
FROM Student
GROUP BY City
ORDER BY Avg (Marks) DESC;

HAVING clauses

- Similar to WHERE
- Used when we want to apply any condition after grouping

SELECT COUNT(Name) , City
FROM Student
GROUP BY City
HAVING MAX(Marks) >90;

General Order

1. SELECT COLUMN
2. FROM Table_name
3. WHERE Condition
4. GROUP BY Column
5. ORDER BY Column ASC;

- HAVING applies Condition on Column
- WHERE Applies Condition on Rows

Update

- to update existing rows

UPDATE Student

SET grades = “O”

WHERE grades = “A”;

SET SQL_SAFE_UPDATES = 0; //To end safe mode

Delete

- to delete existing Table
  DELETE FROM STUDENT
  WHERE Marks<33;

FOREIGN key

- Cascading – changes happening in one table should be done to every data associated with it

On delete Cascade and on update

Create Table student

Id int Primary key

Course ID int

Foreign key (Course ID) references course (id)

ON DELETE CASCADE

ON UPDATE CASCADE;

Alter

- to change the schema

1. Add column

ALTER TABLE table_name

ADD COLUMN age int

2. DROP
3. RENAME
4. CHANGE (data type) // Column old_name new name new data type new constraints
5. MODIFY)/ New datatype New Constraints;

Truncate

- To delete table data

Truncate table student;

Joins in SQL

- combine row from two or more table based on related column between them
- It is not Compulsory to have Foreign Key

#Types of JOIN

- INNER JOIN
- LEFT JOIN --------
- RIGHT JOIN |----- OUTER JOIN
- FULL JOIN --------

INNER JOIN

return records that have Matching value in both table

SELECT Columns

From Table A

INNER JOIN Table B

On table a col_name = table b .col_name

alias - as a // made up name

LEFT JOIN -

LEFT Data(A) + overlapping Data of B

Syntax Same as Inner Join

RIGHT JOIN

VICE Versa of LEFT JOIN

FULL JOIN

- Return all record of both right or left

SELECT * FROM student as a

LEFT JOIN Course as b

ON a. id = b.id

UNION

SELECT * FROM STUDENT as a

RIGHT JOIN course as b

ON a.id = b.id

- UNION -unique DATA

RIGHT EXCLUSIVE JOIN

- Only Right Element of A that don't overal with elements of B

LEFT EXCLUSIVE JOIN

- Only left Element of B that don't overal with elements of A

SELECT*

FROM studies as a

LEFT JOIN course as b

ON a.id= b.id

WHERE b id is null

Full Exclusive JOIN

SELF JOIN - it join itself – To find hierarchy/ to find relevant data

SELECT COLUMN

FROM table as a

JOIN table as b

ON a.col_name = b.col_name;

UNION

- Combine two set giving unique set
- select should have same number of columns
- column must have similar data type
- Column should be in Some order

SELECT COLUMN FROM TABLE A

UNION

SELECT COLUMN From TABLE B

UNION ALL - allow duplicates:

Views

- Virtual tables created from SQL queries.
- CREATE VIEW view_name AS SELECT ...;

CREATE VIEW VIEW1
SELECT Rollno, name FROM Student;
SELECT * FROM View1;

- Views show up to date data

Subqueries

- Nested queries used inside another query.
- SELECT column FROM table WHERE column IN (SELECT ...);

SELECT COLUMN
FROM Table_name
WHERE Col_name OPERATOR

(Subquery);