Intro

30 June 2024 11:59

Types of Data bases :-

① Relational (RDBMS)

Stores data in tables.

Eg - MySQL, Oracle, PostgreSQL

② Non-Relational (NoSQL)

Data Not Stored in tables

Eg - MongoDB

→ SQL

use to interact with RDBMS.

It is used to perform CRUD operation:

- Create
- Read
- Update
- Delete

One database can have multiple tables in it.

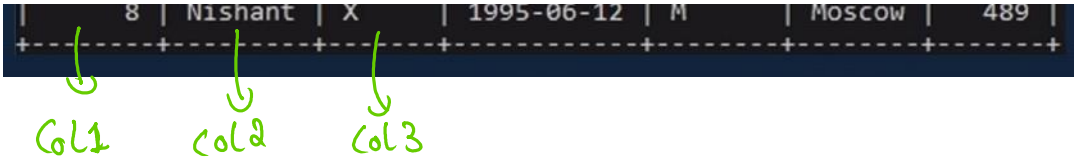
Example of table :-

Student table

RollNo	Name	Class	DOB	Gender	City	Marks
1	Nanda	X	1995-06-06	M	Agra	551
2	Saurabh	XII	1993-05-07	M	Mumbai	462
3	Sonal	XI	1994-05-06	F	Delhi	400
4	Trisla	XII	1995-08-08	F	Mumbai	450
5	Store	XII	1995-10-08	M	Delhi	369
6	Marisla	XI	1994-12-12	F	Dubai	250
7	Neha	X	1995-12-08	F	Moscow	377
8	Nishant	X	1995-06-12	M	Moscow	489

row 1

row 2



Column → design / Schema / Structure

row → individual data.

* Types of SQL Command -

DDL (Data Defination language): Create, alter, rename, truncate & drop

DQL (Data Query language): Select

DML (Data Manipulation Language): insert, update, delete

DCL (Data Control Language): grant & revoke permission to user

TCL (Transaction Control language): Commit, rollback, Start transaction

* → Keys :-

Primary Key -

- a unique id
- Only 1 PK exists for a particular table.
- PK can't be NULL.

Foreign Key -

- used to refer to the PK of another table
- Can be multiple FK.
-

→ Constraints for Data -

① Not NULL

② Unique

③ Primary Key — Combination of unique and Not NULL

↳ Can be Combination of two Col. individual Col can have duplicate vales But the combination of cols must be unique.

```
CREATE TABLE temp1 (
 id INT,
 name VARCHAR(50),
 age INT,
 city VARCHAR(20),
 PRIMARY KEY (id, name)
);
```

④ Foreign Key

```
▷ Run | New Tab
CREATE TABLE temp (
cust_id int,
FOREIGN KEY (cust_id) references customer(id)
);
```

temp table Col name

reference table_name

PK name of reference table

⑤ Default used to set Default value for a Col.

⑥ Check limit the values allows into the Column.

```
▷ Run | New Tab | Copy
CREATE TABLE city (
id INT PRIMARY KEY,
city VARCHAR(50),
age INT,
CONSTRAINT age_check CHECK (age >= 18 AND city="Delhi")
);
```

```
▷ Run | New Tab
CREATE TABLE newTab (
age INT CHECK (age >= 18),
);
```

(7) Aggregate function

Count()

Max()

Min()

Sum()

Avg()

(8) where — applied on rows

(9) Having — groups

Note :- General Order of SQL - Select

FROM

WHERE

GROUP BY

HAVING

ORDER BY

→ Cascading foreign Key's-

On Delete Cascade

FK created using this option, it deletes the referencing row in the child table when the referenced row is deleted in the parent table which has Primary Key.

on update Cascade

FK created using this, the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a Primary Key.

```
CREATE TABLE student (
  id INT PRIMARY KEY,
  courseID INT,
  FOREIGN KEY(courseID) REFERENCES course(id)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

→ Table Related Queries:-

Alter (change Schema)

Add

```
ALTER TABLE table_name
ADD COLUMN column_name datatype constraint;
```

Drop

```
ALTER TABLE table_name
ADD COLUMN column_name datatype constraint;
```

Rename

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Change

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name new_datatype new_constraint;
```

Modify

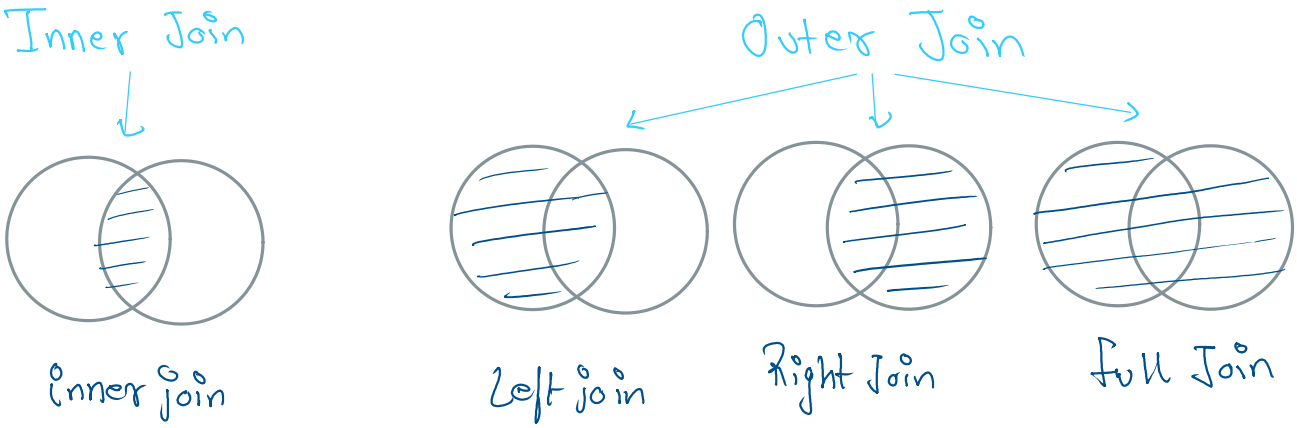
```
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;
```

Truncate

```
TRUNCATE TABLE table_name ;
```

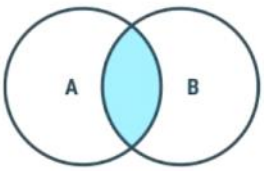
- truncate delets the data of table
- Drop delets the table
- Delete is used to delete specific data

→ Join



Inner Join

Returns records that have matching values in both tables



Syntax

```
SELECT column(s)
FROM tableA
INNER JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Example Select * from Student Inner Join Course on Student.id = Course.id

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

Example

select * from Student
Inner Join Course on Student.id = Course.id

student

student_id	name
101	adam
102	bob
103	casey

course

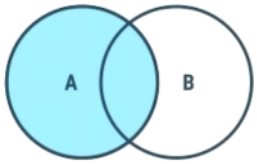
student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	name	course
102	bob	english
103	casey	science

Left Join

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



Syntax

```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Example

Select * From Student left Join Course
on Student.Student_id = Corse.Course_id;

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

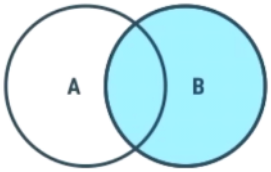
Result

student_id	name	course
101	adam	*null*
102	bob	english

102	bob	english
103	casey	science

Right Join

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



Syntax

```
SELECT column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Right Join

Example

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	course	name
102	english	bob
105	math	*null*
103	science	casey
107	computer science	*null*

Full Join

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

Syntax in MySQL

```
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;
```

LEFT JOIN
UNION
RIGHT JOIN

Full Join

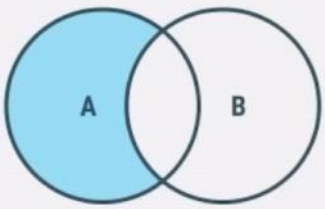
Returns all records when there is a match in either left or right table

Syntax in MySQL

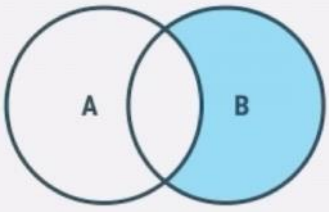
```
SELECT * FROM student as a
LEFT JOIN course as b
ON a.id = b.id
UNION
SELECT * FROM student as a
RIGHT JOIN course as b
ON a.id = b.id;
```

LEFT JOIN
UNION
RIGHT JOIN

Notes —



Left Exclusive Join



Right Exclusive Join

```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

* Self Join

regular join but table is joined with itself.

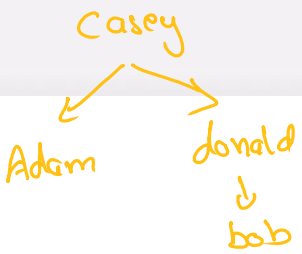
```
Select  col
 from   Table as a
 Join   Table as b
  On    a.col_name = b.col_name;
```

Example

Employee

id	name	manager_id
101	adam	103
102	bob	104
103	casey	*null*
104	donald	103

```
SELECT a.name as manager_name, b.name
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```



→ Union

Combine result of two or more SELECT Statments.
Gives Unique Records.

Restrictions -

- SELECT Statments should have same no. of Columns.
- Col must have similar data type.
- Col in every Select should be of same order.

Note:- UNION ALL give all the duplicate values.

→ Sub Queries :-

① Select Col
FROM table
where Col_name operator (subqueries) ;

② Select Col
FROM (subqueries) As temp ;

③ Select (subqueries)
FROM table ;

{ Subqueries in Select Should return a Single type of row or else it will give error }

→ Views :-

Virtual tables based on result-set of an SQL statments.

```
CREATE VIEW view1 AS
SELECT name, roll_no FROM student;

SELECT * FROM view1 ;
```

Query

30 June 2024

```
▷ Run | New Tab | 🔒 Active Connection
CREATE DATABASE temp1;
▷ Run | New Tab
DROP DATABASE temp1;
```

used to create Database

used to delete Database

```
▷ Run | New Tab
USE college;
```

→ target Database, to use

```
▷ Run | New Tab | Copy
CREATE Table student(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT NOT NULL
);
```

→ used to create table name student into college database

'id' is Primary Key for the table

'name' is a str/char type data, which has max length of 50 & Can be Null

'age' is a int type variable, can't be NULL.

```
▷ Run | New Tab
INSERT INTO student VALUES(1,"Priyanshu", 23);
▷ Run | New Tab
INSERT INTO student VALUES(2,"Priyanshu", 23);
```

} used to add entry into the table

```
▷ Run | New Tab | JSON
SELECT * FROM student;
```

} display the whole Schema for Student.

```
▷ Run | New Tab
DROP TABLE student;
```

used to drop student table.

```
▷ Run | New Tab
INSERT INTO student (rollno, name) VALUES (2,"Aman"), (3,"naman");
```

used to insert multiple rows.

```
▷ Run | New Tab
CREATE DATABASE IF NOT EXISTS college;
```

will only create if table doesn't exist else gives warning.