Batch: B2	Roll No.: 110 and 109
Experiment / assignment / tutorial No. 5	

Title: Queries based on Joins, and Views

Objective: To be able to use SQL JOIN clause to extract data from 2 (or more) tables, we need a relationship between certain columns in these tables.

Expected Outcome of Experiment:
CO 3 : Use SQL for Relational database creation, maintenance and query processing
CO 4 : Applying normalization to design database

Books/ Journals/ Websites referred:

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : "Database Systems Concept", 5th Edition , McGraw Hill
4. Elmasri and Navathe,"Fundamentals of database Systems", 4th Edition,PEARSON Education.

Resources used: Postgresql

Theory

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied. Or JOINS are used to retrieve data from multiple tables. A JOIN is performed whenever two or more tables are joined in a SQL statement.
There are different types of Joins:

- The CROSS JOIN
- The INNER JOIN
- The LEFT OUTER JOIN

- The RIGHT OUTER JOIN
- The FULL OUTER JOIN

A **CROSS JOIN** matches every row of the first table with every row of the second table. If the input tables have x and y columns, respectively, the resulting table will have x+y columns. Because CROSS JOINs have the potential to generate extremely large tables, care must be taken to use them only when appropriate.
Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY CROSS JOIN DEPARTMENT;

A **INNER JOIN** creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows, which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of table1 and table2 are combined into a result row.

Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The **OUTER JOIN** is an extension of the INNER JOIN. SQL standard defines three types of OUTER JOINs: LEFT, RIGHT, and FULL and PostgreSQL supports all of these.

In case of **LEFT OUTER JOIN**, an inner join is performed first. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. Thus, the joined table always has at least one row for each row in T1.

Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The RIGHT OUTER JOIN
First, an inner join is performed. Then, for each row in table T2 that does not satisfy the join condition with any row in table T1, a joined row is added with null values in columns of T1. This is the converse of a left join; the result table will always have a row for each row in T2.
Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY RIGHT OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The **FULL OUTER JOIN**

First, an inner join is performed. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. In addition, for each row of T2 that does not satisfy the join condition with any row in T1, a joined row with null values in the columns of T1 is added.
SELECT EMP_ID, NAME, DEPT FROM COMPANY FULL OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

Views are pseudo-tables. That is, they are not real tables; nevertheless appear as ordinary tables to SELECT. A view can represent a subset of a real table, selecting certain columns or certain rows from an ordinary table. A view can even represent joined tables. Because views are assigned separate permissions, you can use them to restrict table access so that the users see only specific rows or columns of a table.

A view can contain all rows of a table or selected rows from one or more tables. A view can be created from one or many tables, which depends on the written PostgreSQL query to create a view.
Views, which are kind of virtual tables, allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can only see limited data instead of complete table.
- Summarize data from various tables, which can be used to generate reports.

Since views are not ordinary tables, you may not be able to execute a DELETE, INSERT, or UPDATE statement on a view. However, you can create a RULE to correct this problem of using DELETE, INSERT or UPDATE on a view.

Syntax

CREATE [TEMP | TEMPORARY] VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];

Ex

CREATE VIEW COMPANY_VIEW AS

SELECT ID, NAME, AGE

FROM COMPANY;

Dropping Views

Syntax: DROP VIEW view_name;

Implementation Screenshots (Problem Statement, Query and Screenshots of Results):

```
/*inner join using where */
select typename,movename
from pokemontype, pokemonmove
where typeid=mtype


/*inner join using join */
select typename,movename
from pokemontype join pokemonmove
on typeid=mtype

/*cartesian product*/
select typename,movename
from pokemontype natural join pokemonmove;

/*cartesian product using where*/
select typename,movename
from pokemontype , pokemonmove;

/*making a veiw from the cartesian product*/
create view cartesian as
select typename,movename
from pokemontype , pokemonmove;

select * from cartesian
where typename='water'

/*left join*/
insert into pokemontype(typeId,typename) values (6,'normal');

select *
from pokemontype left join pokemonmove
on mtype=typeid;
```

Data Output | Explain | Messages | Notifications

	typename character varying (12)	movename character varying (12)
1	fire	flamethrower
2	water	water gun
3	fire	ember
4	water	whirlpool
5	grass	razor leaf
6	fire	flame blitz
7	psychic	dream eater
8	rock	stone edge
9	grass	giga drain

Data Output | Explain | Messages | Notifications

	typename character varying (12)	movename character varying (12)
1	fire	flamethrower
2	fire	water gun
3	fire	ember
4	fire	whirlpool
5	fire	razor leaf
6	fire	flame blitz
7	fire	dream eater
8	fire	stone edge
9	fire	giga drain
10	water	flamethrower
11	water	water gun
12	water	ember

Data Output | Explain | Messages | Notifications

	typename character varying (12)	movename character varying (12)
34	psychic	dream eater
35	psychic	stone edge
36	psychic	giga drain
37	rock	flamethrower
38	rock	water gun
39	rock	ember
40	rock	whirlpool
41	rock	razor leaf
42	rock	flame blitz
43	rock	dream eater
44	rock	stone edge
45	rock	giga drain

Data Output | Explain | Messages | Notifications

	typeid integer	typename character varying (12)	movename character varying (12)	moveid integer	accuracy integer	mtype integer	pp integer
1	1	fire	flamethrower	1	13	1	50
2	2	water	water gun	2	23	2	75
3	1	fire	ember	3	13	1	80
4	2	water	whirlpool	4	12	2	70
5	3	grass	razor leaf	6	22	3	50
6	1	fire	flame blitz	5	17	1	90
7	4	psychic	dream eater	7	20	4	35
8	5	rock	stone edge	8	15	5	45
9	3	grass	giga drain	9	19	3	45

Conclusion:

In conclusion, to effectively use the SQL JOIN clause to extract data from multiple tables, there must be a relationship between certain columns in the tables. This relationship is typically defined through the use of foreign keys, which establish a link between the related data in different tables. Without this relationship, the JOIN clause cannot effectively match the data from different tables, resulting in inaccurate or incomplete results. Therefore, it is important to properly establish and maintain these relationships in a database to ensure accurate data retrieval using SQL JOIN.

Post Lab Questions:

1. **What is a view?**

   a) A view is a special stored procedure executed when certain event occurs
   b) A view is a virtual table which results of executing a pre-compiled query
   c) A view is a database diagram
   d) None of the Mentioned

2. **What type of join is needed when you wish to include rows that do not have matching values?**

   A. Equi-join
   B. Natural join
   C. Outer join
   D. All of the mentioned

3. **Write SQL query including join operator to get following output:**

   Input Tables:

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

   Output Table:

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT