Data Management in RDBMS using SQL (Advanced SQL Operations)

Contents

- Advanced SQL: Joining
- Different types of Joining:
  - Equijoin
  - Non-Equijoin
  - Outer Join
  - Self Join
- Capabilities of Joining
- Subquery Operations
- Capabilities of Subqueries

Advanced SQL

The Advanced SQL topics we will cover here will show how structured query language can be used for some of the complex calculations on the data that is stored in a database.

Joining

“Joining” is a special way of cross product to multiple database tables, which allows to access data from multiple tables at the same time through a query.

The concept is to join one tuple from a table with other tuple/tuples in another table as we do in calculating Cartesian products.

Example of Joining

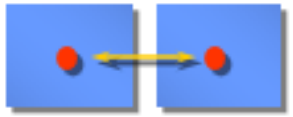
Empno	Name	……	Deptno
7839	King	……	10
7698	Blake	……	30
……	……	……	……
7934	Miller	……	10

Deptno	Name	Location
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Find the Employee's number, name and the location where they are working.

Empno	Name	Location
7839	King	New York
7698	Blake	Chicago
……	……	……
7934	Miller	New York

Types of Joins: Equijoin



Equijoin

Equijoin: If in two different tables there is a common column defined which is a primary key constraint in one table and referred as a foreign key constraint in the other table, then Equijoin is applicable. This is also called inner join.

For instance,

SELECT *table1.column, table2.column*

FROM *table1, table2*

WHERE *table1.column1 = table2.column2*;

Example Equijoin

EMPLOYEE

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

... 14 rows selected.

DEPARTMENT

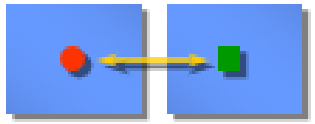
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

... 14 rows selected.

Foreign key Primary key

```
SELECT employee.empno, employee.ename,
       employee.deptno,   department.deptno, department.loc
       FROM   employee, department
       WHERE  employee.deptno = department.deptno;
```

Types of Joins: Non-Equijoin



Non-Equijoin

Non-Equijoin: When there is no direct correspondence between two tables, yet the relationship can be obtained from a common factor by using different comparative operators other than equal "=", non-equijoins are applied.

For instance,

SELECT *table1.column, table2.column*

FROM *table1, table2*

WHERE *table1.column1 > table2.column2*;

[Comparative operators are applied based on the query rules]

Example Non-Equijoin

EMPLOYEE

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950

...
14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"salary in the EMP table is between low salary and high salary in the SALGRADE table"

```
SELECT e.ename, e.sal, s.grade
    FROM employee e, salarygrade s
    WHERE  e.sal  BETWEEN s.losal AND s.hisal;
```

Types of Joins: Outer Joins

Outer Join

Outer Join: While joining two tables if the situation is such that some row/rows do not exist in the referred table as it appears in the primary table then implicitly the join needs to be specified as “LEFT” or “RIGHT” join so that all the available rows in both the tables appears in the result.

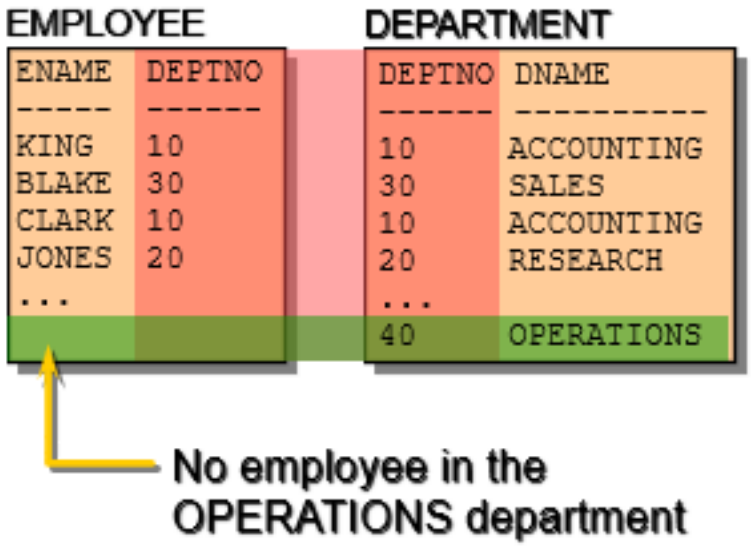
For instance,

SELECT *table1.column1, table2.column2,*

FROM *table1 RIGHT JOIN table2*

ON *table1.column1 = table2.column1*;

Example Outer Join



SELECT employee.name, department.deptno, department.name

FROM employee RIGHT JOIN department ON employee.deptno=department.deptno

Types of Joins: Self Join

Self Join

Self Join: This nothing but joining a column of the same table with another column.

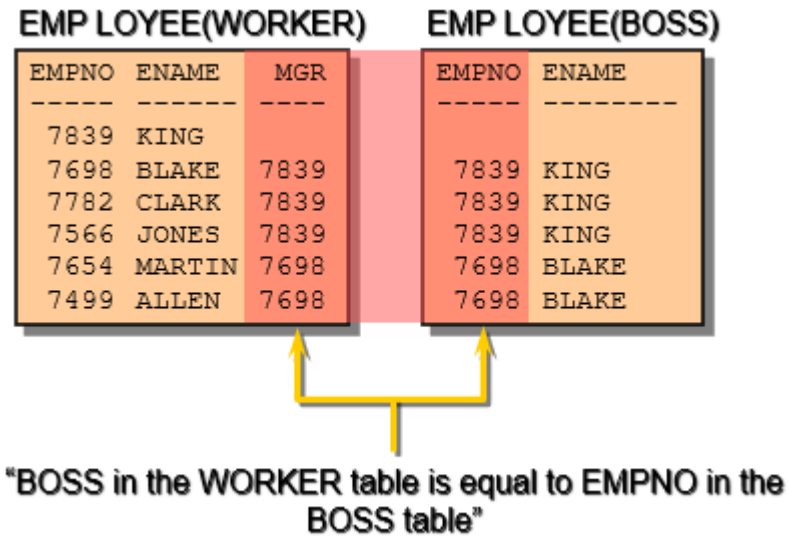
For instance,

SELECT a.*column*, *b.column*

FROM *table a*, *table b*

WHERE *a.column* = *b.column*;

Example Self Join



SELECT worker.name, boss.name

FROM employee worker, employee boss

WHERE worker.boss = boss.empno

Capabilities of “Joining” in the Databases

- Query execution gets faster as whilst joining the columns are specifically indexed and named.
- There are different types of joins which can be utilized to retrieve information from the database even if they belong in different objects.

Subquery/ Nested Query

- Subquery can be inside the select, insert, update, or, delete block of an SQL statement
- Subquery is usually added within the WHERE clause of another SQL SELECT statement within that WHERE clause
- Comparison operators or mathematical operators can be used to perform multi-row function operations using subqueries

Subquery/ Nested Query

```
SELECT [select_list]
FROM table_name
WHERE expr operator (SELECT [select_list]
                     FROM table_name)
```

***The inner block of the subquery works first and then the result of the inner block is sent to the next outer block for further comparison if there is any.

Types of Subquery/ Nested Query

- Single Row Subquery: Returns one or Zero row
- Multiple Row Subquery: Returns more than one row

Example Single Row Subquery

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

```
SELECT ename, job, sal
FROM emp
WHERE sal > (SELECT sal)
            FROM emp
            WHERE empno=7029);
```

Example Multi-Row Subquery

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

```
SELECT ename, job, sal
FROM emp
WHERE sal > ANY (SELECT sal)
                FROM emp
                WHERE job<>'CLERK)';
```

Capabilities of Subquery

- Subquery allows alternative ways to perform operations without complex joining to some extent
- Each part of the SQL statement can be isolated in a structured manner
- Provides better readability with a step-by-step approach


Do you have any
QUESTIONS?