Database Design & Applications (SS ZG 518)

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

SQL

Ex.

Retrieve all employees in department 5 whose salary is between 30,000 and 40,000.

```
SELECT *
FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) and Dno= 5;
```

Order By:

The default ordering of the result is ascending. We can specify the key word DESC if we wish a descending order of values.

Ex.

```
SELECT Fname, Dno, age
FROM EMPLOYEE
WHERE salary > 30,0000
ORDER BY   Dno;
```

```
ORDER BY   dno desc,  age asc
```

Nested Queries

Ex. Retrieve the name of each employee who has a dependent with the same name as the employee.

```
SELECT  E.Fname
FROM EMPLOYEE AS E
WHERE E.ssn IN(SELECT ESSN FROM DEPENDENT
        WHERE E.FNAME = DEPENDENT_NAME);
```

Correlated Nested Queries:

Whenever a condition in the WHERE clause of a nested query references some attribute of a relation declared in the outer query, then the two queries are said to be correlated.

Use of NOT EXISTS clause

Ex.

Retrieve the names, salary of employees who have no dependents

SELECT Fname, Salary
FROM EMPLOYEE
*WHERE NOT EXISTS (SELECT * FROM DEPENDENT WHERE SSN = ESSN);*

We can also use ‘EXISTS’ to check the existence of at least one tuple in the result.

It is also possible to use an explicit set of values in the WHERE – clause.

We can also check whether a value is NULL

Renaming Attributes in the Result

Ex.

```
SELECT name AS Emp_name
FROM EMPLOYEE
WHERE Dno = 5;
```

Join Operation

We can also perform

- Join – using key word ‘JOIN’
- Natural join – using key word ‘NATURAL JOIN’
- Left outer join – using key word ‘LEFT OUTER JOIN’
- Right outer join – using key word ‘RIGHT OUTER JOIN’

Ex: Select * from (Emp join Dept on dno=dnum) where dname=‘HR’;

Aggregate Functions and Grouping

- COUNT
- SUM
- MAX
- MIN
- AVG

Ex. *SELECT SUM (Salary), AVG (Salary) from EMPLOYEE;*

Ex. To retrieve number of rows in Employee table

SELECT count (*)
FROM EMPLOYEE;

Ex. Retrieve the name of employees who have two or more dependents

SELECT Fname
FROM EMPLOYEE
WHERE (SELECT COUNT (*) FROM DEPENDENT WHERE SSN = ESSN) > = 2;

Group by

Ex. For each department retrieve the department number and no of employees.

```
SELECT dno, count (*)
FROM EMPLOYEE
GROUP BY Dno;
```

Group by and Having clause

Ex. Retrieve the department number and no of employees for the departments which have more than 5 employees working for it.

```
SELECT dno, count (*)
FROM EMPLOYEE
GROUP BY Dno
HAVING count(*)>5;
```

INSERT operation

For Inserting a new tuple into the relation

General Form

INSERT INTO <table name>
VALUES(v_1, v_2, v_3,……..v_n);

Ex. *INSERT INTO DEPARTMENT*
VALUES('MARKETING',10, 103, '2000-06-25');

Deleting a tuple

Ex. *DELETE FROM <table name>*
WHERE <condition>;

Ex. *DELETE FROM DEPARTMENT*
WHERE dnumber=10;

If we don't specify the condition all tuples are deleted.

Update command

Ex. *UPDATE EMPLOYEE*
SET salary = 60000
WHERE ssn = 141;

Updates tuples in Employee table for the tuples with ssn = 141, sets the value of the attribute salary to 60,000

Views in SQL

A view in SQL is a single table that is derived from other tables.

These other tables are known as *base tables*.

A view does not necessarily exist in physical form, it can be considered as a *virtual table*.

The tuples of base tables are actually stored in database.

This limits the updates on views.

In fact when a view is updated, the corresponding base tables are the structures which are to be updated.

This makes update operations on views complex.

Creating View

```
CREATE VIEW EMP_DETAILS
AS SELECT name, salary, dname, age, dloc
    FROM EMPLOYEE, DEPARTMENT
    WHERE dno = dnumber;
```

Whenever the view definition is executed, the new temporary table is generated with specified attributes from specified base tables.

View definitions are stored in database, not the result of the view. From then onwards view can be seen as a table and queries can be posed on it.

Ex. *SELECT name, dname FROM EMP_DETAILS WHERE dno = 5;*

Here EMP_DETAILS is a view. Where this query is executed, first the view definition for EMP_DETAILS is executed and the select and where operation are performed on the temporary table.

Note:

- A view is always up to date.
- Updates are generally not possible on views.
- Meant for querying only.
- Some times it is possible to store views for some duration.
- Those views are known as *materialized views*.

Oracle PL/SQL

What is PL/SQL

SQL does not support conditional execution (If-Then-else) and looping(do-while)

PL/SQL(Procedural SQL) is a programming language extension to SQL.

Extensions include:

1. If-then-Else and Do-while for logic representation
2. Variable declaration and assignment
3. Error management

This facilitates isolation of code from application.

SQL-99 defined use of persistent stored modules (PSM).

Persistent stored modules (PSM) :

Is a block of code containing standard SQL statements and procedural extensions that is stored and executed at the DBMS Server.

Admin can control the access to these PSMs.

The procedural code is executed as a unit by the DBMS when invoked.

End users can use Pl/SQL to create:

1. Anonymous Pl/SQL blocks
2. Triggers
3. Stored Procedures
4. PL/SQL functions (different from SQL functions)

PL/SQL Code block

In Oracle we can write PL/SQL code block by enclosing commands in Begin-End clauses.

Ex:

```
Begin
insert into emp_dept values(101, 20);
end;
/
```

DBMS Output messages

innovate achieve lead

```
SQL>  set serveroutput on;

DBMS_OUTPUT.PUTLINE(‘ string ‘);

SQL> Begin
  2  insert into emp_dept values(121, 80);
  3  dbms_output.put_line('Inserted Emp 121 and 80');
  4  end;
  5  /
Inserted Emp 101 and 30
PL/SQL procedure successfully completed.
```

```
SQL> declare
        teamsize number(3);
        begin
        select count(eid) into teamsize from emp;
        dbms_output.put_line('Number of employees are: '||
                                                    teamsize);
        end;
        /
Number of employees are: 5

PL/SQL procedure successfully completed.
```

Triggers

A *trigger* is a procedural SQL code that is automatically invoked by the RDBMS upon the occurrence of a data manipulation event.

1. A trigger is invoked before or after a data row is inserted, deleted or updated.
2. A trigger is associated with a database table.
3. Each table may have one or more triggers.
4. Triggers can be used to enforce constraints
5. Triggers can be used to insert/update records and to call stored procedures.
6. Used for auditing purpose (creating logs)
7. Generation of derived values.

```
SQL> create or replace  trigger T_D1
   after insert on dept
begin
dbms_output.put_line('Insertion done  ');
end;
/

Trigger created.

SQL> insert into dept values(90, ‘Aad', ‘Bho');
Insertion done
1 row created.
```

Check constraints

innovate achieve lead

```
SQL> create table worker(wid int primary key, salary int, constraint
  sal_check check(salary >10000 and salary<20000));
Table created.

SQL> insert into worker values(11, 7000);
insert into worker values(11, 7000)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.SAL_CHECK) violated
```

SQL Stored procedure

innovate achieve lead

A stored procedure is a named collection of procedural and SQL statements.

Like triggers procedures can be stored in the database.

Set of SQL statements that perform a business transaction can be encapsulated within a procedure and stored at the server.

Can be called by invocation as required. This reduces the network traffic.

Helps in reducing the code duplication. Can be called by many applications.

```
SQL> create or replace procedure st_insert(s in number, m in number) as
  2  begin
  3  insert into student values(s,m);
  4  end;
  5  /

Procedure created.

SQL> exec st_insert(11,40);

PL/SQL procedure successfully completed.
```

```
SQL> begin
  2  st_insert(33,70);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> select * from student;

       SID      MARKS
---------- ----------
        11         40
        44         20
        33         70
```

SQL Stored Functions

```
SQL> create function get_marks (s in number)
  2  return number
  3  is m number;
  4  begin
  5  select marks  into m from student where sid=s;
  6  return (m);
  7  end;
  8  /

Function created.
```

Function call

```
SQL> declare
  2  v number;
  3  begin
  4  v:=get_marks(44);
  5  dbms_output.put_line('Marks for student 44 is: '||v   ||'  ');
  6  end;
  7  /
```

Marks for student 44 is: 20

PL/SQL procedure successfully completed.

Summary

- ✓ *What is SQL*
- ✓ *What are the features supported by SQL*
- ✓ *How to create relational schemas using SQL*
- ✓ *How to specify queries in SQL*
- ✓ *How to write nested queries in SQL*
- ✓ *Writing queries using the clauses EXISTS, NOT XISTS, BETWEEN AND, IN, NOT IN*
- ✓ *How to perform explicit JOIN operations*
- ✓ *How to use GROUP BY and HVING*
- ✓ *The concept of views in SQL*
- ✓ *PL/SQL Concepts*