Database Systems (CSF 212)

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Structured Query Language (SQL)

innovate achieve lead

Content

- *Introduction to SQL*
- *Features of SQL*
- *DDL Statements*
- *DML commands*
- *Nested queries and correlated nested queries*
- *Use of EXISTS and NOT EXISTS*
- *Explicit join operations*
- *Aggregate functions*
- *Group by and Having classes*
- *Insert/ Update / Delete operations*
- *Views in SQL*

Introduction to SQL

innovate achieve lead

- SQL (Structured Query language) is the most widely used commercial query language for relational databases.

- SQL was introduced by IBM(1970).

- We study SQL -3 or SQL – 99 which was introduced in 1999 by ANSI (American National Standards Institute) and ISO jointly.
- SQL is a user friendly query language.

- Now-a-days almost all relational databases like – Oracle, MySQL, IBM's DB_2, Informix etc., support SQL.

- SQL is a high-level declarative language to specify data retrieval requests for data stored in relational databases.

- Its declarative because we just specify what to be extracted, rather than how to do it.

- SQL is relationally complete, meaning that any query that is expressed in relational algebra or calculus can also be written in SQL.

- SQL also supports additional features that are not existing in formal languages.

- SQL is a standard and many vendors implement it in their own way without deviating from the standard specifications.

Features of SQL

innovate achieve lead

1. ***DDL*** (Data Definition Language) Set of commands to support creation, deletion and modification of table structures and views.

2. ***DML*** (Data Manipulation Language) Set of commands to pose queries, insert new tuples, and update/delete existing tuples.

3. ***Embedded SQL***: Allows users to call SQL code from host languages like C, C++ & Java.

4. ***Triggers***: Actions executed by the DBMS whenever changes to the database meet specified conditions. Action to be performed and the set of conditions can be defined in "Triggers".

5. ***Transaction Management***: to perform roll-back / commit actions.

6. ***Indexes:*** Indexes can be created to speed up the access to data stored in DB.

DEPT (dnum, dname, dloc)

EMP (eid, ename, esal, dno, ecity)

PROJ(pnum, pname)

EMP_PROJ (eno, pno, hrs)

DDL Commands

CREATE TABLE DEPT(DNUM INT, DNAME VARCHAR(5), DLOC VARCHAR(10), PRIMARY KEY (DNUM), UNIQUE (DNAME));

CREATE TABLE EMP(EID INT PRIMARY KEY, ENAME VARCHAR(20), ESAL INT, DNO INT, ECITY VC(10), FOREIGN KEY (DNO) REFERENCES DEPT(DNUM));

CREATE TABLE PROJ(PNUM INT PRIMARY KEY, PNAME VARCHAR(10));

CREATE TABLE EMP_PROJ(EINO INT , PNO INT, HRS INT, PRIMARY KEY (ENO, PNO), FOREIGN KEY (ENO) REFERENCES EMP(EID), FOREIGN KEY (PNO) REFERENCES PROJ(PNUM));

ALTER TABLE EMPLOYEE ADD CONSTRINT EMPFK FOREIGN KEY (DNO)

DROPPING TABLE EMP

Drop table EMP;

Adding New column to EMP

ALTER TABLE EMP ADD ‘CITY’ VARCHAR (20);

TO DROP A COLUMN

ALTER TABLE EMP DROP AGE CASCADE/RESTRICT;

We can also give names to constraints and later use the names to access those constraints and alter them.

DML Commands

innovate achieve lead

DML (Data Manipulation)

- Selecting tuples, columns (querying)
- Inserting new tuples
- Updating existing tuples
- Deleting existing tuples

DEPT (dnum, dname, dloc)

EMP (eid, ename, esal, dno, ecity)

PROJ(pnum, pname)

EMP_PROJ (eno, pno, hrs)

Basic Query Statements

SQL has 'SELECT' statement for retrieving information from the database. This SELECT has no relationship with select (σ) operation in relational algebra. All the queries mentioned here are specified on the COMPANY database given in Fig. 3.1.

THE SELECT – FROM – WHERE CONSTRUCT:

```
SELECT < attribute list>      // attribute names to be retrieved
FROM < table list >           // names of relation involved
WHERE < condition>            // Boolean expression to identify the
                                 tuples to be extracted.
```

Ex.

```
SELECT eid, esal
FROM EMP
WHERE ename = 'John';
```

Ex. Joining tables using where clause

```
SELECT eid, ename
FROM EMP, DEPT
WHERE dname = 'Research" and dnum = dno;
```

SQL> select * from emp;

EID	ENAME	EAGE	PESAL	DNO
101	Raju	29	12000	10
103	Ramesh	45	34000	30
109	John	34	14000	10
110	Mohan	40	10000	

SQL> select * from dept;

DNUM	DNAME	DLOC
10	HR	HYD
20	MARK	HYD
30	PROD	DEL

SQL> select * from emp where dno=null;

no rows selected

SQL> select * from emp, dept where dno=dnum;

EID	ENAME	EAGE	PESAL	DNO	DNUM	DNAME	DLOC
101	Raju	29	12000	10	10	HR	HYD
103	Ramesh	45	34000	30	30	PROD	DEL
109	John	34	14000	10	10	HR	HYD

SELECT eid
FROM EMPLOYEE;

SQL> select * from emp;

EID	ENAME	EAGE	PESAL	DNO
101	Raju	29	12000	10
103	Ramesh	45	34000	30
109	John	34	14000	10
110	Mohan	40	10000	
117	Malik	50	35000	30

SQL> select dno from emp;

DNO
10
30
10
30

SQL> select distinct dno from emp;

DNO
30
10

Ex.

SELECT eid, dname
FROM EMP, DEPT;

This will retrieve eid, dname from the relation which is result of cross product of employee and department tables.

Ex.
*SELECT **
FROM EMP
WHERE dno = 5;

The above query will retrieve all the columns from employee table for the tuples where Dno = 5.

Ex. *SELECT ALL esal*
FROM EMP;

Retrieves all salaries (including duplicates) from employee table.

Ex. *SELECT DISTINCT esal*
FROM EMP;

Retrieves distinct values for ‘salary’ attribute

We also have following operations in SQL

Union	(for Union)
Except	(for Difference)
Intersect	(for Intersection)

Duplicate tuples are eliminated from the result.

```
SQL> select * from emp;

       EID ENAME          EAGE      PESAL       DNO
---------- ---------- ---------- ---------- ----------
       101 Raju               29      12000         10
       103 Ramesh             45      34000         30
       109 John               34      14000         10
       110 Mohan              40      10000
       117 Malik              50      35000         30

SQL> (select eid, ename from emp where pesal>=35000)
        UNION
        (select eid, ename from emp where dno=10);

       EID ENAME
---------- ----------
       101 Raju
       109 John
       117 Malik
```

```
SQL> select * from emp;

       EID ENAME          EAGE      PESAL       DNO
---------- ---------- ---------- ---------- ----------
       101 Raju               29      12000         10
       103 Ramesh             45      34000         30
       109 John               34      14000         10
       110 Mohan              40      10000
       117 Malik              50      35000         30
```

```
SQL> (select eid, ename from emp where pesal>13000)
        INTERSECT
        (select eid, ename  from emp where dno=10);

       EID     ENAME
----------     ----------
       109      John
```

```
SQL> select * from emp;

       EID ENAME          EAGE      PESAL       DNO
---------- ---------- ---------- ---------- ----------
       101 Raju               29      12000         10
       103 Ramesh             45      34000         30
       109 John               34      14000         10
       110 Mohan              40      10000
       117 Malik              50      35000         30
```

```
SQL> (select eid, ename from emp where pesal>=35000)
        MINUS
        (select eid, ename from emp where dno=10);

       EID ENAME
---------- ----------
       117 Malik
```

Substring Comparisons in SQL

The character '%' replaces an arbitrary number of characters, and '_' (underscore) replaces a single character.

Ex. To retrieve all employees whose name has 'Kumar' as substring.

```
SELECT eid, esal
FROM EMP
WHERE ename LIKE  '% Kumar%';
```

Ex. To retrieve the resulting salaries if every employee working in the 'Accounts' project is given a 10% raise.

```
SELECT eid, ename, 1.1* esal
FROM EMP, EMP_PROJ, PROJECT
WHERE eid = eno AND pno = pnum AND pname = 'Accounts';
```

SQL> select * from emp;

EID	ENAME	EAGE	PESAL	DNO
101	Raju	29	12000	10
103	Ramesh	45	34000	30
109	John	34	14000	10
110	Mohan	40	10000	
117	Malik	50	35000	30

SQL> select * from emp where ename like '_a%';

EID	ENAME	EAGE	PESAL	DNO
101	Raju	29	12000	10
103	Ramesh	45	34000	30
117	Malik	50	35000	30

```
SQL> select * from emp;

       EID ENAME          EAGE      PESAL       DNO
---------- ---------- ---------- ---------- ----------
       101 Raju               29      12000         10
       103 Ramesh             45      34000         30
       109 John               34      14000         10
       110 Mohan              40      10000
       117 Malik              50      35000         30

SQL> select * from emp where ename like '%n';

       EID ENAME          EAGE      PESAL       DNO
---------- ---------- ---------- ---------- ----------
       109 John               34      14000         10
       110 Mohan              40      10000
```

SQL> select * from emp;

EID	ENAME	EAGE	PESAL	DNO
101	Raju	29	12000	10
103	Ramesh	45	34000	30
109	John	34	14000	10
110	Mohan	40	10000	
117	Malik	50	35000	30

SQL> select eid, ename, 1.5*pesal from emp where dno=10;

EID	ENAME	1.5*PESAL
101	Raju	18000
109	John	21000

Ex.

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

```
SELECT *
FROM EMP
WHERE (esal 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 ename, dno
FROM EMP
WHERE esal > 30,000
ORDER BY   dno;
```

```
ORDER BY   dno desc,  name asc
```

SQL> select * from dept;

DNUM	DNAME	DLOC
10	HR	HYD
20	MARK	HYD
30	PROD	DEL

SQL> select * from dept order by dloc asc;

DNUM	DNAME	DLOC
30	PROD	DEL
10	HR	HYD
20	MARK	HYD

SQL> select * from dept order by dname asc;

DNUM	DNAME	DLOC
10	HR	HYD
20	MARK	HYD
30	PROD	DEL

SQL> select * from dept order by dloc desc, dname asc;

DNUM	DNAME	DLOC
10	HR	HYD
20	MARK	HYD
30	PROD	DEL

Nested Queries

Ex. Retrieve the id and name of employees whose department name is located in Chennai

```
SELECT  E.eid, E.ename
FROM EMP AS E
WHERE E.dno IN(SELECT DNUM FROM DEPT
        WHERE DLOC =  'CHENNAI');
```

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 id, salary of employees who have no projects

```
SELECT E.eid, E. esal
FROM EMP AS E
WHERE NOT EXISTS (SELECT * FROM EMP_PROJ AS EP WHERE
E.EID = EP.ENO);
```

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. 3

```
SELECT ename AS Emp_name
FROM EMP
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

SQL> select * from emp, dept where dno=dnum;

EID	ENAME	EAGE	PESAL	DNO	DNUM	DNAME	DLOC
101	Raju	29	12000	10	10	HR	HYD
103	Ramesh	45	34000	30	30	PROD	DEL
109	John	34	14000	10	10	HR	HYD

SQL> select * from (emp left outer join dept on dno=dnum);

EID	ENAME	EAGE	PESAL	DNO	DNUM	DNAME	DLOC
109	John	34	14000	10	10	HR	HYD
101	Raju	29	12000	10	10	HR	HYD
103	Ramesh	45	34000	30	30	PROD	DEL
110	Mohan	40	10000				

SQL> select * from (emp right outer join dept on dno=dnum);

EID	ENAME	EAGE	PESAL	DNO	DNUM	DNAME	DLOC
101	Raju	29	12000	10	10	HR	HYD
103	Ramesh	45	34000	30	30	PROD	DEL
109	John	34	14000	10	10	HR	HYD
					20	MARK	HYD

SQL> select * from (emp full outer join dept on dno=dnum);

EID	ENAME	EAGE	PESAL	DNO	DNUM	DNAME	DLOC
101	Raju	29	12000	10	10	HR	HYD
103	Ramesh	45	34000	30	30	PROD	DEL
109	John	34	14000	10	10	HR	HYD
110	Mohan	40	10000				
					20	MARK	HYD

Ex. *SELECT SUM (esal), AVG (esal) from EMP;*

Ex. To retrieve number of rows in Employee table

SELECT count (*)
FROM EMP;

Ex. Retrieve the eid and name of employees who have two or more Projects

SELECT eid, ename
FROM EMP
WHERE (SELECT COUNT (*) FROM EMP_PROJ WHERE ENO = EID) > = 2;

Group by

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

```
SELECT dno, count (*)
FROM EMP
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 EMP
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 DEPT*
VALUES(10, 'HR', 'MUMBAI');

Deleting a tuple

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

Ex. *DELETE FROM DEPT*
WHERE dnum=10;

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

Update command

Ex.

```
UPDATE EMP
SET esal = 60000
WHERE eid = 141;
```

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 eid, ename, dname,
    FROM EMP, DEPT
    WHERE dno = dnum;
```

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 eid, ename FROM EMP_DETAILS WHERE dname='HR' ;*

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*.

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*