Database Design & Applications (SS ZG 518)

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. Latest version is SQL-2003.

- 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.

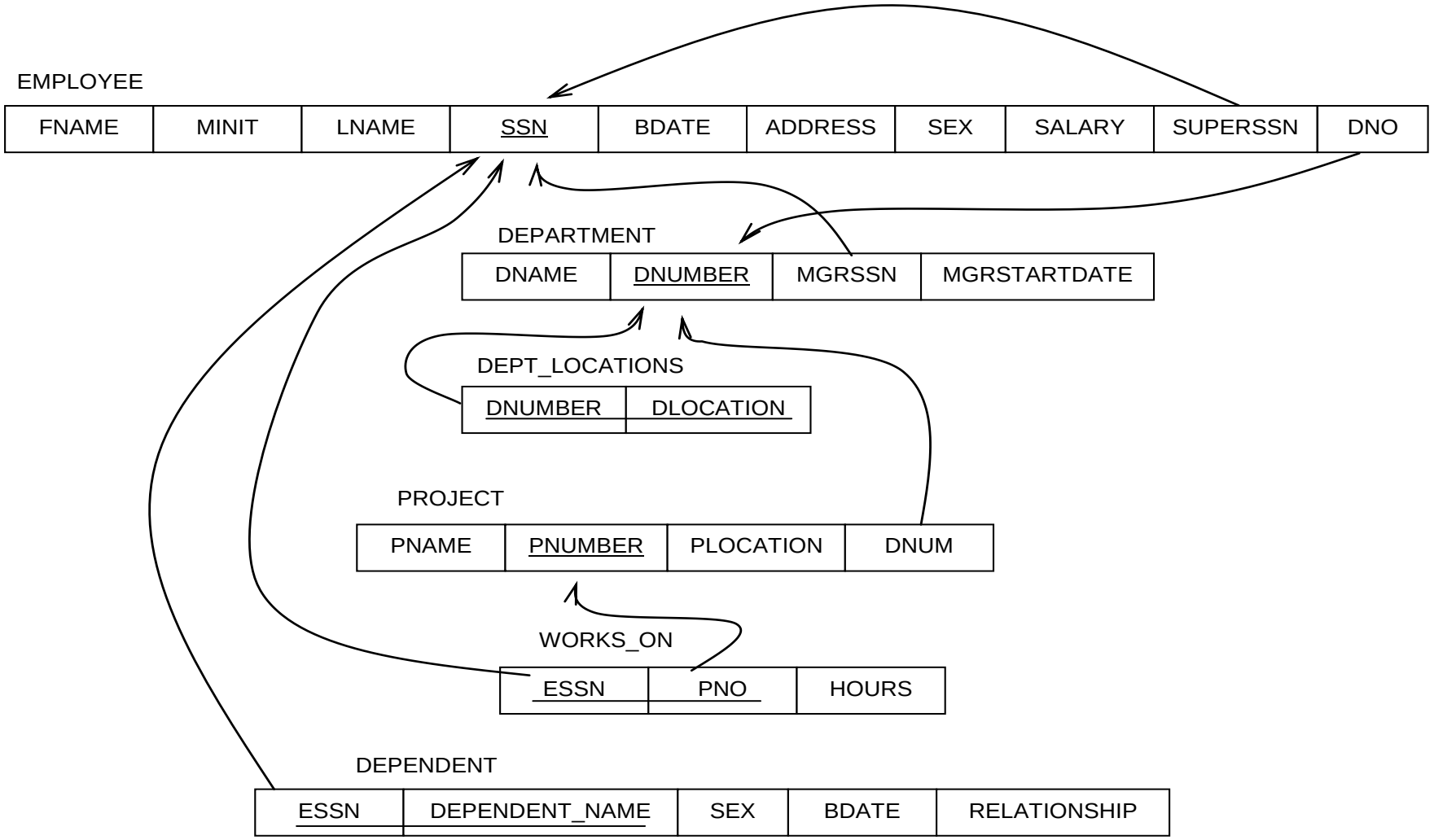
3. ***Embedded SQL***: Allows users to call SQL code from host languages like C, C++ & Java.

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.

Example SQL statements



DDL Commands

innovate achieve lead

The DDL (Create) statement for creating Employee table.

```
CREATE TABLE EMPLOYEE(
FNAME          VARCHAR(15)      NOT NULL,
MINIT          CHAR,
 LNAME         VARCHAR(15)      NOT NULL,
SSN            CHAR(9),
 BDATE         DATE,
 ADDRESS       VARCHAR(30),
 SEX           CHAR,
 SALARY        DECIMAL(10, 2),
 SUPERSSN      CHAR(9),
 DNO           INT              NOT NULL    DEFAULT 1,
PRIMARY KEY (SSN),
FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)
);

ALTER TABLE EMPLOYEE ADD CONSTRINT EMPFK FOREIGN KEY (DNO)

REFERENCES DEPT(DNUMBER) ON DELETE SET DEFAULT/ SET NULL/CASCADE  ON UPDATE
CASCADE;
```

```
CREATE TABLE DEPARTMENT(
DNAME          VARCHAR(15)        NOT NULL,
DNUMBER        INT                NOT NULL,
MGRSSN         CHAR(9)            NOT NULL,
MGRSTARTDATE   DATE,

PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN)REFERENCES EMPLOYEE (SSN));
```

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

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. 1

```
SELECT bdate, address
FROM EMPLOYEE
WHERE Fname = 'john';
```

Retrieves the birthdate & address of the employees whose first name is 'John'.

Ex. 2 Join operation

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname = 'Research" and Dnumber = Dno;
```

Retrieves first name, last name and address from joined truples from employee and department. The joining condition is *Dnumber* in department table is equal to *Dno* in employee table. We can also do aliasing (renaming) of tables to avoid ambiguity.

Ex. 3

```
SELECT   E.Fname, S.Fname
FROM     EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.superssn = S.ssn;
```

Retrieves the employee's first name and his immediate supervisor's first name. This is an example of self joining.

Ex. 4

```
SELECT ssn
FROM EMPLOYEE;
```

Retrieves all 'ssn' from employee table.

Ex. 5

SELECT ssn, dname
FROM EMPLOYEE, DEPARTMENT;

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

Ex. 6: *SELECT **
FROM EMPLOYEE
WHERE Dno = 5;

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

Ex. 7: *SELECT ALL salary*
FROM EMPLOYEE;

Retrieves all salaries (including duplicates) from employee table.

Ex. 8 *SELECT DISTINCT salary*
FROM EMPLOYEE;

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.

Using Union

```
SQL> (select eid from emp where asal>20000)
union (select eid from emp where dno=10);
```

Minus operation

```
SQL> (select eid from emp where asal>20000)
minus (select eid from emp where dno=10);
```

Substring Comparisons in SQL

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

Ex. 9 To retrieve all employees whose address is in Houston, Texas

```
SELECT fname
FROM EMPLOYEE
WHERE Address LIKE '% Houston, Texas %';
```

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

```
SELECT Fname, 1.1* salary
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE ssn = essn AND pno = pnumber AND pname = 'Accounts';
```

```
select eid,ename from emp where dno not
in(select dnum from dept where dloc like '_H_');
```

Ex. 11

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. 12

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

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

```
select * from emp order by dno,asal;
```

Nested Queries

Ex.1 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.