Almasri – Company file:

Step – 1:

**CREATE TABLE** EMPLOYEE(

Fname VARCHAR(15) **NOT NULL**,

Minit CHAR,

Lname VARCHAR(15) **NOT NULL**,

Ssn CHAR(9) **NOT NULL**,

Bdate DATE,

Address VARCHAR(30),

Sex CHAR,

Salary DECIMAL(10,2),

Super\_ssn CHAR(9),

Dno INT **NOT NULL**,

**PRIMARY KEY** (Ssn));

**CREATE TABLE** DEPARTMENT(

Dname VARCHAR(15) **NOT NULL**,

Dnumber INT **NOT NULL**,

Mgr\_ssn CHAR(9) **NOT NULL**,

Mgr\_start\_date DATE,

**PRIMARY KEY** (Dnumber),

**UNIQUE** (Dname));

**CREATE TABLE** EMPLOYEE(

Fname VARCHAR(15) **NOT NULL**,

Minit CHAR,

Lname VARCHAR(15) **NOT NULL**,

Ssn CHAR(9) **NOT NULL**,

Bdate DATE,

Address VARCHAR(30),

Sex CHAR,

Salary DECIMAL(10,2),

Super\_ssn CHAR(9),

Dno INT **NOT NULL**,

**PRIMARY KEY** (Ssn),

**FOREIGN KEY** (Super\_ssn) **REFERENCES** EMPLOYEE(Ssn));

**FOREIGN KEY** (Dno) **REFERENCES** DEPARTMENT(Dnumber) );

**CREATE TABLE** DEPARTMENT(

Dname VARCHAR(15) **NOT NULL**,

Dnumber INT **NOT NULL**,

Mgr\_ssn CHAR(9) **NOT NULL**,

Mgr\_start\_date DATE,

**PRIMARY KEY** (Dnumber),

**UNIQUE** (Dname),

**FOREIGN KEY** (Mgr\_ssn) **REFERENCES** EMPLOYEE(Ssn) );

CREATE TABLE dept\_locations(

Dnumber INT NOT NULL,

Dlocation varchar(15) NOT NULL,

PRIMARY KEY (Dnumber, Dlocation),

FOREIGN KEY (Dnumber) REFERENCES department (Dnumber));

Inserting Data in employee table:

INSERT INTO EMPLOYEE VALUES ('John', 'B', 'Smith', '123456789', '1965-01-09', '731 Fondren, Houston, TX', 'M', 30000, '333445555',5);

INSERT INTO EMPLOYEE VALUES ('Franklin', 'T', 'Wong', '333445555', '1955-12-08', '638 Voss, Houston, TX', 'M', 40000, '888665555',5);

INSERT INTO EMPLOYEE VALUES ('Janifer', 'S', 'Wallace', '987654321', '1941-06-20', '291, Barry, Balaire, TX', 'F', 43000, '888665555',4);

INSERT INTO EMPLOYEE VALUES ('Ahmed', 'V', 'Jabbar', '987987987', '1960-03-20', '980 ,Dallas Houston TX', 'M', 25000, '987654321',4);

INSERT INTO EMPLOYEE VALUES ('James', 'E', 'Borg', '888665555', '1937-11-10', '450 ,Stone, Houston TX', 'M', 55000, NULL,1);

INSERT INTO EMPLOYEE VALUES ('Alicia', 'J', 'Zelaya', '999887777', 1968-01-19, '3321 ,Castle, Spring TX', 'F', 25000, '987654321',4);

Inserting Data in department table:

INSERT INTO DEPARTMENT VALUES ('Administration', 4, '987654321', '1995-01-01');

INSERT INTO DEPARTMENT VALUES ('Research', 5, '333445555', '1988-05-22');

INSERT INTO DEPARTMENT VALUES ('HeadQuarters', 1, '888665555', '1981-08-19');

ALTER TABLE EMPLOYEE ADD CONSTRAINT FOREIGN KEY (Dno) REFERENCES DEPARTMNET (Dnumber);

ALTER TABLE EMPLOYEE ADD CONSTRAINT FOREIGN KEY (Super\_Ssn) REFERENCES EMPLOYEE (Ssn);

alter table student change column salary sal int(4);

UPDATE <TABLE NAME> SET <ATTRIBUTE NAE> = NEW\_VALUE [WHERE <CONDITION>]

**STRUCTURED QUERY LANGUAGE (SQL)**

**Select attr1, attr2, …., attrn FROM <table> <WHERE condition>**

**Query 1.** Retrieve the birth date and address of the employee(s) whose name is ‘John B. Smith’.

Q1: SELECT Fname, Minit, Lname, Bdate, Address

FROM EMPLOYEE

WHERE Fname= 'John' AND Minit= 'B' AND Lname= 'Smith';

SELECT \*

FROM EMPLOYEE

WHERE Fname= 'John' AND Minit= 'B' AND Lname= 'Smith';

SELECT Fname, Ssn, Super\_ssn, dno

FROM EMPLOYEE

WHERE Salary > 25000 AND SEX = 'F';

**Query 2.** Retrieve the name and address of all employees who work for the 'Research' department.

**Q2: SELECT** Fname, Lname, Address

**FROM** EMPLOYEE, DEPARTMENT

**WHERE** Dname='Research' **AND** Dnumber=Dno;

**Query 3.** For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

**Q3: SELECT** Pnumber, Dnum, Lname, Address, Bdate

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn **AND** Plocation='Stafford';

**SELECT** Pnumber, Dname, Lname, Address, Bdate

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn **AND** Plocation='Stafford';

**SELECT** Pnumber AS 'Project\_number', Dname AS 'Department\_name',

Lname AS 'Last\_Name', Address AS 'Emp\_Address', Bdate AS 'Birth\_Date'

**FROM** PROJECT P, DEPARTMENT D, EMPLOYEE E

**WHERE P.**Dnum=D.Dnumber **AND D.**Mgr\_ssn=E.Ssn **AND P.**Plocation='Stafford';

**Query 4:** For each employee, retrieve the employee’s first and last name and the first and last name of his or her immediate supervisor.

**SELECT** E.Fname, E.Lname, S.Fname, S.Lname, E.Super\_Ssn, S.Ssn

**FROM** EMPLOYEE **AS** E, EMPLOYEE **AS** S;

**SELECT** E.Fname, E.Lname, S.Fname, S.Lname

**FROM** EMPLOYEE **AS** E, EMPLOYEE **AS** S

**WHERE** E.Super\_ssn=S.Ssn;

In this case, we are required to declare alternative relation names E and S, called **aliases** or **tuple variables**, for the EMPLOYEE relation. An alias can follow the keyword **AS**, as shown in Q4, or it can directly follow the relation name—for example, by writing EMPLOYEE E, EMPLOYEE S in the FROM clause of Q8. It is also possible to **rename** the relation attributes within the query in SQL by giving them aliases.

USE of DISTINCT key word to avoid duplicate values.

Q5: Retrieve only the salary of employees

SELECT E.salary FROM EMPLOYEE E;

SELECT DISTINCT E.salary FROM EMPLOYEE E;

**UNION OF TWO TABLES**

**Query 6.** Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.

**Q6A: SELECT DISTINCT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Smith';

**Q6B:** **SELECT DISTINCT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Smith';

**Q6C: (SELECT DISTINCT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Smith')

UNION

(**SELECT DISTINCT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Smith');

**Q6A: SELECT DISTINCT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Wong';

**Q6B:** **SELECT DISTINCT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Wong';

**Q6C: (SELECT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Wong')

UNION

(**SELECT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Wong');

**Q7: INTERSECTion OF TWO TABLES**

**(SELECT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Wong')

**INTERSECT**

(**SELECT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Wong');

**Substring Pattern Matching and Arithmetic Operators**

In this section we discuss several more features of SQL. The first feature allows comparison conditions on only parts of a character string, using the **LIKE** comparison operator. This can be used for string **pattern matching**. Partial strings are specified using two reserved characters: % replaces an arbitrary number of zero or more characters, and the underscore (\_) replaces a single character. For example, consider the following query.

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign (%) represents zero, one, or multiple characters
* The underscore sign (\_) represents one, single character

SELECT \* FROM TABLE\_1 WHERE ATTRIBUTE LIKE (‘????’)

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

**Query 8.** Retrieve all employees whose address is in Houston, Texas.

**Q8: SELECT** Fname, Lname

**FROM** EMPLOYEE

**WHERE** Address **LIKE** '%Houston, TX%';

To retrieve all employees who were born during the 1950s, we can use Query 9A.

Here, ‘5’must be the third character of the string (according to our format for date),

so we use the value ‘\_ \_ 5 \_ \_ \_ \_ \_ \_ \_’, with each underscore serving as a placeholder

for an arbitrary character.

**Query 9A.** Find all employees who were born during the 1950s.

**Q9: SELECT** Fname, Lname

**FROM** EMPLOYEE

**WHERE** Bdate **LIKE** '\_\_5\_\_\_\_\_\_\_';

**Query 4.** Make a list of all project numbers for projects that involve an employee whose last name is ‘Wong’, either as a worker or as a manager of the department that controls the project.

**A = {1,2,3,4,5}**

**B = {2,3,5,6,7}**

**A U B = {1,2,3,4,5,6,7}**

**A ∩ B = {2,3,5}**

**UNION/INTERSECTION**

**Q4A:** (**SELECT DISTINCT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Wong' )

**UNION**

( **SELECT DISTINCT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Wong');

**INTERSECTION**

**Q4A:** (**SELECT DISTINCT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn

**AND** Lname='Wong' )

**INTERSECT**

( **SELECT DISTINCT** Pnumber

**FROM** PROJECT, WORKS\_ON, EMPLOYEE

**WHERE** Pnumber=Pno **AND** Essn=Ssn

**AND** Lname='Wong');

**Nested Queries, Tuples, and Set/Multiset Comparisons**

IN operator is used to filter data and allows to determine if the value matches any one of the list or result of subquery.

**SELECT \* FROM EMPLOYEE WHERE fname ='John' OR fname = 'Alicia' OR fname = 'Joyce';**

**Fname = {John, Franklin, Joyce, Ramesh, James, Robert, Jennifer, Ahmad, Alicia}**

**IN**

**{John, Alicia, Joyce}**

**= {John, Alicia, Joyce}**

**SELECT \* FROM EMPLOYEE WHERE fname IN ('John', 'Alicia', 'Joyce');**

Query: Retrieve the name of employees working in ‘Research’ department.

SELECT E.Fname, E.Lname FROM Employee E, Department D

WHERE D.Dname = 'Research' AND D.Dnumber = E.dno;

**Without using JOINing i.e. Cross Product**

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno = (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname = 'Research');

Query: Retrieve the name employees working in ‘Research’ or ‘Administration’ department.

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname = 'Research' OR D.Dname = 'Administration');

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname IN ( 'Research', 'Administration'));

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname NOT IN ( 'Research', 'Administration'));

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno NOT IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname IN ( 'Research', 'Administration'));

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno NOT IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname NOT IN ( 'Research', 'Administration'));

The first nested query selects the project numbers of projects that have an employee with last name ‘Smith’ involved as manager, while the second nested query selects the project numbers of projects that have an employee with last name ‘Smith’ involved as worker. In the outer query, we use the **OR** logical connective to retrieve a PROJECT tuple if the PNUMBER value of that tuple is in the result of either nested query.

**Pnumber = {1, 2, 3, 10, 20, 30}**

**Wong supervising = {1, 2, 3}**

**Set A = {1,2,3,10,20,30} IN {1,2,3} = {1,2,3}**

**IN command returns common of sets**

**Set B = Wong working as member of projects = {2, 3, 10, 20}**

**A union B means Wong as supervisor or member of projects = {1,2,3} U {2, 3, 10, 20} = {1,2,3,10, 20}**

**SELECT DISTINCT** Pnumber **FROM** PROJECT

**WHERE** Pnumber **IN**

(**SELECT** Pnumber

**FROM** PROJECT, DEPARTMENT, EMPLOYEE

**WHERE** Dnum=Dnumber **AND** Mgr\_ssn=Ssn **AND** Lname='Wong' )

**OR**

Pnumber **IN**

( **SELECT** Pno **FROM** WORKS\_ON, EMPLOYEE

**WHERE** Essn=Ssn **AND** Lname='Wong' );

If a nested query returns a single attribute *and* a single tuple, the query result will be a single (scalar) value. In such cases, it is permissible to use = instead of IN for the comparison operator. In general, the nested query will return a **table** (relation), which is a set or multiset of tuples.

SQL allows the use of **tuples** of values in comparisons by placing them within parentheses. To illustrate this, consider the following query:

**SELECT DISTINCT** Essn **FROM** WORKS\_ON

**WHERE** (Pno, Hours) **IN** ( **SELECT** Pno, Hours

**FROM** WORKS\_ON **WHERE** Essn='123456789' );

**Query 16.** Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

SELECT E.Fname, E.Lname From Employee E, Dependent D

WHERE E.Ssn = D.Essn AND E.Fname = D.Dependent\_name AND E.sex = D.sex;

**Q16:**

**SELECT** E.Fname, E.Lname **FROM** EMPLOYEE **AS** E

**WHERE** E.Ssn **=** ( **SELECT** Essn **FROM** DEPENDENT **AS** D

**WHERE** E.Fname=D.Dependent\_name

**AND** E.Sex=D.Sex );

Query 17: Retrieve the name employees working in ‘Research’ department.

SELECT E.Fname, E.Lname FROM Employee E, Department D

WHERE D.Dname <> 'Research' AND D.Dnumber = E.dno;

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname = 'Research');

SELECT E.Fname, E.Lname FROM Employee E

WHERE E.Dno NOT IN (SELECT D.Dnumber FROM DEPARTMENT D WHERE D.Dname = 'Research');