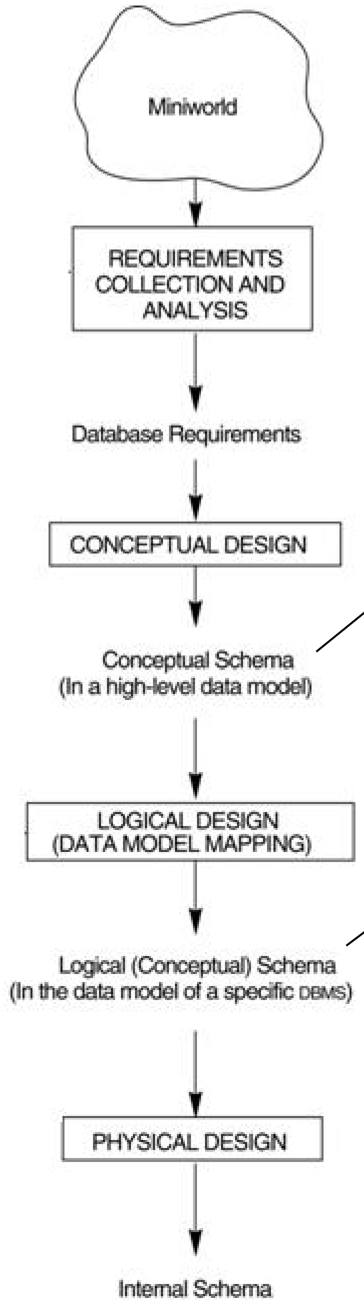
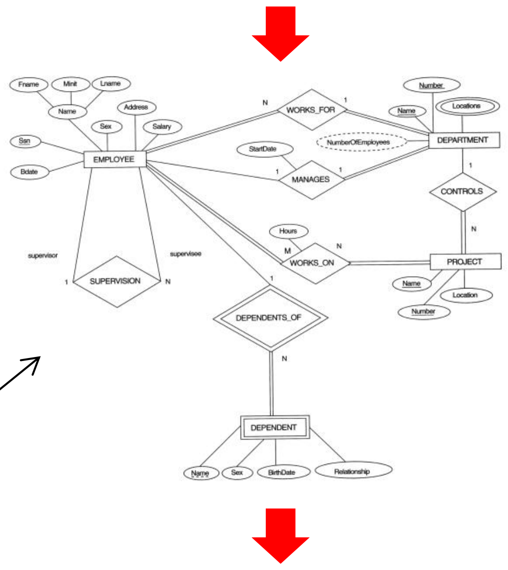
Chapter 4

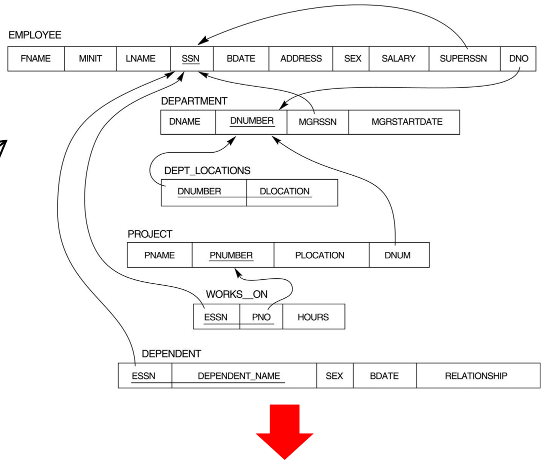
SQL: Data Definition, Constraints, and Basic Queries and Updates

Main phases of database system design


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Database Requirements
CONCEPTUAL DESIGN
Conceptual Schema (In a high-level data model)
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
PHYSICAL DESIGN
Internal Schema

Database Requirements

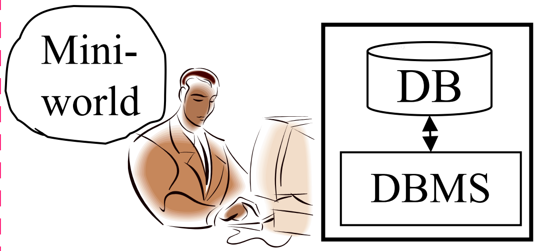




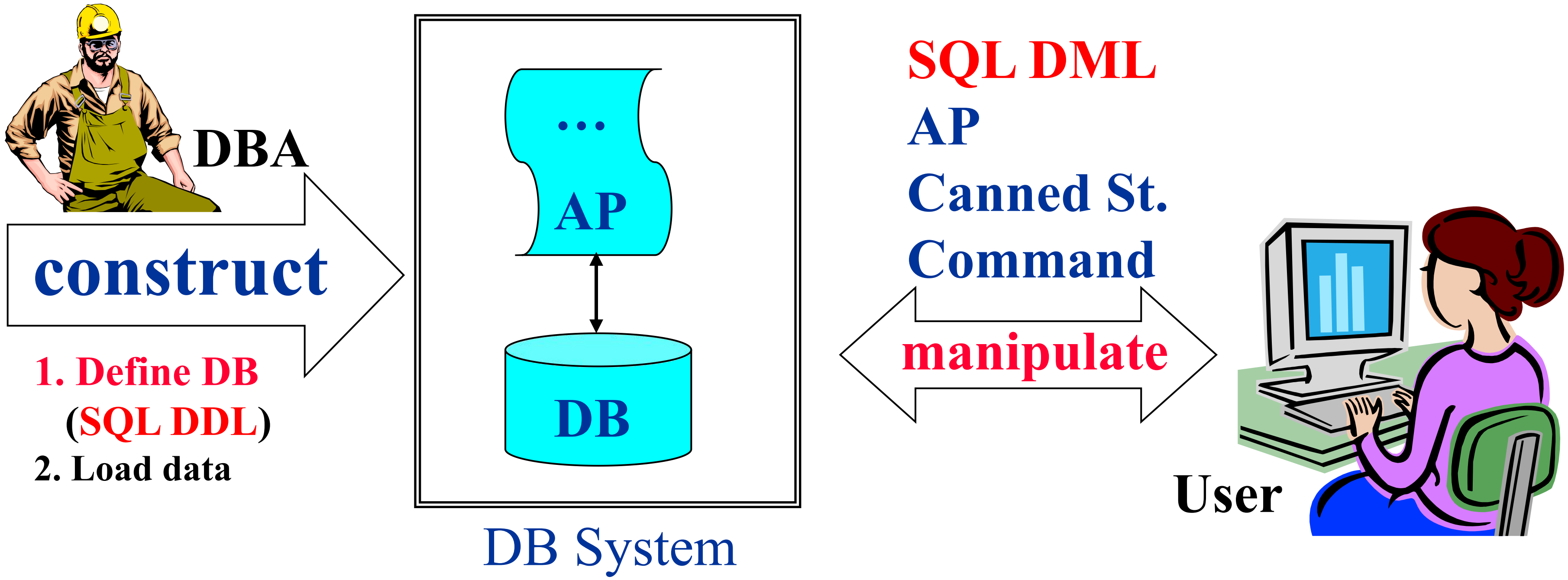
```
CREATE TABLE  DEPARTMENT (
  DNAME          VARCHAR(10)  NOT NULL,
  DNUMBER        INTEGER      NOT NULL,
  MGRSSN         CHAR(9),
  MGRSTARTDATE   CHAR(9)  );
```

Construct DB

1. **Design DB**
2. Define DB
3. Load DB



Construction and Operation



SQL (Ch.4, 5)

- **Data Definition Language**
  - **CREATE, DROP, ALTER**

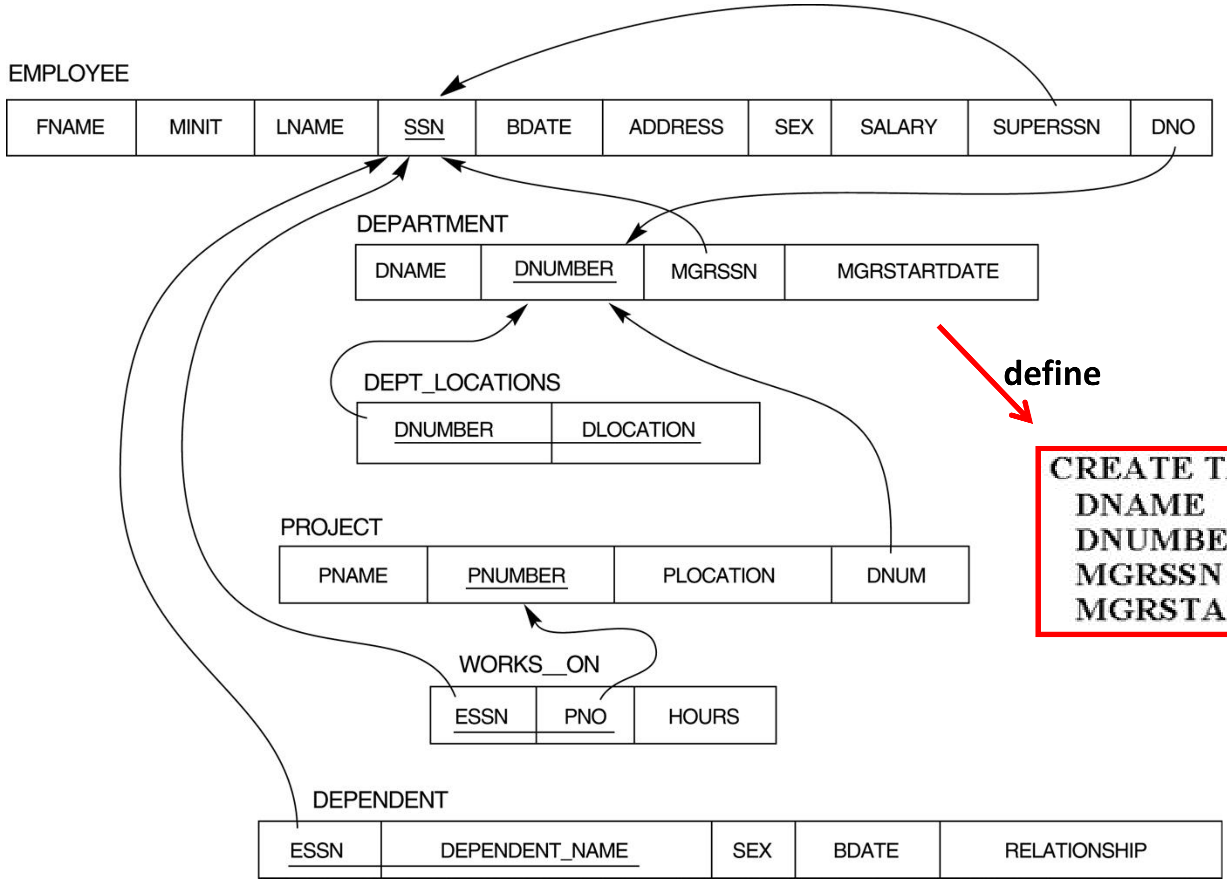
SQL (Ch.4, 5)

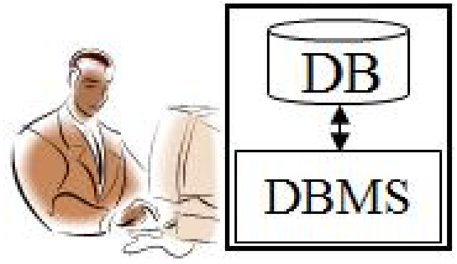
- **Data Manipulation Language**
  - **Query: SELECT**
  - **Update: INSERT, DELETE, UPDATE**

Chapter Outline

Define Database Schema

CREATE SCHEMA COMPANY **AUTHORIZATION** ‘JSmith’;





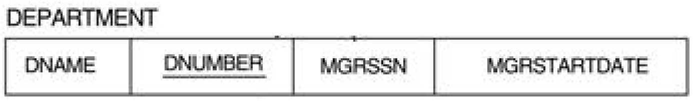
define

```
CREATE TABLE  DEPARTMENT (
  DNAME              VARCHAR(10)   NOT NULL,
  DNUMBER            INTEGER       NOT NULL,
  MGRSSN             CHAR(9),
  MGRSTARTDATE       CHAR(9)  );
```

CREATE TABLE

- Specifies a new base relation by
  - giving it a name, and
  - specifying each of its attributes and their data types (e.g., INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE   DEPARTMENT (
        DNAME            VARCHAR(10)       NOT NULL,
        DNUMBER          INT               NOT NULL,
        MGRSSN           CHAR(9),
        MGRSTARTDATE     CHAR(9)  );
```

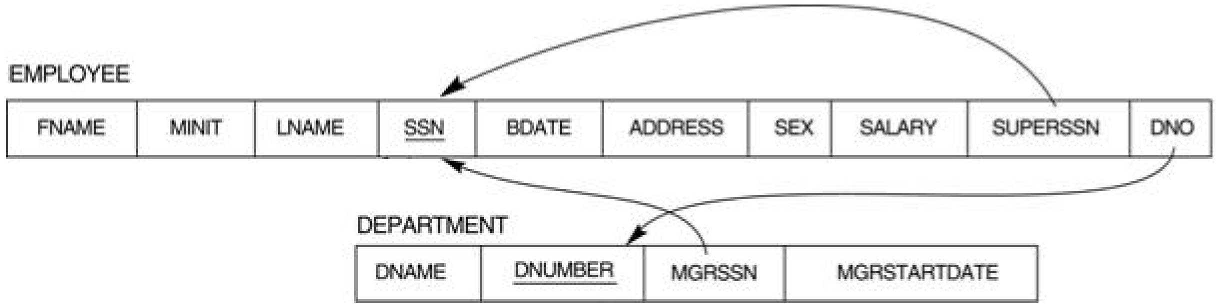


CREATE TABLE

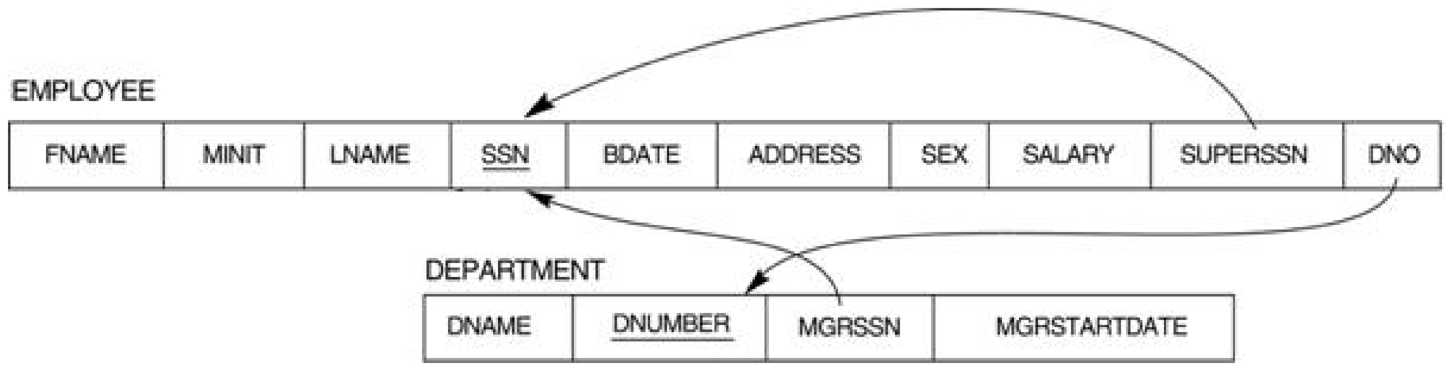
- In SQL2, can use the CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys).
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE   DEPARTMENT
  (  DNAME            VARCHAR(10)      NOT NULL,
     DNUMBER          INT              NOT NULL,
     MGRSSN           CHAR(9),
     MGRSTARTDATE     CHAR(9),

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



```
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) ,
          SUPERSSN              CHAR(9) ,
          DNO                   INT                   NOT NULL ,
   PRIMARY KEY (SSN) ,
   FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ,
   FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER) ) ;
```



```
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) ) ;

CREATE TABLE DEPT_LOCATIONS
        ( DNUMBER            INT               NOT NULL ,
          DLOCATION          VARCHAR(15)       NOT NULL ,
        PRIMARY KEY (DNUMBER, DLOCATION) ,
        FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT(DNUMBER) ) ;
```

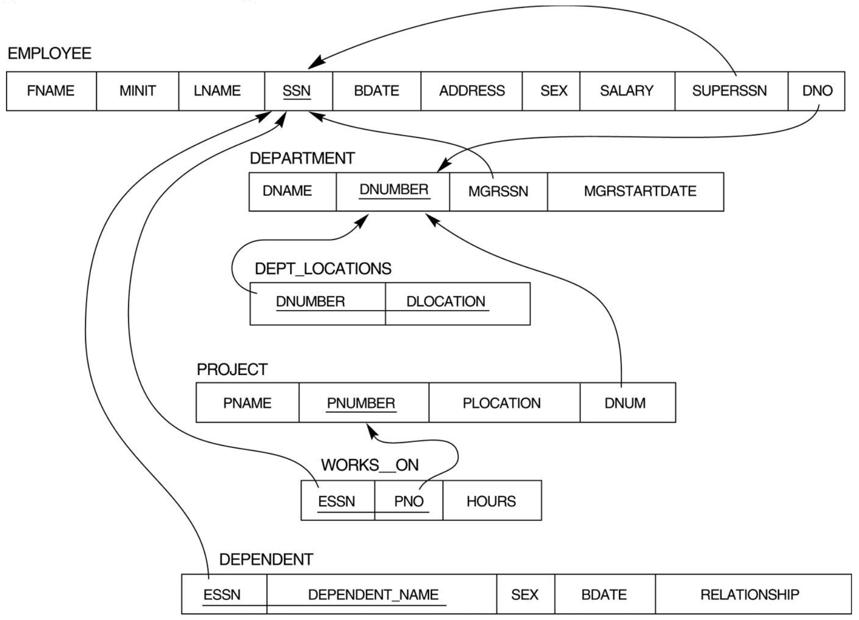


FIGURE (continued)

```
CREATE TABLE PROJECT
        ( PNAME                 VARCHAR(15)         NOT NULL ,
          PNUMBER               INT                 NOT NULL ,
          PLOCATION             VARCHAR(15) ,
          DNUM                  INT                 NOT NULL ,
        PRIMARY KEY (PNUMBER) ,
        UNIQUE (PNAME) ,
        FOREIGN KEY (DNUM) REFERENCES DEPARTMENT(DNUMBER) ) ;
CREATE TABLE WORKS_ON
        ( ESSN                  CHAR(9)             NOT NULL ,
          PNO                   INT                 NOT NULL ,
          HOURS                 DECIMAL(3,1)        NOT NULL ,
        PRIMARY KEY (ESSN, PNO) ,
        FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ,
        FOREIGN KEY (PNO) REFERENCES PROJECT(PNUMBER) ) ;
CREATE TABLE DEPENDENT
        ( ESSN                     CHAR(9)          NOT NULL ,
          DEPENDENT_NAME           VARCHAR(15)      NOT NULL ,
          SEX                      CHAR ,
          BDATE                    DATE ,
          RELATIONSHIP             VARCHAR(8) ,
      PRIMARY KEY (ESSN, DEPENDENT_NAME) ,
      FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ) ;
```

Additional Data Types in SQL2 and SQL-99

Has DATE, TIME, and TIMESTAMP data types

- **DATE:**
  - Made up of year-month-day in the format **yyyy-mm-dd**
- **TIME:**
  - Made up of hour:minute:second in the format **hh:mm:ss**
- **TIME(i):**
  - Made up of hour:minute:second plus i additional digits specifying fractions of a second
  - format is hh:mm:ss:ii...i
- **TIMESTAMP:**
  - Has both DATE and TIME components
- **INTERVAL:**
  - Specifies a relative value rather than an absolute value
  - Can be DAY/TIME intervals or YEAR/MONTH intervals
  - Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

Specifying Constraints in SQL

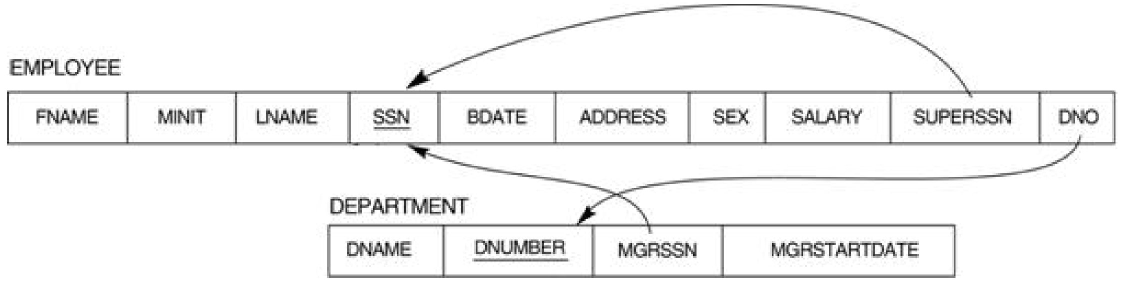
- **Specifying attribute constraints**

```
DNUMBER   INT     NOT NULL   CHECK (DNUMBER > 0 AND DNUMBER < 21)
```

- **REFERENTIAL INTEGRITY OPTIONS**
  - can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE   EMPLOYEE
        (ENAME              VARCHAR(30)       NOT NULL,
        ESSN                CHAR(9),
        BDATE               DATE,
        DNO                 INT               DEFAULT 1,
        SUPERSSN            CHAR(9),

        PRIMARY KEY (ESSN),
        FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)
                ON DELETE SET DEFAULT ON UPDATE CASCADE,
        FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (ESSN)
                ON DELETE SET NULL ON UPDATE CASCADE);
```



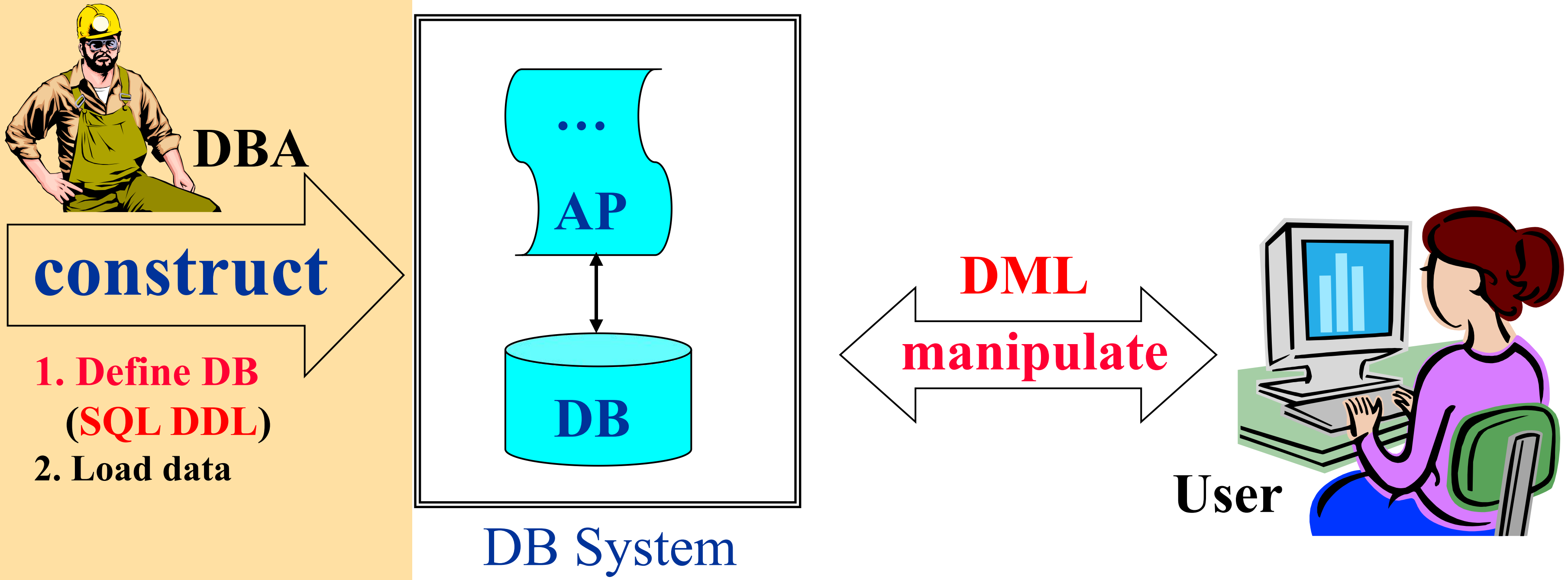
Default attribute values and referential triggered actions can be specified in SQL.

```
CREATE TABLE EMPLOYEE
        ( . . . ,
          DNO              INT    NOT NULL    DEFAULT 1,
        CONSTRAINT EMPPK
          PRIMARY KEY (SSN) ,
        CONSTRAINT EMPSUPERFK
          FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
                         ON DELETE SET NULL    ON UPDATE CASCADE ,
        CONSTRAINT EMPDEPTFK
          FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER)
                         ON DELETE SET DEFAULT    ON UPDATE CASCADE );

CREATE TABLE DEPARTMENT
        ( . . . ,
          MGRSSN   CHAR(9)  NOT NULL DEFAULT  '888665555' ,
          . . . ,
          CONSTRAINT DEPTPK
            PRIMARY KEY (DNUMBER) ,
          CONSTRAINT DEPTSK
            UNIQUE (DNAME),
          CONSTRAINT DEPTMGRFK
            FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
                 ON DELETE SET DEFAULT    ON UPDATE CASCADE );

CREATE TABLE DEPT_LOCATIONS
        ( . . . ,
          PRIMARY KEY (DNUMBER, DLOCATION),
          FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT(DNUMBER)
            ON DELETE CASCADE    ON UPDATE CASCADE ) ;
```

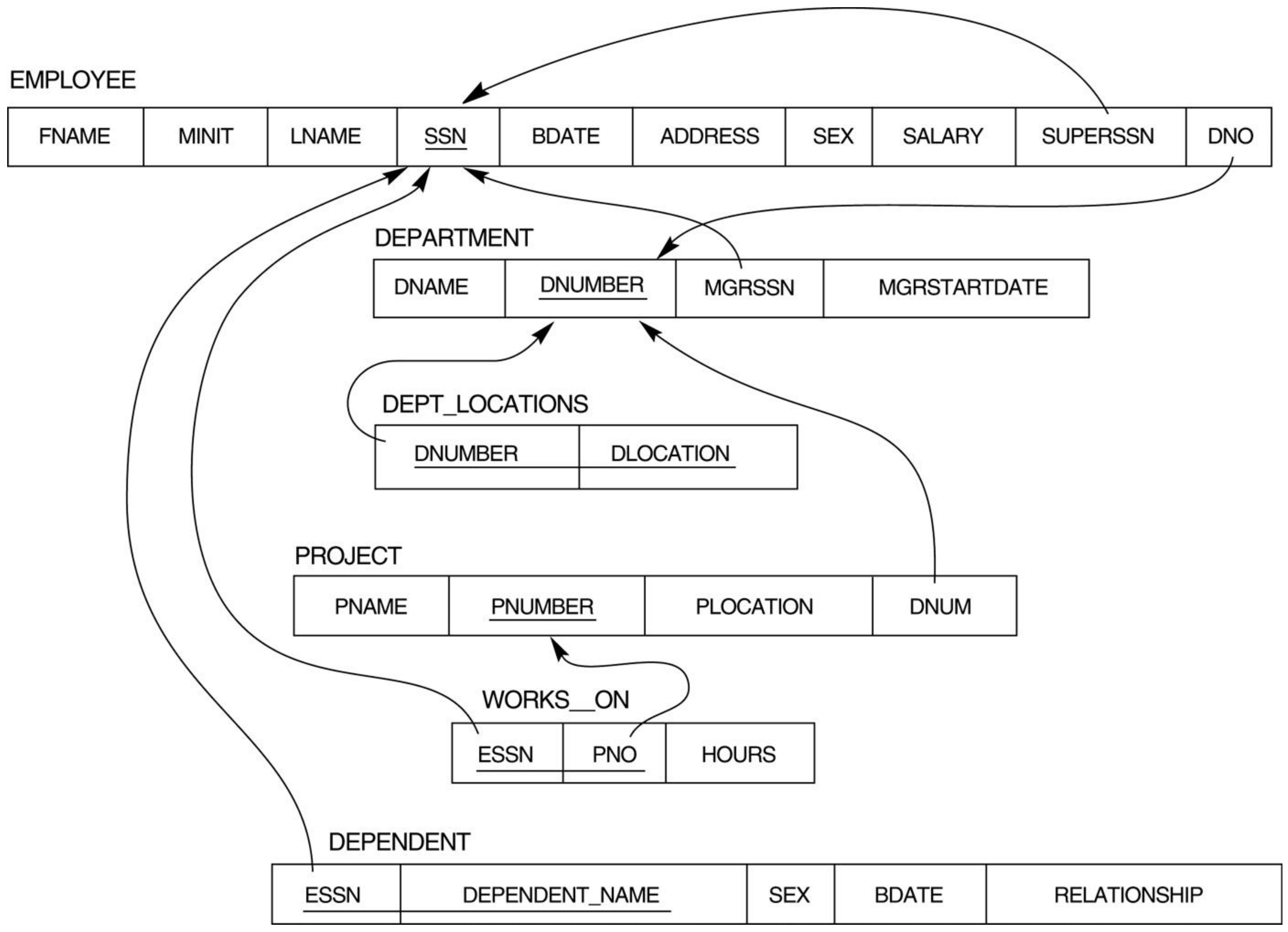
Basic Retrieval Queries in SQL



SQL (Ch.4, 5)
- **Data Manipulation Language**
  - **Query: SELECT**
  - **Update: INSERT, DELETE, UPDATE**

SQL (Ch.4, 5)
- **Data Definition Language**
  - **CREATE, DROP, ALTER**

Relational Database Schema: Company



How to retrieve data from the populated database?

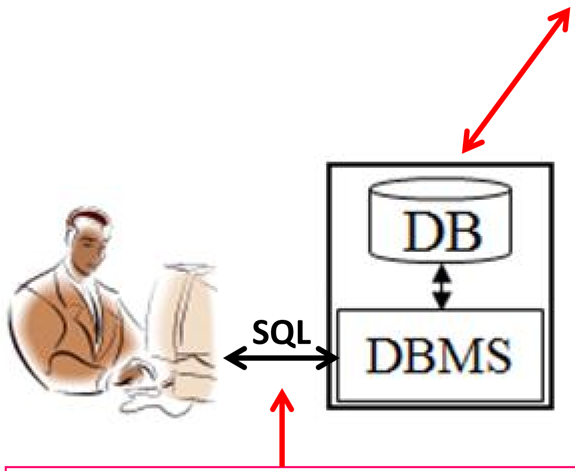
EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4



DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

```
SELECT   BDATE, ADDRESS
FROM     EMPLOYEE
WHERE    FNAME='John' AND MINIT='B’
         AND LNAME='Smith’
```

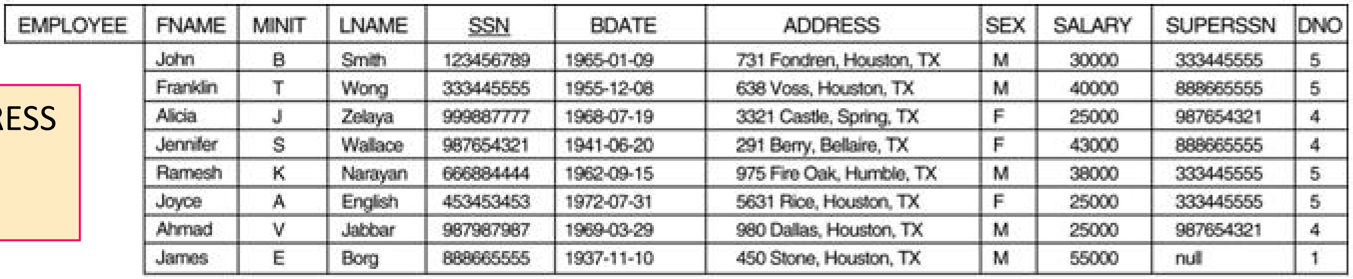
Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a *mapping* or a *SELECT-FROM-WHERE block*

```
SELECT      <attribute list>
FROM        <table list>
WHERE       <condition>
```

  - **<attribute list>** is a list of attribute names whose values are to be retrieved by the query
  - **<table list>** is a list of the relation names required to process the query
  - **<condition>** is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

```
SELECT   FNAME, LNAME, ADDRESS
FROM     EMPLOYEE
WHERE    DNO = 5
```



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

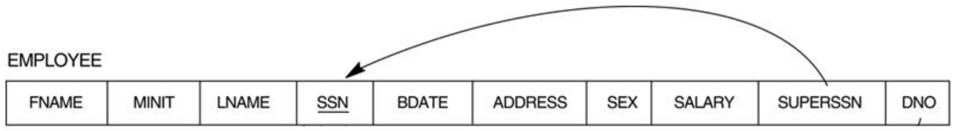
Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra
- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
Q0: SELECT   BDATE, ADDRESS
    FROM     EMPLOYEE
    WHERE    FNAME='John' AND MINIT='B' AND LNAME='Smith'
```

  – Similar to a SELECT-PROJECT pair of relational algebra operations; the SELECT-clause specifies the *projection attributes* and the WHERE-clause specifies the *selection condition*
  – However, the result of the query *may contain* duplicate tuples

BDATE	ADDRESS
1965-01-09	731 Fondren, Houston, TX



SQL Queries with Join

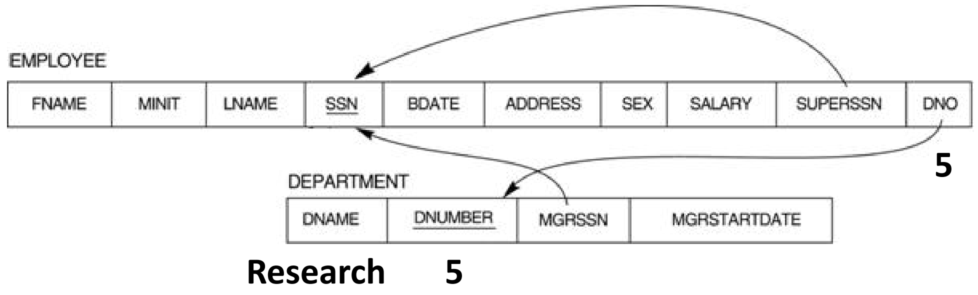
- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1: SELECT   FNAME, LNAME, ADDRESS (姓名 地址)
    FROM     EMPLOYEE, DEPARTMENT
    WHERE    DNAME='Research' AND DNUMBER=DNO(找研究部門)
```

透過DNO找到DNAME

– (DNAME = 'Research') is a *selection condition*
– (DNUMBER = DNO) is a *join condition (牽涉join速度較慢)*

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX



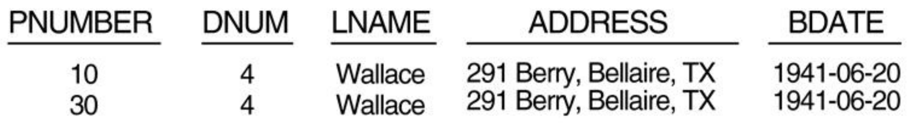
SQL Queries with Two Join

- Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

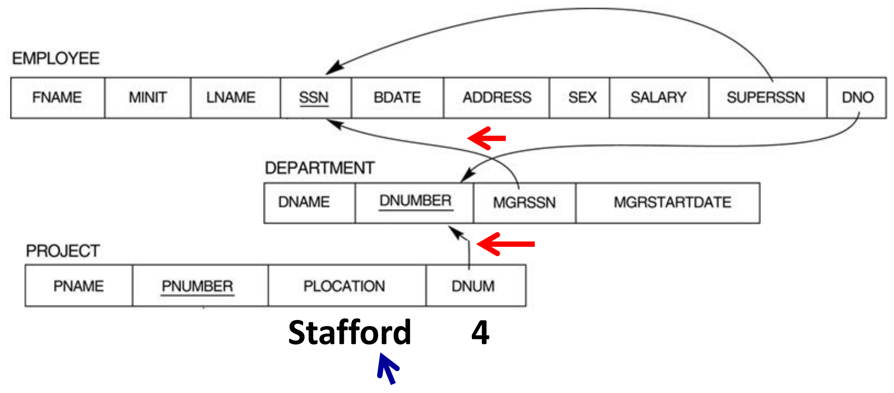
負責的主管的 名字生日地址

```
Q2:  SELECT   PNUMBER, DNUM, LNAME, BDATE, ADDRESS
     FROM     PROJECT, DEPARTMENT, EMPLOYEE
     WHERE    DNUM=DNUMBER AND MGRSSN=SSN AND
              PLOCATION='Stafford' (專案在這做)
```

– In Q2, there are **two** join conditions
– The join condition DNUM=DNUMBER relates a project to its controlling department
– The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department



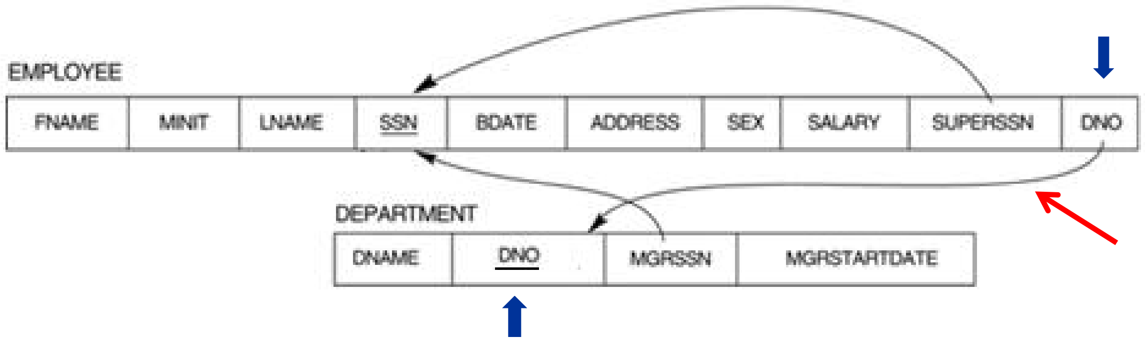
PNUMBER	DNUM	LNAME	ADDRESS	BDATE
10	4	Wallace	291 Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291 Berry, Bellaire, TX	1941-06-20



Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in ***different relations***
- A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name

```
SELECT    FNAME, LNAME, ADDRESS, DNAME
FROM      EMPLOYEE, DEPARTMENT
WHERE     EMPLOYEE.DNO = DEPARTMENT.DNO
```



ALIASES

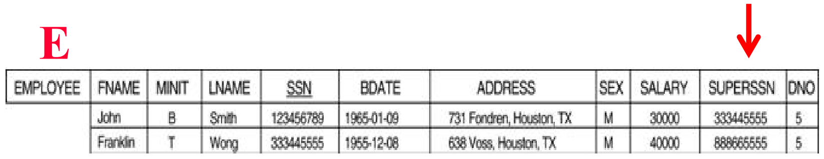
- Some queries need to refer to the same relation twice
- In this case, ***aliases*** are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
Q8:   SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM     EMPLOYEE  E   S  (兩個TABLE都是EMPLOYEE 要相等)
      WHERE    E.SUPERSSN=S.SSN (主管的SSN要相等)
```

  - In Q8, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
  - We can think of E and S as two *different copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*
  - Aliasing can also be used in any SQL query for convenience
    Can also use the AS keyword to specify aliases

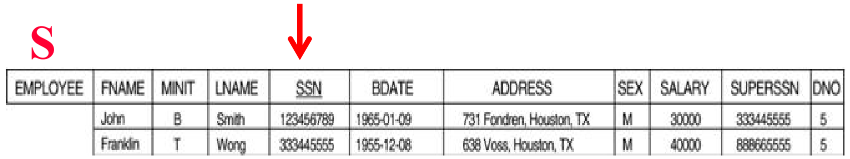
```
Q8:   SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM     EMPLOYEE AS E, EMPLOYEE AS S
      WHERE    E.SUPERSSN=S.SSN
```

E



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

S



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

(a)

BDATE	ADDRESS
1965-01-09	731 Fondren, Houston, TX

(b)

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

(c)

PNUMBER	DNUM	LNAME	ADDRESS	BDATE
10	4	Wallace	291 Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291 Berry, Bellaire, TX	1941-06-20

(d)

E.FNAME	E.LNAME	S.FNAME	S.LNAME
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

(e)

SSN
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

(f)

SSN	DNAME
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

(g)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

FIGURE Results of SQL queries when applied to the COMPANY database
(a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

UNSPECIFIED WHERE-clause

- A ***missing WHERE-clause*** indicates no condition; hence, ***all tuples*** of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

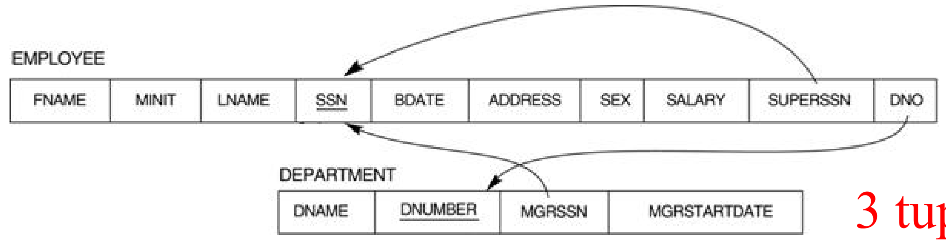
```
Q9:  SELECT  SSN
     FROM    EMPLOYEE
```

SSN
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the ***CARTESIAN PRODUCT*** of tuples is selected

```
Q10:  SELECT  SSN, DNAME  (員工編號 對 部門名稱)
      FROM    EMPLOYEE, DEPARTMENT
```

– It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result



8 tuples (一筆資料對上3筆)

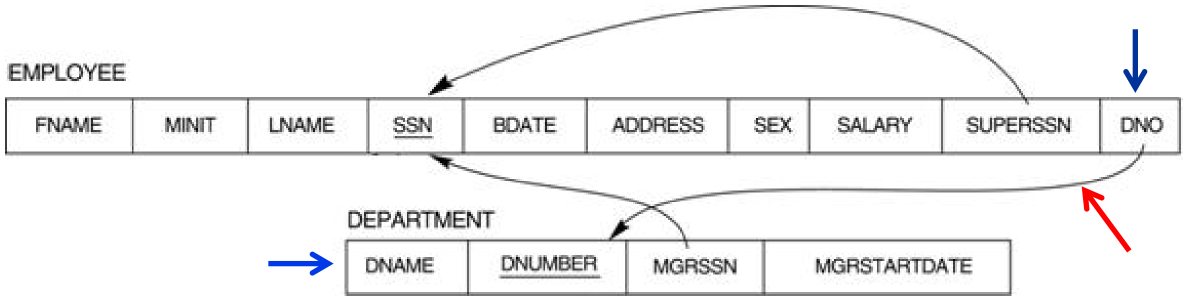
3 tuples (所以一共24筆)

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for ***all the attributes (*表示全選)***
  Examples:

```
Q1C:   SELECT   *
       FROM     EMPLOYEE
       WHERE    DNO=5

Q1D:   SELECT   *
       FROM     EMPLOYEE, DEPARTMENT
       WHERE    DNAME='Research' AND DNO=DNUMBER
```



USE OF DISTINCT

- SQL does ***not*** treat a relation as a set; ***duplicate*** *tuples can appear*
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:   SELECT   SALARY
       FROM     EMPLOYEE

Q11A:  SELECT   DISTINCT SALARY
       FROM     EMPLOYEE
```

(a)

SALARY
30000
40000
25000
43000
38000
25000
25000
55000

(b)

SALARY
30000
40000
25000
43000
38000
55000

SET OPERATIONS

- SQL has directly incorporated some set operations (集合運算)
- There is a union operation (**UNION**), and in *some versions* of SQL there are set difference (**MINUS**) and intersection (**INTERSECT**) operations
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to ***union compatible relations*****:**

1. The two relations must have **the same attributes.**
2. The attributes must appear in **the same order.**

R

A1	A2	A3	A4

S

A1	A2	A3	A4

R **UNION** S ?
R(A1, A3) **MINUS** S(A1, A3) ?
R(A2, A4) **INTERSECT** S(A4) ?

FIGURE
The results of SQL multiset operations.

(a) Two tables, R(A) and S(A).

(a)

R	A
	a1
	a2
	a2
	a3

S	A
	a1
	a2
	a4
	a5

R(A) **UNION ALL** S(A)

(b)

T	A
	a1
	a1
	a2
	a2
	a2
	a3
	a4
	a5

(c)

T	A
	a2
	a3

R(A) **EXCEPT ALL** S(A)

R(A) **INTERSECT ALL** S(A)

(d)

T	A
	a1
	a2

union compatible relations:
1. The two relations must have the **same attributes.**
2. The attributes must appear in the **same order.**

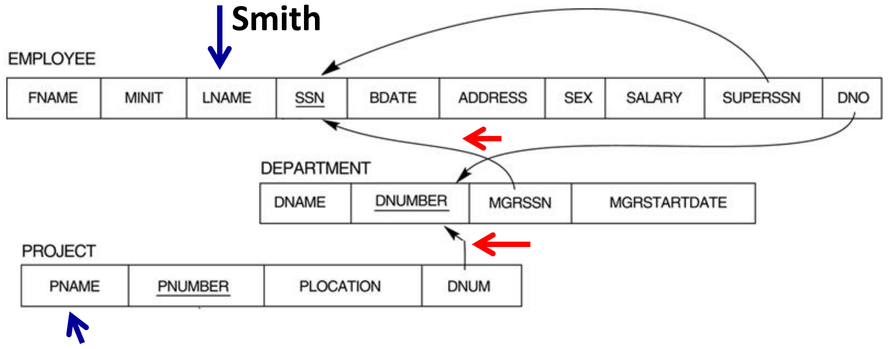
Use the keyword ALL for multiset operations, which will preserve duplicate tuples.

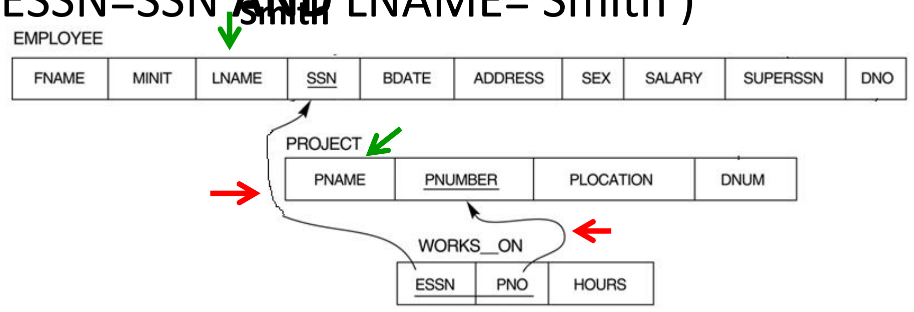
SET OPERATIONS (cont.)

- Query 4: Make a list of all project numbers for projects **that involve an employee whose last name is 'Smith' as a worker** or **as a manager of the department that controls the project**.

Union compatible?

```
Q4:  (SELECT   PNAME
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    DNUM=DNUMBER AND MGRSSN=SSN  AND
LNAME='Smith')
      UNION
     (SELECT   PNAME
      FROM     PROJECT, WORKS_ON, EMPLOYEE
      WHERE    PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith')
```

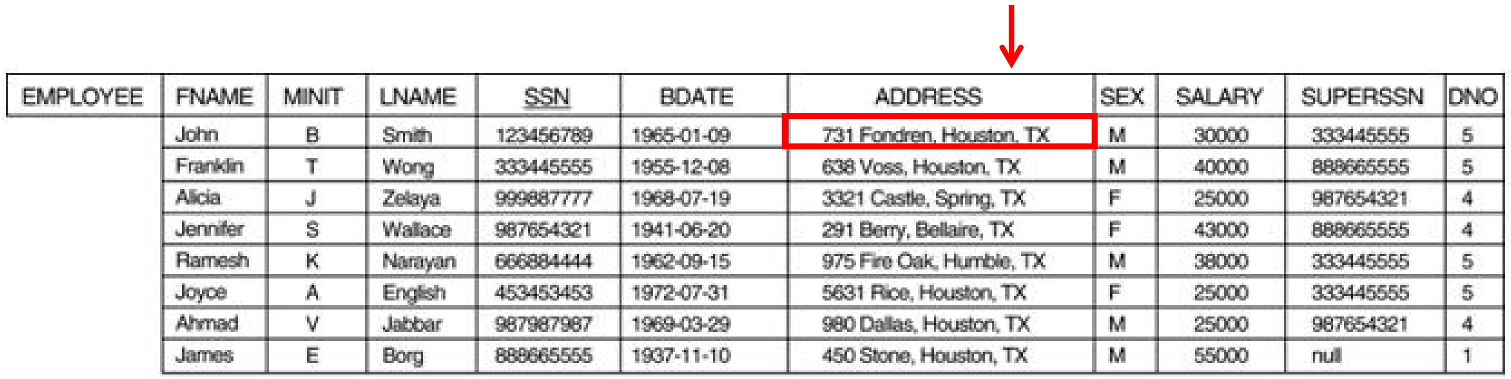




SUBSTRING COMPARISON

- The **LIKE** comparison operator is used to compare partial strings. Two reserved characters are used:
  - '%' (or '*' in some implementations) replaces an arbitrary number of characters, and
  - '_' replaces a single arbitrary character
- Query 12: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX'.

```
Q12: SELECT  FNAME, LNAME
     FROM    EMPLOYEE
     WHERE   ADDRESS LIKE '%Houston, TX%' (TX是地名)
```



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

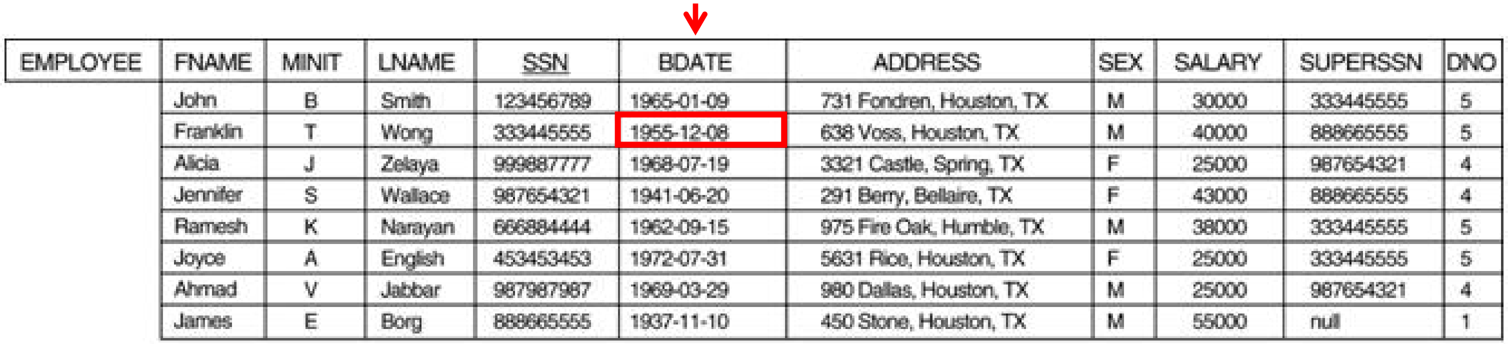
SUBSTRING COMPARISON

- Query 12A: Retrieve all employees who were born during the 1950s. Here, '5' must be the 3th character of the string (according to our format for date), so the BDATE value is '_ _ 5_ _ _ _ _ _ _ _', with each underscore as a place holder for a single arbitrary character.

```
Q12A:  SELECT   FNAME, LNAME
       FROM     EMPLOYEE
       WHERE    BDATE LIKE '_ _5_ _ _ _ _ _ _'
```

BDATE
1955-12-08

- The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible; hence, in SQL, character string attribute values are not atomic

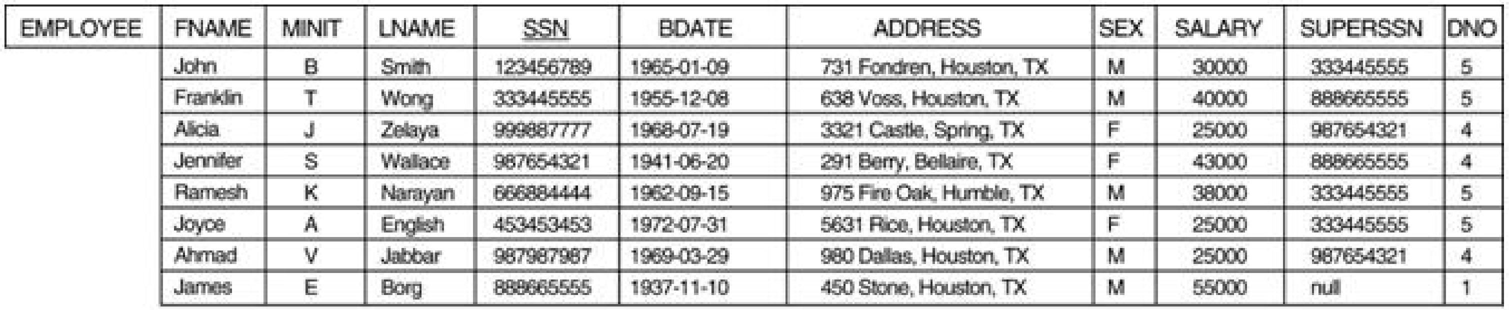


EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-'. '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query 13: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
Q13: SELECT   FNAME, LNAME, 1.1*SALARY (顯示加薪，但selec不動到db)
     FROM     EMPLOYEE, WORKS_ON, PROJECT (實質上沒有加薪)
     WHERE    SSN=ESSN AND PNO=PNUMBER AND
              PNAME='ProductX'
```



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

ORDER BY 排序

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)

  Query 15: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

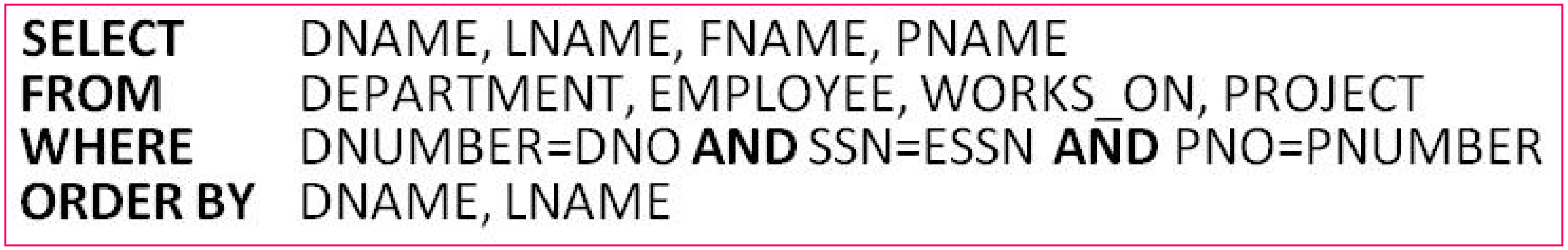
```
Q15:  SELECT     DNAME, LNAME, FNAME, PNAME
      FROM       DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
      WHERE      DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
      ORDER BY   DNAME, LNAME(先排部門名稱 由小到大)
```

- The default order is in ascending(小到大) order of values
- Keyword **DESC** if we want a descending order;
- Keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

Summary of Basic SQL Retrieval Queries

- A basic retrieval query in SQL:

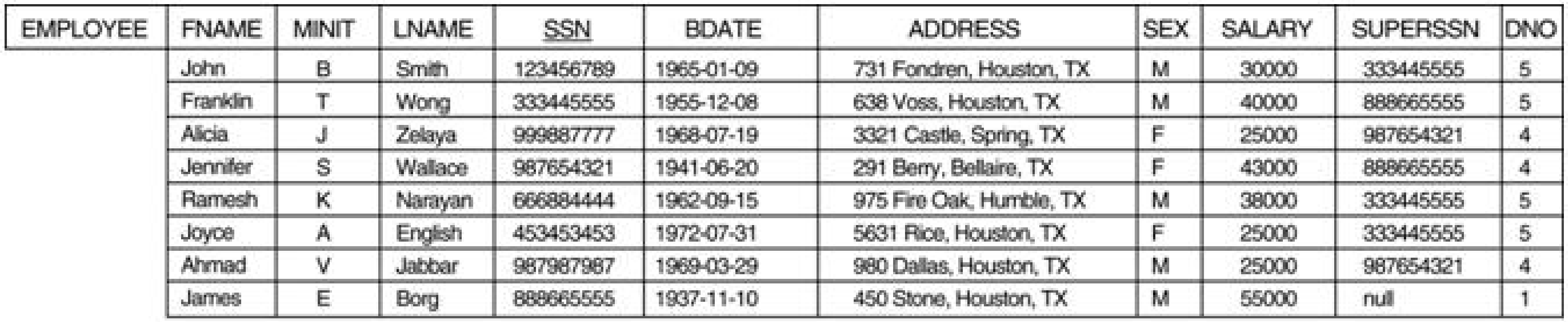
```
SELECT       <attribute list>
FROM         <table list>
[ WHERE      <condition> ]
[ ORDER BY   <attribute list> ]
```



```
SELECT      DNAME, LNAME, FNAME, PNAME
FROM        DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE       DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
ORDER BY    DNAME, LNAME
```

Specifying Updates in SQL

- There are three SQL commands to modify the database;
  - INSERT
  - DELETE
  - UPDATE



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

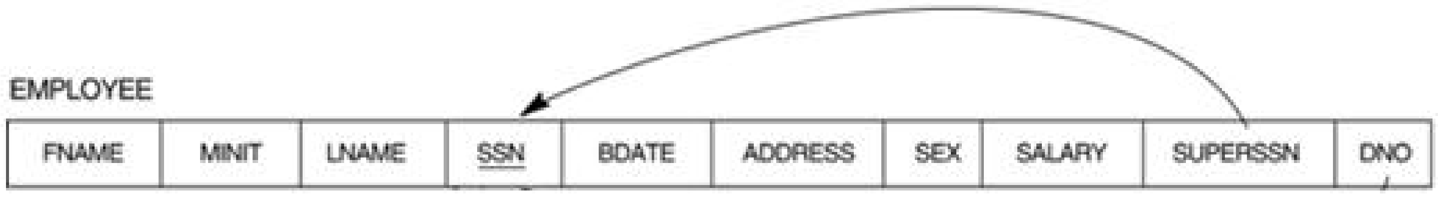
INSERT

- Used to add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE command
- Example:

```
U1:   INSERT INTO  EMPLOYEE
      VALUES       ('Richard','K','Marini', '653298653', '30-DEC-52',
                    '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4 )
```

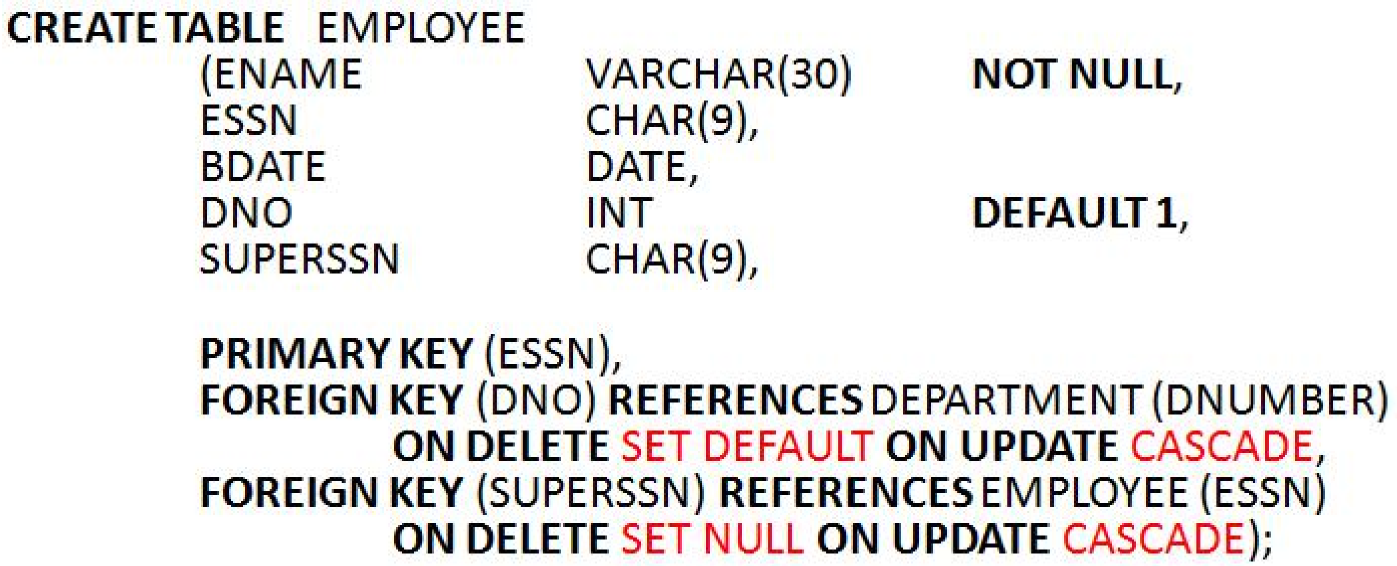
- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple
- Attributes with NULL values can be left out
- Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
U1A:   INSERT INTO   EMPLOYEE (FNAME, LNAME, SSN)
       VALUES        ('Richard', 'Marini', '653298653')
```



INSERT of Multiple Tuples

- Important Note:
  - Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database



```
CREATE TABLE   EMPLOYEE
          (ENAME              VARCHAR(30)      NOT NULL,
          ESSN                CHAR(9),
          BDATE               DATE,
          DNO                 INT              DEFAULT 1,
          SUPERSSN            CHAR(9),

          PRIMARY KEY (ESSN),
          FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)
                    ON DELETE SET DEFAULT ON UPDATE CASCADE,
          FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (ESSN)
                    ON DELETE SET NULL ON UPDATE CASCADE);
```

INSERT of Multiple Tuples

- Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

- Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

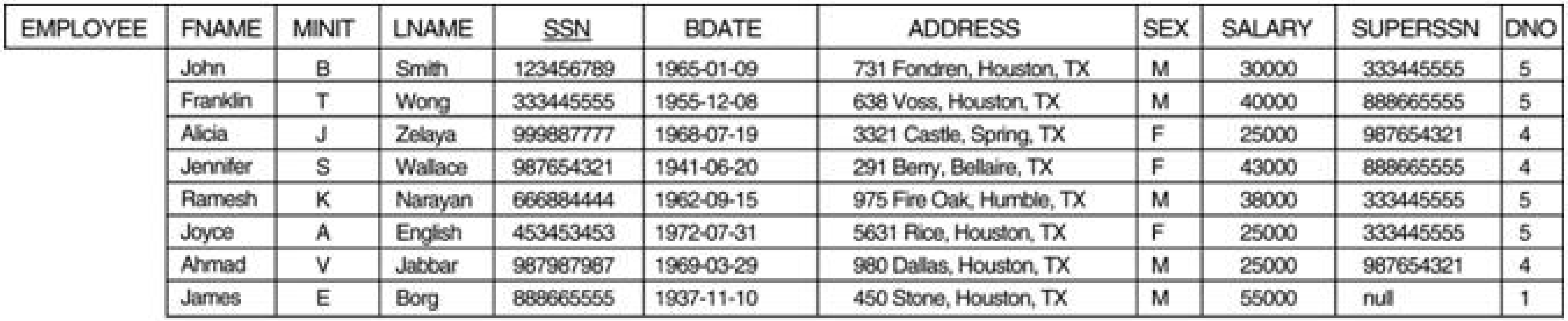
```
U3A:   CREATE TABLE   DEPTS_INFO
                      (DEPT_NAME     VARCHAR(10),
                       NO_OF_EMPS    INTEGER,
                       TOTAL_SAL     INTEGER);

U3B:   INSERT INTO    DEPTS_INFO (DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)
                      SELECT         DNAME, COUNT (*), SUM (SALARY)
                      FROM           DEPARTMENT, EMPLOYEE
                      WHERE          DNUMBER=DNO
                      GROUP BY       DNAME ;
```

DELETE

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause
- Referential integrity should be enforced

```
U4A:    DELETE FROM    EMPLOYEE
        WHERE          LNAME='Brown'
```



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

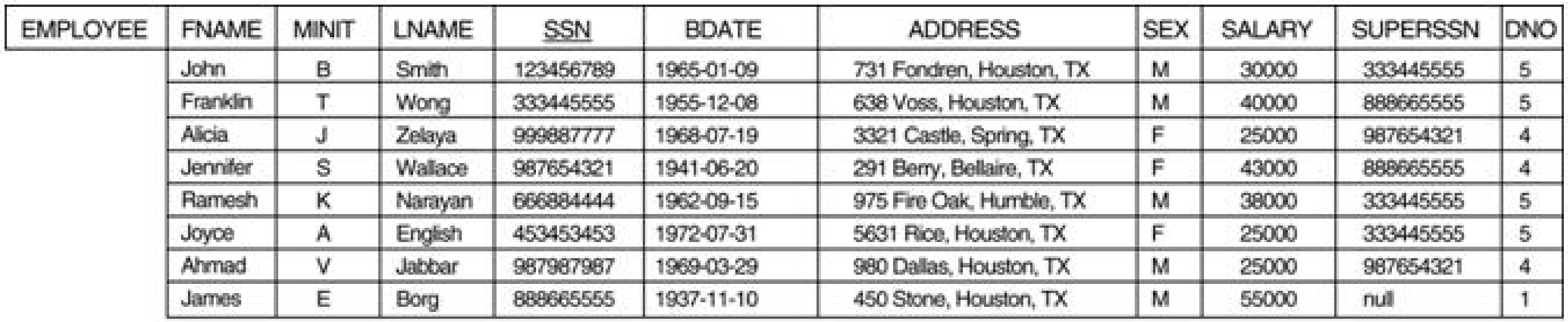
DELETE (cont.)

```
U4B:    DELETE FROM   EMPLOYEE
        WHERE         SSN='123456789’

U4C:    DELETE FROM   EMPLOYEE
        WHERE         DNO  = 5

U4D:    DELETE FROM   EMPLOYEE
```

A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table

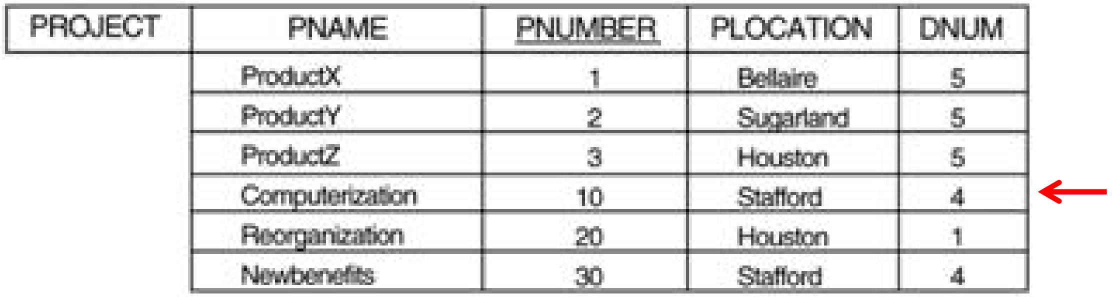


EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples ***in the same relation***
- Referential integrity should be enforced
- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
U5:   UPDATE   PROJECT
      SET      PLOCATION = 'Bellaire', DNUM = 5
      WHERE    PNUMBER=10
```



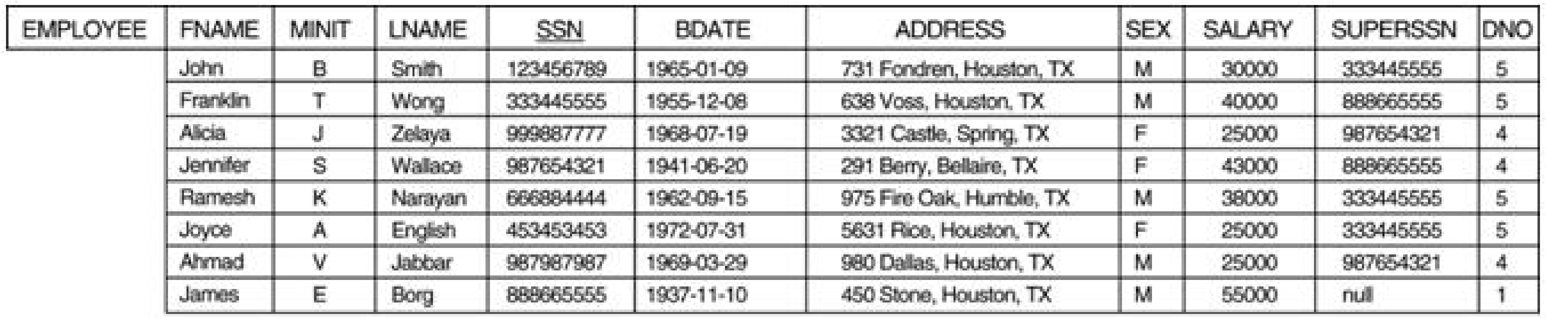
PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

UPDATE (cont.)

- Example: Give all employees in department 5 a 10% raise in salary.

```
U6:  UPDATE  EMPLOYEE
     SET     SALARY = SALARY *1.1
     WHERE   DNO  = 5
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1