Database Systems

Lecture #08

Sang-Wook Kim

Hanyang University

Objectives

- To learn basic concepts of SQL
  - Data definition for SQL
  - Data retrieval queries

Outline

- Introduction to SQL
- Example Database
- Schema and Table Creation
- SELECT Query Basics
- Tables as Sets

Introduction to SQL

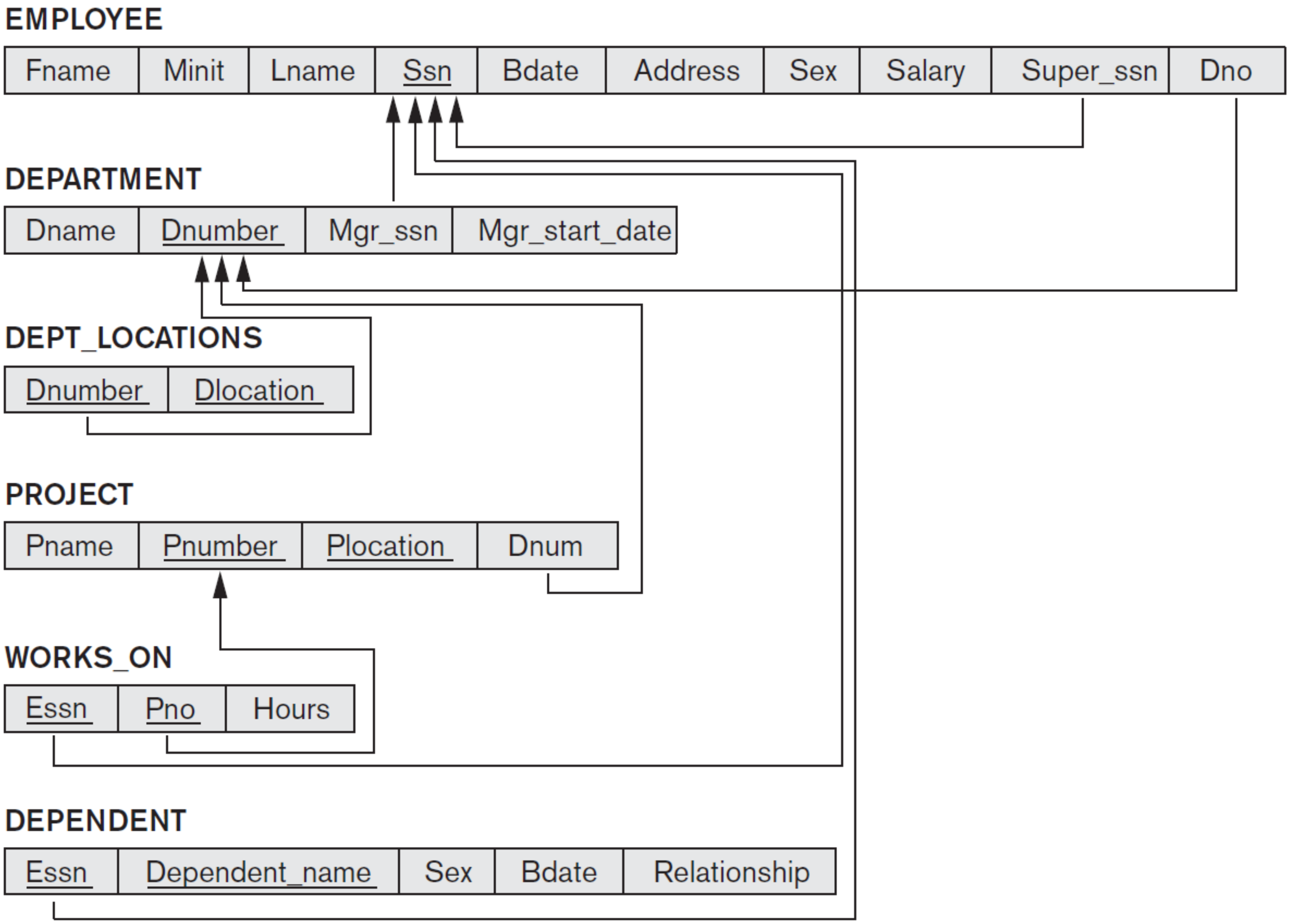
- History
  - Initially developed by IBM Research
    - Special-purpose *declarative* programming language
    - Designed for IBM's first DBMS, System R
    - Originally named SEQUEL (Structured English Query Language)
    - Renamed as SQL (Structured Query Language)
  - Standardized by ANSI
  - Most popular language for DBMS today

Introduction to SQL

◆ Components

- Data Definition Language (DDL) -> manipulate schema
  - Provides commands for databases, tables, and indexes
    - Create, modify, delete
- Data Manipulation Language (DML) -> manipulate instance
  - Provides commands for tuples
    - Retrieval, insert, modify, delete

Database Example: COMPANY



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

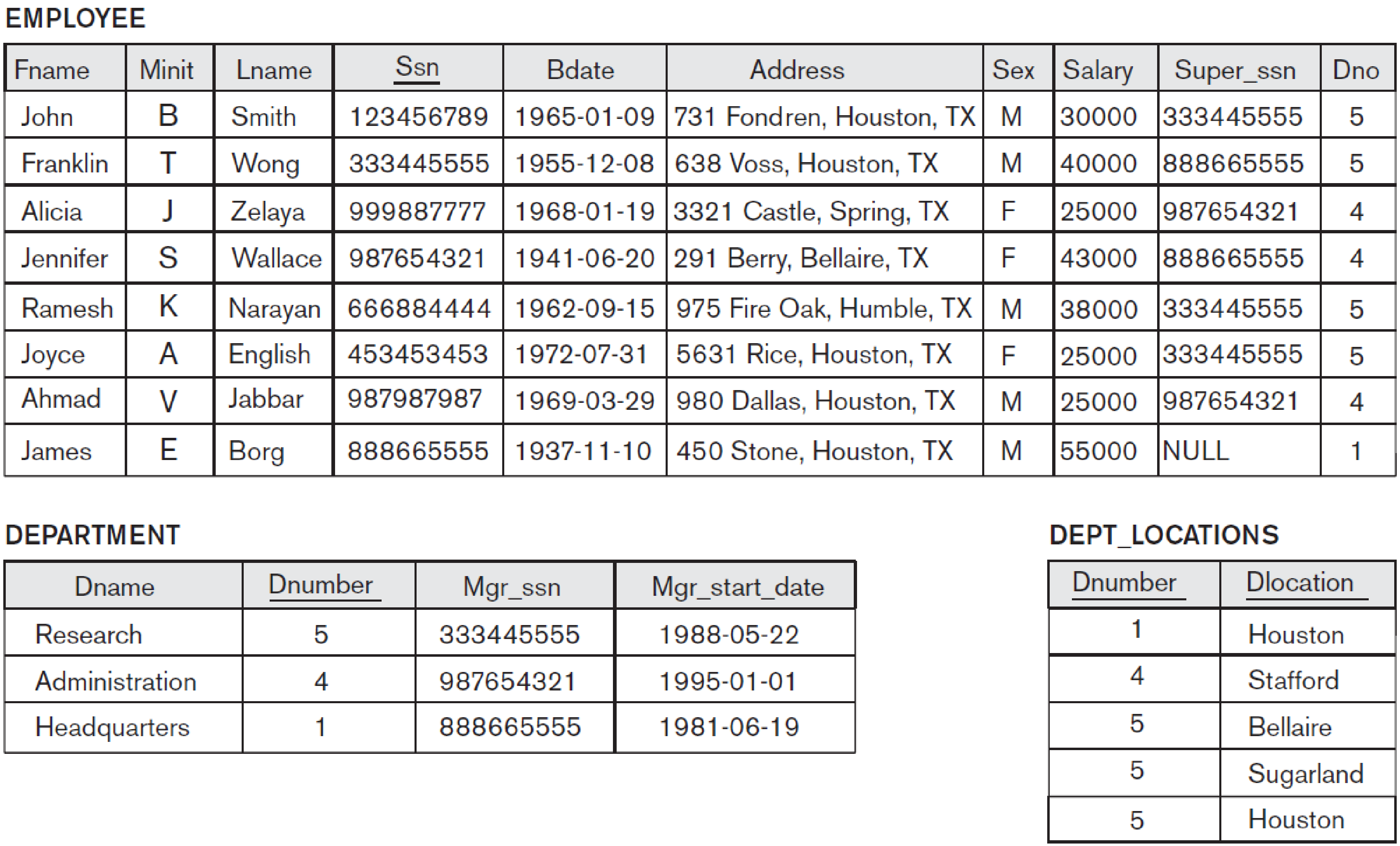
WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Database Example: COMPANY



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

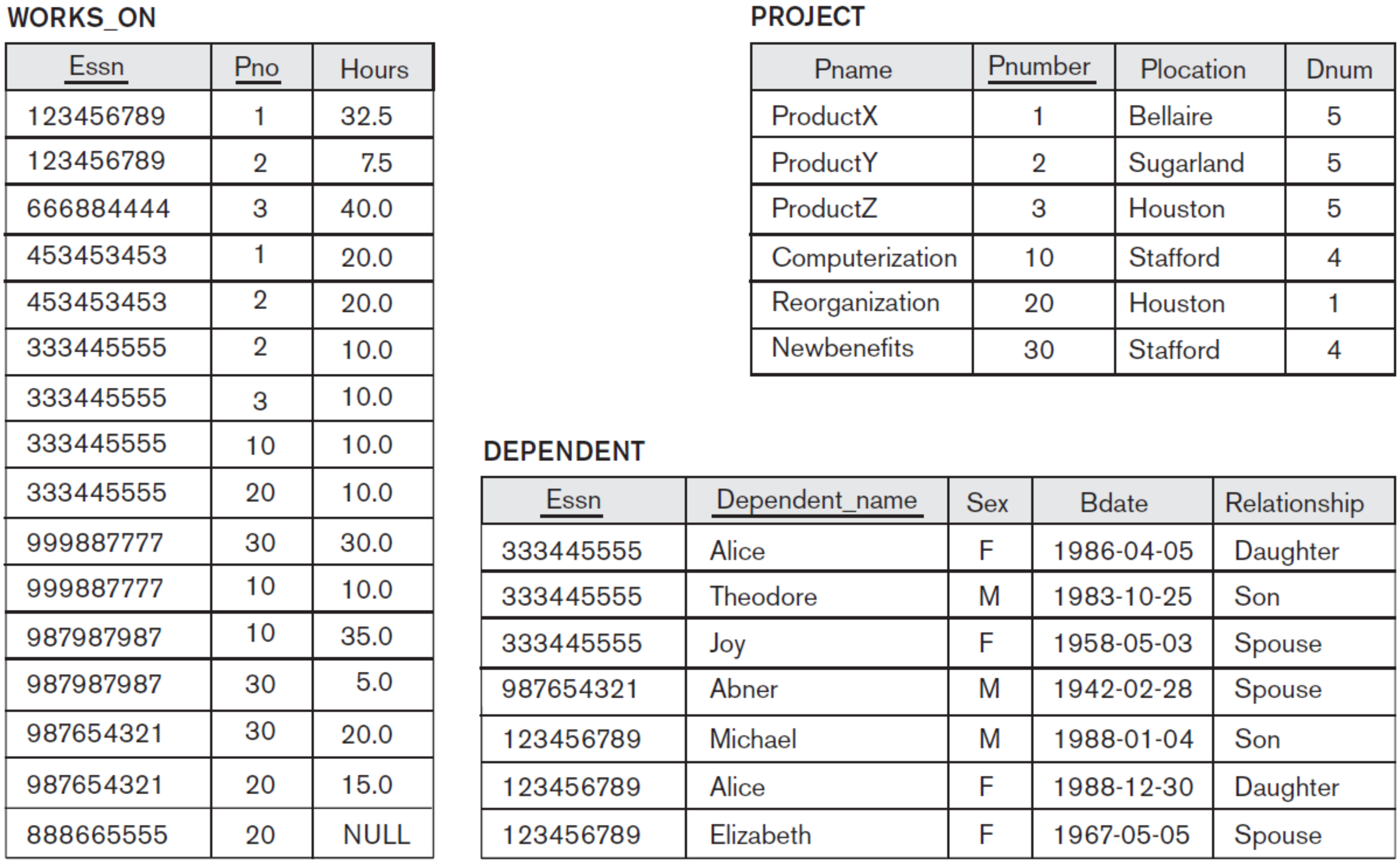
DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Database Example: COMPANY



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

Schema Creation

- SQL schema
  - Group tables and other constructs together
    - Belong to the *same database application*
  - Created using CREATE SCHEMA statement
    - Identified by a *schema name*

Schema Creation

- Example
  - Create a schema named COMPANY
    - Owned by Jsmith
  - **CREATE SCHEMA** COMPANY **AUTHORIZATION** `Jsmith';

Table Creation

- CREATE TABLE statement
  - Specify a new relation
    - Table name
    - Attributes (list of <attribute_name, data_type> pairs)
    - Initial constraints

Table Creation

- Attributes
  - Data types
    - INT, FLOAT, DECIMAL, CHAR(*n*), VARCHAR(*n*), …
  - NOT NULL attribute constraint
    - NULL is not permitted for a particular attribute
  - DEFAULT clause
    - Specify a default value

Table Creation

- Initial constraints
  - PRIMARY KEY clause
    - Specifies one or more attributes that make up the primary key of a relation
  - UNIQUE clause
    - Specifies alternate (secondary) keys candidate key

primary 와 unique 차이
primary의 performance을 unique보다 좋게 설계한다.

Data
And
Knowledge
Engineering

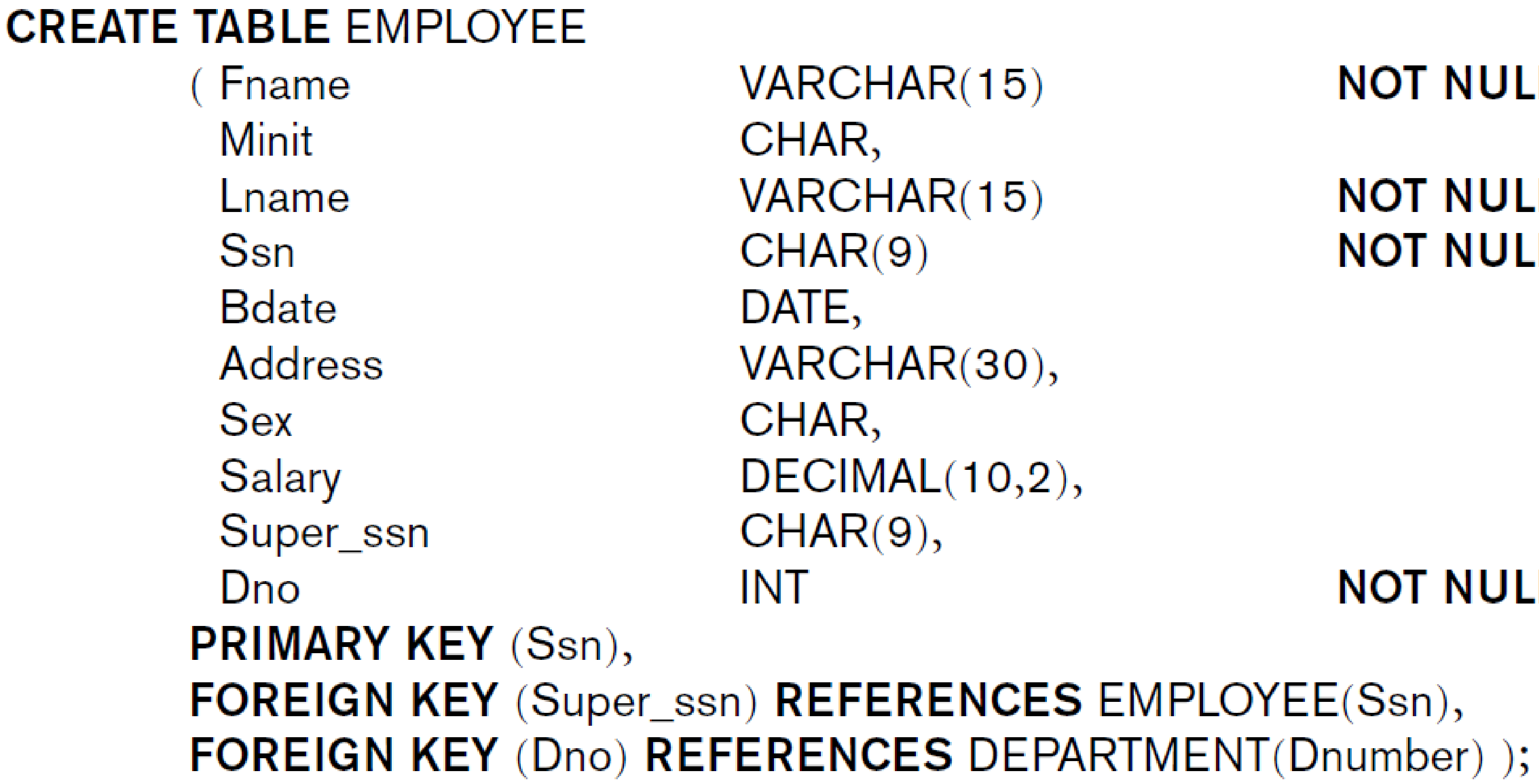
Table Creation

- Initial constraints
  - FOREIGN KEY clause
    - Specifies a foreign key for referential integrity constraint
    - Can specify a *referential triggered action*
      - Actions: SET NULL, CASCADE, SET DEFAULT
      - Triggers: ON DELETE, ON UPDATE (참조하는 primary key 해당)

ON DELETE -	SET NULL	SET DEFAULT	CASCADE
	FK<- NULL	FK<- DEFAULT	FK<-delete
ON UPDATE -	SET NULL	SET DEFAULT	CASCADE
			FK도 PK와 같이 같은 값으로 update

Table Creation: Examples

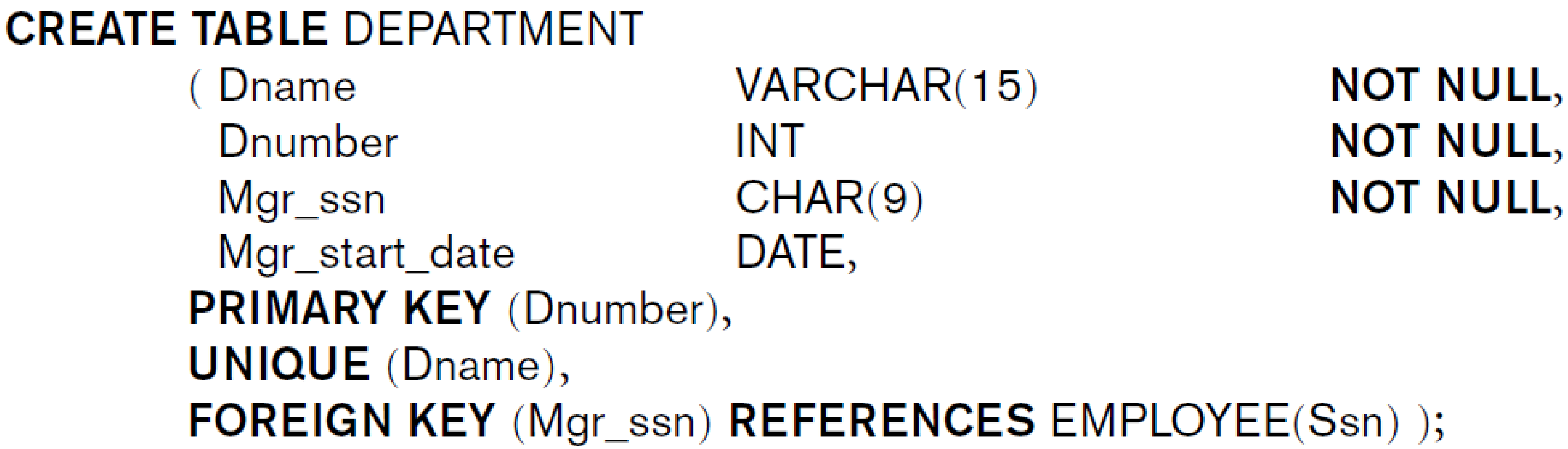
- COMPANY schema



```
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) );
```

Table Creation: Examples

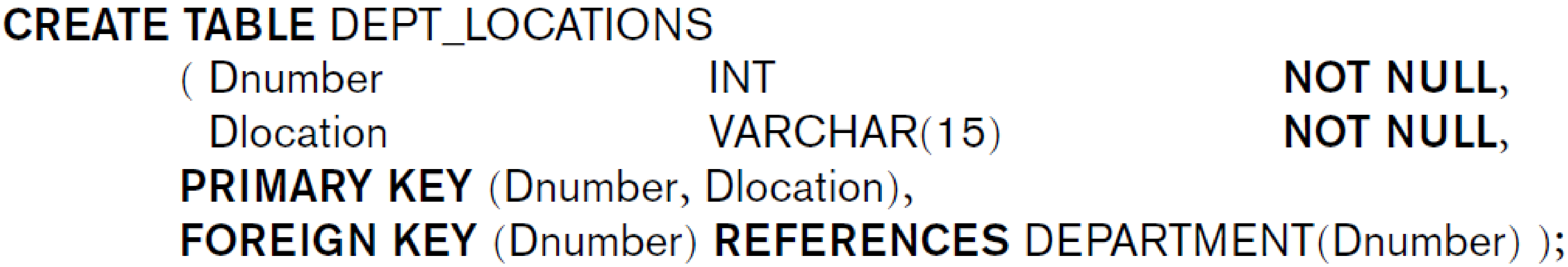
- COMPANY schema



```
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) );
```

Table Creation: Examples

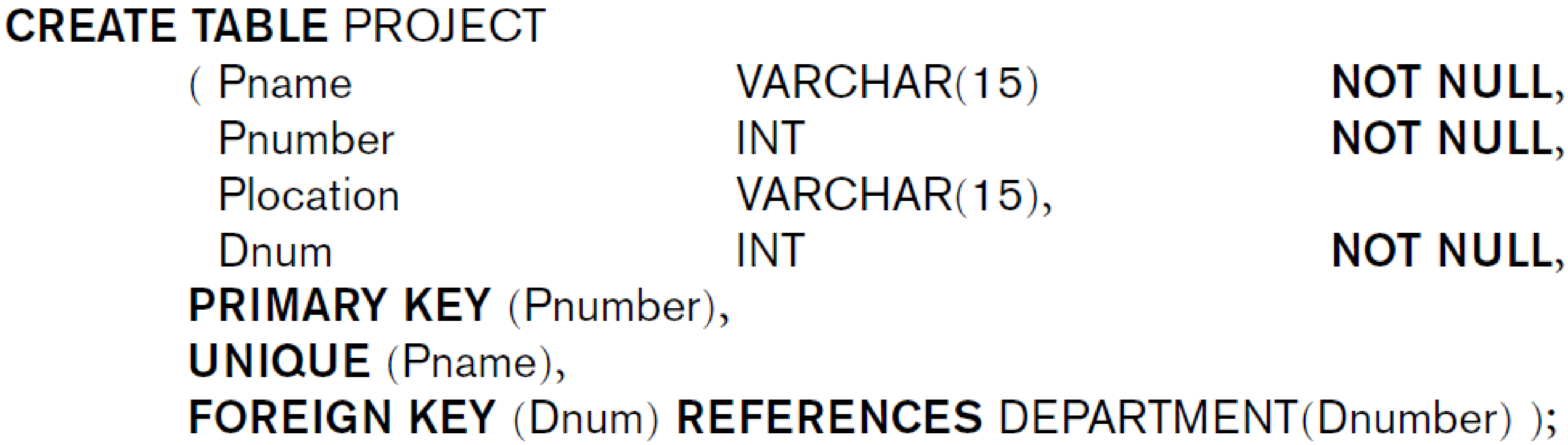
- COMPANY schema



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

Table Creation: Examples

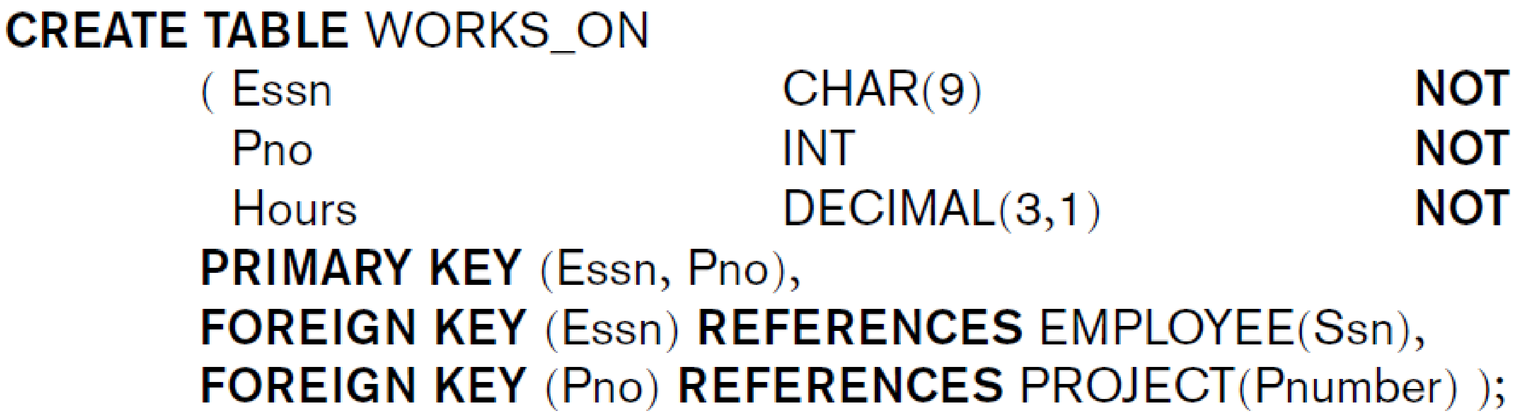
- COMPANY schema



```
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) );
```

Table Creation: Examples

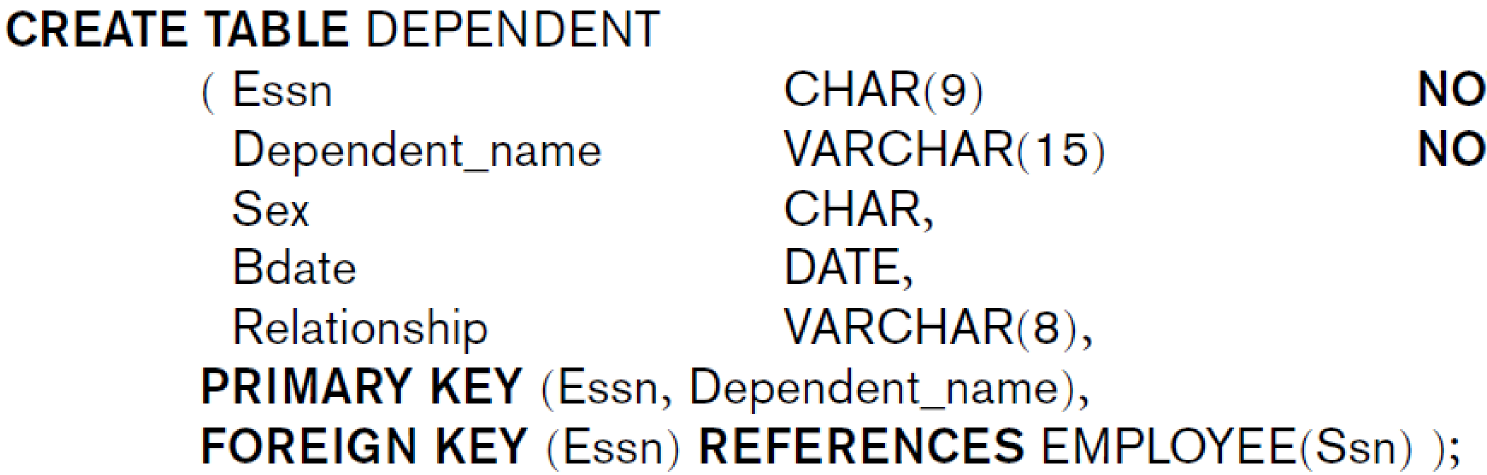
- COMPANY schema



```
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) );
```

Table Creation: Examples

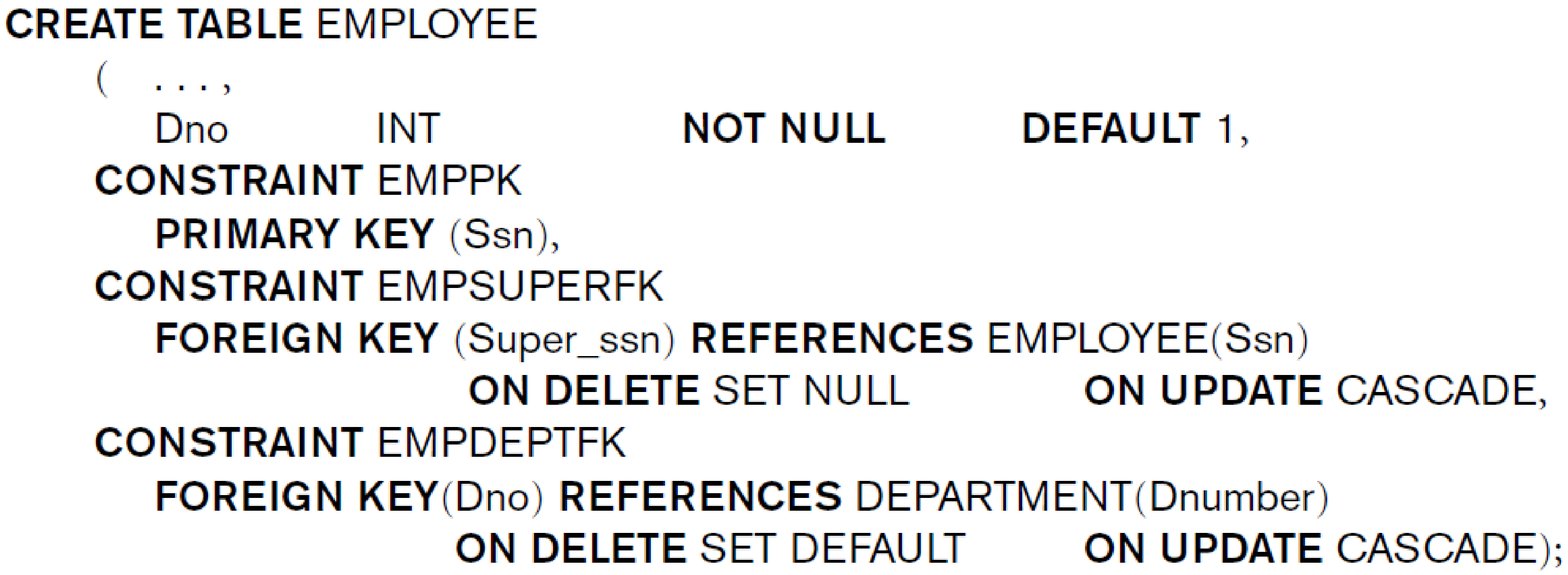
- COMPANY schema



```
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) );
```

Table Creation: Examples

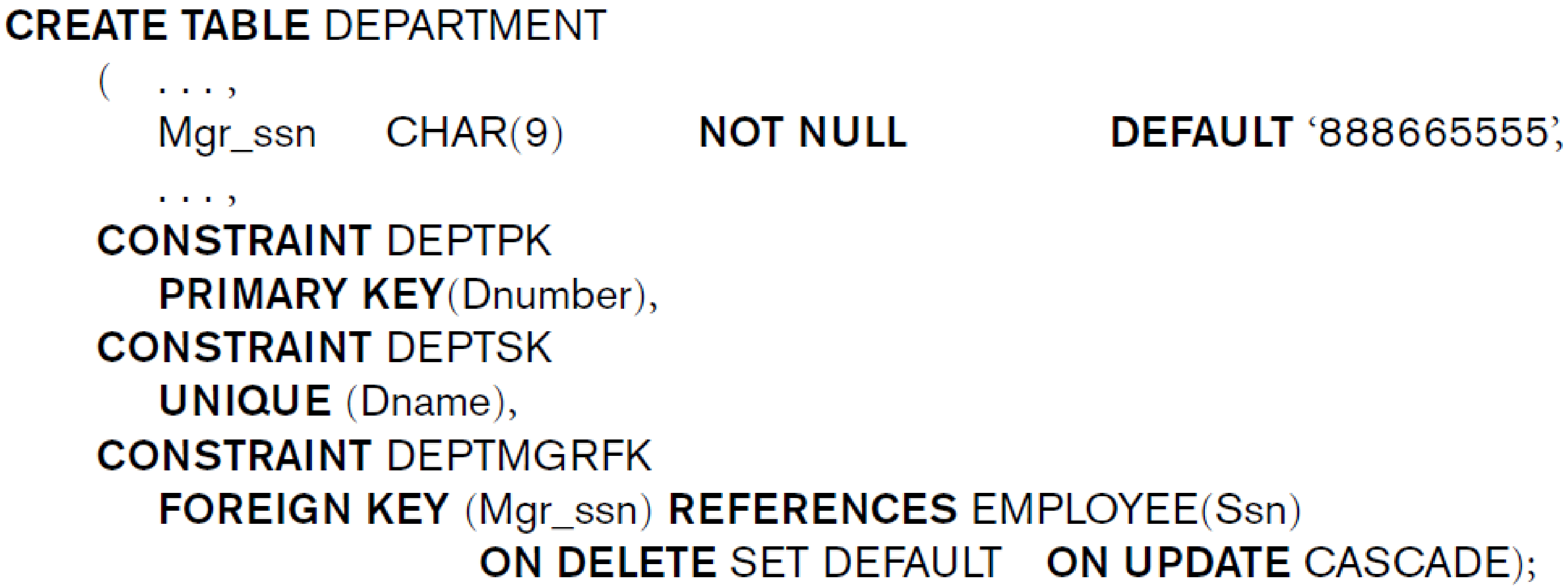
- Referential triggered actions



```
CREATE TABLE EMPLOYEE
    (   . . . ,
        Dno          INT                 NOT NULL         DEFAULT 1,
    CONSTRAINT EMPPK
        PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (Super_ssn) 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);
```

Table Creation: Examples

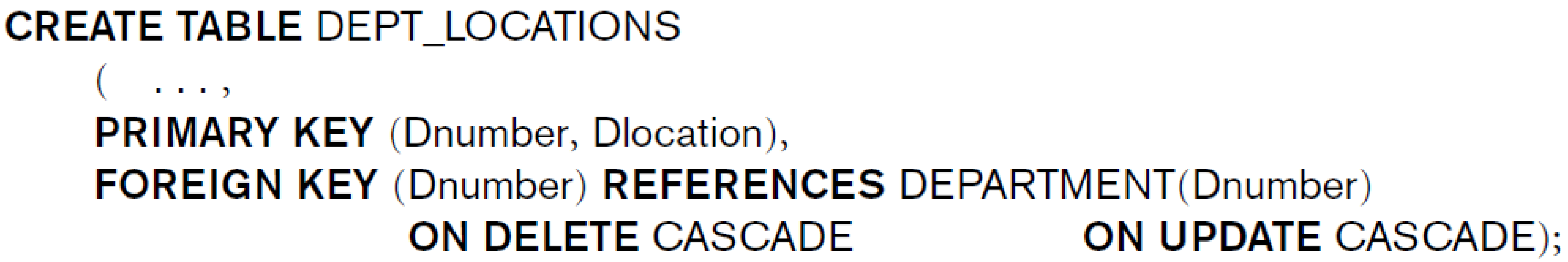
- Referential triggered actions



```
CREATE TABLE DEPARTMENT
    (   . . . ,
        Mgr_ssn     CHAR(9)        NOT NULL              DEFAULT '888665555',
        . . . ,
    CONSTRAINT DEPTPK
        PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
        UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                   ON DELETE SET DEFAULT   ON UPDATE CASCADE);
```

Table Creation: Examples

- Referential triggered actions



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

Schema and Table Deletion

- DROP SCHEMA statement
  - Used when *a whole schema* is no longer needed
  - Options
    - CASCADE
      - Remove the schema and all its elements
    - RESTRICT
      - Remove the schema only if it has *no elements* in it

Schema and Table Deletion

- DROP SCHEMA statement
  - Example
    - Remove COMPANY schema and all its elements
    - **DROP SCHEMA** COMPANY **CASCADE**;

Schema and Table Deletion

- DROP TABLE statement
  - Removes the table definition and all its tuples
  - Options
    - CASCADE
      - Remove the table and *all elements referencing the table* from the schema
    - RESTRICT
      - Remove the table only if it is *not referenced* in any constraints

Schema and Table Deletion

- DROP TABLE statement
  - Example
    - Remove DEPENDENT table from COMPANY schema
    - **DROP TABLE** DEPENDENT **CASCADE**;

Altering Tables

- ALTER TABLE statement
  - Adding or dropping a column (attribute)
  - Changing a column definition
  - Adding or dropping table constraints constraint name 필요!!

Altering Tables

- Examples
  - **ALTER TABLE** COMPANY.EMPLOYEE
    ADD COLUMN Job **VARCHAR**(12);
  - **ALTER TABLE** COMPANY.EMPLOYEE
    DROP COLUMN Address **CASCADE**;
  - **ALTER TABLE** COMPANY.EMPLOYEE
    ALTER COLUMN Mgr_ssn **DROP DEFAULT**;
  - **ALTER TABLE** COMPANY.EMPLOYEE
    ALTER COLUMN Mgr_ssn **SET DEFAULT** ‘333445555’;
  - **ALTER TABLE** COMPANY.EMPLOYEE
    DROP CONSTRAINT EMPSUPERFK;

SELECT Query Basics

- SELECT statement
  - One basic statement for retrieving information from a database
    - *NOTE: Not equivalent to SELECT relational operation*
    - Combination of SELECT, PROJECT, JOIN relational algebra operations

SELECT Query Basics

- SELECT statement
  - SQL allows a table to *have two or more tuples that are identical* in all their attribute values
    - Unlike the relational model
    - Not a set
    - *DISTINCT clause can be used to make it a set*

why ? 1. duplicate elimination is an expensive operation
2. the user may want to see duplicate tuples in the result of a query
3. when an aggregate function is applied to tuples,in most cases we do not want to eliminate duplicates

SELECT Query Basics

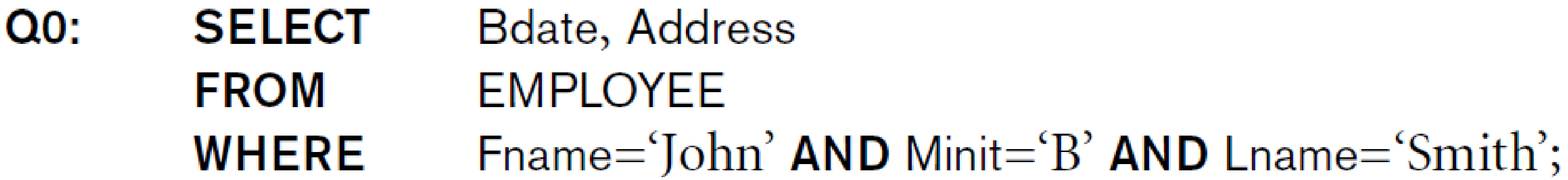
- Basic form:

  - **SELECT** <attribute list>
    FROM <table list>
    WHERE <condition>;

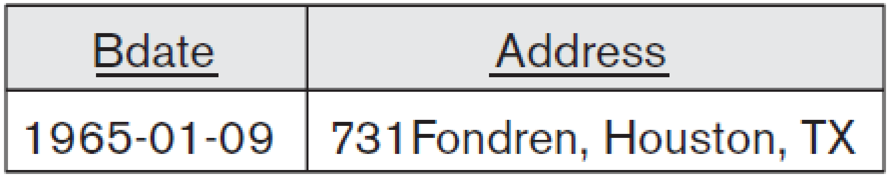
    - <attribute list>: a list of attribute names whose values are to be retrieved by the query
    - <table list>: a list of the relation names required to process the query
    - <condition> a Boolean expression that identifies the tuples to be retrieved by the query

SELECT Query Basics: Examples

- **Query 0**. Retrieve the birth date and address of the employee(s) whose name is ‘John B. Smith’



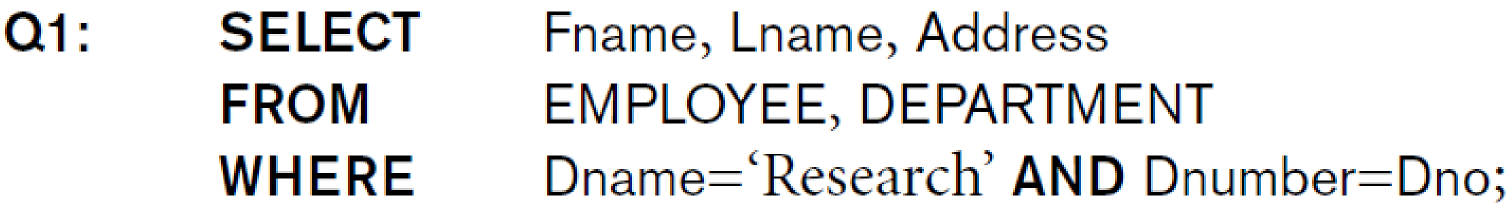
```
Q0:     SELECT    Bdate, Address
        FROM      EMPLOYEE
        WHERE     Fname=‘John’ AND Minit=‘B’ AND Lname=‘Smith’;
```



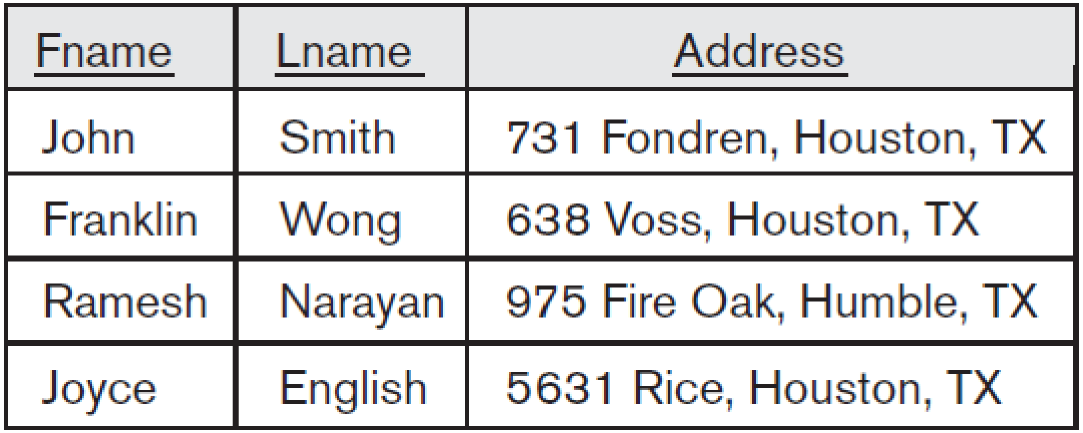
Bdate	Address
1965-01-09	731Fondren, Houston, TX

SELECT Query Basics: Examples

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



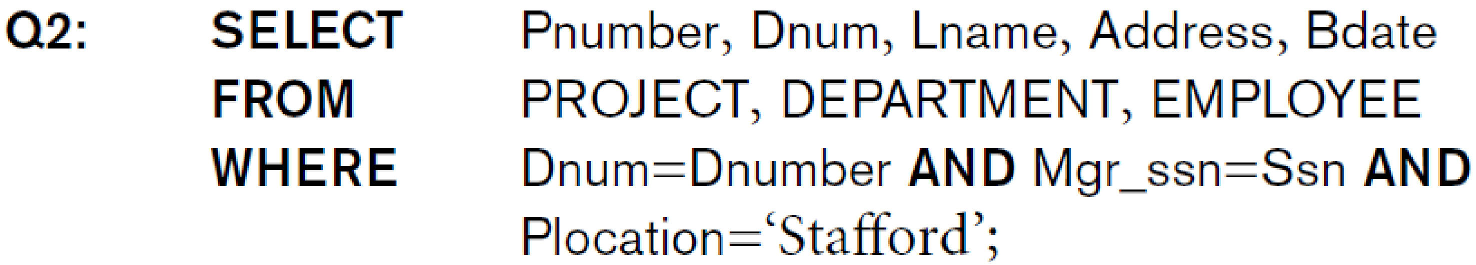
```
Q1:     SELECT   Fname, Lname, Address
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    Dname='Research' AND Dnumber=Dno;
```



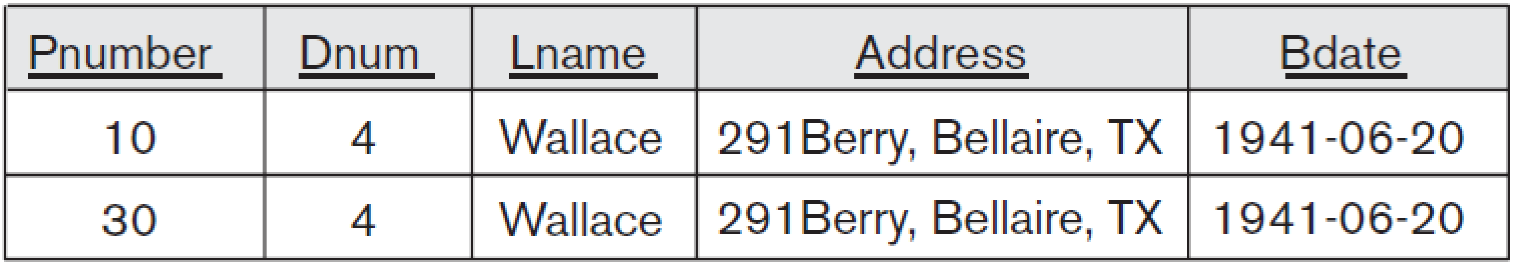
Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

SELECT Query Basics: Examples

- **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 birth date



```
Q2:     SELECT   Pnumber, Dnum, Lname, Address, Bdate
        FROM     PROJECT, DEPARTMENT, EMPLOYEE
        WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
                 Plocation=‘Stafford’;
```



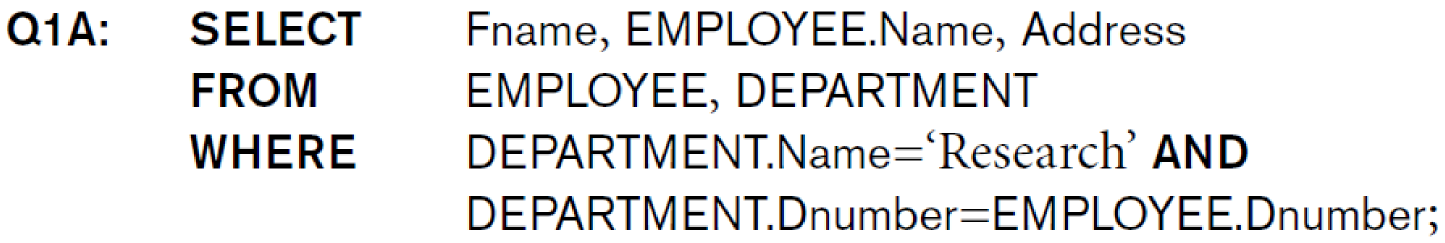
Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Ambiguous Attribute Names

- The same name can be used for two or more attributes
  - In different relations
- Must *qualify* the attribute name *with the relation name* to prevent ambiguity

Ambiguous Attribute Names

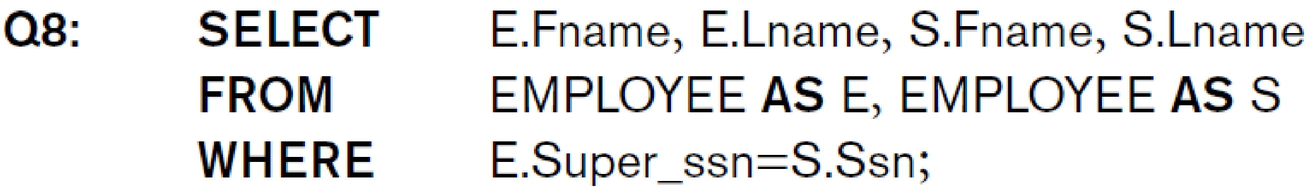
- Example
  - Suppose that there are attributes with the same name in EMPLOYEE and DEPARTMENT table
    - Dnumber, Name



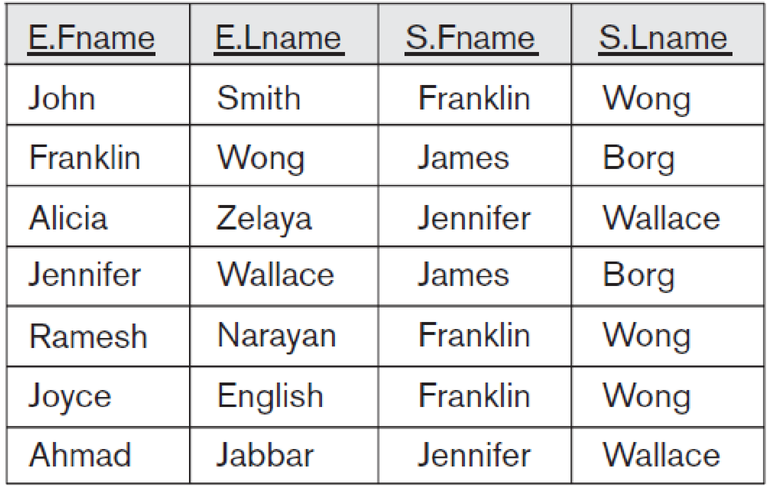
```
Q1A:    SELECT    Fname, EMPLOYEE.Name, Address
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DEPARTMENT.Name='Research' AND
                  DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Table Aliases

- Declare alternative relation names
- **Query 8**. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor



```
Q8:     SELECT   E.Fname, E.Lname, S.Fname, S.Lname
        FROM     EMPLOYEE AS E, EMPLOYEE AS S
        WHERE    E.Super_ssn=S.Ssn;
```



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

Table Aliases

- Attribute names also can be renamed

- Example

  - **SELECT** FN, MI, LN
    FROM EMPLOYEE
    AS E(FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO)
    WHERE E.DNO = 5;

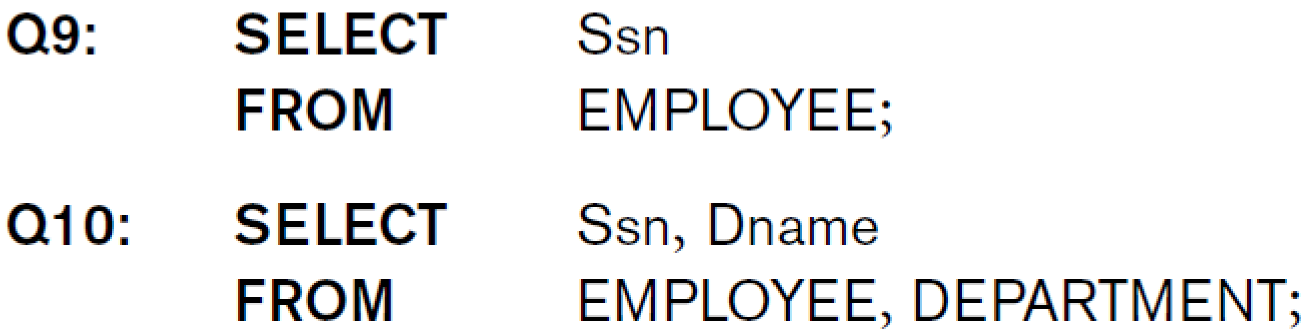
모든 attribute 설정해줘야 한다.

Missing WHERE Clause

- Indicates *no condition* on tuple selection
- Also used to get CROSS PRODUCT of multiple tables

Missing WHERE Clause

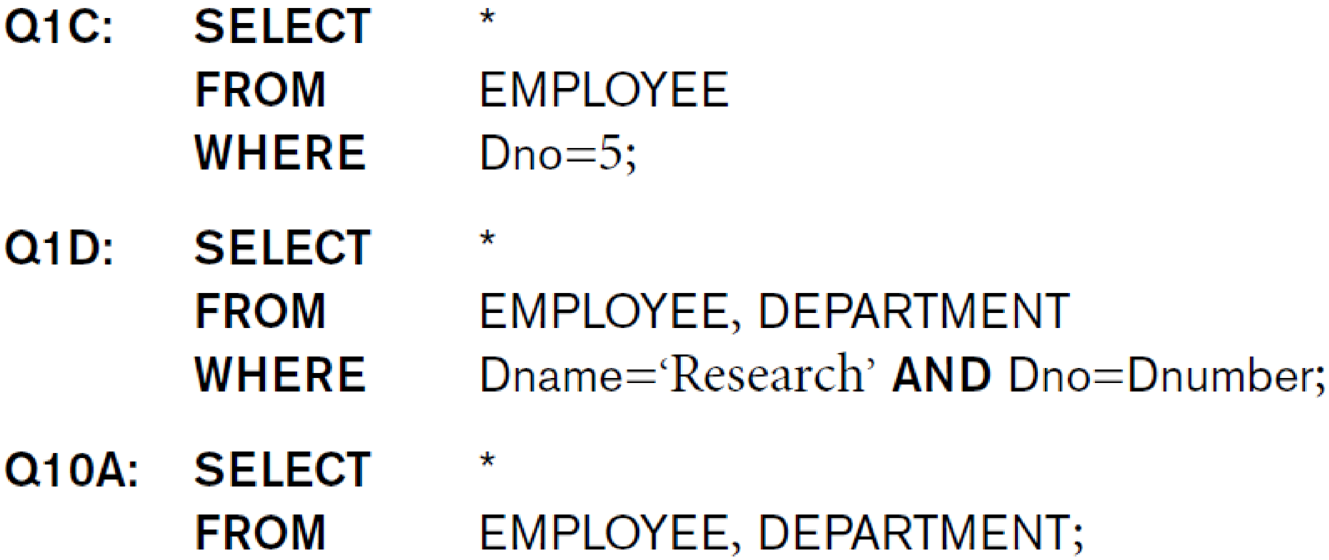
- **Queries 9 and 10**. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database



```
Q9:     SELECT    Ssn
        FROM      EMPLOYEE;

Q10:    SELECT    Ssn, Dname
        FROM      EMPLOYEE, DEPARTMENT;
```

Use of the Asterisk

- Retrieve all the attributes of the selected tuples
- To avoid inconvenience of specifying all attribute names



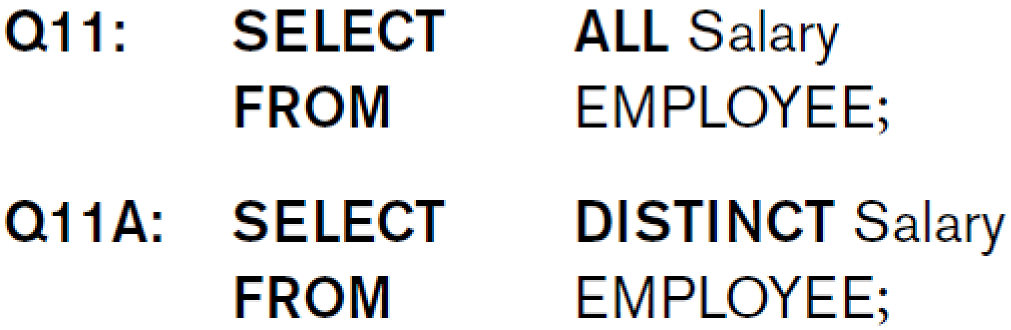
```
Q1C:    SELECT   *
        FROM     EMPLOYEE
        WHERE    Dno=5;

Q1D:    SELECT   *
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    Dname='Research' AND Dno=Dnumber;

Q10A:   SELECT   *
        FROM     EMPLOYEE, DEPARTMENT;
```

Tables as Sets

- SQL does *not automatically eliminate duplicate* tuples in query results
  - Duplicate tuples can be used in some cases
- DISTINCT keyword
  - Only distinct tuples should remain in the result

Tables as Sets

- **Query 11**. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A)



```
Q11:    SELECT    ALL Salary
                  FROM      EMPLOYEE;

Q11A:   SELECT    DISTINCT Salary
                  FROM      EMPLOYEE;
```

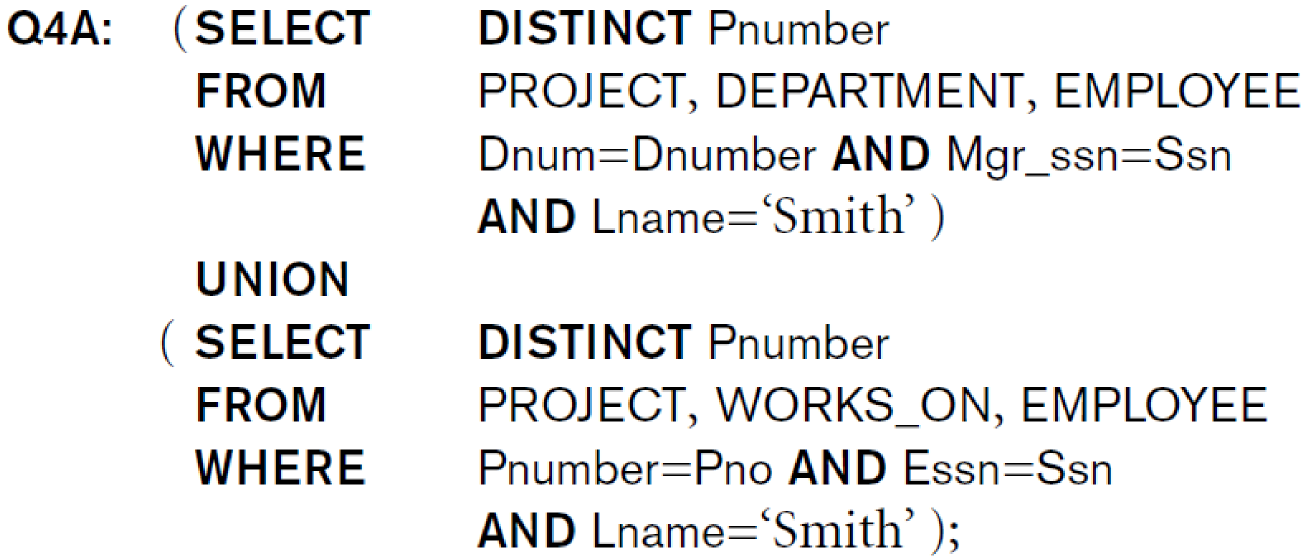
Tables as Sets

- Set operations
  - UNION
  - EXCEPT (difference)
  - INTERSECT
  - Query results of set operations are sets
    - No duplicate tuples

constraint : union compatibilty
1. same number of attributes
2. dom(Ai) = dom(Bi) 1<=i<=n
compatible domain to each other

Tables as Sets

- **Query 4**. 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



```
Q4A:   ( 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' );
```

Summary

- Basic SQL
  - Data definition commands for creating and removing tables
  - Commands for constraint specification
  - Simple retrieval queries

References

1. Boyce, Raymond F., et al. "Specifying queries as relational expressions." *ACM SIGIR Forum*. Vol. 9. No. 3. ACM, 1973.
2. Chamberlin, Donald D., and Raymond F. Boyce. "SEQUEL: A structured English query language." *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*. ACM, 1974.
3. Chamberlin, Donald D., et al. "SEQUEL 2: A unified approach to data definition, manipulation, and control." *IBM Journal of Research and Development* 20.6 (1976): 560-575.

Have a nice day!

Data
And
Knowledge
Engineering