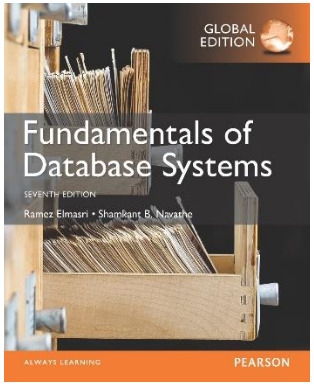
Database Systems

Department of Computer Science
Chungbuk National University

Prof. Jong Yun LEE

CHAPTER 6:
Basic SQL


GLOBAL EDITION
Fundamentals of Database Systems
SEVENTH EDITION
Ramez Elmasri • Shamkant B. Navathe
ALWAYS LEARNING
PEARSON

Chapter 6 Outline

- **SQL Data Definition and Data Types**
- **Specifying Constraints in SQL**
- **Basic Retrieval Queries in SQL**
- **`INSERT`, `DELETE`, and `UPDATE` Statements in SQL**
- **Additional Features of SQL**

Overview

- **SQL language**
  - Considered one of the major reasons for the commercial success of relational databases

- **SQL**
  - The origin of SQL is relational predicate calculus called tuple calculus (see Ch.8) which was proposed initially as the language SQUARE.
  - SQL actually comes from the word "SEQUEL" which was the original term used in the paper: "SEQUEL TO SQUARE" by Chamberlin and Boyce. IBM could not copyright that term, so they abbreviated to SQL and copyrighted the term SQL.
  - Now popularly known as "**Structured Query language**".
  - SQL is an informal or practical rendering of the relational data model with syntax

Relational Operations

- **Relational Algebra**
- **Relational Calculus**
  - Tuple relational calculus
  - Domain relational calculus

Overview

SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce after learning about the relational model from Ted Codd in the early 1970s. This version, initially called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s.

Donald D. Chamberlin (born 21 December 1944) is an American computer scientist who is best known as one of the principal designers of the original SQL language specification with Raymond Boyce. He also made significant contributions to the development of XQuery.

Raymond F. Boyce (1947–1974) was an American computer scientist who was known for his research in relational databases. He is best known for his work co-developing the SQL database language and **Boyce-Codd normal form.**

SQL Data Definition and Data Types

SQL Data Definition, Data Types, Standards

- **Terminology:**
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute

- **`CREATE` statement**
  - Main SQL command for data definition

- **The language has features for : Data definition, Data Manipulation, Transaction control, Indexing, Security specification (`Grant` and `Revoke`), Active databases (`Trigger`), Multi-media, Distributed databases etc.**

SQL Standards

- **SQL has gone through many standards: starting with SQL-86 or SQL 1.A. SQL-92 is referred to as SQL-2.**

- **Later standards (from SQL-1999) are divided into core specification and specialized extensions.**
  - The extensions are implemented for different applications – such as data mining, data warehousing, multimedia etc.

- **SQL-2006 added XML features (Ch. 13); In 2008 they added Object-oriented features (Ch. 12).**

- **SQL-3 is the current standard which started with SQL-1999. It is not fully implemented in any RDBMS.**

Schema and Catalog Concepts in SQL

- **We cover the basic standard SQL syntax – there are variations in existing RDBMS systems**

- **SQL schema**

  in some systems, a schema is called *database*

  - Identified by a **schema name**
  - Includes an **authorization identifier** and **descriptors** for each element

- **Schema elements include**
  - Tables, constraints, views, domains, and other constructs

- **Each statement in SQL ends with a semicolon**

Schema and Catalog Concepts in SQL (cont'd.)

- **`CREATE SCHEMA` statement**
  - `CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;`

- **Catalog**
  - Named <u>collection of schemas </u>in an SQL environment

- **SQL also has the concept of a cluster of catalogs.**

The `CREATE TABLE` Command in SQL

- **Specifying a new relation**
  - Provide name of table
  - Specify attributes, their types and initial constraints

- **Can optionally specify schema:**
  - `CREATE TABLE COMPANY.EMPLOYEE ...`

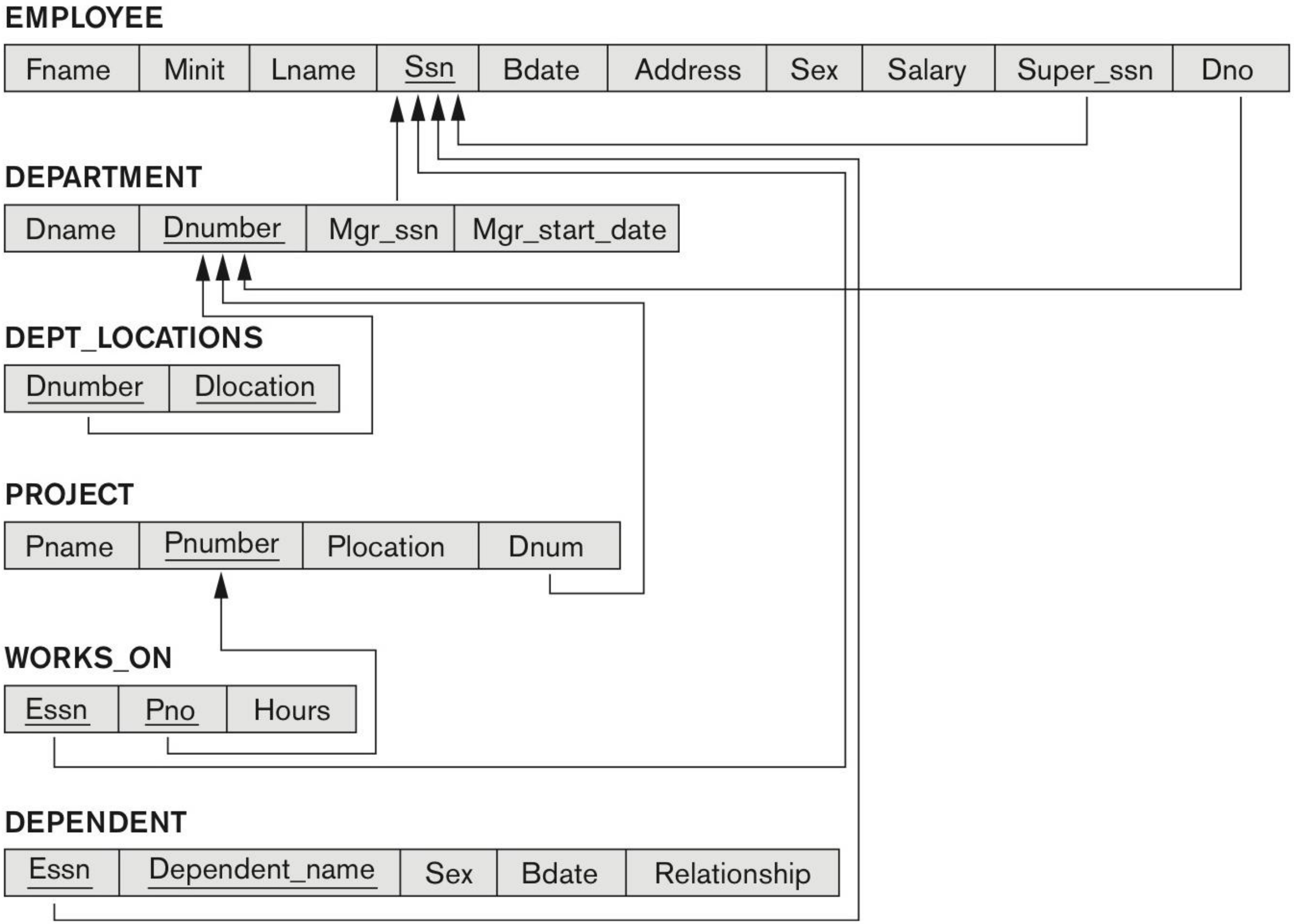
    or
  - `CREATE TABLE EMPLOYEE ...`

The `CREATE TABLE` Command in SQL (cont'd.)

- **Base tables (base relations)**
  - Relation and its tuples are actually created and stored as a file by the DBMS

- **Virtual relations (views)**
  - Created through the `CREATE VIEW` statement.
  - Do not correspond to any physical file.

COMPANY relational database schema (Fig. 5.7)



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

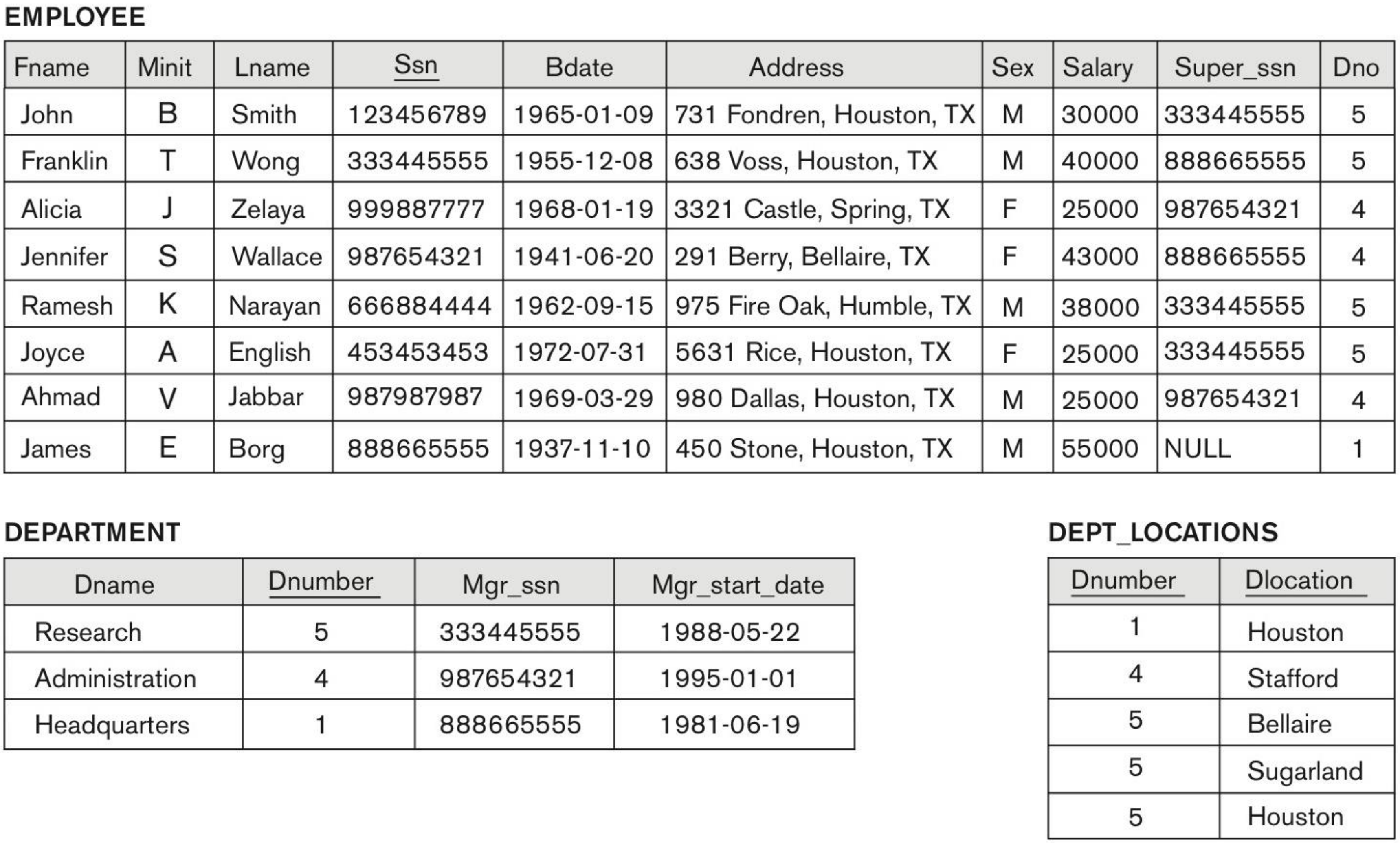
WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

One possible database state for the COMPANY relational database schema (Fig. 5.6)



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

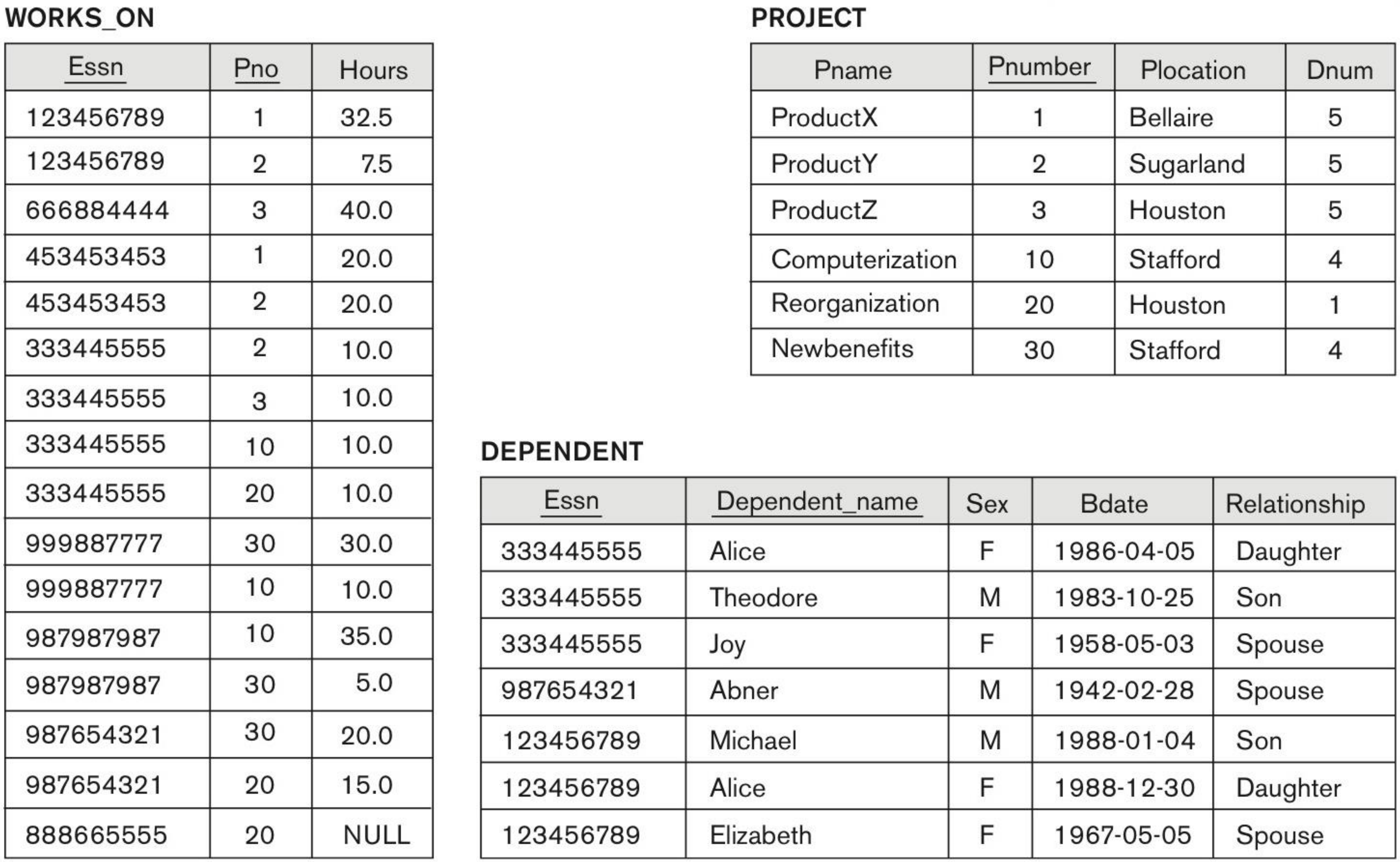
DEPARTMENT

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

One possible database state for the COMPANY relational database schema – continued (Fig. 5.6)



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

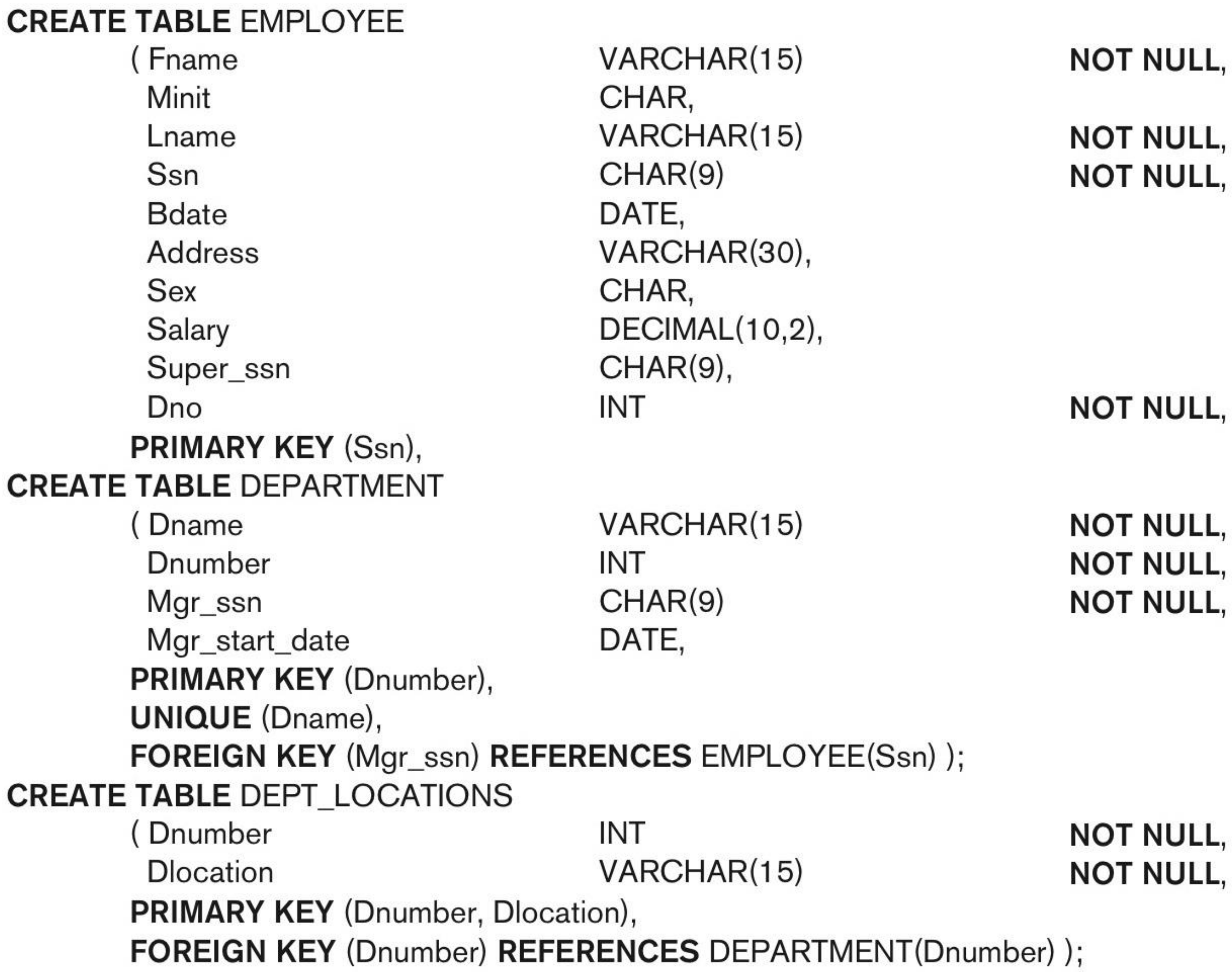
PROJECT

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

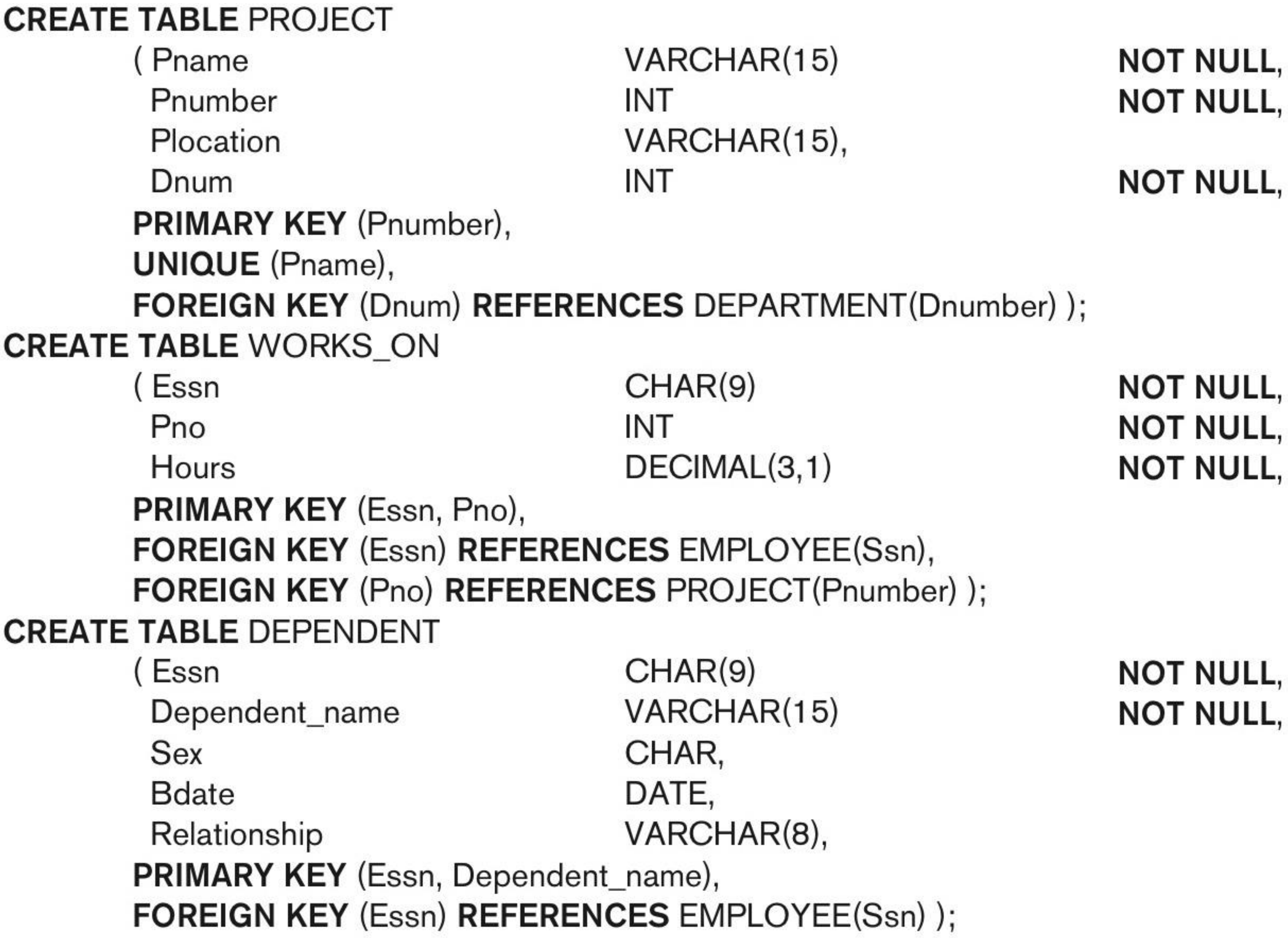
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1)



```
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),
        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) );
```

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1



```
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) );
```

```
mysql> create database company;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> use company;
Database changed
mysql>
mysql> CREATE TABLE EMPLOYEE (
    -> Fname VARCHAR(15) NOT NULL,
    -> Minit CHAR,
    -> Lname VARCHAR(15) NOT NULL,
    -> Ssn CHAR(9) NOT NULL,
    -> Bdata DATE,
    -> Address VARCHAR(30),
    -> Sex CHAR,
    -> Salary DECIMAL(10,2),
    -> Super_ssn CHAR(9),
    -> Dno INT,
    -> PRIMARY KEY(Ssn));
mysql> show tables;
+-------------------+
| Tables_in_company |
+-------------------+
| EMPLOYEE          |
+-------------------+
```

```
mysql> desc EMPLOYEE;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| Fname     | varchar(15)   | NO   |     | NULL    |       |
| Minit     | char(1)       | YES  |     | NULL    |       |
| Lname     | varchar(15)   | NO   |     | NULL    |       |
| Ssn       | char(9)       | NO   | PRI | NULL    |       |
| Bdata     | date          | YES  |     | NULL    |       |
| Address   | varchar(30)   | YES  |     | NULL    |       |
| Sex       | char(1)       | YES  |     | NULL    |       |
| Salary    | decimal(10,2) | YES  |     | NULL    |       |
| Super_ssn | char(9)       | YES  |     | NULL    |       |
| Dno       | int(11)       | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

Attribute Data Types and Domains in SQL

- **Basic data types**
  - **Numeric** data types
    - Integer numbers: **INTEGER, INT, and SMALLINT**
    - Floating-point (real) numbers: **FLOAT** or REAL, and DOUBLE PRECISION
  - **Character-string** data types
    - Fixed length: **CHAR(*n*),** CHARACTER(*n*)
    - Varying length: **VARCHAR(*n*),** CHAR VARYING(*n*), CHARACTER VARYING(*n*)

Attribute Data Types and Domains in SQL (cont'd.)

- **Bit-string** data types
  - Fixed length: `BIT(`*n*`)`
  - Varying length: `BIT VARYING(`*n*`)`
- **Boolean** data type
  - Values of `TRUE` or `FALSE` or `NULL`
- **DATE** data type
  - Ten positions
  - Components are `YEAR`, `MONTH`, and `DAY` in the form `YYYY-MM-DD`
  - Multiple mapping functions available in RDBMSs to change date formats

Attribute Data Types and Domains in SQL (cont'd.)

- **Additional data types**
  - **Timestamp** data type

    Includes the `DATE` and `TIME` fields

    - Plus a minimum of six positions for decimal fractions of seconds
    - Optional `WITH TIME ZONE` qualifier
  - **INTERVAL** data type
    - Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp
  - **DATE, TIME, Timestamp, INTERVAL** data types can be **cast** or converted to string formats for comparison.

Attribute Data Types and Domains in SQL (cont'd.)

- **Domain**
  - Name used with the attribute specification
  - Makes it easier to change the data type for a domain that is used by numerous attributes
  - Improves schema readability
  - Example:
    - `CREATE DOMAIN SSN_TYPE AS CHAR(9);`

- **TYPE**
  - User Defined Types (UDTs) are supported for object-oriented applications. (See Ch.12) Uses the command: `CREATE TYPE`

Specifying Constraints in SQL

Specifying Constraints in SQL

Basic constraints:

- **Relational Model has 3 basic constraint types that are supported in SQL:**
  - **Key** constraint: A primary key value cannot be duplicated
  - **Entity Integrity** Constraint: A primary key value cannot be null
  - **Referential integrity** constraints : The “foreign key “ must have a value that is already present as a primary key, or may be null.

Specifying Attribute Constraints

Other Restrictions on attribute domains:

- **Default value of an attribute**
  - `DEFAULT <value>`
  - `NULL` is not permitted for a particular attribute (`NOT NULL`)

- **`CHECK` clause**
  - `Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);`

Specifying Key and Referential Integrity Constraints

- **`PRIMARY KEY` clause**
  - Specifies one or more attributes that make up the primary key of a relation
  - `Dnumber INT PRIMARY KEY;`

- **`UNIQUE` clause**
  - Specifies alternate (secondary) keys (called `CANDIDATE` keys in the relational model).
  - `Dname VARCHAR(15) UNIQUE;`

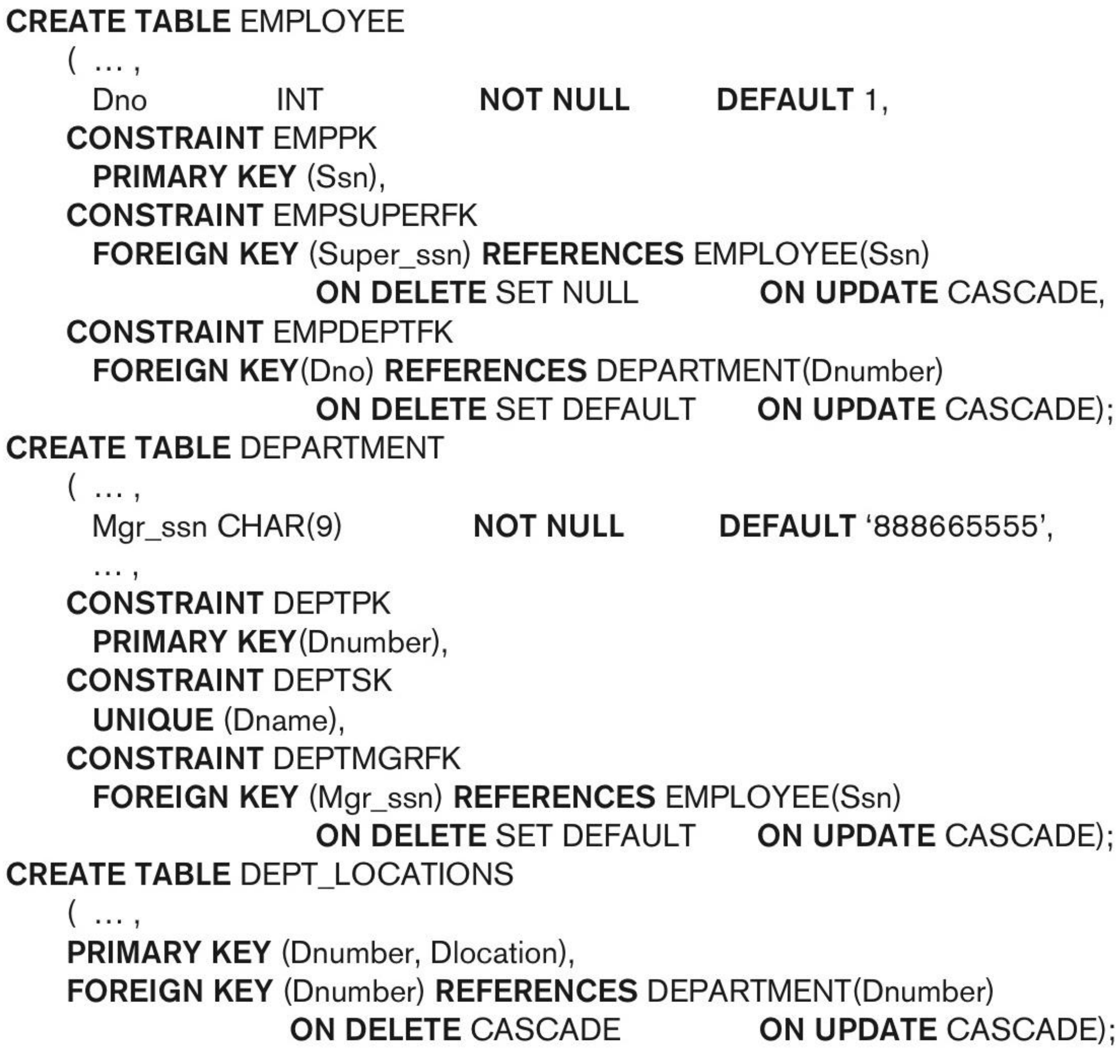
Specifying Key and Referential Integrity Constraints (cont'd.)

- `FOREIGN KEY` **clause**
  - Default operation: reject update on violation
  - Attach **referential triggered action** clause
    - Options include `SET NULL, CASCADE`, and `SET DEFAULT`
    - Action taken by the DBMS for `SET NULL` or `SET DEFAULT` is the same for both `ON DELETE` and `ON UPDATE`
    - `CASCADE` option suitable for "relationship" relations, multivaled attributes, weak entity types

Giving Names to Constraints

- **Using the Keyword `CONSTRAINT`**
  - Name a constraint
  - Useful for later altering

Default attribute values and referential integrity triggered action specification (Fig. 6.2)



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

Specifying Constraints on Tuples Using CHECK

- **Additional Constraints on individual tuples within a relation are also possible using CHECK**

- **CHECK clauses at the end of a CREATE TABLE statement**
  - Apply to each tuple individually
  - `CHECK (Dept_create_date <= Mgr_start_date);`

Basic Retrieval Queries in SQL

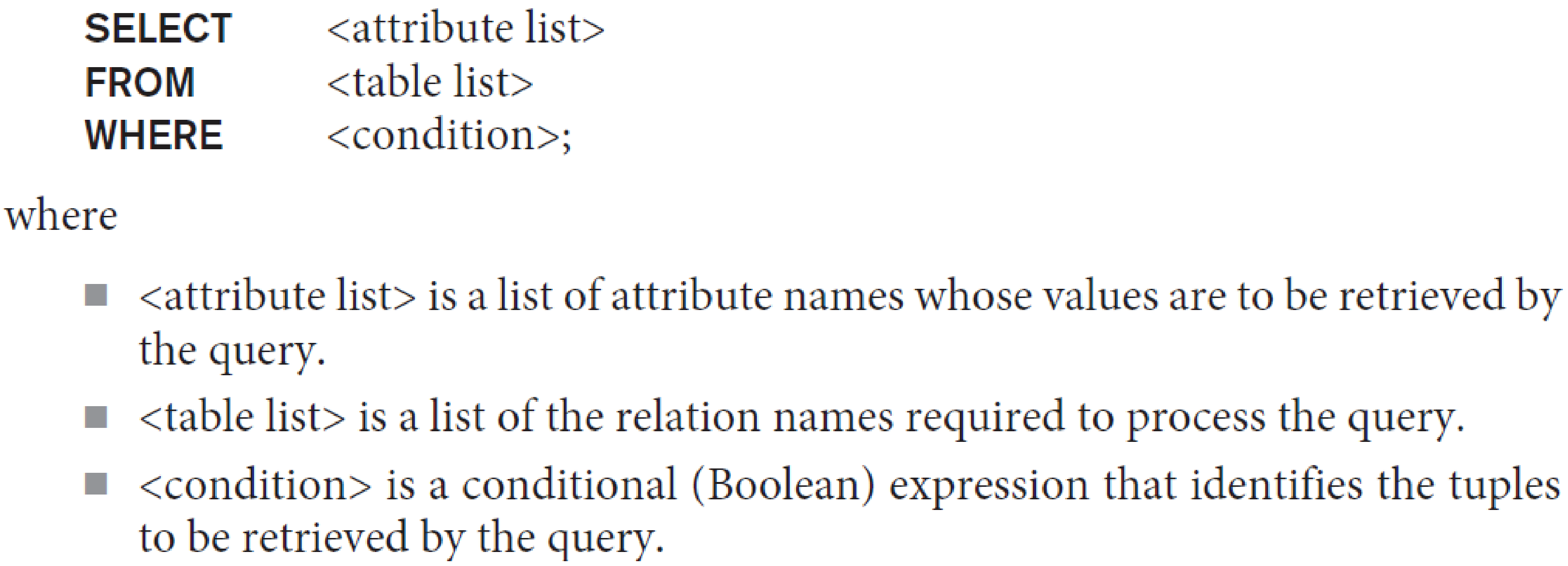
Basic Retrieval Queries in SQL

- **SELECT statement**
  - One basic statement for retrieving information from a database

- **SQL allows a table to have two or more tuples that are identical in all their attribute values**
  - Unlike relational model (relational model is strictly set-theory based)
  - Multiset or bag behavior
  - Tuple-id may be used as a key

The SELECT-FROM-WHERE Structure of Basic SQL Queries

- **Basic form of the SELECT statement:**



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

where

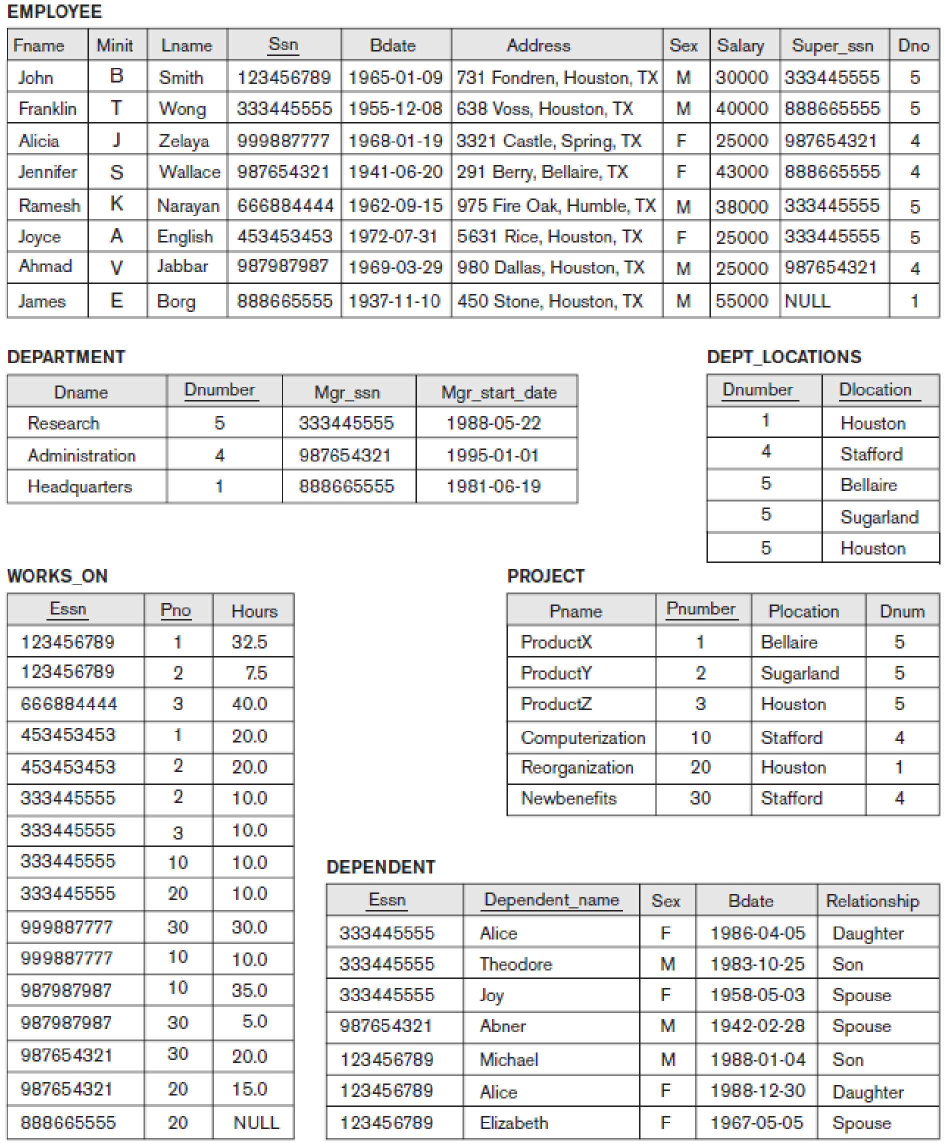
- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

The `SELECT-FROM-WHERE` Structure of Basic SQL Queries

- **Logical comparison operators**
  - `=, <, <=, >, >=, and <>`
- **Projection attributes**
  - Attributes whose values are to be retrieved
- **Selection condition**
  - Boolean condition that must be true for any retrieved tuple. Selection conditions include join conditions (see Ch.8) when multiple relations are involved.

Database state of COMPANY

Figure 5.6
One possible database state for the COMPANY relational database schema.



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

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

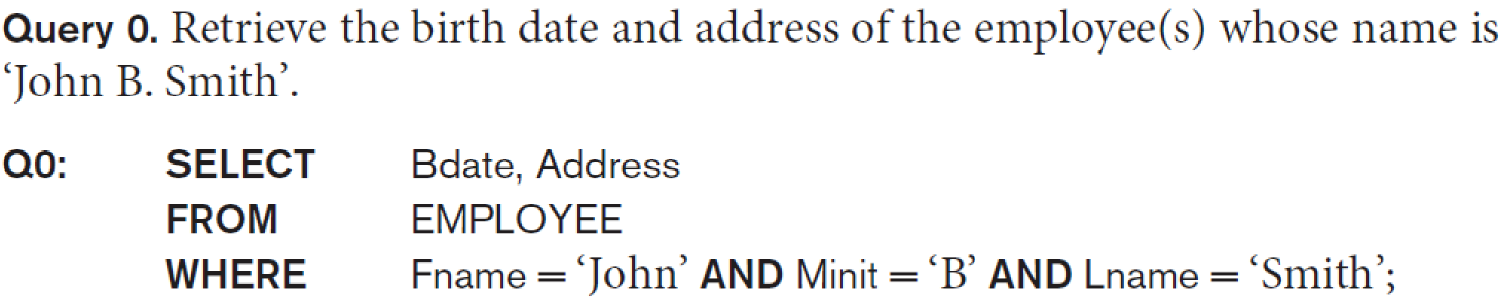
PROJECT

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

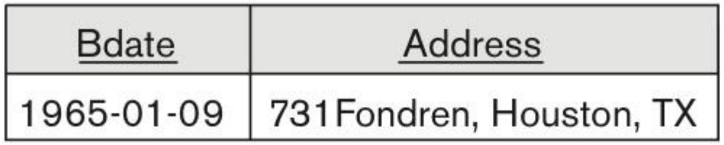
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Basic Retrieval Queries

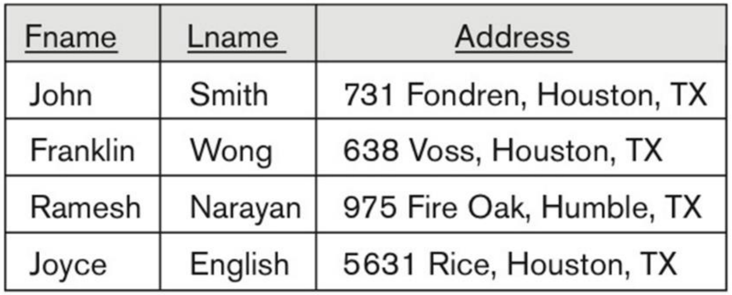


Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

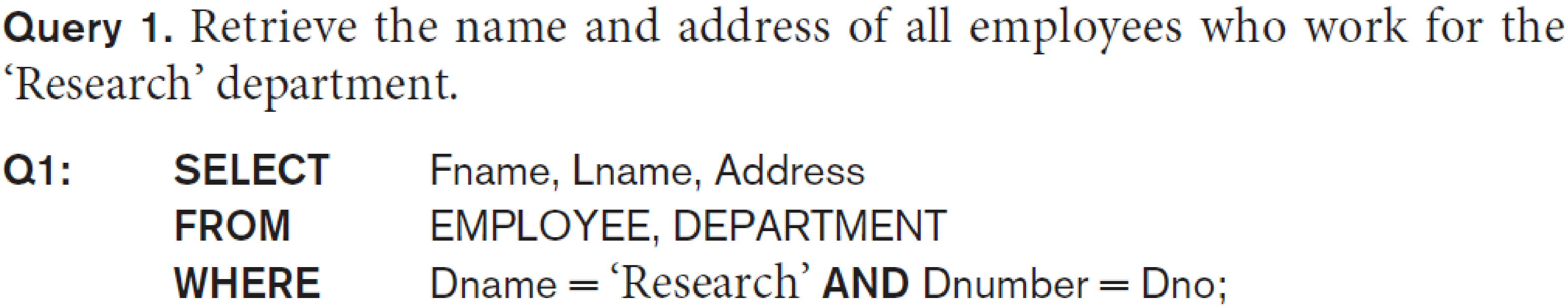
```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```



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



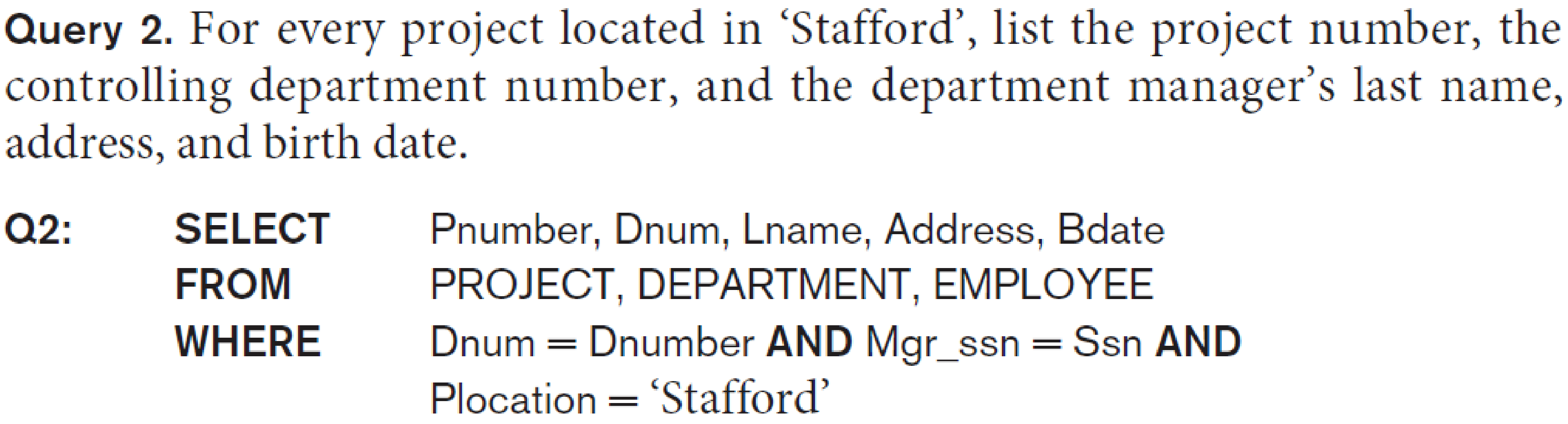
Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX



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

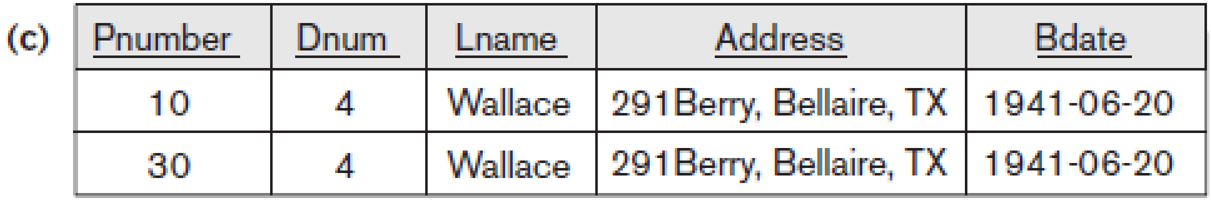
```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname = 'Research' AND Dnumber = Dno;
```

Basic Retrieval Queries (Contd.)



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'
```

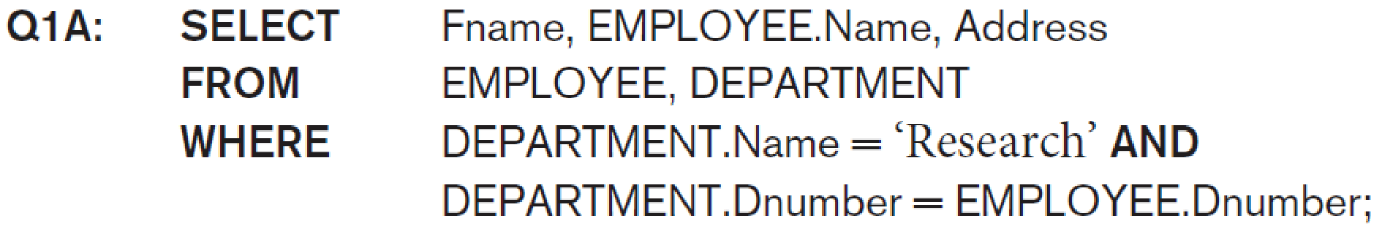


(c)

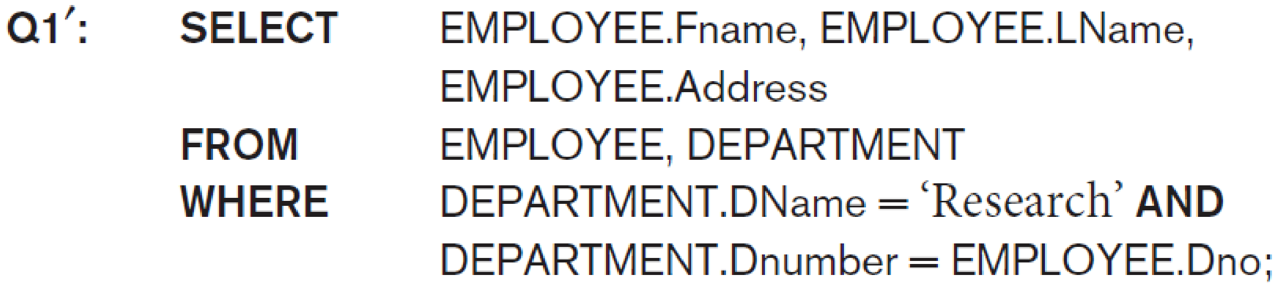
Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Ambiguous Attribute Names

- **Same name can be used for two (or more) attributes in different relations**
  - As long as the attributes are in different relations
  - Must **qualify** the attribute name with the relation name to prevent ambiguity



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



```
Q1′:    SELECT   EMPLOYEE.Fname, EMPLOYEE.LName,
                 EMPLOYEE.Address
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    DEPARTMENT.DName = ‘Research’ AND
                 DEPARTMENT.Dnumber = EMPLOYEE.Dno;
```

Aliasing, and Renaming

- **Aliases or tuple variables**
  - Declare alternative relation names `E` and `S` to refer to the `EMPLOYEE` relation twice in a query:

Query 8:

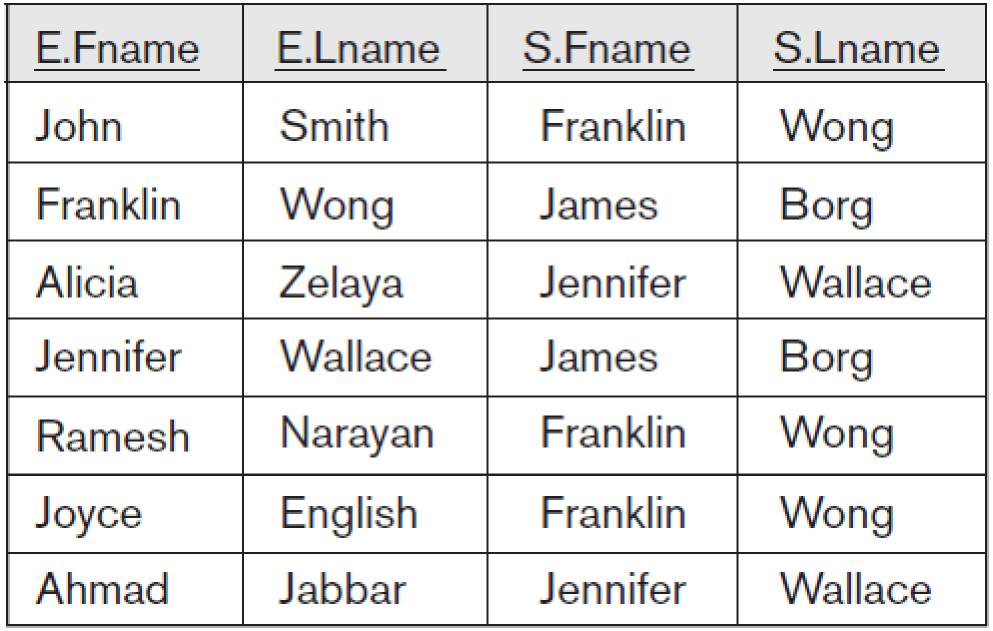
For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

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

  - Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables.

Aliasing, and Renaming

```
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

Aliasing, Renaming and Tuple Variables (contd.)

– The attribute names can also be renamed

```
EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal,
Sssn, Dno)
```

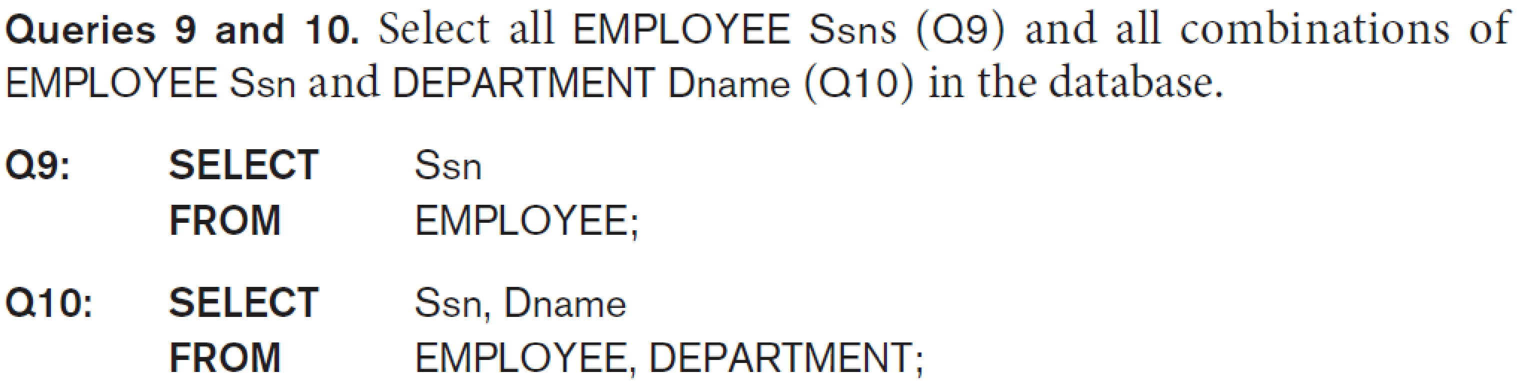
– Note that the relation `EMPLOYEE` now has a variable name `E` which corresponds to a tuple variable

– The “`AS`” may be dropped in most SQL implementations

Unspecified WHERE Clause and Use of the Asterisk

- **Missing WHERE clause**
  - Indicates no condition on tuple selection

- **Effect is a CROSS PRODUCT**
  - Result is all possible tuple combinations (or the Algebra operation of Cartesian Product– see Ch.8) result



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;
```

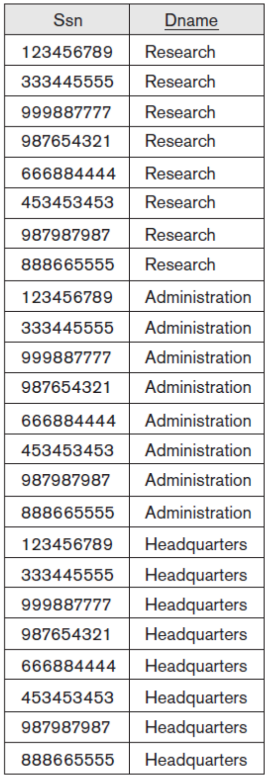
Unspecified `WHERE` Clause and Use of the Asterisk

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;
```

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

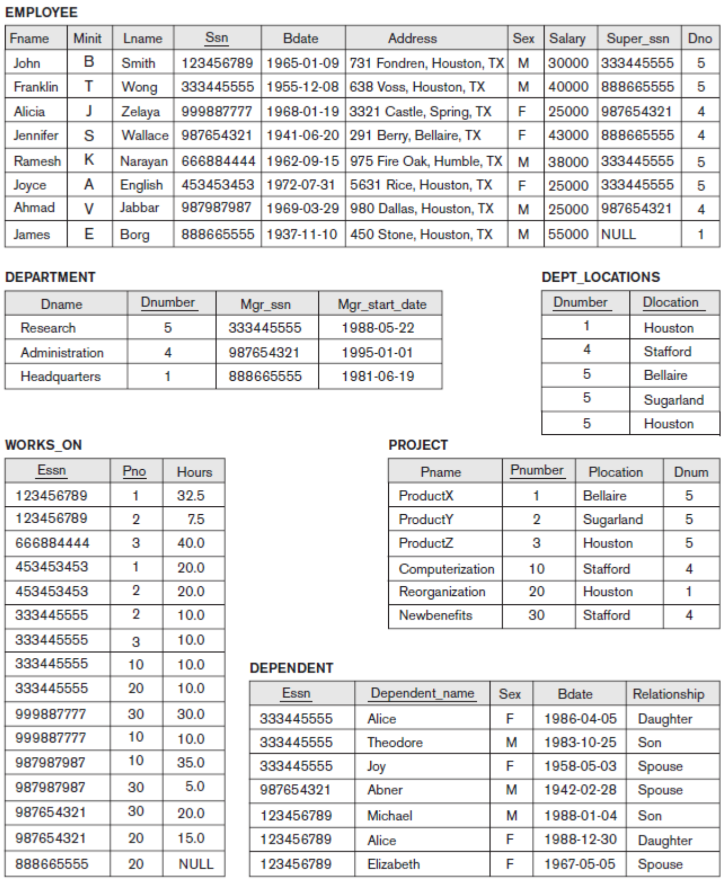


Figure 5.6
One possible database state for the COMPANY relational database schema.

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

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

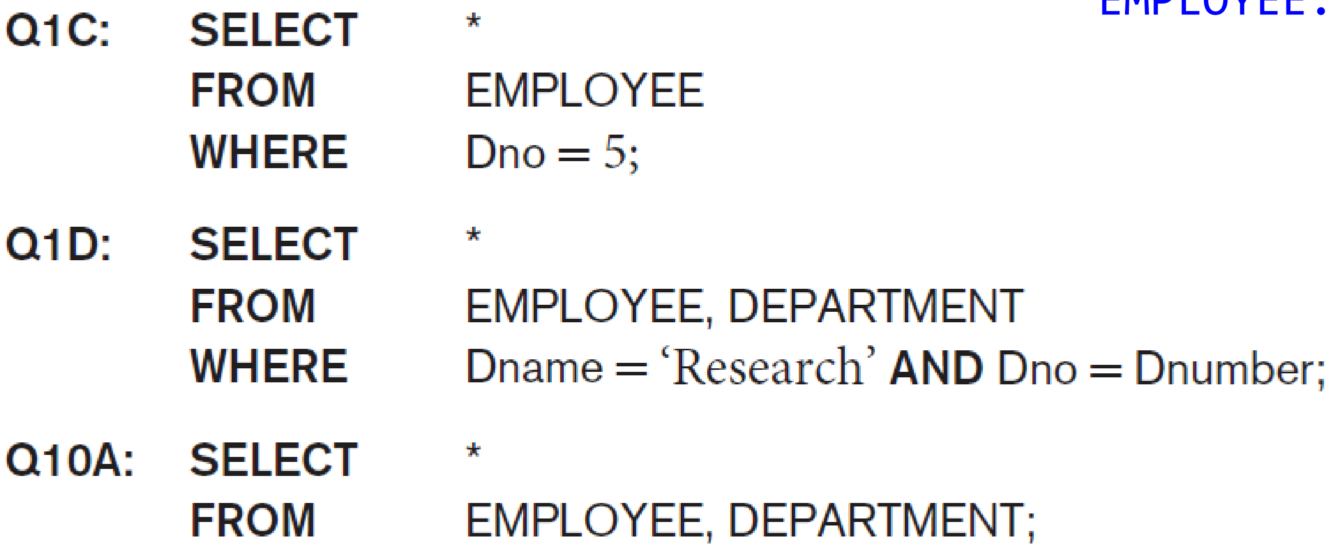
DEPENDENT

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

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)

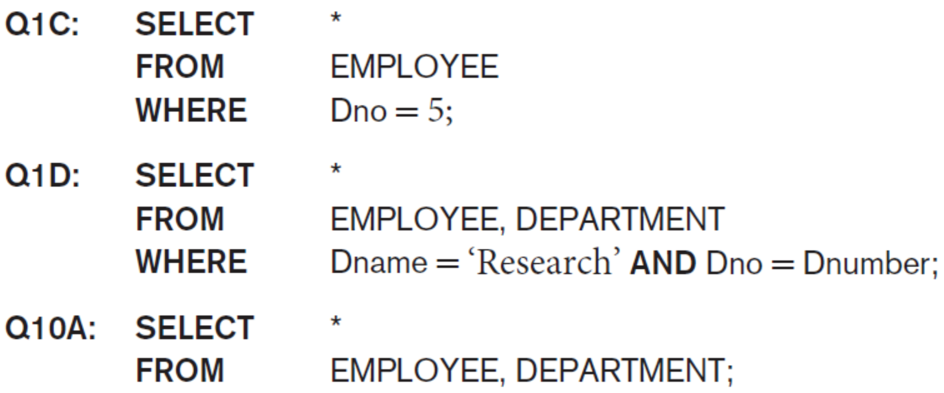
- **Specify an asterisk (*)**
  - Retrieve all the attribute values of the selected tuples
  - The * can be prefixed by the relation name; e.g., EMPLOYEE *

EMPLOYEE.*

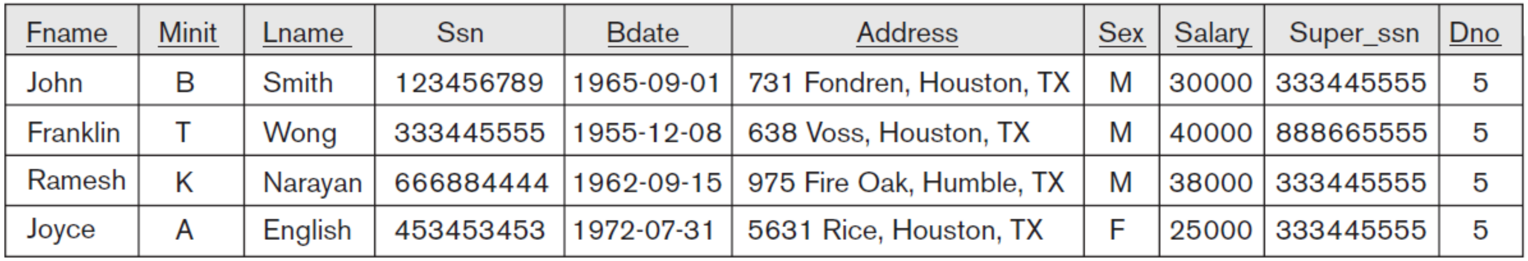


```
Q1C:    SELECT   *
        FROM     EMPLOYEE
        WHERE    Dno = 5;

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

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

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)



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

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

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

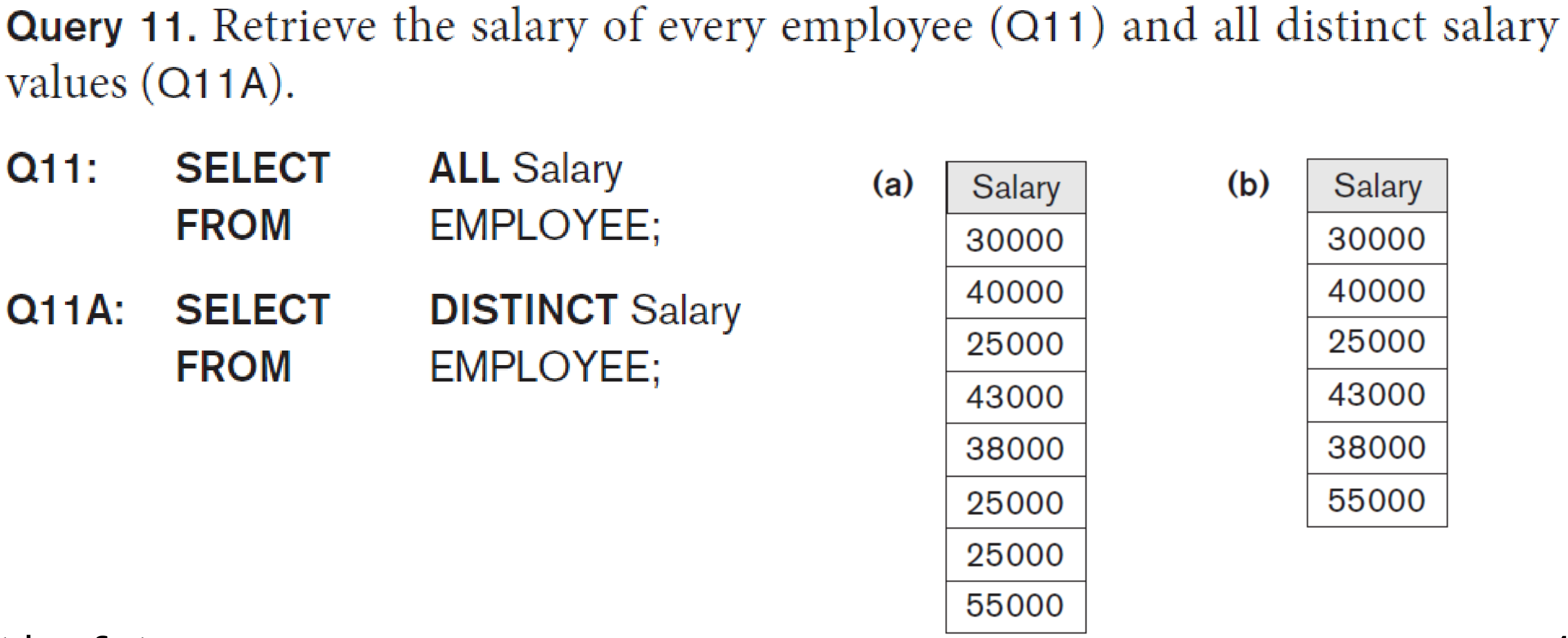


Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

Tables as Sets in SQL

- SQL does not automatically eliminate duplicate tuples in query results

- Use the keyword **DISTINCT** in the `SELECT` clause
  - Only distinct tuples should remain in the result



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;
```

(a)

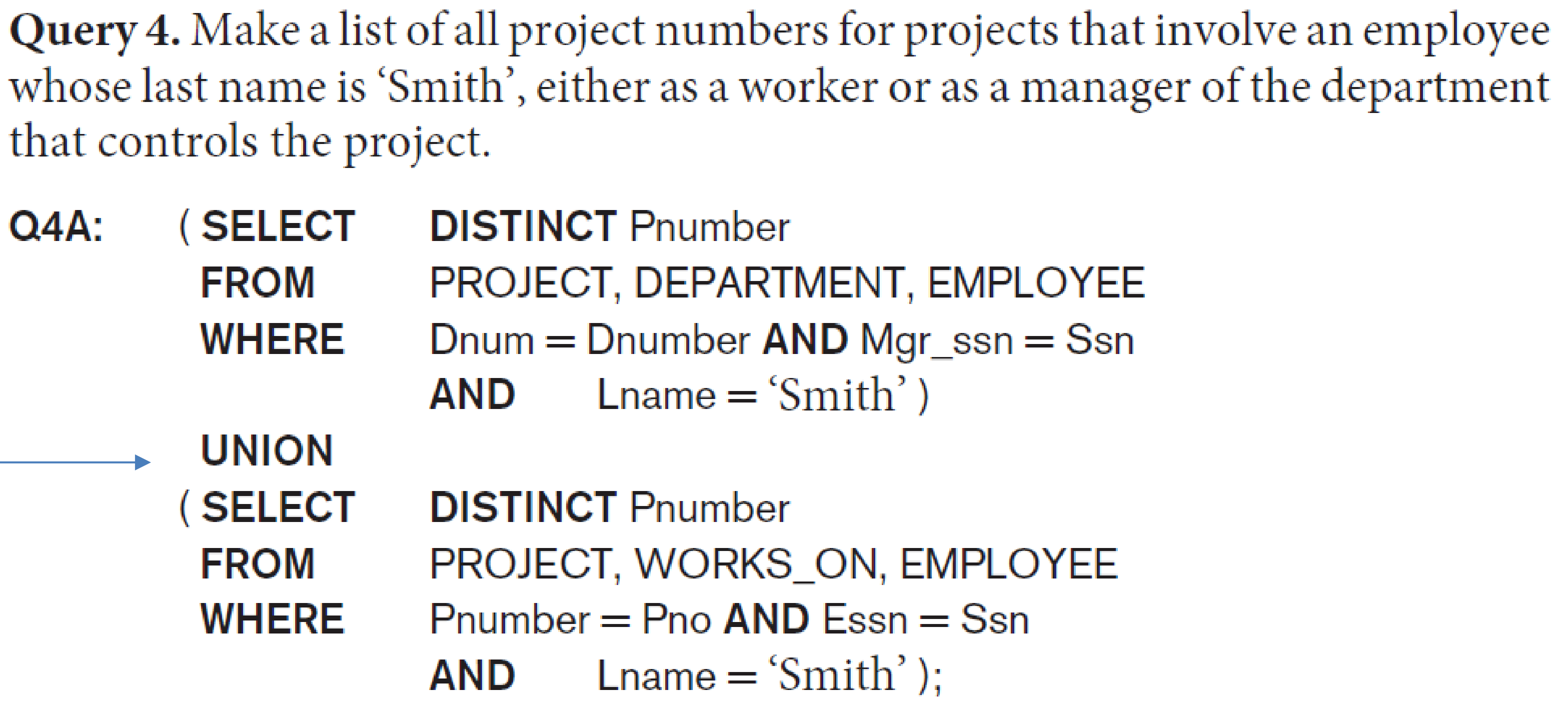
Salary
30000
40000
25000
43000
38000
25000
25000
55000

(b)

Salary
30000
40000
25000
43000
38000
55000

Tables as Sets in SQL (cont'd.)

- **Set operations**
  - **UNION**, **EXCEPT** (difference), **INTERSECT**
  - Type compatibility is needed for these operations to be valid



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' );
```

Tables as Sets in SQL (cont'd.)

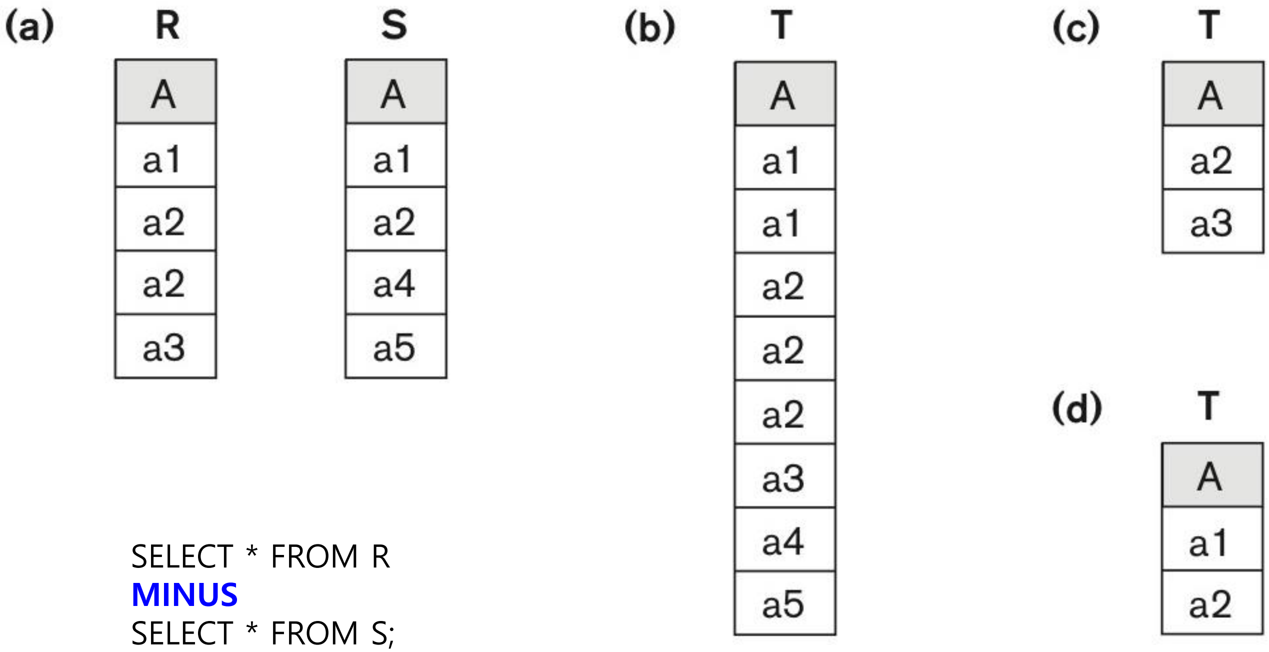
- **Set operations**
  - Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)

```
SELECT * FROM R
UNION
SELECT * FROM S;
```

```
SELECT * FROM R
UNION ALL
SELECT * FROM S;
```

```
SELECT * FROM R
INTERSECT
SELECT * FROM S;
```

```
SELECT * FROM R
INTERSECT ALL
SELECT * FROM S;
```



(a) R

A
a1
a2
a2
a3

S

A
a1
a2
a4
a5

(b) T

A
a1
a1
a2
a2
a2
a3
a4
a5

(c) T

A
a2
a3

(d) T

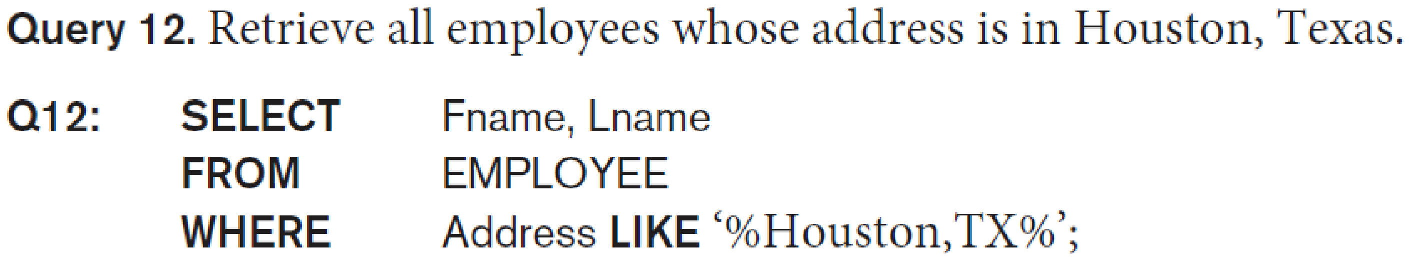
A
a1
a2

```
SELECT * FROM R
MINUS
SELECT * FROM S;
```

Figure 6.5 The results of SQL multiset operations. (a) Two tables, R(A) and S(A). (b) R(A) UNION ALL S(A). (c) R(A) EXCEPT ALL S(A). (d) R(A) INTERSECT ALL S(A).

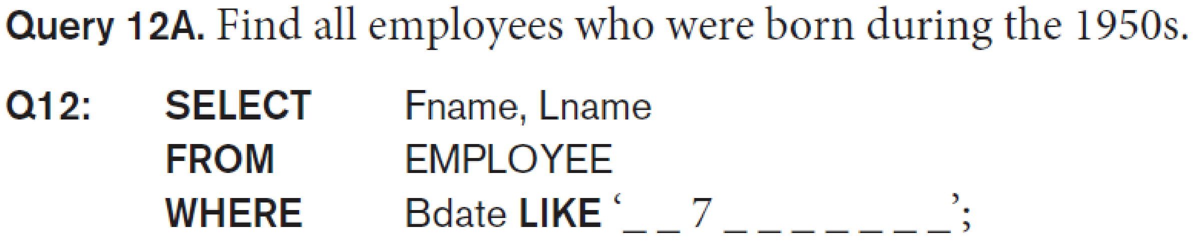
Substring Pattern Matching and Arithmetic Operators

- **LIKE comparison operator**
  - Used for string **pattern matching**
  - % replaces an arbitrary number of zero or more characters
  - underscore (_) replaces a single character
  - Examples: **WHERE** Address **LIKE** '%Houston,TX%';
  - **WHERE** Ssn **LIKE** '_ _ 1_ _ 8901';



Query 12. Retrieve all employees whose address is in Houston, Texas.

```
Q12:   SELECT   Fname, Lname
       FROM     EMPLOYEE
       WHERE    Address LIKE '%Houston,TX%';
```



Query 12A. Find all employees who were born during the 1950s.

```
Q12:   SELECT   Fname, Lname
       FROM     EMPLOYEE
       WHERE    Bdate LIKE '_ _ 7 _ _ _ _ _ _ _';
```

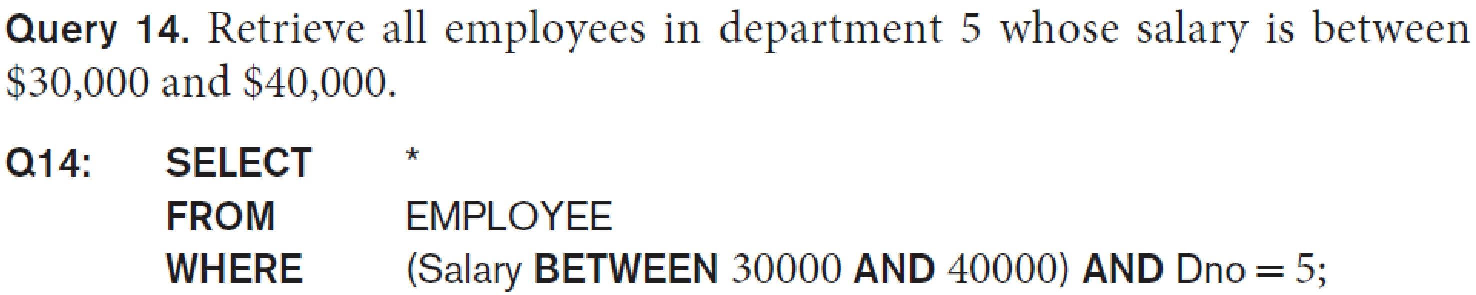
Substring Pattern Matching and Arithmetic Operators

```
mysql> select employees.* from employees where last_name like '%mm%' limit 5;
+--------+------------+------------+--------------+--------+------------+
| emp_no | birth_date | first_name | last_name    | gender | hire_date  |
+--------+------------+------------+--------------+--------+------------+
|  10002 | 1964-06-02 | Bezalel    | Simmel       | F      | 1985-11-21 |
|  10301 | 1962-08-26 | Lucien     | Staudhammer  | M      | 1988-05-23 |
|  10353 | 1953-01-15 | Phule      | Hammerschmidt | M     | 1989-08-24 |
|  10436 | 1963-06-17 | Yahiko     | Lammel       | M      | 1988-01-30 |
|  10777 | 1955-08-11 | Mizuhito   | Kemmerer     | F      | 1988-07-05 |
+--------+------------+------------+--------------+--------+------------+
5 rows in set (0.00 sec)
```

```
mysql> select employees.* from employees where last_name like '_a%' limit 5;
+--------+------------+------------+-------------+--------+------------+
| emp_no | birth_date | first_name | last_name   | gender | hire_date  |
+--------+------------+------------+-------------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello     | M      | 1986-06-26 |
|  10003 | 1959-12-03 | Parto      | Bamford     | M      | 1986-08-28 |
|  10005 | 1955-01-21 | Kyoichi    | Maliniak    | M      | 1989-09-12 |
|  10008 | 1958-02-19 | Saniya     | Kalloufi    | M      | 1994-09-15 |
|  10016 | 1961-05-02 | Kazuhito   | Cappelletti | M      | 1995-01-27 |
+--------+------------+------------+-------------+--------+------------+
5 rows in set (0.00 sec)
```

Substring Pattern Matching and Arithmetic Operators

- **BETWEEN comparison operator**



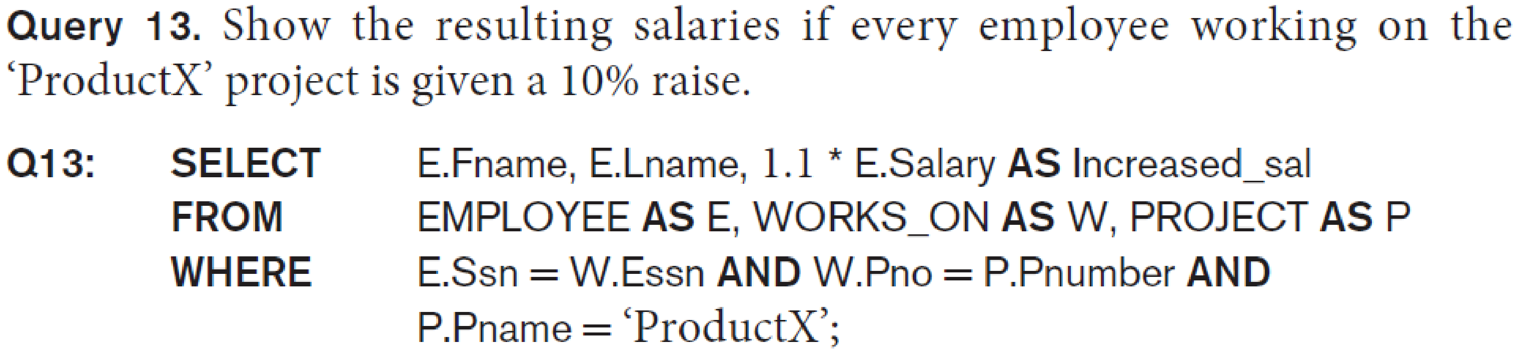
Query 14. Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

```
Q14:   SELECT    *
       FROM      EMPLOYEE
       WHERE     (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

```
mysql> SELECT E.first_name, S.salary  FROM employees E,
salaries S WHERE E.emp_no = S.emp_no AND S.salary BETWEEN
50000 AND 60000 LIMIT 5;
+------------+--------+
| first_name | salary |
+------------+--------+
| Chirstian  |  50594 |
| Chirstian  |  52119 |
| Chirstian  |  54693 |
| Chirstian  |  58326 |
| Anneke     |  52255 |
+------------+--------+
5 rows in set (0.00 sec)
```

Arithmetic Operations

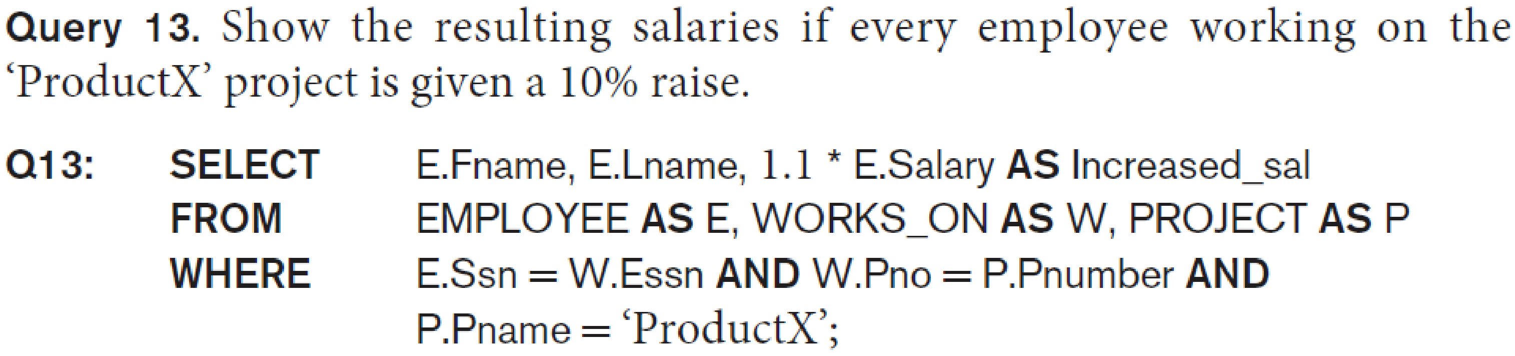
- **Standard arithmetic operators:**
  - Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**



Query 13. Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.

```
Q13:    SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
        FROM    EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
        WHERE   E.Ssn = W.Essn AND W.Pno = P.Pnumber AND
                P.Pname = 'ProductX';
```

Arithmetic Operations



Query 13. Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.

```
Q13:    SELECT    E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
        FROM      EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
        WHERE     E.Ssn = W.Essn AND W.Pno = P.Pnumber AND
                  P.Pname = 'ProductX';
```

```
mysql> select E.first_name, S.salary, 1.1*S.salary as
increased_salary from employees E, salaries S where E.emp_no =
S.emp_no limit 5;
+------------+--------+------------------+
| first_name | salary | increased_salary |
+------------+--------+------------------+
| Georgi     |  60117 |          66128.7 |
| Georgi     |  62102 |          68312.2 |
| Georgi     |  66074 |          72681.4 |
| Georgi     |  66596 |          73255.6 |
| Georgi     |  66961 |          73657.1 |
+------------+--------+------------------+
5 rows in set (0.00 sec)
```

Ordering of Query Results

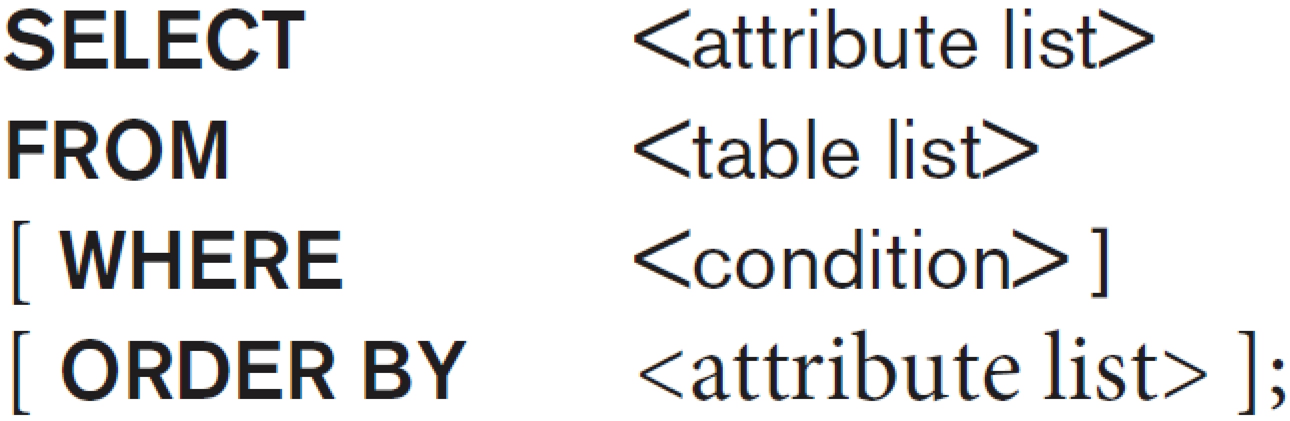
- **Use `ORDER BY` clause**
  - Keyword **`DESC`** to see result in a descending order of values
  - Keyword **`ASC`** to specify ascending order explicitly
  - Typically placed at the end of the query

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

```
mysql> SELECT E.first_name, S.salary  FROM
employees E, salaries S WHERE E.emp_no =
S.emp_no AND S.salary ORDER BY S.salary LIMIT 5;
+------------+--------+
| first_name | salary |
+------------+--------+
| Olivera    |  38623 |
| Fumiya     |  38735 |
| Chuanyi    |  38786 |
| Yurij      |  38812 |
| Mechthild  |  38836 |
+------------+--------+
```

```
mysql> SELECT E.first_name, S.salary  FROM
employees E, salaries S WHERE E.emp_no = S.emp_no
AND S.salary ORDER BY S.salary DESC LIMIT 5;
+------------+--------+
| first_name | salary |
+------------+--------+
| Tokuyasu   | 158220 |
| Tokuyasu   | 157821 |
| Honesty    | 156286 |
| Xiahua     | 155709 |
| Sanjai     | 155513 |
+------------+--------+
```

Basic SQL Retrieval Query Block



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

INSERT, DELETE, and UPDATE Statements in SQL

INSERT, DELETE, UPDATE Statements in SQL

- **Three commands used to modify the database:**
  - INSERT, DELETE, and UPDATE

- **INSERT typically inserts a tuple (row) in a relation (table)**

- **UPDATE may update a number of tuples (rows) in a relation (table) that satisfy the condition**

- **DELETE may also update a number of tuples (rows) in a relation (table) that satisfy the condition**

INSERT

- **In its simplest form, it is used to add one or more tuples to a relation**

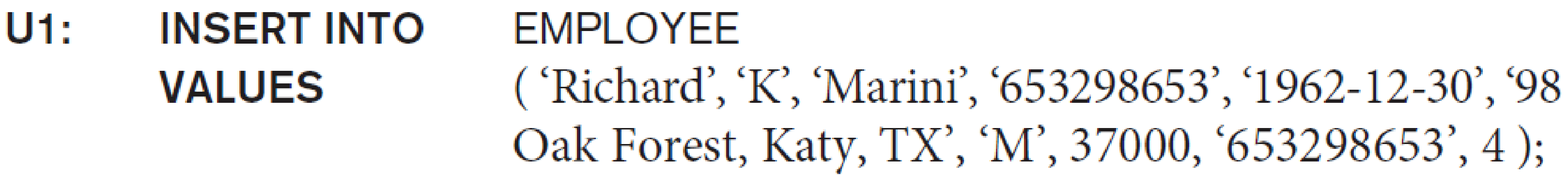
- **Attribute values should be listed in the same order as the attributes were specified in the `CREATE TABLE` command**

- **Constraints on data types are observed automatically**

- **Any integrity constraints as a part of the DDL specification are enforced**

The INSERT Command

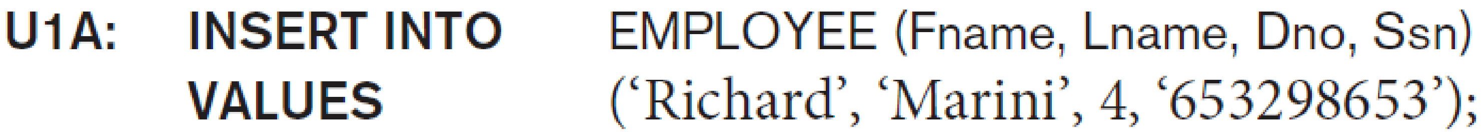
- **Specify the relation name and a list of values for the tuple. All values including nulls are supplied.**



U1: INSERT INTO EMPLOYEE
VALUES (‘Richard’, ‘K’, ‘Marini’, ‘653298653’, ‘1962-12-30’, ‘98 Oak Forest, Katy, TX’, ‘M’, 37000, ‘653298653’, 4);

```
mysql> INSERT INTO EMPLOYEE
       VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest,
Katy, TX', 'M', 37000, '653298653', 4);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * from EMPLOYEE;                                                  +----
-----+-------+--------+-----------+------------+-------------------------+------+----
------+-----------+------+
| Fname   | Minit | Lname  | Ssn       | Bdata      | Address                 | Sex
| Salary   | Super_ssn | Dno  |
+---------+-------+--------+-----------+------------+-------------------------+------
+----------+-----------+------+
| Richard | K     | Marini | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX | M
| 37000.00 | 653298653 |    4 |
+---------+-------+--------+-----------+------------+-------------------------+------
+----------+-----------+------+
```

The INSERT Command

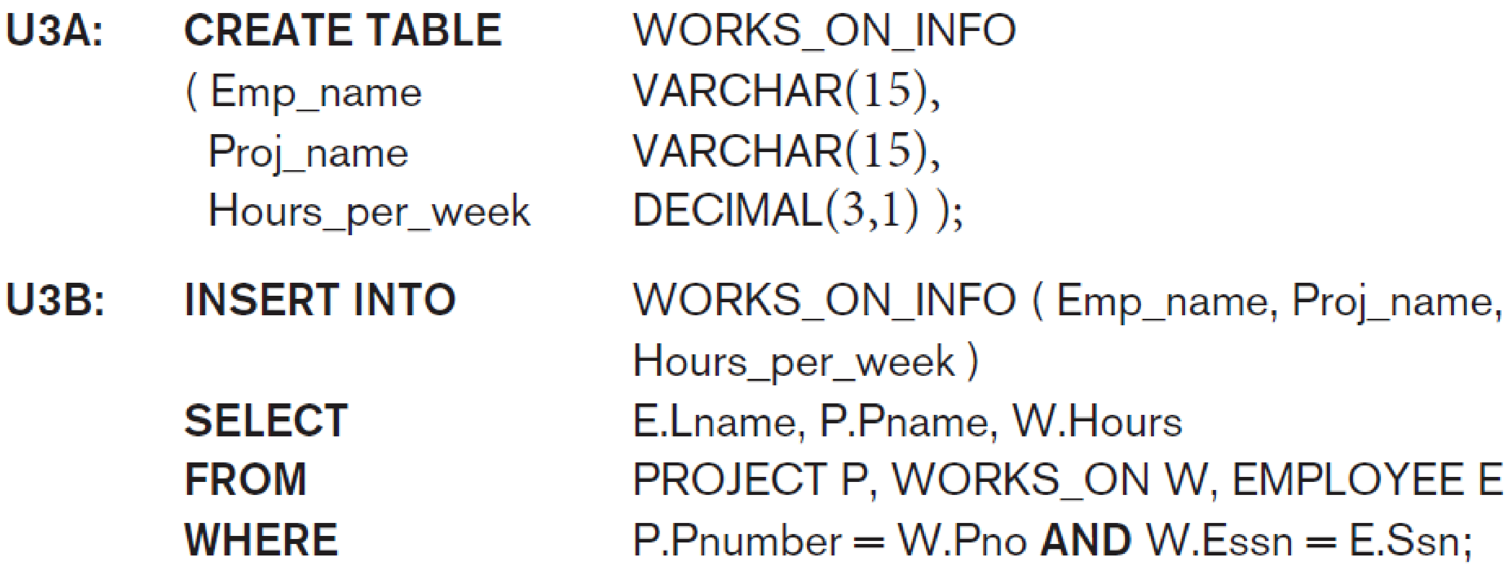


U1A: INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
VALUES (‘Richard’, ‘Marini’, 4, ‘653298653’);

```
mysql> INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
    -> VALUES ('Edgar', 'Codd', 1, '111111111');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> SELECT * from EMPLOYEE;
+---------+-------+--------+-----------+------------+-------------------------+------
+----------+-----------+------+
| Fname   | Minit | Lname  | Ssn       | Bdata      | Address                 | Sex
| Salary   | Super_ssn | Dno  |
+---------+-------+--------+-----------+------------+-------------------------+------
+----------+-----------+------+
| Edgar   | NULL  | Codd   | 111111111 | NULL       | NULL                    | NULL
|     NULL | NULL      |    1 |
| Richard | K     | Marini | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX | M
| 37000.00 | 653298653 |    4 |
+---------+-------+--------+-----------+------------+-------------------------+------
+----------+-----------+------+
2 rows in set (0.00 sec)
```

The `INSERT` Command

- **The variation below inserts multiple tuples where a new table is loaded values from the result of a query.**



```
U3A:    CREATE TABLE        WORKS_ON_INFO
        ( Emp_name          VARCHAR(15),
          Proj_name         VARCHAR(15),
          Hours_per_week    DECIMAL(3,1) );

U3B:    INSERT INTO         WORKS_ON_INFO ( Emp_name, Proj_name,
                            Hours_per_week )
        SELECT              E.Lname, P.Pname, W.Hours
        FROM                PROJECT P, WORKS_ON W, EMPLOYEE E
        WHERE               P.Pnumber = W.Pno AND W.Essn = E.Ssn;
```

Bulk Loading of Tables

- Another variation of `INSERT` is used for bulk-loading of several tuples into tables

- A new table `TNEW` can be created with the same attributes as T and using `LIKE` and `DATA` in the syntax, it can be loaded with entire data.

- EXAMPLE:

```
CREATE TABLE D5EMPS  LIKE  EMPLOYEE
   (      SELECT        E.*
          FROM          EMPLOYEE AS E
          WHERE         E.Dno=5)
WITH DATA;
```

Bulk Loading of Tables

```
mysql> use employees;
mysql> show tables;
+----------------------+
| Tables_in_employees  |
+----------------------+
| current_dept_emp     |
| departments          |
| dept_emp             |
| dept_emp_latest_date |
| dept_manager         |
| employees            |
| salaries             |
| titles               |
+----------------------+
mysql> CREATE TABLE D001EMP LIKE dept_emp;
Query OK, 0 rows affected (0.06 sec)
mysql> show tables;
+----------------------+
| Tables_in_employees  |
+----------------------+
| D001EMP              |
| current_dept_emp     |
| departments          |
| dept_emp             |
| dept_emp_latest_date |
| dept_manager         |
| employees            |
| salaries             |
| titles               |
+----------------------+
```

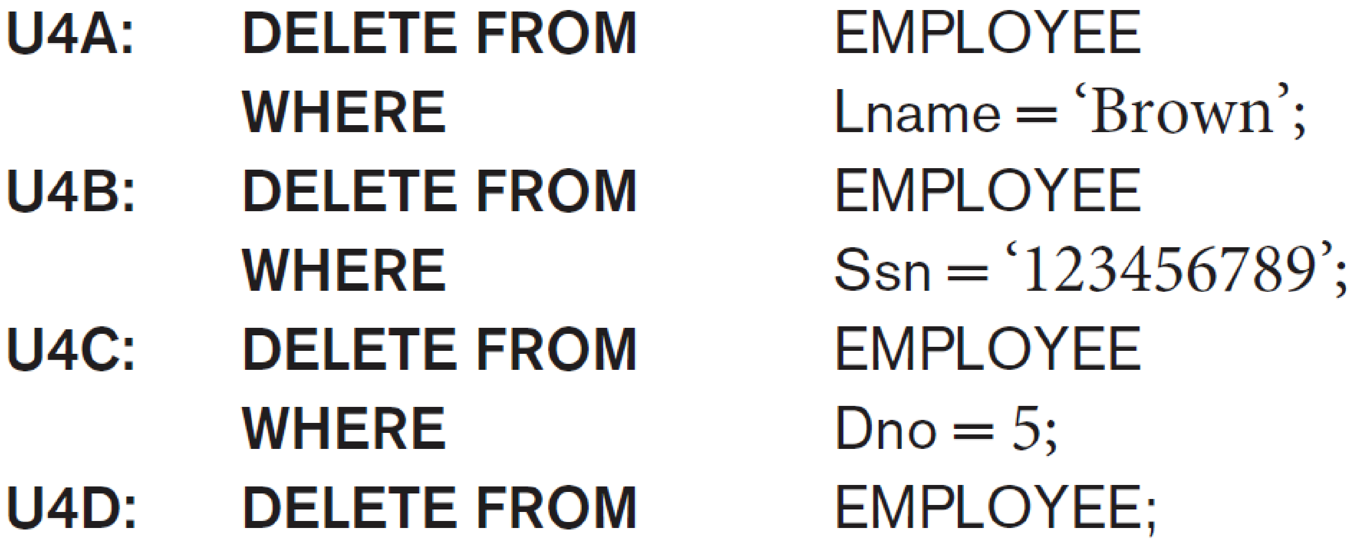
```
mysql> SELECT * from D001EMP;
Empty set (0.00 sec)

mysql> INSERT INTO D001EMP (SELECT * from dept_emp);
Query OK, 331603 rows affected (2.52 sec)
Records: 331603  Duplicates: 0  Warnings: 0

mysql> SELECT * from D001EMP LIMIT 5;
+--------+---------+------------+------------+
| emp_no | dept_no | from_date  | to_date    |
+--------+---------+------------+------------+
|  10001 | d005    | 1986-06-26 | 9999-01-01 |
|  10002 | d007    | 1996-08-03 | 9999-01-01 |
|  10003 | d004    | 1995-12-03 | 9999-01-01 |
|  10004 | d004    | 1986-12-01 | 9999-01-01 |
|  10005 | d003    | 1989-09-12 | 9999-01-01 |
+--------+---------+------------+------------+
5 rows in set (0.00 sec)

mysql>
```

DELETE

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

The DELETE Command

- **Removes tuples from a relation**
  - Includes a WHERE clause to select the tuples to be deleted. The number of tuples deleted will vary.



```
U4A:    DELETE FROM     EMPLOYEE
        WHERE           Lname = ‘Brown’;
U4B:    DELETE FROM     EMPLOYEE
        WHERE           Ssn = ‘123456789’;
U4C:    DELETE FROM     EMPLOYEE
        WHERE           Dno = 5;
U4D:    DELETE FROM     EMPLOYEE;
```

The DELETE Command

```
mysql> select * from employees limit 5;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
|  10002 | 1964-06-02 | Bezalel    | Simmel    | F      | 1985-11-21 |
|  10003 | 1959-12-03 | Parto      | Bamford   | M      | 1986-08-28 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |
|  10005 | 1955-01-21 | Kyoichi    | Maliniak  | M      | 1989-09-12 |
+--------+------------+------------+-----------+--------+------------+
5 rows in set (0.01 sec)

mysql> delete from employees where first_name = 'Georgi';
Query OK, 253 rows affected (0.25 sec)

mysql> select * from employees limit 5;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10002 | 1964-06-02 | Bezalel    | Simmel    | F      | 1985-11-21 |
|  10003 | 1959-12-03 | Parto      | Bamford   | M      | 1986-08-28 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |
|  10005 | 1955-01-21 | Kyoichi    | Maliniak  | M      | 1989-09-12 |
|  10006 | 1953-04-20 | Anneke     | Preusig   | F      | 1989-06-02 |
+--------+------------+------------+-----------+--------+------------+
```

`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 specified as part of DDL specification is enforced**

UPDATE (contd.)

- **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
```

UPDATE (contd.)

- **Example: Give all employees in the 'Research' department a 10% raise in salary.**

```
U6: UPDATE  EMPLOYEE
    SET     SALARY = SALARY *1.1
    WHERE   DNO  IN (  SELECT     DNUMBER
                       FROM       DEPARTMENT
                       WHERE      DNAME='Research')
```

- **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

UPDATE (contd.)

```
mysql> select * from salaries limit 5;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
|  10002 |  65828 | 1996-08-03 | 1997-08-03 |
|  10002 |  65909 | 1997-08-03 | 1998-08-03 |
|  10002 |  67534 | 1998-08-03 | 1999-08-03 |
|  10002 |  69366 | 1999-08-03 | 2000-08-02 |
|  10002 |  71963 | 2000-08-02 | 2001-08-02 |
+--------+--------+------------+------------+
5 rows in set (0.00 sec)
```

```
mysql> UPDATE salaries  SET salary = salary * 1.1
WHERE emp_no = 10002;
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0

mysql> select * from salaries limit 5;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
|  10002 |  72411 | 1996-08-03 | 1997-08-03 |
|  10002 |  72500 | 1997-08-03 | 1998-08-03 |
|  10002 |  74287 | 1998-08-03 | 1999-08-03 |
|  10002 |  76303 | 1999-08-03 | 2000-08-02 |
|  10002 |  79159 | 2000-08-02 | 2001-08-02 |
+--------+--------+------------+------------+
```

Additional Features of SQL

- Techniques for specifying complex retrieval queries (see Ch.7)
- Writing programs in various programming languages that include SQL statements: Embedded and dynamic SQL, SQL/CLI (Call Level Interface) and its predecessor ODBC, SQL/PSM (Persistent Stored Module) (See Ch.10)
- Set of commands for specifying physical database design parameters, file structures for relations, and access paths, e.g., `CREATE INDEX`

Additional Features of SQL (cont'd.)

- **Transaction control commands (Ch.20)**
- **Specifying the granting (`GRANT`) and revoking of privileges (`REVOKE`)to users (Ch.30)**
- **Constructs for creating triggers (Ch.26)**
- **Enhanced relational systems known as object-relational define relations as classes. Abstract data types (called User Defined Types- UDTs) are supported with `CREATE TYPE`**

Summary

- **SQL**
  - A Comprehensive language for relational database management
  - Data definition, queries, updates, constraint specification, and view definition
  - https://dev.mysql.com/doc/refman/8.0/en/sql-syntax.html

- **Covered :**
  - Data definition commands for creating tables
  - Commands for constraint specification
  - Simple retrieval queries
  - Database update commands