Lecture 4

Basic SQL

Agenda

- Introduction to SQL
- SQL Data Definition and Data Types
- Retrieval Queries in SQL

Introduction to SQL

Introduction to SQL

- The name **SQL** is presently expanded as **S**tructured **Q**uery **L**anguage.

- SQL is now the **standard language** for commercial relational DBMSs.

- The standardization of SQL is a joint effort by the American National Standards Institute (ANSI) and the International Standards Organization (ISO).

Introduction to SQL

- SQL is a comprehensive database language: It has statements for **data definitions**, **queries**, and **updates**. Hence, it is both a DDL *and* a DML.
- In addition, it has facilities for **defining views** on the database, for **specifying security and authorization**, for **defining integrity constraints**, and for **specifying transaction controls**.
- It also has rules for **embedding SQL statements** into a general-purpose programming language such as Java or C/C++.

SQL Data Definition and Data Types

CREATE TABLE

- Specifies a new relation by giving it:
  - a name &
  - specifying each of its attributes &
  - their data types (*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             INTEGER     NOT NULL,
   MGRSSN              CHAR(9),
   MGRSTARTDATE         CHAR(9)   );
```

CREATE TABLE

- Can use the CREATE TABLE command for specifying:
  - Primary Key attributes,
  - Secondary Keys, &
  - Referential Integrity Constraints (Foreign Keys).

```
CREATE TABLE DEPT (
 DNAME               VARCHAR(10)    NOT NULL,
 DNUMBER              INTEGER       NOT NULL,
 MGRSSN               CHAR(9),
 MGRSTARTDATE        CHAR(9),
 PRIMARY KEY (DNUMBER),
 UNIQUE (DNAME),
 FOREIGN KEY (MGRSSN) REFERENCES EMP   );
```

DROP TABLE

- Used to remove a relation (base table) and its definition
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

```
DROP  TABLE   DEPENDENT;
```

ALTER TABLE

- Used to add an attribute to one of the base relations
  - The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute.
- Example:

```
 ALTER TABLE EMPLOYEE ADD JOB
VARCHAR(12);
```

- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple.
  - This can be done using the UPDATE command.

REFERENTIAL INTEGRITY OPTIONS

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

```
CREATE TABLE DEPT (
  DNAME          VARCHAR(10) NOT NULL,
  DNUMBER        INTEGER     NOT NULL,
  MGRSSN         CHAR(9),
  MGRSTARTDATE   CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP
    ON DELETE SET DEFAULT
    ON UPDATE CASCADE);
```

REFERENTIAL INTEGRITY OPTIONS

```
CREATE TABLE EMP(
  ENAME       VARCHAR(30)   NOT NULL,
  ESSN        CHAR(9),
  BDATE       DATE,
  DNO         INTEGER  DEFAULT 1,
  SUPERSSN    CHAR(9),
  PRIMARY KEY (ESSN),
  FOREIGN KEY (DNO) REFERENCES DEPT(DNUMBER)
     ON DELETE SET DEFAULT
     ON UPDATE CASCADE,
  FOREIGN KEY (SUPERSSN) REFERENCES EMP (ESSN)
     ON DELETE SET NULL
     ON UPDATE CASCADE);
```

Additional Data Types

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):**

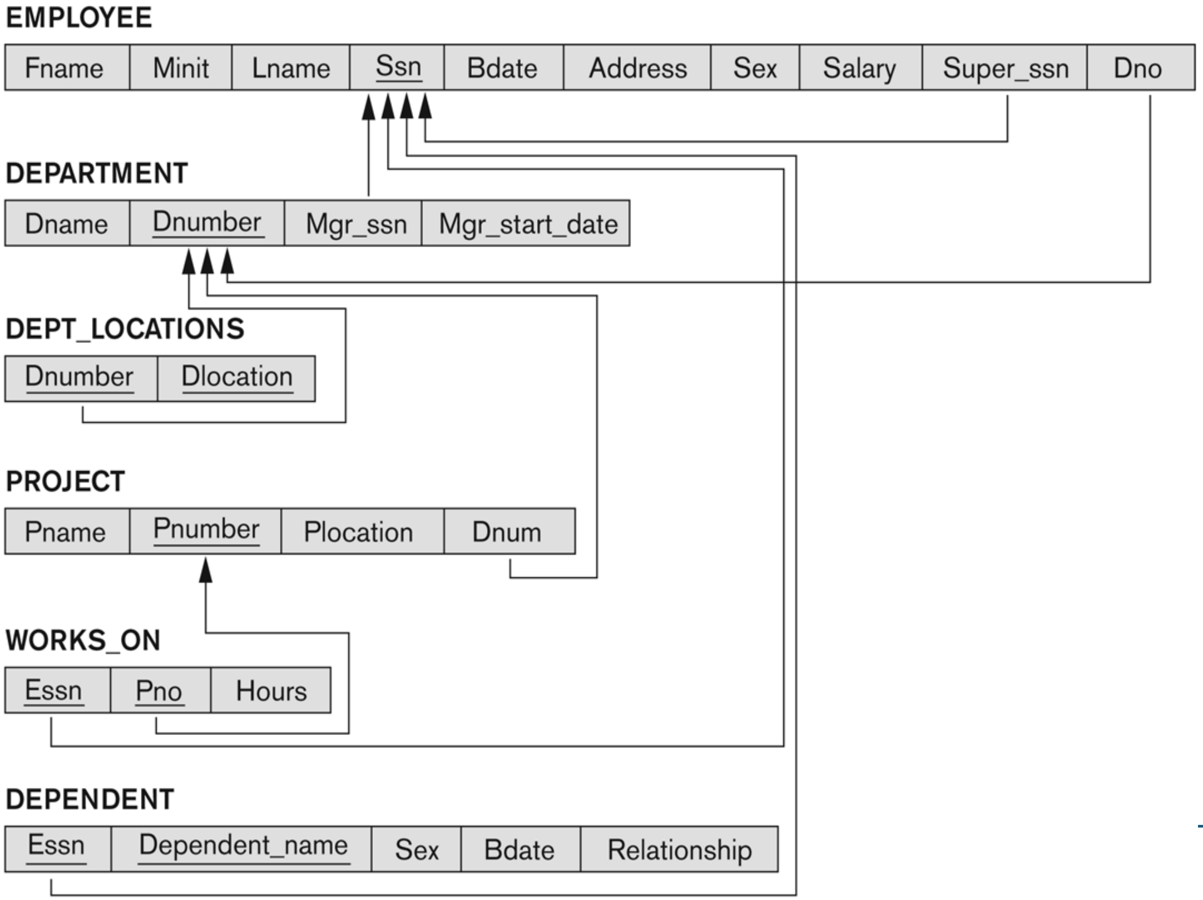
- Made up of hour:minute:second plus i additional digits specifying fractions of a second
- format is hh:mm:ss:ii...i

➢ **TIMESTAMP:**

- Has both DATE and TIME components

Company Relational Database Schema Used for examples in these slides

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



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

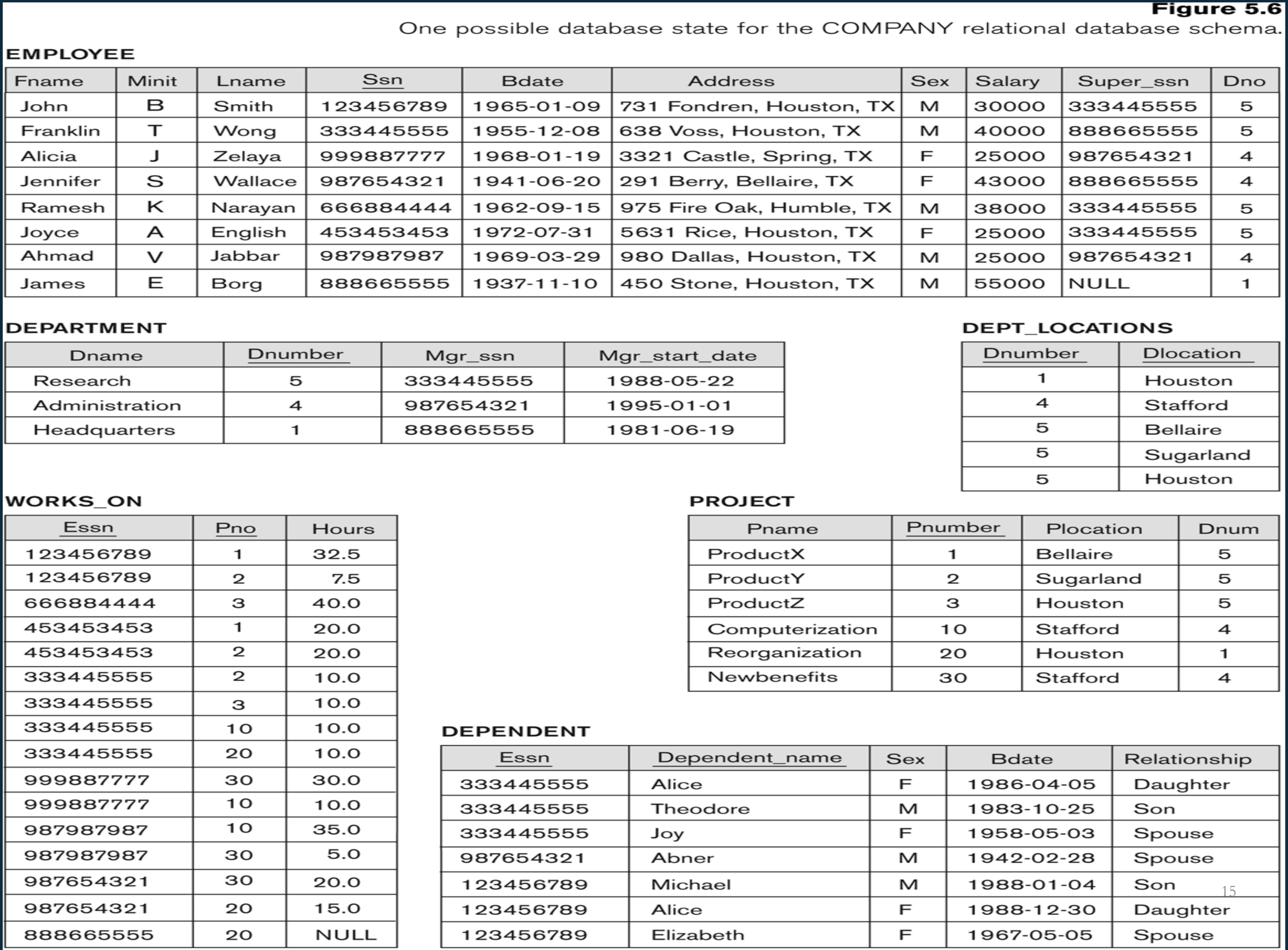


Figure 5.6
One possible database state for the COMPANY relational database schema.

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

Retrieval Queries in SQL

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the **SELECT** statement.
- Basic form of the SQL SELECT statement is called 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

SQL Queries

- 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’
```

- Note: The result of the query may contain duplicate tuples

SQL Queries

- **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
```

SQL Queries

- **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.

```
Q2: SELECT    PNUMBER, DNUM, LNAME, BDATE, ADDRESS
    FROM      PROJECT, DEPARTMENT, EMPLOYEE
    WHERE     DNUM=DNUMBER AND MGRSSN=SSN
              AND PLOCATION='Stafford'
```

- 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

Understanding JOIN

```
Select    *
From      PROJECT , DEPARTMENT
where     DNUM=DNUMBER
```

Join Result:

Pname	Pnumber	Plocation	Dnum	Dnumber	Dname	Mgr_SSN	Mgr_start_date
Product X	1	Bellaire	5	5	Research	333445555	1988-05-22
Product Y	2	Sugerland	5	5	Research	333445555	1988-05-22
…..							

Using RELATION NAME

- 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
- Example:

- **DEPT_LOCATIONS.**DNUMBER, **DEPARTMENT.**DNUMBER

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 3: Retrieve the SSN values for all employees.

- Q3:

```
SELECT    SSN
FROM      EMPLOYEE
```

UNSPECIFIED WHERE-clause

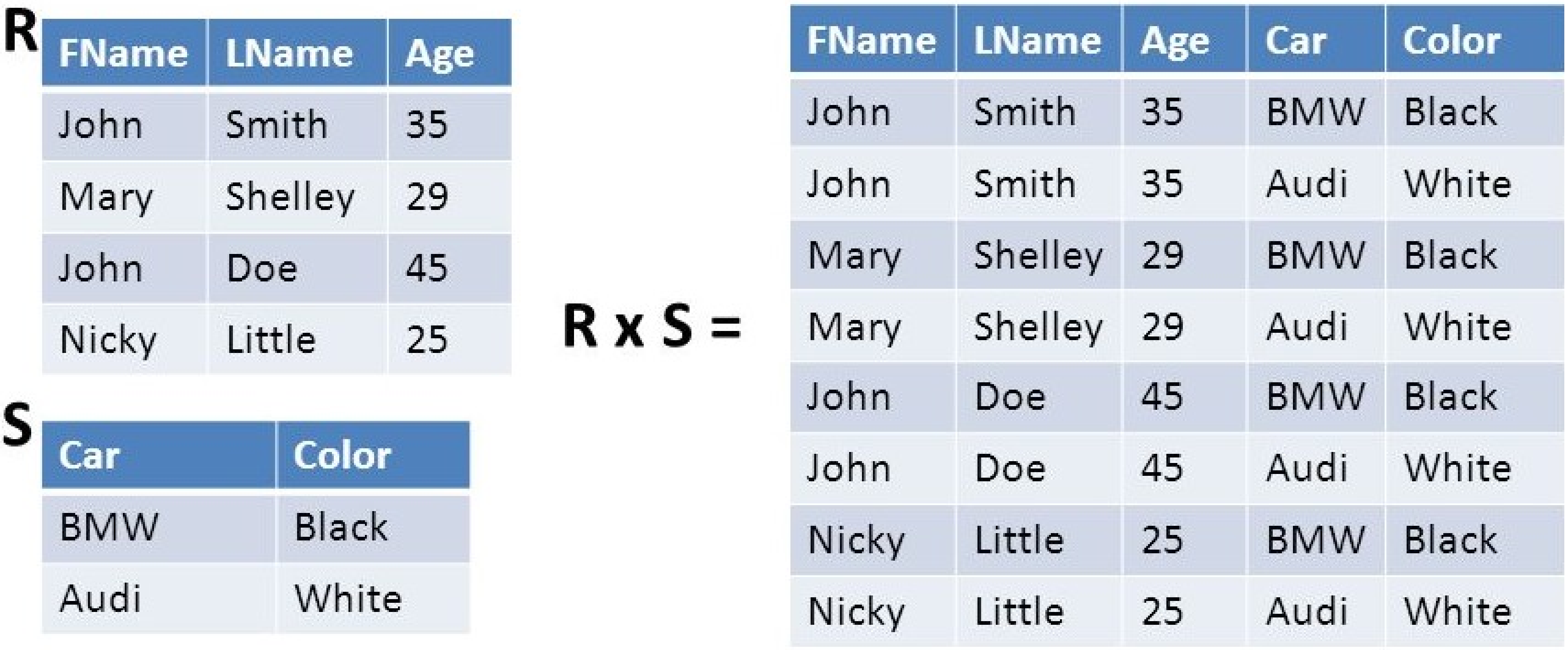
- Example:

```
Q4:     SELECT  SSN, DNAME
        FROM    EMPLOYEE, DEPARTMENT
```

- What is the result of the above Query?
  - 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

UNSPECIFIED WHERE-clause

- ***Cartesian Product:*** All combinations of tuples from the two relations (R x S).



R

FName	LName	Age
John	Smith	35
Mary	Shelley	29
John	Doe	45
Nicky	Little	25

S

Car	Color
BMW	Black
Audi	White

R x S =

FName	LName	Age	Car	Color
John	Smith	35	BMW	Black
John	Smith	35	Audi	White
Mary	Shelley	29	BMW	Black
Mary	Shelley	29	Audi	White
John	Doe	45	BMW	Black
John	Doe	45	Audi	White
Nicky	Little	25	BMW	Black
Nicky	Little	25	Audi	White

USE OF *

- To retrieve all the attribute values of the selected tuples, * is used, which stands for *all the attributes*
- Examples:

```
Q5:     SELECT   *
        FROM     EMPLOYEE
        WHERE    DNO=5

Q6:     SELECT   *
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    DNAME='Research' AND
                 DNO=DNUMBER
```

ALIASES

- Some queries need to refer to the same relation twice
  - In this case, *ALIASES* are given to the relation name
- **Query 7:** For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
Q7:  SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
     FROM     EMPLOYEE E S
     WHERE    E.SUPERSSN=S.SSN
```

  - In Q7, the alternate relation names E and S are called *aliases* 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*

ALIASES

- 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
```

Joining Same Relation

Select *
From EMPLOYEE E S
Where E.SUPERSSN=S.SSN

Join Result:

Fname	Minit	Lname	…	SSN	Super_SSN	Fname	Minit	…
John	B	Smith	…	123456789	333445555	Franklin	T	…
Franklin	T	Wong	…	333445555	888665555	…	….	…
…..								

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 Q9 may have duplicate SALARY values whereas Q10 does not have any duplicate values

```
Q9:     SELECT   SALARY
        FROM     EMPLOYEE

Q10:    SELECT   DISTINCT SALARY
        FROM     EMPLOYEE
```

JOIN TYPES

➢There are different types of join operations that can be applied, such as:

- Natural Join
    - The two attributes to join on should have the same name in both relations.
    - No join condition is added.
    - Only one column of them is kept in the result.
- Left Outer Join
- Right outer Join
- …etc.

JOIN TYPES

- Examples:

```
Q11:    SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
        FROM     EMPLOYEE E S
        WHERE    E.SUPERSSN=S.SSN
```

- If all employee names should be listed this can be written as:

```
Q12:    SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
        FROM     (EMPLOYEE E LEFT OUTER JOIN
                  EMPLOYEE S ON  E.SUPERSSN=S.SSN)
```

JOIN TYPES

- Examples:

```
Q13:    SELECT   FNAME, LNAME, ADDRESS
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    DNAME='Research' AND DNUMBER=DNO
```

- could be written as:

```
Q13:    SELECT   FNAME, LNAME, ADDRESS
        FROM     (EMPLOYEE JOIN DEPARTMENT
                 ON DNUMBER=DNO)
        WHERE    DNAME='Research’
```

- or as:

```
Q13: SELECT   FNAME, LNAME, ADDRESS
     FROM     (EMPLOYEE NATURAL JOIN DEPARTMENT
              AS  DEPT(DNAME, DNO, MSSN, MSDATE)
              WHERE   DNAME='Research’
```

SET OPERATIONS

- SQL has directly incorporated some set operations
  - There is a union operation (UNION),
  - and in *some versions* of SQL (MINUS) and intersection (INTERSECT)
- 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*;
  - the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- **Query 14**: Make a list of all project names 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.

```
Q14:        (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 NAME='Smith')
```

NESTING OF QUERIES

- A complete SELECT query, called a *nested query*, can be specified within the WHERE-clause of another query, called the *outer query*
  - Many of the previous queries can be specified in an alternative form using nesting

- Query 15: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q15:    SELECT   FNAME, LNAME, ADDRESS
        FROM     EMPLOYEE
        WHERE    DNO IN  (SELECT  DNUMBER
                          FROM    DEPARTMENT
                          WHERE   DNAME='Research')
```

NESTING OF QUERIES

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V
- In general, we can have several levels of nested queries
- In this example, the nested query is *not correlated* with the outer query

CORRELATED NESTED QUERIES

- If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query*, the two queries are said to be *correlated*
  - The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) in the outer query
- **Query 16:** Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q16: SELECT      E.FNAME, E.LNAME
     FROM        EMPLOYEE AS E
     WHERE       E.SSN IN
                      ( SELECT   ESSN
                        FROM     DEPENDENT
                        WHERE    ESSN=E.SSN AND
                                 E.FNAME=DEPENDENT_NAME)
```

CORRELATED NESTED QUERIES

- A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can ***always*** be expressed as a single block query. For example, Q16 may be written as in Q17:

```
Q17:    SELECT   E.FNAME, E.LNAME
        FROM     EMPLOYEE E, DEPENDENT D
        WHERE    E.SSN=D.ESSN AND
                 E.FNAME=D.DEPENDENT_NAME
```

THE EXISTS FUNCTION

- EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not
  - We can formulate Query 16 in an alternative form that uses EXISTS as Q18

THE EXISTS FUNCTION

- **Query 18**: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q18:    SELECT    FNAME, LNAME
        FROM      EMPLOYEE
        WHERE     EXISTS  (SELECT     *
                           FROM       DEPENDENT
                           WHERE      SSN=ESSN AND
                          NAME=DEPENDENT_NAME)
```

THE EXISTS FUNCTION

- Query 19: Retrieve the names of employees who have no dependents.

```
Q19: SELECT      FNAME, LNAME
     FROM        EMPLOYEE
     WHERE       NOT EXISTS  (SELECT  *
                         FROM     DEPENDENT
                         WHERE    SSN=ESSN)
```

- In Q19, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If *none exist*, the EMPLOYEE tuple is selected
  - EXISTS is necessary for the expressive power of SQL

EXPLICIT SETS

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- **Query 20:** Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q20:    SELECT  DISTINCT ESSN
        FROM    WORKS_ON
        WHERE   PNO IN  (1, 2, 3)
```

NULLS IN SQL QUERIES

- SQL allows queries that check if a value is **NULL** (missing or undefined or not applicable)
- SQL uses **IS** or **IS NOT** to compare NULLs
- **Query 21:** Retrieve the names of all employees who do not have supervisors.

```
Q21:    SELECT   FNAME, LNAME
        FROM     EMPLOYEE
        WHERE    SUPERSSN  IS  NULL
```

Thank You