KOÇ UNIVERSITY

Database Management Systems

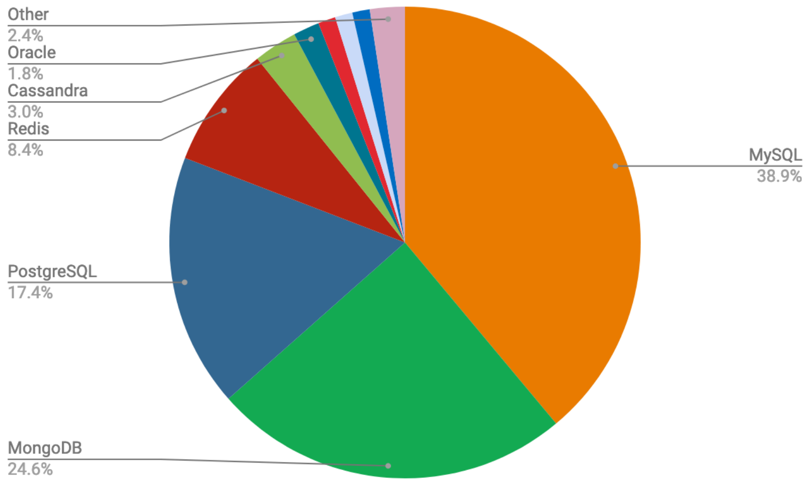
Structured Query Language (SQL)

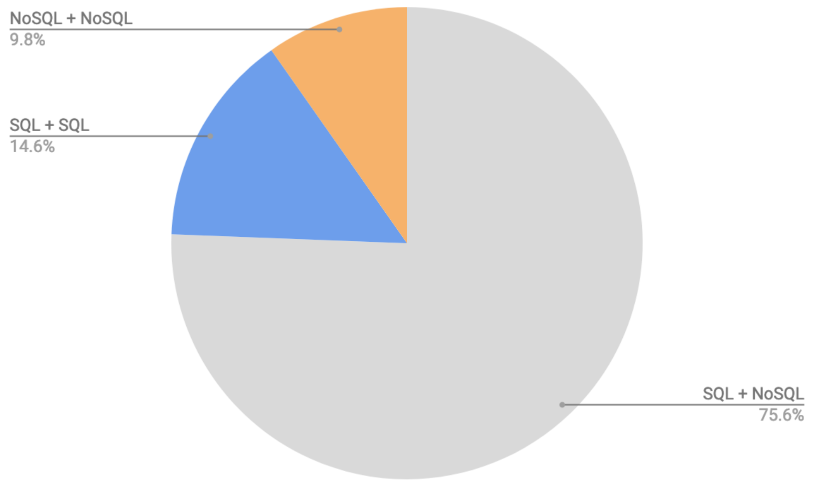
M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering
www.memregursoy.com

Introduction

- Pronounced as "S-Q-L" or "SeQueL"
- SQL is considered to be one of the major reasons for the commercial success of the relational data model
  - IBM, Microsoft, Oracle, ...
  - Main push from IBM in the early days (1970s)





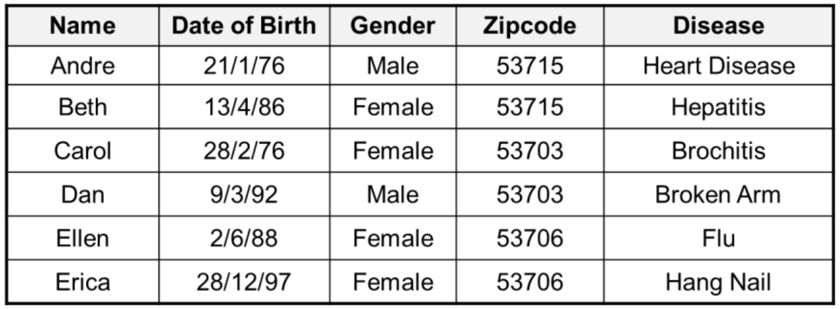
Images from ScaleGrid.io’s 2019 Database Trends study

Some Important Syntax

- CREATE DATABASE *databasename*;
- CREATE TABLE *table_name* (

  column1 datatype,
  column2 datatype,
  column3 datatype,
  ….
  );
- INSERT, UPDATE, DELETE
- SELECT *column1*, *column2*, …

  FROM *table_name*
  WHERE *condition*;
- DROP TABLE *table_name*

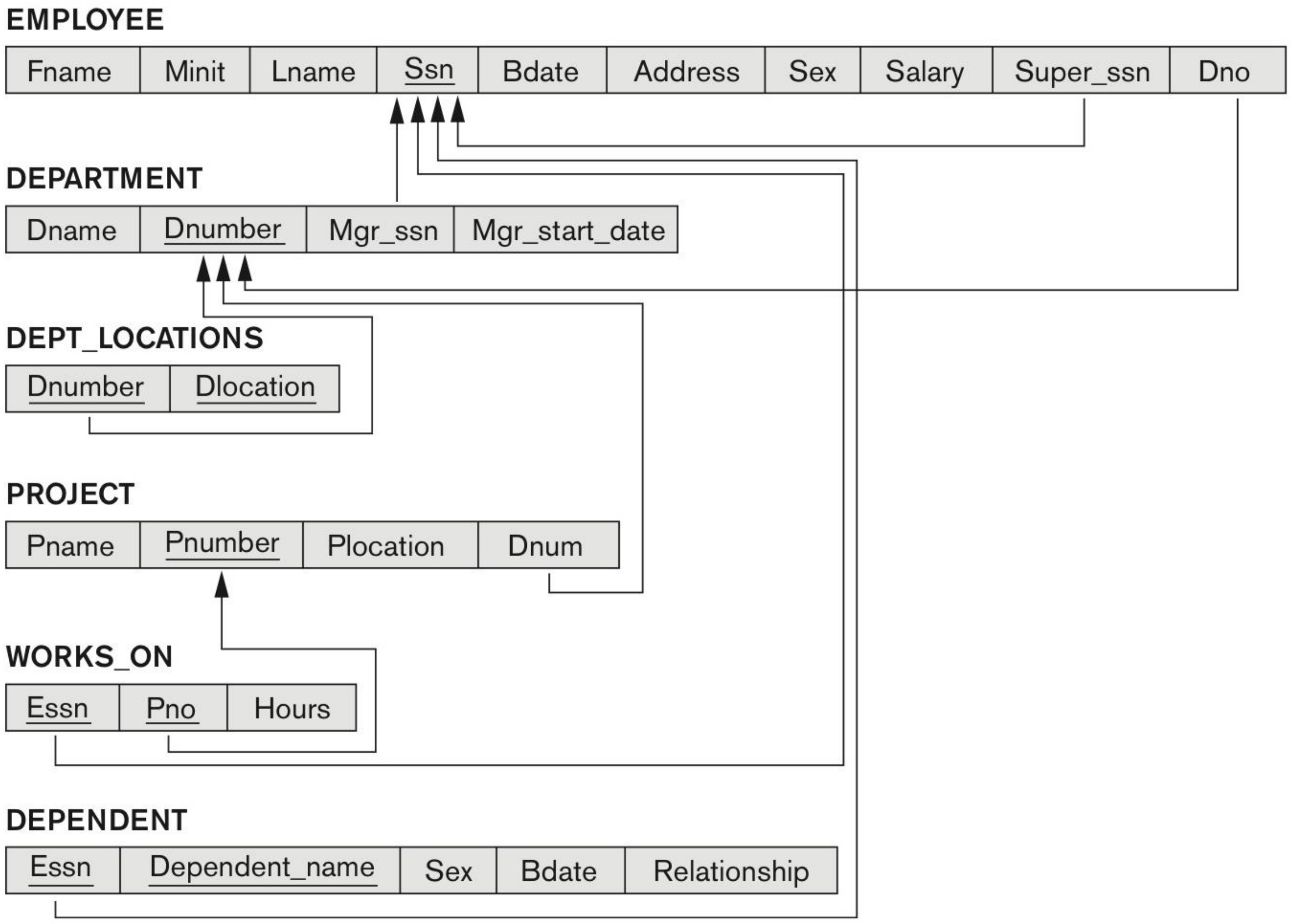


Name	Date of Birth	Gender	Zipcode	Disease
Andre	21/1/76	Male	53715	Heart Disease
Beth	13/4/86	Female	53715	Hepatitis
Carol	28/2/76	Female	53703	Brochitis
Dan	9/3/92	Male	53703	Broken Arm
Ellen	2/6/88	Female	53706	Flu
Erica	28/12/97	Female	53706	Hang Nail

Creating Tables

- `CREATE TABLE EMPLOYEE ...`
  - We have been working in "pseudo"-SQL
- Columns have data types
  - Numeric: `INTEGER`, `FLOAT`, `REAL`, ...
  - String: `CHAR(`*n*`)`, `CHARACTER(`*n*`)`, `VARCHAR(`*n*`)`
  - Bit-string: `BIT(`*n*`)`
  - Boolean: `TRUE` or `FALSE` or `NULL`
  - Date: `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD
  - Timestamp: includes date and time
  - ...
- **UNIQUE** clause: Used to represent a CANDIDATE key that was not selected as the PRIMARY key

Company DB


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

Company DB

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

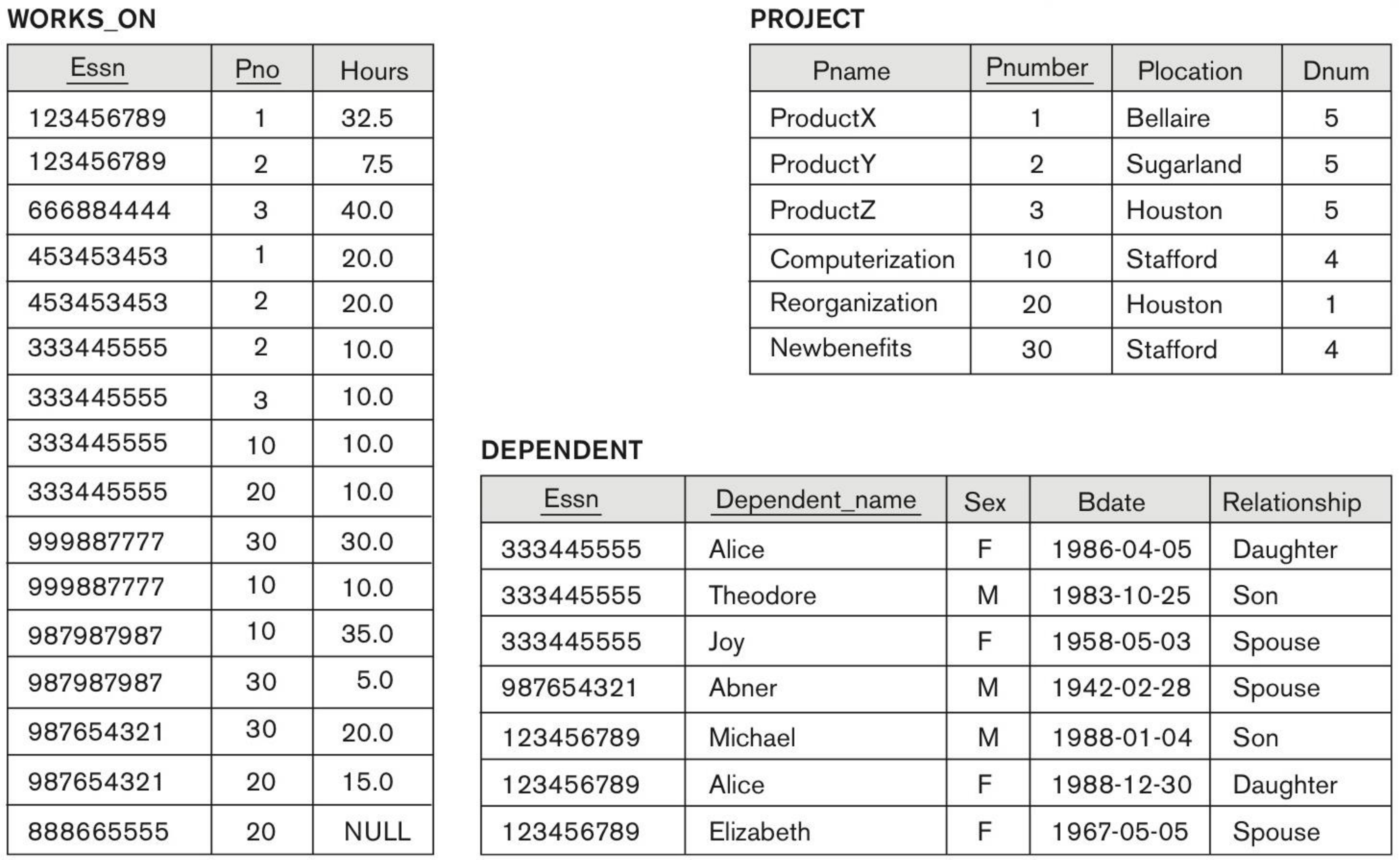
DEPARTMENT

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

Company DB



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

Company DB

```
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) );
```

Company DB

```
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) );
```

Retrieval Queries

```
SELECT      [DISTINCT]  attribute-list
FROM        relation-list
WHERE       condition
```

- **attribute-list** is a list of attribute names whose values are to be retrieved by the query.
- **relation-list** is a list of relation names (table names) required to process the query.
- **condition** is a boolean expression that specifies what conditions the tuples need to satisfy to be retrieved.
- **DISTINCT** is an optional keyword indicating duplicates should be eliminated (otherwise no duplicate elimination).

Retrieval Queries

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 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';
```

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;
```

Retrieval Queries

(c)

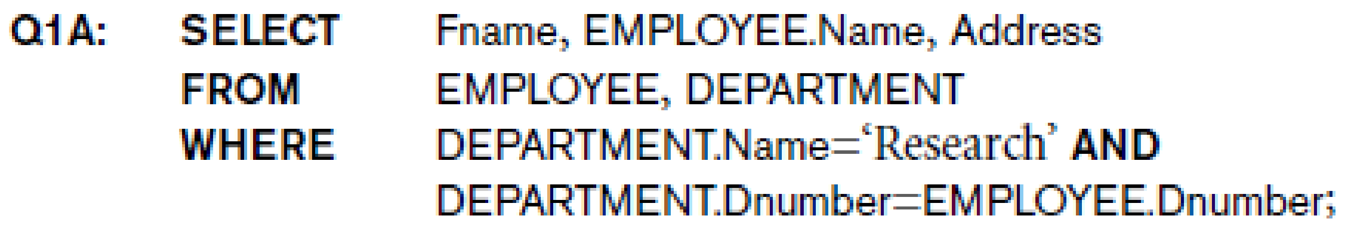
Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

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';
```

Attribute Naming

- If attribute names are unique across the referenced relations, we can omit the relation name in the query
  - We can write Plocation = "Stafford" instead of Project.Plocation = "Stafford"
- But if there exist two attributes with the same name in different relations, we must write the relation explicitly
  - In order to prevent ambiguity
- Some people prefer to always write the relation name



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

Aliasing and Renaming

- **AS:** Declare alternative names for relations
  - EMPLOYEE **AS** E, DEPARTMENT **AS** D, ...
  - This strategy is sometimes used just for brevity, but sometimes it becomes necessary
- Keyword **AS** can also be dropped

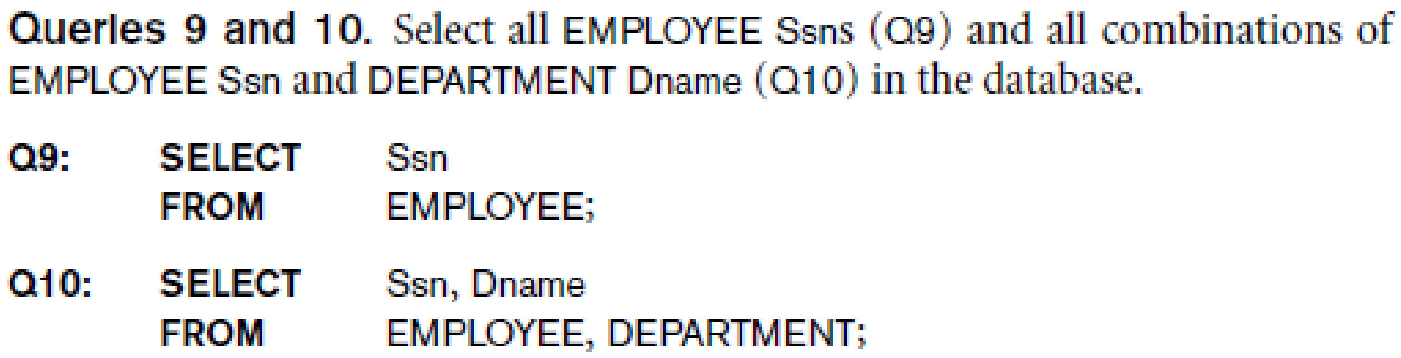
Query 8. For each employee, retrieve the employee's first and last name and the last name of the employee's supervisor.

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

Missing "Where" Clause

- Indicates no condition on tuple selection

- Effect is a CROSS PRODUCT
  - Recall from relational algebra



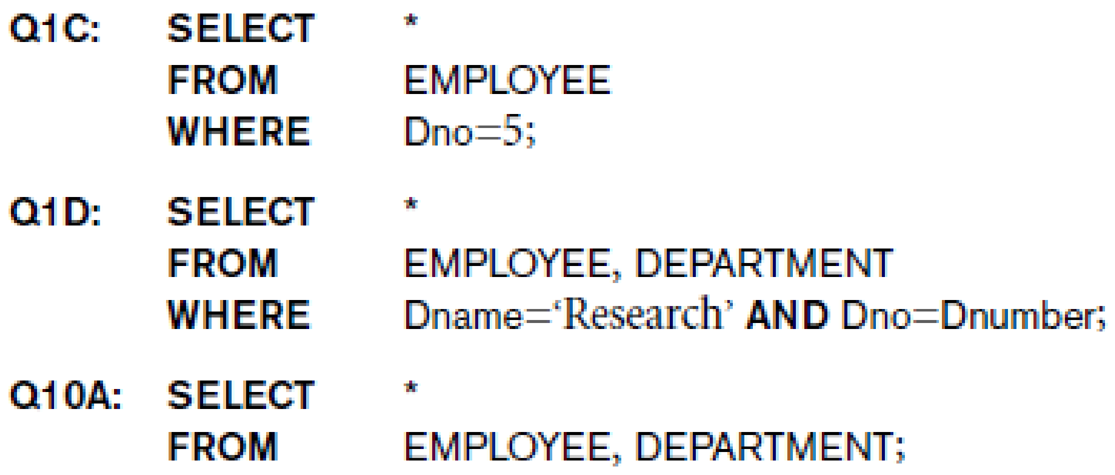
Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

```
Q9:     SELECT    Ssn
        FROM      EMPLOYEE;

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

Use of Asterisk (*)

- Retrieve all attribute values of selected tuples

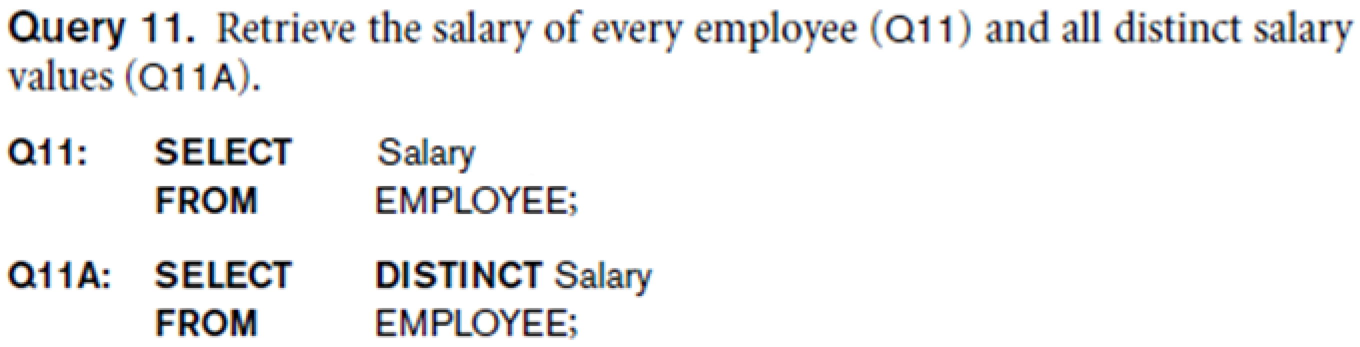
- Effect is "having no projection" (in relational algebra terms)



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

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

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

DISTINCT Keyword

- SQL does not automatically eliminate duplicates
  - This is different from the default behavior of the projection operation in relational algebra
- We use the **DISTINCT** keyword to eliminate duplicates



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

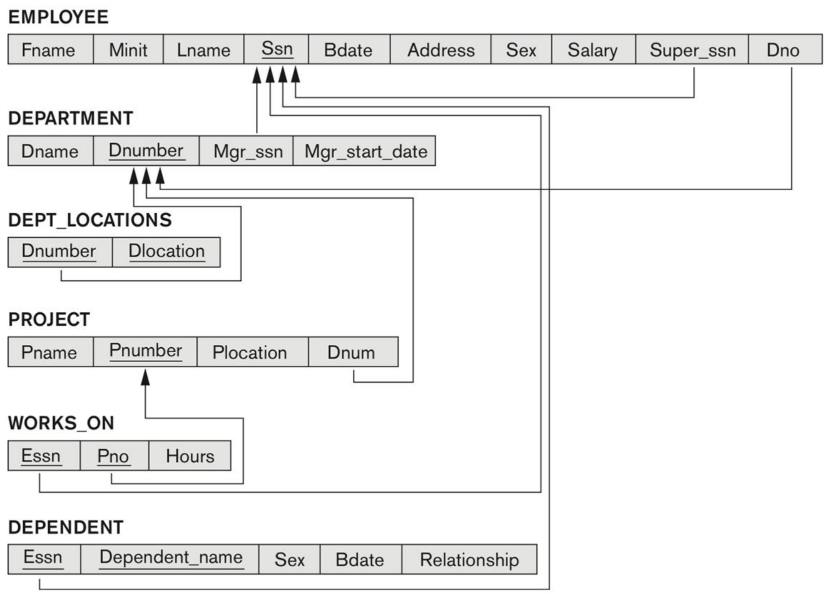
```
Q11:    SELECT    Salary
        FROM      EMPLOYEE;

Q11A:   SELECT    DISTINCT Salary
        FROM      EMPLOYEE;
```

Set Operations

- **UNION**, **EXCEPT** (difference), **INTERSECT**

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' );
```



String Matching (LIKE)

- LIKE is a comparison operator used for string pattern matching (similar to regular expressions).
  - % means an arbitrary number of zero or more characters
  - _ (underscore) means a single character

- Examples:
  - **WHERE Address LIKE ‘%Houston, TX%’;**
  - **WHERE Ssn LIKE ‘_ _ 1_ _ 8901’;**

Arithmetic Operations

- Standard arithmetic operations (+, -, *, /) may be included as part of the **SELECT** clause.

- **Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.**

```
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';
```

Ordering of Results

- We can add the optional **ORDER BY** clause in order to sort query results in a certain order.
  - **ORDER BY** typically comes at the end of the query
  - **DESC:** descending order
  - **ASC:** ascending order

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

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

Examples

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8

Find all students with age 18

```
SELECT  *
FROM  Students S
WHERE  S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2

Examples

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8

Names and logins of all students with age 18

```
SELECT  S.name, S.login
FROM  Students S
WHERE  S.age=18
```

name	login
Jones	jones@cs
Shero	shero@cs

Examples

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8

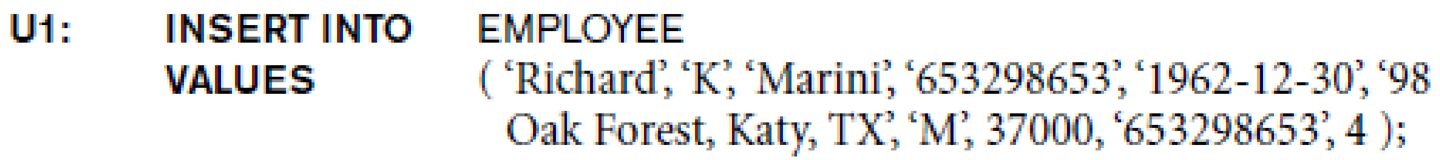
Names and cids of students who received A from one or more of their courses

```
SELECT  S.name, E.cid
FROM  Students S, Enrolled E
WHERE  S.sid=E.sid AND E.grade="A"
```

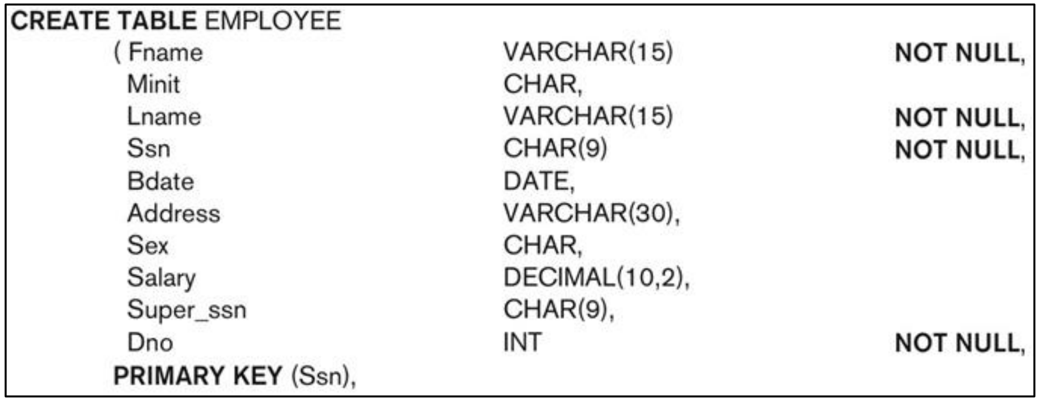
S.name	E.cid
Shero	Topology112

Insertion

- **INSERT**: add tuples to a relation
  - **INSERT INTO** *<table_name>* **VALUES** *(...)*
  - Attribute order should be same as CREATE TABLE
  - Constraints wrt data types (INT, CHAR, ..) and integrity (related to PK, FK, ..) are enforced



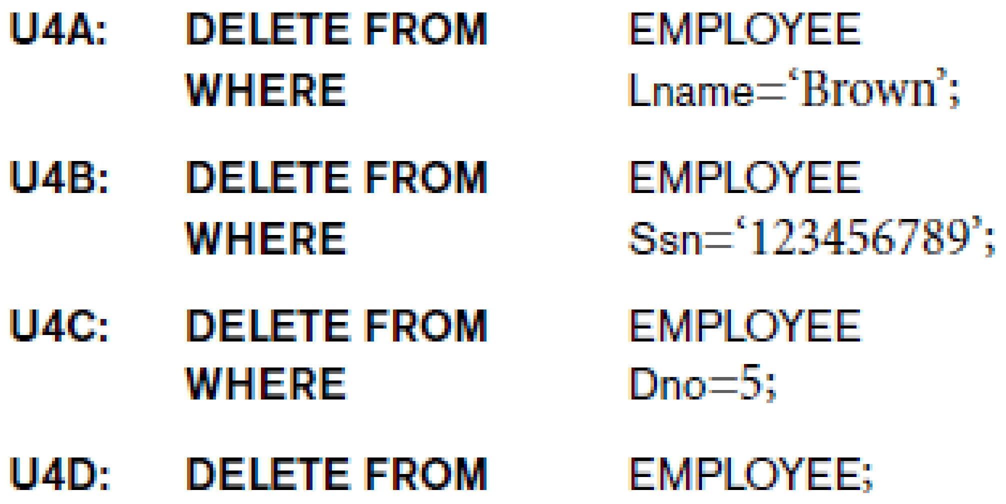
```
U1:     INSERT INTO   EMPLOYEE
        VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                        Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```



```
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),
```

Deletion

- **DELETE**: remove tuples from a relation
  - **DELETE FROM** *<table_name>* **WHERE** *(...)*
  - The WHERE clause indicates the condition regarding which tuples will be deleted



```
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;
```

Update

- **UPDATE**: modify attribute values of tuples
  - **UPDATE** *<table_name>* **SET** *<changes>* **WHERE** *(...)*

Change the location and controlling department number of project #10 to "Bellaire" and 5, respectively.

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

Give all employees in the "Research" department a 10% raise in salary.

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

Examples

```
INSERT INTO   Students
VALUES        (53688, 'Smith', 'smith@ee', 18, 3.2)
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

```
DELETE FROM Students S
WHERE S.name = 'Jones'
```

DROP TABLE Students → Destroy the Students relation. Schema and tuples are deleted.

Exercises

- Consider the following schema:
  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Exercises

- Sailors (<u>sid</u>, sname, rating, age)
- Boats (<u>bid</u>, bname, color)
- Reserves (<u>sid, bid, day</u>)

- Find the names of sailors who reserved boat number 103.

```
SELECT  sname
FROM    Sailors, Reserves
WHERE   Sailors.sid=Reserves.sid AND bid=103
```

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND R.bid=103
```

Exercises

- Sailors (<u>sid</u>, sname, rating, age)
- Boats (<u>bid</u>, bname, color)
- Reserves (<u>sid, bid, day</u>)

- For sailors whose names begin and end with letter B and contain at least three characters, display their name and twice their age.

```
SELECT   S.sname, 2*S.age
FROM     Sailors S
WHERE    S.sname LIKE 'B_%B'
```

Exercises

- Sailors (<u>sid</u>, sname, rating, age)
- Boats (<u>bid</u>, bname, color)
- Reserves (<u>sid, bid, day</u>)

- Find the IDs of sailors who have reserved a red boat or a green boat.

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
      AND (B.color=‘red’ OR B.color=‘green’)
```

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
           AND B.color=‘red’
UNION
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
           AND B.color=‘green’
```

Exercises

  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)
- Find the IDs of sailors who have reserved a red boat but never reserved a green boat.

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
            AND B.color=‘red’
EXCEPT
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
            AND B.color=‘green’
```

Exercises

  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)
- Find the IDs of sailors who have reserved a red boat and a green boat.

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
           AND B.color='red'
INTERSECT
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE   R.bid=B.bid
           AND B.color='green'
```

```
SELECT  R1.sid
FROM  Boats B1, Reserves R1,
        Boats B2, Reserves R2
WHERE  R1.sid = R2.sid AND
    R1.bid=B1.bid AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='green')
```

Conclusion

- SQL is a comprehensive language for relational database management (select, insert, modify, delete, ...)
- So far, we have covered fundamental SQL commands
  - Many advanced aspects and functionalities exist (we will cover some more, but not all)
  - You’ll learn about some of them in the PSs + HWs

- Next topic: **Advanced SQL**
  - Nested SQL queries
  - IN, EXISTS, NOT EXISTS, ...
  - GROUP BY - HAVING
  - Aggregates: COUNT, SUM, MIN, MAX, ...
  - Views