

CSI 2132 Tutorial 6

The Structured Query Language (SQL)

The Structured Query Language

- De facto standard used to interact with relational DB management systems
- Two major branches
 - **DDL (Data Definition Language)**
 - Contains statements that define the structure of the database (create, alter or delete tables, relationships, constraints, etc)
 - **DML (Data Manipulation Language)**
 - Retrieve and modify data

Chapter 4: "Basic SQL"

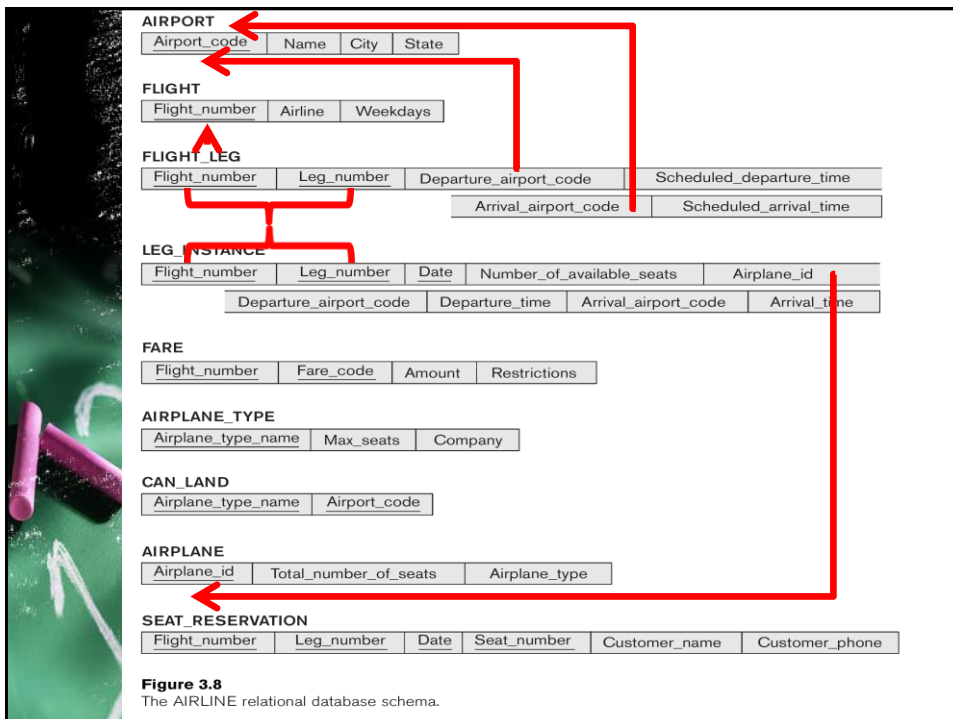
Exercise 4.6

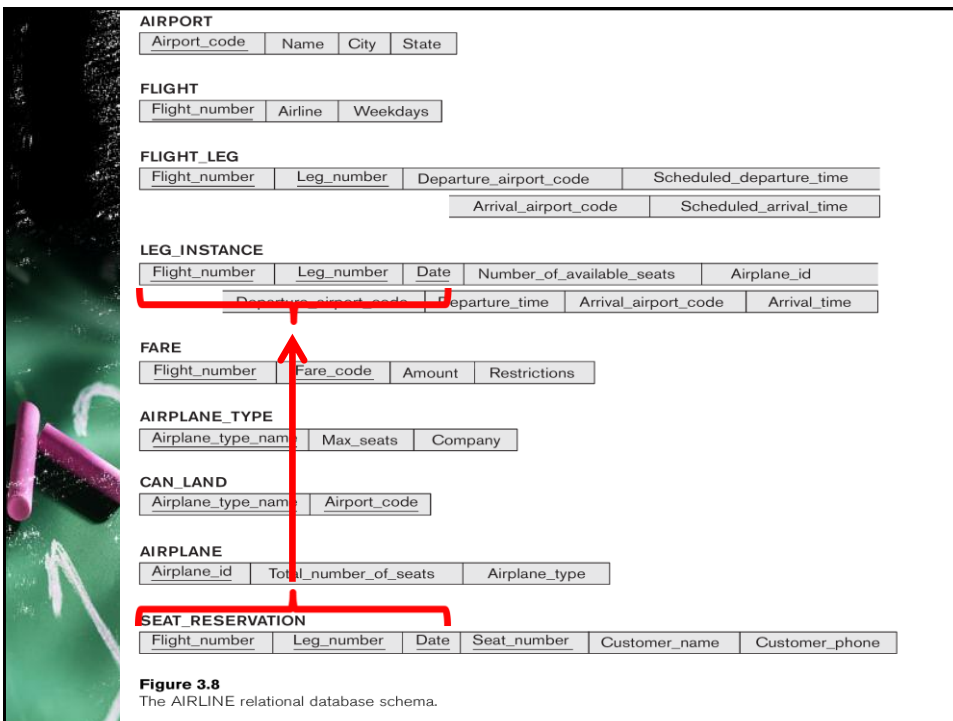
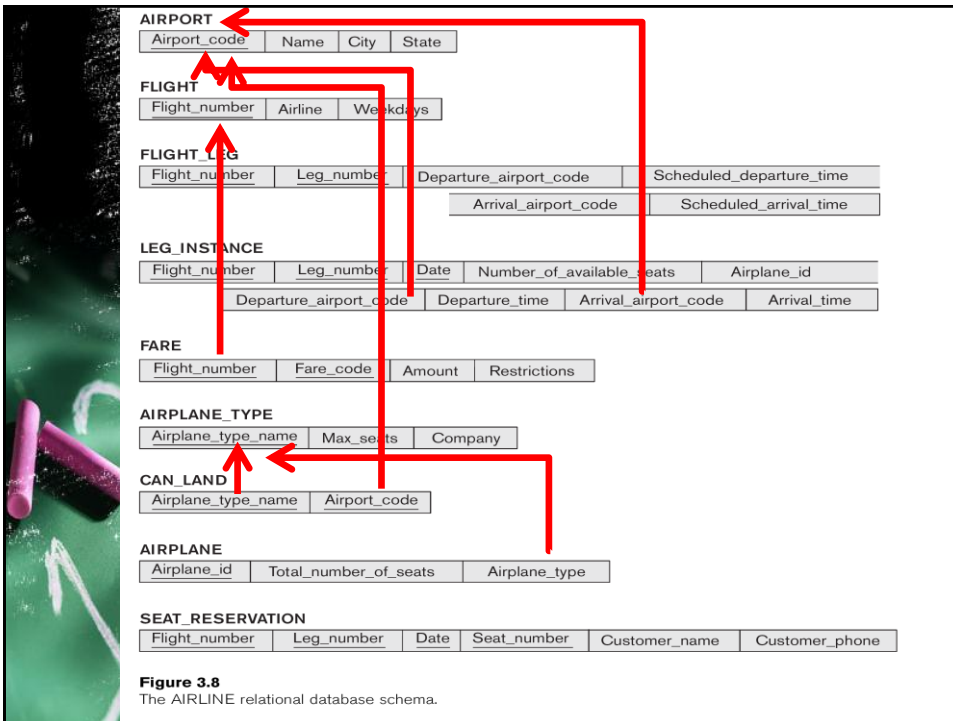
Consider the schema for the **AIRLINE** database in Fig. 3.8

What are the **referential integrity constraints** that should hold on the schema?

Write appropriate **SQL DDL statements** to define the database.

3





Chapter 4: "Basic SQL"

AIRPORT

<u>Airport_code</u>	Name	City	State
---------------------	------	------	-------

```
CREATE TABLE AIRPORT (  
    AIRPORT_CODE CHAR(3) NOT NULL,  
    NAME VARCHAR(30) NOT NULL,  
    CITY VARCHAR(30) NOT NULL,  
    STATE VARCHAR(30),  
    PRIMARY KEY (AIRPORT_CODE)  
);
```

7

Chapter 4: "Basic SQL"

FLIGHT

<u>Flight_number</u>	Airline	Weekdays
----------------------	---------	----------

```
CREATE TABLE FLIGHT (  
    NUMBER VARCHAR(6) NOT NULL,  
    AIRLINE VARCHAR(20) NOT NULL,  
    WEEKDAYS VARCHAR(10) NOT NULL,  
    PRIMARY KEY (NUMBER)  
);
```

8

Chapter 4: "Basic SQL"

FLIGHT_LEG

Flight_number	Leg_number	Departure_airport_code	Scheduled_departure_time
		Arrival_airport_code	Scheduled_arrival_time

```
CREATE TABLE FLIGHT_LEG (
    FLIGHT_NUMBER VARCHAR(6) NOT NULL,
    LEG_NUMBER INTEGER NOT NULL,
    DEPARTURE_AIRPORT_CODE CHAR(3) NOT NULL,
    SCHEDULED_DEPARTURE_TIME TIMESTAMP WITH TIME ZONE,
    ARRIVAL_AIRPORT_CODE CHAR(3) NOT NULL,
    SCHEDULED_ARRIVAL_TIME TIMESTAMP WITH TIME ZONE,
    PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER),
    FOREIGN KEY (FLIGHT_NUMBER) REFERENCES FLIGHT (NUMBER),
    FOREIGN KEY (DEPARTURE_AIRPORT_CODE) REFERENCES
    AIRPORT (AIRPORT_CODE),
    FOREIGN KEY (ARRIVAL_AIRPORT_CODE) REFERENCES
    AIRPORT (AIRPORT_CODE) );
```

9

Chapter 4: "Basic SQL"

LEG_INSTANCE

Flight_number	Leg_number	Date	Number_of_available_seats	Airplane_id	
		Departure_airport_code	Departure_time	Arrival_airport_code	Arrival_time

```
CREATE TABLE LEG_INSTANCE (
    FLIGHT_NUMBER VARCHAR(6) NOT NULL,
    LEG_NUMBER INTEGER NOT NULL,
    LEG_DATE DATE NOT NULL,
    NO_OF_AVAILABLE_SEATS INTEGER,
    AIRPLANE_ID INTEGER,
    DEPARTURE_AIRPORT_CODE CHAR(3),
    DEPARTURE_TIME TIMESTAMP WITH TIME ZONE,
    ARRIVAL_AIRPORT_CODE CHAR(3),
    ARRIVAL_TIME TIMESTAMP WITH TIME ZONE,
    PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE),
    FOREIGN KEY (FLIGHT_NUMBER, LEG_NUMBER) REFERENCES
    FLIGHT_LEG (FLIGHT_NUMBER, LEG_NUMBER),
    FOREIGN KEY (AIRPLANE_ID) REFERENCES
    AIRPLANE (AIRPLANE_ID),
    FOREIGN KEY (DEPARTURE_AIRPORT_CODE) REFERENCES
    AIRPORT (AIRPORT_CODE),
    FOREIGN KEY (ARRIVAL_AIRPORT_CODE) REFERENCES
    AIRPORT (AIRPORT_CODE)
);
```

10

Chapter 4: "Basic SQL"

FARE

<u>Flight_number</u>	<u>Fare_code</u>	Amount	Restrictions
----------------------	------------------	--------	--------------

```
CREATE TABLE FARES (
    FLIGHT_NUMBER VARCHAR(6) NOT NULL,
    FARE_CODE VARCHAR(10) NOT NULL,
    AMOUNT DECIMAL(8,2) NOT NULL,
    RESTRICTIONS VARCHAR(200),
    PRIMARY KEY (FLIGHT_NUMBER, FARE_CODE),
    FOREIGN KEY (FLIGHT_NUMBER) REFERENCES FLIGHT (NUMBER)
);
```

11

Chapter 4: "Basic SQL"

AIRPLANE_TYPE

<u>Airplane_type_name</u>	Max_seats	Company
---------------------------	-----------	---------

```
CREATE TABLE AIRPLANE_TYPE (
    TYPE_NAME VARCHAR(20) NOT NULL,
    MAX_SEATS INTEGER NOT NULL,
    COMPANY VARCHAR(15) NOT NULL,
    PRIMARY KEY (TYPE_NAME)
);
```

12

Chapter 4: "Basic SQL"

CAN_LAND

<u>Airplane_type_name</u>	<u>Airport_code</u>
---------------------------	---------------------

```
CREATE TABLE CAN_LAND (
  AIRPLANE_TYPE_NAME VARCHAR(20) NOT NULL,
  AIRPORT_CODE CHAR(3) NOT NULL,
  PRIMARY KEY (AIRPLANE_TYPE_NAME, AIRPORT_CODE),
  FOREIGN KEY (AIRPLANE_TYPE_NAME) REFERENCES
  AIRPLANE_TYPE (TYPE_NAME),
  FOREIGN KEY (AIRPORT_CODE) REFERENCES
  AIRPORT (AIRPORT_CODE)
);
```

13

Chapter 4: "Basic SQL"

AIRPLANE

<u>Airplane_id</u>	Total_number_of_seats	Airplane_type
--------------------	-----------------------	---------------

```
CREATE TABLE AIRPLANE (
  AIRPLANE_ID INTEGER NOT NULL,
  TOTAL_NUMBER_OF_SEATS INTEGER NOT NULL,
  AIRPLANE_TYPE VARCHAR(20) NOT NULL,
  PRIMARY KEY (AIRPLANE_ID),
  FOREIGN KEY (AIRPLANE_TYPE) REFERENCES AIRPLANE_TYPE (TYPE_NAME)
);
```

14

Chapter 4: "Basic SQL"

SEAT_RESERVATION

<u>Flight_number</u>	<u>Leg_number</u>	<u>Date</u>	<u>Seat_number</u>	Customer_name	Customer_phone
----------------------	-------------------	-------------	--------------------	---------------	----------------

```
CREATE TABLE SEAT_RESERVATION (
  FLIGHT_NUMBER VARCHAR(6) NOT NULL,
  LEG_NUMBER INTEGER NOT NULL,
  LEG_DATE DATE NOT NULL,
  SEAT_NUMBER VARCHAR(4) NOT NULL,
  CUSTOMER_NAME VARCHAR(30) NOT NULL,
  CUSTOMER_PHONE CHAR(12),
  PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE, SEAT_NUMBER),
  FOREIGN KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE) REFERENCES
  LEG_INSTANCE (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE)
);
```

15

Chapter 4: "Basic SQL"

We use the following notation to describe the foreign key constraints.

**FLIGHT_LEG.(FLIGHT_NUMBER) -->
FLIGHT.(NUMBER)**

**SEAT_RESERVATION.(FLIGHT_NUMBER,
LEG_NUMBER, LEG_DATE) -->
LEG_INSTANCE.(FLIGHT_NUMBER,
LEG_NUMBER, LEG_DATE)**

16

Chapter 4: "Basic SQL"

Exercise 4.7

Consider the schema for the **LIBRARY** database in Fig. 4.6

Choose the appropriate action (reject, cascade, set to null, set to default) for each referential integrity constraint, both for a **deletion** of a referenced tuple and for the **update** of a primary key attribute value in a referenced tuple. Justify your choices.

17

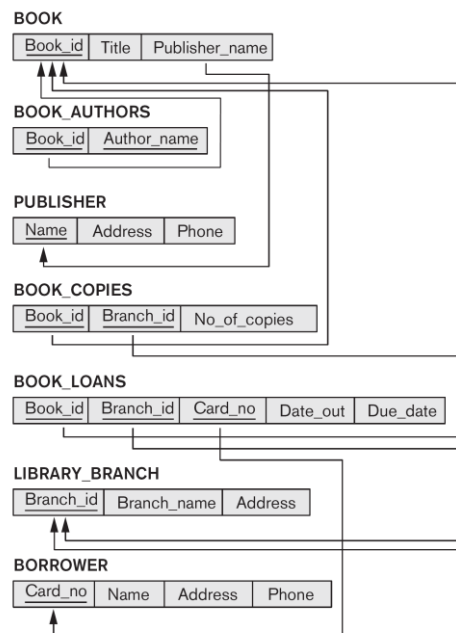


Figure 4.6
A relational database schema for a LIBRARY database.

18

Chapter 4: "Basic SQL"

**BOOK_AUTHORS.(BookId) -->
BOOK.(BookId)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

**Automatically propagate the deletion or
change of a BOOK to the referencing
BOOK_AUTHORS.**

19

Chapter 4: "Basic SQL"

**BOOK.(PublisherName) -->
PUBLISHER.(Name)**

**ON DELETE REJECT
ON UPDATE CASCADE**

**Do not delete a PUBLISHER tuple which
has linked BOOK tuples.**

**Update the PUBLISHER's name on all
BOOK tuples which refer to it.**

20

Chapter 4: "Basic SQL"

**BOOK_LOANS.(BookID) -->
BOOK.(BookID)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

**If a BOOK record is deleted, then delete
all its associated BOOK_LOAN records.
Idem with updates.**

REJECT on DELETE also possible!

21

Chapter 4: "Basic SQL"

**BOOK_COPIES.(BookID) -->
BOOK.(BookID)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

**If a BOOK record is deleted, then delete
all its associated BOOK_COPIES tuples.
Do likewise with updates.**

22

Chapter 4: "Basic SQL"

**BOOK_LOANS.(CardNo) -->
BORROWER.(CardNo)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

If a BORROWER record is deleted, then delete all its associated BOOK_LOANS tuples. Do likewise with updates.

REJECT on DELETE also possible!

23

Chapter 4: "Basic SQL"

**BOOK_COPIES.(BranchID) -->
LIBRARY_BRANCH.(BranchID)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

If a LIBRARY_BRANCH record is deleted, then delete all its linked BOOK_COPIES tuples. Do likewise with updates.

REJECT on DELETE also possible!

24

Chapter 4: "Basic SQL"

**BOOK_LOANS.(BranchID) -->
LIBRARY_BRANCH.(BranchID)**

**ON DELETE CASCADE
ON UPDATE CASCADE**

If a LIBRARY_BRANCH record is deleted,
then delete all its linked BOOK_LOANS
tuples. Do likewise with updates.

REJECT on DELETE also possible!

25

Chapter 4: "Basic SQL"

Exercise 4.15

**Consider the EMPLOYEE table's
constraint EMPSUPERFK as in Fig 4.2**

26

```

CREATE TABLE EMPLOYEE
(
    ...,
    Dno INT NOT NULL DEFAULT 1,
CONSTRAINT EMPCHK
    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);

```

Figure 4.2

Example illustrating how default attribute values and referential integrity triggered actions are specified in SQL.

Chapter 4: "Basic SQL"

Exercise 4.15

If the constraint is changed to read as follows:

```

CONSTRAINT EMPSUPERFK
    FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
        ON DELETE CASCADE ON UPDATE CASCADE

```


Chapter 4: "Basic SQL"

Exercise 4.15

What happens when the following command is run on the **COMPANY** database state shown in Fig. 3.6?

```
DELETE EMPLOYEE WHERE LNAME = 'Borg'
```

29

Figure 3.6

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
Abmad	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

Chapter 4: "Basic SQL"

Answer: The table gets emptied.

➤ *Triggers a deletion of all subordinate records in James Borg's supervision hierarchy.*

31

Chapter 4: "Basic SQL"

Exercise 4.15

Is it better to **CASCADE** or **SET NULL** in case of EMPSUPERFK constraint ON DELETE?

32

Chapter 4: "Basic SQL"

Answer:

SET NULL is preferred, since an **EMPLOYEE** is not fired (deleted) when his/her supervisor is deleted.

Instead, the **SUPERSSN** field should be **SET NULL** so a new supervisor could be assigned later on.

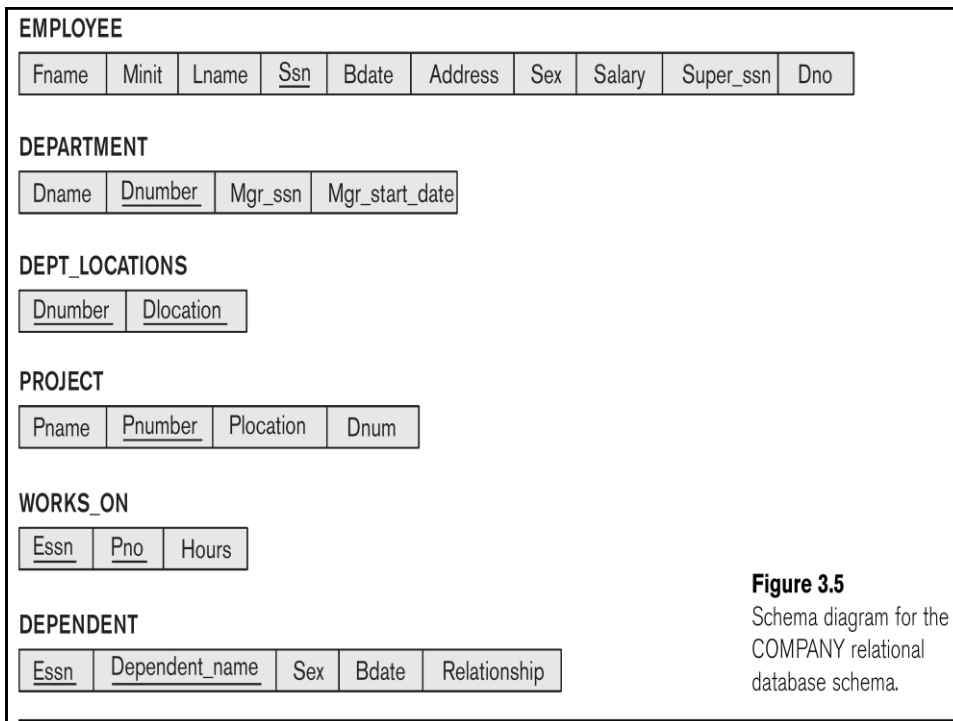
33

Chapter 5: "More SQL"

Exercise 5.5

Specify the following additional SQL queries on the **COMPANY** database of Fig. 3.5

34



Chapter 5: "More SQL"

Exercise 5.5

a) For each department whose average employee salary is over 30K, retrieve the department name and the number of employees working for it.

Chapter 5: "More SQL"

Exercise 5.5

a) For each department whose average employee salary is over 30K, retrieve the department name and the number of employees working for it.

```
SELECT DNAME, COUNT (*)  
FROM DEPARTMENT, EMPLOYEE  
WHERE DNUMBER=DNO  
GROUP BY DNAME  
HAVING AVG (SALARY) > 30000
```

37

Chapter 5: "More SQL"

Exercise 5.5

b) Suppose we want the number of **male** employees in each department rather than all employees.

Can we specify this in SQL? Why or why not?

38

Chapter 5: "More SQL"

Yes, via a **nested query**

```
SELECT DNAME, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE DNUMBER=DNO AND SEX='M' AND DNO IN (
    SELECT DNO
    FROM EMPLOYEE
    GROUP BY DNO
    HAVING AVG (SALARY) > 30000
)
GROUP BY DNAME
```


39

Chapter 5: "More SQL"

Exercise 5.6

Specify the following SQL queries on the **UNIVERSITY** database schema of Fig. 1.2

40



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS


COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

41



GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

42

Chapter 5: "More SQL"

Exercise 5.6

- a) Retrieve the names and major departments of all **straight-A** students (i.e. those who got 'A' in all their courses)

43

Chapter 5: "More SQL"

Exercise 5.6

- a) Retrieve the names and major departments of all **straight-A** students (i.e. those who got 'A' in all their courses)

```
SELECT Name, Major
FROM STUDENT
WHERE NOT EXISTS ( SELECT *
                   FROM GRADE_REPORT
                   WHERE StudentNumber= STUDENT.StudentNumber
                   AND NOT(Grade='A')
                 )
```

44

Chapter 5: "More SQL"

Exercise 5.6

- b) Retrieve the names and major departments of all students who do not have any grade of A **in any of their courses**.

45

Chapter 5: "More SQL"

Exercise 5.6

- b) Retrieve the names and major departments of all students who do not have any grade of A **in any of their courses**.

```
SELECT Name, Major
FROM STUDENT
WHERE NOT EXISTS ( SELECT *
                   FROM GRADE_REPORT
                   WHERE StudentNumber= STUDENT.StudentNumber
                   AND Grade='A'
                   )
```

46

Chapter 5: "More SQL"

Another way

```
SELECT Name, Major
FROM STUDENT
WHERE StudentNumber NOT IN ( SELECT StudentNumber
                              FROM GRADE_REPORT
                              WHERE StudentNumber = STUDENT.StudentNumber
                              AND Grade = 'A'
                              )
```

47

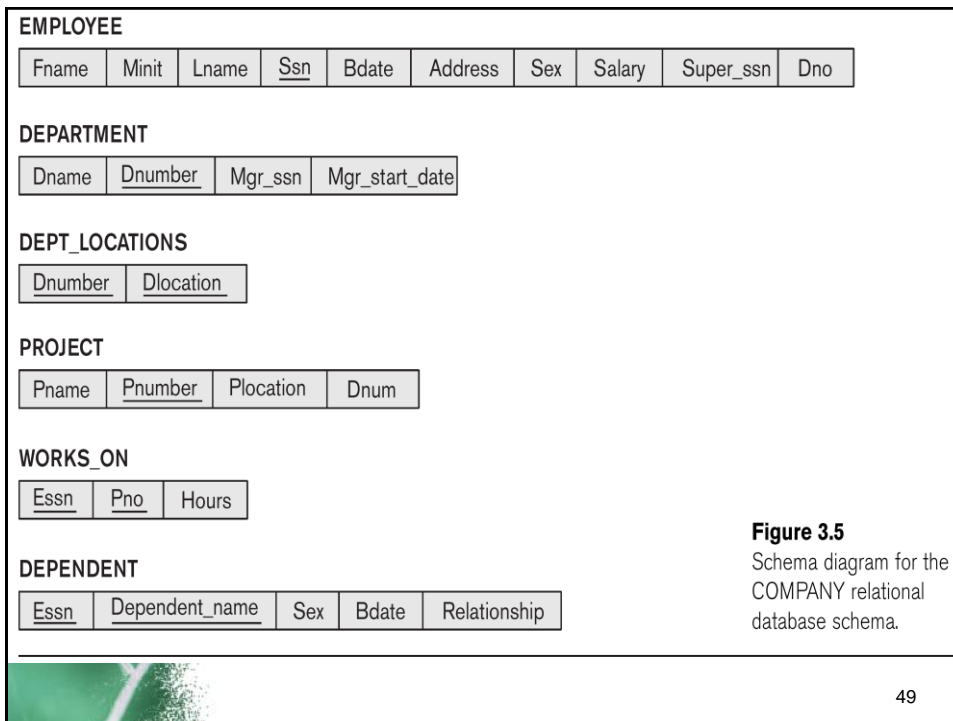
Chapter 5: "More SQL"

Exercise 5.7

In SQL, specify the following queries on the **COMPANY** database in Fig. 3.5 using the concept of **nested queries**

- a) Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.

48



Chapter 5: "More SQL"

a) Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.

```

SELECT LNAME
FROM EMPLOYEE
WHERE DNO = ( SELECT DNO
              FROM EMPLOYEE
              WHERE SALARY = ( SELECT MAX(SALARY)
                              FROM EMPLOYEE
                              )
              )

```

Chapter 5: "More SQL"

Exercise 5.7

b) Retrieve the names of all employees whose supervisor's supervisor has '888665555' for SSN.

51

Chapter 5: "More SQL"

Exercise 5.7

b) Retrieve the names of all employees whose supervisor's supervisor has '888665555' for SSN.

```
SELECT LNAME
FROM EMPLOYEE
WHERE SUPERSSN IN ( SELECT SSN
                    FROM EMPLOYEE
                    WHERE SUPERSSN = '888665555'
                  )
```

52

Chapter 5: "More SQL"

Exercise 5.7

c) Retrieve the names of employees who make at least 10K more than the employee who is paid the least in the company.

53

Chapter 5: "More SQL"

Exercise 5.7

c) Retrieve the names of employees who make at least 10K more than the employee who is paid the least in the company.

```
SELECT LNAME
FROM EMPLOYEE
WHERE SALARY >= 10000 + ( SELECT MIN(SALARY)
                          FROM EMPLOYEE
                          )
```

54

