Universidad Autónoma de Madrid

EDAT

Exercises

Roberto Marabini Ruiz

Contents

1 Database design intro

1.1 Chess Championship - Design

```
Player (player_id, name)
Referee (referee_id, name, birthdate)
Room (room_id, name)
Game (time, white^, black^,
      referee_id^, room_id^, winner^, finished ) -- players do not play
                                                 -- simultaneous games
Move (move_id, piece, origin, target,
      white^, time^)
-- another possible schema  is,
-- Game (game_id, white^, black^,
--      referee_id^, room_id^, move_id^, winner^, finished, time )
-- Move(move_id, piece, origin, target, game_id^)
```

2 SQL

2.1 Chess Championship - SQL

2.1.1 write CREATE TABLE commands needed to implement the chess database

```
CREATE TABLE player (
    id serial primary key,
    name character(64)
);

CREATE TABLE referee (
    id serial primary key,
    name character(64)
```

```
);

CREATE TABLE room (
    id serial primary key,
    name character(64)
);

CREATE TABLE game (
    time timestamp,
    white integer REFERENCES player(id),
    black integer REFERENCES player(id),
    room integer REFERENCES room(id),
    PRIMARY KEY(time, white)
);
```

2.2 Playing with max

Given the following schema use SQL to find the oldest owner

```
Owner (ownerid, name, phone, age)
```

2.2.1 Using EXISTS

```
SELECT age, ownerid
FROM Owner o1
WHERE NOT EXISTS (
     SELECT 1
     FROM Owner o2
     WHERE o2.age > o1.age)
```

2.2.2 Using NOT IN

```
select age, ownerid
from owner
where age not in (
      select o2.age
      from owner as o1 JOIN owner as o2
      WHERE o1.age > o2.age);
```

2.2.3 Using ALL operator

```
select o1.*
from owner o1
where age > all (select age
                 from owner o2
                 where o2.ownerid <> o1.ownerid);
```

2.2.4 Using EXCEPT

```
select o1.*
from owner o1
```

```
EXCEPT

select o2.*
from owner o2, owner o3
where o2.age < o3.age;
```

2.2.5 Using max

```
CREATE VIEW anciano AS
select max(age) as age
from owner

select o2.*
from owner NJ anciano
```

2.2.6 Using ORDER BY and LIMIT

```
CREATE VIEW anciano AS
select age
from owner
ORDER BY age desc LIMIT 1

select o2.*
from owner NJ anciano
```

2.2.7 For each men find the oldest women (her name) that he likes

```
CREATE VIEW older AS
SELECT max(age) as age, nameM
FROM MlikesW NJ Women
GROUP BY nameM

SELECT nameW, nameM
```

```
FROM older NJ Women
```

2.3 Social Network - Likes

Given the following schema use SQL to answer the queries

```
MEN   (NameM, age)
WOMEN (NameW, age)
MlikesW (NameM^, NameW^) -- Man NameM likes woman NameW
WlikesM (NameW^, NameM^) -- Woman NameW likes man NameN
MARRIAGE (NameM^, NameW^)
```

2.3.1 Create Database, and populate it

```
CREATE TABLE men (
   namem char(16) primary key,
   age int
);
CREATE TABLE women (
   namew char(16) primary key,
   age int
);
CREATE TABLE mlikew (
   namem char(16) REFERENCES men(namem),
   namew char(16) REFERENCES women(namew),
   primary key(namem, namew)
);
CREATE TABLE wlikem (
   namew char(16) REFERENCES women(namew),
   namem char(16) REFERENCES men(namem),
   primary key(namew, namem)
);
CREATE TABLE marriage (
   namem char(16) REFERENCES men(namem),
   namew char(16) REFERENCES women(namew),
   primary key(namem, namew)
```

```
);

INSERT INTO men VALUES ('pepe', 23);
INSERT INTO men VALUES ('luis', 21);
INSERT INTO men VALUES ('enrique', 22);

INSERT INTO women VALUES ('maria', 32);
INSERT INTO women VALUES ('teresa', 12);
INSERT INTO women VALUES ('sofia', 22);

INSERT INTO mlikew VALUES ('pepe', 'sofia');
INSERT INTO mlikew VALUES ('pepe', 'maria');
INSERT INTO mlikew VALUES ('luis', 'maria');

INSERT INTO wlikem VALUES ('maria', 'enrique');
INSERT INTO wlikem VALUES ('maria', 'pepe');
INSERT INTO wlikem VALUES ('sofia', 'luis');

INSERT INTO marriage VALUES ('pepe', 'maria');
```

2.3.2 Married women that do not like her husband

```
-- All Married women
SELECT NameW
FROM MARRIAGE

EXCEPT

-- Women that like their husband
SELECT ma.NameW
```

```
FROM MARRIAGE ma, WlikeM l
WHERE ma.NameM = l.NameM
  AND ma.NameW = l.NameW

  -----
  empty
```

2.3.3 Men that do not like any woman

```
--All men
SELECT NameM
FROM MEN
EXCEPT
--men that like at least one woman
SELECT NameM
FROM MlikeW
;
      namem
------------------
 enrique
(1 row)
```

2.3.4 Married women that do not like any married men

```
--All married women
SELECT NameW
FROM MARRIAGE
EXCEPT
--women that like at least one married man
SELECT w.NameM
FROM MARRIAGE m, WlikeM w
WHERE m.NameM = w.NameM
;
      namew
```

```
------------------
 maria
(1 row)
```

2.3.5 Oldest men

```
SELECT namem
FROM men

EXCEPT

SELECT m1.namem
FROM men m1, men m2
WHERE m1.age < m2.age
;
       namem
------------------
 luis
(1 row)
```

2.3.6 Find pairs of men and women that (1) they like each other, (2) are between 30 and 40 years old and (3) are not married to each other

```
-- They like each other
CREATE VIEW likeeachother AS
SELECT m.NameM, m.NameW
FROM MlikeW m, WlikeM w
WHERE m.NameM=w.NameM AND m.NameW=w.NameW;

--Between 30 and 40 years
SELECT m.NameM, w.NameW
FROM likeeachother natural join MEN m natural join WOMEN w
WHERE m.age > 30 and w.age < 40  -- another alternative
```

```
 AND  w.age > 30 and w.age < 40  -- age BETWEEN 30 AND 40
EXCEPT -- remove married couples
SELECT m.NameM, m.NameW
FROM  MARRIAGE

-----
empty
```

2.3.7 Age Oldest woman (aggregation functions)

```
select max(age)
from women;

 max
-----
  32
```

2.3.8 name Oldest woman (aggregation functions)

```
select women.namew
from (select max(age) as age
      from women) p natural join women;

      namew
------------------
 maria
(1 row)
```

2.3.9 For each woman name of the oldest man who likes her

```
--age of oldest man who like each woman
create view old AS
select max(age) as age, namew
```

```
from mlikew natural join men
group by namew;

--name of the man
select *
from old natural join men
         natural join mlikew
;
```

```
      namew        |      namem        | age
-------------------+-------------------+-----
 sofia             | pepe              |  23
 maria             | pepe              |  23
```

2.4 Library

Given the following relational schema

```
Customer(customer_id, name)
Book (book_id, title)
Loan (customer_id^, book_id^, date)
```

2.4.1 Oldest loan

```
CREATE VIEW oldestLoan AS
SELECT min(date) as date
FROM loan

SELECT loan.*
FROM oldestLoan NJ loan
```

2.4.2 Title of the book of the Oldest loan

```
CREATE VIEW oldestLoan AS
SELECT min(date) as date
FROM loan

SELECT book.title
FROM oldestLoan NJ loan NJ book
```

2.4.3 Name of the Patron who has the Oldest loan

```
CREATE VIEW oldestLoan AS
SELECT min(date) as date
FROM loan
```

```
SELECT book.title
FROM oldestLoan NJ loan NJ Customer
```

2.4.4 For each Patron title of the last borrowed book

```
CREATE VIEW lastLoanPerPatron AS
SELECT max(date) as date
FROM loan
GROUP BY customer_id

SELECT book.title, customer_id
FROM lastLoanPerPatron NJ loan NJ book
```

2.4.5 Name of the Patron who has borrowed more books

```
CREATE VIEW bookPerPatron AS
SELECT count(*) as number
FROM loan
GROUP BY customer_id

-- there is no max(count(*))

CREATE VIEW maxBookPerPatron AS
SELECT max(number)
FROM bookPerPatron

SELECT name
FROM maxBookPerPatron NJ bookPerPatron NJ customer
```

2.4.6 Name of the Patron who has borrowed more than 5 books

```
CREATE VIEW bookPerPatron AS
```

```
SELECT count(*) as number
FROM loan
GROUP BY customer_id
HAVING count(*) > 5

-- there is no max(count(*))

CREATE VIEW maxBookPerPatron AS
SELECT max(number)
FROM bookPerPatron

SELECT name
FROM maxBookPerPatron NJ bookPerPatron NJ customer
```

3 ER Modeling Concepts

3.1 Keys

Explain the distinctions among the terms primary key, candidate key, and superkey

A superkey is a set of one or more attributes that uniquely identifies an entity in the entity set. A candidate key is a superkey from which you cannot remove any fields. The primary key is one of the candidate keys that is chosen by the database designer as the principal means of identifying entities within an entity set.

3.2 Entities

Explain the difference between a weak and a strong entity set

A strong entity set has a primary key. All tuples in the set are distinguishable by that key. A weak entity set has no primary key unless the PK of the strong entity set on which it depends are included.

3.3 Attributes

What is an atomic attribute.

Atomic attributes are attributes that can not be further divided

4 E-R diagram for a car-insurance company

Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. The car owner may be different from the car driver (both persons are needed in each accident report).

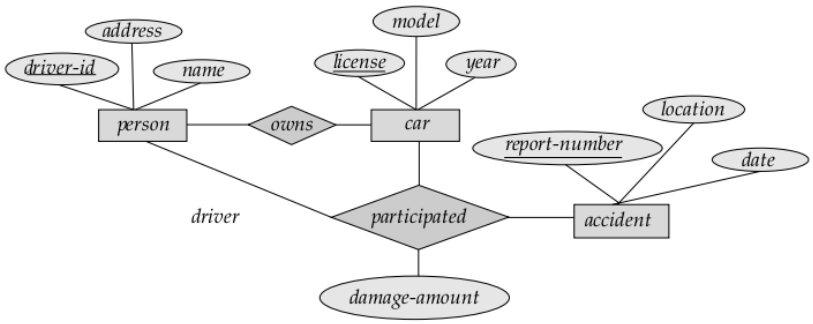


Figure 1: E-R diagram for a car-insurance company

5 Arrange Meeting

An application that satisfies the following requirements must be implemented. The application will persist the data using a database. Model the database using an ER diagram.

a. The user chooses the option to arrange a meeting.

b. The system prompts user for the names of attendees.

c. The user types in a list of names.

d. The system checks that the list is valid.

e. The system prompts the user for meeting constraints.

f. The user types in meeting constraints.

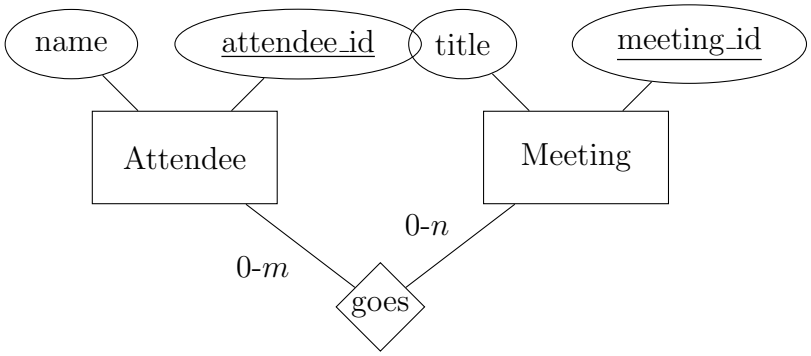
g. The system searches the calendars for a date that satisfies the constraints.

h. The system displays a list of potential dates.

i. The user chooses one of the dates.

j. The system writes the meeting into the calendar.

k. The system emails all the meeting participants informing them of them appointment



PK(goes)=(attendee_id, meeting_id)

6 Normalization

6.1 Exercise 1

Given a relation schema with one single relation $R(\underline{A}, \underline{B}, C, D)$ and a functional dependency $B \rightarrow CD$. Decomposer R so it satisfies the BCNF.

`Answer:`

```
R1(A,B)
R2(B,C,D)
```

6.2 Exercise 2

Given a relation $R(A, B, C, D, E)$ as well as some data:

A	B	C	D	E
1	4	4	3	2
4	1	2	1	2
2	5	5	4	3
1	4	2	1	2
2	5	4	3	2
4	1	4	3	2
3	4	3	3	3

a. Find at least 3 independent, non-trivial functional dependencies that do not conflict with the given data.

b. Can you be sure, if the functional dependencies found in the previous exercise are the "real" functional dependencies, defined on the schema?

Answer:

$A \rightarrow B$
$C \rightarrow D$
$AC \rightarrow E$

No, without further information there is no way to know if the proposed FD are real. A tuple such as (1, 5, 1, 1, 1) would not satisfy $A \rightarrow B$

6.3 Exercise 3

Given following scheme $R(A, B, C, D, E)$. Which normal form is violated in the according schema? What exactly causes the violation?

Case 1

$A \rightarrow BC$
$D \rightarrow E$

Case 2

$A \rightarrow BCD$
$D \rightarrow E$

Answer:
Case 1: No 2NF. E is not prime and depends on part of the PK (A, D). Case 2: No 3NF. E is not prime and depends on D that is nor candidate key.

6.4 Exercise 4

Given a Schema with one single relation $R(A, B, C, D, E.F)$ and a set of functional dependencies

$BC \rightarrow DC$
$B \rightarrow E$
$D \rightarrow EF$
$FC \rightarrow E$
$C \rightarrow A$
$F \rightarrow E$

- find PK
- Find a minimal equivalent set of dependencies
- Normalize the schema into BCNF.

Answer:
PK = BC

cover
$BC \rightarrow D$
$B \rightarrow E$
$D \rightarrow F$
$C \rightarrow A$
$F \rightarrow E$

BCNF

```
R1(B, C, D) ; b,c -> D
R2(B, E) ; B -> E
R3(D, F) ; D -> F
R4(C, A) ; C -> A
F -> E lost.
```

3NF preserving FD

```
R1(B, C, D) ; B, C -> D *
R2(B, E) ; B -> E *
R3(D, F) ; D -> F *
R4(C, A) ; C -> A *
R5(F, E) ; F -> E *
```

6.5 Exercise 5

Given the relation schema $R = (A, B, C, D, E)$ and the set of functional dependencies:

$A \to BC$
$CD \to E$
$B \to D$
$E \to A$

Decompose the relation R so that the resulting set of relations satisfy BCNF.

Answer:

PK = A or E or BC or CD

R is not Boyce Codd since we have $B \to D$ and B is not a candidate key

So we decompose R in:

$R1 = ABCE$
$R2 = BD$

The remaining dependencies are:

for $R1$

$A \to BC$
$E \to A$

for $R2$

$B \to D$.

Since A and E are candidate keys $R1$ and $R2$ satisfy Boyce-Codd NF. $PK(R2) = B$

6.6 Exercise 6

Suppose you are given the following functional dependencies:

```
FD1. ZIP -> { city, state }
FD2. { street, number, city, state } -> ZIP
```

we assume they may exist two `cities` with the same name in different `states`

Then:

a. Give a primary key for the relation `Address(street, number, floor, city, state, ZIP)`

b. Normalize the relation `Address` so it satisfies the BCNF form. Specify the primary keys in the normalized relations.

c. Are we loosing any dependency?

Answer:

6.6.1 Primary keys

```
PK1 = { street, number, floor, city, state }
PK2 = { street, number, floor, ZIP }
```

6.6.2 Normalice to BCNF

If we choose PK1 as primary key then the relation satisfies the 3NF (*city*, *state* and ZIP are PRIME attributes).

We need to decompose the relation as follows

```
R1 = { street, number, floor, ZIP }
R2 = { ZIP, city, state }
```

You should realize that we are loosing functional dependencies FD2

6.7 Exercise 7.

Given The relation R = (A,B,C,D,E). Assume the following functional dependencies holds:

```
FD1. A -> BC
FD2. B -> D
FD3. A -> A
```

Then:

a. Give a primary key for R

b. Normalize the relation so it satisfies the BCNF form. Specify the primary keys in the normalized relations. Specify the new functional dependencies.

c. Are we loosing any dependency?

Answer:

Case 1

We use `A -> BC` then

```
R1. (A,B,C)
R2. (A, D, E)
No possible to recover BC -> D
```

Case 2

We use `A -> BC -> D, A -> BCD` then

```
R1. (A,B,C,D)
R2. (A, E)
No dependency loss BC -> D
```

we apply now `BC -> D`

```
R11. (B,C,D)
R12. (A,B,C)
R2. (A, E)
We are good
```

7 Relational Algebra and Calculus

7.1 Exercise 1

Consider the following schema:
$Suppliers(\underline{sid}, sname, address)$
$Parts(\underline{pid}, pname, color)$
$Catalog(\underline{sid\hat{}}, \underline{pid\hat{}}, cost)$

Write the following queries in relational algebra (RA), tuple relational calculus (RC), and SQL:

7.1.1 Find the names of suppliers who supply some red part

SQL:

```
SELECT sname
FROM Parts NJ Catalog NJ Suppliers
WHERE color = 'red'
```

RA

$\Pi_{sname}\sigma_{color='red'}Parts \bowtie Catalog \bowtie Suppliers$

RC

$\{S.sname \mid Suppliers(S) \wedge (\exists C \exists P$
$(Catalog(C) \wedge Parts(P) \wedge$
$(P.color =' red') \wedge$
$(P.pid = C.pid) \wedge$
$(C.sid = S.did)))\}$

7.1.2 Find the sids of suppliers who supply some red or green part

SQL:

SELECT sid FROM Catalog NJ Parts
WHERE color = 'red' OR color = 'green'

RA

$\Pi_{sid}\sigma_{color='red' \vee color='green'} Parts \bowtie Catalog$

RC

$\{C.sid \mid Catalog(C) \wedge (\exists P$
$(Parts(P) \wedge$
$((P.color =' red') \vee (P.color =' green')) \wedge$
$(P.pid = C.pid)$
$)\}$

7.1.3 Find the sids of suppliers who supply some red part and some green part.

SQL_1:

SELECT sid
FROM Catalog C1, Catalog C2, Parts P1, Parts P2
WHERE C1.pid = P1.pid AND C2.pid = P2.pid
AND C1.sid = C2.sid AND P1.color='red' AND P2.color='green'

SQL_2:

SELECT sid FROM Catalog C, Parts P WHERE C.pid = P.pid AND P.color='red'
INTERSECTS
SELECT sid FROM Catalog C, Parts P WHERE C.pid = P.pid AND P.color='green'

RA

$A \leftarrow \Pi_{sid}\sigma_{color='red'}Parts \bowtie Catalog$
$B \leftarrow \Pi_{sid}\sigma_{color='gree'}Parts \bowtie Catalog$
$A \cap B$

RC_1

$\{C1.sid \mid Catalog(C1) \wedge (\exists P1 \exists P2 \exists C2($
$Parts(P1) \wedge Parts(P2) \wedge Catalog(C2)$
$\wedge(P1.color =' red') \wedge (P2.color =' green')$
$\wedge(P1.pid = C1.pid) \wedge (P2.pid = C2.pid)$
$\wedge(C1.sid = C2.sid)$
$))\}$

RC_2

$\{C1.sid \mid Catalog(C1) \wedge (\exists P1($
$Parts(P1) \wedge (P1.color =' red' \wedge (P1.pid = C1.pid) \wedge (\exists C2($
$Catalog(C2) \wedge (C1.sid = C2.sid) \wedge (existsP2($
$Parts(P2) \wedge (P2.color =' green' \wedge (P2.pid = C2.pid)$
$)))))))\}$

7.1.4 Find the sids of suppliers who supply every part

SQL_1:

```
CREATE view parts_per_supplier as
SELECT sid, count(*) as number
FROM Catalog
GROUP BY sid


CREATE view parts_number as
SELECT count(*) as number
FROM Part
```

SELECT sid
FROM parts_per_supplier NJ parts_number

SQL_2:

```
SELECT C1.sid
FROM Catalog C1
WHERE NOT EXISTS (SELECT P.pid
                  FROM Parts P
                  WHERE NOT EXISTS (SELECT C2.sid
                                    FROM Catalog C2
                                    WHERE C2.sid = C1.sid
                                      AND C2.pid = P.pid))
```

RA

$\Pi_{sid,pid}Catalog/\Pi_{pid}Parts$

RC

$\{S.sid \mid Supplier(S) \wedge (\forall P$
$(Parts(P) \wedge (\exists C$
$(Catalog(C) \wedge (S.sid = C.sid)$
$\wedge(P.pid = C.pid)$
$)))\}$

7.1.5 Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid

SQL:

SELECT c1.sid, c2.sid
FROM Catalog c1, Catalog c2

```
WHERE c1.cost >= c2.cost
AND c1.sid <> c2.sid
AND c1.pid = c2.pid
```

RA

$C1 \leftarrow Catalog$
$C2 \leftarrow Catalog$
$\Pi_{C1.sid,C2.sid}\sigma_{C1.pid=C2.pid \wedge C1.sid \neq C2.sid \wedge C1.cost >= C2.cost} C1 \times C2$

RC_1

$\{C1.sid \mid Catalog(C1) \wedge (\exists C2($
$Catalog(C2) \wedge (C1.sid \neq C2.sid)$
$\wedge(C1.pid = C2.pid)$
$\wedge(C1.cost >= C2.cost)$
$))\}$

>= cover the "patological" case in which all parts have the same cost.

7.1.6 Find the pids of the most expensive parts supplied by supplier named Yosemite Sham (may be 2 parts have the same price)

SQL_1:

```
CREATE VIEW maxCost AS
SELECT max(cost) as cost
FROM Suppliers NJ Parts NJ Catalog
WHERE sname = 'Yosemite Sham';


SELECT *
FROM maxCost NJ Catalog;
```

SQL_2:

```
SELECT C.pid
FROM   Catalog C NJ Suppliers S
WHERE  S.sname = 'Yosemite Sham'
  AND  C.cost >= ALL (Select C2.cost
                      FROM Catalog C2 NJ Suppliers S2
                      WHERE S2.sname = 'Yosemite Sham'
                      )
```

RA:

$maxCost \leftarrow G_{MAX(cost)}\sigma_{sname='YosemiteSham'}Suppliers \bowtie Parts \bowtie Catalog$
$\rho(maxCost(cost))maxCost$
$maxCost \bowtie Catalog$

RC_1

$\{C.pid \mid Catalog(C) \wedge (\exists S($
$Supplier(S) \wedge (S.name =' YosemiteSham') \wedge (S.sid = C.sid)$
$\wedge(NOT\exists C2(Catalog(C2)$
$\wedge(C2.sid = S.sid) \wedge (C2.cost > C1.cost)$
$)))))\}$

7.1.7 Find the pids of parts supplied by at least two different suppliers

SQL

```
SELECT C1.pid
FROM Catalog C1, Catalog C2
WHERE C1.sid <> C2.sids
  AND C1.pid = C2.pid;
```

RA:

$C1 \leftarrow Catalog$
$C2 \leftarrow Catalog$
$\Pi_{C1.pid}\sigma_{C1.sid \neq C2.sid \wedge C1.pid = C2.pid} C1 \times C2$

RC

$\{C1.pid \mid Catalog(C1) \wedge (\exists C2($
$Catalog(C2) \wedge (C1.pid = C2.pid) \wedge (C1.sid \neq C2.sid)$
$))\}$

7.2 Exercise 2

Given two relations R1 and R2, where R1 contains N1 tuples and R2 contains N2 tuples, and $N2 > N1 > 0$, give the maximum and minimum possible sizes (in tuples) for the result relation produced by each of the following relational algebra expressions. In each case, state any assumptions about the schemas for R1 and R2 that are needed to make the expression meaningful.

7.2.1 $R1 \cup R2$

```
Assumption: R1 and R2 are union compatible
Min: N2
Max: N1 + N2
```

7.2.2 $R1 \cap R2$

```
Assumption: R1 and R2 are union compatible
Min: 0
Max: N1
```

7.2.3 $R1 - R2$

```
Assumption: R1 and R2 are union compatible
Min: 0
Max: N1
```

7.2.4 $R1 \times R2$

```
Assumption: R1 and R2 are union compatible
Min: N1 * N2
Max: N1 * N2
```

7.2.5 $\sigma_{a=5}R1$

```
Assumption: R1 has an attribute named a
Min: 0
Max: N1
```

7.2.6 $\Pi_a R1$

```
Assumption: R1 has an attribute named a
Min: 1
Max: N1
```

7.2.7 $R1/R2$

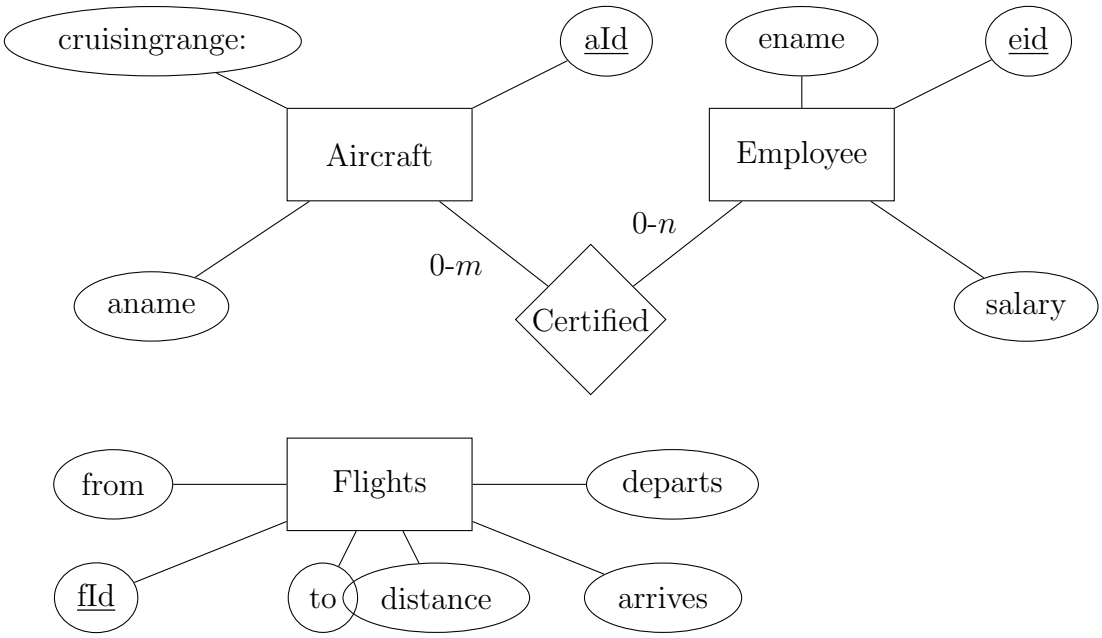
```
Assumption: The set of attributes of R2 is a subset of the attributes of
R1
Min: 0
Max: N1
```

7.2.8 $R2/R1$

```
Assumption: The set of attributes of R1 is a subset of the attributes of
R2
Min: 0
Max: N2/N1
```

7.3 Exercise 3

7.3.1 Convert the following E-R diagram to a relational model. Identify explicitly primary and foreign keys



Flights(flno, from, to, distance, departs, arrives)
Aircraft(aid, aname, cruisingrange)
Employees(eid, ename, salary)
Certified(eid∧, aid∧)

Write the following queries in relational algebra (RA), tuple relational calculus (RC), and SQL:

7.3.2 Identify the flights that can be piloted by every pilot whose salary is more than $100,000. Note: use 'distance' and 'cruisingrange'

SQL:

```
SELECT FId
FROM Aircraft NJ Certified C NJ  Employees E, Flights F
WHERE distance < cruisingrange
  AND salary > 100,000
```

RA

$\Pi_{fId}\sigma_{distance<cruisingrange \wedge salary>100,000} Aircraft \bowtie Certified \bowtie Employees \times Flights$

RC

$\{F.pid \mid Flights(F) \wedge (\exists A($
$Aircrafts(A) \wedge (F.distance < A.cruisingrange)$
$\exists C(Certified(C) \wedge (C.aid = A.aid)$
$\exists E(employee(E) \wedge (E.eid = C.eid) \wedge (E.salary > 100000)$
$)))))\}$

7.4 Find the eids of employees who are certified for exactly two aircrafts.

SQL:

SELECT eid FROM Certified GROUP BY eids HAVING count(*) = 2

RA

$\rho(A(number)) \leftarrow eidG_{count(*)}Certified$
$\Pi_{eid}\sigma_{number=2}A$

RC

$\{C1.eid \mid Certified(C1) \wedge (\exists C2($
$Certified(C2) \wedge (C1.eid = C2.eid) \wedge (C1.aid \neq C2.aid)$
$NOT\exists C3(Certified(C3) \wedge (C1.eid = C3.eid) \wedge ((C3.aid \neq C1.aid) \wedge$
$(C3.aid \neq C2.aid))$
$)))\}$

8 Functional Dependencies

8.1 Exercise 1

The database of a company that buys and shells shares in the stock market contains a relation $R(b, o, a, q, c, d)$ with the following attributes:
b: broker
o: broker's office
a: share
q: number of shares
c: client
d: dividend paid by share

We know that the following functional dependencies are satisfied:
$a \rightarrow d$
$c \rightarrow b$
$c, a \rightarrow q$
$b \rightarrow o$

a. Provide a possible primary key for R.

b. Does R satisfy the 2NF? If it does not decompose R so it does satisfy it.

c. Does R satisfy the 3NF? If it does not decompose R so it does satisfy it.

Let us assume that R is decomposed in the following two relations:
$R1(c, a, q, d)$
$R2(c, b, o)$

d) Does $R1$ and $R2$ satisfy the 3NF? If it does not decompose R so it does satisfy it.

e) Describe a redundancy that is present in the original relation R but not in de decomposition obtained in exercise (c).

ANSWERS:
a) possible PK $\{a, c\}$
b) No 2NF d depends only on a
c) NO 3NF since R does not satisfy 2NF. $r1(c, a, q)$, $r2(a, d)$, $r3(c, b)$, $r4(b, o)$
d) No, in $R2$, o (non-prime attribute) depends only on b (no candidate key)
e) If a brokers has several clients his office information appears many times in R

8.2 Exercise 2

Answer to the following questions. Please, explain your answers:

A) Given a relation $R1(A, B, C, D, E, F, G)$ with primary key B and the following functional dependencies:
$DF1 : B \rightarrow A, C, D, E, F, G$
$DF2 : E \rightarrow F, G$

What is the highest normal form satisfied by $R1$?

B) Given a relation $R2(A, B, C, D)$ with primary key $\{A, B\}$ and the following functional dependencies:
$DF1 : A, B \rightarrow C, D$
$DF2 : C \rightarrow B$

What is the highest normal form satisfied by $R2$?

Decompose $R2$ so it does satisfy Boyce-Codd normal form. Provide the functional dependencies for the decomposition.

ANSWERS:

A) $R1$ satisfies the 1FN since (a) $R1$ has a primary key, (b) attributes are atomic and are not multi-valued. $R1$ satisfies the 2NF because all non-prime attributes fully depend on the primary Key. $R1$ does not satisfy the 3NF because the non-prime attribute G depend on E that is not a candidate key. $R1a(B, A, C, D, E)$ and $R1b(E, F, G)$ satisfy the 3FN.

B) $R2$ satisfies the 1FN since (a) $R2$ has a primary key, (b) attributes are atomic and are not multi-valued. $R2$ satisfies the 2NF because all non-prime attributes fully

depend on the primary Key. $R2$ satisfies the 3NF because the is non-prime attribute do that depends on another attribute that are not part of a candidate key. $R2$ does not satisfy the BCNF because there is one attribute B that depends on C and C is neither a primary key or a candidate key.

Decomposition: $R2a(A, C, D)$ and $R2b(C, B)$ with dependencies $AC \rightarrow D$ and $C \rightarrow B$ satisfy the BCNF.