SQL INTRODUCTION

contraseña: alumnodb

Boolean → True
→ False
→ NULL

to get the list of the databases that already exist.

```
> psql -U alumnodb -h localhost --list
> psql -U alumnodb -h localhost NOMBRE_DEL_DATABASE
edat => CREATE TABLE sample_table(id INTEGRER, b BOOLEAN);
```

table created

```
edat => \d sample_table
```

the table is on the screen now

```
edat => INSERT INTO sample_table VALUES (1, 't');
edat => SELECT * FROM sample_table;
edat => SELECT b, b2, b AND b2 FROM sample_table, sample_table2;
```

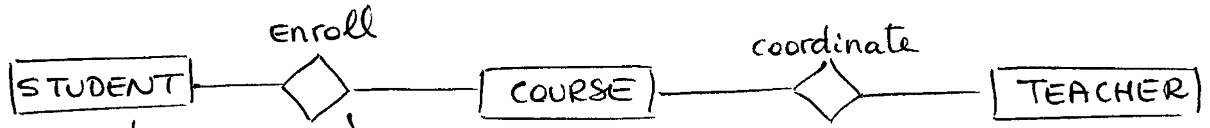
SUMMARY:

- DATA BASE MODELING (to be continued)
- SQL:
  1. DATA DEFINITION LANGUAGE
     create/drop tables
  2. DATA MANIPULATION LANGUAGE
     select / insert
  3. DATA CONTROL
     transition

AND	T	F	NULL
T	T	F	NULL
F	F	F	F
NULL	NULL	F	NULL

OR	T	F	NULL
T	T	T	T
F	T	F	NULL
NULL	T	NULL	NULL

1) DATABASE contains → DATA (structures)
→ RELATIONSHIPS



2) RELATIONAL DATABASE

ID	NAME

STUDENT_ID	COURSE_ID

ID	COURSE-NAME	COORDINATION

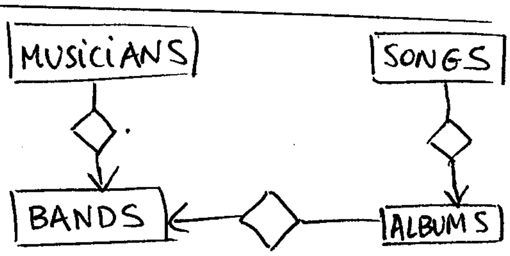
3) REQUIREMENTS

- Query } SQL
- Table } SQL
- Integrity
- Concurrency
- Transaction
- View security

- Primary key
- Foreign key

Exercise 1

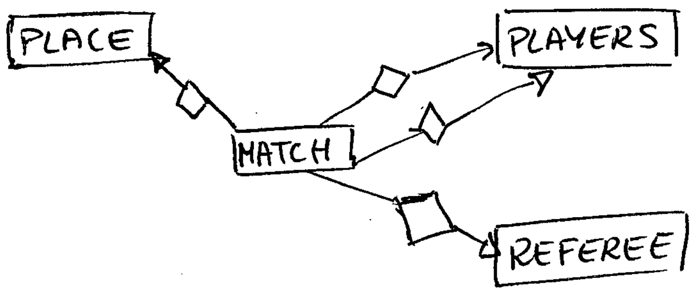
1st: Decide number of "things".



Exercise 2

For each Chess match

- Players
- Place
- Referee
- When
- Who is white/black



INTEGRITY CONSTRAINTS

- unique
- primary key
- not null
- check(name > 0) → for example
- foreign key
- default
- index

* Primary Key = unique + not null + index (ID)

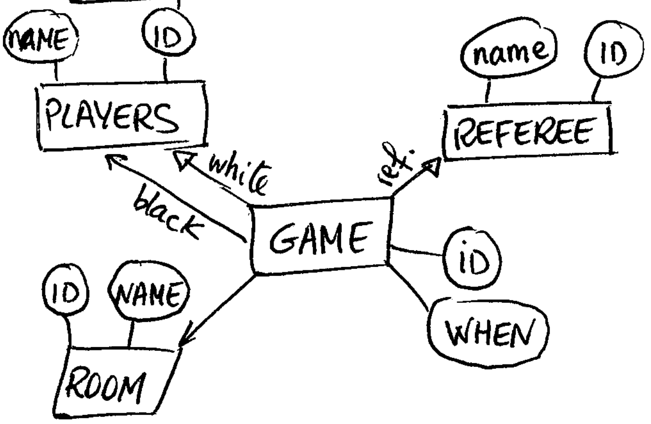
Example:

Example:

serial

```
CREATE TABLE Orders(
    OrderID int PRIMARY KEY,
    OrderNumber int NOT NULL UNIQUE,
    PersonID int REFERENCES Persons(PersonID));
```

Example




```
CREATE TABLE Player(
    ID INT PRIMARY KEY,
    name CHAR(64) NOT NULL
);
```

```
CREATE TABLE Referee(
    ID INT PRIMARY KEY,
    name CHAR(64) NOT NULL
);
```

```
CREATE TABLE Room(
    iD INT PRIMARY KEY,
    name CHAR(64) NOT NULL
);
```

```
CREATE TABLE Game(
    iD SERIAL PRIMARY KEY,
    when TIMESTAMP,
    Player_White INT REFERENCES PLAYER(iD),
    Player_Black INT REFERENCES PLAYER(iD),
    Room_iD INT REFERENCES ROOM(iD);
    Referee_iD INT REFERENCES REFEREE(iD)
);
```

char(N) → normalmente

varchar (N) → normalmente grande

F FLOAT

$\boxed{F=3}$

stored : 2'99999...

N NUMERIC

N= 2'999 identical to the precision you define

```
>SELECT 3.14159;
>SELECT 3.14159::FLOAT;
>SELECT 3.14159::NUMERIC;
>SELECT 3.14159::NUMERIC(10,2);
```

positions

nº of positions that are decimals

DATETIME
- → DATE: fecha
- → TIME: hora
- → TIMESTAMP: fecha y hora

INTERVAL

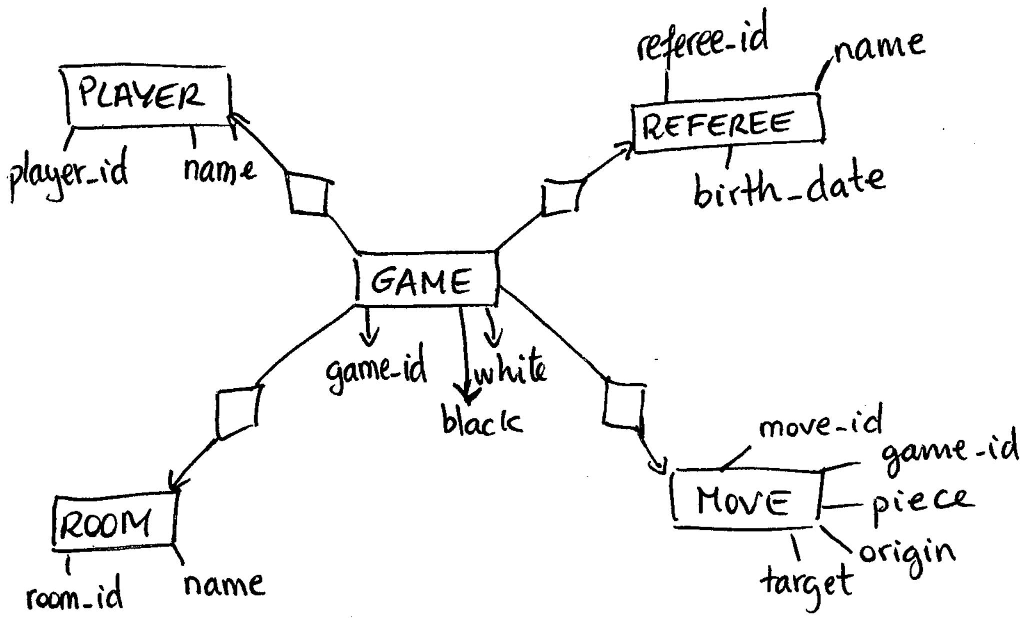
↳ ej: 1 day

- SUMMARY:

3 SELECT	R1.A1, R2.A3, ...	(columna 1, tabla 1; columna 3, tabla 2)
1 FROM	R1, R2, ..., RN ← tablas	
2 WHERE	EXPRESION1 AND/OR EXPRESION2	
4 ORDER BY	R1.A1. [ASC/DESC], R2.A3 [ASC/DESC]	

~~ste~~ select count(*) fom "name of a table";

EXAMPLE:



- Name of the player who had played in the room 'Red'.

SELECT player.name
FROM player, game, room
WHERE (player-id = white OR player-id = black) AND game.room_id = = room.~~id~~room_id AND room.name = 'Red'.

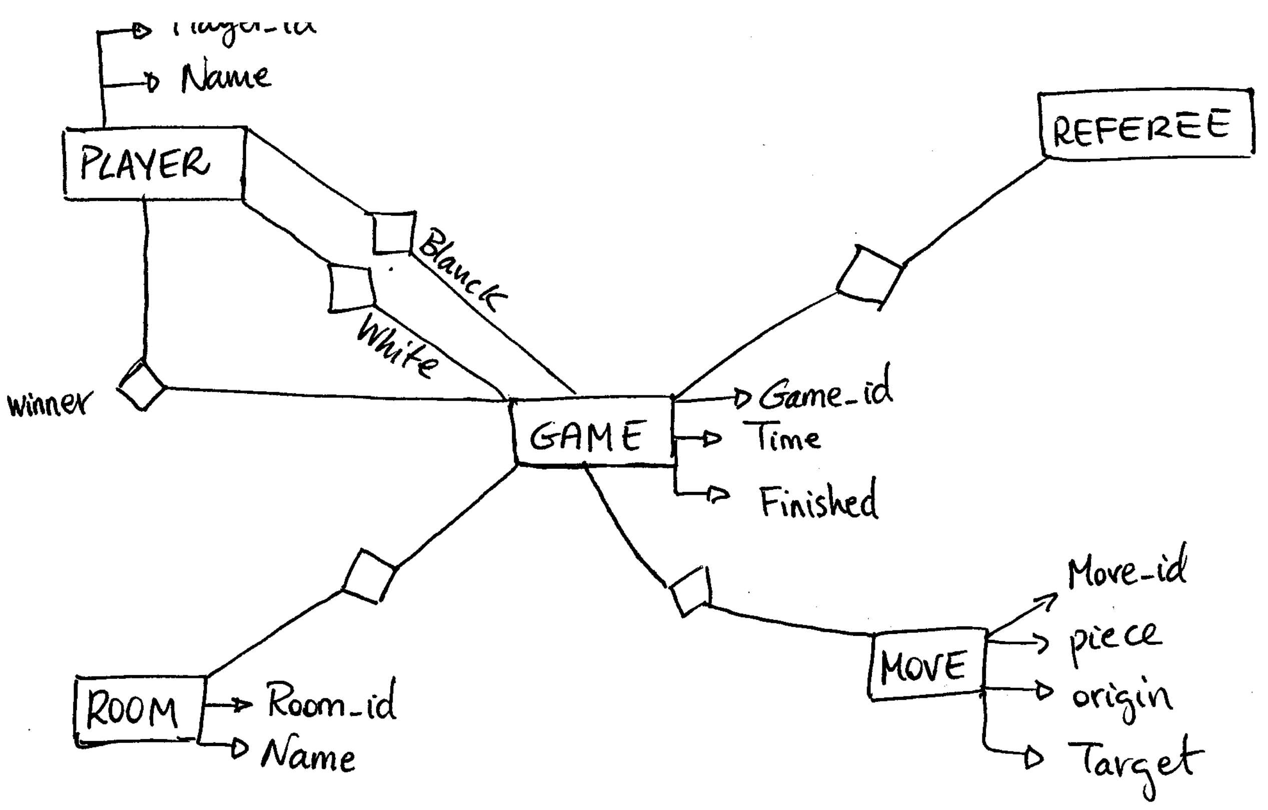
- List of the names of all referees but the oldest one.

```
SELECT distinct(R1.name)
FROM referee R1, referee R2
WHERE R1.birthday > R2.birthday;
```

- Players that have played with any player that has played against 'Betty'.

```
SELECT MAX(Game_id) AS LAST, MIN(Game_id)
FROM Game
WHERE Black > 7
GROUP BY Black
HAVING COUNT(*) > 5
ORDER BY LAST
```

count
max
min
avg



1. Number of games

```
SELECT COUNT(*) FROM GAME;
```

2. Number of games played by each player (player_id) as white.

```
SELECT COUNT(*), white
FROM Game
GROUP BY white;
```

3. Number of games played by each player (player_id).

```
CREATE VIEW AS Both_players AS
SELECT Black AS KK → etiqueta aleatoria
FROM GAME
UNION
SELECT White AS KK
FROM GAME;

SELECT COUNT(*), KK
FROM Both_players
GROUP BY KK;
```

4. Number of games played by each player (name).

```
CREATE VIEW AS Both-players AS
SELECT Black AS Player-id
FROM Game
UNION
SELECT white AS Player-id
FROM Game;
```

etiqueta

```
SELECT Name, player-id, count(*)
FROM Both-player NATURAL JOIN Player
GROUP BY Player-id, Name;
```

The result of a query that contents an aggregate function is a table. This table has as many columns as the attributes used by "GROUP BY" plus an extra column for each aggregation.

I.E.

```
SELECT MAX(ID), MIN(ID), AVG(ID), ID, A1
FROM Relations
GROUP BY ID,A1
```

5. Game-id of the last game.

```
SELECT MAX(Game-id)
FROM Game;
```

6. Player that has played more than five games as white.

```
SELECT White
FROM Game
GROUP BY White
HAVING COUNT(*) > 5;
```

7.) Count number of times the same movement happens.

```
SELECT COUNT(*), PIECE, ORIGIN, TARGET
FROM MOVE
GROUP BY PIECE, ORIGIN, TARGET
```

8.) Most popular movement (piece, origin, target)

```
CREATE VIEW Time_movements AS
   SELECT COUNT(*) AS TIMES, PIECE, ORIGIN, TARGET
   FROM MOVE
   GROUP BY PIECE, ORIGIN, TARGET;

SELECT MAX(TIMES)
FROM Time-movements
```

9.) Oldest referee (name)

a)
```
SELECT Referee.name
FROM Referee
ORDER BY Birthday[ASC]
LIMIT 1;
```

b)
```
CREATE VIEW LastBirthday AS
   SELECT Min(Birthday) AS Birthda
   FROM REFEREE;
SELECT Referee.name
FROM LastBirthday, Referee
WHERE LastBirthday.birthday =
   = Referee.birthday;
```

only if you want to use N

10.) Oldest referee (name) per year.

```
CREATE VIEW OLDEST_REFEREE AS
   SELECT Time, MIN(Birthday)
   FROM REFEREE NJ GAME
   GROUP BY Time;

SELECT Time, Name
FROM Referee NJ Oldest_Referee;
```

Pairs of players who have never played against.

```
SELECT  P1.Player_id, P2.Player_id
FROM  Player P1 , Player P2

EXCEPT
SELECT  White, Black
FROM  Game;
```

3 ER Modeling Concepts

3.1 Keys

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

3.2 Entities

Explain the difference between a weak and a strong entity set

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).

5 University register

A university office maintains data about: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; (d) instructors, including identification number, name, department, and title; and (f) The enrollment of students in courses and grades awarded to students in each course.

A diagram that models this problem can be seem in fig 1. If needed modify the diagram so we can record the marks the students get in different exams in different offerings (November exam, January Exam, June Exam, Lab exams, etc).

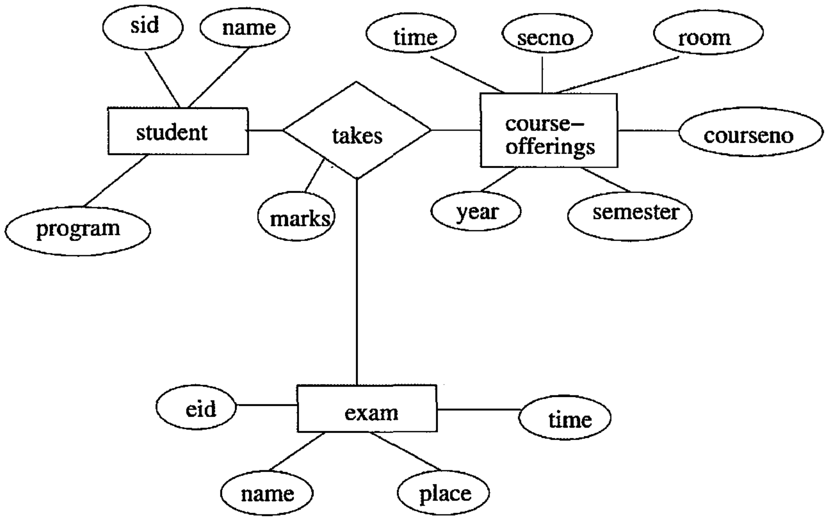


Figure 1: E-R diagram for a car-insurance company

3 ER MODELING CONCEPTS

3.1 Keys

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

~~The PRIMARY KEY is a constraint that identifies uniquely each record in a database table.~~

~~The CADIDATE KEY is a column or a set of columns that can identifies uniquely each record in a database table without referring to any other data.~~

SUPERKEY: Set of columns that are able to identify uniquely each record in a database table.

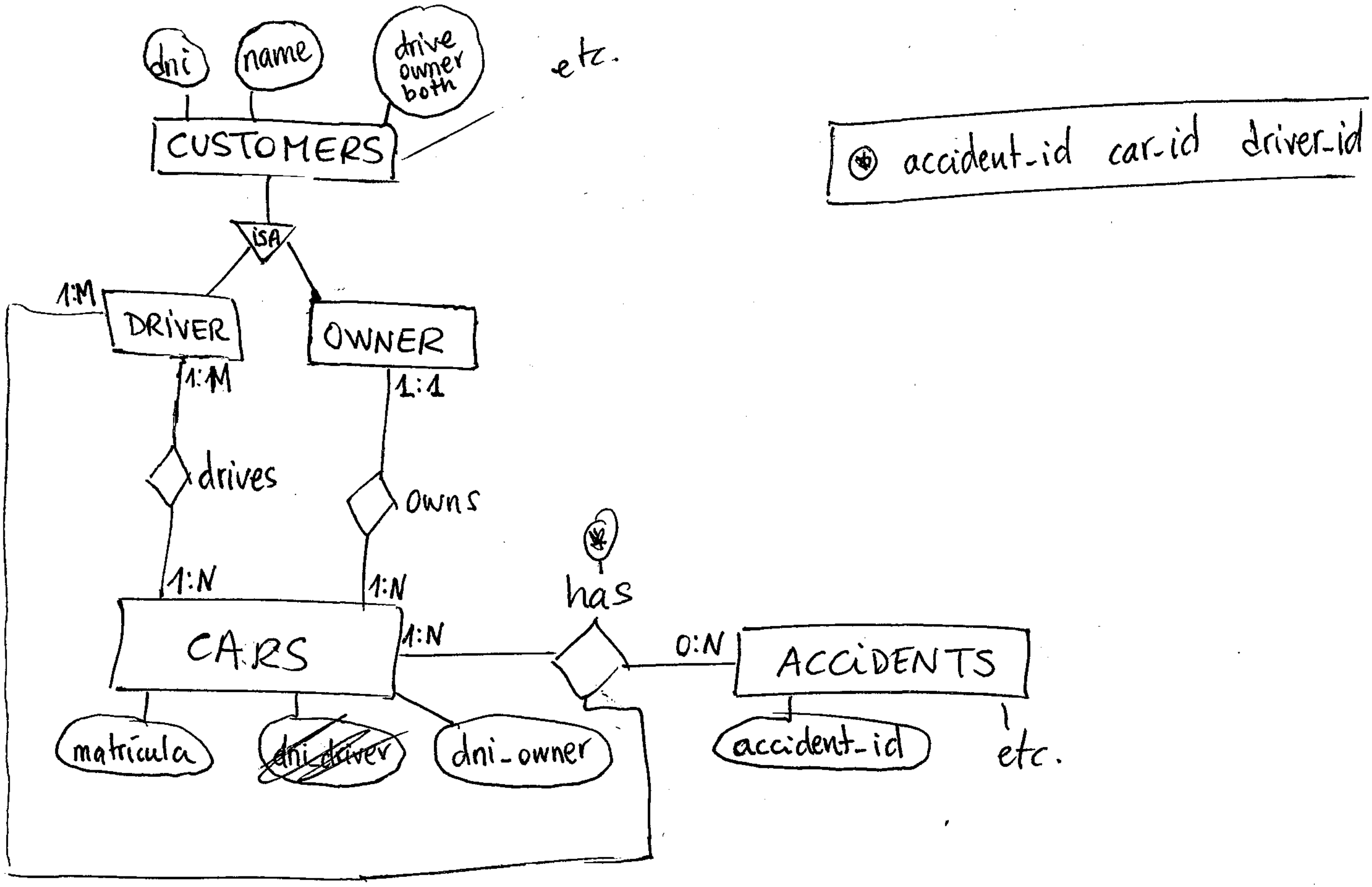
CANDIDATE KEY: Set of columns that are able to identify uniquely each record in a database, but if you remove just one of the columns that are 'candidate key' they are no longer able to identify uniquely each record.

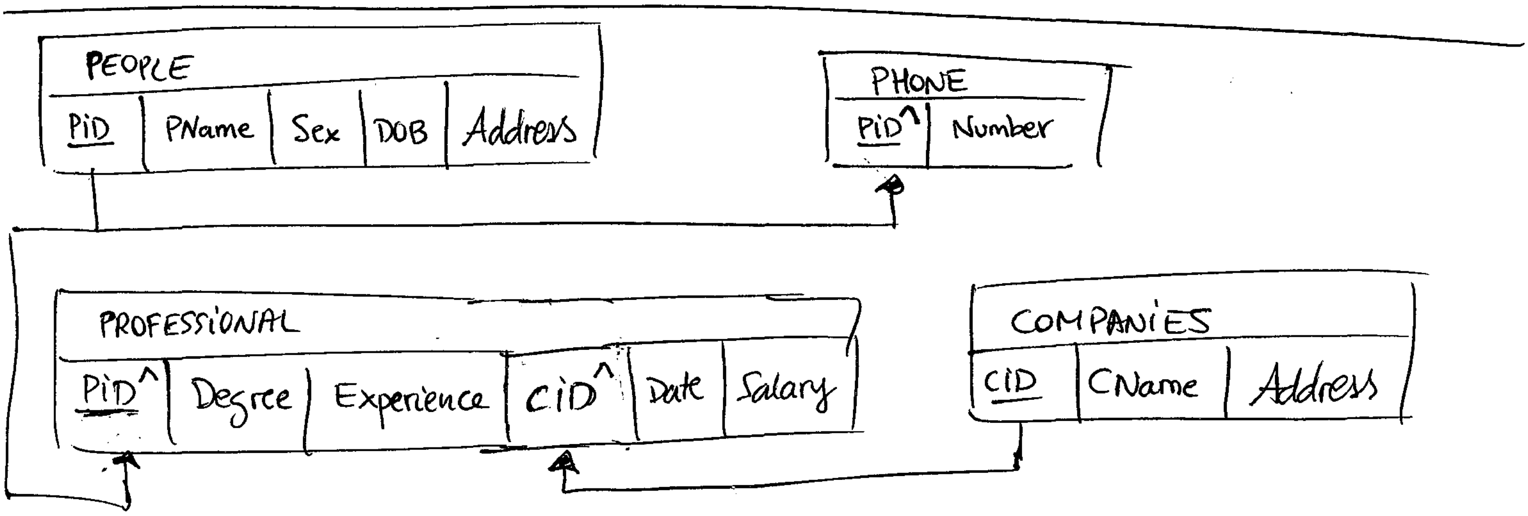
PRIMARY KEY: One of the candidate keys that you choose to implement a database (it is the chosen candidate key).

3.2 Explain the difference between a weak and a strong entity set.

A weak entity set needs an attribute from another entity to identify each record of the database table. A strong entity doesn't.

4





2 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^)
```

2.1 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

2.2 Married women that do not like her husband

2.3 Men that do not like any woman

2.4 Married women that do not like any married men

2.1

(1)

```
SELECT MlikesW.nameM, MlikesW.nameW
FROM   MlikesW NJ WlikesM;
```

→ CREATE VIEW LikeEachOther AS ( );

(2)

```
SELECT Men.nameM, Women.nameW
FROM   LikeEachOther L, Men M, Women W
WHERE  Men.age BETWEEN(30,40) AND Women.age BETWEEN(30,40) AND
   AND L.nameM = M.nameM AND L.nameW = W.nameW;
```

(3)

```
[2]
EXCEPT
SELECT NameM, nameW
FROM   Marriage;
```

2.2

```
SELECT NameW
FROM Marriage

EXCEPT

SELECT Marriage.NameW
FROM Marriage, WlikesM
WHERE Marriage.nameW = Wlikes.nameW AND Marriage.nameM = Wlikes.nameM;
```

2.3

```
SELECT NameM
FROM Men
EXCEPT
SELECT NameM
FROM MlikesW;
```

2.4

```
CREATE VIEW MW AS
SELECT NameW
FROM Marriage

CREATE VIEW MM AS
SELECT NameM
FROM Marriage

CREATE VIEW MW_LIKES_MM AS
SELECT NameW
FROM MW NJ MW_LIKES-MM NJ MM

SELECT NameW
FROM MW
EXCEPT
SELECT NameW
FROM MW-LIKES-MM;
```

STUDENT(Student_id, first_name, last_name, phone)

DEPARTMENT(Department_name, location)

INSTRUCTOR(Instructor_id, phone, last_name, first name, department_name_has↑, department_name_head↑)
↳ most of times it'll be NU

COURSE(Course_id, duration, course_name, instructor_id↑, department_id)

ENROLLED BY(Course_id↑, Student_id↑)

↳ Posible easier solution:
HEAD(Department_name↑, Instructor_id↑

NEW CHAPTER

¿How can we be sure that a database is well design?
¿Can we improve the actual model to a more efficient one?

Non prime attributes MUST depend on the primary key, whole primary key, nothing else than the primary key.

Example

funtional relationships:

$$BC \rightarrow DC \equiv BC \rightarrow D \wedge \underbrace{BC \rightarrow C}_{\text{useless}}$$

$$B \rightarrow E$$

$$D \rightarrow EF$$

$$FC \rightarrow E$$

$$C \rightarrow A$$

1st NF $R(A, \underline{B}, \underline{C}, D, E, F)$ $\Rightarrow PK(B, C)$

$R1(A, \underline{B}, \underline{C}, D, F)$

$R2(\underline{B}, E)$ $\boxed{B \rightarrow E}$

nd NF $\rightarrow$

- $R1.1(\underline{B}, \underline{C}, D, F)$
- $R1.2(\underline{C}, A)$ $\boxed{C \rightarrow A}$

3rd NF $\rightarrow$

- $R.1.1.1(\underline{D}, F)$ $\boxed{D \rightarrow EF}$
- $R.1.1.2(\underline{B}, \underline{C}, D)$ $\boxed{BC \rightarrow D}$
- $R2(\underline{B}, E)$
- $R1.2(\underline{C}, A)$