UNIVERSITÀ DEGLI STUDI DI PISA

CORSO DI LAUREA MAGISTRALE IN

DATA SCIENCE AND BUSINESS INFORMATICS

BASI DI DATI

PROF. GIORGIO GHELLI

ProjectEasyRegatta

Piyush Tada 605988

Francesco Salerno 534622

17/07/2021

Academic Year 2020/2021

Description of the domain

Keeping information about multiple regattas.
People class is divided in two non overlapping cover subclasses: Crew and Committee

Each crew member is related to only one boat. Each boat has at least one member.

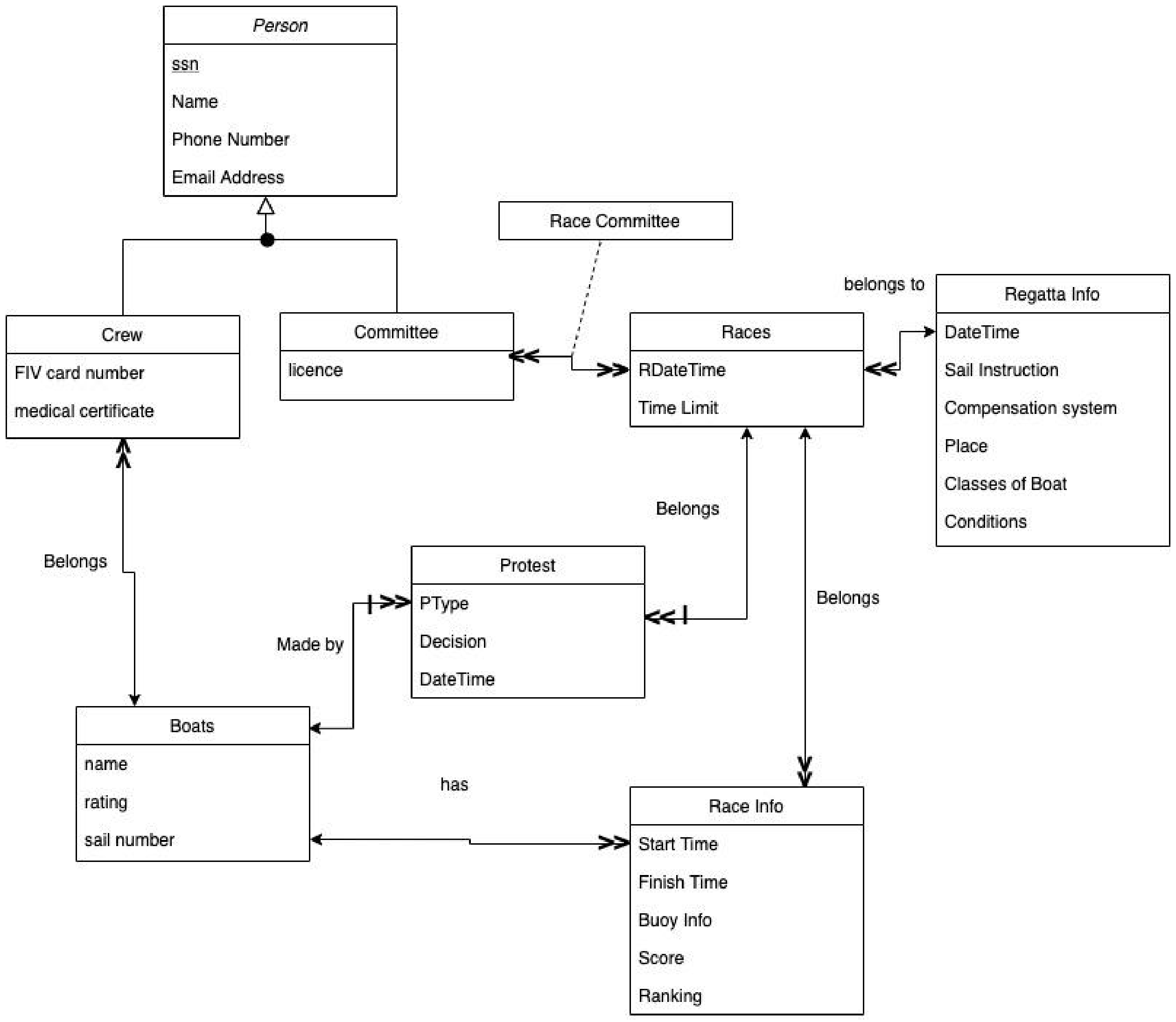
Race info reports information about a single race for a single boat. Different boats can be related to at least one or more race info records.
For each race info we have only one race, but for each race we have multiple race info equal to the number of boats participating in that race.

Races are related to only one Regata/event. Regattas are composed of multiple races.

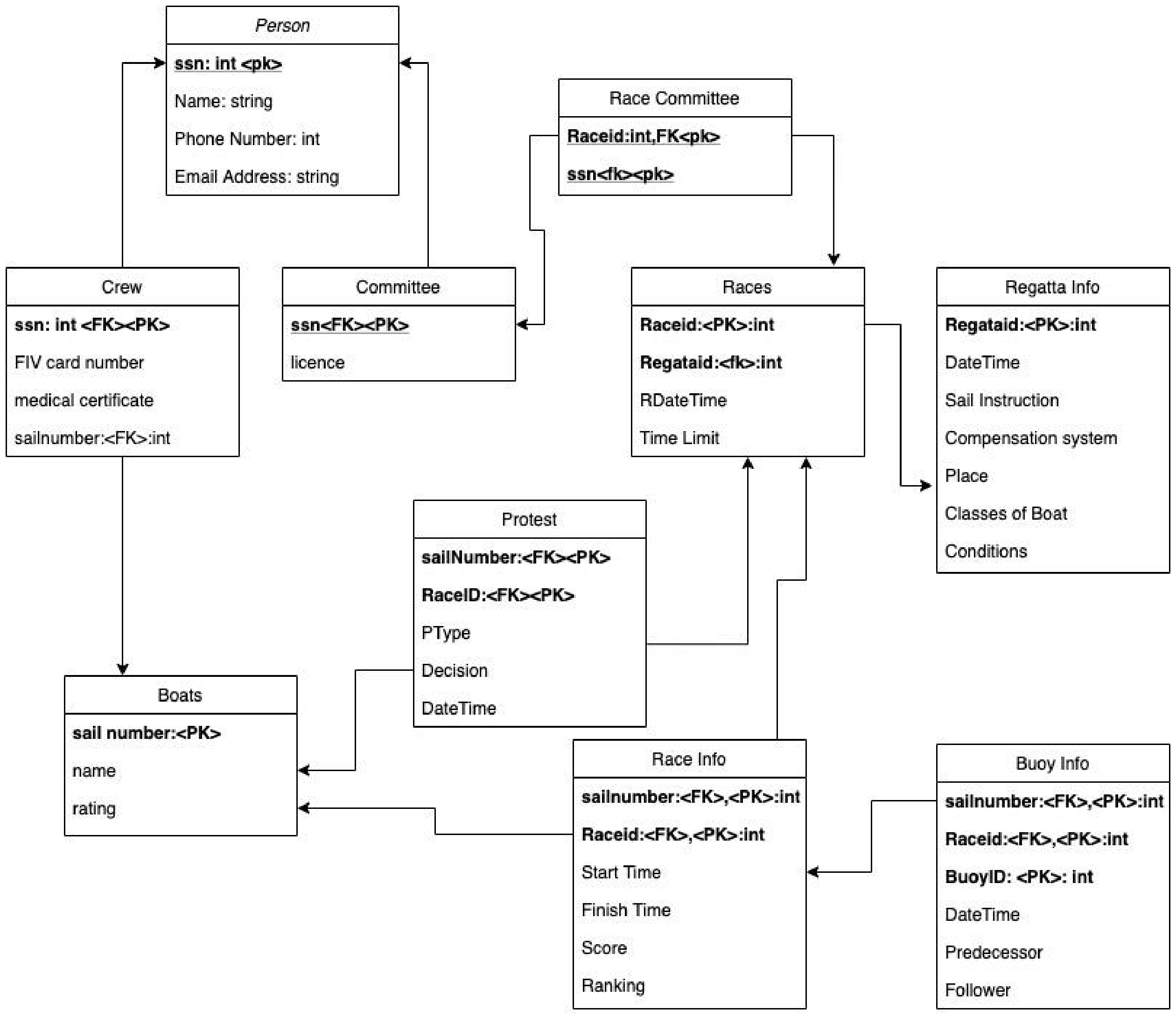
Each race has a race committee. Each committee member can be part of a diverse race committee.

Protest is made by a single boat. A boat can request zero or more protest. Each protest refers to exactly a race. A Race can have zero or more protests.

Conceptual scheme



Relational logic scheme



SPECIFICATION

About Reagatta info the attribute Sail Instruction, Compensation System, Classes boat, Conditions will be represented as key terms which refers to a text which is outside the schema.

Person(**ssn**, name, phone number, email)
Crew(**ssn***,FIV Card number, medical certificate,sailnumber*)
Committee(**ssn***,raceid*, licence)
Boats(**sail number,** name, rating)
Protest(**sail number*, RaceID*,**PType, Decision, DateTime)
Races(**RaceID**,RdataTime, RegattaID*,time limit)
Regatta info (**RegataID**, Date Time, Sail Instruction, Compensation System, Place, Classes boat, Conditions)
RaceInfo(**sailnumber***,**RaceID*,** start time, finish time, score, Ranking)
BuoyInfo(**sailnumber***,**RaceID*, BuoydID**,datetime, predecessor,follower)

DEPENDENCIES

Ssn --> name, phone number, email
Ssn -->FIV Card number, medical certificate,sailnumber
Ssn -->raceid, licence
sail number--> name, rating
sail number*, RaceID-->PType, Decision, DateTime
RaceID -->RdataTime, RegattaID*
RegataID--> Time LImit, Date Time, Sail Instruction, Compensation System, Place, Classes boat, Conditions
sailnumber,**RaceID -->** start time, finish time, buoy time, score, Ranking
sailnumber,**RaceID**, **BUoydID -->**datetime, predecessor, follower

The Functional Dependencies Respect The Normal Form Of Boyce Codd. Because For Every Functional Dependency X-->Y The X Is Part Of A Key.

4. Queries in SQL

a. use of projection, join and restriction;

Raceid of all the races happening in Pisa

```
SELECT r.raceid
FROM Race AS e, regata AS re
WHERE r.regataid=re.regataid and re.palace=”Pisa”;
```

b. use of group by with having, where and sort;

Find the sail number and FIV card number of the boat where more than 2 of the crew members have medical certificate A and sort by FIV card number.

```
SELECT sailnumber, FIV card number
FROM Crew
WHERE medical certificate=”A”
GROUP BY sailnumber
HAVING COUNT(ssn) > 2
ORDER BY FIV card number;
```

c. use of join, group by with having and where;

Find the name of the boat, sailnumber , number of members of the crew for the boat with rating B and crew member greater than 5. Sort by number of crew member

```
SELECT c.sailnumber,name,COUNT(ssn) AS Number of Member
FROM crew AS c ,boat AS b
WHERE c.sailnumber=b.sailnumber
AND b.rating=”B”
GROUP BY c.sailnumber
HAVING COUNT(ssn)>5
ORDER BY Number of Member;
```

d. use of nested select with existential quantification;

Find boat names who have done at least one protest.

```
SELECT name
FROM boats AS b
WHERE EXIST( SELECT *
              FROM protest AS p
              WHERE p.sailnumber=b.sailnumber );
```

e. use of nested select with universal quantification;

Find all the boats name who have done not have any protest

```
SELECT name
FROM boats AS b
WHERE NOT EXIST
             (SELECT *
              FROM protest AS p
              WHERE p.sailnumber=b.sailnumber );
```

f. use of quantified-comparison subqueries (something like_

WHERE xxx <ANY/ =ALL (SELECT FROM WHERE...))

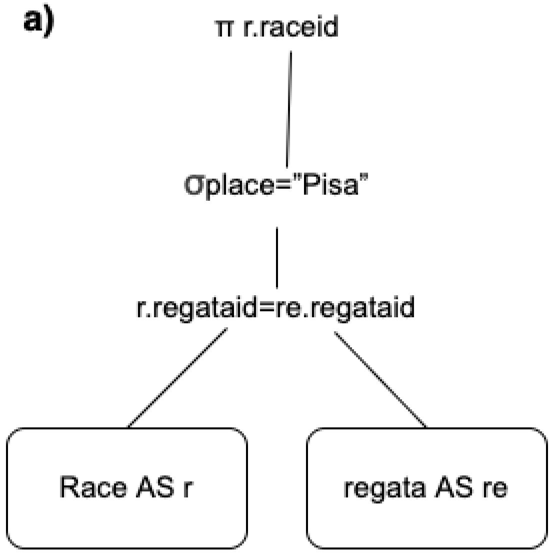
Find the race which the time limit is not the lowest.

```
SELECT b1.raceid
FROM race AS b1
WHERE time limit > ALL( SELECT b2.time limit
             FROM race AS b2)
```

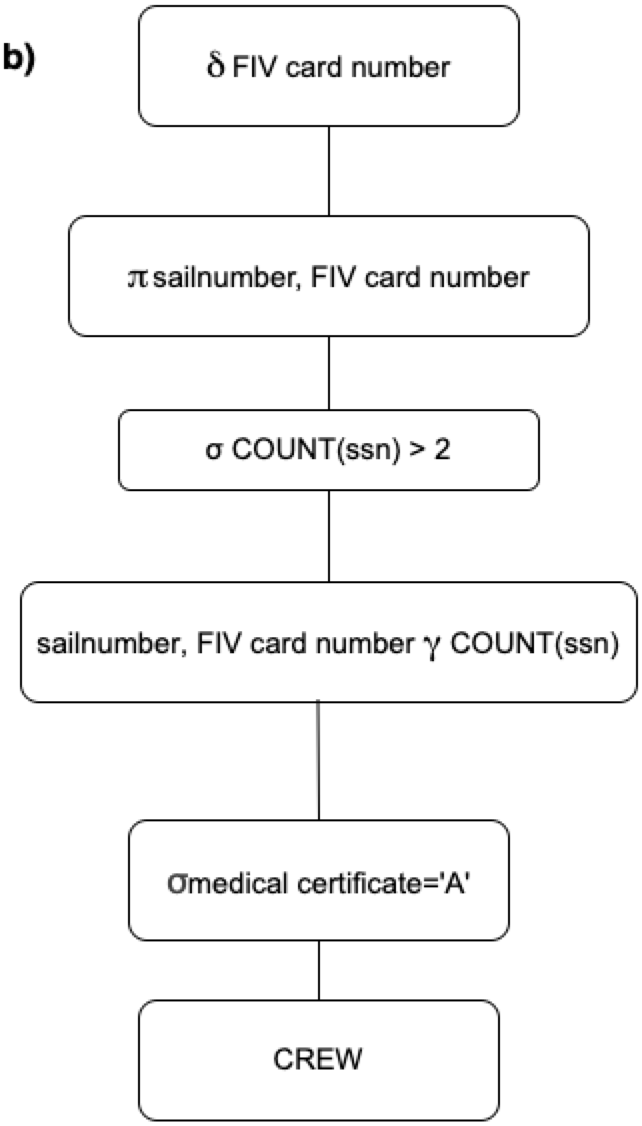
Access plans:

I. Write a logical access plan of queries a), b), c);

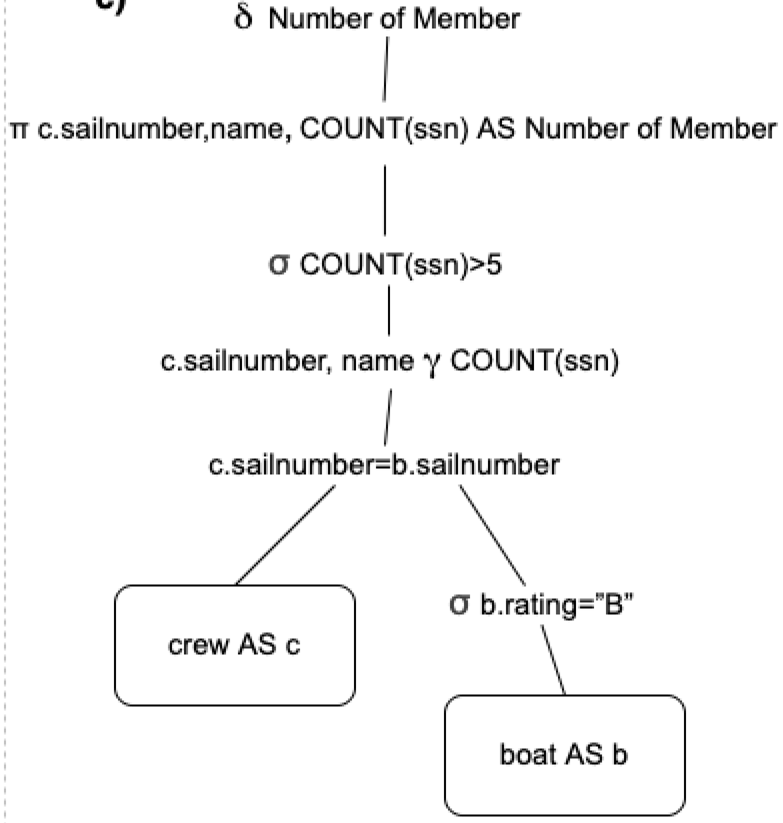
a)



b)

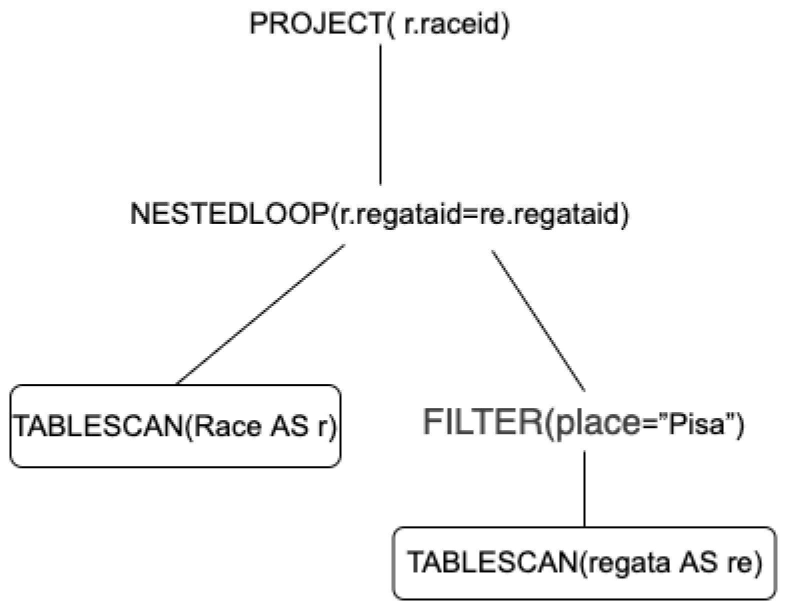

δ FIV card number
π sailnumber, FIV card number
σ COUNT(ssn) > 2
sailnumber, FIV card number γ COUNT(ssn)
σmedical certificate='A'
CREW

c)

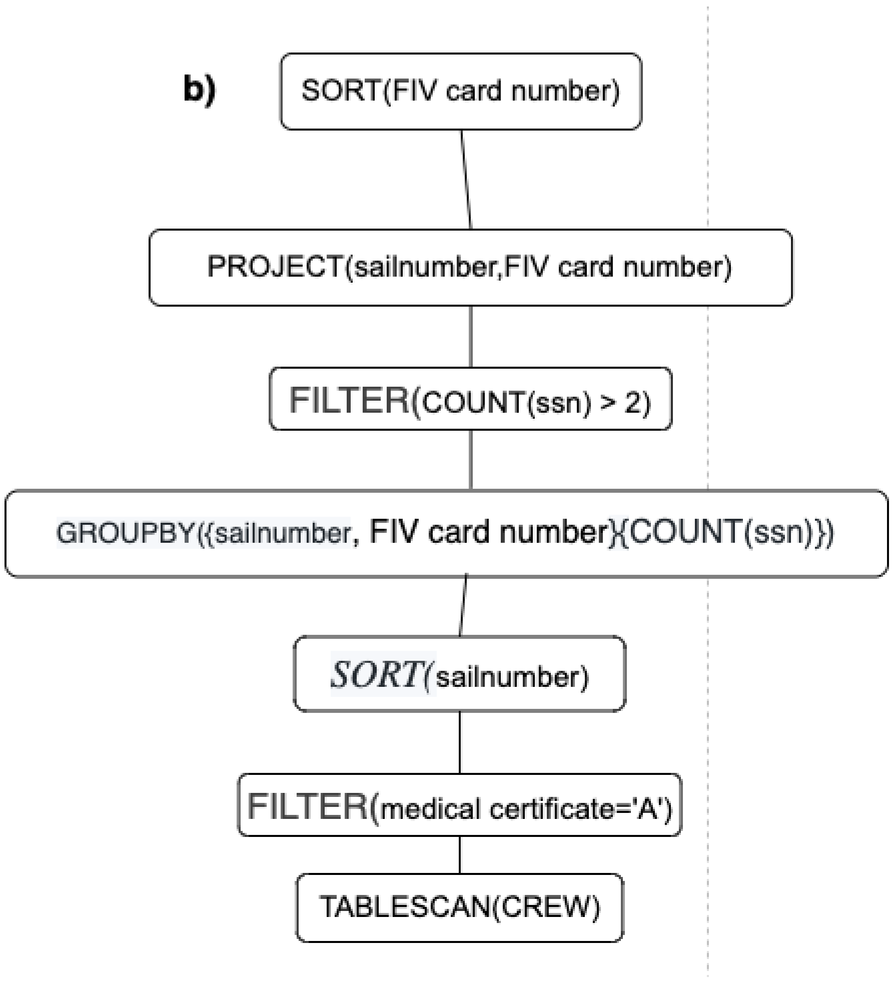

δ Number of Member
π c.sailnumber,name, COUNT(ssn) AS Number of Member
σ COUNT(ssn)>5
c.sailnumber, name γ COUNT(ssn)
c.sailnumber=b.sailnumber
crew AS c
σ b.rating="B"
boat AS b

II. Write an efficient physical access plan for the three logical access plans in point I that does not use indexes and (optional) check if sorting before Group By can be avoided;

a)

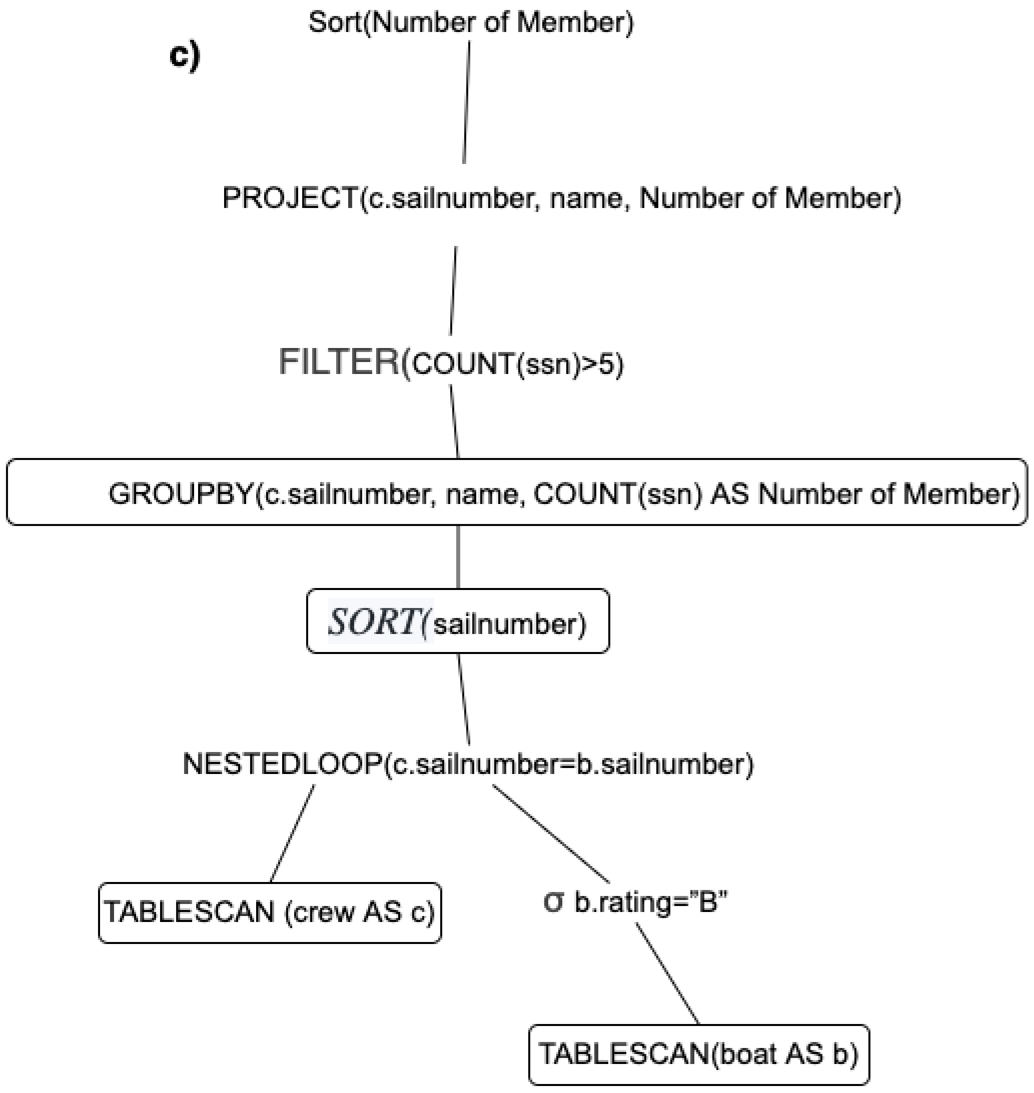


b)



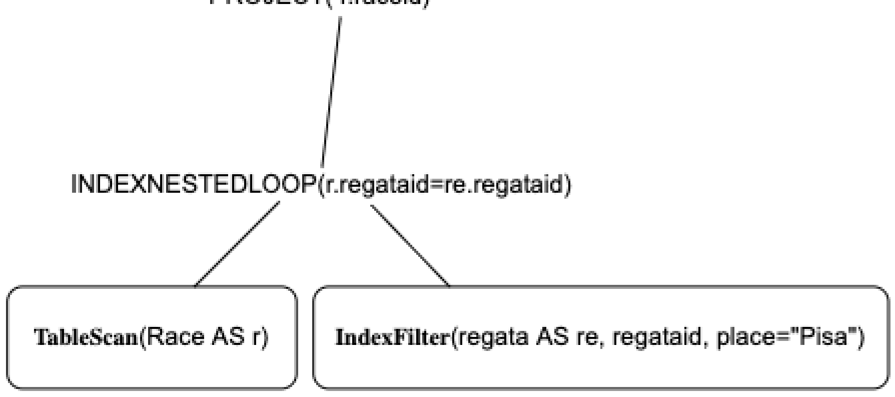
SORT(FIV card number)
PROJECT(sailnumber,FIV card number)
FILTER(COUNT(ssn) > 2)
GROUPBY({sailnumber, FIV card number}{COUNT(ssn)})
SORT(sailnumber)
FILTER(medical certificate='A')
TABLESCAN(CREW)

c)

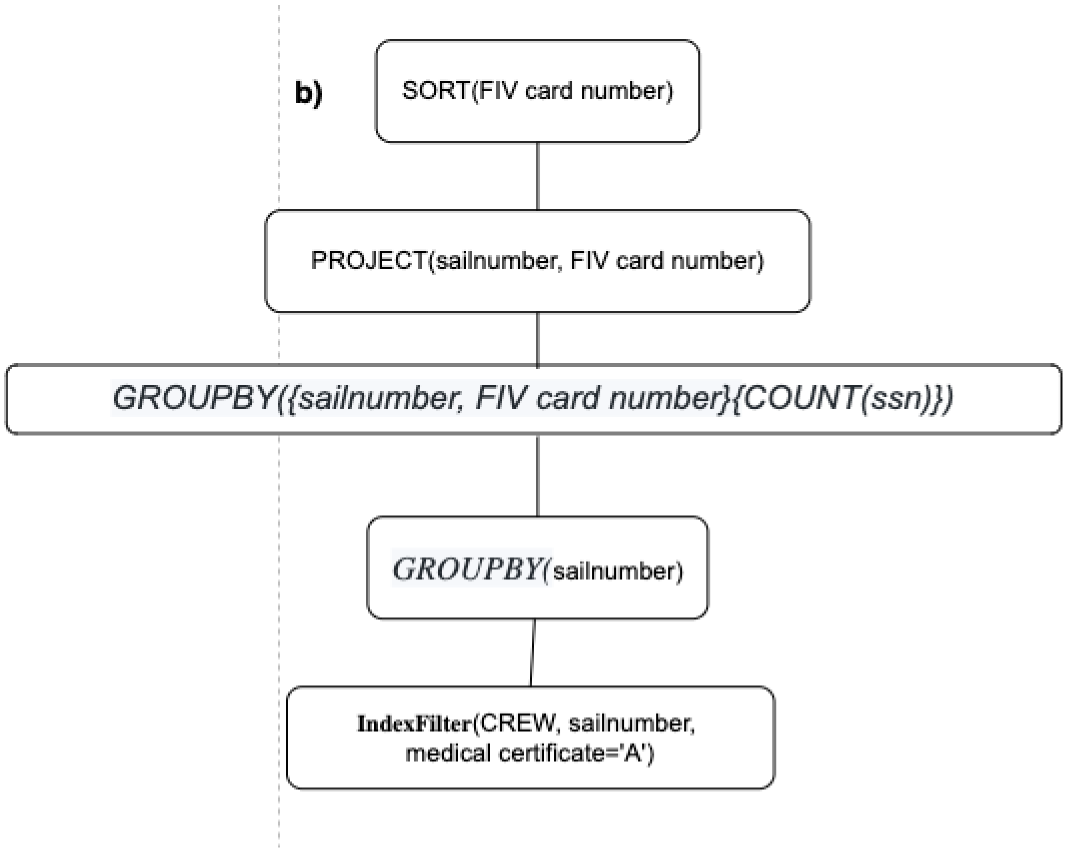

Sort(Number of Member)
PROJECT(c.sailnumber, name, Number of Member)
FILTER(COUNT(ssn)>5)
GROUPBY(c.sailnumber, name, COUNT(ssn) AS Number of Member)
SORT(sailnumber)
NESTEDLOOP(c.sailnumber=b.sailnumber)
TABLESCAN (crew AS c)
σ b.rating="B"
TABLESCAN(boat AS b)

III. Write an efficient physical access plan for the three logical access plans in point I that make use of two indexes (or in any case the maximum number of possible indexes), and (optional) check if the sort before the Group By can be avoided.

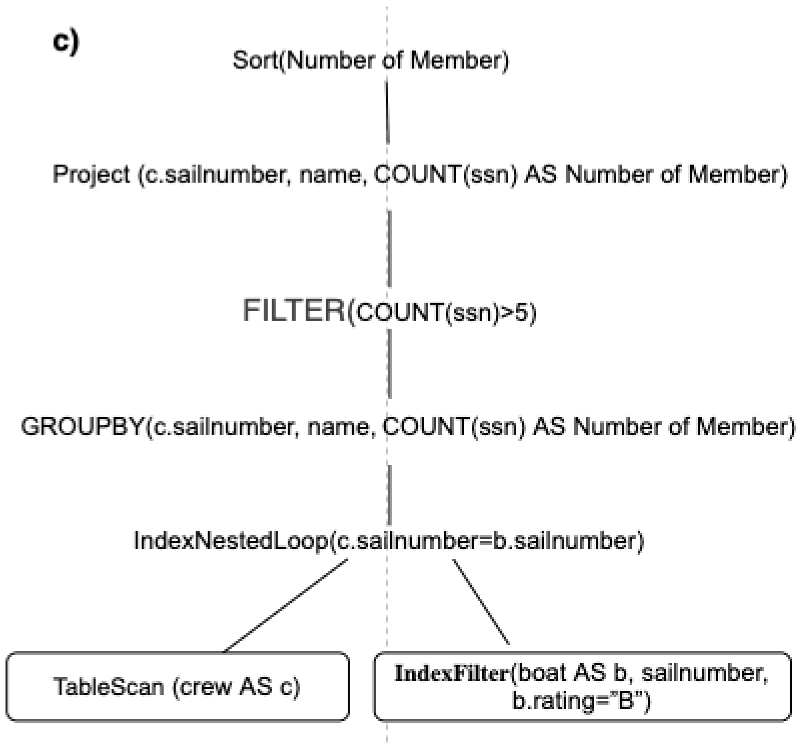
a)


PROJECT(r.raceid)
INDEXNESTEDLOOP(r.regataid=re.regataid)
TableScan(Race AS r)
IndexFilter(regata AS re, regataid, place="Pisa")

b)


SORT(FIV card number)
PROJECT(sailnumber, FIV card number)
GROUPBY({sailnumber, FIV card number}{COUNT(ssn)})
GROUPBY(sailnumber)
IndexFilter(CREW, sailnumber, medical certificate='A')

c)


Sort(Number of Member)
Project (c.sailnumber, name, COUNT(ssn) AS Number of Member)
FILTER(COUNT(ssn)>5)
GROUPBY(c.sailnumber, name, COUNT(ssn) AS Number of Member)
IndexNestedLoop(c.sailnumber=b.sailnumber)
TableScan (crew AS c)
IndexFilter(boat AS b, sailnumber, b.rating="B")