Introduction to Databases

Exam of 08/02/2022

With Solutions

Diego Calvanese

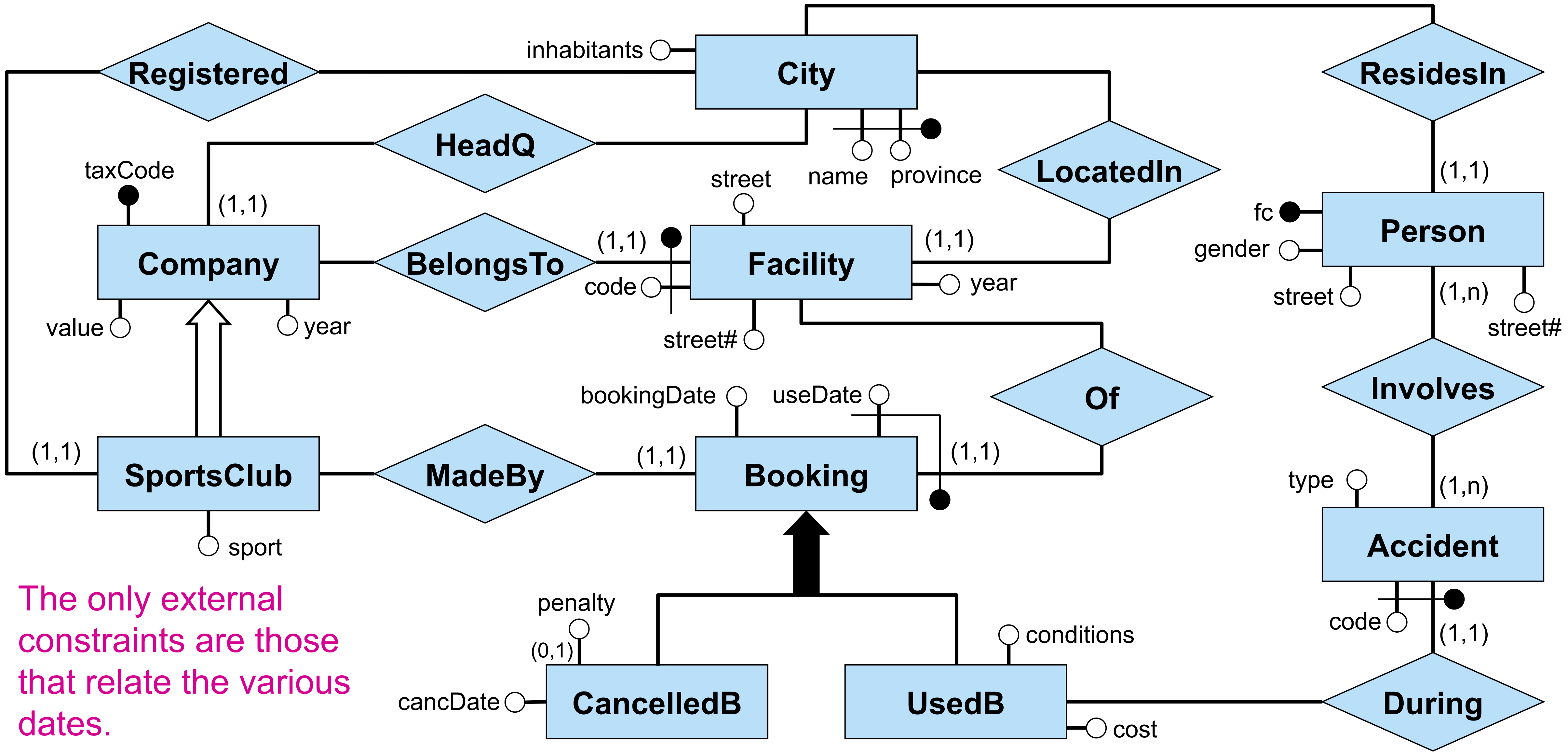
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

http://www.inf.unibz.it/~calvanese/teaching/exams/idb/

Problem 1

Design the Entity-Relationship schema of an application relating to the management of sports facilities (football fields, running tracks, golf fields, etc.). Each **facility** belongs to a company, is identified by a unique code within the company to which it belongs, and we are interested in the year of inauguration, and the street, street number, and city where it is located. For each **company**, we are interested in the tax code (identifier), the city in which its headquarters are located, the year it was founded, and the value. For **sport clubs**, which are special types of companies, we are also interested in the main sport for which the club is registered, and the city in which it is registered (which might coincide or not with the city where the headquarters are located). For each **city**, we are interested in the province, the name (unique within the province), and the number of inhabitants. Facilities are booked by sports clubs: for each **booking**, we are interested in the sports club that made it, the facility that was booked, the date on which the booking was made, and the date on which the facility will be used for that booking (two bookings for the same facility for the same date of use are not accepted). Each booking might be either used or cancelled. For each **used booking**, we are interested in the cost of the use and the conditions in which the facility was left (coded with a number), while for each **cancelled booking**, we are interested in the date of cancellation and the penalty paid for the cancellation (if known). Accidents might occur during the use of a facility, and for each such **accident**, we are interested in the code (unique within a used booking), the type of accident that occurred, and the persons involved (at least one). For each **person**, we are interested in the fiscal code, the gender and the address of residence, complete with the street name, street number, and city. (Notice that the only persons we are interested in are those involved in accidents.)

Problem 1: Conceptual schema



The only external constraints are those that relate the various dates.

Problem 2

Carry out the logical design of the database, producing the complete relational schema with constraints, taking into account the following indications:

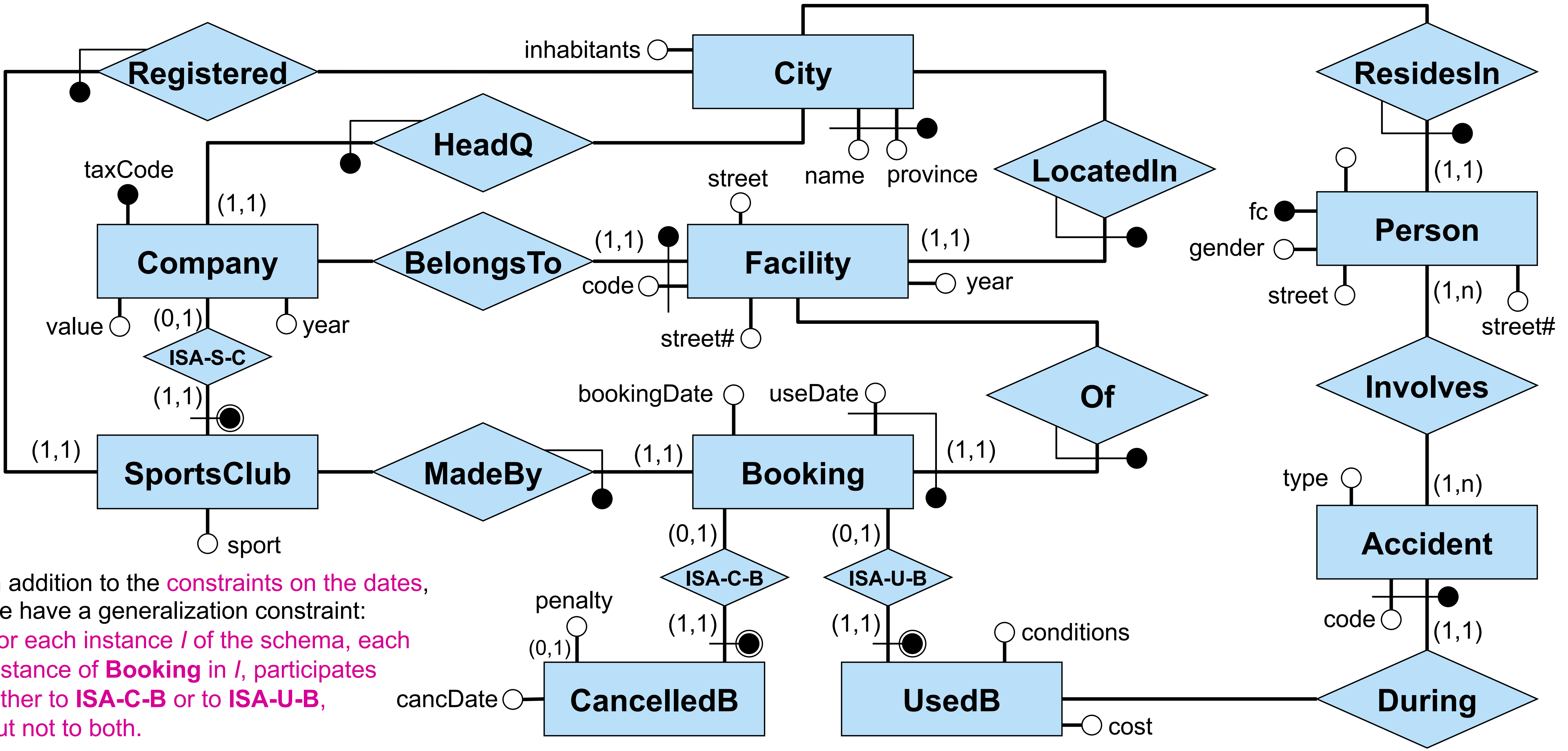
1. Avoid null values in the database.
2. When accessing a facility, we are always interested in knowing the city where it is located.
3. When accessing an accident, we are always interested in knowing the (used) booking during which the accident occurred.

As steps in your design you should produce:

- the restructured ER schema (possibly with external constraints),
- the direct translation into the relational model (possibly with external constraints), and
- the restructured relational schema (again with constraints).

Motivate explicitly how the above indications affect your design.

Problem 2: Restructured conceptual schema



In addition to the constraints on the dates, we have a generalization constraint:
For each instance *I* of the schema, each instance of **Booking** in *I*, participates either to **ISA-C-B** or to **ISA-U-B**, but not to both.

Problem 2: Direct translation (1/2)

Company(taxCode, value, year)
  foreign key: Company[taxCode] $\subseteq$ HeadQ[company]

SportsClub(taxCode, sport)
  foreign key: SportsClub[taxCode] $\subseteq$ Company[taxCode]
  foreign key: SportsClub[taxCode] $\subseteq$ Registered[sportsClub]

City(name, province, inhabitants)

HeadQ(company, city, province)
  foreign key: HeadQ[company] $\subseteq$ Company[taxCode]
  foreign key: HeadQ[city,province] $\subseteq$ City[name,province]

Registered(sportsClub, city, province)
  foreign key: Registered[sportsClub] $\subseteq$ SportsClub[taxCode]
  foreign key: Registered[city,province] $\subseteq$ City[name,province]

Facility(code, company, street, street#, year)
  foreign key: Facility[company] $\subseteq$ Company[taxCode]
  foreign key: Facility[code,company] $\subseteq$ LocatedIn[facility,company]

LocatedIn(city, province, facility, company)
  foreign key: LocatedIn[city,province] $\subseteq$ City[name,province]
  foreign key: LocatedIn[facility,company] $\subseteq$ Facility[code,company]

Person(fc, gender, street, street#)
  foreign key: Person[fc] $\subseteq$ ResidesIn[person]
  inclusion: Person[fc] $\subseteq$ Involves[person]

Problem 2: Direct translation (2/2)

Booking(useDate, facility, company, bookingDate)
  foreign key: Booking[facility,company] ⊆ Facility[code,company]
  foreign key: Booking[useDate,facility,company] ⊆ MadeBy[useDate,facility,company]
  tuple constraint: bookingDate ≤ useDate

CancelledB(useDate, facility, company, cancDate, penalty*)
  foreign key: CancelledB[useDate,facility,company] ⊆ Booking[useDate,facility,company]
  tuple constraint: cancDate ≤ useDate

UsedB(useDate, facility, company, conditions)
  foreign key: UsedB[useDate,facility,company] ⊆ Booking[useDate,facility,company]

MadeBy(useDate, facility, company, sportsClub)
  foreign key: MadeBy[useDate,facility,company] ⊆ Booking[useDate,facility,company]
  foreign key: MadeBy[sportsClub] ⊆ SportsClub[taxCode]

Accident(code, useDate, facility, company, type)
  foreign key: Accident[useate,facility,company] ⊆ UsedB[useDate,facility,company]
  inclusion: Accident[code,useDate,facility,company] ⊆ Involves[accident,useDate,facility,company]

Involves(accident, useDate, facility, company, person)
  foreign key: Involves[accident,useDate,facility,company] ⊆ Accident[code,useDate,facility,company]
  foreign key: Involves[person] ⊆ Person[fc]

ResidesIn(person, city, province)
  foreign key: ResidesIn[person] ⊆ Person[fc]
  foreign key: ResidesIn[city,province] ⊆ City[name,province]

Problem 2: Direct translation – External constraints

Generalization constraints:

$\text{Booking[useDate,facility,company]} \subseteq$
$\quad \text{CancelledB[useDate,facility,company]} \cup \text{UsedB[useDate,facility,company]}$

$\text{CancelledB[useDate,facility,company]} \cap \text{UsedB[useDate,facility,company]} = \emptyset$

Problem 2: Restructuring of the relational schema

1. Avoid null values in the database.
2. When accessing a facility, we are always interested in knowing the city where it is located.
3. When accessing an accident, we are always interested in knowing the (used) booking during which the accident occurred.

We take into account the above indications in the following way:

- We take into account indication 1 by applying a mixed decomposition to **CancelledB**:
  1. We first vertically decompose **CancelledB** into **CancelledB**, which now contains all attributes except for **penalty** (which is nullable), and **CancelledBPenalty**, which contains **penalty**.
  2. We eliminate the part of **CancelledBPenalty** with NULL values for **penalty**.
- We take into account indication 2 by merging the relation **LocatedIn** in the relation **Facility**.
- Indication 3 has already been taken into account by the direct translation, since the relationship **During** has been merged into the relation for the entity **Accident**.

Problem 2: Restructured relational schema

We specify here only the relations with their constraints that have been changed with respect to the schema obtained through the direct translation.

Facility(<u>code</u>, <u>company</u>, street, street#, year, city, province)
  foreign key: Facility[company] $\subseteq$ Company[taxCode]
  foreign key: Facility[city,province] $\subseteq$ City[name,province]

CancelledB(<u>useDate</u>, <u>facility</u>, <u>company</u>, cancDate)
  foreign key: CancelledB[useDate,facility,company] $\subseteq$ Booking[useDate,facility,company]
  tuple constraint: cancDate ≤ useDate

CancelledBPenalty(<u>useDate</u>, <u>facility</u>, <u>company</u>, penalty)
  foreign key: CancelledBPenalty[useDate,facility,company] $\subseteq$
    CancelledB[useDate,facility,company]

There are no constraints that need to be changed or added due to the transformations, so we just keep the generalization constraints that we had.

Problem 3

Consider a database that includes the relations `Vehicle` and `Trip`. The relation `Vehicle(code,owner)` stores for each vehicle, the identification code and the owner, while the relation `Trip(code,city,day,month,year)` stores the trips made by the vehicles in the various cities and on the various dates.

Express the following queries in SQL:

1. For each owner, return the code of the vehicles of that owner that made a trip in Bolzano in 2021.
2. For each owner, count the number of trips made by that owner's vehicles in December.
3. Return how many vehicles are owned by each person who owns at least one vehicle that since 2020 has made only trips in Merano.

Problem 3: Solution (1/2)

```
Vehicle(code,owner)
Trip(code,city,day,month,year)
```

1. For each owner, return the code of the vehicles of that owner that made a trip in Bolzano in 2021.

```
SELECT DISTINCT V.owner, V.code AS vehicle
FROM Vehicle V JOIN Trip T ON V.code = T.code
WHERE T.city = 'Bolzano' AND T.year = 2021
```

2. For each owner, count the number of trips made by that owner's vehicles in December.

```
SELECT V.owner, COUNT(*) AS numTrips
FROM Vehicle V JOIN Trip T ON V.code = T.code
WHERE month = 'December'
GROUP BY V.owner
  UNION
SELECT V.owner, 0 AS numTrips
FROM Vehicle V
WHERE V.owner NOT IN (SELECT V2.owner
                      FROM Vehicle V2 JOIN Trip T ON V2.code = T.code
                      WHERE month = 'December')
```

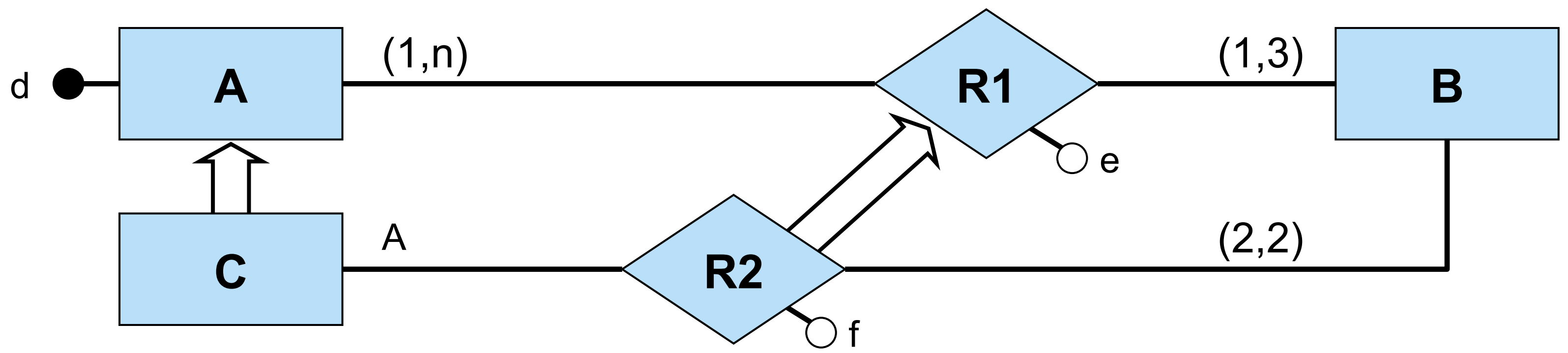
Problem 3: Solution (2/2)

```
Vehicle(code,owner)
Trip(code,city,day,month,year)
```

3. Return how many vehicles are owned by each person who owns at least one vehicle that since 2020 has made only trips in Merano.

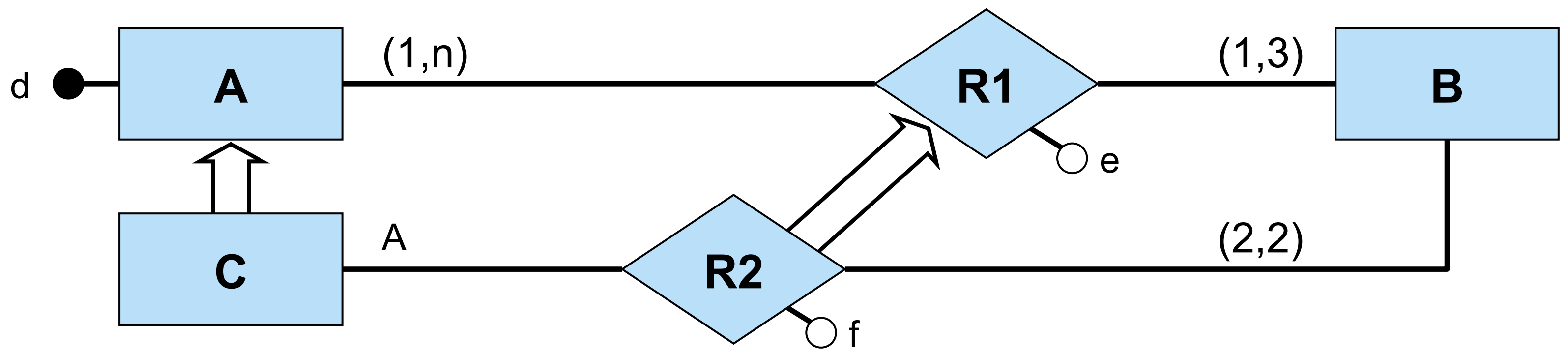
```
WITH ViewOwnerOfVehicleMeranoOnly AS
       (SELECT owner FROM Vehicle
        WHERE code NOT IN (SELECT code FROM Trip
                           WHERE year >= 2020 AND city <> ‘Merano’)
SELECT owner, COUNT(code) AS numVehicles
FROM Vehicle
WHERE owner IN (SELECT owner FROM ViewOwnerOfVehicleMeranoOnly)
GROUP BY owner
```

Problem 4



Consider the conceptual schema ***S*** shown above and say if there is an instance ***I*** of ***S*** in which the entity ***B*** has at least one instance.
If the answer is negative, explain *in detail* why such an instance ***I*** does not exist.
If the answer is positive, describe *precisely* such an instance ***I*** of ***S*** (considering *all* the elements that appear in ***S***).

Problem 4: Solution



We show an instance ***I*** of ***S*** in which the entity ***B*** has exactly one instance:

$instances(\boldsymbol{I},\boldsymbol{A}) = instances(\boldsymbol{I},\boldsymbol{C}) = \{a_1,a_2\}$
$instances(\boldsymbol{I},\boldsymbol{B}) = \{b\}$
$instances(\boldsymbol{I},\boldsymbol{R1}) = instances(\boldsymbol{I},\boldsymbol{R2}) = \{(\boldsymbol{A}{:}a_1,\boldsymbol{B}{:}b), (\boldsymbol{A}{:}a_2,\boldsymbol{B}{:}b)\}$
$instances(\boldsymbol{I},\boldsymbol{d}) = \{(a_1,\text{'}d1\text{'}), (a_2,\text{'}d2\text{'})\}$
$instances(\boldsymbol{I},\boldsymbol{e}) = \{((\boldsymbol{A}{:}a_1,\boldsymbol{B}{:}b),\text{'}e1\text{'}), ((\boldsymbol{A}{:}a_2,\boldsymbol{B}{:}b),\text{'}e2\text{'})\}$
$instances(\boldsymbol{I},\boldsymbol{f}) = \{((\boldsymbol{A}{:}a_1,\boldsymbol{B}{:}b),\text{'}f1\text{'}), ((\boldsymbol{A}{:}a_2,\boldsymbol{B}{:}b),\text{'}f2\text{'})\}$

It is easy to see that ***I*** satisfies all constraints imposed by ***S***, hence it is ideed an instance of ***S***.