Introduction to Databases
Exam of 26/09/2019
Solutions

Diego Calvanese

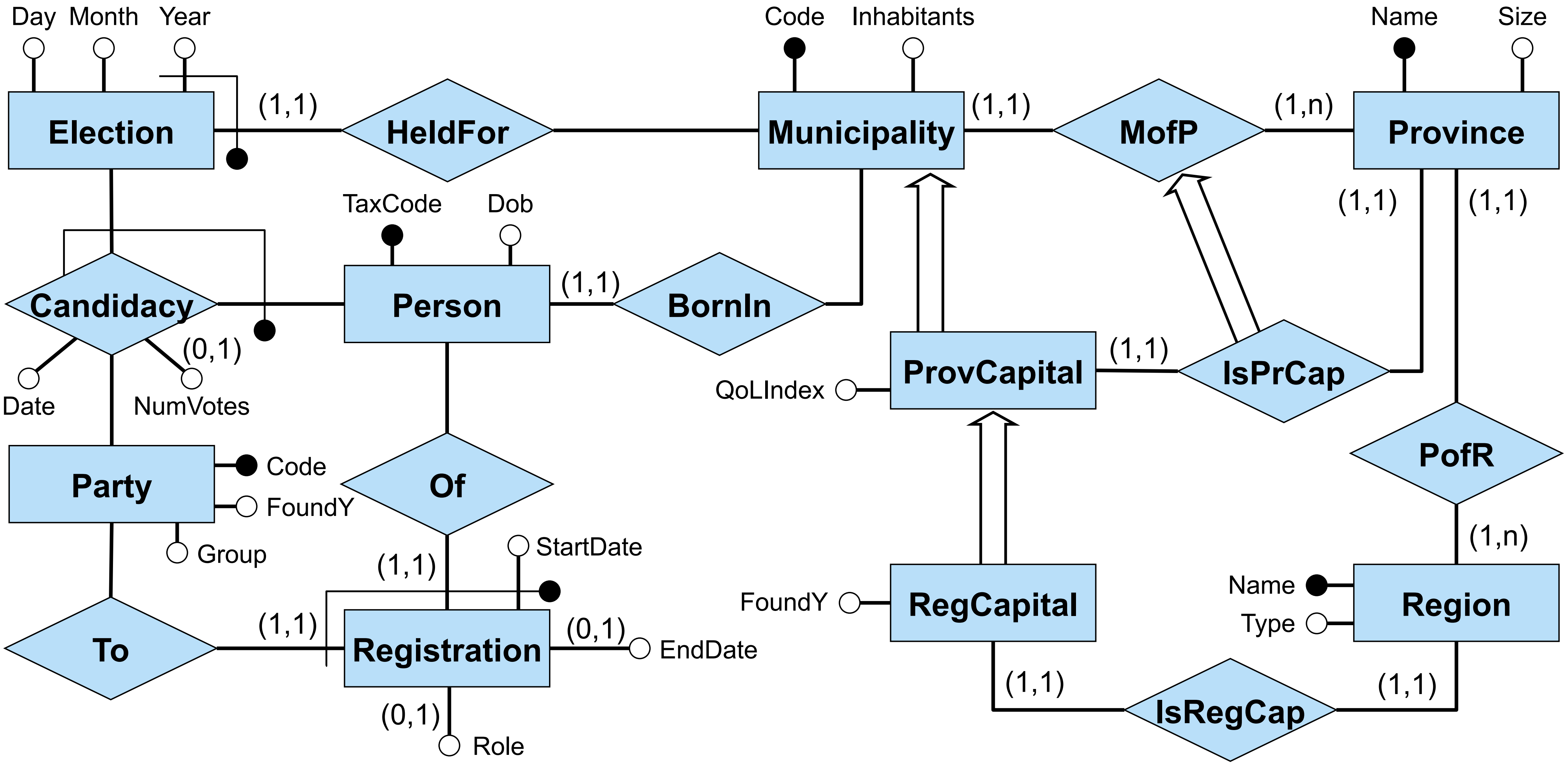
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity Relationship schema of an information system that concerns the elections of municipality councils. Each **election** is held for a municipality on a certain date, with the condition that at most one election per year may be held in the same municipality. For each election, we are interested in the persons who are candidates, with the date where they have presented their candidacy and the political party of reference for the candidacy itself. Note that nobody can present more than one candidacy for the same election. In addition, if a candidate for an election is elected, then we are interested in the number of votes with which the candidate was elected. For each **person**, we are interested in the tax code (identifier), the date of birth, the municipality of birth, the political party (if any) to which the person is currently registered, with the start date of the registration, and the role (if any) played during the registration period (e.g., provincial secretary, regional secretary, etc.). We are also interested in the political parties to which the person has been registered in the past, with the start date and end date of the **registration**, and again the role (if any) played during the registration period. For each **municipality**, we are interested in the code (identifier), the number of inhabitants, and the province to which it belongs. For each **province** we are interested in the name (identifier), the size, the region to which it belongs, and the municipality that, among those that belong to the province itself, is the capital. For each **region** we are interested in the name (identifier), the type (normal, special statute, etc..), and the municipality that is the capital of that region. Note that the municipality that is the capital of a region is also the capital of a province belonging to that region. For the municipalities that are provincial capitals we are interested also in the quality of life index, and for the provincial capitals that are also regional capitals we are interested also in the year of foundation. Finally, for each **political party** we are interested in the code (identifier), the year of foundation, and the political group (left, center, etc..) to which it belongs.

Problem 1: Conceptual schema – Diagram



Problem 1: Conceptual schema – External constraints

External Constraint: For each instance ***I*** of the conceptual schema the following holds:

1. Registration periods of a person to a party do not overlap:
   There are no two instances r_1 and r_2 in ***instances***(***I***,**Registration**) such that for some instance *p* in ***instances***(***I***,**Person**), both (**Registration**:r_1, **Person**:*p*) and (**Registration**:r_2, **Person**:*p*) in ***instances***(***I***,**Of**), and for (r_1,s_1), (r_2,s_2) in ***instances***(***I***,**StartDate**) and (r_1,e_1) in ***instances***(***I***,**EndDate**), we have that $s_1 < s_2 < e_1$.
2. There is at most one current registration of a person to a party:
   For each instance *p* in ***instances***(***I***,**Person**), there is at most one instance *r* in ***instances***(***I***,**Registration**) such that *(1)* (**Registration**:*r*, **Person**:*p*) in ***instances***(***I***,**Of**) and *(2)* there is no *d* such that (*r*,*d*) in ***instances***(***I***,**EndDate**).
3. For each election, there is at most one person that is elected, i.e., for which the number of votes is defined.
   For each instance *e* in ***instances***(***I***,**Election**), there is at most one instance *c* = (**Election**:*e*, **Person**:*p* , **Party**:*q*) in ***instances***(***I***,**Candidacy**) such that (*c*,*n*) is in ***instances***(***I***,**NumVotes**), for some integer *n*.
4. If a provincial capital is also a regional capital, then the province of which it is the capital belongs to the region of which it is the capital:
   For each instance (**RegCapital**:*c*, **Region**:*r*) in ***instances***(***I***,**IsRegCap**), consider the unique *p* in ***instances***(***I***,**Province**) such that (**ProvCapital**:*c*, **Province**:*p*) is in ***instances***(***I***,**IsPrCap**). Then (**Province**:*p*, **Region**:*r*) in ***instances***(***I***,**PofR**).

Problem 2

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

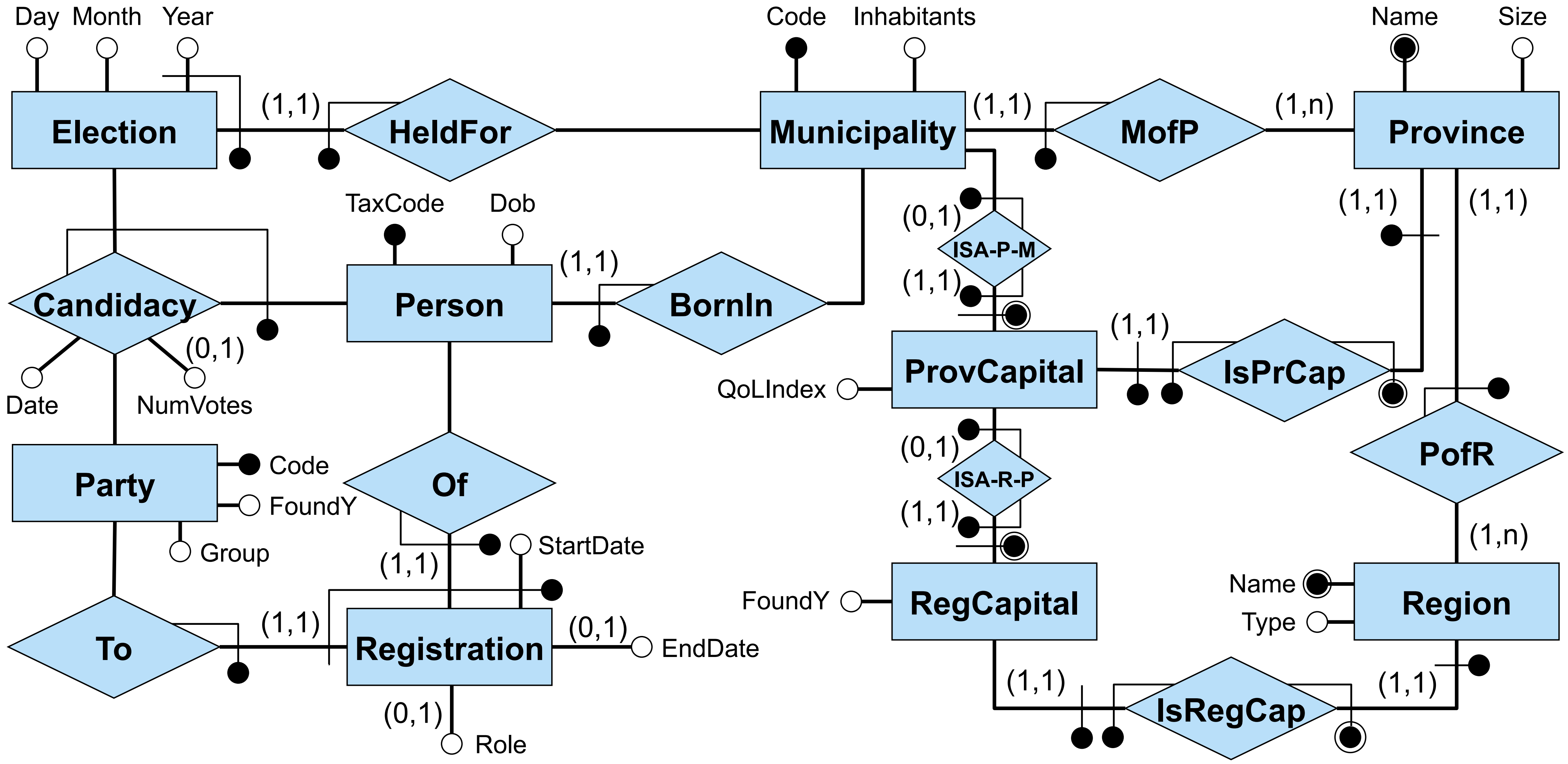
1. When accessing the data of a candidacy, we always want to know the candidate, the political party of reference for the candidacy, the election in which the candidacy was presented and possibly the number of votes with which the person was elected;
2. When accessing a municipality, we always want to know the province to which it belongs.

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



Problem 2: Restructured conceptual schema – External constraints

External Constraint: For each instance ***I*** of the conceptual schema:

1. Is not affected by the restructuring.

2. Is not affected by the restructuring.

3. Is not affected by the restructuring.

4. For each instance (**RegCapital**:*rc*, **Region**:*r*) in ***instances***(***I***,**IsRegCap**), consider the unique *pc* in ***instances***(***I***,**ProvCapital**) such that (**RecCapital**:*rc*, **ProvCapital**:*pc*) in ***instances***(***I***,**ISA-R-P**), and the unique *p* in ***instances***(***I***,**Province**) such that (**ProvCapital**:*pc*, **Province**:*p*) in ***instances***(***I***,**IsPrCap**). Then (**Province**:*p*, **Region**:*r*) in ***instances***(***I***,**PofR**).

5. Constraint introduced by the elimination of ISA between relations:
   For each instance (**ProvCapital**:*pc*, **Province**:*p*) in ***instances***(***I***,**IsPrCap**), consider the unique *m* in ***instances***(***I***,**Municipality**) such that (**ProvCapital**:*pc*, **Municipality**:*m*) in ***instances***(***I***,**ISA-P-M**). Then (**Municipality**:*m*, **Province**:*p*) in ***instances***(***I***,**MofP**).

Problem 2: Direct translation (1/3)

Election(Day, Month, <u>Year</u>, <u>Municipality</u>)
  foreign key: Election[Municipality] $\subseteq$ Municipality[Code]

Person(<u>TaxCode</u>, Dob)
  foreign key: Person[TaxCode] $\subseteq$ BornIn[Person]

Party(<u>Code</u>, FoundY, Group)

Candidacy(<u>Year</u>, <u>Municipality</u>, <u>Person</u>, Party, Date, NumVotes*)
  foreign key: Candidacy[Year,Municipality] $\subseteq$ Election[Year, Municipality]
  foreign key: Candidacy[Person] $\subseteq$ Person[TaxCode]
  foreign key: Candidacy[Party] $\subseteq$ Party[Code]

Registration(<u>Person</u>, <u>Party</u>, <u>StartDate</u>, EndDate*, Role*)
  foreign key: Registration[Person] $\subseteq$ Person[TaxCode]
  foreign key: Registration[Party] $\subseteq$ Party[Code]

Problem 2: Direct translation (2/3)

Municipality(<u>Code</u>, Inhabitants)
  foreign key: Municipality[Code] ⊆ MofP[Municipality]

Province(<u>Name</u>, Size)
  inclusion: Province[Name] ⊆ MofP[Province]
  foreign key: Province[Name] ⊆ IsPrCap[Province]
  foreign key: Province[Name] ⊆ PofR[Province]

MofP(<u>Municipality</u>, Province)
  foreign key: MofP[Municipality] ⊆ Municipality[Code]
  foreign key: MofP[Province] ⊆ Province[Name]

ProvCapital(<u>Code</u>, QoLIndex)
  foreign key: ProvCapital[Code] ⊆ Municipality[Code]
  foreign key: ProvCapital[Code] ⊆ IsPrCap[ProvCapital]

IsPrCap(ProvCapital, <u>Province</u>)
  key: ProvCapital
  foreign key: IsPrCap[ProvCapital] ⊆ ProvCapital[Code]
  foreign key: IsPrCap[Province] ⊆ Province[Name]
  foreign key: IsPrCap[ProvCapital,Province] ⊆ MofP[Municipality,Province]

Problem 2: Direct translation (3/3)

RegCapital(<u>Code</u>, FoundY)
  foreign key: RegCapital[Code] $\subseteq$ ProvCapital[Code]
  foreign key: RegCapital[Code] $\subseteq$ IsRegCap[RegCapital]

Region(<u>Name</u>, Type)
  foreign key: Region[Name] $\subseteq$ IsRegCap[Region]
  inclusion: Region[Name] $\subseteq$ PofR[Region]

IsRegCap(RegCapital, <u>Region</u>)
  key: RegCapital
  foreign key: IsRegCapital[RegCapital] $\subseteq$ RegCapital[Code]
  foreign key: IsRegCapital[Region] $\subseteq$ Region[Name]

PofR(<u>Province</u>, Region)
  foreign key: PofR[Province] $\subseteq$ Province[Name]
  foreign key: PofR[Region] $\subseteq$ Region[Name]

BornIn(<u>Person</u>, Municipality)
  foreign key: BornIn[Person] $\subseteq$ Person[TaxCode]
  foreign key: BornIn[Municipality] $\subseteq$ Municipality[Code]

Problem 2: Direct translation – Constraints

Constraints:

1. There are no two tuples (*person*, *party1*, *sd1*, *ed1, role1*) and (*person*, *party2*, *sd2*, *ed2, role2*) in **Registration** such that *sd1* < *sd2* < *ed1*.
2. For each value *person*, there is at most one tuple (*person*, *party*, *sd*, *ed, role*) in **Registration** such that *ed* is NULL.
3. For each pair of values *y*, *mun*, there is at most one tuple (*y*, *mun*, *person*, *party*, *d*, *numvotes*) in **Candidacy** such that *numvotes* is not NULL.
4. For each tuple (*regcap*, *region*) in **IsRegCap**, consider the unique value *province* such that there is a tuple (*regcap*, *province*) in **IsPrCap**. Then the tuple (*province*, *region*) is in **PofR**.
5. This has been translated into the foreign key of **IsPrCap**:

   IsPrCap[ProvCapital,Province] $\subseteq$ MofP[Municipality,Province]

Problem 2: Restructuring of the relational schema

1. When accessing the data of a candidacy, we always want to know the candidate, the political party of reference for the candidacy, the election in which the candidacy was presented and possibly the number of votes with which the person was elected;
2. When accessing a municipality, we always want to know the province to which it belongs.

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

- Indication 1 is already taken into account by the fact that the **Candidacy** relation contains already all attributes that identify the candidate (i.e., Person), the political party (i.e., Party), the election (i.e., Year, Municipality), and the number of votes (i.e., NumVotes).
- We take into account indication 2 by merging MofP with Municipality, IsPrCap with ProvCapital, and IsRegCap with RegCapital.

Problem 2: Restructured relational schema (1/3)

Election(Day, Month, Year, Municipality)
  foreign key: Election[Municipality] $\subseteq$ Municipality[Code]

Person(TaxCode, Dob)
  foreign key: Person[TaxCode] $\subseteq$ BornIn[Person]

Party(Code, FoundY, Group)

Candidacy(Year, Municipality, Person, Party, Date, NumVotes*)
  foreign key: Candidacy[Year,Municipality] $\subseteq$ Election[Year, Municipality]
  foreign key: Candidacy[Person] $\subseteq$ Person[TaxCode]
  foreign key: Candidacy[Party] $\subseteq$ Party[Code]

Registration(Person, Party, StartDate, EndDate*, Role*)
  foreign key: Registration[Person] $\subseteq$ Person[TaxCode]
  foreign key: Registration[Party] $\subseteq$ Party[Code]

Problem 2: Restructured relational schema (2/3)

Municipality(<u>Code</u>, Inhabitants, Province)

foreign key: Municipality[Code] ⊆ MofP[Municipality]
foreign key: Municipality[Province] ⊆ Province[Name]

Province(<u>Name</u>, Size)

inclusion: Province[Name] ⊆ Municipality[Province]
foreign key: Province[Name] ⊆ ProvCapital[Province]
foreign key: Province[Name] ⊆ PofR[Province]

ProvCapital(<u>Code</u>, QoLIndex, Province)

key: Province
foreign key: ProvCapital[Code] ⊆ Municipality[Code]
foreign key: ProvCapital[Province] ⊆ Province[Name]
foreign key: ProvCapital[Code,Province] ⊆ Municipality[Code,Province]

Problem 2: Restructured relational schema (3/3)

RegCapital(<u>Code</u>, FoundY, Region)
  key: Region
  foreign key: RegCapital[Code] $\subseteq$ ProvCapital[Code]
  foreign key: RegCapital[Region] $\subseteq$ Region[Name]

Region(<u>Name</u>, Type)
  foreign key: Region[Name] $\subseteq$ RegCapital[Region]
  inclusion: Region[Name] $\subseteq$ PofR[Region]

PofR(<u>Province</u>, Region)
  foreign key: PofR[Province] $\subseteq$ Province[Name]
  foreign key: PofR[Region] $\subseteq$ Region[Name]

BornIn(<u>Person</u>, Municipality)
  foreign key: BornIn[Person] $\subseteq$ Person[TaxCode]
  foreign key: BornIn[Municipality] $\subseteq$ Municipality[Code]

Problem 2: Restructured relational schema – Constraints

Constraints:

1. Is not affected by the restructuring.
2. Is not affected by the restructuring.
3. Is not affected by the restructuring.
4. For each tuple (*regcap*, *y*, *region*) in **RegCapital**, consider the unique value *province* such that there is a tuple (*regcap*, *qoli*, *province*) in **ProvCapital**, for some value *qoli*. Then the tuple (*province*, *region*) is in **PofR**.
5. This becomes now a foreign key of **ProvCapital**:

$$\text{ProvCapital[Code,Province]} \subseteq \text{Municipality[Code,Province]}$$

Problem 3

The relation `City(name,population,level)` stores the data about population and pollution level of cities, the relation `Sensor(code,day,numVehicles)` stores, for each sensor represented by the code, and for each day (of the current year), the number of vehicle passages measured by the sensor, while the relation `Location(sensorCode,cityName)` stores, for each sensor, the city in which it is located.

Express in SQL the following queries over the above relations:

1. Calculate the name, population, and pollution level of the cities in which at least one sensor is located that has detected more than 100 vehicle passages in at least one day.
2. For each city with a pollution level greater than 5, calculate how many vehicle passages in total were detected throughout the year by all sensors located in that city.
3. For each day of the current year and for each city with at least 10,000 inhabitants, calculate the average number of vehicle passages measured in that day by the sensors located in that city.

Problem 3: Solution (1/3)

```
City(name,population,level)
Sensor(code,day,numVehicles)
Location(sensorCode,cityName)
```

1. Calculate the name, population, and pollution level of the cities in which at least one sensor is located that has detected more than 100 vehicle passages in at least one day.

```
SELECT DISTINCT C.name, C.population, C.level
FROM (City C JOIN Location L ON C.name = L.cityName) JOIN
     Sensor S ON S.code = L.sensorCode
WHERE S.numVehicles > 100
```

Problem 3: Solution (2/3)

```
City(name,population,level)
Sensor(code,day,numVehicles)
Location(sensorCode,cityName)
```

2. For each city with a pollution level greater than 5, calculate how many vehicle passages in total were detected throughout the year by all sensors located in that city.

```
SELECT C.name, SUM(S.numVehicles)
FROM (City C JOIN Location L ON C.name = L.cityName) JOIN
     Sensor S ON S.code = L.sensorCode
WHERE C.level > 5
GROUP BY C.name
```

Problem 3: Solution (3/3)

```
City(name,population,level)
Sensor(code,day,numVehicles)
Location(sensorCode,cityName)
```

3. For each day of the current year and for each city with at least 10,000 inhabitants, calculate the average number of vehicle passages measured in that day by the sensors located in that city.

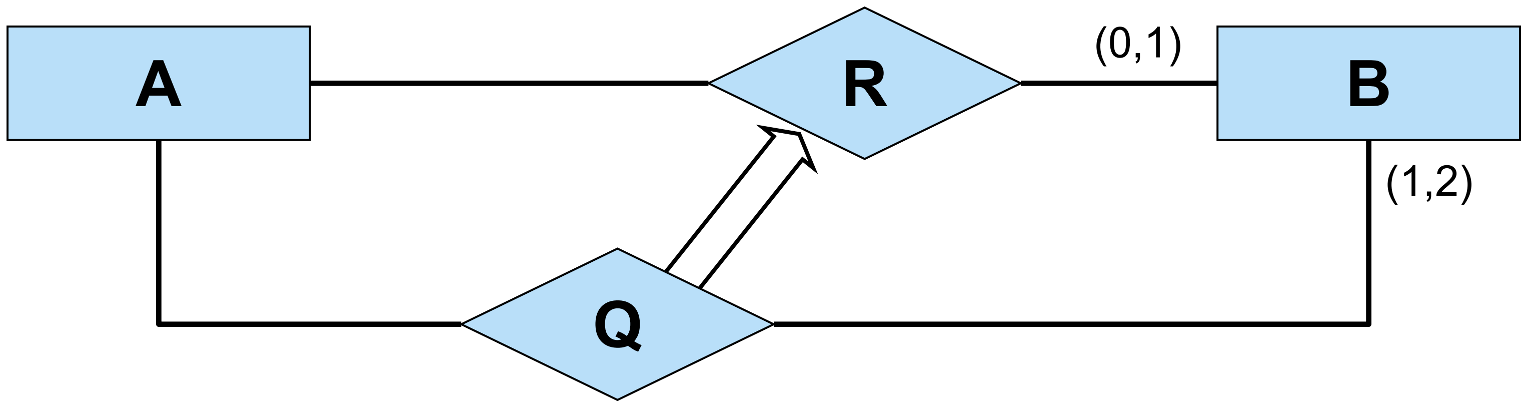
```
SELECT C.name, S.day, AVG(S.numVehicles)
FROM (City C JOIN Location L ON C.name = L.cityName) JOIN
     Sensor S ON S.code = L.sensorCode
WHERE C.population >= 10000
GROUP BY C.name, S.day
```

Problem 4

Consider the conceptual schema *S* shown below. If we change the schema *S* as follows, do we get a schema equivalent to *S* (i.e., that admits the same instances)?

1. Setting the minimum cardinality of *Q* in the role *B* to 0.
2. Setting the maximum cardinality of *Q* in the role *B* to 1.

Motivate you answer in both cases.



Problem 4: Solution (1/2)

1. By setting the minimum cardinality of *Q* in the role *B* to 0, we obtain a schema S_1 that is **not** equivalent to *S*.

For example, consider the following instance *I*:

instances(*I*,*A*) = { },

instances(*I*,*B*) = { b },

instances(*I*,*R*) = { },

instances(*I*,*Q*) = { }

I is legal for schema S_1 but not for schema *S*, since instance *b* violates the minimum cardinality 1 of *Q* in the role *B*.

Problem 4: Solution (2/2)

2. By setting the maximum cardinality of *Q* in the role *B* to 1, we obtain a schema S_2 that is equivalent to *S*.

This holds because the maximum cardinality of a relation is inherited by its subrelations.

In the example, there is no instance *I* of *S* such that there is some *b* in *instances*(*I*,*B*) and a_1 and a_2 both in *instances*(*I*,*A*) such that both (*A*:a_1, *B*:*b*) and (*A*:a_2, *B*:*b*) are in *instances*(*I*,*Q*). Indeed, if we had such an instance *I*, then both (*A*:a_1, *B*:*b*) and (*A*:a_2, *B*:*b*) would be also in *instances*(*I*,*Q*), and *b* would violate the maximum cardinality 1 of *R* in the role *B*. Hence, every instance of schema *S* is already an instance of S_2.

On the other hand, every instance of S_2 is already an instance of *S*, since S_2 is identical to S, except that it imposes a stricter cardinality on *Q* in the role *B*.

Hence, *S* and S_2 have the same instances, which means that they are equivalent.