Introduction to Databases

Exam of 25/01/2023

With Solutions

Diego Calvanese

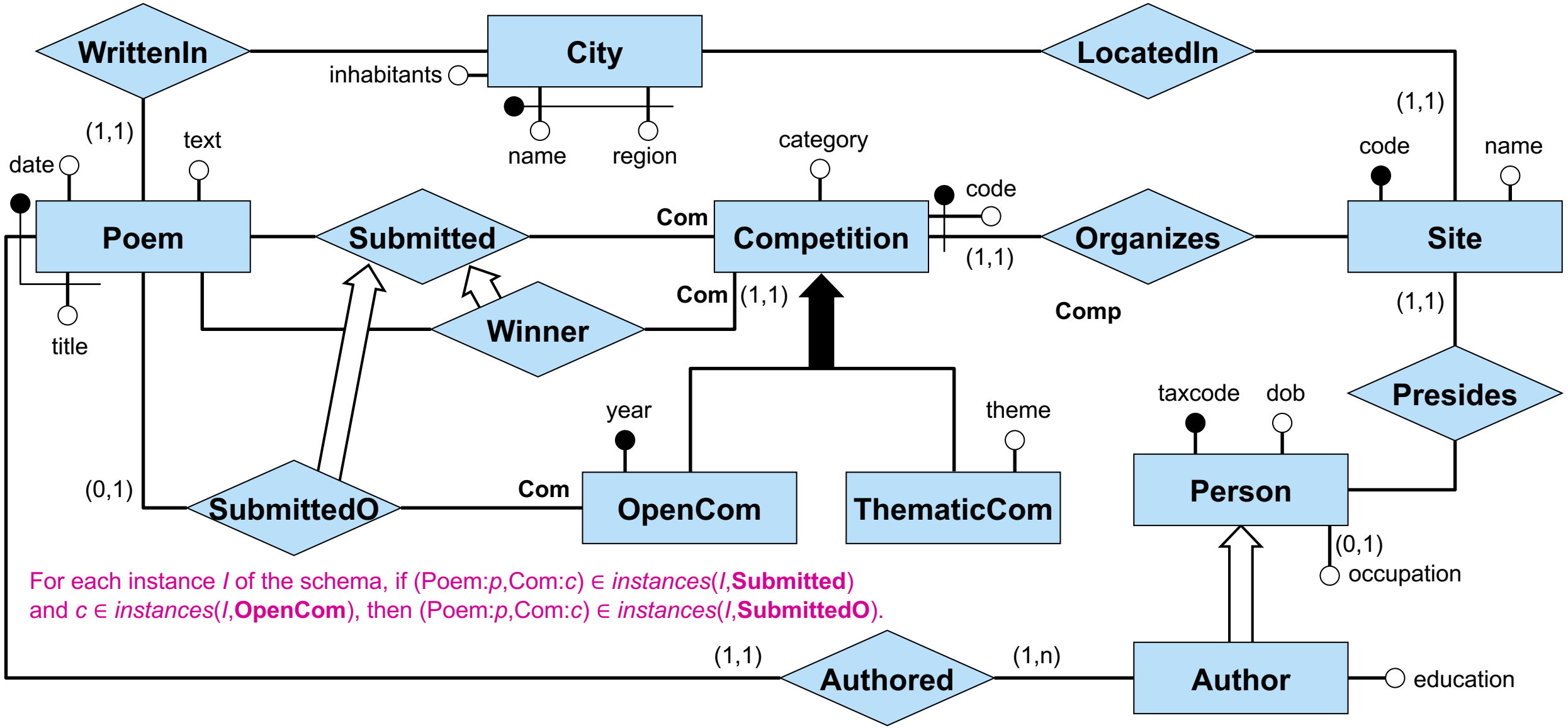
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema of an application for managing the poetry competitions that have been organized by an association. The association consists of different sites, which are responsible for organizing the competitions. Of each **competition**, we are interested in the site that organized it, the code (unique within the site that organized it), and the category. There are exactly two types of competitions, open ones and thematic ones. Of each **open competition**, we are interested in the year in which it is held (in each year a maximum of one open competition is allowed), while of each **thematic competition**, we are interested in the theme covered. Of each competition, we are interested in knowing the various poems that have been submitted and among them which is the unique winner. Of each **poem**, we are interested in the person who authored it, the title (unique to the person who authored it), the text, and the date and city where it was written. Note that a poem may be submitted to multiple competitions, but at most to one open competition. Of each **site**, we are interested in the code (identifier), the name, the person who is its president, and the city where it is located. Of each **person**, we are interested in the tax code (identifier), the date of birth, and the occupation, if any. Of each **poem author**, we are also interested in the education level. Of each **city**, we are interested in the region in which it is located, the name (unique within the region in which it is located), and the number of inhabitants.

Problem 1: Conceptual schema



Problem 2

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

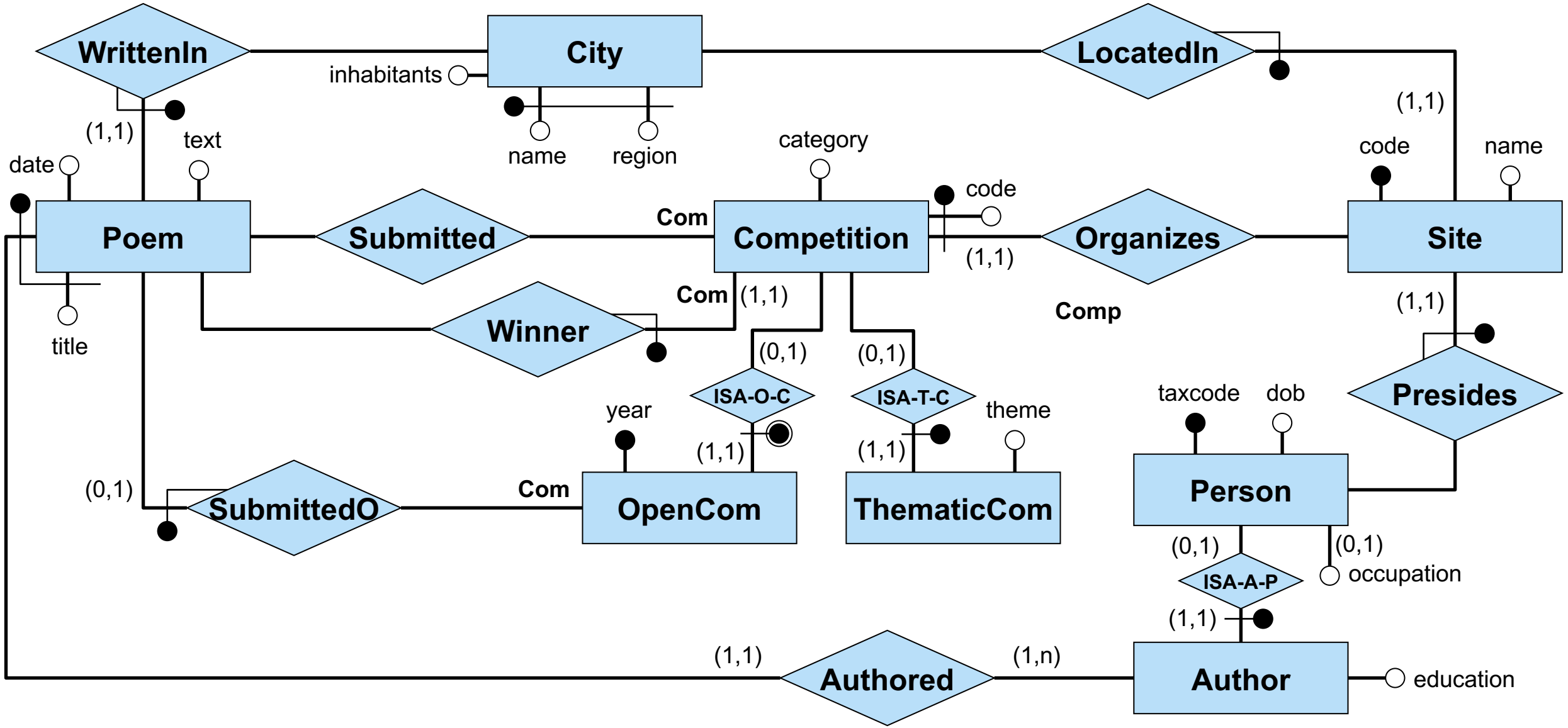
1. Null values in the database should be avoided.
2. When accessing the data on a poem, we always want to know its author.

In your design you should follow the methodology adopted in the course, and 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).

You should motivate explicitly how the above indications affect your design.

Problem 2: Restructured conceptual schema



Problem 2: Restrucured Conceptual Schema – External constraints

1. For each instance *I* of the schema, if (Poem:*p*,Com:*c*) ∈ *instances*(*I*,**Submitted**) and there is an *o* ∈ *instances*(*I*,**OpenCom**) such that (OpenCom:*o*,Competition:*c*) ∈ *instances*(*I*,**ISA-O-C**), then (Poem:*p*,Com:*o*) ∈ *instances*(*I*,**SubmittedO**).
2. Generalization constraint for the entity **Competition**:
   For each instance *I* of the schema, each instance of **Competition** in *I*, participates either to **ISA-O-C** or to **ISA-T-C**, but not to both.
3. Constraint resulting from the elimination of ISA between relationships **Winner** and **Submitted**:
   For each instance *I* of the schema, each instance of **Winner** in *I* is also an instance of **Submitted** in *I*.
4. Constraint resulting from the elimination of ISA between relationships **SubmittedO** and **Submitted**:
   For each instance *I* of the schema, if (Poem:*p*,Com:*o*) ∈ *instances*(*I*,**SubmittedO**) and (OpenCom:*o*,Competition:*c*) ∈ *instances*(*I*,**ISA-O-C**), then (Poem:*p*,Com:*c*) ∈ *instances*(*I*,**Submitted**).

Problem 2: Direct translation (1/2)

City(<u>name</u>, <u>region</u>, inhabitants)

Site(<u>code</u>, name)
- foreign key: Site[code] $\subseteq$ LocatedIn[site]
- foreign key: Site[code] $\subseteq$ Presides[site]

LocatedIn(<u>site</u>, city, region)
- foreign key: LocatedIn[site] $\subseteq$ Site[code]
- foreign key: LocatedIn[city,region] $\subseteq$ City[name,region]

Person(<u>taxcode</u>, dob, occupation*)

Author(<u>taxcode</u>, education)
- foreign key: Author[taxcode] $\subseteq$ Person[taxcode]
- inclusion: Author[taxcode] $\subseteq$ Poem[author]

Presides(<u>site</u>, person)
- foreign key: Presides[site] $\subseteq$ Site[code]
- foreign key: Presides[person] $\subseteq$ Person[taxcode]

Competition(<u>code</u>, <u>site</u>, category)
- foreign key: Competition[site] $\subseteq$ Site[code]
- foreign key: Competition[code,site] $\subseteq$ Winner[competition,site]

Problem 2: Direct translation (2/2)

OpenCom(<u>code</u>, <u>site</u>, year)
  foreign key: OpenCom[code,site] $\subseteq$ Competition[code,site]
  key: year

ThematicCom(<u>code</u>, <u>site</u>, theme)
  foreign key: ThematicCom[code,site] $\subseteq$ Competition[code,site]

Poem(<u>title</u>, <u>author</u>, date, text)
  foreign key: Poem[author] $\subseteq$ Author[taxcode]
  foreign key: Poem[title,author] $\subseteq$ WrittenIn[poem,author]

WrittenIn(<u>poem</u>, <u>author</u>, city, region)
  foreign key: WrittenIn[poem,author] $\subseteq$ Poem[title,author]
  foreign key: WrittenIn[city,region] $\subseteq$ City[name,region]

Submitted(<u>poem</u>, <u>author</u>, <u>competition</u>, <u>site</u>)
  foreign key: Submitted[poem,author] $\subseteq$ Poem[title,author]
  foreign key: Submitted[competition,site] $\subseteq$ Competition[code,site]

Winner(poem, author, <u>competition</u>, <u>site</u>)
  foreign key: Winner[poem,author,competition,site] $\subseteq$ Submitted[poem,author,competition,site]

SubmittedO(<u>poem</u>, <u>author</u>, competition, site)
  foreign key: SubmittedO[poem,author,competition,site] $\subseteq$ Submitted[poem,author,competition,site]
  foreign key: SubmittedO[competition,site] $\subseteq$ OpenCom[code,site]

Problem 2: Direct translation – External constraints

1. Submitted[competition,site] $\subseteq$
   SubmittedO[competition,site] $\cup$ ThematicCom[code,site]
2. Generalization constraints:

   Competition[code,site] $\subseteq$ OpenCom[code,site] $\cup$ ThematicCom[code,site]

   OpenCom[code,site] $\cap$ ThematicCom[code,site] = $\emptyset$

3. The constraint resulting from the elimination of ISA between the relationships **Winner** and **Submitted** has already been expressed as a foreign key on **Winner**.

4. The constraint resulting from the elimination of ISA between the relationships **SubmittedO** and **Submitted** has already been expressed as a foreign key on **SubmittedO**.

Problem 2: Restructuring of the relational schema

1. Null values in the database should be avoided.
2. When accessing the data on a poem, we always want to know its author.

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

- We take into account indication 1 by performing a mixed decomposition of the relation `Person`.
- Indication 2 is already taken into account since the author is part of the (external) identifier of the **Poem** entity. Therefore the **Authored** relationship has been merged into **Poem** during the direct translation, hence the primary key `taxcode` of the `Author` relation has become part of the primary key of the `Poem` relation.

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.

Person(taxcode, dob)

PersonWithOccupation(taxcode, occupation)

foreign key: PersonWithOccupation[taxcode] $\subseteq$ Person[taxcode]

There are no constraints that need to be changed or added due to the transformation.

Problem 3

Consider a database that includes the relations `Company` and `City`.
The relation `Company(name,type,city,numEmp)` stores for each company the name, the type, the city where it is registered, and the number of employees.
The relation `City(name,region)` stores for each city the region in which it is located.

1. Express **in SQL** the query that returns for each company type and for each region, the total number of employees of companies of that type registered in that region.
2. Express **in SQL** the query that returns the company type (or types) that have the highest total number of employees.
3. Express **in relational algebra** the query that computes the company that has (or the companies that have) the smallest number of employees, showing the name of the company, the region where it is registered, and the number of employees.

Problem 3: Solution (1/3)

```
Company(name,type,city,numEmp)
City(name,region)
```

1. Express **in SQL** the query that returns for each company type and for each region, the total number of employees of companies of that type registered in that region.

```
SELECT type, region, SUM(numEmp) AS totalEmp
FROM Company Co JOIN City Ci ON Co.city = Ci.name
GROUP BY type, region
```

Problem 3: Solution (2/3)

```
Company(name,type,city,numEmp)
City(name,region)
```

2. Express **in SQL** the query that returns the company type (or types) that have the highest total number of employees.

```
WITH totalEmpView AS
   (SELECT type, SUM(numEmp) AS totalEmp
    FROM Company
    GROUP BY type)
SELECT type
FROM totalEmpView
WHERE totalEmp >= ALL (SELECT totalEmp FROM totalEmpView)
```

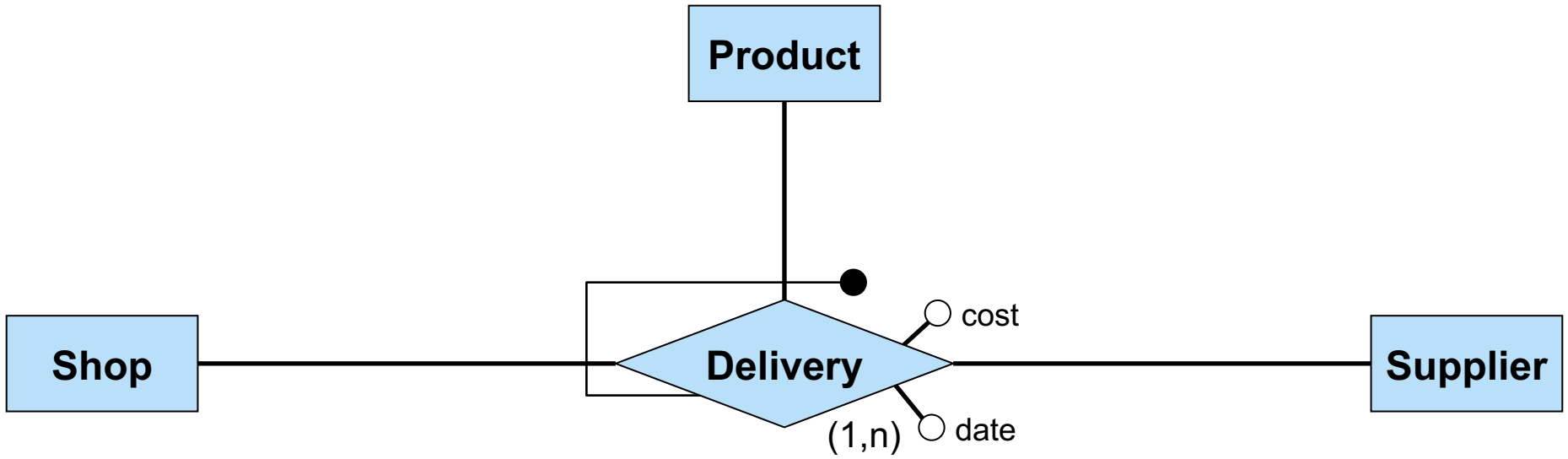
Problem 3: Solution (3/3)

Company(<u>name</u>,type,city,numEmp)
City(<u>name</u>,region)

3. Express **in relational algebra** the query that computes the company that has (or the companies that have) the smallest number of employees, showing the name of the company, the region where it is registered, and the number of employees.

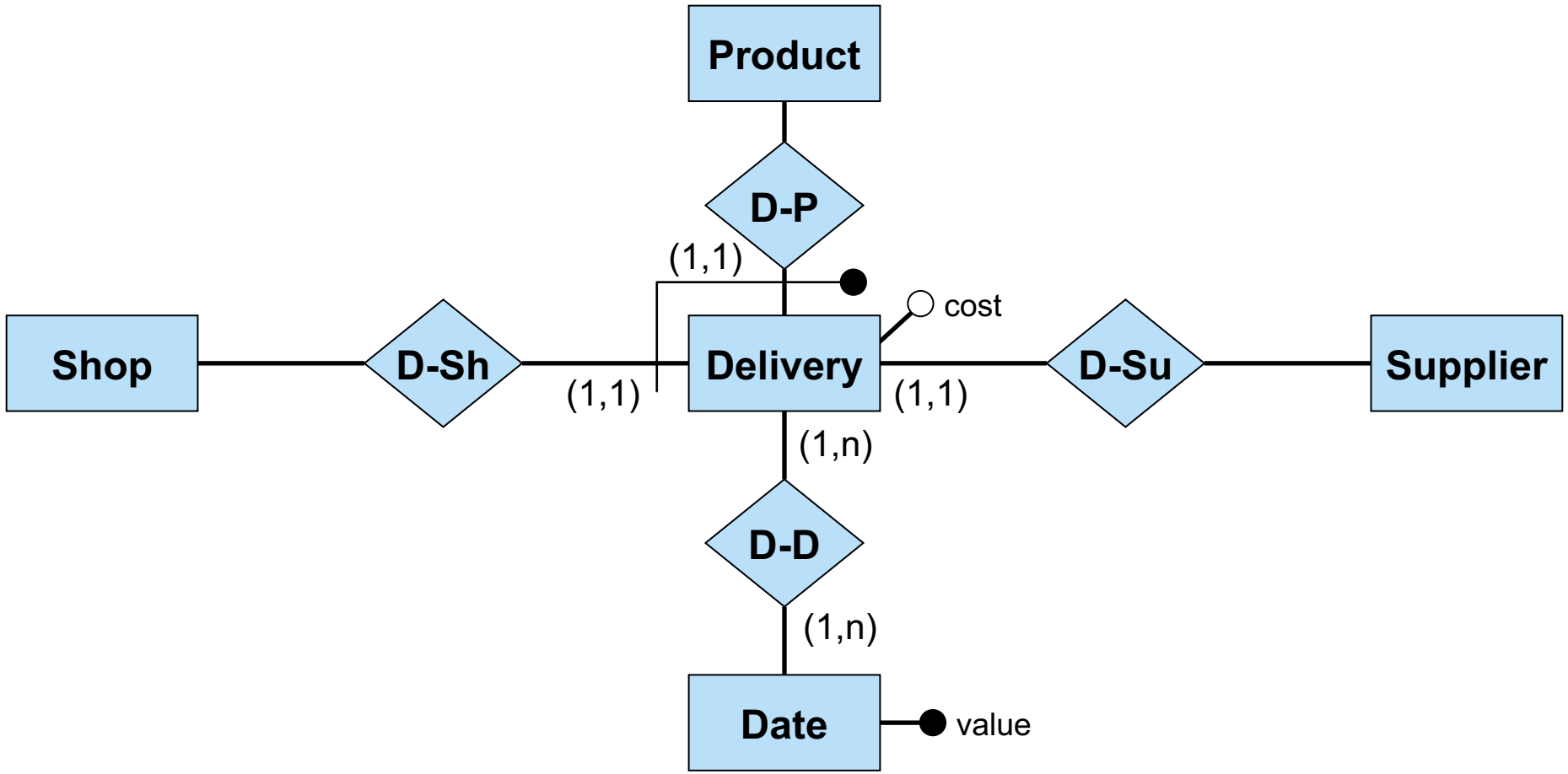
$$\textbf{PROJ}_{\text{name,region,numEmp}} (\textbf{REN}_{\text{city} \leftarrow \text{name}} (\text{City}) \textbf{ JOIN } (\textbf{PROJ}_{\text{name,city,numEmp}} (\text{Company}) - \textbf{PROJ}_{\text{name,city,numEmp}} (\textbf{SEL}_{\text{numEmp} > \text{e2}} (\text{Company } \textbf{JOIN } \textbf{PROJ}_{\text{n2,e2}} (\textbf{REN}_{\text{n2} \leftarrow \text{name, e2} \leftarrow \text{numEmp}} (\text{Company})))))))$$

Problem 4

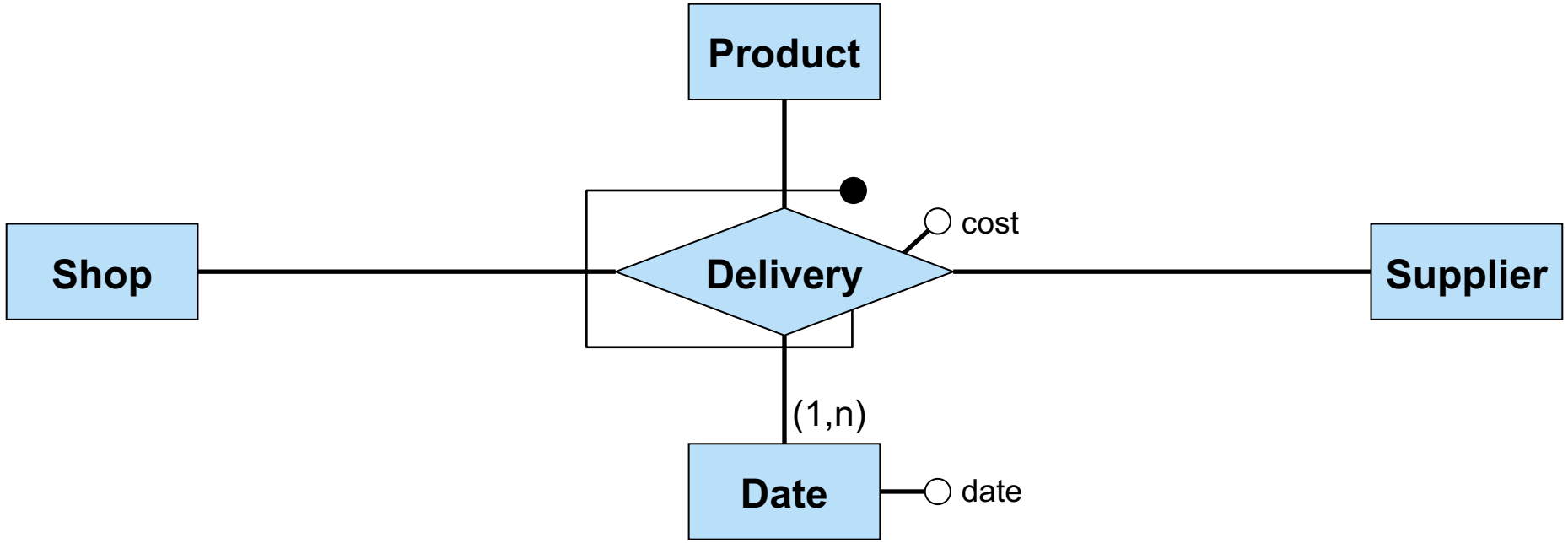


Consider the Entity-Relationship schema ***S*** shown above and show the corresponding restructured Entity-Relationship schema S_r that correctly captures its semantics (i.e., such that the instances of ***S*** and those of S_r can be put into a one-to-one correspondence).

Problem 4: Solution



Problem 4: Alternative solution with external constraint



In this alternative solution, one needs to add external constraints stating that, if two instances of **Delivery** agree on their **Product** and **Shop** components, then they must also agree on the **Supplier** component and they must have the same value for the **cost** attribute. Hence, this solution is less elegant.