Introduction to Databases

Exam of 29/01/2024

With Solutions

Diego Calvanese

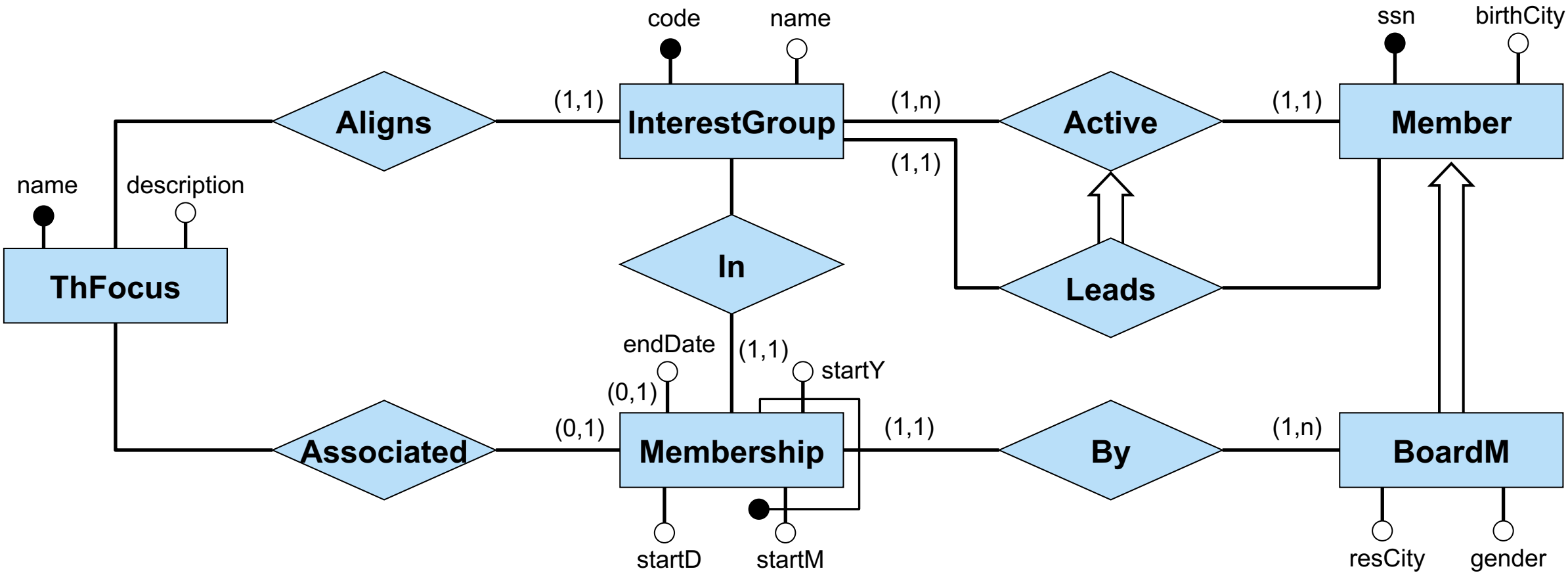
Bachelor in Computer Science
Faculty of Engineering
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema of an application related to an association with members and organized in various interest groups, for which the following information is of interest. Every **interest group** within the association has an identifier code, a name, a thematic focus it aligns with, a non-empty set of members currently active within the interest group, and a member among its currently active ones who leads it. For every **member** of the association, their social security number (which is an identifier) and the city of birth are of interest. A member is currently active in exactly one interest group. Some of the association members are **board members**, and for each of them, the city of residence, gender, and the interest groups in which they have been active in the past are also of interest. Every time a board member has been active in an interest group (currently or in the past), the date of joining the interest group, optionally the thematic focus they were associated with at that time, and the date of departure have been recorded. Notice that the date of departure is undefined for the interest group within which a board member is currently active. Take into account that a board member can switch interest group at most once per month. Of each **thematic focus**, the name (identifier) and a description are of interest.

Problem 1: Conceptual schema



Problem 1: Conceptual schema – External constraints

For each instance *I* of the schema:

1. If *b* ∈ *instances*(*I*,**BoardM**), there is exactly one *m* ∈ *instances*(*I*,**Membership**) such that (Membership:*m*,BoardM:*b*) ∈ *instances*(*I*,**By**) and for which there is no *d* such that (*m*,*d*) ∈ *instances*(*I*,**endDate**).
2. If *m* ∈ *instances*(*I*,**Membership**), there is no *d* such that (*m*,*d*) ∈ *instances*(*I*,**endDate**), (Membership:*m*,InterestGroup:*ig*) ∈ *instances*(*I*,**In**), and (Membership:*m*,BoardM:*b*) ∈ *instances*(*I*,**By**), then (Member:*b*,InterestGroup:*ig*) ∈ *instances*(*I*,**Active**).

In addition, one could express constraints related to the mutual consistency of the start and end dates of each instance of Membership, and of the dates of different instances of Membership for the same board member. For simplicity, we assume that such constraints are enforced at the application level, and not directly at the database level.

Problem 2

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

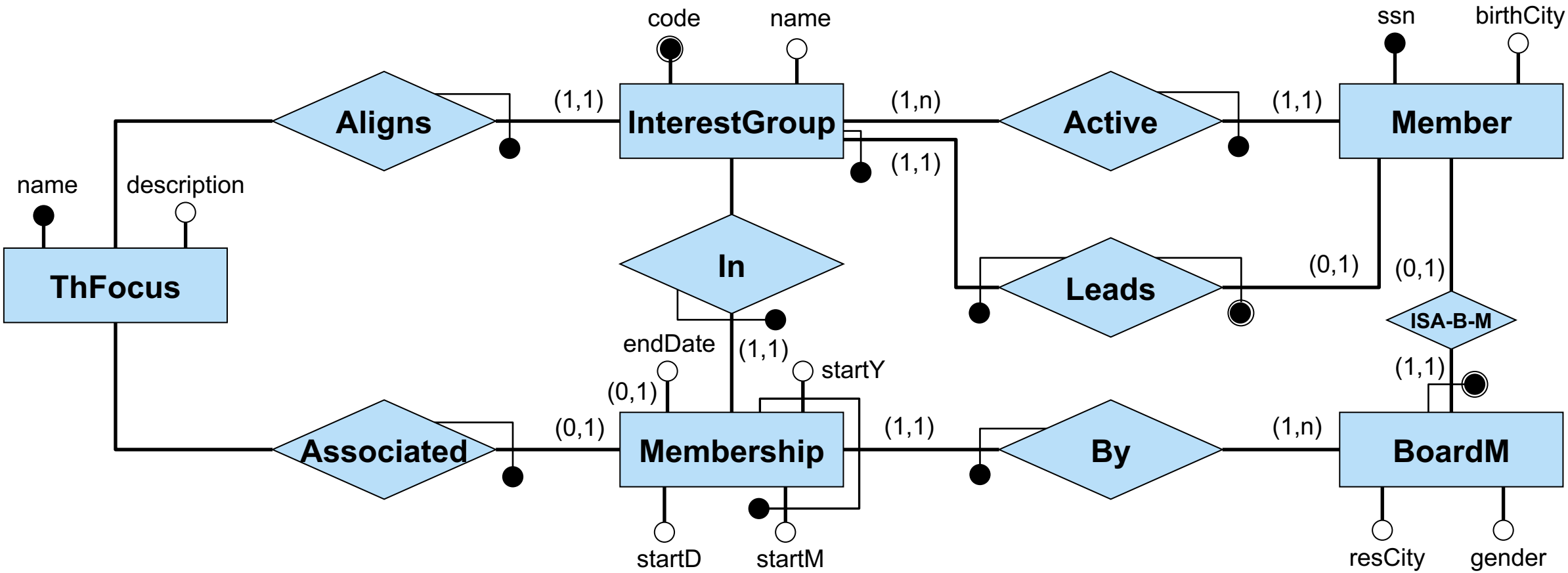
1. we want to avoid null values in the database;
2. every time we access the information about an association member, we always want to know whether they are a board member, and if so, we want to know their gender and city of residence.

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: Restructured conceptual schema – External constraints

For each instance I of the schema:

1. If $b \in instances(I,$**BoardM**$)$, there is exactly one $m \in instances(I,$**Membership**$)$ such that (Membership:m,BoardM:b) $\in instances(I,$**By**$)$ and for which there is no d such that $(m,d) \in instances(I,$**endDate**$)$.
2. If $m \in instances(I,$**Membership**$)$, there is no d such that $(m,d) \in instances(I,$**endDate**$)$, (Membership:m,InterestGroup:ig) $\in instances(I,$**In**$)$, (Membership:m,BoardM:b) $\in instances(I,$**By**$)$, and (BoardM:b,Member:mb) $\in instances(I,$**ISA-B-M**$)$, then (Member:mb,InterestGroup:ig) $\in instances(I,$**Active**$)$.
3. Constraint resulting from the elimination of ISA between relationships **Leads** and **Active**:
   If (Member:m,InterestGroup:ig) $\in instances(I,$**Leads**$)$ then (Member:m,InterestGroup:ig) $\in instances(I,$**Active**$)$.

Problem 2: Direct translation (1/2)

Member(ssn, birthCity)
  foreign key: Member[ssn] $\subseteq$ Active[member]

BoardM(ssn, resCity, gender)
  foreign key: BoardM[ssn] $\subseteq$ Member[ssn]
  inclusion: BoardM[ssn] $\subseteq$ Membership[boardM]

InterestGroup(code, name)
  inclusion: InterestGroup[code] $\subseteq$ Active[interestGroup]
  foreign key: InterestGroup[code] $\subseteq$ Leads[interestGroup]
  foreign key: InterestGroup[code] $\subseteq$ Aligns[interestGroup]

Active(member, interestGroup)
  foreign key: Active[member] $\subseteq$ Member[ssn]
  foreign key: Active[interestGroup] $\subseteq$ InterestGroup[code]

Leads(member, interestGroup)
  foreign key: Leads[member, interestGroup] $\subseteq$ Active[member, interestGroup]
  key: interestGroup

Problem 2: Direct translation (2/2)

ThFocus(name, description)

Aligns(interestGroup, thFocus)

foreign key: Aligns[interestGroup] $\subseteq$ InterestGroup[code]

foreign key: Aligns[thFocus] $\subseteq$ ThFocus[name]

Membership(boardM, startD, startM, startY, endDate*)

foreign key: Membership[boardM] $\subseteq$ BoardM[ssn]

foreign key: Membership[boardM,startM,startY] $\subseteq$ In[boardM,startM,startY]

In(boardM, startM, startY, interestGroup)

foreign key: In[boardM,startM,startY] $\subseteq$ Membership[boardM,startM,startY]

foreign key: In[interestGroup] $\subseteq$ InterestGroup[code]

Associated(boardM, startM, startY, thFocus)

foreign key: Associated[boardM,startM,startY] $\subseteq$ Membership[boardM,startM,startY]

foreign key: Associated[thFocus] $\subseteq$ ThFocus[name]

Problem 2: Direct translation – External constraints

1. This becomes a CHECK constraint on relation **Membership**:

```
NOT EXISTS (SELECT M1.boardM
            FROM Membership M1 JOIN Membership M2 ON M1.boardM = M2.boardM
            WHERE (M1.startM <> M2.startM OR M1.startY <> M2.startY) AND
                  M1.endDate IS NULL AND M2.endDate IS NULL)
AND NOT EXISTS
      (SELECT boardM FROM Membership M1
       WHERE NOT EXISTS (SELECT boardM FROM Membership M2
                         WHERE M1.boardM = M2.boardM AND M2.endDate IS NULL))
```

2. This becomes a CHECK constraint on relation **Membership**:

```
NOT EXISTS
      (SELECT In.boardM, In.interestGroup
       FROM Membership M JOIN In ON M.startM = In.startM AND M.startY = In.startY
                                    AND M.boardM = In.boardM
       WHERE M.endDate IS NULL AND
             NOT EXISTS (SELECT A.member, A.interestGroup FROM Active A
                         WHERE In.boardM = A.member AND
                               In.interestGroup = A.interestGroup))
```

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

Problem 2: Restructuring of the relational schema

1. we want to avoid null values in the database;
2. every time we access the information about an association member, we always want to know whether they are a board member, and if so, we want to know their gender and city of residence.

- We take into account indication 1 by performing a mixed decomposition of relation **Membership** into **PastBoardMembership** and **ActiveBoardMembership**.
- We take into account indication 2 by merging relation **BoardM** into **Member**. However, this would require making the attributes **resCity** and **gender** nullable, violating indication 1. Therefore, we perform a mixed decomposition of the resulting relation **Member** into **BoardMember** and **NonBoardMember**. Notice that is this way, we do not have anymore a single relation with all association members, which means that we have given priority to indication 1, while we cannot fully satisfy indication 2.
- Since due to constraint 1, there is exactly one tuple in **ActiveBoardMembership** for each tuple in **BoardMember** and vice-versa, we also merge relations **ActiveBoardMembership** and **BoardMember**, ending up with the three relations **NonBoardMember**, **BoardMember**, and **PastBoardMembership**.

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.

The relations **Member**, **BoardM**, and **Membership** are replaced by the following ones:

NonBoardMember(ssn, birthCity)
  foreign key: NonBoardMember[ssn] $\subseteq$ Active[member]

BoardMember(ssn, birthCity, resCity, gender, startD, startM, startY)
  foreign key: BoardMember[ssn] $\subseteq$ Active[member]
  foreign key: BoardMember[ssn] $\subseteq$ In[boardM]

PastBoardMembership(ssn, startD, startM, startY, endDate)
  foreign key: PastBoardMembership[ssn] $\subseteq$ BoardMember[ssn]
  foreign key: PastBoardMembership[ssn,startM,startY] $\subseteq$ In[boardM,startM,startY]

The foreign key constraint from **Active** to **Member** becomes an inclusion in a union:
  foreign key: Active[member] $\subseteq$ NonBoardMember[ssn] $\cup$ BoardMember[ssn]

Similarly, the foreign key constraint from **In** to **Membership** becomes an inclusion in a union:
  foreign key: In[boardM,startM,startY] $\subseteq$ BoardMember[ssn,startM,startY] $\cup$
  PastBoardMembership[ssn,startM,startY]

The constraints 1 and 2 (and also 3) are now directly enforced through the key and foreign key constraints of the restructured relational schema.

Problem 3

Consider a database containing the two tables `Delivery(courier,item,date,district)`, which stores the information regarding courier deliveries, each with the courier, the delivered item, and the date and city district where the delivery was done, and `Express(courier,item,date)`, which stores the same information (without the district), but only for the express deliveries (these are a subset of all deliveries).

1. Write a **SQL** query that returns, for each courier and each district in which the courier has delivered some item, the courier, the district, and the date on which the courier has done their first delivery in that district.
2. Write a **SQL** query that computes, for each courier that has done some express delivery and for each district, the average number of items per day that that courier has delivered express to that district, returning that number together with the courier and the district.
3. Write a **relational algebra** query that computes all couriers that on at least one date have done both an express delivery and a non-express delivery.

Problem 3: Solution (1/3)

```
Delivery(courier,item,date,district)
Express(courier,item,date)
```

1. Write a **SQL** query that returns, for each courier and each district in which the courier has delivered some item, the courier, the district, and the date on which the courier has done their first delivery in that district.

```
SELECT courier, district, MIN(date)
FROM Delivery
GROUP BY courier, district
```

Problem 3: Solution (2/3)

```
Delivery(courier,item,date,district)
Express(courier,item,date)
```

2. Write a **SQL** query that computes, for each courier that has done some express delivery and for each district, the average number of items per day that that courier has delivered express to that district, returning that number together with the courier and the district.

```
WITH CountItems AS
  (SELECT D.courier, D.district, D.date, COUNT(D.item) AS numitems
   FROM Delivery D JOIN Express E
          ON D.courier = E.courier AND
             D.item = E.item AND D.date = E.date
   GROUP BY D.courier, D.district, D.date)

SELECT courier, district, AVG(numitems)
FROM  CountItems
GROUP BY courier, district
```

Problem 3: Solution (3/3)

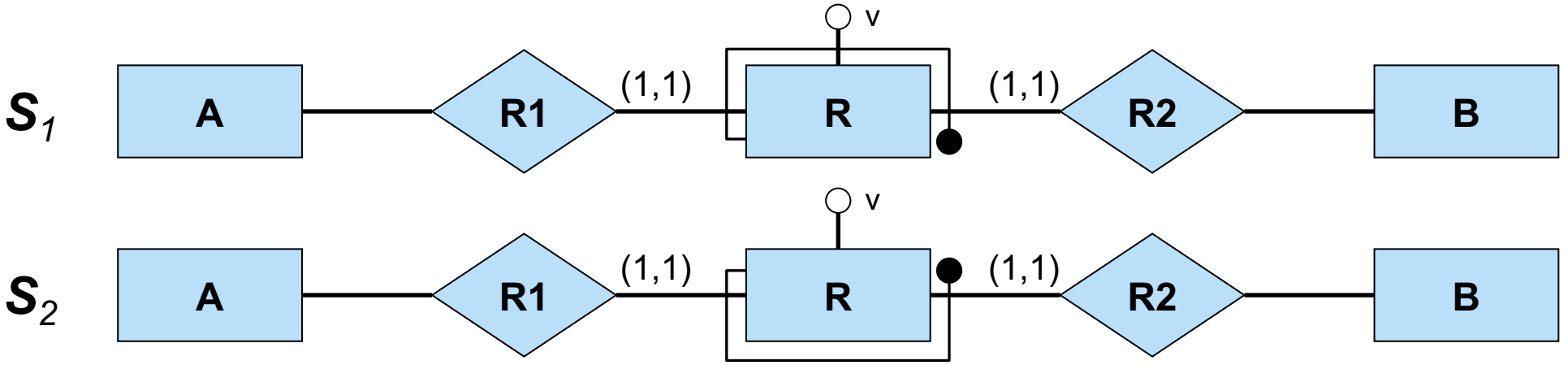
Delivery(<u>courier</u>,<u>item</u>,<u>date</u>,district)
Express(<u>courier</u>,<u>item</u>,<u>date</u>)

3. Write a **relational algebra** query that computes all couriers that on at least one date have done both an express delivery and a non-express delivery.

$$\text{PROJ}_{\text{courier}}((\text{PROJ}_{\text{courier,date}}(\text{PROJ}_{\text{courier,item,date}}(\text{Delivery}) - \text{Express})) \text{ INTERSECT } \text{PROJ}_{\text{courier,date}}(\text{Express}))$$

Notice that we have to take into account that the relation **Delivery** contains also all express deliveries. Therefore we obtain the non-express deliveries via set difference.

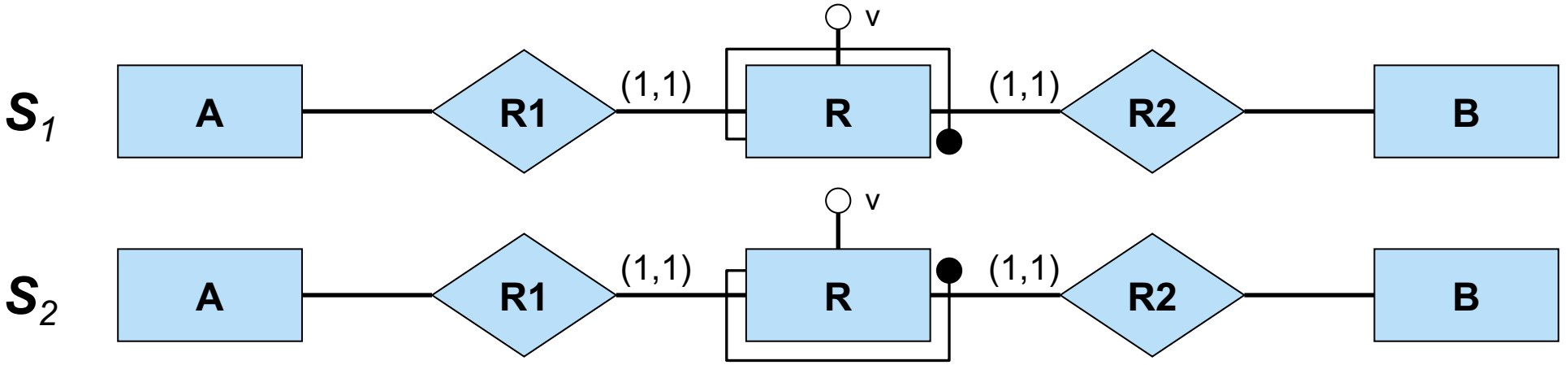
Problem 4



Consider the two ER schemas S_1 and S_2 shown above.

1. Is there a legal instance of S_1 that is *not* a legal instance of S_2?
   If yes, show such an instance;
   if no, argue why such an instance cannot exist.
2. Is there a legal instance of S_2 that is *not* a legal instance of S_1?
   If yes, show such an instance;
   if no, argue why such an instance cannot exist.

Problem 4: Solution



1. Yes. The following instance ***I*** is a legal instance of $\boldsymbol{S_1}$ but *not* a legal instance of $\boldsymbol{S_2}$: *instances*(***I***,***R***) = {r_1,r_2}, *instances*(***I***,***A***) = {a}, *instances*(***I***,***B***) = {b}, *instances*(***I***,***R1***) = {(R:r_1,A:a), (R:r_2,A:a)}, *instances*(***I***,***R2***) = {(R:r_1,B:b), (R:r_2,B:b)}, *instances*(***I***,***v***) = {(r_1,v_1), (r_2,v_2)}.
2. No, since every legal instance of $\boldsymbol{S_2}$ is also a legal instance of $\boldsymbol{S_1}$. Indeed, consider a legal instance ***I*** of $\boldsymbol{S_2}$. Due to the identifier of ***R***, in ***I*** there cannot be two instances of ***R*** that coincide in their participation to ***R1*** and ***R2***. But this obviously means also that there cannot be two instances of ***R*** that coincide in their participation to ***R1*** and ***R2***, and in addition coincide in their value of attribute ***v***. Hence ***I*** is also a legal instance of $\boldsymbol{S_1}$.