Introduction to Databases

Exam of 04/07/2025

With Solutions

Diego Calvanese

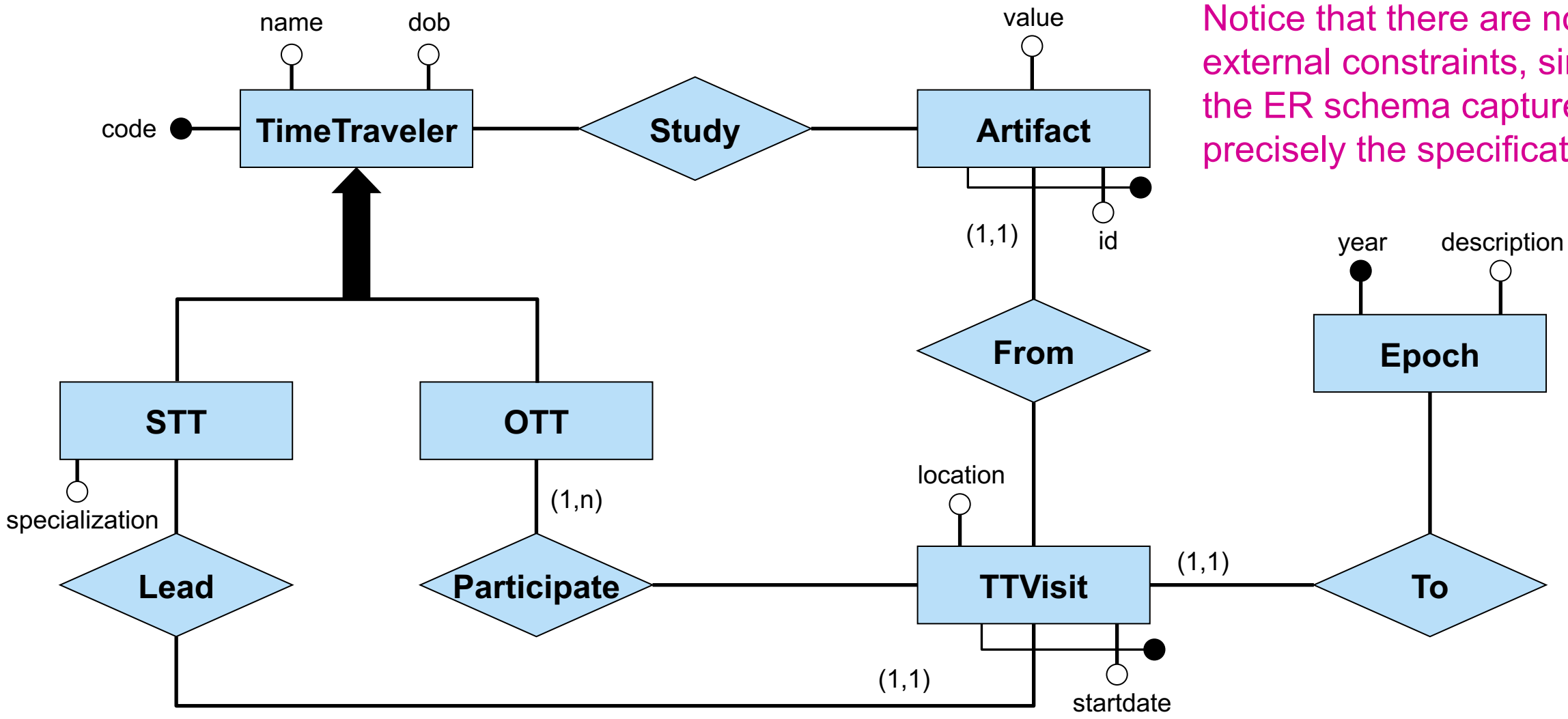
Bachelor in Computer Science
Faculty of Engineering
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema of an application for the Time Travel Agency, which is interested in maintaining the information about time travel (TT) visits carried out by time travelers to different epochs. Each **epoch** is identified by a year, and we are also interested in its description. Each **time traveler** is identified by their tta-code, and they have a name and a date of birth. There are exactly two types of time travelers: **ordinary time travelers** (OTTs), who participate in TT-visits, and they must have participated in at least one; and **specialized time travelers** (STTs), who have a specialization and who may lead TT-visits (but otherwise do not participate in them). Each **TT-visit** is led by exactly one STT, is to a certain epoch, and has a start-date and a location. Notice that there may not be two TT-visits with the same start-date led by the same STT. As mentioned, we are interested in the OTTs who participate to a TT-visit (who may be zero, one, or more). The Time Travel Agency also maintains artifacts that have been obtained from TT-visits, where each **artifact** is identified by an id that is unique within the TT-visit from which it has been obtained. Of artifacts, we are also interested in their value and in the time travelers who are studying them.

Problem 1: Conceptual schema



Notice that there are no external constraints, since the ER schema captures precisely the specification.

Problem 2

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

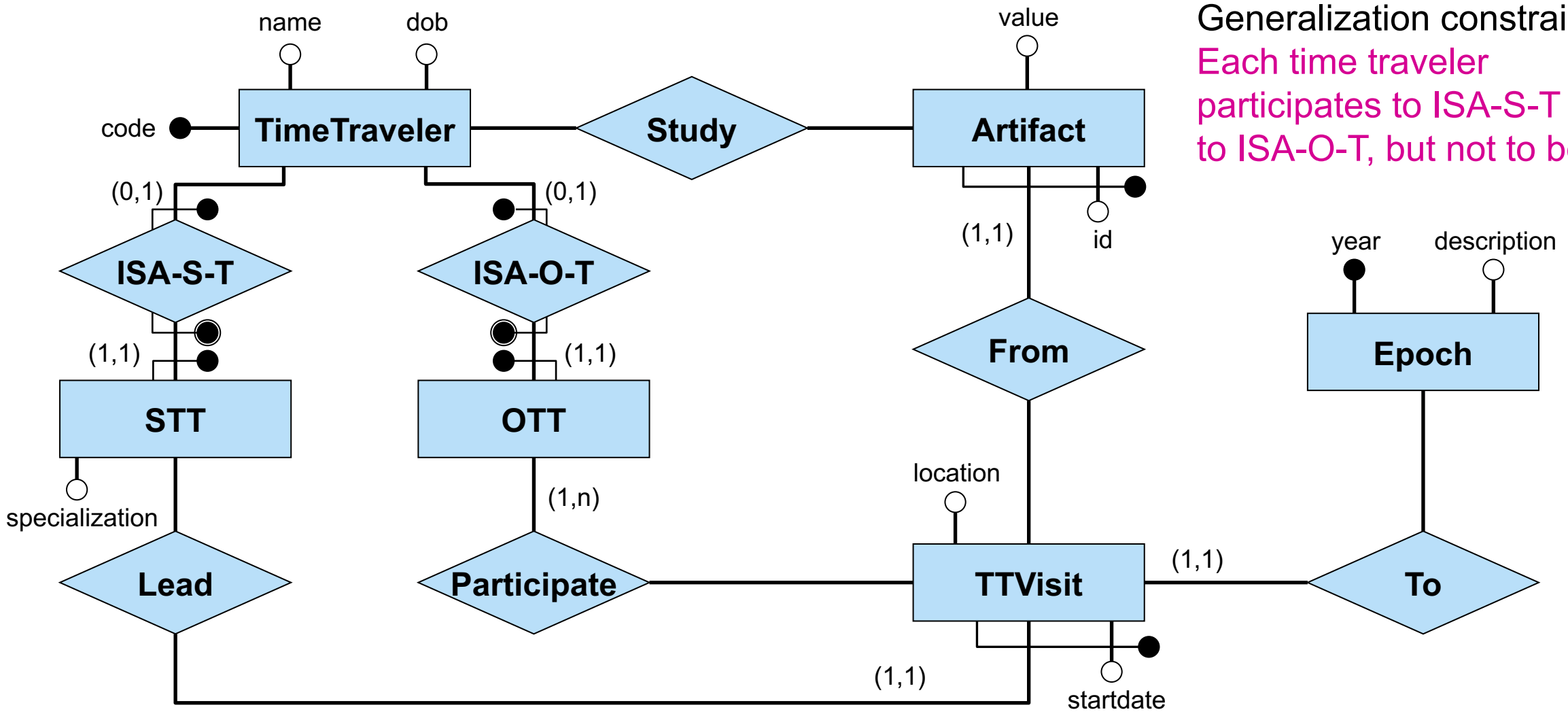
1. We access STTs separately from OTTs.
2. Every time we access a TT-visit, we always want to know the epoch to which it was done.

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



Generalization constraint:
Each time traveler participates to ISA-S-T or to ISA-O-T, but not to both.

Problem 2: Direct translation (1/2)

TimeTraveler(code, name, dob)

STT(code, specialization)
  foreign key: STT[code] $\subseteq$ TimeTraveler[code]

OTT(code)
  foreign key: OTT[code] $\subseteq$ TimeTraveler[code]
  inclusion: OTT[code] $\subseteq$ Participate[ott]

TTVisit(startDate, leader, location)
  foreign key: TTVisit[stt] $\subseteq$ STT[code]
  foreign key: TTVisit[startdate,leader] $\subseteq$ To[ttVisitStartdate,ttVisitLeader]

Participate(ott, ttVisitStartdate, ttVisitLeader)
  foreign key: Participate[ott] $\subseteq$ OTT[code]
  foreign key: Participate[ttVisitStartdate,ttVisitLeader] $\subseteq$ TTVisit[startDate,leader]

Problem 2: Direct translation (2/2)

Artifact(id, ttVisitStartdate, ttVisitLeader, value)
  foreign key: Artifact[ttVisitStartdate,ttVisitLeader] $\subseteq$ TTVisit[startDate,leader]

Study(timeTraveler, artifact, ttVisitStartDate, ttVisitLeader)
  foreign key: Study[timeTraveler] $\subseteq$ TimeTraveler[code]
  foreign key: Study[artifact,ttVisitStartdate,ttVisitLeader] $\subseteq$
    Artifact[id,ttVisitStartdate,ttVisitLeader]

Epoch(year, description)

To(ttVisitStartdate, ttVisitLeader, epoch)
  foreign key: To[ttVisitStartdate,ttVisitLeader] $\subseteq$ TTVisit[startDate,leader]
  foreign key: To[epoch] $\subseteq$ Epoch[year]

Generalization constraints:

$STT[code] \cap OTT[code] = \emptyset$

$TimeTraveler[code] \subseteq STT[code] \cup OTT[code]$

Problem 2: Restructuring of the relational schema

Indications:

1. We access STTs separately from OTTs.
2. Every time we access a TT-visit, we always want to know the epoch to which it was done.

- We take into account Indication 1 by performing a horizontal decomposition of relation **TimeTraveler** into **STimeTraveler** and **OTimeTraveler**. We further merge **STimeTraveler** with **STT** and **OTimeTraveler** with **OTT**, since after the horizontal decomposition, due to the generalization constraints, these two pairs of relations are strongly coupled.
- We take into account Indication 2 by merging **To** and **TTVisit** (which are strongly coupled).

Problem 2: Restructured relational schema

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

Specifically, the relations **TimeTraveler** and **To** are removed, and the relations **STT**, **OTT**, and **TTVisit** are modified as follows:

STT(code, specialization, name, dob)

OTT(code , name, dob)
  inclusion: OTT[code] $\subseteq$ Participate[ott]

TTVisit(startDate, leader, location, epoch)
  foreign key: TTVisit[stt] $\subseteq$ STT[code]
  foreign key: TTVisit[epoch] $\subseteq$ Epoch[year]

Moreover, the foreign key: Study[timeTraveler] $\subseteq$ TimeTraveler[code] of relation **Study**, becomes the following constraint:
  foreign key: Study[timeTraveler] $\subseteq$ STT[code] $\cup$ OTT[code]

The generalization constraint simplifies to:
  STT[code] $\cap$ OTT[code] = $\emptyset$

Problem 3

Consider a database *D* containing the two relations:

i. `Teacher(code,cityOfBirth,dateOfBirth)`, which stores the code (primary key), the city of birth, and the date of birth of a set of teachers;

ii. `Competition(teacherCode,year,score)`, which store the code of the teacher, the year, and the score obtained in a university competition in which the teacher has participated (the rule is that no teacher can participate in university competitions more than once in a year).

We know that when the score for a competition for a teacher is not known, the attribute `score` for that competition stores the null value, while in the other attributes of the two relations, null values never appear. We also know that a foreign key constraint is defined from `teacherCode` in `Competition` to `code` in `Teacher`.

Write in **SQL** the following queries over *D*:

1. Compute the **code** of all teachers, each with the **number of competitions** in which they have participated.
2. For each teacher born in `Bolzano`, compute the **code** and the **minimum score** obtained in any competitions they have participated in.
3. For those teachers who have participated in at least 5 competitions, compute the **code**, the **date of birth**, and the **average score** obtained in the competitions in which the score is known.

Problem 3: Solution 1

```
Teacher(code,cityOfBirth,dateOfBirth)
Competition(teacherCode,year,score)
```

1. Compute the **code** of all teachers, each with the **number of competitions** in which they have participated.

```
SELECT T.code, COUNT(C.year) AS numCompetitions
FROM Teacher T LEFT OUTER JOIN Competition C ON
       T.code = C.teacherCode
GROUP BY T.code
```

Problem 3: Solution 2

```
Teacher(code,cityOfBirth,dateOfBirth)
Competition(teacherCode,year,score)
```

2. For each teacher born in `Bolzano`, compute the **code** and the **minimum score** obtained in any competitions they have participated in.

```
SELECT T.code, MIN(C.score) AS minScore
FROM Teacher T LEFT OUTER JOIN Competition C ON
       T.code = C.teacherCode
WHERE T.cityOfBirth = 'Bolzano'
GROUP BY T.code
```

Problem 3: Solution 3

```
Teacher(code,cityOfBirth,dateOfBirth)
Competition(teacherCode,year,score)
```

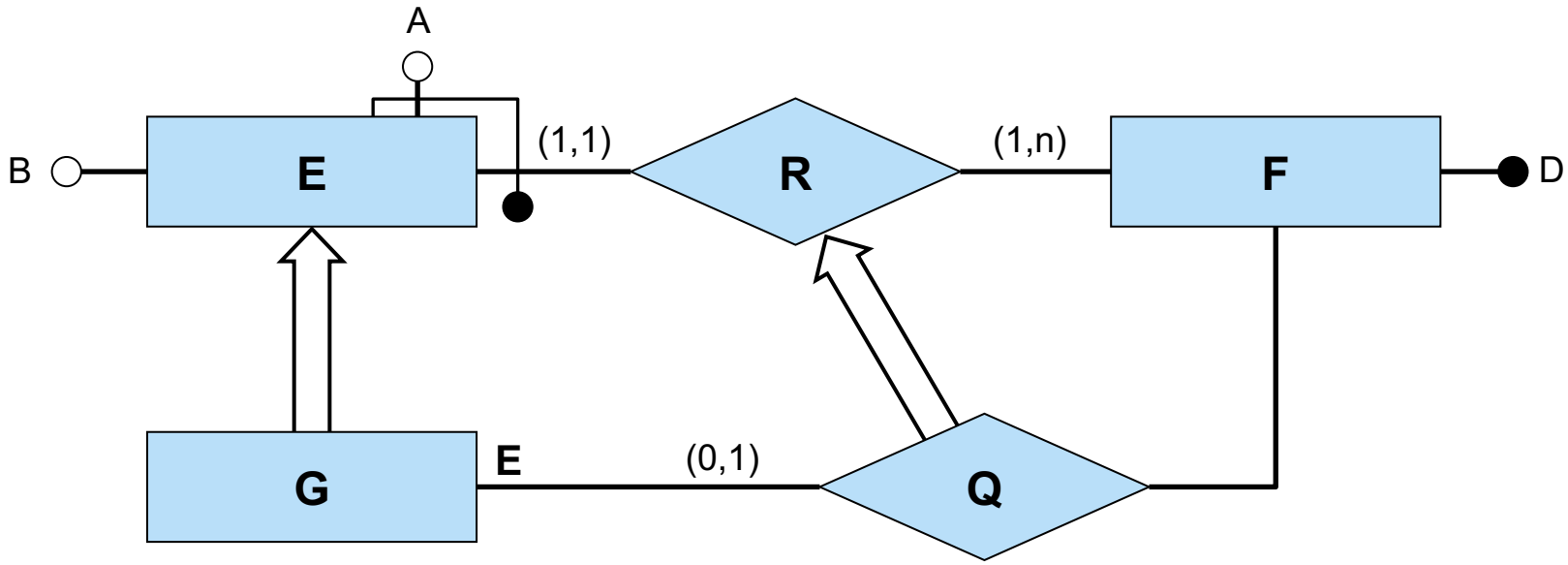
3. For those teachers who have participated in at least 5 competitions, compute the **code**, the **date of birth**, and the **average score** obtained in the competitions in which the score is known.

```
SELECT T.code, T.dateOfBirth, AVG(C.score) AS avgScore
FROM Teacher T JOIN Competition C ON T.code = C.teacherCode
GROUP BY T.code, T.dateOfBirth
HAVING COUNT(C.year) >= 5
```

Notice that when computing the average, NULL values for the score are ignored, while in `COUNT(C.year)`, also competitions with a NULL score are considered.

Problem 4

Consider the conceptual schema ***S*** shown below and carry out the logical design starting from ***S***, considering that you should produce the logical schema (with integrity constraints) with as few relations as possible.



Problem 4: Solution

We first eliminate the ISA relations between ***G*** and ***E***, and between ***Q*** and ***R***, obtaining a conceptual schema ***S'***.

The direct translation of ***S'***, results in the following relational schema:

F(D), inclusion: F[D] $\subseteq$ E[F]
E(A, F, B), foreign key: E[F] $\subseteq$ F[D]
G(A, F), foreign key: G[A,F] $\subseteq$ E[A,F]
Q(A, F, F1), foreign key: Q[A,F] $\subseteq$ G[A,F], for each tuple *t* of Q, we have *t* [F] = *t* [F1]

Restructuring of the relational schema:

- Since F[D] = E[F], the relation ***F*** is redundant and can be eliminated.
- The attribute ***F1*** of ***Q***, being always equal to attribute ***F***, can be eliminated.
- We can then merge ***Q*** into ***G***, adding to ***G*** a boolean flag attribute ***Q***, which indicates whether the tuple participates to ***Q*** or not.
- We can then further merge ***G*** into ***E***, changing the boolean flag attribute ***Q*** to a three-valued flag attribute. A tuple in ***E*** will have:
  - value `NULL` for ***Q***, if the tuple is in ***E*** but not in ***G*** (and hence does not participate to ***Q***);
  - value `g` for ***Q***, if the tuple is in ***G*** but does not participate to ***Q***; and
  - value `q` for ***Q***, if the tuple is in ***G*** and participates to ***Q***.

Summing up, a relational schema that captures the semantics of the conceptual schema ***S*** is
E(A, F, B, Q*), where E[Q] $\subseteq$ {`NULL`, `g`, `q`}.