Introduction to Databases

Exam of 20/02/2026

With Solutions

Diego Calvanese

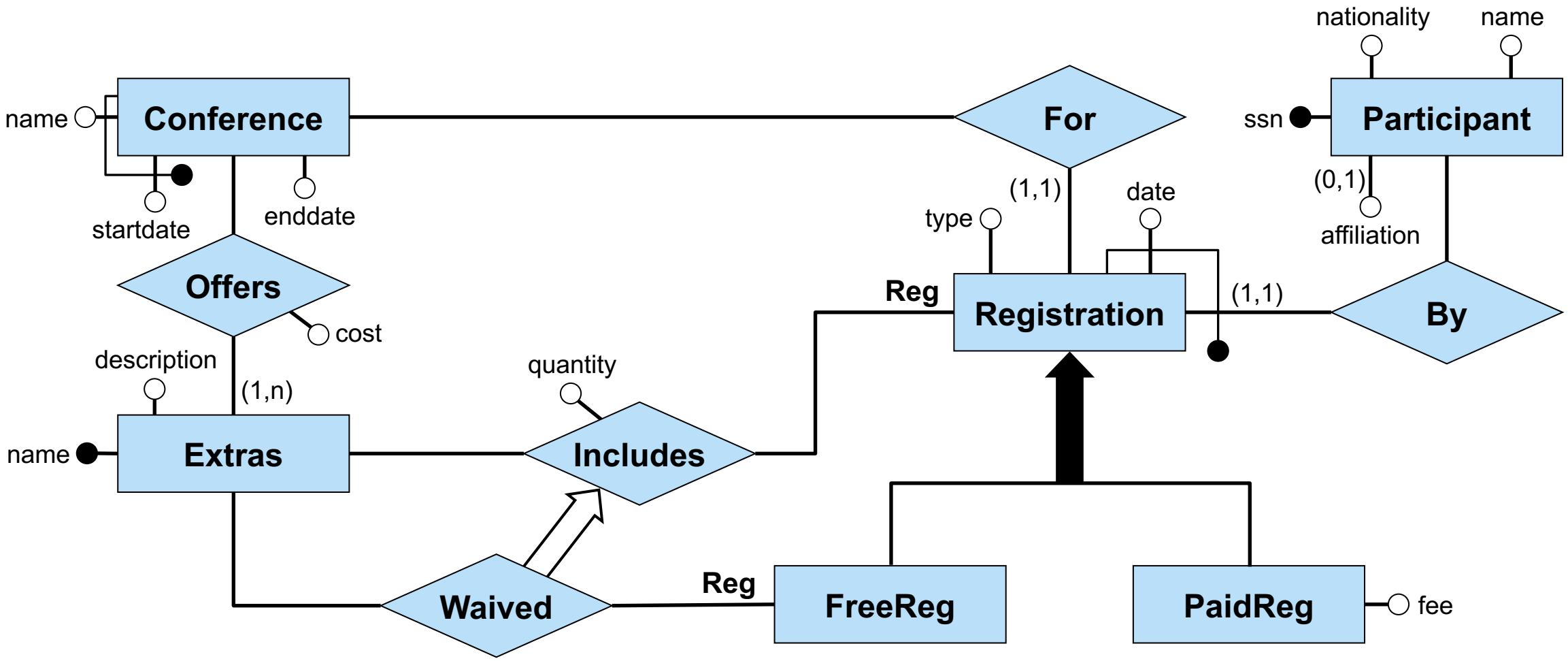
Bachelor in Computer Science
Faculty of Engineering
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema for an application that manages conferences and the participants who register for them. Each **participant** is uniquely identified by an ssn, and we are also interested in their name, nationality, and affiliation (if available). Each **conference** is uniquely identified by its name and start-date, and for each conference we also want to know its end-date. Each **registration** refers to exactly one participant and exactly one conference, and we are also interested in the registration date and the registration type (e.g., full, workshop-only, etc.). Notice that a participant cannot have more than one registration on the same date. A conference may offer some extras (e.g., lunch vouchers), whose cost depends on the conference offering them. Each **extra** is uniquely identified by its name, and we also want to know its description, the conferences that offer it (at least one), and the corresponding cost at each conference. A registration may include some extras, with a quantity (e.g., 5 lunch vouchers). Note that these extras have to be among those offered by the conference for which the registration is done. There are exactly two forms of registrations: **paid registrations**, which are associated with a fee; and **free registrations**, for which some of the included extras may be waived (e.g., if a free registration includes lunch vouchers, they may be waived).

Problem 1: Conceptual schema



Problem 1: Conceptual schema – External constraints

1. “The extras included in a registration have to be among those offered by the conference for which the registration is done.”
   Formally: For each instance I of the schema, if (Reg:r, Extras:e) $\in$ *instances*(I,**Includes**) and (Registration:r, Conference:c) $\in$ *instances*(I,**For**), then (Conference:c, Extras:e) $\in$ *instances*(I,**Offers**).

2. The enddate of a conference must not be before the startdate.
   Formally: For each instance I of the schema, if $c \in$ *instances*(I,**Conference**), (c,s) $\in$ *instances*(I,**startdate**), and (c,e) $\in$ *instances*(I,**enddate**), then $s \leq e$.

Problem 2

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

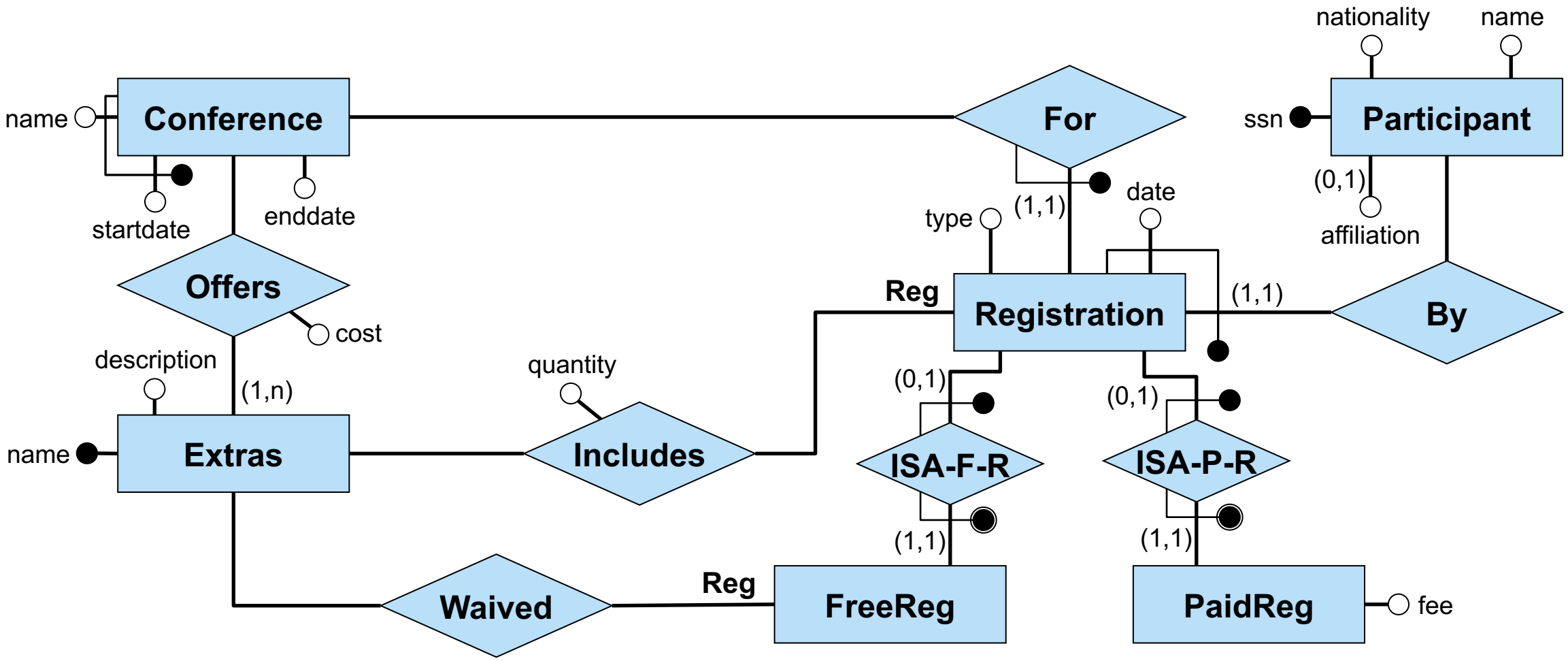
1. We access free and paid registrations together, and every time we access a registration, we always want to know whether it is free or paid.
2. Every time we access a registration, we always want to know the conference for which it is done.

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



Problem 1: Conceptual schema – External constraints

1. "The extras included in a registration have to be among those offered by the conference for which the registration is done."
   Formally: For each instance I of the schema, if (Reg:r, Extras:e) $\in$ *instances*(I,**Includes**), consider the unique $c \in$ *instances*(I,**Conference**) such that (Registration:r, Conference:c) $\in$ *instances*(I,**For**).
   Then (Conference:c, Extras:e) $\in$ *instances*(I,**Offers**).
2. As for the original ER schema.
3. Generalization constraint: Each instance of **Registration** participates either to **ISA-F-R** or to **ISA-P-R**, but not to both.
4. Constraint resulting from the elimination of ISA between **Waived** and **Includes**:
   For each instance I of the schema, if (Reg:f, Extras:e) $\in$ *instances*(I,**Waived**), consider the unique $r \in$ *instances*(I,**Registration**) such that (FreeReg:f, Registration:r) $\in$ *instances*(I,**ISA-F-R**).
   Then (Reg:r, Extras:e) $\in$ *instances*(I,**Includes**).

Problem 2: Direct translation (1/2)

Conference(name, sdate, edate)

Participant(ssn, name, nationality, affiliation*)

Extras(name, description)

inclusion: Extras[name] $\subseteq$ Offers[extras]

Offers(conference, sdate, extras, cost)

foreign key: Offers[conference,sdate] $\subseteq$ Conference[name,sdate]

foreign key: Offers[extras] $\subseteq$ Extras[name]

Registration(date, participant, type)

foreign key: Registration[date,participant] $\subseteq$ For[date,participant]

foreign key: Registration[participant] $\subseteq$ Participant[ssn]

For(date, participant, conference, sdate)

foreign key: For[date,participant] $\subseteq$ Registration[date,participant]

foreign key: For[conference,sdate] $\subseteq$ Conference[name,sdate]

Problem 2: Direct translation (2/2)

FreeReg(<u>date</u>, <u>participant</u>)
  foreign key: FreeReg[date,participant] $\subseteq$ Registration[date,participant]

PaidReg(<u>date</u>, <u>participant</u>, fee)
  foreign key: PaidReg[date,participant] $\subseteq$ Registration[date,participant]

Includes(<u>date</u>, <u>participant</u>, <u>extras</u>, quantity)
  foreign key: Includes[date,participant] $\subseteq$ Registration[date,participant]
  foreign key: Includes[extras] $\subseteq$ Extras[name]

Waived(<u>date</u>, <u>participant</u>, <u>extras</u>)
  foreign key: Waived[date,participant] $\subseteq$ FreeReg[date,participant]
  foreign key: Waived[extras] $\subseteq$ Extras[name] (Notice that this FK is implied by the next one and the FK on Includes, thus it could be omitted.)
  foreign key: Waived[date,participant,extras] $\subseteq$ Includes[date,participant,extras]
(This FK expresses the constraint resulting from the elimination of ISA between **Waived** and **Includes**.)

Problem 2: Direct translation – Constraints

1. The constraint "The extras included in a registration have to be among those offered by the conference for which the registration is done." can be expressed as:

   $\mathbf{PROJ}_{conference,sdate,extras}(\text{Includes } \mathbf{JOIN} \text{ FOR}) \subseteq \mathbf{PROJ}_{conference,sdate,extras}(\text{Offers})$

2. Check clause on relation **Conference**: `CHECK (startdate <= enddate)`.

3. Generalization constraints:
   FreeReg[date,participant] ∩ PaidReg[date,participant] = ∅
   Registration[date,participant] ⊆
   FreeReg[date,participant] ∪ PaidReg[date,participant]

4. The constraint resulting from the elimination of ISA between **Waived** and **Includes** has been represented as a foreign key for **Waived**.

Problem 2: Restructuring of the relational schema

Indications:

1. We access free and paid registrations together, and every time we access a registration, we always want to know whether it is free or paid.
2. Every time we access a registration, we always want to know the conference for which it is done.

- We take into account Indication 1 by merging relations **FreeReg** and **PaidReg** into **Registration**, and making the attribute **fee** (coming from **PaidReg**) optional. This is possible since the three relations result from the translation of a complete hierarchy in the ER diagram. We use **fee** to indicate whether a registration is free or paid: it is free exactly when **fee** is NULL.
- We take into account Indication 2 by merging **Registration** and **For** (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 **FreeReg**, **PaidReg**, and **For** are removed, and the relation **Registration** is modified as follows:

Registration(<u>date</u>, <u>participant</u>, conference, sdate, type, fee*)
  foreign key: Registration[conference,sdate] $\subseteq$ Conference[name,sdate]
  foreign key: Registration[participant] $\subseteq$ Participant[ssn]

Moreover, the foreign key: Waived[date,participant] $\subseteq$ FreeReg[date,participant] of relation **Waived**, becomes the following constraint:

$$\textbf{PROJ}_{date,participant}(\text{Waived}) \subseteq \textbf{PROJ}_{date,participant}(\textbf{SEL}_{fee\ \text{IS NULL}}(\text{Registration}))$$

We also observe that after merging **For** and **Registration**, external constraint 1 has to be expressed as:

$$\textbf{PROJ}_{conference,sdate,extras}(\text{Includes } \textbf{JOIN} \text{ Registration}) \subseteq \textbf{PROJ}_{conference,sdate,extras}(\text{Offers})$$

Problem 3

Consider a database *D* containing the two relations:

i. `Review(viewer,movie,rating)`, which stores the ratings given by viewers to movies;

ii. `Subscribed(viewer,platform)`, which stores which viewers are subscribed to which viewing platforms.

Write the following queries over *D*:

1. Write a query in **relational algebra** that computes the platforms that do not have any movie with some rating given by their subscribed viewers that is less than 4.
2. Write a query in **SQL** that computes which viewers subscribed to the platform "`Netflix`" have an average rating for the movies they reviewed less than 3.
3. Write a query in **SQL** that computes the platforms that do not have any movie for which the average rating given by their subscribed viewers is less than 4.

Problem 3: Solution 1

```
Review(viewer,movie,rating)
Subscribed(viewer,platform)
```

1. Write a query in **relational algebra** that computes the platforms that do not have any movie with some rating given by their subscribed viewers that is less than 4.

PROJ$_{platform}$ (Subscribed) **–**
(**PROJ**$_{platform}$ (Subscribed **JOIN SEL**$_{rating < 4}$ (Review)))

Problem 3: Solution 2

```
Review(viewer,movie,rating)
Subscribed(viewer,platform)
```

2. Write a query in **SQL** that computes which viewers subscribed to the platform "`Netflix`" have an average rating for the movies they reviewed less than 3.

```
SELECT R.viewer
FROM Review R JOIN Subscribed S ON S.viewer = R.viewer
WHERE S.platform = 'Netflix'
GROUP BY R.viewer
HAVING AVG(R.rating) < 3
```

Problem 3: Solution 3

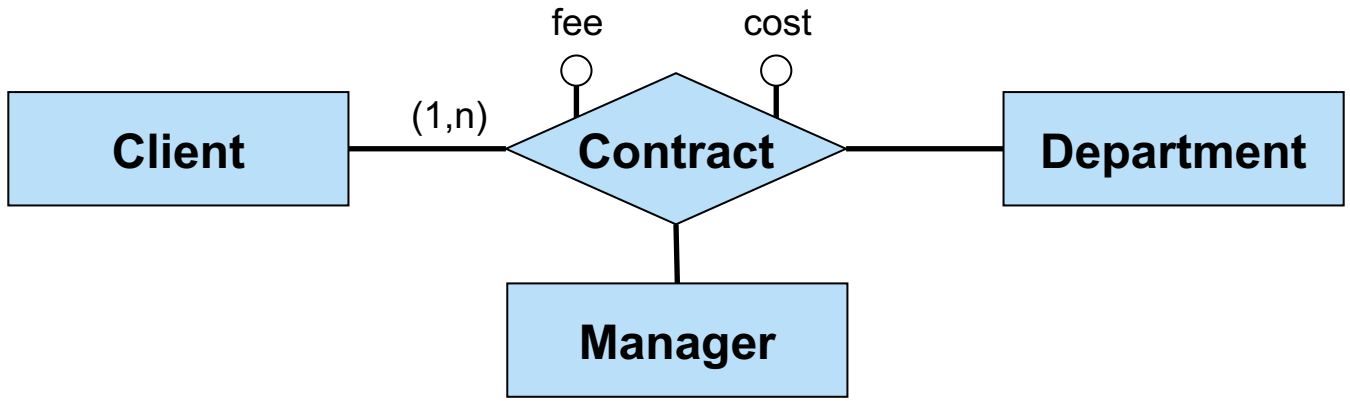
```
Review(viewer,movie,rating)
Subscribed(viewer,platform)
```

3. Write a query in **SQL** that computes the platforms that do not have any movie for which the average rating given by their subscribed viewers is less than 4.

```
SELECT DISTINCT platform
FROM Subscribed
WHERE platform NOT IN
    (SELECT S.platform
     FROM Review R JOIN Subscribed S ON S.viewer = R.viewer
     GROUP BY R.movie, S.platform
     HAVING AVG(R.rating) < 4)
```

Problem 4

The following conceptual schema ***S*** models the contracts between a department, a responsible manager, and a client of a service company: for each service contract, the relevant information is: *(1)* the fee charged to the client for the basic service, and *(2)* the extra cost charged to the client for each additional product delivered under the contract (the same cost for all products). Following a change decided by the company, the requirements are revised: *(i)* each department may not enter into more than 5 contracts, and *(ii)* for each contract, the different products delivered under that contract should also be modeled, and the extra cost will no longer be unique for all products, but its value may vary among such products. Show the conceptual schema resulting from the revision of schema ***S*** so as to comply with the new requirements.

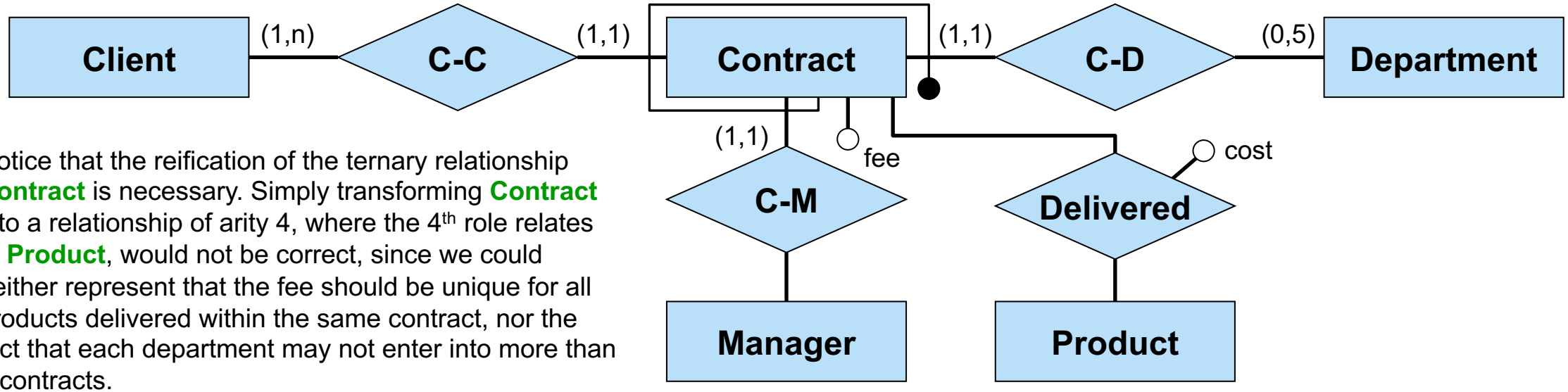


Problem 4: Solution

In order to capture the new requirements, we need to represent a relationship between contracts and products, in such a way that to each pair (*c*,*p*) of this new relationship we can associate the extra cost charged for product *p* under contract *c*. Starting from the original ER schema, we proceed as follows:

1. We reify the relationship **Contract**, i.e., we transform it into an entity, whose instances are identified by the three roles of binary relationships to **Client**, **Manager**, and **Department**, respectively.
2. We introduce an entity **Product** and a relationship, called **Delivered**, between the entity **Contract** and this new entity **Product**, with an attribute that models the extra cost. In this way, we can associate the extra cost to each pair (*c*,*p*) in the relationship **Delivered**, where *c* is a contract and *p* is a product.
3. We set to 5 the maximum cardinality of the **Department** role in the new relationship relating **Department** to **Contract**.

The resulting ER schema is as follows:



Notice that the reification of the ternary relationship **Contract** is necessary. Simply transforming **Contract** into a relationship of arity 4, where the 4th role relates to **Product**, would not be correct, since we could neither represent that the fee should be unique for all products delivered within the same contract, nor the fact that each department may not enter into more than 5 contracts.