Introduction to Databases

Exam of 29/09/2023

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/

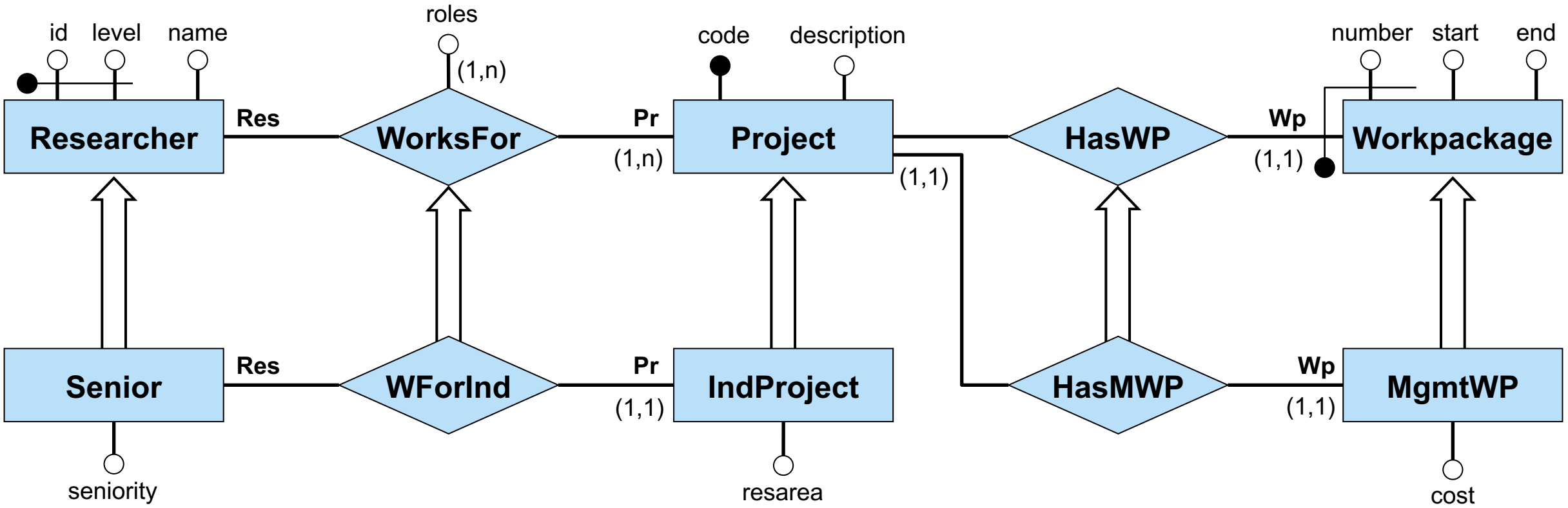
Problem 1

Design the Entity-Relationship schema of an application related to the management of research projects. Of each **researcher**, we are interested in the id and level (that together identify the researcher), the name, and the projects for which they work, with the roles (at least one) they hold in each project (e.g., responsible for publicity, budget responsible, etc.). Some researchers are **seniors** and for them we are also interested in their seniority. Of each **project**, we are interested in the code (identifier), the description (a string), and the researchers (at least one) that work for it. Some projects are **individual projects** and for them we are also interested in the research area. Note that only a single researcher participates to an individual projects, and such researcher must be senior. For each project, we are also interested in the **workpackages** that belong to it. Of each workpackage, we are interested in its number, which is unique within the project to which the workpackage belongs, and in its start and end date. In the case where the workpackage is a **management workpackage**, we are also interested in its cost. Note that each project has exactly one management workpackage, and the start and end dates of all workpackages of a project must fall within the start and end dates of the management workpackage of that project.

Problem 1: Conceptual schema

For each instance *I* of the schema:

1. If *w* ∈ *instances*(*I*,**Workpackage**), (*w*,*sw*) ∈ *instances*(*I*,**start**), and (*w*,*ew*) ∈ *instances*(*I*,**end**), then *sw* ≤ *ew*.
2. If (Project:*p*,Wp:*w*) ∈ *instances*(*I*,**HasWP**), (Project:*p*,Wp:*m*) ∈ *instances*(*I*,**HasMWP**), (*w*,*sw*) ∈ *instances*(*I*,**start**), (*w*,*ew*) ∈ *instances*(*I*,**end**), (*m*,*sm*) ∈ *instances*(*I*,**start**), and (*m*,*em*) ∈ *instances*(*I*,**end**), then *sm* ≤ *sw* and *ew* ≤ *em*.
3. If *ip* ∈ *instances*(*I*,**IndProject**) and (Res:*r*,Pr:*ip*) ∈ *instances*(*I*,**WorksFor**), then (Res:*r*,Pr:*ip*) ∈ *instances*(*I*,**WForInd**).



Problem 2

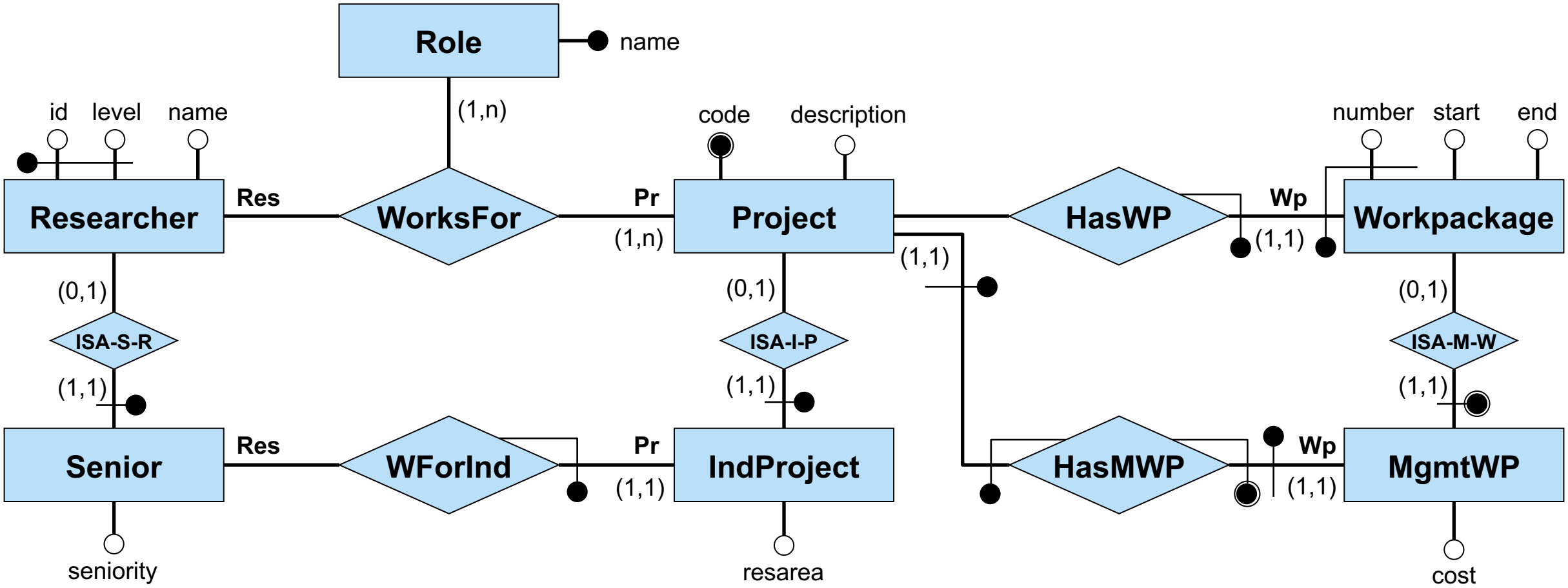
Carry out the logical design of the database, producing the complete relational schema with constraints, taking into account taking into account that when we access the information about a project, we always want to know whether it is an individual project, and if it is so, we always want to know the research area and the senior researcher that works for the project.

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 $w \in instances(I,\textbf{Workpackage})$, $(w,sw) \in instances(I,\textbf{start})$, and $(w,ew) \in instances(I,\textbf{end})$, then $sw \leq ew$.
2. If $(\text{Project}:p,\text{Wp}:w) \in instances(I,\textbf{HasWP})$, $(\text{Project}:p,\text{Wp}:m) \in instances(I,\textbf{HasMWP})$, $(w,sw) \in instances(I,\textbf{start})$, $(w,ew) \in instances(I,\textbf{end})$, $(m,sm) \in instances(I,\textbf{start})$, and $(m,em) \in instances(I,\textbf{end})$, then $sm \leq sw$ and $ew \leq em$.
3. Constraint resulting from the elimination of ISA between relationships **WForInd** and **WorksFor**, and also capturing constraint 3 of the original schema:
   If $(\text{Res}:s,\text{Pr}:ip) \in instances(I,\textbf{WForInd})$ and p is the (unique) instance of **Project** such that $(\text{IndProject}:ip,\text{Project}:p) \in instances(I,\textbf{ISA-I-P})$, then there is a unique instance r of **Researcher** such that $(\text{Res}:r,\text{Role}:ro,\text{Pr}:p) \in instances(I,\textbf{WorksFor})$ for some instance ro of **Role**, and moreover $(\text{Senior}:s,\text{Researcher}:r) \in instances(I,\textbf{ISA-S-R})$.
4. Constraint resulting from the elimination of ISA between relationships **HasMWP** and **HasWP**:
   If $(\text{Project}:p,\text{Wp}:m) \in instances(I,\textbf{HasMWP})$ and w is the (unique) instance of **Workpackage** such that $(\text{MgmtWP}:m,\text{Workpackage}:w) \in instances(I,\textbf{ISA-M-W})$, then $(\text{Project}:p,\text{Wp}:w) \in instances(I,\textbf{HasWP})$.

Problem 2: Direct translation (1/2)

Researcher(id, level, name)

Senior(id, level, seniority)
  foreign key: Senior[id,level] $\subseteq$ Researcher[id,level]

Project(code, description)
  inclusion: Project[code] $\subseteq$ WorksFor[project]
  foreign key: Project[code] $\subseteq$ HasMWP[project]

IndProject(code, resarea)
  foreign key: IndProject[code] $\subseteq$ Project[code]
  foreign key: IndProject[code] $\subseteq$ WForInd[project]

Role(name)
  inclusion: Role[name] $\subseteq$ WorksFor[role]

WorksFor(resid, reslevel, role, project)
  foreign key: WorksFor[resid,reslevel] $\subseteq$ Researcher[id,level]
  foreign key: WorksFor[role] $\subseteq$ Role[name]
  foreign key: WorksFor[project] $\subseteq$ Project[code]

Problem 2: Direct translation (2/2)

WForInd(resid, reslevel, project)
  foreign key: WForInd[resid,reslevel] $\subseteq$ Senior[id,level]
  foreign key: WForInd[project] $\subseteq$ IndProject[code]
  foreign key: WForInd[resid,reslevel,project] $\subseteq$ WorksFor[resid,reslevel,project]

Workpackage(number, project, start, end)
  foreign key: Workpackage[project] $\subseteq$ Project[code]
  constraint: start $\leq$ end

MgmtWP(number, project, cost)
  foreign key: MgmtWP[number,project] $\subseteq$ Workpackage[number,project]
  foreign key: MgmtWP[project] $\subseteq$ HasMWP[project]

HasMWP(project, project2)
  foreign key: HasMWP[project] $\subseteq$ MgmtWP[project]
  foreign key: HasMWP[project2] $\subseteq$ Project[code]
  constraint: project = project2

Note that the fact that the primary key of **MgmtWP** is just the project attribute is a consequence of the fact that each project has a unique management workpackage.

Also, the constraint project = project2 holding for **HasMWP** is a consequence of the constraint deriving from the inclusion between relationships **HasMWP** in **HasWP** in the original ER schema and the fact that project is (part of) the primary key of **MgmtWP** and of **Workpackage**.

Problem 2: Direct translation – External constraints

1. Has been specified as tuple constraint of the **Workpackage** relation.
2. CHECK constraints on relation **Workpackage**:
   - ```
     start >= (SELECT WP.start FROM Workpackage WP, MgmtWP M
               WHERE WP.number = M.number AND WP.project = M.project)
     ```
   - ```
     end <= (SELECT WP.end FROM Workpackage WP, MgmtWP M
             WHERE WP.number = M.number AND WP.project = M.project)
     ```
3. The constraint resulting from the elimination of ISA between relationships **WForInd** and **WorksFor** has already been expressed as a foreign key on **WForInd**. The additional constraint 3 of the original conceptual schema can be expressed through a CHECK constraint:

   ```
   CHECK (SELECT WF.resid, WF.reslevel, WF.project
          FROM WorksFor WF, IndProject IP WHERE WF.project = IP.code)
         IN (SELECT resid, reslevel, project FROM WForInd)
   ```
4. The constraint resulting from the elimination of ISA between relationships **HasMWP** and **HasWP** has already been expressed through the foreign key between **MgmtWP** and **Workpackage**, together with the fact that the project code is part of the primary key of **Workpackage**.

Problem 2: Restructuring of the relational schema

When we access the information about a project, we always want to know whether it is an individual project, and if it is so, we always want to know the research area and the senior researcher that works for the project.

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

- We merge relation **IndProject** into **Project**, making the attribute **resarea** nullable, and using it as a flag to distinguish the instances of **IndProject**.
- We also merge relation **WForInd** into **Project**, making the two attributes **resid** and **reslevel** nullable, and constraining them to be null exactly when attribute **resarea** is null.

Additionally, we remove the redundant attribute **project2** from **HasMWP**, at which point the relation **HasMWP**, becomes redundant, since the information about the project of an instance of **MgmtWP** is already contained in **MgmtWP**.

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 IndProject, WForInd, and HasMWP have been removed.

Project(code, description, resarea*, resid*, reslevel*)
- inclusion: Project[code] $\subseteq$ WorksFor[project]
- foreign key: Project[code] $\subseteq$ HasMWP[project]
- foreign key: Project[resid,reslevel] $\subseteq$ Senior[id,level]
- foreign key: Project[resid,reslevel,code] $\subseteq$ WorksFor[resid,reslevel,project]

The constraints 1, 2, and 4 stay unchanged, while the additional constraint 3 of the original conceptual schema can now be expressed through a CHECK constraint:

```
CHECK (SELECT WF.resid, WF.reslevel, WF.project
       FROM WorksFor WF, Project P
       WHERE WF.project = P.code AND IS NOT NULL P.resarea)
      IN (SELECT resid, reslevel, code FROM Project)
```

Problem 3

Consider a database containing the tables `Worker(ssn,age,department)`, where null values are allowed for the attributes `age` and `department`, and `Purchase(ssn,item,year)`, where null values are not allowed. The first table stores the ssn (primary key), age, and department of a set of workers. Each tuple in the second table represents a purchase made by a worker for an item in a certain year. Note that there is nothing that prevents a purchase *p* from having a "twin", i.e., a purchase done by the same worker of the same item in the same year as those of *p*. We know that the database satisfies the foreign key constraint from `Purchase[ssn]` to `Worker[ssn]`.

1. Express **in SQL** a query that, calculates how many purchases in the `Purchase` table have no twins.
2. Express **in relational algebra** a query that, computes the ssn of workers who have done at least one purchase from 2020 onwards, and for whom if the department is not known, the age is known.

Problem 3: Solution (1/2)

```
Worker(ssn,age,department)
Purchase(ssn,item,year)
```

1. Express **in SQL** a query that, calculates how many purchases in the `Purchase` table have no twins.

```
SELECT COUNT(*)
FROM (SELECT ssn, item, year
      FROM Purchase
      GROUP BY ssn, item, year
      HAVING COUNT(*) = 1)
```

Problem 3: Solution (2/2)

```
Worker(ssn,age,department)
Purchase(ssn,item,year)
```

2. Express **in relational algebra** a query that, computes the ssn of workers who have done at least one purchase after 2020 and for whom if the department is not known, the age is known.

$$\textbf{PROJ}_{\text{ssn}}\ ((\textbf{SEL}_{\text{(department }\textbf{IS NOT NULL}\text{) }\textbf{OR}\text{ (age }\textbf{IS NOT NULL}\text{)}}\text{Worker})\ \textbf{JOIN}\ (\textbf{SEL}_{\text{year >= 2020}}\ \text{Purchase}))$$

Problem 4

Consider the relational schema *S* consisting of two relations `P(X,Y)` and `Q(Z,W)`, where:

1. `P` has `X` as primary key and `Y` as additional key;
2. the default value for attribute `Y` of `P` is 1;
3. `Q` has `Z` as primary key;
4. `P` and `Q` have mutual foreign keys with delete policies as follows:

```
P(X,Y)  UNIQUE Y, DEFAULT Y=1,
        FOREIGN KEY P[Y] ⊆ Q[Z] ON DELETE SET DEFAULT
Q(Z,W)  FOREIGN KEY Q[W] ⊆ P[X] ON DELETE CASCADE
```

Answer the following question: Is there a database *D* consistent with *S* such that the deletion from the database *D* of a tuple *t* of `P` causes an error?

If the answer is negative, motivate your answer.

If the answer is positive, show such a database *D* and the corresponding tuple *t* and describe the reason for the error.

Problem 4: Solution

P

X	Y
7	4
2	5
3	1

Q

Z	W
4	2
5	7
1	2

Consider the above database *D*, and assume that we delete tuple (7,4) from `P`. Then, the `DELETE CASCADE` policy for `Q[W]` $\subseteq$ `P[X]` (and since there can be no other tuple *t* in `P` such that *t* [`X`] = 7) forces to delete tuple (5,7) from `Q`.
Then, the `DELETE SET DEFAULT` policy for `P[Y]` $\subseteq$ `Q[Z]` forces to set to 1 (which is the default value for `P[Y]`) the value 5 in tuple (2,5).

In this way, we obtain the two tuples (2,1) and (3,1) in `P`, which causes a violation of the key constraint `UNIQUE Y` of `P`.

Variant of Problem 4 with Solution

Consider now a variant of Problem 4 where Condition 2 now states that the default value for attribute `Y` (instead of `X`) of `P` is 1. Then the solution of the problem is as follows.

Consider an arbitrary database D consistent with S and a tuple t of `P` that is deleted.

- If there is no tuple s in `Q` such that s[`W`] = t[`X`], there is no further change to make.
- If instead there are tuples $s_1,...,s_n$ in `Q` such that s_1[`W`] = ... = s_n[`W`] = t[`X`], then the `DELETE CASCADE` policy for `Q[W]⊆P[X]` (and the fact that `X` is a key of `P`, hence there can be no other tuple t' in `P` such that t' [`X`] = t [`X`]) forces to delete $s_1,...,s_n$ from `Q`.
- If for every i with $1 \le i \le n$, there is no tuple t' in `P` such that t' [`Y`] = s_i [`Z`], there is no further change to make.
- If instead all the tuples t_j in `P` such that t_j [`Y`] = s_i [`Z`], for some i with $1 \le i \le n$, are $t_1,...,t_m$, then the `DELETE SET DEFAULT` policy for `P[Y]⊆Q[Z]` (and the fact that `Z` is a key of `Q`) forces to set all of t_1[`Y`],..., t_m[`Y`] to `NULL` (which is the default value for `P[Y]`). None of the constraints is violated by this (in particular the key constraint on `P[Y]` is not violated).
- No further change needs to be made, and the resulting database is consistent with S.

Hence, there is no database D consistent with S such that the deletion of a tuple t of `P` causes an error.