Introduction to Databases

Exam of 14/09/2021

With Solutions

Diego Calvanese

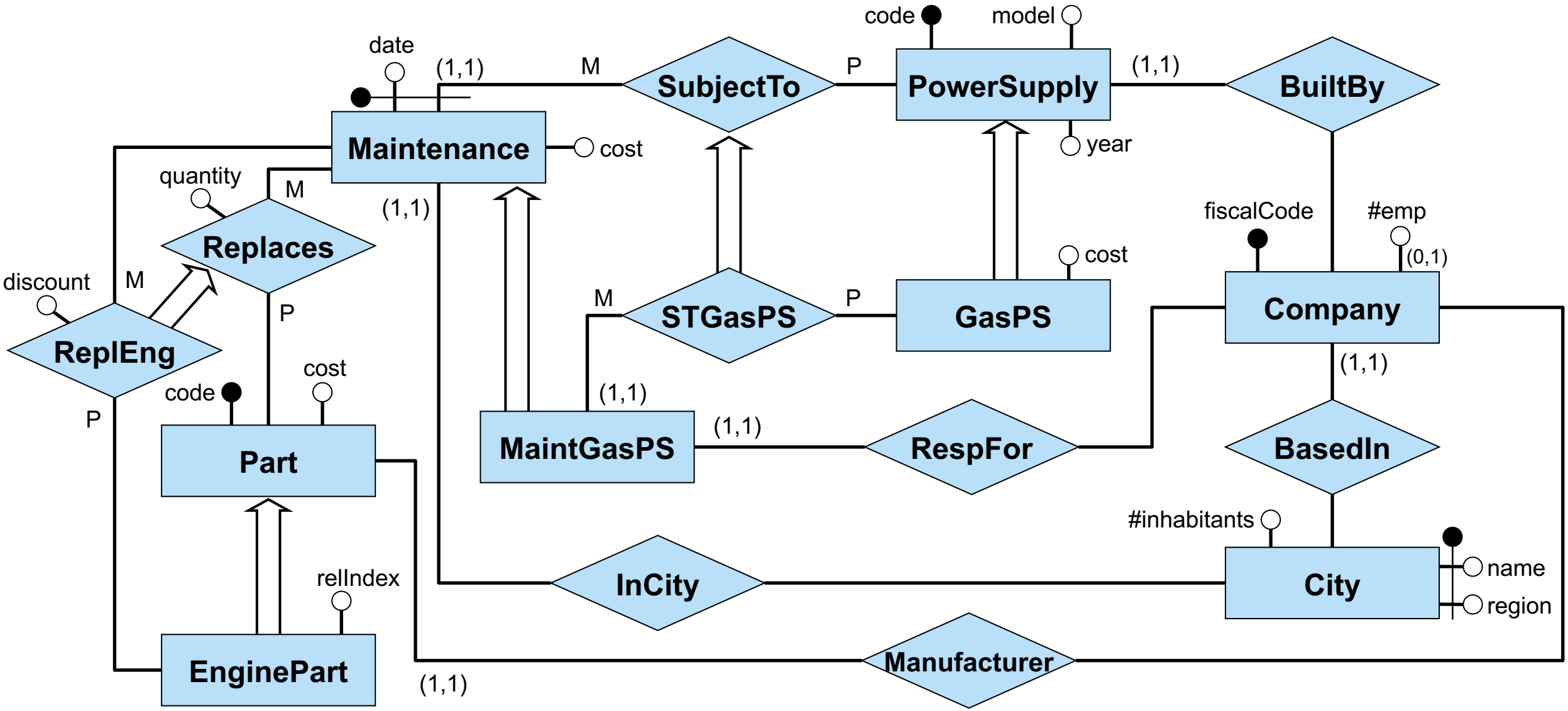
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema of an application relating to maintenance operations on movable power supplies. For each **power supply**, the code (identifier), the model, the company that built it, and the year of construction are of interest. **Gas power supplies** are a special kind of power supplies, the cost of which is also of interest. For each **company**, the fiscal code (identifier), the city where it is based, and the number of employees are of interest (this last information may not be available). For each **city**, the name (unique within the region), the region, and the number of inhabitants are of interest. Power supplies are subject to maintenance: for each **maintenance operation** we are interested in the power supply to which it is applied, the date, the labour cost, and the city where the operation took place. A power supply cannot be subject to more than one maintenance operation per day. For a **maintenance operation that concerns a gas power supply**, we are interested also in the company responsible for the maintenance operation. A maintenance operation may require the replacement of parts. For each **part**, the code (identifier), the unit cost, and the manufacturing company are of interest. If the part is an **engine part**, it is also important to know the reliability index (positive integer). Finally, for each part to be replaced in a maintenance operation, it is of interest to know the quantity that has been replaced (for example, 3 valves may be replaced in a maintenance operation of a power supply), and in the case of an engine part, the discount applied on the unit price.

Problem 1: Conceptual schema – Diagram



Problem 1: Conceptual schema – External constraints

1) For each instance I of the schema, for each (M:m,P:p) $\in$ *instances*(I,**SubjectTo**), if $p \in$ *instances*(I,**GasPS**), then $m \in$ *instances*(I,**MaintGasPS**) (and hence (M:m,P:p) $\in$ *instances*(I,**STGasPS**)).
2) For each instance I of the schema, for each (M:m,P:p) $\in$ *instances*(I,**Replaces**), if $p \in$ *instances*(I,**EnginePart**), then (M:m,P:p) $\in$ *instances*(I,**RepEng**).

Problem 2

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

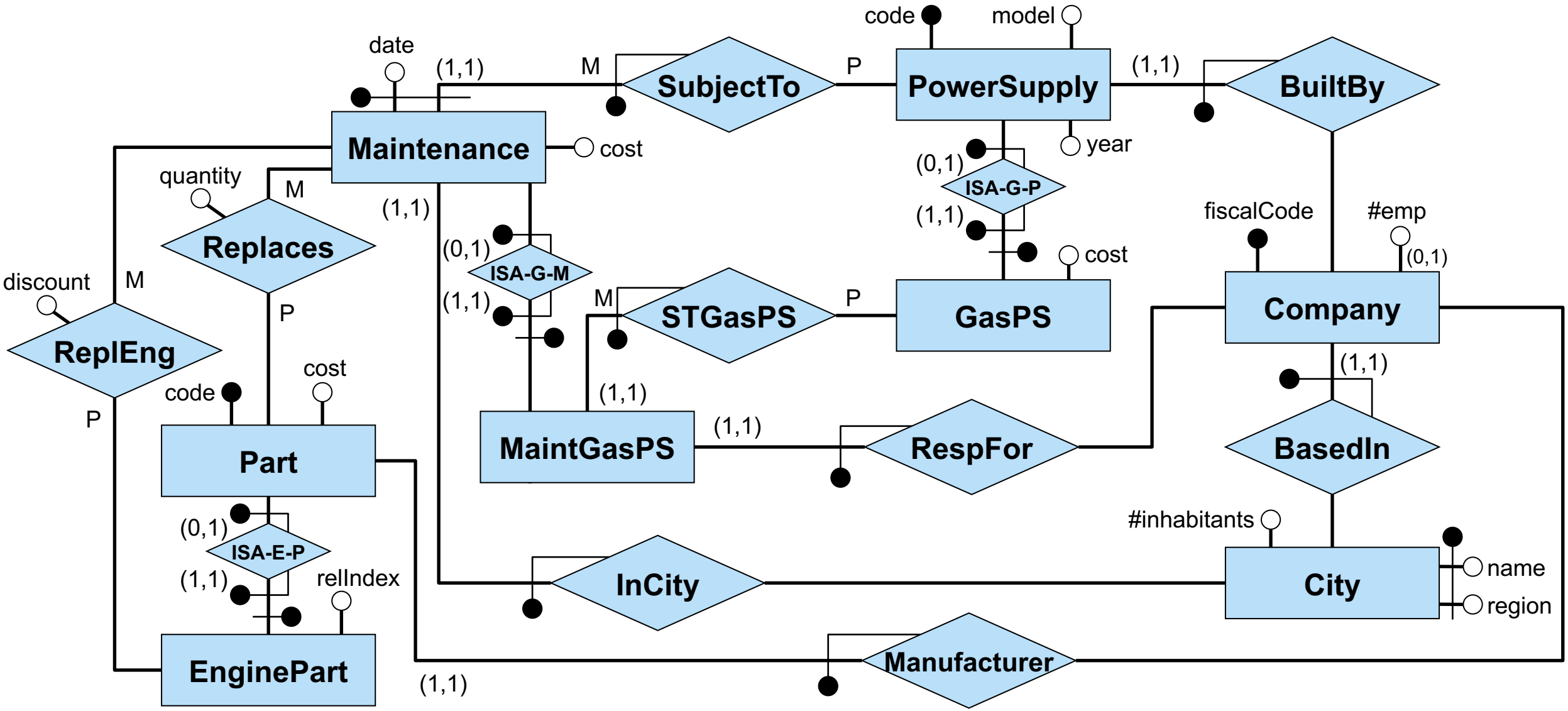
1. When accessing data about a maintenance operation, we are always interested in knowing the city where it took place and whether it is a maintenance operation concerning a gas power supply, and in this case also the company responsible for it.
2. When accessing data about a power supply, we are always interested in knowing the company that built it.

As steps in your design 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).

Motivate explicitly how the above indications affect your design.

Problem 2: Restructured conceptual schema – Diagram



Problem 2: Restructured conceptual schema – External constraints

1) For each instance *I* of the schema, for each (M:*m*,P:*p*) ∈ *instances*(*I*,**SubjectTo**), if there is a *g* ∈ *instances*(*I*,**GasPS**) such that (GasPS:*g*,PowerSupply:*p*) ∈ *instances*(*I*,**ISA-G-P**), then there is a (unique) *mg* ∈ *instances*(*I*,**MaintGasPS**) such that (MaintGasPS:*mg*,Maintenance:*m*) ∈ *instances*(*I*,**ISA-G-M**) and (M:*mg*,P:*g)* ∈ *instances*(*I*,**STGasPS**).

2) For each instance *I* of the schema, for each (M:*m*,P:*p*) ∈ *instances*(*I*,**Replaces**), if there is an *e* ∈ *instances*(*I*,**EnginePart**) such that (EnginePart:*e*,Part:*p*) ∈ *instances*(*I*,**ISA-E-P**), then (M:*m*,P:*e*) ∈ *instances*(*I*,**RepEng**).

3) Constraint resulting from the elimination of ISA between the relationships **STGasPS** and **SubjectTo**: For each instance *I* of the schema, if (M:*mg*,P:*g)* ∈ *instances*(*I*,**STGasPS**), *m* is the unique element of *instances*(*I*,**Maintenance**) such that (MaintGasPS:*mg*,Maintenance:*m*) ∈ *instances*(*I*,**ISA-G-M**), and *p* is the unique element of *instances*(*I*,**PowerSupply**) such that (GasPS:*g*,PowerSupply:*p*) ∈ *instances*(*I*,**ISA-G-P**), then (M:*m*,P:*p*) ∈ *instances*(*I*,**SubjectTo**).

4) Constraint resulting from the elimination of ISA between the relationships **ReplEng** and **Replaces**: For each instance *I* of the schema, if (M:*m*,P:*e)* ∈ *instances*(*I*,**ReplEng**) and *p* is the unique element of *instances*(*I*,**Part**) such that (EnginePart:*e*,Part:*p*) ∈ *instances*(*I*,**ISA-E-P**), then (M:*m*,P:*p*) ∈ *instances*(*I*,**Replaces**).

Problem 2: Direct translation (1/2)

PowerSupply(code, model, year)
  foreign key: PowerSupply[code] $\subseteq$ BuiltBy[powerSupply]

Company(fiscalCode, #emp*)
  foreign key: Company[fiscalCode] $\subseteq$ BasedIn[company]

Maintenance(date, powerSupply, cost)
  foreign key: Maintenance[powerSupply] $\subseteq$ PowerSupply[code]
  foreign key: Maintenance[date,powerSupply] $\subseteq$ InCity[date,PowerSupply]

GasPS(code, cost)
  foreign key: GasPS[code] $\subseteq$ PowerSupply[code]

MaintGasPS(date, powerSupply)
  foreign key: MaintGasPS[date,powerSupply] $\subseteq$ Maintenance[date,powerSupply]
  foreign key: MaintGasPS[date,powerSupply] $\subseteq$ RespFor[date,powerSupply]
  foreign key: MaintGasPS[date,powerSupply] $\subseteq$ STGasPS[date,powerSupply]

City(name, region, #inhabitants)

Part(code, cost)
  foreign key: Part[code] $\subseteq$ Manufacturer[part]

EnginePart(code, relIndex)
  foreign key: EnginePart[code] $\subseteq$ Part[code]

Problem 2: Direct translation (2/2)

BuiltBy(powerSupply, company)
  foreign key: BuiltBy[powerSupply] $\subseteq$ PowerSupply[code]
  foreign key: BuiltBy[company] $\subseteq$ Company[fiscalCode]

BasedIn(company, city, region)
  foreign key: BasedIn[company] $\subseteq$ Company[fiscalCode]
  foreign key: BasedIn[city,region] $\subseteq$ City[name,region]

RespFor(date, powerSupply, company)
  foreign key: RespFor[date,powerSupply] $\subseteq$ MaintGasPS[date,powerSupply]
  foreign key: RespFor[company] $\subseteq$ Company[fiscalCode]

STGasPS(date, powerSupply, gasPS)
  foreign key: STGasPS[date,powerSupply] $\subseteq$ MaintGasPS[date,powerSupply]
  foreign key: STGasPS[gasPS] $\subseteq$ GasPS[code]

InCity(date, powerSupply, city, region)
  foreign key: InCity[date,powerSupply] $\subseteq$ Maintenance[date,powerSupply]
  foreign key: InCity[city,region] $\subseteq$ City[name,region]

Replaces(date, powerSupply, part, quantity)
  foreign key: Replaces[date,powerSupply] $\subseteq$ Maintenance[date,powerSupply]
  foreign key: Replaces[part] $\subseteq$ Part[code]

ReplEng(date, powerSupply, enginePart, discount)
  foreign key: ReplEng[date,powerSupply] $\subseteq$ Maintenance[date,powerSupply]
  foreign key: ReplEng[enginePart] $\subseteq$ EnginePart[code]
  foreign key: ReplEng[date,powerSupply,enginePart] $\subseteq$ Replaces[date,powerSupply,part]

Manufacturer(part, company)
  foreign key: Manufacturer[part] $\subseteq$ Part[code]
  foreign key: Manufacturer[company] $\subseteq$ Company[fiscalCode]

Problem 2: Direct translation – External constraints

1) $PROJ_{date,powerSupply}(Maintenance\ JOIN_{powerSupply=code}\ GasPS) \subseteq MaintGasPS$.

2) $PROJ_{date,powerSupply,part}(Replaces\ JOIN_{part=code}\ EnginePart) \subseteq PROJ_{date,powerSupply,enginePart}(ReplEng)$.

3) The constraint resulting from the elimination of ISA between the relationships **STGasPS** and **SubjectTo** is already implied by the foreign key between the relations **STGasPS** and **MaintGasPS** and the foreign key between the relations **MaintGasPS** and **Maintenance** (into which the relationship **SubjectTo** has been merged).

4) The constraint resulting from the elimination of ISA between the relationships **ReplEng** and **Replaces** has already been specified as foreign key between the corresponding relations **ReplEng** and **Replaces**.

Problem 2: Restructuring of the relational schema

1. When accessing data about a maintenance operation, we are always interested in knowing the city where it took place and whether it is a maintenance operation concerning a gas power supply, and in this case also the company responsible for it.
2. When accessing data about a power supply, we are always interested in knowing the company that built it.

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

- We take into account indication 1 by:
  1. Merging into the relation **MaintGasPS** the relation **RespFor**, from which and to which **MaintGasPS** has a foreign key, and then merging the resulting relation into **Maintenance**. Since there is a foreign key from **MaintGasPS** to **Maintenance** but not vice-versa, the latter merging operation might introduce NULL values in **Maintenance**.
  2. Merging into the relation **Maintenance** the relation **InCity**, from which and to which **Maintenance** has a foreign key.
- We take into account indication 2 by merging into the relation **PowerSupply** the relation **BuiltBy**, from which and to which **PowerSupply** has a foreign key.

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.

Maintenance(date, powerSupply, cost, company*, city, region)
- foreign key: Maintenance[powerSupply] $\subseteq$ PowerSupply[code]
- foreign key: Maintenance[company] $\subseteq$ Company[fiscalCode]
- foreign key: Maintenance[date,powerSupply] $\subseteq$ InCity[date,PowerSupply]
- foreign key: Maintenance[city,region] $\subseteq$ City[name,region]

STGasPS(date, powerSupply, gasPS)
- foreign key: STGasPS[date,powerSupply] $\subseteq$ Maintenance[date,powerSupply]
- foreign key: STGasPS[gasPS] $\subseteq$ GasPS[code]

PowerSupply(code, model, year, company)
- foreign key: PowerSupply[company] $\subseteq$ Company[fiscalCode]

Moreover, we remove the relations **MaintGasPS**, **RespFor**, and **BuiltBy**.

Problem 2: Restructuring of the relational schema – External constraints

1) $SEL_{company\ IS\ NULL}(Maintenance\ JOIN_{powerSupply=code}\ GasPS) = \emptyset$.
2) This constraint is not affected by the restructuring.
3) The constraint resulting from the elimination of ISA between the relationships **STGasPS** and **SubjectTo** is already specified as foreign key between the relations **STGasPS** and **Maintenance** (into which the relationship **SubjectTo** has been merged).
4) This constraint is not affected by the restructuring and stays specified as foreign key between the relations **ReplEng** and **Replaces**.
5) The foreign keys $MaintGasPS[date,powerSupply] \subseteq STGasPS[date,powerSupply]$ and $STGasPS[date,powerSupply] \subseteq MaintGasPS[date,powerSupply]$ become the following external constraint:

$$PROJ_{powerSupply}(SEL_{company\ IS\ NOT\ NULL}(Maintenance)) \subseteq PROJ_{code}(GasPS)$$
$$PROJ_{code}(GasPS) \subseteq PROJ_{powerSupply}(SEL_{company\ IS\ NOT\ NULL}(Maintenance)).$$

Problem 3

Consider a database that includes the relations `Graduated` and `School`. The relation `Graduated(studId,grade,schoolCode)` stores for each graduated student, the student id, the grade obtained at the graduation, and the school where they obtained it. The relation `School(schoolCode,city)` stores for each school the city.

1. We call “exit indicator” of a school the average of the grades obtained by the graduates in that school, and “average exit indicator” the average of the exit indicators of all the schools. Then we call “school quality indicator” of a city the percentage of schools that are located in that city and that have the exit indicator higher than the average exit indicator. Write a *SQL query* that returns for each city its school quality indicator.
2. We call a city a “top city” if all its schools have at least one graduate with a grade of 100. Write a query in *relational algebra* that returns the top cities.

Problem 3: Solution (1/2)

```
Graduated(studId,grade,schoolCode)
School(schoolCode,city)
```

1. We call “exit indicator” of a school the average of the grades obtained by the graduates in that school, and “average exit indicator” the average of the exit indicators of all the schools. Then we call “school quality indicator” of a city the percentage of schools that are located in that city and that have the exit indicator higher than the average exit indicator. Write a *SQL query* that returns for each city its school quality indicator.

```
WITH ViewEI AS
      (SELECT S.schoolCode, S.city, AVG(G.grade) AS ei
       FROM School S JOIN Graduated G ON S.schoolCode = G.schoolCode
       GROUP BY S.schoolCode, S.city)
SELECT V1.city, (COUNT(DISTINCT V1.schoolCode)::NUMERIC /
                 COUNT(DISTINCT V2.schoolCode)) * 100 AS sqi
FROM ViewEI V1 JOIN ViewEI V2 ON V1.city = V2.city
WHERE V1.ei > (SELECT AVG(ei) FROM ViewEI)
GROUP BY V1.city
```

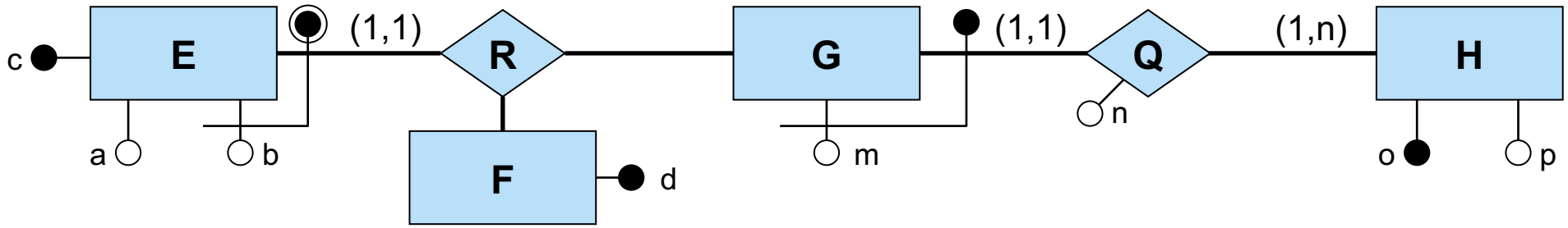
Problem 3: Solution (2/2)

```
Graduated(studId,grade,schoolCode)
School(schoolCode,city)
```

2. We call a city a “top city” if all its schools have at least one graduate with a grade of 100. Write a query in *relational algebra* that returns the top cities.

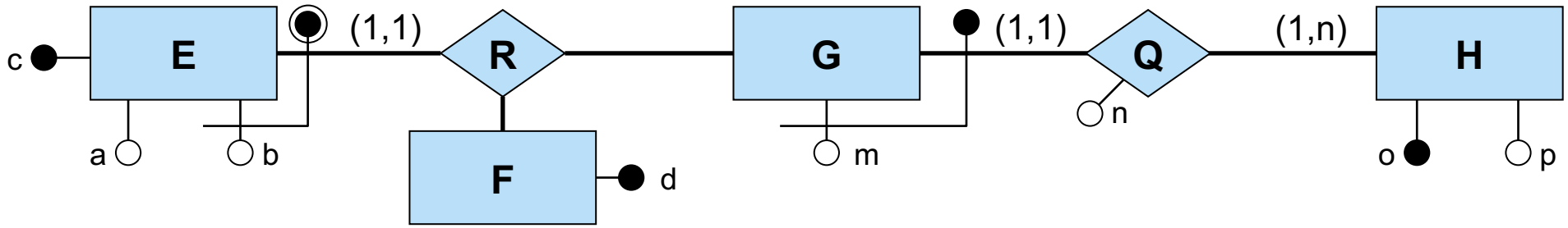
$$\text{PROJ}_{\text{city}}(\text{School}) - \text{PROJ}_{\text{city}}(\text{School JOIN } (\text{PROJ}_{\text{schoolCode}}(\text{School}) - \text{PROJ}_{\text{schoolCode}}(\text{SEL}_{\text{grade = 100}}(\text{Graduated}))))$$

Problem 4



Consider the restructured conceptual schema shown above, and carry out the direct translation step, illustrating the relational schema complete with constraints resulting from that step.

Problem 4: Solution



E(a, b, c, f, g, h)
  foreign key: E[f] ⊆ F[d]
  foreign key: E[g,h] ⊆ G[m,h]
  key: c

F(d)

G(m, n, h)
  foreign key: G[h] ⊆ H[o]

H(o, p)
  inclusion: H[o] ⊆ G[h]

Notes:

- Since the primary identifier of entity **E** involves a role of relationship **R**, **R** gets merged into relation **E**.
- Since the primary identifier of entity **G** involves a role of relationship **Q**, **Q** gets merged into relation **G**.
- We have chosen as names for the attributes that refer to external entities, the name of the entity rather than the name of the attribute (when such attribute is part of the primay identifier of the external entity).