

Introduction to Databases

Exam of 22/09/2026

With Solutions

Diego Calvanese

Bachelor in Computer Science

Faculty of Engineering

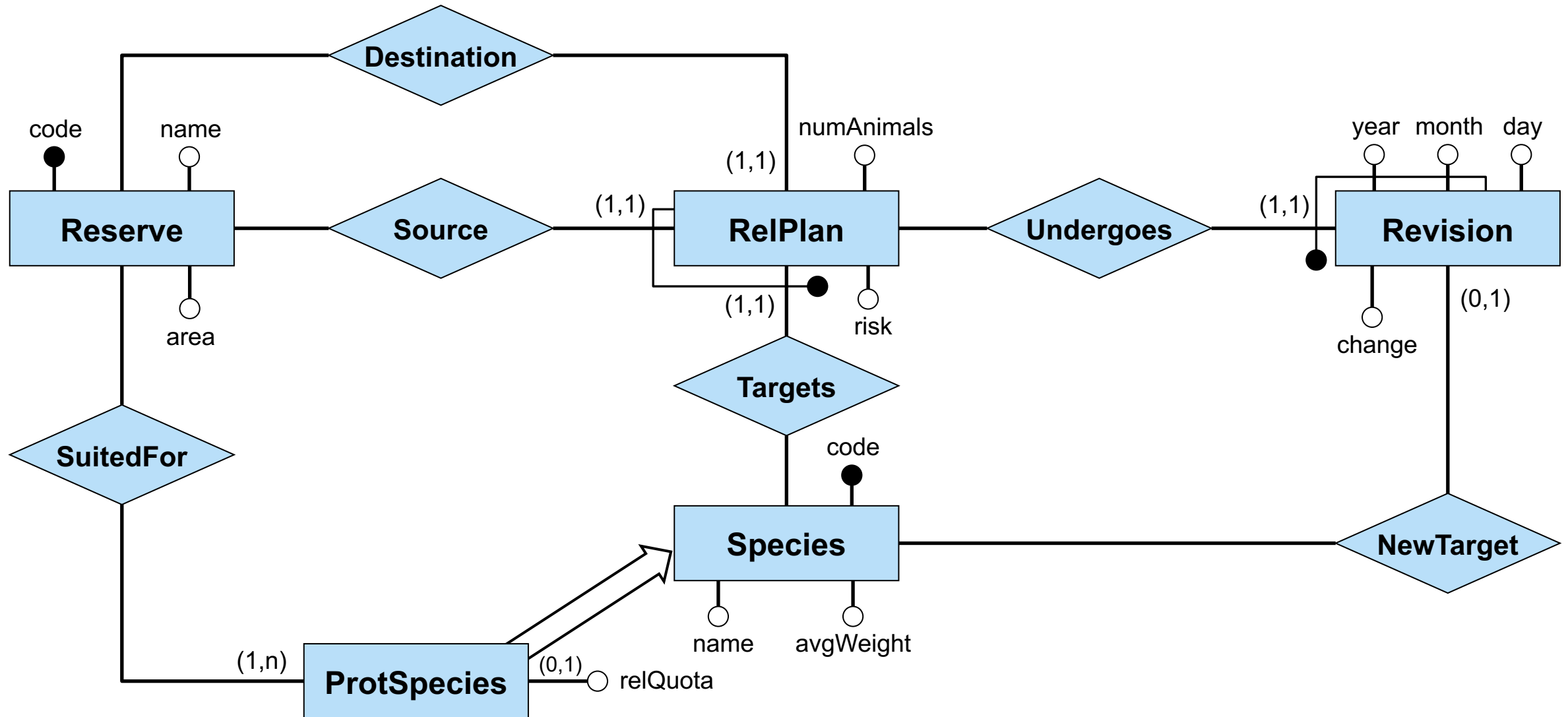
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema for an application that concerns wildlife relocation plans among a set of nature reserves, according to the following requirements. For each **nature reserve**, we are interested in its reserve code (identifier), its name, its area, and the relocation plans in which it is involved. Each **relocation plan** concerns a species (called the target species of the plan), is established between a source reserve and a destination reserve, and has an associated planned number of animals and a risk level. Notice that a species may be the target species of any number of relocation plans (possibly none), and that each nature reserve may not be the source of more than one relocation plan having the same target species. A relocation plan may undergo **revisions** (at most one per month might be approved) in order to record a change in the planned number of animals and possibly to change the species that is the target of the relocation. For each revision of a relocation plan, we are interested in both of these pieces of information, namely, the change in the planned number of animals (which must be positive) and the possible new target species, and also in the date (i.e., day, month, and year) in which the revision is approved. For each **species**, identified by a code, we are interested in its name and average weight. **Protected species** are particular kinds of species that may have an associated recommended relocation quota, and for which we are interested in the reserves that are suited for them (at least one). Notice that a reserve may be suited for any number of protected species (possibly none).

Problem 1: Conceptual schema



Problem 1: External constraints

- 1) Domain constraint: **change of Revision must be a positive integer.**
- 2) The source and destination reserve of a relocation plan must be different.
Formally: For each instance I of the ER schema and for each $rp \in \text{instances}(I, \text{RelPlan})$, if there are sr and dr such that $\langle \text{RelPlan}:rp, \text{Reserve}:sr \rangle \in \text{instances}(I, \text{Source})$ and $\langle \text{RelPlan}:rp, \text{Reserve}:dr \rangle \in \text{instances}(I, \text{Destination})$, then $sr \neq dr$.
- 3) The species that is the new target of a revision must be different from the original target species.
Formally: For each instance I of the ER schema and for each $r \in \text{instances}(I, \text{Revision})$, if there are rp , s , and ns such that $\langle \text{RelPlan}:rp, \text{Revision}:r \rangle \in \text{instances}(I, \text{Undergoes})$, $\langle \text{RelPlan}:rp, \text{Species}:s \rangle \in \text{instances}(I, \text{Targets})$, and $\langle \text{Revision}:r, \text{Species}:ns \rangle \in \text{instances}(I, \text{NewTarget})$, then $ns \neq s$.

Problem 2

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

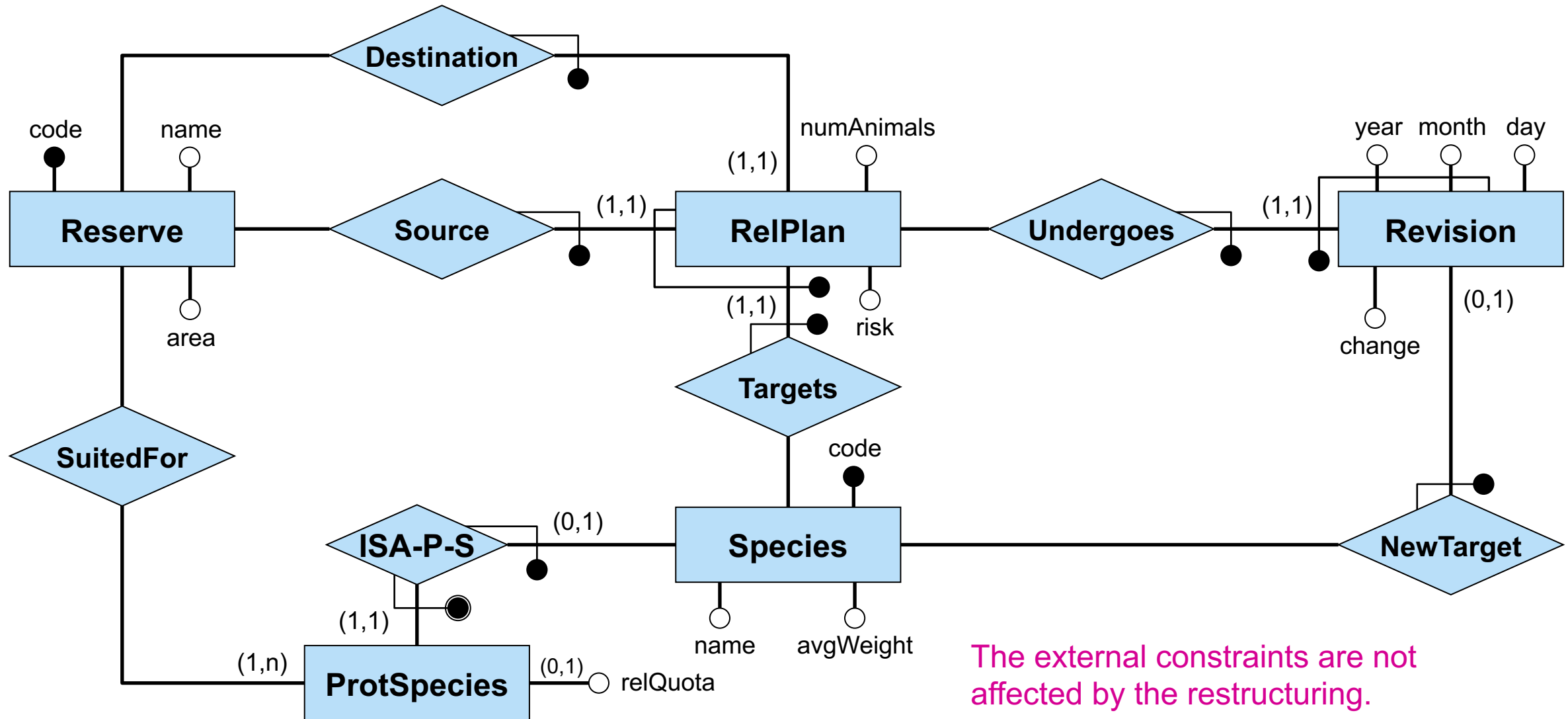
1. Whenever a relocation plan is accessed, we always want to know the source and destination reserves.
2. Whenever a revision of a relocation plan is accessed, we always want to know the possible new target species established by that revision.
3. Whenever a species is accessed, we always want to know whether it is a protected species or not and, if it is, its possible recommended relocation quota.

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: Direct translation (1/2)

Reserve(code, name, area)

Species(code, name, avgWeight)

ProtSpecies(code, relQuota*)

foreign key: ProtSpecies[code] $\subseteq$ Species[code]

inclusion: ProtSpecies[code] $\subseteq$ SuitedFor[protSpecies]

SuitedFor(reserve, protSpecies)

foreign key: SuitedFor[reserve] $\subseteq$ Reserve[code]

foreign key: SuitedFor[protSpecies] $\subseteq$ protSpecies[code]

RelPlan(reserve, species, numAnimals, risk)

foreign key: RelPlan[reserve] $\subseteq$ Reserve[code]

foreign key: RelPlan[species] $\subseteq$ Species[code]

foreign key: RelPlan[reserve,species] $\subseteq$ Destination[rpReserve,rpSpecies]

Problem 2: Direct translation (2/2)

Destination(rpReserve, rpSpecies, destReserve)

foreign key: Destination[rpReserve,rpSpecies] $\subseteq$ RelPlan[reserve,species]

foreign key: Destination[destReserve] $\subseteq$ Reserve[code]

tuple constraint: rpReserve $\neq$ destReserve (realizes external constraint 2)

Revision(reserve, species, year, month, day, change)

foreign key: Revision[reserve,species] $\subseteq$ RelPlan[reserve,species]

domain constraint: change > 0 (realizes external constraint 1)

NewTarget(reserve, species, year, month, newSpecies)

foreign key: NewTarget[reserve,species,year,month] $\subseteq$
Revision[reserve,species,year,month]

foreign key: NewTarget[newSpecies] $\subseteq$ Species[code]

tuple constraint: species $\neq$ newSpecies (realizes external constraint 3)

Problem 2: Restructuring of the relational schema

1. Whenever a relocation plan is accessed, we always want to know the source and destination reserves.

This induces merging **Destination** and **RelPlan**, which are strongly coupled.

2. Whenever a revision of a relocation plan is accessed, we always want to know the possible new target species established by that revision.

This induces merging **NewTarget** and **Revision**, which are weakly coupled.

3. Whenever a species is accessed, we always want to know whether it is a protected species or not and, if it is, its possible recommended relocation quota.

This induces merging **ProtSpecies** and **Species**, which are weakly coupled.

The restructured relational schema is shown on the next slide.

We show also the views that reconstruct the merged relations.

Problem 2: Restructured relational schema (1/2)

Reserve(code, name, area)

Species(code, name, avgWeight, flagProtected, relQuota*)

tuple constraint: flagProtected = 'true' OR relQuota IS NULL

constraint: $\text{PROJECT}_{\text{code}}(\text{SELECT}_{\text{flagProtected} = \text{'true'}}(\text{Species})) \subseteq \text{PROJECT}_{\text{protSpecies}}(\text{SuitedFor})$

SuitedFor(reserve, protSpecies)

foreign key: SuitedFor[reserve] $\subseteq$ Reserve[code]

foreign key: SuitedFor[protSpecies] $\subseteq$ protSpecies[code]

RelPlan(reserve, species, numAnimals, risk, destReserve)

foreign key: RelPlan[reserve] $\subseteq$ Reserve[code]

foreign key: RelPlan[species] $\subseteq$ Species[code]

foreign key: RelPlan[destReserve] $\subseteq$ Reserve[code]

tuple constraint: reserve $\neq$ destReserve (realizes external constraint 2)

Problem 2: Restructured relational schema (2/2)

Revision(reserve, species, year, month, day, change, newSpecies*)

foreign key: Revision[reserve,species] $\subseteq$ RelPlan[reserve,species]

foreign key: Revision[newSpecies] $\subseteq$ Species[code]

domain constraint: change > 0 (realizes external constraint 1)

tuple constraint: newSpecies IS NULL OR species $\neq$ newSpecies (realizes ext. con. 3)

Views to reconstruct the merged relations:

View Destination(rpr,rps,dr) =

```
SELECT reserve, species, destReserve FROM RelPlan
```

View NewTarget(r,s,y,m,ns) =

```
SELECT reserve, species, year, month, newSpecies
FROM Revision WHERE newSpecies IS NOT NULL
```

View ProtSpecies(c,rq) =

```
SELECT code, relQuota
FROM Species WHERE flagProtected = 'true'
```

Problem 3

Consider a database D containing the following relations:

- i. **Spacecraft** (scCode, agency, launchYear), which stores relevant information about spacecrafts;
- ii. **CelestialBody** (bCode, type, distance), which stores relevant information about celestial bodies.
- iii. **Visit** (scCode, bCode, date, duration), which stores visits carried out by spacecraft to celestial bodies.

The attribute **type** has 'planet' among its possible values.

Write the following queries over D :

- 1. A query in **relational algebra** that computes the code and launch year of each spacecraft, together with the code of the celestial body and the date of the visit, for all visits carried out by spacecraft of agency 'ESA' to planets.
- 2. A query in **SQL** that for each spacecraft that has visited at least three distinct celestial bodies, computes the spacecraft code, the number of distinct celestial bodies it has visited, and the average duration of its visits.
- 3. A query in **SQL** that computes the code and launch year of all spacecraft that have visited *every planet* stored in the database at least once.

Problem 3: Solution 1

Spacecraft (scCode, agency, launchYear)

CelestialBody (bCode, type, distance)

Visit (scCode, bCode, date, duration)

1. Write a query in **relational algebra** that computes the code and launch year of each spacecraft, together with the code of the celestial body and the date of the visit, for all visits carried out by spacecraft of agency 'ESA' to planets.

PROJECT_{scCode, launchYear, bCode, date}
(**SELECT**_{agency = 'ESA' and type = 'planet'} (Spacecraft **JOIN** Visit **JOIN** CelestialBody))

Problem 3: Solution 2

Spacecraft (scCode, agency, launchYear)

CelestialBody (bCode, type, distance)

Visit (scCode, bCode, date, duration)

2. Write a query in **SQL** that for each spacecraft that has visited at least three distinct celestial bodies, computes the spacecraft code, the number of distinct celestial bodies it has visited, and the average duration of its visits.

```
SELECT V.scCode,  
       COUNT(DISTINCT V.bCode) AS numberOfBodies,  
       AVG(V.duration) AS averageDuration  
FROM Visit V  
GROUP BY V.scCode  
HAVING COUNT(DISTINCT V.bCode) >= 3
```

We are assuming that the FK constraint $\text{Visit}[\text{scCode}] \subseteq \text{Spacecraft}[\text{scCode}]$ holds.

Problem 3: Solution 3

Spacecraft (scCode, agency, launchYear)

CelestialBody (bCode, type, distance)

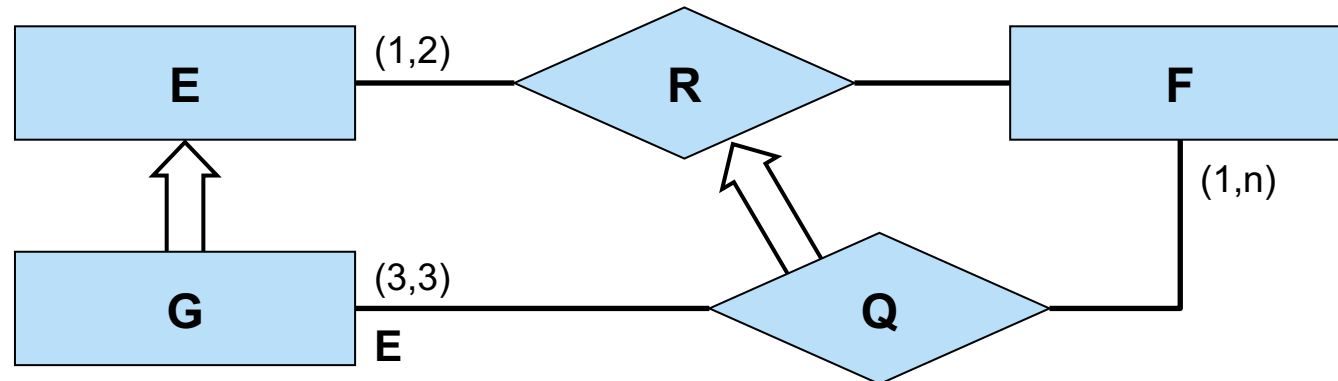
Visit (scCode, bCode, date, duration)

3. Write a query in **SQL** that computes the code and launch year of all spacecraft that have visited every planet stored in the database at least once.

```
SELECT S.scCode, S.launchYear
FROM Spacecraft S
WHERE NOT EXISTS (SELECT *
                  FROM CelestialBody B
                  WHERE B.type = 'planet'
                     AND B.bCode NOT IN (SELECT V.bCode
                                         FROM Visit V
                                         WHERE V.scCode = S.scCode))
```

Problem 4

Give the definition of an *integrity constraint* in general and of a *cardinality constraint* in particular. Then consider the conceptual schema **S** shown below and state whether there exists an instance of **S** in which the set of instances of **F** is non-empty. If the answer is positive, exhibit such an instance; otherwise, provide an appropriate justification. Recall that an instance of a conceptual schema is an extensional level in which all the properties and all the integrity constraints of the conceptual schema are satisfied.



Problem 4: Solution

There is no instance of the schema **S** in which the set of instances of **F** is non-empty. Indeed, let **I** be any instance of **S** and assume that there is a $z \in \text{Instances}(\mathbf{I}, \mathbf{F})$. Then z must participate in the relationship **Q** in the role **F**, and therefore $\text{Instances}(\mathbf{I}, \mathbf{Q})$ contains a tuple $\langle \mathbf{E}:e, \mathbf{F}:z \rangle$, with $e \in \text{Instances}(\mathbf{I}, \mathbf{G})$. Since e is an instance of **G**, it must participate three times in the relationship **Q**, and therefore $\text{Instances}(\mathbf{I}, \mathbf{Q})$ contains two *additional* tuples $\langle \mathbf{E}:e, \mathbf{F}:z_2 \rangle$ and $\langle \mathbf{E}:e, \mathbf{F}:z_3 \rangle$, with z, z_2 , and z_3 pairwise distinct. However, all tuples that are instances of **Q** are also instances of **R**. Hence the three tuples $\langle \mathbf{E}:e, \mathbf{F}:z \rangle$, $\langle \mathbf{E}:e, \mathbf{F}:z_2 \rangle$, and $\langle \mathbf{E}:e, \mathbf{F}:z_3 \rangle$ all belong to $\text{Instances}(\mathbf{I}, \mathbf{R})$. This violates the cardinality constraint (1,2), which requires each instance of **E**, in particular e , to participate in the relationship **R** in the role **E** at most twice. Hence, such a z cannot exist.

