Introduction to Databases

Exam of 09/07/2026

With Solutions

Diego Calvanese

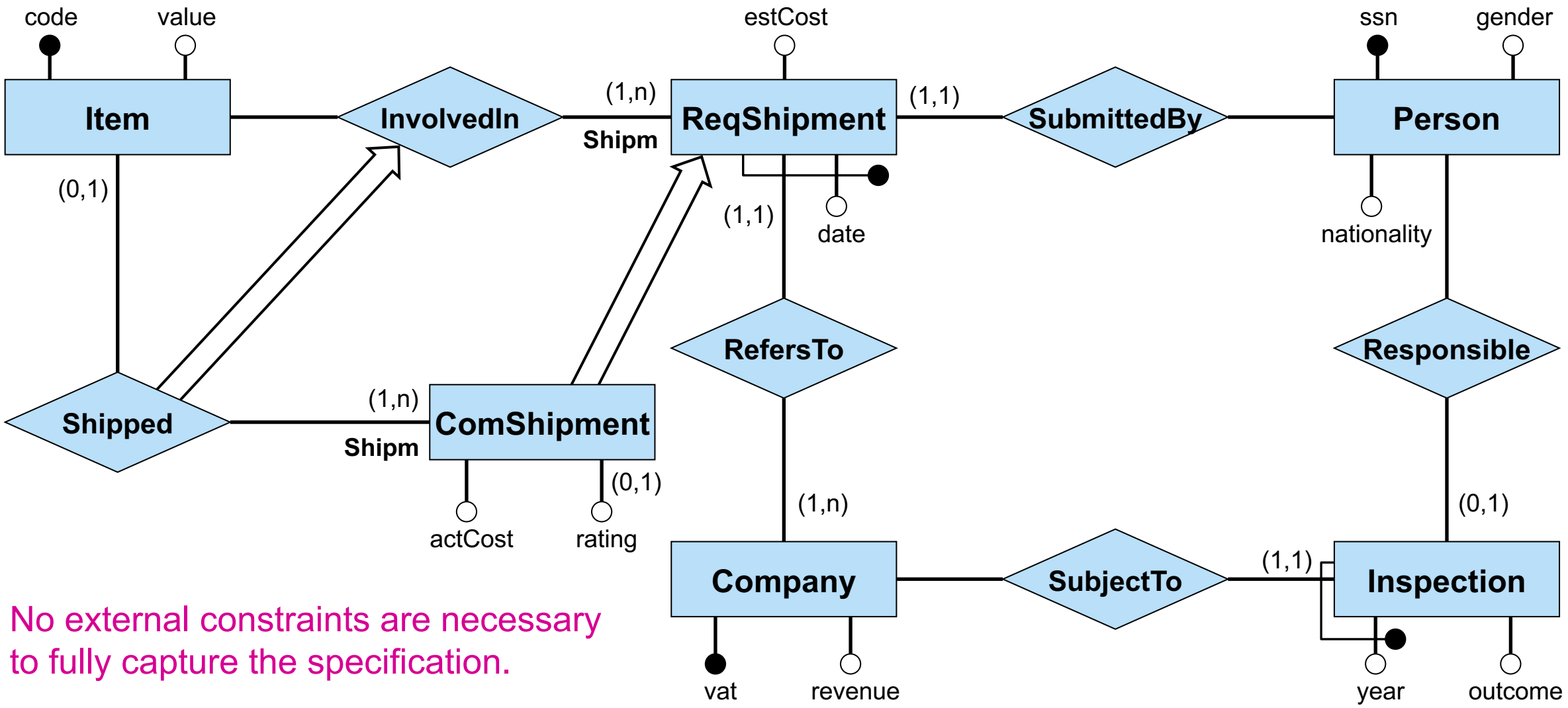
Bachelor in Computer Science
Faculty of Engineering
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema for an application that manages the shipment of merchandise items for an e-commerce business. Of each **item** registered in the system, we are interested in its code (identifier), its value, and the requested shipments that involve it. Each **requested shipment** is submitted by a person on a certain date, concerns a set of items (at least one), has an associated estimated cost, and refers to a shipping company that has been assigned to carry it out. Two requested shipments submitted on the same date cannot refer to the same shipping company. A **completed shipment**, which is a requested shipment that has been carried out, concerns a non-empty subset of the requested items. Of each completed shipment, we are interested in the actual cost and the quality rating (if available). Note that an item may be involved in several (possibly none) requested shipments, but in at most one completed shipment. Of each **person** registered in the system, we are interested in their social security number (identifier), gender, and nationality. Of each **shipping company**, which should be assigned to at least one shipment, we are interested in its VAT number (identifier), its revenue, and the quality inspections (no more than one per year) to which it has been subjected. Of each **quality inspection**, we are interested in the year in which it was carried out, the person responsible (if known), and the outcome.

Problem 1: Conceptual schema



No external constraints are necessary to fully capture the specification.

Problem 2

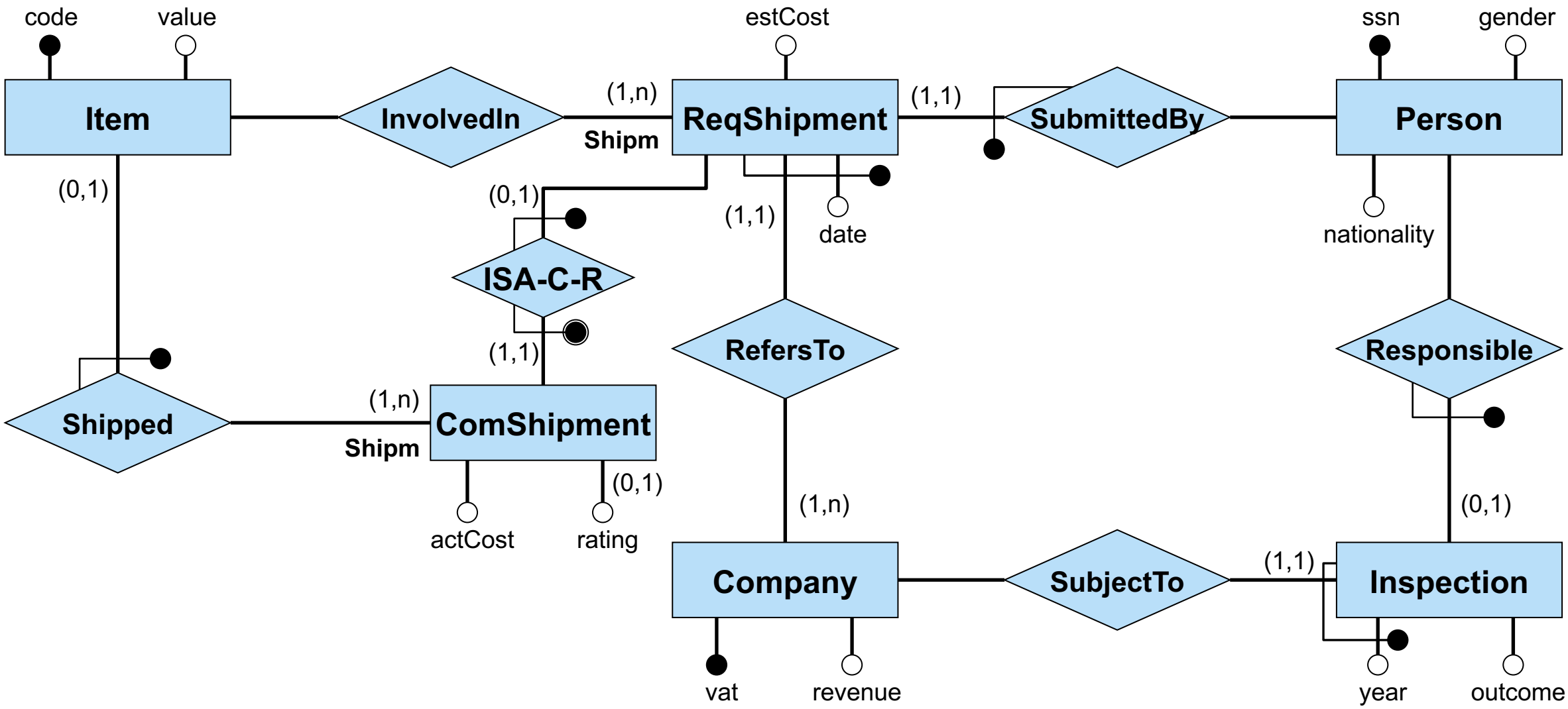
Carry out the logical design of the database, producing the complete relational schema with constraints, taking into account the following requirement: whenever a quality inspection of a shipping company is accessed, we are always interested in knowing the person responsible (if any), together with their nationality.

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: Conceptual schema – External constraint

Constraint resulting from the elimination of ISA between **Shipped** and **InvolvedIn**:

For each instance *I* of the schema, if (Item:*i*, Shipm:*s*) ∈ *instances*(*I*,**Shipped**), consider the unique *r* ∈ *instances*(*I*,**ReqShipment**) such that (ComShipment:*s*, ReqShipment:*r*) ∈ *instances*(*I*,**ISA-C-R**).
Then (Item:*i*, Shipm:*r*) ∈ *instances*(*I*,**InvolvedIn**).

Problem 2: Direct translation (1/2)

Item(code, value)

Person(ssn, gender, nationality)

Company(vat, revenue)
  inclusion: Company[vat] $\subseteq$ ReqShipment[company]

ReqShipment(date, company, estCost)
  foreign key: ReqShipment[company] $\subseteq$ Company[vat]
  foreign key: ReqShipment[date,company] $\subseteq$ SubmittedBy[sDate,sCompany]
  inclusion: ReqShipment[date,company] $\subseteq$ InvolvedIn[sDate,sCompany]

ComShipment(date, company, actCost, rating*)
  foreign key: ComShipment[date,company] $\subseteq$ ReqShipment[date,company]
  inclusion: ComShipment[date,company] $\subseteq$ Shipped[sDate,sCompany]

InvolvedIn(item, sDate, sCompany)
  foreign key: InvolvedIn[item] $\subseteq$ Item[code]
  foreign key: InvolvedIn[sDate,sCompany] $\subseteq$ ReqShipment[date,company]

Problem 2: Direct translation (2/2)

Shipped(item, sDate, sCompany)
- foreign key: Shipped[sDate,sCompany] $\subseteq$ ComShipment[date,company]
- foreign key: Shipped[item] $\subseteq$ Item[code] (Notice that this FK is implied by the next one and the FK on **InvolvedIn**, thus it could be omitted.)
- foreign key: Shipped[item,sDate,sCompany] $\subseteq$ InvolvedIn[item,sDate,sCompany] (This FK expresses the constraint resulting from the elimination of ISA between **Shipped** and **InvolvedIn**.)

SubmittedBy(sDate, sCompany, person)
- foreign key: SubmittedBy[sDate,sCompany] $\subseteq$ ReqShipment[date,company]
- foreign key: SubmittedBy[person] $\subseteq$ Person[ssn]

Inspection(year, company, outcome)
- foreign key: Inspection[company] $\subseteq$ Company[vat]

Responsible(inspYear, inspCompany, person)
- foreign key: Responsible[inspYear,inspCompany] $\subseteq$ Inspection[year,company]
- foreign key: Responsible[person] $\subseteq$ Person[ssn]

Problem 2: Restructuring of the relational schema

Indication: whenever a quality inspection of a shipping company is accessed, we are always interested in knowing the person responsible (if any), together with their nationality.

We take into account the indication by merging relations **Inspection** and **Responsible**, which are weakly coupled, and by further merging the resulting relation, which we still call **Inspection**, with relation **Person**, so as to bring also the attribute **nationality** into the relation **Inspection** (which becomes "denormalized"). As a result of this restructuring, relation **Responsible** is removed from the schema and table **Inspection** becomes:

Inspection(year, company, outcome, person*, nationality*)
  foreign key: Inspection[company] $\subseteq$ Company[vat]
  foreign key: Inspection[person,nationality] $\subseteq$ Person[ssn,nationality]
  tuple constraint: person is null if and only if nationality is null

Notice that we can reconstruct the relation **Responsible** by means of the following view:

view Responsible(inspYear,inspCompany,person) = $\mathbf{PROJ}_{year,company,person}$(Inspection)

Problem 3

Consider a database *D* containing the following relations:

i. `Person(taxCode,dateOfBirth)`, which stores the tax code (primary key) and the date of birth of persons;

ii. `Birth(taxCode,municipality)` and `WorkedIn(taxCode,municipality)`, which store, for each person, respectively the municipality of birth (if known) and the municipalities in which they have worked (a person may have never worked).

It is known that the only foreign keys that are satisfied in the database are the one from `Birth[taxCode]` to `Person[taxCode]` and the one from `WorkedIn[taxCode]` to `Person[taxCode]`.

Write the following queries over *D*:

1. Write a query in **SQL** that computes *all* persons, with their tax code, date of birth, and, if known, also their municipality of birth.
2. Write a query in **relational algebra** that computes the tax code of all persons who have worked *only* in their municipality of birth.
3. Write a query in **SQL** that computes, for *each* person, their tax code and the number of municipalities in which they have worked.

Problem 3: Solution 1

```
Person(taxCode,dateOfBirth)
Birth(taxCode,municipality)
WorkedIn(taxCode,municipality)
```

1. Write a query in **SQL** that computes *all* persons, with their tax code, date of birth, and, if known, also their municipality of birth.

```
SELECT P.taxCode, P.dateOfBirth, B.municipality
FROM Person P LEFT OUTER JOIN Birth B ON
       P.taxCode = B.taxCode
```

Problem 3: Solution 2

```
Person(taxCode,dateOfBirth)
Birth(taxCode,municipality)
WorkedIn(taxCode,municipality)
```

2. Write a query in **relational algebra** that computes the tax code of all persons who have worked *only* in their municipality of birth.

$$\textbf{PROJ}_{taxCode}\ (\text{Person}) - \textbf{PROJ}_{taxCode}\ (\text{Birth}\ \textbf{JOIN}_{taxCode = tc\ AND\ municipality <> mun}\ \textbf{REN}_{tc \leftarrow taxCode,\ mun \leftarrow municipality}\ \text{WorkedIn}))$$

Notice that a person who has never worked qualifies as a person who has worked *only* in their municipality of birth, and therefore is included in the answer.

Problem 3: Solution 3

```
Person(taxCode,dateOfBirth)
Birth(taxCode,municipality)
WorkedIn(taxCode,municipality)
```

3. Write a query in **SQL** that computes , for *each* person, their tax code and the number of municipalities in which they have worked.

```
SELECT P.taxCode, COUNT(W.municipality) AS numMunicipalities
FROM Person P LEFT OUTER JOIN WorkedIn W
        ON P.taxCode = W.taxCode
GROUP BY P.taxCode
```

Notice that due to the primary key of `WorkedIn`, the same municipality cannot occur twice for the same person in `WorkedIn`, and therefore we do *not* need to use `COUNT(DISTINCT W.municipality)`.

Problem 4

R

A	B	C	D
5	7	1	5
5	5	2	5
null	6	3	7
5	5	4	6

S

E
5
8
9

Consider the relational database shown above and answer the following questions:

1. Which key constraints are satisfied by relation **R**?
   In other words, what are the keys of **R**?
2. Which superkey constraints are satisfied by relation **R**?
   In other words, what are the superkeys of **R**?
3. Which foreign key constraints are satisfied by the database?

Problem 4: Solution

R

A	B	C	D
5	7	1	5
5	5	2	5
null	6	3	7
5	5	4	6

S

E
5
8
9

1. The keys of **R** are: {C}, {B,D}.
2. The superkeys of **R** are: {C}, {C,A}, {C,B}, {C,D}, {C,A,B}, {C,A,D}, {C,B,D}, {C,A,B,D}, {B,D}, {B,D,A}.
3. The only foreign key constraint satisfied by the database is $R[A] \subseteq S[E]$. Remember that in a foreign key constraint, the target attributes must be a key of the target relation. And indeed, {E} is a key of **S**.