# Introduction to Databases

## Exam of 28/06/2022

*With Solutions*

**Diego Calvanese**

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

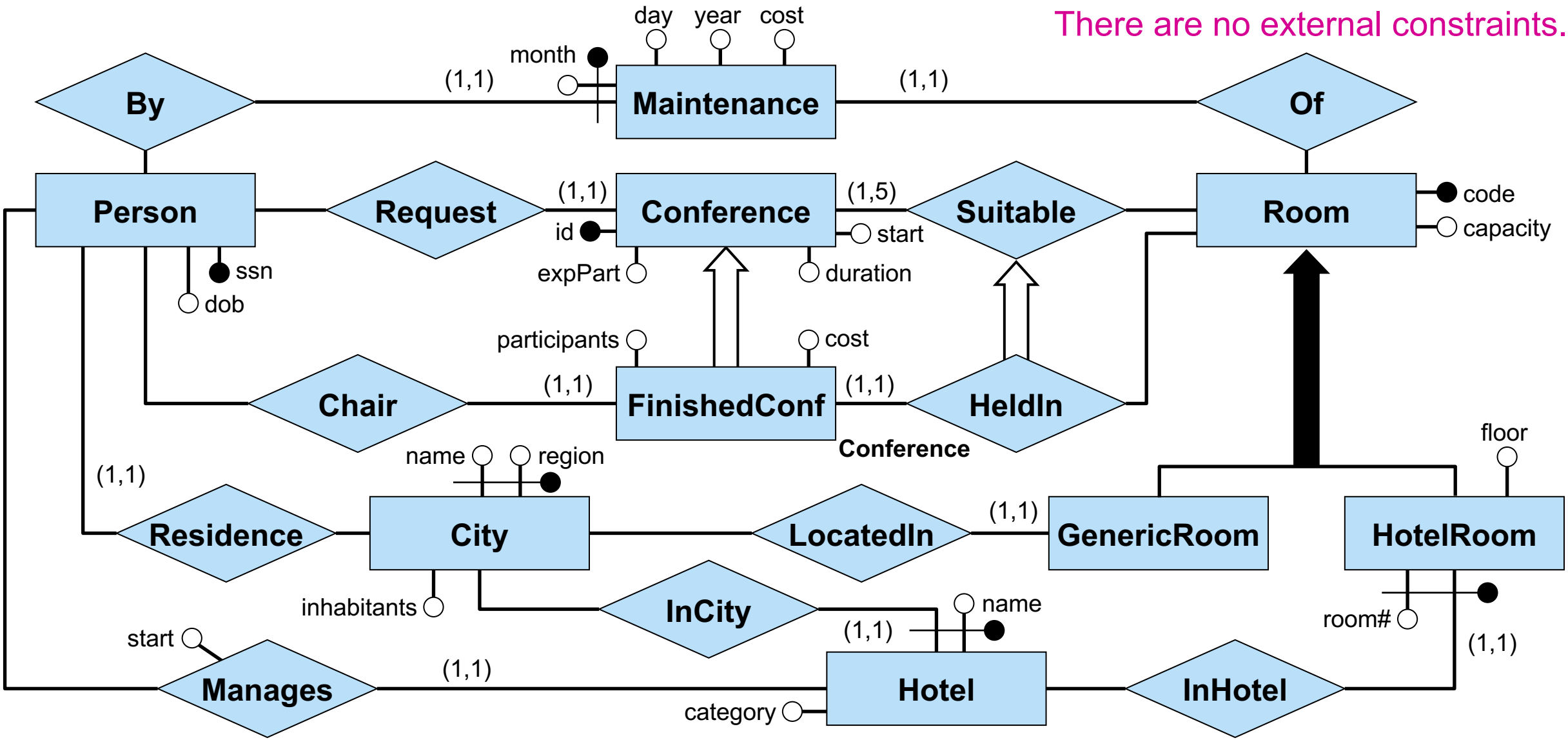
**http://www.inf.unibz.it/~calvanese/teaching/exams/idb/**

# Problem 1

Design the Entity-Relationship schema of an application supporting the organization of conferences. When a person requests to organize a conference, the **conference** is assigned a unique identifier and the person who issued the request is recorded, together with the expected number of participants, the start date, the duration, and the rooms (at least one and at most 5) that are considered suitable for the conference. For **finished conferences**, we are interested also in the actual number of participants, the total cost, the person who acted as conference chair, and the room in which the conference was actually held (this room must be one of those that were reported as suitable at the time of the initial request). Of each **room** we are interested in the code (identifier), the capacity, and the maintenance interventions undergone by that room. Each **maintenance intervention** is related to a room, occurs on a date, has a cost, and is carried out under the responsibility of one person, with the rule that no person can be responsible for more than one maintenance intervention per month. The rooms are classified into two categories: generic rooms and hotel rooms. Of each **generic room**, we are interested in the area and the city in which it is located. Of each **hotel room** we are interested in the hotel in which it is located, the floor, and the room number (which is unique within the hotel in which the room is located). Of each **hotel** we are interested in the city in which it is located, the name (unique within the city), the category, and the person who manages it (with the date when the direction started). Of each **city** we are interested in the name (unique within the region), the region, and the number of inhabitants. Of each **person** we are interested in the social security number (identifier), the date of birth, and the city of residence.

# Problem 1: Conceptual schema

There are no external constraints.



# Problem 2

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

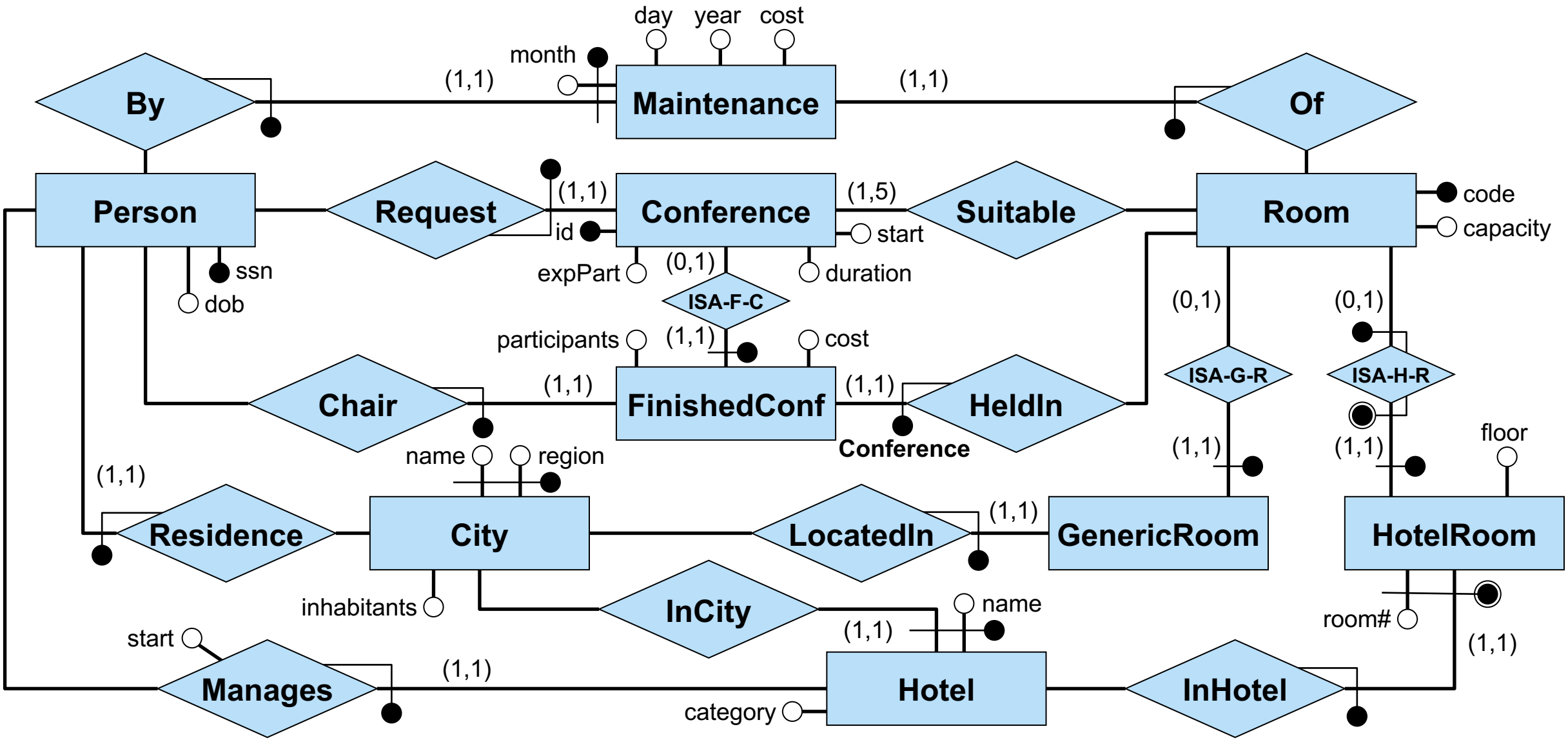
1. Each hotel room is always accessed by the room number and the hotel in which it is located.
2. When accessing a hotel we always want to know who its manager is.

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



# Problem 2: Restrucured Conceptual Schema – External constraints

1) Generalization constraint for the entity **Room**:
   For each instance *I* of the schema, each instance of **Room** in *I*, participates either to **ISA-G-R** or to **ISA-H-R**, but not to both.
2) Constraint resulting from the elimination of ISA between the relationships **HeldIn** and **Suitable**:
   For each instance *I* of the schema, if (Conference:*f*, Room:*r)* ∈ *instances*(*I*,**HeldIn**) and *c* is the unique element of *instances*(*I*,**Conference**) such that (FinishedConf:*f*,Conference:*c*) ∈ *instances*(*I*,**ISA-F-C**), then (Conference:*c*,Room:*r*) ∈ *instances*(*I*,**Suitable**).

# Problem 2: Direct translation (1/3)

Conference(id, expPart, start, duration)
  foreign key: Conference[id] $\subseteq$ Request[conference]
  inclusion: Conference[id] $\subseteq$ Suitable[conference]

FinishedConf(id, sparticipants, cost)
  foreign key: FinishedConf[id] $\subseteq$ Conference[id]
  foreign key: FinishedConf[id] $\subseteq$ Chair[conference]
  foreign key: FinishedConf[id] $\subseteq$ HeldIn[conference]

Person(ssn, dob)
  foreign key: Person[ssn] $\subseteq$ Residence[person]

City(name, region, inhabitants)

Hotel(name, city, region, category)
  foreign key: Hotel[name,city,region] $\subseteq$ Manages[hotelName,hotelCity,hotelRegion]
  foreign key: Hotel[city,region] $\subseteq$ City[name,region]

Room(code, capacity)

GenericRoom(code)
  foreign key: GenericRoom[code] $\subseteq$ Room[code]
  foreign key: GenericRoom[code] $\subseteq$ LocatedIn[room]

# Problem 2: Direct translation (2/3)

HotelRoom(room#, hotel, city, region, floor)
  foreign key: HotelRoom[hotel,city,region] $\subseteq$ Hotel[name,city,region]
  foreign key: HotelRoom[room#,hotel,city,region] $\subseteq$ ISA-H-R[room#,hotel,city,region]

Maintenance(person, month, day, year, cost)
  foreign key: Maintenance[person] $\subseteq$ Person[ssn]
  foreign key: Maintenance[person,month] $\subseteq$ Of[maintPerson,maintMonth]

Request(conference, person)
  foreign key: Request[conference] $\subseteq$ Conference[id]
  foreign key: Request[person] $\subseteq$ Person[ssn]

Suitable(conference, room)
  foreign key: Suitable[conference] $\subseteq$ Conference[id]
  foreign key: Suitable[room] $\subseteq$ Room[code]

Chair(conference, person)
  foreign key: Chair[conference] $\subseteq$ FinishedConf[id]
  foreign key: Chair[person] $\subseteq$ Person[ssn]

HeldIn(conference, room)
  foreign key: HeldIn[conference] $\subseteq$ FinishedConf[id]
  foreign key: HedlIn[room] $\subseteq$ Room[code]
  foreign key: HeldIn[conference,room] $\subseteq$ Suitable[conference,room]

# Problem 2: Direct translation (3/3)

Residence(person, city, region)
  foreign key: Residence[person] $\subseteq$ Person[ssn]
  foreign key: Residence[city,region] $\subseteq$ City[name,region]

LocatedIn(room, city, region)
  foreign key: LocatedIn[room] $\subseteq$ GenericRoom[code]
  foreign key: LocatedIn[city,region] $\subseteq$ City[name,region]

Manages(hotel, city, region, person, start)
  foreign key: Manages[hotel,city,region] $\subseteq$ Hotel[name,city,region]
  foreign key: Manages[person] $\subseteq$ Person[ssn]

ISA-H-R(room#, hotel, city, region, room)
  foreign key: ISA-H-R[room#,hotel,city,region] $\subseteq$ HotelRoom[room#,hotel,city,region]
  foreign key: ISA-H-R[room] $\subseteq$ Room[code]
  key: room

Of(maintPerson, maintMonth, room)
  foreign key: Of[maintPerson,maintMonth] $\subseteq$ Maintenance[person,month]
  foreign key: Of[room] $\subseteq$ Room[code]

# Problem 2: Direct translation – External constraints

1. Generalization constraints:

   $Room[code] \subseteq GenericRoom[code] \cup ISA\text{-}H\text{-}R[room]$

   $GenericRoom[code] \cap ISA\text{-}H\text{-}R[room] = \emptyset$

2. The constraint resulting from the elimination of ISA between the relationships **HeldIn** and **Suitable** has already been specified as the foreign key of the **HeldIn** relation:

   $HeldIn[conference,room] \subseteq Suitable[conference,room]$

3. The constraint resulting from the maximum cardinality 5 on the **Conference** role of **Suitable** can be realized through a `CHECK` clause on `Suitable`:

```
CHECK (5 >= (SELECT COUNT(*)
             FROM Suitable S
             WHERE conference = S.conference))
```

# Problem 2: Restructuring of the relational schema

1. Each hotel room is always accessed by the room number and the hotel in which it is located.
2. When accessing a hotel we always want to know who its manager is.

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

- We have already taken into account indication 1 by choosing as primary identifier of the entity **HotelRoom** the external identifier involving the relationship **InHotel**.
- We take into account indication 2 by merging the relation **Manages** in the relation **Hotel**.

# 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.

Hotel(name, city, region, category, person, start)

foreign key: Hotel[city,region] $\subseteq$ City[name,region]

foreign key: Hotel[person] $\subseteq$ Person[ssn]

There are no constraints that need to be changed or added due to the transformations, so we just keep the generalization constraints that we had.

# Problem 3

Consider a database of a library that includes the relations `Book`, `Client`, and `Borrowed`. The relation `Book(name,length)` stores for each book in the library the name and its length in pages, the relation `Client(ssn,city)` stores for each registered client of the library the ssn and the city of birth, and the relation `Borrowed(ssn,name,year)` stores which books have been borrowed by which clients in which year.

Express the following queries in SQL:

1. Return the ssn of the clients born in Bolzano who since 2020 have borrowed at least one book longer than 200 pages.
2. For each client, return the average length of the books they have borrowed.
3. Return the ssn and the city of birth of those clients who have borrowed all books in the library that are at least 1000 pages long.

# Problem 3: Solution (1/2)

```
Book(name,length)
Client(ssn,city)
Borrowed(ssn,name,year)
```

1. Return the ssn of the clients born in Bolzano who since 2020 have borrowed at least one book longer than 200 pages.

```
SELECT DISTINCT C.ssn
FROM (Client C JOIN Borrowed Bo ON C.ssn = Bo.ssn) JOIN
     Book B ON Bo.name = B.name
WHERE C.city = 'Bolzano' AND Bo.year >= 2020 AND B.length > 200
```

2. For each client, return the average length of the books they have borrowed.

```
SELECT Bo.ssn, AVG(B.length)
FROM Borrowed Bo JOIN Book B ON Bo.name = B.name
GROUP BY Bo.ssn
```

# Problem 3: Solution (2/2)

```
Book(name,length)
Client(ssn,city)
Borrowed(ssn,name,year)
```

3. Return the ssn and the city of birth of those clients who have borrowed all books in the library that are at least 1000 pages long.

```
SELECT C.ssn, C.city
FROM Client C
WHERE NOT EXISTS (SELECT B.name
                  FROM Book B
                  WHERE B.length >= 1000 AND
                        B.name NOT IN (SELECT Bo.name
                                       FROM Borrowed Bo
                                       WHERE Bo.ssn = C.ssn))
```

# Problem 4

Consider a relational schema containing two relations whose schemas are `R(A,B,C,D)` and `Q(E,F)`. Provide the SQL `CREATE TABLE` statement for relation `R` taking into account that:

1. all attributes are of type `INTEGER`;
2. attribute `A` is the primary key of `R`;
3. attribute `B` cannot take null values;
4. every value that appears in `C` also appears in `E` (where `E` is the primary key of `Q`);
5. every value that appears in $\text{PROJ}_D(R)$ also appears in $\text{PROJ}_F(\text{SEL}_{E=1}(Q))$.

# Problem 4: Solution

```
CREATE TABLE R (
  A INTEGER PRIMARY KEY,
  B INTEGER NOT NULL,
  C INTEGER REFERENCES Q(E),
  D INTEGER CHECK (D IN (SELECT F FROM Q WHERE E=1))
)
```