# Introduction to Databases

## Exam of 19/09/2022

*With Solutions*

**Diego Calvanese**

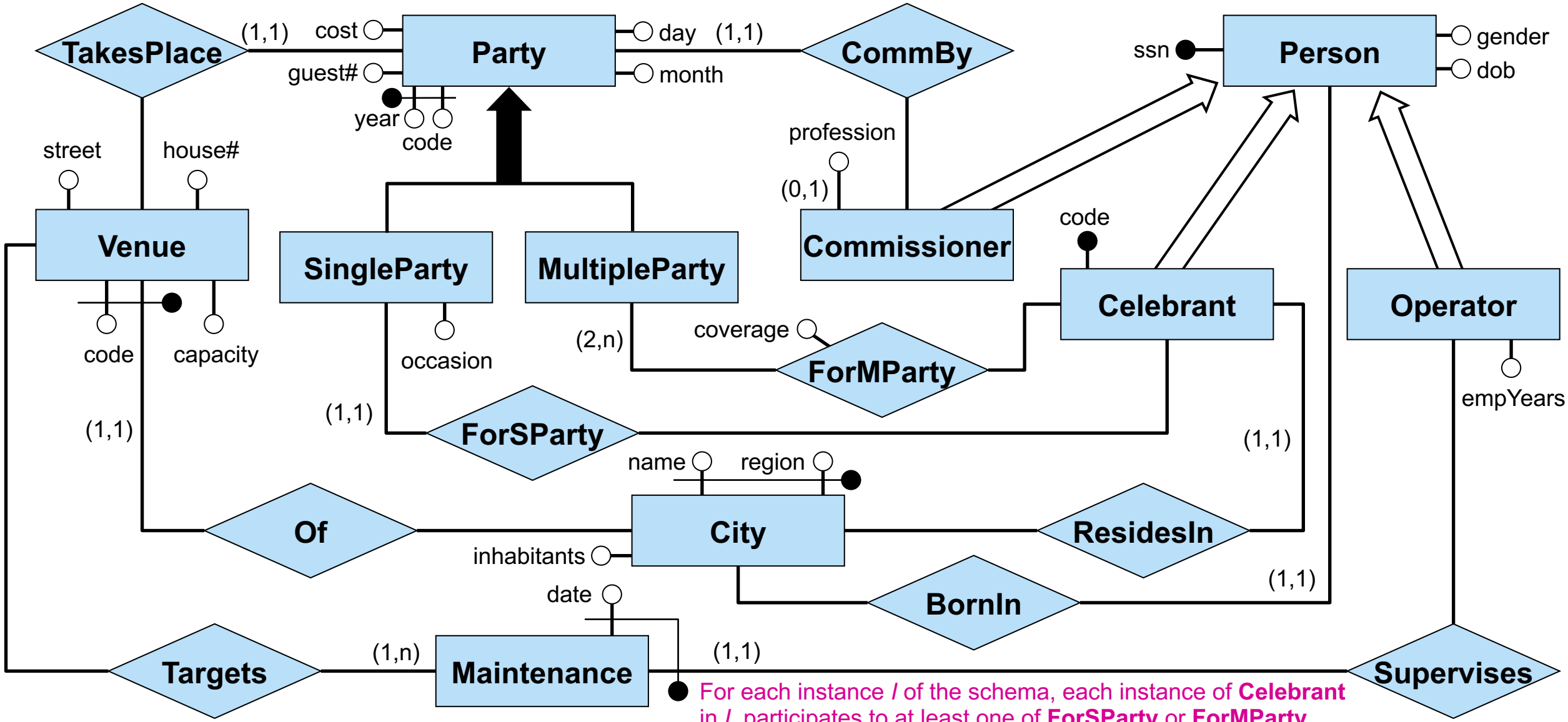
*Bachelor in Computer Science*
*Faculty of Computer Science*
*Free University of Bozen-Bolzano*

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

# Problem 1

Design the Entity-Relationship schema of an application supporting a company in the organization of parties celebrating a person. Of each **party** we are interested in the code (unique within the year in which the party takes place), the date on which it takes place, the venue in which it takes place, the number of expected guests, the expected cost, and the person who commissioned it. There are two types of parties, depending on how many persons are being celebrated: single parties (one celebrant) and multiple parties (at least two celebrants). Of each **single party**, we are interested to know who is the celebrant, and for what occasion the party is held (birthday, promotion, etc.). Of each **multiple party** we are interested to know who are the celebrants, and what percentage of the expenses each of them will cover. Of each **person** we are interested in the social security number, gender, date of birth, and city of birth. In addition, of each **celebrant** we are interested in the code (identifier assigned by the company) and the city in which the person resides. Of each **person commissioning a party** we are interested in the profession (which, however, is significant only if the person is employed). Of each **venue** we are interested in the code (unique within the city), the address (consisting of city, street, and house number), and the capacity. Of each **city** we are interested in the name (unique with the region), the region, and the number of inhabitants. Moreover, of each venue we are also interested in all the maintenance operations that it has undergone, where each **maintenance operation** targets one or more venues, takes place on a certain date, and is supervised by an operator, with the rule that no operator can supervise more than one maintenance operation per day. Notice that an **operator** is a person employed by the company whose year of employment is of interest. There are no restrictions on who can be operator, celebrant, or commission a party.

# Problem 1: Conceptual schema



# Problem 2

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

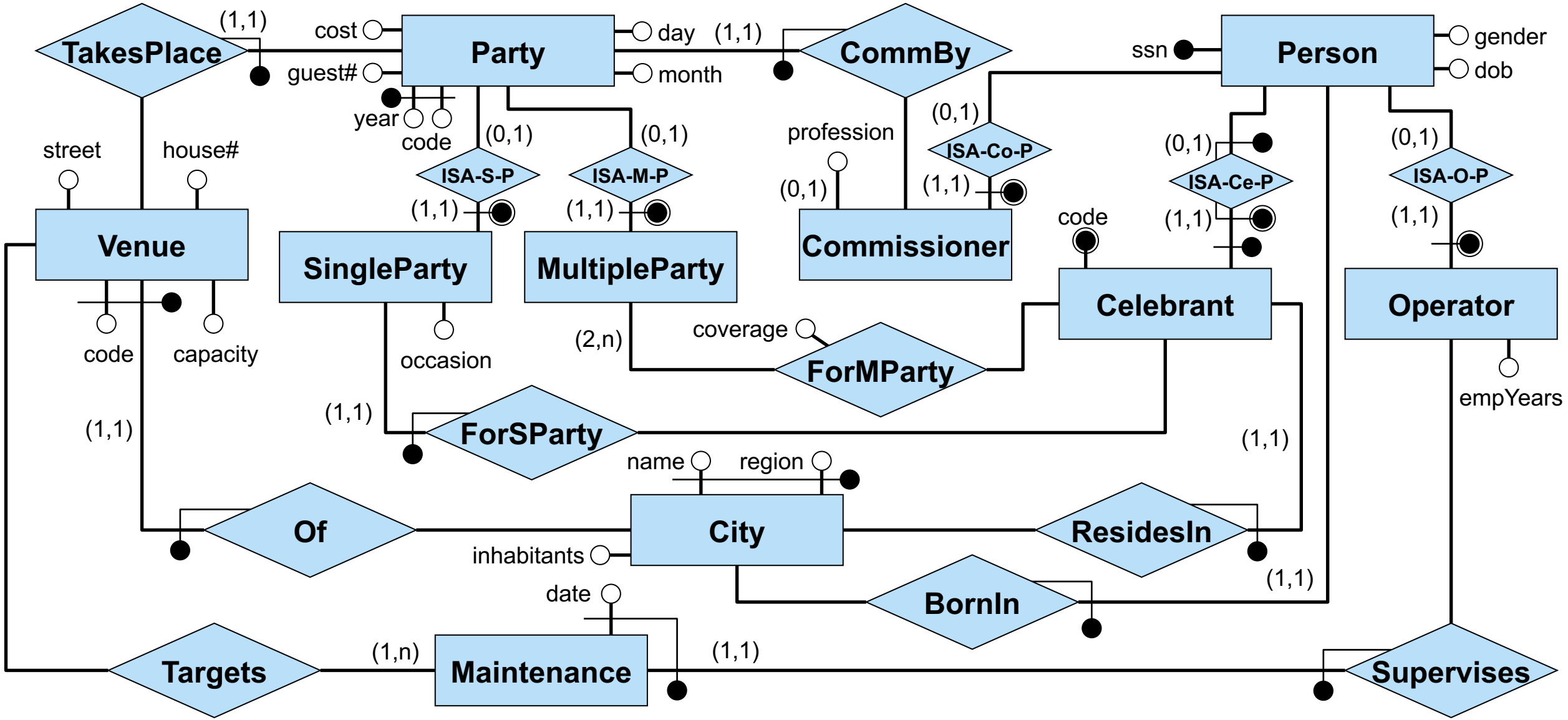
1. Celebrant's data are accessed using the code assigned by the company.
2. Null values in the database should be avoided.
3. When accessing the data on a maintenance operation, we always want to know the operator who supervised it.

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



# Problem 2: Restrucured Conceptual Schema – External constraints

1. Generalization constraint for the entity **Party**:
   For each instance *I* of the schema, each instance of **Party** in *I*, participates either to **ISA-S-P** or to **ISA-M-P**, but not to both.
2. Constraint on the entity **Celebrant**:
   For each instance *I* of the schema, each instance of **Celebrant** in *I*, participates to at least one of **ForSParty** or **ForMParty**.

# Problem 2: Direct translation (1/3)

Party(year, code, day, month, cost, guest#)
  foreign key: Party[year,code] $\subseteq$ CommBy[partyYear,partyCode,commissioner]
  foreign key: Party[year,code] $\subseteq$ TakesPlace[partyYear,partyCode]

SingleParty(year, code, occasion)
  foreign key: SingleParty[year,code] $\subseteq$ Party[year,code]
  foreign key: SingleParty[year,code] $\subseteq$ ForSParty[partyY,partyC]

MultipleParty(year, code)
  foreign key: MultipleParty[year,code] $\subseteq$ Party[year,code]
  inclusion: MultipleParty[year,code] $\subseteq$ ForMParty[partyY,partyC]

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

Commissioner(ssn, profession*)
  foreign key: Commissioner[ssn] $\subseteq$ Person[ssn]

Operator(ssn, empYears)
  foreign key: Operator[ssn] $\subseteq$ Person[ssn]

Celebrant(code)
  foreign key: Celebrant[code] $\subseteq$ IsaCeP[celebrant]
  foreign key: Celebrant[code] $\subseteq$ ResidesIn[celebrant]

# Problem 2: Direct translation (2/3)

City(name, region, inhabitants)

Maintenance(date, operator)
- foreign key: Maintenance[operator] $\subseteq$ Operator[ssn]
- inclusion: Maintenance[date,operator] $\subseteq$ Targets[maintenanceDate,maintenanceOperator]

Venue(code, city, region, street, house#, capacity)
- foreign key: Venue[city,region] $\subseteq$ City[name,region]

Targets(venue, city, region, maintenanceDate, maintenanceOperator)
- foreign key: Targets[venue,city,region] $\subseteq$ Venue[code,city,region]
- foreign key: Targets[maintenanceDate,maintenanceOperator] $\subseteq$ Maintenance[date,operator]

ResidesIn(celebrant, city, region)
- foreign key: ResidesIn[celebrant] $\subseteq$ Celebrant[code]
- foreign key: ResidesIn[city,region] $\subseteq$ City[name,region]

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

# Problem 2: Direct translation (3/3)

ForSParty(partyYear, partyCode, celebrant)
  foreign key: ForSParty[partyYear,partyCode] $\subseteq$ SingleParty[year,code]
  foreign key: ForSParty[celebrant] $\subseteq$ Celebrant[code]

ForMParty(partyYear, partyCode, celebrant, coverage)
  foreign key: ForMParty[partyYear,partyCode] $\subseteq$ MultipleParty[year,code]
  foreign key: ForMParty[celebrant] $\subseteq$ Celebrant[code]

CommBy(partyYear, partyCode, commissioner)
  foreign key: CommBy[partyYear,partyCode] $\subseteq$ Party[year,code]
  foreign key: CommBy[commissioner] $\subseteq$ Commissioner[ssn]

TakesPlace(partyYear, partyCode, venueCode, city, region)
  foreign key: TakesPlace[partyYear,partyCode] $\subseteq$ Party[year,code]
  foreign key: TakesPlace[venueCode,city,region] $\subseteq$ Venue[code,city,region]

IsaCeP(celebrant, person)
  foreign key: IsaCeP[celebrant] $\subseteq$ Celebrant[code]
  foreign key: IsaCeP[person] $\subseteq$ Person[ssn]
  key: person

# Problem 2: Direct translation – External constraints

1. Generalization constraints:

   $Party[year,code] \subseteq SingleParty[year,code] \cup MultipleParty[year,code]$

   $SingleParty[year,code] \cap MultipleParty[year,code] = \emptyset$

2. Constraint on the entity **Celebrant**:
   $Celebrant[code] \subseteq ForSParty[celebrant] \cup ForMParty[celebrant]$

3. The constraint resulting from the minimum cardinality 2 on the **MultipleParty** role of **ForMParty** can be realized through a `CHECK` clause on `MultipleParty`:

```
CHECK (2 <= (SELECT COUNT(*)
             FROM ForMParty F
             WHERE year = F.partyYear AND
                   code = F.partyCode))
```

# Problem 2: Restructuring of the relational schema

1. Celebrant's data are accessed using the code assigned by the company.
2. Null values in the database should be avoided.
3. When accessing the data on a maintenance operation, we always want to know the operator who supervised it.

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 **Celebrant** the **code** attribute.
- We take into account indication 2 by performing a mixed decomposition of the relation `Commissioner`.
- Indication 3 is already taken into account since the operator is part of the (external) identifier of the **Maintenance** entity.

# 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.

Commissioner(ssn)

foreign key: Commissioner[ssn] $\subseteq$ Person[ssn]

CommissionerWithProfession(ssn, profession)

foreign key: CommissionerWithProfession[ssn] $\subseteq$ Commissioner[ssn]

There are no constraints that need to be changed or added due to the transformation.

# Problem 3

Consider a database that includes the relations `Person` and `WorksIn`.
The relation `Person(ssn,cityof,yearob,gender)` stores for each person the social security number, the city and year of birth, and the gender.
The relation `WorksIn(ssn,city,year)` stores information about which persons (represented by the ssn) have worked in which cities and in which years (for example, a person may have worked in Bolzano in the years 2015, 2016, 2020, and 2021).

Express the following queries in SQL:

1. Compute the women who have worked at least once in the city in which they were born.
2. For each person who has worked at least once, compute the age at which they first worked.
3. Compute the cities in which at least one male worked and in which no female born in that city has ever worked.

# Problem 3: Solution (1/2)

```
Person(ssn,cityof,yearob,gender)
WorksIn(ssn,city,year)
```

1. Compute the women who have worked at least once in the city in which they were born.

```
SELECT DISTINCT P.ssn
FROM Person P JOIN WorksIn W ON P.ssn = W.ssn
WHERE P.gender = 'F' AND W.city = P.cityob
```

2. For each person who has worked at least once, compute the age at which they first worked.

```
SELECT P.ssn, MIN(W.year-P.yearob)
FROM Person P JOIN WorksIn W ON P.ssn = W.ssn
GROUP BY P.ssn
```

# Problem 3: Solution (2/2)

```
Person(ssn,cityof,yearob,gender)
WorksIn(ssn,city,year)
```

3. Compute the cities in which at least one male worked and in which no female born in that city has ever worked.

```
SELECT DISTINCT W.city
FROM WorksIn W JOIN Person P ON W.ssn = P.ssn
WHERE P.gender = 'M' AND
      W.city NOT IN (SELECT W2.city
                     FROM WorksIn W2 JOIN Person P2
                          ON W2.ssn = P2.ssn
                     WHERE P2.gender = 'F' AND
                           W2.city = P2.cityob)
```

# Problem 4

Consider a relational schema containing two relations whose schemas are `R(A,B,C)` and `Q(D,E,F)`, with all attributes of type `INTEGER`. Show how the following constraints can be expressed within the SQL `CREATE TABLE` statement related to relation `R`:

1. No pair of values appears simultaneously in $\text{PROJ}_{A,B}(R)$ and $\text{PROJ}_{E,F}(Q)$.
2. Every value that appears in $\text{PROJ}_{C}(R)$ also appears in $\text{PROJ}_{F}(\text{SEL}_{D=1}(Q))$.

# Problem 4: Solution

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