Exercise on data warehousing

Data Management

A.Y. 2023/24

Maurizio Lenzerini

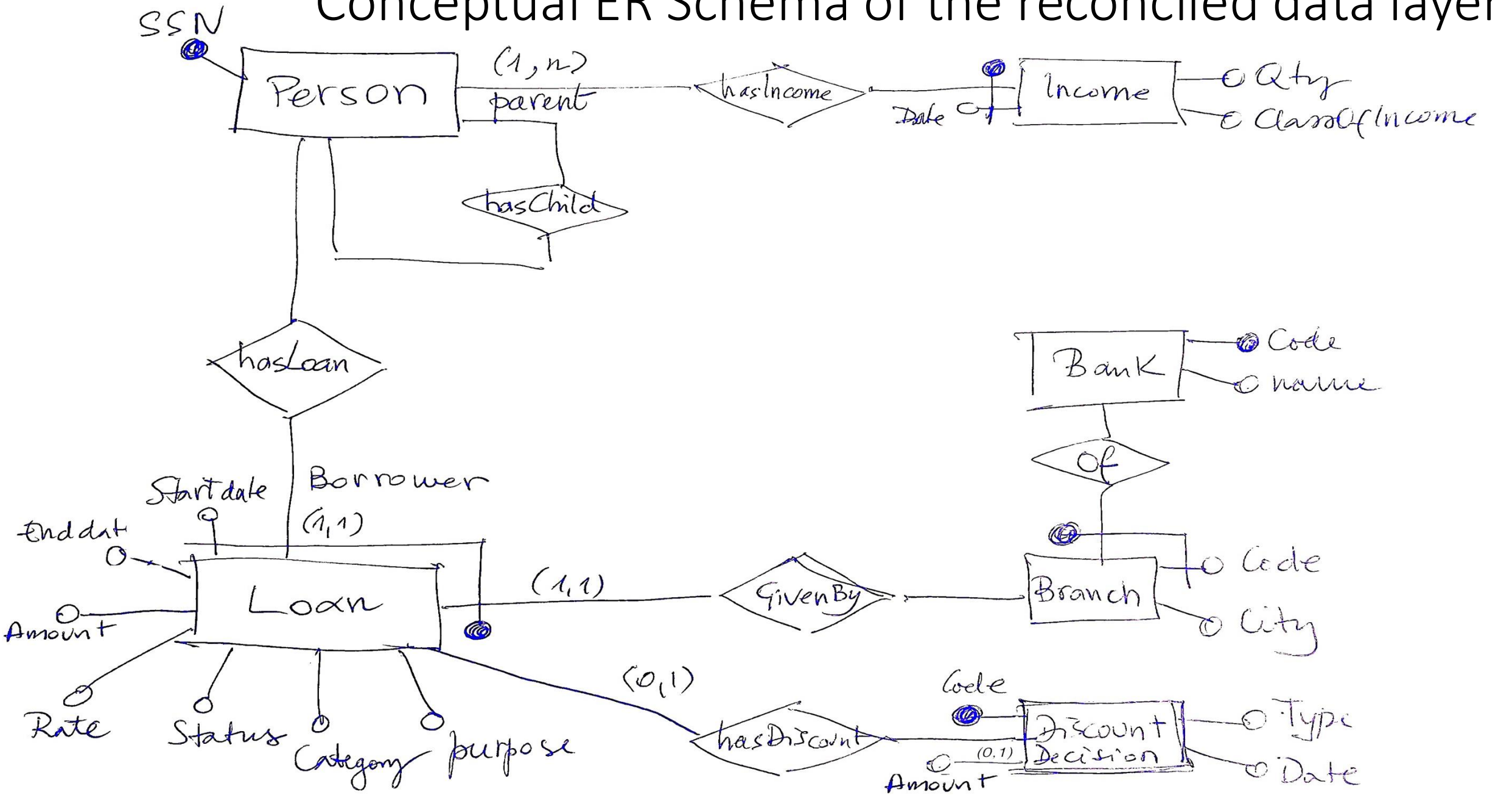
Plan for the exercise

We assume we have a complete documentation of the reconciled data layer derived from the operational data sources in a three-layers architecture. Our goal is to design a (very simple) data warehouse, suitably populated with data from the reconciled view.

- We first illustrate the relevant portion of the conceptual schema of the reconciled data layer
- Then we illustrate the logical (relational) schema of such portion
- Finally, we describe the requirements for the data warehouse

The design of the data warehouse should be done according to the given requirements.

Conceptual ER Schema of the reconciled data layer



Logical, relational schema of the reconciled data layer

Person(SSN)
    foreign key Person[SSN] ⊆ HasIncome[person]
HasIncome(person,date,qty,incomeClass)
    foreign key HasIncome[person] ⊆ Person[SSN]
HasChild(parent,child)
    foreign key HasChild[parent] ⊆ Person[SSN]
    foreign key HasChild[child] ⊆ Person[SSN]
Loan(borrower,branchcode,bankcode,startdate,enddate,amount,rate,status,category,purpose)
    foreign key Loan[borrower] ⊆ Person[SSN]
    foreign key Loan[branchcode,bankcode] ⊆ Branch[code,bank]
Branch(code,bank,city)
    foreign key Branch[bank] ⊆ Bank[code]
Bank(code,name)
HasDiscount(borrower,branchcode,bankcode,startdate,dcode)
    foreign key HasDiscount[borrower,branchcode,bankcode,startdate] ⊆
        Loan[borrower,branchcode,bankcode,startdate]
    foreign key HasDiscount[dcode] ⊆ DiscountDecision[code]
DiscountDecision(code,date,type,amount*)

Requirements for the data warehouse

In order to decide the content of the Data Warehouse, consider the following specification.

For every loan we are interested in the starting date (day, month and year), the end date (day, month and year), the borrower (with SSN, number of children, income at the starting date, class of income), the branch of the bank where the loan was issued (with name and city of the branch), the category of the loan (e.g., fixed rate or one of different types of variable rate), the purpose of the loan (e.g., to buy a car, a house, a personal loan, ...), the discount type (but not all loans have a discount type), with the date of the corresponding decision and the status of the loan (fully repaid, defaulted, ...). For every loan, the following values are of interest: the amount of the loan, the interest rate and the amount of discount (if it applies and is known).

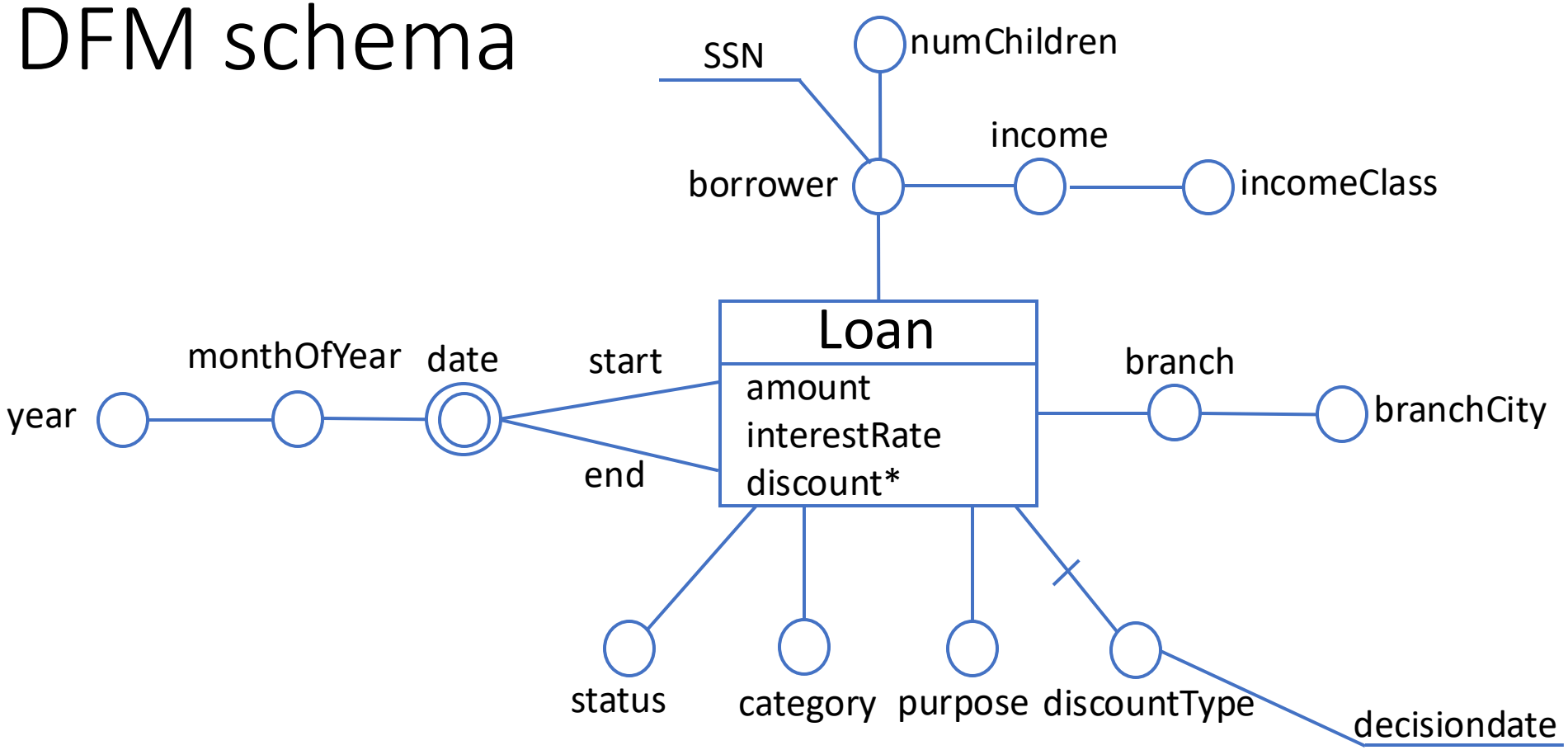
The following are some typical questions for building the data marts used to start OLAP sessions on the data warehouse by business analysts:

- Compute the average interest rate, depending on the loan category, the branch of the bank where the loan was issued and the month of the year of the start date.
- For every bank branch, compute the minimum interest rate and the maximum interest rate of the issued loans, depending on the loan purpose and the class of income of the borrower.
- Compute the number of loans and the average duration (in years) depending on the branch of the bank where the loan was issued and the number of children of the borrower.
- Compute the percentage of defaulted loans depending on the month of the end year and the city of the branch of the bank where the loan was issued.

What to do in the exercise

1. Produce the DFM schema of the data warehouse
2. Derive the Star schema corresponding to the DFM schema
3. Illustrate the ETL processes (in terms of queries over the reconciled data layer) that perform the first load of the tables in the Star schema
4. Write the query on the Star schema for computing the basic cube of the data warehouse
5. Write the queries on the Star schema for capturing the above-mentioned questions for building the data marts that will be built by the business analysts in order to start the relevant OLAP sessions

1. DFM schema

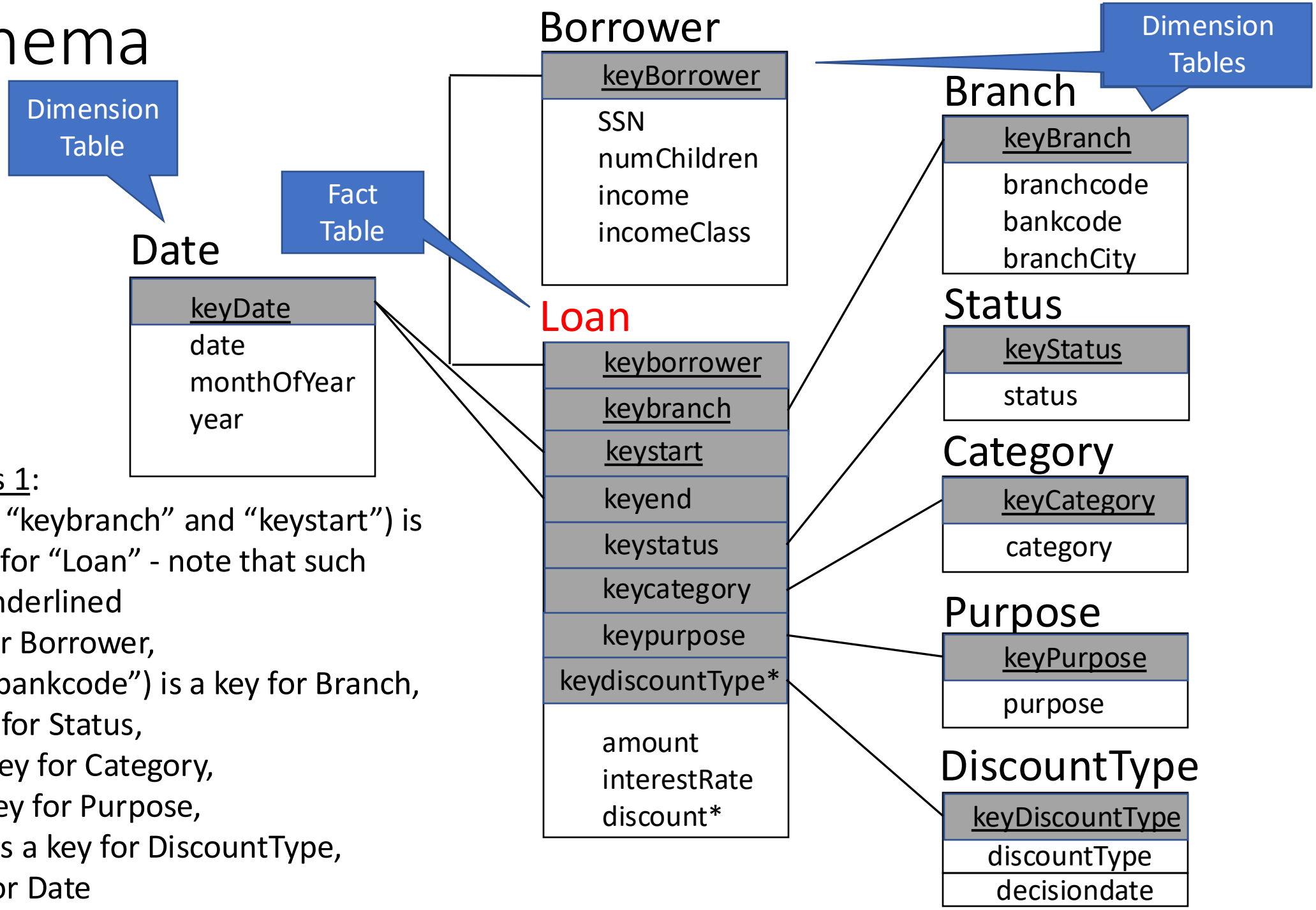


Note:
Symbol * associated to a measure means that some fact may exists with no value associated to that measure (e.g., “discount”).

Integrity constraints:

- “borrower”, “branch” and “start” form an identifier for “Loan”
- For every fact F of type Loan, the fact F cannot have a value for “discount” without a value for the dimension “discountType”.

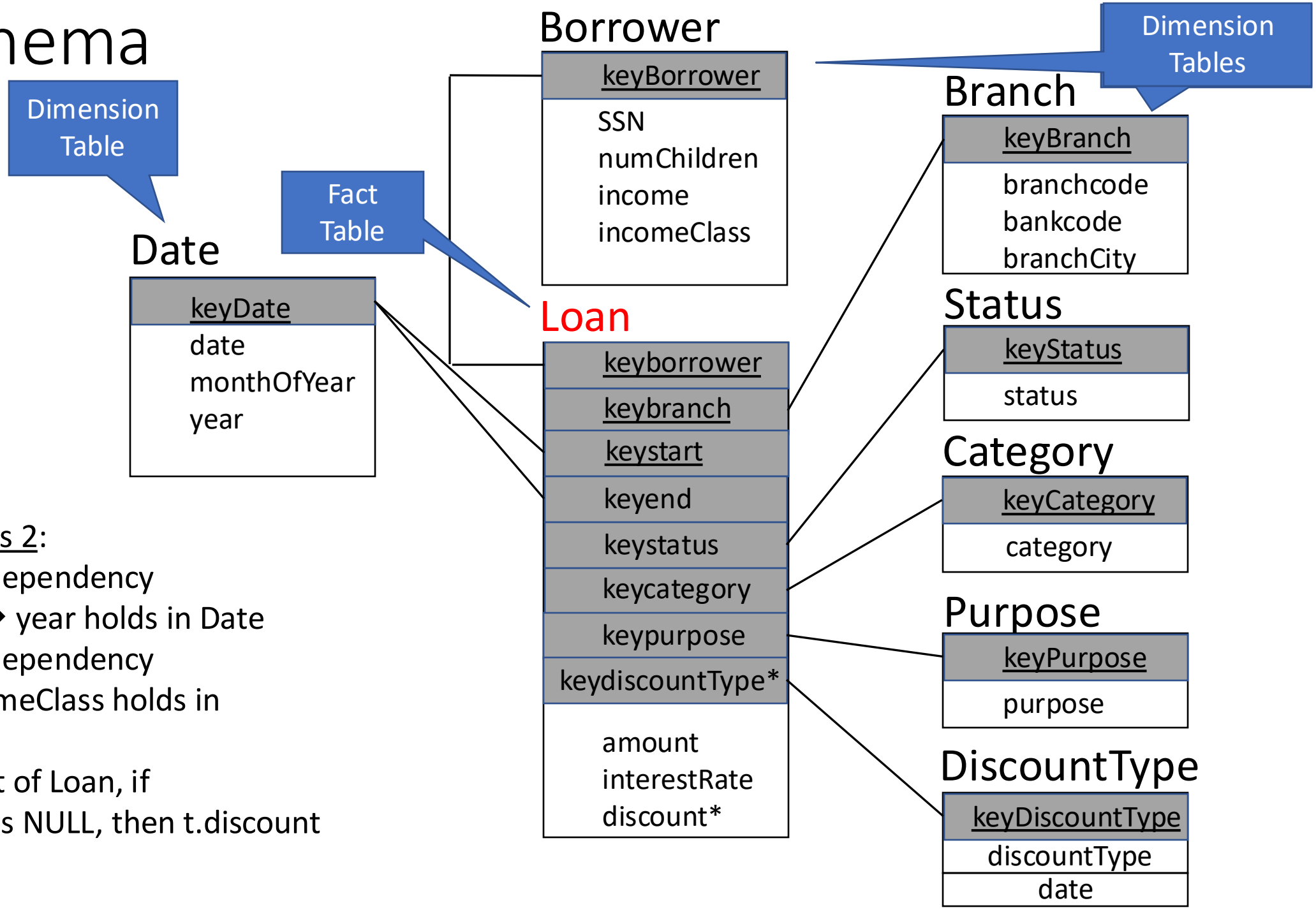
2. Star schema



Integrity constraints 1:

- (“keyborrower”, “keybranch” and “keystart”) is the primary key for “Loan” - note that such attributes are underlined
- “SSN” is a key for Borrower,
- (“branchcode”,”bankcode”) is a key for Branch,
- “status” is a key for Status,
- “category” is a key for Category,
- “purpose” is a key for Purpose,
- ”discountType” is a key for DiscountType,
- “date” is a key for Date

2. Star schema



Integrity constraints 2:

- the functional dependency monthOfYear → year holds in Date
- the functional dependency income → incomeClass holds in Borrower
- for every tuple t of Loan, if t.discountType is NULL, then t.discount is NULL

3. How to crate “surrogate keys” in SQL

A surrogate key in a Data Warehouse is a value that is invented for identifying objects or tuples in the tables of the Data Warehouse. In the following we assume that we define a surrogate key for every dimension table.

How to use SERIAL:

1. If you want to define a surrogate key for a table T in SQL, what you can do is to declare an attribute A (e.g., the first one) of type SERIAL in the “create table” for T.

2. Then, when you insert a new tuple in T, you simply specify no value for that attribute A, and the system will generate a value (in general, with autoincrement) identifying the tuples in the table T.

3. The ETL queries

Here is the SQL code for loading the dimension table Date:

```
insert into Date(keyDate,date,monthOfYear,year) as  --  keyDate declared as SERIAL
select date, month, year
from (select startdate as date,
                MakeMonth(startdate.month, startdate.year) as month,
                 startdate.year as year
        from Loan
        union
        select enddate as date,
                 MakeMonth(enddate.month, enddate.year) as month,
                 enddate.year as year
        from Loan
)
```

We assume that MakeMonth() is a function to build a monthOfYear as a pair (month,year)

3. The ETL queries

Here is the SQL code for loading the dimension table Borrower:

```
insert into Borrower(keyBorrower,SSN,income,incomeClass,numChildren) as
                -- keyBorrower declared as SERIAL
select keyBorrower, p.SSN, n.Qty as Income, n.IncomeClass, count(c.child) as numChildren
from Person p join HasIncome n on p.SSN = n.person join Loan a on n.date = a.startdate
        left join HasChild c on p.SSN = c.parent
group by p.SSN, n.Qty, n.IncomeClass
```

Here is the SQL code for loading the dimension table Branch:

```
insert into Branch(keyBranch,branchcode,bankcode,branchCity) as
                -- keyBranch declared as SERIAL
      select code, bank, city
      from Branch
```

3. The ETL queries

Here is the SQL code for loading the dimension table Status:

```
insert into Status(keyStatus,status) as -- keyStatus declared as SERIAL
select status
from Loan
```

Here is the SQL code for loading the dimension table Category:

```
insert into Category(keyCategory,category) as -- keyCategory declared as SERIAL
select category
from Loan
```

Here is the SQL code for loading the dimension table Purpose:

```
insert into Category(keyPurpose,purpose) as -- keyPurpose declared as SERIAL
select purpose
from Loan
```

Here is the SQL code for loading the dimension table DiscountType:

```
insert into DiscountType(keyDiscountType,discountType,date) as -- keyDiscountType declared as SERIAL
select d.Type, d.date
from HasDiscount h join DiscountDecision d on h.dcode = d.code
```

3. The ETL queries

Here is the SQL code loading the fact table Loan (using both tables in the reconciled layer, and the dimension tables):

```
insert into Loan(keyborrower,keybranch,keystart,keyend,keystatus, keycategory,keypurpose,
                    keydiscountType,amount,interestrate,discount) as
select b.keyBorrower, r.keybranch, d1.keydate, d2.keydate, s.keystatus, c.keycategory,
       p.keypurpose, dd.Type, a.amount, a.rate, dd.amount
from Loan a join Date as d1 on a.startdate = d1.date join Date d2 on a.enddate = d2.date
     join  Borrower b on a.borrower = b.borrower join Branch r on a.branchcode = r.branchcode
     join  Status s on a.status =  s.status join Category c on a.category = c.category
     join Purpose p on a.purpose = p.purpose  left join
     (HasDiscountType h left join DiscountDecision dd on h.code = dd.code)
     on a.borrower = h.borrower and a.branchcode = h.branchcode and a.bankcode = h.bankcode
          and a.startdate = h.startdate
```

4. The query for building the basic cube

Here is the SQL code corresponding to such query (left joins are used to cope with loans not having discount information):

```
select *
from Loan a join Date as d1 on a.startdate = d1.date
        join Date d2 on a.enddate = d2.date
        join Borrower b on a.borrower = b.borrower
        join Branch r on a.branchcode = r.branchcode
        join Status s on a.status =  s.status
        join Category c on a.category = c.category
        join Purpose p on a.purpose = p.purpose
        left join  (HasDiscountType h left join DiscountDecision dd on h.code = dd.code)
                    on a.borrower = h.borrower and a.branchcode = h.branchcode and
                        a.bankcode = h.bankcode and a.startdate = h.startdate
```

Example of OLAP query

We consider the first query: *Compute the average interest rate, depending on the loan category, the branch of the bank where the loan was issued and the month of the year of the start date.*

Here is the SQL code corresponding to such query:

```
select c.category, b.branchcode, b.bankcode, d.monthOfYear, avg(a.interestRate)
from Loan a, Category c, Branch b, Date d
where a.keycategory = c.keyCategory and a.keybranch = b.keyBranch and
        a.keystart = d.keyDate and a.keyborrower = b.keyBorrower
group by d.monthOfYear, c.category, b.branchcode, b.bankcode
```