Homework #1

Matteo Corain S256654

Data Science and Database Technology – A.Y. 2018-19

1 Conceptual design

1.1 Facts and measures identification

The requirements state that the company for which the data warehouse is designed is interested in analyzing the revenues and the number of tickets purchased by their customers. Ticket sales are therefore the relevant fact for the analysis, characterized by the two values "revenues" and "number of tickets": this can be modeled by introducing, in the DFM schema, a `TicketSales` fact, characterized by the two measures `Revenues` and `NumberOfTickets`.

1.2 Dimensions identification

The requirements state that the company for which the data warehouse is designed is interested in performing the analysis on:

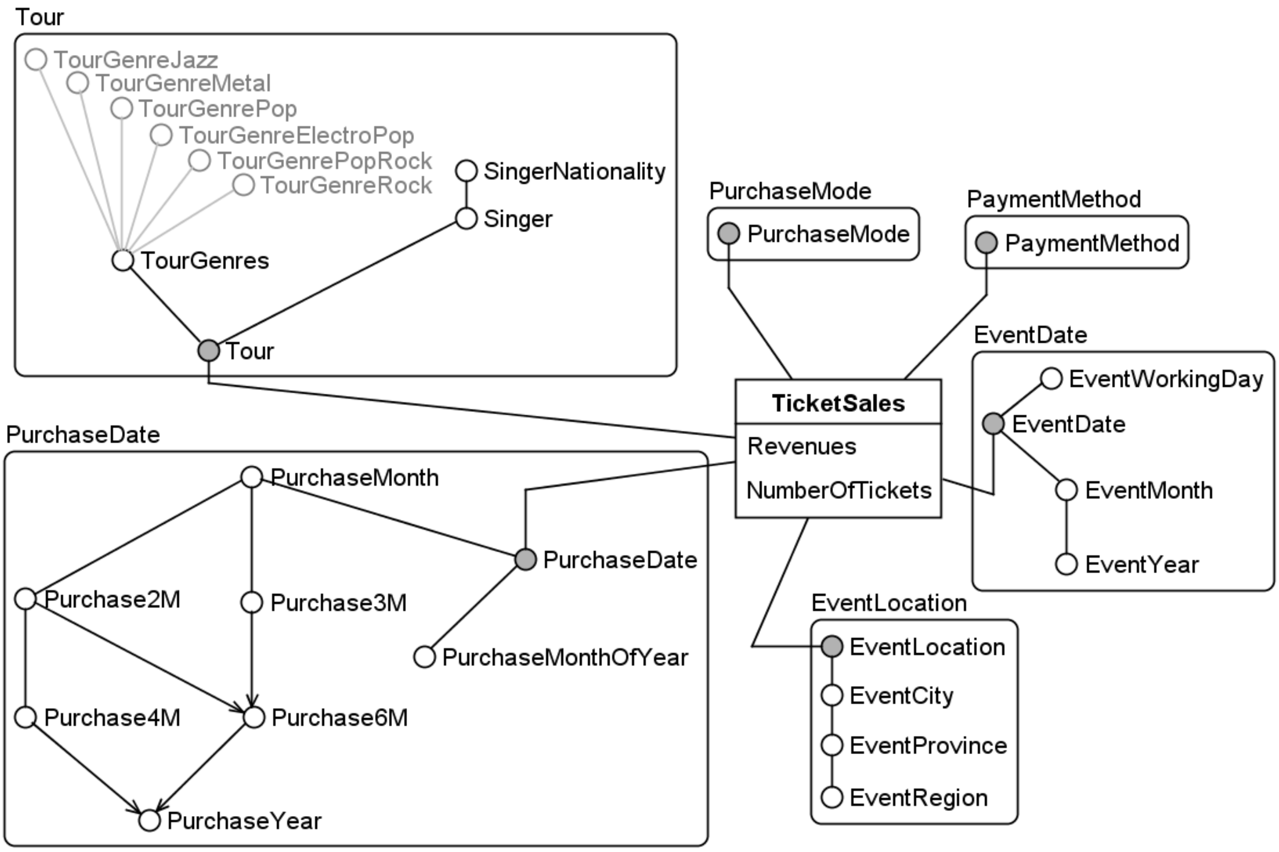
- Tour name and musical genres: this can be achieved by introducing a `Tour` hierarchy, made up of two dimensional attributes: the base `Tour` attribute and the `TourGenres` configuration attribute (see the following paragraph for more information on this particular modeling choice);
- Singer and nationality: since every tour is bound to a specific singer/band, this can be achieved by introducing a `Singer` attribute and a `SingerNationality` attribute in the `Tour` hierarchy;
- Event location, city, province and region: this can be achieved by introducing an `EventLocation` hierarchy, made up of the four dimensional attributes `EventLocation`, `EventCity`, `EventProvince` and `EventRegion`;
- Event date, month, year and working day/holiday: this can be achieved by introducing an `EventDate` hierarchy, made up of the base `EventDate` attribute and of two sub-hierarchies:
  - The first is made up of the two dimensional attributes `EventMonth` and `EventYear`;
  - The second is made up of the dimensional attribute `EventWorkingDay`.
- Purchase date, month, month of the year, two-month period, three-month period, four-month period and year: this can be achieved by introducing a `PurchaseDate` hierarchy, made up of the base `PurchaseDate` attribute and of two sub-hierarchies:
  - The first one is a complex hierarchy that organizes the attributes `PurchaseMonth`, `Purchase2M`, `Purchase3M`, `Purchase4M`, `Purchase6M` and `PurchaseYear` in a subtree to comply with the inclusion property of the functional dependencies (time period multiples);
  - The second one is made up of the `PurchaseMonthOfTheYear` attribute.
- Purchase mode: this can be achieved by introducing a degenerate `PurchaseMode` hierarchy made up of the single dimensional attribute `PurchaseMode` (with its own peculiar domain);
- Payment method: this can be achieved by introducing a degenerate `PaymentMethod` hierarchy made up of the single dimensional attribute `PaymentMethod`.

1.3 Musical genres modeling

Some care should be used to model the musical genres dimension. The requirements state in fact that a single tour may belong to one or more musical genres, but at the same time the list of possible genres is known and of limited size. This means that there is no need to introduce a multiple edge in the DFM schema: that would have been necessary in case the list of possible genres is not known at design time, but, since it is, the problem can be solved using the techniques for configuration modeling. For this reason, in the DFM schema a single configuration attribute called `TourGenres` has been added in the `Tour` hierarchy; this attribute may assume, at the same time, multiple different values from the list of known items (rock, pop rock, electropop, pop, metal, jazz). Those elements are reported in grey in the DFM schema, since they do not represent additional aggregation levels, but just possible values for the multi-value attribute.

1.4 Final conceptual schema

The final DFM schema obtained by modeling the problem following the aforementioned procedure is reported below.



2 Logical design

2.1 Dimension tables

One table has been introduced for every non-degenerate dimension, meaning every dimension for which the hierarchy is not limited to a single level, including all the attributes of the hierarchy plus a surrogate primary key (the field noted as ID) to be used for the join operations with the fact table. The degenerate dimensions have been instead pushed back to the fact table in order to reduce the number of useless join operations to be performed to rebuild the OLAP hypercube.

```
D_EVENTDATE(EventDate_ID, EventDate, EventWorkingDay, EventMonth, EventYear)
D_EVENTLOCATION(EventLocation_ID, EventLocation, EventCity, EventProvince, EventRegion)
D_PURCHASEDATE(PurchaseDate_ID, PurchaseDate, PurchaseMonthOfYear, PurchaseMonth, Purchase2M, Purchase3M, Purchase4M, Purchase6M, PurchaseYear)
```

For the translation of the `Tour` dimension, containing the configuration attribute `TourGenres`, an ad-hoc procedure should be used. That conceptually single attribute, in fact, needs to be split at the logical level into a number of different attributes, each one of them relating to a single music genre, to be interpreted as boolean value (the column relative to a genre is true if the corresponding tour belongs to that genre). Those attributes, contrarily to standard dimensional attributes, are not linked by any functional dependency, so they can assume any value for any object. In this way, the creation of an additional table is avoided, at the cost of a possible redundancy increase.

`D_TOUR(Tour_ID, Tour, Singer, SingerNationality, TourGenreRock, TourGenrePopRock, TourGenreElectroPop, TourGenrePop, TourGenreMetal, TourGenreJazz)`

2.2 Fact tables

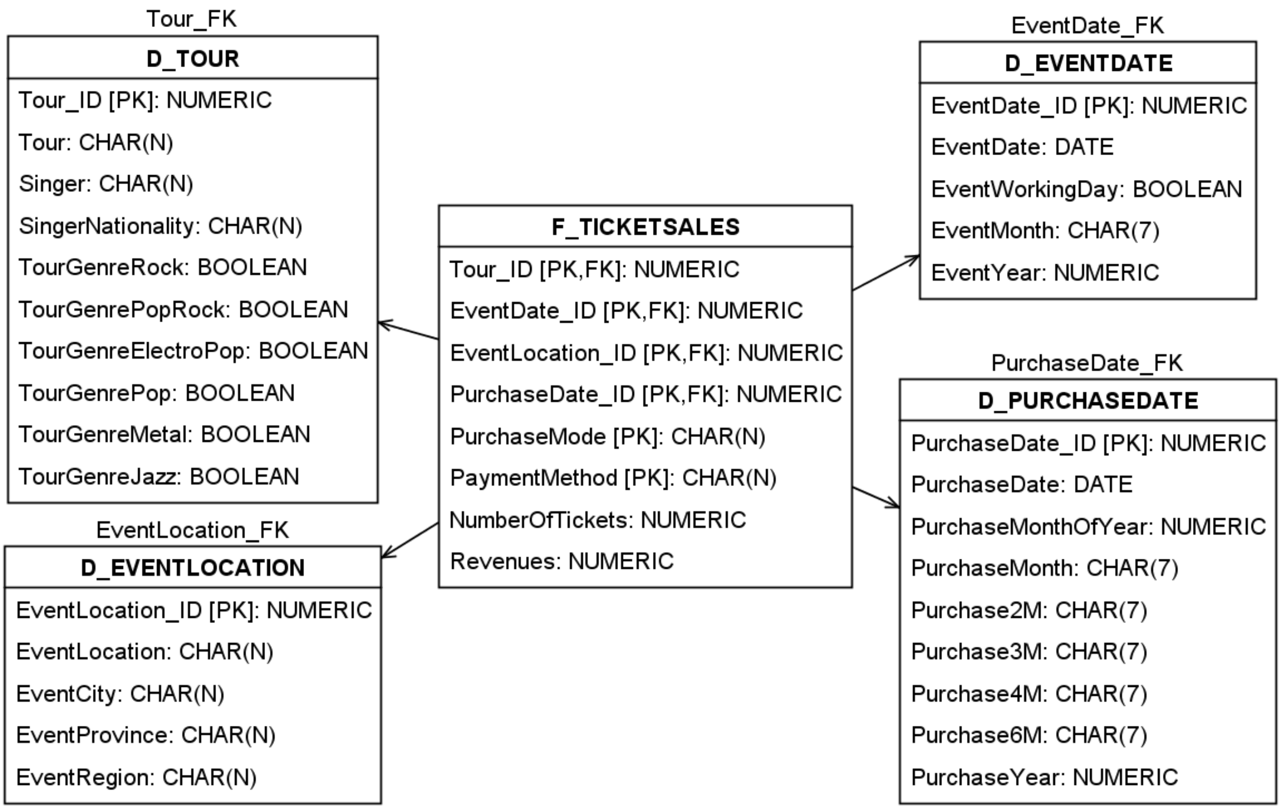
A single fact table has been introduced for the single fact of interest identified in the DFM schema, having as attributes the measures identified in the conceptual design phase and as primary key:

- All the surrogate keys of the non-degenerate dimensions, linked to the fact table by using a foreign key constraint;
- All the values of the single attributes in the degenerate dimensions.

`F_TICKETSALES(Tour_ID, EventDate_ID, EventLocation_ID, PurchaseDate_ID, PurchaseMode, PaymentMethod, NumberOfTickets, Revenues)`

2.3 Final logical schema

The final logical schema, which include also a tentative data type definition for every field of the tables, is reported below.



3 Queries

3.1 Query #1

Separately for each purchase mode and for each purchase month, analyze: the average daily revenue; the cumulative revenue from the beginning of the year; the percentage of tickets related to the considered purchase mode over the total number of tickets of the month.

```
SELECT      PurchaseYear,
            PurchaseMonth,
            PurchaseMode,
            SUM(Revenues)/COUNT(*)
                AS AvgDailyRevenue,
            SUM(SUM(Revenues)) OVER (
                PARTITION BY PurchaseYear, PurchaseMode
                ORDER BY PurchaseMonth
                ROWS UNBOUNDED PRECEDING
                ) AS RevenueFromBeginningOfYear,
            SUM(NumberOfTickets)/SUM(SUM(NumberOfTickets))
                OVER (PARTITION BY PurchaseMonth) * 100
                AS PercentageOverMonth
FROM        F_TICKETSALES TS, D_PURCHASEDATE PD
WHERE       TS.PurchaseDate_ID = PD.PurchaseDate_ID
GROUP BY    PurchaseMode, PurchaseMonth, PurchaseYear
ORDER BY    PurchaseYear, PurchaseMonth, PurchaseMode;
```

3.2 Query #2

Considering the events that took place in 2017, separately for each singer/band nationality and for each city, analyze: the average revenue for a ticket; the percentage of revenue over the total revenue for the corresponding province; the percentage of revenue over the total revenue for the corresponding region.

```
SELECT      SingerNationality,
            EventCity,
            EventProvince,
            EventRegion,
            SUM(Revenues)/SUM(NumberOfTickets) AS AvgTicketRevenue,
            SUM(Revenues)/SUM(SUM(Revenues))
                OVER (PARTITION BY SingerNationality, EventProvince) * 100
                AS PercentageOverProvince,
            SUM(Revenues)/SUM(SUM(Revenues))
                OVER (PARTITION BY SingerNationality, EventRegion) * 100
                AS PercentageOverRegion
FROM        F_TICKETSALES TS, D_EVENTDATE ED,
            D_TOUR T, D_EVENTLOCATION EL
WHERE       TS.EventDate_ID = ED.EventDate_ID
AND         TS.Tour_ID = T.Tour_ID
AND         TS.EventLocation_ID = EL.EventLocation_ID
AND         EventYear = 2017
GROUP BY    SingerNationality, EventCity, EventProvince, EventRegion
ORDER BY    SingerNationality, EventCity, EventProvince, EventRegion;
```