UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences
Département d'Informatique

INFO-H-419 Data Warehouses

Assignment - Part I

Teacher :

Toon Calders

Assistant :

Hatem Haddad

Group E members :

Raymond Lochner (000443637)

Patrick Randriambololona (000444674)

Aldar Saranov (000435170)

Antoine Vandevenne (000333030)

Academic year 2017 - 2018

Contents

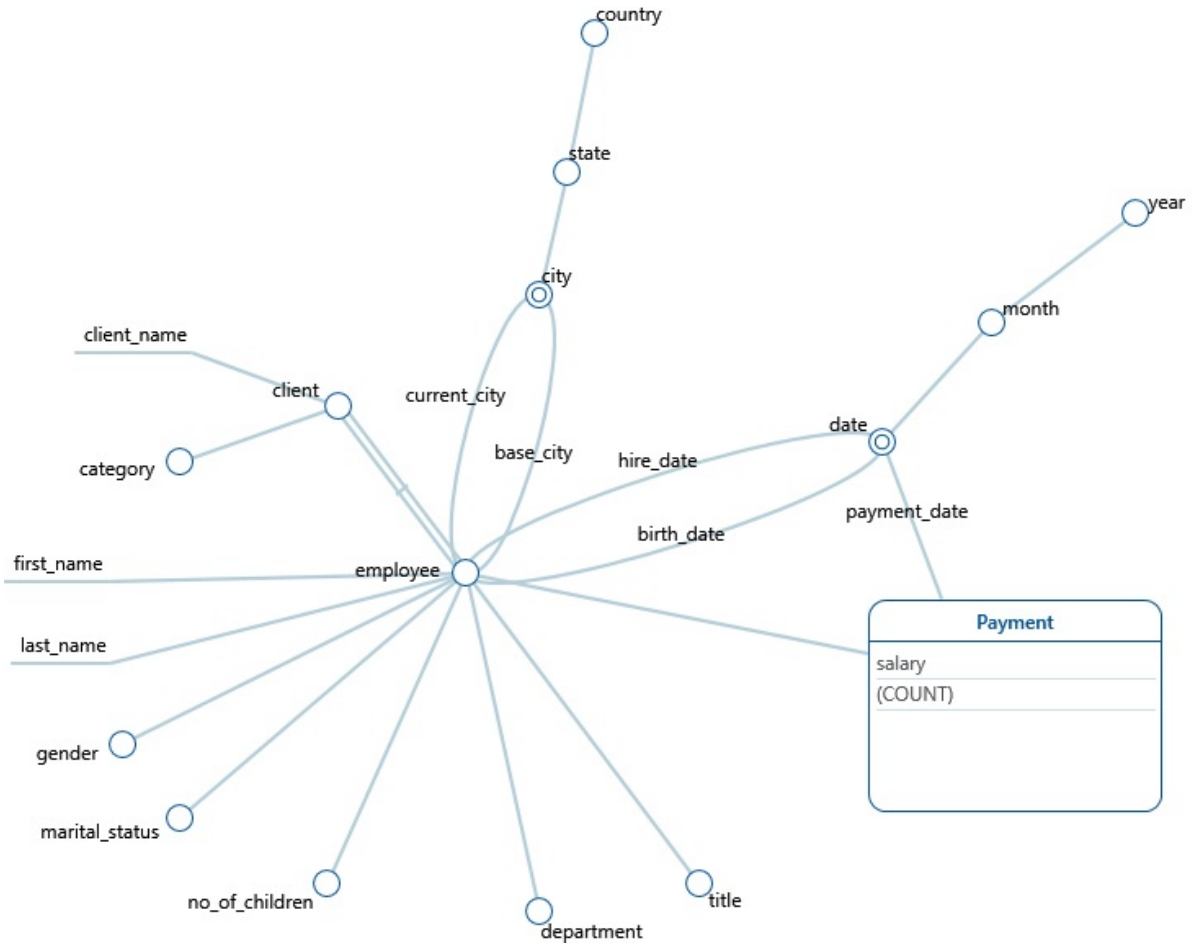

Figure 1: Dimensional Fact Model

1 Dimensional Fact Model

The first step of the conception of the Dimensional Fact Model (DFM) consists in choosing on which fact we will be working on, along with the related measures. Our fact could be stated as follows : "An employee whose ID is *sEmployeeID* was paid an amount of *salary* on *payment_date*. Because of the kind of operations that we are expected to run (Average/Sum), we choose the monthly salary, the date of payment and the (COUNT) operation as measures. Although it would have been possible to define our fact as a period defined by a start date and an end date, we choose opted for the simplest worked choice, that is a single point in time.

Thanks to the choice of the fact, all major aggregations can be anticipated in future, i.e. Min, Max, Average and Sum. The (COUNT) operation is meant to help executing the query №3, which returns employee distribution over certain levels of the geographical dimension.

We identified a convergence in the DFM for the *Date* and *City* hierarchies, respectively shared by *hire_date*, *birth_date* and *payment_date* dimensions, and *base_city* and *current_city* dimensions.

Beside those dimensions, there are several other mandatory dimensions related to *employee*: *title*, *department*, *gender*, *no_of_children*, *marital_status*.

Relation Employee-Customer is marked as many-to-many, since one employee can serve several clients and one client can be served by several employees. This relation dimension is optional since an Employee could not serve any client.

No incomplete hierarchies or cross-dimensional attribute are supposed in this model. One could have chosen the *state* level as optional but, as no row in the data-set showed a missing state, we assumed that it is a mandatory dimension. We identified the name of the client and the first name and last name of an employee as descriptive attributes because it is required to store them in the database but they cannot be used for aggregation.

Given the mentioned set of potential queries, at least the following data cubes are required:

- (title, departments, payment_date) for query 1
- (gender, payment_date) for query 2
- (city, payment_date), (state, payment_date), (country, payment_date) for query 3
- (client, payment_date) for query 4
- (title, payment_date) for query 5

And certainly the OLAP service should store the primary grouping set. Additional data cubes can also be the ones that need to be materialized, depending on the new requirements.

2 Anticipating potential changes

Before discussing about which case requires which updating system, we define our update policy as follow:

- Type-1 updates are used to correct mistakes or unexpected changes (as the gender).
- Type-2 updates are used to perform updates on records that are subject to change and for which we are interested in keeping the history of the changes. More specifically, we use the Type-2B system in order to keep the start and end dates of the validity period of a fact in the record table. Moreover, notice that the valid flag is implied when the *end* column is not defined or null. Thus, it is not necessary to include it.

Given that employees are expected to see their professional (title, department, salary) and personal (marital status, number of children) change, the *employee* table require a Type-2B updating system.
In addition, clients may move to another category (small/medium/large) over time. As we want to be able to analyze old salaries in regard to the category of each client at a given time, we also use Type-2B updating system for clients.
Any other record will be managed by Type-1 updating system.

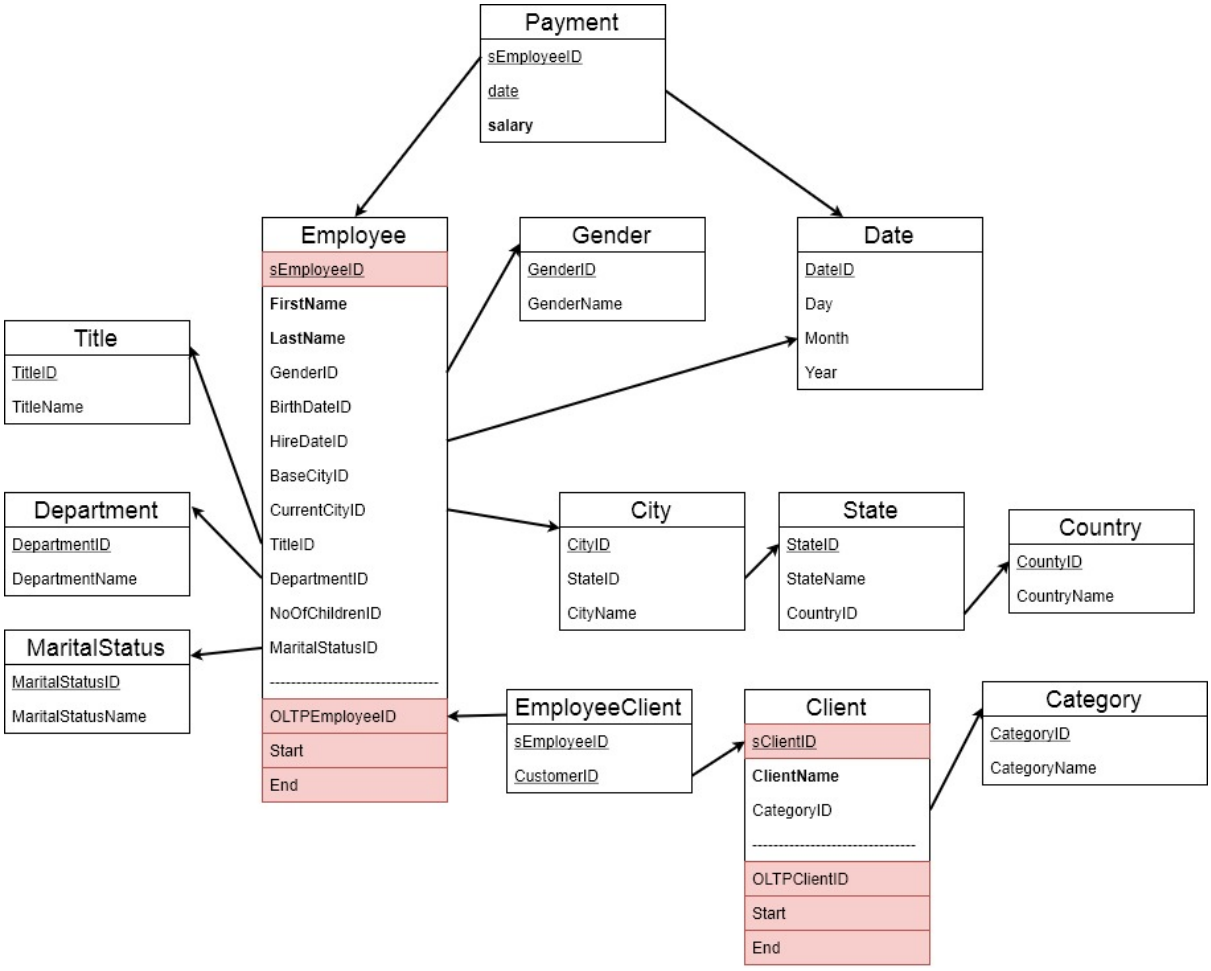

Figure 2: Relation Model

3 Relational Model

As detailed on the model, we decided to establish the following tables: *Employee*, *Title*, *Department Payment*, *Gender*, *City*, *State*, *Country*, *Date*, *EmployeeClient*, *Client*, *Category*.

As *no_of_children* is an integer value, no table is required to identify the number of children that an employee has.
Similarly for the *Date*, it is not necessary to assign IDs to each year or month. However, we do define a table for *Date* in order to be able to roll up on it.

A bridge is used as it is a many-to-many relation. *EmployeeClient* table is composed of two foreign surrogate keys, namely *sEmployeeID* and *sClientID*, and thus a row of this table represents a single relation between a concrete employee version and a client version.
If we register changes either in Employee or in Client tables, we would need to add a new version of the row of corresponding table and copy its corresponding relations from

EmployeeClient by substituting the foreign key to the table that is changed.
If we need to change a relation between two existing employee and client, we add one new version of both employee and client, and new version of *EmployeeClient* if the relation has been established; or we add nothing there if the relation has been destroyed.