Data Warehousing Assigment - Part 2 Report

Abbas Al-Eryani 000439330 | Geo Jolly 000439156 | Alberto Parravicini 000437917 | Almas Zhanibek 000439431

1 Notes about the files

- "Package.dtsx" is the **ETL** package. For completeness, the entire *integration* project is provided, in the "Assignment2" folder.
- The **ETL** phase doesn't use any external connection or flat files beside *source* and *target*, i.e. the provided **OLTP** and **OLAP** databases. Note that the **OLAP** database has to conform with what is contained in the next section for the script to work.
- The cube is contained in the "CubeFinal" folder.

2 Notes about the Data Warehouse Schema

As starting point of the second assignment, we have been provided with an *SQL script* to create the tables of the data warehouse. However, the script had some issues that had to be fixed before starting the **ETL procedure**:

- Tables *actor, address, category, city, country, film, inventory, language, rental* were set to have **auto-incrementing keys**: however, these tables do not use *surrogate keys*, and their keys are the same of the oltp database. As such, it was necessary to remove the *auto-increment* setting.
- Table *date* was not set to have an *auto-incrementing* key. Given that the tuples of this table have to be loaded from multiple sources, having an auto-incrementing key is definitely a good idea to simplify the data-warehouse management. We decided to modify the table and add an *auto-incrementing* key.
- The cyclic *foreign key* constraint between the tables *store* and *staff* (relatively to the store manager, and the store where he works) proved to troublesome to handle. Indeed, it is not possible to add a new store without previously having added its manager, and viceversa it's not possible to add a manager without adding his store beforehand.
  To deal with this problem during the **ETL** p, it was necessary to temporarily disable these two *foreign key* constraints. More details in the specific section.

3 ETL

This section includes pictures of the steps of the ETL procedure, with comments added where needed.
Note that for brevity some data-flows are not reported, in case the data-flow simply transfers data from the **OLTP** database to the data warehouse.

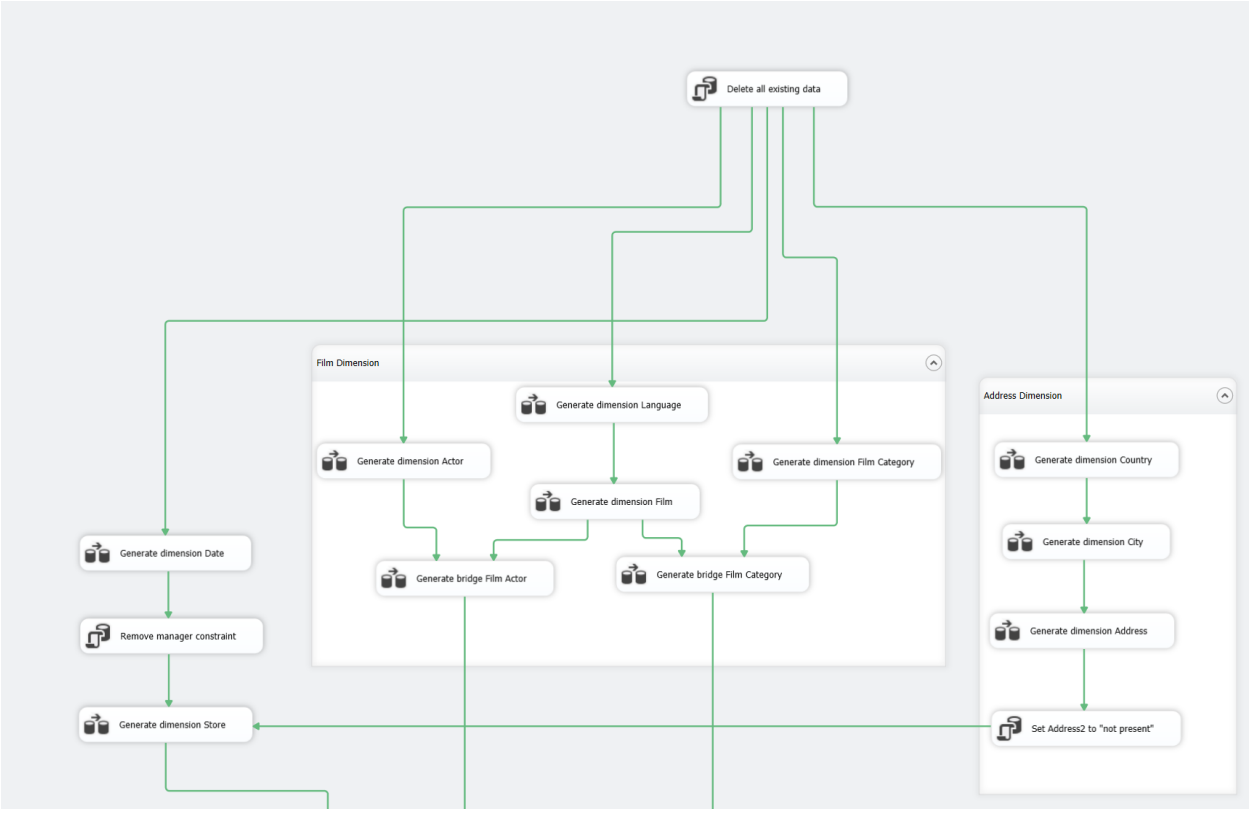


Figure 1: This phase loads the data for *film* and *address*, and the related tables. At the beginning, all data are erased: if the data-warehouse is already empty, this part is redundant; we added it so that we could more easily test the *ETL* script.
Before generating the dimensions *store*, the *foreign key* constraint *[FK_store_staff]* is removed.

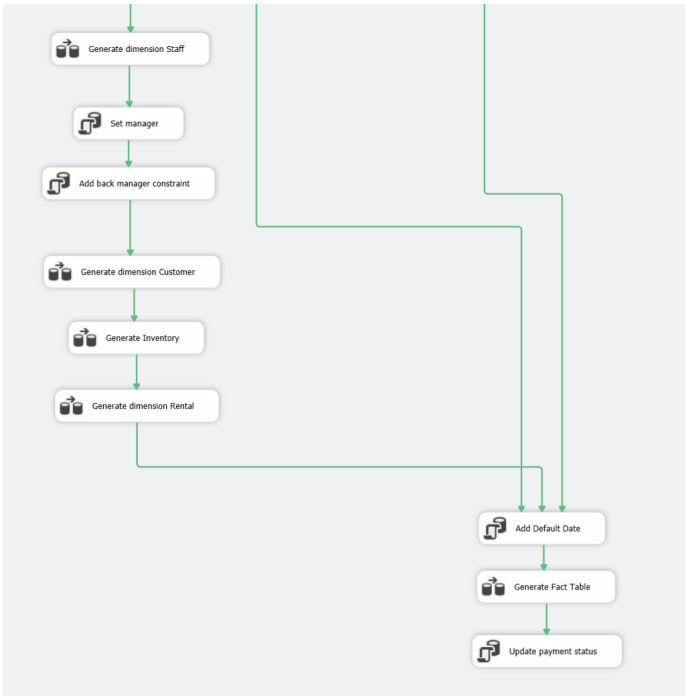
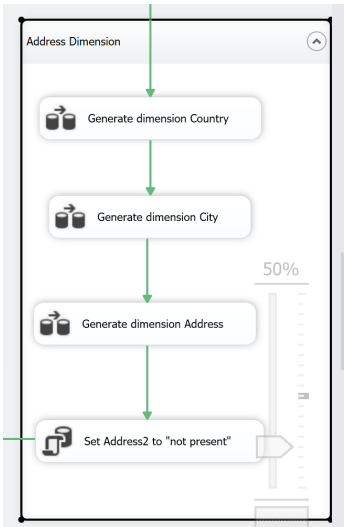
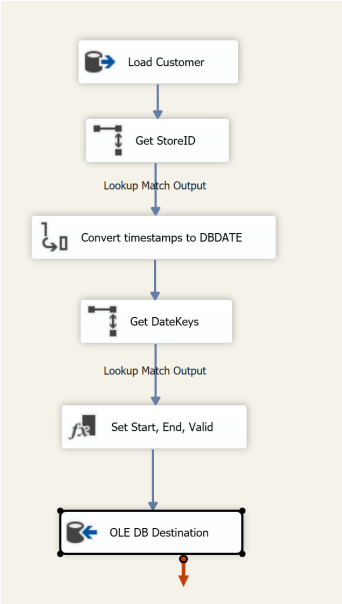


Figure 2: This phase loads the data of the other dimensions, and populates the fact table.
After populating the table *staff*, the constraint *[FK_store_staff]* is restored. Before populating the fact table, the default date *1900-01-01* is added to the database (see the specific section for more details). As last step, rentals without payment have the attribute *payment_status* set to *false*.

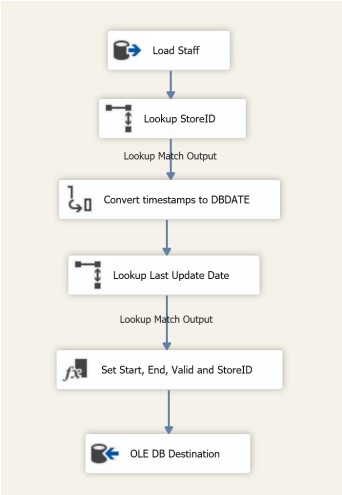
• **Note**: the data-flow for the film dimensions is trivial and is not reported.



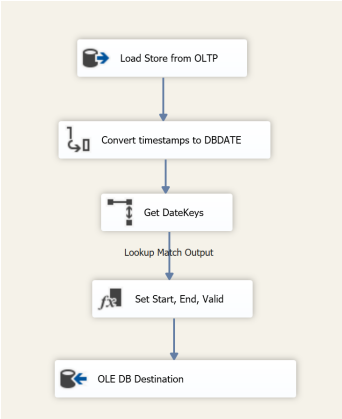
(a) Loading of *address*. At the end, empty values of *address2* are set to *not_present*.



(b) Loading of *customer*



(a) Loading of *staff*



(b) Loading of *store*

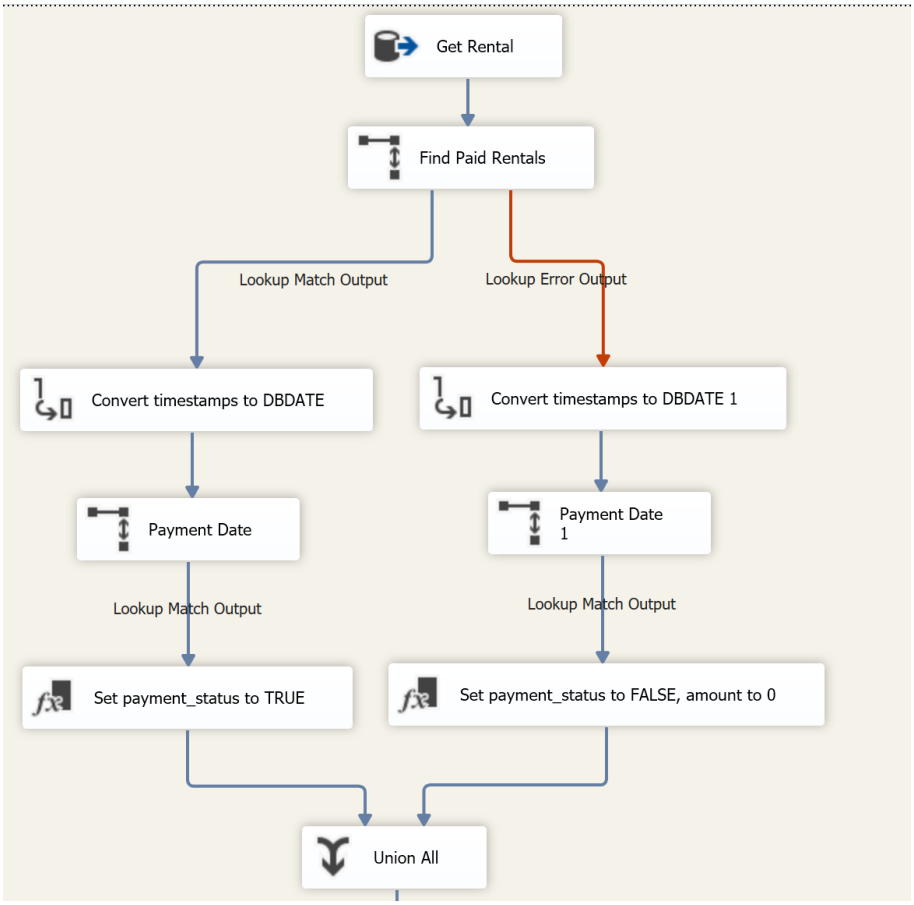


Figure 5: Loading of the *fact table*

- First of all, data are loaded from the *rental* table; then a lookup is done on the *payment* table, by comparing **rental_id**.
- If a match is found, the rental was paid. In this case, the values of *customer, staff* are updated to be the one in the *payment* table. In fact, the customer and the staff who performed a rental could be different from the ones who perform the payment. The data warehouse could have been designed to hold both couples, where available, but in our case only one couple can be kept.
- If no match was found, the rental was not paid: in this case, *amount* is set to 0, and the customer and staff who performed the rental are stored (as it is not possible to have *null* values).
- If no match was found, the *payment date* is temporarily set to the *rental date*. Later, it is substituted with the default value *1900-01-01*, as it's not possible to use *null* values.

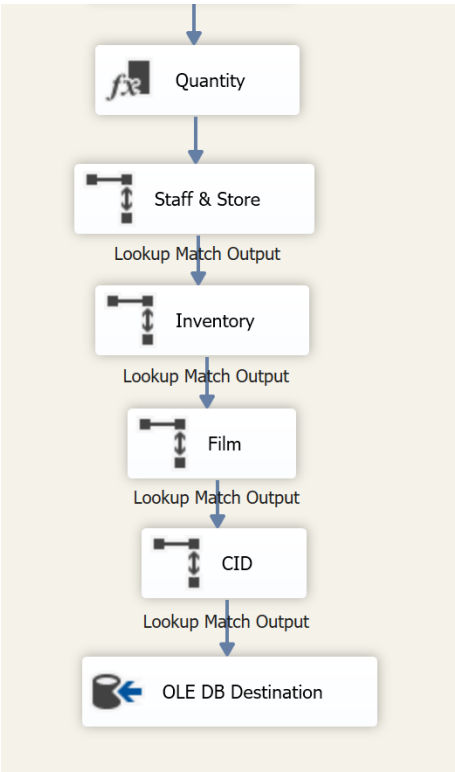


Figure 6: Loading of the *fact table*

- From the description, it is not clear what the *quantity* measure is.
  From the **OLTP** schema it is possible to infer that each payment refers to just one copy of a movie, so we assumed that *quantity* is simply used to count the number of facts. As such, it is set to 1 for all tuples.
- Then, all the keys of the fact table are loaded.

4 Data-Cube

This section includes an explanation about the structure of the *data-cube*, with pictures and comments where needed.

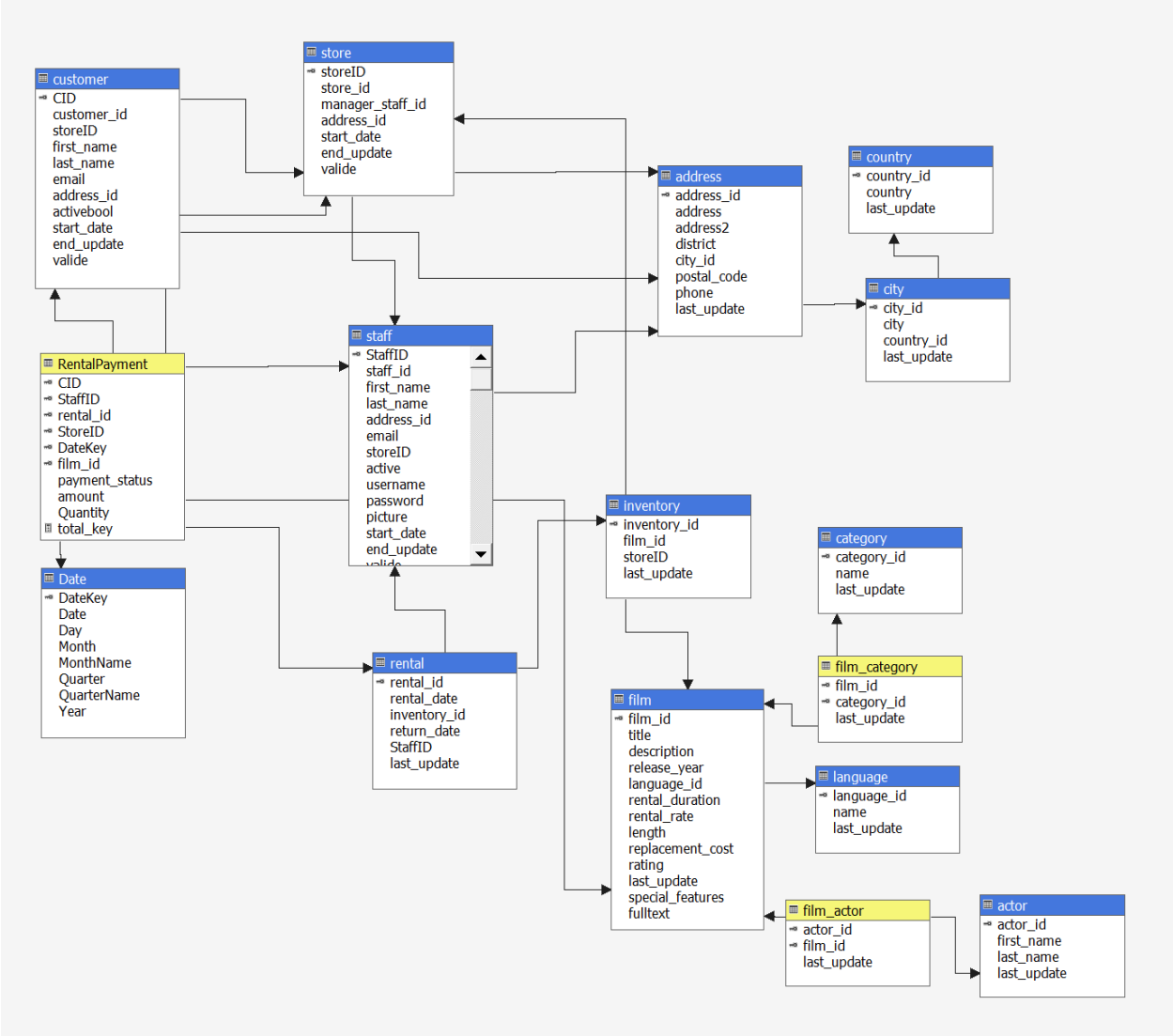
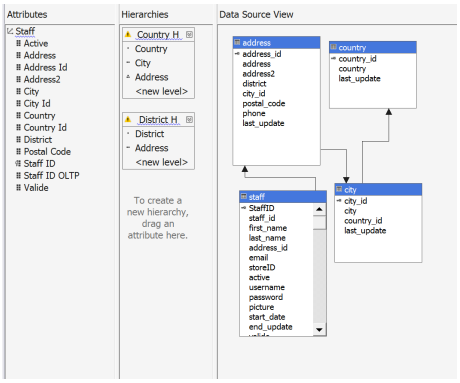
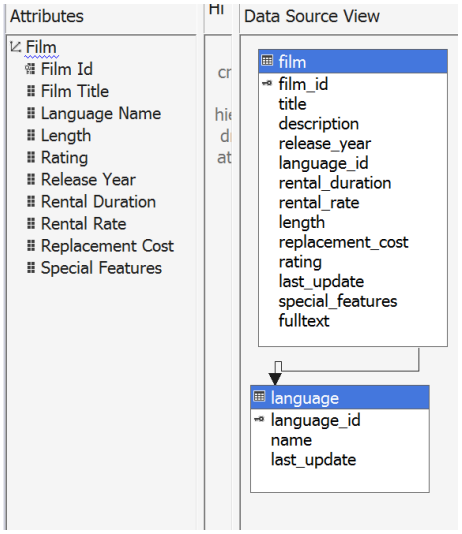


Figure 7: Loading of the *Data-Cube*. In yellow, the *fact table* and the *bridge tables*. *total_key* is the overall key of the *fact table*, and is used by the *payment status* dimension.

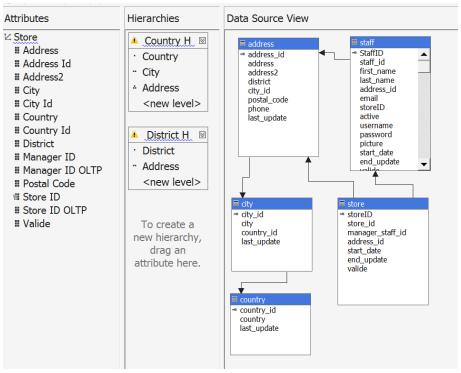
4.2 *Dimensions*



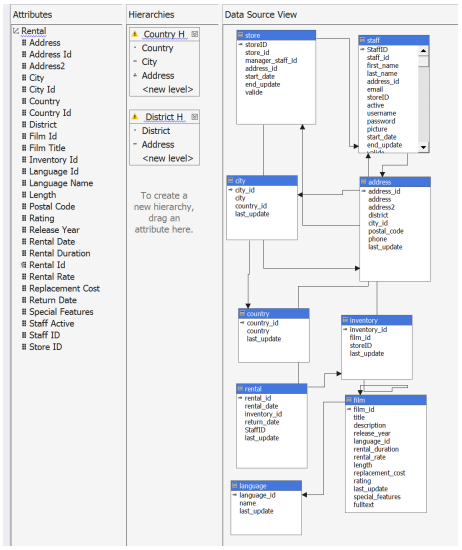
(a) *Staff* dimension



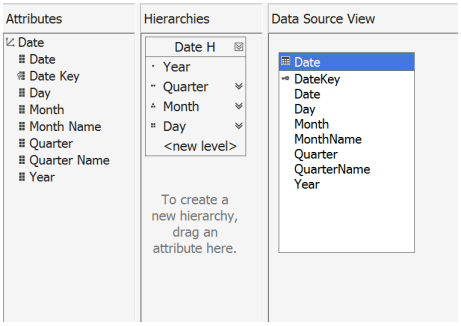
(b) *Film* dimension



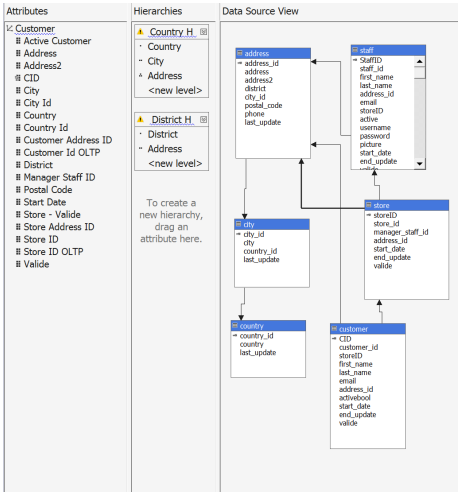
(a) *Store* dimension



(b) *Rental* dimension



(a) *Date* dimension



(b) *Customer* dimension

• Not shown, the supporting *Category* and *Actor* dimensions, connected to *Film* through bridge tables. Also not shown, the *Payment Status* dimension, which allows to aggregate distinguishing whether a rental was paid or not.

4.3 *Attribute Relations*

What follows are two examples of *attribute relationships* structures. Other dimensions follow a similar pattern, and are not shown for brevity.

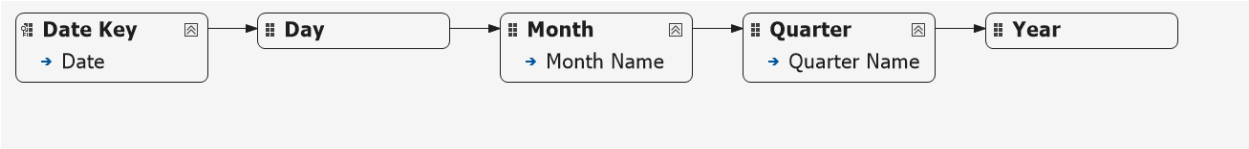


Figure 11: Attribute relationship of the dimension **Date**. Note that the hierarchy is built on the numerical values, but when displaying, the textual names for month and semester are used. Also, level of the hierarchy use composite keys: as an example, *month* has both *month* and *year* as key.

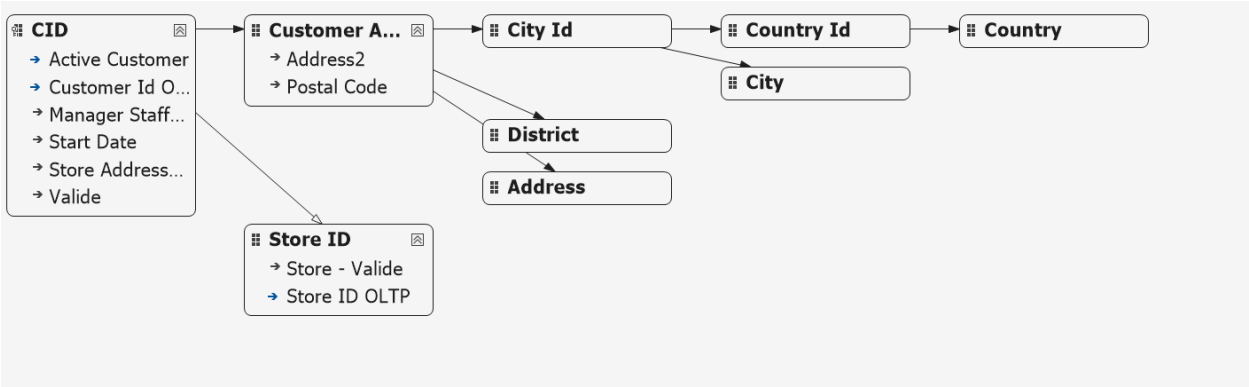
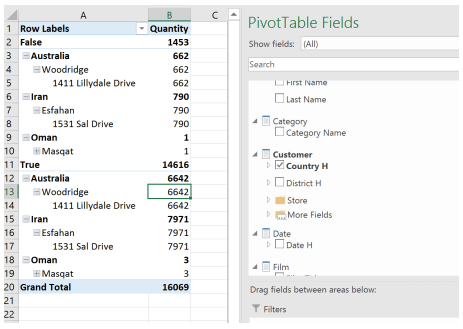


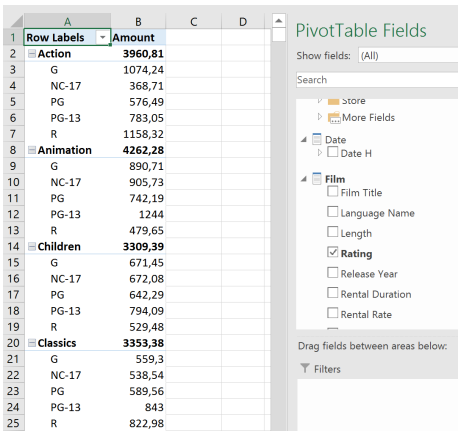
Figure 12: Attribute relationship of the dimension **Customer**. Dimensions *Rental*, *Store* and *Staff* have a similar structure. Note that the *valide* attribute is kept so that it is possible to aggregate only using the values that are currently valid.

4.4 *Cube Usage Examples*



Row Labels	Quantity
False	1453
Australia	662
Woodridge	662
1411 Lillydale Drive	662
Iran	790
Esfahan	790
1531 Sal Drive	790
Oman	1
Masqat	1
True	14616
Australia	6642
Woodridge	6642
1411 Lillydale Drive	6642
Iran	7971
Esfahan	7971
1531 Sal Drive	7971
Oman	3
Masqat	3
Grand Total	16069

(a) Number of rentals, divided by *paid* and *not paid*, and by *country*, *city*, *address*.



Row Labels	Amount
Action	3960,81
G	1074,24
NC-17	368,71
PG	576,49
PG-13	783,05
R	1158,32
Animation	4262,28
G	890,71
NC-17	905,73
PG	742,19
PG-13	1244
R	479,65
Children	3309,39
G	671,45
NC-17	672,08
PG	642,29
PG-13	794,09
R	529,48
Classics	3353,38
G	559,3
NC-17	538,54
PG	589,56
PG-13	843
R	822,98

(b) Rental income divided by *film category* and *film rating*.