Build the data warehouse

Simone Di Luna 544322

The data mart was designed to have a star schema (see figure 1): the fact table at the center describing facts about CPU sales (`cpu_sales_fact`) and four dimension tables surrounding it, each having a one-to-many relationship with the fact table, which provide the context for analyzing sales (`cpu_product, geography, time_by_day, vendor`).

To design the schema and establish the various constraints, a preliminary analysis was conducted to ascertain the domain of the variables, their data types, sizes, the presence of missing values, the form in which they occur and their frequencies. In the case of qualitative attributes, the strings were also tested to understand their character encodings; this was useful to differentiate between `char`/`varchar` and `nchar`/`nvarchar` data types in MS SQL Server. The report of the complete analysis can be viewed in the `preliminary_analysis_report.txt` file.

The size of the qualitative attributes were determined by the max length of the strings in the column incremented by 1/4 to have a margin in case the data mart is updated. However, this offset was not applied to those attributes for which we know for sure the number of characters they are made of, like `continent, day_name, the_quarter, currency`.

On the other hand, the domain of the numerical attributes helped in choosing between `int`, `smallint`, and `tinyint` data types.

The minimum and maximum values of the numerical variables also allowed us to assess the semantic accuracy of the measures in the fact table. Although all the measures in the `fact` file have positive values, it was decided to impose a *default constraint* of 0 on these attributes to cope with possible anomalies that might occur during future updates of the data mart (Albano and Ruggieri 2015, p. 60).

The default constraint was also applied to the required new attribute, *cost*, which was computed with the formula $C = \frac{R}{k+1}$, where C is the estimated cost, while R and k are the revenue and the markup, respectively. In turn, this formula was retrieved from that of the markup: $k = \frac{R-C}{C}$. The markup was randomly drawn from a uniform distribution and then constrained to take values between 15% and 40% through the min-man normalization technique.

In the fact table, all of the missing values occur as empty strings. Moreover, as stated in the assignment description, they affect only the variables `cpu_code`, `gpu_code`, and `ram_code`. Specifically, when one of these three attributes takes a non-missing value, the other two must necessarily be empty strings. Therefore, all missing values in the fact table were removed by dropping the `gpu_code` and `ram_code` attributes from the file and skipping the rows where `cpu_code` was missing. In addition, during these operations, the type of the `cpu_code` variable was also cast from float to int.

For what concern the other tables, the only missing value found was the `Unknown` value in the `cpu` file. More precisely, this missing value affected the attribute `series` of the record with `cpu_code` equal to 467. The `Unknown` value was then replaced by the actual series name inferred from the CPU name. In addition, this record also had another problem: the redundant information concerning the series in the `name` field. We solved this issue by removing the `E-Series` pieace of string from the CPU name.

In general, we noticed that the `cpu` file was affected by several similar problems. In

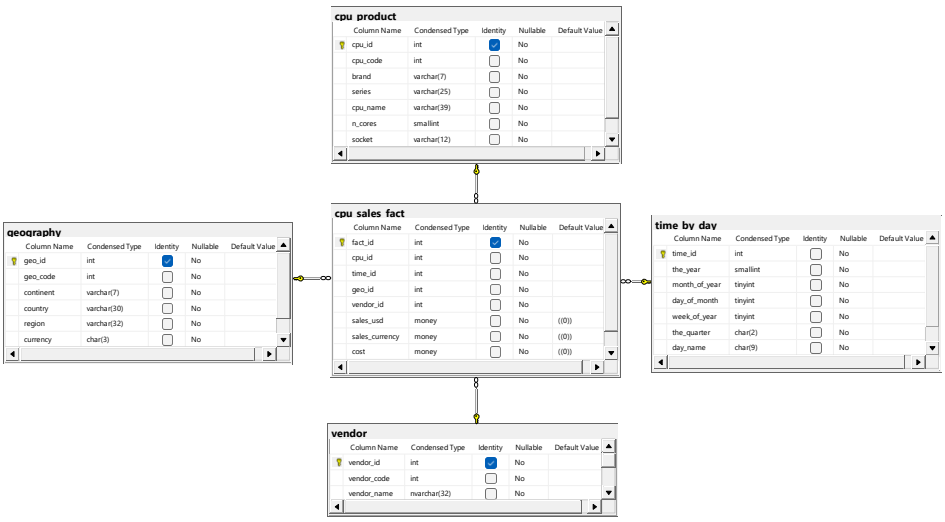

Figure 1. Database diagram

particular, the first four records had the CPU model name in place of the series name in the `series` column. Again, we solved this by inferring the correct series name from the `name` field. Also the instances with the `cpu_code` 187, 200, 240, and 467 were affected by these kind of issues which were solved in a manner similar to what we already discussed. Moreover, records 200 and 240 had an additional inconsistency related to the `socket` attribute. The former showed an incorrect prefix (Intel in place of AMD), the latter even an incorrect socket. The first problem was solved by completely removing the brand + `'socket'` prefix from the socket field of all records within the file. Indeed, since each socket uniquely determines the brand name, this information is irrelevant because already provided by the `series` column. The second problem was solved by fetching the correct socket code from AMD's official site.

There are probably other inconsistencies in the `cpu` file, but identifying them requires deep domain knowledge.

Each of the dimension tables has a unique attribute whose name terminates in `_code` which is supposed to be a primary key coming from the operational database. The approach adopted was that suggested by Albano and Ruggieri (2015, pp. 43–44): each dimension table should have a positive autogenerated primary key (*surrogate key*) that can coexist with any other primary key used in the original data. Therefore, new primary keys were created using MS SQL Server's `IDENTITY` function, one for each of the dimension tables, except for the time dimension, where we chose to use `time_id` (ibid., p. 43) directly. According to these changes, we also renamed the foreign key attributes of the fact table and their references. The old keys were kept so to support slowly changing dimensions with the *Type 2* strategy, thus they were considered a sort of natural keys.

For efficiency reasons, it was also created an autogenerated primary key for the fact table, called `fact_id`. This key also replaces the original `id` variable which appeared to be inconsistent and meaningless. Indeed, by sorting the data by the attributes `gpu_code`,

`cpu_code`, `ram_code` we noticed that the `id` increased whenever one of the three `_code` attributes increased.

Initially, the old primary keys were treated by adding a *unique constraint* on each of them. However, these constraints were only temporary: their purpose was to avoid inserting duplicate records. After populating the dimension tables, these constraints were removed.

Other minor changes were also made, such as renaming attributes in the time dimension to bring the name of the variables closer to their semantics and to disambiguate in case of homonymous functions. It was also decided to rename attributes of different dimensions that have the same names, as suggested by Albano and Ruggieri (2015, p. 60).

As requested, two new attributes were created in the time dimension: one, via the `compute_quarter` function, which shows the quarters (`the_quarter`), and the other, via the `find_day_name` function, which computes the name of the day of the week (`day_name`). For more details, see the file `time_transformations.py`.

Finally, in addition to the clustered indexes on primary keys automatically created by MS SQL Server, after populating the entire data mart, we also decided to create a few other indexes to improve query performance. In particular, a non-clustered index was created on each foreign key of the fact table to speed up the queries requiring multiple joins with the dimension tables. In addition, a non-clustered columnar index was created for each dimension table, specifically on the attribute that was assumed to be used most frequently in the where clause of queries (this also aims to improve slicing/dicing performance on the ROLAP cube).

The `create_tables.sql` and `create_indexes.sql` files include all data mart design information in a complete and structured manner.

References

Albano, Antonio and Salvatore Ruggieri (2015). *Decision Support Databases Essentials*.