ETL process

Simone Di Luna 544322

The second task of the project requires performing the following analysis using SQL Server Integration Services:

> for each year and CPU series, the name of the CPU that sold the most that year and the percentage in sales with respect to the total sales of the corresponding CPU series.

The first part of the assignment is unambiguous, but the same cannot be said for the second part. Specifically, it has been interpreted in the following way:

1. Find the best-selling (in terms of sales revenues) CPU for each year and series.

2. For each CPU at point 1, relate its sales for that year to the overall sales of the series to which it belongs, regardless of the year.

According to the above interpretation, the result that should be achieved in SSIS is the same computed by the analytic SQL query shown in Listing 1.

The Integration Services solution shown in figure 1 produces the same output. This SSIS project consists of a unique package having a *control flow* made up of only one task, namely a *data flow task*.

As you can see, the data flow starts with an *OLE DB source component* which connects to the personal database in the `lds.di.unipi.it` server and extract data from the fact table. However, not all the attributes were useful for our analysis. Specifically, we made a projection only on the sales in USD and the foreign keys referencing the `cpu_product` and `time_by_day` tables.

Next, the flow continues with two transformation components connected through a sequence path. They aim to join the fact table with the two dimension tables to retrieve three attributes: `the_year`, `series`, and `cpu_name`.

```sql
WITH tmp AS (
   SELECT the_year, series, cpu_name,
      RANK() OVER (PARTITION BY the_year, series
         ORDER BY SUM(sales_usd) DESC) AS rk,
      100.0 * SUM(sales_usd) / SUM(SUM(sales_usd))
         OVER (PARTITION BY series) AS PercSalesWRTTotalSeriesSales
   FROM cpu_sales_fact f
   JOIN time_by_day t ON f.time_id = t.time_id
   JOIN cpu_product c ON f.cpu_id = c.cpu_id
   GROUP BY the_year, series, cpu_name
)
SELECT the_year, series, cpu_name, sales,
    PercSalesWRTTotalSeriesSales
FROM tmp
WHERE rk = 1
ORDER BY the_year, series;
```

Listing 1. Assignment solved by means of SQL analytic functions.

The element used to perform these two transformations is the *lookup component*, because it allows us to exploit the design of the data warehouse, hence improving performances: for each value of a foreign key in the fact table there is exactly one correspondence in the referenced dimension table.

We have different facts for each year and series where the same CPU may appear many times. Therefore, to find the best-selling CPU for each period and series, we first need to compute the overall sales for each CPU in each year and series. In the SSIS project, this is achieved by means of an *aggregate component*, the first one (from top to bottom) in the diagram in figure 1.

After doing this, we can compare the different CPUs for each year and series and find the best CPU in each of these groups. The second *aggregate component* serves this purpose. However, what we actually find in this step is the revenue of the best CPU in each year and series, not its name. Therefore, we need to attach these figures to the corresponding records in the previous table. Therefore, before the second *aggregate component*, we need to split the flow into two branches by means of a *multicast component*.

Moreover, to compute the "percentage in sales with respect to the total sales of the corresponding CPU series" we need also to compute the total revenue by CPU series (third *aggregate component*). This can be efficiently done through the results aggregated by year and sales computed in the previous step. Therefore, we need another *multicast component* splitting the flow into two branches in order to retrieve these partial results before attaching them to the original table by means of a *mergejoin component* (where the join on the `series` attribute).

Finally, we need to attach back all these new partial results to the table before the first *multicast component*. Here, the join predicate consists of the conjunction of three equijoins:

```
the_year==the_year
AND series==series
AND sales==MaxCpuSalesByYearSeries
```

Alternatively, we could have performed the join only on `the_year` and `series` attributes and then used a *conditional split component* to take only those CPUs for which the total revenue is equal to the maximum revenue for a CPU belonging to the same series in that year.

Eventually, we use the *derived column component* to combine all the metrics we obtained so far to compute the ratio (and then the percentage) between the revenue of the best-selling CPU for each year and series and the total revenue by the corresponding series.

Finally, a *flat file destination component* was used to select only the information required from the assignment (i.e., year, series, CPU name, percentage in sales with respect to the total sales of the corresponding CPU series) and save them in a local `.txt` file.

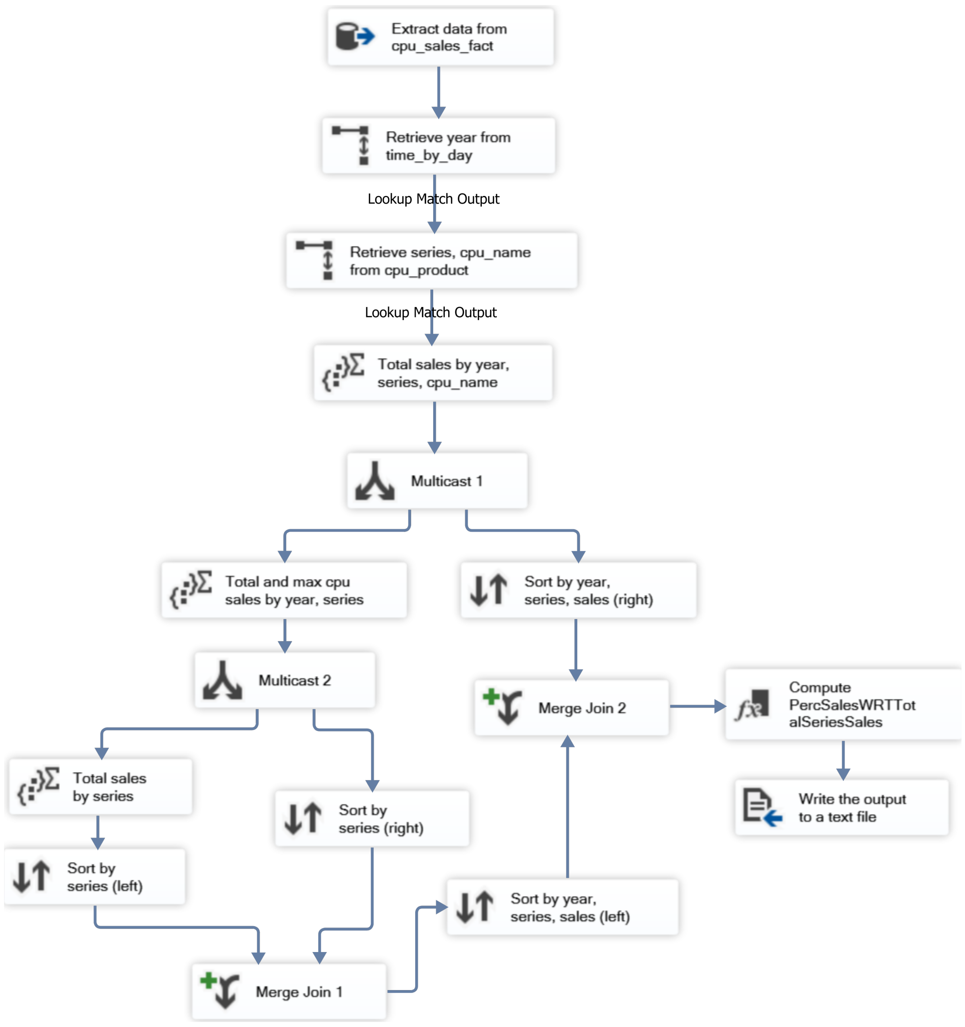
Extract data from cpu_sales_fact
Retrieve year from time_by_day
Lookup Match Output
Retrieve series, cpu_name from cpu_product
Lookup Match Output
Total sales by year, series, cpu_name
Multicast 1
Total and max cpu sales by year, series
Sort by year, series, sales (right)
Multicast 2
Merge Join 2
Compute PercSalesWRTTotalSeriesSales
Write the output to a text file
Total sales by series
Sort by series (right)
Sort by series (left)
Sort by year, series, sales (left)
Merge Join 1

Figure 1. SSIS solution.

Before concluding, two things should be highlighted:

- In the output file, we reported the names of the best CPUs by year and series but not their revenues, according to the requirements of the assignment.
- Given a year and a series, having more than one best-selling CPU is possible. Both the SSIS (through the mergejoin components) and the SQL solution report all of them. This seemed to be the most reasonable choice.