# ***CHAPTER 2: ETL TRADITIONAL FOR THE CREATION OF DATA MART***

The occurrence of the first need to integrated data, has led companies to address the issue internally, as the market did not offer sufficiently flexible and reliable solutions. For address the need for integrated data it was to develop in-house ad hoc software especially, to perform the extraction, transformation and loading of data into a single, integrated environment. Despite recent advances, today most companies use customized ETL solutions to meet the need for integration.

However, the most recent market developments have led to an increased demand for data Integration products, bringing to 60% the percentage of companies using an integration software to carry out activities business intelligence [8].

The recent economic downturn has also led to a decrease of the budget assigned to the development of information technology in enterprises, resulting in increased adoption of open source integration solutions.

One can therefore conclude that the ETL market today is characterized by the coexistence of three types of products:

* *Custom Software: I*ntegration of data internally developed to be able to meet the specific needs of their business scope. With the maturation of the market, this approach has become less and less affordable. In addition, the emergence of SOA and SaaS application architectures is decreeing the end of the products developed “at home”. Today's data integration Suite on the market offer definitely functionality and improved reliability;
* *Software owners:* The data integration products have matured steadily over the years, providing a range of functionality more and more rich and varied making these applications suitable to support the majority of business scenarios. The number of applications on the market today is high, providing suite of products able to cover almost all of the business needs to specialized products in specific business contexts or specific issues;
* *Open Source Software:* the limit of the largest owners products on the market are the costs necessary for their implementation. To come in against the needs of smaller companies, recently, the first open source products entered in the market. Are products that can support a fair amount of features but with a much lower cost than proprietary products (costs license void, reduced infrastructure costs, services paid for with use).

For my thesis, I decided to use an open source software that offers all the features necessary for carrying, possible minimizing costs: Talend Open Studio.

## ***2.1 TALEND OPEN SOURCE***

The Talend open source approach provides two products:

* *Talend Open Studio*: Free downloadable suite with an open source license (GPL). Talend Open Studio is presented as comprehensive data integration product and characterized by a wide range of functionality, sufficient for most needs;
* *Talend Integration Suite:* is an enhanced version of the free product that adds advanced capabilities such as collaborative development, advanced monitoring of the project and the Data Masking.

For those who do not have the hardware necessary to support the system, there is a third option consists of Talend On Demand, which is a type of software to offer a Service (SaaS).

The Talend products today offer the following features:

* Development environment user-friendly (based on the Eclipse platform);
* High number of connectors;
* Common Warehouse Metadata;
* Support to collaborative development.
* Data Processing Services;
* Trend monitoring;
* Data Profiling and Data Quality.

Let's see what are the strengths of the approach [7]:

* *No barriers to the adoption*: Free availability of the basic product makes almost immediate the installation of the software. Talend supports the customer through tutorial on the basic use and it is also possible to rely to a large community of users;
* *Fast Learning Curve:* The product is presented graphically user-friendly. Graphical interface is intuitive and the use of the basic functionality does not require special training;
* *Model of stable and predictable prices:* Proprietary products often involve high costs as that expand the functionality and capabilities of the product. This often makes it difficult to properly costed in the early stages of the project. Talend, however, provides a cost model based on the number of developers and the use of the service, independent of licenses, hardware and quantity of data to be integrated;
* *The importance of supporting communities:* The online community of experts and users of the product is already very broad and today is a very important factor to facilitate the implementation and maintenance of solutions offered by Talend. Forums, wiki, guides and free user contributions represent an added value that only a product of this type can offer;
* *Broad support to different types of data:* With more than 400 preset connections the Talend solution ensures compatibility with a large number of systems, databases, software packages, business applications, web services, etc. No other solution on the market offers a high number of possible connections.
* *Flexibility, versatility and product reuse:* Talend is not limited to a support to standard techniques of ETL but allows the implementation of different integration strategies. The ability to reuse already finalized projects also constitutes another point of strength of open source approach;
* *Functionality and Performance*: The level of functionality offered is comparable to proprietary products. However, there are some gaps in the field of data modeling, data quality and data mining. the research and development team dedicated allows the product to be up to the latest market needs and to propose innovative features;
* *Costs and time-optimized*: The solutions offered by Talend resulting from a 50% to 80% cheaper than traditional products, being less expensive to acquire and maintain and because allow a more rapid development of the integration system.

## ***2.2 CREATION OF DATA MART***

Specifically, a Data Mart is an analytical database designed to meet the specific needs of a business. Being logical or physical subset of a data warehouse, larger in size, it follows the same design rules but with aggregated data at various levels of detail, although it may sometimes also be formed in the absence of an integrated data system [11].

Table 3: Data Warehouse Vs. Data Mart

|  |  |  |
| --- | --- | --- |
| **Definition** | **Data Warehouse** | **data mart** |
| **Purposes** | neutral  Centralized Application  and shared in the  entire company | specific applications  Departments or areas |
| **Data** | Low denormalisation | High denormalisation |
| **Subjects users** | Subjects in many areas | Subjects in a single area |
| **Data Sources** | Many  External Data Operational | few  External Data Operational |
| **Features** | Flexible, extensible  Long life,  Data-oriented | Rigid, non-extendable,  short life,  Project-orientation |
| **Implementation Time** | 9-18 months for the first stage | 4-12 months |

Deployment can be of two types: top-down, building the data warehouse with a subsequent aggregation and export in various data marts, and bottom-up, build various data marts focusing on specific areas of the business you will to build a data warehouse. In this way you will have a scalable approach.

Immagine che contiene testo

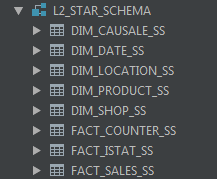
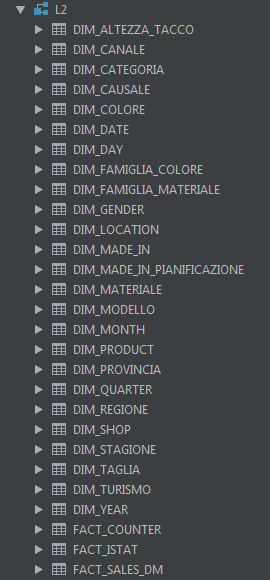
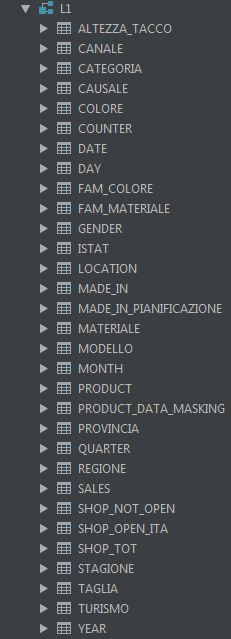
Descrizione generata con affidabilità elevata******

Figure 13: Level Tree

The phase of the ETL for the Fashion Retail project is based on creating a top-down data mart in order to get a fact table of sales with specific data to extract information that currently the customer does not know and that could bring it to produce a competitive advantage or increasing its economic potential. Any data system strictly follows the ACID (Atomicity, Consistency, Isolation and Durability).

To facilitate the recognition of files, tables and any reloading need, we will use a specific nomenclature for each level that make easier to identify the individual files. The tables created must then reside in a tree structure that allows the historical and conceptual navigation through levels.

Table 4: ETL Levels

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SCHEME** | **L0** | **L1** | **L2** | **L2\_STAR**  **SCHEME** |
| **Definition** | Extraction of data from various types of files without transforma-tion. | Major changes and data quality operations. | Very fast and slender tables. use of surrogate key to connect the various attributes. | Few tables but full-bodied, to have all the necessary fields data for reports. |
| **Area** | Staging / Extraction Area. | Transformation Area. | ETL  Area. | Visualization Area. |
| **Primary key** | NO. | YES. | YES. | YES. |
| **Surrogate key** | NO. | YES. | YES. | NO. |
| **Action on Table** | Truncate table but maintain scheme. | Nothing. | Nothing. | Nothing. |
| **Action on data** | Insert. | Update /  Insert. | Update /  Insert. | Update /  Insert. |

***2.2.3 Delta Of Data***

* The power of the DWH normally starts with the Initial Load, the process that provides a complete population of data warehouse with consistent datas from which you can update and loading new data with delta loads.

The load of delta portion can be done in different ways:

* *CDC (Change Data Capture):* Tables are subjected to a change data capture mechanism that automatically intercepts or replicate, for each table that needs, the delta extraction of new data with respect to the L0 data already loaded (recommended to replicate the tables very large, so as not having to download moles of unnecessary data);
* *MINUS*: The are not subjected sources tables to any of change detection mechanism and we have to independently extract the amount of data and find the delta performing a minus of the data that we possess with new ones (recommended for the master data tables that contain one mole of note data and do not provide too many changes in the high number of records);
* *FULL*: Daily replication of the table data of the small size and low variability of the content.

### ***2.2.2 Historicization***

The DLT tables can have a historicizing required to recalculate the data warehouse, to avoid a loss of data or load partial data due to problems, such as server problem.

To have a historicizing of the data we have two choices:

* Creation of partitioned shadow tables by DLT JOB\_ID extraction, called HIS. The tables Delta DLT will not be partitioned with the option truncate / insert (truncate the table keeping the initial scheme and insert the new data) and they will contain only the data of the current JOB\_ID while the HIS, will insert with partitioning for JOB\_ID, with a speed up of the extraction process.
* Have the history directly on the DLTs in insert with the partitioning for JOB\_ID, always to speed up the extractions.

You can use only one of these two modes in order to standardize the architecture of the tables to a single model and, depending on the mode used, it will be necessary to differentiate the code of an eventual recalculation.

In the thesis project, the Delta and Historicization tables will not be used because the data come from local csv or Excel file with no possibility of recalculation, not being connected to a source with a constant daily / monthly update

### ***2.2.3 The Multidimensional Model - Dimensional Fact Model***

The ER model (Entity-Relationship Model), spread to design relational information systems, it is not suitable to express and analyze in detail large data sets [10].

The multidimensional model or DFM (Dimensional Fact Model) is a conceptual model where is possible represent data within a Hypercube whose edges represent the dimensions of analysis, which subsequently will be divided into many "cubes", each identified by a triple of coordinates. Each cube ideally contains the values assumed by the measures for that data triad and is commonly referred to as "Fact" because it represents the occurrence of an event of interest for the business domain.

A multidimensional model is mainly based on four key concepts:

* *Fact*: Table that typically models a specific business area (Sales, Orders, Production, etc.) and is characterized by a more measures;
* *Measurement*: It is the quantitative aspect of the fact and it is of high importance for the analysis. From measures are extracted the KPIs (Key Performance Indicator) that will guide enterprises in their business strategies. Some examples can be the Quantity produced, the profit, and price;
* *Dimension*: It represents the coordinates of the analysis Done. Among these we can find Date, Product, Shop;
* *Dimensional Attribute*: It is a logical grouping of some elements of a same size. Are classes of elements that allow the user to select the data for specific characteristics.

To navigate in the multidimensional cube there are different operations that allow you to organize your data in it, through different perspectives [10].

The first is the Pivoting which allows to quickly change the display of data by turning the cube axes and has the purpose to change the point of view from which analyzes the data. The second, however, is the Slice & Dice to select and project the cube data. Specifically, they will extract sub-cubic filtering on a (Slice) or more (Dice) dimensions. Finally, we have the Roll-Up and Drill-Down to help you navigate within a hierarchy, choosing the level of aggregation according to which the user wants to analyze the data. Specifically, it will rise to a hierarchical level with the roll-up, while you will drop one level with drill down.

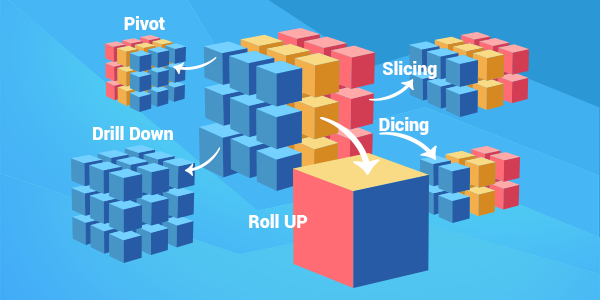


Figure 14: Hypercube OLAP

This system has been idealized to accomplish certain purposes such as provide support to the conceptual design, create an environment where users can make queries intuitively, to interrogate effectively the supplied reports, facilitate communication between designers and users in order to formalize the project requirements, build a stable logic design platform and finally, create and publish a clear and effective documentation.

## ***2.3 L0 LEVEL - DATA INGESTION***

The L0 level is the initial level and represent the Extraction phase of the information from source systems for the creation of a data warehouse. The tables are achieved in the Staging Area where the discharge takes place without transformations of data.

The source systems can be of different types but the most common are the operational systems, database or files:

* *From table*: Reading the daily data sets or replication of the entire data set via database;
* *From file*: Extracting information directly from documents.

The data ingestion is the process of acquiring and importing data for the use or the immediate storage in a database.

Data can be transmitted in real-time streaming or ingested into periodic lots of data:

When data is ingested in real time, each data element is imported immediately while it is being emitted by the source. When the data are imported in batches, the data elements are imported into blocks at periodic time intervals. An effective process of data acquisition starts by prioritizing data sources, validating individual files and loads the data items to the correct destination.

If there are several sources of large data in different formats, it can be difficult for companies to acquire data at a reasonable speed and process them efficiently in order to maintain a competitive advantage. To this end, manufacturers offer custom software for specific processing or application environments. When importing data is automated, the software used to run the process, can include data preparation capabilities to structure and organize data; so, they can be analyzed through the Business Intelligence (BI) or, in the specific, through several algorithms of Business Analytics (BA).

Tables that we will create in this level are all prefixed by "STG" as Staging Area, that corresponding to the total import of the source document with no changes to the schema and with only small changes due to the capacity of the database variables of SQL Server, used for the project, to avoid the data truncation errors.

### **2.3.1 Metadata**

The term "metadata", in the context of data warehousing where they play a substantial role, they indicate the sources, the value, the use and functions of data stored in the DWH and describe the altered and processed data during the different levels of architecture.

So, the metadata tables are closely linked to the DWH and its applications are use from both supply side and analysis side.

It is possible to distinguish two categories of metadata, according to various uses

• Internal: Of interest to the administrator, describe the sources, transformations, power policies, logical and physical patterns, constraints and user profiles;

• *External*: Of interest to users, are, for example, the definitions, the quality, the units and significant aggregations.

The metadata is stored in a special container which can access all the other components of the architecture.

It can be classified also about the level that consider:

• *Global*: Contain metadata related to all levels and processes;

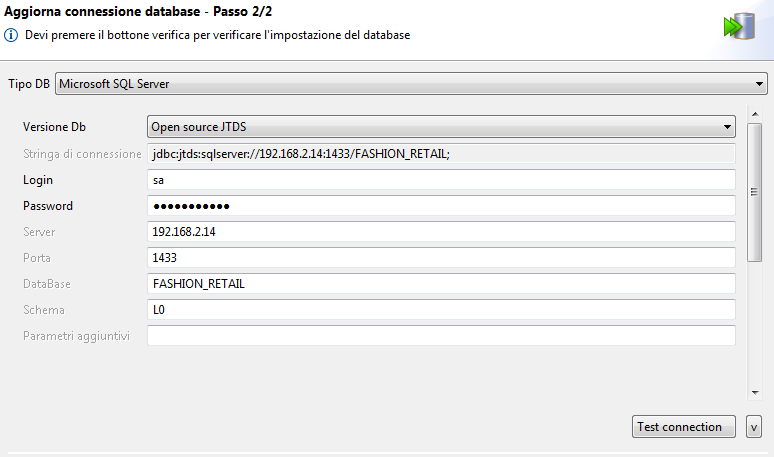
• *Proces*s: We distinguish the metadata depending on the feeding system and the process in which they are involved. The metadata describing the individual process or less related to a particular system (internal sync point between tables, their percentage of fault-tolerant system) must be own for each of them.

The concept of metadata is very critical within the DWH management and is often debated, whether or not, to keep it inside the project or manage it with external logic that release the technology and the product used from the actual purpose of the metadata.

Normally, a metadata is recorded in a table, for usability reasons on the part of the heterogeneous systems, and it has the utility to describe uniquely and precisely in what state is a process, in order to avoid multiple instances launches, launch a process at a specific time, say if the process is completed in correct or with errors or provide the time interval in which the process has been completed or running to extracted the data.

Through the capabilities of ETL software Talend Open Studio it has been possible to import various file types to build a new Database (Data Mart) in SQL Server called FASHION\_RETAIL, for evaluating the entire sales department.

To use it on Talend I need to create a database connection for each level, where i will implement the anagrafiche (all the dimensions tables) and the movements (all the fact tables) by importing metadata.



Server, port, and database name to start the connection

DB Version

DB Type

Figure 15: Connecting to the Database

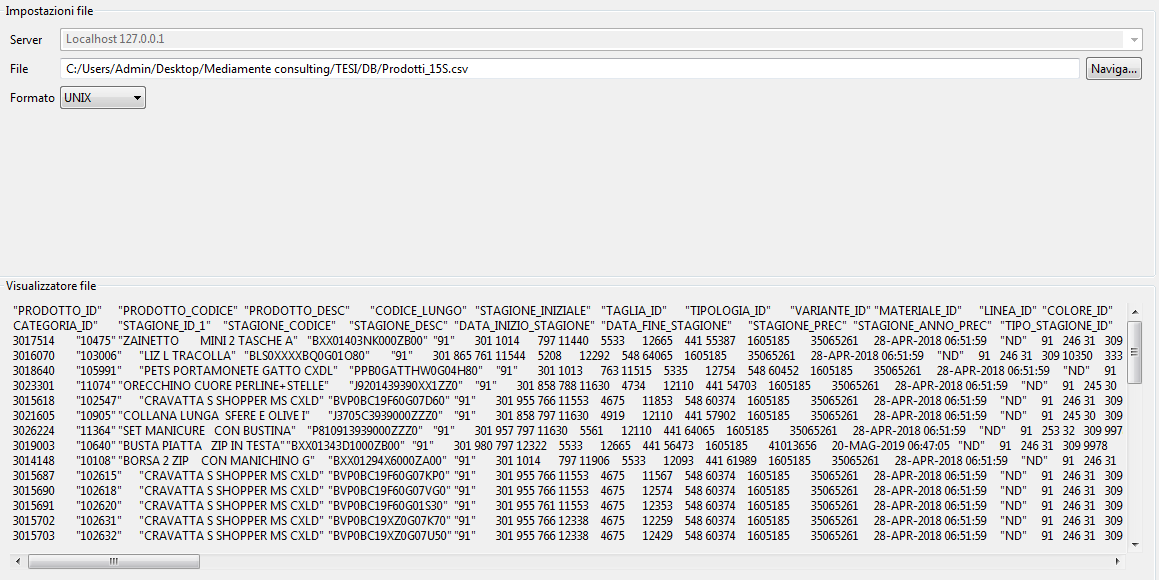
The table below lists the tables created at the L0 level - Data Ingestion with the extracted source files for the project and the mainly type: Excel or .csv (Delimited Files).

Table 5: Metadata

|  |  |  |
| --- | --- | --- |
| **Database Table** | **Metadata Type** | **Metadata Name** |
| STG\_Causale | File: .Csv | Reasons |
| STG\_Made\_In | File: .Csv | Made\_In |
| STG\_Made\_In\_Pianificazione | File: .Csv | Made\_In\_Pianificazione |
| STG \_Product | File: .Csv | ProdottiS15, Products <S15, Products <S15n2 |
| STG \_Shop\_Open | File: .Excel | stores open |
| STG \_Shop\_Closed | File: .Excel | Shops closed |
| STG\_Fam\_Colore | File: .Csv | colors |
| STG\_Fam\_Materiale | File: .Csv | Materials |
| STG \_Sales | File: .Csv | PIAF, Scontrini2019 |
| STG \_Provincia | File: .Csv | province |
| STG \_Counter | File: .Csv | Visitor counter, Contapersone2019 |

Having a set of files in CSV and Excel, the phase of import in the software development, at the storage level of the ETL process, will incorporate them into folders, one for each type, in the form of metadata.

The project implemented in Talend will have these two types of loading views:



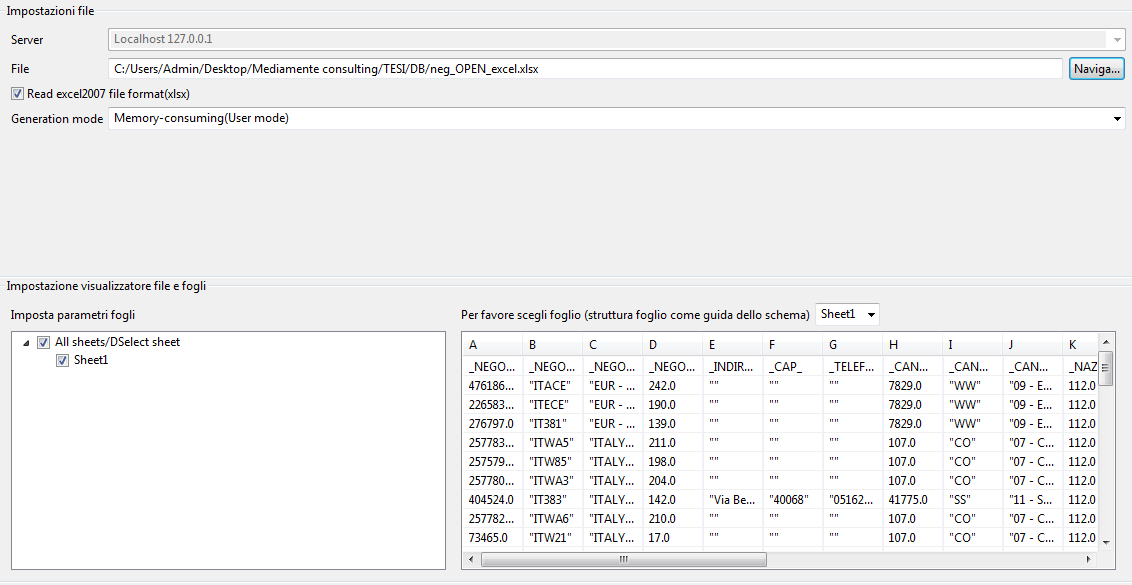
data in CSV files

Source file

Server

Figure 16: Loading from Delimited File

Server



Data in EXCEL files

Source file

Figure 17: Excel File Upload

Metadata, once imported, must be reworked to create the database on the server. Conceptually, for every job created in Talend, the following four transactions took place:

* Import csv or Excel file;
* Union files via the Tunite instrument, if necessary;
* Change and mapping names, lengths or types of attributes or JOIN tables thanks to primary keys via the TMAP tool;
* Creating a table in SQL SERVER using the tool tDBOutput (tMSSQLOutput).

To perform these tasks you need to create a new JOB, which will contain the various import methods.

A very significant example is the creation of the STG\_PRODUCT, the table in the staging area of ​​the Product dimension.

In it you can observe how the import of various files merged via a Palette called Tunite, which captures the patterns of different source files to create one that suits everyone; if sources have different schemes you will be marked with a warning, even if, the process continues to operate normally.

TMAP tool (2.2.4 data quality) is use to transform the input patterns to optimize outputs (L1) and create join relationships among the various tables.

In this specific case, it helps to identify which products are accepted according to the scheme previously defined and which rejected, respectively, with the creation of the staging area of ​​the product and an excel file with the rejected products.

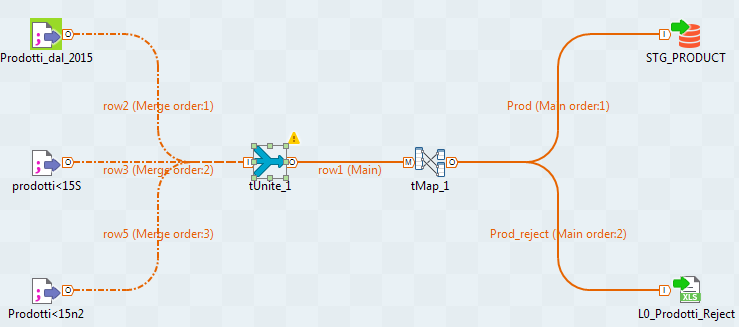


Figure 18: Multi-Loading

The same procedure was carried out for the other tables, with the most common case defined by the direct import of files into the database.

It is important to note that each file is made on a different Job (work area). This is necessary to prevent errors during the data loading or to isolate problems.

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figure 19: L0 Single Loading

## 

## ***2.4 L1 LEVEL - DATA OPERATION***

The L1 level include data quality, data normalization, and all the transformations of the source files, and is definitely the heart of the entire ETL process.

Unlike the L0 extraction level, data are extracted exclusively from the tables created in the Staging Area previously created, to then be subsequently processed and loaded in the same Database FASHION\_RETAIL, but in a normalize manner, to enhance the process and have a continuous control over activities taking place.

For tables already in the Staging Area the process is very simple, as you only have to select attributes (mapping) in the TMAP that interest me and make the appropriate changes for the normalization of the data. In the transfer, after a first check of integrity, they create the Primary Key, that will go to uniquely identify the specific attributes of each table, often represented by an ID field.

Important phase is the Pre-Loading, namely the extraction of secondary dimensions from other tables in the L0 layer.

Using as input the STG\_PRODUCT table, for example, we can observe the possible formation of new dimensions and perform specific queries for each of them, extracting univocally from the source table fields related to the category, thanks to the "SELECT DISTINCT" function of the SQL language.

***~~Immagine che contiene testo, targa

Descrizione generata con affidabilità molto elevata~~***Immagine che contiene testo, mappa

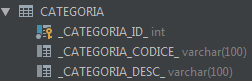
Descrizione generata con affidabilità molto elevata***~~~~***

Figure 20: Pre-Loading L1

"**SELECT DISTINCT**

CATEGORIA\_ID,

CATEGORIA\_CODICE,

CATEGORIA\_DESC

**FROM**

L0. STG\_PRODUCT "

### **2.4.1 Data Quality**

Data quality is a perception or evaluation of the suitability of the data for a purpose in a given context. Data quality is determined by factors such as accuracy, completeness, reliability, relevance and frequency. Since the data have become more closely linked to the operations of the organizations, the emphasis on data quality has gained more attention.

The check is divided mainly into two parts:

* *Referential integration*: Control through the foreign key checks. It is carried out by means of join with the L1 tables containing the fathers of which verify the relations;
* *Record validation*: Data must undergo checks to reject records that do not meet the requirement not null or other simple and complex conditions (date falling in intervals or text fields of defined length etc...)

The following table shows (one for each type) all the data quality operations performed during the design of the data warehouse, in the TMAP tool.

In some cases, they were sometimes further processing at the metadata level, particularly in the size of fields. These changes have been made to avoid the data truncation, in order to avoid subsequent inconsistencies in the final data.

Poor quality data is often considered as the source of inaccurate reports and strategies in companies. The economic damage due to data quality issues can range from miscellaneous added expenses when the packages are sent to wrong addresses, up to fines of regulatory compliance for improper financial dealings.

Table 6: Data Quality

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type Source-**  **Destination** | **Source** | **Transformation for Data Normalization** | **Destination** |
| Datetime- Date | "Dd-MMM-yyyy hh: mm: ss" | Change in the variable type directly in TMAP options | dd-mm-yyyy |
| String-String | "Piemonte" | Regione.toUpperCase () | "Piemonte" |
| String-String | "Trentino Alto Adige" | Regione.replace  ( "-", "") | "Trentino Alto Adige" |
| String-String | Bolzano /  Bolzen | Provincia.replace  ( "/ Bolzen", "") | "Bolzano" |
| String-String | "Piemonte" | Regione.trim () | "Piemonte" |
| Integer - Integer | null | Totale\_ Arrivi\_2018 == null? 0:  Total \_Arrivi\_2018 | 0 |
| Integer -Integer | 23 | 23 |
| String-Double | "23" | Double.parseDouble  (DISCOUNT) | 23.0 |

### ***2.4.2 TMAP Componentt: Talend Open Studio***

The TMAP component [24] is one of the main components of processing in Talend Open Studio and is used mainly for mapping the input data to output data or a source pattern on a target.

INNER JOIN via Primary Key

Output Creating containing the records rejected

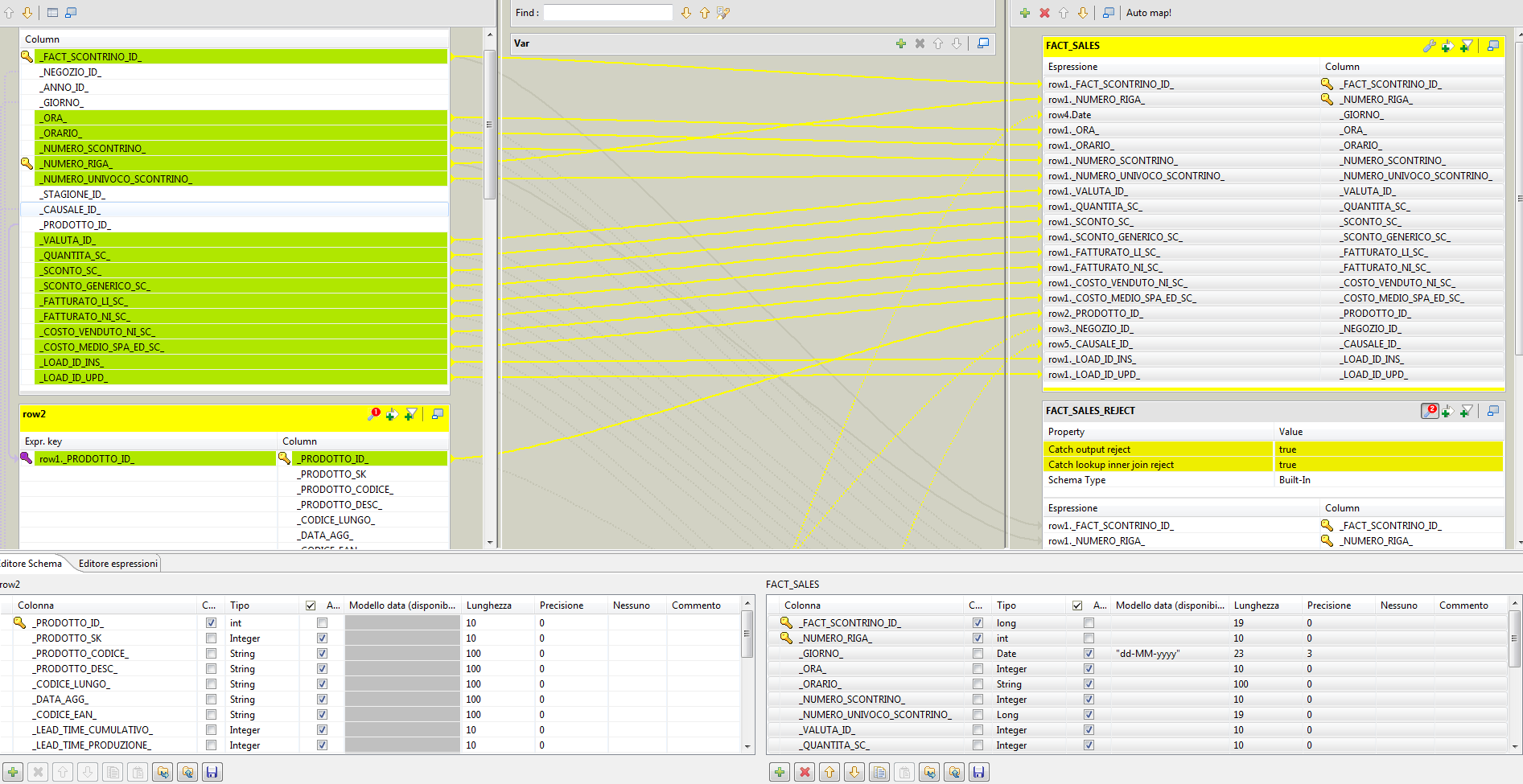


Figure 21: Join & Mapping in TMAP

Mapping

I link the file attributes with the attributes that I will create in the DB table. The full view of them is present in the lower part of the representation

In addition to perform mapping functions, the Tmap can be also used to merge multiple input tables combining data into a single target table.

All transformations previously listed in the table refers to the Data Quality and they are carried out in Tmap with a further possibility to filter data.

A mapping expression must be done for each columns of the input patterns. For do that you can use the editor of Talend that contains any available Java class or Talend routine.

Immagine che contiene screenshot, mappa, interni, computer

Descrizione generata con affidabilità elevata

TRANSFORMATION

Figure 22: Transformation in TMAP

Mapping

As you can see from the examples above, each mapping may have a different complexity according to our needs.

In the best practice you should always make sure that the data and the created models will be always available for reuse. In fact, it is very likely that in a recent future, we can extend our ability to a specific attribute of a table, for example by controlling the format. Therefore, to have a code and good process performance is better to change the logic in a single routine, rather than operate several times in the individual mapping expressions.

## ***2.5 LEVEL L2 - ETL DATA MART BEST PRACTICE***

In the new data analytics projects, one of the best practical implementation of a data warehouse or a data mart is the SnowflakeDB, mainly developed to have an ETL process effective and fast.

It is defined very different with respect to the relational database because it is built on the principles of the cloud, it is very elastic, it is virtually maintenance-free, it is almost in real-time and has a native support for unstructured and semi-structured data (JSON). But, at the same time, it is similar with respect to the relational database because it is a columnar stored relational database, in fact, the SnowflakeDB is founded by Oracle, the same company founder of the relational database.

In particular, my goal is to bring a SQL Server relational database, a design level in Snowflake Style:

* Data must be classified and marked appropriately, especially if highly protected;
* Data must preserve its history through audits and verification of data that will have to be validated at the source, if possible;
* Data must be processed in micro-batches;
* The data must be ingested and loaded and then processed according to the rules of ETL or ELT;
* Process the CDC end-to-end data to avoid performance problems;
* Create separately data mart instances according to the specific requirements of the company's business, creating systems of governance.

There are some best practices that apply to the implementation of a Snowflake Database, specific for its unique architectural differences with other relational databases or platforms of big data:

* Use independent DWH Multi-Cluster with storage capabilities and scalability of shared data to optimize processing requirements of different workloads. For example, the Staging Area (Level L0) can be located on a different database than the Core Layer (Level L1);
* Assign a separate virtual data mart for each business to have an optimized data consumption;
* Keep, if possible, the semi-structured data in its original format to increase processing performance data. Often the JSON data is processed faster than those converted into relational tables. When you store semi-structured data, SnowflakeDB optimizes the storage on the basis of repeated elements within the structure of semi-strings;
* Upload the data into smaller chunks instead of one large file and load them in parallel using multiple nodes. For one customer we were able to charge 24 months telemetry event data in less than two weeks with a small cluster nodes;
* Assign virtual clusters to separate patterns of a data warehouse to optimize the performance
* Consider the clustering of large tables to improve query performance.
* Recluster if performance degrades. Sometimes the cluster / reclustering may cause poor query performance, although it is recommended to analyze the data in the table before you make these changes;
* SnowflakeDB stores the metadata (min and max values, distinct values, etc.) effectively reducing the stock micro partitions needed to scan a query;
* Is recommended running the ingestion of data based on events to allow the chronological order of the data.The data pipeline based on flexible framework must be completely separated from the actual processing structure. This operation is done by the creation of type Integer surrogate keys (SK\_KEY) to improve the efficiency allowing quick connection and loading of data for the large tables. This allows minimum changes to the code if there are any changes in the target or in the source schema, still be able to perform all the loading operations, transformation, aggregation and processing of data;
* Building a robust audit of the budgetary control framework that tracks not only the lineage of data and data quality, but optimizes database performance than the processing costs;
* Create database clones for testing or for validation to avoid duplication of data.

SnowflakeDB is a powerful database with unique features that enable rapid implementation of data analysis projects, but as all other databases requires careful planning. Following the best practices just listed, the first part to play in the project is the creation of the Surrogate Keys for all data tables of the level L1, creating parent and child relationships between dimensional tables.

For example, with regard to the pre-loading of the product table, surrogate key will be created with the following SQL:

* ***ALTER TABLE*** *L1.PRODUCT ADD \_PRODOTTO\_DM\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.ALTEZZA\_TACCO ADD \_ALTEZZA\_TACCO\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.CATEGORIA ADD \_CATEGORIA\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.COLORE ADD \_COLORE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.GENDER ADD \_GENDER\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.FAM\_COLORE ADD \_FAMIGLIA\_COLORE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.FAM\_MATERIALE ADD \_FAMIGLIA\_MATERIALE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.MADE\_IN ADD \_MADE\_IN\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.MADE\_IN\_PIANIFICAZIONE ADD \_MADE\_IN\_PIANIFICAZIONE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.MATERIALE ADD \_MATERIALE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.MODELLO ADD \_MODELLO\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.STAGIONE ADD \_STAGIONE\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.TAGLIA ADD \_TAGLIA\_SK INT IDENTITY (1,1) NOT NULL;*
* ***ALTER TABLE*** *L1.TURISMO ADD \_TURISMO\_SK INT IDENTITY (1,1) NOT NULL;*

At this point, through the creation of a connection to the schema L1 of Fashion Retail database in the form of metadata, have need to proceed with the extraction of the tables having the Surrogate Key present in it, that, through the processing in Talend Tmap tool [24], will become the primary key of the new table size.

Specifically, it is seen how the imput "row1" defined as the table category of level L1 have as primary key the Id, is set to level L2 with a identical scheme to the previous one, but with primary key, the surrogate key. In general:

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

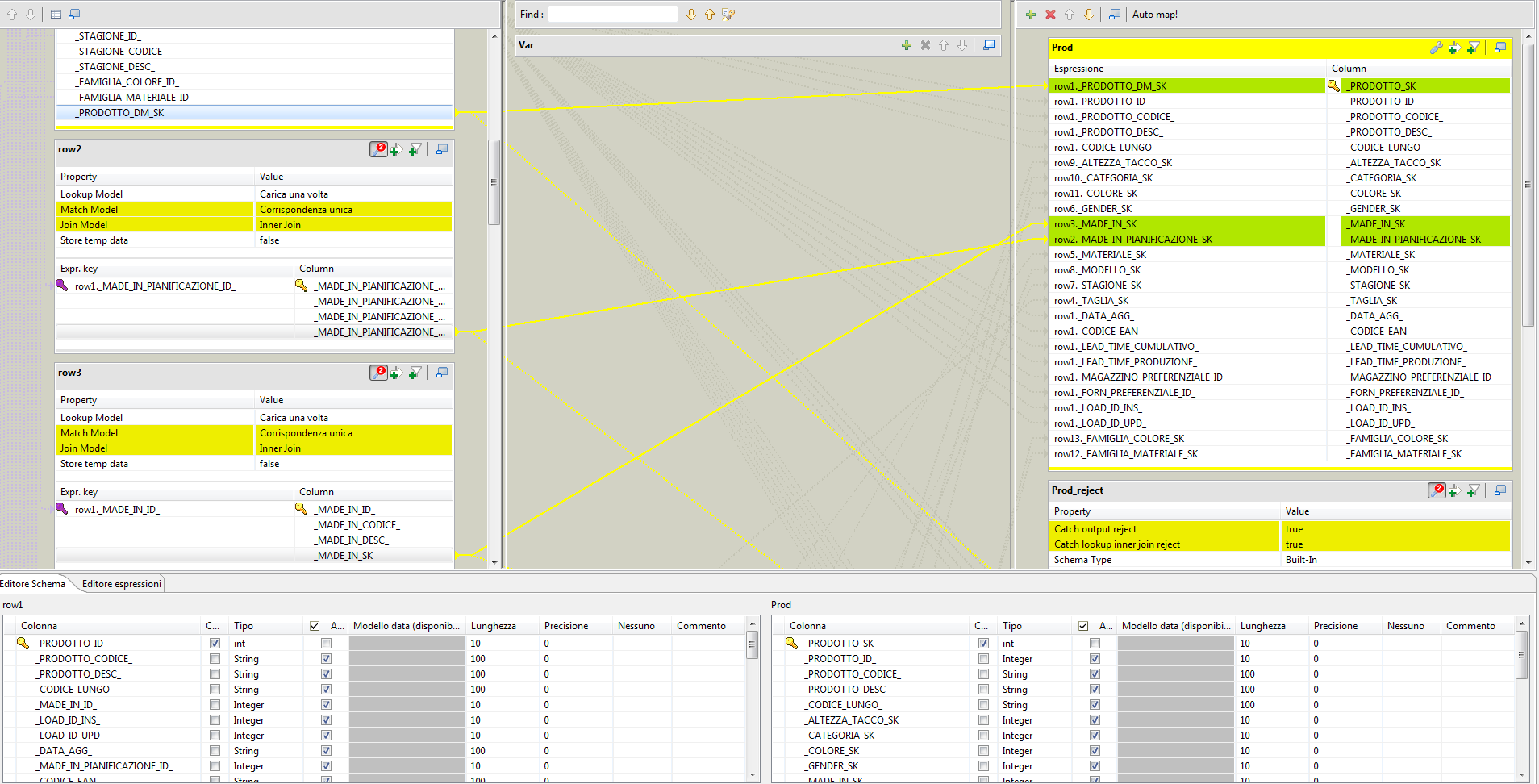
******

Figure 23: SnowflakeDB Dimension

Moreover, it is very important to be careful in the construction of the fact tables, as, being the father tables of dimension tables, will not have its own surrogate key. In them, it will be shown only the surrogate key of dimension tables that characterize it, ignoring all the information such as, for example, the Id and the description, since, in this process has no need to go into detail in a single table, but it is a tree structured system with a system of hierarchies, defined precisely by Joins between the surrogate keys of dimension tables.

***Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata***

Figure 24: FACT SnowflakeDB

### ***2.6.1 Snowflake Schema***

A database is in 3NF (third normal form) whether all non-key attributes depend on one andd only one key, ie there are no non-key attributes that depend on other non-key attributes. This normalization eliminates the transitive dependency of attributes from the key and is called SnowFlake scheme.

The name comes from the fact that the dimension tables branch and resemble, like a snowflake. Observing the model, is highlighted as a fact table is surrounded by the dimensional tables, with which it will create the aforementioned branching. Unlike the star schema, tables of dimensions in a snowflake schema may have their own categories. The dominant idea behind the scheme is that Snowflake dimension tables are fully normalized. Each dimension table can be described by one or more lookup tables. This is repeated until the model is not completely normalized.

Obviously, the normalization creates a greater complexity in performing the query of the snowflake schema, as, for example, we will have to dig deeper to get the name of the type of product or the municipality of a store. The structure is based on a series of nested JOIN, where we must add to a simple JOIN another JOIN for each new level within the same dimension. Of course, there is a number of standard annidation, but depends on the given level that you want to extract. More data is in depth, the more the process of writing query will be complex [23].

Basically, a query against a data mart based on Snowflake schema will run slower than one in Starschema. In most cases, this is not a problem: it does not matter much if we get the result in a second or a millisecond.

In the proposed project, a comprehensive view of the snowflake is as follows:

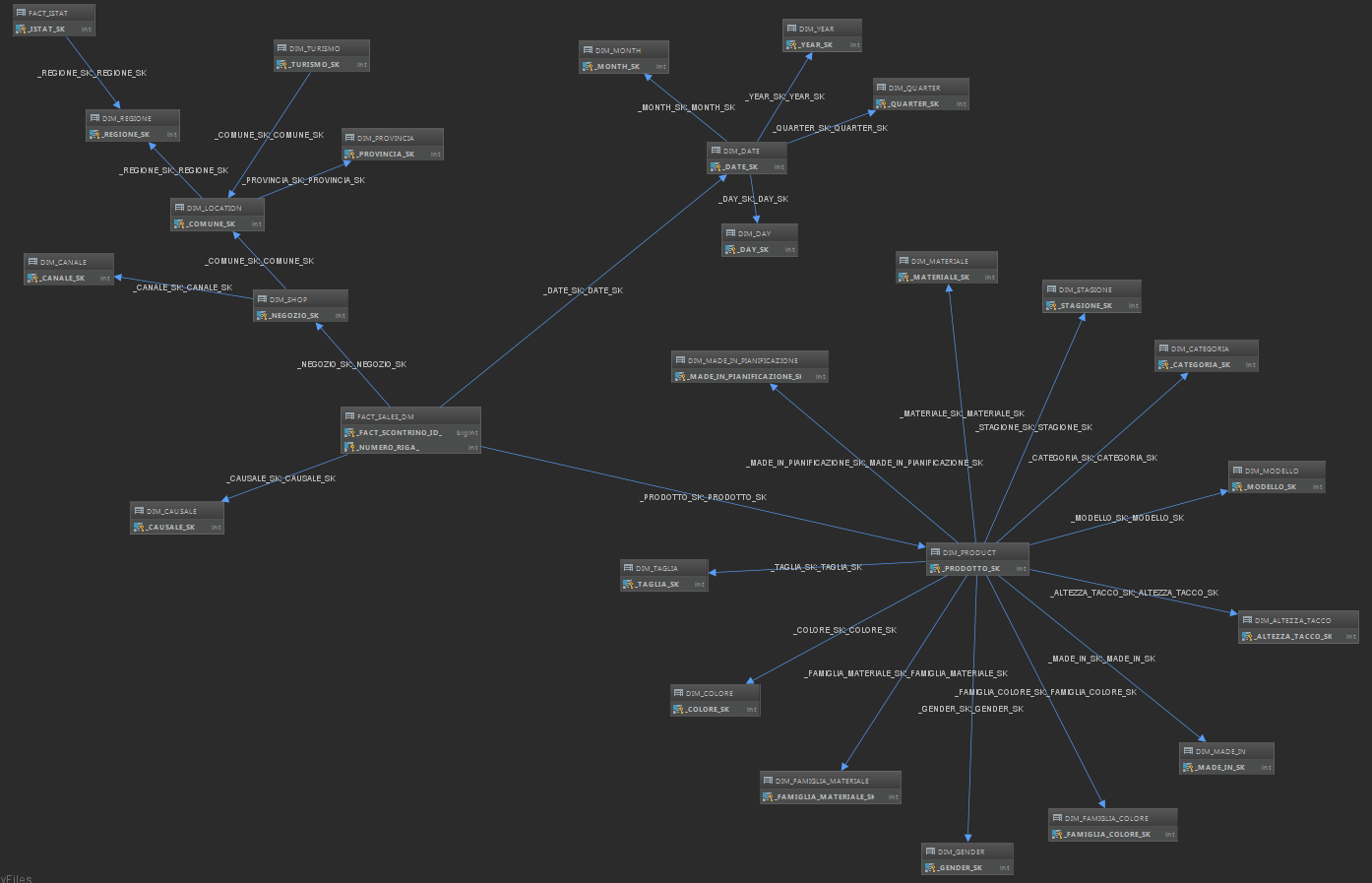


Figure 25: Snowflake Schema

## ***2.6 LEVEL L2 - VISUALIZATION DATA MART BEST PRACTICE***

The Relational Database, the most widely used, compared to SnoflakeDB exhibit the same levels of Staging Area and Transformation Area but with a substantial differentiation in the final level L2.

In this case, the goal is not to have an ETL super powerful process, but to have fewer end large tables with more information to facilitate, through data visualization software, reporting to give effective future directive to implement better decision strategies or to provide a simple financial audit on the economic performance of the company.

The Job schema does not change compared to the SnowflakeDB, but change the mapping of variables in the Tmap [24].

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figure 26: Job Product Star Schema

In fact, the tables of level L2, in this case it can defined like aggregate tables of level L1 tables and it is possible to obtain them with a series of JOIN between tables, no longer connected to the Surrogate Key, but directly to the Primary Key, with the necessary integrity checks.

The final table attributes can also result from different tables, as each attribute of a table is connected to the attribute of the final table through the mapping created by JOIN operation.

Immagine che contiene screenshot, mappa

Descrizione generata con affidabilità molto elevata

Figure 27: Product TMAP Star Schema

The same procedure was done for the fact tables, remembering to make JOIN not with the primary keys of the dimension tables, but make them directly with the surrogate keys, because the fact tables are in 3NF, like in snowflake schema.

### 

### ***2.6.1 Star Schema***

Once built the Data Fact Model, the logic diagram must be implemented. It is represented according to a Star Schema, which the center is constituted by a fact table; the points of the star represent instead the dimension tables that branch out from the center. The main features of a Star Scheme are as follows:

* simple structure & easy to understand;
* High performing queries, because they reduce the joins to be made between tables;
* Loading time of the relatively long data, because the data redundancy due to the de-normalization, causes an increase in size of the table;
* Widely supported by a large number of business intelligence tools;
* The fact tables in a star schema is in third normal form, while the dimensional tables are de-normalized [10].

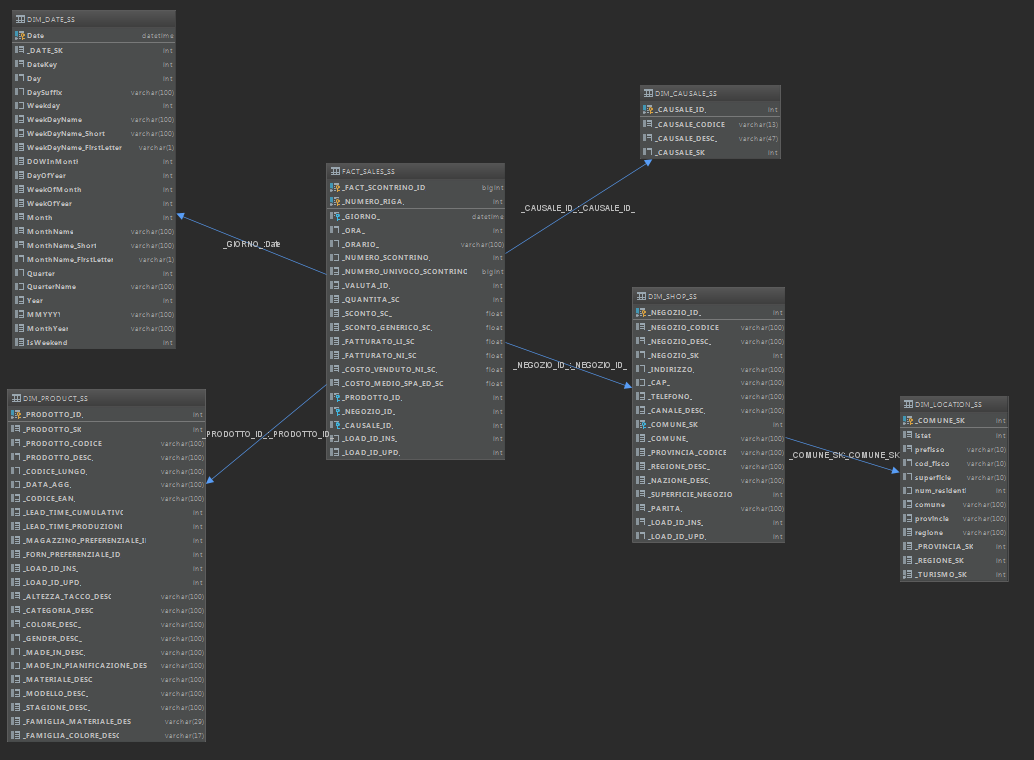


Figure 28: Star Schema

## 

## ***2.7 FULL LOAD ETL***

To speed the whole process explained above, the best method is create jobs that contain other job. In this way, I can enclose into subgroups the master data and the movements for each level, and then, run them all at the same time.

For example, the job STG\_ANAGRAFICHE in the level L0, where I collect all of the job of level L0 relating to all master data, as shown in the image below.

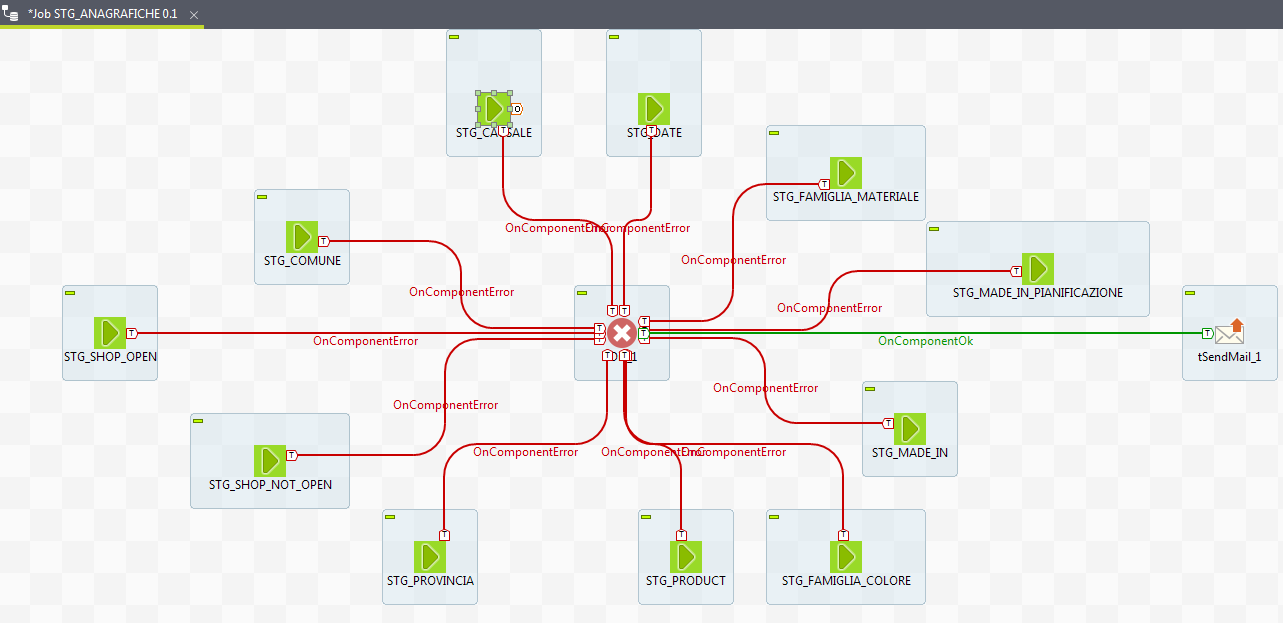
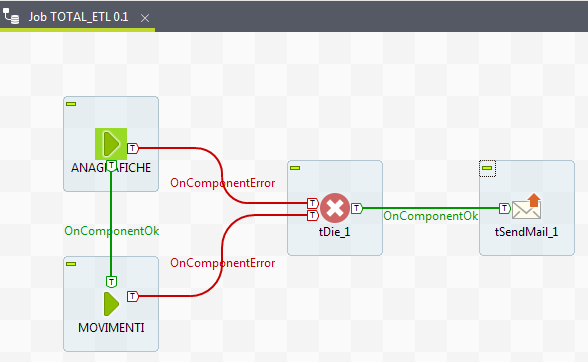


Figure 29: Job STG Anagrafiche

The same procedure is done for the master data and the movements of each level until the L2 level with only two jobs to be joined, respectively, one relating to the master data and to the movements.

As a final step of the ETL process, it has to combine the two types of configuration data with all their hierarchy in one final job. The advantage of Full Load ETL is the ability through a unique command to load entirely a database.

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figure 30: Full Load ETL & Auditing

It is very important to check the flow of data in the various stages of the process. In the figure above because, in addition to the last step is important to focus on two tools: TDIE and tSendEmail. They work together, and when the process is in error, it sent a signal to the TDI e, which by the help of tSendEmail tool, send an email with the attachment in the form of error Script, both the owner and the manager Database.

An example covering the entire explanation process is explained in Appendix A5.

### ***2.7.1 Auditing ETL***

The control in a process of extraction, transformation and loading has the aim to satisfy the following objectives:

* Check the data anomalies as well as controlling just the serious mistakes;
* Capture and store an electronic track of any material changes to the data during processing.

The ETL auditing helps to confirm that there are no anomalies in the data even in the absence of errors. A well-designed auditing mechanism also adds to the integrity of the ETL process by eliminating the ambiguity in the transformation logic, trapping and tracing each modification to the data along the path. Even in the most rudimentary ETL architectures, you can check out some high-level metrics to confirm that the loaded data are those provided.

In general, the ETL auditing processes should ensure the following to confirm that correspond to the input output:

* Counting general of the rows;
* Total aggregate (which may include financial amounts or other summary data).

Some processes require a more comprehensive audit. In other cases, it might be necessary to check whether the data are within reasonable limits or if they support these values. Another aspect that we must not forget to check the cases in which no data has been loaded. Unfortunately, it often happens, and the two main causes are due to a source file that contains no data, an incorrectly configured query that returns no rows or a directory of empty originally intended to contain one or more files could lead to the successful completion of ETL process to load but exactly zero rows of data. However, if a given process should always include an uploaded file number other than zero, be sure to add a control step to check it.

The ETL auditing is rarely the most visible element in the architecture, but it is an insurance policy necessary to protect the integrity of the data and the process.