30/11/2021

SSIS & SQL Server project report

Guillaume Holdorf – Monique Rollier Manuel Gawert – Nelson Lopez

DSTI – S21

Table of Content

[Introduction 2](#_Toc89207751)

[Data 2](#_Toc89207752)

[Pipeline design 2](#_Toc89207753)

[Staging database 3](#_Toc89207754)

[Crimes Table 3](#_Toc89207755)

[Departments Table 7](#_Toc89207756)

[Operational Data Store 9](#_Toc89207757)

[Crimes Table 9](#_Toc89207758)

[Departments Table 15](#_Toc89207759)

[Datawarehouse 18](#_Toc89207760)

[Database design 18](#_Toc89207761)

[Integration of the Date dimension 19](#_Toc89207762)

[Integration of the Departments dimension 21](#_Toc89207763)

[Integration of the Crimes Facts table 23](#_Toc89207764)

[Project Deployment 28](#_Toc89207765)

[Use Case 29](#_Toc89207766)

[Conclusion 30](#_Toc89207767)

# Introduction

To be able to use and extract value from available data, it first needs to be integrated into an IT system. This imply that all the data coming from various sources need to be unified and standardized to be used by other programs. One way to achieve this is to implement a Data Warehouse.

In this project, we are going to design and implement a Data Warehouse with historical data of crimes in France. For this, we will use SQL Server and SSIS.

## Data

Our data is composed of two Excel files.

The first file “Crimes\_in\_France\_1996\_2016.xls” has historical data of crimes in Metropolitan France, from 1996 to 2016. The data is arranged as follows [[1]](#footnote-1):

- Each column indicates the year and month statistics in the format YYYY\_MM

- Each line represents a specific type of crime or offense

- Each tab (sheet) represents a French department

The second file “Departments mapping.csv” has the following data about the “Départements”:

- “Code Postal”, the zip code.

- “Département”, the name.

- “Indicatif Téléphonique”, the first digit of a fixed-line phone number.

- “Région”, the county.

- “Zone Vacances”, the code of zone associated with a holiday schedule.

## Pipeline design

The pipeline will be done in three steps:

* The Staging area (STA) will allow to load the data as is, or with minimal changes.
* The Operational Data Store area (ODS) will allow to clean and standardize the data.  
  If the data don’t pass the quality criteriums, they will be put in the “Technical\_Rejects” table as technical rejects.
* The Data WareHouse area (DWH) will organize the data in one fact table related to multiple dimensions tables. If records can’t be integrated in the schema, they will be put in the “Functional\_Rejects” table as functional reject. Alternatively, some placeholder relations can be created.

There will be one STA and ODS packages per file.

# Staging database

Here, the role of the staging database is to store all the data coming from the different sources. We want to accept all available data.

## Crimes Table

Here is an extract of one sheets of “Crimes\_in\_France\_1996\_2016.xls” :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Index | Libellé index | 2016\_03 | 2016\_02 | 2016\_01 | 2015\_12 | 2015\_11 |
| 1 | Règlements de compte entre malfaireurs | 5 | 3 | 1 | 0 | 6 |
| 2 | Homicides pour voler et à l'occasion de vols | 0 | 0 | 0 | 0 | 0 |
| 3 | Homicides pour d'autres motifs | 1 | 1 | 1 | 2 | 0 |
| 4 | Tentatives d'homicides pour voler et à l'occasion de vols | 0 | 0 | 0 | 1 | 0 |
| 5 | Tentatives homicides pour d'autres motifs | 13 | 11 | 6 | 9 | 11 |

In addition, the data is separated into one sheet by “Département” :



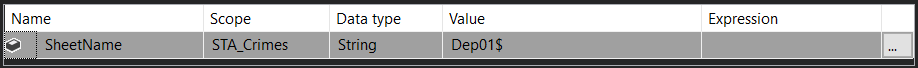
To be able to use the data in the file, we need to put it into one table. This means that we need to extract the data from each data sheet, while also keeping track of the source sheet (“Département”).

To go into all data sheets, we use a “Foreach Loop Container” where we put the extraction dataflow inside. The data flow in the container will be able to load one sheet at the time. Here, we also truncate the data from the previous runs.

Une image contenant texte

Description générée automatiquement

The foreach loop is iterating over a variable “SheetName” that allow to address each sheet separately. The variable is of the format “DepXX$” , with “XX” being the department number.

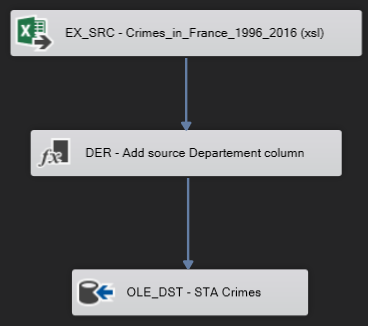


We use the enumerator “ADO.NET Schema Rowset Enumerator” to be able to iterate over the tables of the (Excel) data source.

Une image contenant texte

Description générée automatiquement

Next, the dataflow is defined as follow :



To keep track of the source “Département” sheet, we define a derived column “Departement” using the variable “SheetName” as value.



Once we have extracted the data, we can load it into our target table “Crimes” into the STA database.

First, we check if the table already exists and drop it. This is to prevent interference of the previous runs of the package.

Une image contenant texte

Description générée automatiquement

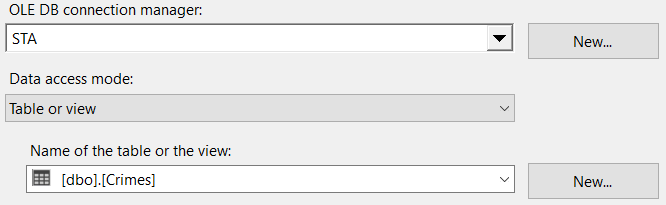
Then we create the table with the following script:

Une image contenant table

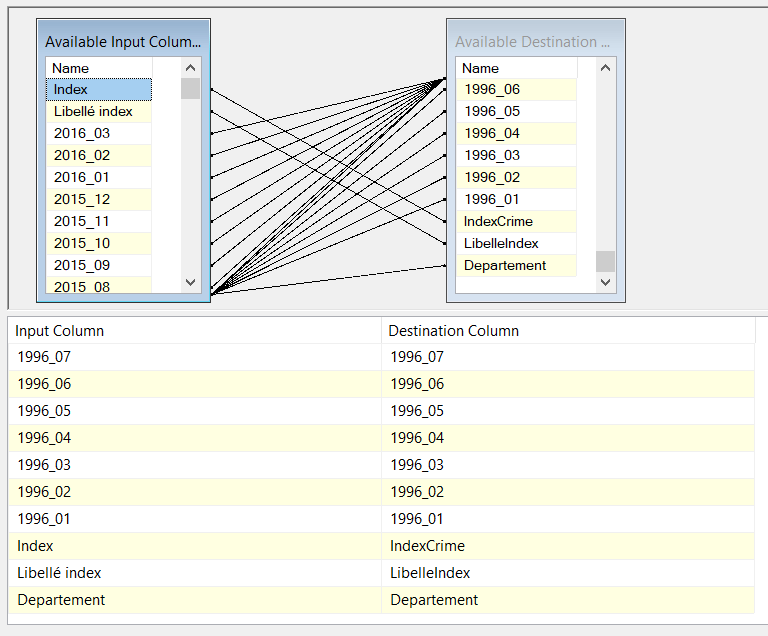
Description générée automatiquement

All values are set to nvarchar(255) for now , to be able to accept all data.

We can now define the target destination.



And finally, we define the columns mapping as follow:



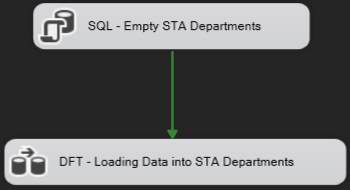
Here is the first ten lines of the results:

Une image contenant texte, rayon, capture d’écran

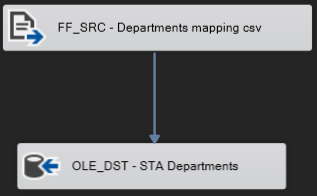
Description générée automatiquement

## Departments Table

Next is the extraction of the data from “Departments mapping.csv”. For this table, we don’t need to add additional data. Here we just make sure to truncate the table “Departments” before running the package:



The dataflow is an import of a flat file :



We need to create the destination table with the following command:

Une image contenant texte

Description générée automatiquement

The data is then loaded into the table “Departments”:

Une image contenant texte

Description générée automatiquement

We also make sure to change some of the names to avoid problems with accents, spaces and SQL reserved words:

Une image contenant table

Description générée automatiquement

Here is the first ten lines of the results:

Une image contenant table

Description générée automatiquement

# Operational Data Store

The second step of the pipeline is to load usable data into the Operational Data Store. This means we need to transform the data into a usable format. We also need to clean and standardize the data. All the data that do not respect the “quality standards “will be rejected as a technical reject.

The quality standards are based on the “correctness” of the data. The output data must me consistent in data types and in values. We also need to ensure that the data can used in queries, so we might need to reorganize and enrich the data.

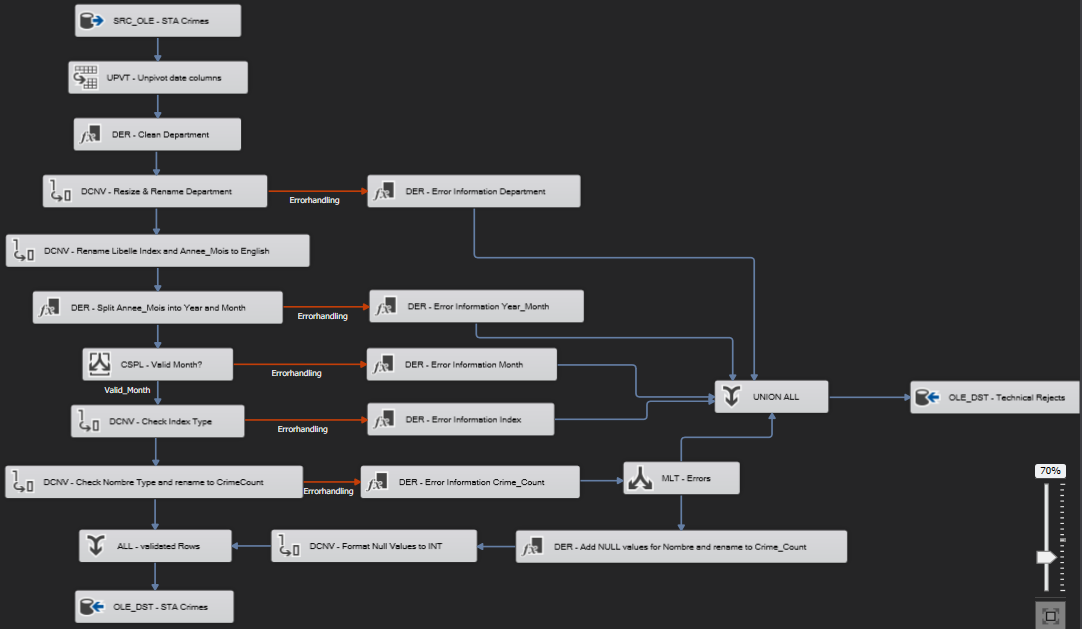
## Crimes Table

Like before, we truncate the data from the previous runs.



Here we can see a warning, this is due to a possible truncation on the error messages.

The data flow is defined as follow:



We are going to detail the three segments of this dataflow.

In the first segment bellow, we change the format of the table into one that is easier to query:

Une image contenant texte

Description générée automatiquement

The initial data format, of the Crimes table, have the time series data organized into columns and the categories are defined in the rows:

|  |  |  |  |
| --- | --- | --- | --- |
| Libellé index | 2016\_03 | 2016\_02 | 2016\_01 |
| Règlements de compte entre malfaiteurs | 8 | 6 | 11 |
| Homicides pour voler et à l'occasion de vols | 3 | 1 | 5 |

The desired output would have one row per date, with the associated category and value:

|  |  |  |
| --- | --- | --- |
| Libellé index | Période | Valeur |
| Règlements de compte entre malfaiteurs | 2016\_03 | 8 |
| Règlements de compte entre malfaiteurs | 2016\_02 | 6 |
| Règlements de compte entre malfaiteurs | 2016\_01 | 11 |
| Homicides pour voler et à l'occasion de vols | 2016\_03 | 3 |
| Homicides pour voler et à l'occasion de vols | 2016\_02 | 1 |
| Homicides pour voler et à l'occasion de vols | 2016\_01 | 5 |

This operation is called “Unpivot” and we apply it to the date columns:

Une image contenant table

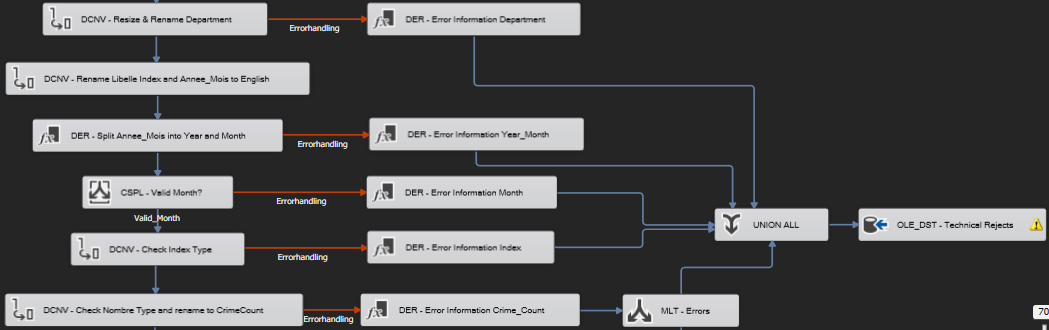
Description générée automatiquement

For this, we need to define wich collumns will be expanded. In our case, we need to expand the dates and move the data to a new column “Nombre”. We also rename the date column “Annee\_Mois”.

Then, we clean the “Department” column that has still the format of the variable. We only keep the two characters of the department (because of “2A” and “2B”, we choose characters):



In the second segment, we are cleaning and enriching the data.



We have multiple checks and the errors are redirected to the “Technical\_Rejects” Table created as follow:

Une image contenant texte

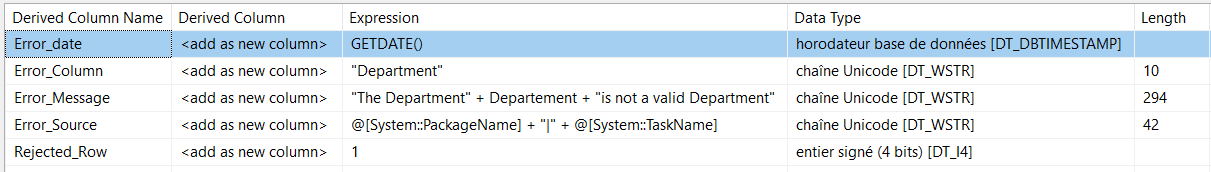
Description générée automatiquement

First, we rename and resize the “Departement” column. To keep track of the changes and distinguish the columns, we add a suffix with the change in length or type.

Une image contenant texte

Description générée automatiquement

If that step fails, we track the error and put it in the “Technical\_Rejects” table.



As seen above the data inserted in the technical rejects are:

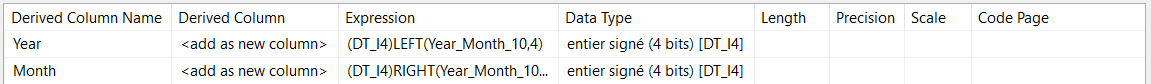
* The date of the error.
* The column making the error.
* An error message.
* The package and the task causing the error.
* A rejection code: 1 for reject; 0 for a warning.

Next, we rename and resize the columns

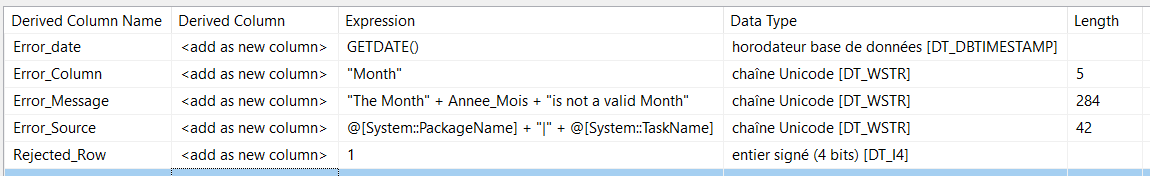
Une image contenant table

Description générée automatiquement

Then, we enrich the data by splitting the column “Year\_Month\_10” into “Year” and “Month” columns. We explain this choice in the Data Warehouse section.



In case of an error, we track it like before:

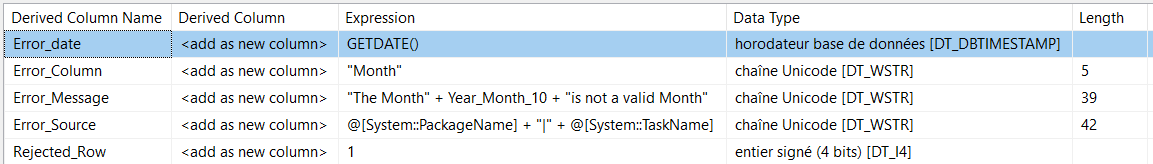


We also do an additional check on the months to make sure that the value makes sense (range 0-12):

Une image contenant table

Description générée automatiquement

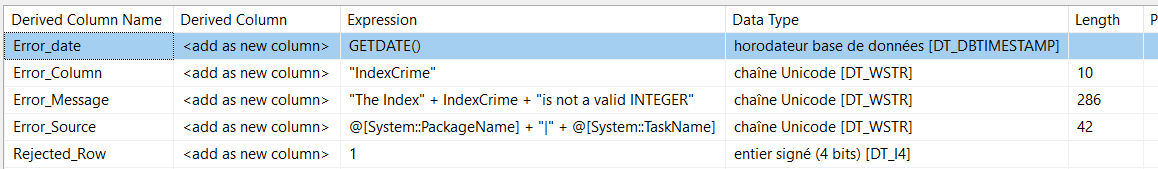
In case of an error:



Next, we resize the “IndexCrime” column:



In case of an error:



Finally, we rename and resize the “Nombre” column containing the numerical data about the crimes. The last check is of Warning level, so we integrate it in the database with a null value. We choose to do this to not lose the record as it could generate errors in the data integration (see DWH\_Crimes) and we can still manage it in the queries.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

In the last segment, we want to insert a null value. So, we first generate the “NULL” then we convert it to the destination type.

Une image contenant texte, périphérique

Description générée automatiquement





Now we can create the ODS “Crimes” table as follow:

Une image contenant texte

Description générée automatiquement

The last task is to insert the data in the ODS database:

Une image contenant texte

Description générée automatiquement

The final mapping is:

Une image contenant table

Description générée automatiquement

Here is the first ten lines of the results:

Une image contenant table

Description générée automatiquement

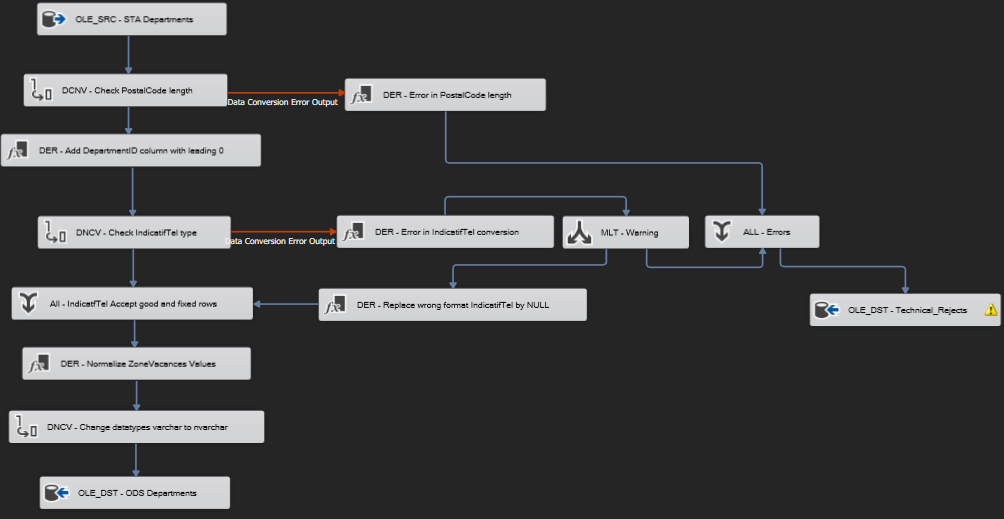
## Departments Table

Now we need to process the data from the “Departments” table. We also have a warning for the same reason as before.

Une image contenant texte

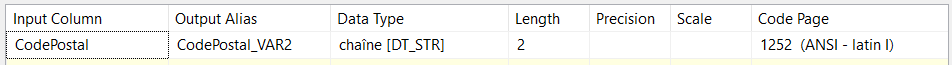
Description générée automatiquement

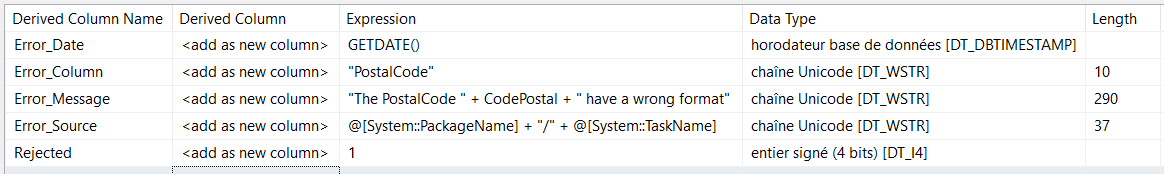
We process here is similar, we process the data, collect the errors, and reintegrate some of the data into the table with null values.



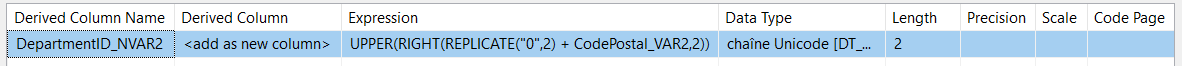
We import the data from the staging area, then the first processing step is to clean some of the data.

First, we resize the reference of the department at two characters and handle the associated errors.

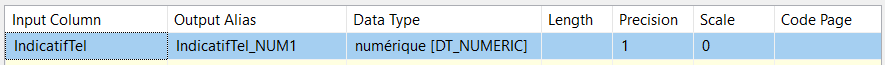


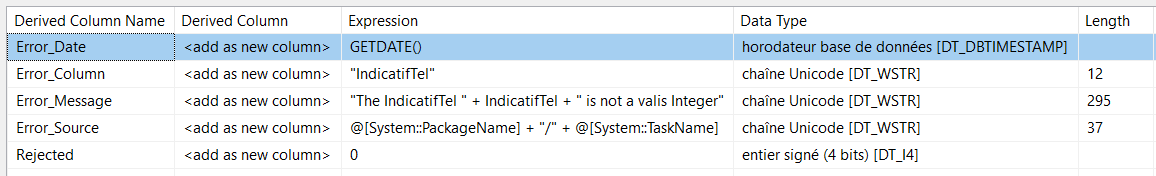


Next, to have a valid zip code, we need to add a leading zero.



Then we process the phone number data.

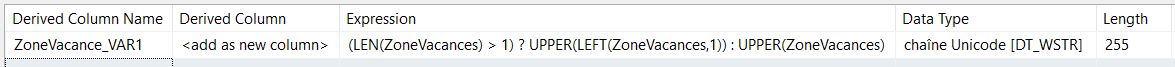




Here, we consider that the leading “0” can be replaced with “+33”. So, the real data is one number. In addition, if we have an error, we replace it with a null value to not lost the entire department.



Then we normalize the holiday areas as one upper character (i.e., “S” instead of “Special”).



The final cleaning step is to make sure that all the text data is of nvarchar type.

Une image contenant texte

Description générée automatiquement

Finally, we can create the ODS “Departments” table with the following script:

Une image contenant texte

Description générée automatiquement

Then, we load the data into the table.

Une image contenant texte

Description générée automatiquement

The final mapping is as follow:

Une image contenant table

Description générée automatiquement

Here is the first ten lines of the results:

Une image contenant texte, capture d’écran, intérieur

Description générée automatiquement

All the errors are sent to the same “Technical\_Rejects” table.

# Datawarehouse

The last step in the data pipeline is to integrate the data into the Data WareHouse. One common database schema is the Star schema. In this schema, we have one main table called the “Fact Table” That is surrounded by “Dimension Tables”. The fact table contains the most important data called “facts”, whereas the dimensions tables give addition descriptive information. We will design our database around this schema.

## Database design

For our data, we can consider that the important data are the statistics about the crimes. Therefore, we choose to build the fact table with the “Crimes” table.

For the dimensions, one common dimension is the “Date” dimension. It allows to describe the dates with multiple temporal aggregate categories (year, month, quarter). The table will contain multiple variation of those three data to be able to adapt to various styles of queries. We define the relation to the fact table with a technical key with the format “YYYYMM”.

The second dimension we choose is the Department dimension. It will help to describe geographical data. We choose to use an incremental indices technical key for the relation to the fact table. There is also a business key with a two-character zip code of the department.

We could consider an additional dimension describing the crimes in more details. However, with the current available data, we choose to keep everything in the fact table.

## Integration of the Date dimension

This dimension is describing dates, in particular months. Since we don’t need external data to build this dimension, we use an TSQL script to make it.

First, we create an empty table:

Une image contenant texte

Description générée automatiquement

The available data with different formats are:

* MonthKey, the technical key.
* Month (1,2,3…)
* MonthName (January, February, March…)
* MonthName\_Short (JAN, FEB, MAR…)
* MonthName\_FirstLetter (J, F, M…)
* Quarter (1,2…)
* QuarterName (first, second…)
* Year (1996, 1997…)
* MMYYYY (011996, 021996…)
* YYYY\_MM (1996\_01, 1996\_02)
* MonthYear (1996JAN, 1996FEB…)

To build that data we use the following script:

Une image contenant texte

Description générée automatiquement

Here is the first ten lines of the results:

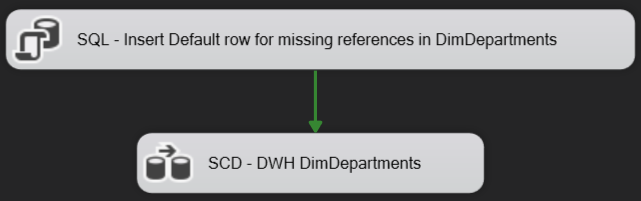
Une image contenant table

Description générée automatiquement

To accept new data in the data warehouse, this dimension can be updated before the beginning of a new year. We would need to change the “@EndDate” variable.

## Integration of the Departments dimension

In ODS we have clean data about the departments. However, there is the possibility that an error in the fact table could prevent to make a link to the fact table. To handle this, we add a default (-1) value for the department key.



First, we need to create the table.

Une image contenant texte

Description générée automatiquement

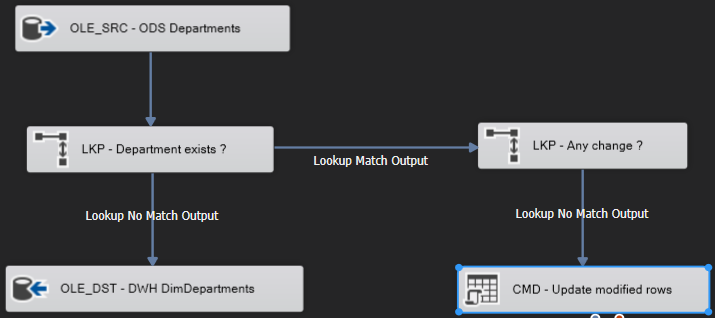
We decide to create a row that will be a reference for any unknown department found in other data.  
It is created at pre-sql step of Department Dimension.   
We decide to give it a special DepartmentKey -1.

Script testing if this DepartmentKey exist and create it if not found:

Une image contenant texte

Description générée automatiquement

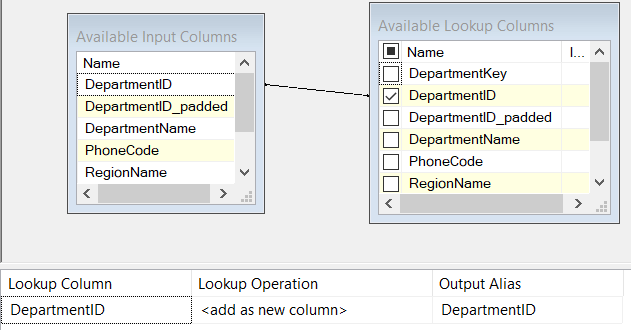
Then we can load the data with the process bellow:



We need to make sure that we can make joins between the fact table and the dimension table, so we check the link with “DepartmentID”.

Une image contenant texte

Description générée automatiquement



For the dimensions tables, we also need to have an update policy. We choose the SCD1 strategy, implemented by checking is there is any change and updating the table if necessary.

Une image contenant table

Description générée automatiquement

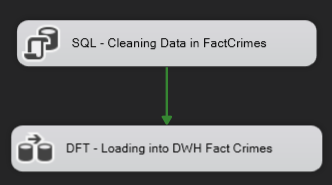
Here is the first ten lines of the results:

Une image contenant texte

Description générée automatiquement

## Integration of the Crimes Facts table

Now that we finally have our dimensions tables, we can build our fact table while checking valid relations with the dimensions.



Delete strategy:

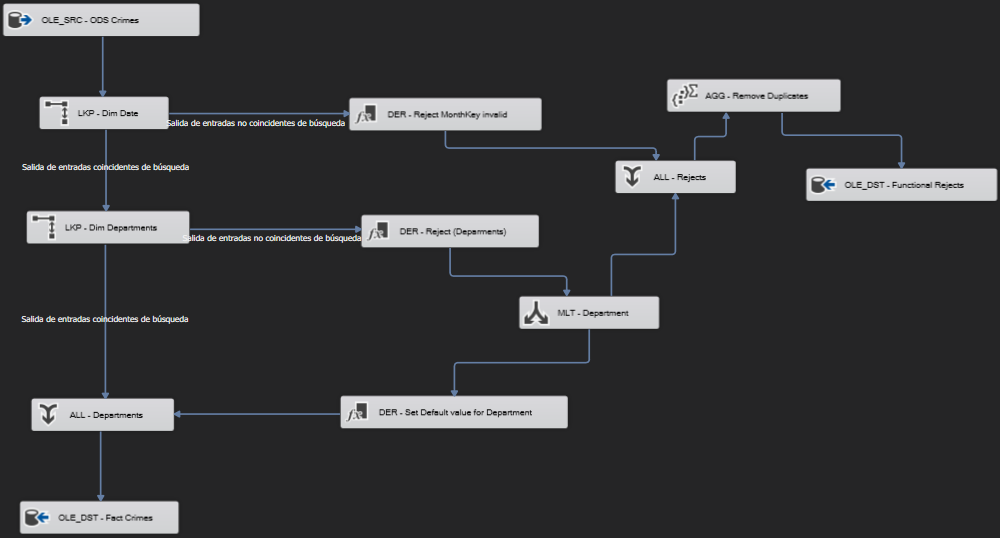
As this is a fact table it is supposed to cumulate historical data with new data coming based on a regular basis. The original spreadsheet is organized by departments and by months, probably the data ingestion is made on a monthly basis. But the spreadsheet presents much more than a month of data.  
We choose to consider that data about a month X coming in the spreadsheet can be either new data or updated data. So, a month Y not present in the spreadsheet have to be consider as stable in the fact table.   
What about the departments? can we consider that if a department is missing in next spreadsheet, we have to keep the data in Fact table about it?   
We would have answer Yes but it requires to be able to identify the department to be able to delete only those coming from the new spreadsheet. But we don’t keep the original Crimes department information in the fact table, we replace it by a surrogate key to “DimDepartment” table.  
We choose to delete in “FactCrimes” table the data based on “Year\_Month” present in ODS Crimes table.

Graphical user interface, text, application

Description automatically generated

This strategy will ensure not to duplicate data coming this month with data coming next month.

Once we made sure that we have a clean table to insert the data into the data warehouse, we verify and load the data as follow:

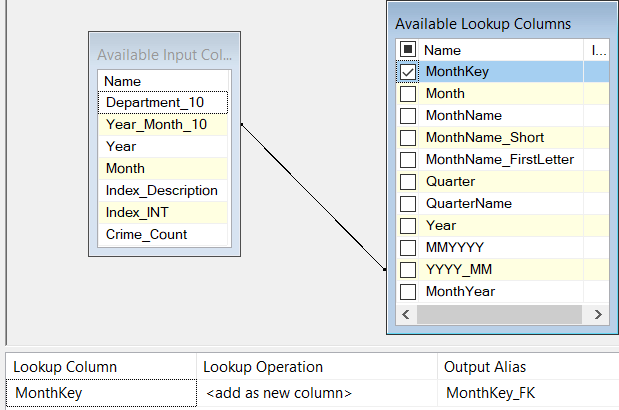


In case of an error in the processing, we generate a functional reject and insert it into a table “Functional\_Rejects” created by the following script:

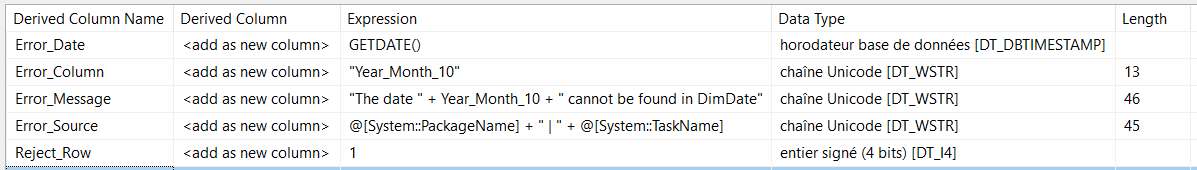
Une image contenant texte

Description générée automatiquement

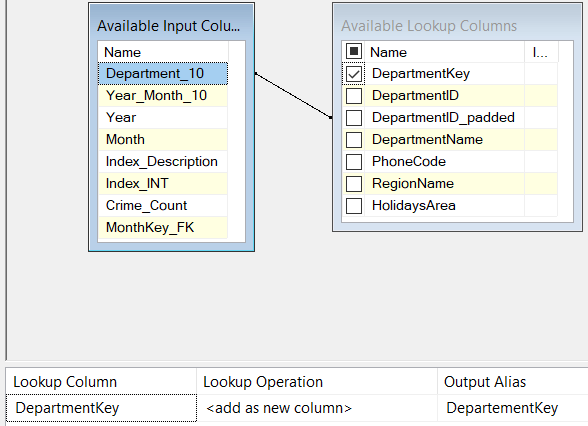
The first dimension we check is “DimMonth”. We check it with the business key between “Year\_Month\_10” and “YYYY\_MM”. We also add a technical key “MonthKey\_FK”.



We tack the errors with the same method that in ODS. For a missing relation with “DimMonth”, we generate the following data:



Similarly, we check the relation with “DimDepartment” with the business key “Department\_10” and “DepartmentID\_padded”. We also add the technical key “DepartmentKey”.

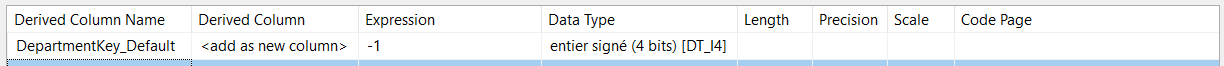


In case of an error, we have the following:

Une image contenant texte

Description générée automatiquement

In this case, we will integrate the record with the default value (-1) for the Department.



Since we have a lot of records, a missing department can generate a lot of data. To alleviate this problem, we check for duplicates during insertions in “Functional\_Rejects”.

Une image contenant table

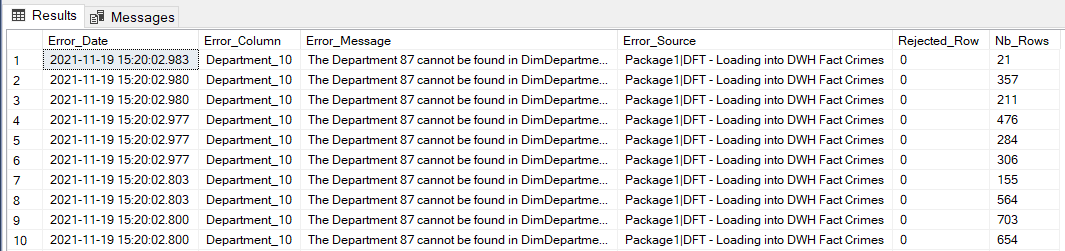
Description générée automatiquement

As shown here, we are keeping track of the number of duplicates:

Une image contenant table

Description générée automatiquement

Here is the first ten lines of the results of “Functional\_Rejects”:



Finally, we can create and load the fact table into the data warehouse.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

During the mapping we do a final change to the names.

Une image contenant table

Description générée automatiquement

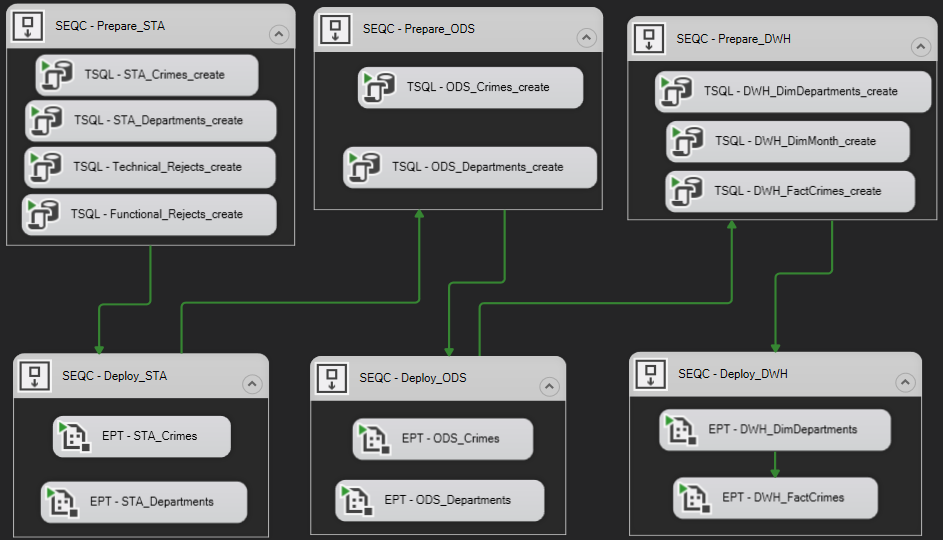
Here is the first ten lines of the results of the facts table “FactCrimes”:

Une image contenant table

Description générée automatiquement

# Project Deployment

We have defined all required steps to deploy our data warehouse with the available data. To ensure that the pipeline is executed in a reproducible manner, we define an additional package “ETL\_Pipeline” with the DAG of the tasks.



Using this package allow to use one interface for the entire pipeline. The creation of the tables is done with TSQL tasks and the package execution tasks launches the data flows.

## Use Case

Once we have deployed the data warehouse, we can start to query it for making analysis reports. For instance, we can generate a view that can then be used in a BI tool to report on the geographical and temporal distribution of crimes.

The following view get geographical data from “DimDepartment” and temporal data from “DimMonth”.

Une image contenant texte

Description générée automatiquement

Then, we aggregate by month and region to analyse if there is any “seasonal” pattern and variation across regions. Here we could use numerical and literal labels for months and crimes categories.

Here is the first ten lines of the results of the facts view “V\_All\_Crimes”:

Une image contenant table

Description générée automatiquement

# Conclusion

We made some choices:

Months

This table is a regular time dimension. Here, we limit the grain to month because the fact data is at this granularity. All possible values are known, so it’s not necessary to create a reference for unknown month.

Department

This table is Dimension data. It is almost stable. Therefore, we consider that it is not necessary to keep any old version of data  
We can’t truncate the whole “DimDepartment” table because it is reference for other tables. So, it is necessary to merge existing data with new one to accept changes.  
It is also necessary to create a special key for unknown Department found in Crimes.

Crimes

This table is the fact table. Our design choice was to replace the value for month and year by the computed key “YYYYMM” of “DimMonths”. The Department reference is replaced by the surrogate “DepartmentKey” found in “DimDepartment”, with a default value for missing references.

This data model we can answer questions about Crimes using the two dimensions: Time and Department. By using different levels of aggregations, we can extract data per quarter or year and per region. Views are a good tool to prepare data for BI tools.

Future expansions

In the future, it is possible that crimes will be classified by consequences like “prison” and/or “fine”, or regrouped in categories “aggressions”, “vols”, …  
Having more information about crime types will drive to create a new dimension “DimCrimesTypes”. Then, we would need to change the “FactCrimes” table structure to reflect this new dimension.

If we want to create it today, we would create a new package using ODS “Crimes” table. This package will extract crimes index and descriptions and make a set of distinct crimes to form it as a dimension. This would require a lot of checks to unify it correctly. The merge of data coming from ODS “Crimes” with a “DimCrimesType” will be a source of issues too.

Another expansion can be a fact table of contacts by department. The current design will adapt easily to this kind of evolution.

1. « Subject of the project.docx », Emerick Duval. [↑](#footnote-ref-1)