Sergio Hervás Aragón

Sergiohervas9@gmail.com

Brief description

This project is a case study of a Business Intelligence (BI) solution for a telecommunications multinational. It involves designing the data analysis system architecture, modeling the database, loading extracted data, and performing data exploitation to support decision-making.

Final practice

The job of the data scientist

Index

Content

[1. Analysis of initial data and proposed data model. 2](#_Toc189044172)

[1.1. Understanding initial data 2](#_Toc189044173)

[1.2. Data processing 3](#_Toc189044174)

[1.3. Design of the proposed data model 4](#_Toc189044175)

[1.4. Why this model? 5](#_Toc189044176)

[2. Model development in MySQL 6](#_Toc189044177)

[2.1. Creating tables and defining keys 6](#_Toc189044178)

[2.2. Loading initial data 8](#_Toc189044179)

[3. Developing loads with Pentaho Data Integration 9](#_Toc189044180)

[3.1. Staging 9](#_Toc189044181)

[3.2. Masters 14](#_Toc189044182)

# Analysis of initial data and proposed data model.

## Understanding initial data

Dataset data:

* **Fecha de creación**: Specifies the time when the ticket is created. It will be used to perform the temporal analysis.
* **Numero de incidente**: Unique ticket identifier.
* **Descripción**: Brief description of the request or incident.
* **Servicio**: The different services that the company provides to its users and clients.
* **Tipo de servicio**: Specify whether it is a request or an incident:
  + **Incidencias**: Restoration of infrastructure or restoration of service to user.
  + **Peticiones**: Service request by user
* **Prioridad**: Specifies the priority with which the ticket should be treated.
  + There are four priorities, ordered from most important to least: Critical, High, Medium, and Low.
  + Only incidents can have critical priority and should only be considered in production environments.
* **Estado**: Indicates the status of the ticket.
  + A ticket is considered finished if it is in Closed or Resolved status.
  + A ticket is considered open if it is pending or assigned.
  + Cancelled tickets do not count.
* **Torre**: It is a functional grouping of services. Each service belongs to a management tower.
* **Entorno**: Indicates the environment that the ticket affects. Production environments are all those that contain a PRO.
* **Estado cumplimiento**: Indicates whether the ticket has been resolved within the time required by the client or not.
  + Within the service objective: ticket resolved appropriately
  + Service objectives not met: ticket resolved late

**Duración días**: Number of days it took to resolve the ticket.

## Data processing

An analysis of the data is performed to verify nulls, and/or data that may be erroneous, in a first instance of .isnull():

Imagen que contiene texto, grande

Descripción generada automáticamente

With the empty data, proceed by columns to:

* Descripción: Nulls are filled with the value “No description”.
* Servicio: Nulls are filled with the value “Unspecified”.
* Torre: Nulls are filled with the value “Unspecified”.
* Entorno: Nulls are filled with the value “Unspecified”.
* Estado cumplimiento SLA: Nulls are filled with the value “Unspecified”.

Imagen que contiene texto, tabla, calle, grande

Descripción generada automáticamente

## Design of the proposed data model

After analyzing the data, the star model was chosen, which will consist of a central fact table that is related to several dimension tables:

* Fact table (fincidencias): Table which will contain numerical information about the incidents.
* Dimension tables:
  + dsevicio: Contains details about the services.
    - id
    - servicio
  + dtipo\_servicio: Contains details about service types.
    - id
    - tipo\_servicio
  + dprioridad: Contains details about priority.
    - id
    - prioridad
  + destado: Contains details about the status.
    - id
    - estado
  + dtorre: Contains details about management.
    - id
    - torre
  + dentorno: Contains details about the environment.
    - id
    - entorno
  + dsla: Contains details of the compliance status.
    - id
    - sla

## Why this model?

The use of this model is based on the use of the term ‘DISTINCT’ in MySQL queries by columns of our Dataset, and based on the data obtained, if it is a large amount, the column is discarded as an identifier:

* fecha\_creacion: 58376 different records, so it is not in our model.
* num\_incidente: 58564 distinct records, so it is not in our model.
* descripción: 51925 distinct records, so it is not in our model.
* servicio: 114 different records, so it is in our model.
* tipo\_servicio: 4 different records, so it is in our model.
* prioridad: 4 different records, so it is in our model.
* estado: 5 different records, so it is in our model.
* torre: 19 different records, so it is in our model.
* entorno: 881 different records, so it is in our model.
* sla: 3 different records, so it is in our model.
* duración\_dia: 55467 distinct records, so it is not in our model.

# Model development in MySQL

## Creating tables and defining keys

Based on the chosen model:

* Dimension tables:
  + dentorno:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| identorno | INT | X | X |  |  |  |  |  |  |
| entorno | VARCHAR |  |  |  |  |  |  |  |  |

* + destado:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idestado | INT | X | X |  |  |  |  |  |  |
| estado | VARCHAR |  |  |  |  |  |  |  |  |

* + dprioridad:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idprioridad | INT | X | X |  |  |  |  |  |  |
| prioridad | VARCHAR |  |  |  |  |  |  |  |  |

* + dservicio:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idservicio | INT | X | X |  |  |  |  |  |  |
| servicio | VARCHAR |  |  |  |  |  |  |  |  |

* + dsla:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idsla | INT | X | X |  |  |  |  |  |  |
| sla | VARCHAR |  |  |  |  |  |  |  |  |

* + dtipo\_servicio:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idtipo\_servicio | INT | X | X |  |  |  |  |  |  |
| tipo\_servicio | VARCHAR |  |  |  |  |  |  |  |  |

* + dtorre:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| idtorre | INT | X | X |  |  |  |  |  |  |
| torre | VARCHAR |  |  |  |  |  |  |  |  |

* Fact table:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| id | INT | X | X |  |  |  |  |  |  |
| identorno | INT | X | X |  |  |  |  |  |  |
| idestado | INT | X | X |  |  |  |  |  |  |
| idprioridad | INT | X | X |  |  |  |  |  |  |
| idservicio | INT | X | X |  |  |  |  |  |  |
| idsla | INT | X | X |  |  |  |  |  |  |
| idtipo\_servicio | INT | X | X |  |  |  |  |  |  |
| idtorre | INT | X | X |  |  |  |  |  |  |

## Loading initial data

When loading the data, it involved creating a table that reflected the structure of the data contained in the XLS file, so a table was designed with column names that matched the column headers in the XLS file, and the data types of each column were adjusted accordingly.

Initial data loading table (staging):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G |
| fecha\_creacion | DATETIME |  |  |  |  |  |  |  |  |
| num\_incidente | VARCHAR |  |  |  |  |  |  |  |  |
| descripción | VARCHAR |  |  |  |  |  |  |  |  |
| servicio | VARCHAR |  |  |  |  |  |  |  |  |
| tipo\_servicio | VARCHAR |  |  |  |  |  |  |  |  |
| prioridad | VARCHAR |  |  |  |  |  |  |  |  |
| estado | VARCHAR |  |  |  |  |  |  |  |  |
| torre | VARCHAR |  |  |  |  |  |  |  |  |
| entorno | VARCHAR |  |  |  |  |  |  |  |  |
| sla | VARCHAR |  |  |  |  |  |  |  |  |
| duración\_dia | DOUBLE |  |  |  |  |  |  |  |  |

# Developing loads with Pentaho Data Integration

## Staging

In the first data loading process, we will try to work with the information in our Dataset, obtain the data, and transfer it to MySQL for storage.

Components:

**Excel Entry**: Component which we use to load the data from our Excel file:

* We will look for the ‘Microsoft Excel Input’ component in our design column in the Input section and drag it to the canvas:

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

* Once dragged, we will double-click on the component with the left mouse button, where we will change the name of the component to the XLS file that we are going to use, and in the file or directory section, we will have to add the file that we are going to work with:

Captura de pantalla de computadora

Descripción generada automáticamente

Within the component you can find sections such as:

* Sheets: Tab where you can see the sheets of the imported file.
* Content: Tab where you can adjust the table settings.
* Fields: Tab where you can see the columns of the file.

**Output Table**: Component which we use to load the data from our Excel file into MySQL:

* We will look for the ‘Output Table’ component in our design column in the Output section and drag it to the canvas. Once done, with the mouse wheel, we will join the input component with the output component, to work with the data, we receive, and once all this is done, we will double-left click on the output component:

Captura de pantalla de computadora

Descripción generada automáticamente

* Once inside the component, we will set the step name to the name of our table in MySQL, then we will set the connection, filling in the Host Name, Database Name, UserName and Password, we will click on Test to make sure our connection is successful:

Interfaz de usuario gráfica

Descripción generada automáticamente

* Going back, now that we have a connection to the database, we will examine the destination table, where the data will be saved:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* We will accept, and we will return to the table output screen, where we will have already completed all the connection issues. Now we will activate the empty table section (to avoid overwriting), and the specify fields section, and in the Database fields tab, by clicking on ‘Enter Field mapping’, we will relate the fields we received with those in our database:

Captura de pantalla de computadora

Descripción generada automáticamente

* Final result of our staging:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## Masters

In the second process, although it has already been discussed, we will look at how we will try to fill the dimension tables already created in our database with information.

After a first study of the data, a search for nulls has been carried out to verify which columns had a null value, which may be:

* Descripción
* Servicio
* Torre
* Estado cumplimiento SLA

**Table Entry:** Component which we use to access data from our staging table in our database, after having filled it with information in the staging.

* First, we will call this component with the name of the column from which we want to obtain the information, then we will connect to our database, and enter the statement to obtain the desired data, in this case we expect to receive the unique data from the service column, so we do a search within our general table (staging table):

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tables that contain nulls use a different query model than those that do not contain nulls:

* With Nulls:

SELECT DISTINCT case WHEN ‘Column Name’ IS NULL THEN 'Unspecified'

ELSE ‘Column Name’

END AS ‘Column Name’

FROM staging;

* No Nulls:

SELECT DISTINCT ‘Column Name’ FROM staging;

**Obtain value from database sequence:** Component which is responsible for assigning an identifier to the obtained values, easy to use, where the only modifications we will have to make will be the name of the step, the name of the component, and the name of the value, which will be the name with which it is saved in our database.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Between component and component, these will have to be linked together so that the information we receive is passed from component to component and thus we can work with them.

**Output Table:** Component which in this case we will use to load the data into MySQL:

* We will complete the data as we have seen previously, we change the step name to the name of the destination table in the database, we configure the connection, we select the destination table, we empty the table and we specify the data:

Captura de pantalla de computadora

Descripción generada automáticamente

Once this chain of processes is completed, we will repeat it with each branch of our model, exchanging names, destination tables, statements, etc., with their respective names.

* Final Result of our Masters:

Captura de pantalla de computadora

Descripción generada automáticamente