Professor José Curto Díaz

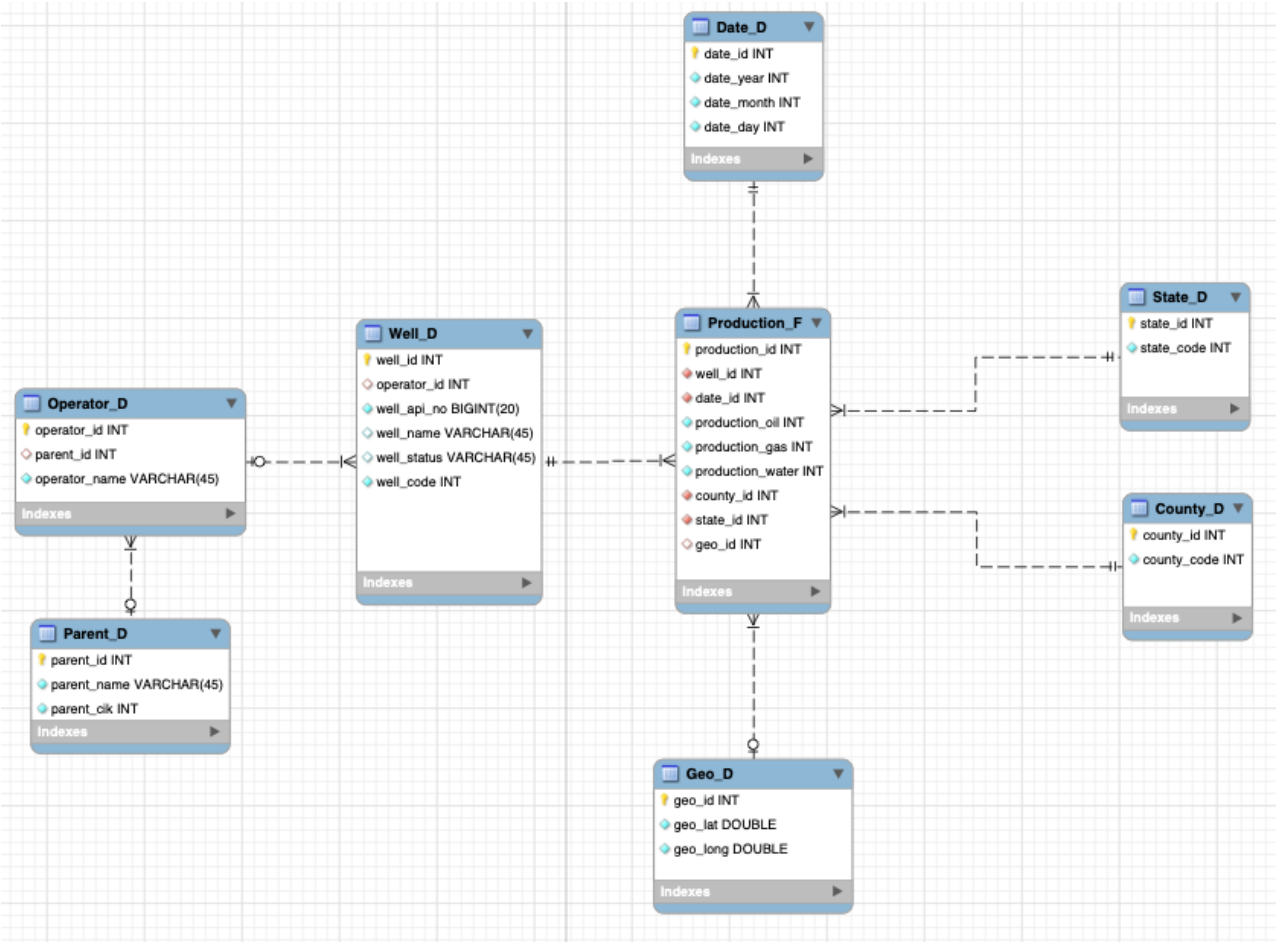
ETL WORKGROUP

MBD 0-1-1

Madrid, November 25th, 2018

Table of Content

1. UPDATED DATA MART MODEL



2. EXTRACTING STRATEGY

Default strategy

Working with two .json files, we were able to extract the data with the native data extractors, available in PDI. Adding the file/directory *${Internal.Entry.Current.Directory}/input/wells_mexico_all.json* and the fields of the JSON file to this input step, we are able to start preprocessing its tables. The same goes for the *production_mexico_all dataset*.

Organisation of Jobs and Transformations

The ETL process will consist of one parent job (*WELLS_PRODUCTION_ETL*) which will contain two jobs: first, a ***DATA_PREPARATION & DIMENSIONS*** job, and second a ***FACT TABLE JOB***. Opening these jobs will bring us to different transformations, discussed in more detail in the next paragraphs.

3. DATA MAPPING PROCESS

A. DATA_PREPARATION

The first step in the *Data_Preparation_Dimensions_tables* job will be to clean the two datasets. This will happen in one transformation, called ***DATA_PREPARATION***. We envision this transformation to have three parts. Note that it will still be a single transformation, in which all following steps will be executed.

a. PREPARATION OF WELLS

We input the JSON file *well_mexico_all* and we extract from *api_number* the *state_code_w* (first two digits), the *county_code_w* (from 3rd to 5th digit), and the *well_code_w* (last 5 digits). At the end of the process, we sort by *api_number* in ascending order, in order to make the merge process faster.

Derived fields:

- ***state_code_w***
- ***county_code_w***
- ***well_code_w***

b. PREPARATION OF PRODUCTION

Secondly, we will input the JSON file *pruduction_mexico_all* and. In the original production dataset, we find the month table. In order to properly be able to process our data, we will, therefore, create three new attributes, called *p_year*, *p_month*, and *p_day*, which respectively are the year, month and day of production. As we will have done for the *wells_preparation* transformation, at the end of the process, we sort the data by *api_number* in ascending order.

It was found that there are ***53 production records*** for ***8 api_numbers*** that ***don't exist*** in the *well_mexico_all* dataset. Therefore we decided to derive *state_code_p*, *county_code_p*, and *well_code_p* as well here.

Derived fields:

- ***state_code_p***
- ***county_code_p***
- ***well_code_p***
- ***p_year***
- ***p_month***
- ***p_day***

c. MERGING THE TWO PREPARATIONS

It was found that there are ***5 production records*** for the *api_number* ***3501539188***, which ***are not in New Mexico state***.

In addition to the ***53 production records*** for that ***don't exist*** in the *well_mexico_all* dataset.

Finally, there are ***42742 wells records without any production data*** in the production dataset.

Therefore, we decided to perform an ***Outer Merge*** for wells & production data, which means the final result will have 3 types of records:

i. Rows ***with well data*** (name, operator, parent, status, geo coordinates) and ***no production data*** (month, year, day, oil, water, gas)
ii. Rows with ***both well data and production data*** for a specific date.
iii. Rows ***with production data and no detailed well info*** (Only API number and derived codes)

We merge by *api_number* the two files with an outer join in order to create a ***merged_production.csv*** file in the ***/output*** directory. After merging we replace all numeric fields with *NULL* values to 0 (*oil*, *gas*, and *water, latitude, longitude)*.
Additionally, we wrote a Java Expression to pick the *api_number*, *state_code*, *county_code*, and *well_code* form the merged output:

Java Expressions:

```
state_code_w == null ? state_code_p :
state_code_w
county_code_w == null ? county_code_p :
county_code_w
```

Finally we save the data in ***/output/merged_production.csv*** to be used in later jobs and transformations.

B. WELL_DIMENSION

We input the *merged_production.csv* file and since we have two types of wells records here:

1. ***Wells with complete detailed info*** (*well_name*, and *operator_id*)
2. ***Wells with only api_number*** retrieved from the 53 production records with no well data.

We **Filter Rows** by well_name to differentiate between the two types and for the records with well data we **lookup** the **operator_id** from the Operator_D dimension and then sort data and pick unique rows to insert into Well_D.
For wells with minimal info, we sort and pick unique values then insert right away **without lookup**.

Result: 101738 rows

C. OPERATOR_DIMENSION

We import the file *merged_production.csv*, and we filter by keeping only the rows where the *parent* is *NOT NULL* and the *cik* is *NOT 0*. We look up in our *Parent_D* table the fields *parent_name* and *parent_cik*, and we match them with *parent* and *cik* in the input .csv file. We generate a new auto incremental variable

parent_id in the stream, and we select the values *operator_name* and *parent_id*, sorting operator_name both in ascending order. We remove the duplicates from these fields and we insert the selected variables into the database in the table *Operator_D*.

Result: 1391 rows

D. PARENT_DIMENSION

We import the file *merged_production.csv*, and we only select the fields *parent* (renamed *parent_name*), and *cik* (renamed *parent_cik*). Then, we sort by *parent_name* in ascending order, and *parent_cik*, also in ascending order. We remove the duplicates from these fields and we keep only the values where *parent_cik* is *NOT 0* and *parent_name* is *NOT NULL*. We insert the selected variables in the table *Parent_D* in our database.

Result: 24 rows

E. GEO_DIMENSION

We import the file *merged_production.csv*, and we filter the rows where *latitude* and *longitude* are *NOT NULL* or *NOT 0*. Then we select the values *latitude* (renamed to *geo_lat*) and *longitude* (*geo_long*). We sort first by *geo_lat* and then *geo_long*, both in ascending order. We remove the duplicates by filtering on the unique pairs of latitude and longitude. Finally, we insert the selected values into the table *Geo_D* of our database.

Result: 96568 rows

F. STATE_DIMENSION

We import the file *merged_production.csv*, selecting only the *state_code* and sorting it in ascending order. Then, we only keep the unique rows of this field. Then, we filter the rows where *state_code* is *NOT NULL* and we insert the selected values into the *State_D* table in our target schema.

Result: 2 rows

G. COUNTY_DIMENSION

We import the file *merged_production.csv*, selecting only the *county_code* and sorting it in ascending order. Only the unique rows of this field will be kept. Then, we filter the rows where *county_code* is *NOT NULL* and we insert the selected values into the *County_D* table in our target schema.

Result: 31 rows

H. DATE_DIMENSION

We import the file *merged_production.csv*, selecting only the values *p_year*, *p_month*, and *p_day*, renaming them, respectively, *date_year*, *date_month*, and *date_day*. Then, we sort the rows by *date_year*, *date_month*, and *date_day*, all in ascending order. We remove the duplicates from these fields, and we filter the rows where *date_year*, *date_month*, and *date_day* are *NOT NULL*. Last, we insert the selected values into the *Date_D* table in our target schema.

Result: 2 rows

I. PRODUCTION_FACT

The Fact Table transformation integrates the dimension tables with the Production_F table. We do this by first filtering by date values to remove rows without production data (*oil*, *water*, *gas*) and then looking up the FKs from all dimension tables. These are the *well_id*, *state_id*, *county_id*, *geo_id* and if they exist, *date_id*. The values are then filled in when the FKs are known. The last step is importing the attributes, which are in this case the production variables *oil*, *gas*, and *water*.

Result: 1,195,489 rows

4. DATA QUALITY & METADATA APPROACH

Data quality

Through the ETL transformations, we managed to elevate the quality of our data in several ways. A first crucial improvement is the separation of the month field in the *production_all_mexico* to the separate fields year, month and date.

Also, we made sure to represent any missing data by the appropriate data type and value such as NULL in FK fields for missing entities like parent or geo, and filling with 0 for missing production data.

Metadata Approach

At first, we considered adding a table with the source file as a relevant metadata approach. However, after the decision to merge the wells & production files with an outer join, adding this metadata would be redundant. It is also clear, when there are particular missing values, if the source file is

- the wells table: production variables are null because no API number was matched with the one in the wells table (42742 cases in total)
- the production table: all well variables are null because no API number was matched with one of the production table (53 cases in total)

The calculation of the amount of cases is based on the following information.

- From production, we query 58955 distinct *well_numbers*
- From wells, we query 101730 distinct *well_numbers*

5. ETL PROCESS BOTTLENECKS

In the data_preparation transformation, we will face a bottleneck in PDI. First in *Geo_D* due to a floating point comparison of latitude and longitude. Second, since the comparison is for a unique tuple with two fields instead of one single field makes this operation slower.

6. ETL PROCESS VALIDATION

Entity	Source Count from dataset	Target Table Count	Result
Wells_D	• Count of distinct API numbers in wells_mexico_all.json: **101730** • Count of API numbers in production_mexico_all.json that doesn't exist in wells dataset: **8** **Total: 101738**	**101738**	***Matching***
Operator_D	Count of distinct *operator_name* in wells_mexico_all.json: **1391**	**1391**	***Matching***
Parent_D	Count of distinct *parent* in wells_mexico_all.json: **24**	**24**	***Matching***
Geo_D	Count of distinct *latitude & longitude tuple* in wells_mexico_all.json: **96568**	**96568**	***Matching***
State_D	Count of distinct first 2 digits in API Numbers in Well_D: **2**	**2**	***Matching***
County_D	Count of distinct 3rd to 5th digits in API Numbers in Well_D: **31**	**31**	***Matching***
Date_D	Count of distinct *month* in wells_production_all.json: **22**	**22**	***Matching***
Production_F	Count of records in wells_production_all.json: **1195489**	**1195489**	***Matching***

Finalization Checklist

Name of the Project: ETL Wells_Production dataset MBD O-1-1
Date of the Review: Week of 25th of November
Name of the Reviewer: Josep Curto

Item	Response	Comments
Did you use Annotate Stream to improve your model results?	NO	
Did you use filters on your tables or charts?	YES	

Source: Data Explorer in Pentaho Data Integration (PDI) [PDF] Hitachi Retrieved from https://support.pentaho.com/hc/en-us/article_attachments/360003978672/Best_Practices_for_Data_Explorer.pdf