

# RAPPORT : PROJET PENTAHO

## MODULE : BUSINESS INTELLIGENCE

Réalisé par **YASSER EL BACHIRI**

**Master IDDL**

### OBJECTIF :

Une société de construction confie de déployer un entrepôt de données pour avoir la capacité de suivre ses travaux.

- ➔ L'objectif est de créer un système décisionnel pour analyser quelques mesures pertinentes, par exemple :
  - Le chiffre d'affaire par projet, région et type de région
  - La superficie par projet, région et semestre.
- ➔ Automatiser l'insertion de flux des données (le Job) de façon périodique

### PLAN :

- Proposer un schéma d'entrepôt de données selon les besoins précités.
- Déployer la solution sous la suite PENTAHO BI :
  - ETL PDI: Pentaho Data Integration
  - Analyse OLAP: PSW (Pentaho Schema Workbench)
  - Pentaho Server : pour la création des rapports.
- Automatiser le système avec la technologie Jenkins.

# I. SCHEMA D'ENTREPOT DE DONNEES

Pour déterminer notre table des faits et les tableaux de dimension il faut parcourir l'**algorithme de conception d'Entrepôt de Données**. Il faut d'abord choisir le sujet, dans notre cas c'est la gestion des projets. Ensuite, on détermine la granularité comme par exemple : Analyser le chiffre d'affaire, le cout, la superficie par temps, région, projet et opération. Alors nos tableaux de dimension seront par conséquence : La dimension du **temps, région, projet et opération**.

Pour chaque dimension on détermine le niveau, la hiérarchie et les attributs. Et enfin on précise les mesures de notre table des faits : **Chiffre d'affaire, superficie, cout d'opération**. Notre schéma d'entrepôt de données est le suivant :

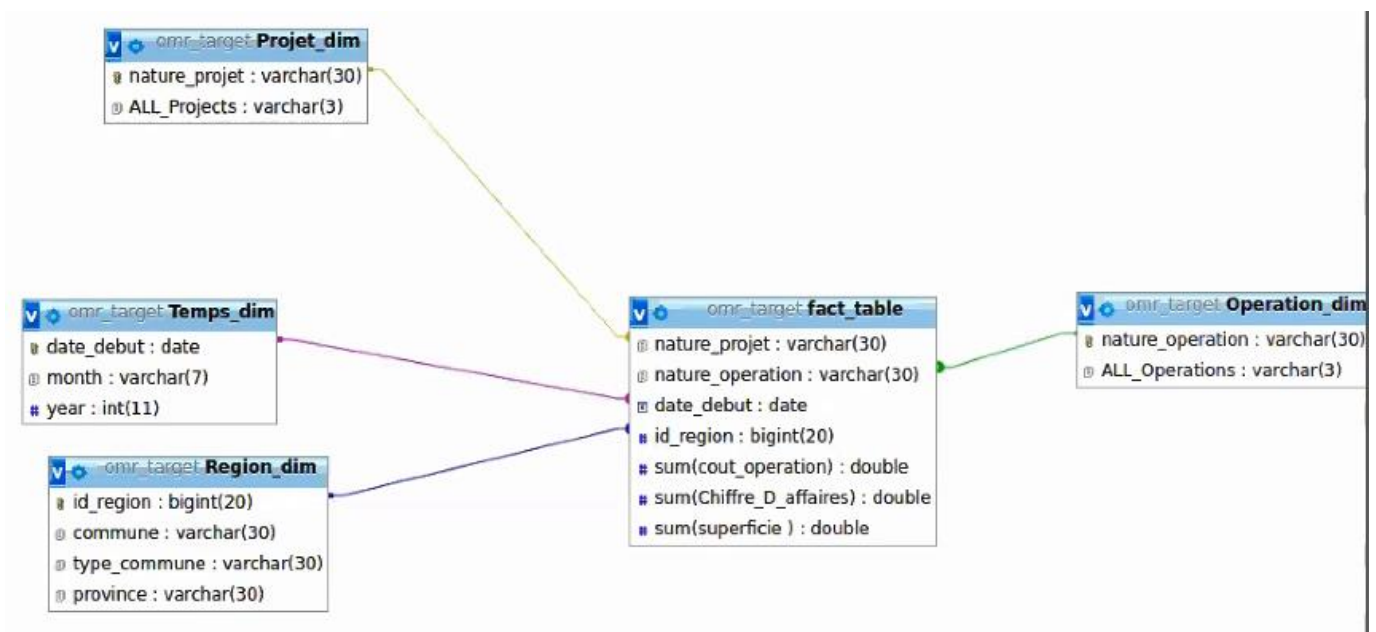


Schéma d'entrepôt de données du projet

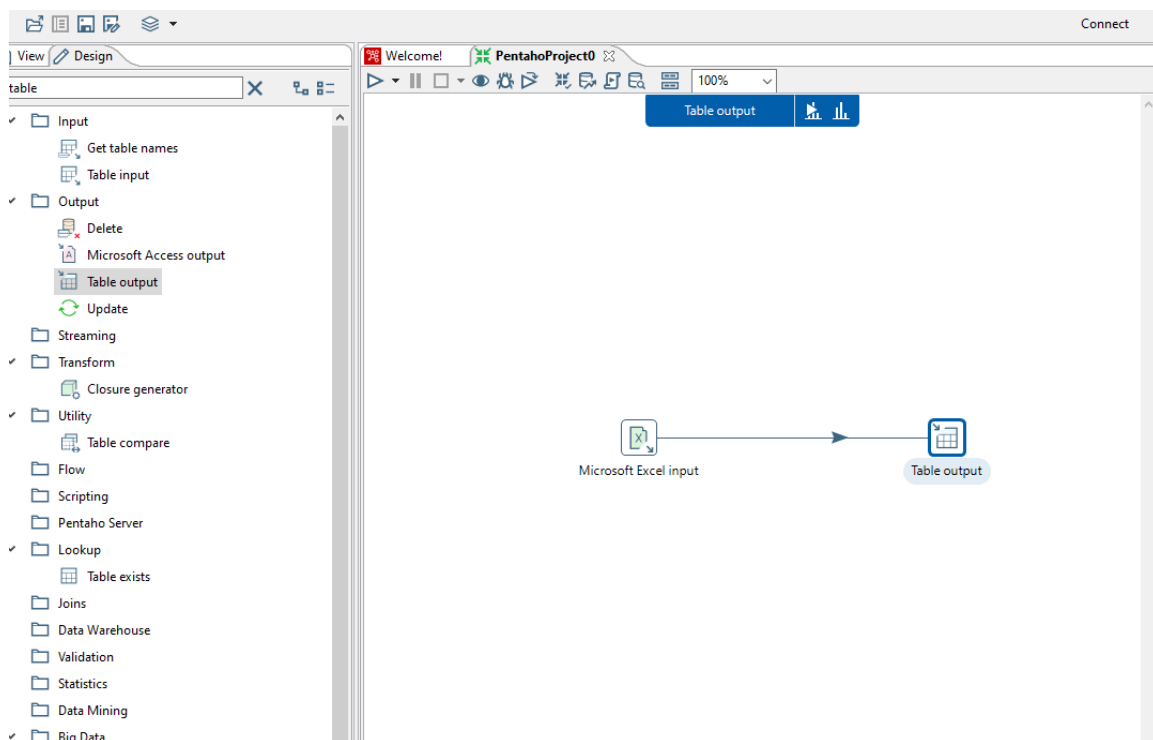
## II. DEPLOYER LA SOLUTION SOUS LA SUITE PENTAHO BI :

### 1. ETL PDI: PENTAHO DATA INTEGRATION

Pour transformer nos données depuis le fichier CSV aux structures d'entrepôt de données comme illustrer avant, on va suivre les étapes suivantes :

- On va charger tous nos données depuis notre fichier CSV vers une base de données intermédiaire pour bien valider les types et normaliser les données.
- Charger les données depuis notre table intermédiaire vers la base de données destinataire qui garde et respecte la structure de notre entrepôt de données.

#### 1. 1- CHARGER LES DONNEES DEPUIS UN FICHIER CSV VERS UNE BASE DE DONNEES INTERMEDIAIRE



*Schéma de flux de données depuis le fichier CSV vers la base de données intermédiaire*

Step name: **Microsoft Excel input**

Add sheet(s)

Files | Sheets | Content | Error Handling | Fields | Additional output fields

Spread sheet type (engine): Excel 97-2003 XLS (JXL)

File or directory:  Add Browse...

Regular Expression:

Exclude Regular Expression:

Password:

Selected files:

#	File/Directory	
1	E:\Yasser\Study\Master 2019-2020\7- Systemes Information Decisionnels - BI\Seance_07_04_2020\input_Qu	Delete

Accept filenames from previous steps

Accept filenames from previous step ☐

Step to read filenames from:

Field in the input to use as filename:

Show filename(s)...

OK Previous steps Cancel

Choisir l'emplacement du fichier CSV

Step name: **Table output**

Connection: **omraneDB\_src** Edit... New... Wizard...

Target schema:  Browse...

Target table:  Browse...

Commit size: 1000

Truncate table ☐

Ignore insert errors ☐

Specify database fields ☐

Main options | Database fields

Partition data over tables ☐

Partitioning field:

Partition data per month ☒

Partition data per day ☐

Use batch update for inserts ☒

Is the name of the table defined in a field? ☐

Field that contains name of table:

Store the tablename field ☒

Return auto-generated key ☐

Name of auto-generated key field:

Help OK Cancel SQL

1- La fenêtre des paramètres du tableau intermédiaire

Connection name: **omraneDB\_src**

Connection type: **MySQL**

Settings:

Host Name: **localhost**

Database Name: **omraneDB\_src**

Port Number: **3306**

Username: **root**

Password: **\*\*\*\*\***

☒ Use Result Streaming Cursor

Test Feature List Explore

OK Cancel

2- Vérifier la connexion avec la base de données nommée «omraneDB\_src »

Connection tested successfully



Connection to omraneDB\_src was successful.

Hostname: localhost  
Port: 3306  
Database name: omraneDB\_src

OK

Le test de la connexion avec la base de données a été bien établie

**Table output**

Step name: Table output

Connection: omraneDB\_src [Edit... New... Wizard...]

Target schema: [Browse...]

Target table: Data\_src [Browse...]

Commit size: 1000

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options / Database fields

Fields to insert:

#	Table field	Stream field
1	Id	Id
2	nature_projet	nature_projet
3	nature_operation	nature_operation
4	province	province
5	commune	commune
6	type_commune	type_commune
7	cout operation	cout operation
8	Chiffre_D_affaires	Chiffre_D_affaires
9	superficie	superficie
10	date_debut	date_debut

[Get fields] [Enter field mapping]

[Help] [OK] [Cancel] [SQL]

**Simple SQL editor**

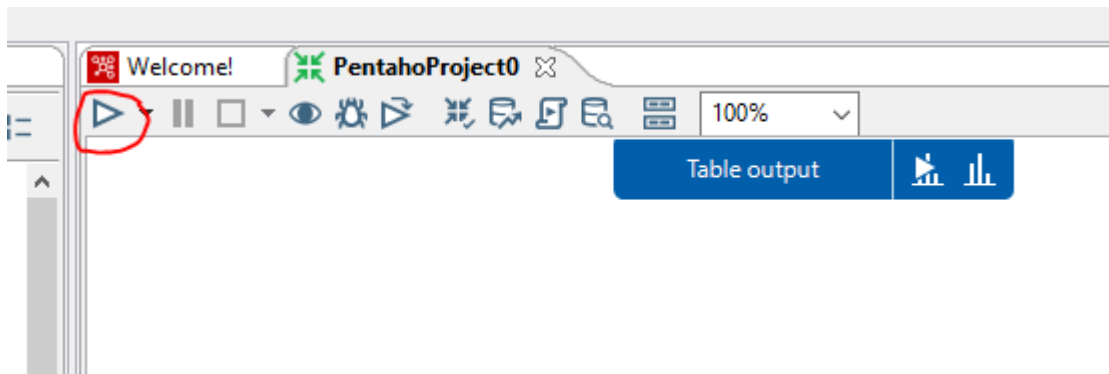
SQL statements, separated by semicolon ';'

```
CREATE TABLE Data_src
(
  Id DOUBLE
, nature_projet VARCHAR(40)
, nature_operation VARCHAR(40)
, province VARCHAR(40)
, commune VARCHAR(40)
, type_commune VARCHAR(40)
, `cout operation` DOUBLE
, Chiffre_D_affaires DOUBLE
, superficie DOUBLE
, date_debut DATE
)
;
```

Line 8 column 26

[Execute] [Clear cache] [Close]

En cliquant sur « Get Fields » sur l'image en dessus à gauche on récupère tous les champs du fichiers CSV comme des attributs dans notre nouvelle table intermédiaire. On peut voir la requête SQL (l'image à droite) qui va être exécuter.



On lance l'opération sur le bouton « Run » après avoir vérifié tous les paramètres.



**Execution Results**

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

2020/05/09 05:08:04 - Spoon - Running transformation using the Kettle execution engine

2020/05/09 05:08:04 - Spoon - Transformation opened.

2020/05/09 05:08:04 - Spoon - Launching transformation [PentahoProject0]...

2020/05/09 05:08:04 - Spoon - Started the transformation execution.

2020/05/09 05:08:04 - PentahoProject0 - Dispatching started for transformation [PentahoProject0]

2020/05/09 05:08:04 - Table output.0 - Connected to database [omraneDB\_src] (commit=1000)

2020/05/09 05:08:06 - Microsoft Excel input.0 - Finished processing (I=2991, O=0, R=0, W=2991, U=0, E=0)

2020/05/09 05:08:07 - Table output.0 - Finished processing (I=0, O=2991, R=2991, W=2991, U=0, E=0)

2020/05/09 05:08:07 - Spoon - The transformation has finished!!

9

10 • `select * from Data_src;`

	Id	nature_projet	nature_operation	province	commune	type_commune	cout_operation	Chiffre_D_affaires	superficie	date
1	1	Construction	Logements	TIZNIT	TIZNIT	Urbaine	45.46747294	10.204	0.4561	1998
2	2	Amenagement Foncier	Lotissement Habitat	TIZNIT	TARSOUAT	Rurale	0.70397788	1.127525	0.489475	1998
3	3	Construction	Logements	TIZNIT	TIZNIT	Urbaine	1.19616497	9.962399	0.442004	2000
4	4	Amenagement Foncier	Lotissement Habitat	TIZNIT	REGGADA	Rurale	9.570037	12.0169	2.1237	2000
5	5	Construction	Logements	TIZNIT	SIDI IFNI	Urbaine	3.5792041	0.57	0.74131	2000
6	6	Amenagement Foncier	Lotissement Habitat	OUARZAZATE	OURZAZATE	Urbaine	136.38171985	152.71615	20.03	2000
7	7	Amenagement Foncier	Lotissement Habitat	OUARZAZATE	TINGHIR	Urbaine	133.61247475	92.31805	102.376	2000
8	8	Amenagement Foncier	Lotissement Habitat	GUELMIM	BOUIZAKARNE	Urbaine	45.97500729	92.31805	22.1439	2000
9	9	Amenagement Foncier	Lotissement Habitat	GUELMIM	BOUIZAKARNE	Urbaine	0.222439	0	23.9634	2000
10	10	Amenagement Foncier	Lotissement Habitat	AGADIR	AGADIR	Urbaine	600.53554664	7485	19.819483	2000
11	11	Amenagement Foncier	Lotissement Habitat	AGADIR	AGADIR	Urbaine	0.3636	0	71	2000
12	12	Construction	Logements	AGADIR	AGADIR	Urbaine	38.1520227	41.85	0.6663	2000
13	13	Construction	Logements	AGADIR	AGADIR	Urbaine	15.16749775	18.4372035	0.0532	2000
14	14	Construction	Logements	AGADIR	AGADIR	Urbaine	18.250993	34.144	0.216	2000
15	15	Amenagement Foncier	Lotissement Habitat	AGADIR	AGADIR	Urbaine	42.57135453	2.0847	5.3784	2000
16	16	Construction	Logements	AGADIR	AGADIR	Urbaine	33.61154182	56.9628	0.429	2000
17	17	Construction	Logements	AGADIR	AGADIR	Urbaine	86.11510951	51.1544	2.2097	2000
18	18	Amenagement Foncier	Lotissement Habitat	AGADIR	AGADIR	Urbaine	37895	37895	0.2962	1998
19	19	Construction	Logements	TIZNIT	TIZNIT	Urbaine	2.48421633	37895	0.7927	1998
20	20	Amenagement Foncier	Lotissement Habitat	TIZNIT	TIZNIT	Urbaine	0.00000000	1.000000	1.000000	1998

Data\_src 2 x Read Only

Output

*Les enregistrements de la table intermédiaire après l'exécution de l'opération avec succès*

## 1. 2- CHARGER LES DONNEES DEPUIS LA TABLE INTERMEDIAIRE VERS LA BASE DE DONNEES DESTINATAIRE

Pour transformer les données vers la base de données destinataire « **omraneDB\_target** », on va utiliser la base de données intermédiaire « **omraneDB\_src** » qui contient la table « **Data\_src** ». On a quatre dimensions, donc on va créer quatre tables avec la table des faits dans la base de données « **omraneDB\_target** ».

Pour les tables de la dimension « **Projet** » et « **Opération** » on a utilisé **la conception des dimensions à évolution lente**. Le cas où le client a changé une information concernant ces deux dimensions, la table va garder l'ancien valeur et ajoute un nouvel enregistrement avec des clés de substitution. Avec ce concept, on peut suivre l'évolution des attributs et aussi segmenter la table de faits en fonction de l'historique.

The screenshot displays the Pentaho Data Integration (PDI) interface. The top pane shows a data flow diagram with four steps, each represented by a green checkmark icon:

- Table input data\_src to Temps → Table output temps\_dim
- Table input data\_src to Projet → Table output projet\_dim
- Table input data\_src to Region → Table output region\_dim
- Table input data\_src to Operation → Table output operation\_dim

The bottom pane shows the "Execution Results" tab, which includes a logging window with the following text:

```

2020/06/01 17:17:27 - Table input data_src to Region.0 - Finished processing (I=2991, O=0, R=0, W=2991, U=0, E=0)
2020/06/01 17:17:27 - Table input data_src to Temps.0 - Finished reading query, closing connection.
2020/06/01 17:17:27 - Table input data_src to Temps.0 - Finished processing (I=1306, O=0, R=0, W=1306, U=0, E=0)
2020/06/01 17:17:27 - Table input data_src to Operation.0 - Finished reading query, closing connection.
2020/06/01 17:17:27 - Table input data_src to Operation.0 - Finished processing (I=2991, O=0, R=0, W=2991, U=0, E=0)
2020/06/01 17:17:27 - Table input data_src to Projet.0 - Finished processing (I=2991, O=0, R=0, W=2991, U=0, E=0)
2020/06/01 17:17:27 - Table output temps_dim.0 - Finished processing (I=0, O=1306, R=1306, W=1306, U=0, E=0)
2020/06/01 17:17:28 - Table output region_dim.0 - Finished processing (I=0, O=2991, R=2991, W=2991, U=0, E=0)
2020/06/01 17:17:30 - Table output projet_dim.0 - Finished processing (I=2991, O=2991, R=2991, W=2991, U=0, E=0)
2020/06/01 17:17:30 - Table output operation_dim.0 - Finished processing (I=2991, O=2991, R=2991, W=2991, U=0, E=0)
2020/06/01 17:17:30 - Spoon - The transformation has finished!!
  
```

L'installation des quatre flux pour chaque dimension depuis la base de données « **omraneDB\_src** » vers la base de données « **omraneDB\_target** ». Les tables « **projet\_dim** » et « **operation\_dim** » utilise la conception des dimensions à variations lente.

The screenshot shows the "Table input" configuration dialog. The "Step name" is "Table input" and the "Connection" is "omraneDB\_src". The "SQL" field contains the following query:

```

select distinct(date_debut),
date_format(date_debut, '%Y-%m') as Month,
year(date_debut) as Year FROM data_src;
  
```

La requête SQL depuis la table input de la dimension Temps

Step name: Table output

Connection: omraneDB\_target

Target schema:

Target table: temps\_dim

Commit size: 1000

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	date_debut	date_debut
2	Month	Month
3	Year	Year

Les attributs récupérés pour la nouvelle table  
« temps\_dim »

Simple SQL editor

SQL statements, separated by semicolon ';'

```
CREATE TABLE temps_dim
(
  date_debut DATE PRIMARY KEY,
  Month VARCHAR(7)
, Year INT
)
```

La requête SQL qui va être exécuté

Step name: Table input data\_src to Projet

Connection: omraneDB\_src

SQL

```
select Id, nature_projet, date_debut, 'all' as all_project
from data_src order by date_debut asc
```

La requête SQL depuis la table input de la dimension Projet

Step name: Table output projet\_dim

Update the dimension?: ☒

Connection: omraneDB\_target

Target schema:

Target table: projet\_dim

Commit size: 3000

Enable the cache?: ☐

Pre-load the cache?: ☐

Cache size in rows (0 = cache all):

Keys Fields

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	id	Id

Technical key field: tkey\_pro

Creation of technical key:

☒ Use table maximum + 1

☐ Use sequence

☐ Use auto increment field

Version field: version

Stream Datefield: date\_debut

Date range start field: date\_from Min. year: 1900

Use an alternative start date?: ☐ <Select Option>

Table date range end: date\_to Max. year: 2199

OK Cancel Get Fields SQL

Simple SQL editor

SQL statements, separated by semicolon ';'

```
CREATE TABLE projet_dim
(
  tkey_pro INT NOT NULL PRIMARY KEY
, version INT
, date_from DATETIME
, date_to DATETIME
, id DOUBLE
, nature_projet VARCHAR(40)
, date_debut DATE
, all_project VARCHAR(3)
)
CREATE INDEX idx_projet_dim_lookup ON projet_dim(id)
CREATE INDEX idx_projet_dim_tk ON projet_dim(tkey_pro)
```

La table dimension à variation lente « projet\_dim » contient la clé de substitution « tkey\_pro », la version qui représente l'indicateur du changement de l'enregistrement, « date\_from » et « date\_to » représente la durée entre l'insertion de l'enregistrement à la base de données et l'insertion de sa nouvelle version, « id, nature\_projet, date\_debut, all\_project » les attributs extraite depuis la base de donnée « omraneDB\_src ».



**Table input**

Step name: Table input\_data\_src to Operation

Connection: omraneDB\_src

SQL:

```
select Id, nature_operation, date_debut, 'all' as all_operation
from data_src order by date_debut asc
```

Get SQL select statement...

La requête SQL depuis la table input de **la dimension Opération**

**Dimension lookup/update**

Step name: Table output operation\_dim

Update the dimension? ☒

Connection: omraneDB\_target

Target schema:

Target table: operation\_dim

Commit size: 3000

Enable the cache? ☒

Pre-load the cache? ☐

Cache size in rows (0 = cache all): 5000

Keys Fields

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	id	Id

Technical key field: tkey\_opr

Creation of technical key:

☒ Use table maximum + 1

☐ Use sequence

☐ Use auto increment field

Version field: version

Stream Datefield: date\_debut

Date range start field: date\_from Min. year: 1900

Use an alternative start date? ☐ <Select Option> date\_debut

Table date range end: date\_to Max. year: 2199

OK Cancel Get Fields SQL

Simple SQL editor

SQL statements, separated by semicolon ;

```
CREATE TABLE operation_dim
(
  tkey_opr INT NOT NULL PRIMARY KEY
  , version INT
  , date_from DATETIME
  , date_to DATETIME
  , id DOUBLE
  , nature_operation VARCHAR(40)
  , date_debut DATE
  , all_operation VARCHAR(3)
)
CREATE INDEX idx_operation_dim_lookup ON operation_dim(id)
CREATE INDEX idx_operation_dim_tk ON operation_dim(tkey_opr)
```

De même pour la table « **operation\_dim** », contient la clé de substitution « **tkey\_opr** », la **version** qui représente l'indicateur du changement de l'enregistrement, « **date\_from** » et « **date\_to** » représente la durée entre l'insertion de l'enregistrement à la base de données et l'insertion de sa nouvelle version, « **id, nature\_ooperation, date\_debut, all\_ooperation** » les attributs extraite depuis la base de donnée « **omraneDB\_src** ».

The screenshot displays the Pentaho Data Integration (Kettle) interface. On the left, a tree view shows the 'Transform' folder expanded, containing 'Closure generator', 'Table compare', 'Flow', 'Scripting', 'Pentaho Server', 'Lookup', 'Table exists', 'Joins', 'Data Warehouse', 'Validation', 'Statistics', 'Data Mining', and 'Big Data'. The 'Big Data' folder is expanded, showing 'Cassandra input', 'Cassandra output', 'HBase input', 'HBase output', 'SSTable output', 'Agile', 'Table Agile Mart', and 'Cryptography'. The main workspace shows a transformation job with four steps: 'Table input operation dimension', 'Table output operation dimension', 'Table input region dimension', and 'Table output region dimension'. Below the workspace, the 'Execution Results' tab is active, showing a log of the transformation process. The log indicates that the transformation was successful, with all four dimensions being processed and outputted to the database.

**Execution Results**

Logging Execution History Step Metrics Performance Graph Metrics Preview data

2020/05/09 06:04:42 - Spoon - Running transformation using the Kettle execution engine  
 2020/05/09 06:04:42 - Spoon - Transformation opened.  
 2020/05/09 06:04:42 - Spoon - Launching transformation [All\_dim]...  
 2020/05/09 06:04:42 - Spoon - Started the transformation execution.  
 2020/05/09 06:04:42 - All\_dim - Dispatching started for transformation [All\_dim]  
 2020/05/09 06:04:42 - Table output temps dimension.0 - Connected to database [omraneDB\_target] (commit=1000)  
 2020/05/09 06:04:42 - Table output projet dimension.0 - Connected to database [omraneDB\_target] (commit=1000)  
 2020/05/09 06:04:42 - Table output operation dimension.0 - Connected to database [omraneDB\_target] (commit=1000)  
 2020/05/09 06:04:42 - Table output region dimension.0 - Connected to database [omraneDB\_target] (commit=1000)  
 2020/05/09 06:04:42 - Table input operation dimension.0 - Finished reading query, closing connection.  
 2020/05/09 06:04:42 - Table input operation dimension.0 - Finished processing (I=11, O=0, R=0, W=11, U=0, E=0)  
 2020/05/09 06:04:42 - Table output projet dimension.0 - Finished reading query, closing connection.  
 2020/05/09 06:04:42 - Table input projet dimension.0 - Finished processing (I=3, O=0, R=0, W=3, U=0, E=0)  
 2020/05/09 06:04:42 - Table input region dimension.0 - Finished reading query, closing connection.  
 2020/05/09 06:04:42 - Table input region dimension.0 - Finished processing (I=380, O=0, R=0, W=380, U=0, E=0)  
 2020/05/09 06:04:42 - Table input temps dimension.0 - Finished reading query, closing connection.  
 2020/05/09 06:04:42 - Table input temps dimension.0 - Finished processing (I=1306, O=0, R=0, W=1306, U=0, E=0)  
 2020/05/09 06:04:42 - Table output projet dimension.0 - Finished processing (I=0, O=3, R=3, W=3, U=0, E=0)  
 2020/05/09 06:04:42 - Table output operation dimension.0 - Finished processing (I=0, O=11, R=11, W=11, U=0, E=0)  
 2020/05/09 06:04:42 - Table output region dimension.0 - Finished processing (I=0, O=380, R=380, W=380, U=0, E=0)  
 2020/05/09 06:04:42 - Table output temps dimension.0 - Finished processing (I=0, O=1306, R=1306, W=1306, U=0, E=0)  
 2020/05/09 06:04:42 - Spoon - The transformation has finished!!

*L'exécution de la transformation a été réussie sur les quatre dimensions*

On peut maintenant diriger vers nos bases de données sur « **MySQL Workbench** » pour vérifier est-ce que la transformation a été bien installé.

Result Grid

Filter Rows:

	date_debut	Month	Year
▶	1901-01-01	1901-01	1901
	1973-09-15	1973-09	1973
	1974-01-01	1974-01	1974
	1974-04-19	1974-04	1974
	1974-08-15	1974-08	1974
	1975-02-28	1975-02	1975
	1975-04-05	1975-04	1975
	1975-05-01	1975-05	1975
	1975-05-10	1975-05	1975
	1975-07-17	1975-07	1975
	1976-02-12	1976-02	1976
	1976-03-31	1976-03	1976
	1977-03-06	1977-03	1977
	1977-03-30	1977-03	1977
	1977-12-31	1977-12	1977
	1978-03-31	1978-03	1978

temps\_dim 1

	id_region	commune	type_commune	province	country
▶	1	TIZNIT	Urbaine	TIZNIT	Maroc
	2	TARSOUAT	Rurale	TIZNIT	Maroc
	3	TIZNIT	Urbaine	TIZNIT	Maroc
	4	REGGADA	Rurale	TIZNIT	Maroc
	5	SIDI IFNI	Urbaine	TIZNIT	Maroc
	6	OURZAZATE	Urbaine	OUARZAZATE	Maroc
	7	TINGHIR	Urbaine	OUARZAZATE	Maroc
	8	BOUIZAKARNE	Urbaine	GUELMIM	Maroc
	9	BOUIZAKARNE	Urbaine	GUELMIM	Maroc
	10	AGADIR	Urbaine	AGADIR	Maroc
	11	AGADIR	Urbaine	AGADIR	Maroc
	12	AGADIR	Urbaine	AGADIR	Maroc
	13	AGADIR	Urbaine	AGADIR	Maroc
	14	AGADIR	Urbaine	AGADIR	Maroc
	15	AGADIR	Urbaine	AGADIR	Maroc
	16	AGADIR	Urbaine	AGADIR	Maroc
	17	AGADIR	Urbaine	AGADIR	Maroc
	18	AGADIR	Urbaine	AGADIR	Maroc
	19	TIZNIT	Urbaine	TIZNIT	Maroc

region\_dim 19

*Les enregistrements des tables « temps\_dim » et « region\_dim » après l'exécution de la transformation*

	tkey_pro	version	date_from	date_to	id	nature_projet	date_debut	all_project
▶	0	1	NULL	NULL	NULL	NULL	NULL	NULL
	1	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1869	Amenagement Foncier	1901-01-01	all
	2	1	1900-01-01 00:00:00	2200-01-01 00:00:00	2521	Amenagement Foncier	1901-01-01	all
	3	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1461	Amenagement Foncier	1973-09-15	all
	4	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1471	Amenagement Foncier	1974-01-01	all
	5	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1462	Construction	1974-04-19	all
	6	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1888	Construction	1974-08-15	all
	7	1	1900-01-01 00:00:00	2200-01-01 00:00:00	125	Construction	1975-02-28	all
	8	1	1900-01-01 00:00:00	2200-01-01 00:00:00	258	Construction	1975-04-05	all
	9	1	1900-01-01 00:00:00	2200-01-01 00:00:00	279	Construction	1975-05-01	all
	10	1	1900-01-01 00:00:00	2200-01-01 00:00:00	239	Construction	1975-05-10	all
	11	1	1900-01-01 00:00:00	2200-01-01 00:00:00	2192	Construction	1975-07-17	all
	12	1	1900-01-01 00:00:00	2200-01-01 00:00:00	958	Amenagement Foncier	1976-02-12	all
	13	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1450	Amenagement Foncier	1976-02-21	all

projet\_dim 20 x

Les enregistrements de la table « **projet\_dim** » après l'exécution de la transformation

	tkey_opr	version	date_from	date_to	id	nature_operation	date_debut	all_operation
▶	0	1	NULL	NULL	NULL	NULL	NULL	NULL
	1	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1869	Lotissement Habitat	1901-01-01	all
	2	1	1900-01-01 00:00:00	2200-01-01 00:00:00	2521	Lotissement Habitat	1901-01-01	all
	3	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1461	Lotissement Habitat	1973-09-15	all
	4	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1471	Lotissement Habitat	1974-01-01	all
	5	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1462	Logements	1974-04-19	all
	6	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1888	Logements	1974-08-15	all
	7	1	1900-01-01 00:00:00	2200-01-01 00:00:00	125	Logements	1975-02-28	all
	8	1	1900-01-01 00:00:00	2200-01-01 00:00:00	258	Logements	1975-04-05	all
	9	1	1900-01-01 00:00:00	2200-01-01 00:00:00	279	Logements	1975-05-01	all
	10	1	1900-01-01 00:00:00	2200-01-01 00:00:00	239	Logements	1975-05-10	all
	11	1	1900-01-01 00:00:00	2200-01-01 00:00:00	2192	Logements	1975-07-17	all
	12	1	1900-01-01 00:00:00	2200-01-01 00:00:00	958	Lotissement Habitat	1976-02-12	all
	13	1	1900-01-01 00:00:00	2200-01-01 00:00:00	1450	Lotissement Habitat	1976-02-21	all

operation\_dim 21 x

Les enregistrements de la table « **operation\_dim** » après l'exécution de la transformation

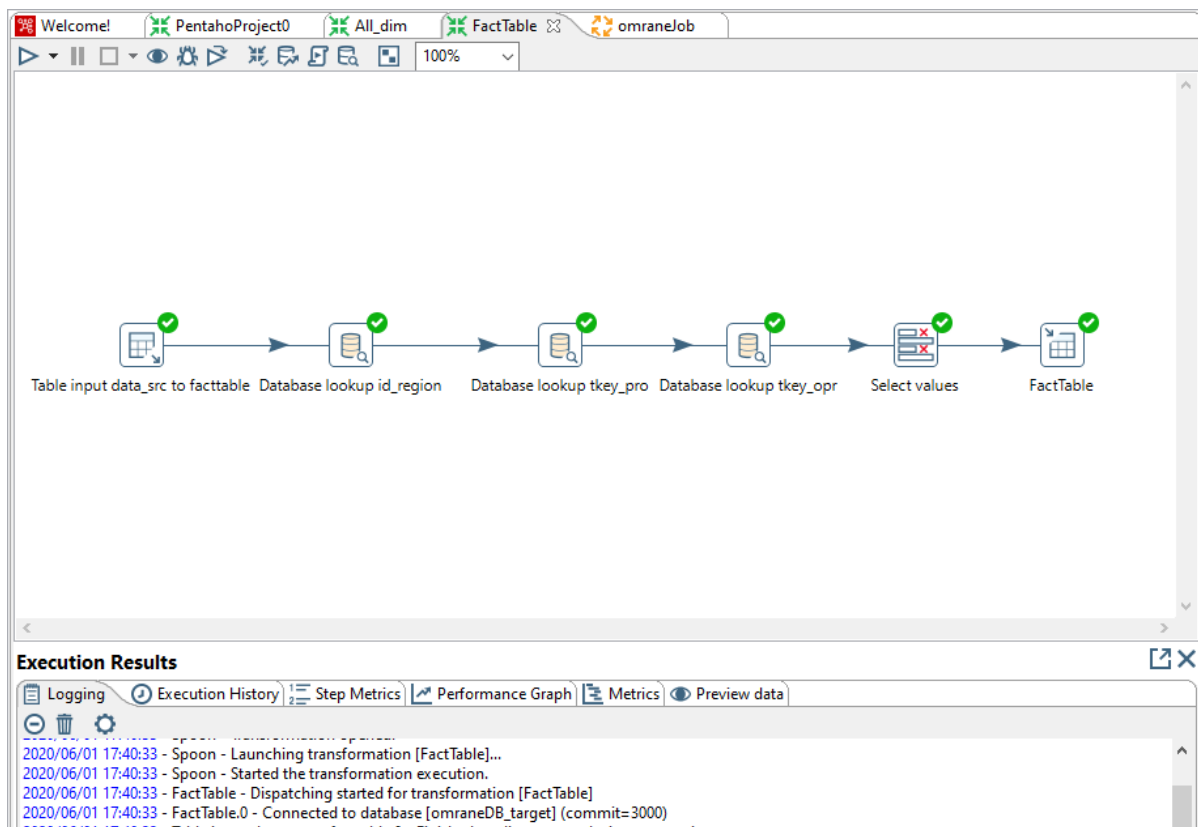
## • Création de la table des faits

Maintenant, après avoir créé les quatre dimensions, il faut créer la **table des faits** qui contient comme attribut les clés primaires de chaque table dimension et comme condition la table des faits contient que **des attributs numériques**, dans notre cas on va utiliser **les id des dimensions** et **les clés de substitution** des table « projet\_dim » et « opeartion\_dim ». Pour les mesures de notre table des faits on ajoute le calcul des attributs « **sum(cout\_operation)** », « **sum(Chiffre\_D\_affaires)** » et « **sum(superficie)** ».

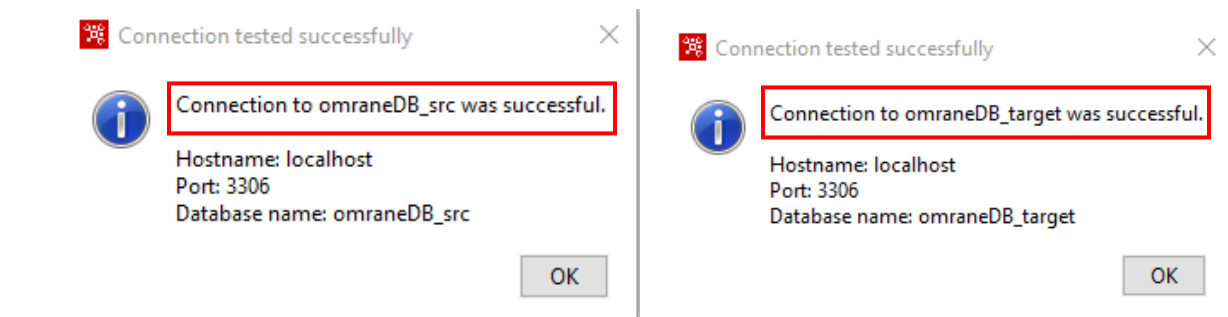
Pour atteindre à cet objectif, on va utiliser notre base de données source « **omraneDB\_src** » comme input, et aussi un outil appelé « **Database Lookup** » qui va nous aider à faire des tests et il joue comme un rôle d'une jointure entre deux tables pour éliminer les occurrences, dans notre cas on va ajouter trois tests :

- L'égalité entre deux valeurs de l'attributs « **commune** » de la table « Data\_src » et la table « FactTable », va récupérer la valeur de "**id\_region**" de cet enregistrement pour la table « FactTable ».
- L'égalité entre **les id** de la table « Data\_src » et la table « FactTable », va récupérer la valeur de clés de substitution « **tkey\_pro** » de cet enregistrement pour la table « FactTable ».
- L'égalité entre **les id** de la table « Data\_src » et la table « FactTable », va récupérer la valeur de clés de substitution « **tkey\_opr** » de cet enregistrement pour la table « FactTable ».

On va utiliser aussi un outil appelé « **Select Values** » qui va nous aider à sélectionner les attributs passant par le flux et éliminer l'attributs désirer, dans notre cas on va supprimer les attributs « **nature\_projet** », « **nature\_operation** » et « **commune** ».



*L'installation de flux depuis la table source de la base de données « **omraneDB\_src** » vers la table destinataire « **FactTable** » la base de données « **omraneDB\_target** » en passant par les trois outils « **Database Lookup** » et l'outil « **Select Values** »*



*Vérification de la connexion sur chaque base de données*

## Table input

Step name: Table input data\_src to facttable

Connection: omraneDB\_src

SQL

```
select Id, nature_projet, nature_operation, date_debut, commune,
sum(cout_operation) as cout_operation, sum(Chiffre_D_affaires) as Chiffre_D_affaires,
sum(superficie) as superficie
FROM data_src group by Id, nature_projet, nature_operation,
commune, date_debut;
```

La requête SQL depuis la table source input de **la base de données « omraneDB\_src »**

## Table output

Step name: FactTable

Connection: omraneDB\_target

Target schema: omraneDB\_target

Target table: facttable

Commit size: 3000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☒

### Main options Database fields

#### Fields to insert:

#	Table field	Stream field
1	Id	Id
2	date_debut	date_debut
3	cout_operation	cout_operation
4	Chiffre_D_affaires	Chiffre_D_affaires
5	superficie	superficie
6	id_region	id_region
7	tkey_pro	tkey_pro
8	tkey_opr	tkey_opr
9		

#### Simple SQL editor

SQL statements, separated by semicolon ';'.

```
CREATE TABLE omraneDB_target.facttable
(
  Id INT
, date_debut DATE
, cout_operation DOUBLE
, Chiffre_D_affaires DOUBLE
, superficie DOUBLE
, id_region INT
, tkey_pro INT
, tkey_opr INT
)
```

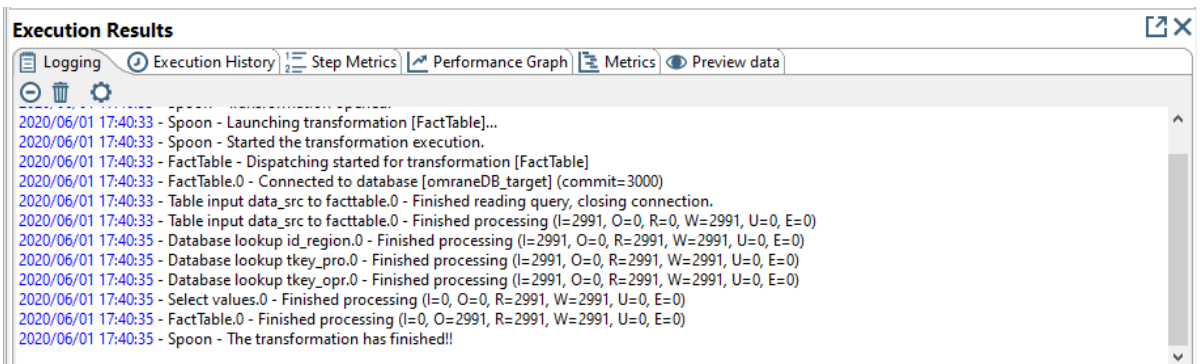
Help

OK

Cancel

SQL

Les attributs récupérés pour la nouvelle table « **FactTable** » et La requête SQL qui va être exécuté



L'exécution de la transformation a été réussie sur la nouvelle table des faits « **FactTable** »

Id	date_debut	cout_operation	Chiffre_D_affaires	superficie	id_region	tkey_pro	tkey_opr
1	1998-02-21	45.46747294	10.204	0.4561	1	788	788
2	1999-11-24	0.70397788	1.127525	0.489475	2	847	847
3	2002-05-01	1.19616497	9.962399	0.442004	1	971	971
4	2006-01-01	9.570037	12.0169	2.1237	4	2164	2164
5	2003-02-01	3.5792041	0.57	0.74131	5	1015	1015
6	2005-10-05	136.38171985	152.71615	20.03	6	2121	2121
7	2005-12-07	133.61247475	92.31805	102.376	7	2153	2153
8	2009-02-28	45.97500729	92.31805	22.1439	8	2598	2598
9	2003-10-01	0.222439	0	23.9634	8	1061	1061
10	2006-11-04	600.53554664	7485	19.819483	10	2237	2237
11	2006-11-04	0.3636	0	71	10	2238	2238
12	2007-05-30	38.1520227	41.85	0.6663	10	2307	2307
13	2007-06-01	15.16749775	18.4372035	0.0532	10	2308	2308
14	2007-06-01	18.250003	24.144	0.215	10	2300	2300

Les enregistrements sur MySQL Workbench de table des faits « **FactTable** » après l'exécution de la transformation

## • Création du “Job”

Pour cette étape, on va créer un “**Job**” qui garantit l'exécution des trois transformations sans erreurs sinon il arrête l'exécution et affiche la partie concernée par l'erreur. Si le “Job” a terminé avec succès, on a lui donné la possibilité d'envoyer un **e-mail** à l'administrateur de base de données contenant :

- La date et le temps de l'exécution
- Un Log file contient les détails de l'exécution les erreurs
- Et bien autre information...

Welcome! PentahoProject0 All\_dim FactTable omraneJob

100%

Start Extraction depuis le CSV Dimensions Update Update FactTable Mail Success

### Execution Results

Logging History Job metrics Metrics

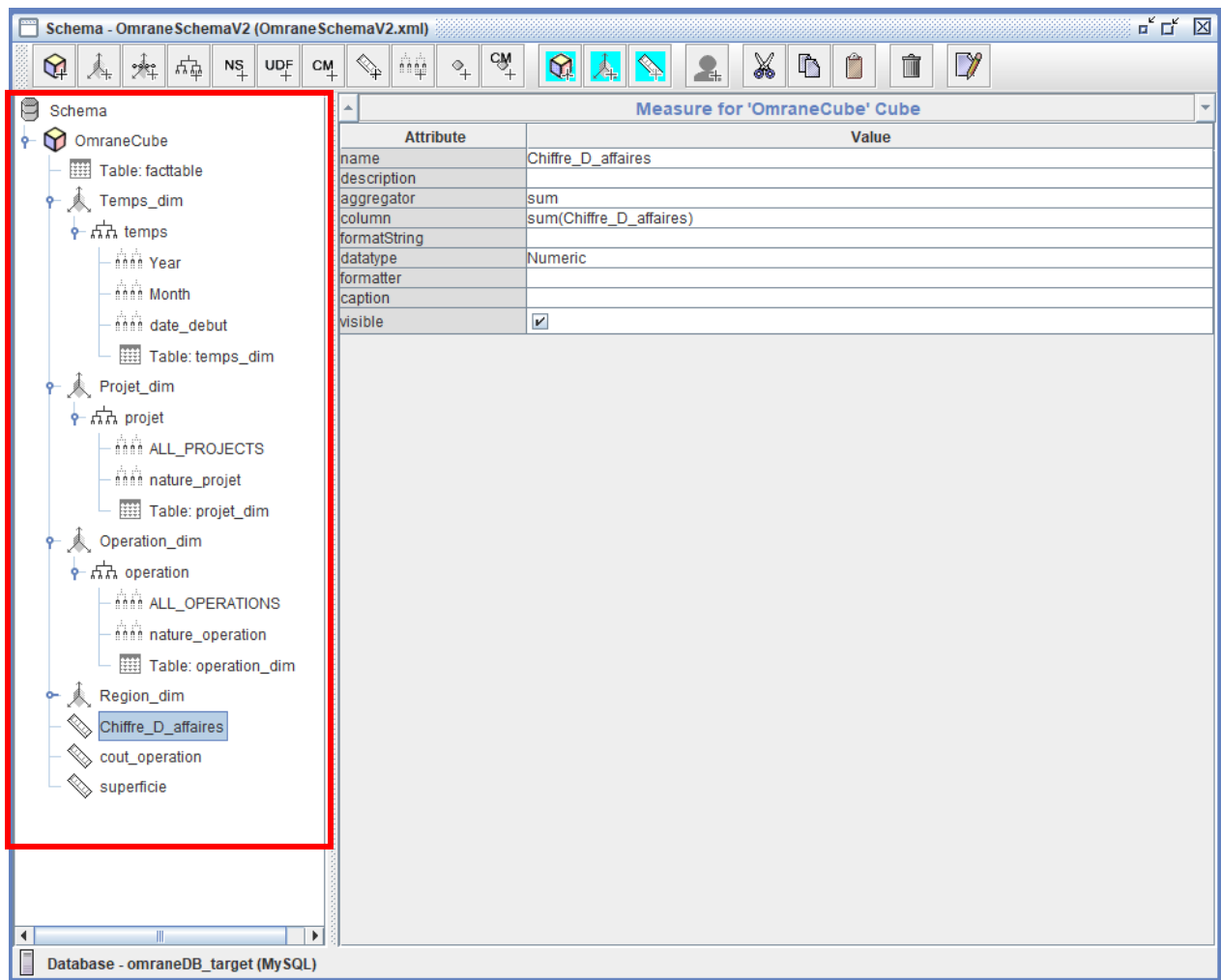
2020/06/01 18:09:27 - Database lookup tkey\_opr.0 - Finished processing (I=2991, O=0, R=2991, W=2991, U=0, E=0)  
2020/06/01 18:09:27 - Select values.0 - Finished processing (I=0, O=0, R=2991, W=2991, U=0, E=0)  
2020/06/01 18:09:27 - FactTable.0 - Finished processing (I=0, O=2991, R=2991, W=2991, U=0, E=0)  
2020/06/01 18:09:27 - omraneJob - Starting entry [Mail]  
2020/06/01 18:09:29 - omraneJob - Starting entry [Success]  
2020/06/01 18:09:29 - omraneJob - Finished job entry [Success] (result=[true])  
2020/06/01 18:09:29 - omraneJob - Finished job entry [Mail] (result=[true])  
2020/06/01 18:09:29 - omraneJob - Finished job entry [Update FactTable] (result=[true])  
2020/06/01 18:09:29 - omraneJob - Finished job entry [Dimensions Update] (result=[true])  
2020/06/01 18:09:29 - omraneJob - Finished job entry [Extraction depuis le CSV] (result=[true])  
2020/06/01 18:09:29 - omraneJob - Job execution finished  
2020/06/01 18:09:29 - Spoon - Job has ended.

Le “**Job**” exécute les trois transformations avec succès et envoie un mail à l’administrateur de base de données



## 2. ANALYSE OLAP: PSW (PENTAHO SCHEMA WORKBENCH)

Dans cette étape, l'outil "**PENTAHO SCHEMA WORKBENCH**" va nous aider à créer le fichier .XML qui contient la structure de CUBE. Il faut bien structurer et lier les clés primaires et étrangères et les dimensions sinon le cube va donner des erreurs sur le PENTAHO SERVER.



*Schéma de la structure de cube sur **PENTAHO SCHEMA WORKBENCH***

Après la vérification de toutes les paramètres, on publie notre schéma sur le **PENTAHO SERVER**

### 3. PENTAHO SERVER : POUR LA CREATION DES RAPPORTS

**PENTAHO SERVER** est l'outil qui va nous aider à visualiser notre cube, donc on peut achever notre objectif pour analyser des mesures comme le chiffre d'affaire par projet, région et type de région.

Browse Files

Create New

Manage Data Sources

Documentation

#### Recents

You haven't opened anything recently. Browse your files.

Browse Files

#### Favorites

You haven't selected any favorites yet. Add

#### Pentaho Business Analytics

Get help and contribute with your knowledge. Find here some ways how you can do it.



Documentation

Forums

Mailing Lists

L'affichage de la première fenêtre de **PENTAHO SERVER**

## Database Connection

General  
Advanced  
Options  
Pooling

Connection Name:  
omranedb\_targetV2

Database Type:  
Generic database  
H2  
Hypersonic  
MonetDB  
MySQL  
Pentaho Data Services  
PostgreSQL  
Snowflake  
SparkSQL

Settings  
Host Name:  
localhost  
Database Name:  
omraneDB\_target  
Port Number:  
3306  
User Name:  
root  
Password:

Access:  
Adding Databases ?

Test Connection  
Connection to database [ omraneDB\_target ] succeeded  
OK

OK Cancel

Création d'une nouvelle connexion avec la base de données « **omraneDB\_target** »

Import Analysis

Mondrian File:  
OmraneSchemaV2.mondrian.xml

Select from available data sources.  
Manually enter data source parameter values.

Parameters

Name	Value
DataSource	omranedb_targetV2
EnableXmla	false

Save Close

## Manage Data Sources

New Data Source

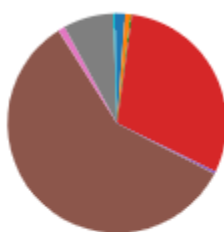
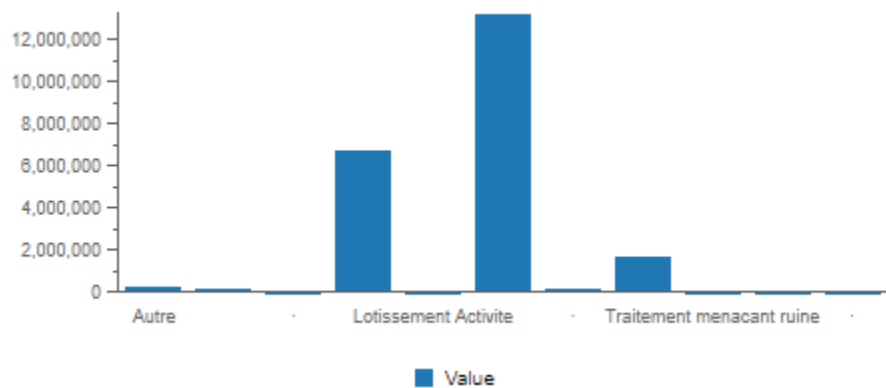
Datasource	Type
AgileBI	JDBC
omranedb_targetV2	JDBC
OmraneSchemaV2	Analysis
SampleData	Analysis
SampleData	JDBC

Close

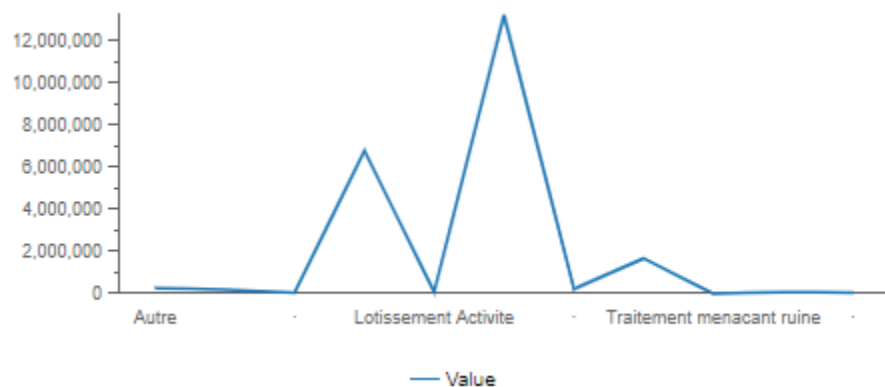
Premièrement, on peut remarquer que notre schéma « **OmraneSchemaV2** » et lister dans le “Manage Data Sources”.

Maintenant on va liée notre connexion avec le schéma et enregistre les modifications.

On peut visualiser maintenant notre schéma sur le “**Dashboard CDE**”. Par exemple, analyser la mesure : le chiffre d’affaire par projet.



■ Autre ■ Commerces ■ Equipements ■ Logements ■ Lotissement Activite  
 ■ Lotissement Habitat ■ Rehabilitation ■ Restruturation QHNR  
 ■ Traitement menacant ruine ■ ZAP ■ ZUN - Ville nouvelle



*Les différents diagrammes qu’on peut visualiser pour la mesure du chiffre d’affaire par projet*

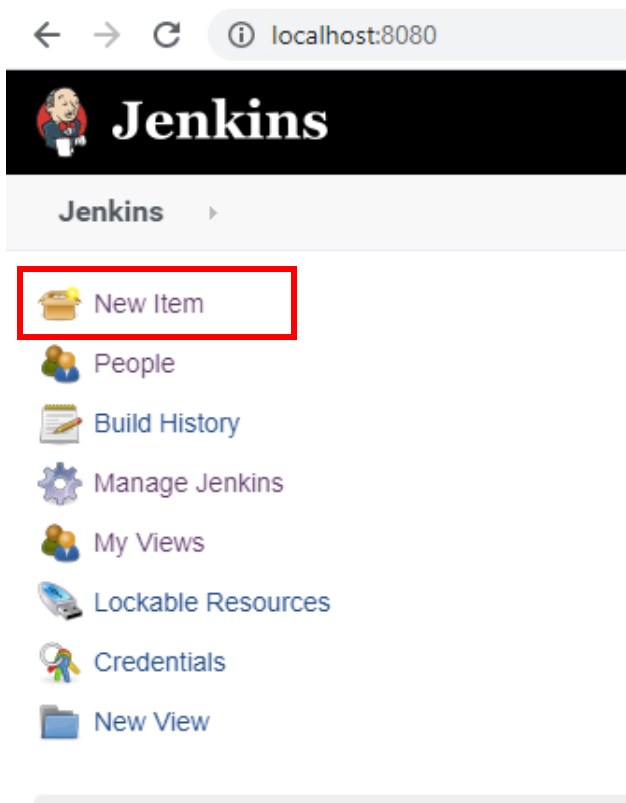
### III. AUTOMATISER LE SYSTEME AVEC LA TECHNOLOGIE JENKINS

**Jenkins** est un outil open source d'intégration continue, il génère des projets qui peuvent être amorcées par différents moyens, tels que les mécanismes de planification similaires au "cron".



# Jenkins

On va utiliser **Jenkins** pour automatiser l'exécution périodique du "Job" qu'on a créé juste avant, par exemple le lancement du "Job" chaque heure. Pour faire cela, il faut d'abord installer Jenkins et le lancer sur le serveur localhost.



Dans cette interface, on va créer un nouvel objet de type "Freestyle Project", on lui donne un nom, dans notre cas « **omraneAutomation** » et on continue dans la configuration.

Dans la partie “Build Triggers”, on choisit « **Build periodically** » pour donner l’aspect périodique de notre automatisation. Pour notre cas, on a donné comme exemple l’exécution de l’automatisation chaque heure.



**Build Triggers**


- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☒ Build periodically

Schedule `H * * * *`

Would last have run at mardi 2 juin 2020 07 h 43 WEST; would next run at mardi 2 juin 2020 08 h 43 WEST.

*L’interface de la configuration de l’automatisation, la partie du “Build Triggers”*

Pour la partie de l’exécution de la commande Windows batch, on va spécifier l’emplacement de la configuration “**Kitchen.bat**” de Pentaho Data Integration, et l’emplacement de notre **fichier “Job”**.



**Build**

Execute Windows batch command

Command `C:\Pentaho\design-tools\data-integration\Kitchen.bat -file=C:\Users\User\Desktop\JenkinsRun\omraneJob.kjb`

See [the list of available environment variables](#)

*L’interface de la configuration de l’automatisation, la partie du “Build”*

En sauvegarde la configuration de l’automatisation et en lance le “Build” en cliquant sur “Build now”.

## Console Output

```
Démarré par l'utilisateur admin
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\omraneAutomation
[omraneAutomation] $ cmd /c call C:\Windows\TEMP\jenkins2522037065648610985.bat

C:\Program Files (x86)\Jenkins\workspace\omraneAutomation>C:\Pentaho\design-tools\data-integration\Kitchen.bat -file=C:\Users\User\Desktop\JenkinsRun\omraneJob.kjb
DEBUG: Using PENTAHO_JAVA_HOME
DEBUG: _PENTAHO_JAVA_HOME=C:\Pentaho\java
DEBUG: _PENTAHO_JAVA=C:\Pentaho\java\bin\java.exe
DEBUG: PENTAHO_INSTALLED_LICENSE_PATH=C:\Pentaho\installedLicenses.xml

C:\Pentaho\design-tools\data-integration>"C:\Pentaho\java\bin\java.exe" "-Dpentaho.installed.licenses.file=C:\Pentaho\installedLicenses.xml" "-Xms1024m" "-Xmx2048m" "-XX:MaxPermSize=256m" "-Dhttps.protocols=TLSv1,TLSv1.1,TLSv1.2" "-Djava.library.path=libs\win64" "-Djava.endorsed.dirs=C:\Pentaho\java\jre\lib\endorsed;C:\Pentaho\java\lib\endorsed;C:\Pentaho\design-tools\data-integration\system\karaf\lib\endorsed" "-DKETTLE_HOME=" "-DKETTLE_REPOSITORY=" "-DKETTLE_USER=" "-DKETTLE_PASSWORD=" "-DKETTLE_PLUGIN_PACKAGES=" "-DKETTLE_LOG_SIZE_LIMIT=" "-DKETTLE_JNDI_ROOT=" "-Dpentaho.installed.licenses.file=C:\Pentaho\installedLicenses.xml" -jar launcher\launcher.jar -lib ..\libs\win64 -main org.pentaho.di.kitchen.Kitchen -InitialDir "C:\Program Files (x86)\Jenkins\workspace\omraneAutomation" -file C:\Users\User\Desktop\JenkinsRun\omraneJob.kjb
07:57:08,609 INFO [KarafBoot] Checking to see if org.pentaho.clean.karaf.cache is enabled
07:57:14,697 INFO [KarafInstance]
*****
*** Karaf Instance Number: 1 at C:\Pentaho\design-tools\data-integration\ \ ***
*** system\karaf\caches\kitchen\data-1 ***
*** Karaf Port:8802 ***
*** OSGI Service Port:9051 ***
*****
juin 02, 2020 7:57:15 AM org.apache.karaf.main.Main$KarafLockCallback lockAcquired
INFO: Lock acquired. Setting startlevel to 100
2020/06/02 07:57:15 - Kitchen - Démarrage.
juin 02, 2020 7:57:16 AM org.apache.aries.spifly.BaseActivator log
INFO: Examining bundle for SPI provider: org.eclipse.jetty.http
juin 02, 2020 7:57:16 AM org.apache.aries.spifly.BaseActivator log
INFO: Found SPI resource: bundle://142.0:0/META-INF/services/org.eclipse.jetty.http.HttpFieldPreEncoder
juin 02, 2020 7:57:16 AM org.apache.aries.spifly.BaseActivator log
INFO: Loaded SPI provider: class org.eclipse.jetty.http.HttpFieldPreEncoder
juin 02, 2020 7:57:16 AM org.apache.aries.spifly.BaseActivator log
INFO: Registered service: org.apache.felix.framework.ServiceRegistrationImpl@761d3848
juin 02, 2020 7:57:16 AM org.apache.aries.spifly.BaseActivator log
INFO: Registered provider: org.eclipse.jetty.http.HttpFieldPreEncoder in bundle org.eclipse.jetty.http
```

*L'interface du Console Log qui affiche tous les détails de l'exécution qui a terminé avec succès !*

## • Conclusion :

Dans ce projet, on a traité le cas réel de la simulation de la transformation des données et des informations de la société depuis un fichier Excel ou fichier csv vers un entrepôt de données, toute en gardant la traçabilité et le suivre des données en fonction du temps avec l'outil **Pentaho Data Integration**. Même aussi la visualisation de ces données de façon plus concrète à l'aide des graphes et des diagrammes de l'outil **Pentaho Server**. L'outil **Jenkins** aide à l'automatisation de ce système au niveau de la totalité des transformations.