UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences
Département d'Informatique

INFO-H-419 Data Warehouses

Assignment - Part II

Teacher :

Toon Calders

Assistant :

Hatem Haddad

Group E members :

Raymond Lochner (000443637)

Patrick Randriambololona (000444674)

Aldar Saranov (000435170)

Antoine Vandevenne (000333030)

Academic year 2017 - 2018

Contents

1 Introduction

In this report, we review all the steps taken to load and process the provided data, deploy a cube over it and finally generate a report.

1.1 Assumptions

Over the course of the second part of this project, a few assumptions had to be made. Indeed, although the assignment includes a lot of details, we sometimes encountered some vagueness or missing details. In order to be able to carry out this assignment, the following assumptions have been made:

- The fact date corresponds to the end date of the snapshot timespan.
- The salary of an employee as given in the initial database and snapshots is always to be considered a monthly payment. If a yearly payment was in fact to be considered, the queries taking the earnings of employees over months into account should be modified to divide the salary by 12.
- The solution delivered along with this report consists of a proof-of-concept. What it means is that we show that we are capable of devising a model to process the data as well as using it to fetch any requested data. However, the model itself is not meant to be used extensively nor released as a final product. For this reason and as an example, it was decided not to create indexes. It implies that the processing time will not benefit from indexing but that we consider it a minor issue in regards to the assumption of proof-of-concept.

1.2 Modification of the proposed model

In order to answer the future queries, that are related to the history of employee-client mapping, it was decided to add start and end attributes to the [bridgeEmployee] table. However, it should not be confused with type-2 changes because it has no historical attributes. The finishing and adding new relations have to be done manually by means of SQL tasks instead of using standard SSIS SCD task.

1.3 Optimization

Because the SQL date type is a simple data type and does not requires much storage space, it was decided to generate an entry including an id as the primary key along with the actual date value for each day between 01-01-1900 and 01-01-2100. Such a wide range should easily cover any birth date or hire date of future employees for a reasonable period. Thanks to this, it will not be needed to create a new entry on-fly for each new employee whose birth date or hire date does not match an already existing entry in the data warehouse. Instead, we make the processing of new snapshots faster by sacrificing a negligible and constant storage space. If it happens that a wider range or dates is required, the current one could be easily extended.

1.4 Selection of the project variant

After careful reading and discussion about ambiguous parts of the assignment, we decided to aim for the level 4 and full-size variant of this project, which we hopefully managed to carry on to its full completion.

2 Initial load of the data warehouse

The entire process of the initial load of the data warehouse is displayed in Figure 1. The sequence of operations is split in two phases:
First, the tasks on the left side load the data from the CSV files into a temporary database, hence corresponding to a 3-layer architecture. It is worth noting that some tasks performs several loads in parallel, as shown in Figure 2 for the task 'Load country, department, title, clients', while the following tasks only take care of one load each. This is due to the fact that a key dependence exists between the tables country and state, state and city, city and employee, departments and employee, client_emp and clients and employee. In order to maintain this dependence, the loads are performed in this particular order.
Second, the tasks of the right side perform ETL's to load the data from the temporary database and into the data warehouse.

A data flow view of all remaining tasks, namely 'Load state', 'Load city', 'Load employee', 'Load client emp', 'Convert city and clients', 'Convert employee', 'Convert factSalary and bridgeEmployee', is available respectively in Figures 3, 4, 5, 6, 7, 8 and 9.
In 'Convert Employee' task we use derived column to specify the start and end dates. For the start value we use a constant value of the "1991-01-01" date. For the end value we use null value, since all the employees that are currently being loaded are considered actual.

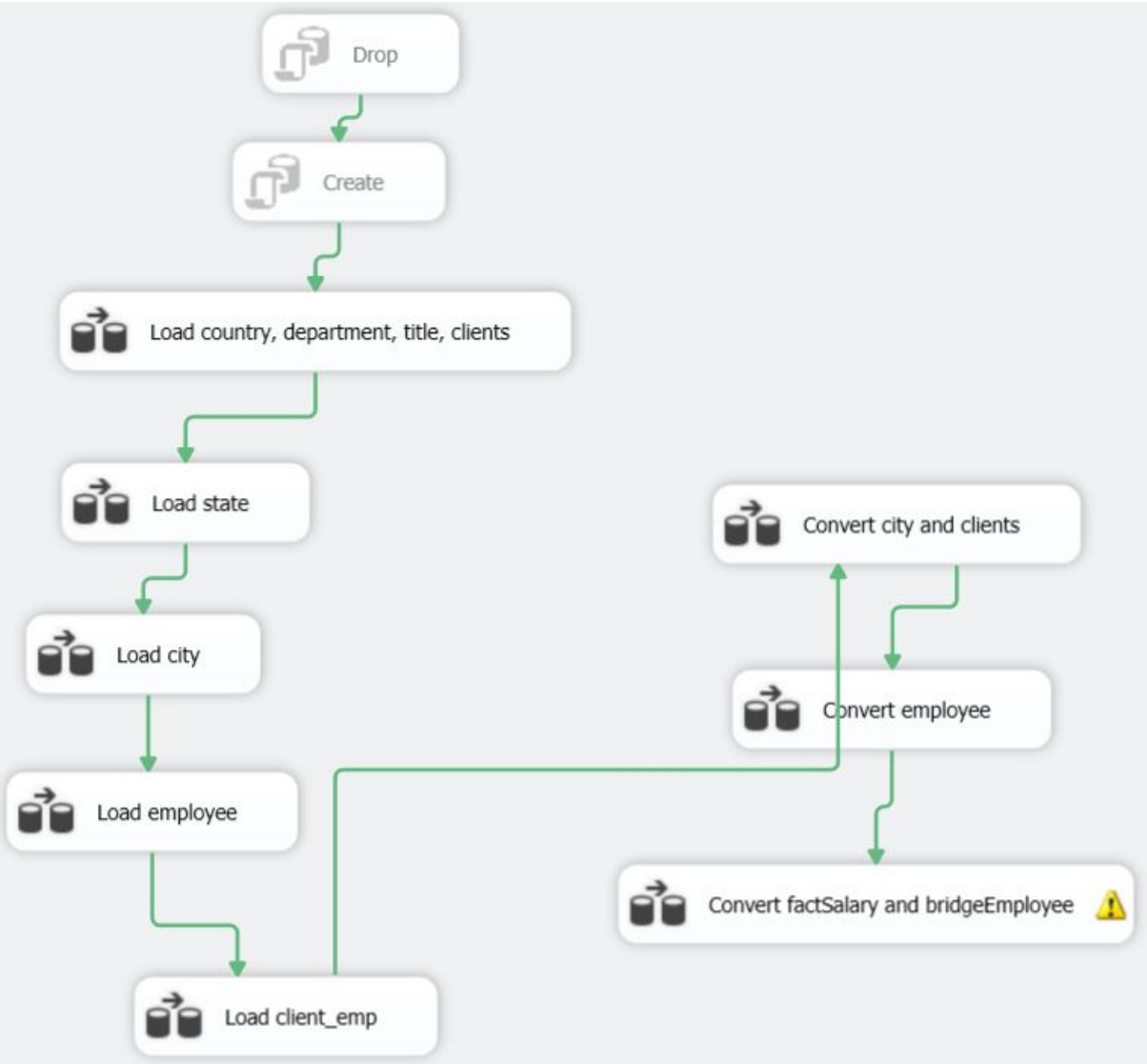


Figure 1: Control flow view of the initial load of the data warehouse

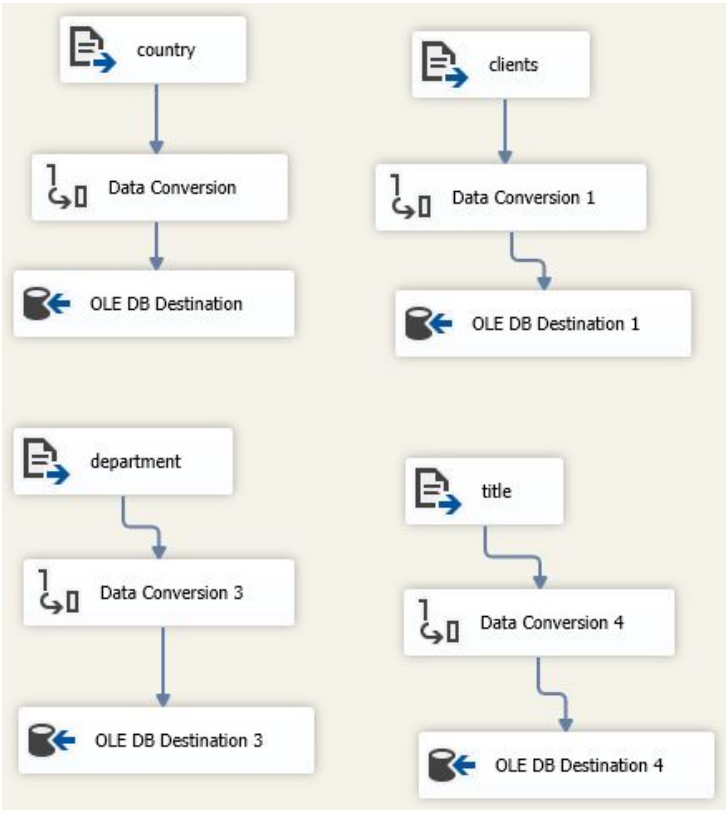


Figure 2: Data flow view of the task performing the load of the tables country, clients, department and title

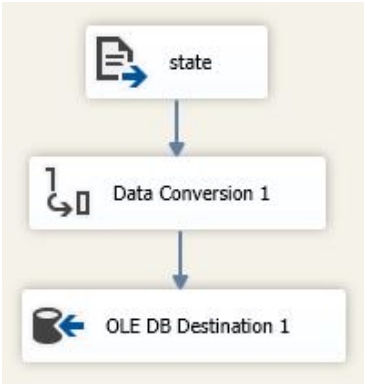


Figure 3: Data flow view of the task performing the load of the table state

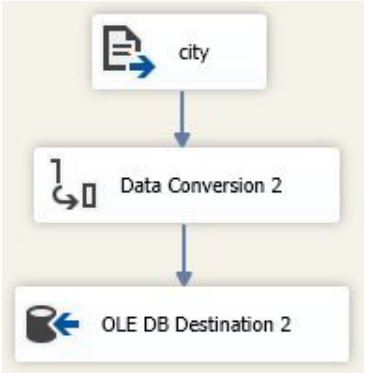


Figure 4: Data flow view of the task performing the load of the table city

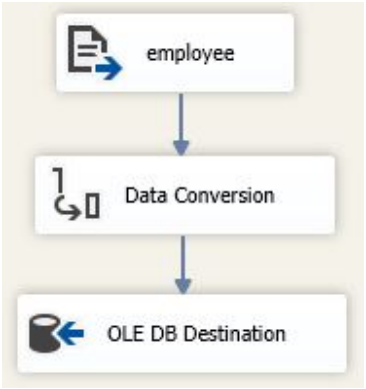


Figure 5: Data flow view of the task performing the load of the table employee

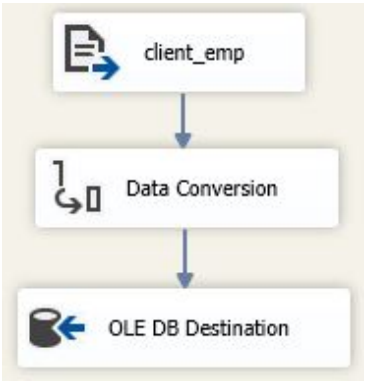


Figure 6: Data flow view of the task performing the load of the table client emp

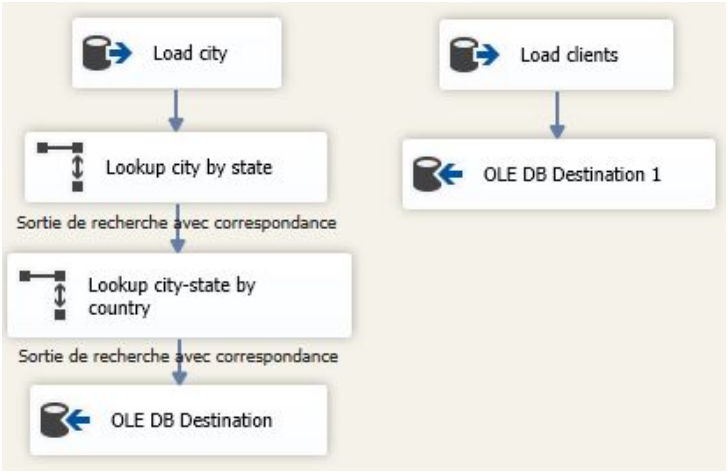


Figure 7: Data flow view of the task performing the conversion of the tables city and clients

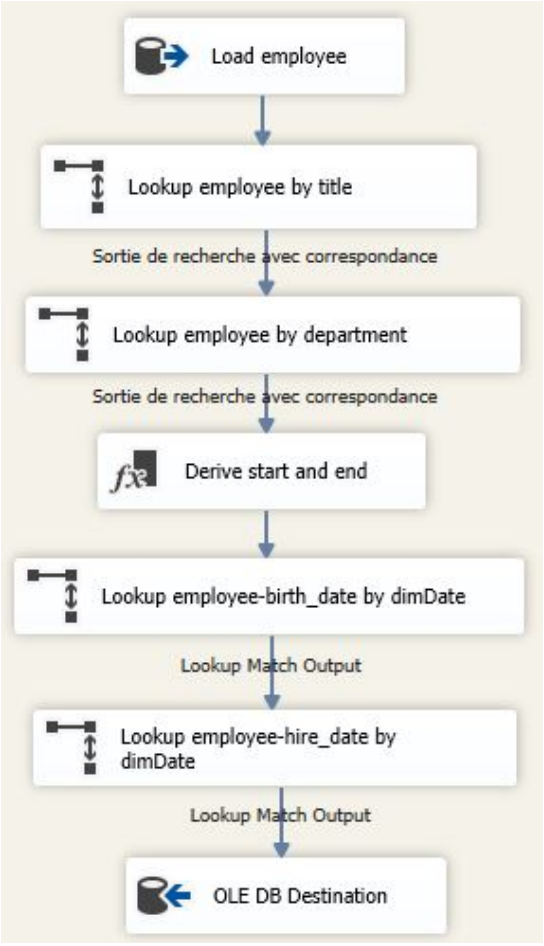


Figure 8: Data flow view of the task performing the conversion of the table employee

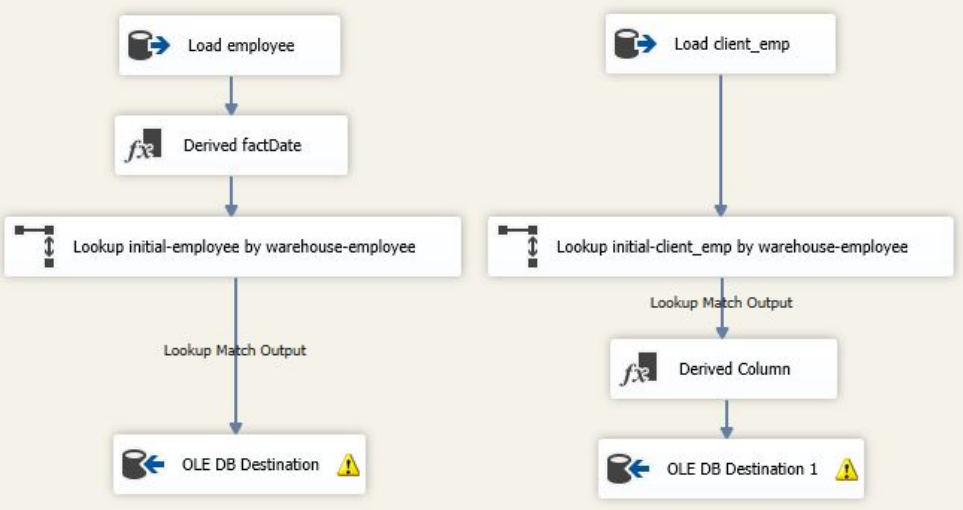


Figure 9: Data flow view of the task performing the conversion of the tables factSalary and bridgeEmployee

3 Processing of the snapshots and update of the data warehouse

The control flow of the ETL flow is depicted in Figure 10. The proposed looping template was used as a base for the script. As mentioned, we could use the data reducing task in order to switch the size complexity.

3.1 SnapDateId caching

In addition to the template loop tasks, we have established a 'Determine variable @snapDateId' SQL task that sets the variable of @snapDateId to the ID of the corresponsing date of @snapDate in the [dimDate] table. Thus, we optimize the insertion of fact dates and end dates of the dimensions and avoid excessive [dimDate] look-ups. The snapshots being processed can be specified in the for-loop parameters.

3.2 Slowly changing dimensions

The task that changes [dimCity], uses standard SSIS SCD task triggered by changing values of any of the attributes according to type-1 changing. The table [dimEmployee] is a subject for type-2 changes triggered by the attributes [current_city_id], [dept_id], [dept_name], [marital_status], [no_of_children], [title], [title_id], and type-1 triggered by the rest of the attributes. The SCD works in two-flag mode with start and end dates. The table [dimClients] is also subject to type-2 changes triggered by attributes [client] and [category]. However, unlike the table [dimEmployee], one-flag mode is applied and therefore instead of the effective date range, one can find a single boolean flag of actuality.

3.3 Fact filling

In order to insert new facts, the script looks-up the [EID] of the corresponding employee version of the fact by disregarding the expired version of this employee. This is done by creating a derived [end] column full of null-values with consequent look-up by this column and the employee natural key. The [DateId] is assigned to the pre-computed @snapDateId foreign key value.

3.4 Updating and adding bridge relations

As it was previously said, the [bridgeEmployee] updating process should not be confused with applying SCD type-2 changing, despite it may remind one. First an ad-hoc task determines the table of the employee-client mapping, expressed in surrogate keys, in order to compare it to the one stored in the warehouse. After this, one determines the mapping relations that existed in the previous state and does not exist anymore in the current state. This is done by the mean of a SQL task. Later, we assign the current snapshot end date to the end date of these employees. Similarly, one determines the mapping relations that did not exist before, but does now. These relations are new in the relation mapping and therefore should be appended to the [bridgeEmployee] table. The surrogate keys are inserted as well as the start date being equal to the current snapshot start date, and end date being equal to null.

Once again, a data flow view of the tasks 'Change dimCity', 'Change dimEmployee', 'Change dimClients', 'Change factSalary', 'Determine new client emp surrogate' and 'Append new relations' is available respectively in Figures 11, 12, 13, 14, 15 and 16.

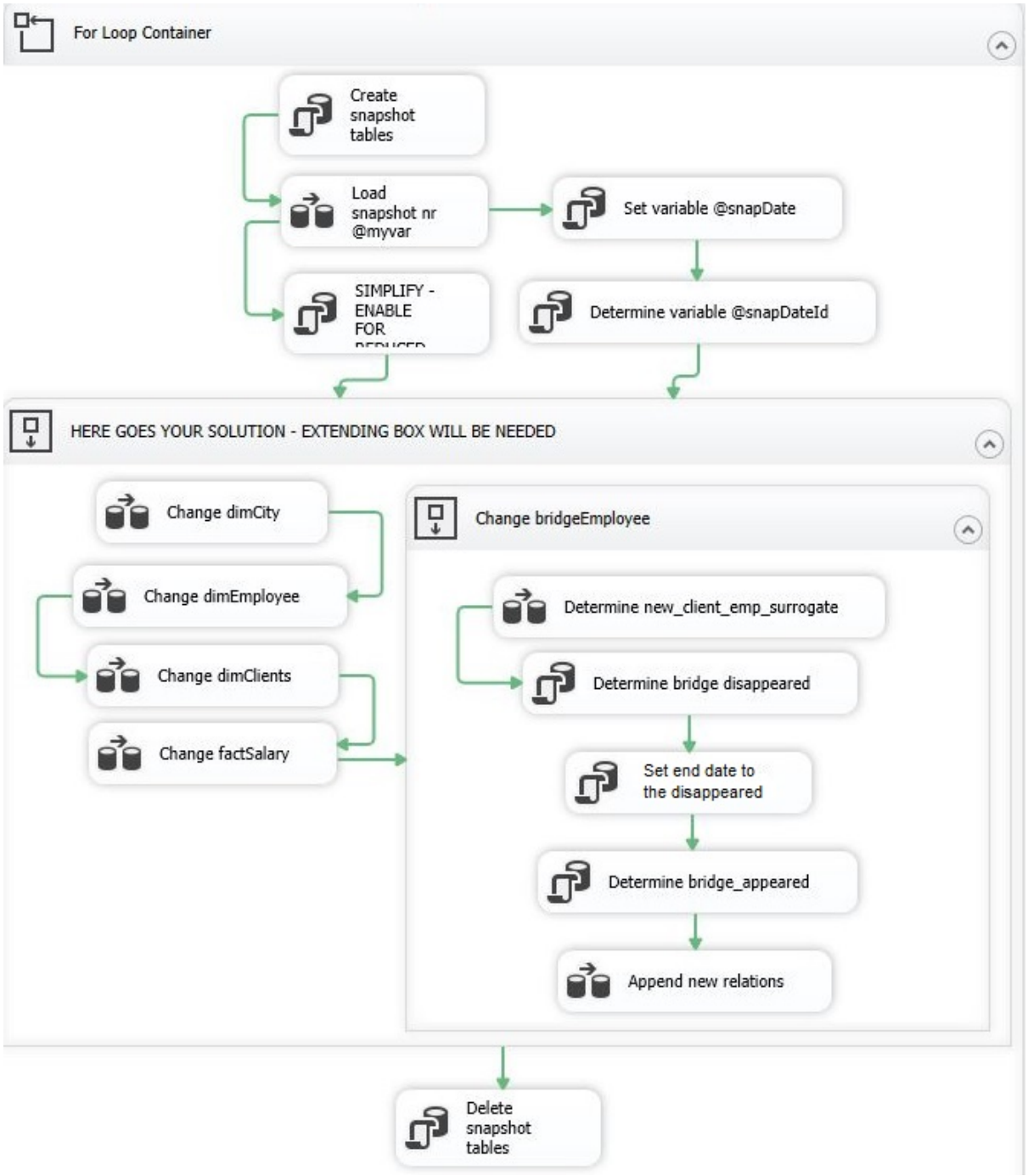


Figure 10: Control flow view of the load of the snapshots into the data warehouse

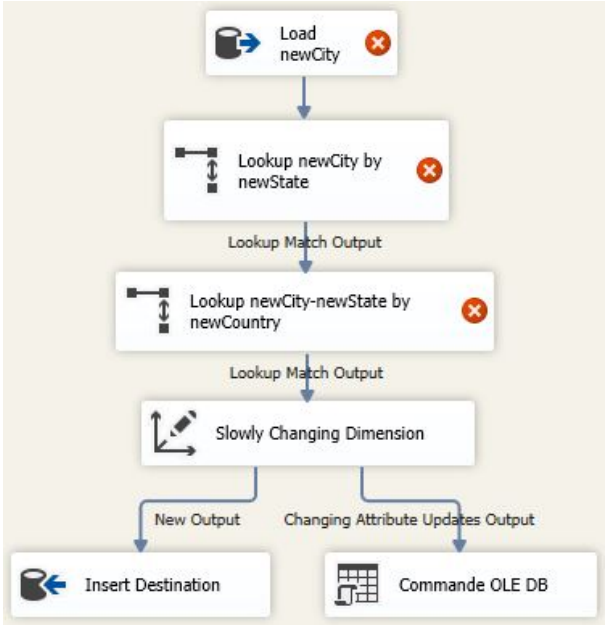


Figure 11: Data flow view of the task performing ETL process of the [dimCity] dimension

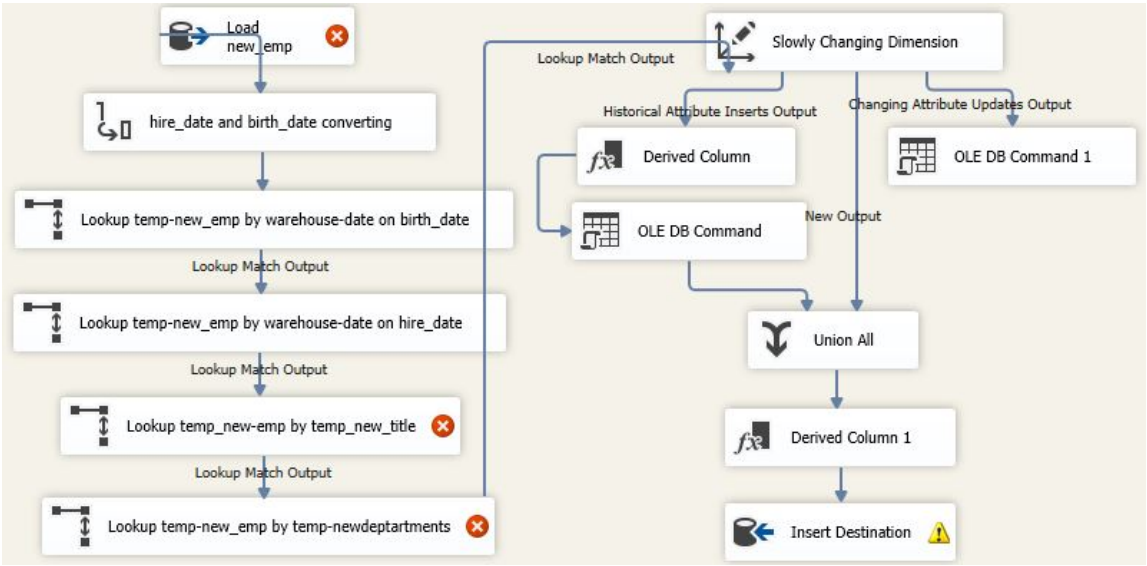


Figure 12: Data flow view of the task performing ETL process of the [dimEmployee] dimension

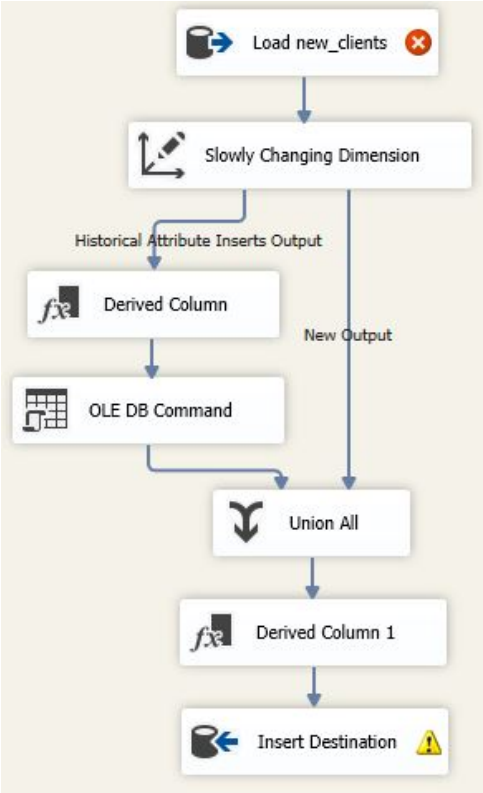


Figure 13: Data flow view of the task performing ETL process of the [dimClients] dimension

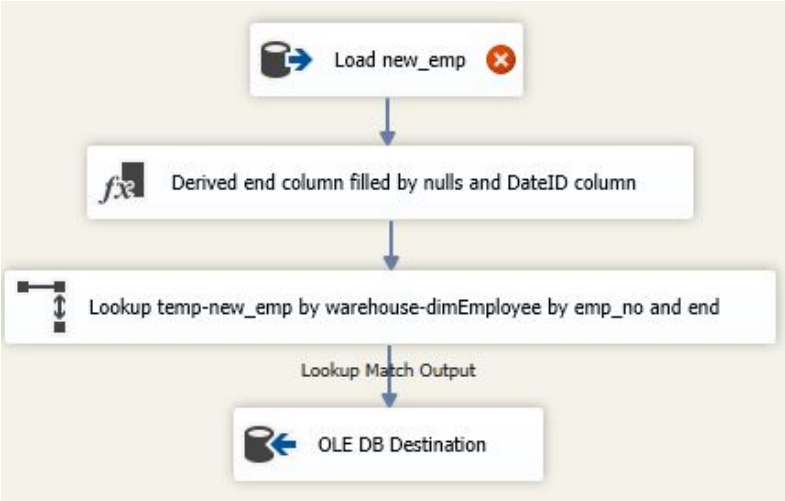


Figure 14: Data flow view of the task performing the ETL of the [factEmployee] fact

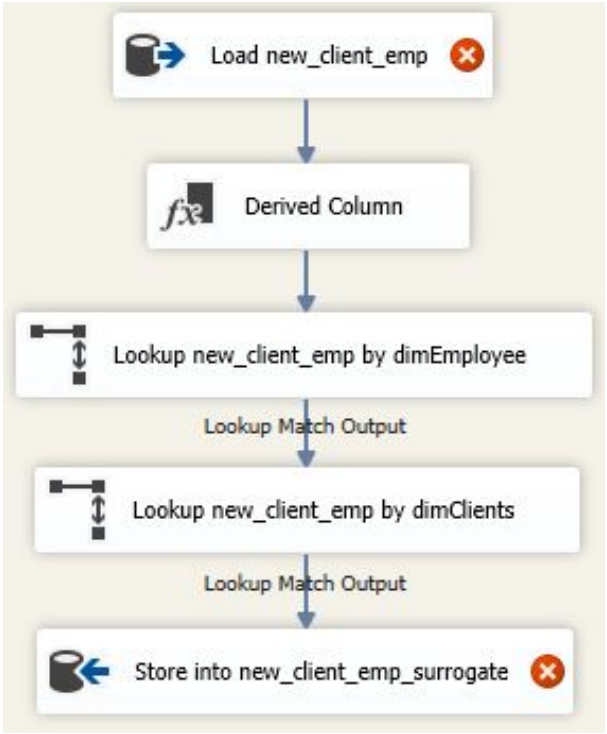


Figure 15: Data flow view of the task performing the computation of the new employee-client relations

Figure 16: Data flow view of the task performing appending of the new employee-client relations to the database

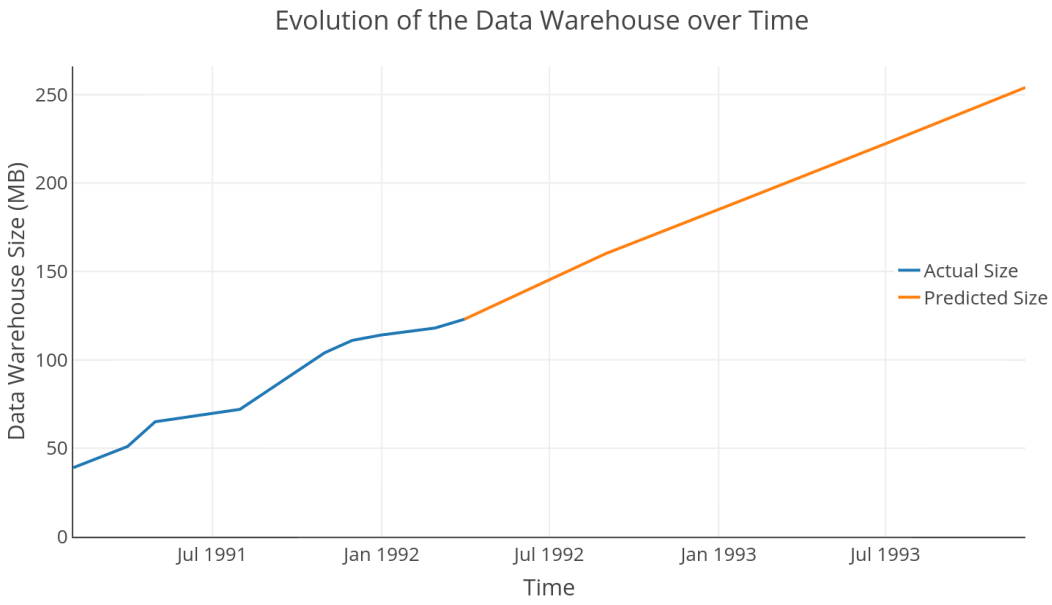


Figure 17: Historical and predicted size of the data Warehouse over time. Blue line: actual size, Red line: predicted size

4 Estimates of the growth of the data warehouse

The actual size of the data warehouse resulting from the processing of snapshots is shown in Table 1a. Using nonlinear regression (4-parameter logistic regression (4PL)) to predict the data warehouse size in the future results in the values shown in Table 1b. A graph with the actual and predicted values is displayed in Figure 17.
If we assume that the company will experience similar workload as over the 15 first snapshots and hence the same amount of changes resulting in the same increase of size in the data warehouse, we can assume that future snapshots processing will bring the data warehouse size to a value close to that of the predicted line. This estimation takes into account the increasing level of complexity of the last snapshots.
Our prediction is then that the data warehouse will experience a linear growth, similar to that observed after the processing of the provided snapshots.

(a) Actual data warehouse size

Snapshot	Time	Size(MB)
1	1991-01-31	39
3	1991-03-31	51
4	1991-04-30	65
7	1991-07-31	72
10	1991-10-31	104
11	1991-11-30	111
12	1991-12-31	114
13	1992-01-31	116
14	1992-02-28	118
15	1992-03-31	123

(b) Predicted data warehouse size

Time	Snapshot	Size(MB)
1992-08-31	20	160
1993-01-30	25	191
1993-06-30	30	222
1993-11-30	35	254

5 Deployment of a data cube and generation of a report

The data cube was established using an SSAS project. Four dimensions were created: [dimCity], [dimClients], [dimDate] and [dimEmployee]. The hierarchies that correspond to these dimensions are specified in the Figures 18, 19, 20, 21.
Two facts present - [factEmployee] and [bridgeEmployee] since it is used as a bridge between employees and clients. For [factEmployee] we have [salary] and [Fact Salary Count] as measures. For [bridgeEmployee], [Bridge Employee Count] is the measure. The relationships between the facts and the dimensions are described in Figure 22.

Based on the problem description of the first assignment, we have created the five reports requested by the company by means of an SSRS project. The actual *.xlsx* files can be found in the project submission package but screenshots of them can be found in the appendix.

- Query 1 (Average salary of managers per year for the sales and marketing departments) in Appendix A.1
- Query 2 (Monthly average salary per gender) in Appendix A.2
- Query 3 (Geographic employee distribution) in Appendix A.3
- Query 4 (Monthly average salary for all employees working for a specific client between January 1990 and July 1990) in Appendix A.4
- Query 5 (Total salary per job title since 1989) in Appendix A.5

The negative values in query 2 and 5 are due to the fact that we used the *int* data type for the salary fact. This is because the *int* type can only store 4 bytes, which is integer numbers between $-2,147,483,648$ to $2,147,483,647$. Changing the data type to *bigint* in the database or casting the arithmetic functions used in the queries like *sum* to

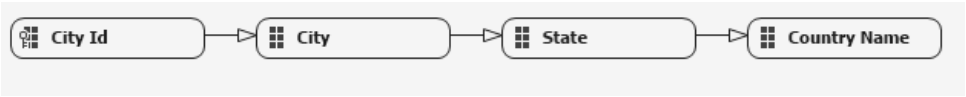


Figure 18: The attribute hierarchy of the [dimCity] dimension

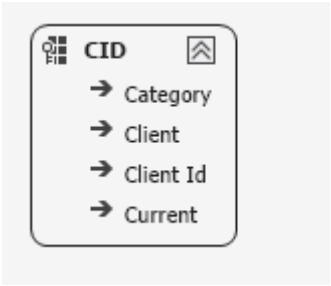


Figure 19: The attribute hierarchy of the [dimClients] dimension

bigint would resolve this problem[1]. However, under our assumption that we are devising a proof of concept, we decided not to invest too much time in the resolution of this minor issue, as we consider it to be irrelevant to the purpose of this project.

[1]As described in the Microsoft documentation: `https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql`

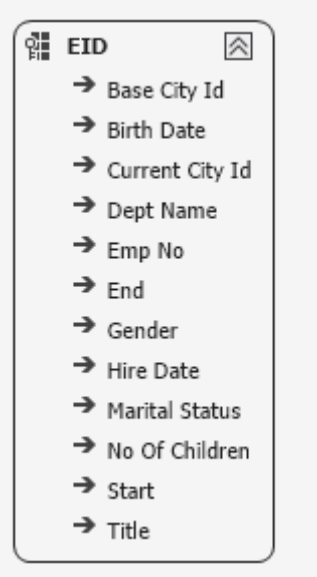


Figure 20: The attribute hierarchy of the [dimEmployee] dimension

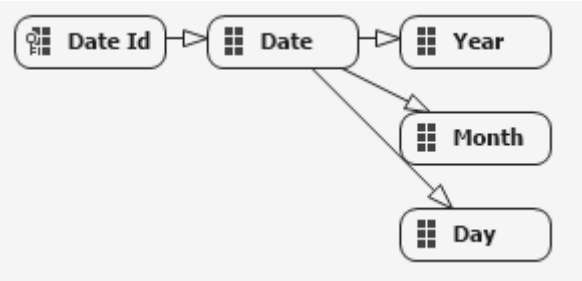


Figure 21: The attribute hierarchy of the [dimDate] dimension

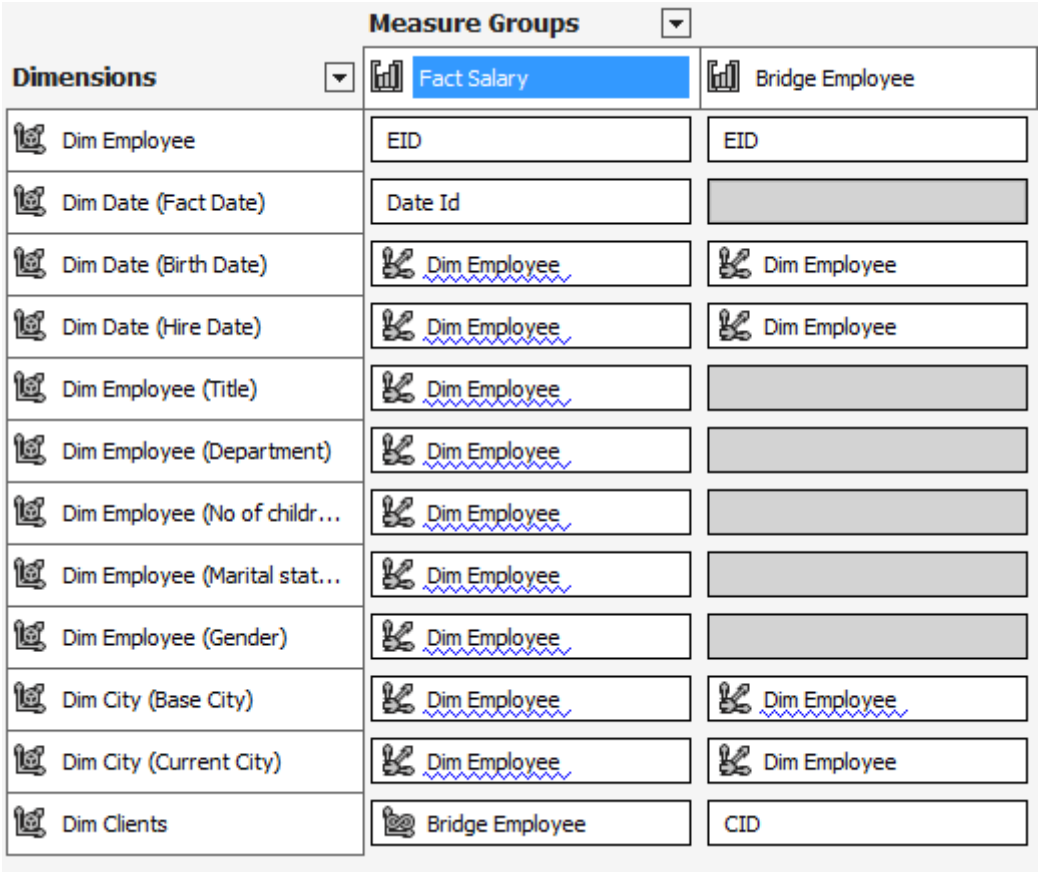


Figure 22: Dimension usage table that describes relationships between facts and dimensions

A Reports generated to answer asked queries

A.1 Query 1

	A	B
1	1-salary-title-dept	
2	Year	Avg Salary
3	1991	82230.53846
4	1992	82704

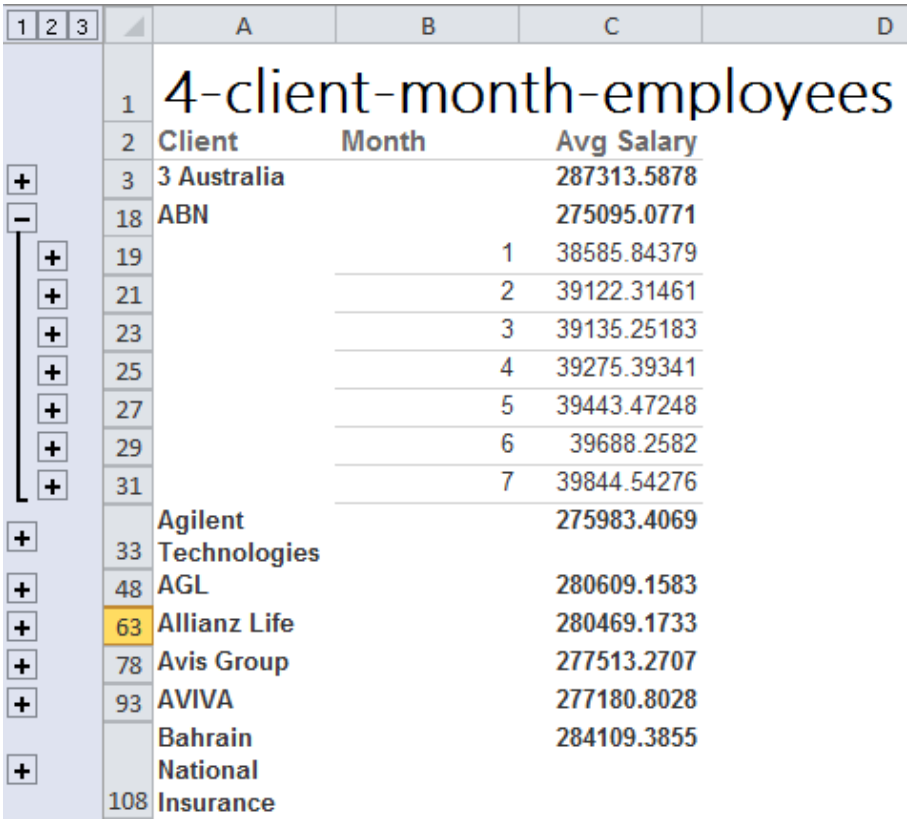
A.2 Query 2

	A	B
1	2-salary-gender	
2	Gender	Avg Salary
3	F	-50.55801454
4	M	-60.12645809

A.3 Query 3

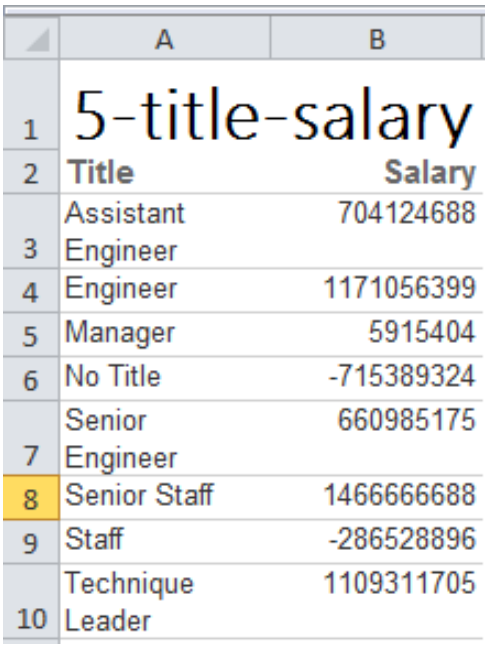
	A	B	C	D
1	3			
2	Country Name	State	City	Fact Salary Count
3	Brazil			106950
191	France			7667
192		Ile-de-France		2559
193			Kundan	1292
195			Paris	1267
197		Provence-Alpes-Cote d'Azur		3847
204		Rhone-Alpes		1261
207	Germany			39397
278	India			56202

A.4 Query 4



	A	B	C	D
1	4-client-month-employees			
2	Client	Month	Avg Salary	
3	3 Australia		287313.5878	
18	ABN		275095.0771	
19		1	38585.84379	
21		2	39122.31461	
23		3	39135.25183	
25		4	39275.39341	
27		5	39443.47248	
29		6	39688.2582	
31		7	39844.54276	
33	Agilent Technologies		275983.4069	
48	AGL		280609.1583	
63	Allianz Life		280469.1733	
78	Avis Group		277513.2707	
93	AVIVA		277180.8028	
108	Bahrain National Insurance		284109.3855	

A.5 Query 5



	A	B
1	5-title-salary	
2	Title	Salary
3	Assistant Engineer	704124688
4	Engineer	1171056399
5	Manager	5915404
6	No Title	-715389324
7	Senior Engineer	660985175
8	Senior Staff	1466666688
9	Staff	-286528896
10	Technique Leader	1109311705