

|  |
| --- |
| EPAM Systems, RD Dep.  data warehouse for Auto sales |
| Картинки по запросу car sales |
|  |

|  |  |
| --- | --- |
| Abbreviations and Acronyms | |
| DWH | Data Warehouse |
| 3NF | Third normal form |

Contents

[1. Business Description 4](#_Toc500250524)

[1.1 Business background 4](#_Toc500250525)

[1.2 Problems because of poor data management 4](#_Toc500250526)

[1.3 Benefits from implementing a Data Warehouse 4](#_Toc500250527)

[2. DIMENSIONS OF A BUSINESS 5](#_Toc500250528)

[3. DIMENSIONAL MODEL. STAR SCHEMA 7](#_Toc500250529)

[4. 3NF SCHEMA 8](#_Toc500250530)

[5. Data Flow 9](#_Toc500250531)

[6. Fact Table Partitioning Strategy 9](#_Toc500250532)

[7. report layout 11](#_Toc500250533)

1. Business Description

## Business background

Nowadays there are many different ways of buying cars. Thousands of cars are sold and bought daily. Moreover, sometimes it is complicated to keep track of all changes in car prices and the popularity of models especially because of the huge amount of different characteristic such as vehicle type, fuel type, and gearbox.

## Problems because of poor data management

The problems that business is facing because of poor data management are that it is so complicated to find all the information about the cars such as price, year of registration, model and kilometer in one place. Usually this information divided into small pieces in different places. In addition, it is hardly to find the information about different types of model in one place.

## Benefits from implementing a Data Warehouse

With the help of data warehouse, it would be easier to find the information about cars, building different types of reports. In addition, it could be possible to see the difference in sales per years or per model, or per price. Completed information about car sold will be in one place.

This DWH would be suitable either for people, who wanted to buy a car, or for that people, who are interested in evaluation of the situation in the car market.

2. DIMENSIONS OF A BUSINESS

The business of the project is a car sale. Thousands of cars are sold and bought daily, so it becomes a problem to keep it in one place. Based on this, the grain of the model is a car for sale.

Fact table will include Seller\_id, Store\_id, Car\_id, Order\_Date\_id, Customer\_id, Order\_code, Cost, Min\_Price, Avg\_Price, SD\_Price.

Dimensions:

1. Seller. Here will be information about seller such as:

Table 1 DIM\_SELLER

| DIMENSION SELLER | | | |
| --- | --- | --- | --- |
| Column name | Column type | Other | Description |
| 1. Seller\_id 2. Seller\_name 3. Seller\_surname 4. Seller\_rating 5. Phone 6. Email | * Number(8) * Varchar2(200) * Varchar2(200) * Number(8) * Varchar2(200) * Varchar2(200) | PK | This dimension reflect the information about car seller such as name/surname, rating, phone and email. |

1. Store. In Store there is information about the location:

Table 2 DIM\_STORE

| DIMENSION STORE | | | |
| --- | --- | --- | --- |
| Column name | Column type | Other | Description |
| 1. Store\_id 2. Store\_name 3. Phone 4. Email 5. Street\_name 6. House\_number 7. City\_name 8. Country\_name | * Number(8) * Varchar2(200) * Varchar2(200) * Varchar2(200) * varchar2(200) * Varchar2(200) * Varchar2(200) * Varchar2(200) | PK | This dimension reflect the information about car stores such as store name, phone, email and location. |

1. Date dimension. Here will be:

Table 3 DIM\_DATE

| DIMENSION DATE | | | |
| --- | --- | --- | --- |
| Column name | Column type | Other | Description |
| 1. Date\_id 2. Day\_per\_week 3. Day\_per\_month 4. Day\_per\_year 5. Week\_per\_month 6. Week\_per\_year 7. Month\_number 8. Month\_name 9. Year 10. Day-Month 11. Year-Month | * Date * Number(8) * Number(8) * Number(8) * Number(8) * Number(8) * Number(8) * Varchar2(200) * Number(8) * Varchar2(200) * Varchar2(200) | PK | This dimension reflect the information about date of order registration. |

1. Cars. Here will be:

Table 4 DIM\_CAR

| DIMENSION | | | |
| --- | --- | --- | --- |
| Column name | Column type | Other | Description |
| 1. Car\_id 2. Car\_number 3. Car\_name 4. Car\_Registration\_Date 5. Vehicle\_type\_name 6. Engine\_Type\_name 7. Gearbox\_type\_name 8. Brand\_name 9. Model\_name 10. Repair\_status\_name 11. Power\_PS 12. Kilometers | * Number(8) * Number(8) * Varchar2(200) * Date * Varchar2(200) * Varchar2(200) * Varchar2(200) * Varchar2(200) * Varchar2(200) * Varchar2(200) * Number(8) * Number(8) | PK | This dimension reflect the information about car stores such as store name, phone, email and location. |

1. Customers. Here will be:

| DIMENSION SELLER | | | |
| --- | --- | --- | --- |
| Column name | Column type | Other | Description |
| 1. Customer\_id 2. Customer\_name 3. Customer\_surname 4. Age 5. Gender | * Number(8) * Varchar2(200) * Varchar2(200) * Number(8) * Number(8) | PK | This dimension reflect the information about car customer such as name/surname, age and gender. |

3. DIMENSIONAL MODEL. STAR SCHEMA

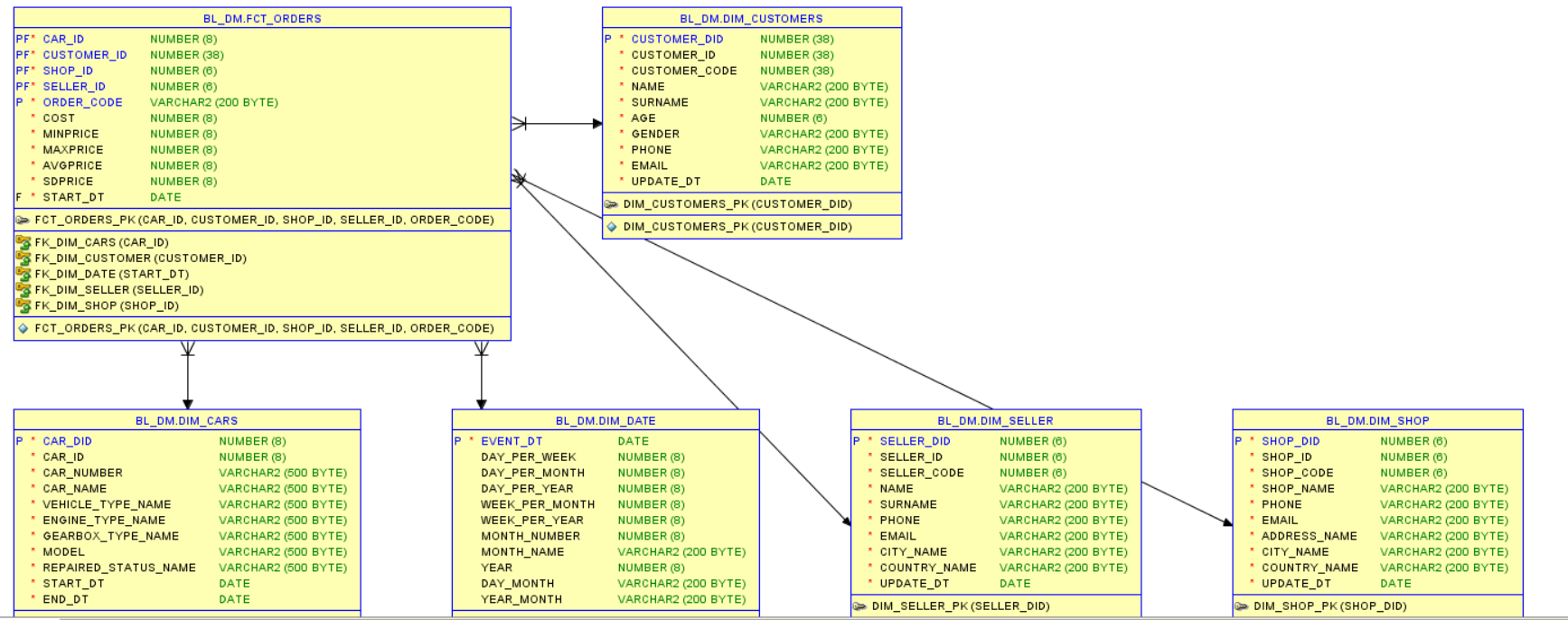


Figure 1 Star Schema

Advantages of star schema:

1. Higher productivity
2. While changes in one dimension, other will stay in the same conditional.

Disadvantages of star schema:

1. Need a lot of space

4. 3NF SCHEMA

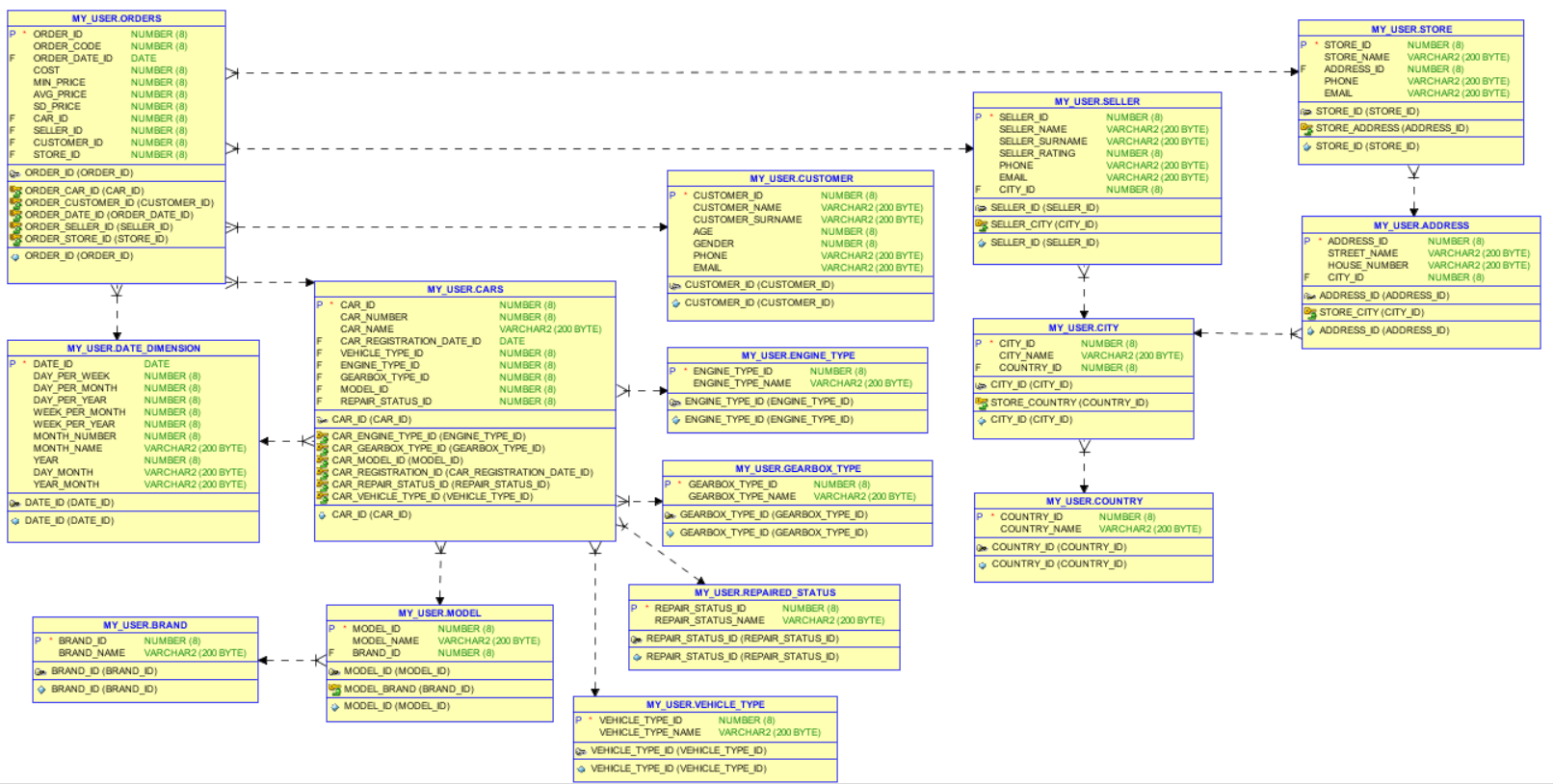


Figure 2 3NF Schema

5. Data Flow

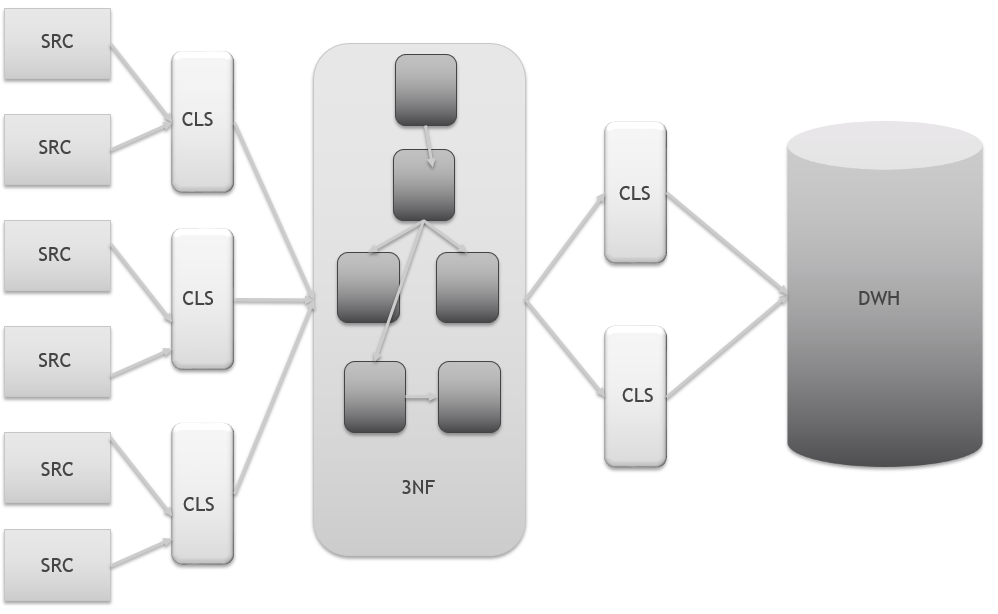
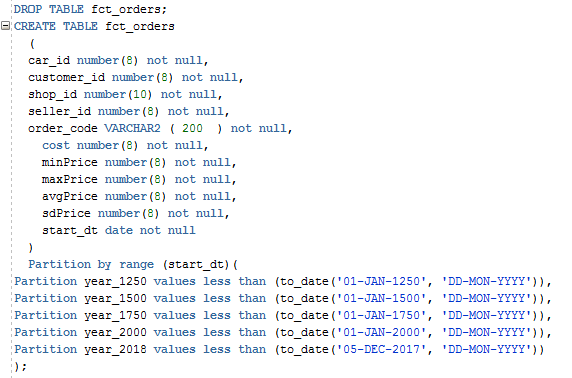


Figure 3 Data Flow

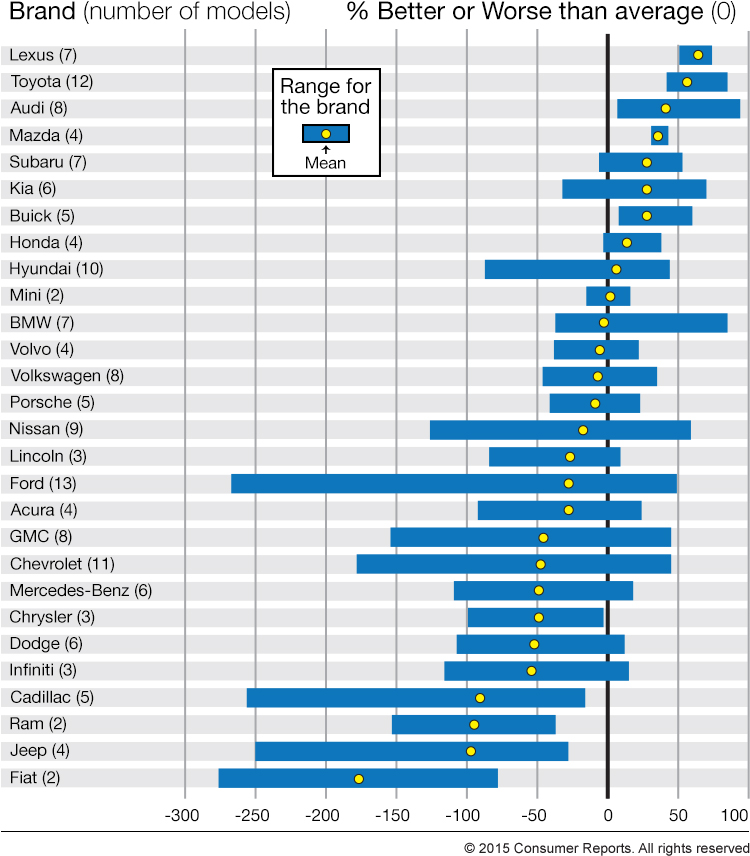
# 6. Fact Table Partitioning Strategy

Fact table was partitioning by years. It is really useful because my DWH had 1018 years and it is really complicated to use such amount of different years for searching.

* From 1000 to 1250
* From 1250 to 1500
* From 1500 to 1750
* From 1750 to 2000
* From 200 to 2017



7. report layout



8. Data model

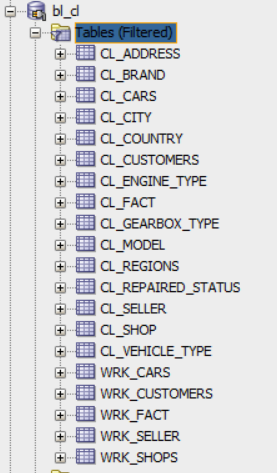
In my DWH there are 5 dimensions and two of them are SCD2 with start\_dt and end\_dt attributes. Moreover with date dimension you can analyze by year, month or even day.

Information for cars dimension and fact table were taken from <https://www.kaggle.com>. Information about customers and seller were taken from <http://www.fakenamegenerator.com>. Information about shops and date dimension were generated on my own.

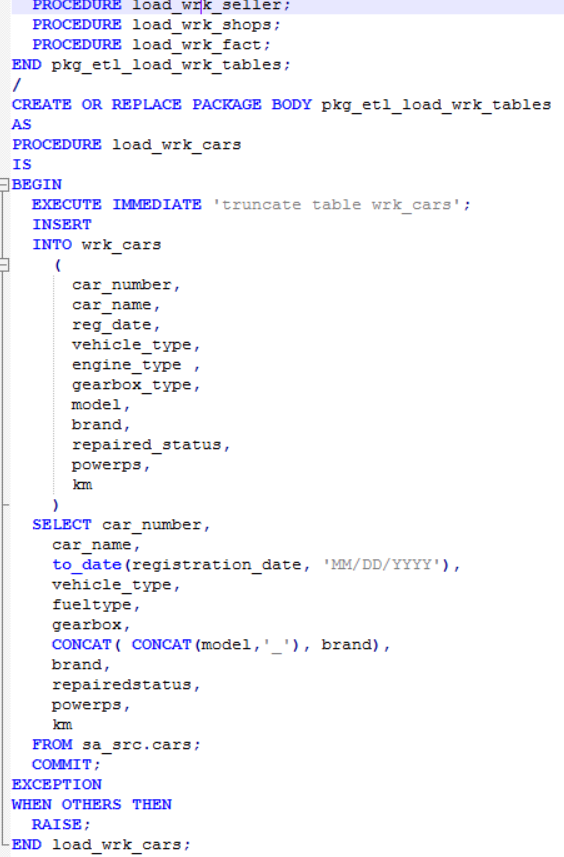
On BL\_CL layer there are two types of tables: wrk and cls.

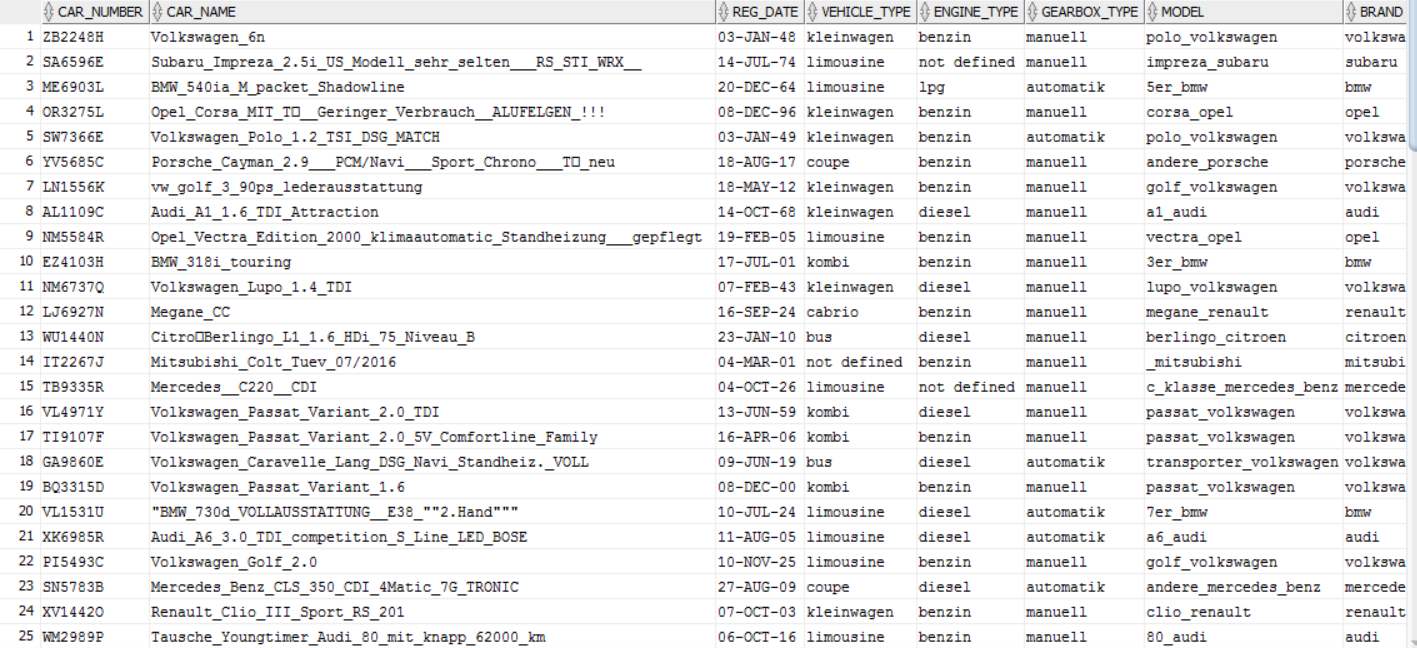
WRK tables were needed for loading external tables because external tables need a lot of space and work worse. Because of that we made a WRK tables which had the same structure as external tables, but their performance higher.

CLS tables were made for 3nf scheme, because they have the same structure as tables from 3nf, and they were made for cleansing some unsuitable information and divided some information for using in 3nf.



For loading wrk tables I used script below





Also in this tables I used only the columns, which I need. So this scrips become a little bit more complicated. Moreover, I made such stuff, like -99, -98 and not defined for null values. So on this layer I have already replace nulls.

On CLS Layer I divide WRK tables into tables that I will have on 3nf layer. Also I got rid of repetitions and generate some date columns where needed.

For loading data from wrk to cls I used only implicit cursor.

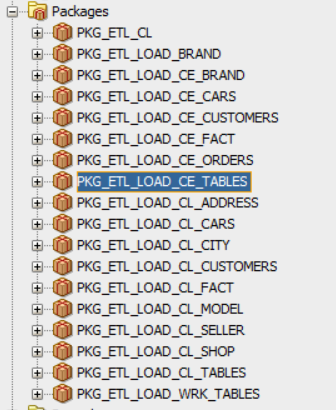


The script above was used for loading data from CLS to CE layer. Here I used merge for defining whether we already has this information or its updating. Here I also have date of last update.

Moreover on this layer I generated surr key which is primary key in CE for linking tables.

For loading data from cls to ce I used implicit and explicit cursors.

Packages, which were used for inserting data into ce are below:



Here is the script for explicit cursor:

CREATE OR REPLACE PACKAGE pkg\_etl\_load\_ce\_brand

AS

PROCEDURE load\_ce\_brand;

END pkg\_etl\_load\_ce\_brand;

/

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_load\_ce\_brand

AS

PROCEDURE load\_ce\_brand

IS

CURSOR c\_data

IS

SELECT DISTINCT cebr.rowid AS rid,

clbr.brand\_name,

trunc(sysdate) update\_dt

FROM cl\_brand clbr

LEFT OUTER JOIN bl\_3nf.ce\_brand cebr

ON clbr.brand\_name <> cebr.brand\_name;

type t\_\_data

IS

TABLE OF c\_data%rowtype INDEX BY binary\_integer;

t\_data t\_\_data;

BEGIN

EXECUTE IMMEDIATE 'truncate table bl\_3nf.ce\_brand';

OPEN c\_data;

LOOP

FETCH c\_data bulk collect INTO t\_data ;

EXIT

WHEN t\_data.count = 0;

FOR idx IN t\_data.first .. t\_data.last

LOOP

IF t\_data(idx).rid IS NULL THEN

INSERT

INTO bl\_3nf.ce\_brand

(

brand\_id,

brand\_name,

update\_dt

)

VALUES

(

bl\_3nf.seq\_brand.nextval,

t\_data(idx).brand\_name,

sysdate

);

ELSE

UPDATE bl\_3nf.ce\_brand

SET update\_dt = sysdate

WHERE rowid = t\_data(idx).rid ;

END IF;

END LOOP;

END LOOP;

CLOSE c\_data;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

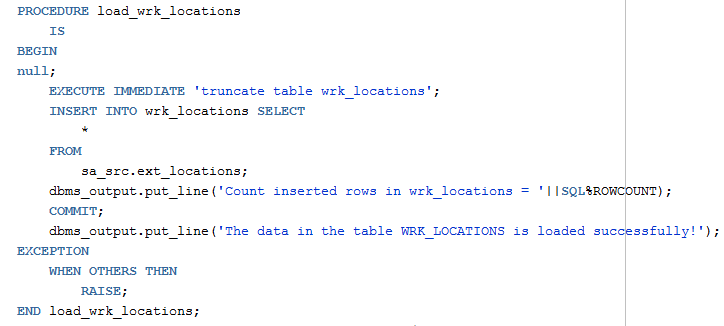
RAISE;

END load\_ce\_brand;

END pkg\_etl\_load\_ce\_brand;

/

Implicit cursor are used while INSERT, UPDATE or SELECT INTO. You don’t need either declare them or open



Here is the script for loading cars table from cls to ce.

PROCEDURE load\_ce\_cars

IS

BEGIN

EXECUTE IMMEDIATE 'truncate table bl\_3nf.ce\_cars';

MERGE INTO bl\_3nf.ce\_cars cc

USING ( SELECT car\_number, car\_name, vehicle\_type\_id, engine\_type\_id,

gearbox\_type\_id, model\_id, repaired\_status\_id, start\_dt, end\_dt FROM cl\_cars ) clc

ON (cc.car\_number = clc.car\_number and cc.start\_dt = clc.start\_dt and

cc.end\_dt = clc.end\_dt )

WHEN MATCHED THEN

UPDATE SET cc.car\_name=clc.car\_name ,

cc.vehicle\_type\_id=clc.vehicle\_type\_id,cc.engine\_type\_id = clc.engine\_type\_id ,

cc.gearbox\_type\_id= clc.gearbox\_type\_id ,cc.model\_id= clc.model\_id ,

cc.repaired\_status\_id=clc.repaired\_status\_id

WHERE DECODE(cc.car\_name,clc.car\_name,0,1)

+DECODE(cc.vehicle\_type\_id,clc.vehicle\_type\_id,0,1)

+DECODE(cc.engine\_type\_id, clc.engine\_type\_id,0,1)+DECODE(cc.gearbox\_type\_id,clc.gearbox\_type\_id,0,1)

+DECODE(cc.model\_id,clc.model\_id,0,1)+DECODE(cc.repaired\_status\_id,clc.repaired\_status\_id,0,1)>0

WHEN NOT MATCHED THEN

INSERT (cc.car\_id, cc.car\_number, cc.car\_name, cc.vehicle\_type\_id, cc.engine\_type\_id,

cc.gearbox\_type\_id, cc.model\_id, cc.repaired\_status\_id, cc.start\_dt, cc.end\_dt)

values (bl\_3nf.seq\_cars.nextval, clc.car\_number, clc.car\_name, clc.vehicle\_type\_id, clc.engine\_type\_id,

clc.gearbox\_type\_id, clc.model\_id, clc.repaired\_status\_id, clc.start\_dt, clc.end\_dt) ;

commit;

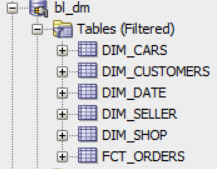
EXCEPTION

WHEN OTHERS THEN

RAISE;

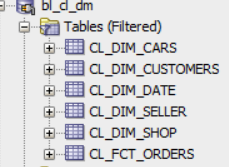
END load\_ce\_cars;

On BL\_DM layer there are such tables:



For each dim table we created sequence for the primary key, which were used for linked tables. With it we gave grants to cl\_dm layer for select sequence and select, update, insert and delete on table.

On BL\_CL\_DM layer there are such tables:



Also were used packages.

Bulk collect

CREATE OR REPLACE PACKAGE pkg\_etl\_load\_ce\_brand

AS

PROCEDURE load\_ce\_brand;

END pkg\_etl\_load\_ce\_brand;

/

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_load\_ce\_brand

AS

PROCEDURE load\_ce\_brand

IS

CURSOR c\_data

IS

SELECT DISTINCT cebr.rowid AS rid,

clbr.brand\_name,

trunc(sysdate) update\_dt

FROM cl\_brand clbr

LEFT OUTER JOIN bl\_3nf.ce\_brand cebr

ON clbr.brand\_name <> cebr.brand\_name;

type t\_\_datafa

IS

TABLE OF c\_data%rowtype INDEX BY binary\_integer;

t\_data t\_\_data;

BEGIN

EXECUTE IMMEDIATE 'truncate table bl\_3nf.ce\_brand';

OPEN c\_data;

LOOP

FETCH c\_data bulk collect INTO t\_data ;

EXIT

WHEN t\_data.count = 0;

FOR idx IN t\_data.first .. t\_data.last

LOOP

IF t\_data(idx).rid IS NULL THEN

INSERT

INTO bl\_3nf.ce\_brand

(

brand\_id,

brand\_name,

update\_dt

)

VALUES

(

bl\_3nf.seq\_brand.nextval,

t\_data(idx).brand\_name,

sysdate

);

ELSE

UPDATE bl\_3nf.ce\_brand

SET update\_dt = sysdate

WHERE rowid = t\_data(idx).rid ;

END IF;

END LOOP;

END LOOP;

CLOSE c\_data;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_ce\_brand;

END pkg\_etl\_load\_ce\_brand;

/

FORALL

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_load\_cl\_tables

AS

PROCEDURE load\_cl\_dim\_cars

IS

BEGIN

DECLARE

CURSOR c\_data IS

SELECT c.car\_id,

c.car\_number,

c.car\_name,

vt.vehicle\_type\_name,

et.engine\_type\_name,

gt.gearbox\_type\_name,

m.model\_name,

rs.repaired\_status\_name,

c.start\_dt,

c.end\_dt

FROM bl\_3nf.ce\_cars c

left join bl\_3nf.ce\_vehicle\_type vt on c.vehicle\_type\_id = vt.vehicle\_type\_id

left join bl\_3nf.ce\_engine\_type et on c.engine\_type\_id=et.engine\_type\_id

left join bl\_3nf.ce\_gearbox\_type gt on c.gearbox\_type\_id = gt.gearbox\_type\_id

left join bl\_3nf.ce\_model m on c.model\_id=m.model\_id

left join bl\_3nf.ce\_repaired\_status rs on c.repaired\_status\_id = rs.repaired\_status\_id;

TYPE fetch\_array IS TABLE OF c\_data%ROWTYPE ;

s\_array fetch\_array;

BEGIN

OPEN c\_data;

LOOP

FETCH c\_data BULK COLLECT INTO s\_array;

FORALL i IN 1..s\_array.COUNT

INSERT INTO fct\_sales

VALUES s\_array(i) ;

EXIT WHEN c\_data%NOTFOUND;

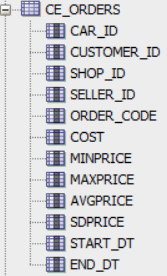
END LOOP;

CLOSE c\_data;

COMMIT;

END load\_cl\_dim\_cars;

For 3nf I have one fact table, which is there:



PROCEDURE load\_ce\_orders

IS

BEGIN

EXECUTE IMMEDIATE 'truncate table bl\_3nf.ce\_orders';

INSERT

INTO bl\_3nf.ce\_orders(

car\_id,

customer\_id,

shop\_id,

seller\_id,

order\_code,

cost,

minPrice,

maxPrice,

avgPrice,

sdPrice,

start\_dt,

end\_dt

)

SELECT

ROUND ( dbms\_random.value (

( SELECT MIN ( car\_id ) FROM bl\_3nf.ce\_cars ) ,

(SELECT MAX ( car\_id ) FROM bl\_3nf.ce\_cars)

)

) AS car\_id,

ROUND ( dbms\_random.value (

(SELECT MIN ( customer\_id ) FROM bl\_3nf.ce\_customers ) ,

(SELECT MAX ( customer\_id ) FROM bl\_3nf.ce\_customers )

)

) AS customer\_id,

ROUND ( dbms\_random.value (

(SELECT MIN ( shop\_id ) FROM bl\_3nf.ce\_shop ) ,

(SELECT MAX ( shop\_id ) FROM bl\_3nf.ce\_shop )

)

) AS shop\_id,

ROUND ( dbms\_random.value (

(SELECT MIN ( seller\_id ) FROM bl\_3nf.ce\_seller ) ,

(SELECT MAX ( seller\_id ) FROM bl\_3nf.ce\_seller)

)

) AS seller\_id,

clf.order\_code, clf.cost, clf.minPrice, clf.maxPrice,

clf.avgPrice, clf.sdPrice, clf.start\_dt, clf.end\_dt

FROM ( SELECT distinct order\_code, cost, minPrice, maxPrice,

avgPrice, sdPrice, start\_dt, end\_dt from cl\_fact)clf,

(select \* from dual connect by level <= 1000);

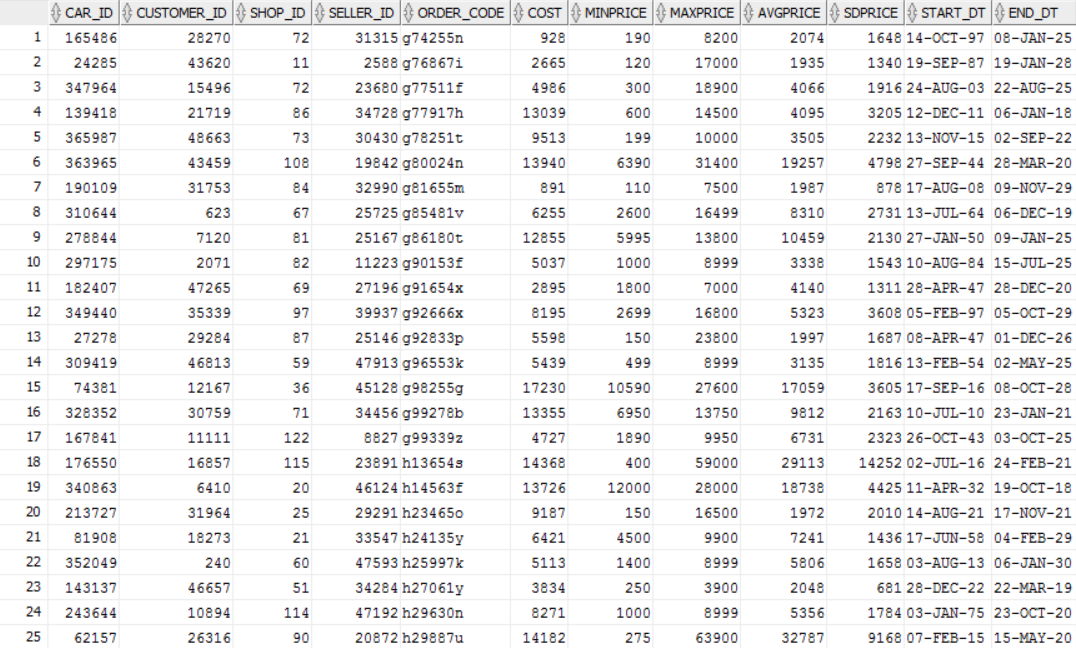
commit;

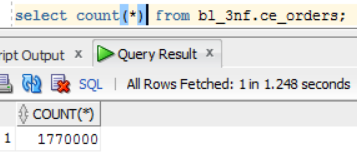
EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_ce\_orders;





Batch



| REVISION HISTORY | | | | | |
| --- | --- | --- | --- | --- | --- |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| n.1 | Added information about business description dimensions of a business, dimensional model star schema, 3nf schema. | Arina Marchenko | 15-Nov-2017 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |