

|  |
| --- |
| Хранилище данных  RELOUIS COMPANY |
| Relouis-Cosmetics |

Contents

[RELOUIS COMPANY 1](#_Toc500088295)

[1 Описание предметной области 7](#_Toc500088296)

[1.1 бизес основа 7](#_Toc500088297)

[1.2 недостатки существующих решений 7](#_Toc500088298)

[1.3 преимущества внедрения хранилища данных 7](#_Toc500088299)

[2 Бизнес измерения 8](#_Toc500088300)

[2.1 описание бизнес-процесса 8](#_Toc500088301)

[2.2 Определение зерна 8](#_Toc500088302)

[2.3 описание измерений 8](#_Toc500088303)

[2.4 Описание фактовой таблицы 11](#_Toc500088378)

[3 логическая схема 12](#_Toc500088390)

[3.1 схема слоя bl\_3nf: 12](#_Toc500088391)

[3.2 Схема слоя bl\_dm 13](#_Toc500088392)

[4 Объекты партиционирования 14](#_Toc500088393)

[5 Бизнес процессы 15](#_Toc500088394)

[6 Модель данных 16](#_Toc500088396)

[6.1 описание типов измерений 16](#_Toc500088397)

[6.2 Генерация данных 16](#_Toc500088398)

[6.3 пРОЦЕСС ФУНКЦИОНИРОВАНИЯ ХРАНИЛИЩА 18](#_Toc500088399)

[7 использование хранилище 25](#_Toc500088407)

[7.1 Пример функционирования хранилища 25](#_Toc500088408)

[7.2 deploy пакет 25](#_Toc500088409)

# Описание предметной области

## бизес основа

Бренд Relouis представляет собой декоративную косметику белорусского производства. Ассортиментный перечень содержит более 600 наименований и включает все группы декоративной косметики. На данный момент насчитывается 5 коллекций и 5 основных групп товара.

## недостатки существующих решений

Когда разрабатывалось первое хранилище, никто не думал, что компания за столько небольшой срок наберет такую широкую массу покупателей и так хорошо наладит бизнес, поэтому и возникла проблема в создании нового хранилища данных.

Трудности, которые возникли в связи с использованием старого хранилища данных:

* + Различные источники данных
  + Данные содержать в “грядном” виде, включаю отсутствие описания ключевых полей и т.п
  + Отсутствие истории продаж
  + Заказчик понес значительные убытки, т.к. некоторые из его решений были приняты на основе использования не консистентных данных в предыдущем хранилище

## преимущества внедрения хранилища данных

* Ускорится процесс обработки данных
* Улучшиться доступ к данным
* Наличие историчности данных
* Расширится круг разрезов, по которым менеджеры смогут проводить аналитику,

# Бизнес измерения

## описание бизнес-процесса

Учет продажи товаров для будущего анализа компании с использованием различных метрик.

## Определение зерна

Зерном будет являться продажа какого(их)–то товаров определенном покупателю, определенным сотрудником в конкретном магазине в определенную дату.

## описание измерений

Схема будет содержать следующие измерения:

| DIM\_Customers | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| customer\_id | NUMBER(10) | ПК,уникальный идентификатор для каждого покупателя, заполняется с помощью последовательностей (20222) |
| customer\_surr\_id | NUMBER(10) | Суррогатный ключ, который заполняется на уровне 3НФ. |
| last\_name | VARCHAR2(100 CHAR) | Фамилия покупателя (Макарец) |
| first\_name | VARCHAR2(100 CHAR) | Имя покупателя (Алина) |
| middle\_name | VARCHAR2(100 CHAR) | Отчество покупателя (Павловна) |
| phone | VARCHAR2(50 CHAR) | Телефон покупателя (+375294444444) |
| email | VARCHAR2(150 CHAR) | Почтовый адрес для рассылки с информацией о новинках/акциях (alinamakarets@tut.by) |
| age | NUMBER(10) | Возраст покупателя (20) |
| discount | NUMBER(10) | Персональная скидка для покупателя (5-имеется в виду 5%) |
| gender | VARCHAR2(50 CHAR) | Пол покупателя ( Ж ) |
| city\_id | NUMBER(10) | Идентификатор города покупателя ( 52 ) |
| city\_name | VARCHAR2(200 CHAR) | Название города покупателя ( Минск ) |
| region\_id | NUMBER(10) | Идентификатор региона покупателя ( 7 ) |
| region\_name | VARCHAR2(200 CHAR) | Название региона покупателя ( город Минск ) |
| Insert\_dt | DATE | Дата вставки строки |
| Update\_dt | DATE | Дата обновления строки |

| DIM\_Products | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| product\_id | NUMBER(10) | ПК,уникальный идентификатор для каждого продукта, заполняется с помощью последовательностей (236) |
| product\_surr\_id | NUMBER(10) | Суррогатный ключ, который заполняется на уровне 3НФ. |
| product\_name | VARCHAR2(200 CHAR) | Название продукта (Тушь с эффектом накладных ресниц) |
| product\_desc | VARCHAR2(300 CHAR) | Уникальный код продукта (456-12) |
| price | NUMBER(20) | Цена (12) |
| color\_id | NUMBER(10) | Код цвета (85) |
| color\_name | VARCHAR2(200 CHAR) | Название цвета (Черный) |
| collection \_id | NUMBER(10) | Код коллекции (1, *-999 для N/A)* |
| collection\_name | VARCHAR2(200 CHAR) | Название коллекции (Professional) |
| category\_id | NUMBER(10) | Код категории ( 4) |
| category\_name | VARCHAR2(200 CHAR) | Название категории (Глаза) |
| subcategory\_id | NUMBER(10) | Код подкатегории ( 4) |
| subcategory\_name | VARCHAR2(200 CHAR) | Название подкатегории (Тушь для глаз) |
| start\_dt | DATE | Дата старта продаж коллекции |
| end\_dt | DATE | Дата окончания продаж коллекции |
| is\_active | VARCHAR2(4 BYTE) | Показывает актуальность продаж по данной коллекции |

| DIM\_Shops | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| shop\_id | NUMBER(10) | ПК, код магазина (101) |
| shop\_surr\_id | NUMBER(10) | Суррогатный ключ, который заполняется на уровне 3НФ. |
| shop\_name | VARCHAR2(40 BYTE) | Название магазина (Фирменный Магазин Relouis) |
| phone | VARCHAR2(40 BYTE) | Телефон магазина (+375172569849) |
| address | NUMBER(10) | Развернутый адрес магазина (2092) |
| city\_id | NUMBER(10) | Идентификатор города магазина ( 52 ) |
| city\_name | VARCHAR2(200 CHAR) | Название города магазина ( Минск ) |
| region\_id | NUMBER(10) | Идентификатор региона магазина ( 7 ) |
| region\_name | VARCHAR2(200 CHAR) | Название региона магазина ( город Минск ) |
| manager\_id | NUMBER(10) | Идентификатор работника, который ответственный за магазин (2223) |
| manager\_last\_name | VARCHAR2(100 CHAR) | Фамилия менеджера (Макарец) |
| manager first\_name | VARCHAR2(100 CHAR) | Имя менеджера (Алина) |
| manager middle\_name | VARCHAR2(100 CHAR) | Отчество менеджера (Павловна) |
| manager\_gender | VARCHAR2(50 CHAR) | Пол менеджера |
| Insert\_dt | DATE | Дата вставки строки |
| Update\_dt | DATE | Дата обновления строки |

| DIM\_Employees | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| employee\_id | NUMBER(10) | ПК,Уникальный идентификатор для работника (2237) |
| employee\_surr\_id | NUMBER(10) | Суррогатный ключ, который заполняется на уровне 3НФ. |
| last\_name | VARCHAR2(100 CHAR) | Фамилия работника (Макарец) |
| first\_name | VARCHAR2(100 CHAR) | Имя работника (Алина) |
| middle\_name | VARCHAR2(100 CHAR) | Отчество работника (Павловна) |
| phone | VARCHAR2(50 CHAR) | Телефон работника (+375294444444) |
| email | VARCHAR2(150 CHAR) | Почтовый адрес работника для рассылки информации, связанной с работой (alinamakarets@tut.by) |
| age | NUMBER(10) | Возраст работника (50) |
| gender | VARCHAR2(50 CHAR) | Пол работника ( Ж ) |
| passport | VARCHAR2(40 CHAR) | Пасспорт работника (МР5236623) |
| Start\_dt | DATE | Дата принятия на работу |
| End\_dt | DATE | Дата окончания работы на нанимателя |
| Is\_current | VARCHAR2(10 CHAR) | Показывает актуальность работы на данный момент ( работает такой человек или нет) |

| DIM\_Time\_Day | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| date\_dt | DATE | ПК, Дата (04-01-2017) |
| week\_name | VARCHAR2(25 CHAR) | Название дня недели ( Понедельник ) |
| week\_number\_of\_month | NUMBER(10) | Номер недели в месяце ( 2 ) |
| week\_number\_of\_year | NUMBER(10) | Номер недели в году ( 35 ) |
| day\_number\_of\_week | NUMBER(10) | Числовой день недели (2 ) |
| day\_number\_of\_month | NUMBER(10) | День в месяце ( 25 ) |
| day\_number\_of\_year | NUMBER(10) | День в году (320 ) |
| month\_year | VARCHAR2(25 CHAR) | Значение месяца и года (11-2017) |
| month\_number | NUMBER(10) | День месяца в году (4) |
| month\_name | VARCHAR2(25 CHAR) | Название месяца ( Апрель ) |
| quarter | NUMBER(10) | Число квартала (1) |
| year\_quater | VARCHAR2(25 BYTE) | Значение квартала и года ( 2-2017) |
| Year | NUMBER(10) | Год (2017) |

## Описание фактовой таблицы

Таблица фактов будет выглядеть следующим образом:

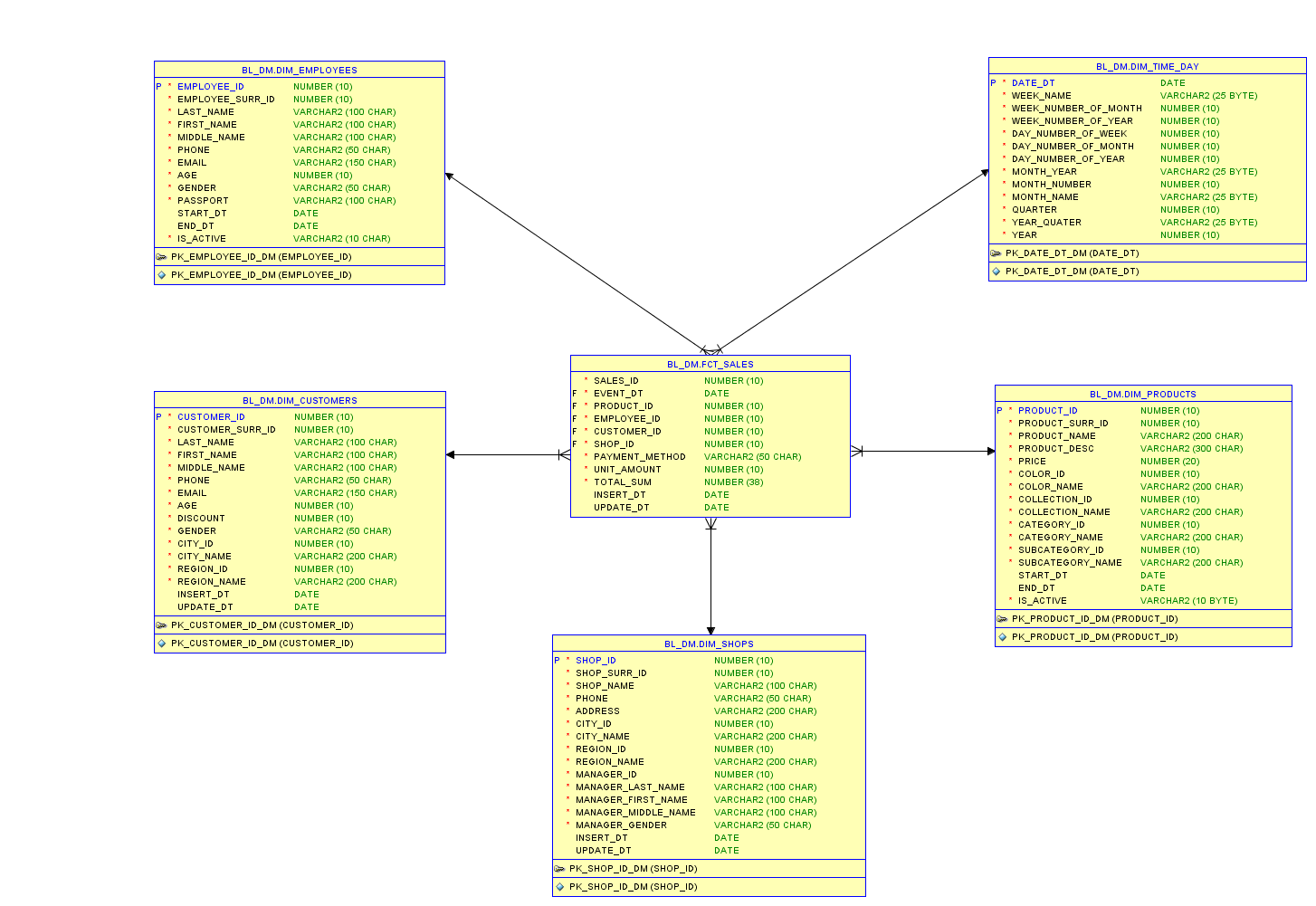
| FCT\_Sales | | |
| --- | --- | --- |
| Название колонки | Тип колонки | Описание (Пример) |
| sales\_id | NUMBER(10) | Суррогатный ключ, который заполняется на уровне 3НФ. |
| event\_dt | DATE | ВК к таблице с датой (04-01-2017) |
| product\_id | NUMBER(10) | ВК к таблице с продуктами (203) |
| employee\_id | NUMBER(10) | ВК к таблице с сотрудниками (2237) |
| customer\_id | NUMBER(10) | ВК к таблице с покупателями (20222) |
| shop\_id | NUMBER(10) | ВК к таблица с магазинами (101) |
| payment\_method | VARCHAR2(50 CHAR) | Тип Платежа ( Наличные / Карта ) |
| unit\_amount | NUMBER(10) | Количество продуктов (2) |
| total\_sum | NUMBER(38) | Общая сумма (20) |
| insert\_dt | DATE | Дата вставки строки |
| update\_dt | DATE | Дата обновления строки |

# логическая схема

## D:\Alina_Makarets\DWH\3nf.pngсхема слоя bl\_3nf:

Picture bl\_3nf

## Схема слоя bl\_dm

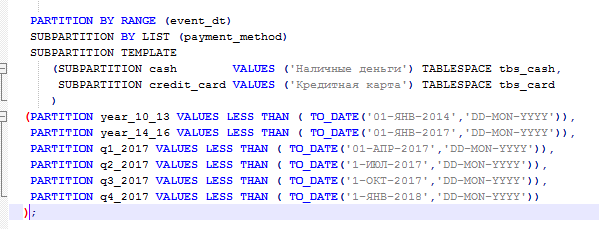


В данном хранилище использовалась схема типа «Звезда». Она подразумевает минимальное количество (а лучше их отсутствие) соединения таблиц при обработке запросов и позволяет ответить практически на все поставленные вопросы, к примеру, для построения отчетов по продажам.

# Объекты партиционирования

Конечная таблица фактов была разбита на partition по годам, которые в свою очередь были разбиты на subpartition по типу оплаты (одно из требований заказчика, иметь возможность проводить анализ по данному критерию). Ниже приведены partition и его реализация:

* + 2010-2013
    - Тип оплаты ( Наличные деньги / Кредитная карта )
  + 2014-2016
    - Тип оплаты ( Наличные деньги / Кредитная карта )
  + 1-й квартал 2017
    - Тип оплаты ( Наличные деньги / Кредитная карта )
  + 2-й квартал 2017
    - Тип оплаты ( Наличные деньги / Кредитная карта )
  + 3-й квартал 2017
    - Тип оплаты ( Наличные деньги / Кредитная карта )
  + 4-й квартал 2017
    - Тип оплаты ( Наличные деньги / Кредитная карта )



# Бизнес процессы

Пример отчета, который построен на данных нашего хранилища. Здесь исследуется общая сумма продаж по регионам и городам.

# 

Такие же отчеты могут быть построены и для определенного года в разрезе всех категорий или коллекций.

Для построенных выше отчетов использовался Tibco Sportfire.

# Модель данных

## описание типов измерений

В данном хранилище данных использовано 5 измерений, 2 из которых имеют тип SCD2 type с атрибутами “Start\_dt”, “End\_dt”, “Is\_current”. В данном хранилище данных использовано измерение календарного типа DIM\_TIME\_DAY, позволяющее нам делать анализ продаж в разрезе лет, месяцев и даже дней.

## Генерация данных

Данные для данного хранилища были преимущественно сгенерированы вручную. Информация о продуктах, категориях продуктов, описание, цвета и т.д. была взята из официального сайта компании

<http://relouis.by/product_category/tush-dlya-resnits/>



Данные о покупателях ( ФИО) были взяты на сайте <https://igra.evroopt.by/archive/>. С помощью excel были сгенерирован возраст, личная скидка покупателя. С помощью сайта <http://roman-dushkin.narod.ru/generator_02_em.html>. Города и регионы для покупателей были сделаны вручную.

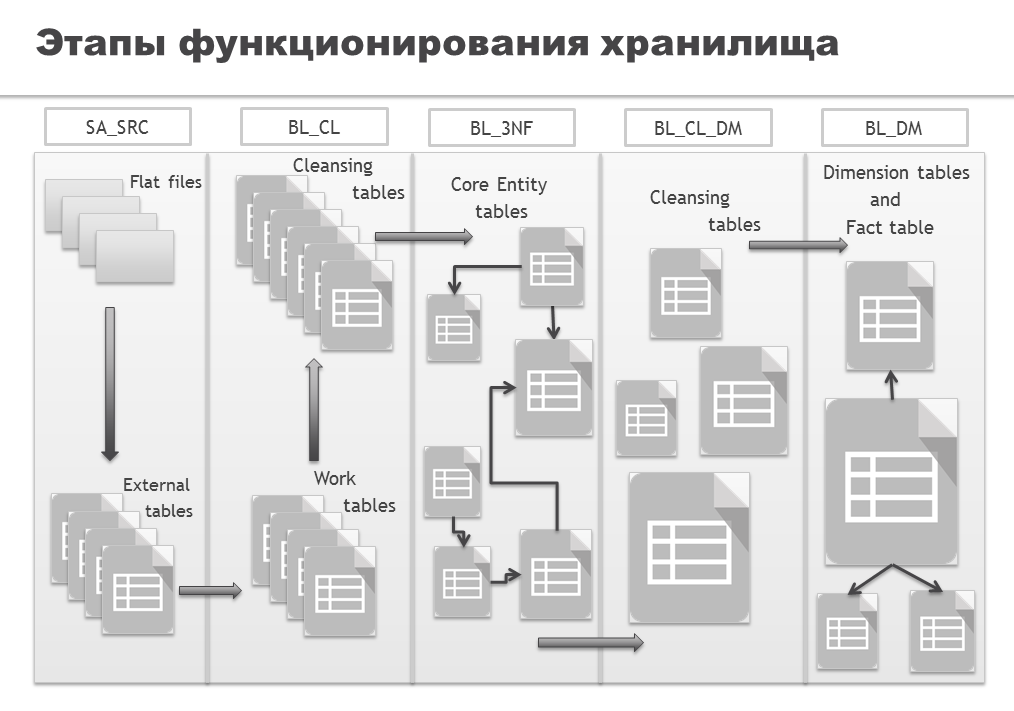
Данные о магазинах были взяты с официального сайта компании Relouis. Продукты данного бренда продаются в сетях магазинов «Мила» и «Остров Чистоты». Соответственно с сайтов компаний <http://ostrov-chistoty.by/> и <http://mila.by/> были взяты информация о расположении магазинов.

Данные о сотрудников были взяты частично из информации о покупателях, т.к. изначальный файл состоял из более 200 000 человек.

Генерация информации о времени была произведена с помощью скрипты на уровне BL\_CL.

Генерация таблицы фактов, так же была произведена с помощью скрипта на уровне BL\_CL, при этом была использована информация о цене продукта и личной скидки покупателя из уровня BL\_3NF.

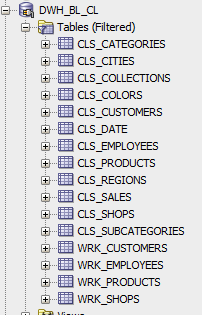
## пРОЦЕСС ФУНКЦИОНИРОВАНИЯ ХРАНИЛИЩА



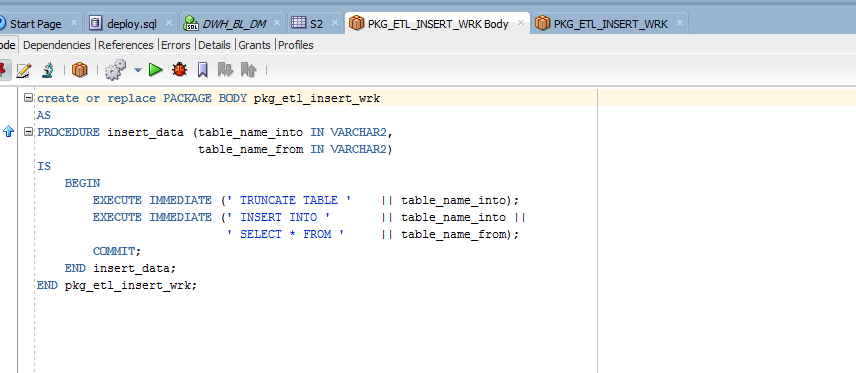
Picture 3 Data\_flow

Для функционирования хранилища данных было выделено 5 основных стадии:

1. SA\_SRC слой. На данном этапе у нас был набор csv файлов с данными. Для данного набора данных были сделаны external таблицы с такой же структурой данных, как и в файлах.
2. BL\_CL слой подразумевает собой наличие двух типов таблиц wrk и cls. Таблиц типов wrk было столько же, сколько и external таблиц, повторяющие полностью их структуру.



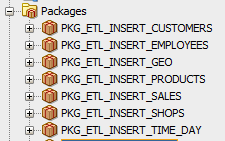
Для перезаливки данных в данные таблицы был написан один пакет, который использовался для всех таблиц, просто заменяя имя.



Таблицы типа cls повторяли структуру таблиц, находящихся на следующем слое. При вставке данных в cls таблицы была использована очистка данных:

* Определения пола покупателя и работника
* Сопоставления номера региона и его названию ( 7 – город Минск)
* Определен уникальный ключ для продукта при помощи конкатенации кода продукта и его цвета

Пакеты, которые были использованы для вставки данных на уровени 3NF.



* 1. Использование Explicit Cursor

Явный курсор - это курсор, который нужно создать явно, а затем управлять им. Нужно объявить и определить явный курсор, указать ему имя и связать его с запросом.

----------------------------------------------------

PROCEDURE insert\_table\_employees

IS

CURSOR emp\_cursor

IS

SELECT TO\_NUMBER(employee\_code) as employee\_id,

first\_name,

last\_name,

middle\_name,

phone,

email,

age,

CASE

WHEN substr (middle\_name,-1) = 'а'

THEN 'Ж'

ELSE 'М'

END AS gender,

passport\_code || passport\_letter as passport,

NVL(to\_date(start\_date, 'DD-MM-YYYY'),'01-ЯНВ-1990') as start\_dt,

NVL(to\_date(end\_date, 'DD-MM-YYYY'),'31-ДЕК-9999') as end\_dt,

CASE

WHEN active = 1

THEN 'Y'

WHEN active = 0

THEN 'N'

END as is\_active

FROM wrk\_employees

WHERE first\_name IS NOT NULL

AND last\_name IS NOT NULL

AND middle\_name IS NOT NULL

AND email IS NOT NULL;

rt\_emp\_cursor cls\_employees%rowtype;

BEGIN

EXECUTE IMMEDIATE 'TRUNCATE TABLE cls\_employees';

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO rt\_emp\_cursor;

IF emp\_cursor%found THEN

INSERT INTO cls\_employees

(

employee\_id,

last\_name,

first\_name,

middle\_name,

phone,

email,

age,

gender,

passport,

start\_dt,

end\_dt,

is\_active

)

VALUES

(

rt\_emp\_cursor.employee\_id,

rt\_emp\_cursor.last\_name,

rt\_emp\_cursor.first\_name,

rt\_emp\_cursor.middle\_name,

rt\_emp\_cursor.phone,

rt\_emp\_cursor.email,

rt\_emp\_cursor.age,

rt\_emp\_cursor.gender,

rt\_emp\_cursor.passport,

rt\_emp\_cursor.start\_dt,

rt\_emp\_cursor.end\_dt,

rt\_emp\_cursor.is\_active

)

;

END IF;

EXIT WHEN emp\_cursor%notfound;

END LOOP;

COMMIT;

CLOSE emp\_cursor;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END insert\_table\_employees;

1. Использование implicit курсора

Примерами неявных курсоров являются INSERT, UPDATE, или SELECT INTOНеявные курсоры не обязательно должны быть объявлены ( DECLARE) , открыты ( OPEN ), использование FETCH. Неявные атрибуты курсора указываются с помощью курсора SQL ( SQL%ROWCOUNT, использующийся для подсчет записей в курсоре).

PROCEDURE insert\_table\_products

IS

BEGIN

DECLARE

CURSOR pro\_cursor IS

SELECT product\_code||'-'||color || '-'|| subcat.subcategory\_surr\_id AS product\_id,

product\_name,

NVL(product\_desc,wrk.subcategory\_name) as product\_desc,

to\_number ( price, '99999999.99') as price,

clr.color\_surr\_id,

col.collection\_surr\_id,

cat.category\_surr\_id,

subcat.subcategory\_surr\_id,

to\_date(wrk.start\_date,'DD-MM-YYYY') as start\_dt

FROM wrk\_products wrk JOIN ce\_colors clr ON wrk.color=clr.color\_id

JOIN ce\_categories cat ON wrk.category\_name=cat.category\_name

JOIN ce\_subcategories subcat ON wrk.subcategory\_name=subcat.subcategory\_name

JOIN ce\_collections col ON wrk.collection\_name=col.collection\_name;

BEGIN

EXECUTE IMMEDIATE (' TRUNCATE TABLE cls\_products');

FOR prod\_cursor\_val IN pro\_cursor LOOP

INSERT INTO cls\_products (

product\_id,

product\_name,

product\_desc,

price,

color\_id,

collection\_id,

category\_id,

subcategory\_id,

start\_dt,

end\_dt,

is\_active

)

VALUES ( prod\_cursor\_val.product\_id,

prod\_cursor\_val.product\_name,

prod\_cursor\_val.product\_desc,

prod\_cursor\_val.price,

prod\_cursor\_val.color\_surr\_id,

prod\_cursor\_val.collection\_surr\_id,

prod\_cursor\_val.category\_surr\_id,

prod\_cursor\_val.subcategory\_surr\_id,

prod\_cursor\_val.start\_dt,

TO\_DATE('31-ДЕК-9999','DD-MON-YYYY'),

'Y'

);

END LOOP;

COMMIT;

END;

EXCEPTION

WHEN OTHERS

THEN RAISE;

END insert\_table\_products;

1. Использование MERGE

Практически во всех пакетах был использован merge, но как показала практика данный способ не сильно эффективен на больших объемах данных при использовании UPDATE statement, т.к. данная процедура занимает много времени, но он прекрасно справляется при первой вставке данных в таблицы, находящихся на BL\_3NF слое

PROCEDURE merge\_table\_ce\_colors

IS

BEGIN

MERGE INTO ce\_colors t USING

( SELECT color\_id,

color\_name

FROM cls\_colors

MINUS

SELECT color\_id,

color\_name

FROM ce\_colors

) c

ON ( c.color\_id=t.color\_id )

WHEN matched THEN

UPDATE SET t.color\_name=c.color\_name

WHEN NOT matched THEN

INSERT

(color\_surr\_id,

color\_id,

color\_name,

insert\_dt,

update\_dt)

VALUES

(ce\_colors\_seq.nextval,

c.color\_id,

c.color\_name,

SYSDATE,

SYSDATE) ;

COMMIT ;

EXCEPTION

WHEN OTHERS

THEN RAISE;

END merge\_table\_ce\_colors;

На уровне BL\_CL также были произведены генерации данных для временной таблицы и таблицы фактов.

1. Пример генерации фактов:

PROCEDURE insert\_table\_sales

IS

BEGIN

EXECUTE IMMEDIATE ('TRUNCATE TABLE cls\_sales');

BEGIN

FOR i IN 1..1000000

LOOP

dbms\_random.seed

(

i \* 5

)

;

INSERT INTO cls\_sales

( sales\_number,

event\_dt,

product\_id,

employee\_id,

customer\_id,

shop\_id,

payment\_method,

unit\_amount,

total\_sum

)

SELECT

sal.sales\_number,

TRUNC(sal.event\_dt),

sal.product\_id,

sal.employee\_id,

sal.customer\_id,

sal.shop\_id,

sal.payment\_method,

sal.unit\_amount,

ce.price\*unit\_amount\*(1-(cus.discount/100)) as total\_sum

FROM

(

SELECT

trunc( dbms\_random.value( 100000000, 999999999 ) ) as sales\_number,

to\_date( trunc (

dbms\_random.value (

( SELECT to\_char( MIN ( date\_dt ), 'J' ) FROM ce\_date ),

( SELECT to\_char( MAX ( date\_dt ), 'J' ) FROM ce\_date )

)

), 'J' ) as event\_dt,

round ( dbms\_random.value (

( SELECT MIN ( product\_surr\_id ) FROM ce\_products ) ,

( SELECT MAX ( product\_surr\_id ) FROM ce\_products )

)

) as product\_id,

round ( dbms\_random.value (

( SELECT MIN ( employee\_surr\_id ) FROM ce\_employees ) ,

( SELECT MAX ( employee\_surr\_id ) FROM ce\_employees )

)

) as employee\_id,

round ( dbms\_random.value (

( SELECT MIN ( customer\_surr\_id ) FROM ce\_customers ) ,

( SELECT MAX ( customer\_surr\_id ) FROM ce\_customers )

)

) as customer\_id,

round ( dbms\_random.value (

( SELECT MIN ( shop\_surr\_id ) FROM ce\_shops ) ,

( SELECT MAX ( shop\_surr\_id ) FROM ce\_shops )

)

) as shop\_id,

(SELECT CASE

WHEN (SELECT

abs(trunc(dbms\_random.normal))

FROM dual)=1

THEN 'Наличные деньги'

WHEN (SELECT

abs(trunc(dbms\_random.normal))

FROM dual)=0

THEN 'Кредитная карта'

ELSE 'Наличные деньги'

END

FROM dual) as payment\_method,

round ( dbms\_random.value ( 1 , 10 ) ) as unit\_amount

FROM dual)sal

JOIN ce\_products ce ON sal.product\_id=ce.product\_surr\_id

JOIN ce\_customers cus ON sal.customer\_id=cus.customer\_surr\_id;

END LOOP;

END;

COMMIT;

EXCEPTION

WHEN OTHERS

THEN RAISE;

END insert\_table\_sales;

1. BL\_3NF слой содержал в себе таблицы, которые находились в нормализованном виде. Процесс вставки данных в таблицы происходил на предыдущем уровне, поэтому заранее были предусмотрены соответствующие гранты для пользователей, а также были созданы заранее последовательности и на их использование так же были выделены гранты.
2. BL\_CL\_DM слой должен иметь доступ сразу к двум слоям (BL\_3NF и BL\_DM). На данном слое никакой очистки не производилось. Был произведен процесс mapping и соединения нужных таблиц с данными для вставки необходимой информации на следующий слой. Для вставки данных в соответствующие таблицы были использованы BULK COLLECT и FORALL
3. Пример использования BULK COLLECT

PROCEDURE insert\_table\_products

IS

BEGIN

DECLARE

CURSOR c\_data

IS

SELECT DISTINCT product\_surr\_id as product\_id,

product\_name,

product\_desc,

price,

col.color\_surr\_id,

col.color\_name,

coll.collection\_surr\_id,

coll.collection\_name,

cat.category\_surr\_id,

cat.category\_name,

subcat.subcategory\_surr\_id,

subcat.subcategory\_name,

start\_dt,

end\_dt,

is\_active

FROM ce\_products prod LEFT JOIN ce\_colors col ON prod.color\_surr\_id=col.color\_surr\_id

LEFT JOIN ce\_collections coll ON prod.collection\_surr\_id=coll.collection\_surr\_id

LEFT JOIN ce\_subcategories subcat ON prod.subcategory\_surr\_id=subcat.subcategory\_surr\_id

LEFT JOIN ce\_categories cat ON subcat.category\_surr\_id=cat.category\_surr\_id;

type t\_\_data

IS

TABLE OF c\_data%rowtype;

t\_data t\_\_data;

BEGIN

EXECUTE IMMEDIATE ('TRUNCATE TABLE cls\_dim\_products');

OPEN c\_data;

LOOP

FETCH c\_data bulk collect INTO t\_data;

EXIT WHEN t\_data.count = 0;

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

LOOP

INSERT INTO cls\_dim\_products

VALUES

(

t\_data(i).product\_id,

t\_data(i).product\_name,

t\_data(i).product\_desc,

t\_data(i).price,

t\_data(i).color\_surr\_id,

t\_data(i).color\_name,

t\_data(i).collection\_surr\_id,

t\_data(i).collection\_name,

t\_data(i).category\_surr\_id,

t\_data(i).category\_name,

t\_data(i).subcategory\_surr\_id,

t\_data(i).subcategory\_name,

t\_data(i).start\_dt,

t\_data(i).end\_dt,

t\_data(i).is\_active

);

END LOOP;

END LOOP;

CLOSE c\_data;

END;

END insert\_table\_products;

1. Пример использования FORALL

PROCEDURE insert\_table\_dim\_fct\_table

IS

BEGIN

DECLARE

CURSOR c\_data IS

SELECT ce.sales\_surr\_id,

ce.event\_dt,

dim.product\_id,

emp.employee\_id,

cus.customer\_id,

dshop.shop\_id ,

ce.payment\_method,

ce.unit\_amount,

ce.total\_sum,

SYSDATE as insert\_dt,

SYSDATE as update\_dt

FROM ce\_sales ce LEFT JOIN dim\_products dim ON ce.product\_surr\_id=Dim.Product\_surr\_Id

LEFT JOIN dim\_shops dshop ON ce.shop\_surr\_id=dshop.shop\_surr\_id

LEFT JOIN dim\_employees emp ON ce.employee\_surr\_id=emp.employee\_surr\_id

LEFT JOIN dim\_customers cus ON ce.customer\_surr\_id=cus.customer\_surr\_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;

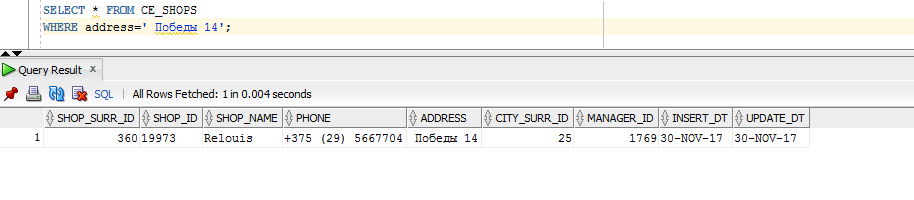
END insert\_table\_dim\_fct\_table;

1. BL\_DM слой является конечным слоем нашего хранилища данных. На данном слое можно создавать различные витрины данных для построения необходимых отчетов.

# использование хранилище

## Пример функционирования хранилища

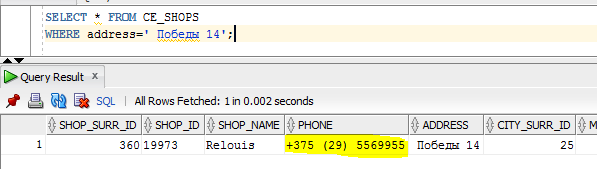
Изначально в файлах с данными был магазин с определенным телефоном:



Потом вручную был изменен телефон на уровне файла:



Заново запустив все пакеты, проверили работает ли наше хранилище:



Результат: хранилище хорошо функционирует .

## deploy пакет

Для ускорения проверки работоспособности проверки был создан batch файл, который позволяет с помощью F5 создать абсолютно все, начиная от пользователей, заканчивая заполнения данными на финальном уровне. Ниже приведен кусочек кода:

D:\dwh\system\tablespace.sql

@D:\dwh\system\sa\_src\sa\_src\_user.sql

@D:\dwh\system\sa\_src\sa\_src\_grants.sql

@D:\dwh\system\bl\_cl\bl\_cl\_user.sql

@D:\dwh\system\bl\_cl\bl\_cl\_grants.sql

@D:\dwh\system\bl\_3nf\bl\_3nf\_user.sql

@D:\dwh\system\bl\_3nf\bl\_3nf\_grants.sql

@D:\dwh\system\bl\_cl\_dm\bl\_cl\_dm\_user.sql

@D:\dwh\system\bl\_cl\_dm\bl\_cl\_dm\_grants.sql

@D:\dwh\system\bl\_dm\bl\_dm\_user.sql

@D:\dwh\system\bl\_dm\bl\_dm\_grants.sql

connect sa\_src/123;

show user;

@D:\dwh\sa\_src\create\_directory.sql

@D:\dwh\sa\_src\tables\ext\_customers.sql

@D:\dwh\sa\_src\tables\ext\_employees.sql

@D:\dwh\sa\_src\tables\ext\_products.sql

@D:\dwh\sa\_src\tables\ext\_shops.sql

@D:\dwh\sa\_src\create\_public\_synonyms.sql

@D:\dwh\sa\_src\grants\_to\_bl\_cl.sql

| REVISION HISTORY | | | | | |
| --- | --- | --- | --- | --- | --- |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 3.1 | Изменена тема шаблон | Alina Makarets | 16-NOV-2017 |  |  |
| 3.2 | Изменения структуры | Alina Makarets | 03-DEC-2017 |  |  |
|  |  |  |  |  |  |