**Лабораторная работа №4**

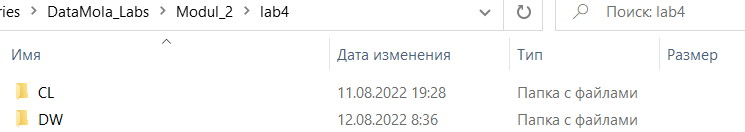
**Невейков Андрей, 2022**

**All scripts here:**

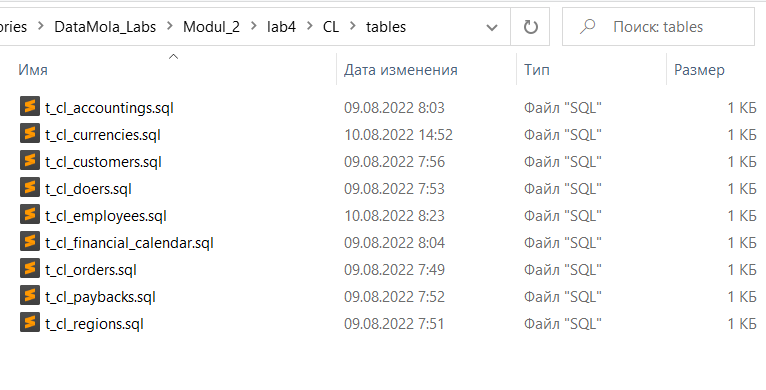
**https://github.com/AndreyNeveikov/DataMola\_Labs/tree/main/Modul\_2/lab4**

/\*Структура хранилища\*/

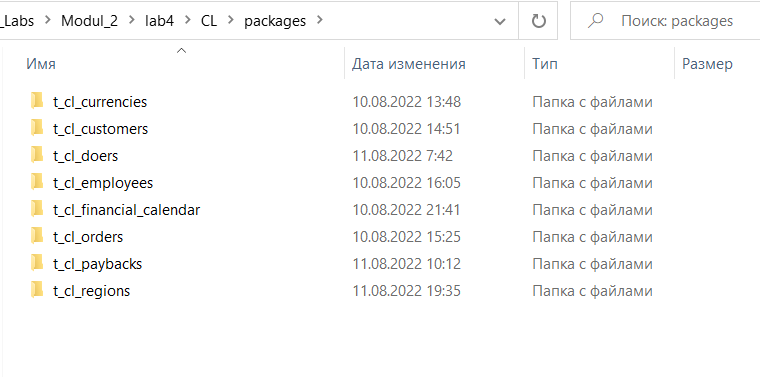
Лабораторная работа состоит из двух частей: на CL уровне и на DW уровне.



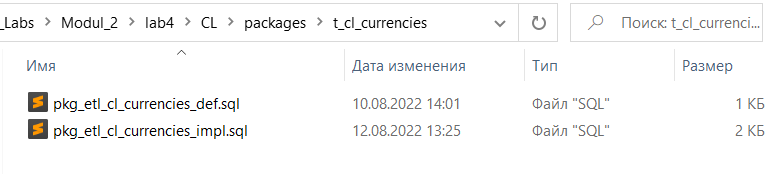
Созданы таблицы на соответствующем уровне.



Пакеты лежат в папках, соответствующих названию таблицы.



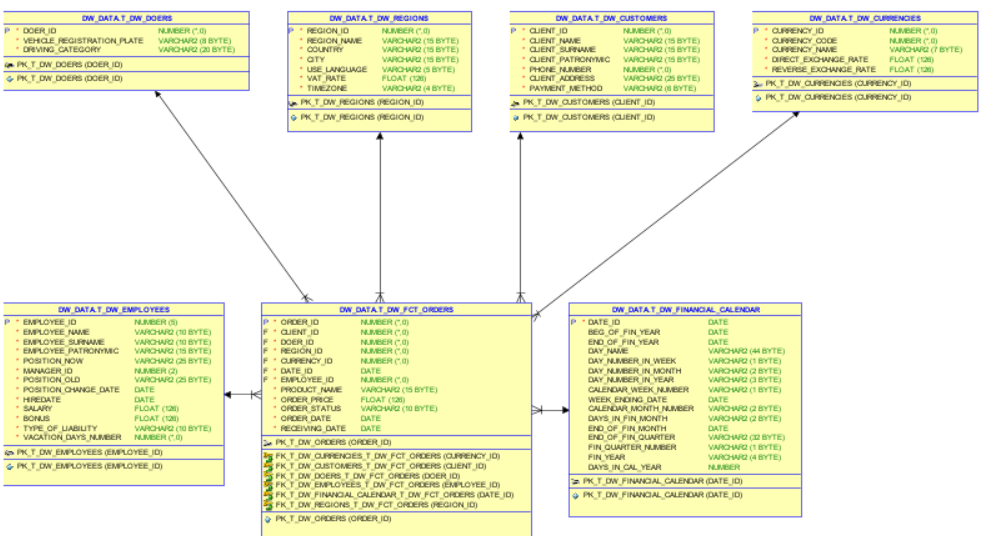
Каждый пакет состоит из def и impl частей. В def – объявление пакета, в impl – логика.



**/\*Task\_1\*/ Соответствие заданию**

Создать объекты на DW-уровне





Раздать привилегии на CL уровень

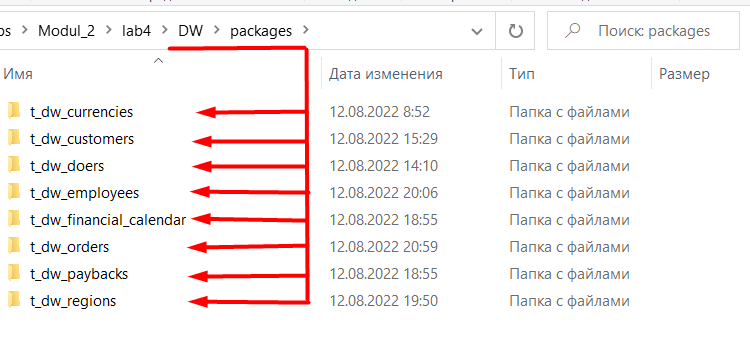


Вверху каждого пакета предоставляются привилегии на уровень ниже:

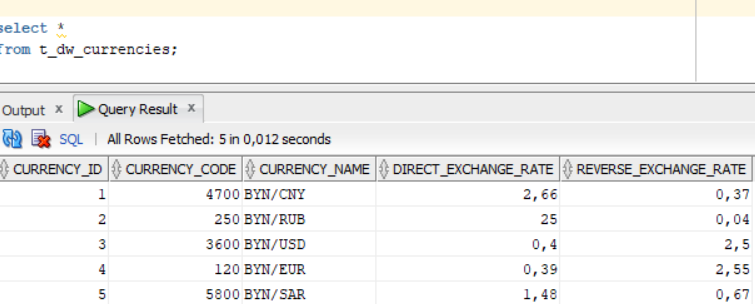


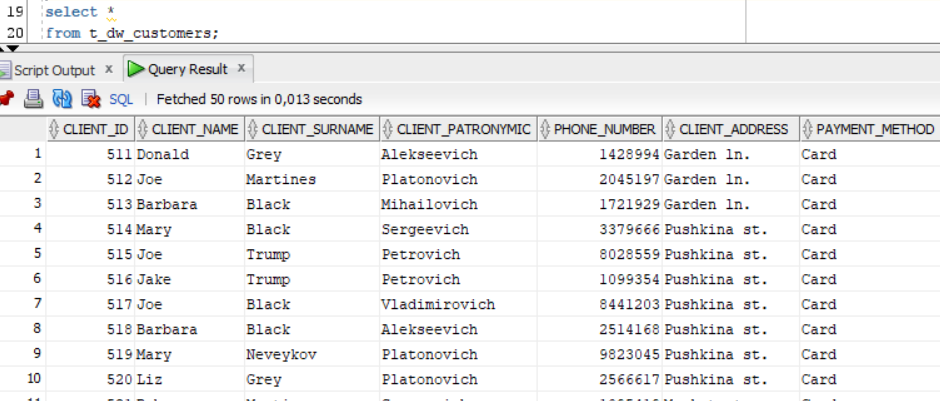
Создать отдельный пакет каждому dimention:

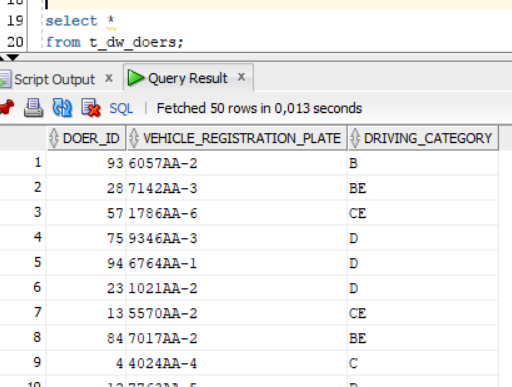


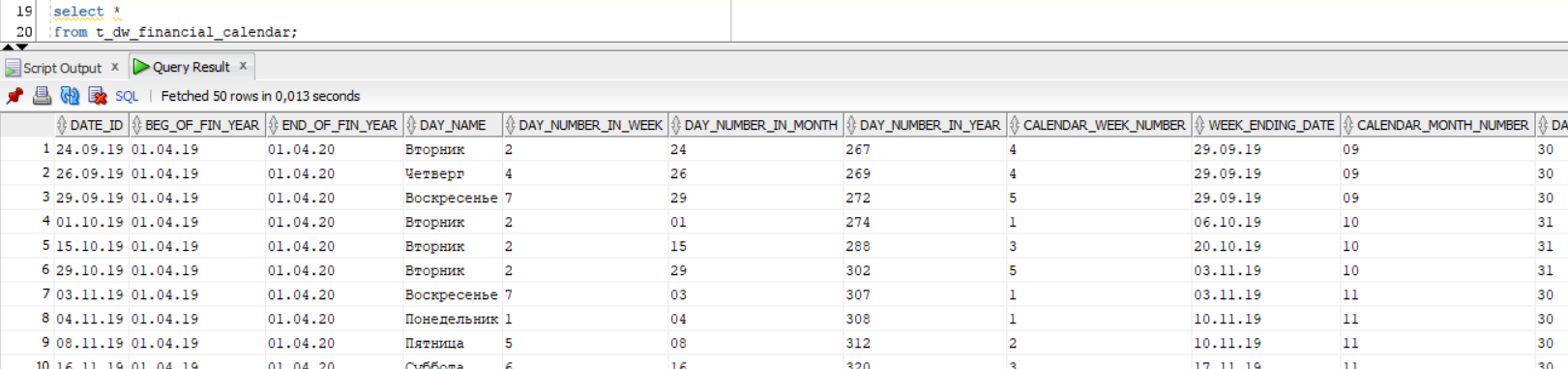
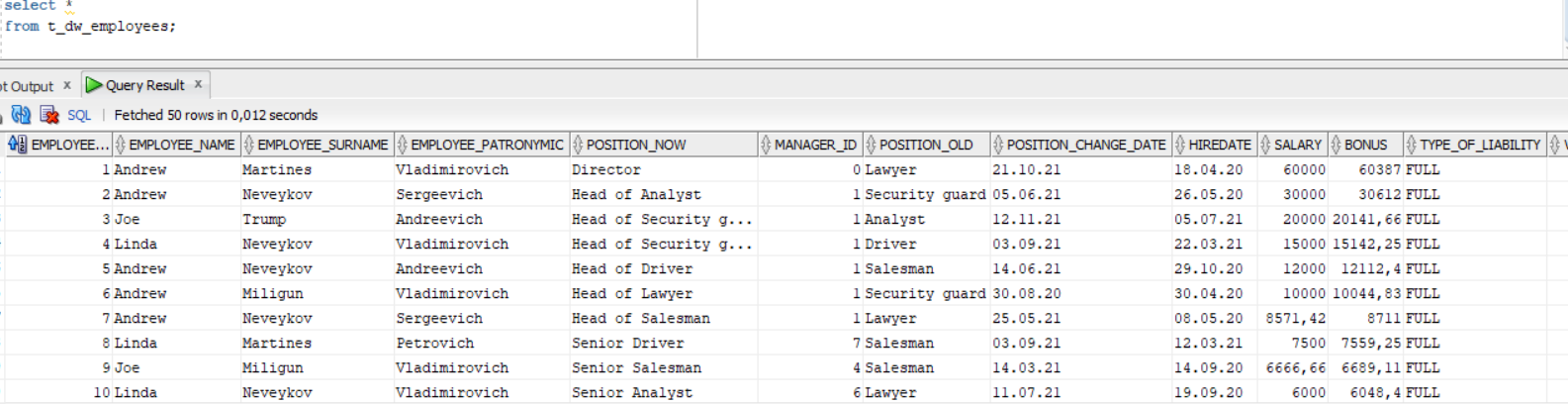


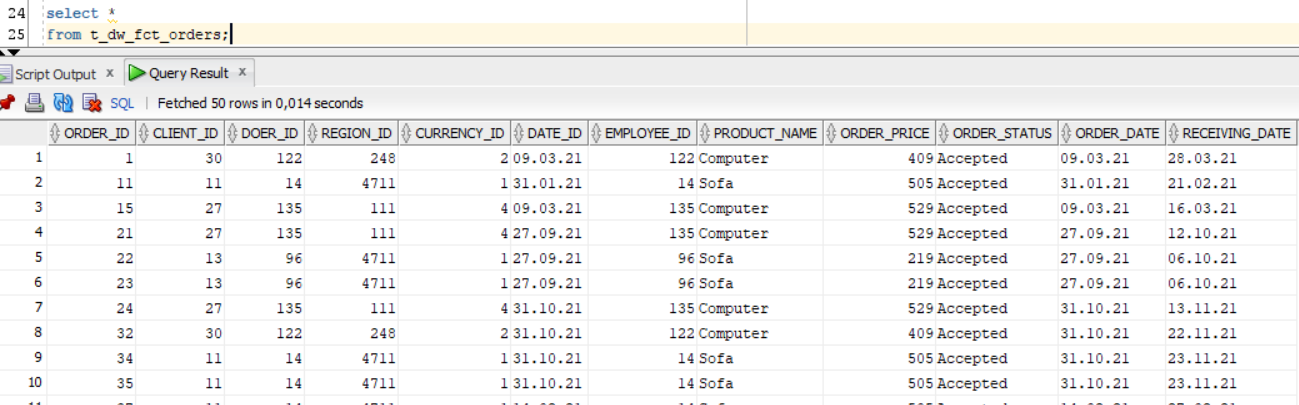
Скриншоты с данным на dimentions:

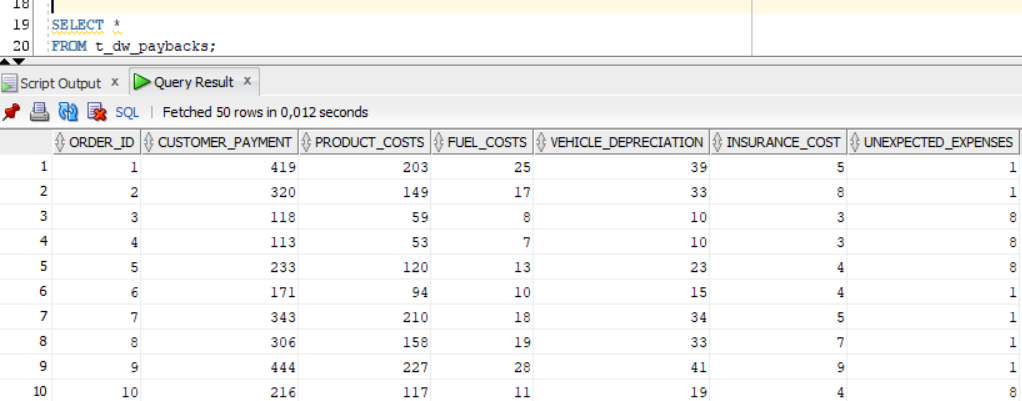


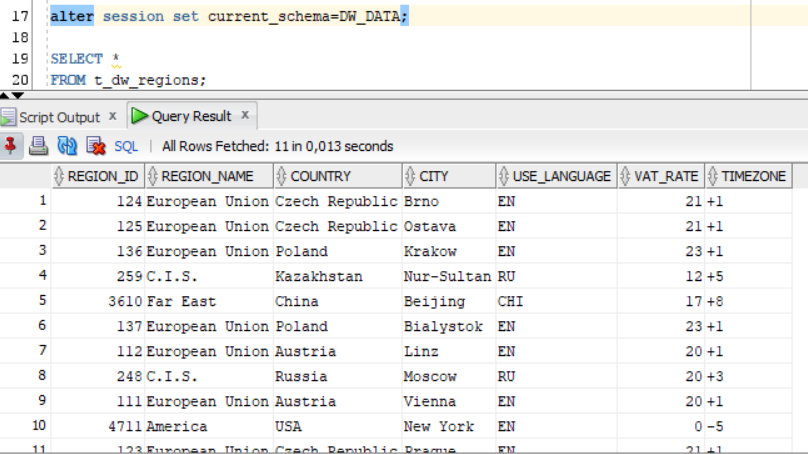




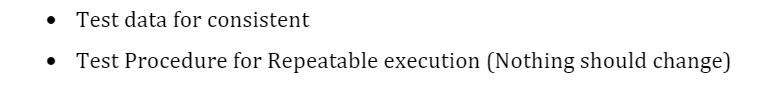






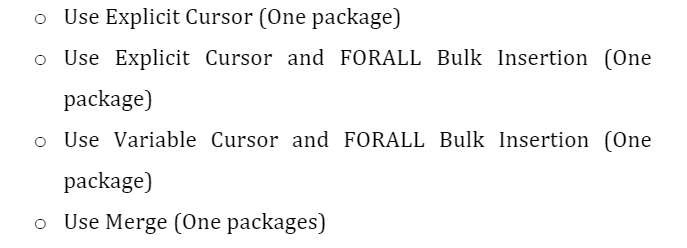


Протестировать процедуры:



***Сделано;)***

Использовать следующие виды курсоров.



**Explicit Cursor применил в пакете для CL уровня pkg\_etl\_cl\_employees:**

CREATE OR REPLACE PACKAGE body pkg\_etl\_cl\_employees

AS

PROCEDURE LOAD\_CLEAN\_EMPLOYEES

AS

CURSOR curs\_cl\_employees

IS

SELECT DISTINCT

employee\_id

, employee\_name

, employee\_surname

, employee\_patronymic

, position\_now

, manager\_id

, position\_old

, position\_change\_date

, hiredate

, salary

, bonus

, type\_of\_liability

, vacation\_days\_number

FROM SA\_CUSTOMERS.t\_sa\_employees

WHERE employee\_id IS NOT NULL

AND employee\_name IS NOT NULL

AND employee\_surname IS NOT NULL

AND employee\_patronymic IS NOT NULL

AND position\_now IS NOT NULL

AND manager\_id IS NOT NULL

AND position\_old IS NOT NULL

AND position\_change\_date IS NOT NULL

AND hiredate IS NOT NULL

AND salary IS NOT NULL

AND bonus IS NOT NULL

AND type\_of\_liability IS NOT NULL

AND vacation\_days\_number IS NOT NULL;

BEGIN

EXECUTE IMMEDIATE 'TRUNCATE TABLE DW\_CL.t\_cl\_employees';

FOR i IN curs\_cl\_employees LOOP

INSERT INTO DW\_CL.t\_cl\_employees(

employee\_id

, employee\_name

, employee\_surname

, employee\_patronymic

, position\_now

, manager\_id

, position\_old

, position\_change\_date

, hiredate

, salary

, bonus

, type\_of\_liability

, vacation\_days\_number)

VALUES ( i.employee\_id

, i.employee\_name

, i.employee\_surname

, i.employee\_patronymic

, i.position\_now

, i.manager\_id

, i.position\_old

, i.position\_change\_date

, i.hiredate

, i.salary

, i.bonus

, i.type\_of\_liability

, i.vacation\_days\_number);

EXIT WHEN curs\_cl\_employees%NOTFOUND;

END LOOP;

COMMIT;

END LOAD\_CLEAN\_EMPLOYEES;

END pkg\_etl\_cl\_employees;

**Explicit Cursor and FOR ALL BLUNK Insertion применил в пакете для DW уровня pkg\_etl\_dw\_currencies**

CREATE OR REPLACE PACKAGE body pkg\_etl\_dw\_currencies

AS

PROCEDURE LOAD\_DW\_CURRENCY

AS

BEGIN

DECLARE

TYPE CURSOR\_INT IS TABLE OF INT;

TYPE CURSOR\_VARCHAR IS TABLE OF varchar2(7);

TYPE CURSOR\_FLOAT IS TABLE OF FLOAT;

TYPE BIG\_CURSOR IS REF CURSOR ;

ALL\_INF BIG\_CURSOR;

currency\_code CURSOR\_INT;

currency\_name CURSOR\_VARCHAR;

direct\_exchange\_rate CURSOR\_FLOAT;

reverse\_exchange\_rate CURSOR\_FLOAT;

currency\_code\_STAGE CURSOR\_INT;

currency\_id CURSOR\_INT;

BEGIN

OPEN ALL\_INF FOR

SELECT

source\_CL.currency\_code AS currency\_code\_source\_CL,

source\_CL.currency\_name AS currency\_name\_source\_CL,

source\_CL.direct\_exchange\_rate AS direct\_exchange\_rate\_source\_CL,

source\_CL.reverse\_exchange\_rate AS reverse\_exchange\_rate\_source\_CL,

stage.currency\_code AS currency\_code\_stage,

stage.currency\_id AS currency\_id

FROM (SELECT DISTINCT \* FROM DW\_CL.t\_cl\_currencies) source\_CL

LEFT JOIN DW\_DATA.t\_dw\_currencies stage

ON (source\_CL.currency\_code = stage.currency\_code);

FETCH ALL\_INF

BULK COLLECT INTO

currency\_code,

currency\_name,

direct\_exchange\_rate,

reverse\_exchange\_rate,

currency\_code\_STAGE,

currency\_id;

CLOSE ALL\_INF;

FOR i IN currency\_id.FIRST .. currency\_id.LAST LOOP

IF ( currency\_id ( i ) IS NULL ) THEN

INSERT INTO DW\_DATA.t\_dw\_currencies (

currency\_id,

currency\_code,

currency\_name,

direct\_exchange\_rate,

reverse\_exchange\_rate)

VALUES ( SEQ\_CURRENCIES.NEXTVAL,

currency\_code(i),

currency\_name(i),

direct\_exchange\_rate(i),

reverse\_exchange\_rate(i));

COMMIT;

ELSE UPDATE DW\_DATA.t\_dw\_currencies

SET

currency\_code = currency\_code(i),

currency\_name = currency\_name(i),

direct\_exchange\_rate = direct\_exchange\_rate(i),

reverse\_exchange\_rate = reverse\_exchange\_rate(i)

WHERE currency\_id = currency\_id(i);

COMMIT;

END IF;

END LOOP;

END;

END LOAD\_DW\_CURRENCY;

END pkg\_etl\_dw\_currencies;

**Variable Cursor and FOR ALL BLUNK Insertion применил в пакете для DW уровня pkg\_etl\_dw\_fct\_orders**

alter session set current\_schema=DW\_CL;

GRANT SELECT ON DW\_CL.t\_cl\_transactions TO DW\_DATA;

alter session set current\_schema = DW\_DATA;

CREATE OR REPLACE PACKAGE body pkg\_etl\_dw\_fct\_orders

AS

PROCEDURE LOAD\_DW\_FCT\_ORDERS

AS

BEGIN

DECLARE

TYPE CURSOR\_INT IS TABLE OF int;

TYPE CURSOR\_VARCHAR IS TABLE OF varchar2(15);

TYPE CURSOR\_DATE IS TABLE OF date;

TYPE CURSOR\_FLOAT IS TABLE OF float;

TYPE BIG\_CURSOR IS REF CURSOR ;

ALL\_INF BIG\_CURSOR;

order\_id CURSOR\_INT;

client\_id CURSOR\_INT;

doer\_id CURSOR\_INT;

region\_id CURSOR\_INT;

currency\_id CURSOR\_INT;

date\_id CURSOR\_DATE;

employee\_id CURSOR\_INT;

product\_name CURSOR\_VARCHAR;

order\_price CURSOR\_FLOAT;

order\_status CURSOR\_VARCHAR;

order\_date CURSOR\_DATE;

receiving\_date CURSOR\_DATE;

BEGIN

OPEN ALL\_INF FOR

SELECT

dw\_ord.order\_id

, cl.client\_id

, dr.doer\_id

, reg.region\_id

, cur.currency\_id

, calen.date\_id

, emp.employee\_id

, source\_cl.product\_name

, source\_cl.order\_price

, source\_cl.order\_status

, source\_cl.order\_date

, source\_cl.receiving\_date

FROM (SELECT DISTINCT \*

FROM DW\_CL.t\_cl\_transactions) source\_cl

INNER JOIN

DW\_DATA.t\_dw\_customers cl

ON (source\_cl.client\_id=cl.client\_id)

INNER JOIN

DW\_DATA.t\_dw\_doers dr

ON (source\_cl.doer\_id=dr.doer\_id)

INNER JOIN

DW\_DATA.t\_dw\_currencies cur

ON (source\_cl.currency\_code=cur.currency\_code)

INNER JOIN

DW\_DATA.t\_dw\_financial\_calendar calen

ON (source\_cl.date\_id=calen.date\_id)

INNER JOIN

DW\_DATA.t\_dw\_regions reg

ON (source\_cl.region\_id=reg.region\_id)

INNER JOIN

DW\_DATA.t\_dw\_employees emp

ON (source\_cl.employee\_id=emp.employee\_id)

LEFT JOIN

DW\_DATA.t\_dw\_fct\_orders dw\_ord

ON (cl.client\_id=dw\_ord.client\_id AND calen.date\_id=dw\_ord.date\_id AND

cur.currency\_id=dw\_ord.currency\_id AND dr.doer\_id=dw\_ord.doer\_id);

FETCH ALL\_INF

BULK COLLECT INTO

order\_id

, client\_id

, doer\_id

, region\_id

, currency\_id

, date\_id

, employee\_id

, product\_name

, order\_price

, order\_status

, order\_date

, receiving\_date;

CLOSE ALL\_INF;

FOR i IN ORDER\_ID.FIRST .. ORDER\_ID.LAST LOOP

IF ( ORDER\_ID ( i ) IS NULL ) THEN

INSERT INTO dw\_data.t\_dw\_fct\_orders (ORDER\_ID

, client\_id

, doer\_id

, region\_id

, currency\_id

, date\_id

, employee\_id

, product\_name

, order\_price

, order\_status

, order\_date

, receiving\_date)

VALUES ( SEQ\_FCT\_ORDERS.NEXTVAL

, client\_id (i)

, doer\_id (i)

, region\_id (i)

, currency\_id (i)

, date\_id (i)

, employee\_id (i)

, product\_name (i)

, order\_price (i)

, order\_status (i)

, order\_date (i)

, receiving\_date (i));

COMMIT;

END IF;

END LOOP;

END;

END LOAD\_DW\_FCT\_ORDERS;

END pkg\_etl\_dw\_fct\_orders;

alter session set current\_schema=DW\_DATA;

exec pkg\_etl\_dw\_fct\_orders.LOAD\_DW\_FCT\_ORDERS;

select \* from t\_dw\_fct\_orders;

commit;

**Merge применил в пакете для DW уровня pkg\_etl\_dw\_employees**

CREATE OR REPLACE PACKAGE body pkg\_etl\_dw\_employees

AS

PROCEDURE LOAD\_DW\_EMPLOYEES

AS

BEGIN

MERGE INTO DW\_DATA.t\_dw\_employees A

USING ( SELECT employee\_id, employee\_name, employee\_surname, employee\_patronymic, position\_now,

manager\_id, position\_old, position\_change\_date, hiredate, salary, bonus,

type\_of\_liability, vacation\_days\_number

FROM DW\_CL.t\_cl\_employees) B

ON (a.employee\_id = b.employee\_id)

WHEN MATCHED THEN

UPDATE SET a.salary = b.salary

WHEN NOT MATCHED THEN

INSERT (a.employee\_id, a.employee\_name, a.employee\_surname, a.employee\_patronymic, a.position\_now,

a.manager\_id, a.position\_old, a.position\_change\_date, a.hiredate, a.salary, a.bonus,

a.type\_of\_liability, a.vacation\_days\_number)

VALUES (b.employee\_id, b.employee\_name, b.employee\_surname, b.employee\_patronymic, b.position\_now,

b.manager\_id, b.position\_old, b.position\_change\_date, b.hiredate, b.salary, b.bonus,

b.type\_of\_liability, b.vacation\_days\_number);

COMMIT;

END LOAD\_DW\_EMPLOYEES;

END pkg\_etl\_dw\_employees;