Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра АСОІУ

Звіт з лабораторної роботи № 4

з дисципліни

«OLAP та сховища даних»

Виконав:

студент 3-го курсу
групи ІП-52, ФІОТ
Набоков Едуард

Київ 2017

Завдання

Завдання на підготовку до комп'ютерного практикуму

Матеріали лекцій.

Завдання 1.Створення процедур завантаження даних

Маємо 2 схеми - task1, task3 (КР1, КР3). Створіть пакет з набором процедур/функцій, який виконає завантаження та перетворення даних (INVOICE, SALES, STORE) до відповідних таблиць схеми task3.

Зауваження 1

Дані, яких не вистачає заповнити за власне сформованими правилами (наприклад, якщо в таблицях відсутні дані про позицію працівника – у таблиці POSITION створити позицію з ідентифікатором «1», назвою позиції «Продавець касир» та заробітною платою «4000»)

При заповненні таблиць даними виконувати перевірку, чи вже існують такі дані (merge)

Зауваження 2

Данні мають бути коректними

Проблемні дані занести до таблиці LOG, що має наступну структуру:

ID	DATE	PROBLEM_DATA
1	10.01.2014	TABLE STUFF COLUMN NAME_STUFF ID 123

Так, якщо маємо початкову таблицю **INVOICE** у схемі task1, то дані у нормалізованій схемі можуть мати наступний вигляд.

SOURCE

INVOICE

ID_STUFF	5
STAFF_NAME	Simon Hughes
E_MAIL	Hughes .Simon@shop.com
INVOICE	INV-15
SUPPLIER	LG
PRODUCT	TV-10
QUANTITY	1

PRICE	3192
INVOICE_DATE	08.05.2011

TARGET

STUFF

ID_STUFF 5
ID_POSITION 1
NAME Simon
SURNAME Hughes
PHONE
ADDRESS
EMAIL Hughes .Simon@shop.com

INVOICE_DETAIL

ID_INVOICE 15
ID_PRODUCT 1
QUANTITY 1
DESCRIPTION

PRODUCT

ID_PRODUCT 1
ID_PRODUCT_TYPE 1
ID_SUPPLIER 1
ID_UNIT 1
PRODUCT_NAME LG TV 10
DESCRIPTION
PRICE 3192

INVOICE

ID_INVOICE 15
ID_TYPE 1
ID_STUFF 5
PURCHASE_TIME 08.05.2011

STORE

STAFF_NAME	Barry Davies
SUPPLIER	Sony
SHELF	2
PRODUCT	TV-23
QUANTITY	8
OPER_TYPE	in
STORE_DATE	10.02.2013

STUFF

ID_STUFF
ID_POSITION
NAME Barry
SURNAME Davies
PHONE
ADDRESS
EMAIL

STORE

ID_STORE 1
ID_PRODUCT 2
SHELF 2
DATE_OPER 10.02.2013
ID_OPER_TYPE 1
QUANTITY 8

SUPPLIER

ID_SUPPLIER 2
SUPPLIER_NAME SONY
SUPPLIER_INFO

PRODUCT

ID_PRODUCT 2
ID_PRODUCT_TYPE 1
ID_SUPPLIER 2
ID_UNIT 1
PRODUCT_NAME Sony TV - 23
DESCRIPTION
PRICE 3192

TYPE_OPER

ID_OPER_TYPE 1
NAME_OPER IN
DESCRIPTION

Вимоги до звіту комп'ютерного практикуму

Звіт до комп'ютерного практикуму виконується у Microsoft Word та повинен містити наступні розділи:

1. Відомості про виконавця
2. Текст процедур.

Контрольні запитання та завдання

1. Дайте визначення ETL.
2. На чому базується процес ETL?

Виконання роботи

```
create table INVOICE_S  (
   ID_INVOICE NUMBER NOT NULL,
   ID_TYPE NUMBER NOT NULL,
   ID_STUFF NUMBER NOT NULL,
   PURCHASE_TIME DATE NOT NULL,
   CONSTRAINT PK_INVOICE_S PRIMARY KEY ("ID_INVOICE")
   --CONSTRAINT FK_STUFF FOREIGN KEY ("ID_STUFF") REFERENCES "STUFF"("ID_STUFF")
);

create table INVOICE_D  (
   ID_INVOICE NUMBER NOT NULL,
   ID_TYPE NUMBER NOT NULL,
   ID_STUFF NUMBER NOT NULL,
   PURCHASE_TIME DATE NOT NULL,
   CONSTRAINT PK_INVOICE_D PRIMARY KEY ("ID_INVOICE")
   --CONSTRAINT FK_STUFF FOREIGN KEY ("ID_STUFF") REFERENCES "STUFF"("ID_STUFF")
);

CREATE TABLE PRODUCT_S (
  ID_PRODUCT NUMBER NOT NULL,
  ID_PRODUCT_TYPE NUMBER NOT NULL,
  ID_SUPPLIER NUMBER NOT NULL,
  ID_UNIT NUMBER NOT NULL,
  PRODUCT_NAME VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(255),
  PRICE NUMBER NOT NULL,
  CONSTRAINT PK_PRODUCT_S PRIMARY KEY ("ID_PRODUCT")
  --CONSTRAINT FK_STUFF FOREIGN KEY ("ID_STUFF") REFERENCES "STUFF"("ID_STUFF")
);

CREATE TABLE PRODUCT_D (
  ID_PRODUCT NUMBER NOT NULL,
  ID_PRODUCT_TYPE NUMBER NOT NULL,
  ID_SUPPLIER NUMBER NOT NULL,
  ID_UNIT NUMBER NOT NULL,
  PRODUCT_NAME VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(255),
  PRICE NUMBER NOT NULL,
  CONSTRAINT PK_PRODUCT_D PRIMARY KEY ("ID_PRODUCT")
  --CONSTRAINT FK_STUFF FOREIGN KEY ("ID_STUFF") REFERENCES "STUFF"("ID_STUFF")
);

CREATE TABLE STUFF_S (
  ID_STUFF NUMBER NOT NULL,
  ID_POSITION NUMBER NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  SURNAME VARCHAR(255) NOT NULL,
  PHONE VARCHAR(20),
  ADDRESS VARCHAR(255),
  EMAIL VARCHAR(255) NOT NULL,
  CONSTRAINT PK_STUFF_S PRIMARY KEY ("ID_STUFF")
);

CREATE TABLE STUFF_D (
  ID_STUFF NUMBER NOT NULL,
  ID_POSITION NUMBER NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  SURNAME VARCHAR(255) NOT NULL,
  PHONE VARCHAR(20),
  ADDRESS VARCHAR(255),
```

```
  EMAIL VARCHAR(255) NOT NULL,
  CONSTRAINT PK_STUFF_D PRIMARY KEY ("ID_STUFF")
);

CREATE TABLE INVOICE_DETAIL_S (
  ID_INVOICE NUMBER NOT NULL,
  ID_PRODUCT NUMBER NOT NULL,
  QUANTITY NUMBER NOT NULL,
  DESCRIPTION VARCHAR(255)
);

CREATE TABLE INVOICE_DETAIL_D (
  ID_INVOICE NUMBER NOT NULL,
  ID_PRODUCT NUMBER NOT NULL,
  QUANTITY NUMBER NOT NULL,
  DESCRIPTION VARCHAR(255)
);


CREATE TABLE SUPPLIER_S (
  ID_SUPPLIER NUMBER NOT NULL,
  SUPPLIER_NAME VARCHAR(255) NOT NULL,
  SUPPLIER_INFO VARCHAR(255),
  CONSTRAINT PK_SUPPLIER_S PRIMARY KEY ("ID_SUPPLIER")
);

CREATE TABLE SUPPLIER_D (
  ID_SUPPLIER NUMBER NOT NULL,
  SUPPLIER_NAME VARCHAR(255) NOT NULL,
  SUPPLIER_INFO VARCHAR(255),
  CONSTRAINT PK_SUPPLIER_D PRIMARY KEY ("ID_SUPPLIER")
);

CREATE TABLE STORE_S (
  ID_STORE NUMBER NOT NULL,
  ID_PRODUCT NUMBER NOT NULL,
  SHELF NUMBER NOT NULL,
  DATE_OPER DATE NOT NULL,
  ID_OPER_TYPE NUMBER NOT NULL,
  QUANTITY NUMBER NOT NULL,
  CONSTRAINT PK_STORE_S PRIMARY KEY ("ID_STORE")
);

CREATE TABLE STORE_D (
  ID_STORE NUMBER NOT NULL,
  ID_PRODUCT NUMBER NOT NULL,
  SHELF NUMBER NOT NULL,
  DATE_OPER DATE NOT NULL,
  ID_OPER_TYPE NUMBER NOT NULL,
  QUANTITY NUMBER NOT NULL,
  CONSTRAINT PK_STORE_D PRIMARY KEY ("ID_STORE")
);

CREATE TABLE TYPE_OPER_S (
  ID_OPER_TYPE NUMBER NOT NULL,
  NAME_OPER VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(255),
  CONSTRAINT PK_OPER_TYPE_S PRIMARY KEY ("ID_OPER_TYPE")
);

CREATE TABLE TYPE_OPER_D (
  ID_OPER_TYPE NUMBER NOT NULL,
```

```
  NAME_OPER VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(255),
  CONSTRAINT PK_OPER_TYPE_D PRIMARY KEY ("ID_OPER_TYPE")
);



-- MERGING
CREATE OR REPLACE PROCEDURE LOAD_STUFF IS
 CURSOR C1 IS
    SELECT DISTINCT ID_STUFF, STAFF_NAME, E_MAIL FROM INVOICE WHERE REGEXP_LIKE (E_MAIL,
'^[A-Za-z]+[A-Za-z0-9. ]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$');
BEGIN
  FOR STUFF_REC IN C1
  LOOP
  INSERT INTO STUFF_S (ID_STUFF, ID_POSITION, NAME, SURNAME, EMAIL) VALUES
(STUFF_REC.ID_STUFF, 1,
  REGEXP_SUBSTR (STUFF_REC.STAFF_NAME, '[^ ]+', 1, 1), REGEXP_SUBSTR
(STUFF_REC.STAFF_NAME, '[^ ]+', 1, 2), STUFF_REC.E_MAIL);
  END LOOP;
END;

EXECUTE LOAD_STUFF;

select * from stuff_s;
truncate table stuff_s;
-- 2 load valid supplier records
SELECT DISTINCT SUPPLIER FROM INVOICE;

CREATE OR REPLACE PROCEDURE LOAD_SUPPLIER IS
 CURSOR C2 IS
    SELECT DISTINCT SUPPLIER FROM INVOICE WHERE REGEXP_LIKE (SUPPLIER, '^[A-Za-z]');
 ID_NUM INTEGER := 1;
BEGIN
  FOR SUPPLIER_REC IN C2
  LOOP
  INSERT INTO SUPPLIER_S (ID_SUPPLIER, SUPPLIER_NAME) VALUES (ID_NUM,
SUPPLIER_REC.SUPPLIER);
  ID_NUM := ID_NUM + 1;
  END LOOP;
END;

EXECUTE LOAD_SUPPLIER;
SELECT * FROM SUPPLIER_S;

-- 3 load valid product records

CREATE OR REPLACE PROCEDURE LOAD_PRODUCT IS
 CURSOR C3 IS
    SELECT DISTINCT SUPPLIER, PRODUCT, PRICE FROM INVOICE WHERE REGEXP_LIKE (SUPPLIER,
'^[A-Za-z]') AND PRODUCT LIKE 'TV-__';
 ID_NUM INTEGER := 1;
 ID_SUP INTEGER;
BEGIN
  FOR PRODUCT_REC IN C3
  LOOP
  SELECT ID_SUPPLIER INTO ID_SUP FROM SUPPLIER_S WHERE SUPPLIER_NAME =
PRODUCT_REC.SUPPLIER;
  INSERT INTO PRODUCT_S (ID_PRODUCT, ID_PRODUCT_TYPE, ID_SUPPLIER, ID_UNIT,
PRODUCT_NAME, PRICE)
  VALUES (ID_NUM, 1, ID_SUP, 1, PRODUCT_REC.PRODUCT, PRODUCT_REC.PRICE);
```

```
  ID_NUM := ID_NUM + 1;
  END LOOP;
END;

EXECUTE LOAD_PRODUCT;
SELECT * FROM PRODUCT_S;
SELECT * FROM SUPPLIER_S;
SELECT * FROM INVOICE;
-- 4 load valid invoice records

CREATE FUNCTION is_number (p_string IN VARCHAR2)
   RETURN INT
IS
   v_new_num NUMBER;
BEGIN
   v_new_num := TO_NUMBER(p_string);
   RETURN 1;
EXCEPTION
WHEN VALUE_ERROR THEN
   RETURN 0;
END is_number;

CREATE OR REPLACE PROCEDURE LOAD_INVOICE IS
 CURSOR C4 IS
    SELECT DISTINCT INVOICE, QUANTITY, ID_STUFF, INVOICE_DATE FROM INVOICE WHERE INVOICE
LIKE 'INV-%' AND LENGTH(INVOICE) < 8
    AND IS_NUMBER(QUANTITY) = 1 AND
    REGEXP_LIKE (INVOICE_DATE, '^(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[012])\.\d{4}$');
BEGIN
  FOR INVOICE_REC IN C4
  LOOP
  IF INVOICE_REC.QUANTITY BETWEEN 0 AND 100 THEN
    INSERT INTO INVOICE_S (ID_INVOICE, ID_TYPE, ID_STUFF, PURCHASE_TIME)
    VALUES (REGEXP_SUBSTR (INVOICE_REC.INVOICE, '[^-]+', 1, 2), 1, INVOICE_REC.ID_STUFF,
INVOICE_REC.INVOICE_DATE);
  END IF;
  END LOOP;
END;

EXECUTE LOAD_INVOICE;
SELECT * FROM INVOICE_S;
TRUNCATE TABLE INVOICE_S;
SELECT DISTINCT INVOICE_DATE FROM INVOICE;
-- 5 load valid invoice info records

CREATE OR REPLACE PROCEDURE LOAD_INVOICE_DETAILS IS
 CURSOR C5 IS
    SELECT DISTINCT REGEXP_SUBSTR (INVOICE, '[^-]+', 1, 2) AS ID_INVOICE, QUANTITY, SUPPLIER,
PRODUCT, PRICE FROM INVOICE
    WHERE INVOICE LIKE 'INV-%' AND LENGTH(INVOICE) < 8
    AND IS_NUMBER(QUANTITY) = 1 AND REGEXP_LIKE(SUPPLIER,'^[A-Za-z]') AND PRODUCT LIKE 'TV-
__' AND
    REGEXP_LIKE (INVOICE_DATE, '^(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[012])\.\d{4}$');
    ID_P NUMBER;
BEGIN
  FOR INVOICE_REC IN C5
  LOOP
  IF INVOICE_REC.QUANTITY BETWEEN 0 AND 100 THEN
    SELECT DISTINCT ID_PRODUCT INTO ID_P FROM PRODUCT_S T1 JOIN SUPPLIER_S T2 ON
(T1.ID_SUPPLIER = T2.ID_SUPPLIER) WHERE SUPPLIER_NAME = INVOICE_REC.SUPPLIER
    AND PRODUCT_NAME = INVOICE_REC.PRODUCT AND PRICE = INVOICE_REC.PRICE;
    INSERT INTO INVOICE_DETAIL_S (ID_INVOICE, ID_PRODUCT, QUANTITY)
```

```
    VALUES (INVOICE_REC.ID_INVOICE, ID_P, INVOICE_REC.QUANTITY);
  END IF;
  END LOOP;
END;

EXECUTE LOAD_INVOICE_DETAILS;
SELECT * FROM INVOICE_DETAIL_S;
SELECT * FROM INVOICE;

-- 6 load vaild product records from store

-- insert oper types
INSERT INTO TYPE_OPER_S (ID_OPER_TYPE, NAME_OPER) VALUES
(1, 'IN');

INSERT INTO TYPE_OPER_S (ID_OPER_TYPE, NAME_OPER) VALUES
(2, 'OUT');

CREATE OR REPLACE PROCEDURE LOAD_PRODUCT_STORE IS
 CURSOR C6 IS
    SELECT DISTINCT SUPPLIER, PRODUCT, PRODUCT_NAME FROM STORE T1 LEFT JOIN PRODUCT_S
T2 ON (T1.PRODUCT = T2.PRODUCT_NAME)
    WHERE SUPPLIER IN (SELECT SUPPLIER_NAME FROM SUPPLIER_S)
    AND PRODUCT LIKE 'TV-__';
    ID_S NUMBER;
    ID_P NUMBER;
BEGIN
  SELECT MAX(ID_PRODUCT) INTO ID_P FROM PRODUCT_S;
  ID_P := ID_P + 1;
  FOR P_REC IN C6
  LOOP
  IF P_REC.PRODUCT_NAME IS NULL THEN
   SELECT ID_SUPPLIER INTO ID_S FROM SUPPLIER_S WHERE SUPPLIER_NAME = P_REC.SUPPLIER;
   INSERT INTO PRODUCT_S (ID_PRODUCT, ID_PRODUCT_TYPE, ID_SUPPLIER, ID_UNIT,
PRODUCT_NAME, PRICE) VALUES
   (ID_P, 1, ID_S, 1, P_REC.PRODUCT, 228);
   ID_P := ID_P + 1;
  END IF;
  END LOOP;
END;

EXECUTE LOAD_PRODUCT_STORE;
SELECT * FROM PRODUCT_S;

CREATE FUNCTION IS_DATE (p_string IN VARCHAR2)
   RETURN INT
IS
   V_NEW_DATE DATE;
BEGIN
   V_NEW_DATE := TO_DATE(p_string);
   RETURN 1;
EXCEPTION
WHEN others THEN
   RETURN 0;
END IS_DATE;

CREATE OR REPLACE PROCEDURE LOAD_STORE IS
 CURSOR C7 IS
    SELECT SHELF, QUANTITY, STORE_DATE, REGEXP_SUBSTR (STUFF_NAME, '[^ ]+', 1, 1) AS NAME,
    REGEXP_SUBSTR (STUFF_NAME, '[^ ]+', 1, 2) AS SURNAME, SUPPLIER, PRODUCT, OPER_TYPE
    FROM STORE WHERE REGEXP_SUBSTR (STUFF_NAME, '[^ ]+', 1, 1) IN (SELECT NAME FROM
STUFF_S) AND REGEXP_SUBSTR (STUFF_NAME, '[^ ]+', 1, 2) IN
```

```
    (SELECT SURNAME FROM STUFF_S) AND SUPPLIER IN (SELECT SUPPLIER_NAME FROM
SUPPLIER_S) AND SHELF < 100 AND
    REGEXP_LIKE (STORE_DATE, '^(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[012])\.\d{4}$') AND
    PRODUCT IN (SELECT PRODUCT_NAME FROM PRODUCT_S) AND UPPER(OPER_TYPE) IN (SELECT
NAME_OPER FROM TYPE_OPER_S);
    ID_P NUMBER;
    ID_S NUMBER := 1;
    ID_SUP NUMBER;
    ID_OP NUMBER;
BEGIN
  FOR INVOICE_REC IN C7
  LOOP
 IF IS_NUMBER(INVOICE_REC.QUANTITY) = 1 AND INVOICE_REC.QUANTITY BETWEEN 0 AND 100 AND
IS_DATE(INVOICE_REC.STORE_DATE) = 1 THEN
    DBMS_OUTPUT.PUT_LINE('WTF');
    SELECT ID_SUPPLIER INTO ID_SUP FROM SUPPLIER_S WHERE SUPPLIER_S.SUPPLIER_NAME =
INVOICE_REC.SUPPLIER;
    --SELECT ID_PRODUCT INTO ID_P FROM PRODUCT_S WHERE PRODUCT_S.PRODUCT_NAME =
INVOICE_REC.PRODUCT
    --AND PRODUCT_S.ID_SUPPLIER = ID_SUP;
    SELECT ID_OPER_TYPE INTO ID_OP FROM TYPE_OPER_S WHERE NAME_OPER =
UPPER(INVOICE_REC.OPER_TYPE);
    DBMS_OUTPUT.PUT_LINE('ID_S' || ID_S || ' ID_P' || ID_P || ' SHELF' || INVOICE_REC.SHELF || ' DATE' ||
INVOICE_REC.STORE_DATE);
    INSERT INTO STORE_S (ID_STORE, ID_PRODUCT, SHELF, DATE_OPER, ID_OPER_TYPE, QUANTITY)
    VALUES (ID_S, ID_S, INVOICE_REC.SHELF, TO_DATE(INVOICE_REC.STORE_DATE), ID_OP,
INVOICE_REC.QUANTITY);
    ID_S := ID_S + 1;
  END IF;
  END LOOP;
END;

EXECUTE LOAD_STORE;

-- 7 merge into destination tables

MERGE INTO INVOICE_D TARGET USING (
  SELECT DISTINCT T1.ID_INVOICE, T1.ID_TYPE, T1.ID_STUFF, T1.PURCHASE_TIME FROM INVOICE_S
T1 LEFT JOIN INVOICE_D T2 ON
  (T1.ID_INVOICE = T2.ID_INVOICE)) SOURCE ON (TARGET.ID_INVOICE = SOURCE.ID_INVOICE)
  WHEN MATCHED THEN
    UPDATE SET ID_TYPE = SOURCE.ID_TYPE, ID_STUFF = SOURCE.ID_STUFF, PURCHASE_TIME =
SOURCE.PURCHASE_TIME
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_INVOICE, SOURCE.ID_TYPE, SOURCE.ID_STUFF,
SOURCE.PURCHASE_TIME);

MERGE INTO PRODUCT_D TARGET USING (
  SELECT DISTINCT T1.ID_PRODUCT, T1.ID_PRODUCT_TYPE, T1.ID_SUPPLIER, T1.ID_UNIT,
T1.PRODUCT_NAME,
  T1.DESCRIPTION, T1.PRICE FROM PRODUCT_S T1 LEFT JOIN PRODUCT_D T2 ON
  (T1.ID_PRODUCT = T2.ID_PRODUCT)) SOURCE ON (TARGET.ID_PRODUCT = SOURCE.ID_PRODUCT)
  WHEN MATCHED THEN
    UPDATE SET ID_PRODUCT_TYPE = SOURCE.ID_PRODUCT_TYPE, ID_SUPPLIER =
SOURCE.ID_SUPPLIER, ID_UNIT = SOURCE.ID_UNIT,
    PRODUCT_NAME = SOURCE.PRODUCT_NAME, DESCRIPTION = SOURCE.DESCRIPTION, PRICE =
SOURCE.PRICE
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_PRODUCT, SOURCE.ID_PRODUCT_TYPE, SOURCE.ID_SUPPLIER,
SOURCE.ID_UNIT,
     SOURCE.PRODUCT_NAME, SOURCE.DESCRIPTION, SOURCE.PRICE);
```

```
MERGE INTO INVOICE_DETAIL_D TARGET USING (
  SELECT DISTINCT T1.ID_INVOICE, T1.ID_PRODUCT, T1.QUANTITY, T1.DESCRIPTION FROM
INVOICE_DETAIL_S T1 LEFT JOIN INVOICE_DETAIL_D T2 ON
  (T1.ID_INVOICE = T2.ID_INVOICE)) SOURCE ON (TARGET.ID_INVOICE = SOURCE.ID_INVOICE)
  WHEN MATCHED THEN
    UPDATE SET ID_PRODUCT = SOURCE.ID_PRODUCT, DESCRIPTION = SOURCE.DESCRIPTION
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_INVOICE, SOURCE.ID_PRODUCT, SOURCE.QUANTITY,
SOURCE.DESCRIPTION);

MERGE INTO STORE_D TARGET USING (
  SELECT DISTINCT T1.ID_STORE, T1.ID_PRODUCT, T1.SHELF, T1.DATE_OPER, T1.ID_OPER_TYPE,
T1.QUANTITY FROM STORE_S T1
  LEFT JOIN STORE_D T2 ON
  (T1.ID_STORE = T2.ID_STORE)) SOURCE ON (TARGET.ID_STORE = SOURCE.ID_STORE)
  WHEN MATCHED THEN
    UPDATE SET ID_PRODUCT = SOURCE.ID_PRODUCT, SHELF = SOURCE.SHELF, DATE_OPER =
SOURCE.DATE_OPER, ID_OPER_TYPE = SOURCE.ID_OPER_TYPE,
    QUANTITY = SOURCE.QUANTITY
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_STORE, SOURCE.ID_PRODUCT, SOURCE.SHELF,
SOURCE.DATE_OPER, SOURCE.ID_OPER_TYPE, SOURCE.QUANTITY);

MERGE INTO STUFF_D TARGET USING (
  SELECT DISTINCT T1.ID_STUFF, T1.ID_POSITION, T1.NAME, T1.SURNAME, T1.PHONE, T1.ADDRESS,
T1.EMAIL FROM STUFF_S T1
  LEFT JOIN STUFF_D T2 ON
  (T1.ID_STUFF = T2.ID_STUFF)) SOURCE ON (TARGET.ID_STUFF = SOURCE.ID_STUFF)
  WHEN MATCHED THEN
    UPDATE SET ID_POSITION = SOURCE.ID_POSITION, NAME = SOURCE.NAME, SURNAME =
SOURCE.SURNAME, PHONE = SOURCE.PHONE,
    ADDRESS = SOURCE.ADDRESS, EMAIL = SOURCE.EMAIL
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_STUFF, SOURCE.ID_POSITION, SOURCE.NAME, SOURCE.SURNAME,
SOURCE.PHONE, SOURCE.ADDRESS, SOURCE.EMAIL);

MERGE INTO SUPPLIER_D TARGET USING (
  SELECT DISTINCT T1.ID_SUPPLIER, T1.SUPPLIER_NAME, T1.SUPPLIER_INFO FROM SUPPLIER_S T1
  LEFT JOIN SUPPLIER_D T2 ON
  (T1.ID_SUPPLIER = T2.ID_SUPPLIER)) SOURCE ON (TARGET.ID_SUPPLIER = SOURCE.ID_SUPPLIER)
  WHEN MATCHED THEN
    UPDATE SET SUPPLIER_NAME = SOURCE.SUPPLIER_NAME, SUPPLIER_INFO =
SOURCE.SUPPLIER_INFO
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_SUPPLIER, SOURCE.SUPPLIER_NAME, SOURCE.SUPPLIER_INFO);

MERGE INTO TYPE_OPER_D TARGET USING (
  SELECT DISTINCT T1.ID_OPER_TYPE, T1.NAME_OPER, T1.DESCRIPTION FROM TYPE_OPER_S T1
  LEFT JOIN TYPE_OPER_D T2 ON
  (T1.ID_OPER_TYPE = T2.ID_OPER_TYPE)) SOURCE ON (TARGET.ID_OPER_TYPE =
SOURCE.ID_OPER_TYPE)
  WHEN MATCHED THEN
    UPDATE SET NAME_OPER = SOURCE.NAME_OPER, DESCRIPTION = SOURCE.DESCRIPTION
    WHEN NOT MATCHED THEN
     INSERT VALUES (SOURCE.ID_OPER_TYPE, SOURCE.NAME_OPER, SOURCE.DESCRIPTION);
```