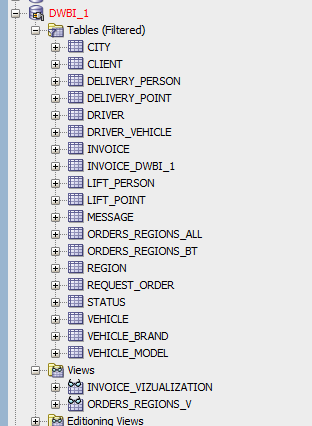
**Proiect**

1. **Crearea bazelor de date și a utilizatorilor – obligatoriu**

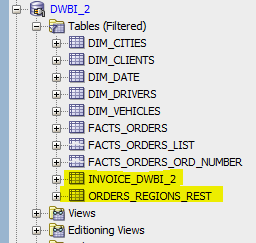
Așa cum am specificat și în modelul de analiză, din motive tehnice, nu am reușit să realizăm configurația cu două servere, așa că am încercat să adaptăm cerințele folosind doi useri din aceeasi baza de date.

Am folosit aceeasi useri creați pentru DW&BI, la care am adaugat o tabela globala suplimentară și tabelele aferente fragmetărilor realizate.

Astfel, avem user-ul DWBI\_1 (considerat ca fiind aflat pe baza de date de pe serverul 1)

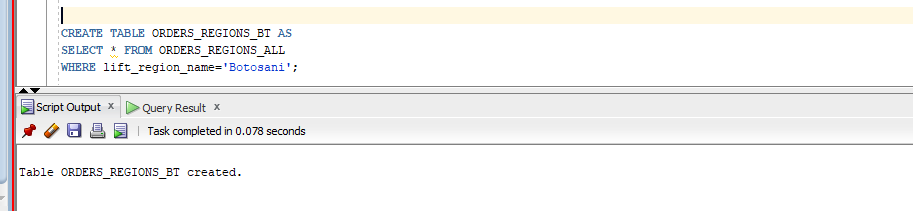


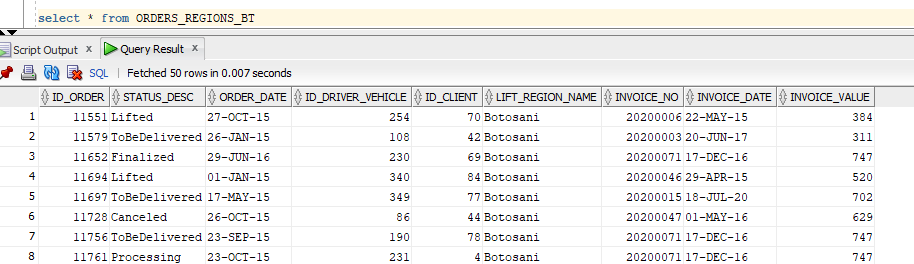
Dar și user-ul DWBI\_2 (considerat ca fiind aflat pe baza de date a serverului 2) – din acest user vom folosi pe parcursul proiectului, doar tabelele fragmentate și constrângerile aferente.



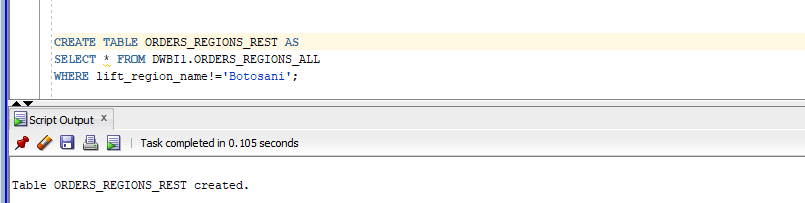
1. **Crearea relațiilor și a fragmentelor – obligatoriu**
2. De creat tabele din fragmentarea orizontala primara

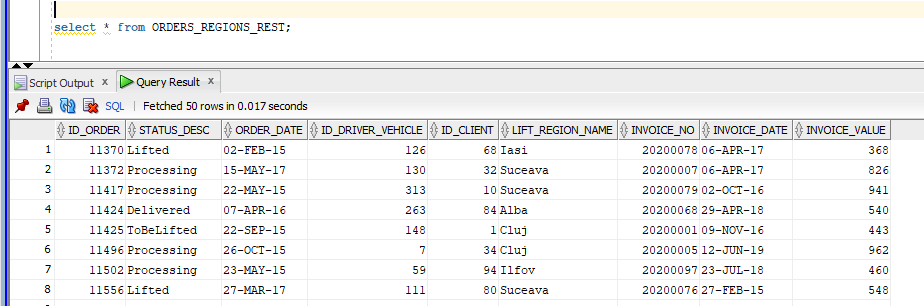
Pe user-ul DWBI1, am creat și populat fragmentul ORDERS\_REGIONS\_BT ce conține înregistrările regiunii Botoșani.





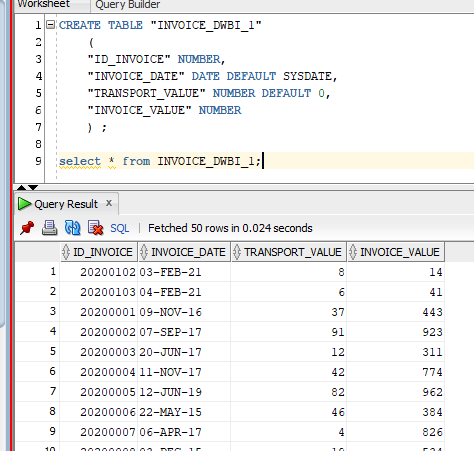
Pe user-ul DWBI2, am creat și populat fragmentul ORDERS\_REGIONS\_REST ce conține înregistrările pentru regiunile diferite de Botoșani.



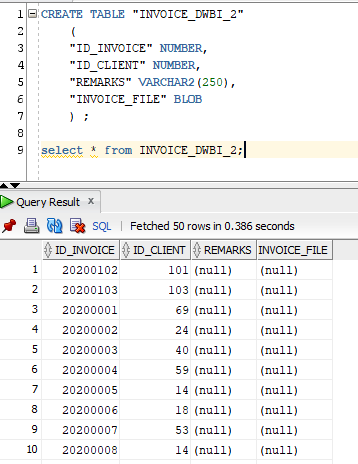


1. De creat tabele din fragmentarea verticala

Pe user-ul DWBI1 am creat și populat fragmentul INVOICE\_DWBI\_1, ce are structura conform algoritmului aplicat în partea de analiză.



Pe user-ul DWBI2 am creat și populat fragmentul INVOICE\_DWBI\_2, ce are structura conform algoritmului aplicat în fișierul de analiză.

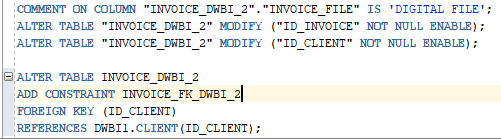


1. De configurat cheile externe

Furnizăm grant pentru a putea crea cheia externă pe user-ul 2.



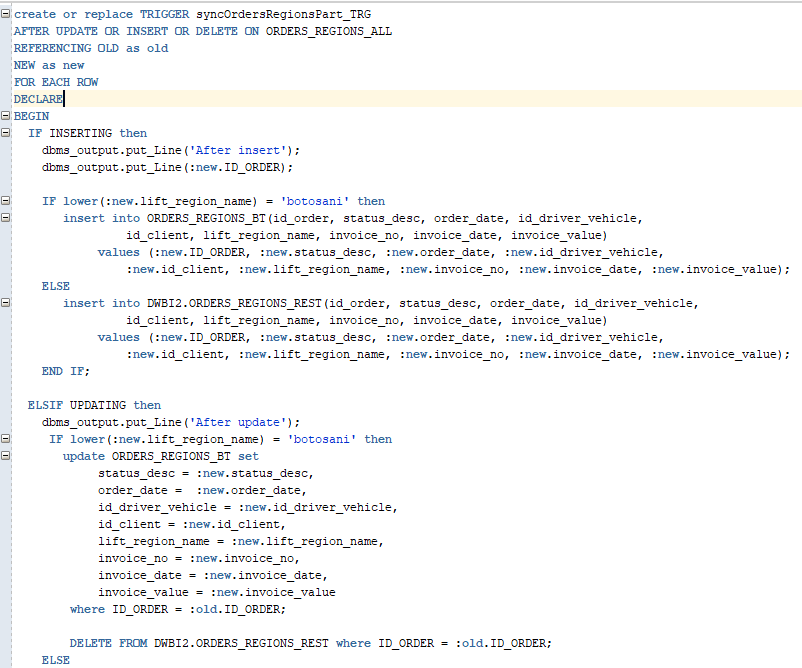
Astfel putem rula scriptul pentru FK.

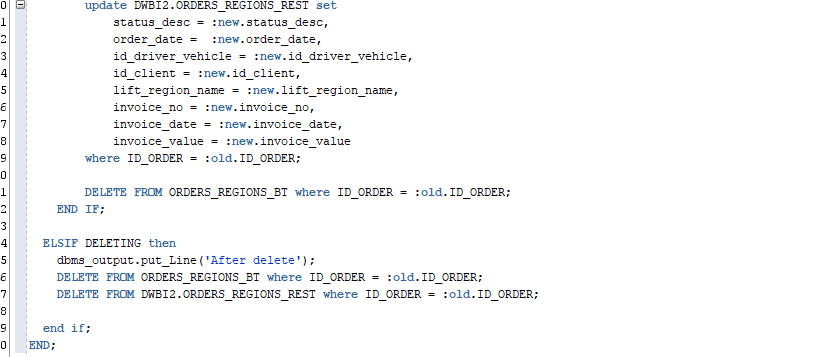


1. **Popularea cu date a bazelor de date – obligatoriu**

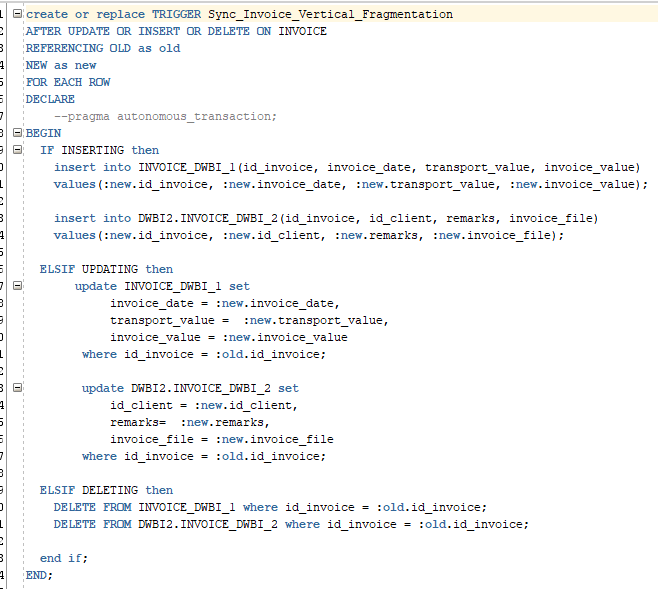
Tabelele fragmentate s-au populat anterior direct la crearea lor iar pentru a ne asigura că datele se vor sincroniza și pe viitor, cream câte un trigger after update/insert/delete pentru fiecare tabelă globală.

Pentru ORDERS\_REGIONS\_ALL – trigger SyncOrdersRegionsPart\_TRG

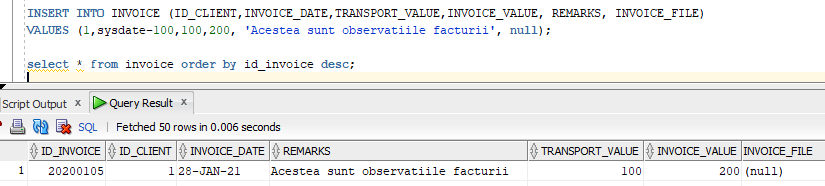




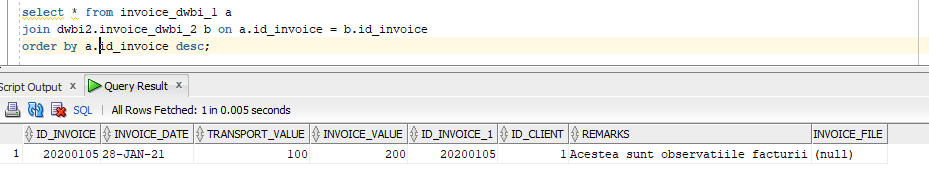
Pentru INVOICE – trigger Sync\_Invoice\_Vertical\_Fragmentation



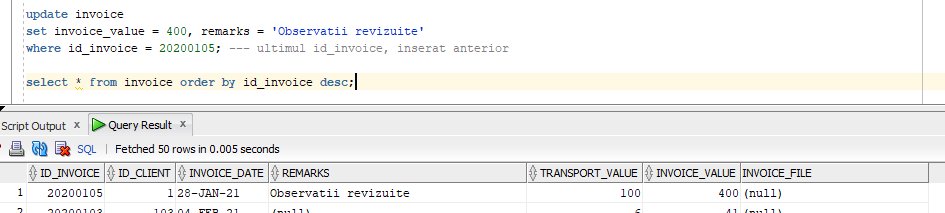
Verificare Select și Insert



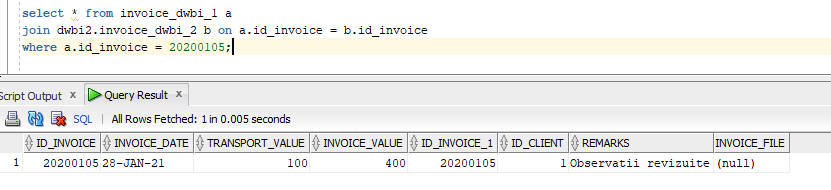
Datele se sincronizeaza si in tabelele fragmentate vertical – se observa din join



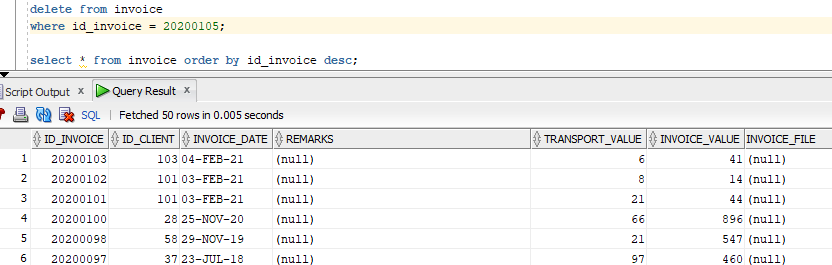
Update-ul



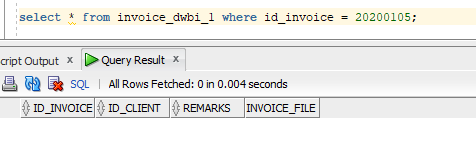
Datele sunt sincronizate si in tablele fragmentate, tot datorita trigger-ului

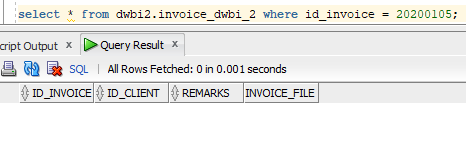


Rulam comanda DELETE pentru facture inserata si dupa cum se observa inregistrarea nu mai apare in tabela principala.



Si nici in tabelele fragmentate

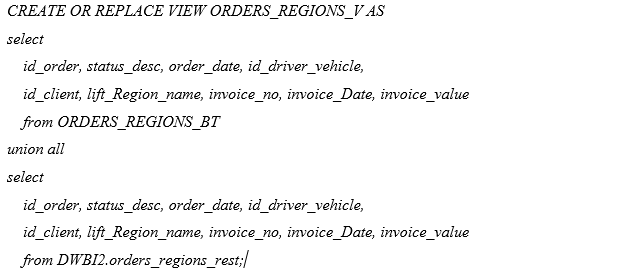




1. **Furnizarea formelor de transparență pentru întreg modelul ales**
2. transparență pentru fragmentele verticale

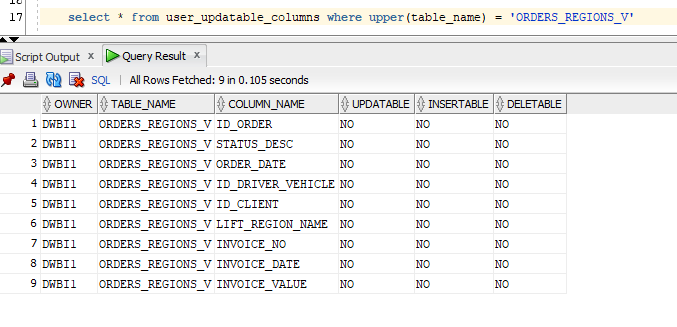
Pentru obținerea transparenței pentru fragmentele verticale vom utiliza vizualizările locale. Acestea pot dermina transparența localizării tabelorlor dintr-o bază de date distribuită, deoarece utilizatorul care accesează vizualizarea nu știu unde sunt localizate fizic datele.

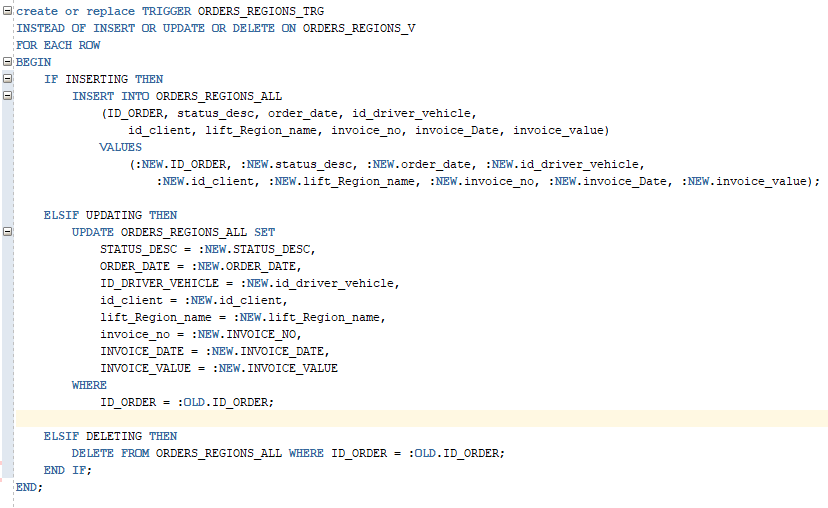
Astfel, pentru reconstrucția relației ORDERS\_REGIONS\_ALL se va defini vizualizarea ORDERS\_REGIONS\_V ce va folosi datele din fragmentele orizonatale ORDERS\_REGIONS\_BT (dwbi1 – server1) și ORDERS\_REGIONS\_REST (dwbi2 – server2).



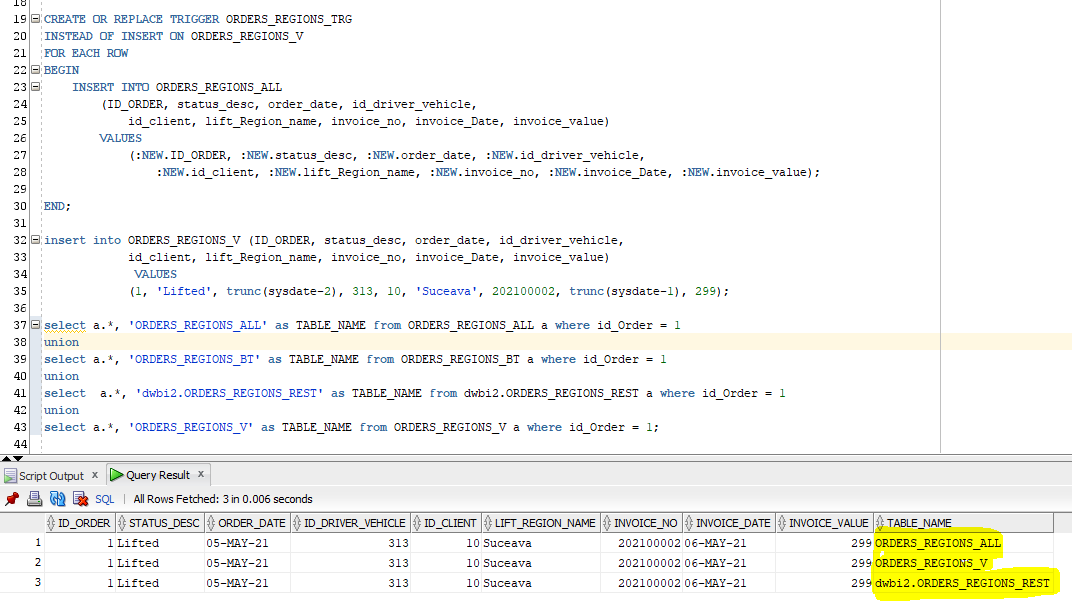
Menționăm faptul că, în cazul în care ORDERS\_REGIONS\_REST se afla pe un alt server bd, atunci în loc de DWBI2.orders\_regions\_rest s-ar fi folosit sintaxa orders\_regions\_rest@nume\_dblink.

Verificând dictionarul datelor, observăm faptul ca niciuna din coloanele vizualizarii create nu permite propagarea operațiilor LMD asupra tabelelor de baza, astfel încât vom fi nevoiți să definim un trigger de tipul INSTEAD OF.





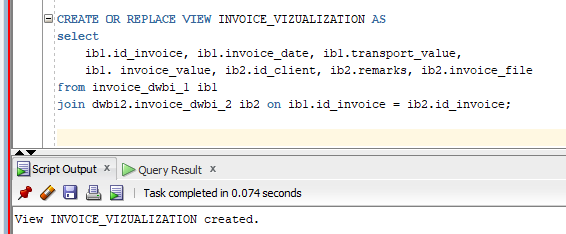
După crearea trigger-ului am efectuat și verificarea corectitudinii datelor, inserând o noua linie folosind vizualizarea nou creată. Datele inserate s-au propagat în tabelele principale (tabela principală și tabla fragment corectă), după cum se poate observa în urma selectului rulat.



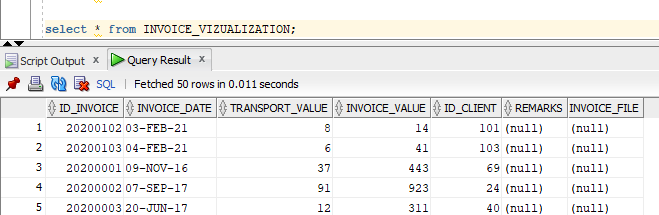
1. transparență pentru fragmentele orizontale

Asemănător punctului a), realizăm și view-ul pentru fragmentele orizontale

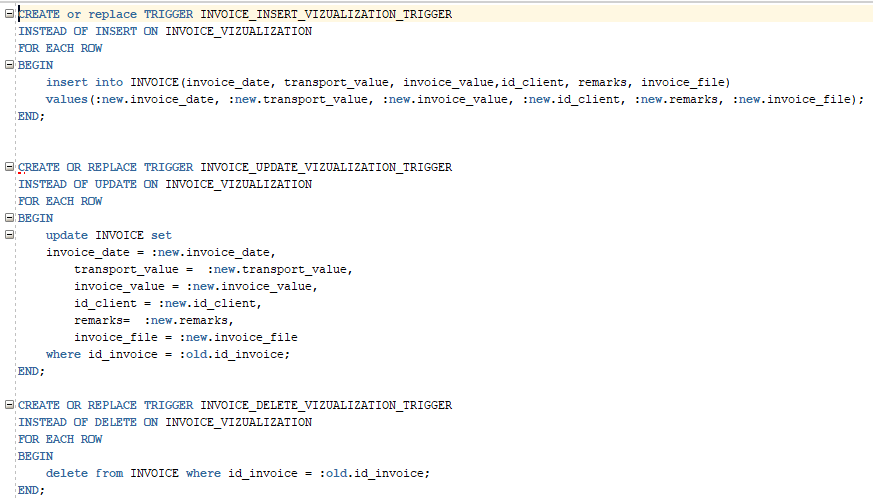
Se creaza vizualizarea



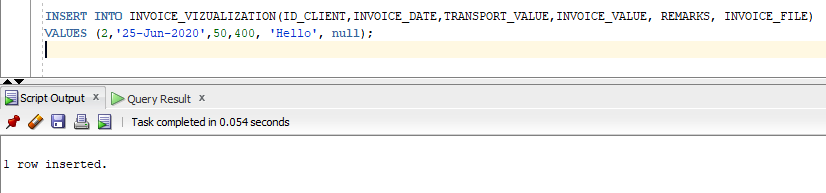
Testare vizualizare

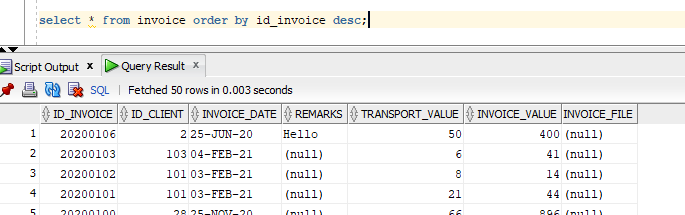


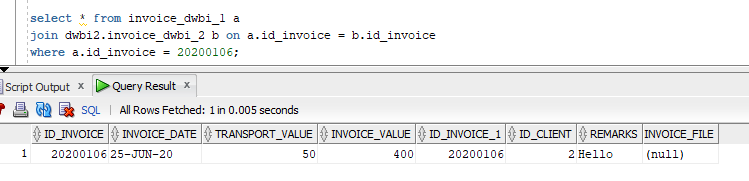
Creăm trei triggeri INSTEAD OF pentru fiecare din comenzile Insert/Update/Delete

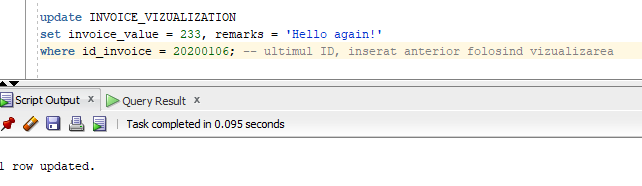


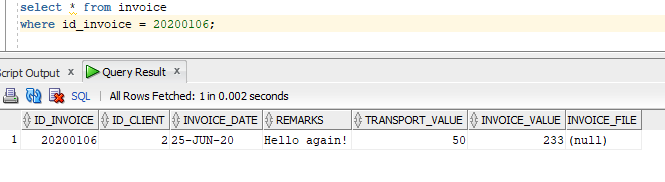
Verificare Triggeri instead of

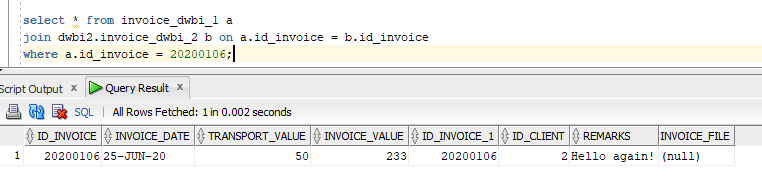


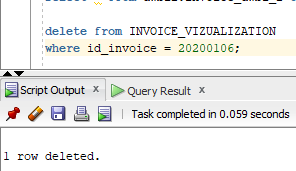


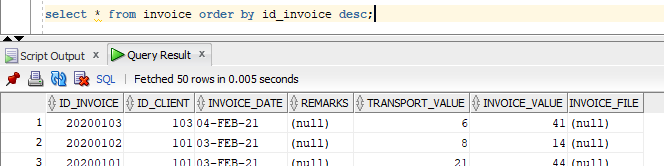


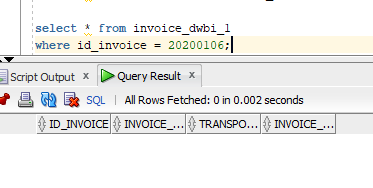


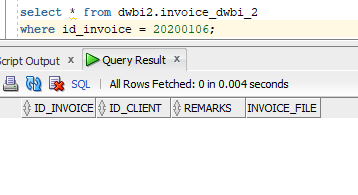












1. **Asigurarea sincronizării datelor pentru relațiile replicate**

Sincronizarea datelor este asigurată cu ajutorul triggerilor creați anterior la punctele 3 și 4.

**6.**

**A.i. Pentru unicitatea locala am folosit constrangeri**

alter table DWBI1.DRIVER

add constraint DRIVER\_CNP\_UK UNIQUE(CNP);

**A.ii Pentru unicitate glabala pe fragmente orizontale folosim trigeri**

CREATE OR REPLACE TRIGGER PK\_VECHICLE\_MODEL

BEFORE INSERT ON DWBI1.VEHICLE\_MODEL

FOR EACH ROW

DECLARE

nr NUMBER (6);

BEGIN

SELECT COUNT(\*) INTO nr

FROM DWBI1.VEHICLE

WHERE ID\_VEHICLE\_MODEL=:NEW.ID\_VEHICLE\_MODEL;

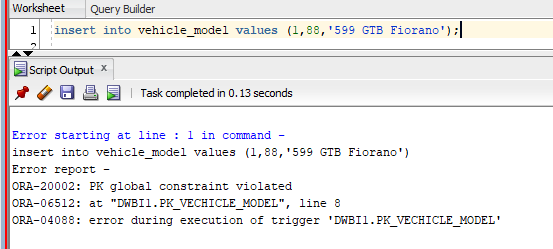
IF (nr<>0) THEN

RAISE\_APPLICATION\_ERROR (-20002,

'PK global constraint violated');

END IF;

END;



**B. Cheie primara la nivel local**

alter table DRIVER

add constraint PK\_ID\_DRIVER primary key(ID\_DRIVER);

Cheie primara la nivel global

CREATE OR REPLACE TRIGGER PK\_VECHICLE\_MODEL

BEFORE INSERT ON DWBI1.VEHICLE\_MODEL

FOR EACH ROW

DECLARE

nr NUMBER (6);

BEGIN

SELECT COUNT(\*) INTO nr

FROM DWBI1.VEHICLE

WHERE ID\_VEHICLE\_MODEL=:NEW.ID\_VEHICLE\_MODEL;

IF (nr>0) THEN

RAISE\_APPLICATION\_ERROR (-20002,

'PK global constraint violated');

END IF;

END;

**C. Cheia externa la nivel local se face prin constrangere**

alter table DWBI1.INVOICE

add constraint FK\_ID\_CLIENT foreign key (ID\_CLIENT)

references DWBI1.CLIENT(ID\_CLIENT);

**Cheia externa la nivel global se face prin triger**

CREATE OR REPLACE TRIGGER FK\_GLOBAL\_INVOICE

BEFORE INSERT ON DWBI2.INVOICE\_DWBI\_2

FOR EACH ROW

DECLARE

nr NUMBER (6);

BEGIN

SELECT COUNT(\*) INTO nr

FROM DWBI1.CLIENT

WHERE ID\_CLIENT=:NEW.ID\_CLIENT;

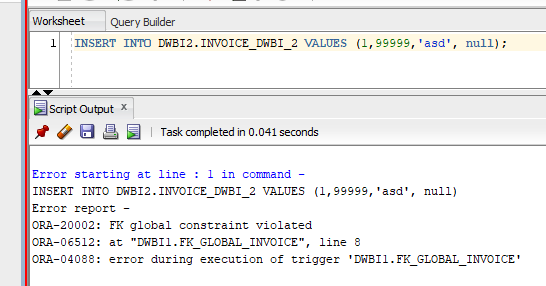
IF (nr=0) THEN

RAISE\_APPLICATION\_ERROR (-20002,

'FK global constraint violated');

END IF;

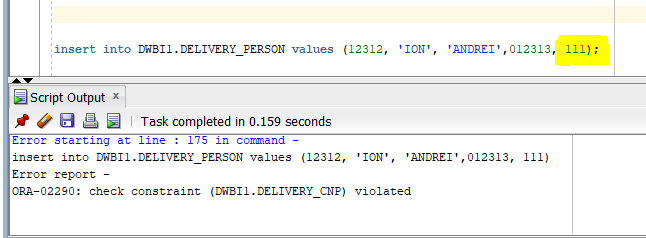
END;



**D. Pentru validari folosim constrangeri**

alter table DWBI1.DRIVER

add constraint DRIVER\_CNP check(length(CNP)=13);



alter table DWBI1.VEHICLE

add constraint VEHICLE\_VIN check(length(VIN)>=11 AND length(VIN)<=17);

alter table DWBI1.VEHICLE

add constraint VEHICLE\_VIN\_UK UNIQUE(VIN);