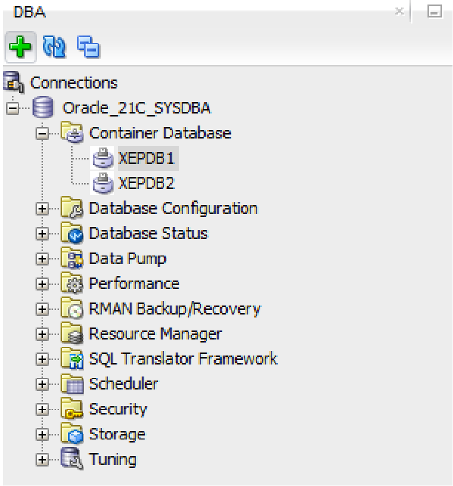
1. Crearea bazelor de date și a utilizatorilor

Pentru realizarea proiectului a fost create două baze de date de tip pluggable numite XEPDB1 și XEPDB2.

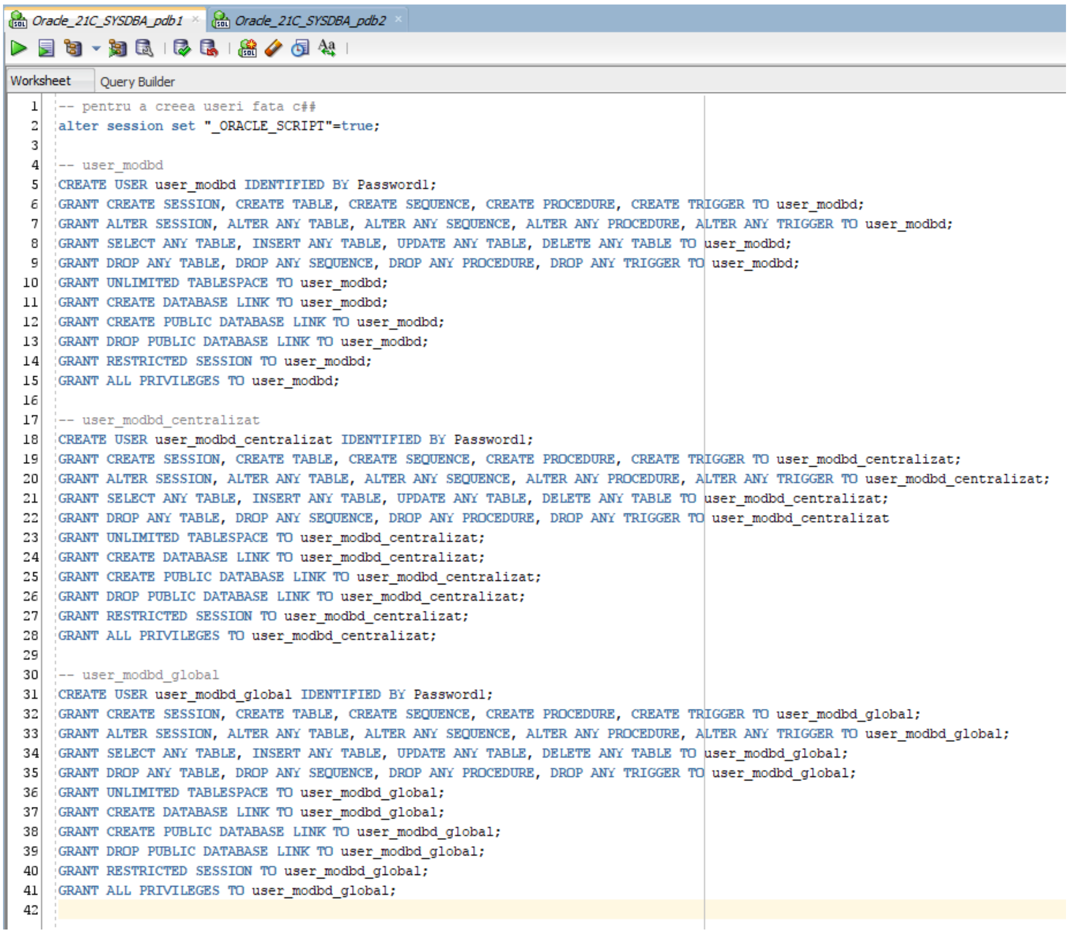


În baza XEPDB1 au fost creați utilizatori user_modbd, user_modbd_centralizat și user_modbd_global, iar în XEPDB2 a fost creat utilizatorul user_modbd.

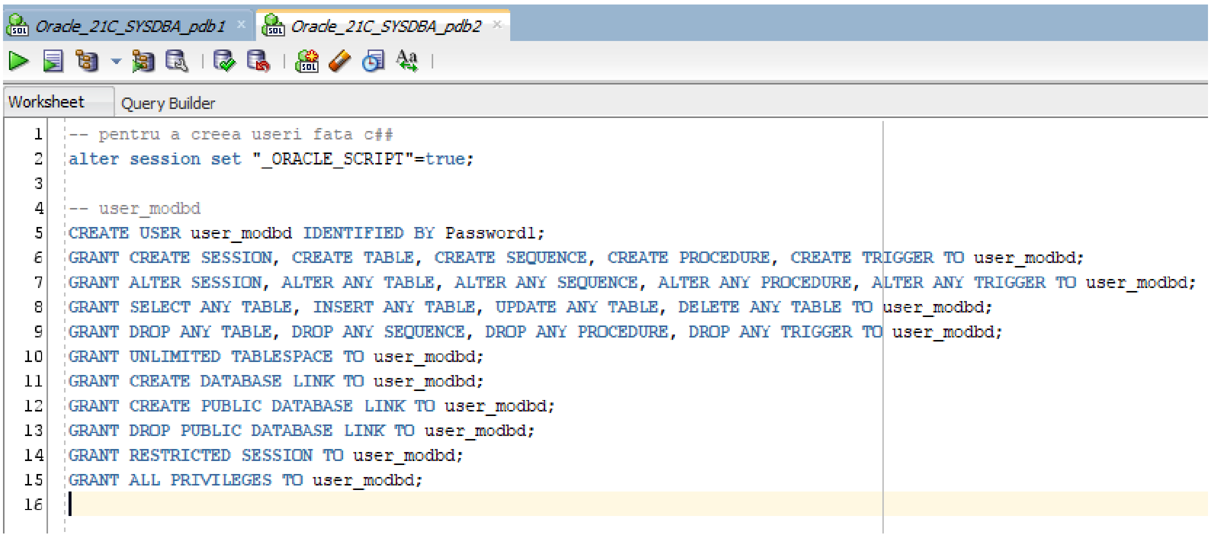
```
-- pentru a creea useri fata c##
alter session set "_ORACLE_SCRIPT"=true;

-- user_modbd
CREATE USER user_modbd IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE,
CREATE TRIGGER TO user_modbd;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY
PROCEDURE, ALTER ANY TRIGGER TO user_modbd;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO
user_modbd;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY
TRIGGER TO user_modbd;
GRANT UNLIMITED TABLESPACE TO user_modbd;
GRANT CREATE DATABASE LINK TO user_modbd;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd;
GRANT RESTRICTED SESSION TO user_modbd;
GRANT ALL PRIVILEGES TO user_modbd;

-- user_modbd_centralizat
CREATE USER user_modbd_centralizat IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE,
CREATE TRIGGER TO user_modbd_centralizat;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY
PROCEDURE, ALTER ANY TRIGGER TO user_modbd_centralizat;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO
user_modbd_centralizat;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY
TRIGGER TO user_modbd_centralizat
GRANT UNLIMITED TABLESPACE TO user_modbd_centralizat;
GRANT CREATE DATABASE LINK TO user_modbd_centralizat;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd_centralizat;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd_centralizat;
```

GRANT RESTRICTED SESSION TO user_modbd_centralizat;
GRANT ALL PRIVILEGES TO user_modbd_centralizat;

-- user_modbd_global
CREATE USER user_modbd_global IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO user_modbd_global;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY PROCEDURE, ALTER ANY TRIGGER TO user_modbd_global;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO user_modbd_global;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY TRIGGER TO user_modbd_global;
GRANT UNLIMITED TABLESPACE TO user_modbd_global;
GRANT CREATE DATABASE LINK TO user_modbd_global;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd_global;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd_global;
GRANT RESTRICTED SESSION TO user_modbd_global;
GRANT ALL PRIVILEGES TO user_modbd_global;



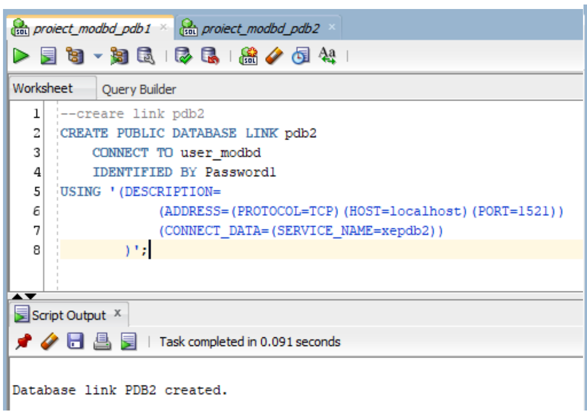
```
-- pentru a creea useri fata c##
alter session set "_ORACLE_SCRIPT"=true;

-- user_modbd
CREATE USER user_modbd IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO user_modbd;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY PROCEDURE, ALTER ANY TRIGGER TO user_modbd;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO user_modbd;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY TRIGGER TO user_modbd;
GRANT UNLIMITED TABLESPACE TO user_modbd;
GRANT CREATE DATABASE LINK TO user_modbd;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd;
GRANT RESTRICTED SESSION TO user_modbd;
GRANT ALL PRIVILEGES TO user_modbd;

-- user_modbd_centralizat
CREATE USER user_modbd_centralizat IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO user_modbd_centralizat;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY PROCEDURE, ALTER ANY TRIGGER TO user_modbd_centralizat;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO user_modbd_centralizat;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY TRIGGER TO user_modbd_centralizat
GRANT UNLIMITED TABLESPACE TO user_modbd_centralizat;
GRANT CREATE DATABASE LINK TO user_modbd_centralizat;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd_centralizat;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd_centralizat;
GRANT RESTRICTED SESSION TO user_modbd_centralizat;
GRANT ALL PRIVILEGES TO user_modbd_centralizat;

-- user_modbd_global
CREATE USER user_modbd_global IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO user_modbd_global;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY PROCEDURE, ALTER ANY TRIGGER TO user_modbd_global;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO user_modbd_global;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY TRIGGER TO user_modbd_global;
GRANT UNLIMITED TABLESPACE TO user_modbd_global;
GRANT CREATE DATABASE LINK TO user_modbd_global;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd_global;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd_global;
GRANT RESTRICTED SESSION TO user_modbd_global;
GRANT ALL PRIVILEGES TO user_modbd_global;
```



```
-- pentru a creea useri fata c##
alter session set "_ORACLE_SCRIPT"=true;

-- user_modbd
CREATE USER user_modbd IDENTIFIED BY Password1;
GRANT CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO user_modbd;
GRANT ALTER SESSION, ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY PROCEDURE, ALTER ANY TRIGGER TO user_modbd;
GRANT SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE TO user_modbd;
GRANT DROP ANY TABLE, DROP ANY SEQUENCE, DROP ANY PROCEDURE, DROP ANY TRIGGER TO user_modbd;
GRANT UNLIMITED TABLESPACE TO user_modbd;
GRANT CREATE DATABASE LINK TO user_modbd;
GRANT CREATE PUBLIC DATABASE LINK TO user_modbd;
GRANT DROP PUBLIC DATABASE LINK TO user_modbd;
GRANT RESTRICTED SESSION TO user_modbd;
GRANT ALL PRIVILEGES TO user_modbd;
```

Apoi am ne-am conectat ca user_modbd la pdb1 și pdb2 și am creat link-urile dintre acestea

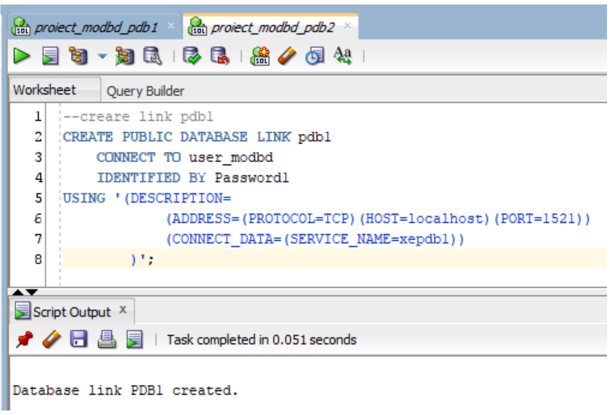
```
--creare link pdb2
CREATE PUBLIC DATABASE LINK pdb2
   CONNECT TO user_modbd
   IDENTIFIED BY Password1
USING '(DESCRIPTION=
         (ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
         (CONNECT_DATA=(SERVICE_NAME=xepdb2))
      )';
--creare link pdb1
CREATE PUBLIC DATABASE LINK pdb1
   CONNECT TO user_modbd
   IDENTIFIED BY Password1
USING '(DESCRIPTION=
         (ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
         (CONNECT_DATA=(SERVICE_NAME=xepdb1))
      )';
```



```
--creare link pdb2
CREATE PUBLIC DATABASE LINK pdb2
    CONNECT TO user_modbd
    IDENTIFIED BY Password1
USING '(DESCRIPTION=
            (ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
            (CONNECT_DATA=(SERVICE_NAME=xepdb2))
        )';
```

Task completed in 0.091 seconds

```
Database link PDB2 created.
```



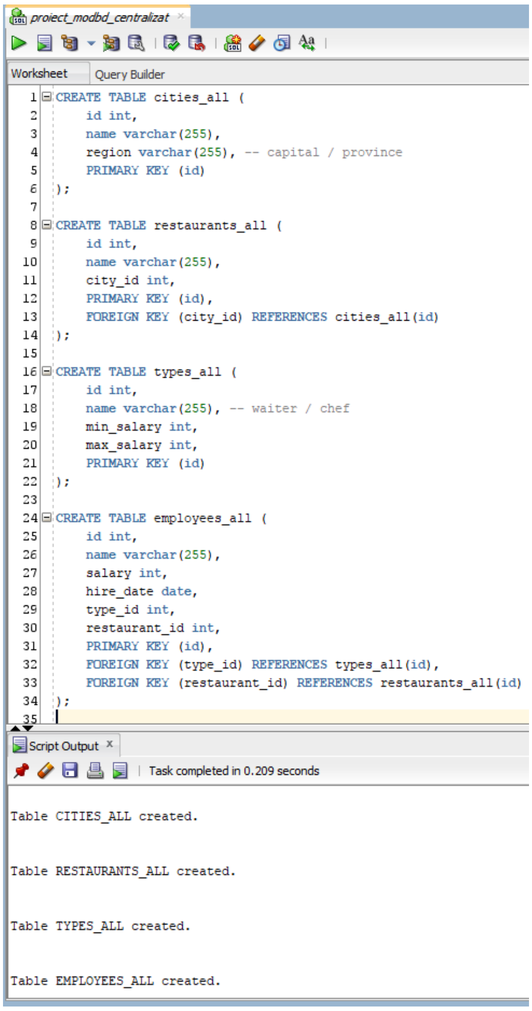
```
--creare link pdb1
CREATE PUBLIC DATABASE LINK pdb1
    CONNECT TO user_modbd
    IDENTIFIED BY Password1
USING '(DESCRIPTION=
            (ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
            (CONNECT_DATA=(SERVICE_NAME=xepdb1))
        )';
```

Task completed in 0.051 seconds

```
Database link PDB1 created.
```

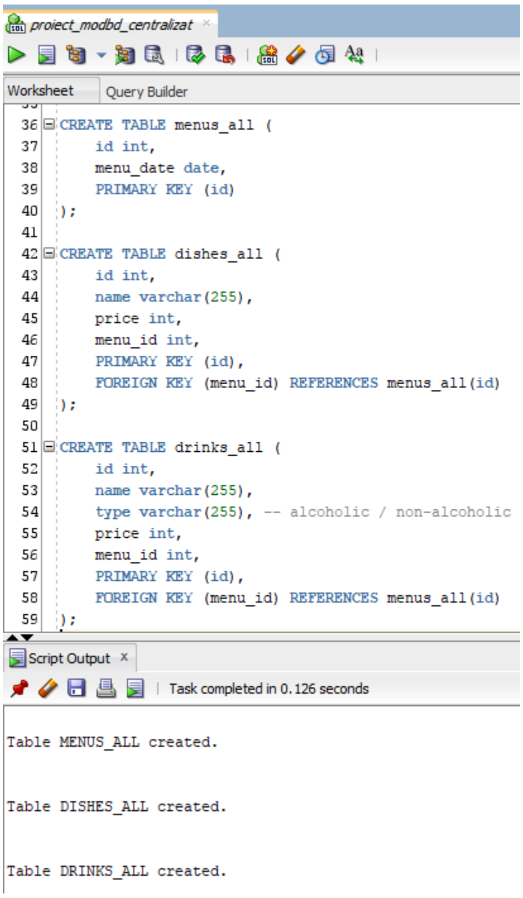
Următorul pas a fost crearea tabelelor în baza de date centralizat.

```
--creare bd centralizat
CREATE TABLE cities_all (
   id int,
   name varchar(255),
   region varchar(255), -- capital / province
   PRIMARY KEY (id)
);
CREATE TABLE restaurants_all (
   id int,
   name varchar(255),
   city_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (city_id) REFERENCES cities_all(id)
);
CREATE TABLE types_all (
   id int,
   name varchar(255), -- waiter / chef
   min_salary int,
   max_salary int,
   PRIMARY KEY (id)
);
CREATE TABLE employees_all (
   id int,
   name varchar(255),
   salary int,
   hire_date date,
   type_id int,
   restaurant_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (type_id) REFERENCES types_all(id),
   FOREIGN KEY (restaurant_id) REFERENCES restaurants_all(id)
);
CREATE TABLE menus_all (
   id int,
   menu_date date,
   PRIMARY KEY (id)
);
CREATE TABLE dishes_all (
   id int,
   name varchar(255),
   price int,
   menu_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (menu_id) REFERENCES menus_all(id)
);
CREATE TABLE drinks_all (
   id int,
   name varchar(255),
   type varchar(255), -- alcoholic / non-alcoholic
   price int,
   menu_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (menu_id) REFERENCES menus_all(id)
);
CREATE TABLE orders_all (
   id int,
   order_date date,
   total int,
   tip int,
   waiter_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (waiter_id) REFERENCES employees_all(id)
);
CREATE TABLE orders_drinks_all (
   id int,
   count int,
   price int,
   order_id int,
   drink_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (order_id) REFERENCES orders_all(id),
   FOREIGN KEY (drink_id) REFERENCES drinks_all(id)
);
CREATE TABLE chefs_orders_dishes_all (
   id int,
   count int,
   price int,
   chef_id int,
   order_id int,
   dish_id int,
   PRIMARY KEY (id),
   FOREIGN KEY (chef_id) REFERENCES employees_all(id),
   FOREIGN KEY (order_id) REFERENCES orders_all(id),
   FOREIGN KEY (dish_id) REFERENCES dishes_all(id)
);
```



```
CREATE TABLE cities_all (
    id int,
    name varchar(255),
    region varchar(255), -- capital / province
    PRIMARY KEY (id)
);

CREATE TABLE restaurants_all (
    id int,
    name varchar(255),
    city_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (city_id) REFERENCES cities_all(id)
);

CREATE TABLE types_all (
    id int,
    name varchar(255), -- waiter / chef
    min_salary int,
    max_salary int,
    PRIMARY KEY (id)
);

CREATE TABLE employees_all (
    id int,
    name varchar(255),
    salary int,
    hire_date date,
    type_id int,
    restaurant_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (type_id) REFERENCES types_all(id),
    FOREIGN KEY (restaurant_id) REFERENCES restaurants_all(id)
);
```

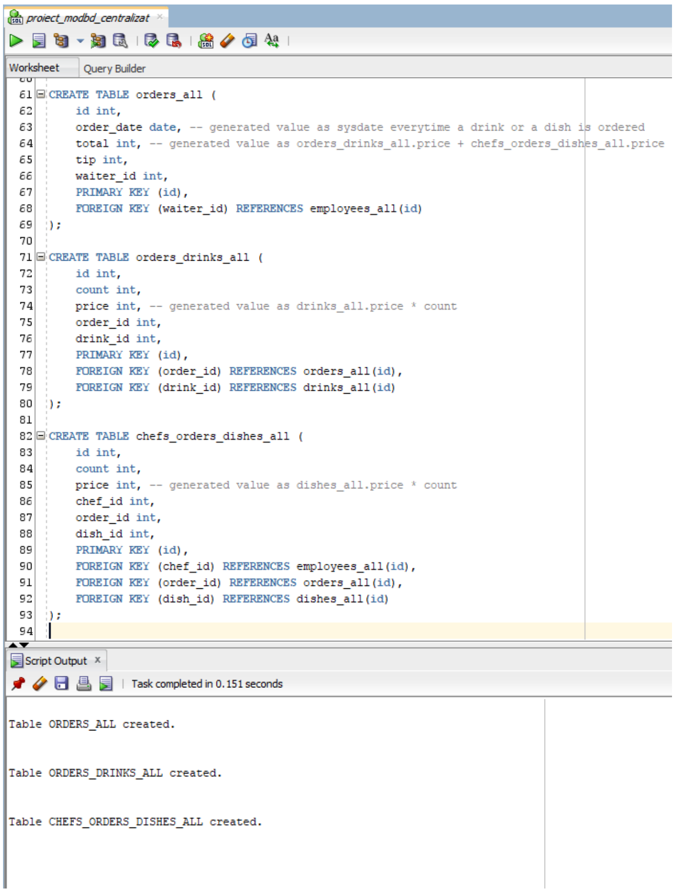
```
Table CITIES_ALL created.


Table RESTAURANTS_ALL created.


Table TYPES_ALL created.


Table EMPLOYEES_ALL created.
```



```sql
CREATE TABLE menus_all (
    id int,
    menu_date date,
    PRIMARY KEY (id)
);

CREATE TABLE dishes_all (
    id int,
    name varchar(255),
    price int,
    menu_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (menu_id) REFERENCES menus_all(id)
);

CREATE TABLE drinks_all (
    id int,
    name varchar(255),
    type varchar(255), -- alcoholic / non-alcoholic
    price int,
    menu_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (menu_id) REFERENCES menus_all(id)
);
```

Task completed in 0.126 seconds

```
Table MENUS_ALL created.


Table DISHES_ALL created.


Table DRINKS_ALL created.
```



```sql
CREATE TABLE orders_all (
    id int,
    order_date date, -- generated value as sysdate everytime a drink or a dish is ordered
    total int, -- generated value as orders_drinks_all.price + chefs_orders_dishes_all.price
    tip int,
    waiter_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (waiter_id) REFERENCES employees_all(id)
);

CREATE TABLE orders_drinks_all (
    id int,
    count int,
    price int, -- generated value as drinks_all.price * count
    order_id int,
    drink_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (order_id) REFERENCES orders_all(id),
    FOREIGN KEY (drink_id) REFERENCES drinks_all(id)
);

CREATE TABLE chefs_orders_dishes_all (
    id int,
    count int,
    price int, -- generated value as dishes_all.price * count
    chef_id int,
    order_id int,
    dish_id int,
    PRIMARY KEY (id),
    FOREIGN KEY (chef_id) REFERENCES employees_all(id),
    FOREIGN KEY (order_id) REFERENCES orders_all(id),
    FOREIGN KEY (dish_id) REFERENCES dishes_all(id)
);
```

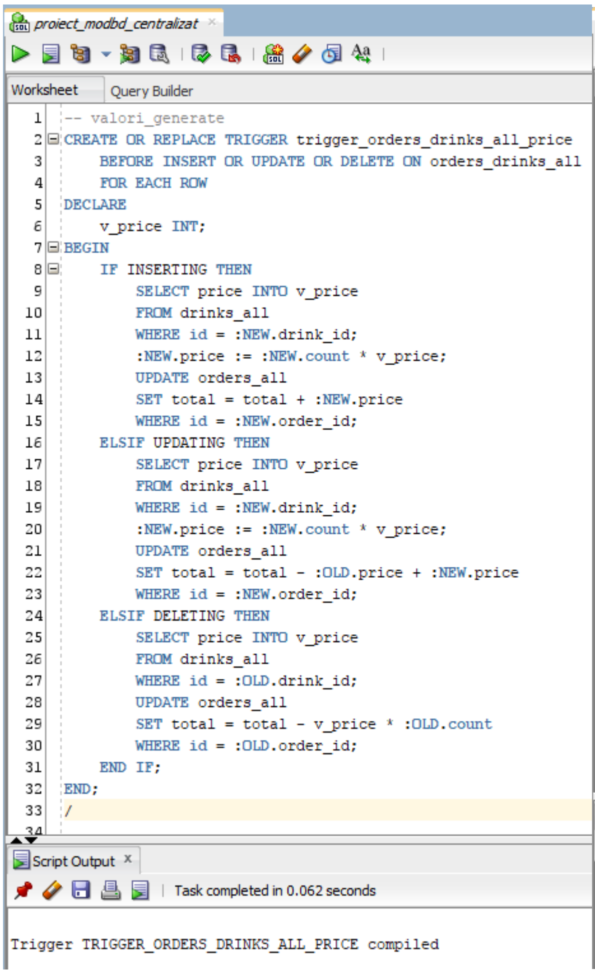
Task completed in 0.151 seconds

```
Table ORDERS_ALL created.


Table ORDERS_DRINKS_ALL created.


Table CHEFS_ORDERS_DISHES_ALL created.
```

Pentru valorile generate am folosit următoarele triggere:

```sql
-- valori_generate
CREATE OR REPLACE TRIGGER trigger_orders_drinks_all_price
   BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_all
   FOR EACH ROW
DECLARE
   v_price INT;
BEGIN
   IF INSERTING THEN
      SELECT price INTO v_price
      FROM drinks_all
      WHERE id = :NEW.drink_id;
      :NEW.price := :NEW.count * v_price;
      UPDATE orders_all
      SET total = total + :NEW.price
      WHERE id = :NEW.order_id;
   ELSIF UPDATING THEN
      SELECT price INTO v_price
      FROM drinks_all
      WHERE id = :NEW.drink_id;
      :NEW.price := :NEW.count * v_price;
      UPDATE orders_all
      SET total = total - :OLD.price + :NEW.price
      WHERE id = :NEW.order_id;
   ELSIF DELETING THEN
      SELECT price INTO v_price
      FROM drinks_all
      WHERE id = :OLD.drink_id;
      UPDATE orders_all
      SET total = total - v_price * :OLD.count
```

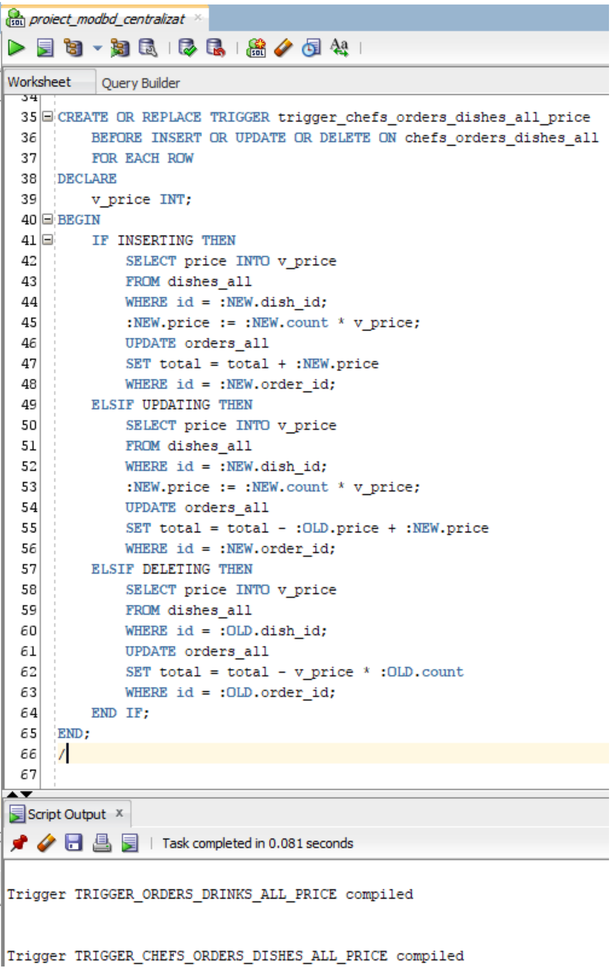
```
        WHERE id = :OLD.order_id;
    END IF;
END;
/
CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_all_price
    BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_all
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :OLD.dish_id;
        UPDATE orders_all
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
CREATE OR REPLACE TRIGGER trigger_orders_all_date
    BEFORE INSERT OR UPDATE ON orders_all
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/
```



```
-- valori_generate
CREATE OR REPLACE TRIGGER trigger_orders_drinks_all_price
    BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_all
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM drinks_all
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM drinks_all
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM drinks_all
        WHERE id = :OLD.drink_id;
        UPDATE orders_all
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```

Task completed in 0.062 seconds

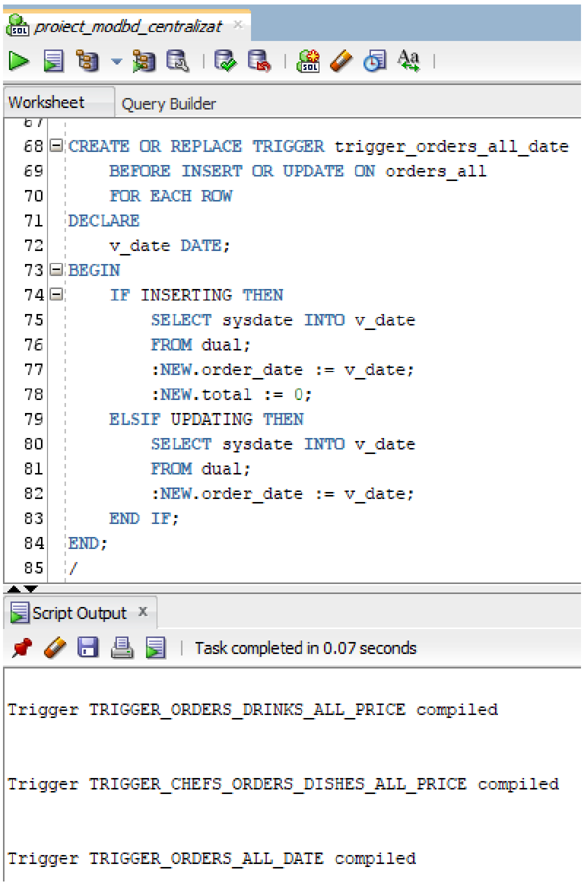
```
Trigger TRIGGER_ORDERS_DRINKS_ALL_PRICE compiled
```



```
CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_all_price
    BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_all
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_all
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM dishes_all
        WHERE id = :OLD.dish_id;
        UPDATE orders_all
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```

Task completed in 0.081 seconds

```
Trigger TRIGGER_ORDERS_DRINKS_ALL_PRICE compiled


Trigger TRIGGER_CHEFS_ORDERS_DISHES_ALL_PRICE compiled
```



```
CREATE OR REPLACE TRIGGER trigger_orders_all_date
    BEFORE INSERT OR UPDATE ON orders_all
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/
```

Task completed in 0.07 seconds

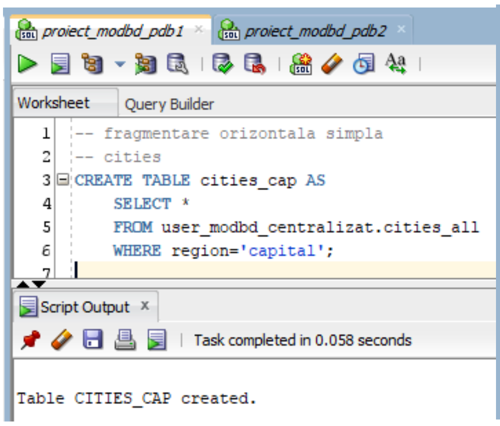
```
Trigger TRIGGER_ORDERS_DRINKS_ALL_PRICE compiled


Trigger TRIGGER_CHEFS_ORDERS_DISHES_ALL_PRICE compiled


Trigger TRIGGER_ORDERS_ALL_DATE compiled
```

2. Crearea relațiilor și a fragmentelor

a. Fragmentare orizontală primară

Fragmentarea orizontală primară am ales tabela cities_all, in funcție de atributul region. Astfel, în PBD1 vom avea cities_cap, unde vor fi stocate orașele capitale, iar in PDB2 în cities_prov vor fi stocate orașele din provincie.

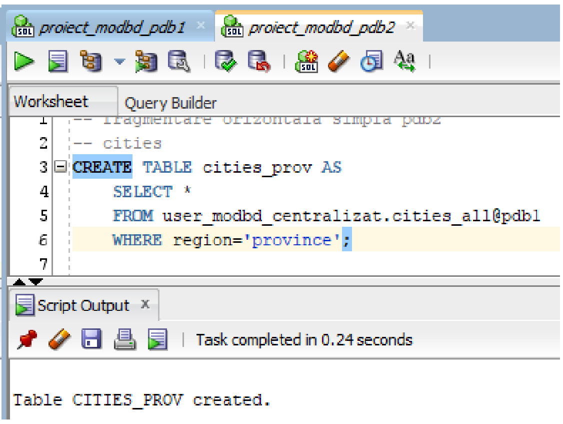
```
CREATE TABLE cities_cap AS
   SELECT *
   FROM user_modbd_centralizat.cities_all
   WHERE region='capital';

CREATE TABLE cities_prov AS
   SELECT *
   FROM user_modbd_centralizat.cities_all@pdb1
   WHERE region='province';
```



```
-- fragmentare orizontala simpla
-- cities
CREATE TABLE cities_cap AS
    SELECT *
    FROM user_modbd_centralizat.cities_all
    WHERE region='capital';
```

Task completed in 0.058 seconds

```
Table CITIES_CAP created.
```



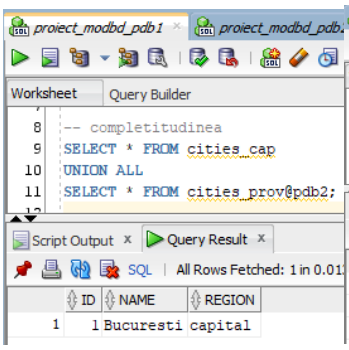
```
-- fragmentare orizontala simpla pdb2
-- cities
CREATE TABLE cities_prov AS
    SELECT *
    FROM user_modbd_centralizat.cities_all@pdb1
    WHERE region='province';
```

Task completed in 0.24 seconds

```
Table CITIES_PROV created.
```

Pentru testarea corectitudinii fragmentării, am realizat operațiile de completitudine, reconstrucție și disjuncție

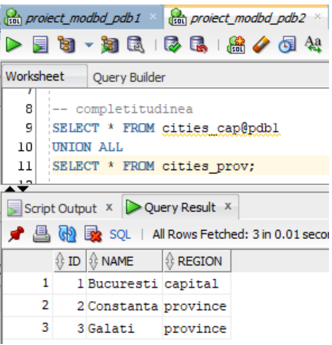
```
-- completitudinea
SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2;

SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov;
```



```
-- completitudinea
SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2;
```

All Rows Fetched: 1 in 0.01

	ID	NAME	REGION
1	1	Bucuresti	capital

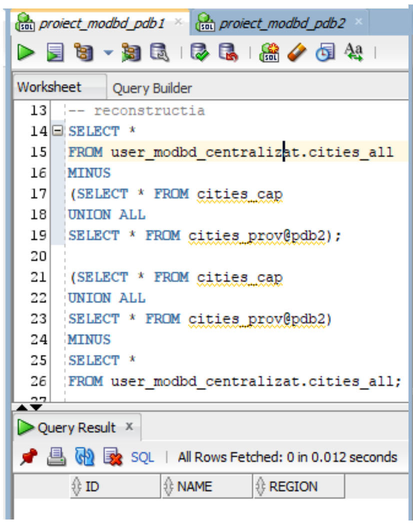


```
-- completitudinea
SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov;
```

All Rows Fetched: 3 in 0.01 seco

	ID	NAME	REGION
1	1	Bucuresti	capital
2	2	Constanta	province
3	3	Galati	province

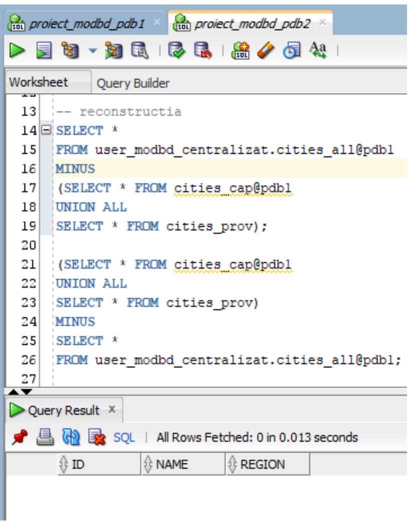
```
-- reconstructia
SELECT *
FROM user_modbd_centralizat.cities_all
MINUS
(SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2);

(SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2)
MINUS
SELECT *
FROM user_modbd_centralizat.cities_all;

SELECT *
FROM user_modbd_centralizat.cities_all@pdb1
MINUS
(SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov);

(SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov)
MINUS
SELECT *
FROM user_modbd_centralizat.cities_all@pdb1;
```



```
-- reconstructia
SELECT *
FROM user_modbd_centralizat.cities_all
MINUS
(SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2);

(SELECT * FROM cities_cap
UNION ALL
SELECT * FROM cities_prov@pdb2)
MINUS
SELECT *
FROM user_modbd_centralizat.cities_all;
```

All Rows Fetched: 0 in 0.012 seconds

ID	NAME	REGION

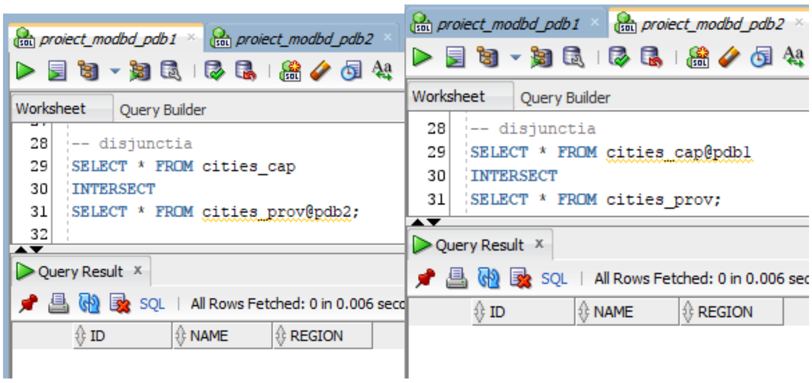


```
-- reconstructia
SELECT *
FROM user_modbd_centralizat.cities_all@pdb1
MINUS
(SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov);

(SELECT * FROM cities_cap@pdb1
UNION ALL
SELECT * FROM cities_prov)
MINUS
SELECT *
FROM user_modbd_centralizat.cities_all@pdb1;
```

All Rows Fetched: 0 in 0.013 seconds

ID	NAME	REGION

```
-- disjunctia
SELECT * FROM cities_cap
INTERSECT
SELECT * FROM cities_prov@pdb2;

SELECT * FROM cities_cap@pdb1
INTERSECT
SELECT * FROM cities_prov;
```



```
-- disjunctia
SELECT * FROM cities_cap
INTERSECT
SELECT * FROM cities_prov@pdb2;
```

All Rows Fetched: 0 in 0.006 seco

ID	NAME	REGION

```
-- disjunctia
SELECT * FROM cities_cap@pdb1
INTERSECT
SELECT * FROM cities_prov;
```

All Rows Fetched: 0 in 0.006 sec

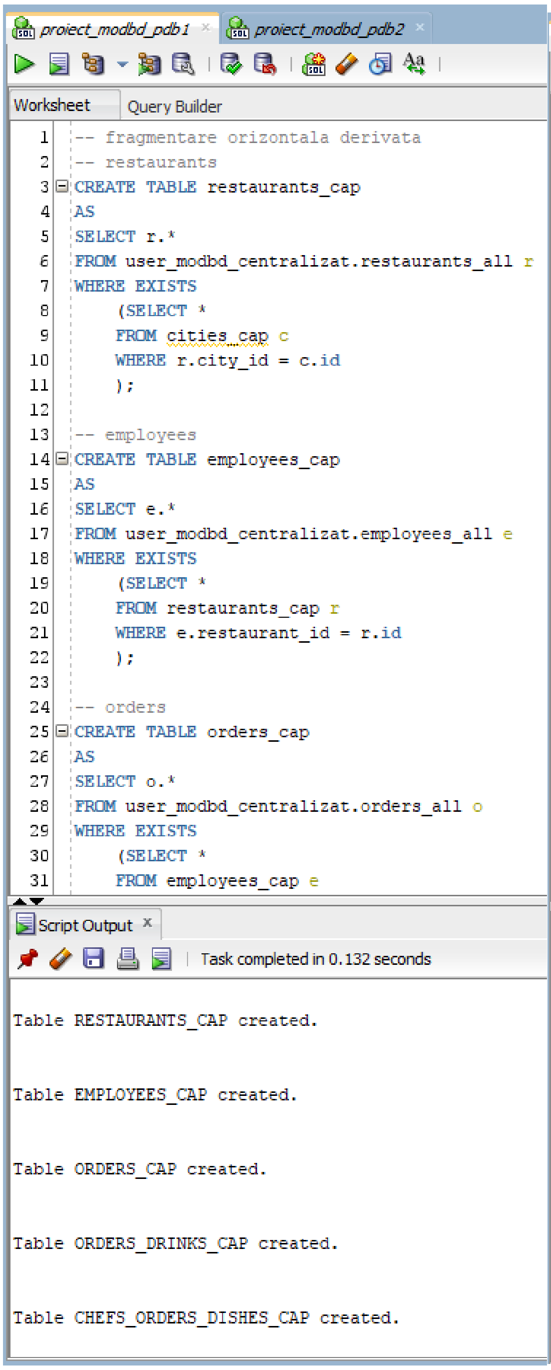
ID	NAME	REGION

b. Fragmentare orizontală derivată

Fragmentarea orizonatală derivată a fost realizată asupra tabelelor restaurants, employees, orders, orders_drinks și chef_orders_dishes.

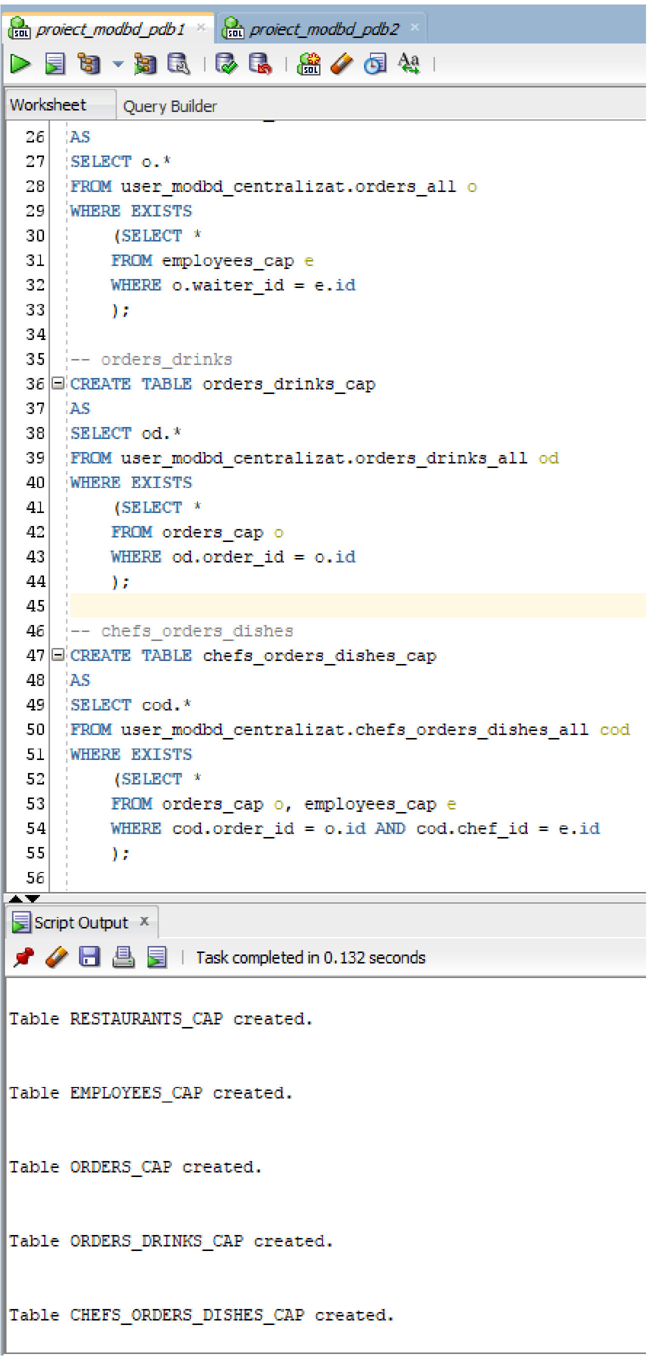
```
-- fragmentare orizontala derivata pdb1
-- restaurants
CREATE TABLE restaurants_cap
AS
SELECT r.*
FROM user_modbd_centralizat.restaurants_all r
WHERE EXISTS
   (SELECT *
   FROM cities_cap c
   WHERE r.city_id = c.id
   );

-- employees
CREATE TABLE employees_cap
AS
SELECT e.*
FROM user_modbd_centralizat.employees_all e
WHERE EXISTS
   (SELECT *
   FROM restaurants_cap r
   WHERE e.restaurant_id = r.id
   );

-- orders
CREATE TABLE orders_cap
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all o
WHERE EXISTS
   (SELECT *
   FROM employees_cap e
   WHERE o.waiter_id = e.id
   );

-- orders_drinks
CREATE TABLE orders_drinks_cap
AS
SELECT od.*
FROM user_modbd_centralizat.orders_drinks_all od
WHERE EXISTS
```

```
   (SELECT *
   FROM orders_cap o
   WHERE od.order_id = o.id
   );

-- chefs_orders_dishes
CREATE TABLE chefs_orders_dishes_cap
AS
SELECT cod.*
FROM user_modbd_centralizat.chefs_orders_dishes_all cod
WHERE EXISTS
   (SELECT *
   FROM orders_cap o, employees_cap e
   WHERE cod.order_id = o.id AND cod.chef_id = e.id
   );
```



```
-- fragmentare orizontala derivata
-- restaurants
CREATE TABLE restaurants_cap
AS
SELECT r.*
FROM user_modbd_centralizat.restaurants_all r
WHERE EXISTS
    (SELECT *
    FROM cities_cap c
    WHERE r.city_id = c.id
    );

-- employees
CREATE TABLE employees_cap
AS
SELECT e.*
FROM user_modbd_centralizat.employees_all e
WHERE EXISTS
    (SELECT *
    FROM restaurants_cap r
    WHERE e.restaurant_id = r.id
    );

-- orders
CREATE TABLE orders_cap
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all o
WHERE EXISTS
    (SELECT *
    FROM employees_cap e
```

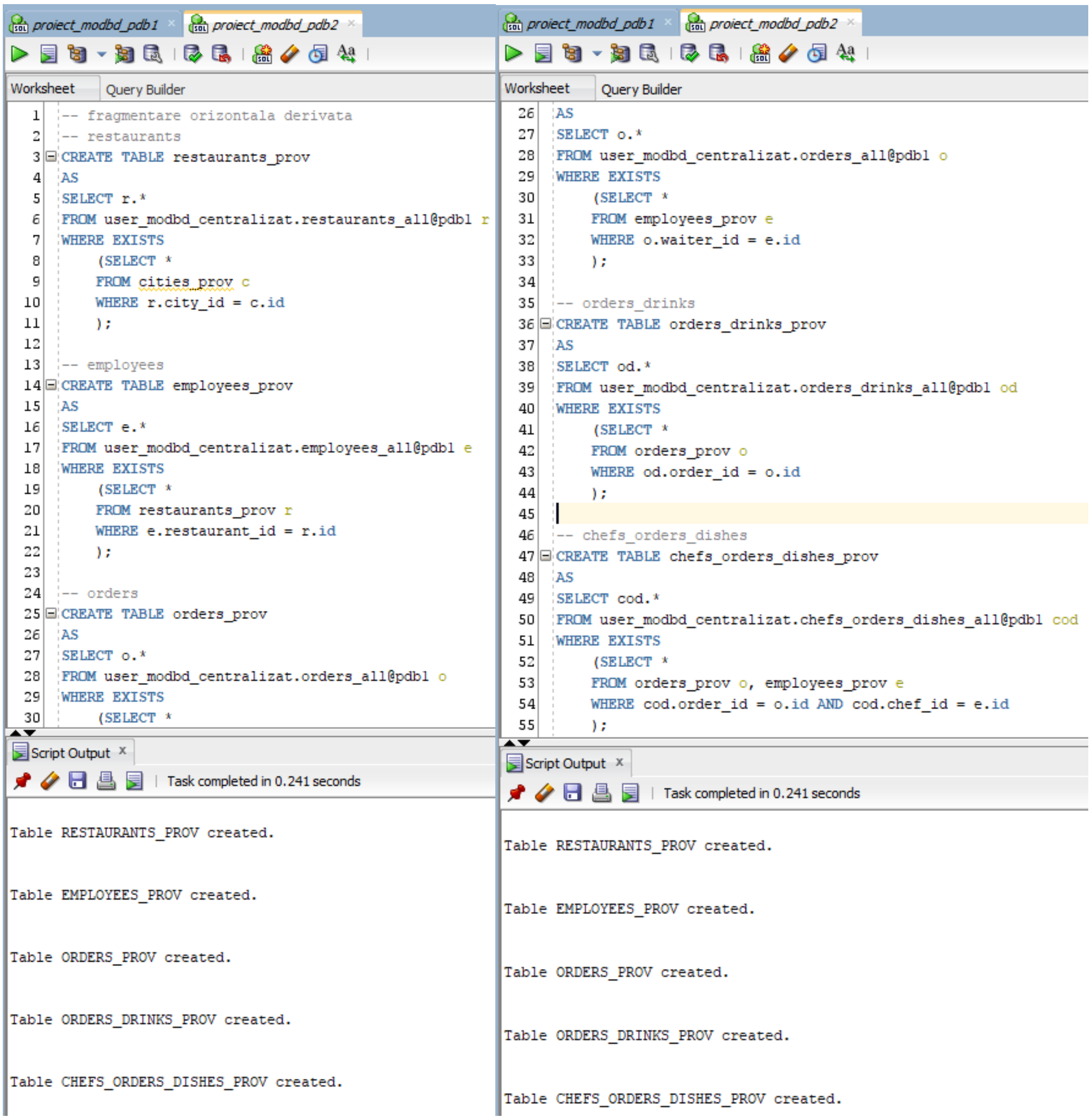
```
Table RESTAURANTS_CAP created.


Table EMPLOYEES_CAP created.


Table ORDERS_CAP created.


Table ORDERS_DRINKS_CAP created.


Table CHEFS_ORDERS_DISHES_CAP created.
```



```
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all o
WHERE EXISTS
    (SELECT *
    FROM employees_cap e
    WHERE o.waiter_id = e.id
    );

-- orders_drinks
CREATE TABLE orders_drinks_cap
AS
SELECT od.*
FROM user_modbd_centralizat.orders_drinks_all od
WHERE EXISTS
    (SELECT *
    FROM orders_cap o
    WHERE od.order_id = o.id
    );

-- chefs_orders_dishes
CREATE TABLE chefs_orders_dishes_cap
AS
SELECT cod.*
FROM user_modbd_centralizat.chefs_orders_dishes_all cod
WHERE EXISTS
    (SELECT *
    FROM orders_cap o, employees_cap e
    WHERE cod.order_id = o.id AND cod.chef_id = e.id
    );
```

```
Table RESTAURANTS_CAP created.


Table EMPLOYEES_CAP created.


Table ORDERS_CAP created.


Table ORDERS_DRINKS_CAP created.


Table CHEFS_ORDERS_DISHES_CAP created.
```

```
-- fragmentare orizontala derivata pdb2
-- restaurants
CREATE TABLE restaurants_prov
AS
SELECT r.*
FROM user_modbd_centralizat.restaurants_all@pdb1 r
WHERE EXISTS
   (SELECT *
   FROM cities_prov c
   WHERE r.city_id = c.id
   );

-- employees
CREATE TABLE employees_prov
AS
SELECT e.*
FROM user_modbd_centralizat.employees_all@pdb1 e
WHERE EXISTS
   (SELECT *
   FROM restaurants_prov r
   WHERE e.restaurant_id = r.id
   );

-- orders
CREATE TABLE orders_prov
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all@pdb1 o
WHERE EXISTS
   (SELECT *
   FROM employees_prov e
   WHERE o.waiter_id = e.id
   );

-- orders_drinks
CREATE TABLE orders_drinks_prov
AS
SELECT od.*
FROM user_modbd_centralizat.orders_drinks_all@pdb1 od
WHERE EXISTS
   (SELECT *
   FROM orders_prov o
   WHERE od.order_id = o.id
   );

-- chefs_orders_dishes
CREATE TABLE chefs_orders_dishes_prov
AS
SELECT cod.*
FROM user_modbd_centralizat.chefs_orders_dishes_all@pdb1 cod
WHERE EXISTS
   (SELECT *
   FROM orders_prov o, employees_prov e
   WHERE cod.order_id = o.id AND cod.chef_id = e.id
   );
```



```
-- fragmentare orizontala derivata
-- restaurants
CREATE TABLE restaurants_prov
AS
SELECT r.*
FROM user_modbd_centralizat.restaurants_all@pdb1 r
WHERE EXISTS
    (SELECT *
    FROM cities_prov c
    WHERE r.city_id = c.id
    );

-- employees
CREATE TABLE employees_prov
AS
SELECT e.*
FROM user_modbd_centralizat.employees_all@pdb1 e
WHERE EXISTS
    (SELECT *
    FROM restaurants_prov r
    WHERE e.restaurant_id = r.id
    );

-- orders
CREATE TABLE orders_prov
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all@pdb1 o
WHERE EXISTS
    (SELECT *
```

Task completed in 0.241 seconds

```
Table RESTAURANTS_PROV created.


Table EMPLOYEES_PROV created.


Table ORDERS_PROV created.


Table ORDERS_DRINKS_PROV created.


Table CHEFS_ORDERS_DISHES_PROV created.
```

```
AS
SELECT o.*
FROM user_modbd_centralizat.orders_all@pdb1 o
WHERE EXISTS
    (SELECT *
    FROM employees_prov e
    WHERE o.waiter_id = e.id
    );

-- orders_drinks
CREATE TABLE orders_drinks_prov
AS
SELECT od.*
FROM user_modbd_centralizat.orders_drinks_all@pdb1 od
WHERE EXISTS
    (SELECT *
    FROM orders_prov o
    WHERE od.order_id = o.id
    );

-- chefs_orders_dishes
CREATE TABLE chefs_orders_dishes_prov
AS
SELECT cod.*
FROM user_modbd_centralizat.chefs_orders_dishes_all@pdb1 cod
WHERE EXISTS
    (SELECT *
    FROM orders_prov o, employees_prov e
    WHERE cod.order_id = o.id AND cod.chef_id = e.id
    );
```

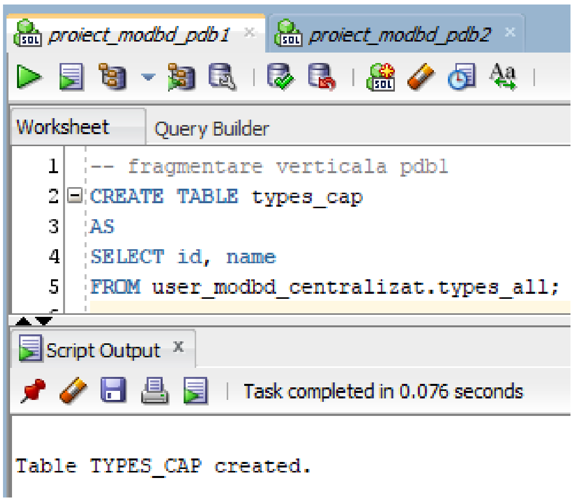
Task completed in 0.241 seconds

```
Table RESTAURANTS_PROV created.


Table EMPLOYEES_PROV created.


Table ORDERS_PROV created.


Table ORDERS_DRINKS_PROV created.


Table CHEFS_ORDERS_DISHES_PROV created.
```

c. Fragmentare verticală

Pentru fragmentarea verticală, am ales tabela types, motivul fiind doar unul demonstrativ. Astfel, în tabela types_cap vom avea coloanele id și name, iar in types_prov vor fi id, min_salary și max_salary

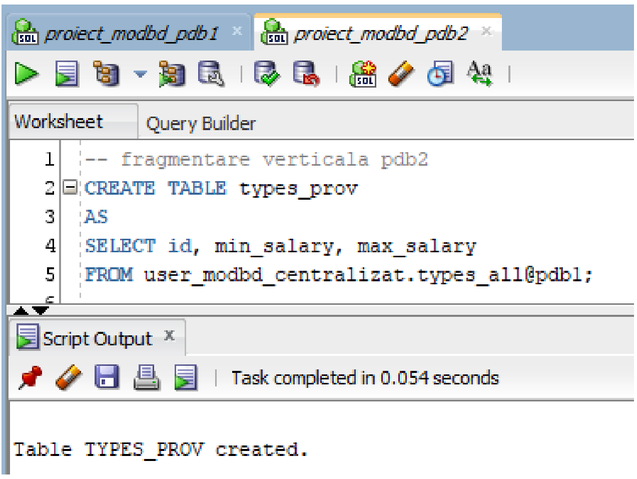
```
-- fragmentare verticala pdb1
CREATE TABLE types_cap
AS
SELECT id, name
FROM user_modbd_centralizat.types_all;
```



```
-- fragmentare verticala pdb1
CREATE TABLE types_cap
AS
SELECT id, name
FROM user_modbd_centralizat.types_all;
```

Task completed in 0.076 seconds

```
Table TYPES_CAP created.
```

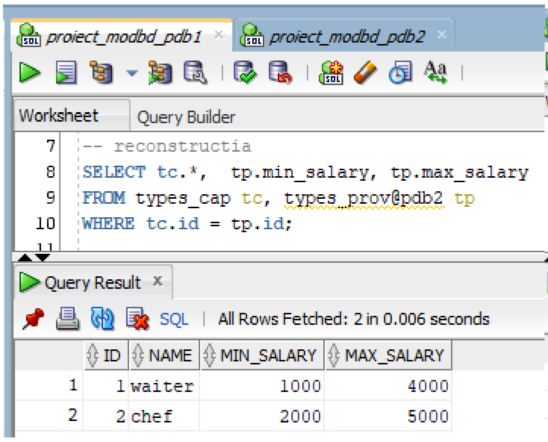


```
-- fragmentare verticala pdb2
CREATE TABLE types_prov
AS
SELECT id, min_salary, max_salary
FROM user_modbd_centralizat.types_all@pdb1;
```

Task completed in 0.054 seconds

```
Table TYPES_PROV created.
```

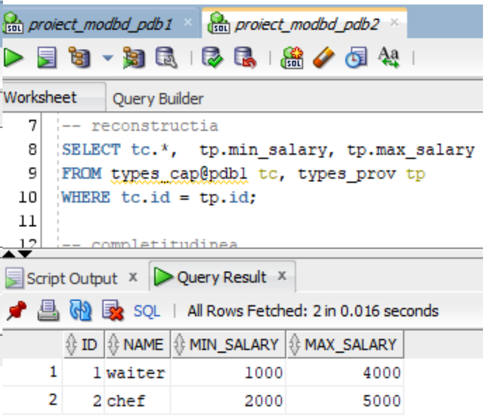
```
-- reconstructia
SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap tc, types_prov@pdb2 tp
WHERE tc.id = tp.id;

SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap@pdb1 tc, types_prov tp
WHERE tc.id = tp.id;
```



```
-- reconstructia
SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap tc, types_prov@pdb2 tp
WHERE tc.id = tp.id;
```

All Rows Fetched: 2 in 0.006 seconds

	ID	NAME	MIN_SALARY	MAX_SALARY
1	1	waiter	1000	4000
2	2	chef	2000	5000

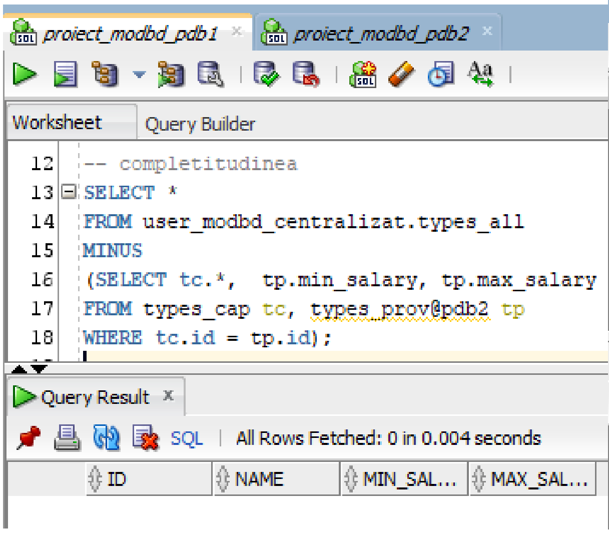


```
-- reconstructia
SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap@pdb1 tc, types_prov tp
WHERE tc.id = tp.id;

-- completitudinea
```

All Rows Fetched: 2 in 0.016 seconds

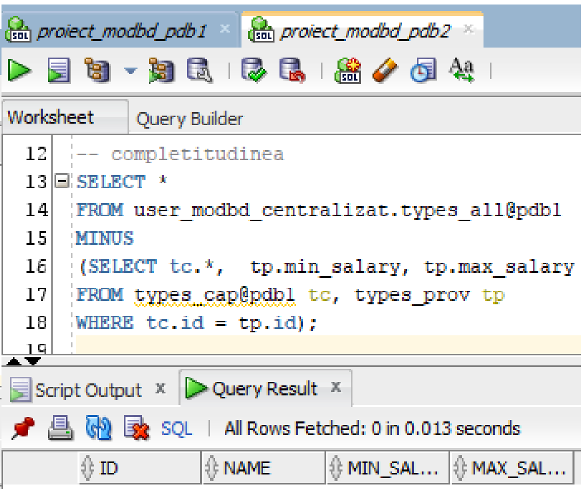
	ID	NAME	MIN_SALARY	MAX_SALARY
1	1	waiter	1000	4000
2	2	chef	2000	5000

```
-- completitudinea
SELECT *
FROM user_modbd_centralizat.types_all
MINUS
(SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap tc, types_prov@pdb2 tp
WHERE tc.id = tp.id);

SELECT *
FROM user_modbd_centralizat.types_all@pdb1
MINUS
(SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap@pdb1 tc, types_prov tp
WHERE tc.id = tp.id);
```



```
-- completitudinea
SELECT *
FROM user_modbd_centralizat.types_all
MINUS
(SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap tc, types_prov@pdb2 tp
WHERE tc.id = tp.id);
```

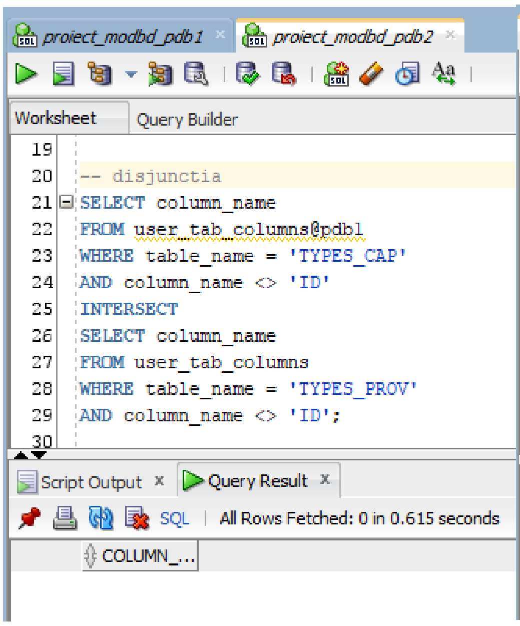
```
All Rows Fetched: 0 in 0.004 seconds
ID	NAME	MIN_SAL...	MAX_SAL...
```



```
-- completitudinea
SELECT *
FROM user_modbd_centralizat.types_all@pdb1
MINUS
(SELECT tc.*,  tp.min_salary, tp.max_salary
FROM types_cap@pdb1 tc, types_prov tp
WHERE tc.id = tp.id);
```

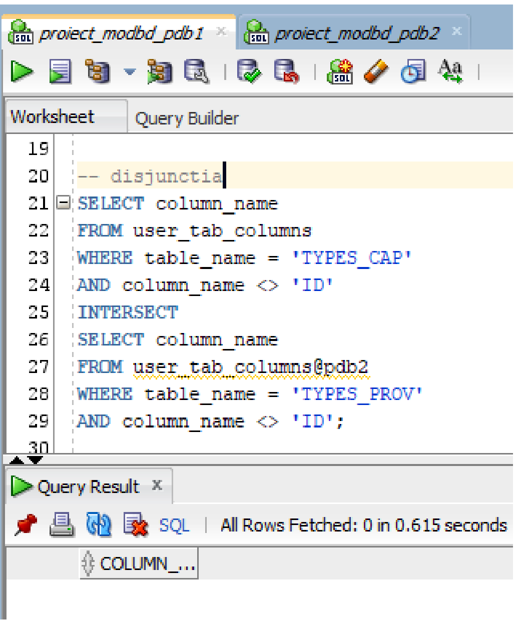
```
All Rows Fetched: 0 in 0.013 seconds
ID	NAME	MIN_SAL...	MAX_SAL...
```

```
-- disjunctia
SELECT column_name
FROM user_tab_columns
WHERE table_name = 'TYPES_CAP'
AND column_name <> 'ID'
INTERSECT
SELECT column_name
FROM user_tab_columns@pdb2
WHERE table_name = 'TYPES_PROV'
AND column_name <> 'ID';

SELECT column_name
FROM user_tab_columns@pdb1
WHERE table_name = 'TYPES_CAP'
AND column_name <> 'ID'
INTERSECT
SELECT column_name
FROM user_tab_columns
WHERE table_name = 'TYPES_PROV'
AND column_name <> 'ID';
```



```
-- disjunctia
SELECT column_name
FROM user_tab_columns@pdb1
WHERE table_name = 'TYPES_CAP'
AND column_name <> 'ID'
INTERSECT
SELECT column_name
FROM user_tab_columns
WHERE table_name = 'TYPES_PROV'
AND column_name <> 'ID';
```

```
All Rows Fetched: 0 in 0.615 seconds
COLUMN_...
```

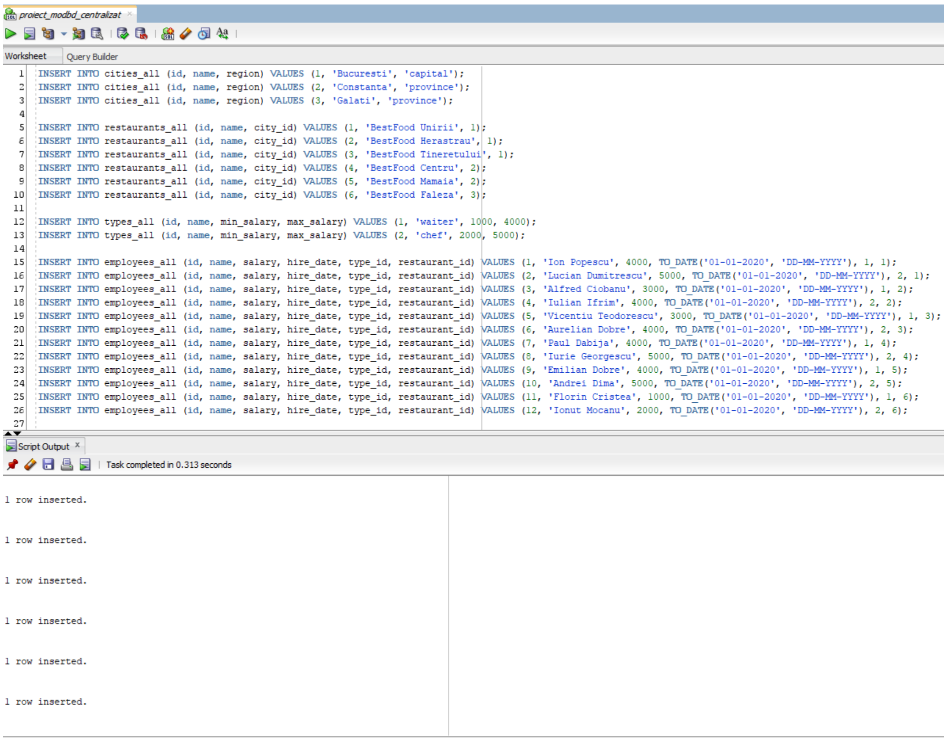


```
-- disjunctia
SELECT column_name
FROM user_tab_columns
WHERE table_name = 'TYPES_CAP'
AND column_name <> 'ID'
INTERSECT
SELECT column_name
FROM user_tab_columns@pdb2
WHERE table_name = 'TYPES_PROV'
AND column_name <> 'ID';
```

```
All Rows Fetched: 0 in 0.615 seconds
COLUMN_...
```

3. Popularea cu date a bazelor de date

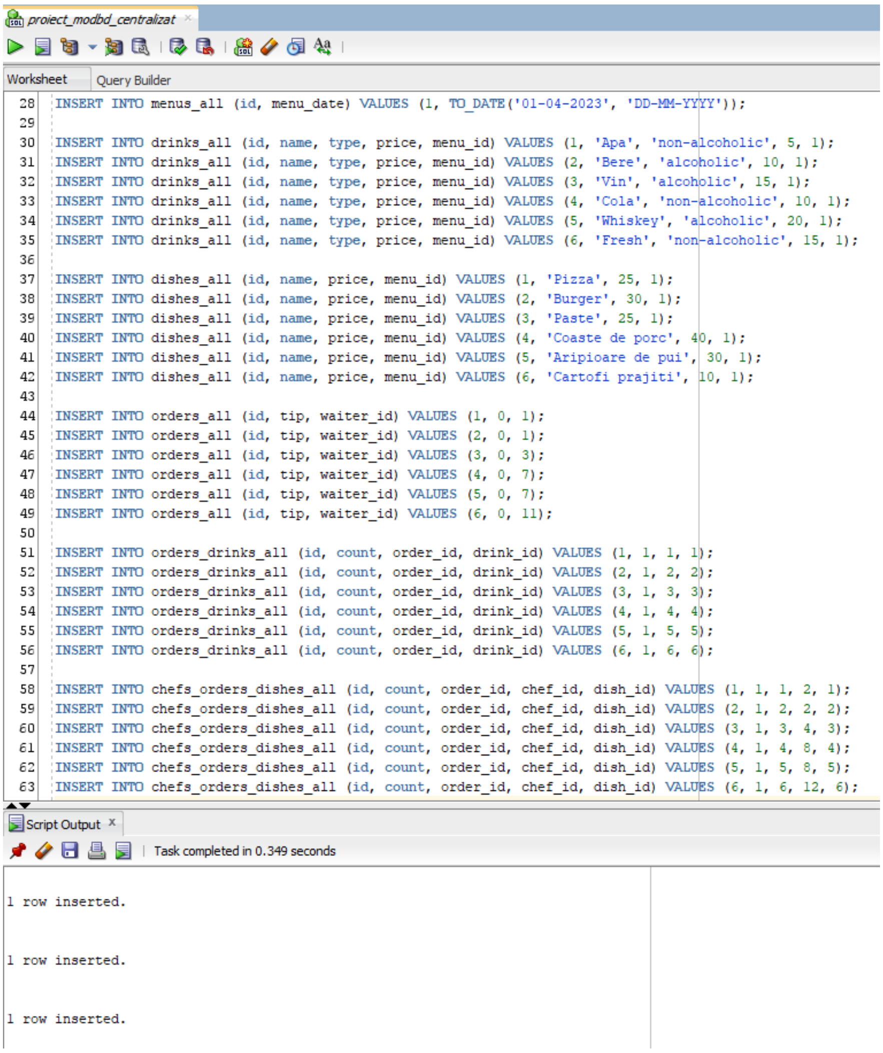
```
INSERT INTO cities_all (id, name, region) VALUES (1, 'Bucuresti', 'capital');
INSERT INTO cities_all (id, name, region) VALUES (2, 'Constanta', 'province');
INSERT INTO cities_all (id, name, region) VALUES (3, 'Galati', 'province');

INSERT INTO restaurants_all (id, name, city_id) VALUES (1, 'BestFood Unirii', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (2, 'BestFood Herastrau', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (3, 'BestFood Tineretului', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (4, 'BestFood Centru', 2);
INSERT INTO restaurants_all (id, name, city_id) VALUES (5, 'BestFood Mamaia', 2);
INSERT INTO restaurants_all (id, name, city_id) VALUES (6, 'BestFood Faleza', 3);

INSERT INTO types_all (id, name, min_salary, max_salary) VALUES (1, 'waiter', 1000, 4000);
INSERT INTO types_all (id, name, min_salary, max_salary) VALUES (2, 'chef', 2000, 5000);

INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (1, 'Ion
Popescu', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 1);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (2, 'Lucian
Dumitrescu', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 1);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (3, 'Alfred
Ciobanu', 3000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 2);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (4, 'Iulian
Ifrim', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 2);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (5,
'Vicentiu Teodorescu', 3000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 3);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (6,
'Aurelian Dobre', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 3);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (7, 'Paul
Dabija', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 4);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (8, 'Iurie
Georgescu', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 4);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (9, 'Emilian
Dobre', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 5);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (10,
'Andrei Dima', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 5);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (11, 'Florin
Cristea', 1000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 6);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (12, 'Ionut
Mocanu', 2000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 6);

INSERT INTO menus_all (id, menu_date) VALUES (1, TO_DATE('01-04-2023', 'DD-MM-YYYY'));

INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (1, 'Apa', 'non-alcoholic', 5, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (2, 'Bere', 'alcoholic', 10, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (3, 'Vin', 'alcoholic', 15, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (4, 'Cola', 'non-alcoholic', 10, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (5, 'Whiskey', 'alcoholic', 20, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (6, 'Fresh', 'non-alcoholic', 15, 1);

INSERT INTO dishes_all (id, name, price, menu_id) VALUES (1, 'Pizza', 25, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (2, 'Burger', 30, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (3, 'Paste', 25, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (4, 'Coaste de porc', 40, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (5, 'Aripioare de pui', 30, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (6, 'Cartofi prajiti', 10, 1);

INSERT INTO orders_all (id, tip, waiter_id) VALUES (1, 0, 1);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (2, 0, 1);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (3, 0, 3);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (4, 0, 7);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (5, 0, 7);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (6, 0, 11);

INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (1, 1, 1, 1);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (2, 1, 2, 2);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (3, 1, 3, 3);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (4, 1, 4, 4);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (5, 1, 5, 5);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (6, 1, 6, 6);

INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (1, 1, 1, 2, 1);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (2, 1, 2, 2, 2);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (3, 1, 3, 4, 3);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (4, 1, 4, 8, 4);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (5, 1, 5, 8, 5);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (6, 1, 6, 12, 6);
```



```
INSERT INTO cities_all (id, name, region) VALUES (1, 'Bucuresti', 'capital');
INSERT INTO cities_all (id, name, region) VALUES (2, 'Constanta', 'province');
INSERT INTO cities_all (id, name, region) VALUES (3, 'Galati', 'province');

INSERT INTO restaurants_all (id, name, city_id) VALUES (1, 'BestFood Unirii', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (2, 'BestFood Herastrau', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (3, 'BestFood Tineretului', 1);
INSERT INTO restaurants_all (id, name, city_id) VALUES (4, 'BestFood Centru', 2);
INSERT INTO restaurants_all (id, name, city_id) VALUES (5, 'BestFood Mamaia', 2);
INSERT INTO restaurants_all (id, name, city_id) VALUES (6, 'BestFood Faleza', 3);

INSERT INTO types_all (id, name, min_salary, max_salary) VALUES (1, 'waiter', 1000, 4000);
INSERT INTO types_all (id, name, min_salary, max_salary) VALUES (2, 'chef', 2000, 5000);

INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (1, 'Ion Popescu', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 1);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (2, 'Lucian Dumitrescu', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 1);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (3, 'Alfred Ciobanu', 3000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 2);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (4, 'Iulian Ifrim', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 2);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (5, 'Vicentiu Teodorescu', 3000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 3);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (6, 'Aurelian Dobre', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 3);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (7, 'Paul Dabija', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 4);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (8, 'Iurie Georgescu', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 4);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (9, 'Emilian Dobre', 4000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 5);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (10, 'Andrei Dima', 5000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 5);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (11, 'Florin Cristea', 1000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 1, 6);
INSERT INTO employees_all (id, name, salary, hire_date, type_id, restaurant_id) VALUES (12, 'Ionut Mocanu', 2000, TO_DATE('01-01-2020', 'DD-MM-YYYY'), 2, 6);
```

Task completed in 0.313 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```



```
INSERT INTO menus_all (id, menu_date) VALUES (1, TO_DATE('01-04-2023', 'DD-MM-YYYY'));

INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (1, 'Apa', 'non-alcoholic', 5, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (2, 'Bere', 'alcoholic', 10, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (3, 'Vin', 'alcoholic', 15, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (4, 'Cola', 'non-alcoholic', 10, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (5, 'Whiskey', 'alcoholic', 20, 1);
INSERT INTO drinks_all (id, name, type, price, menu_id) VALUES (6, 'Fresh', 'non-alcoholic', 15, 1);

INSERT INTO dishes_all (id, name, price, menu_id) VALUES (1, 'Pizza', 25, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (2, 'Burger', 30, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (3, 'Paste', 25, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (4, 'Coaste de porc', 40, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (5, 'Aripioare de pui', 30, 1);
INSERT INTO dishes_all (id, name, price, menu_id) VALUES (6, 'Cartofi prajiti', 10, 1);

INSERT INTO orders_all (id, tip, waiter_id) VALUES (1, 0, 1);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (2, 0, 1);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (3, 0, 3);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (4, 0, 7);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (5, 0, 7);
INSERT INTO orders_all (id, tip, waiter_id) VALUES (6, 0, 11);

INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (1, 1, 1, 1);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (2, 1, 2, 2);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (3, 1, 3, 3);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (4, 1, 4, 4);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (5, 1, 5, 5);
INSERT INTO orders_drinks_all (id, count, order_id, drink_id) VALUES (6, 1, 6, 6);

INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (1, 1, 1, 2, 1);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (2, 1, 2, 2, 2);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (3, 1, 3, 4, 3);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (4, 1, 4, 8, 4);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (5, 1, 5, 8, 5);
INSERT INTO chefs_orders_dishes_all (id, count, order_id, chef_id, dish_id) VALUES (6, 1, 6, 12, 6);
```

Task completed in 0.349 seconds

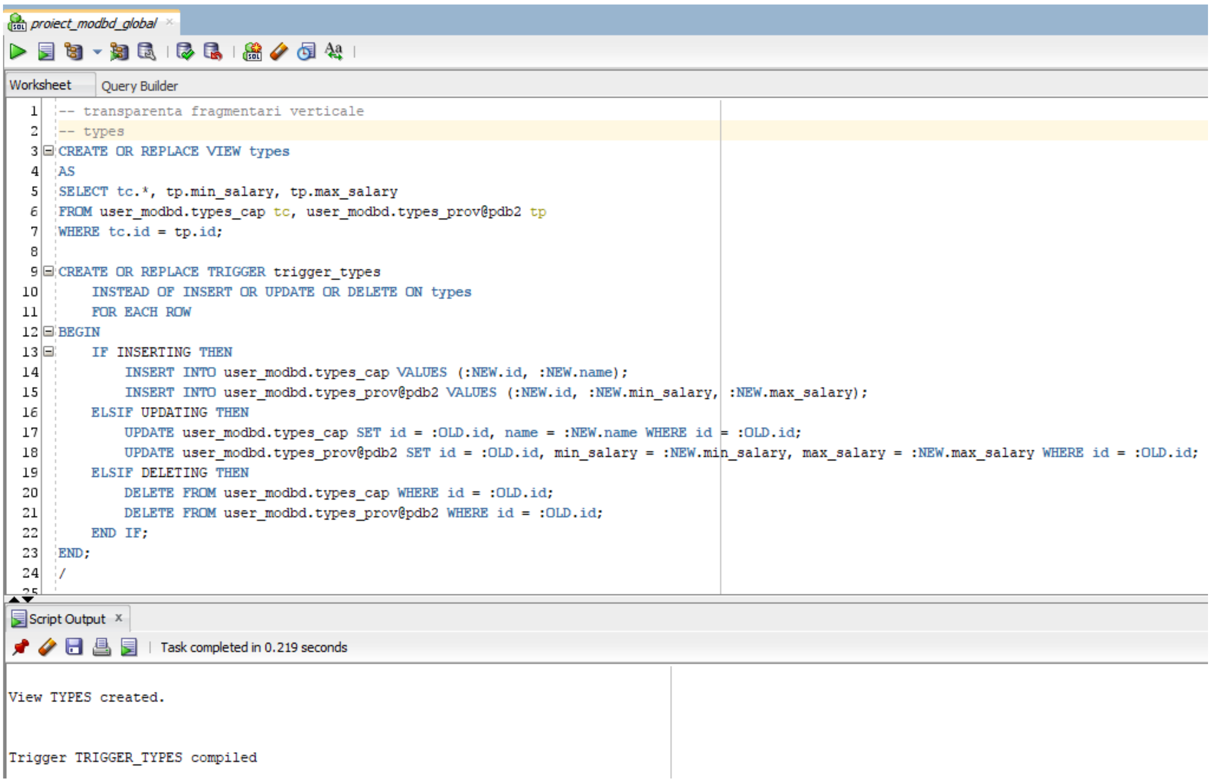
```
1 row inserted.


1 row inserted.


1 row inserted.
```

4. Furnizarea formelor de transparență pentru întreg modelul ales

a. Transparență pentru fragmentele verticale

Pentru a realiza transparența fragmentelor verticale, în global, am create un view numit types ca fiind un join al tabelelor types_cap și types_prov. Apoi, am definit un trigger care, atunci când se realizează o operație de insert, update sau delete asupra view-ului, modificările să aibă loc asupra tabelelor locale din bazele pdb1 și pdb2.

```
-- transparenta fragmentari verticale
-- types
CREATE OR REPLACE VIEW types
AS
SELECT tc.*, tp.min_salary, tp.max_salary
FROM user_modbd.types_cap tc, user_modbd.types_prov@pdb2 tp
WHERE tc.id = tp.id;

CREATE OR REPLACE TRIGGER trigger_types
   INSTEAD OF INSERT OR UPDATE OR DELETE ON types
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO user_modbd.types_cap VALUES (:NEW.id, :NEW.name);
      INSERT INTO user_modbd.types_prov@pdb2 VALUES (:NEW.id, :NEW.min_salary,
:NEW.max_salary);
   ELSIF UPDATING THEN
      UPDATE user_modbd.types_cap SET id = :OLD.id, name = :NEW.name WHERE id = :OLD.id;
      UPDATE user_modbd.types_prov@pdb2 SET id = :OLD.id, min_salary = :NEW.min_salary,
max_salary = :NEW.max_salary WHERE id = :OLD.id;
   ELSIF DELETING THEN
      DELETE FROM user_modbd.types_cap WHERE id = :OLD.id;
      DELETE FROM user_modbd.types_prov@pdb2 WHERE id = :OLD.id;
   END IF;
END;
/
```



```
-- transparenta fragmentari verticale
-- types
CREATE OR REPLACE VIEW types
AS
SELECT tc.*, tp.min_salary, tp.max_salary
FROM user_modbd.types_cap tc, user_modbd.types_prov@pdb2 tp
WHERE tc.id = tp.id;

CREATE OR REPLACE TRIGGER trigger_types
    INSTEAD OF INSERT OR UPDATE OR DELETE ON types
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_modbd.types_cap VALUES (:NEW.id, :NEW.name);
        INSERT INTO user_modbd.types_prov@pdb2 VALUES (:NEW.id, :NEW.min_salary, :NEW.max_salary);
    ELSIF UPDATING THEN
        UPDATE user_modbd.types_cap SET id = :OLD.id, name = :NEW.name WHERE id = :OLD.id;
        UPDATE user_modbd.types_prov@pdb2 SET id = :OLD.id, min_salary = :NEW.min_salary, max_salary = :NEW.max_salary WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM user_modbd.types_cap WHERE id = :OLD.id;
        DELETE FROM user_modbd.types_prov@pdb2 WHERE id = :OLD.id;
    END IF;
END;
/
```

Task completed in 0.219 seconds

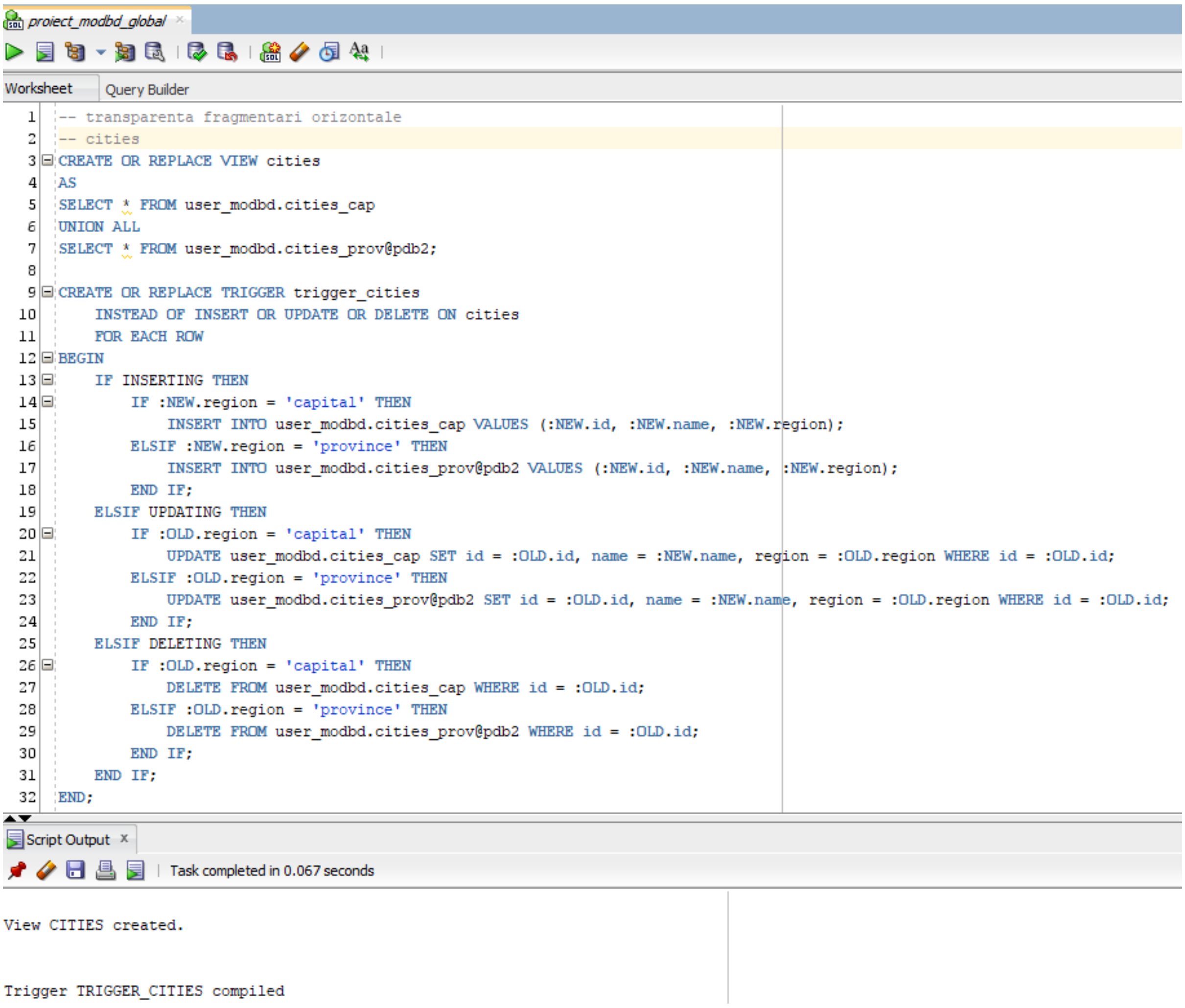
```
View TYPES created.


Trigger TRIGGER_TYPES compiled
```

b. Transparență pentru fragmentele orizontale

Pentru transparența fragmentelor orizontale, am creat view-uri ca uniuni ale tabelelor din pdb1 și pdb2, apoi am definit trigger-i care, în caz de insert, update sau delete, să efectueze modificări asupra bazelor de date aferente. De exemplu, în cazul cities, trigger-ul verifică dacă regiunea este capitală sau provincie și efectuează operațiile asupra bazei respective.

```
-- transparenta fragmentari orizontale
-- cities
CREATE OR REPLACE VIEW cities
AS
SELECT * FROM user_modbd.cities_cap
UNION ALL
```

```
SELECT * FROM user_modbd.cities_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_cities
   INSTEAD OF INSERT OR UPDATE OR DELETE ON cities
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.region = 'capital' THEN
         INSERT INTO user_modbd.cities_cap VALUES (:NEW.id, :NEW.name, :NEW.region);
      ELSIF :NEW.region = 'province' THEN
         INSERT INTO user_modbd.cities_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.region);
      END IF;
   ELSIF UPDATING THEN
      IF :OLD.region = 'capital' THEN
         UPDATE user_modbd.cities_cap SET id = :OLD.id, name = :NEW.name, region = :OLD.region
WHERE id = :OLD.id;
      ELSIF :OLD.region = 'province' THEN
         UPDATE user_modbd.cities_prov@pdb2 SET id = :OLD.id, name = :NEW.name, region =
:OLD.region WHERE id = :OLD.id;
      END IF;
   ELSIF DELETING THEN
      IF :OLD.region = 'capital' THEN
         DELETE FROM user_modbd.cities_cap WHERE id = :OLD.id;
      ELSIF :OLD.region = 'province' THEN
         DELETE FROM user_modbd.cities_prov@pdb2 WHERE id = :OLD.id;
      END IF;
   END IF;
END;
/
```

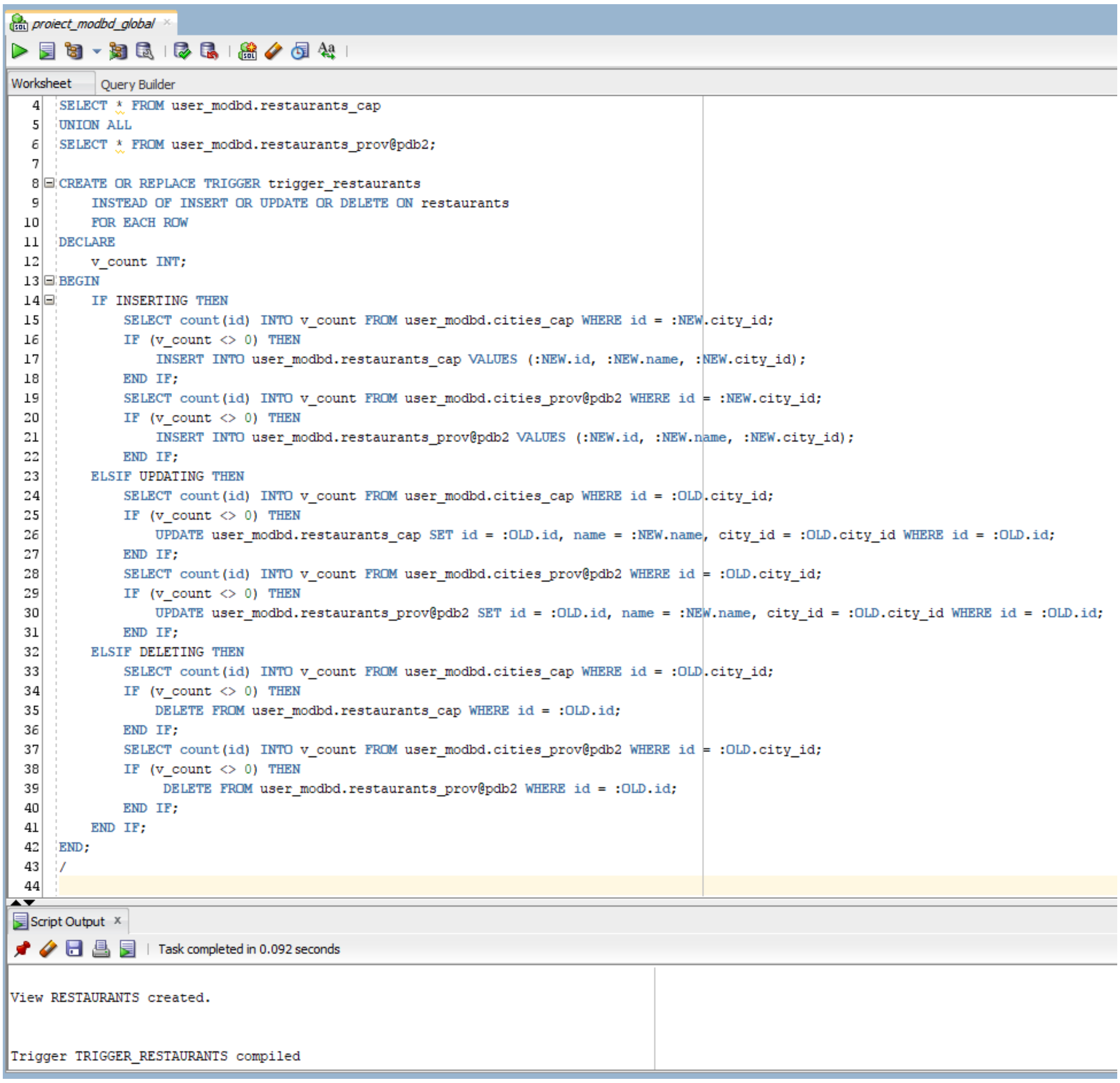


```
-- transparenta fragmentari orizontale
-- cities
CREATE OR REPLACE VIEW cities
AS
SELECT * FROM user_modbd.cities_cap
UNION ALL
SELECT * FROM user_modbd.cities_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_cities
    INSTEAD OF INSERT OR UPDATE OR DELETE ON cities
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        IF :NEW.region = 'capital' THEN
            INSERT INTO user_modbd.cities_cap VALUES (:NEW.id, :NEW.name, :NEW.region);
        ELSIF :NEW.region = 'province' THEN
            INSERT INTO user_modbd.cities_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.region);
        END IF;
    ELSIF UPDATING THEN
        IF :OLD.region = 'capital' THEN
            UPDATE user_modbd.cities_cap SET id = :OLD.id, name = :NEW.name, region = :OLD.region WHERE id = :OLD.id;
        ELSIF :OLD.region = 'province' THEN
            UPDATE user_modbd.cities_prov@pdb2 SET id = :OLD.id, name = :NEW.name, region = :OLD.region WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        IF :OLD.region = 'capital' THEN
            DELETE FROM user_modbd.cities_cap WHERE id = :OLD.id;
        ELSIF :OLD.region = 'province' THEN
            DELETE FROM user_modbd.cities_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
END;
```

```
View CITIES created.


Trigger TRIGGER_CITIES compiled
```

Pentru tabelele fragmentate orizontal derivat, trigger-i verifică în tabela părinte dacă se află cheia externă.

```
-- restaurants
CREATE OR REPLACE VIEW restaurants
AS
SELECT * FROM user_modbd.restaurants_cap
UNION ALL
SELECT * FROM user_modbd.restaurants_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_restaurants
   INSTEAD OF INSERT OR UPDATE OR DELETE ON restaurants
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   IF INSERTING THEN
      SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :NEW.city_id;
      IF (v_count <> 0) THEN
         INSERT INTO user_modbd.restaurants_cap VALUES (:NEW.id, :NEW.name, :NEW.city_id);
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id =
:NEW.city_id;
      IF (v_count <> 0) THEN
      INSERT INTO user_modbd.restaurants_prov@pdb2 VALUES (:NEW.id, :NEW.name,
:NEW.city_id);
    END IF;
  ELSIF UPDATING THEN
    SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :OLD.city_id;
    IF (v_count <> 0) THEN
      UPDATE user_modbd.restaurants_cap SET id = :OLD.id, name = :NEW.name, city_id =
:OLD.city_id WHERE id = :OLD.id;
    END IF;
    SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id =
:OLD.city_id;
    IF (v_count <> 0) THEN
      UPDATE user_modbd.restaurants_prov@pdb2 SET id = :OLD.id, name = :NEW.name, city_id
= :OLD.city_id WHERE id = :OLD.id;
    END IF;
  ELSIF DELETING THEN
    SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :OLD.city_id;
    IF (v_count <> 0) THEN
      DELETE FROM user_modbd.restaurants_cap WHERE id = :OLD.id;
    END IF;
    SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id =
:OLD.city_id;
    IF (v_count <> 0) THEN
       DELETE FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.id;
    END IF;
  END IF;
END;
/
```

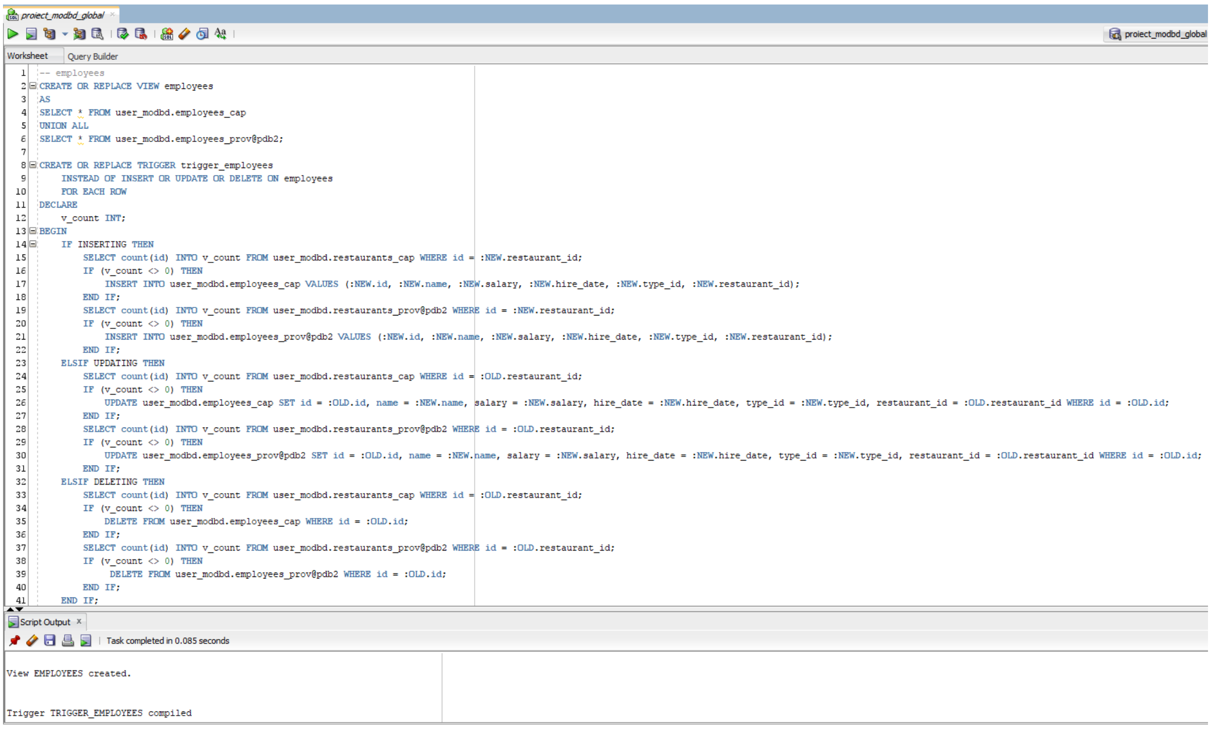


```
SELECT * FROM user_modbd.restaurants_cap
UNION ALL
SELECT * FROM user_modbd.restaurants_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_restaurants
    INSTEAD OF INSERT OR UPDATE OR DELETE ON restaurants
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :NEW.city_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.restaurants_cap VALUES (:NEW.id, :NEW.name, :NEW.city_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id = :NEW.city_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.restaurants_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.city_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :OLD.city_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.restaurants_cap SET id = :OLD.id, name = :NEW.name, city_id = :OLD.city_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id = :OLD.city_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.restaurants_prov@pdb2 SET id = :OLD.id, name = :NEW.name, city_id = :OLD.city_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.cities_cap WHERE id = :OLD.city_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.restaurants_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.cities_prov@pdb2 WHERE id = :OLD.city_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
END;
/
```

Task completed in 0.092 seconds

```
View RESTAURANTS created.


Trigger TRIGGER_RESTAURANTS compiled
```

```
-- employees
CREATE OR REPLACE VIEW employees
AS
SELECT * FROM user_modbd.employees_cap
UNION ALL
SELECT * FROM user_modbd.employees_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_employees
    INSTEAD OF INSERT OR UPDATE OR DELETE ON employees
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :NEW.restaurant_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.employees_cap VALUES (:NEW.id, :NEW.name, :NEW.salary, :NEW.hire_date, :NEW.type_id, :NEW.restaurant_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :NEW.restaurant_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.employees_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.salary, :NEW.hire_date, :NEW.type_id, :NEW.restaurant_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.employees_cap SET id = :OLD.id, name = :NEW.name, salary = :NEW.salary, hire_date = :NEW.hire_date, type_id = :NEW.type_id, restaurant_id = :OLD.restaurant_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.employees_prov@pdb2 SET id = :OLD.id, name = :NEW.name, salary = :NEW.salary, hire_date = :NEW.hire_date, type_id = :NEW.type_id, restaurant_id = :OLD.restaurant_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.employees_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.employees_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
END;
/
```



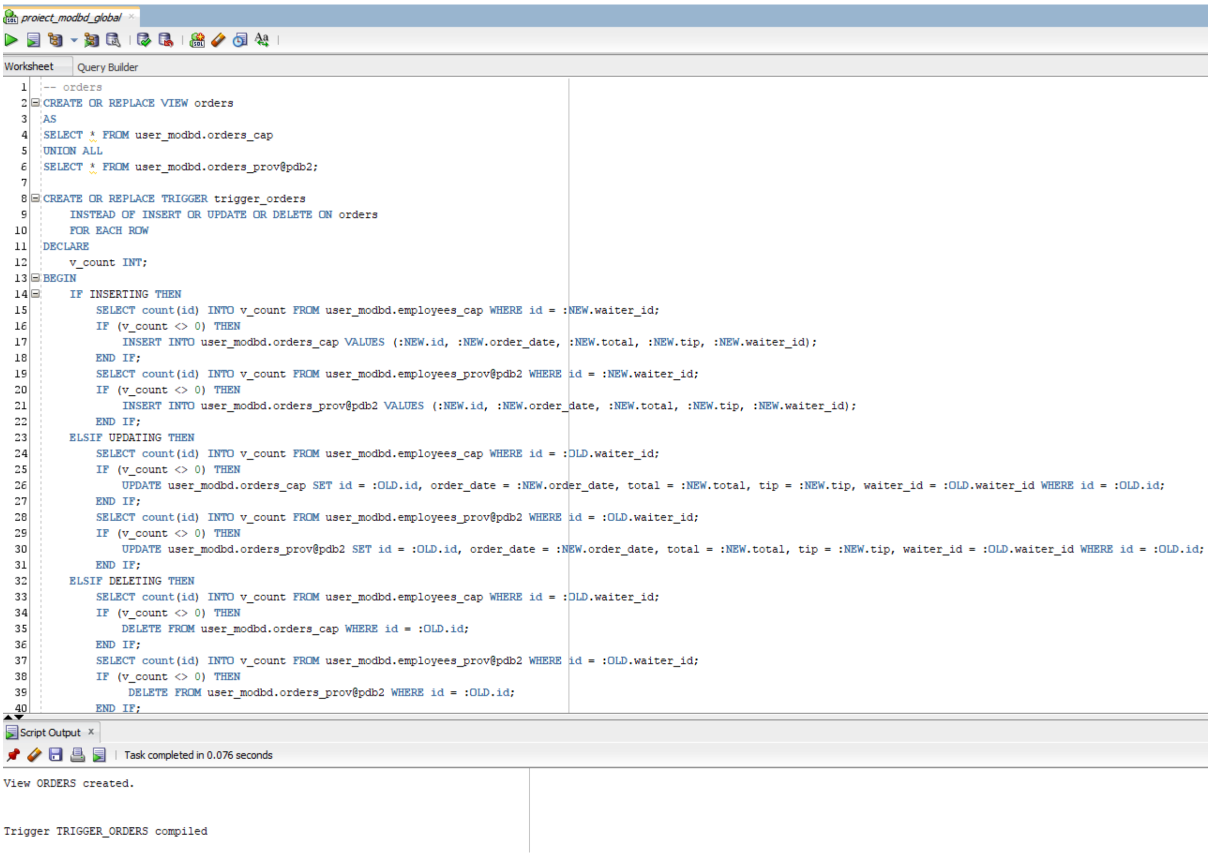
```
-- employees
CREATE OR REPLACE VIEW employees
AS
SELECT * FROM user_modbd.employees_cap
UNION ALL
SELECT * FROM user_modbd.employees_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_employees
    INSTEAD OF INSERT OR UPDATE OR DELETE ON employees
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :NEW.restaurant_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.employees_cap VALUES (:NEW.id, :NEW.name, :NEW.salary, :NEW.hire_date, :NEW.type_id, :NEW.restaurant_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :NEW.restaurant_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.employees_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.salary, :NEW.hire_date, :NEW.type_id, :NEW.restaurant_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.employees_cap SET id = :OLD.id, name = :NEW.name, salary = :NEW.salary, hire_date = :NEW.hire_date, type_id = :NEW.type_id, restaurant_id = :OLD.restaurant_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.employees_prov@pdb2 SET id = :OLD.id, name = :NEW.name, salary = :NEW.salary, hire_date = :NEW.hire_date, type_id = :NEW.type_id, restaurant_id = :OLD.restaurant_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_cap WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.employees_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.restaurants_prov@pdb2 WHERE id = :OLD.restaurant_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.employees_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
```

Task completed in 0.085 seconds

```
View EMPLOYEES created.


Trigger TRIGGER_EMPLOYEES compiled
```

```
-- orders
CREATE OR REPLACE VIEW orders
```

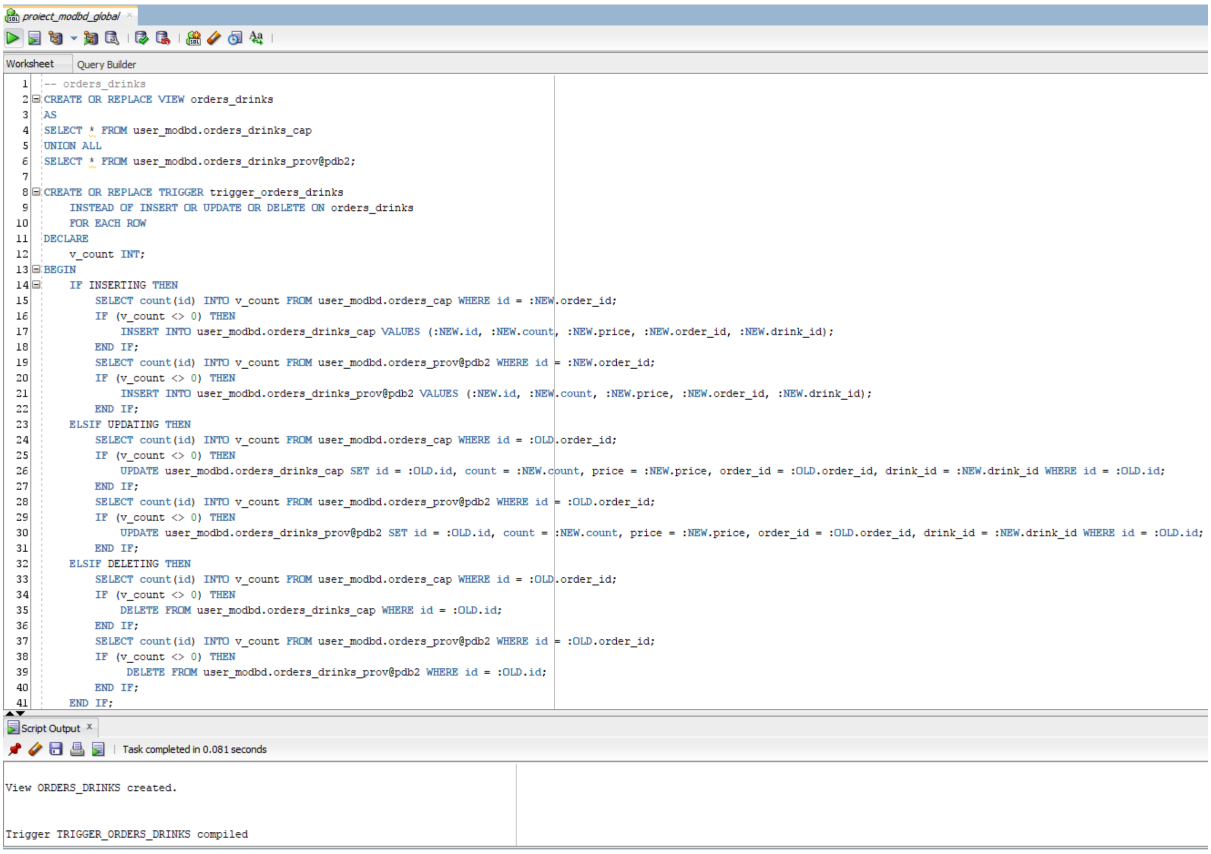
```
AS
SELECT * FROM user_modbd.orders_cap
UNION ALL
SELECT * FROM user_modbd.orders_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_orders
   INSTEAD OF INSERT OR UPDATE OR DELETE ON orders
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   IF INSERTING THEN
      SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id =
:NEW.waiter_id;
      IF (v_count <> 0) THEN
         INSERT INTO user_modbd.orders_cap VALUES (:NEW.id, :NEW.order_date, :NEW.total,
:NEW.tip, :NEW.waiter_id);
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id =
:NEW.waiter_id;
      IF (v_count <> 0) THEN
         INSERT INTO user_modbd.orders_prov@pdb2 VALUES (:NEW.id, :NEW.order_date,
:NEW.total, :NEW.tip, :NEW.waiter_id);
      END IF;
   ELSIF UPDATING THEN
      SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id =
:OLD.waiter_id;
      IF (v_count <> 0) THEN
         UPDATE user_modbd.orders_cap SET id = :OLD.id, order_date = :NEW.order_date, total =
:NEW.total, tip = :NEW.tip, waiter_id = :OLD.waiter_id WHERE id = :OLD.id;
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id =
:OLD.waiter_id;
      IF (v_count <> 0) THEN
         UPDATE user_modbd.orders_prov@pdb2 SET id = :OLD.id, order_date = :NEW.order_date,
total = :NEW.total, tip = :NEW.tip, waiter_id = :OLD.waiter_id WHERE id = :OLD.id;
      END IF;
   ELSIF DELETING THEN
      SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id =
:OLD.waiter_id;
      IF (v_count <> 0) THEN
         DELETE FROM user_modbd.orders_cap WHERE id = :OLD.id;
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id =
:OLD.waiter_id;
      IF (v_count <> 0) THEN
          DELETE FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.id;
      END IF;
   END IF;
END;
/
```



```
-- orders
CREATE OR REPLACE VIEW orders
AS
SELECT * FROM user_modbd.orders_cap
UNION ALL
SELECT * FROM user_modbd.orders_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_orders
    INSTEAD OF INSERT OR UPDATE OR DELETE ON orders
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id = :NEW.waiter_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_cap VALUES (:NEW.id, :NEW.order_date, :NEW.total, :NEW.tip, :NEW.waiter_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id = :NEW.waiter_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_prov@pdb2 VALUES (:NEW.id, :NEW.order_date, :NEW.total, :NEW.tip, :NEW.waiter_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id = :OLD.waiter_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_cap SET id = :OLD.id, order_date = :NEW.order_date, total = :NEW.total, tip = :NEW.tip, waiter_id = :OLD.waiter_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id = :OLD.waiter_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_prov@pdb2 SET id = :OLD.id, order_date = :NEW.order_date, total = :NEW.total, tip = :NEW.tip, waiter_id = :OLD.waiter_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.employees_cap WHERE id = :OLD.waiter_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.orders_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.employees_prov@pdb2 WHERE id = :OLD.waiter_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.id;
        END IF;
```

```
View ORDERS created.


Trigger TRIGGER_ORDERS compiled
```

```
-- orders_drinks
CREATE OR REPLACE VIEW orders_drinks
AS
SELECT * FROM user_modbd.orders_drinks_cap
UNION ALL
SELECT * FROM user_modbd.orders_drinks_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_orders_drinks
    INSTEAD OF INSERT OR UPDATE OR DELETE ON orders_drinks
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_drinks_cap VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.order_id, :NEW.drink_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_drinks_prov@pdb2 VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.order_id, :NEW.drink_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_drinks_cap SET id = :OLD.id, count = :NEW.count, price = :NEW.price, order_id = :OLD.order_id, drink_id = :NEW.drink_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id =
:OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_drinks_prov@pdb2 SET id = :OLD.id, count = :NEW.count,
price = :NEW.price, order_id = :OLD.order_id, drink_id = :NEW.drink_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.orders_drinks_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id =
:OLD.order_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.orders_drinks_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
END;
/
```

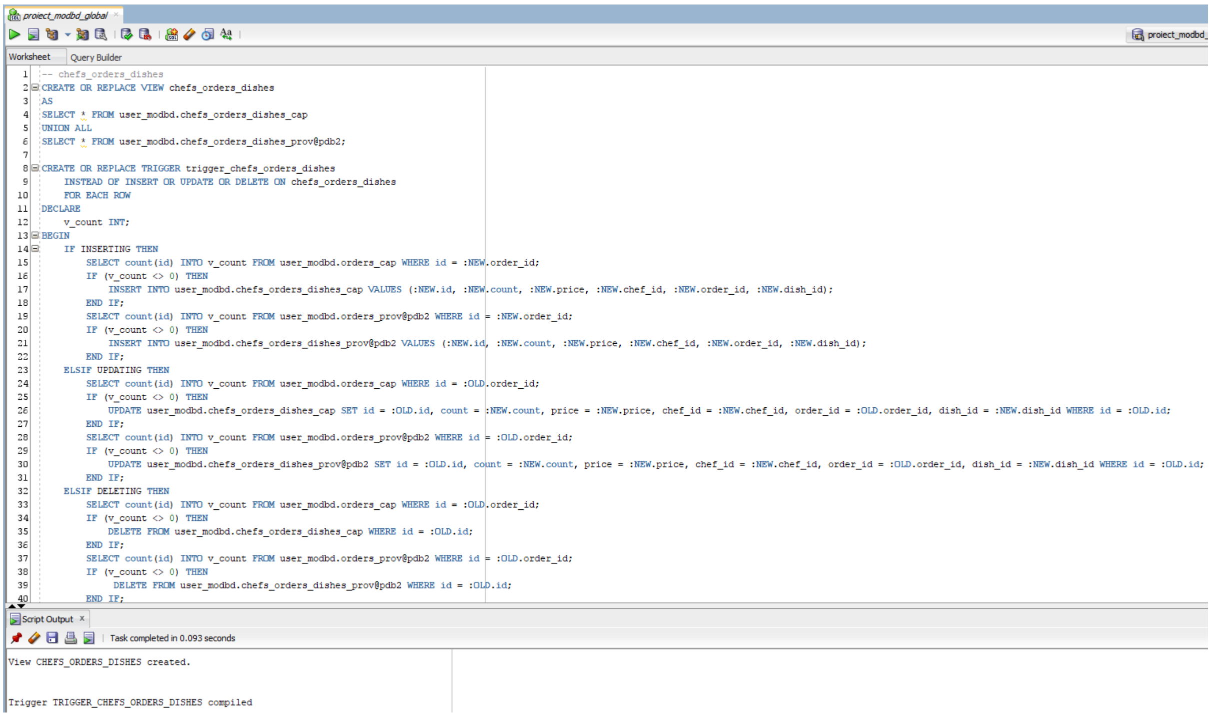


```
-- orders_drinks
CREATE OR REPLACE VIEW orders_drinks
AS
SELECT * FROM user_modbd.orders_drinks_cap
UNION ALL
SELECT * FROM user_modbd.orders_drinks_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_orders_drinks
    INSTEAD OF INSERT OR UPDATE OR DELETE ON orders_drinks
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_drinks_cap VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.order_id, :NEW.drink_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.orders_drinks_prov@pdb2 VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.order_id, :NEW.drink_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_drinks_cap SET id = :OLD.id, count = :NEW.count, price = :NEW.price, order_id = :OLD.order_id, drink_id = :NEW.drink_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.orders_drinks_prov@pdb2 SET id = :OLD.id, count = :NEW.count, price = :NEW.price, order_id = :OLD.order_id, drink_id = :NEW.drink_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.orders_drinks_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.orders_drinks_prov@pdb2 WHERE id = :OLD.id;
        END IF;
    END IF;
```

```
View ORDERS_DRINKS created.


Trigger TRIGGER_ORDERS_DRINKS compiled
```

```
-- chefs_orders_dishes
CREATE OR REPLACE VIEW chefs_orders_dishes
AS
SELECT * FROM user_modbd.chefs_orders_dishes_cap
UNION ALL
SELECT * FROM user_modbd.chefs_orders_dishes_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes
    INSTEAD OF INSERT OR UPDATE OR DELETE ON chefs_orders_dishes
    FOR EACH ROW
DECLARE
    v_count INT;
```

```
BEGIN
   IF INSERTING THEN
      SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :NEW.order_id;
      IF (v_count <> 0) THEN
         INSERT INTO user_modbd.chefs_orders_dishes_cap VALUES (:NEW.id, :NEW.count,
:NEW.price, :NEW.chef_id, :NEW.order_id, :NEW.dish_id);
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id =
:NEW.order_id;
      IF (v_count <> 0) THEN
         INSERT INTO user_modbd.chefs_orders_dishes_prov@pdb2 VALUES (:NEW.id,
:NEW.count, :NEW.price, :NEW.chef_id, :NEW.order_id, :NEW.dish_id);
      END IF;
   ELSIF UPDATING THEN
      SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
      IF (v_count <> 0) THEN
         UPDATE user_modbd.chefs_orders_dishes_cap SET id = :OLD.id, count = :NEW.count, price
= :NEW.price, chef_id = :NEW.chef_id, order_id = :OLD.order_id, dish_id = :NEW.dish_id WHERE id
= :OLD.id;
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id =
:OLD.order_id;
      IF (v_count <> 0) THEN
         UPDATE user_modbd.chefs_orders_dishes_prov@pdb2 SET id = :OLD.id, count =
:NEW.count, price = :NEW.price, chef_id = :NEW.chef_id, order_id = :OLD.order_id, dish_id =
:NEW.dish_id WHERE id = :OLD.id;
      END IF;
   ELSIF DELETING THEN
      SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
      IF (v_count <> 0) THEN
         DELETE FROM user_modbd.chefs_orders_dishes_cap WHERE id = :OLD.id;
      END IF;
      SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id =
:OLD.order_id;
      IF (v_count <> 0) THEN
          DELETE FROM user_modbd.chefs_orders_dishes_prov@pdb2 WHERE id = :OLD.id;
      END IF;
   END IF;
END;
/
```



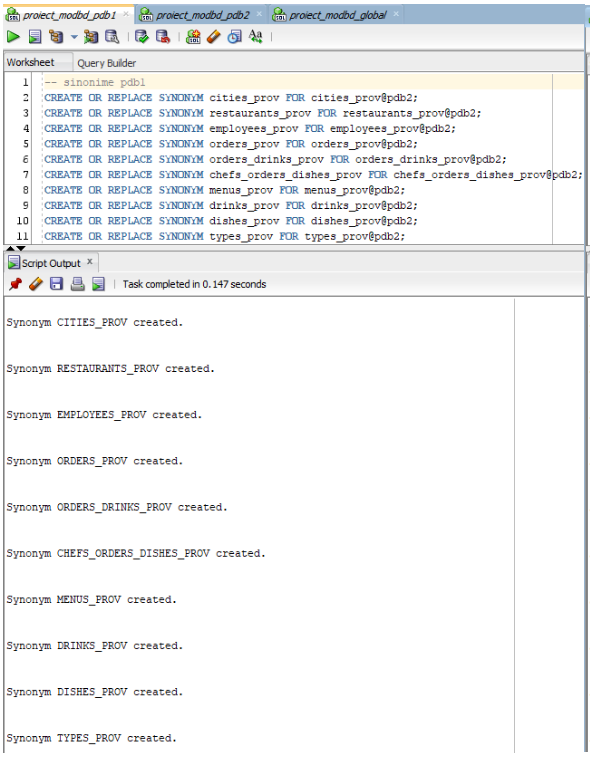
```sql
-- chefs_orders_dishes
CREATE OR REPLACE VIEW chefs_orders_dishes
AS
SELECT * FROM user_modbd.chefs_orders_dishes_cap
UNION ALL
SELECT * FROM user_modbd.chefs_orders_dishes_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes
    INSTEAD OF INSERT OR UPDATE OR DELETE ON chefs_orders_dishes
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    IF INSERTING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.chefs_orders_dishes_cap VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.chef_id, :NEW.order_id, :NEW.dish_id);
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :NEW.order_id;
        IF (v_count <> 0) THEN
            INSERT INTO user_modbd.chefs_orders_dishes_prov@pdb2 VALUES (:NEW.id, :NEW.count, :NEW.price, :NEW.chef_id, :NEW.order_id, :NEW.dish_id);
        END IF;
    ELSIF UPDATING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.chefs_orders_dishes_cap SET id = :OLD.id, count = :NEW.count, price = :NEW.price, chef_id = :NEW.chef_id, order_id = :OLD.order_id, dish_id = :NEW.dish_id WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            UPDATE user_modbd.chefs_orders_dishes_prov@pdb2 SET id = :OLD.id, count = :NEW.count, price = :NEW.price, chef_id = :NEW.chef_id, order_id = :OLD.order_id, dish_id = :NEW.dish_id WHERE id = :OLD.id;
        END IF;
    ELSIF DELETING THEN
        SELECT count(id) INTO v_count FROM user_modbd.orders_cap WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
            DELETE FROM user_modbd.chefs_orders_dishes_cap WHERE id = :OLD.id;
        END IF;
        SELECT count(id) INTO v_count FROM user_modbd.orders_prov@pdb2 WHERE id = :OLD.order_id;
        IF (v_count <> 0) THEN
             DELETE FROM user_modbd.chefs_orders_dishes_prov@pdb2 WHERE id = :OLD.id;
        END IF;
```

```
View CHEFS_ORDERS_DISHES created.


Trigger TRIGGER_CHEFS_ORDERS_DISHES compiled
```

c. Transparență pentru tabelele stocate în altă bază de date față de cea la care se conectează aplicația

Pentru transparența tabelelor stocate în altă bază de date, am create sinonime atât în pdb1 și pdb2, cât și în global.

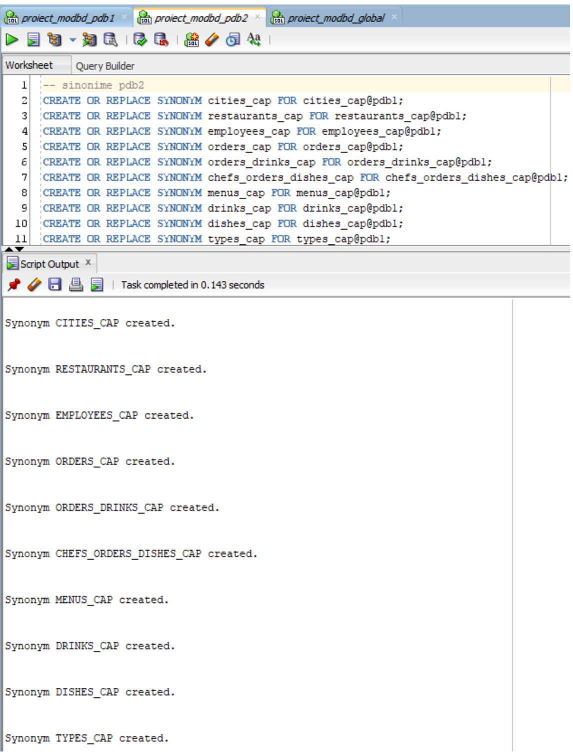
```sql
-- sinonime pdb1
CREATE OR REPLACE SYNONYM cities_prov FOR cities_prov@pdb2;
CREATE OR REPLACE SYNONYM restaurants_prov FOR restaurants_prov@pdb2;
CREATE OR REPLACE SYNONYM employees_prov FOR employees_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_prov FOR orders_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_drinks_prov FOR orders_drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_prov FOR
chefs_orders_dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM menus_prov FOR menus_prov@pdb2;
CREATE OR REPLACE SYNONYM drinks_prov FOR drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM dishes_prov FOR dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM types_prov FOR types_prov@pdb2;

-- sinonime pdb2
CREATE OR REPLACE SYNONYM cities_cap FOR cities_cap@pdb1;
CREATE OR REPLACE SYNONYM restaurants_cap FOR restaurants_cap@pdb1;
CREATE OR REPLACE SYNONYM employees_cap FOR employees_cap@pdb1;
CREATE OR REPLACE SYNONYM orders_cap FOR orders_cap@pdb1;
CREATE OR REPLACE SYNONYM orders_drinks_cap FOR orders_drinks_cap@pdb1;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_cap FOR
chefs_orders_dishes_cap@pdb1;
CREATE OR REPLACE SYNONYM menus_cap FOR menus_cap@pdb1;
CREATE OR REPLACE SYNONYM drinks_cap FOR drinks_cap@pdb1;
CREATE OR REPLACE SYNONYM dishes_cap FOR dishes_cap@pdb1;
CREATE OR REPLACE SYNONYM types_cap FOR types_cap@pdb1;
```



```
-- sinonime pdb1
CREATE OR REPLACE SYNONYM cities_prov FOR cities_prov@pdb2;
CREATE OR REPLACE SYNONYM restaurants_prov FOR restaurants_prov@pdb2;
CREATE OR REPLACE SYNONYM employees_prov FOR employees_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_prov FOR orders_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_drinks_prov FOR orders_drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_prov FOR chefs_orders_dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM menus_prov FOR menus_prov@pdb2;
CREATE OR REPLACE SYNONYM drinks_prov FOR drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM dishes_prov FOR dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM types_prov FOR types_prov@pdb2;
```

Task completed in 0.147 seconds

```
Synonym CITIES_PROV created.


Synonym RESTAURANTS_PROV created.


Synonym EMPLOYEES_PROV created.


Synonym ORDERS_PROV created.


Synonym ORDERS_DRINKS_PROV created.


Synonym CHEFS_ORDERS_DISHES_PROV created.


Synonym MENUS_PROV created.


Synonym DRINKS_PROV created.


Synonym DISHES_PROV created.


Synonym TYPES_PROV created.
```

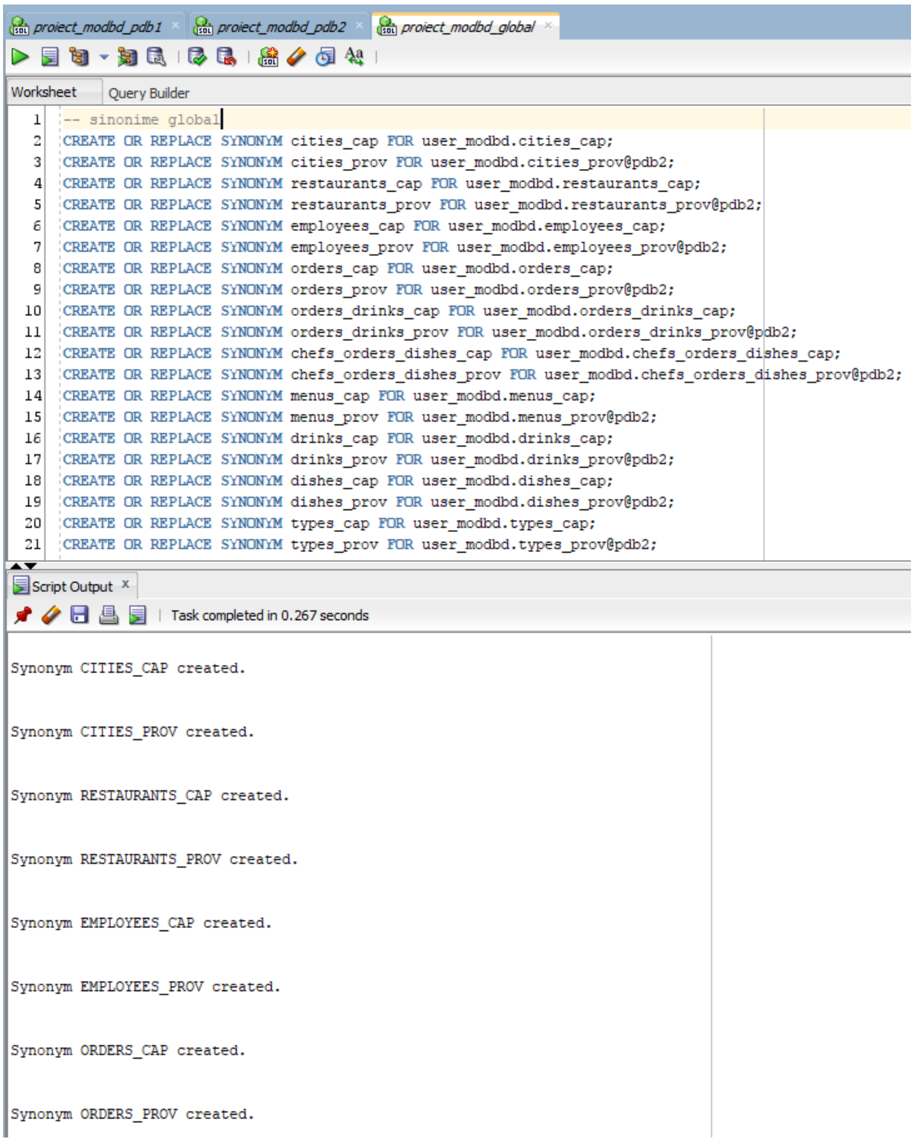


```
-- sinonime pdb2
CREATE OR REPLACE SYNONYM cities_cap FOR cities_cap@pdb1;
CREATE OR REPLACE SYNONYM restaurants_cap FOR restaurants_cap@pdb1;
CREATE OR REPLACE SYNONYM employees_cap FOR employees_cap@pdb1;
CREATE OR REPLACE SYNONYM orders_cap FOR orders_cap@pdb1;
CREATE OR REPLACE SYNONYM orders_drinks_cap FOR orders_drinks_cap@pdb1;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_cap FOR chefs_orders_dishes_cap@pdb1;
CREATE OR REPLACE SYNONYM menus_cap FOR menus_cap@pdb1;
CREATE OR REPLACE SYNONYM drinks_cap FOR drinks_cap@pdb1;
CREATE OR REPLACE SYNONYM dishes_cap FOR dishes_cap@pdb1;
CREATE OR REPLACE SYNONYM types_cap FOR types_cap@pdb1;
```

Task completed in 0.143 seconds

```
Synonym CITIES_CAP created.


Synonym RESTAURANTS_CAP created.


Synonym EMPLOYEES_CAP created.


Synonym ORDERS_CAP created.


Synonym ORDERS_DRINKS_CAP created.


Synonym CHEFS_ORDERS_DISHES_CAP created.


Synonym MENUS_CAP created.


Synonym DRINKS_CAP created.


Synonym DISHES_CAP created.


Synonym TYPES_CAP created.
```

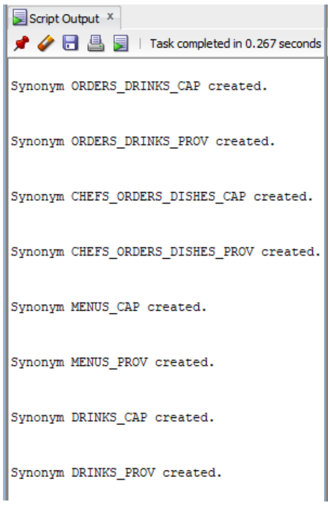
```
-- sinonime global
CREATE OR REPLACE SYNONYM cities_cap FOR user_modbd.cities_cap;
CREATE OR REPLACE SYNONYM cities_prov FOR user_modbd.cities_prov@pdb2;
CREATE OR REPLACE SYNONYM restaurants_cap FOR user_modbd.restaurants_cap;
CREATE OR REPLACE SYNONYM restaurants_prov FOR user_modbd.restaurants_prov@pdb2;
CREATE OR REPLACE SYNONYM employees_cap FOR user_modbd.employees_cap;
CREATE OR REPLACE SYNONYM employees_prov FOR user_modbd.employees_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_cap FOR user_modbd.orders_cap;
CREATE OR REPLACE SYNONYM orders_prov FOR user_modbd.orders_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_drinks_cap FOR user_modbd.orders_drinks_cap;
CREATE OR REPLACE SYNONYM orders_drinks_prov FOR
user_modbd.orders_drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_cap FOR
user_modbd.chefs_orders_dishes_cap;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_prov FOR
user_modbd.chefs_orders_dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM menus_cap FOR user_modbd.menus_cap;
CREATE OR REPLACE SYNONYM menus_prov FOR user_modbd.menus_prov@pdb2;
CREATE OR REPLACE SYNONYM drinks_cap FOR user_modbd.drinks_cap;
CREATE OR REPLACE SYNONYM drinks_prov FOR user_modbd.drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM dishes_cap FOR user_modbd.dishes_cap;
CREATE OR REPLACE SYNONYM dishes_prov FOR user_modbd.dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM types_cap FOR user_modbd.types_cap;
CREATE OR REPLACE SYNONYM types_prov FOR user_modbd.types_prov@pdb2;
```



```
-- sinonime global
CREATE OR REPLACE SYNONYM cities_cap FOR user_modbd.cities_cap;
CREATE OR REPLACE SYNONYM cities_prov FOR user_modbd.cities_prov@pdb2;
CREATE OR REPLACE SYNONYM restaurants_cap FOR user_modbd.restaurants_cap;
CREATE OR REPLACE SYNONYM restaurants_prov FOR user_modbd.restaurants_prov@pdb2;
CREATE OR REPLACE SYNONYM employees_cap FOR user_modbd.employees_cap;
CREATE OR REPLACE SYNONYM employees_prov FOR user_modbd.employees_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_cap FOR user_modbd.orders_cap;
CREATE OR REPLACE SYNONYM orders_prov FOR user_modbd.orders_prov@pdb2;
CREATE OR REPLACE SYNONYM orders_drinks_cap FOR user_modbd.orders_drinks_cap;
CREATE OR REPLACE SYNONYM orders_drinks_prov FOR user_modbd.orders_drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_cap FOR user_modbd.chefs_orders_dishes_cap;
CREATE OR REPLACE SYNONYM chefs_orders_dishes_prov FOR user_modbd.chefs_orders_dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM menus_cap FOR user_modbd.menus_cap;
CREATE OR REPLACE SYNONYM menus_prov FOR user_modbd.menus_prov@pdb2;
CREATE OR REPLACE SYNONYM drinks_cap FOR user_modbd.drinks_cap;
CREATE OR REPLACE SYNONYM drinks_prov FOR user_modbd.drinks_prov@pdb2;
CREATE OR REPLACE SYNONYM dishes_cap FOR user_modbd.dishes_cap;
CREATE OR REPLACE SYNONYM dishes_prov FOR user_modbd.dishes_prov@pdb2;
CREATE OR REPLACE SYNONYM types_cap FOR user_modbd.types_cap;
CREATE OR REPLACE SYNONYM types_prov FOR user_modbd.types_prov@pdb2;
```

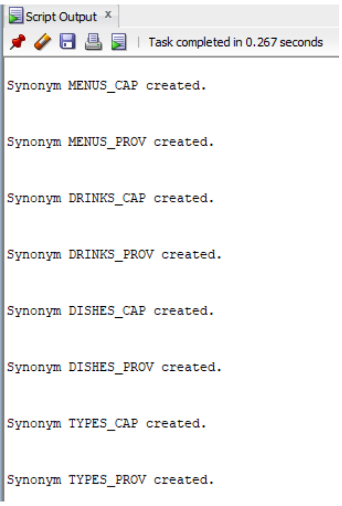
Task completed in 0.267 seconds

```
Synonym CITIES_CAP created.


Synonym CITIES_PROV created.


Synonym RESTAURANTS_CAP created.


Synonym RESTAURANTS_PROV created.


Synonym EMPLOYEES_CAP created.


Synonym EMPLOYEES_PROV created.


Synonym ORDERS_CAP created.


Synonym ORDERS_PROV created.
```



Task completed in 0.267 seconds

```
Synonym ORDERS_DRINKS_CAP created.


Synonym ORDERS_DRINKS_PROV created.


Synonym CHEFS_ORDERS_DISHES_CAP created.


Synonym CHEFS_ORDERS_DISHES_PROV created.


Synonym MENUS_CAP created.


Synonym MENUS_PROV created.


Synonym DRINKS_CAP created.


Synonym DRINKS_PROV created.
```



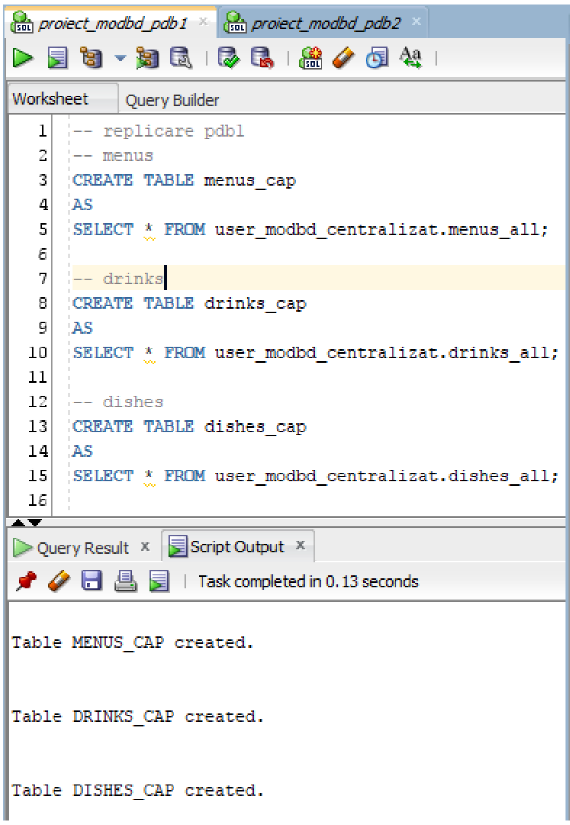
Task completed in 0.267 seconds

```
Synonym MENUS_CAP created.


Synonym MENUS_PROV created.


Synonym DRINKS_CAP created.


Synonym DRINKS_PROV created.


Synonym DISHES_CAP created.


Synonym DISHES_PROV created.


Synonym TYPES_CAP created.


Synonym TYPES_PROV created.
```

5. Asigurarea sincronizării datelor pentru relațiile replicate

Tabele menus, drinks și dishes nu depind de oraș, deoarece toate restaurantele din rețea au aceleași produse. Astfel, datele din acestea vor fi replicate. De ex, menus_cap și menus_prov vor conține aceleași înregistrări.

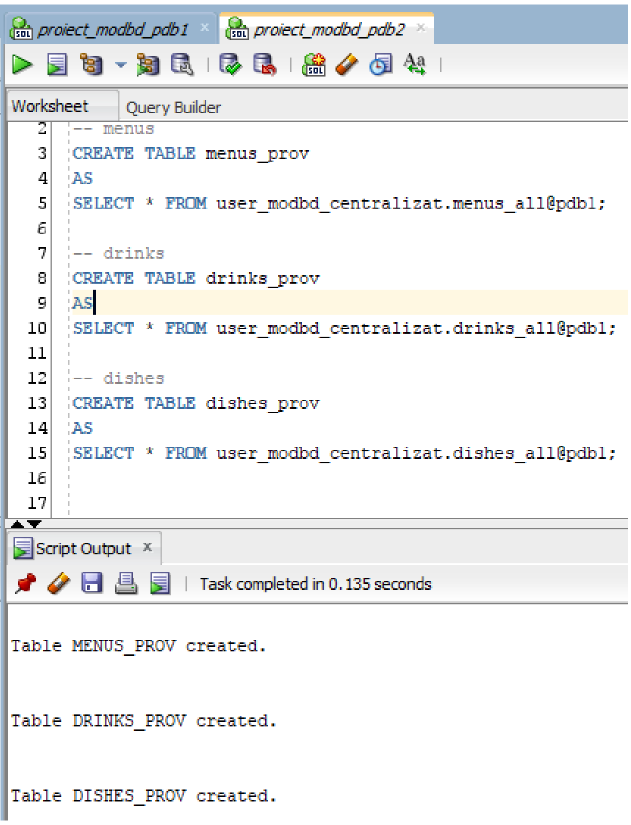
```
-- replicare pdb1
-- menus
CREATE TABLE menus_cap
AS
SELECT * FROM user_modbd_centralizat.menus_all;

-- drinks
CREATE TABLE drinks_cap
AS
SELECT * FROM user_modbd_centralizat.drinks_all;

-- dishes
CREATE TABLE dishes_cap
AS
SELECT * FROM user_modbd_centralizat.dishes_all;

-- replicare pdb2
-- menus
CREATE TABLE menus_prov
AS
SELECT * FROM user_modbd_centralizat.menus_all@pdb1;

-- drinks
CREATE TABLE drinks_prov
AS
SELECT * FROM user_modbd_centralizat.drinks_all@pdb1;

-- dishes
CREATE TABLE dishes_prov
AS
SELECT * FROM user_modbd_centralizat.dishes_all@pdb1;
```



```
-- replicare pdb1
-- menus
CREATE TABLE menus_cap
AS
SELECT * FROM user_modbd_centralizat.menus_all;

-- drinks
CREATE TABLE drinks_cap
AS
SELECT * FROM user_modbd_centralizat.drinks_all;

-- dishes
CREATE TABLE dishes_cap
AS
SELECT * FROM user_modbd_centralizat.dishes_all;
```

Task completed in 0.13 seconds

```
Table MENUS_CAP created.


Table DRINKS_CAP created.


Table DISHES_CAP created.
```



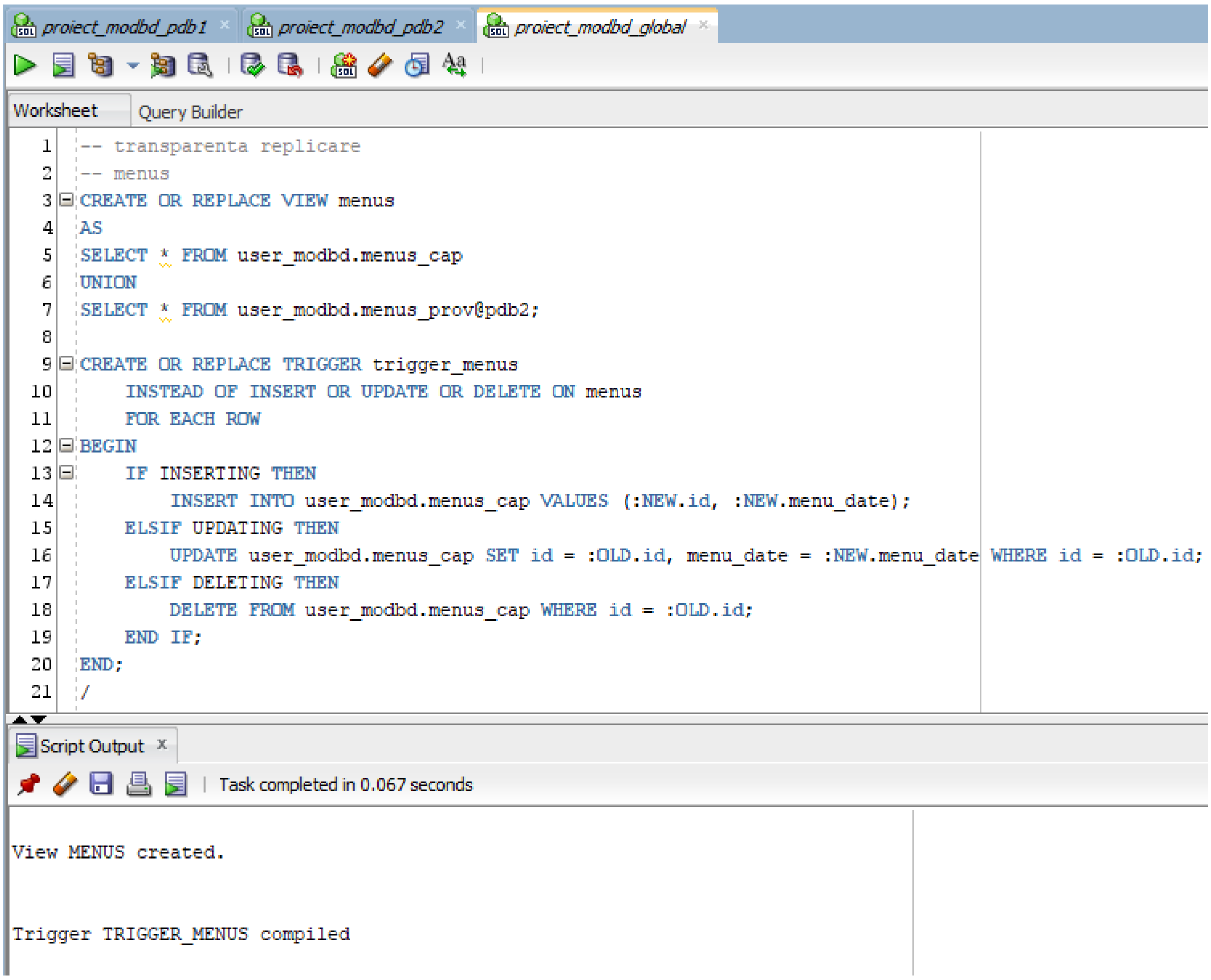
```
-- menus
CREATE TABLE menus_prov
AS
SELECT * FROM user_modbd_centralizat.menus_all@pdb1;

-- drinks
CREATE TABLE drinks_prov
AS
SELECT * FROM user_modbd_centralizat.drinks_all@pdb1;

-- dishes
CREATE TABLE dishes_prov
AS
SELECT * FROM user_modbd_centralizat.dishes_all@pdb1;
```

Task completed in 0.135 seconds

```
Table MENUS_PROV created.


Table DRINKS_PROV created.


Table DISHES_PROV created.
```

Pentru transparență, triggerele pentru insert, update și delete se vor conecta numai la tabelel din pdb1, umând ca datele să fie copiate în pdb2.

```
-- transparenta replicare
-- menus
CREATE OR REPLACE VIEW menus
AS
SELECT * FROM user_modbd.menus_cap
UNION
SELECT * FROM user_modbd.menus_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_menus
   INSTEAD OF INSERT OR UPDATE OR DELETE ON menus
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO user_modbd.menus_cap VALUES (:NEW.id, :NEW.menu_date);
   ELSIF UPDATING THEN
      UPDATE user_modbd.menus_cap SET id = :OLD.id, menu_date = :NEW.menu_date WHERE id
= :OLD.id;
   ELSIF DELETING THEN
      DELETE FROM user_modbd.menus_cap WHERE id = :OLD.id;
   END IF;
END;
/
```

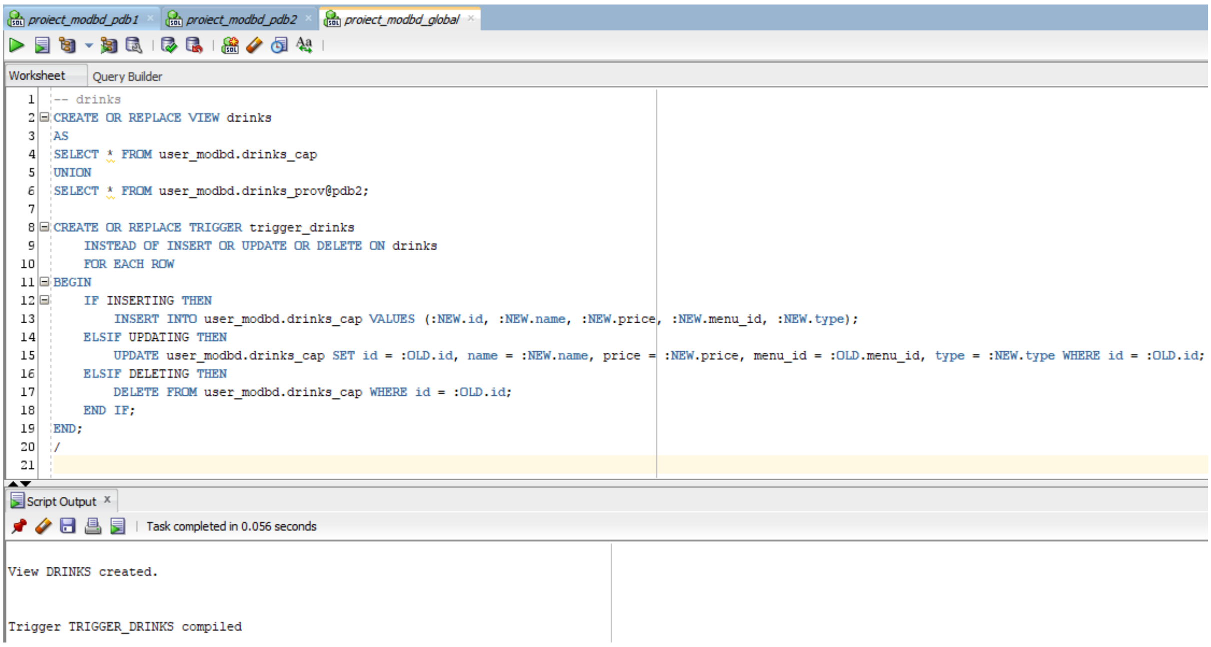


```
-- transparenta replicare
-- menus
CREATE OR REPLACE VIEW menus
AS
SELECT * FROM user_modbd.menus_cap
UNION
SELECT * FROM user_modbd.menus_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_menus
    INSTEAD OF INSERT OR UPDATE OR DELETE ON menus
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_modbd.menus_cap VALUES (:NEW.id, :NEW.menu_date);
    ELSIF UPDATING THEN
        UPDATE user_modbd.menus_cap SET id = :OLD.id, menu_date = :NEW.menu_date WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM user_modbd.menus_cap WHERE id = :OLD.id;
    END IF;
END;
/
```

Task completed in 0.067 seconds

```
View MENUS created.


Trigger TRIGGER_MENUS compiled
```

```
-- drinks
CREATE OR REPLACE VIEW drinks
AS
SELECT * FROM user_modbd.drinks_cap
UNION
SELECT * FROM user_modbd.drinks_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_drinks
   INSTEAD OF INSERT OR UPDATE OR DELETE ON drinks
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO user_modbd.drinks_cap VALUES (:NEW.id, :NEW.name, :NEW.type, :NEW.price,
:NEW.menu_id);
   ELSIF UPDATING THEN
      UPDATE user_modbd.drinks_cap SET id = :OLD.id, name = :NEW.name, type = :NEW.type,
price = :NEW.price, menu_id = :OLD.menu_id WHERE id = :OLD.id;
   ELSIF DELETING THEN
      DELETE FROM user_modbd.drinks_cap WHERE id = :OLD.id;
   END IF;
END;
/
```

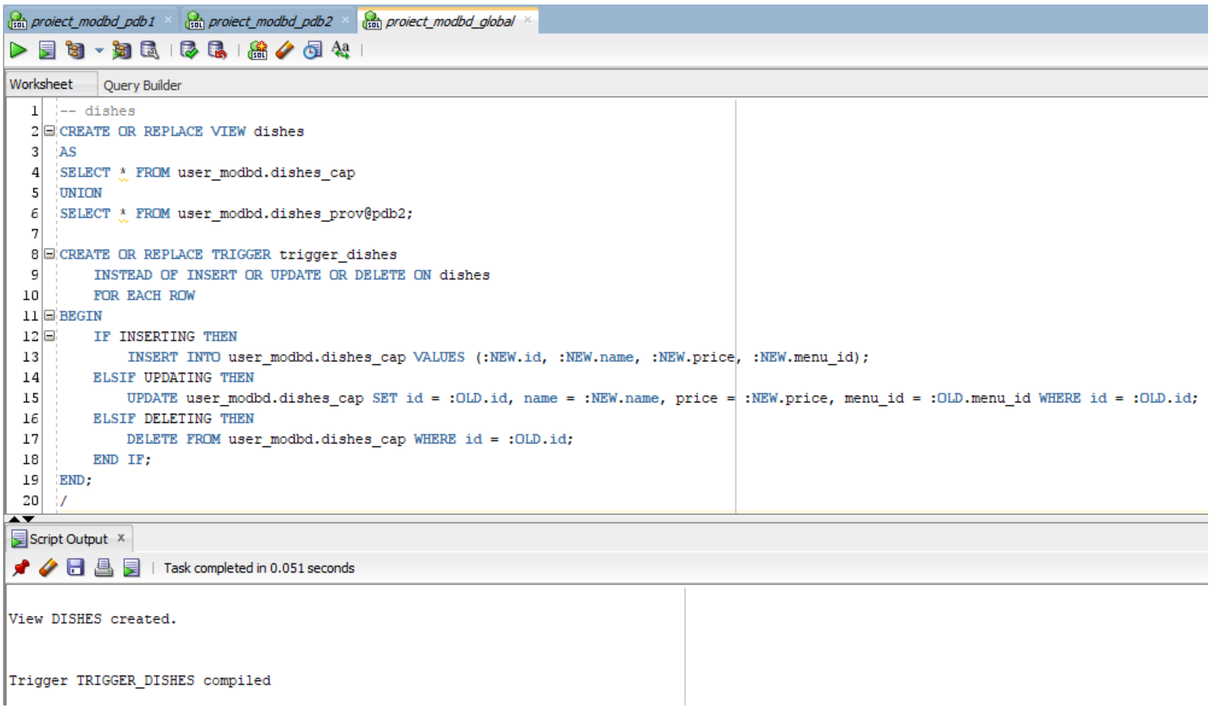


```
-- drinks
CREATE OR REPLACE VIEW drinks
AS
SELECT * FROM user_modbd.drinks_cap
UNION
SELECT * FROM user_modbd.drinks_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_drinks
    INSTEAD OF INSERT OR UPDATE OR DELETE ON drinks
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_modbd.drinks_cap VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id, :NEW.type);
    ELSIF UPDATING THEN
        UPDATE user_modbd.drinks_cap SET id = :OLD.id, name = :NEW.name, price = :NEW.price, menu_id = :OLD.menu_id, type = :NEW.type WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM user_modbd.drinks_cap WHERE id = :OLD.id;
    END IF;
END;
/
```

Task completed in 0.056 seconds

```
View DRINKS created.


Trigger TRIGGER_DRINKS compiled
```

```
-- dishes
CREATE OR REPLACE VIEW dishes
AS
SELECT * FROM user_modbd.dishes_cap
UNION
SELECT * FROM user_modbd.dishes_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_dishes
    INSTEAD OF INSERT OR UPDATE OR DELETE ON dishes
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_modbd.dishes_cap VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id);
    ELSIF UPDATING THEN
        UPDATE user_modbd.dishes_cap SET id = :OLD.id, name = :NEW.name, price = :NEW.price, menu_id = :OLD.menu_id WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM user_modbd.dishes_cap WHERE id = :OLD.id;
    END IF;
END;
/
```



```
-- dishes
CREATE OR REPLACE VIEW dishes
AS
SELECT * FROM user_modbd.dishes_cap
UNION
SELECT * FROM user_modbd.dishes_prov@pdb2;

CREATE OR REPLACE TRIGGER trigger_dishes
    INSTEAD OF INSERT OR UPDATE OR DELETE ON dishes
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_modbd.dishes_cap VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id);
    ELSIF UPDATING THEN
        UPDATE user_modbd.dishes_cap SET id = :OLD.id, name = :NEW.name, price = :NEW.price, menu_id = :OLD.menu_id WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM user_modbd.dishes_cap WHERE id = :OLD.id;
    END IF;
END;
/
```

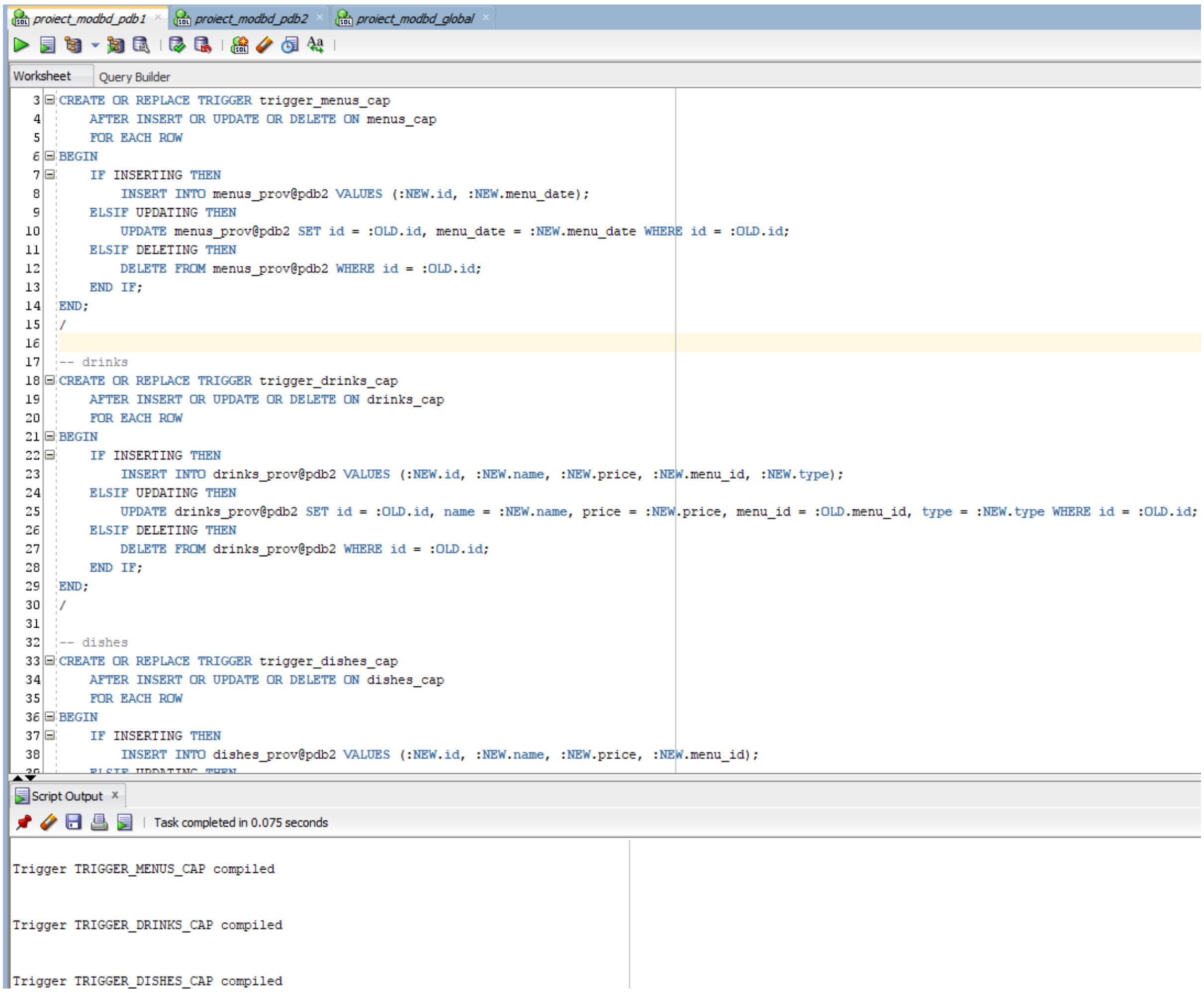
Task completed in 0.051 seconds

```
View DISHES created.


Trigger TRIGGER_DISHES compiled
```

Pentru a sincroniza datele din pdb1 în pdb2, au fost creați trigger-i care să copieze rezultatul fiecărei operații de insert, update sau delete.

```
-- replicare pdb1
-- menus
CREATE OR REPLACE TRIGGER trigger_menus_cap
   AFTER INSERT OR UPDATE OR DELETE ON menus_cap
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO menus_prov@pdb2 VALUES (:NEW.id, :NEW.menu_date);
   ELSIF UPDATING THEN
      UPDATE menus_prov@pdb2 SET id = :OLD.id, menu_date = :NEW.menu_date WHERE id =
:OLD.id;
   ELSIF DELETING THEN
      DELETE FROM menus_prov@pdb2 WHERE id = :OLD.id;
   END IF;
END;
/

-- drinks
CREATE OR REPLACE TRIGGER trigger_drinks_cap
   AFTER INSERT OR UPDATE OR DELETE ON drinks_cap
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO drinks_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.type, :NEW.price,
:NEW.menu_id);
   ELSIF UPDATING THEN
      UPDATE drinks_prov@pdb2 SET id = :OLD.id, name = :NEW.name, type = :NEW.type, price =
:NEW.price, menu_id = :OLD.menu_id WHERE id = :OLD.id;
   ELSIF DELETING THEN
      DELETE FROM drinks_prov@pdb2 WHERE id = :OLD.id;
   END IF;
END;
/
```

```
-- dishes
CREATE OR REPLACE TRIGGER trigger_dishes_cap
   AFTER INSERT OR UPDATE OR DELETE ON dishes_cap
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO dishes_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id);
   ELSIF UPDATING THEN
      UPDATE dishes_prov@pdb2 SET id = :OLD.id, name = :NEW.name, price = :NEW.price,
menu_id = :OLD.menu_id WHERE id = :OLD.id;
   ELSIF DELETING THEN
      DELETE FROM dishes_prov@pdb2 WHERE id = :OLD.id;
   END IF;
END;
/
```



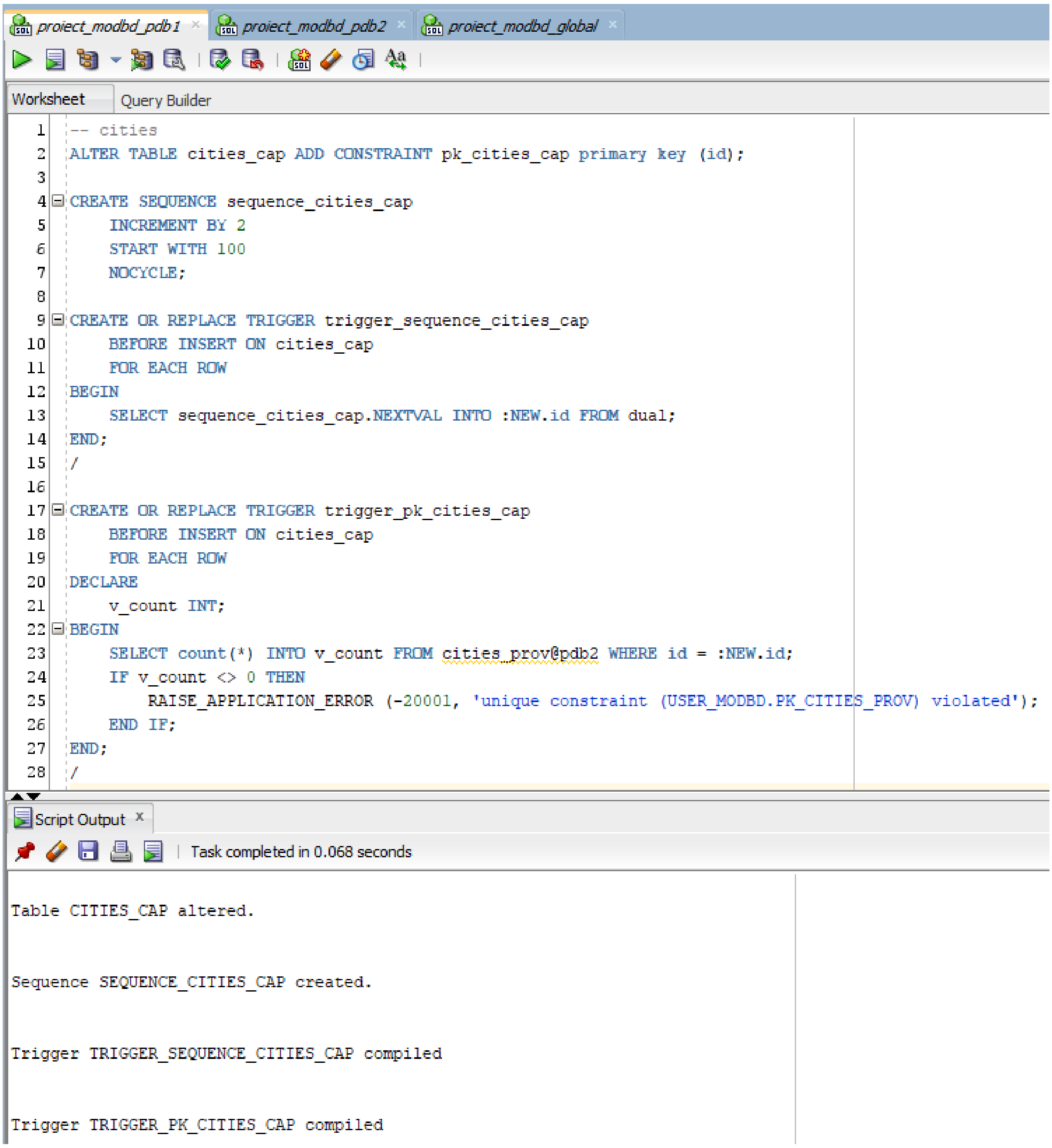
```
CREATE OR REPLACE TRIGGER trigger_menus_cap
    AFTER INSERT OR UPDATE OR DELETE ON menus_cap
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO menus_prov@pdb2 VALUES (:NEW.id, :NEW.menu_date);
    ELSIF UPDATING THEN
        UPDATE menus_prov@pdb2 SET id = :OLD.id, menu_date = :NEW.menu_date WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM menus_prov@pdb2 WHERE id = :OLD.id;
    END IF;
END;
/

-- drinks
CREATE OR REPLACE TRIGGER trigger_drinks_cap
    AFTER INSERT OR UPDATE OR DELETE ON drinks_cap
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO drinks_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id, :NEW.type);
    ELSIF UPDATING THEN
        UPDATE drinks_prov@pdb2 SET id = :OLD.id, name = :NEW.name, price = :NEW.price, menu_id = :OLD.menu_id, type = :NEW.type WHERE id = :OLD.id;
    ELSIF DELETING THEN
        DELETE FROM drinks_prov@pdb2 WHERE id = :OLD.id;
    END IF;
END;
/

-- dishes
CREATE OR REPLACE TRIGGER trigger_dishes_cap
    AFTER INSERT OR UPDATE OR DELETE ON dishes_cap
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO dishes_prov@pdb2 VALUES (:NEW.id, :NEW.name, :NEW.price, :NEW.menu_id);
    ELSIF UPDATING THEN
```

```
Trigger TRIGGER_MENUS_CAP compiled


Trigger TRIGGER_DRINKS_CAP compiled


Trigger TRIGGER_DISHES_CAP compiled
```

6. Asigurarea tuturor constrângerilor de integritate folosite în model

Pentru definirea constrângerilor, am adăugat constrângerile de cheie primară si de chei externe pentru fiecare tabel din pdb1 și pdb2. Apoi, am create cate o secvență care să genereze automat id-uri noi. Pentru tabelele replicate orizontal, înregistrările din pdb1 vor avea valori pare, iar cele din pdb2 vor avea valori impare. Am creat trigger-e care să adauge automat câte un nou id folosind secvența creată. În plus, avem câte un trigger în cazul tabelelor fragmentate orizontal care să verifice unicitatea cheii primare la nivel global: dacă id-ul din pdb1 nu există deja în pdb2 și invers.

```
-- cities
ALTER TABLE cities_cap ADD CONSTRAINT pk_cities_cap primary key (id);

CREATE SEQUENCE sequence_cities_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_cities_cap
   BEFORE INSERT ON cities_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_cities_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_cities_cap
   BEFORE INSERT ON cities_cap
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM cities_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_CITIES_PROV) violated');
   END IF;
END;
/
```

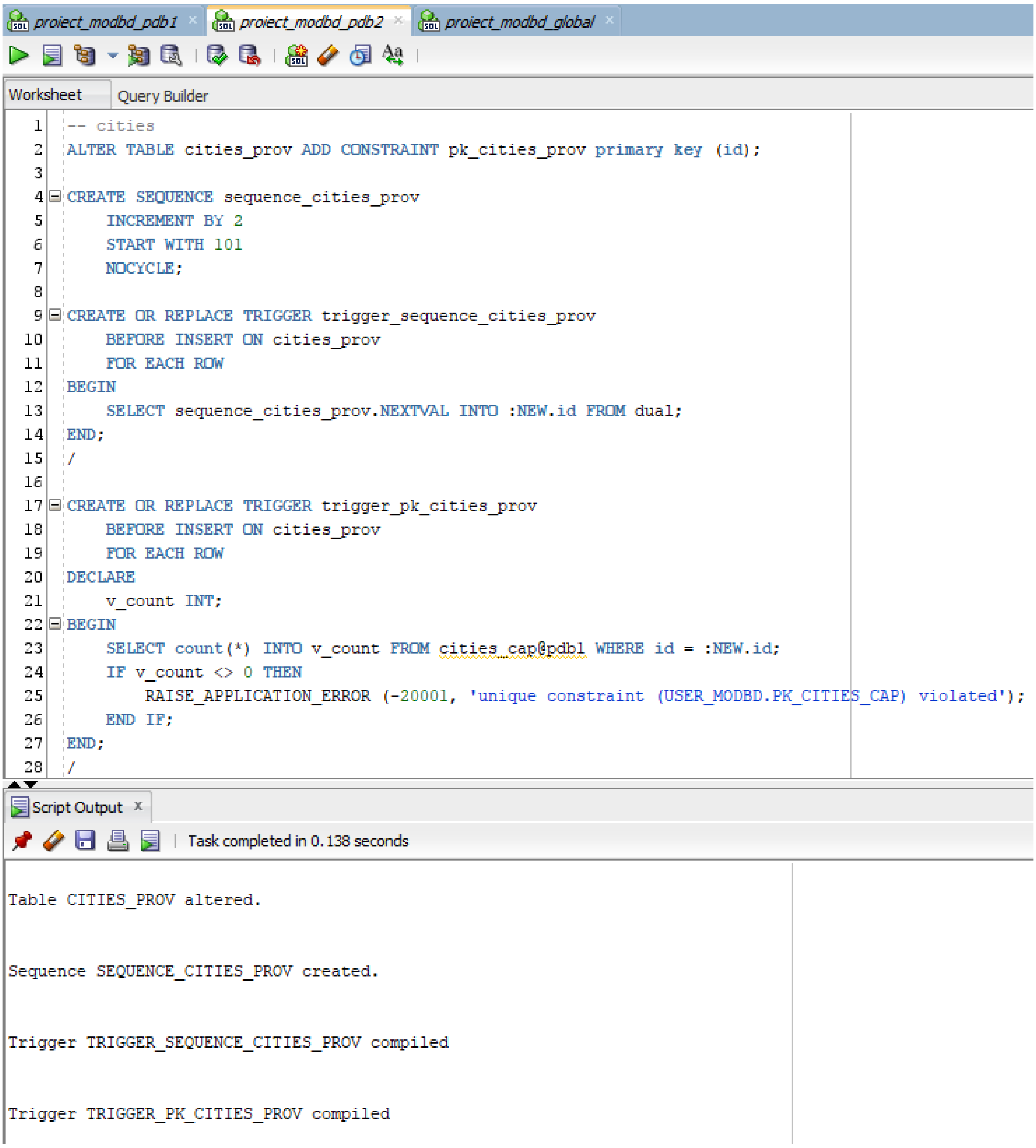


```
-- cities
ALTER TABLE cities_cap ADD CONSTRAINT pk_cities_cap primary key (id);

CREATE SEQUENCE sequence_cities_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_cities_cap
    BEFORE INSERT ON cities_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_cities_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_cities_cap
    BEFORE INSERT ON cities_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM cities_prov@pdb2 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_CITIES_PROV) violated');
    END IF;
END;
/
```

```
Table CITIES_CAP altered.


Sequence SEQUENCE_CITIES_CAP created.


Trigger TRIGGER_SEQUENCE_CITIES_CAP compiled


Trigger TRIGGER_PK_CITIES_CAP compiled
```

```
-- cities
ALTER TABLE cities_prov ADD CONSTRAINT pk_cities_prov primary key (id);

CREATE SEQUENCE sequence_cities_prov
   INCREMENT BY 2
   START WITH 101
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_cities_prov
   BEFORE INSERT ON cities_prov
   FOR EACH ROW
BEGIN
   SELECT sequence_cities_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_cities_prov
```

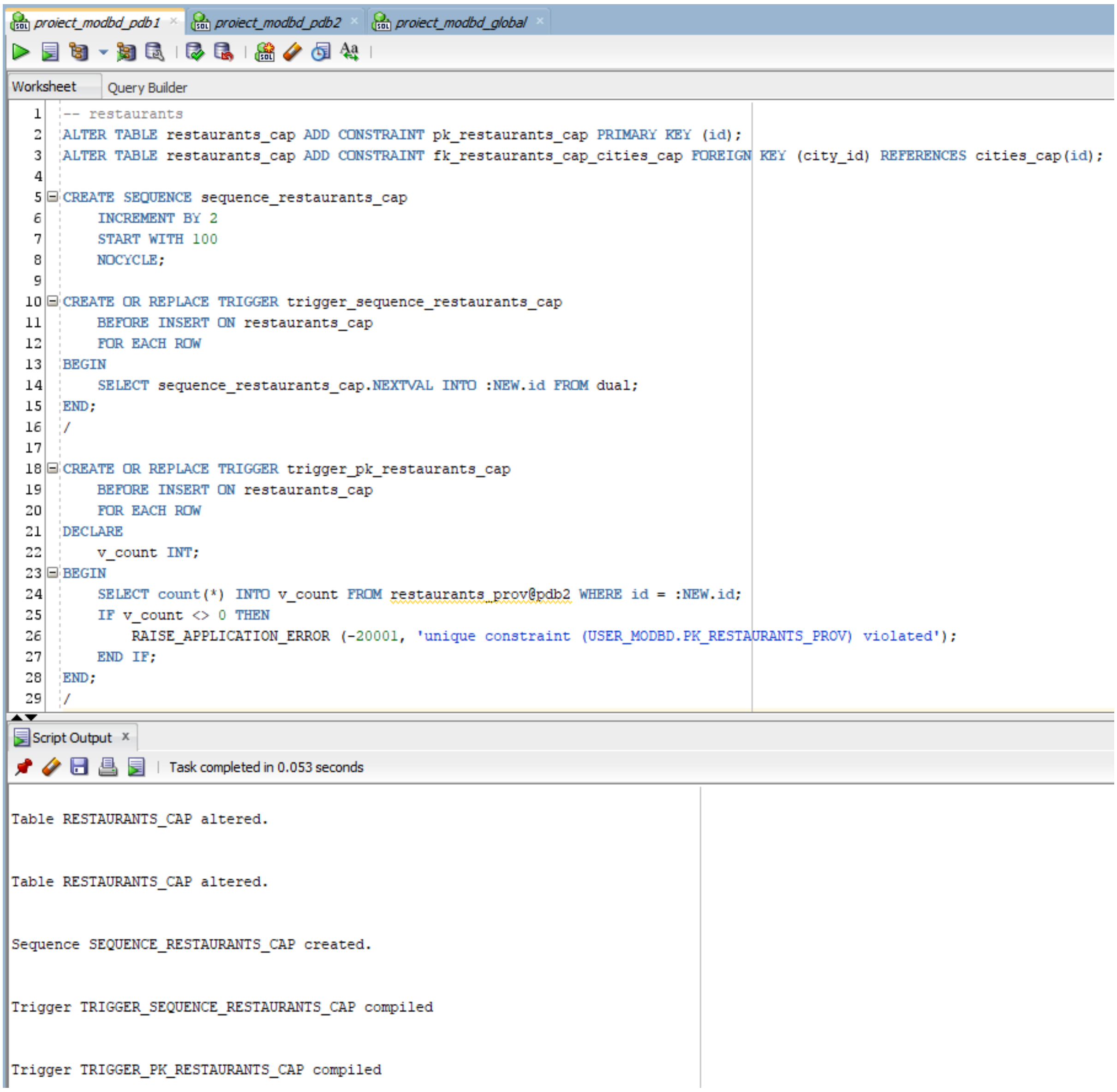
```
    BEFORE INSERT ON cities_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM cities_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_CITIES_CAP)
violated');
    END IF;
END;
/
```



```
-- cities
ALTER TABLE cities_prov ADD CONSTRAINT pk_cities_prov primary key (id);

CREATE SEQUENCE sequence_cities_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_cities_prov
    BEFORE INSERT ON cities_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_cities_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_cities_prov
    BEFORE INSERT ON cities_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM cities_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_CITIES_CAP) violated');
    END IF;
END;
/
```

```
Table CITIES_PROV altered.


Sequence SEQUENCE_CITIES_PROV created.


Trigger TRIGGER_SEQUENCE_CITIES_PROV compiled


Trigger TRIGGER_PK_CITIES_PROV compiled
```

```
-- restaurants
ALTER TABLE restaurants_cap ADD CONSTRAINT pk_restaurants_cap PRIMARY KEY (id);
ALTER TABLE restaurants_cap ADD CONSTRAINT fk_restaurants_cap_cities_cap FOREIGN KEY
(city_id) REFERENCES cities_cap(id);

CREATE SEQUENCE sequence_restaurants_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_restaurants_cap
   BEFORE INSERT ON restaurants_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_restaurants_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_restaurants_cap
   BEFORE INSERT ON restaurants_cap
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM restaurants_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_RESTAURANTS_PROV) violated');
   END IF;
END;
/
```



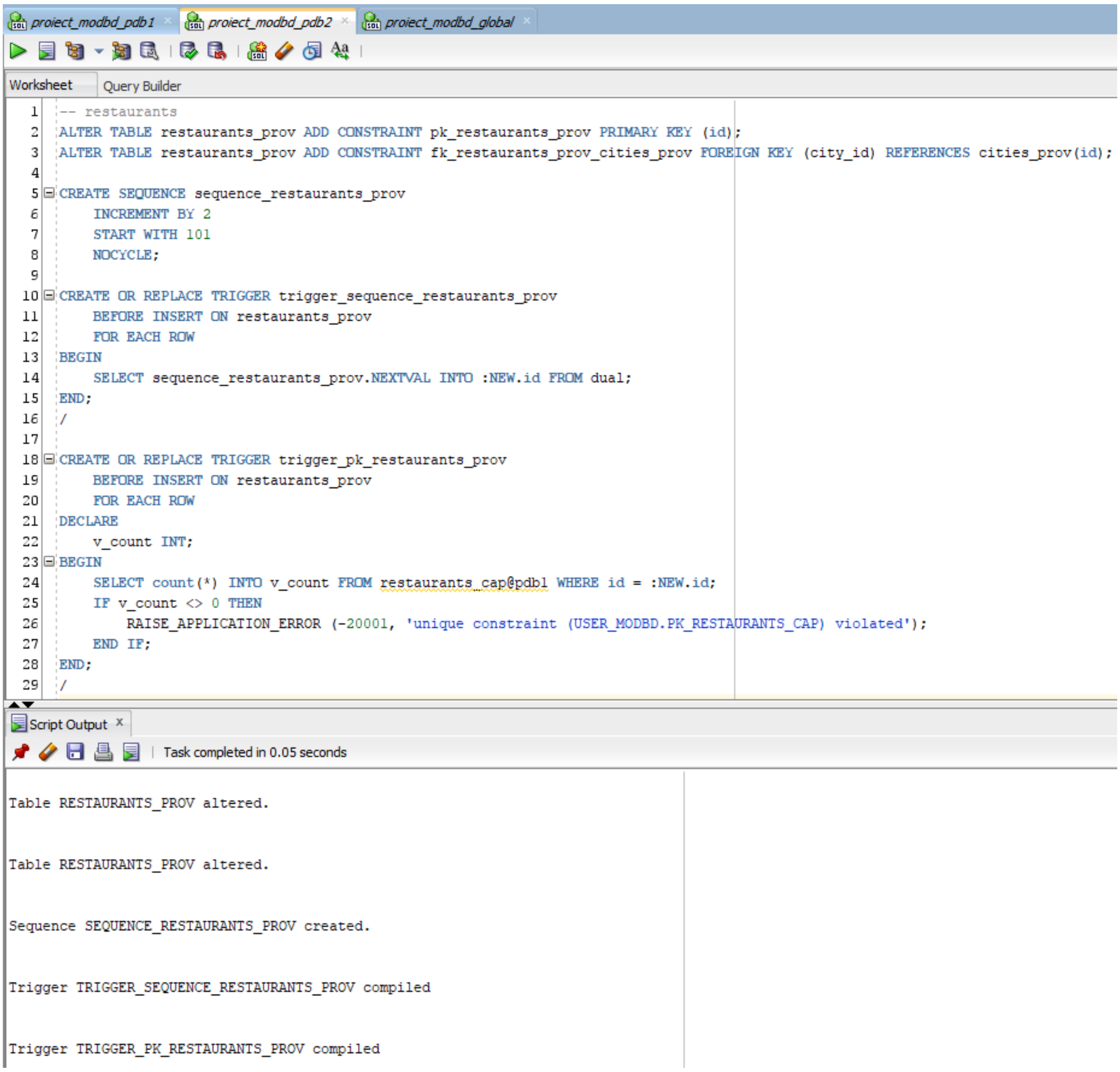
```
-- restaurants
ALTER TABLE restaurants_cap ADD CONSTRAINT pk_restaurants_cap PRIMARY KEY (id);
ALTER TABLE restaurants_cap ADD CONSTRAINT fk_restaurants_cap_cities_cap FOREIGN KEY (city_id) REFERENCES cities_cap(id);

CREATE SEQUENCE sequence_restaurants_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_restaurants_cap
    BEFORE INSERT ON restaurants_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_restaurants_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_restaurants_cap
    BEFORE INSERT ON restaurants_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM restaurants_prov@pdb2 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_RESTAURANTS_PROV) violated');
    END IF;
END;
/
```

Task completed in 0.053 seconds

```
Table RESTAURANTS_CAP altered.


Table RESTAURANTS_CAP altered.


Sequence SEQUENCE_RESTAURANTS_CAP created.


Trigger TRIGGER_SEQUENCE_RESTAURANTS_CAP compiled


Trigger TRIGGER_PK_RESTAURANTS_CAP compiled
```

```
-- restaurants
ALTER TABLE restaurants_prov ADD CONSTRAINT pk_restaurants_prov PRIMARY KEY (id);
ALTER TABLE restaurants_prov ADD CONSTRAINT fk_restaurants_prov_cities_prov FOREIGN KEY
(city_id) REFERENCES cities_prov(id);

CREATE SEQUENCE sequence_restaurants_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_restaurants_prov
    BEFORE INSERT ON restaurants_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_restaurants_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_restaurants_prov
    BEFORE INSERT ON restaurants_prov
    FOR EACH ROW
```

```
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM restaurants_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_RESTAURANTS_CAP) violated');
    END IF;
END;
/
```



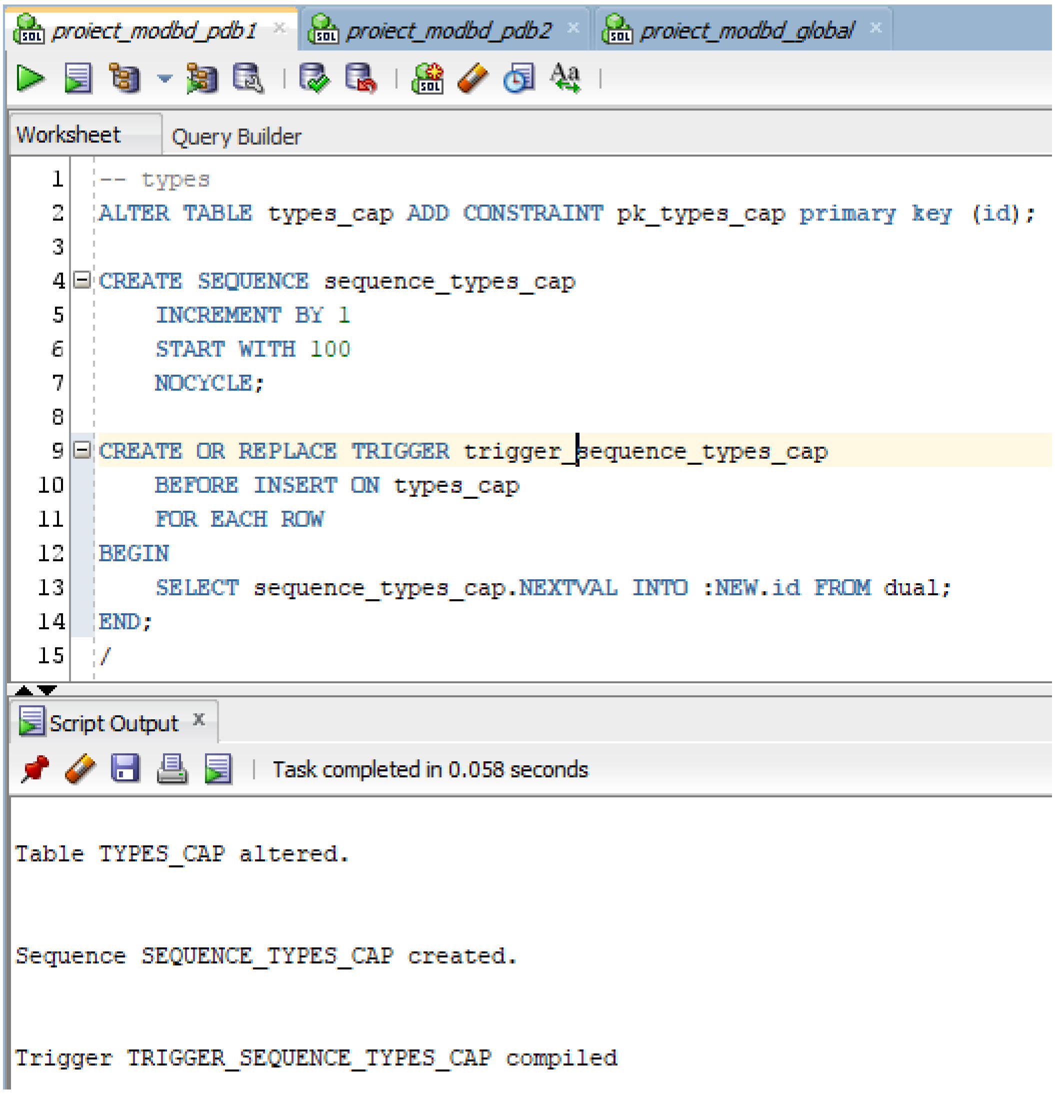
```
-- restaurants
ALTER TABLE restaurants_prov ADD CONSTRAINT pk_restaurants_prov PRIMARY KEY (id);
ALTER TABLE restaurants_prov ADD CONSTRAINT fk_restaurants_prov_cities_prov FOREIGN KEY (city_id) REFERENCES cities_prov(id);

CREATE SEQUENCE sequence_restaurants_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_restaurants_prov
    BEFORE INSERT ON restaurants_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_restaurants_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_restaurants_prov
    BEFORE INSERT ON restaurants_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM restaurants_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_RESTAURANTS_CAP) violated');
    END IF;
END;
/
```

Task completed in 0.05 seconds

```
Table RESTAURANTS_PROV altered.


Table RESTAURANTS_PROV altered.


Sequence SEQUENCE_RESTAURANTS_PROV created.


Trigger TRIGGER_SEQUENCE_RESTAURANTS_PROV compiled


Trigger TRIGGER_PK_RESTAURANTS_PROV compiled
```

```
-- types
ALTER TABLE types_cap ADD CONSTRAINT pk_types_cap primary key (id);

CREATE SEQUENCE sequence_types_cap
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_types_cap
    BEFORE INSERT ON types_cap
    FOR EACH ROW
```

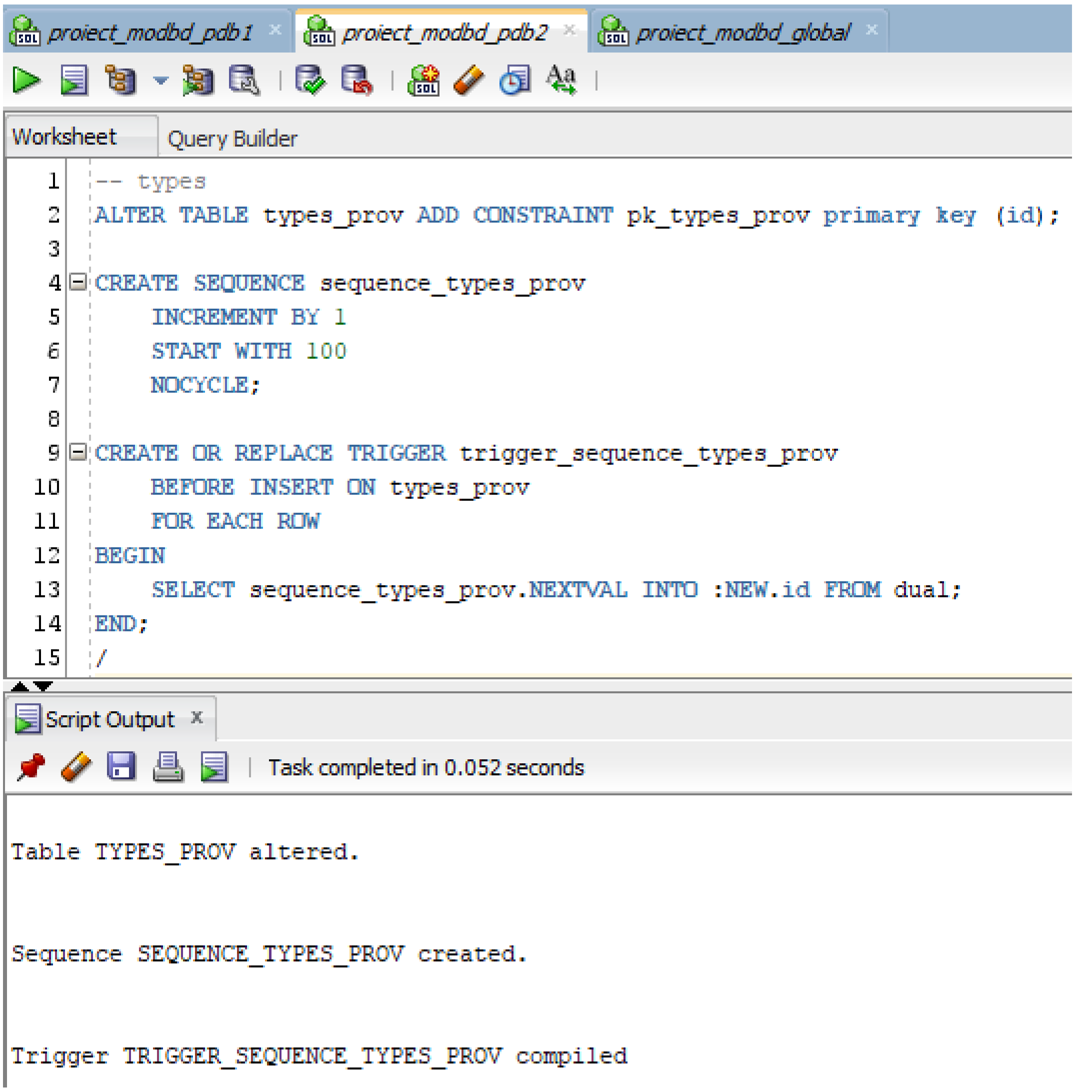
```
BEGIN
   SELECT sequence_types_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```



```
-- types
ALTER TABLE types_cap ADD CONSTRAINT pk_types_cap primary key (id);

CREATE SEQUENCE sequence_types_cap
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_types_cap
    BEFORE INSERT ON types_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_types_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

```
Table TYPES_CAP altered.


Sequence SEQUENCE_TYPES_CAP created.


Trigger TRIGGER_SEQUENCE_TYPES_CAP compiled
```

```
-- types
ALTER TABLE types_prov ADD CONSTRAINT pk_types_prov primary key (id);

CREATE SEQUENCE sequence_types_prov
   INCREMENT BY 1
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_types_prov
   BEFORE INSERT ON types_prov
   FOR EACH ROW
BEGIN
   SELECT sequence_types_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

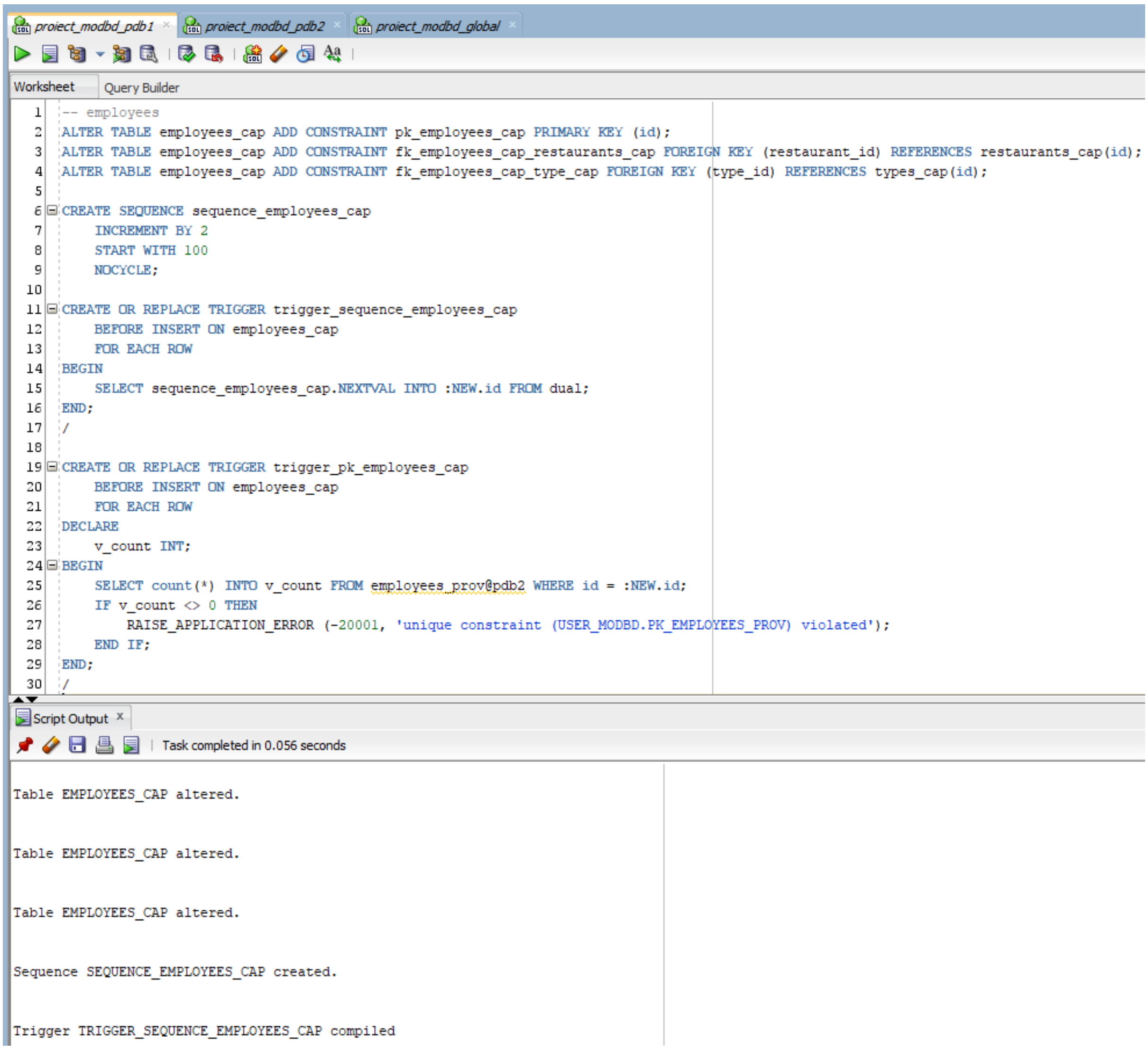


```
-- types
ALTER TABLE types_prov ADD CONSTRAINT pk_types_prov primary key (id);

CREATE SEQUENCE sequence_types_prov
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_types_prov
    BEFORE INSERT ON types_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_types_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

```
Table TYPES_PROV altered.


Sequence SEQUENCE_TYPES_PROV created.


Trigger TRIGGER_SEQUENCE_TYPES_PROV compiled
```

```
-- employees
ALTER TABLE employees_cap ADD CONSTRAINT pk_employees_cap PRIMARY KEY (id);
ALTER TABLE employees_cap ADD CONSTRAINT fk_employees_cap_restaurants_cap FOREIGN
KEY (restaurant_id) REFERENCES restaurants_cap(id);
ALTER TABLE employees_cap ADD CONSTRAINT fk_employees_cap_type_cap FOREIGN KEY
(type_id) REFERENCES types_cap(id);

CREATE SEQUENCE sequence_employees_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_employees_cap
   BEFORE INSERT ON employees_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_employees_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_employees_cap
   BEFORE INSERT ON employees_cap
   FOR EACH ROW
DECLARE
```

```
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM employees_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_EMPLOYEES_PROV) violated');
   END IF;
END;
/
```



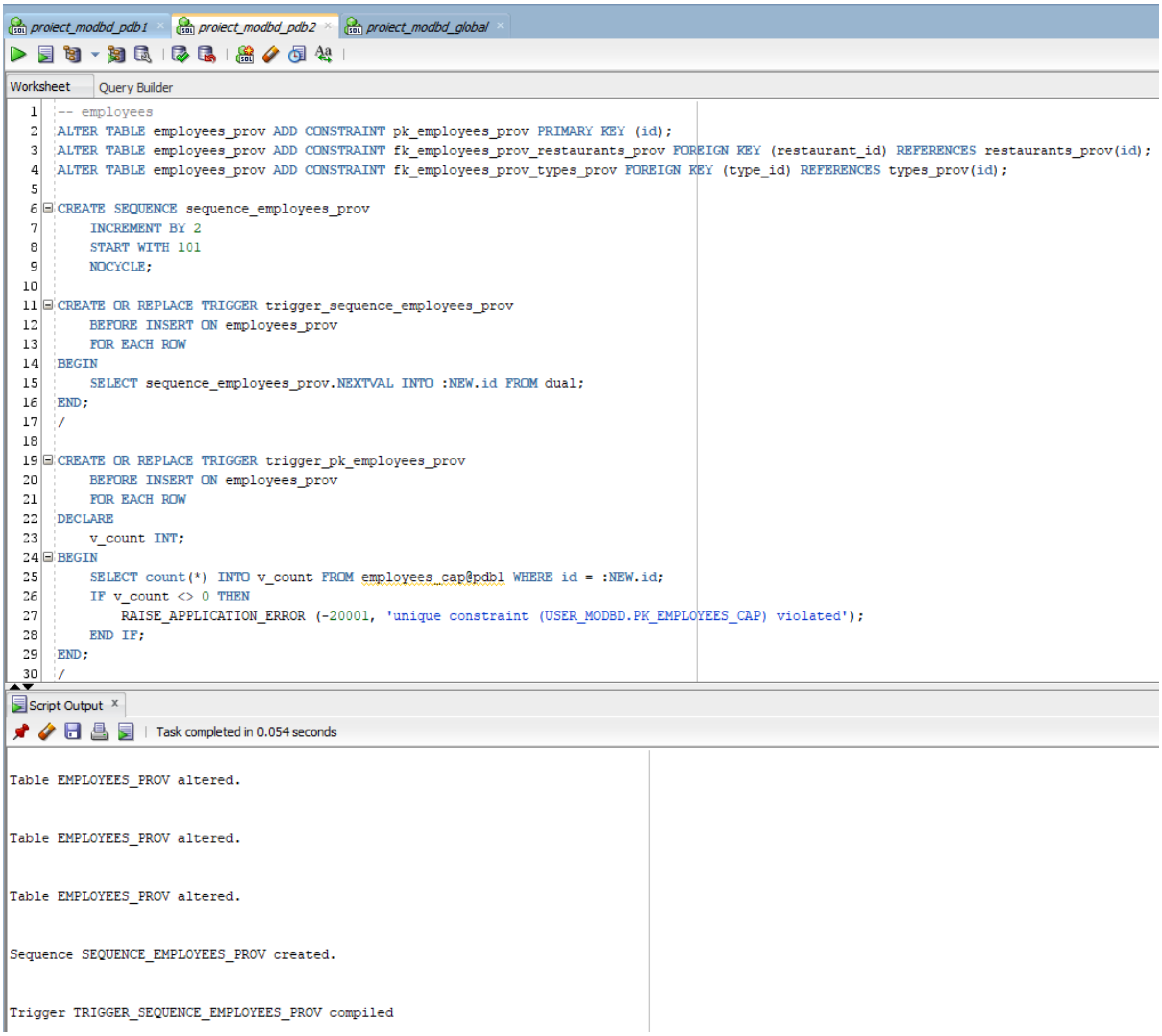
```
-- employees
ALTER TABLE employees_cap ADD CONSTRAINT pk_employees_cap PRIMARY KEY (id);
ALTER TABLE employees_cap ADD CONSTRAINT fk_employees_cap_restaurants_cap FOREIGN KEY (restaurant_id) REFERENCES restaurants_cap(id);
ALTER TABLE employees_cap ADD CONSTRAINT fk_employees_cap_type_cap FOREIGN KEY (type_id) REFERENCES types_cap(id);

CREATE SEQUENCE sequence_employees_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_employees_cap
    BEFORE INSERT ON employees_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_employees_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_employees_cap
    BEFORE INSERT ON employees_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM employees_prov@pdb2 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_EMPLOYEES_PROV) violated');
    END IF;
END;
/
```

Task completed in 0.056 seconds

```
Table EMPLOYEES_CAP altered.


Table EMPLOYEES_CAP altered.


Table EMPLOYEES_CAP altered.


Sequence SEQUENCE_EMPLOYEES_CAP created.


Trigger TRIGGER_SEQUENCE_EMPLOYEES_CAP compiled
```

```
-- employees
ALTER TABLE employees_prov ADD CONSTRAINT pk_employees_prov PRIMARY KEY (id);
ALTER TABLE employees_prov ADD CONSTRAINT fk_employees_prov_restaurants_prov FOREIGN
KEY (restaurant_id) REFERENCES restaurants_prov(id);
ALTER TABLE employees_prov ADD CONSTRAINT fk_employees_prov_types_prov FOREIGN KEY
(type_id) REFERENCES types_prov(id);

CREATE SEQUENCE sequence_employees_prov
   INCREMENT BY 2
   START WITH 101
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_employees_prov
   BEFORE INSERT ON employees_prov
```

```
    FOR EACH ROW
BEGIN
    SELECT sequence_employees_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_employees_prov
    BEFORE INSERT ON employees_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM employees_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_EMPLOYEES_CAP) violated');
    END IF;
END;
/
```



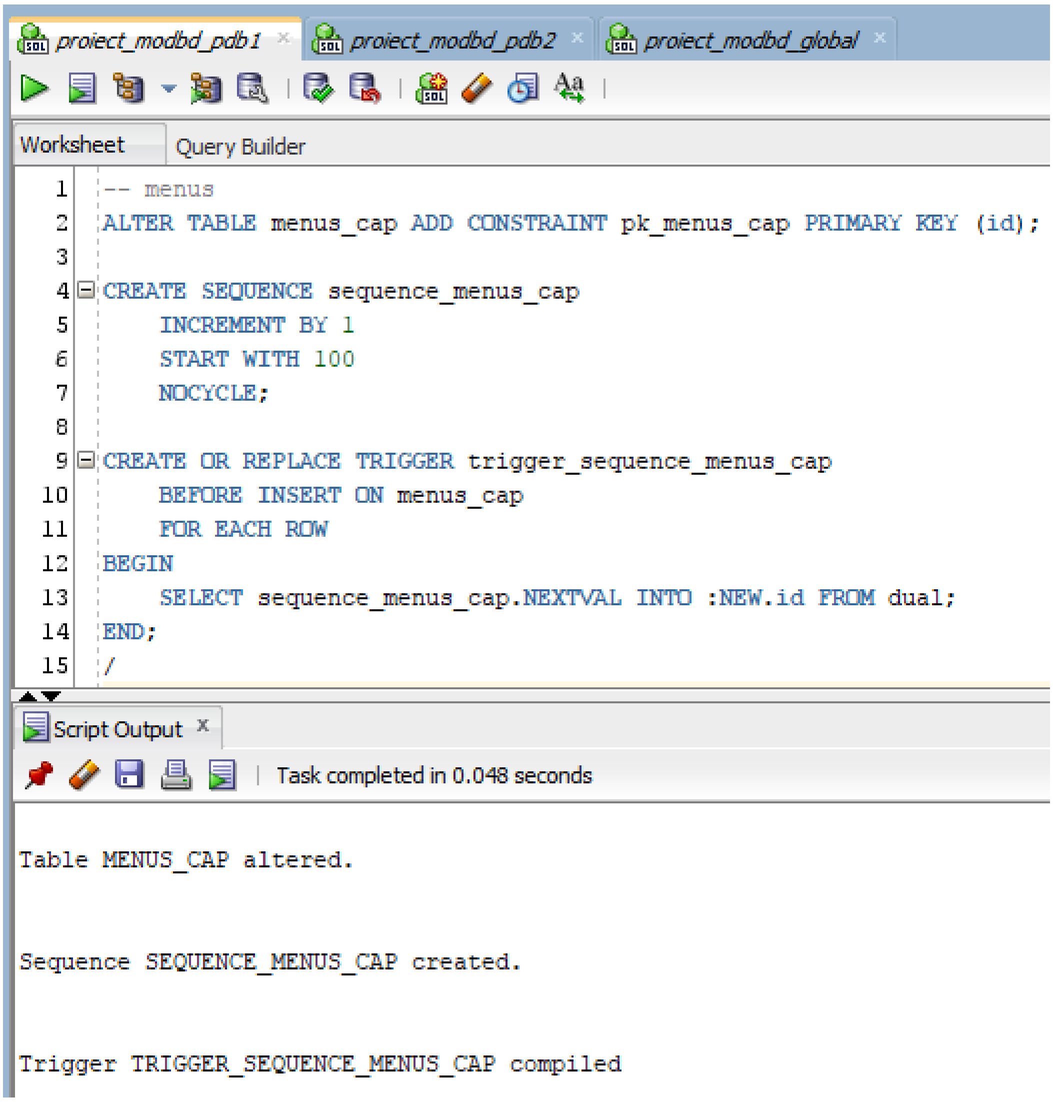
```
-- employees
ALTER TABLE employees_prov ADD CONSTRAINT pk_employees_prov PRIMARY KEY (id);
ALTER TABLE employees_prov ADD CONSTRAINT fk_employees_prov_restaurants_prov FOREIGN KEY (restaurant_id) REFERENCES restaurants_prov(id);
ALTER TABLE employees_prov ADD CONSTRAINT fk_employees_prov_types_prov FOREIGN KEY (type_id) REFERENCES types_prov(id);

CREATE SEQUENCE sequence_employees_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_employees_prov
    BEFORE INSERT ON employees_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_employees_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_employees_prov
    BEFORE INSERT ON employees_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM employees_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_EMPLOYEES_CAP) violated');
    END IF;
END;
/
```

Task completed in 0.054 seconds

```
Table EMPLOYEES_PROV altered.


Table EMPLOYEES_PROV altered.


Table EMPLOYEES_PROV altered.


Sequence SEQUENCE_EMPLOYEES_PROV created.


Trigger TRIGGER_SEQUENCE_EMPLOYEES_PROV compiled
```

```
-- menus
ALTER TABLE menus_cap ADD CONSTRAINT pk_menus_cap PRIMARY KEY (id);

CREATE SEQUENCE sequence_menus_cap
    INCREMENT BY 1
```

```
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_menus_cap
   BEFORE INSERT ON menus_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_menus_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

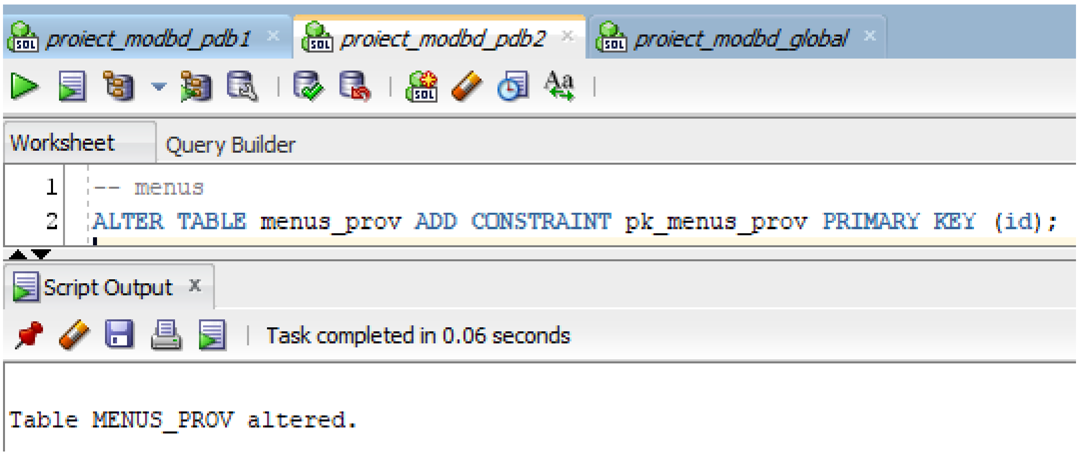


```
-- menus
ALTER TABLE menus_cap ADD CONSTRAINT pk_menus_cap PRIMARY KEY (id);

CREATE SEQUENCE sequence_menus_cap
     INCREMENT BY 1
     START WITH 100
     NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_menus_cap
     BEFORE INSERT ON menus_cap
     FOR EACH ROW
BEGIN
     SELECT sequence_menus_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

Task completed in 0.048 seconds

```
Table MENUS_CAP altered.


Sequence SEQUENCE_MENUS_CAP created.


Trigger TRIGGER_SEQUENCE_MENUS_CAP compiled
```

```
-- menus
ALTER TABLE menus_prov ADD CONSTRAINT pk_menus_prov PRIMARY KEY (id);
```

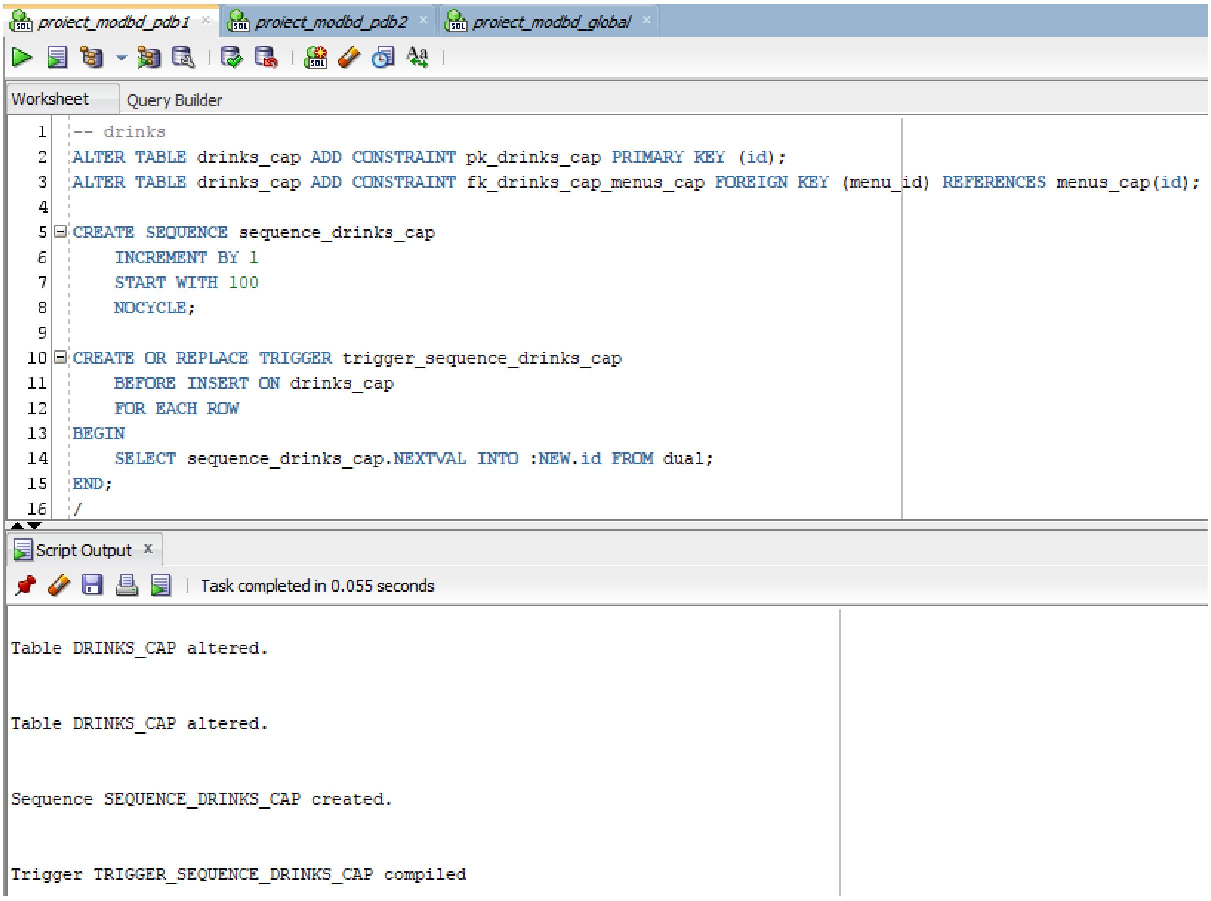


```
-- menus
ALTER TABLE menus_prov ADD CONSTRAINT pk_menus_prov PRIMARY KEY (id);
```

Task completed in 0.06 seconds

```
Table MENUS_PROV altered.
```

```
-- drinks
ALTER TABLE drinks_cap ADD CONSTRAINT pk_drinks_cap PRIMARY KEY (id);
ALTER TABLE drinks_cap ADD CONSTRAINT fk_drinks_cap_menus_cap FOREIGN KEY (menu_id)
REFERENCES menus_cap(id);

CREATE SEQUENCE sequence_drinks_cap
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_drinks_cap
    BEFORE INSERT ON drinks_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_drinks_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

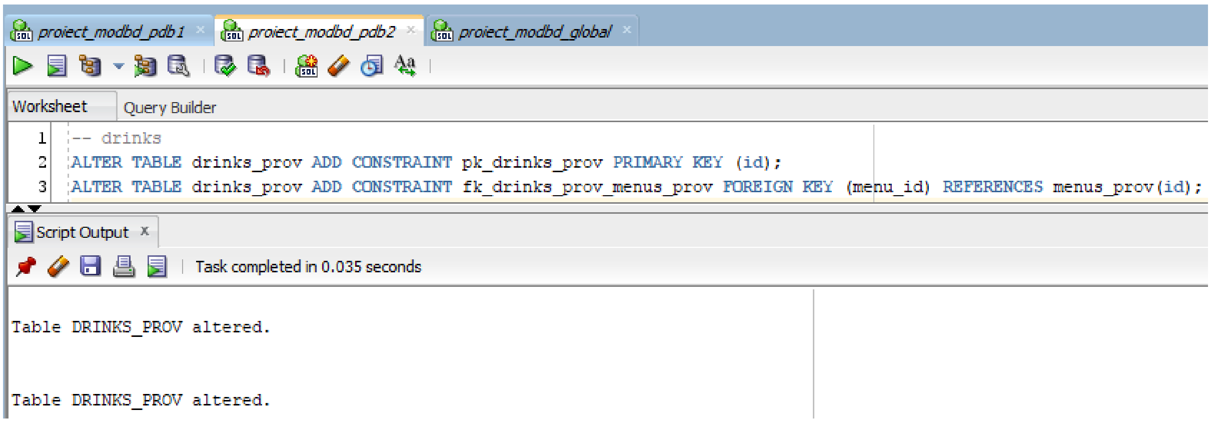


```
-- drinks
ALTER TABLE drinks_cap ADD CONSTRAINT pk_drinks_cap PRIMARY KEY (id);
ALTER TABLE drinks_cap ADD CONSTRAINT fk_drinks_cap_menus_cap FOREIGN KEY (menu_id) REFERENCES menus_cap(id);

CREATE SEQUENCE sequence_drinks_cap
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_drinks_cap
    BEFORE INSERT ON drinks_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_drinks_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

Task completed in 0.055 seconds

```
Table DRINKS_CAP altered.


Table DRINKS_CAP altered.


Sequence SEQUENCE_DRINKS_CAP created.


Trigger TRIGGER_SEQUENCE_DRINKS_CAP compiled
```

-- drinks
ALTER TABLE drinks_prov ADD CONSTRAINT pk_drinks_prov PRIMARY KEY (id);
ALTER TABLE drinks_prov ADD CONSTRAINT fk_drinks_prov_menus_prov FOREIGN KEY (menu_id) REFERENCES menus_prov(id);



```
-- drinks
ALTER TABLE drinks_prov ADD CONSTRAINT pk_drinks_prov PRIMARY KEY (id);
ALTER TABLE drinks_prov ADD CONSTRAINT fk_drinks_prov_menus_prov FOREIGN KEY (menu_id) REFERENCES menus_prov(id);
```

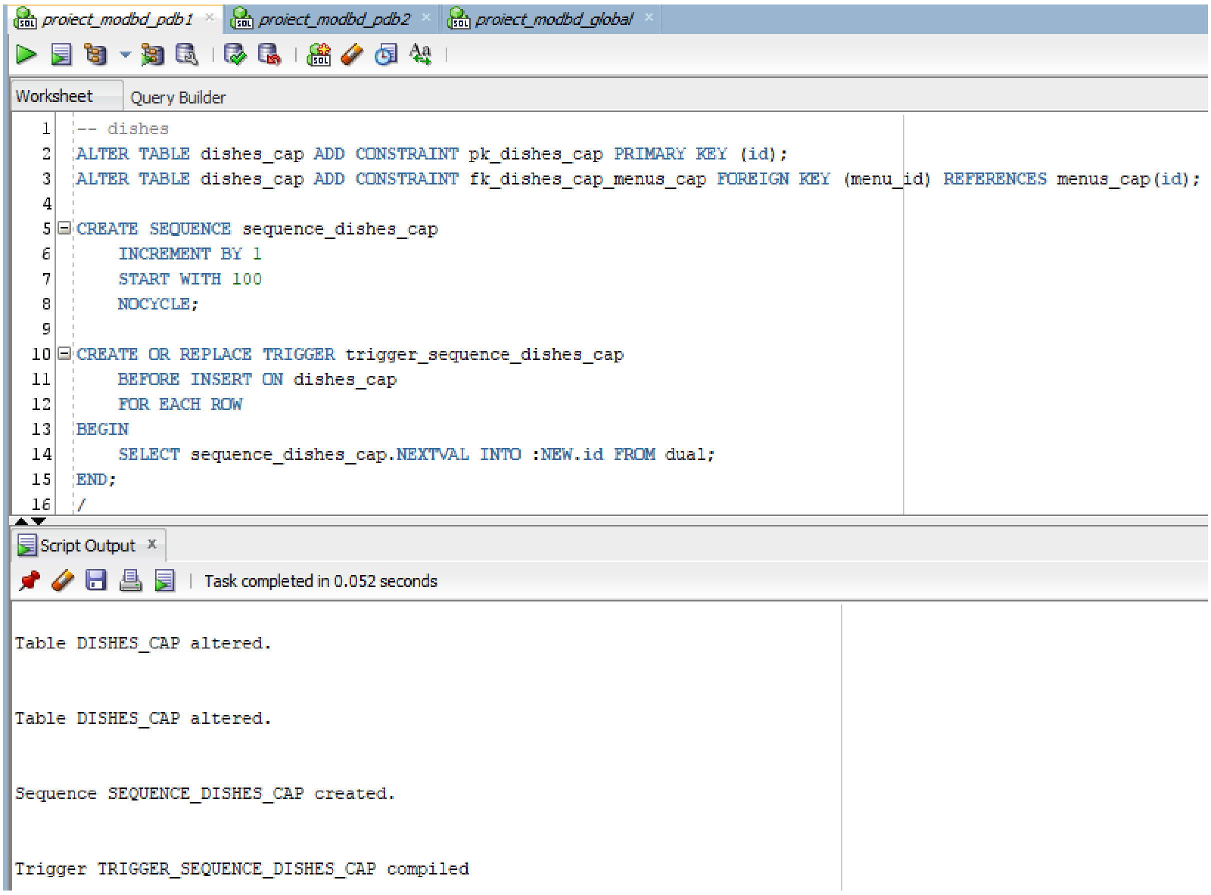
Task completed in 0.035 seconds

```
Table DRINKS_PROV altered.


Table DRINKS_PROV altered.
```

-- dishes
ALTER TABLE dishes_cap ADD CONSTRAINT pk_dishes_cap PRIMARY KEY (id);
ALTER TABLE dishes_cap ADD CONSTRAINT fk_dishes_cap_menus_cap FOREIGN KEY (menu_id) REFERENCES menus_cap(id);

```
CREATE SEQUENCE sequence_dishes_cap
   INCREMENT BY 1
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_dishes_cap
```

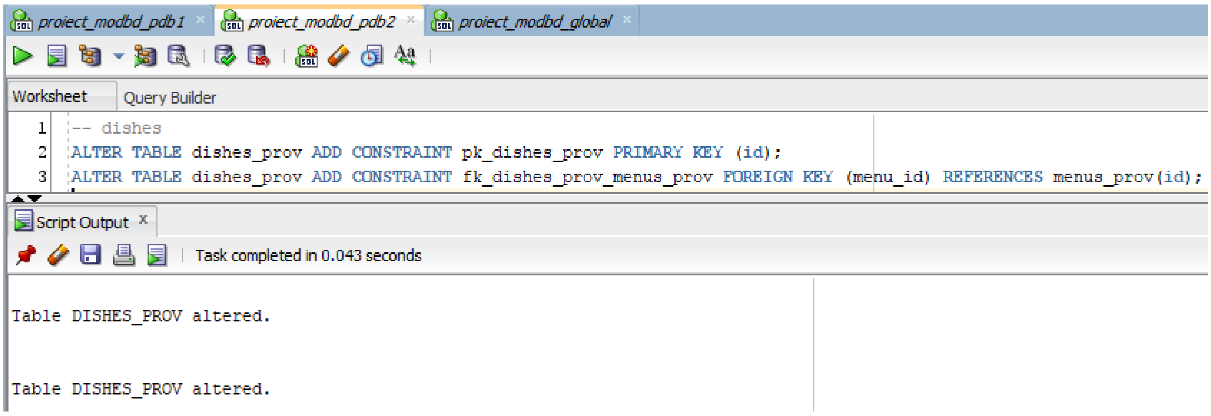
```
    BEFORE INSERT ON dishes_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_dishes_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```



```
-- dishes
ALTER TABLE dishes_cap ADD CONSTRAINT pk_dishes_cap PRIMARY KEY (id);
ALTER TABLE dishes_cap ADD CONSTRAINT fk_dishes_cap_menus_cap FOREIGN KEY (menu_id) REFERENCES menus_cap(id);

CREATE SEQUENCE sequence_dishes_cap
    INCREMENT BY 1
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_dishes_cap
    BEFORE INSERT ON dishes_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_dishes_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

```
Table DISHES_CAP altered.


Table DISHES_CAP altered.


Sequence SEQUENCE_DISHES_CAP created.


Trigger TRIGGER_SEQUENCE_DISHES_CAP compiled
```

```
-- dishes
ALTER TABLE dishes_prov ADD CONSTRAINT pk_dishes_prov PRIMARY KEY (id);
ALTER TABLE dishes_prov ADD CONSTRAINT fk_dishes_prov_menus_prov FOREIGN KEY
(menu_id) REFERENCES menus_prov(id);
```

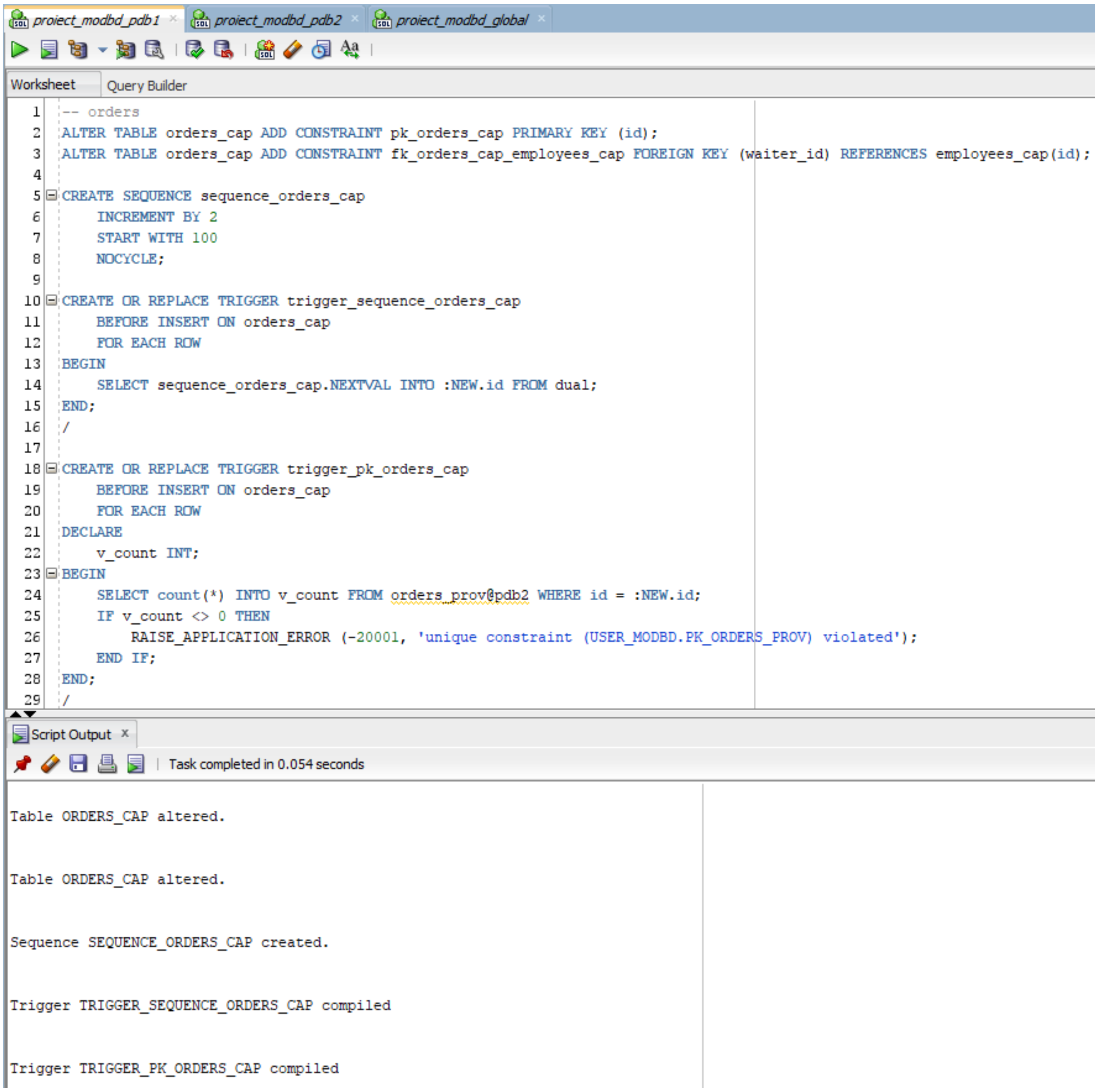


```
-- dishes
ALTER TABLE dishes_prov ADD CONSTRAINT pk_dishes_prov PRIMARY KEY (id);
ALTER TABLE dishes_prov ADD CONSTRAINT fk_dishes_prov_menus_prov FOREIGN KEY (menu_id) REFERENCES menus_prov(id);
```

```
Table DISHES_PROV altered.


Table DISHES_PROV altered.
```

```
-- orders
ALTER TABLE orders_cap ADD CONSTRAINT pk_orders_cap PRIMARY KEY (id);
ALTER TABLE orders_cap ADD CONSTRAINT fk_orders_cap_employees_cap FOREIGN KEY
(waiter_id) REFERENCES employees_cap(id);
```

```
CREATE SEQUENCE sequence_orders_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_cap
   BEFORE INSERT ON orders_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_orders_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_cap
   BEFORE INSERT ON orders_cap
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM orders_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_ORDERS_PROV) violated');
   END IF;
END;
/
```

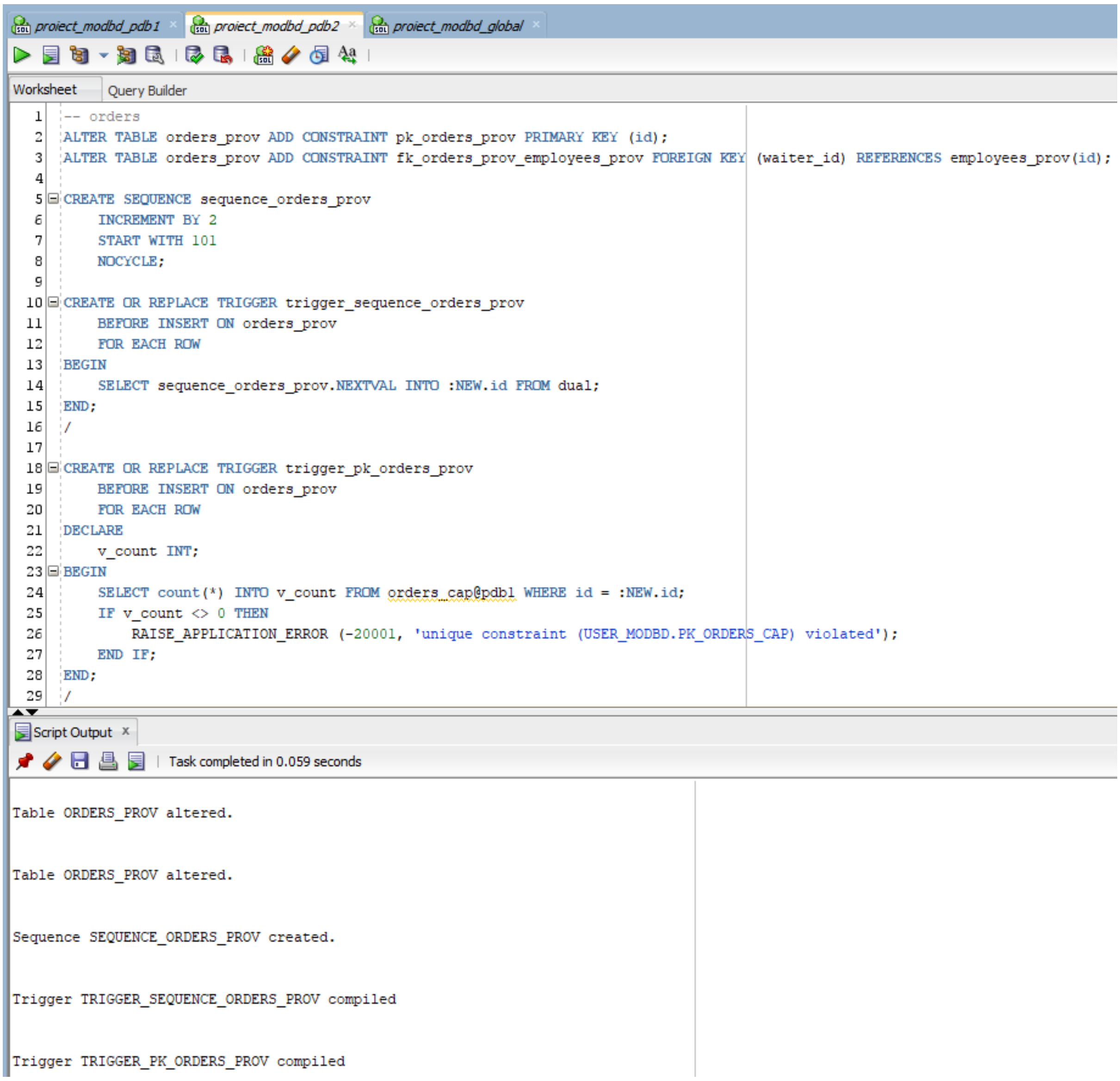


```
-- orders
ALTER TABLE orders_cap ADD CONSTRAINT pk_orders_cap PRIMARY KEY (id);
ALTER TABLE orders_cap ADD CONSTRAINT fk_orders_cap_employees_cap FOREIGN KEY (waiter_id) REFERENCES employees_cap(id);

CREATE SEQUENCE sequence_orders_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_cap
    BEFORE INSERT ON orders_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_orders_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_cap
    BEFORE INSERT ON orders_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM orders_prov@pdb2 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_ORDERS_PROV) violated');
    END IF;
END;
/
```

```
Table ORDERS_CAP altered.


Table ORDERS_CAP altered.


Sequence SEQUENCE_ORDERS_CAP created.


Trigger TRIGGER_SEQUENCE_ORDERS_CAP compiled


Trigger TRIGGER_PK_ORDERS_CAP compiled
```

```
-- orders
ALTER TABLE orders_prov ADD CONSTRAINT pk_orders_prov PRIMARY KEY (id);
ALTER TABLE orders_prov ADD CONSTRAINT fk_orders_prov_employees_prov FOREIGN KEY
(waiter_id) REFERENCES employees_prov(id);

CREATE SEQUENCE sequence_orders_prov
   INCREMENT BY 2
   START WITH 101
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_prov
   BEFORE INSERT ON orders_prov
   FOR EACH ROW
BEGIN
   SELECT sequence_orders_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_prov
   BEFORE INSERT ON orders_prov
   FOR EACH ROW
```

```
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM orders_cap@pdb1 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_ORDERS_CAP) violated');
   END IF;
END;
/
```

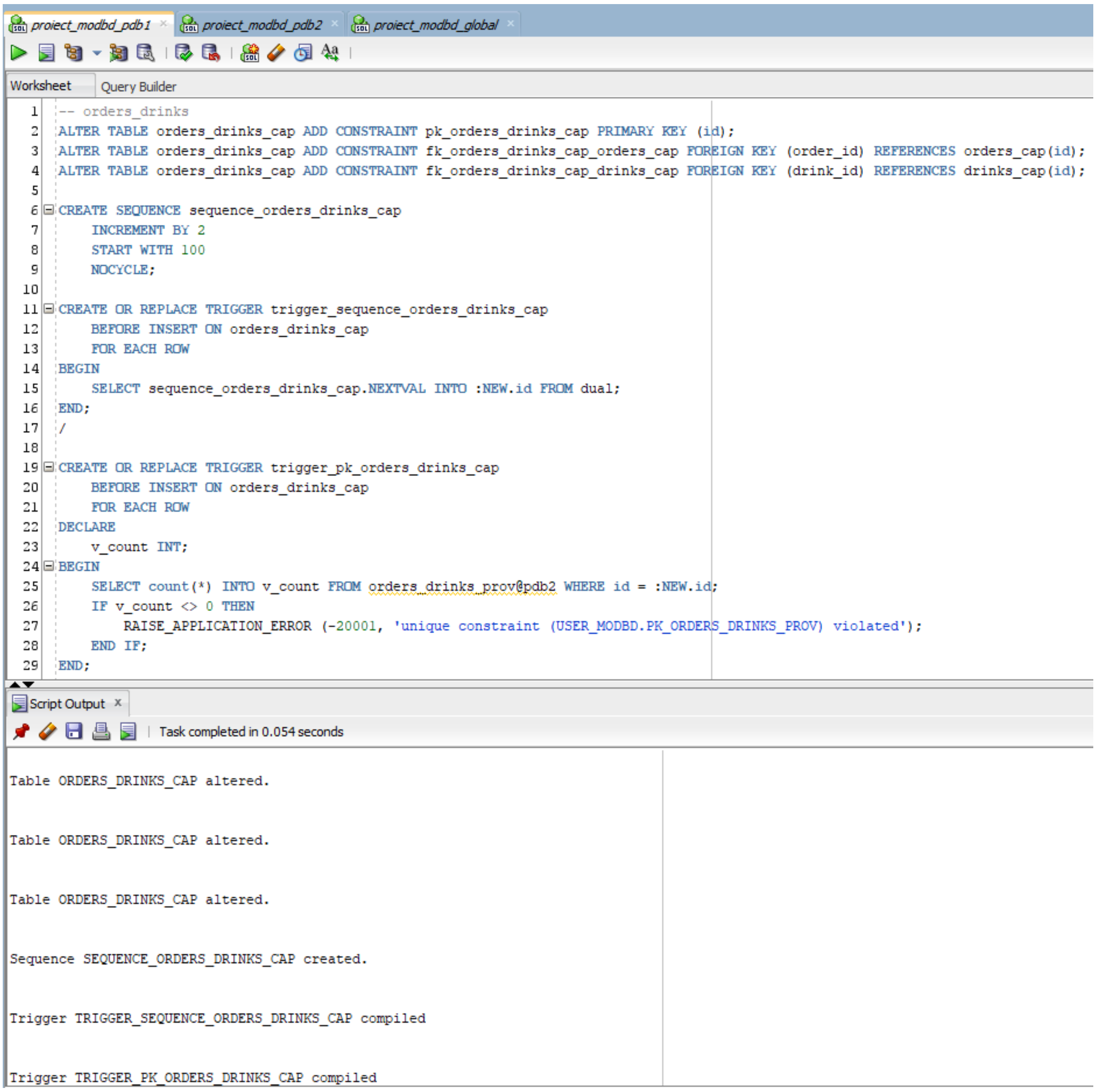


```
-- orders
ALTER TABLE orders_prov ADD CONSTRAINT pk_orders_prov PRIMARY KEY (id);
ALTER TABLE orders_prov ADD CONSTRAINT fk_orders_prov_employees_prov FOREIGN KEY (waiter_id) REFERENCES employees_prov(id);

CREATE SEQUENCE sequence_orders_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_prov
    BEFORE INSERT ON orders_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_orders_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_prov
    BEFORE INSERT ON orders_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM orders_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_ORDERS_CAP) violated');
    END IF;
END;
/
```

```
Table ORDERS_PROV altered.


Table ORDERS_PROV altered.


Sequence SEQUENCE_ORDERS_PROV created.


Trigger TRIGGER_SEQUENCE_ORDERS_PROV compiled


Trigger TRIGGER_PK_ORDERS_PROV compiled
```

```
-- orders_drinks
ALTER TABLE orders_drinks_cap ADD CONSTRAINT pk_orders_drinks_cap PRIMARY KEY (id);
ALTER TABLE orders_drinks_cap ADD CONSTRAINT fk_orders_drinks_cap_orders_cap FOREIGN
KEY (order_id) REFERENCES orders_cap(id);
ALTER TABLE orders_drinks_cap ADD CONSTRAINT fk_orders_drinks_cap_drinks_cap FOREIGN
KEY (drink_id) REFERENCES drinks_cap(id);

CREATE SEQUENCE sequence_orders_drinks_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;
```

```
CREATE OR REPLACE TRIGGER trigger_sequence_orders_drinks_cap
   BEFORE INSERT ON orders_drinks_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_orders_drinks_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_drinks_cap
   BEFORE INSERT ON orders_drinks_cap
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM orders_drinks_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_ORDERS_DRINKS_PROV) violated');
   END IF;
END;
/
```

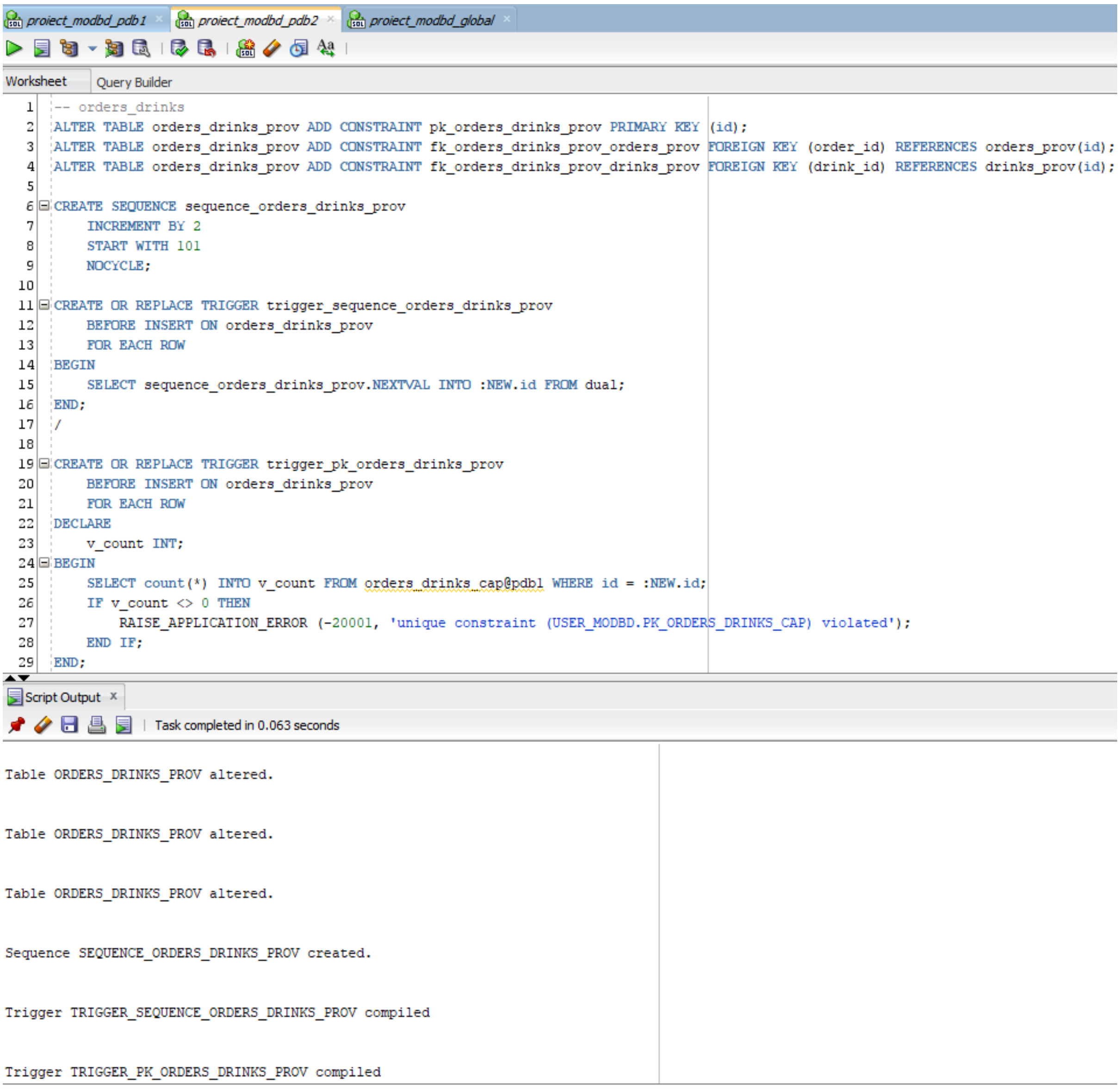


```
-- orders_drinks
ALTER TABLE orders_drinks_cap ADD CONSTRAINT pk_orders_drinks_cap PRIMARY KEY (id);
ALTER TABLE orders_drinks_cap ADD CONSTRAINT fk_orders_drinks_cap_orders_cap FOREIGN KEY (order_id) REFERENCES orders_cap(id);
ALTER TABLE orders_drinks_cap ADD CONSTRAINT fk_orders_drinks_cap_drinks_cap FOREIGN KEY (drink_id) REFERENCES drinks_cap(id);

CREATE SEQUENCE sequence_orders_drinks_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_drinks_cap
    BEFORE INSERT ON orders_drinks_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_orders_drinks_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_drinks_cap
    BEFORE INSERT ON orders_drinks_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM orders_drinks_prov@pdb2 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_ORDERS_DRINKS_PROV) violated');
    END IF;
END;
```

```
Table ORDERS_DRINKS_CAP altered.


Table ORDERS_DRINKS_CAP altered.


Table ORDERS_DRINKS_CAP altered.


Sequence SEQUENCE_ORDERS_DRINKS_CAP created.


Trigger TRIGGER_SEQUENCE_ORDERS_DRINKS_CAP compiled


Trigger TRIGGER_PK_ORDERS_DRINKS_CAP compiled
```

```
-- orders_drinks
ALTER TABLE orders_drinks_prov ADD CONSTRAINT pk_orders_drinks_prov PRIMARY KEY (id);
ALTER TABLE orders_drinks_prov ADD CONSTRAINT fk_orders_drinks_prov_orders_prov
FOREIGN KEY (order_id) REFERENCES orders_prov(id);
ALTER TABLE orders_drinks_prov ADD CONSTRAINT fk_orders_drinks_prov_drinks_prov
FOREIGN KEY (drink_id) REFERENCES drinks_prov(id);

CREATE SEQUENCE sequence_orders_drinks_prov
   INCREMENT BY 2
   START WITH 101
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_drinks_prov
   BEFORE INSERT ON orders_drinks_prov
   FOR EACH ROW
BEGIN
   SELECT sequence_orders_drinks_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_drinks_prov
```

```
    BEFORE INSERT ON orders_drinks_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM orders_drinks_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_ORDERS_DRINKS_CAP) violated');
    END IF;
END;
/
```



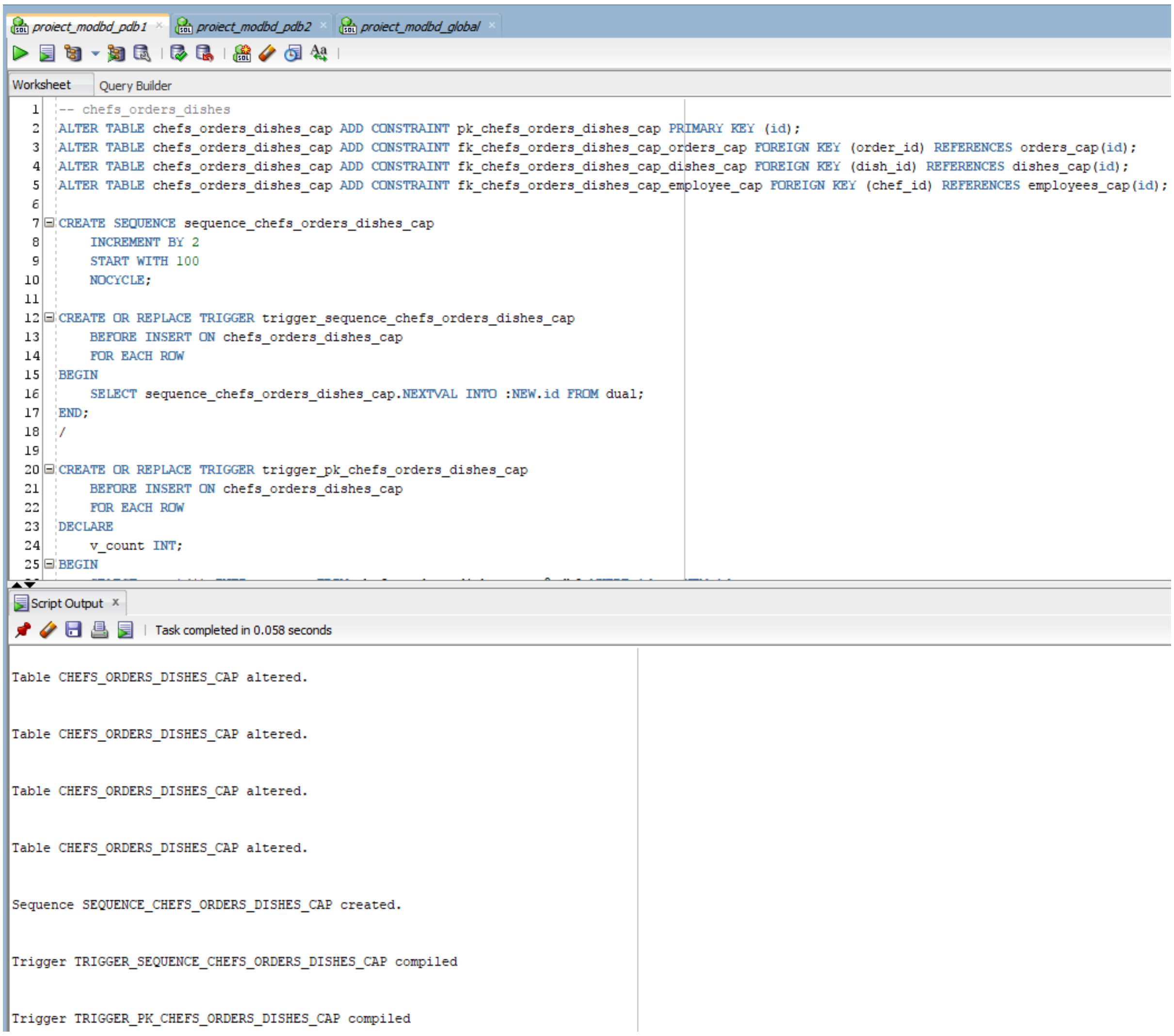
```
-- orders_drinks
ALTER TABLE orders_drinks_prov ADD CONSTRAINT pk_orders_drinks_prov PRIMARY KEY (id);
ALTER TABLE orders_drinks_prov ADD CONSTRAINT fk_orders_drinks_prov_orders_prov FOREIGN KEY (order_id) REFERENCES orders_prov(id);
ALTER TABLE orders_drinks_prov ADD CONSTRAINT fk_orders_drinks_prov_drinks_prov FOREIGN KEY (drink_id) REFERENCES drinks_prov(id);

CREATE SEQUENCE sequence_orders_drinks_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_orders_drinks_prov
    BEFORE INSERT ON orders_drinks_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_orders_drinks_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_orders_drinks_prov
    BEFORE INSERT ON orders_drinks_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
    SELECT count(*) INTO v_count FROM orders_drinks_cap@pdb1 WHERE id = :NEW.id;
    IF v_count <> 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'unique constraint (USER_MODBD.PK_ORDERS_DRINKS_CAP) violated');
    END IF;
END;
```

Task completed in 0.063 seconds

```
Table ORDERS_DRINKS_PROV altered.


Table ORDERS_DRINKS_PROV altered.


Table ORDERS_DRINKS_PROV altered.


Sequence SEQUENCE_ORDERS_DRINKS_PROV created.


Trigger TRIGGER_SEQUENCE_ORDERS_DRINKS_PROV compiled


Trigger TRIGGER_PK_ORDERS_DRINKS_PROV compiled
```

```
-- chefs_orders_dishes
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT pk_chefs_orders_dishes_cap
PRIMARY KEY (id);
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT
fk_chefs_orders_dishes_cap_orders_cap FOREIGN KEY (order_id) REFERENCES orders_cap(id);
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT
fk_chefs_orders_dishes_cap_dishes_cap FOREIGN KEY (dish_id) REFERENCES dishes_cap(id);
```

```
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT
fk_chefs_orders_dishes_cap_employee_cap FOREIGN KEY (chef_id) REFERENCES
employees_cap(id);

CREATE SEQUENCE sequence_chefs_orders_dishes_cap
   INCREMENT BY 2
   START WITH 100
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_chefs_orders_dishes_cap
   BEFORE INSERT ON chefs_orders_dishes_cap
   FOR EACH ROW
BEGIN
   SELECT sequence_chefs_orders_dishes_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_chefs_orders_dishes_cap
   BEFORE INSERT ON chefs_orders_dishes_cap
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM chefs_orders_dishes_prov@pdb2 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_CHEFS_ORDERS_DISHES_PROV) violated');
   END IF;
END;
/
```

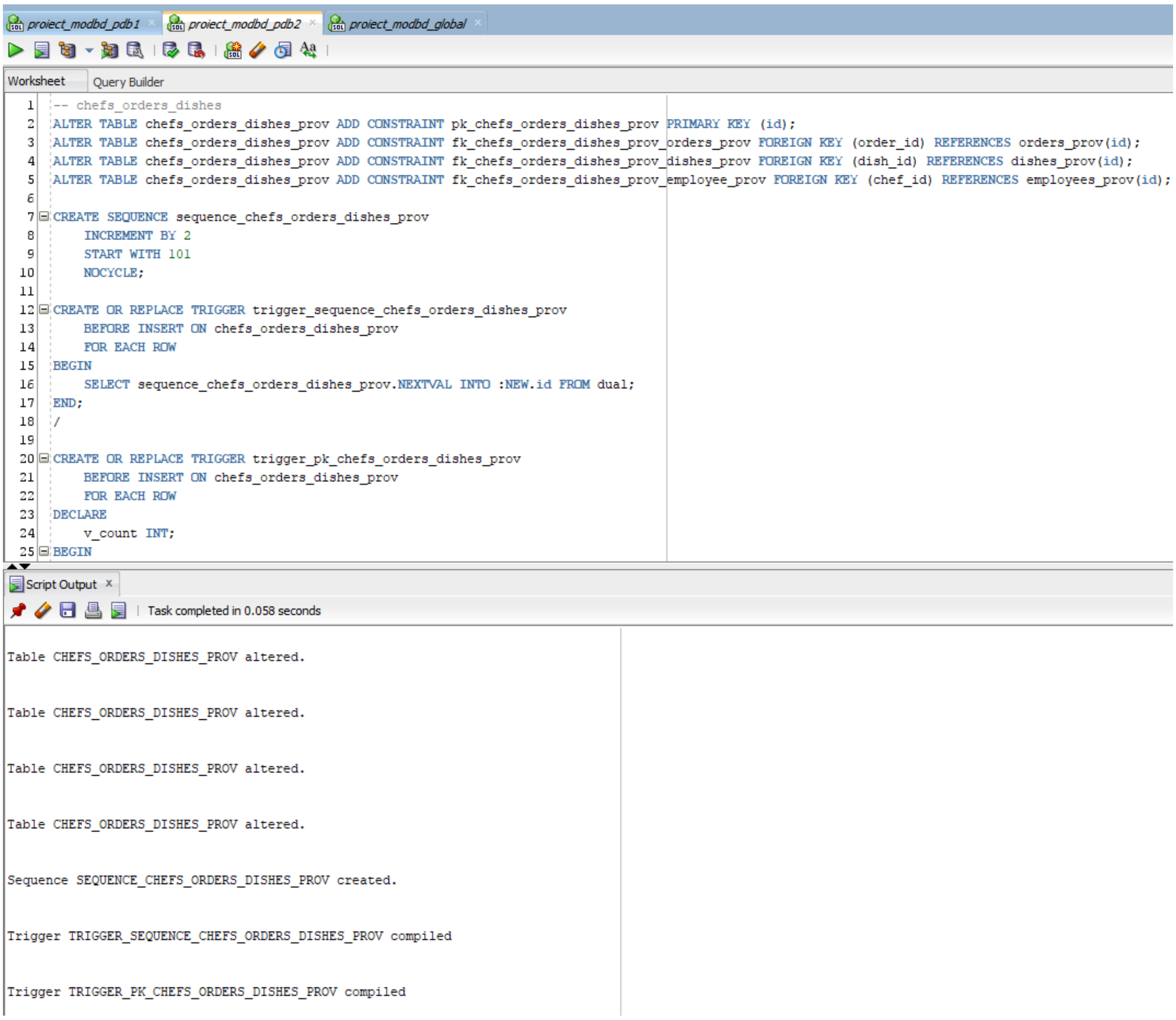


```
-- chefs_orders_dishes
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT pk_chefs_orders_dishes_cap PRIMARY KEY (id);
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT fk_chefs_orders_dishes_cap_orders_cap FOREIGN KEY (order_id) REFERENCES orders_cap(id);
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT fk_chefs_orders_dishes_cap_dishes_cap FOREIGN KEY (dish_id) REFERENCES dishes_cap(id);
ALTER TABLE chefs_orders_dishes_cap ADD CONSTRAINT fk_chefs_orders_dishes_cap_employee_cap FOREIGN KEY (chef_id) REFERENCES employees_cap(id);

CREATE SEQUENCE sequence_chefs_orders_dishes_cap
    INCREMENT BY 2
    START WITH 100
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_chefs_orders_dishes_cap
    BEFORE INSERT ON chefs_orders_dishes_cap
    FOR EACH ROW
BEGIN
    SELECT sequence_chefs_orders_dishes_cap.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_chefs_orders_dishes_cap
    BEFORE INSERT ON chefs_orders_dishes_cap
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
```

```
Table CHEFS_ORDERS_DISHES_CAP altered.


Table CHEFS_ORDERS_DISHES_CAP altered.


Table CHEFS_ORDERS_DISHES_CAP altered.


Table CHEFS_ORDERS_DISHES_CAP altered.


Sequence SEQUENCE_CHEFS_ORDERS_DISHES_CAP created.


Trigger TRIGGER_SEQUENCE_CHEFS_ORDERS_DISHES_CAP compiled


Trigger TRIGGER_PK_CHEFS_ORDERS_DISHES_CAP compiled
```

```
-- chefs_orders_dishes
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT pk_chefs_orders_dishes_prov
PRIMARY KEY (id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT
fk_chefs_orders_dishes_prov_orders_prov FOREIGN KEY (order_id) REFERENCES orders_prov(id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT
fk_chefs_orders_dishes_prov_dishes_prov FOREIGN KEY (dish_id) REFERENCES dishes_prov(id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT
fk_chefs_orders_dishes_prov_employee_prov FOREIGN KEY (chef_id) REFERENCES
employees_prov(id);

CREATE SEQUENCE sequence_chefs_orders_dishes_prov
   INCREMENT BY 2
   START WITH 101
   NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_chefs_orders_dishes_prov
   BEFORE INSERT ON chefs_orders_dishes_prov
   FOR EACH ROW
BEGIN
   SELECT sequence_chefs_orders_dishes_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_chefs_orders_dishes_prov
```

```
   BEFORE INSERT ON chefs_orders_dishes_prov
   FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   SELECT count(*) INTO v_count FROM chefs_orders_dishes_cap@pdb1 WHERE id = :NEW.id;
   IF v_count <> 0 THEN
      RAISE_APPLICATION_ERROR (-20001, 'unique constraint
(USER_MODBD.PK_CHEFS_ORDERS_DISHES_CAP) violated');
   END IF;
END;
/
```



```
-- chefs_orders_dishes
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT pk_chefs_orders_dishes_prov PRIMARY KEY (id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT fk_chefs_orders_dishes_prov_orders_prov FOREIGN KEY (order_id) REFERENCES orders_prov(id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT fk_chefs_orders_dishes_prov_dishes_prov FOREIGN KEY (dish_id) REFERENCES dishes_prov(id);
ALTER TABLE chefs_orders_dishes_prov ADD CONSTRAINT fk_chefs_orders_dishes_prov_employee_prov FOREIGN KEY (chef_id) REFERENCES employees_prov(id);

CREATE SEQUENCE sequence_chefs_orders_dishes_prov
    INCREMENT BY 2
    START WITH 101
    NOCYCLE;

CREATE OR REPLACE TRIGGER trigger_sequence_chefs_orders_dishes_prov
    BEFORE INSERT ON chefs_orders_dishes_prov
    FOR EACH ROW
BEGIN
    SELECT sequence_chefs_orders_dishes_prov.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER trigger_pk_chefs_orders_dishes_prov
    BEFORE INSERT ON chefs_orders_dishes_prov
    FOR EACH ROW
DECLARE
    v_count INT;
BEGIN
```

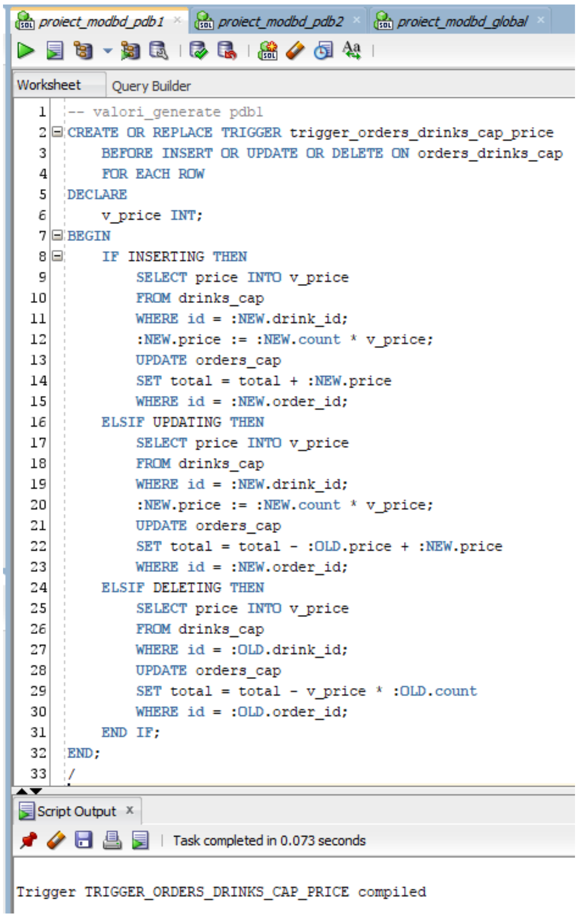
Task completed in 0.058 seconds

```
Table CHEFS_ORDERS_DISHES_PROV altered.


Table CHEFS_ORDERS_DISHES_PROV altered.


Table CHEFS_ORDERS_DISHES_PROV altered.


Table CHEFS_ORDERS_DISHES_PROV altered.


Sequence SEQUENCE_CHEFS_ORDERS_DISHES_PROV created.


Trigger TRIGGER_SEQUENCE_CHEFS_ORDERS_DISHES_PROV compiled


Trigger TRIGGER_PK_CHEFS_ORDERS_DISHES_PROV compiled
```

Am decis să adăugăm aici și generarea automata a valorilor pentru atributele price din orders_drinks și chef_orders_dishes, precum și a atributelor total și order_date din tabela orders. Trigger-ele corespunzătoare sunt create local, in pdb1 și pdb2

```
CREATE OR REPLACE TRIGGER trigger_orders_drinks_cap_price
   BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_cap
   FOR EACH ROW
DECLARE
   v_price INT;
BEGIN
   IF INSERTING THEN
```

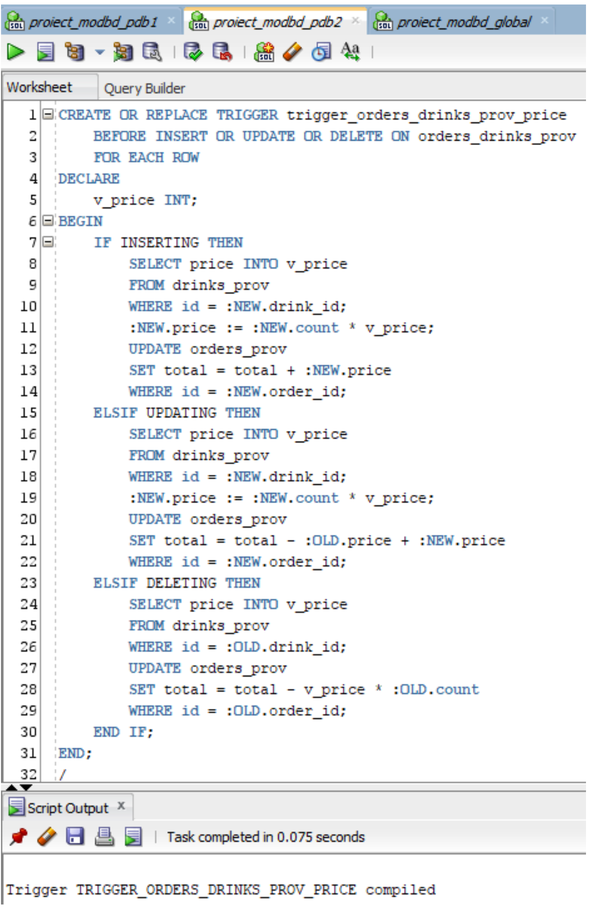
```
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :OLD.drink_id;
        UPDATE orders_cap
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trigger_orders_drinks_prov_price
    BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_prov
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :OLD.drink_id;
        UPDATE orders_prov
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```



```
-- valori generate pdb1
CREATE OR REPLACE TRIGGER trigger_orders_drinks_cap_price
    BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_cap
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM drinks_cap
        WHERE id = :OLD.drink_id;
        UPDATE orders_cap
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```

Task completed in 0.073 seconds

```
Trigger TRIGGER_ORDERS_DRINKS_CAP_PRICE compiled
```



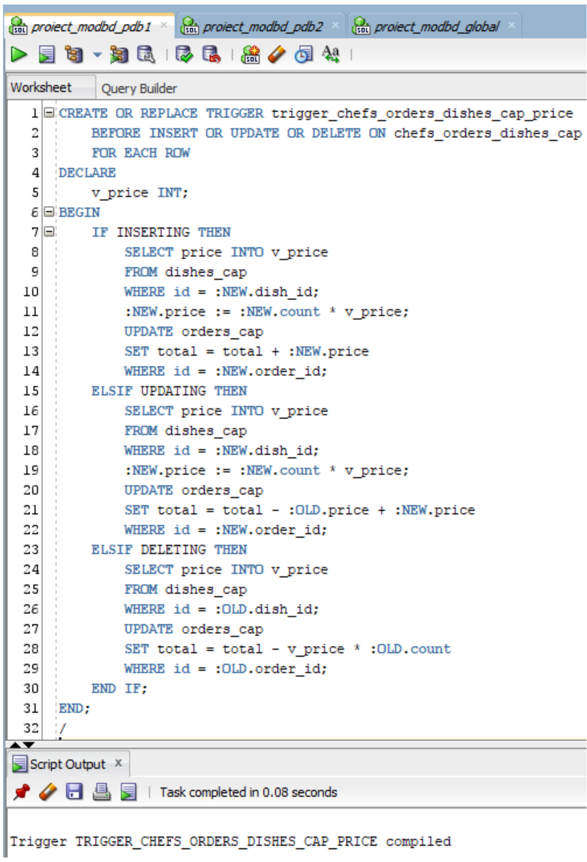
```
CREATE OR REPLACE TRIGGER trigger_orders_drinks_prov_price
    BEFORE INSERT OR UPDATE OR DELETE ON orders_drinks_prov
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :NEW.drink_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM drinks_prov
        WHERE id = :OLD.drink_id;
        UPDATE orders_prov
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```

Task completed in 0.075 seconds

```
Trigger TRIGGER_ORDERS_DRINKS_PROV_PRICE compiled
```

```
CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_cap_price
  BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_cap
  FOR EACH ROW
DECLARE
  v_price INT;
BEGIN
  IF INSERTING THEN
    SELECT price INTO v_price
    FROM dishes_cap
    WHERE id = :NEW.dish_id;
    :NEW.price := :NEW.count * v_price;
    UPDATE orders_cap
    SET total = total + :NEW.price
    WHERE id = :NEW.order_id;
  ELSIF UPDATING THEN
    SELECT price INTO v_price
    FROM dishes_cap
    WHERE id = :NEW.dish_id;
    :NEW.price := :NEW.count * v_price;
    UPDATE orders_cap
    SET total = total - :OLD.price + :NEW.price
    WHERE id = :NEW.order_id;
  ELSIF DELETING THEN
    SELECT price INTO v_price
    FROM dishes_cap
    WHERE id = :OLD.dish_id;
    UPDATE orders_cap
    SET total = total - v_price * :OLD.count
    WHERE id = :OLD.order_id;
  END IF;
```

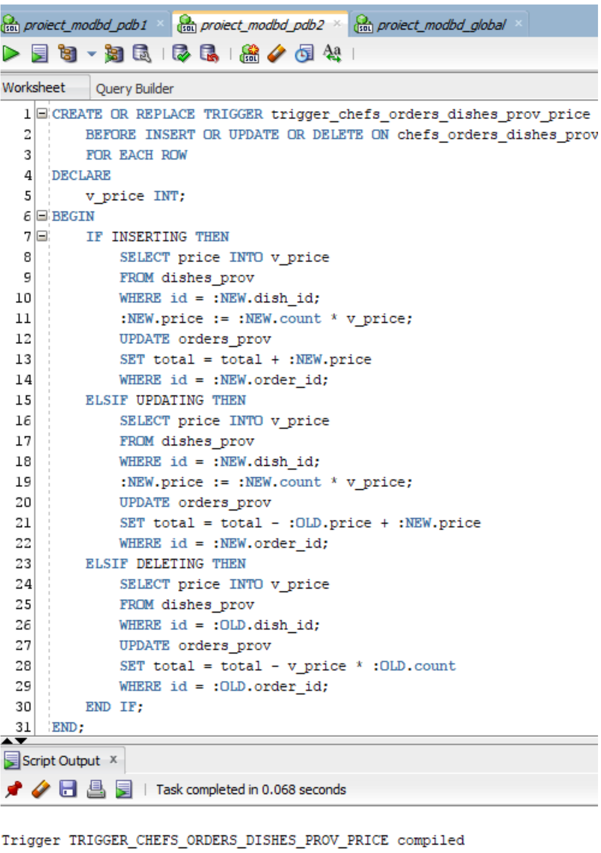
```
END;
/

CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_prov_price
   BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_prov
   FOR EACH ROW
DECLARE
   v_price INT;
BEGIN
   IF INSERTING THEN
      SELECT price INTO v_price
      FROM dishes_prov
      WHERE id = :NEW.dish_id;
      :NEW.price := :NEW.count * v_price;
      UPDATE orders_prov
      SET total = total + :NEW.price
      WHERE id = :NEW.order_id;
   ELSIF UPDATING THEN
      SELECT price INTO v_price
      FROM dishes_prov
      WHERE id = :NEW.dish_id;
      :NEW.price := :NEW.count * v_price;
      UPDATE orders_prov
      SET total = total - :OLD.price + :NEW.price
      WHERE id = :NEW.order_id;
   ELSIF DELETING THEN
      SELECT price INTO v_price
      FROM dishes_prov
      WHERE id = :OLD.dish_id;
      UPDATE orders_prov
      SET total = total - v_price * :OLD.count
      WHERE id = :OLD.order_id;
   END IF;
END;
/
```



```
CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_cap_price
    BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_cap
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM dishes_cap
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM dishes_cap
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_cap
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM dishes_cap
        WHERE id = :OLD.dish_id;
        UPDATE orders_cap
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
/
```

Task completed in 0.08 seconds

```
Trigger TRIGGER_CHEFS_ORDERS_DISHES_CAP_PRICE compiled
```

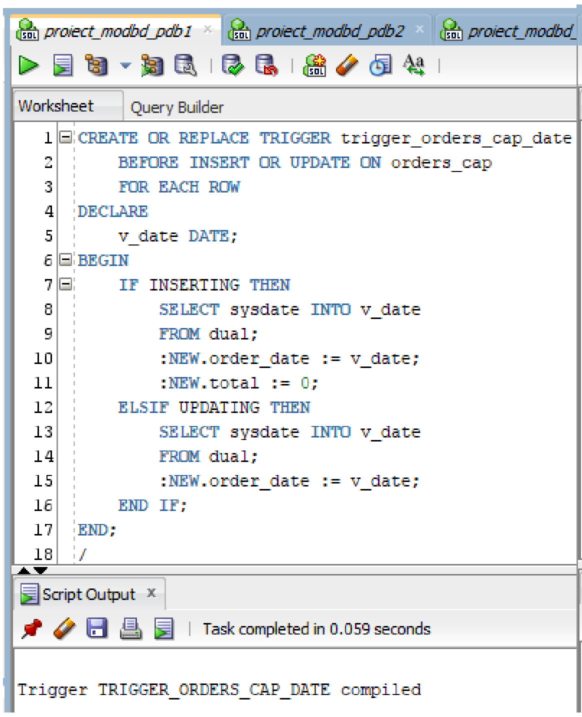


```
CREATE OR REPLACE TRIGGER trigger_chefs_orders_dishes_prov_price
    BEFORE INSERT OR UPDATE OR DELETE ON chefs_orders_dishes_prov
    FOR EACH ROW
DECLARE
    v_price INT;
BEGIN
    IF INSERTING THEN
        SELECT price INTO v_price
        FROM dishes_prov
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF UPDATING THEN
        SELECT price INTO v_price
        FROM dishes_prov
        WHERE id = :NEW.dish_id;
        :NEW.price := :NEW.count * v_price;
        UPDATE orders_prov
        SET total = total - :OLD.price + :NEW.price
        WHERE id = :NEW.order_id;
    ELSIF DELETING THEN
        SELECT price INTO v_price
        FROM dishes_prov
        WHERE id = :OLD.dish_id;
        UPDATE orders_prov
        SET total = total - v_price * :OLD.count
        WHERE id = :OLD.order_id;
    END IF;
END;
```

Task completed in 0.068 seconds

```
Trigger TRIGGER_CHEFS_ORDERS_DISHES_PROV_PRICE compiled
```

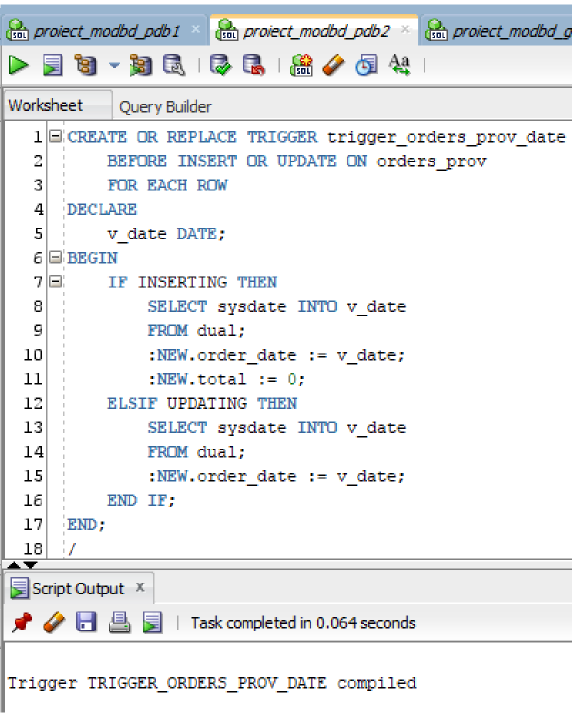
```
CREATE OR REPLACE TRIGGER trigger_orders_cap_date
    BEFORE INSERT OR UPDATE ON orders_cap
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trigger_orders_prov_date
    BEFORE INSERT OR UPDATE ON orders_prov
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/
```



```
CREATE OR REPLACE TRIGGER trigger_orders_cap_date
    BEFORE INSERT OR UPDATE ON orders_cap
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/
```

Task completed in 0.059 seconds

```
Trigger TRIGGER_ORDERS_CAP_DATE compiled
```



```
CREATE OR REPLACE TRIGGER trigger_orders_prov_date
    BEFORE INSERT OR UPDATE ON orders_prov
    FOR EACH ROW
DECLARE
    v_date DATE;
BEGIN
    IF INSERTING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
        :NEW.total := 0;
    ELSIF UPDATING THEN
        SELECT sysdate INTO v_date
        FROM dual;
        :NEW.order_date := v_date;
    END IF;
END;
/
```

Task completed in 0.064 seconds

```
Trigger TRIGGER_ORDERS_PROV_DATE compiled
```