Proiect Ferma- EABD

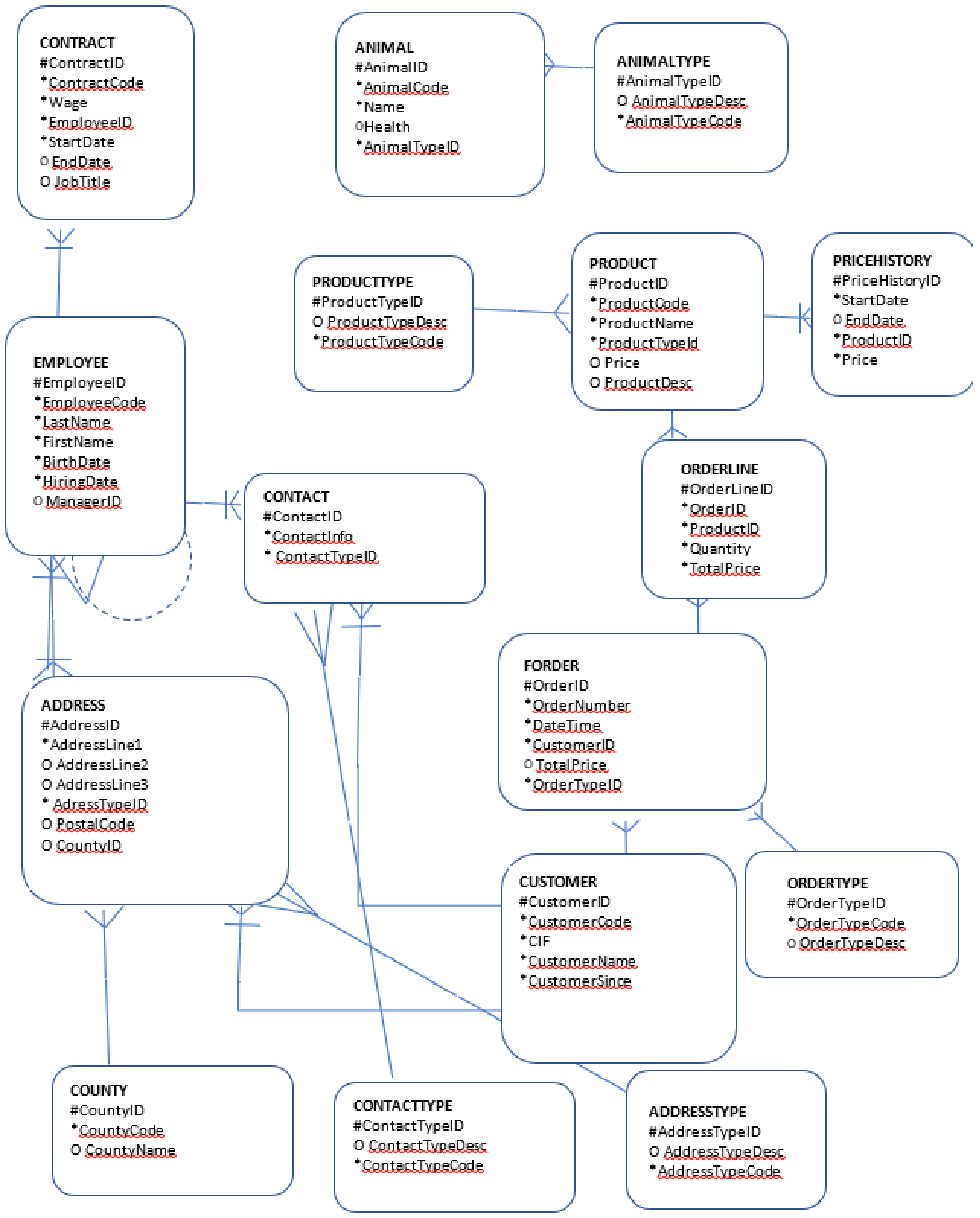
Student: Florea Darius George

Specializare: IE ID, an 3, gr 2

Proiectul ales de catre mine anul trecut a fost “Ferma”. Aplicabilitatea bazei de date sau a unei eventuale aplicatii create peste aceasta baza de date ar fi administrarea mult mai usoara a unei ferme(tot ce tine de stoc, comenzi, client, bonusuri, preturi, istoric preturi) ulterior acest proces putand fi spre automatizarea livrarii comenzilor. S-ar putea integra cu sisteme de navigare, oferind astfel soferilor ore de plecare/sosire si trasee pentru a facilita o livrare a comenzilor cat mai rapida si simpla.

Baza de date doreste sa stocheze cea mai valoroasa resursa pe care o avem in zilele noastre: informatia. Astfel vom dori sa stocam cat mai multe informatii despre clienti, produsele din stoc, comenzile trecute precum si preturile. O modificare adusa este adaugarea fieldului “ContractDocument” in tabela “Contract”. Un camp de tip CLOB, care stocheaza contractul dintre angajat si angajator.

O sa las si diagrama ERD predate anul anterior pentru a fi mai usor de urmarit.


CONTRACT
#ContractID
*ContractCode
*Wage
*EmployeeID
*StartDate
O EndDate
O JobTitle
ANIMAL
#AnimalID
*AnimalCode
*Name
OHealth
*AnimalTypeID
ANIMALTYPE
#AnimalTypeID
O AnimalTypeDesc
*AnimalTypeCode
PRODUCTTYPE
#ProductTypeID
O ProductTypeDesc
*ProductTypeCode
PRODUCT
#ProductID
*ProductCode
*ProductName
*ProductTypeId
O Price
O ProductDesc
PRICEHISTORY
#PriceHistoryID
*StartDate
O EndDate
*ProductID
*Price
EMPLOYEE
#EmployeeID
*EmployeeCode
*LastName
*FirstName
*BirthDate
*HiringDate
O ManagerID
CONTACT
#ContactID
*ContactInfo
* ContactTypeID
ORDERLINE
#OrderLineID
*OrderID
*ProductID
*Quantity
*TotalPrice
FORDER
#OrderID
*OrderNumber
*DateTime
*CustomerID
O TotalPrice
*OrderTypeID
ADDRESS
#AddressID
*AddressLine1
O AddressLine2
O AddressLine3
* AdressTypeID
O PostalCode
O CountyID
CUSTOMER
#CustomerID
*CustomerCode
*CIF
*CustomerName
*CustomerSince
ORDERTYPE
#OrderTypeID
*OrderTypeCode
O OrderTypeDesc
COUNTY
#CountyID
*CountyCode
O CountyName
CONTACTTYPE
#ContactTypeID
O ContactTypeDesc
*ContactTypeCode
ADDRESSTYPE
#AddressTypeID
O AddressTypeDesc
*AddressTypeCode

Crearea tabelelor in baza de date

Comenzile pentru crearea bazei de date sunt urmatoarele:

```
CREATE TABLE employee(EmployeeID NUMBER PRIMARY KEY,EmployeeCode VARCHAR(30) NOT NULL
UNIQUE,

FirstName VARCHAR(15) NOT NULL,

LastName VARCHAR(15) NOT NULL,

BirthDate DATE,

HiringDate DATE NOT NULL,

ManagerID NUMBER);

CREATE TABLE AnimalType(AnimalTypeID NUMBER PRIMARY KEY, AnimalTypeDesc
VARCHAR(30), AnimalTypeCode VARCHAR(15) NOT NULL UNIQUE);

CREATE TABLE ProductType(ProductTypeID NUMBER PRIMARY KEY, ProductTypeDesc
VARCHAR(30), ProductTypeCode VARCHAR(15) NOT NULL UNIQUE);

CREATE TABLE ContactType(ContactTypeID NUMBER PRIMARY KEY, ContactTypeDesc
VARCHAR(30), ContactTypeCode VARCHAR(15) NOT NULL UNIQUE);

CREATE TABLE AddressType(AddressTypeID NUMBER PRIMARY KEY, AddressTypeDesc
VARCHAR(30), AddressTypeCode VARCHAR(15) NOT NULL UNIQUE);

CREATE TABLE OrderType(OrderTypeID NUMBER PRIMARY KEY, OrderTypeDesc VARCHAR(30),
OrderTypeCode VARCHAR(15) NOT NULL UNIQUE);

CREATE TABLE County (CountyID NUMBER PRIMARY KEY, CountyCode VARCHAR(5) NOT NULL
UNIQUE, CountyName VARCHAR(15));

CREATE TABLE Animal(AnimalID NUMBER PRIMARY KEY, AnimalCode VARCHAR(15) NOT NULL
UNIQUE, Name VARCHAR(15) NOT NULL, Health VARCHAR(35), AnimalTypeID NUMBER
REFERENCES AnimalType(AnimalTypeID));

CREATE TABLE Contact(ContactID NUMBER PRIMARY KEY, ContactInfo VARCHAR(30) NOT NULL
UNIQUE, ContactTypeID NUMBER REFERENCES ContactType(ContactTypeID));

CREATE TABLE Address(AddressID NUMBER PRIMARY KEY, AddressLine1 VARCHAR(35) NOT
NULL UNIQUE, AddressLine2 VARCHAR(35), AddressLine3 VARCHAR(35), AddressTypeID
NUMBER REFERENCES AddressType(AddressTypeID), PostalCode CHAR(6), CountyID NUMBER
REFERENCES County(CountyID));
```

```sql
CREATE TABLE Contract(ContractID NUMBER PRIMARY KEY, ContractCode VARCHAR(25) NOT
NULL UNIQUE, Wage NUMBER NOT NULL, EmployeeID NUMBER REFERENCES
Employee(EmployeeID), StartDate DATE NOT NULL, EndDate DATE, JobTitle VARCHAR(15) NOT
NULL);

CREATE TABLE Product(ProductID NUMBER PRIMARY KEY, ProductCode VARCHAR(15) NOT
NULL UNIQUE, ProductName VARCHAR(30) NOT NULL, ProductTypeID REFERENCES
ProductType(ProductTypeID), Price NUMBER, ProductDesc VARCHAR(30));

CREATE TABLE PriceHistory(PriceHistoryID NUMBER PRIMARY KEY, StartDate DATE NOT NULL,
EndDate DATE, Price NUMBER, ProductID NUMBER REFERENCES Product(ProductID));

CREATE TABLE Customer(CustomerID NUMBER PRIMARY KEY, CustomerCode VARCHAR(15)
NOT NULL UNIQUE, Cif VARCHAR(20) NOT NULL, CustomerName VARCHAR(30), CustomerSince
DATE NOT NULL, Notes VARCHAR(50));

CREATE TABLE FOrder(OrderID NUMBER PRIMARY KEY, OrderNumber VARCHAR(30) NOT NULL
UNIQUE, DateTime TIMESTAMP NOT NULL, CustomerID REFERENCES Customer(CustomerID),
TotalPrice NUMBER, OrderTypeID NUMBER REFERENCES OrderType(OrderTypeID));

CREATE TABLE OrderLine(OrderLineID NUMBER PRIMARY KEY, OrderID NUMBER REFERENCES
FOrder(OrderID), ProductID NUMBER REFERENCES Product(ProductID), Quantity NUMBER NOT
NULL, TotalPrice NUMBER NOT NULL, CreatedDT TIMESTAMP NOT NULL);

CREATE TABLE CustomerAddress(CustomerAddressID NUMBER PRIMARY KEY, AddressID
NUMBER REFERENCES Address(AddressID), CustomerID NUMBER REFERENCES
Customer(CustomerID), AddressTypeID NUMBER REFERENCES AddressType(AddressTypeID));

CREATE TABLE EmployeeAddress(EmployeeAddressID NUMBER PRIMARY KEY, AddressID
NUMBER REFERENCES Address(AddressID), EmployeeID NUMBER REFERENCES
Employee(EmployeeID), AddressTypeID NUMBER REFERENCES AddressType(AddressTypeID));

CREATE TABLE CustomerContact(CustomerContactID NUMBER PRIMARY KEY, ContactID
NUMBER REFERENCES Contact(ContactID), CustomerID NUMBER REFERENCES
Customer(CustomerID), ContactTypeID NUMBER REFERENCES ContactType(ContactTypeID));

CREATE TABLE EmployeeContact(EmployeeContactID NUMBER PRIMARY KEY, ContactID
NUMBER REFERENCES Contact(ContactID), EmployeeID NUMBER REFERENCES
Employee(EmployeeID), ContactTypeID NUMBER REFERENCES ContactType(ContactTypeID));
```

Populare tabele

Vom popula tabelele folosindu-ne de urmatoarele comenzi:

(id, cod, nume)

```
INSERT INTO County VALUES(1, 'AB', 'Alba');
INSERT INTO County VALUES(2, 'CJ', 'Cluj');
INSERT INTO County VALUES(3, 'BN', 'Bistrita-Nasaud');
INSERT INTO County VALUES(4, 'HD', 'Hunedoara');
INSERT INTO County VALUES(5, 'BV', 'Brasov');
INSERT INTO County VALUES(6, 'MM', 'Maramures');
```

(id, desc, cod)

```
INSERT INTO AnimalType VALUES (1, 'Porc calitate 1', 'PORC');
INSERT INTO AnimalType VALUES (2, 'Pui calitate 1', 'PUI');
INSERT INTO AnimalType VALUES (3, '', 'RATA');
INSERT INTO AnimalType VALUES (4, '', 'IEPURE');
INSERT INTO AnimalType VALUES (5, 'Curcan calitate 2', 'CURCAN');
INSERT INTO AnimalType VALUES (6, 'Peste', 'PESTE');
```

(id, desc, cod)

```
INSERT INTO ContactType VALUES(1, '', 'TELEFON');
INSERT INTO ContactType VALUES(2, '', 'FACEBOOK');
INSERT INTO ContactType VALUES(3, '', 'LINKEDIN');
INSERT INTO ContactType VALUES(4, '', 'EMAIL');
INSERT INTO ContactType VALUES(5, '', 'MOBIL');
INSERT INTO ContactType VALUES(6, '', 'TWITTER');
```

```
(id, desc, cod)

INSERT INTO AddressType VALUES(1, 'Main mailing address', 'MAILING1');

INSERT INTO AddressType VALUES(2, 'Secondary mailing address', 'MAILING2');

INSERT INTO AddressType VALUES(3, 'Third mailing address', 'MAILING3');

INSERT INTO AddressType VALUES(4, 'Bills only address', 'BILL');

INSERT INTO AddressType VALUES(5, 'Main delivery address', 'DELIVERY1');

INSERT INTO AddressType VALUES(6, 'Secondary delivery address', 'DELIVERY2');

(ProductTypeID, ProductTypeDesc,ProductTypeCode)

INSERT INTO ProductType VALUES(1, '', 'ANIMAL');

INSERT INTO ProductType VALUES(2, '', 'LACTAT');

INSERT INTO ProductType VALUES(3, '', 'BLANA');

INSERT INTO ProductType VALUES(4, '', 'REST');

INSERT INTO ProductType VALUES(5, '', 'TEST_PRODUCT');

INSERT INTO ProductType VALUES(6, '', 'TEST_PRODUCT2');

(id, desc, cod)

INSERT INTO OrderType VALUES(1, '', 'ORDER');

INSERT INTO OrderType VALUES(2, '', 'CERERE');

INSERT INTO OrderType VALUES(3, '', 'DEBARASARE');

INSERT INTO OrderType VALUES(4, '', 'PROMOTIE_ORDER');

INSERT INTO OrderType VALUES(5, '', 'TEST_ORDER');

INSERT INTO OrderType VALUES(6, '', 'TEST_ORDER2');

(id, cod, prenume, nume, data nasterii, data angajarii, id manager)

INSERT INTO Employee VALUES(1, 'MRC123', 'Marcel', 'Aluminiu', '12/Sep/1967', '01/Jan/2021',
0);

INSERT INTO Employee VALUES(2, 'ABL222', 'Abel', 'Bunul', '15/Jan/1991', '21/Jan/2021', 1);

INSERT INTO Employee VALUES(3, 'ANA111', 'Ana', 'Bunea', '15/Jan/1991', '21/Jan/2021', 1);

INSERT INTO Employee VALUES(4, 'MAR111', 'Maria', 'Bunea', '15/Jan/1991', '21/Jan/2021', 1);
INSERT INTO Employee VALUES(5, 'MIH333', 'Mihai', 'Nedumeritul', '15/OCt/1989', '21/Jan/2021', 1);

INSERT INTO Employee VALUES(6, 'PET333', 'Petre', 'Namidee', '22/Dec/1978', '21/Feb/2021', 1);
```

(id, cod, nume, stare sanatate, cod tip animal)

```
INSERT INTO Animal VALUES(1, 'PORC1', 'Porc aleator', 'Fara probleme', 1);

INSERT INTO Animal VALUES(2, 'PORC2', 'Porc aleator', '', 1);

INSERT INTO Animal VALUES(3, 'IEP1', 'Iepure aleator', '', 4);

INSERT INTO Animal VALUES(4, 'IEP2', 'Iepure aleator', '', 4);

INSERT INTO Animal VALUES(5, 'CURC1', 'Curcan generic', '', 5);

INSERT INTO Animal VALUES(6, 'CURC2', 'Curcan generic', '', 5);
```

(id, info, contact type id)

```
INSERT INTO Contact VALUES(1, '+40755123456', 5);

INSERT INTO Contact VALUES(2, '+40755133456', 5);

INSERT INTO Contact VALUES(3, 'www.facebook.com/marcel123', 2);

INSERT INTO Contact VALUES(4, 'www.facebook.com/anabanana', 2);

INSERT INTO Contact VALUES(5, 'maria.mar@gmail.com', 4);

INSERT INTO Contact VALUES(6, 'abel.bunul@gmail.com', 4);
```

(id, addressline1, addressline2, addressline3, addresstype id, cod postal, id judet)

```
INSERT INTO Address VALUES (1, 'Str pietrelor nr 1', 'bl 4 sc 1 et 4 ap 29', '', 1, '422220', 2);

INSERT INTO Address VALUES (2, 'Str pietrelor nr 2', 'bl 5 sc 1 et 4 ap 29', '', 1, '411567', 1);

INSERT INTO Address VALUES (3, 'Str pietrelor nr 3', 'bl 5 sc 1 et 4 ap 29', '', 1, '411567', 6);

INSERT INTO Address VALUES (4, 'Str pietrelor nr 4', 'bl 1 sc 1 et 1 ap 1', '', 1, '411567', 5);

INSERT INTO Address VALUES (5, 'Str pietrelor nr 5', '', 'test add3', 1, '344999', 3);

INSERT INTO Address VALUES (6, 'Str pietrelor nr 6', '', 'test add3', 1, '344999', 3);
```

(id, cod, salariu, id employee, start date, end date, titlu pozitie)

```
INSERT INTO Contract VALUES (1, '9000000', 5000, 1, '01/Jan/2021', '', 'CEO');

INSERT INTO Contract VALUES (2, '9000001', 2000, 2, '21/Jan/2021', '', 'MUNCITOR');
INSERT INTO Contract VALUES (3, '9000002', 2000, 3, '21/Jan/2021', '', 'MUNCITOR');

INSERT INTO Contract VALUES (4, '9000003', 2000, 4, '21/Jan/2021', '', 'MUNCITOR');

INSERT INTO Contract VALUES (5, '9000004', 2000, 5, '21/Jan/2021', '', 'MUNCITOR');

INSERT INTO Contract VALUES (6, '9000005', 1750, 5, '21/FEB/2021', '21/MAY/2021',
'MUNCITOR_PROBA');

(id, cod, nume, producttypeid, pret, desc)

INSERT INTO Product VALUES (1, 'X0000001','Porc', 1, 200, '');

INSERT INTO Product VALUES (2, 'L0000001','Iaurt 1.5', 2, 8, '');

INSERT INTO Product VALUES (3, 'B0000001','Blana urs maro', 3, 999, '');

INSERT INTO Product VALUES (4, 'T10000001','Test', 3, 0, '');

INSERT INTO Product VALUES (5, 'T10000002','Test', 3, 0, '');

INSERT INTO Product VALUES (6, 'T10000003','Test', 5, 11111, '');

(id, startdate, enddate, pret, productid)

INSERT INTO PriceHistory VALUES (1, '05/Jan/2021','27/Feb/2021', 598, 1);

INSERT INTO PriceHistory VALUES (2, '27/Feb/2021','27/Jun/2021', 667, 1);

INSERT INTO PriceHistory VALUES (3, '27/Jun/2021','', 200, 1);

INSERT INTO PriceHistory VALUES (4, '27/Jun/2021','', 8, 2);

INSERT INTO PriceHistory VALUES (5, '27/Jun/2021','', 999, 3);

INSERT INTO PriceHistory VALUES (6, '27/Jan/2021','', 1111, 6);

(id, cod, cif, nume, customersince, notes)

INSERT INTO customer VALUES (1, 'C00001', '245674347', 'Tester SRL', '27/Feb/2021','');

INSERT INTO customer VALUES (2, 'C00002', '245674323247', 'Tester SRL 2', '22/Feb/2021','');

INSERT INTO customer VALUES (3, 'C00003', '243247', 'Tester SRL 3', '22/Mar/2021','');

INSERT INTO customer VALUES (4, 'C00004', '24111113247', 'Tester SRL 4', '22/Apr/2021','');

INSERT INTO customer VALUES (5, 'C00005', '2453247', 'Tester SRL 5', '21/Jan/2021','');

INSERT INTO customer VALUES (6, 'C00006', '2453247', 'Old SRL', '01/Jan/2021','oldest');

(id, cod, datetime, customerid, price, ordertypeid)
```

```
INSERT INTO FOrder VALUES (1, 'O000001', '29/Jun/2021', 1, 0, 1);
INSERT INTO FOrder VALUES (2, 'O000002', '29/Jun/2021', 2, 0, 2);
INSERT INTO FOrder VALUES (3, 'O000003', '29/May/2021', 3, 0, 3);
INSERT INTO FOrder VALUES (4, 'O000004', '29/May/2021', 4, 0, 4);
INSERT INTO FOrder VALUES (5, 'O000005', '29/May/2021', 5, 0, 5);
INSERT INTO FOrder VALUES (6, 'O000006', '29/May/2021', 1, 0, 1);
(orderlineid, orderid, productid, quantity, totalprice, createddt)
INSERT INTO OrderLine VALUES (1, 1, 1, 15, 0, '29/Jun/2021');
INSERT INTO OrderLine VALUES (2, 2, 2, 15, 0, '29/Jun/2021');
INSERT INTO OrderLine VALUES (3, 3, 3, 15, 0, '29/Jun/2021');
INSERT INTO OrderLine VALUES (4, 4, 4, 15, 0, '29/Jun/2021');
INSERT INTO OrderLine VALUES (5, 1, 5, 15, 0, '29/Jun/2021');
INSERT INTO OrderLine VALUES (6, 2, 2, 15, 0, '29/Jun/2021');
(customeraddressid, customerid, addressid, addresstypeid)
INSERT INTO CustomerAddress VALUES (1,1,1,1);
INSERT INTO CustomerAddress VALUES (2,2,2,1);
INSERT INTO CustomerAddress VALUES (3,3,3,1);
INSERT INTO CustomerAddress VALUES (4,4,4,1);
INSERT INTO CustomerAddress VALUES (5,5,5,1);
INSERT INTO CustomerAddress VALUES (6,6,6,6);
(employeeaddressid, employeeid, addressid, addresstypeid):
INSERT INTO EmployeeAddress VALUES (6,6,6,6);
INSERT INTO EmployeeAddress VALUES (1,1,1,1);
INSERT INTO EmployeeAddress VALUES (2,2,2,1);
INSERT INTO EmployeeAddress VALUES (3,3,3,1);
INSERT INTO EmployeeAddress VALUES (4,4,4,1);
INSERT INTO EmployeeAddress VALUES (5,5,5,1);
```

```
(customercontactid, customerid, contactid, contacttypeid):
INSERT INTO CustomerContact VALUES (5,5,5,1);
INSERT INTO CustomerContact VALUES (6,6,6,1);
INSERT INTO CustomerContact VALUES (1,1,1,1);
INSERT INTO CustomerContact VALUES (2,2,2,1);
INSERT INTO CustomerContact VALUES (3,3,3,4);
INSERT INTO CustomerContact VALUES (4,4,4,1);
(employeecontactid, employeeid, contactid, contactypeid)
INSERT INTO EmployeeContact VALUES (1,1,3,1);
INSERT INTO EmployeeContact VALUES (2,3,4,1);
INSERT INTO EmployeeContact VALUES (3,4,5,1);
INSERT INTO EmployeeContact VALUES (4,2,6,1);
INSERT INTO EmployeeContact VALUES (5,5,1,1);
INSERT INTO EmployeeContact VALUES (6,6,2,1);
```

Toate comenzile de mai sus au fost rulate pe contul RO_A722_PLSQL_S35(unele manual rand pe rand, ultimele au fost introduse intr-un script numit : “populare” pentru a eficientiza procesul).

Este de asemenea necesar un update al tabelelor de mapare:

```
UPDATE CustomerContact
  SET (ContactTypeID) = (SELECT Contact.ContactTypeID FROM Contact WHERE
Contact.ContactID = CustomerContact.ContactID)
```

Similar vom face si in cazul EmployeeContact:

```
UPDATE EmployeeContact
  SET (ContactTypeID) = (SELECT Contact.ContactTypeID FROM Contact WHERE
Contact.ContactID = EmployeeContact.ContactID)
```

EmployeeAddress:

UPDATE EmployeeAddress

  SET (AddressTypeID) = (SELECT Address.AddressTypeID FROM Address WHERE Address.AddressID = EmployeeAddress.AddressID)

CustomerAddress:

UPDATE CustomerAddress

  SET (AddressTypeID) = (SELECT Address.AddressTypeID FROM Address WHERE Address.AddressID = CustomerAddress.AddressID)

Vom face un update pe TotalPrice pentru tabelele FOrder si OrderLine:

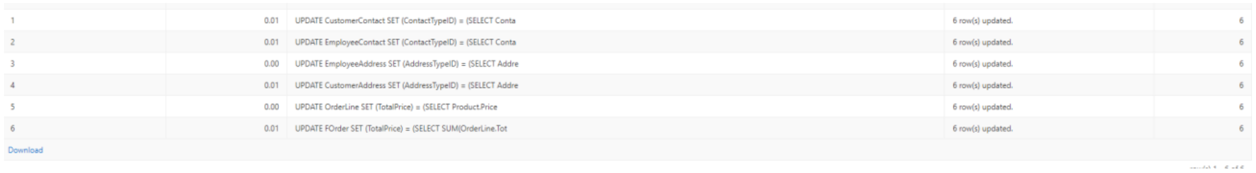
UPDATE OrderLine

  SET (TotalPrice) = (SELECT Product.Price FROM Product WHERE Product.ProductID = OrderLine.ProductID) * OrderLine.Quantity

Aceasta instructiune va face un calcul inmultind pretul produsului de pe un OrderLine cu cantitatea descrisa in OrderLine.Quantity.

UPDATE FOrder

  SET (TotalPrice) = (SELECT SUM(OrderLine.TotalPrice) FROM OrderLine WHERE OrderLine.OrderID = FOrder.OrderID)



1	0.01	UPDATE CustomerContact SET (ContactTypeID) = (SELECT Conta	6 row(s) updated.	6
2	0.01	UPDATE EmployeeContact SET (ContactTypeID) = (SELECT Conta	6 row(s) updated.	6
3	0.00	UPDATE EmployeeAddress SET (AddressTypeID) = (SELECT Addre	6 row(s) updated.	6
4	0.01	UPDATE CustomerAddress SET (AddressTypeID) = (SELECT Addre	6 row(s) updated.	6
5	0.00	UPDATE OrderLine SET (TotalPrice) = (SELECT Product.Price	6 row(s) updated.	6
6	0.01	UPDATE FOrder SET (TotalPrice) = (SELECT SUM(OrderLine.Tot	6 row(s) updated.	6

Download

Comenzile de mai sus au fost rulate in contul RO_A722_PLSQL_S35 prin scriptul “update_continut”.

De asemenea vom adauga un field de tip CLOB in tabela Contract care sa stocheze contractual propriu zis realizat intre angajat si angajator. Acest field se va numi “ContractDocument”:

alter table "CONTRACT" add ("CONTRACTDOCUMENT" CLOB);

Triggere si secvente

Vom incepe prin creerea unor secvente pentru id-urile tabelelor Employee,Customer,Contract,Forder, Address,Contact,Product

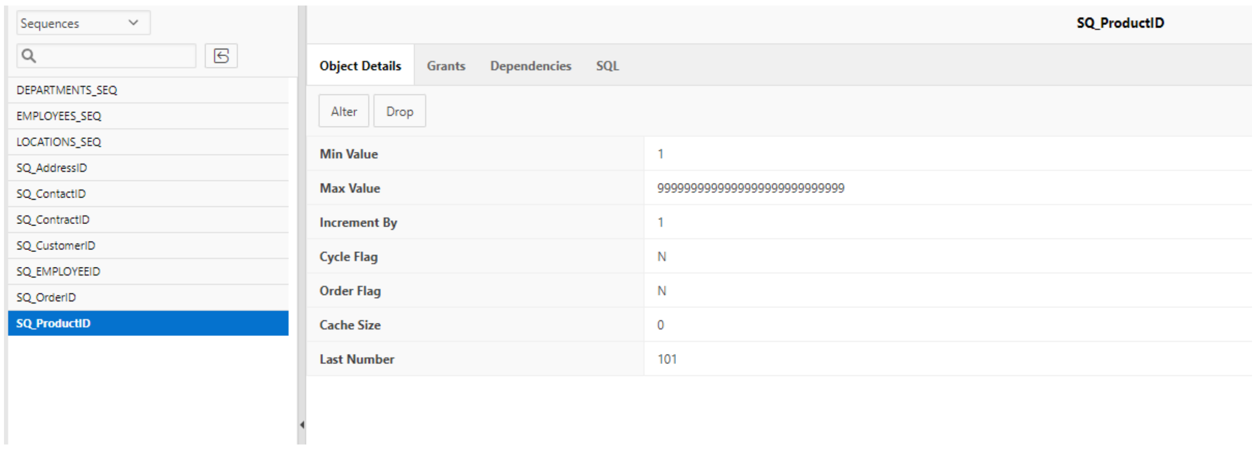
```
create sequence "SQ_EMPLOYEEID"
start with 1
increment by 1
nocache
nocycle
noorder;
create sequence "SQ_CustomerID"
start with 1
increment by 1
nocache
nocycle
noorder;
create sequence "SQ_ContractID"
start with 1
increment by 1
nocache
nocycle
noorder;
create sequence "SQ_OrderID"
start with 1
increment by 1
nocache
nocycle
```

```
noorder;
create sequence "SQ_ContactID"
start with 1
increment by 1
nocache
nocycle
noorder;
create sequence "SQ_AddressID"
start with 1
increment by 1
nocache
nocycle
noorder;
create sequence "SQ_ProductID"
start with 1
increment by 1
nocache
nocycle
noorder;
```

Ne vom folosi de triggere pentru a atribui in mod automat un nou ID atunci cand cream un nou Employee,Customer,Contract,Forder, Address,Contact sau Product. Astfel ca secventele vor “urmari” reprezenta valoarea urmatorului ID. In cazul de fata avem deja introduse date in baza de date astfel ca va trebui sa crestem valoarea secventei, pentru simplitate ii vom da 101.

Pentru a realiza acest lucru vom altera secventa crescandu-I proprietatea de increment cu 100, apoi ii vom creste valoarea, dupa care revenim la proprietatea increment “1”.

```
ALTER SEQUENCE "SQ_ProductID" INCREMENT BY 100;
select "SQ_ProductID".nextval from dual;
ALTER SEQUENCE "SQ_ProductID"  INCREMENT BY 1;
```



SQ_ProductID

Min Value	1
Max Value	9999999999999999999999999999
Increment By	1
Cycle Flag	N
Order Flag	N
Cache Size	0
Last Number	101

Similar vom face si pentru celelalte secvente:

ALTER SEQUENCE "SQ_AddressID" INCREMENT BY 100;

select "SQ_AddressID".nextval from dual;

ALTER SEQUENCE "SQ_AddressID" INCREMENT BY 1;

ALTER SEQUENCE "SQ_ContactID" INCREMENT BY 100;

select "SQ_ContactID".nextval from dual;

ALTER SEQUENCE "SQ_ContactID" INCREMENT BY 1;

ALTER SEQUENCE "SQ_OrderID" INCREMENT BY 100;

select "SQ_OrderID".nextval from dual;

ALTER SEQUENCE "SQ_OrderID" INCREMENT BY 1;

ALTER SEQUENCE "SQ_ContractID" INCREMENT BY 100;

select "SQ_ContractID".nextval from dual;

ALTER SEQUENCE "SQ_ContractID" INCREMENT BY 1;

ALTER SEQUENCE "SQ_CustomerID" INCREMENT BY 100;

select "SQ_CustomerID".nextval from dual;

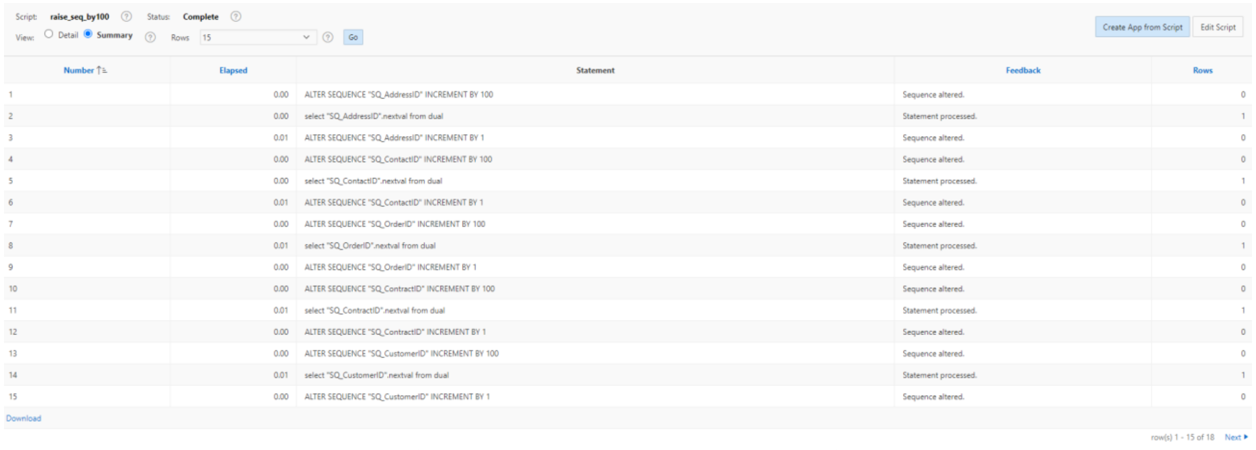
ALTER SEQUENCE "SQ_CustomerID" INCREMENT BY 1;

ALTER SEQUENCE "SQ_EMPLOYEEID" INCREMENT BY 100;

select "SQ_EMPLOYEEID".nextval from dual;

ALTER SEQUENCE "SQ_EMPLOYEEID" INCREMENT BY 1;

Am rulat aceste comenzi prin scriptul raise_seq_by100:



Script: **raise_seq_by100** Status: **Complete**

View: Detail [✓ Summary] Rows 15

Number	Elapsed	Statement	Feedback	Rows
1	0.00	ALTER SEQUENCE "SQ_AddressID" INCREMENT BY 100	Sequence altered.	0
2	0.00	select "SQ_AddressID".nextval from dual	Statement processed.	1
3	0.01	ALTER SEQUENCE "SQ_AddressID" INCREMENT BY 1	Sequence altered.	0
4	0.00	ALTER SEQUENCE "SQ_ContactID" INCREMENT BY 100	Sequence altered.	0
5	0.00	select "SQ_ContactID".nextval from dual	Statement processed.	1
6	0.01	ALTER SEQUENCE "SQ_ContactID" INCREMENT BY 1	Sequence altered.	0
7	0.00	ALTER SEQUENCE "SQ_OrderID" INCREMENT BY 100	Sequence altered.	0
8	0.01	select "SQ_OrderID".nextval from dual	Statement processed.	1
9	0.00	ALTER SEQUENCE "SQ_OrderID" INCREMENT BY 1	Sequence altered.	0
10	0.00	ALTER SEQUENCE "SQ_ContractID" INCREMENT BY 100	Sequence altered.	0
11	0.01	select "SQ_ContractID".nextval from dual	Statement processed.	1
12	0.00	ALTER SEQUENCE "SQ_ContractID" INCREMENT BY 1	Sequence altered.	0
13	0.00	ALTER SEQUENCE "SQ_CustomerID" INCREMENT BY 100	Sequence altered.	0
14	0.01	select "SQ_CustomerID".nextval from dual	Statement processed.	1
15	0.00	ALTER SEQUENCE "SQ_CustomerID" INCREMENT BY 1	Sequence altered.	0

Download

row(s) 1 - 15 of 18 Next

Alternativ se putea folosi:

alter sequence "SQ_EMPLOYEEID" restart start with 101;

alter sequence "SQ_CustomerID" restart start with 101;

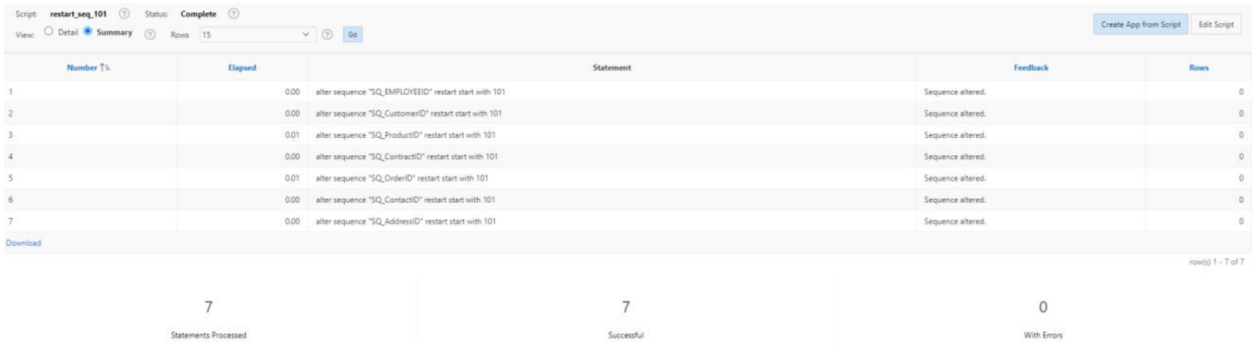
alter sequence "SQ_ProductID" restart start with 101;

alter sequence "SQ_ContractID" restart start with 101;

alter sequence "SQ_OrderID" restart start with 101;

alter sequence "SQ_ContactID" restart start with 101;

alter sequence "SQ_AddressID" restart start with 101;



Script: **restart_seq_101** Status: **Complete**

View: Detail [✓ Summary] Rows 15

Number	Elapsed	Statement	Feedback	Rows
1	0.00	alter sequence "SQ_EMPLOYEEID" restart start with 101	Sequence altered.	0
2	0.00	alter sequence "SQ_CustomerID" restart start with 101	Sequence altered.	0
3	0.01	alter sequence "SQ_ProductID" restart start with 101	Sequence altered.	0
4	0.00	alter sequence "SQ_ContractID" restart start with 101	Sequence altered.	0
5	0.01	alter sequence "SQ_OrderID" restart start with 101	Sequence altered.	0
6	0.00	alter sequence "SQ_ContactID" restart start with 101	Sequence altered.	0
7	0.00	alter sequence "SQ_AddressID" restart start with 101	Sequence altered.	0

Download

row(s) 1 - 7 of 7

7	7	0
Statements Processed	Successful	With Errors

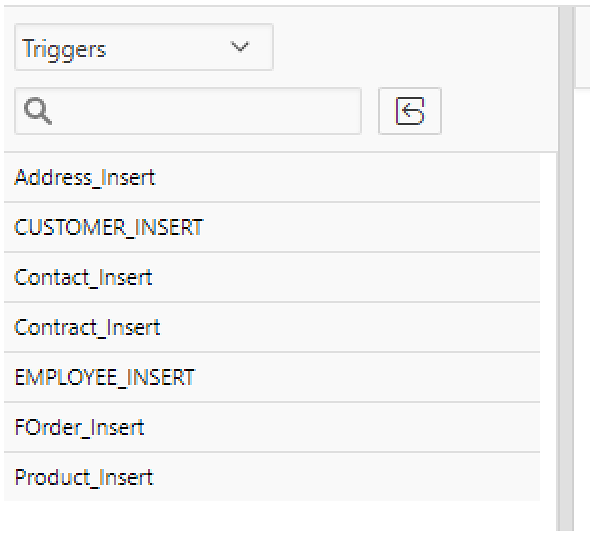
```
create or replace trigger "CUSTOMER_INSERT"
BEFORE
insert on "CUSTOMER"
for each row
DECLARE
BEGIN
  SELECT "SQ_CustomerID".nextval
    INTO :new.CustomerID
    FROM dual;
END;

create or replace trigger "EMPLOYEE_INSERT"
BEFORE
insert on "EMPLOYEE"
for each row
DECLARE
BEGIN
  SELECT "SQ_EMPLOYEEID".nextval
    INTO :new.EmployeeID
    FROM dual;
END;

create or replace trigger "Product_Insert"
BEFORE
insert on "PRODUCT"
for each row
DECLARE
BEGIN
  SELECT "SQ_ProductID".nextval
```

```
    INTO :new.ProductID
    FROM dual;
END;

create or replace trigger "Contract_Insert"
BEFORE
insert on "CONTRACT"
for each row
DECLARE
BEGIN
  SELECT "SQ_ContractID".nextval
    INTO :new.ContractID
    FROM dual;
END;

create or replace trigger "FOrder_Insert"
BEFORE
insert on "FORDER"
for each row
DECLARE
BEGIN
  SELECT "SQ_OrderID".nextval
    INTO :new.OrderID
    FROM dual;
END;

create or replace trigger "Contact_Insert"
BEFORE
insert on "CONTACT"
```

```
for each row
DECLARE
BEGIN
  SELECT "SQ_ContactID".nextval
    INTO :new.ContactID
    FROM dual;
END;

create or replace trigger "Address_Insert"
BEFORE
insert on "ADDRESS"
for each row
DECLARE
BEGIN
  SELECT "SQ_AddressID".nextval
    INTO :new.AddressID
    FROM dual;
END;
```

Scopul acestor triggere este sa automatizeze procesul de atribuire al cheii primare eliminand astfel riscul ca utilizatorii sa introduca chei primare invalide sau nule, lucru care sa genereze un crash al aplicatiei.

Triggerele au fost create cu ajutorul comenzilor de mai sus in contul RO_A722_PLSQL_S35



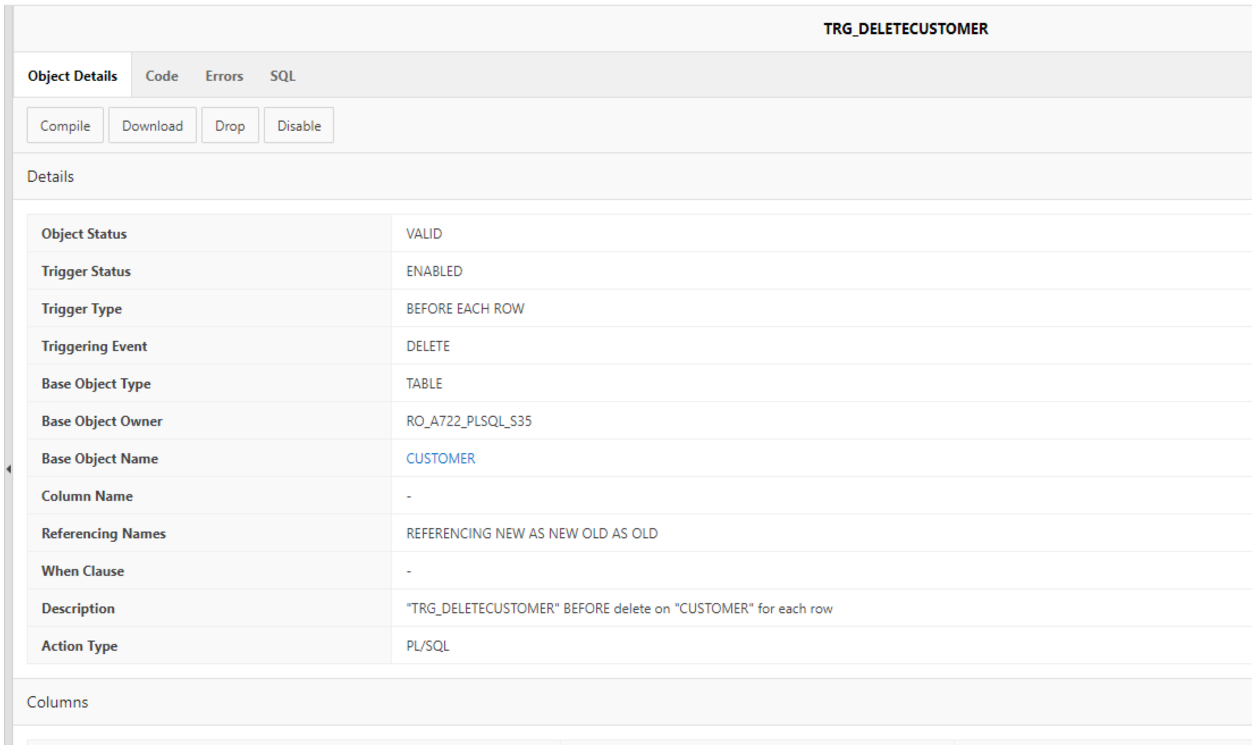
Address_Insert
CUSTOMER_INSERT
Contact_Insert
Contract_Insert
EMPLOYEE_INSERT
FOrder_Insert
Product_Insert

Declansatorii pentru alte tipuri de operatiuni vor fi definiti dupa ce avem si procedurile stocate.

Mai jos este definit un trigger care apeleaza procedura de delete_customer_information din package-ul create.

```
create or replace trigger "TRG_DELETECUSTOMER"

BEFORE

delete on "CUSTOMER"

for each row

begin

"proiect_eadb_fdg".DELETE_CUSTOMER_INFORMATION(:old.CUSTOMERCODE);


end;
```

Astfel, atunci cand este sters un customer, informatiile lui sunt sterse automat.


TRG_DELETECUSTOMER

Object Status	VALID
Trigger Status	ENABLED
Trigger Type	BEFORE EACH ROW
Triggering Event	DELETE
Base Object Type	TABLE
Base Object Owner	RO_A722_PLSQL_S35
Base Object Name	CUSTOMER
Column Name	-
Referencing Names	REFERENCING NEW AS NEW OLD AS OLD
When Clause	-
Description	"TRG_DELETECUSTOMER" BEFORE delete on "CUSTOMER" for each row
Action Type	PL/SQL

Proceduri stocate

Vom incepe prin a crea o functie care sa verifice unicitatea unui cod dat. Scopul ei este de a ne asigura ca atunci cand cream un record nou, acesta va avea un cod unic. Va fi folosita in cadrul procedurilor de tip ‘insert’.

```
create or replace FUNCTION UniqueCustomerCode(c_customercode Customer.CustomerCode%type)
RETURN BOOLEAN IS
  isuniq BOOLEAN;
  c_dbCode Customer.CustomerCode%type;

BEGIN
  SELECT CustomerCode into c_dbCode
  FROM Customer WHERE CustomerCode = c_customercode;
  IF sql%found THEN isuniq := FALSE;

  ELSIF sql%notfound THEN isuniq := TRUE;

  END IF;


  RETURN isuniq ;

END;
```

Pentru simplitate vom crea o functie generica, care poate fi refolosita in mai multe situatii pentru a verifica unicitatea unui camp.

```
create or replace function "GENERICVERIFYCODE" (

   c_code in varchar2,

   c_tablename in varchar2,

   c_fieldname in varchar2)

return boolean

is

  isuniq boolean;

  n_count number;

  v_sql varchar2(256);


BEGIN

  v_sql := 'select ' || 'count(1)' ||

  ' from ' || c_tablename ||

  ' where ' || c_fieldname || ' = ' || chr(39)|| c_code ||chr(39);

  execute immediate v_sql into n_count;


  if n_count > 0 then return false;

  else return true;

  end if;
end "GENERICVERIFYCODE";
```

Prima procedura se va numi ‘addCustomer’, scopul ei fiind acela de a adauga un customer. Va afisa un mesaj simplu daca acest customer a fost adaugat/nu a fost adaugat cu succes. Aceasta procedura va fi de tip private pentru ca modifica date in baza de date cu acces direct, iar noi vrem sa oprim utilizatorii rau voitori sa foloseasca aceasta procedura pentru a face rau bazei de date. Pentru moment o vom crea normal, ca si un “stored procedura” ulterior se va face drop iar aceasta va putea fi apelata doar in package-ul pe care il vom crea.

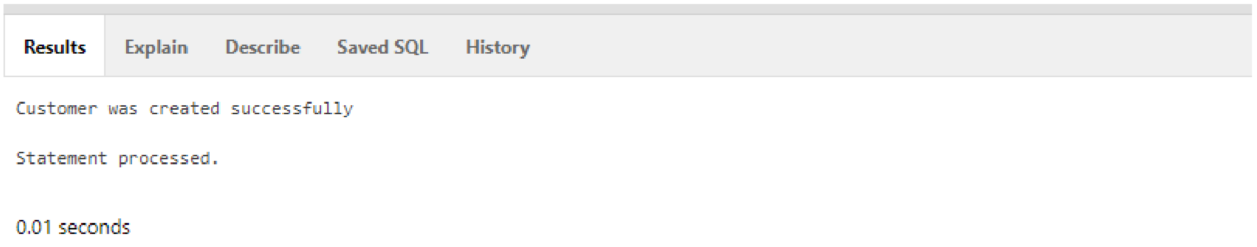
```
create or replace PROCEDURE "ADDCUSTOMER"(
    c_name varchar2,
    c_code  varchar2,
    c_cif  varchar2,
    c_since   date,
          c_notes varchar2)
AS
BEGIN
  INSERT INTO Customer (CustomerName,CustomerCode,CIF,CustomerSince,Notes)
    VALUES(c_name, c_code, c_cif, c_since,c_notes);
  IF sql%found THEN
    dbms_output.put_line('Customer was created successfully');
    elsif sql%notfound THEN dbms_output.put_line('Unable to create customer');
  END IF;
END "ADDCUSTOMER";
```

Procedura foloseste 2 cursori impliciti pentru a determina daca acest customer a fost creat cu succes. Sql%found precizeaza daca 1 sau mai multe randuri au fost afectate de catre instructiunea INSERT, pe cand sql%notfound face opusul.
Folosim functia GenericVerifyCode pentru a ne asigura ca avem coduri unice.

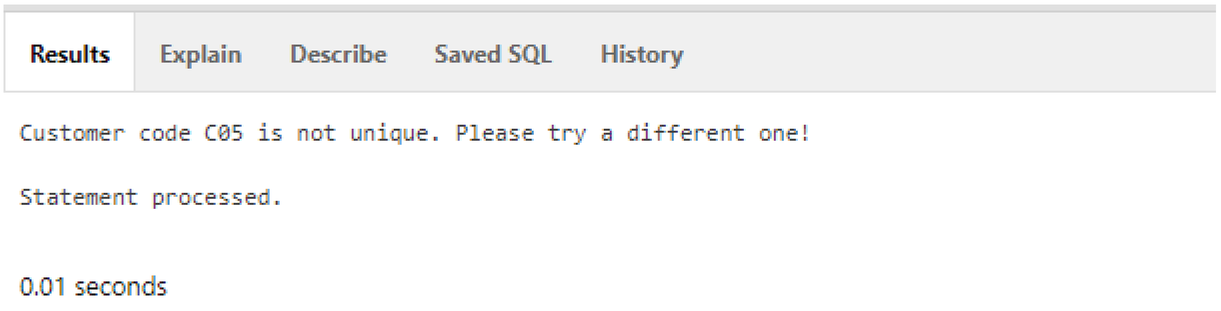
Pentru a testa procedura am folosit codul de mai jos:

```
declare
c_name Customer.CustomerName%type;
    c_code  Customer.CustomerCode%type;
    c_cif  Customer.CIF%type;
    c_since   Customer.CustomerSince%type;
c_notes Customer.Notes%type;
begin
c_name := 'Marcel';
c_code := 'C05';
c_cif := '123123432';
SELECT SYSDATE into c_since FROM DUAL;
c_notes := 'testing if proc works';
```

addCustomer(c_name,c_code,c_cif,c_since,c_notes);
end;



```
Customer was created successfully

Statement processed.

0.01 seconds
```



```
Customer code C05 is not unique. Please try a different one!

Statement processed.

0.01 seconds
```

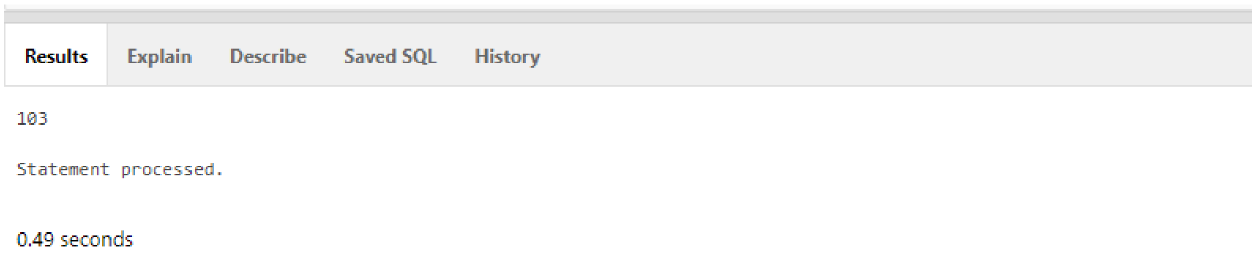
A doua functie pe care o vom crea este una care ne va ajuta sa lucram cu date in baza primary key-urilor. Aceasta functie va fi una private si ne va returna valoarea cheii primare a unui tabel.

```
create or replace function "GETRECORDID"

(c_tablename in VARCHAR2,

c_fieldname in VARCHAR2,

c_value in VARCHAR2)

return number

is

v_sql varchar2(256);

c_primarycolumnname varchar2(50);

n_dbID number;


BEGIN
```

```
    v_sql := 'SELECT cols.column_name
    FROM all_constraints cons NATURAL JOIN all_cons_columns cols
    WHERE cons.constraint_type = ''P'' AND table_name = ' || chr(39)|| UPPER(c_tablename) || chr(39);
    EXECUTE IMMEDIATE v_sql INTO c_primarycolumnname;

    v_sql := 'SELECT ' || c_primarycolumnname ||
     ' FROM ' || c_tablename ||
     ' WHERE ' || c_fieldname || ' = ' || chr(39)|| c_value ||chr(39);
    EXECUTE IMMEDIATE v_sql INTO n_dbID;

    RETURN n_dbID ;

    EXCEPTION
      WHEN no_data_found THEN n_dbID := 0;

    RETURN n_dbID ;
END;
```

Pentru exemplu si totodata si ca test avem urmatorul cod:

```
declare
result number;
begin
 result := getrecordid('Customer','CustomerCode','C05');
  dbms_output.put_line(result);
end;
```



```
103

Statement processed.

0.49 seconds
```

Ceea ce inseamna ca valoarea campului “CustomerID” care este cheia primara pentru tabela “Customer” unde “CustomerCode” este ‘C05’ este 103. Se poate folosi in multe situatii, cum ar fi ca parametru de intrare pentru o procedura de creere a unui Forder sau pentru a face anumite calcule, join-uri.

In situatia in care nu va gasi record-ul, functia va returna 0.

Vom crea un cascade delete pentru customer, cu scopul de a sterge toate datele despre un customer. Aceasta procedura va deveni privata in momentul in care vom define package-ul, dar pentru moment codul ei arata asa:

```
create or replace procedure "CASCADEDELETECUSTOMER" (
  n_customerid  NUMBER)
is
  CURSOR c_CustomerAddress IS SELECT CustomerAddressID FROM CustomerAddress WHERE
CustomerID = n_customerid FOR UPDATE;
  CURSOR c_CustomerContact IS SELECT CustomerContactID FROM CustomerContact WHERE
CustomerID = n_customerid FOR UPDATE;
  CURSOR c_FOrder_Cascade IS SELECT FOrder.OrderID,OrderLine.OrderLineID FROM FOrder,OrderLine
WHERE FOrder.OrderID = OrderLine.OrderID AND CustomerID = n_customerid;

  c_CA_rowtype c_CustomerAddress%rowtype;
  c_CC_rowtype c_CustomerContact%rowtype;

  n_OrderID number;
  n_OrderLineID number;
```

```
begin
   OPEN c_CustomerAddress;
   LOOP
      FETCH c_CustomerAddress INTO c_CA_rowtype;
         EXIT WHEN c_CustomerAddress%notfound;
      DELETE FROM CustomerAddress WHERE CURRENT OF c_CustomerAddress;
   END LOOP;
   CLOSE c_CustomerAddress;

   OPEN c_CustomerContact;
   LOOP
      FETCH c_CustomerContact INTO c_CC_rowtype;
         EXIT WHEN c_CustomerContact%notfound;
      DELETE FROM CustomerContact WHERE CURRENT OF c_CustomerContact;
   END LOOP;
   CLOSE c_CustomerContact;

   OPEN c_FOrder_Cascade;
   LOOP
      FETCH c_FOrder_Cascade INTO n_OrderID,n_OrderLineID;
         EXIT WHEN c_FOrder_Cascade%notfound;
      DELETE FROM OrderLine WHERE OrderLine.OrderLineID = n_OrderLineID;
      DELETE FROM FOrder WHERE FOrder.OrderID = n_OrderID;

   END LOOP;
   CLOSE c_FOrder_Cascade;

end "CASCADEDELETECUSTOMER";
```

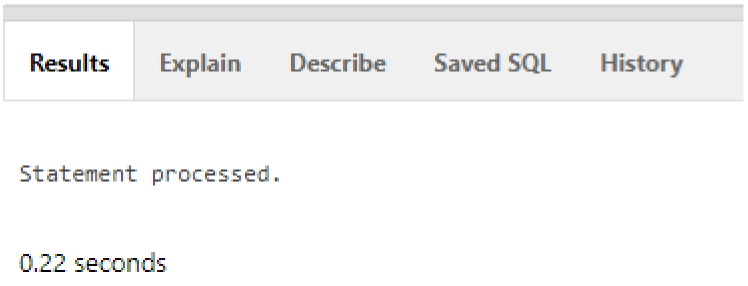
Aceasta procedura foloseste mai multi cursori explicit, unul pentru CustomerAddress, unul pentru CustomerContact si unul multiplu, compus din Forder si OrderLine. Atunci cand un Customer va fi sters, in trigger-ul customer vom avea un apel catre o procedura care va apela procedura noastra.

Va trebui sa renuntam la un foreign key constraint fie in forder sau orderline pentru ca altfel nu le vom putea sterge. Am ales sa renunt la cel care apartine unui orderline, pentru ca putem sterge orderline fara order, dar nu order fara orderline:

Alter table ORDERLINE drop constraint SYS_C0010283816;

Procedura a fost testata folosind codul:

```
declare

n number;

begin

n:=getrecordid('Customer','CustomerName','Tester SRL');

cascadedeletecustomer(n);

end;
```



```
Statement processed.

0.22 seconds
```

S-a sters tabela de mapare “CustomerAddress”, nu Adresa propriu zisa, care poate fi inca legata de alti customeri, acelasi lucru si in cazul Contactului.

In continuare vom crea o procedura care sa apeleze procedura privata de add. Acecasta procedura este publica si valideaza datele de intrare. Customer code-ul va fi unul generat in baza unei formule. CIF trebuie sa fim de minim 6 caractere, iar data folosita va fi cea din sistem. Desigur, se verifica si unicitatea CIF-ului. Astfel avem procedura:

```
create or replace procedure "ADD_CUSTOMER_PUBLIC" (

   c_customername  VARCHAR2,

   c_cif  VARCHAR2,

   c_notes  VARCHAR2)

is
```

```
    any_rows_found number;
    c_customercode_generated varchar(50);
    i number := 123;

begin
  if length(c_cif) < 7
  then
      dbms_output.put_line('CIF provided is too short, minimum length is 6. Complete with 0. Example
000005');
  else
      select count(1) into any_rows_found from CUSTOMER where CIF = c_cif;
      if any_rows_found > 0 then
          dbms_output.put_line('This customer already exists. Please use a different CIF');
      else
          loop
              c_customercode_generated := substr(c_customername, 1, 2) || substr(c_cif, 1, 2) ||
to_number(substr(c_cif,3,2) * 123);
              if GENERICVERIFYCODE(c_customercode_generated, 'Customer', 'CustomerCode') then
                  exit;
              else
                  i := i + 1;
              end if;
          end loop;
          addCustomer(c_customername,c_customercode_generated,c_cif,g_currentdate,c_notes);
      end if;
    end if;
end "ADD_CUSTOMER_PUBLIC";
```

Vom crea o procedura publica pentru stergerea datelor unui customer. In cazul in care acest customer doreste sa fie sters sau sa I se stearga datele, vom putea folosi procedura pentru a realiza acest lucru.

```
create or replace procedure "DELETE_CUSTOMER_INFORMATION"
```

```
(c_customercode  VARCHAR2)
is
n_customerID number;
begin
   n_customerID := GetRecordID('Customer','CustomerCode', c_customercode);
   if n_customerID = 0 then
      dbms_output.put_line('Customer was not found. No operation performed.');
   else
      CascadeDeleteCustomer(n_customerID);
      dbms_output.put_line('Customer data removed for ' || c_customercode);
   end if;
end;
```

Package

Vom crea un package cu numele proiect_eadb_fdg, unde vom defini functiile si procedurile de mai sus.

Cele care sunt private nu vor aparea in package specification, doar in package body, astfel, utilizatorii pachetului nu vor putea sa le apeleze.

Vom define 2 variabile globale, “de context” una g_currentuser, care ne va spune care este user-ul current atunci cand pachetul este incarcat, aceasta se poate si modifica ulterior si folosita in diverse situatii, cum ar fi pentru field-uri precum “CreatedBY”. Si o variable de tip date, care afiseaza data curenta numita g_currentdate. Aceasta va fi folosita pentru field-ul CustomerSince de tip date pentru a ii atribui o valoare in mod automat.

Astfel avem inferfata pachetului:

```
create or replace package "proiect_eadb_fdg" as
```

```
g_currentuser varchar2(50);
g_currentdate date;

function GenericVerifyCode
   (c_code in varchar2, c_tablename in varchar2, c_fieldname in varchar2)
   return boolean;

procedure add_customer_public(
   c_customername  VARCHAR2,
   c_cif  VARCHAR2,
   c_notes  VARCHAR2);

procedure delete_customer_information (
   c_customercode  VARCHAR2);



end;
```

Si body-ul sau implementarea pachetului:

```
create or replace package body "proiect_eadb_fdg" as

/*forward private functions and procedures*/
PROCEDURE "ADDCUSTOMER"(
     c_name varchar2,
     c_code  varchar2,
     c_cif  varchar2,
   c_since   date,
          c_notes varchar2);

PROCEDURE "CASCADEDELETECUSTOMER" (
  n_customerid  NUMBER);

FUNCTION "GETRECORDID" (
  c_tablename in VARCHAR2,
  c_fieldname in VARCHAR2,
  c_value in VARCHAR2)
  RETURN NUMBER;
/*end forwarding*/

function "GENERICVERIFYCODE" (
  c_code in varchar2,
  c_tablename in varchar2,
  c_fieldname in varchar2)
return boolean
is
  isuniq boolean;
  n_count number;
  v_sql varchar2(256);

BEGIN
  v_sql := 'select ' || 'count(1)' ||
  ' from ' || c_tablename ||
  ' where ' || c_fieldname || ' = ' || chr(39)|| c_code ||chr(39);
  execute immediate v_sql into n_count;
```

```
   if n_count > 0 then return false;
   else return true;
   end if;

end "GENERICVERIFYCODE";

procedure "DELETE_CUSTOMER_INFORMATION" (
       c_customercode  VARCHAR2)
is
    n_customerID number;
begin
    n_customerID := GetRecordID('Customer','CustomerCode', c_customercode);
    if n_customerID = 0 then
       dbms_output.put_line('Customer was not found. No operation performed.');
    else
       CascadeDeleteCustomer(n_customerID);
       dbms_output.put_line('Customer data removed for ' || c_customercode);
    end if;
end "DELETE_CUSTOMER_INFORMATION";

procedure "ADD_CUSTOMER_PUBLIC" (
    c_customername  VARCHAR2,
    c_cif  VARCHAR2,
    c_notes  VARCHAR2)
is
    any_rows_found number;
    c_customercode_generated varchar(50);
    i number := 123;
```

```
begin
  if length(c_cif) < 7
  then
      dbms_output.put_line('CIF provided is too short, minimum length is 6. Complete with 0. Example
000005');
  else
      select count(1) into any_rows_found from CUSTOMER where CIF = c_cif;
      if any_rows_found > 0 then
          dbms_output.put_line('This customer already exists. Please use a different CIF');
      else
          loop
              c_customercode_generated := substr(c_customername, 1, 2) || substr(c_cif, 1, 2) ||
to_number(substr(c_cif,3,2) * 123);
              if GENERICVERIFYCODE(c_customercode_generated, 'Customer', 'CustomerCode') then
                  exit;
              else
                  i := i + 1;
              end if;
          end loop;

"proiect_eadb_fdg".addCustomer(c_customername,c_customercode_generated,c_cif,g_currentdate,c_
notes);
      end if;
   end if;
end "ADD_CUSTOMER_PUBLIC";


/*Urmeaza functiile private*/


function "GETRECORDID" (
   c_tablename in VARCHAR2,
   c_fieldname in VARCHAR2,
   c_value in VARCHAR2)
return number
is
   v_sql varchar2(256);
   c_primarycolumnname varchar2(50);
   n_dbID number;
BEGIN
   v_sql := 'SELECT cols.column_name
   FROM all_constraints cons NATURAL JOIN all_cons_columns cols
   WHERE cons.constraint_type = ''P'' AND table_name = ' || chr(39)|| UPPER(c_tablename) || chr(39);
   EXECUTE IMMEDIATE v_sql INTO c_primarycolumnname;

   v_sql := 'SELECT ' || c_primarycolumnname ||
    ' FROM ' || c_tablename ||
    ' WHERE ' || c_fieldname || ' = ' || chr(39)|| c_value ||chr(39);
   EXECUTE IMMEDIATE v_sql INTO n_dbID;

   RETURN n_dbID ;

   EXCEPTION
     WHEN no_data_found THEN n_dbID := 0;

   RETURN n_dbID ;
END "GETRECORDID";

/*Urmeaza procedurile private*/

PROCEDURE "ADDCUSTOMER"(
     c_name varchar2,
     c_code  varchar2,
     c_cif  varchar2,
     c_since   date,
            c_notes varchar2)
AS
BEGIN
  INSERT INTO Customer (CustomerName,CustomerCode,CIF,CustomerSince,Notes)
    VALUES(c_name, c_code, c_cif, c_since,c_notes);
  IF sql%found THEN
   dbms_output.put_line('Customer was created successfully');
   elsif sql%notfound THEN dbms_output.put_line('Unable to create customer');
  END IF;
END "ADDCUSTOMER";


procedure "CASCADEDELETECUSTOMER" (
   n_customerid  NUMBER)
is
   CURSOR c_CustomerAddress IS SELECT CustomerAddressID FROM CustomerAddress WHERE
CustomerID = n_customerid FOR UPDATE;
   CURSOR c_CustomerContact IS SELECT CustomerContactID FROM CustomerContact WHERE
CustomerID = n_customerid FOR UPDATE;
   CURSOR c_FOrder_Cascade IS SELECT FOrder.OrderID,OrderLine.OrderLineID FROM FOrder,OrderLine
WHERE FOrder.OrderID = OrderLine.OrderID AND CustomerID = n_customerid;


   c_CA_rowtype c_CustomerAddress%rowtype;
   c_CC_rowtype c_CustomerContact%rowtype;


   n_OrderID number;
   n_OrderLineID number;
```

```
begin
   OPEN c_CustomerAddress;
   LOOP
      FETCH c_CustomerAddress INTO c_CA_rowtype;
         EXIT WHEN c_CustomerAddress%notfound;
      DELETE FROM CustomerAddress WHERE CURRENT OF c_CustomerAddress;
   END LOOP;
   CLOSE c_CustomerAddress;

   OPEN c_CustomerContact;
   LOOP
      FETCH c_CustomerContact INTO c_CC_rowtype;
         EXIT WHEN c_CustomerContact%notfound;
      DELETE FROM CustomerContact WHERE CURRENT OF c_CustomerContact;
   END LOOP;
   CLOSE c_CustomerContact;

   OPEN c_FOrder_Cascade;
   LOOP
      FETCH c_FOrder_Cascade INTO n_OrderID,n_OrderLineID;
         EXIT WHEN c_FOrder_Cascade%notfound;
      DELETE FROM OrderLine WHERE OrderLine.OrderLineID = n_OrderLineID;
      DELETE FROM FOrder WHERE FOrder.OrderID = n_OrderID;

   END LOOP;
   CLOSE c_FOrder_Cascade;

end "CASCADEDELETECUSTOMER";
```

begin

g_currentuser := USER;

select SYSDATE into g_currentdate from dual;

dbms_output.put_line('Package proiect_eadb_fdg loaded.');

dbms_output.put_line('Current user is ' || g_currentuser);

dbms_output.put_line('Current date is ' || g_currentdate);

end "proiect_eadb_fdg";

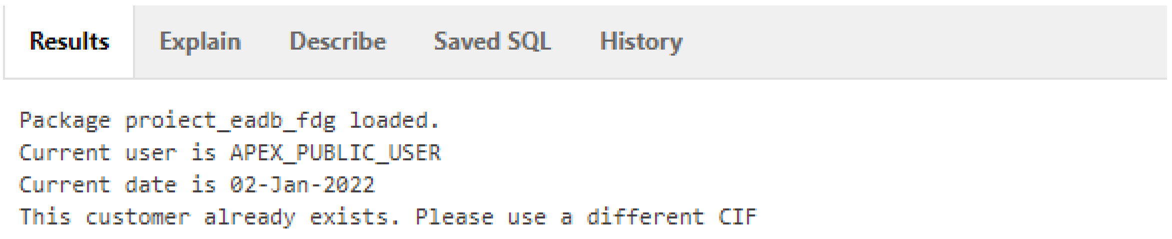
Dupa cum se poate observa, mai sus am facut un forward functiilor si procedurilor private. Acest lucru deoarece dorim sa folosim functii si proceduri private, definite mai jos, pe care pachetul nu le cunoaste.

declare

begin

"proiect_eadb_fdg".ADD_CUSTOMER_PUBLIC('Darius package','12312434','Testing to see if package works');

end;



```
Package proiect_eadb_fdg loaded.
Current user is APEX_PUBLIC_USER
Current date is 02-Jan-2022
This customer already exists. Please use a different CIF
```

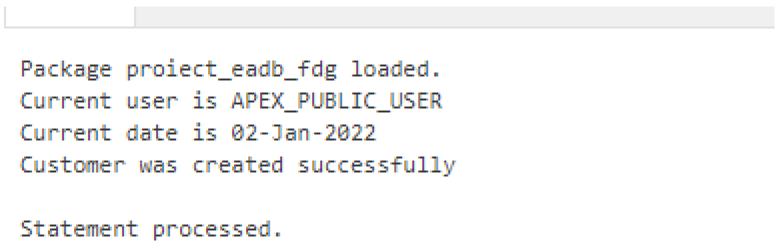
Daca oferim un CIF diferit, va adauga un alt customer, cu un alt cod generat.

declare

begin

"proiect_eadb_fdg".ADD_CUSTOMER_PUBLIC('Darius package','15555555','Testing to see if package works');

end;



```
Package proiect_eadb_fdg loaded.
Current user is APEX_PUBLIC_USER
Current date is 02-Jan-2022
Customer was created successfully

Statement processed.
```