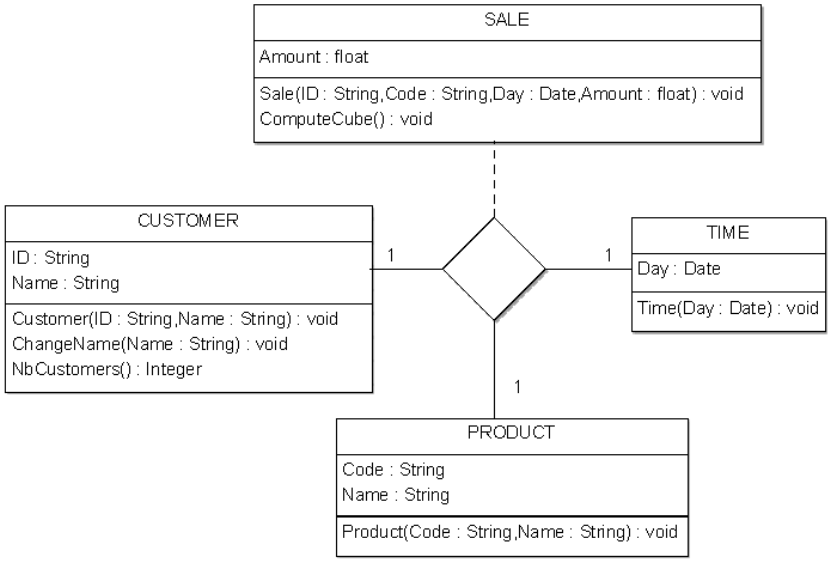
Erasmus Mundus master course in **Data Mining and Knowledge Management** a european master

Complex data warehouses
Academic year 2014-2015
Labwork #1: Star object datamart
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 29/08/14

Let us consider the sample datamart from the course (slide #9), whose conceptual model is recalled below. The objective of this labwork is to implement it under Oracle using its object-relational features.



1. Define types (with names prefixed by T_) corresponding to model classes CUSTOMER, PRODUCT and TIME. Ignore methods for now.

2. In a test PL/SQL anonymous block, instantiate each class (i.e., create a new object of type T_CUSTOMER, T_PRODUCT and T_TIME, respectively).

3. Create 3 object tables aimed at storing instances of classes CUSTOMER, PRODUCT and TIME, respectively. When running again the test PL/SQL block from question #2, are any objects stored in these tables?

4. Modify the test PL/SQL block so that created objects are inserted in the corresponding tables. Then display the tables' contents with an SQL query.

5. Add into the test PL/SQL block a new object of type T_CUSTOMER, and load into it the customer stored in table customer.

6. Define the type corresponding to association SALE (still with no method), an object table to store its instances, populate it with a fact by inserting the corresponding command in the test PL/SQL block, and display its contents with an SQL query.

7. Rerun the fact insertion into table SALE. Should it succeed? Do duplicate insertions in the other tables work? Why?

8. Experiment with the implicit join queries from slide #18.

9. Enforce integrity constraints on table SALES (slides #20-#22). Try again to insert duplicates into table SALE.

10. Modify all types to include the declarations of all methods indicated in the model. Then, write type bodies with the actual code of these methods. Constructors should include object insertions into the corresponding table. Ignore method ComputeCube for now.

11. In the test PL/SQL block, comment all insertions, declare a T_SALE object and instantiate it.

12. Add a (dummy, just to see how it works) MAP function to type T_CUSTOMER, which returns the length of the customer's name. Test the equality of the two customers in the test PL/SQL block (the one created through the constructor and the one loaded from the CUSTOMER table, which should be identical).

13. Create a new type T_CUBE(CustomerName, ProductName, Day, SumAmount) and an object view CUBE based on T_CUBE. Query CUBE with SQL. What do the NULL values stand for in the result?

14. Include method ComputeCube into type T_SALE and its body. ComputeCube basically displays the contents of object view CUBE. Invoke method ComputeCube in the test PL/SQL block.

Solution

```
-- Clean-up

DROP VIEW CUBE;
DROP TABLE SALE;
DROP TABLE CUSTOMER;
DROP TABLE PRODUCT;
DROP TABLE TIME;
DROP TYPE T_CUBE;
DROP TYPE T_SALE;
DROP TYPE T_CUSTOMER;
DROP TYPE T_PRODUCT;
DROP TYPE T_TIME;

-- Types

CREATE OR REPLACE TYPE T_CUSTOMER AS OBJECT (
  ID VARCHAR(25),
  Name VARCHAR(255),
  CONSTRUCTOR FUNCTION T_Customer (ID VARCHAR, Name VARCHAR) RETURN SELF AS
RESULT,
  MEMBER PROCEDURE ChangeName (Name VARCHAR),
  STATIC FUNCTION NbCustomers RETURN NUMBER,
  MAP MEMBER FUNCTION Len RETURN NUMBER
)
/

CREATE OR REPLACE TYPE T_PRODUCT AS OBJECT (
  Code VARCHAR(25),
  Name VARCHAR(255),
  CONSTRUCTOR FUNCTION T_Product (Code VARCHAR, Name VARCHAR) RETURN SELF AS
RESULT
)
/

CREATE OR REPLACE TYPE T_TIME AS OBJECT (
  Day DATE,
  CONSTRUCTOR FUNCTION T_Time (Day DATE) RETURN SELF AS RESULT
)
/

CREATE OR REPLACE TYPE T_SALE AS OBJECT (
  RefCustomer REF T_CUSTOMER,
  RefProduct REF T_PRODUCT,
  RefTime REF T_TIME,
  Amount FLOAT,
  CONSTRUCTOR FUNCTION T_Sale (ID VARCHAR, Code VARCHAR, Day DATE, Amount FLOAT)
RETURN SELF AS RESULT,
  STATIC PROCEDURE ComputeCube
)
/

CREATE OR REPLACE TYPE T_CUBE AS OBJECT (
  CustomerName VARCHAR(255),
  ProductName VARCHAR(255),
  Day DATE,
  SumAmount FLOAT
)
/
```



```
-- Object tables & views

CREATE TABLE CUSTOMER OF T_CUSTOMER (
  CONSTRAINT CUSTOMER_PK PRIMARY KEY (ID)
);

CREATE TABLE PRODUCT OF T_PRODUCT (
  CONSTRAINT PRODUCT_PK PRIMARY KEY (Code)
);

CREATE TABLE TIME OF T_TIME (
  CONSTRAINT TIME_PK PRIMARY KEY (Day)
);

CREATE TABLE SALE OF T_SALE (
  CONSTRAINT SALE_CUSTREF RefCustomer    REFERENCES CUSTOMER,
  CONSTRAINT SALE_PRODREF RefProduct REFERENCES PRODUCT,
  CONSTRAINT SALE_TIMEREF RefTime   REFERENCES TIME
);

CREATE OR REPLACE TRIGGER TRIG_SALE_PK
BEFORE INSERT OR UPDATE ON SALE
FOR EACH ROW
DECLARE
  n INTEGER;
  null_problem EXCEPTION;
  double_problem EXCEPTION;
BEGIN
  IF :NEW.RefCustomer IS NULL OR :NEW.RefProduct IS NULL OR :NEW.RefTime IS NULL
THEN
    RAISE null_problem;
  END IF;
  SELECT COUNT(*) INTO n FROM SALE s WHERE s.RefCustomer = :NEW.RefCustomer  AND
s.RefProduct = :NEW.RefProduct     AND s.RefTime = :NEW.RefTime;
  IF n > 0 THEN
    RAISE double_problem;
  END IF;
EXCEPTION
  WHEN null_problem THEN RAISE_APPLICATION_ERROR(-20501, 'Primary key with NULL
value(s).');
  WHEN double_problem THEN RAISE_APPLICATION_ERROR(-20502, 'Primary key already
exists.');
END;
/

CREATE VIEW CUBE OF T_CUBE
WITH OBJECT IDENTIFIER (CustomerName, ProductName, Day) AS
  SELECT s.RefCustomer.Name, s.RefProduct.Name, s.RefTime.Day, SUM (Amount)
  FROM SALE s
  GROUP BY ROLLUP (s.RefCustomer.Name, s.RefProduct.Name, s.RefTime.Day);

-- Type bodies

CREATE OR REPLACE TYPE BODY T_CUSTOMER AS
    CONSTRUCTOR FUNCTION T_Customer (ID VARCHAR, Name VARCHAR) RETURN SELF AS
RESULT IS
         BEGIN
              SELF.ID := ID;
              SELF.Name := Name;
              INSERT INTO CUSTOMER VALUES (SELF);
              RETURN;
         END;
```

```
    MEMBER PROCEDURE ChangeName (Name VARCHAR) IS
         BEGIN
              SELF.Name := Name;
              UPDATE CUSTOMER
                  SET OBJECT_VALUE = SELF
                  WHERE ID = SELF.ID;
         END;
    STATIC FUNCTION NbCustomers RETURN NUMBER IS
         n INTEGER;
         BEGIN
              SELECT COUNT(*) INTO n FROM CUSTOMER;
              RETURN n;
         END;
  MAP MEMBER FUNCTION Len RETURN NUMBER IS
    BEGIN
      RETURN LENGTH(SELF.Name);
    END;
END;
/

CREATE OR REPLACE TYPE BODY T_PRODUCT AS
    CONSTRUCTOR FUNCTION T_Product (Code VARCHAR, Name VARCHAR) RETURN SELF AS
RESULT IS
         BEGIN
              SELF.Code := Code;
              SELF.Name := Name;
              INSERT INTO PRODUCT VALUES (SELF);
              RETURN;
         END;
END;
/

CREATE OR REPLACE TYPE BODY T_TIME AS
    CONSTRUCTOR FUNCTION T_Time (Day DATE) RETURN SELF AS RESULT IS
         BEGIN
              SELF.Day := Day;
              INSERT INTO TIME VALUES (SELF);
              RETURN;
         END;
END;
/

CREATE OR REPLACE TYPE BODY T_SALE AS
    CONSTRUCTOR FUNCTION T_Sale (ID VARCHAR, Code VARCHAR, Day DATE, Amount
FLOAT) RETURN SELF AS RESULT IS
         BEGIN
      SELECT REF(c) INTO SELF.RefCustomer FROM CUSTOMER c WHERE c.ID = ID;
      SELECT REF(p) INTO SELF.RefProduct FROM PRODUCT p WHERE p.Code = Code;
      SELECT REF(t) INTO SELF.RefTime FROM TIME t WHERE t.Day = Day;
              SELF.Amount := Amount;
              INSERT INTO SALE VALUES (SELF);
              RETURN;
         END;
  STATIC PROCEDURE ComputeCube IS
      CURSOR ListCube IS SELECT * FROM CUBE;
      t ListCube%ROWTYPE;
    BEGIN
      FOR t IN ListCube LOOP
        DBMS_OUTPUT.PUT_LINE(t.CustomerName || ' + ' || t.ProductName || ' + '
|| t.Day || ' = ' || t.SumAmount);
      END LOOP;
    END;
END;
/
```



```
-- Test PL/SQL block

DECLARE
  c T_CUSTOMER;
  c2 T_CUSTOMER;
  p T_PRODUCT;
  t T_TIME;
  s T_SALE;
BEGIN
  -- 2
  c := NEW T_CUSTOMER('Cust001', 'Darmont');
  p := NEW T_PRODUCT('Pr32blue', 'Blue thing');
  t := NEW T_TIME(SYSDATE);
  -- 4
  -- INSERT INTO CUSTOMER VALUES (c);
  -- INSERT INTO PRODUCT VALUES (p);
  -- INSERT INTO TIME VALUES (t);
  -- 5
  SELECT VALUE(c) INTO c2 FROM CUSTOMER c WHERE c.ID = 'Cust001';
  -- 6
  /* INSERT INTO SALE VALUES (
    (SELECT REF(c) FROM CUSTOMER c WHERE c.ID = 'Cust001'),
    (SELECT REF(p) FROM PRODUCT p WHERE p.Code = 'Pr32blue'),
    (SELECT REF(t) FROM TIME t WHERE t.Day = SYSDATE),
    500.75
  ); */
  -- 11
  s := NEW T_SALE(c.ID, p.Code, t.Day, 500.75);
  -- 12
  IF c = c2 THEN
    DBMS_OUTPUT.PUT_LINE('c = c2, no way!');
  END IF;
  -- 14
  T_SALE.ComputeCube;
END;
/

-- Tables' contents (4)

SELECT * FROM CUSTOMER;
SELECT * FROM PRODUCT;
SELECT * FROM TIME;
SELECT * FROM SALE;

-- Implicit join queries (8)

SELECT s.Amount FROM SALE s
     WHERE s.RefCustomer.Name = 'Darmont';
SELECT DISTINCT s.RefCustomer.Name FROM SALE s WHERE s.Amount > 500;
SELECT DEREF(RefCustomer) FROM SALE s;

-- Object view (13)

SELECT * FROM CUBE;
```