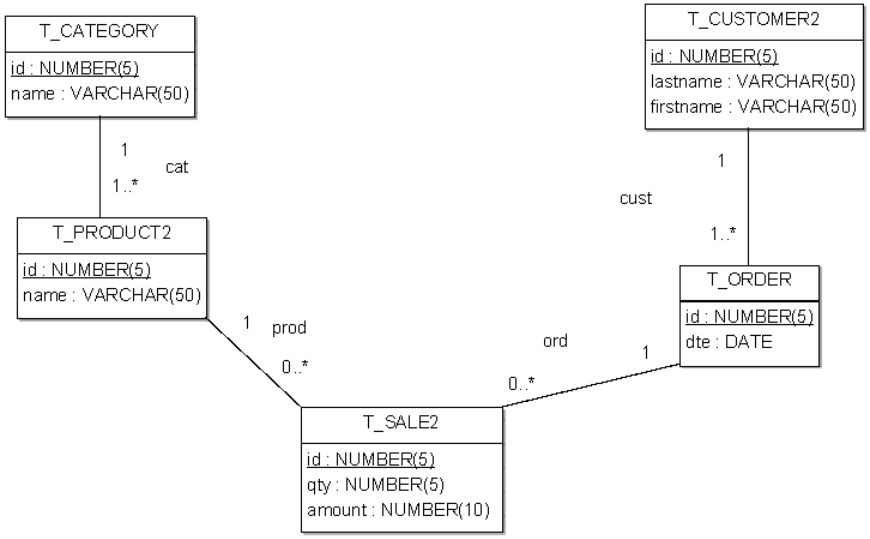
Erasmus Mundus master course in **Data Mining and Knowledge Management** a european master

Complex data warehouses
Academic year 2014-2015
Labwork #2: Snowflake object datamart
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 29/08/14

Let us consider the datamart whose conceptual snowflake (i.e., with hierarchies) schema is represented below.



1. Create the Oracle types corresponding to this conceptual model. Represent associations with references, e.g., in T_PRODUCT2, *cat* is a reference to an object of type T_CATEGORY.

2. Create the object tables indicated in the table below. D_ and F_ stand for dimension and fact, respectively.

Object table	Associate type
D_CATEGORY	T_CATEGORY
D_PRODUCT	T_PRODUCT2
D_CUSTOMER	T_CUSTOMER2
D_ORDER	T_ORDER
F_SALE	T_SALE2

3. Feed table D_CATEGORY with the help of an SQL query that associates a numerical identifier (e.g., the ROWNUM pseudocolumn) to each distinct category from table DARMONT.DEMO_PRODUCT_INFO. Display table D_CATEGORY.

4. Feed table D_PRODUCT with the help of an SQL query that selects attributes PRODUCT_ID and PRODUCT_NAME from table DARMONT.DEMO_PRODUCT_INFO and the reference to CATEGORY from table D_CATEGORY. Display table D_PRODUCT.

5. Feed table D_CUSTOMER with the help of an SQL query that selects attributes CUSTOMER_ID, CUST_LAST_NAME and CUST_FIRST_NAME from table DARMONT.DEMO_CUSTOMERS. Display table D_CUSTOMER.

6. Feed table D_ORDER with the help of an SQL query that selects attributes ORDER_ID and ORDER_TIMESTAMP from table DARMONT.DEMO_ORDERS and the reference to CUSTOMER_ID from table D_CUSTOMER. Display table D_ORDER.

7. Finally, feed table F_SALE with the help of an SQL query that selects attributes ORDER_ITEM_ID, QUANTITY and QUANTITY * LIST_PRICE (calculated field) from tables DARMONT.DEMO_ORDER_ITEMS and DARMONT.DEMO_PRODUCT_INFO (join), and the references to ORDER_ID from D_ORDER and to PRODUCT_ID from D_PRODUCT. Display table F_SALE.

8. From table F_SALE, compute the cube providing the sum of quantities and amounts per product (indicate product name) with an SQL query using the GROUP BY CUBE clause. Use the implicit join allowed by object references, and the GROUPING[1] and DECODE[2] functions to identify superaggregates as ALL instead of NULL.

9. Same query as #8, but only for products of category “Computer”.

10. Activate the query execution plan’s display (SET AUTOTRACE ON EXPLAIN).

11. From object table F_SALE, compute the cube providing the sum of quantities and amounts per product category (indicate category name) and per customer (indicate customer last name) with an SQL query using the GROUP BY ROLLUP clause. Use the implicit join allowed by object references, and the GROUPING and DECODE functions to identify superaggregates as ALL.

12. Same query as #11, but from source relational tables DARMONT.DEMO_ORDER_ITEMS, DARMONT.DEMO_ORDERS, DARMONT.DEMO_CUSTOMERS and DARMONT.DEMO_PRODUCT_INFO. Conclusion (query writing, efficiency)?

13. Same query as #11, using the GROUP BY CUBE clause. What is the difference between GROUP BY CUBE and GROUP BY ROLLUP (in terms of result and efficiency)?

[1] http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions064.htm#SQLRF00647
[2] http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions042.htm#SQLRF00631

Solution

```
-- Datamart class schema

create type t_category as object(
  id number(5),
  name varchar(50)
)
/

create type t_product2 as object(
  id number(5),
  name varchar(50),
  cat ref t_category
)
/

create type t_customer2 as object(
  id number(5),
  lastname varchar(50),
  firstname varchar(50)
)
/

create type t_order as object(
  id number(5),
  dte date,
  cust ref t_customer2
)
/

create type t_sale2 as object(
  id number(5),
  qty number(5),
  amount number(10),
  ord ref t_order,
  prod ref t_product2
)
/

-- Object tables

create table d_category of t_category (
     constraint d_category_pk primary key(id)
);

create table d_product of t_product2 (
     constraint d_product_pk primary key(id),
     constraint d_category_ref cat references d_category
);

create table d_customer of t_customer2 (
     constraint d_customer_pk primary key(id)
);

create table d_order of t_order (
     constraint d_order_pk primary key(id),
     constraint d_customer_ref cust references d_customer
);
```



```
create table f_sale of t_sale (
     constraint f_sale_pk primary key(id),
     constraint d_product_ref prod references d_product,
     constraint d_order_ref ord references d_order
);

-- ETL

insert into d_category select rownum, category from (select distinct category
from darmont.demo_product_info);

select * from d_category;

insert into d_product
  select product_id, product_name,
    (select ref(c) from d_category c where c.name = category)
  from darmont.demo_product_info;

select * from d_product;

insert into d_customer
  select customer_id, cust_last_name, cust_first_name from
darmont.demo_customers;

select * from d_customer;

insert into d_order
  select order_id, order_timestamp,
    (select ref(c) from d_customer c where c.id = customer_id)
  from darmont.demo_orders;

select * from d_order;

insert into f_sale
  select order_item_id, quantity, quantity * list_price,
    (select ref(o) from d_order o where o.id = order_id),
    (select ref(p) from d_product p where p.id = oi.product_id)
  from darmont.demo_order_items oi, darmont.demo_product_info pi
  where oi.product_id = pi.product_id ;

select * from f_sale;

-- Decision queries by implicit join

select
  decode(grouping(s.prod.name), 1, 'ALL', s.prod.name) product,
  sum(qty), sum(amount)
from f_sale s
group by cube(s.prod.name);

select
  decode(grouping(s.prod.name), 1, 'ALL', s.prod.name) product,
  sum(qty), sum(amount)
from f_sale s
where s.prod.cat.name = 'Computer'
group by cube(s.prod.name);
```

```
-- Comparison of implicit join on object table vs. relational join

set autotrace on explain

select
  decode(grouping(s.prod.cat.name), 1, 'ALL', s.prod.cat.name) category,
  decode(grouping(s.ord.cust.lastname), 1, 'ALL', s.ord.cust.lastname) customer,
  sum(qty), sum(amount)
from f_sale s
group by rollup(s.prod.cat.name, s.ord.cust.lastname);

select
  decode(grouping(category), 1, 'ALL', category) categorie,
  decode(grouping(cust_last_name), 1, 'ALL', cust_last_name) client,
  sum(quantity) qte, sum(quantity*list_price) montant
from darmont.demo_order_items doi, darmont.demo_orders do,
darmont.demo_customers dc, darmont.demo_product_info dpi
where doi.order_id = do.order_id
and do.customer_id = dc.customer_id
and doi.product_id = dpi.product_id
group by rollup(category, cust_last_name);
select
  decode(grouping(s.prod.cat.name), 1, 'ALL', s.prod.cat.name) category,
  decode(grouping(s.ord.cust.lastname), 1, 'ALL', s.ord.cust.lastname) customer,
  sum(qty), sum(amount)
from f_sale s
group by cube(s.prod.cat.name, s.ord.cust.lastname);
```