UDC 683.1

Hryhorovych A., *Hryhorovych V.
Drohobych Pedagogic Lyceum,
Ivan Franko Str., 36, Drohobych, 82100, Ukraine,
*Ivan Franko Drohobych State Pedagogic University,
cathedra of informational systems and technologies,
Ivan Franko Str., 24, Drohobych, 82100, Ukraine,
E-mail: a_grygorovych@mail.ru, vgrig@bigmir.net

REALIZATION OF OPERATIONS WITH NESTED RELATIONS

There have been reviewed object-relational peculiarities and work with nested relations in the industrial SMDB. There have been given the examples of operations with nested relations by means of Oracle, SQL3, SQL4 and XML.

Keywords –.nested relations, Oracle, SQL3, SQL4, XML.

Nested relations are especially important for the programs of data base elaboration, specifically a subclass of nested relations called relations in *partitioned normal form (PNF)*. A relation is in partitioned normal form when its atomic attributes are a superkey of the relation and when any non-atomic component of this relation's tuple also is in partitioned normal form (PNF). Relations in partitioned normal form were investigated in [1-5]. This question arised for the first time in 1977 when A.Makinouchi said [6] that conditions of the first normal form (1NF), put on relational data base do not fit different programs for data base elaboration, such as *Information Retrieval Systems* or *Computer-Aided Design and Manufacturing, CAD/CAM*. Such programs need to operate with structured substances when the first normal form lets only atomic (non-divided) meanings for attributes. Nested relations got a new stimulus for the development because of the wide use of *Object-Relational Data Model*, specifically beause of insertion of the proper possibilities to industrial systems of management of data base (SMDB) (for instance, Oracle) and to the existing standards – SQL3 (SQL-99). Object-relational data models (ORDM) make a relational model wider through insertion of object-oriented possibilities and providing of design of non-standard data types; they allow complicated data types for attributes in tuples, including non-atomic meanings, such as nested relation. ORDM reserve relational basement, specifically delarative access to the data, and make the possibilities of a data model wider; they are compatible with the existing relational languages.

1. Object-relational peculiarities and work with nested tables in Oracle

Defining of the object types that include nested relations

Oracle allows us to define types like the types SQL. For example, defining of a point signed by two coordinated looks like:

```
CREATE TYPE PointType AS OBJECT (
        x NUMBER,
        y NUMBER );
```

We may use the object type like any other type in the further defining of object types or types-tables. For example, we may define a type of a line:

```
CREATE TYPE LineType AS OBJECT (
       end1 PointType,
       end2 PointType  );
```

After this we may create a relation that includes lines of the type LineType:

```
CREATE TABLE Lines (
       lineID INT,
       line   LineType );
```

Design of object quantities

Like C++, Orale provides inluded constructors for the results of the announces type that have the type's name. So, a result of the type `PointType` is formed with a word PointType and a list of the corresponding meanings put into brackets. For example, to create a line segment with ID27 in `Lines` that lies from the eginning of the coordinates to the point (3; 4), we shall write:

```
INSERT INTO Lines
         VALUES(27, LineType(
```

```
                    PointType(0.0, 0.0),
                    PointType(3.0, 4.0)  )  );
```

I. e., we have created two meanings of the type PointType, these meanings are been used to create meanings of the type `LineType`, and this meaning is been used with the whole 27 to create a tuple for `Lines`.

Nested tables

Extremely powerful possibility of the using the object types in Oracle is that the *column types*, i.e. types of attributes can be a table-type [7,8]. I. e., meaning of attributes in one tuple can be a full relation, as depicted in a Fig. 1, where a relation with a scheme (a, b) has the meaning b, that by its turn is a relation with a scheme (x, y, z)

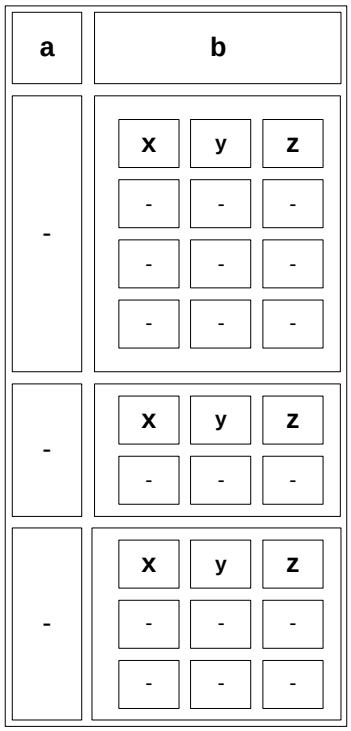

Fig. 1 Scheme of a nested relation R(a, b(x,y,z,))

To receive a relation as a type of some attribute, we need to define a type at once, using an attribute `AS TABLE OF`. For example,

```
CREATE TYPE PolygonType AS TABLE OF PointType;
```

The given definition points us that a type `PolygonType` is a relation, which tuples have a type `PointType`, i.e., they have two real components x and y.

Now we may to define the relation, one of the columns of which has meaningh that perform polygons; i.e., they are a complex of points. There the following definition may take place, in which the polygons are performed with names and complexes of points:

```
CREATE TABLE Polygons (
        name   VARCHAR2(20),
        points PolygonType )
        NESTED TABLE points STORE AS PointsTable;
```

Here “elementary” relations, that perform corresponding polygons, are retained not directly as meanings of attributes `points`, but in a single table `PointsTable`, that has to be described after put into brackets of the list attributes of the table `Polygons`, and that is used for retaining of relations of a type `PolygonType`.

To insert lines into relations which have one or several columns that are nested relations (e.g., into the relation `Polygons`), we use a constructor of a type for a nested relation (in our example `PolygonType`) to surround the meaning of the nested relation. The meaning of the nested relation is performed with the list of the meaning of a corresponding type; in our example the type `PointType` is performed with the constructor of the type with the same name.

With the help of the following construction we insert a polygon “square” that consists of four points and depicts a single square:

```
INSERT INTO Polygons VALUES(
```

```
'square', PolygonType(PointType(0.0, 0.0), PointType(0.0, 1.0),
          PointType(1.0, 0.0), PointType(1.0, 1.0)  ) );
```

We may receive the points of this square using the inquire

```
SELECT points
  FROM Polygons
  WHERE name = 'square';
```

We also may receive the details of a nested relation with the help of the inquire `FROM` using the key word `THE` applied to a subinquire the result of which is the relation. If the inquire abow reverts the whole relation, the result of the following will be the points of the square which are placed on the main diagonal (i.e., x = y):

```
SELECT ss.x
FROM THE(SELECT points
         FROM Polygons
         WHERE name = 'square'
        ) ss
WHERE ss.x = ss.y;
```

In this inquire the nested relation received a supplemental parameter ss that is being used in requires SELET and WHERE as if it were a usual relation.

Combination of nested relations and references

In a case of creating of nested tables the tuples of which in fact are indicators (reference books) to the tuples of another table (we can see it in normalization of the data), a problem arises that the attribute of the nested table has no name. For using in such cases Oracle gives the `name COLUMN_VALUE`.

Let's change the relation that depicts polygons to receive nested tables of indicators (reference books). First, we create a new type that is a nested table of indicators (of reference books) to the points:

```
CREATE TYPE PolygonRefType AS TABLE OF REF PointType;
```

After that we create a new relation, similar to `Polygons`, but with the points retained as a nested table of indicators:

```
CREATE TABLE PolygonsRef (
         name   VARCHAR2(20),
         pointsRef PolygonRefType)
         NESTED TABLE pointsRef STORE AS PointsRefTable;
```

It is necessary to remember that the points have to be directly retained in some relation of the type `PointType`. In this case, constructing a require to find the points that are situated on a main diagonal, it is necessary to use `COLUMN_VALUE` to address to the column of the nested table. The require will look like:

```
SELECT ss.COLUMN_VALUE.x
FROM THE(SELECT pointsRef
         FROM PolygonsRef
         WHERE name = 'square'
        ) ss
WHERE ss.COLUMN_VALUE.x = ss.COLUMN_VALUE.y;
```

2. II. Realization of nested tables with the means SQL3 and SQL4

To create nested tables where a column of one table includes in fact another table we use collections. Collections are constructors of types that are used to define collections of other types. Collections are used to retain several meanings in one column of a table. As a result, there can be created one table that includes several levels on nestedness of the type "a main/a subordinate". So, collections let us fulfill projecting of physical structure of the data base more flexible.

In a standard SQL3 there is паратремизований type of the collection `ARRAY`, and in a standatd SQL4 it is expected an additional introducing паратремизованих types of the collections `LIST, SET` and `MULTISET` [9]. In any case a parameter, called an *element of a type*, can have a previously defined type, a typeof a user UDT (User-Defined Types), a line type or to be a collection, but it cannot be an indicating type or a type UDT that includes an indicating type. Besides, Each collection has to be homogenous, i.e. all its elements must have the same type or arise from hierarchy of one type. The types of the collections have the following definitions:

`ARRAY`. One-dimension massif, maximum elements are indicated for which.

`LIST`. A put in order collection where duplicated are allowed.

`SET`. Not in order collection where duplicates are forbidden.

`MULTISET`. Not in order collection where duplicates are allowed.

These types are analogous to the types defined in a standard ODMG 3.0 [9, p. 1012-1036], exepting that the name Bag is changed with the data type MULTISET of the language SQL. Such operations ate expected for these collections:

Two sets SET are used as operandes and return a resulr of he type SET;

Two multisets MULTISET are used as operandes and return a result of a type MULTISET;

A multiset MULTISET is used as an operand and returns a result of a type SET;

Any collection is used as an operand and returns a result equal to the number of the elements in this collection;

Two lists LIST are used as operandes and return a result of a type LIST;

Two lists LIST are used as operandes and return a result of a type INTEGER;

A list LIST and one or two whole numbers are used as operandes and return a result of a type LIST;

An array ARRAY and one or two whole numbers are used as operandes and return an element of an array ARRAY;

Two massifs ARRAY are used as operandes and concatenation is fulfilled of these two arrays in one array ARRAY in an indicated order .The next examples show us the possibilities of the work with the collections in SQL.

Using of the collection of the type SET

It is necessary to supplement a table STAFF, so that it would be possible to save the data about the number of family members (next of kin), and then to find the names and last names of the members with the name John White.

The collection of the data type SET allows us to include a column into the table STAFF in this way:

```
NextOfKin SET (PersonType)
```

In this case the inquire will look like:

```
SELECT n.fName, n.lName
FROM Staff s, TABLE (s.NextOfKin) n
WHERE s.lName = 'White' AND s.fName = 'John';
```

Using the collection of the type ARRAY

If the limit is expected according to which we allow retaining of the data about not more that about three members of the family, we may this column like the data of the type ARRAY.

```
NextOfKin PersonType ARRAY(3)
```

In this case the inquire that expects a selection of the data only about the first member of the family will look like:

```
SELECT s.NextOfKin[1].fName, s.nextOfKin[1].lName
FROM Staff s
WHERE s.lName = 'White' AND s.fName = 'John';
```

Because it is not permitted to use the array as an indicator to a table, it is necessary to ise the whole form of a note in the list of the selection of an operator.

3. Nested relations in XML

In the language XML (eXtensible Markup Language) the data are given as lines of a text that includes insertions of markup that serves to description of qualities of the data. Using of the markup allows us to supplement the text with the information that touches its contents or its shape.

In XML we may use two approaches to present the relational data:

- **Relation of the type "one to many"** are presented as separate tables of lines that are linked with the common columns. These common columns are defined in the collections as the keys, and the relation takes place among them.

- **Nested relations** are presented hierarchically, i.e., paternal elements include nested filial elements. In the following example there are showed orders Orders of a customer, nested to the element Customers [10]. I.e., the data about customers and all their orders are grouped in one hyerarch with Customers.

```
<CustomerOrders>
  <Customers>
    <CustomerID>ALFKI</CustomerID>
    <Orders>
      <OrderID>10643</OrderID>
        <OrderDetails>
      <CustomerID>ALFKI</CustomerID>
      <OrderDate>1997-08-25</OrderDate>
```

```
      </Orders>
      <Orders>
        <OrderID>10692</OrderID>
        <CustomerID>ALFKI</CustomerID>
        <OrderDate>1997-10-03</OrderDate>
      </Orders>
      <CompanyName>Alfreds Futterkiste</CompanyName>
    </Customers>
    <Customers>
      <CustomerID>ANATR</CustomerID>
      <Orders>
        <OrderID>10308</OrderID>
        <CustomerID>ANATR</CustomerID>
        <OrderDate>1996-09-18</OrderDate>
      </Orders>
      <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
    </Customers>
  </CustomerOrders>
```

4. Conclusion

The rewieved possibilities allowed us to approve that it is already realized the corresponding widening of the functions of Systems of managing of data bases (SMDB) to provide the support to the operations upon the nested relations. Many existing SMDB include instrumental means for elaboration of nested relations, thanks to which there is solved the problem of proper presentation of the data in informational systems that continues to be actual: on one side – for the existing problems new models of the data that more properly present the subject's sphere have being built; on the other side – new problems arise that need the elaborating of the data of new nature. The usage of the nested relations in investigation of the models of spatial data is especially perspective, - it will allow us to realize scaling at the level of the data model: depending on the scale a object can be considerated as "simple" or as such one that has some spatial structure.

References

[1] F. Bancilhon, P. Richard and M. Scholl. On line processing of compacted relations. In Proc. 8th International Conference on Very Large Databases, pages 263-269, Mexico City, Sep. 1982.

[2] S. Abiteboul and N. Bidoit. Non first normal form relations: An Algebra allowing data restructuring. Journal of Computer and System Sciences, 33: 361-393, 1986.

[3] M. A. Roth, H. F. Korth and A. Silberschatz. Extended algebra and calculus for nested relational databases. ACM Transactions on Database Systems, 13 (4): 389-417, Dec. 1988.

[4] G. Hulin. A Relational Algebra for Nested Relations in Partitioned Normal Form. Manuscript M318, Philips Research Laboratory Brussels, Oct. 1989.

[5] G. Hulin. On Restructuring Nested Relations in Partitioned Normal Form. In Proc. 16th VLDB Conference, p. 626-637. Brisbane, Australia 1990. www.vldb.org/conf/1990/P626.PDF

[6] A. Makinouchi. A consideration on normal form of not-necessarily-normalized relation in the relational data model. In Proc. 3rd International Conference on Very Large Databases, pages 447-453, Tokyo, Oct. 1977.

[7] Фейерштейн С., Прибыл Б. Oracle PL/SQL для профессионалов 3-е изд. – Спб.: Питер, 2003. С. 799 – 857.

[8]Jeff Ullman. Object-Relational Features of Oracle. www-db.stanford.edu/~ullman/fcdb/oracle/or-objects.html

[9] Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. – М.:Издательский дом «Вильямс», 2003.

[10]http://msdn.microsoft.com/library/rus/default.asp?url=/library/rus/vbcon/html/vbconnestedrelationshipsinxml.asp