Evaluating Relationship Implementations Performance in Object-Relational Databases

María Fernanda Golobisky, Universidad Tecnológica Nacional, Instituto de Desarrollo y Diseño INGAR (CONICET – UTN), Argentina; E-mail: mfgolo@ceride.gov.ar

Aldo Vecchietti, Universidad Tecnológica Nacional, Instituto de Desarrollo y Diseño INGAR (CONICET – UTN), Argentina; E-mail: aldovec@ceride.gov.ar

ABSTRACT

In this work an evaluation of an object-relational schema implementation representing different relationships of an UML class diagram against the relational approach was made. To perform this test we have implemented both object-relational and relational schemas from a UML class diagram in a commercial database leader in the market. The main goal has been to prove the competitiveness of the object-relational technology. The methodology used for this work was to present several schema implementations of association, composition, aggregation and inheritance relationships, propose a set of representative queries to evaluate their behavior, compare the results and make an analysis based on response times. Four alternatives implementations of the schema diagram were made for a composition relationship presented in the proposed UML class diagram. The queries have been executed with no flush to the database buffer pool among runnings to simulate a real situation. In some object-relational queries several built-in functions and operations have been used. As a consequence of this work we are proposing some extensions to the relational schema diagram to add the object-relational alternatives (references, arrays, multisets, etc.) proposed by the SQL:2003 standard.

Keywords: Performance test. Object-relational schema. SQL:2003. Array. Multiset. Scoped references.

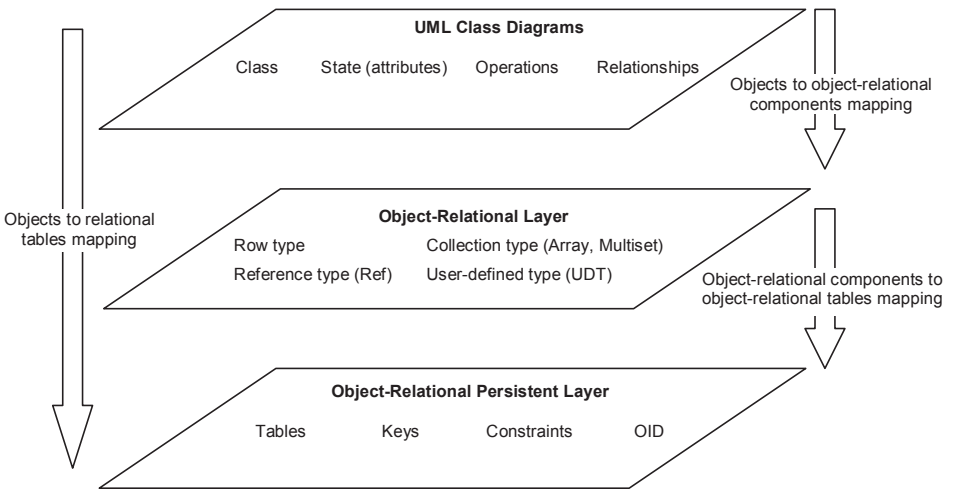
INTRODUCTION

Object-relational database management systems (ORDBMS) based on the SQL:2003 standard offer several new capabilities to implement inheritance, association, aggregation and composition relationships among objects, comparing to the relational approach based on the SQL'92 standard. These novel capabilities are based on the use of user-defined types (UDTs), references and collections. A reference is a logical pointer to a row object that is constructed from its object identifier (OID). In the object-relational (O-R) approach association and aggregation relationships can be implemented by means of single references or collection of references, depending on the relationship cardinality. Composition relationship which is a stronger whole-part relationship than aggregation can be implemented by including a single object or a collection of them into other objects, again depending on the relationship cardinality. Collections can be implemented by two different structures: array and multiset; the main difference between them is the prediction of a given maximum size (the array) or not (multiset).

In relational database management systems (RDBMS) relationships are implemented via tables, foreign and primary keys. Depending on the relationship degree and cardinality a join table is defined in order to hold it. A join table must contain at least a foreign key column for each primary key of the entities participating in the association.

In this work we evaluate the implementation of inheritance, association, aggregation and composition relationships over Oracle 10g to prove the competitiveness of the object-relational technology. We have used this ORDBMS for both object-relational and relational implementations. The reason for choosing Oracle 10g is because it is leader in the database market and includes many of the SQL:2003 features. To perform the implementation evaluations several queries have been selected considering the use of special built-in functions applied to references (REF, DEREF) and collections (TABLE). Those queries were executed and compared to their relational equivalent, which take the form of join operations.

Figure 1. Layers involved in mapping objects into ORDBMS



The results obtained in terms of the elapsed time and execution plans proposed by the optimizer are given.

Two important works are found in the literature about the study of the ORDBMS performance [1, 5]. They were done several years ago, when the O-R technology did not offer the nowadays features. At that moment arrays and multisets were not implemented and scoped references were not supported. We used them as a reference for this paper. Furthermore, [6] was taking into account in our research since it refers to the benefits and contributions of the O-R technology in the software development process and [7] where some concepts about O-R mappings are proposed.

MAPPING LAYERS OF ORDBMS

In [4] we have defined three layers involved in the transformation of UML class diagrams into ORDBMS persistent objects. The first one corresponds to the UML class diagram, the second is the object-relational layer composed of the O-R elements proposed by the SQL:2003 standard [8] -UDTs, arrays, multisets, references, row types-, and the third is the object-relational persistent layer composed of typed tables which are defined from the elements of the second layer containing keys, constraints and OIDs, among other things. Unlike the relational model the additional layer of the object-relational model adds a greater complexity.

The layers involved in the transformations and the elements composing them are presented in Figure 1. It shows that the relational transformations complying with SQL'92 standard are made in one step from UML to relational tables; while O-R transformations take two steps from UML to object-relational components and from the latter to persistent object tables.

DATA MODEL EXAMPLE

We have used a book case model of a purchase order administration in a business company whose UML class diagram for the schema implementations is shown in Figure 2. This model contains many of the relationship types needed to perform the evaluation. It should be noted that no aggregation relationships are presented, this gap was overcame by implementing the composition relationship in a "weak" manner treating it as an aggregation as will be shown later in this paper.

The UML class diagram was translated into an object-relational schema compliance with the SQL:2003 standard and into a relational schema designed under the SQL'92 standard and following the normalization rules. This was done in order to compare the performance of both technologies.

RELATIONAL SCHEMA DEFINITION

The UML class diagram mapping into a relational schema is based on the definitions made on [3].

For the inheritance hierarchy of classes, three ways are presented in the literature [2, 3]: flat, vertical and horizontal. We have implemented the three methods but in this paper it is only shown the flat model by creating one single table for all classes (super and subtypes) in the hierarchy. In the hierarchy it is assumed that Person and Company sets are disjoints and the three object types (customers, persons and companies) must be represented, then in the table where those attributes not corresponding to the type stored in a row, contains NULL values.

Figure 2. Class diagram for a purchase order application

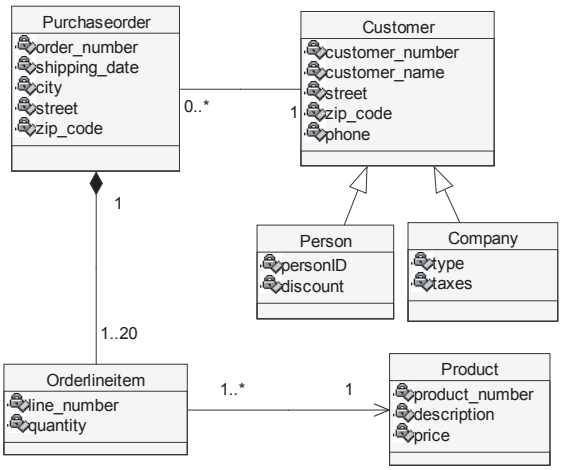


Table 1. Object-relational mapping layers

UML layer components	Object-Relational layer components	Persistent layer components
Customer class	Customer UDT	Customer type table with substitutability property
Person class	Person UDT under Customer	
Company class	Company UDT under Customer	
Purchaseorder class	Purchaseorder UDT	Purchaseorder type table
Orderlineitem class	Orderlineitem UDT	
Product class	Product UDT	Product type table
Customer - Purchaseorder association	Purchaseorder reference multiset (attribute of Customer UDT) Customer reference (attribute of Purchaseorder UDT)	
Purchaseorder - Orderlineitem composition	1. Orderlineitem object array (attribute of Purchaseorder UDT)	
	2. Orderlineitem reference array (attribute of Purchaseorder UDT)	Orderlineitem type table
	3. Orderlineitem reference multiset (attribute of Purchaseorder UDT)	Orderlineitem type table
	4. Orderlineitem object multiset (attribute of Purchaseorder UDT)	
Orderlineitem - Product association	Product reference (attribute of Orderlineitem UDT)	

Association, aggregation and composition relationships are implemented by means of primary and foreign keys.

OBJECT-RELATIONAL SCHEMA DEFINITION

The O-R schema is generated by using references, arrays and multisets and/or a combination of them according to the definitions made in [4].

In Table 1 we present the elements composing the three layers involved in the O-R schema definition. Observe that Purchaseorder-Orderlineitem composition relationship has been implemented in four different ways:

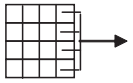
- The first one (1.) is by defining an Orderlineitem type array of dimension 20 in Purchaseorder type table, this is the most natural implementation according to the relationship defined in the UML class diagram. We included the objects of the "part" into the "whole" due to it's a strong relationship where the part life depends on the whole life. We used an Orderlineitem type array into Purchaseorder type table because the multiplicity of the part is well known having a maximal number of 20.
- The second one (2.) is by defining an array of references to Orderlineitem objects in Purchaseorder type table, implemented by the orderline_va attribute. This implementation was made in order to use references within the composition relationship so that it can be treated like an aggregation relationship. It is important to note that if the "whole" is deleted some procedure to eliminate the "parts" must be implemented in order to maintain the integrity of the references. This is not a natural implementation of a composition relationship, it is done in this case in order to evaluate this relationship type. Although for some cases and depending on the nature of the relationship this can be an alternative for a composition.
- The third (3.) and fourth (4.) alternative implementations include a multiset of references and a multiset of objects respectively. The difference between these two and the previous two is that for multiset it is not known the maximum size of the collection. The considerations made about using references or objects are the same than the previous paragraphs.

The relationship between Orderlineitem and Product is an unidirectional association, so we have included a reference to Product as an attribute in Orderlineitem UDT.

Observe that the persistent layer is composed of fewer elements than the O-R layer, depending on the way the composition is implemented three or four tables are defined.

Considering that there are no symbols proposed to represent the O-R elements in a database schema diagram we introduce a graphical notation for this purpose which is shown in Table 2.

Table 2. Object-relational extensions to the relational schema diagram

Graph	Element
	Reference (single arrow)
	Array of references (double arrow)
	Multiset of references (quadruple arrow)
	Object array
	Object multiset
	Object array containing references to other object
	Object multiset containing references to other object

According to the graphical elements proposed, the resultant object-relational schema diagram corresponding to the first implementation of the composition relationship is shown in Figure 3. The other schemas of the remainder implementations of the composition relationship are shown in Figures 4 to 6.

In Fig. 3, the Customer class has a multiset of references to Purchaseorder class. The Purchaseorder class has a single reference to the Customer class and an array of Orderlineitem objects containing a reference to Product class.

Figure 4 shows that the Customer2 class has a multiset of references to Purchaseorder2 class. The Purchaseorder2 class has a single reference to the Customer2 class and an array of references to the Orderlineitem2 class. Orderlineitem2 class contains a single reference to Product2 class.

In Fig. 5, the Customer3 class has a multiset of references to Purchaseorder3 class. The Purchaseorder3 class has a single reference to the Customer3 class and a multiset of references to the Orderlineitem3 class. Orderlineitem3 class contains a single reference to Product3 class.

Figure 6 shows that the Customer4 class has a multiset of references to Purchaseorder4 class. The Purchaseorder4 class has a single reference to the Customer4

Figure 3. Schema diagram for the object-relational implementation. Composition relationship implemented using an array of objects

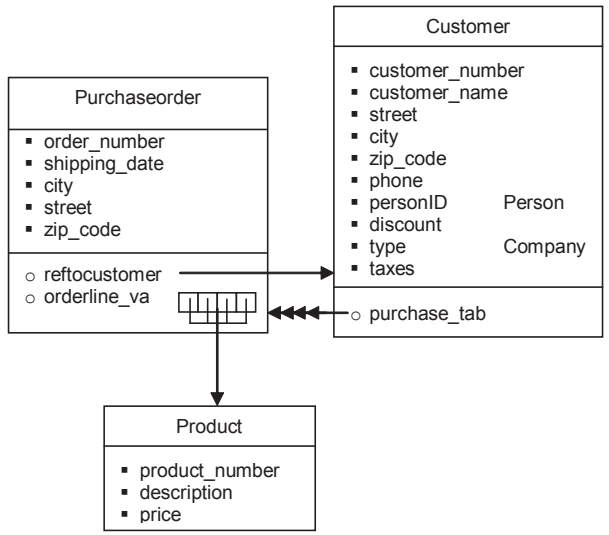


Figure 4. Composition relationship implemented by means of an array of references

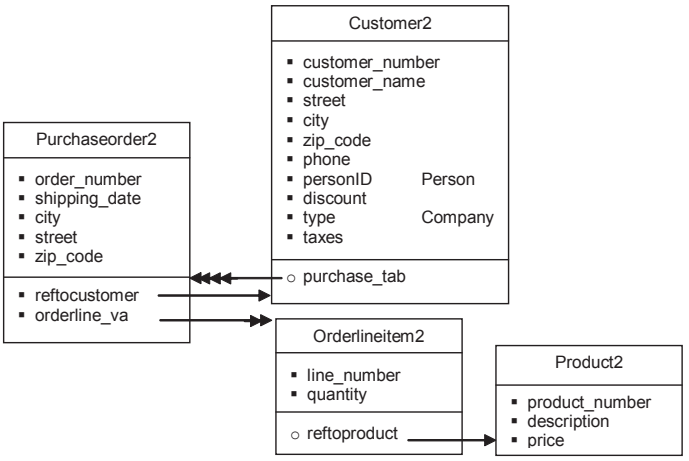


Figure 5. Composition relationship implemented by means of a multiset of references

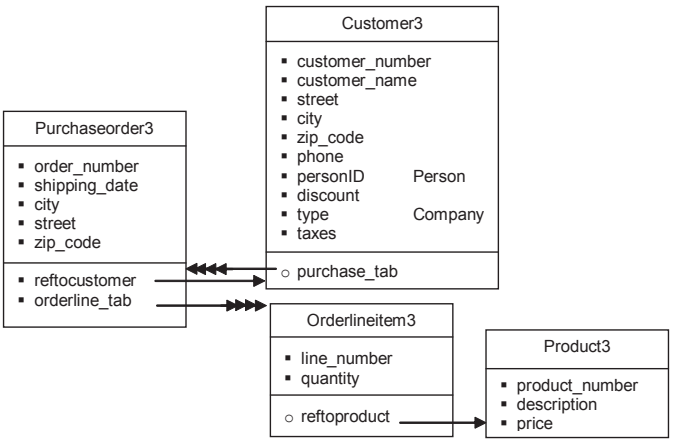
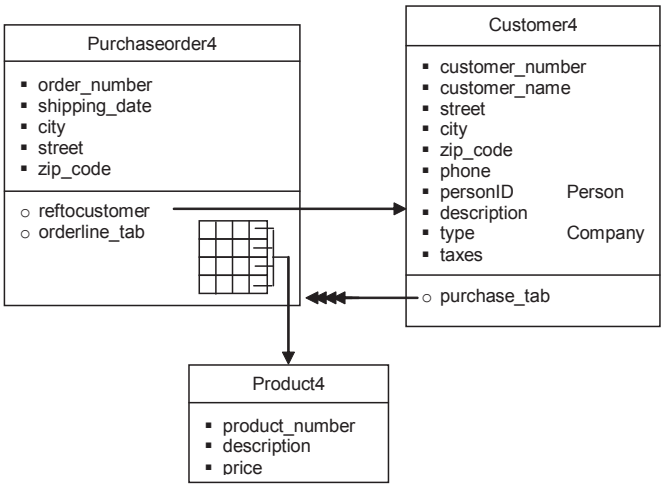


Figure 6. Composition relationship implemented by means of a multiset of objects with references to other object



class and an embedding multiset of Orderlineitem4 objects with single reference to Product4 class.

DATABASE IMPLEMENTATION

In order to make a proper evaluation among the different implementations the tables of the database were populated with thousands of object/tuples using store procedures written in the programming language provided by the ORDBMS containing random values generation functions for the data.

In the object-relational schema every row object in an object table (type table) has an associated OID that uniquely identifies it. The OID allows the corresponding row object to be referenced by other objects. A built-in data type called REF is used for such references. We have used scoped REF to constrain that only references to a specified object table can be implemented, because they are stored more efficiently than unscoped REFs. In the relational schema every row in a table has a primary key that uniquely identifies it allowing table joins.

The number of generated instances of each class is shown in Table 3.

Table 3. Number of objects of each class in the data model

Class	Number of instances
Company	451
Customer	1,000
Orderlineitem	63,578
Purchaseorder	10,000
Person	320
Product	10,000

EVALUATION TEST BETWEEN THE SCHEMAS

We have defined several relational and object-relational queries to compare the performance of the schemas proposed. As can be seen the selected queries explore the use of collections (array and/or multiset) of objects and references, single references and inheritance hierarchy. These queries have been selected due to collections and references make the difference between the O-R approach and the relational one, and the reason of the performance comparison made.

We have executed each one 10 times in different moments and we have calculated the average elapsed time for them. We decided not to flush the database buffer pool among runnings because in real life users execute several applications at the same time all of them consuming system resources. The hardware used for the implementation and testing is an Intel Pentium IV CPU 3.00 GHz, with 1 GB of main memory, running the Microsoft Windows XP operating system.

The goal of the comparison among the queries is to make a relative evaluation of the proposed implementations and analyze the use of references, arrays and multiset of the object-relational technology against the joins of the relational approach. In this analysis we considered the response times and the execution plans defined by the optimizer.

Query 1. Find the order numbers and the detail of line numbers and quantity ordered.

In this query we are analyzing the behavior of the four implementations of the composition relationship in order to find out the most convenient alternative in terms of the response time.

1.1 Array of objects

```
SELECT p.order_number, o.line_number, o.quantity FROM purchaseorder_t p,
TABLE(p.orderline_va) o;
```

1.2 Array of references

```
SELECT p.order_number, o.column_value.line_number, o.column_value.quantity
FROM purchaseorder2_t p, TABLE(p.orderline_va) o;
```

1.3 Multiset of references

```
SELECT p.order_number, o.column_value.line_number, o.column_value.quantity
FROM purchaseorder3_t p, TABLE(p.orderline_tab) o;
```

1.4 Multiset of objects

```
SELECT p.order_number, o.line_number, o.quantity FROM purchaseorder4_t p,
TABLE(p.orderline_tab) o;
```

1.5 Relational model

```
SELECT p.order_number, o.line_number, o.quantity FROM purchaseorder p,
orderlineitem o
WHERE p.order_number = o.order_number;
```

The results obtained are shown in Table 4.

Table 4 shows that the use of a multiset of objects (query 1.4) has the same response time than the join (query 1.5) proposed for the relational query; in this case both

Table 4. Results of the query 1

Query	Rows selected	Response time (hh:mm:ss)
1.1	63578	00:00:03
1.2		00:00:07
1.3		00:00:07
1.4		00:00:02
1.5		00:00:02

technologies are competitive. The use of collections of references (queries 1.2 and 1.3) have the worst response time, the explanation for this behavior relays on the size of the references, which are more than 40 bytes long requiring an extra time to solve it. Looking at the execution plans the use of arrays requires a collection iterator operation (PICKLER FETCH) that is not present for multisets, it is traduced in a higher cost and number of bytes involved.

Query 2. Find the customers, their order numbers together the line numbers and quantity ordered.

This query is similar to query 1 but in this case we are starting from the customer typed table adding an extra multiset of references. When the composition is implemented like an aggregation two hop references are employed. For this case we are using the traversal of two collections.

2.1 Multiset of references + Array of objects

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.line_number, o.quantity
FROM customer_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.orderline_va) o;
```

2.2 Multiset of references + Array of references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.column_value.line_number, o.column_value.quantity FROM customer2_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.orderline_va) o;
```

2.3 Multiset of references + Multiset of references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.column_value.line_number, o.column_value.quantity FROM customer3_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.orderline_tab) o;
```

2.4 Multiset of references + Multiset of objects

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.line_number, o.quantity
FROM customer4_t c, TABLE(c.purchase_tab) p, TABLE(p. column_value.orderline_tab) o;
```

2.5 Relational model

```
SELECT c.customer_number, c.customer_name, p.order_number, o.line_number, o.quantity
FROM customer_plano c, purchaseorder p, orderlineitem o
WHERE c.customer_number = p.customer_number AND p.order_number = o.order_number;
```

The results obtained are shown in Table 5.

Looking at the results shown in Table 5 the relational approach is more efficient in terms of response time than the O-R technology. The cause is that the multiset of references implementing the association between Customer and Purchaseorder add an extra time for solving the query as was mentioned before. The multisets used for the composition relationship consumes much more time than the arrays. Looking at the execution plans the operations, cost, number of bytes and the other variables of the plans gave us no clue about this behavior. What it is clear in this case is that when two collections are involved in a query is better to implement it by means of arrays if possible.

Arrays perform much better than multisets in the case that the entire collection is manipulated as a single unit in the application because the array is stored in packed form and do not require joins to retrieve the data, unlike multiset, using Oracle 10g.

Query 3. Find the products ordered by the customers.

In this query we are using two collections plus single references to retrieve products information, that is to say it were employed three hop references.

3.1 Multiset of references + Array of objects + Single references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.reftoproduct.product_number
FROM customer_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.orderline_va) o;
```

3.2 Multiset of references + Array of references + Single references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.column_value.reftoproduct.product_number
FROM customer2_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.orderline_va) o;
```

Table 5. Results of the query 2

Query	Rows selected	Response time (hh:mm:ss)
2.1	63578	00:00:05
2.2		00:00:10
2.3		00:01:14
2.4		00:01:14
2.5		00:00:03

3.3 Multiset of references + Multiset of references + Single references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.column_value.reftoproduct.description
FROM customer3_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.order-line_tab) o;
```

3.4 Multiset of references + Multiset of objects + Single references

```
SELECT c.customer_number, c.customer_name, p.column_value.order_number, o.reftoproduct.product_number
FROM customer4_t c, TABLE(c.purchase_tab) p, TABLE(p.column_value.order-line_tab) o;
```

3.5 Relational model

```
SELECT c.customer_number, c.customer_name, p.order_number, pr.description
FROM customer_plano c, purchaseorder p, orderlineitem o, product pr
WHERE c.customer_number = p.customer_number AND p.order_number = o.order_number
AND o.product_number = pr.product_number;
```

The results obtained are shown in Table 6.

As it can be noted queries with the third added hop have the same performance than query 2.

Query 4. Find customer information for all customers of person type.

In this query we are using the inheritance hierarchy of Customer obtaining the supertype information of person subtype. The substitutability property allows the storage of any subtype in the supertype table.

4.1 Object-relational

```
SELECT p.customer_number, p.customer_name, p.street, p.city
FROM customer_t p WHERE VALUE(p) IS OF (person_ob);
```

4.2 Relational

```
SELECT customer_number, customer_name, street, city
FROM customer_plano WHERE type = 'P';
```

The results obtained are shown in Table 7.

The response time for both queries is similar; due to the few rows involved in the query the time is very low. Analyzing the execution plans both are very similar, and no differences can be found. The advantage of the O-R approach is that the model evolution can be easily implemented, subtyes can be added to the hierarchy and can be stored in the supertype table.

Query 5. Find customer and person information for all customers of person type.

The difference between this query and query 3 is that in this case we are treating supertype instances as subtype instances.

5.1 Object-relational

```
SELECT p.customer_number, p.customer_name, p.street, p.city, TREAT(VALUE(p) AS person_ob).person_id, TREAT (VALUE(p) AS person_ob).discount FROM customer_t p WHERE VALUE(P) IS OF (person_ob);
```

5.2 Relational

```
SELECT customer_number, customer_name, street, city, person_id, discount
FROM customer_plano WHERE type = 'P';
```

The results obtained are shown in Table 8.

The result analysis made for query 4 is the same for this one. Due to in both queries (4 and 5) the flat model is used, so the optimizer makes a sweeping of the entire tables.

Table 6. Results of the query 3

Query	Rows selected	Response time (hh:mm:ss)
3.1	63578	00:00:10
3.2		00:00:12
3.3		00:01:13
3.4		00:01:11
3.5		00:00:07

Table 7. Results of the query 4

Query	Rows selected	Response time (hh:mm:ss)
4.1	320	Less than 1 second
4.2		Less than 1 second

Table 8. Results of the query 5

Query	Rows selected	Response time (hh:mm:ss)
5.1	320	Less than 1 second
5.2		Less than 1 second

CONCLUSIONS

In this work we have evaluated the implementations of relationships of different type into an object-relational schema and have made the comparison of them against a relational approach. Oracle 10g was used for the implementations. We started with a UML class diagram of a book case example. In order to define the O-R schema we have transformed the class diagram into O-R elements of an intermediate layer and then they were transformed into persistent typed tables. These tasks are more complex than the relational model which involves a more direct mapping. Several O-R schemas have been defined involving different alternatives for the implementation of composition relationship. Arrays and multiset of references and objects have been used for this purpose. We have proposed graph elements to support object-relational extensions for the relational schema diagram. Those elements are very useful for database developers since the complexity of the object-relational model can be represented graphically facilitating their interpretation.

The evaluation of the O-R implementations against the relational approach has been driven by a set of queries, their response time and execution plans. As a result of this study, comparing the use of arrays, multiset, objects and references, for the implementation of composition and aggregation relationships, no general conclusion of which one is better can be made. Each case can be analyzed according to the business rule to be implemented, several alternatives for them are open, and it is worthy to make some evaluations before making a final decision. The performance of the inheritance hierarchy is the same in both technologies analyzed, having the O-R technology more flexibility for type evolution.

Even though the relational technology threw the best results, the object-relational technology had good ones in some cases, not so far the relational behavior. In the future work our plan is to implement the mappings in an OO language such that it is possible to evaluate if the O-R technology can reduce the impedance mismatch existing between the OO programming languages and the relational approach. A priori, the expectation is to get certain advantages from the O-R technology regarding to this issue.

REFERENCES

1. Carey, M., DeWitt, D., Naughton, J., Asgarian, Brown, P., M., Gehrke; J. and Shah, D.: The Bucky Object-Relational Benchmark. In Proc. of ACM SIGMOD International Conference on Management of Data, pp. 135-146 (1997).
2. Carey, M., Chamberlin, D., Narayanan, S., Vance, B., Doole, D., Rielau, S., Swagerman, R. and Mattos, N.: O-O, What Have They Done to DB2?. In Proc. of 25th International Conference on Very Large Data Bases. Edinburgh, Scotland (1999).
3. Elmasri, R. and Navathe S.: Fundamentals of Database Systems, Third Edition. Addison Wesley. (2000).
4. Golobisky, M.F. and Vecchietti, A.: Mapping UML Class Diagrams into Object-Relational Schemas. In Proc. of the Argentinian Symposium of Software Engineering (ASSE 2005). ISSN: 1666-1087. Pág. 65-79. 34 JAIIO, Rosario, Santa Fe, Argentina (2005).
5. Lee, S.H, Kim, S. J. and Kim, W.: The BORD Benchmark for Object-Relational Databases. In Proc. of the 11th International Conference on Database and Expert Systems Applications (2000).
6. Mahnke, W.: Towards a modular, object-relational schema design. In Proc. of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'2002) (2002).
7. Marcos, E., Vela, B., Cavero, J.M. and Cáceres, P.: Aggregation and Composition in Object-Relational Database Design. In Proc. of the Fifth East-European Conference on Advances in Databases and Information Sytems, Vilnius Lithuania (2001)
8. Melton, J. ISO/IEC 9075-2:2003 (SQL/Foundation), ISO standard. (2003).