International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 3, June 2012 www.ijcsn.org ISSN 2277-5420

Use of Inheritance Feature in Relational Database Development

[1]A.V.Saurkar, [2]Prof. A.R. Itkikar

[1,2]Department of Computer Science & Engineering
Sipna's COET, SGBAU, Amravati (MH), India

Abstract

Currently object-relational database technology is setting the direction for the future of data management. The appearance of object-relational database (ORDB) technology into the business database market caused the database user's attention in search of how to utilize its object-oriented features in the database development. Although the ORDB technology is already available for use in all DBMS products, its industrial acceptance rate is not relatively high. The goal of the paper is to present how to use the object-relational database management system (ORDBMS) to overcome relational database problems and improve database performance in the database development using features of ORDB technology. This paper introduces how to use the specific ORDB technology to provide the solutions for data complexity problems with specific ORDBMS techniques: object inheritance.

Keywords-- OBJECT-RELATIONAL DATABASE, DATABASE CURRICULUM, ORACLE DATABASE, NORMALIZATION.

1. Introduction

Object-relational DBMS (ORDBMS) are the next great wave. They have been proposed for all applications that need both complex queries and complex data types. An object-relational database or object-relational database management system (ORDBMS), is a database management system similar to a relational database, but with an object-oriented database models like objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, it supports extension of the data model with custom data-types and methods.

Object-relational database technology has developed as a way of enhancing object-oriented features in relational database management systems (RDBMSs).

By extending traditional RDBMS with object-oriented features developer develop ORDBMS which actually enhances object-oriented technology into the relational database management system (RDBMS) [2]. As an evolutionary technology, ORDBMS allows users to take advantages of reusing features of object-oriented technology, to map objects into relations and maintaining a consistent data structure in the present RDBMS.

The achievement of relational DBMSs cannot be ignored, but it has more difficulty when confronted with the "complex data", found in advanced application areas such as hardware and software design, science and medicine etc. To overcome the pitfalls of RDBMS , Oracle, IBM and Microsoft have moved to introduce object-oriented database features into their relational DBMSs under the name of object-relational DBMSs. Even if the ORDB technology is already available in all the major DBMS products, its industrial acceptance rate is very low. One of the major drawback of ORDBMS is that its complexity, which results in the loss of the simplicity and transparency of the relational database model. It is challenging work for industrial application developers who are very familiar with traditional relational database background to adopt the developing ORDB technology.

The purpose of the paper is to support utilization of ORDBMS features in industrial database application and providing proper solutions to the problems in database development. The paper first introduces Background of ORDBMS, and then explains how to use ORDBMS's special functionality: object inheritance to solve data complexity problems.

Inheritance allows for hierarchies in which each child has characteristics of its parent. Each level in the type hierarchy has properties, which can be shared by those beneath it; lower levels can have their own specialized attributes or functions as well.

2. Background of ORDB

The object-relational database technology is initiate in the middle of 1990s after emergence of object-oriented database (OODB). Stonebraker and Moore [10] define their four-quadrant view (two by two matrix).

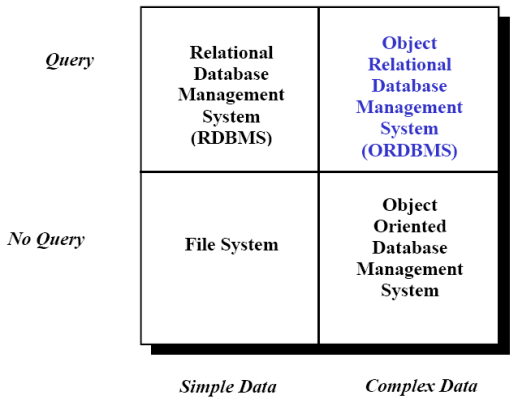

Figure 1: Database classification matrix.

Figure 1 shows database classification matrix. Four quadrant views show the data processing world: relational database (RDBMS), object-relational database (ORDBMS), file processing system, and object-oriented database (OODBMS). The idea behind this model is to point out the kinds of problems which are solved by each of four-quadrants. As will be seen there is no DBMS that solves all the applications. They suggest that there is a natural selection of data manager for each of the four database applications.

In the four-quadrant view of the database world, ORDBMS has been the most suitable DBMS that processes complex data and complex queries.

An object-relational database (ORD), or object-relational database management system (ORDBMS), is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. An object-relational database can be said to provide a middle ground between relational databases and object-oriented databases (OODBMS). Object-relational database management systems grew out of research that occurred in the early 1990s. That research extended existing relational database concepts by adding object concepts.

To define Object-Relational Database Management System (ORDBMS) it is enough to take simple equation:

ORDBMS = ODBMS + RDBMS = (O + R) * DB * MS.

The vendors of relational systems have integrated many object-oriented database features into their DBMS products. As a result, many DBMS products that used to be called "relational" are now called "object-relational" [6].

The gap between OODBMS and RDBMS is bridges by ORDBMS. It allows user to take advantage of OODB'MSs great productivity and complex data type without losing existing investment of OODB in relational data [2].

Actually, an ORDBMS engine supports both relational and object- relational features in an integrated fashion [5]. The original ORDB data model is relational because object data is stored in tables or columns. while assimilating new object-oriented features ORDB designers actually work with well-known tabular structures and data definition languages (DDLs) [8].It is essentially a relational data model with object-oriented extensions. In response to the evolutional change of ORDB technology, SQL:1999 started supporting object-relational data modeling features in database management standardization and SQL:2003 continues this evolution [4]. Currently, all the major database vendors have upgrade their relational database products to object-relational database management systems to imitate the new SQL standards [7] and ready to be used by industrial practitioners.

All the ORDBMSs have the capability to store object data and methods in databases. Many of SQL:2003 standard ORDBMS features appear in Oracle. They are listed as follows [11].

Object Types:
User-defined data types (UDT) or abstract types (ADT) are referred to as object types.
Functions/Methods:
For each object type, the user can define the methods for data access. Methods define the behaviour of data.
Varray:
The varray is a collection type that allows the user to embed homogenous data into an array to form an object in a pre-defined array data type.
Nested table:
A nested table is a collection type that can be stored within another table. With a nested table, a collection of multiple columns from one table can be placed into a single column in another table.
Inheritance:
With Object type inheritance, users can build subtypes in hierarchies of database types in ORDBs.
Object View:
Object view allows users to develop object structures in existing relational tables. The data are really stored in a traditional relational format but object view allows data to be accessed or viewed in an object- oriented way.

3. Details of topic

The beauty of ORDBMSs is reusability and sharing. Reusability mainly comes from storing data and methods together in object types and performing their functionality on the ORDBMS server, rather than have the methods coded separately in each application. Sharing comes from using user-defined standard data types to make the database structure more standardized [1].

3.1 Inheritance for Object Reuse

The main advantages of extending the relational data model come from reuse and sharing. If multiple applications use the same set of database objects, then you have created a de facto standard for the database objects, and these objects can be extended [12]. ORDBMSs allow users to define hierarchies of data types. With this feature, users can build subtypes in hierarchies of database types. If users create standard data types to use for all employees, then all of the employees in the database will use the same internal format.

This reuse and sharing of database object is called Inheritance in Object Oriented DB. Inheritance allows for hierarchies in which each child has characteristics of its parent. Each level in the type hierarchy has properties, which can be shared by those beneath it; lower levels can have their own specialized attributes or functions as well.

Users might want to define a full time employee object type and have that type inherit existing attributes from employee_ty [11]. The full_time_ty type can extend employee_ty with attributes to store the full time employee's salary. The part_time_ty type can extend employee_ty with attributes to store the part-time employee's hourly rates and wages. Inheritance allows for the reuse of the employee_ty object data type. The details are illustrated in the following class diagram:

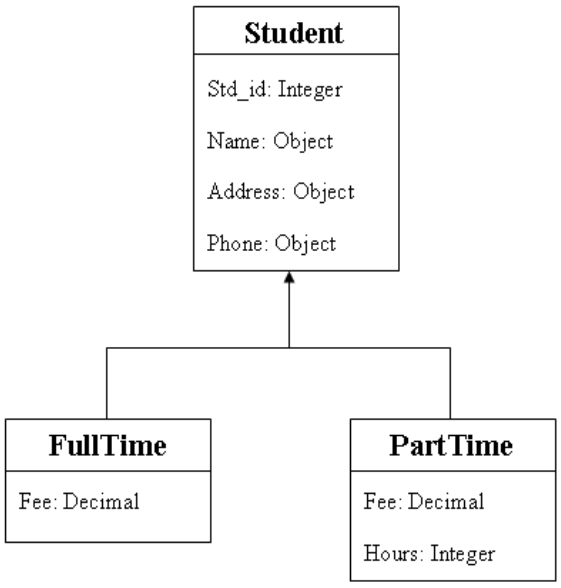

Figure 2: Employee relationship

Object type inheritance is one of new features of Oracle 9i. For employee_ty to be inherited from, it must be defined using the NOT FINAL clause because the default is FINAL, meaning that object type cannot be inherited. Oracle 9i can also mark an object type as NOT INSTANTIABLE; this prevents objects of that type derived. Users can mark an object type as NOT INSTANTIABLE when they use the type only as part of another type or as a super_type with NOT FINAL. The following example marks address type as NOT INSTANTIABLE:

```
CREATE TYPE Student_ty AS OBJECT (
std_id NUMBER,
name name_ty,
dob DATE,
phone varray_phone_ty,
address address_ty
) NOT FINAL NOT INSTANTIABLE;
```

To define a new subtype full_time_ty inheriting attributes and methods from existing types, users need to use the UNDER clause. Users can then use full_time_ty to define column objects or table objects. For example, the following statement creates an object table named FullTimeEmp.

```
CREATE TYPE full_time_Stu_ty UNDER student_ty ( fee NUMBER(8,2));
CREATE TABLE FullTimeStudent of full_time_Stu_ty;
```

The preceding statement creates full_time_Stu_ty as a subtype of student_ty. As a subtype of student_ty, full_time_stu_ty inherits all the attributes declared in student_ty and any methods declared in student_ty. The statement that defines full_time_stu_ty specializes student_ty by adding a new attribute "fee". New attributes declared in a subtype must have names that are different from the names of any attributes or methods declared in any of its supertypes, higher up in its type hierarchy. The following example inserts row into the FullTimeStudent table. Notice that the additional salary attribute is supplied :

```
INSERT INTO FullTimeStudent VALUES
(10, name_ty('Milind', 'Kevat', 'K'), '12-MAY-1991',
varray_phone_ty('(626)123-5678', '(323)343-2983',
'(626)789-1234'),
Address_ty ('3 M.G. Road', 'Eden Gardan', 'Colkata', 32145),
45000.00);

SELECT * FROM FullTimeStudent;
```

EMP_ID	NAME(F_NAME, L_NAME, INITIALS)	DOB	PHONE	ADDRESS (STREET, CITY,STATE, ZIP)	SALARY
10	name_ty('Milind', 'Kevat', 'K')	12 MAY -1991	varray_phone_ty ('(626)123-5678', '(323)343-2983', '(626)789-1234')	Address_ty ('3 M.G. Road', 'Eden Gardan', 'Colkata', 32145)	45000.00

Table 1: Output of 'SELECT * FROM FullTimeStudent;' query

3.2 Another Example of Inheritance using SQL:

When you create a user-defined structured type, the syntax is very similar to that of creating a table: the name of the type is specified, along with its attribute names and the data types of the attributes. If you create subtypes under a structured type, those subtypes will automatically inherit the attributes of the type above it in the hierarchy (the super type).

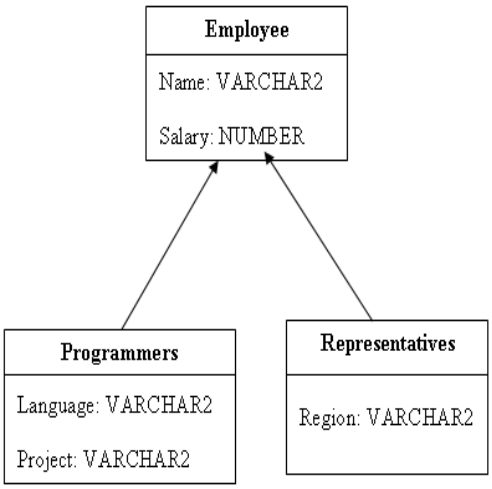

Figure 3: Programmer- representative relationship

Hierarchy within structured complex data offers an additional property, type inheritance. That is, a structured type can have subtypes that reuse all of its attributes and contain additional attributes specific to the subtype. Below are the data suggested to consider.

Table 2: Employees

Name	Salary
S. Swaminathan	30000.00

Table 3: Programmers

Name	Salary	Language	Project
Vanat Vinayak	45000.00	JAVA	Seestorm

Table 4: Representatives

Name	Salary	Region
Aniket	50000.00	India

The data types make up the hierarchy illustrated here:

```
SQL representation
CREATE TYPE Employee AS OBJECT (
  Name      VARCHAR2(20),
  Salary    NUMBER(6,2)
) NOT FINAL;
CREATE TYPE Programmer UNDER Employee (
  Language   VARCHAR2(12),
  Project    VARCHAR2(30)
);
CREATE TYPE Representative UNDER Employee (
  Region    VARCHAR2(30)
);
CREATE TABLE employees OF Employee;
CREATE TABLE programmers OF Programmer;
CREATE TABLE representatives OF Representative;
INSERT INTO employees
  VALUES (Employee(' S. Swaminathan ', 30000.00));
INSERT INTO programmers
  VALUES (Programmer(' Vanat Vinayak ', 45000.00,
'JAVA', 'Seestorm'));
INSERT INTO representatives
  VALUES (Representative('Aniket', 50000.00, 'India'));
```

The "Programmer" and "Representative" subtypes inherit all the attributes from the "Employee" supertype. A request for "employees" objects of the "Employee" type means also request for objects of subtypes, namely "programmers" and "representatives". For example, the result of the following SQL statement:

```
SELECT e.Name
  FROM employees e;

Would be:
Name
--------------------
S. Swaminathan
Vanat Vinayak
Aniket
```

4. Conclusion

The significance of the paper is to support the utilization of ORDBMS features for object reuse and integration among database practitioners. Use of ORDBMS is to develop applications can implement the reuse of varying user-defined object types, provide an integrated view of data and allow multiple database applications to operate cooperatively. This paper introduces how to use the specific ORDB technology to provide the solutions for data complexity problems with specific ORDBMS techniques: object inheritance.

References

[1] Begg, C., & Connolly, T. (2010). Database systems: A practical approach to design, implementation, and management, 5th Ed. Addison Wesley.

[2] Connolly, T. and Begg, C. (2006). Database systems: A practical approach to design, implementation, and management, 4th Ed. Addison Wesley.

[3] Elmasri, R. & Navathe, S. (2011). Fundamentals of Database Systems, 6th Edition, Addison Wesley.

[4] Fortier, P. (1999). SQL3: Implementing the Object-Relational Database, Osborne McGraw-Hill.

[5] Frank, M. (1995). Object-relational Hybrids, DBMS, 8/8, 46-56.

[6] Garcia-Molina, H., Ullman, J. & Widom, J. 2003. Database Systems: The Complete Book, Prentice Hall, Upper Saddle River.

[7] Hoffer, J., Prescott, M., & Topi, H., 2009 Modern Database Management, 9th Edition, Pearson Prentice Hall.

[8] Krishnamurthy, Banerjee and Nori, 1999. Bringing object-relational technology to the mainstream, Proceedings of the ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems, Philadelphia, PA .

[9] Loney, K. & Koch, G. (2002) Oracle 9i: The complete reference, Oracle Press/McGraw-Hill/Osborne.

[10] Stonebraker M. and Moore, D. 1996. Object-relational DBMSs: the Next Great Wave. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

[11] Ming Wang, California State University, ming.wang@calstatela.edu using object-relational database technology To solve problems in database development, Issues in Information Systems, Volume XI, No. 1, 2010. Price, J. 2002. Oracle9i, JDBC Programming, Oracle Press/McGraw-Hill/Osborne