Object Relational Database Management Systems

Mr.Harsha Wijayawardhana

Definition of ORDBMS

- Object Relational Databases Management systems have emerged to enhance the capabilities of relational database management systems
- the idea that object-oriented database concepts can be superimposed on relational databases, is more commonly encountered in available products

Reasons for the evolution of ORDBMS

- Today's applications deal with graphics, images, weather forecasting, biological gnome data etc
- Further we have to deal with audio and video streaming data

Inadequacy of Relational model

- The handling above challenges was a problem since many of these applications handle the above as objects.
- Some of these applications were developed with Object Oriented Languages

Object Database Management System

- An object database management system (ODBMS, also referred to as object-oriented database management system or OODBMS), is a database management system (DBMS) that supports the modeling and creation of data as objects. This includes some kind of support for classes of objects and the inheritance of class properties and methods by subclasses and their objects.
- (Fundamentals of Database Systems page 359)

History of ODMS

- **Early 1980s - Orion Research Project at MCC**

  Won Kim at MCC (Microelectronics and Computer Technology Corporation) in Austin, Texas, begins a research project on ORION. Two products will later trace their history to ORION: ITASCA (no longer around) and Versant.

Commercial Products

- Late 1980

A Lisp-based system, Graphael, appears from the French nuclear regulatory efforts. Eventually, Graphael goes through a re-write and becomes Matisse.

Servo-Logic begins work on GemStone. Servo-Logic is now GemStone Systems.

Commercial Products cont …

- Start of O2 development at INRIA (France). The founder of O2 is Francois Bencilhon, also from MCC.

  Tom Atwood at Ontologic produced Vbase, which supports the proprietary language COP (for C Object Processor). COP is eventually eclipsed by C++, Ontologic becomes ONTOS, and the database is rewritten to support C++. Tom left Ontologic in the late 1980s and founded Object Design (now part of Progres Software) with ObjectStore (based on C++).

Commercial Products cont …

- **1991 - ODMG**

  Rick Cattell (SunSoft) initiates the ODMG with 5 major OODBMS vendors. The first standard, ODMG 1.0, was released in 1993. Throughout the 1990s, the ODMG works with the X3H2 (SQL) committee on a common query language. Though no specific goal is achieved, the efforts heavily influence the ODMG OQL (object query language) and, to a lesser extent, SQL:1999.

Commercial Products cont …

- **2001 - Final ODMG 3.0 standards released.**
  Final ODMG 3.0 standards is released. Shortly thereafter, the ODMG submits the ODMG Java Binding to the Java Community Process as a basis for the Java Data Objects (JDO) Specification. Afterwards, the ODMG disbands.

Review of Object Oriented concepts

- Origins of OO concepts can be traced to Object Oriented Programming languages
- Today Object Oriented concepts are applied to many areas: databases, Software Engineering, Knowledge bases.
- OOPLs have its roots to SIMULA which was proposed in 1960

Review cont …

- In SIMULA class group together the internal data structures in an object in a class. Researchers later proposed the concept of abstract data type which hides the internal data structures and specifies all possible external operations can be applied that can be applied to an object, leading to the concept encapsulation.

Review cont …

- SMALL TALK developed at XEROX PARC (Palo Alto Research Centre, California) in 1970 was one of the first languages to incorporate additional OO concepts
- Object : State (value) and behavior (operations)
- Another key concept in OO systems is that of type and class hierarchies and inheritance :Multiple Inheritance or Selective inheritance (pp 362 -367 and 380-381)

Review cont …

- Other OO concepts are:
  - Polymorphism which refers sometime as Operator overloading (This gives rise to early binding for strong typed and late binding weak typed)

The Object Model

- The basic modeling primitives are the object and the literal (constant). Each object has a unique identifier. A literal has no identifier.

- Objects and literals can be categorized by their types. All elements of a given type have a common range of states (i.e., the same set of properties) and common behavior (i.e., the same set of defined operations). An object is sometimes referred to as an instance of its type.

- The state of an object is defined by the values it carries for a set of properties. These properties can be attributes of the object itself or relationships between the object and one or more other objects. Typically the values of an object's properties can change over time.

Object Model cont …

- The behavior of an object is defined by the set of operations that can be executed on or by the object. Operations may have a list of input and output parameters, each with a specified type. Each operation may also return a typed result.

- A database stores objects, enabling them to be shared by multiple users and applications. A database is based on a schema that is defined in ODL and contains instances of the types defined by its schema.

Object specifications languages

- The primary objective of these languages is to facilitate the portability of databases across ODMG compliant implementations. These languages also provide a step toward the interoperability of ODBMSs from multiple vendors.

Object specification language cont ...

- Object Definition Language (ODL) and Object Interchange Format (OIF).

Object Definition Language (ODL)

- ODL should support all semantic constructs of the ODMG Object Model.
- ODL should not be a full programming language, but rather a definition language for object specifications.
- ODL should be programming-language independent.

- ODL should be compatible with the OMG's Interface Definition Language (IDL).
- ODL should be extensible, not only for future functionality, but also for physical optimizations.
- ODL should be practical, providing value to application developers, while being supportable by the ODBMS vendors within a relatively short time frame after publication of the specification.
- (for further reading on Object Definition Language Fundamentals of Database Systems pp 399-404)
- (www.odmg.org)

Object Query Language (OQL)

- OQL relies on the ODMG object model.
- OQL is very close to SQL 92. Extensions concern object-oriented notions, like complex objects, object identity, path expressions, polymorphism, operation invocation, late binding.
- OQL provides high-level primitives to deal with sets of objects but is not restricted to this collection construct. It also provides primitives to deal with structures, lists, arrays, and treats such constructs with the same efficiency.

OQL cont …

- OQL is a functional language where operators can freely be composed, as long as the operands respect the type system. This is a consequence of the fact that the result of any query has a type which belongs to the ODMG type model, and thus can be queried again .

- OQL is not computationally complete. It is a simple-to-use query language which provides easy access to an ODBMS.

- Based on the same type system, OQL can be invoked from within programming languages for which an ODMG binding is defined. Conversely, OQL can invoke operations programmed in these languages.

OQL cont …

- OQL does not provide explicit update operators but rather invokes operations defined on objects for that purpose, and thus does not breach the semantics of an ODBMS which, by definition, is managed by the "methods" defined on the objects.
- OQL provides declarative access to objects. Thus OQL queries can be easily optimized by virtue of this declarative nature.
- The formal semantics of OQL can easily be defined.

Simple queries using OQL (syntax for O_2)

- Basic syntax is select …from…where …

```
SELECT <list of values>
FROM <list of collections and variable assignments>
WHERE <condition>
```

```
SELECT SName: p.name
FROM p in People
WHERE p.age > 26
```

OQL cont …

- Dot notation and Path Expressions
  - Let variables t and ta range over objects in extents (persistent names) of Tutors and TAs (i.e., range over objects in sets Tutors and TAs).

    ta.salary -> real
    t.students -> set of tuples of type tuple(name: string, fee: real) representing students
    t.salary -> real
  - Cascade of dots can be used if all names represent objects and not a collection.

OQL cont …

- Find the names of the students of all tutors:

```
SELECT s.name
 FROM Tutors t, t.students s
```

OQL cont …

- Sub queries
  - Give the names of the Tutors which have a salary greater than Rs.3000 and have a student paying more than Rs. 3000:

```
SELECT t.name
 FROM ( SELECT t FROM Tutors t WHERE t.salary >
 300 ) r, r.students s
 WHERE s.fee > 3000
```

(pp 405 – 409)

ORDBMS

- Some of the Object Relational Database Management Systems are:
  - Oracle 8
  - Postgresql
  - Informix Universal Server
- In addition, we will discuss SQL 3 language latest version of SQL language which extends SQL 2 by incorporating object database and other features such as extended data types

ORDMS cont …

- SQL 3
  - SQL3 object facilities primarily involve extensions to SQL's *type* facilities; however, extensions to SQL *table* facilities can also be considered relevant. Additional facilities include control structures to make SQL a computationally complete language for creating, managing, and querying persistent object-like data structures.

SQL 3 cont …

- The parts of SQL3 that provide the primary basis for supporting object-oriented structures are:
  - *user-defined types* (*ADT*s, *named row types*, and *distinct types*)
  - type constructors for *row types* and *reference types*
  - type constructors for *collection types* (sets, lists, and multi sets)
  - *user-defined functions and procedures*
  - support for *large objects* (BLOBs and CLOBs)

SQL 3 cont …

- A *row type* is a sequence of field name/data type pairs resembling a table definition. Two rows are type-equivalent if both have the same number of fields and every pair of fields in the same position have compatible types. The row type provides a data type that can represent the types of rows in tables, so that complete rows can be stored.

SQL 3 cont …

- Operations that may be invoked in SQL include defined operations on tables (SELECT, INSERT, UPDATE, DELETE), the implicitly defined functions defined for ADT attributes, and routines either explicitly associated with ADTs or defined separately.
- Routines associated with ADTs are FUNCTION definitions for type-specific user-defined behavior. The FUNCTION definitions specify the operations on the ADT and return a single value of a defined data type. Functions may either be SQL functions, completely defined in an SQL schema definition, or external functions, defined in standard programming languages.

SQL 3 cont …

- QL functions associated with ADTs are invoked using either a functionalnotation or a dot notation (the dot notation, double dot notation, is syntactic sugar for the functional notation). For example:

BEGIN DECLARE r real_estate ... SET r..area = 2540; /* same as area(r,2540) SET ... = r..area; /* same as area(r) ... SET ... = r..location..state; /* same as state(location(r)) SET r..location..city =

'LA'; /* same as city(location(r),'LA') ...

SQL 3 cont …

- Different routines may have the same name.
- This is referred to as *overloading, and* may be required, for example, to allow an ADT subtype to redefine an operation inherited from a super type.

SQL 3 cont …

- SQL3 implements what is sometimes known as a *generalized* object model, meaning that the types of *all* arguments of a routine are taken into consideration when determining what routine to invoke, rather than using only a single type specified in the invocation as, for example, in C++ or Smalltalk

SQL 3 cont …

- Inheritance
  - An ADT can be defined as a subtype of one or more ADTs by defining itas UNDER those ADTs (multiple inheritance is supported). In this case,the ADT is referred to as a *direct subtype* of the ADTs specifiedin the UNDER clause, and these ADTs are *direct supertypes*.

SQL 3 cont …

– A typecan have more than one subtype and more than one supertype. A subtype inherits all the attributes and behavior of its supertypes; additional attributes and behavior can also be defined. An instance of a subtype is consideredan instance of all of its supertypes. An instance of a subtype can be usedwherever an instance of any of its supertypes is expected.

- CREATE TABLE person (name CHAR(20), sex CHAR(1), age INTEGER);
- CREATE TABLE employee UNDER person (salary FLOAT);
- CREATE TABLE customer UNDER person (account INTEGER);

SQL 3 cont …

- The number of other statements which are added:
  - An assignment statement that allows the result of an SQL value expressionto be assigned to a free standing local variable, a column, or an attributeof an ADT.
  - A CALL statement that allows invocation of an SQL procedure.
  - A RETURN statement that allows the result of an SQL value expressionto be returned as the RETURNS value of the SQL function.

- A CASE statement to allow selection of an execution path based on alternative choices.
- An IF statement with THEN, ELSE, and ELSEIF alternatives to allow selection of an execution path based on the truth value of one or more conditions.
- Statements for LOOP, WHILE, and REPEAT to allow repeated execution of a block of SQL statements. WHILE checks a <search condition> before execution of the block, and REPEAT checks it afterwards. All three statements are allowed to have a statement label.

PostgreSQL

- PostgreSQL 8.1
- POSTGRESQL's ancestor was Ingres, developed at the University of California at Berkeley (1977-1985). The Ingres code was later enhanced by Relational Technologies/Ingres Corporation, which produced one of the first commercially successful relational database servers.

Create

- CREATE [[GLOBAL | LOCAL] { TEMPORARY | TEMP }] TABLE *table_name* ([{ *column_name data_type* [DEFAULT *default_expr*]
  [*column_constraint* [...]] | *table_constraint* | LIKE *parent_table* [{ INCLUDING | EXCLUDING } DEFAULTS] } [, ...]])
  [INHERITS (*parent_table* [, ...])]
  [WITH OIDS | WITHOUT OIDS] [ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP }]
  [TABLESPACE *tablespace*]

Inheritance in PostgreSQL

- CREATE TABLE cities (name text, population float, altitude int -- (in ft));

CREATE TABLE capitals (state char(2)) INHERITS (cities);

Capitals inherits from cities.

- SELECT name, altitude FROM cities WHERE altitude > 500;
- name | altitude ----------+

  Las Vegas | 2174

  Mariposa | 1953

  Madison | 845

- SELECT name, altitude FROM ONLY cities WHERE altitude > 500;
- name | altitude

  Las Vegas | 2174

  Mariposa | 1953