O-R DBMS

IF 4035 – Basis Data Non Relasional

Ir. Hira Laksmiwati MSc,

September 2010

Difference between relational databases and OO databases

- The difference between relational databases and OO databases is the way in which relationships are handled.
- In OO databases, the relationships are represented with OIDs, which improves the data access performance.
- In relational databases, relationships among tuples are specified by attributes having the same domain.

OODBMSs Problems

- The main drawback of OODBMSs has been poor performance. Unlike RDBMSs, query optimization for OODBMs is highly complex.

- OODBMSs also suffer from problems of scalability, and are unable to support large-scale systems. Some examples of OODBMSs are O2 (now called Ardent) developed by Ardent Software, and the ObjectStore system produced by Object Design Inc.

Object-Relational DBMS (ORDBMS)

- An ORDBMS supports an extended form of SQL called SQL3 that is still in the development stages. The extensions are needed because ORDBMSs have to support ADT's.

- The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns and SQL is used as the query language and the result of a query is also table or tuples (rows).

Introduction to O-R database systems

RDBMSs currently dominant database technology with estimated sales $50 billion with tools sales included, and growing rate possibly 25% per yr.

OODBMS market still small, with sales of $150 million in 1996 and a 3% market share in 1997.

- Some expect OODBMS market to grow at over 50% per year, but unlikely to overtake RDBMS
- Vendors of RDBMSs conscious of threat and promise of OODBMS.
- Agree that RDBMSs not currently suited to advanced database apps,
- Reject claim that extended RDBMSs will not provide sufficient functionality/be too slow to cope adequately with new complexity.

- Can remedy shortcomings of relational model by extending with OO features.

• Some expect OODBMS market to grow at over 50% per year, but unlikely to overtake RDBMS

•too slow to cope adequately with new complexity.

Can remedy shortcomings of relational model by extending with OO features.

characteristics of an ORDBMSs

- Base datatype extension,
- Support complex objects,
- Inheritance, and
- Rule Systems.

Advantages/disadvantages of O-R database systems

Advantages:

- Resolves many known weaknesses of RDBMS.
- Reuse and sharing:
  - Reuse: from ability to extend server to perform standard functionality centrally.
  - increased productivity for developer and end-user.
- Preserves significant body of knowledge and experience gone into developing relational applications.

Disadvantages:

- Complexity.
- Increased costs.
- Proponents of relational approach believe simplicity and purity of relational model are lost.
- Some believe RDBMS is being extended for what will be a minority of applications.
- OO purists not attracted by extensions either.
- SQL now extremely complex.

OODBMS vs. ORDBMS

- Use an OODBMS:
  - When your application retrieves relatively few (large) complex objects and works on them for a long period before moving to the next object
  - When you are happy programming in an OO language
- Use an ORDBMS:
  - When your application processes a large number of short-lived transactions (e.g. ad-hoc queries) on complex data items.

An early example – Postgres

Postgres ('Post Ingres') is a research DBMS designed to be potential successor to INGRES.

Some of the objectives of the project were to provide:

- better support for complex objects.
- user extensibility for data types, operators & access methods.
- active database facilities (alerters & triggers) & inferencing support.
- Make as few changes as possible to the relational model.

Postgres extended RDM (used "the first way") to include:

- Abstract Data Types,
- Data of type 'procedure',
- Rules.

Supported OO constructs such as aggregation, generalization, complex objects with shared subobjects, and attributes that reference tuples in other relations.

SQL3 – the way forward for ORDBMS?

- *SQL3 is a superset of SQL/92. Therefore, whatever worked in an implementation of SQL/92 should also work in an implementation of SQL3*
- *(Now known as SQL:1999)*

Some of the new OO Features in SQL3:

- NEW TYPES: extended base types, row types, user-defined types, reference types, collection types (arrays, sets, lists etc.)
- SUPERTYPES: the ability to define a hierarchy of super-types and sub-types
- SUPERTABLES: the ability to define a hierarchy of super-tables and sub-tables.
- USER-DEFINED PROCEDURES, functions, and operators.

Release of SQL3 fell significantly behind schedule and was only finalized in 1999 (SQL2 in 1992). Some features have been deferred to SQL4.

SQL3: Row types

- A column in a table can store a composite value of type ROW.
- A value of type ROW is a collection of (presumably related) data items
- A ROW is defined by giving a sequence of field names (each with their type)

Allows composite (aggregated) data to be:

- stored in variables,
- passed as arguments to routines,
- returned as return values from function calls.

Example:

```
CREATE TABLE Branch (branchNo CHAR(4),
                     address ROW(street VARCHAR(25), city VARCHAR(15),
                                 postcode ROW(cityIdentifier VARCHAR(4),
                                 subPart VARCHAR(4))));
INSERT INTO Branch VALUES ('B005', ROW ('22 Deer Rd', 'London',ROW('SW1', '4EH')));
```

SQL3: Collection types

Collections: OO "collections" (arrays, sets, lists) of other types.

- E.g. A single column can be a SET and store multiple values (SQL4).
- Can result in nested tables.
- SQL3 has parameterized ARRAY collection type.
  - (may be an ARRAY of basic type, UDT, row type, or another collection).

Example: Use of a collection SET of staff names at a given branch

```
CREATE TABLE Branch (branchNo CHAR(4),
                     address ROW(street VARCHAR(25), city VARCHAR(15),
                                 postcode ROW(cityIdentifier VARCHAR(4),
                                 subPart VARCHAR(4))),
                     staff  SET (VARCHAR(20)));
```

SQL3: UDTs

- UDT = User Defined Type
- Two categories of UDTs:
  - Distinct types (simplest)
  - Structured types
- May be used in same way as built-in types.

Distinct types:

- allows built-in error-checking to differentiate between same underlying built-in types (only) that should be used in different ways:

```
CREATE TYPE OwnerNoType AS VARCHAR(5);
CREATE TYPE StaffNoType AS VARCHAR(5);
```

  - (Would get error if attempt to treat as instance of wrong type)
- Not the same as SQL domains, which constrain the set of storable valid values

SQL3: UDTs

Structured Types

- May have one or more attributes (of any type, including UDTs)
- All aspects of their behaviour are provided through methods, functions and procedures, defined by the user as part of the type (*like OO Class*)
- System-generated (hidden) “get” and “set” functions provide access to attribute values
- Comparisons of values done only through user-defined functions
- Example:

```
CREATE TYPE emp_type AS (EMP_ID INTEGER, SALARY REAL)
INSTANTIABLE
INSTANCE METHOD GIVE_RAISE (AMOUNT REAL) RETURNS REAL;
```

- Can access attribute value using common dot notation: p.fName

SQL3: Reference types and object identity

- Reference type provides similar functionality as OID of OODBMSs.
- It is a reference to a row in a table of instantiations of a given Structure type
  - I.e. a table where each row is an instantiation of a Structure type
  - So the columns are the attributes of the Structure type
  - For example:

    CREATE TABLE empls OF emp_type;

- A value can be declared to be a REF type but must be scoped. Example:

  CREATE TYPE co_type AS (co_name VARCHAR(20), ceo REF(emp_type));
  CREATE TABLE company of co_type (SCOPE FOR ceo is empls);
- SQL3 syntax for following a reference: SELECT co_type.ceo->salary;
  - allows “path expressions”.
  - also give optimizer alternative way to navigate data instead of joins.

SQL3: Subtypes and Supertypes

UDTs can participate in subtype/supertype hierarchy using UNDER clause.

- Single inheritance only (originally, SQL3 allowed multiple inheritance)
  - Subtype inherits all attributes and behavior of supertype.
- Can define additional attributes and functions and can override inherited functions.
- Concept of substitutability supported: whenever instance of supertype expected instance of subtype can be used in its place.

SQL3: Subtypes and Supertypes

Example:

```
CREATE TYPE PersonType AS ( PRIVATE date_of_birth DATE,
PUBLIC   name VARCHAR(15),  address VARCHAR(50), tel VARCHAR(13)) NOT FINAL;

CREATE TYPE StaffType UNDER PersonType AS (
        staffNo VARCHAR(5),        position VARCHAR(10) DEFAULT 'Assistant',
        salary DECIMAL(7, 2),      branchNo CHAR(4),
        CREATE FUNCTION isManager (s StaffType) RETURNS BOOLEAN
        BEGIN   IF s.position = 'Manager' THEN RETURN TRUE;
                ELSE RETURN FALSE;
                END IF
        END)
        INSTANTIABLE
        NOT FINAL;
```

SQL3: User-defined Routines (UDRs)

UDRs define methods for manipulating data.

- Defined as part of a Structured type
  - Or separately as part of a schema
- May be a procedure (returns no value) or function (returns a value)
- May be externally provided in standard programming language
  - or defined completely in SQL.
- May be invoked from SQL CALL statement.
- May have parameters, each of which may be IN, OUT or INOUT,
  - and may have a body if defined fully within SQL.

SQL3: Polymorphism

Routine (procedure, function) names may be overloaded, provided:

- no two functions in same schema have same signature;
- no two procedures in same schema have same name and number of parameters.

SQL3 uses generalized object model,

- so types of all arguments considered when deciding which routine to invoke (left to right).

- Precedence lists used to determine closest match.

SQL3: Persistent stored modules

SQL3 has some new statement types to make it computationally complete.

Behavior (methods) can be stored/executed from within database as SQL statements.

Can group statements into a compound statement (block), with its own local variables.

Some of the new statements are:

- An assignment statement.
- An IF … THEN … ELSE … END IF statement.
- CASE statement.
- A set of statements for iteration: FOR, WHILE, and REPEAT.
- A CALL statement to invoke procedures and a RETURN statement.

SQL3: Triggers

Trigger: An SQL (compound) statement executed automatically by DBMS as side effect of a modification to named table.

Use of triggers include:

- Validating input data and maintaining complex integrity constraints that otherwise would be difficult/impossible.
- Providing alerts.
- Maintaining audit information.
- Supporting replication.

Major advantage - standard functions can be stored within database and enforced consistently.

disadvantages:

- Complexity.
- Hidden functionality.
- Performance overhead.

Syntax:

```
CREATE TRIGGER TriggerName
  BEFORE | AFTER <triggerEvent>  ON <TableName>
  [REFERENCING <oldOrNewValuesAliasList>]
  [FOR EACH {ROW | STATEMENT}]
  [ WHEN (triggerCondition) ]      <triggerBody>
```

The differences between the three approaches

Criteria	RDBMS	ODBMS	ORDBMS
Defining standard	SQL2	ODMG-2.0	SQL3 (in process)
Support for object-oriented features	Does not support; It is difficult to map program object to the database	Supports extensively	Limited support; mostly to new data type
Usage	Easy to use	OK for programmers; some SQL access for end users	Easy to use except for some extensions
Support for complex relationships	Does not support abstract datatypes	Supports a wide variety of datatypes and data with complex inter-relationships	Supports Abstract datatypes and complex relationships
Performance	Very good	Relatively less	Expected to perform

The differences between the three approaches

Criteria	RDBMS	ODBMS	ORDBMS
Product maturity	Relatively old and so very mature	This concept is few years old and so relatively matur feature	Still in development stage so immature
The use of SQL	Extensive supports SQL	OQL is similar to SQL, but with additional features like Complex objects and object-oriented features	SQL3 is being developed with OO features incorporated in it
Advantages	Its dependence on SQL, relatively simple query optimization hence good performance	It can handle all types of complex applications, reusability of code, less coding	Ability to query complex applications and ability to handle large and complex applications
Disadvantage	Inability to handle complex applications	Low performance due to complex query optimization, inability to support large-scale systems	Low performance in web application
Support from vendors	It is considered to be highly successful so the market size is very large but many vendors are moving towards ORDBMS	Presently lacking vendor support due to vast size of RDBMS market	All major RDBMS vendors are after this so has very good future

Slowly move

- The other current ORDBMSs include Oracle8, from Oracle Corporation, and Universal DB (UDB) from IBM. Also, Stonebraker point out that applications from Relational DBMSs (simple data with query) will slowly move towards the Object-Relational DBMSs (complex data with query).

five architectural options given by Dr. Stonebraker

- Supply plug-in code to make function calls to other applications.
- Add separate API's and server subsystems to support object functionality.
- Simulate specialized object-relational functionality in a middleware layer.
- Completely redesign the database engine.
- Add a new object-oriented layer to support rich datatypes atop a proven relational database engine.

Conclusion

- In spite of many advantages, ORDBMSs also have a drawback. The architecture of object-relational model is not appropriate for high-speed web applications. However, with advantages like large storage capacity, access speed, and manipulation power of object databases, ORDBMSs are set to conquer the database market.

TUGAS OODB :

- Deskripsi Tugas :
  - Serupa dengan deskripsi Tugas Temporal DB per kelompok
  - Aplikasi yang dipergunakan Postgress (SQL 3)
  - Kelompok sama seperti tugas Temporal
  - Laporan tugas, diserahkan Senin 29 Nopember 2010 , 12.00 WIB, di TU Lab Basis Data
  - Asumsi yang dibuat agar disampaikan dalam laporan yang ditulis.
  - Format Laporan mengikuti Assignment yang lalu.
  - Presentasi global dilakukan Rabu, 24 Nopember 2010