Object-Relational Databases

Ashutosh Tiwary

What is it?

- RDB + Objects + Rules
- Extensibility
  - user defined types and methods
  - user defined functions and operators
  - inheritance and polymorphism
- Richer SQL
  - SQL3

Some history

- POSTGRES (Stonebraker et al. 1986)
  - RDB + ADT + Procedures + Rules
  - POSTQUEL with transitive closure
  - Illustra (1992), Informix Universal Server (1996)
- STARBURST (L. Haas, B. Lindsay, C. Mohan 1984-1992)
  - RDB (Aries) + Extensibility + Rules + Query Processing
  - IBM DB2 New Edition (1995)
- PREDATOR (P. Seshadri 1996)
  - Extensibility

Some history

- UniSQL (Won Kim 1990)
  - OODB based (ORION)
  - ODMG object model + relations + extensibility
  - SQL3 support
  - Client side language bindings to reduce impedance mismatch
- Oracle 8
  - RDB + weird stuff

What applications need it?

- Manipulate complex data (images, CAD data)
- Complex queries that need to execute on the server

```
slides (        id          int,
                date        date,
                caption     document,
                picture     photo_CD_image);
landmarks(      name        varchar(30),
                location    point);
```

Find a sunset picture taken within 20 miles of Sacramento

```
select id
from slides P, landmarks L, landmarks S
where sunset (P.picture) and contains (P.caption, L.name) and
L.location || S.location and S.name = 'Sacramento'
```

What applications need it?

Extensible

```
bond (      rate            float;
            date_bought     date;
            date_sold       date;
            interest        float;      )
update bond set interest = rate * (date_sold - date_bought)
```

The DBMS Matrix

	Simple Data	Complex Data
Query (SQL)	RDB ($8B, 25% pa) **100**	ORDB **150**
No Query	File System	OODB ($100M, 50% pa) **1**

Comparing different types of DBs

- RDB: simple data, not extensible, high security, many concurrent users doing simple things, executes on the server (TP)
- OODB: complex data, extensible, trade security for performance, fewer concurrent users doing more complex things, executes mostly on the client (ECAD)
- ORDB: simple and complex data, extensible, high security, many concurrent users doing complex things, executes on the server (GIS VLDB)

Enhanced Data Types

- Base Data Type -- define new types and operations

```
create type mytype_t(
                internallength = 8;
                input = mytypeInput;
                output = mytypeOutput;)

create function function_name (type_name_1,.. type-name-k)
returns type-name as (file-name or SQL expression);

create operator binding oper_name to function_name;
```

- Base types are opaque.

Enhanced Data Types

- Complex Objects -- Base Types + Type Constructors
  - basic: composites (records), sets, references
    - setof(T), ref(T), stored in tables
    - use a reference (OID) instead of foreign-key/primary-key
  - advanced: list, stack, queue, array….
- Tables are distinct from types: Containers of instances
- Inheritance of data types and functions
  - queries span sub-types unless restricted by *only*.

RULES: Create rule new_update_update as

```
        on update to emp.salary where current.name = 'Mike'
        do update emp
                set salary = new.salary where name = 'Jane'
```

Extensibility

- Dynamic Linking
  - simple to add, easy to change, no down time, small footprint
- Client or Server Activation
  - number and overhead of invocation, data transfer, dynamic??
- Callbacks
  - SQL queries inside functions -- *sunset()* and private database
- User-defined access methods
  - indices on user-defined types and functions
- Arbitrary-length types
  - images cannot be fixed length -- not BLOB in RDBs

Extensibility

- Security: how do you execute user-defined functions
  - client: only trashes the user's data
  - server: could trash everyone's data
  - separate address space: cost of crossing and data transfer
  - firewalls or sandboxing: code to monitor jumps and references outside a region of memory
  - interpreted type safe languages like Java
    - Godfrey [SIGMOD 98] show that Java performance is not too bad
  - rings of protection (Multics)

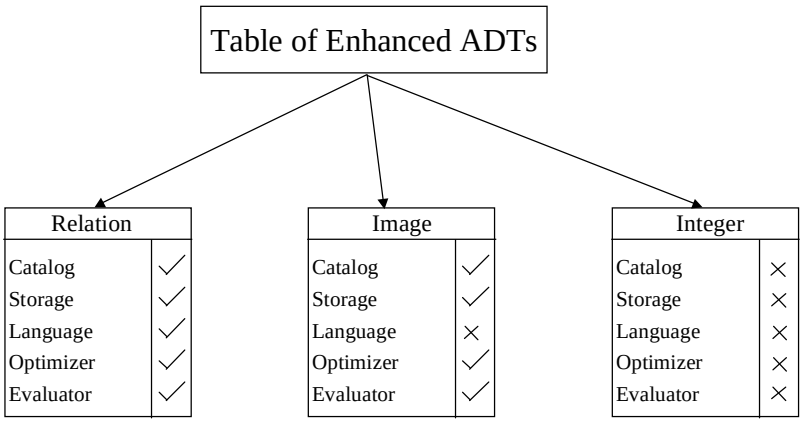
Richer Query Language

- SQL + OQL = SQL3
  - normal SQL
  - path expressions
  - user-defined types, functions and operators
  - computationally complete

Query Optimization

- Operator or function notation
- Generic B-trees and user-defined comparison operators
- Indices on a function of the data
- User-defined access methods (indices)
- User-defined selectivity and cost for functions

Query Optimization: The Predator Approach



Parallelization and scaling is a problem

- RDB parallel execution: properties of relational operators
  - table partitioning across multiple CPUs
  - pipelining: tuples over multiple operators
  - partitioned execution: split and combine operators over CPUs
- Partitioning complex objects and UDFs is hard
  - execution skew due to complex data and UDFs
  - Refs lead to non-local execution
- How do you parallelize UDFs

Architectural Options for relational Vendors

- Write one from scratch: ILLUSTRA
  - needs the features and reliability of the old system
  - better than the old system, under-estimate effort (DEC RDB)
- Evolve incrementally: INGRES, IBM DB2, MS OLE/DB
- Write a wrapper: HP OpenDB and Odaptor
  - wrapper maps ORDB data/queries to a RDB
  - this mapping is inefficient: poor performance

Architectural Options for relational Vendors

- Write a gateway: don't understand
  - break a query into relational and object-relational part
  - execute on RDB and ORDB and combine results
  - inter-operability strategy: why do you need this?
- Object-relational top on relational storage manger: Tandem
  - table access, index access, page management, locking, recovery
  - Illustra + Tandem non-stop storage manager

Architectural Options for OODB Vendors

- Extend an OODB: executes on the client
  - build full relational and SQL support
  - security and client-only execution don't mix
  - extend to execute code on the client and the server
  - build a good optimizer for SQL3/OQL
  - don't have enough resources to focus on different markets
- Object-relational engine on a persistent store: UniSQL/Versant
  - easy and efficient
  - reduce impedance mismatch between ORDB and existing OODB language bindings.