Introduction to Object-Oriented and Object-Relational Database Systems

Database Design

Logical Database Design:
How to translate a schema in the conceptual data model (e.g. extended ER-schemas) to a schema in the DBMS data model (e.g relational tables)

PROBLEM:
Semantics may disappear or be blurred when data is translated from extended ER-model to less expressive relational data model

Physical Database Design:
E.g by indexes:
- permit fast matching of records in table satisfying certain search conditions.

PROBLEM:
New applications may require data and index structures that are not supported by the DBMS.

E.g. calendars, numerical arrays, geographical data, images, text, voice, etc.
Database Manipulation

• Typical query language operations are:
  - Searching for records fulfilling certain selection conditions
  - Iterating over entire tables applying update operations

PROBLEM: Would like to be able to customize and extend query language for different application areas, maps, time series, images, etc.

Evolution of DBMS technology

Classical DBMSs

• Applications:
  Administrative applications
e.g. banking (ATMs)

• Properties:
  Very large structured data volumes
  Very many small Transactions On-line (High transaction rates)
  Occasional batch programs
  High Security/Consistency

New DBMS applications areas

CAD Computer Aided Design
Multi-media databases (images, maps, voice, time series,...)
Scientific Applications (measurements, logs)
Hypertext databases/documents (WWW/HTML/XML)
New DBMS applications (for OODB)

- New needs for e.g. CAD and scientific databases:
  - Extensibility (on all levels)
  - Better performance
  - Expressability
    - E.g. Object-Orientation needed
    - Tight programming language interfaces
    - E.g. C++, Java
  - Long transactions
    - E.g. Engineering requires checkin/checkout model
  - Very large objects

Classification of Database applications

<table>
<thead>
<tr>
<th>Query</th>
<th>No Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>VOD</td>
</tr>
<tr>
<td>Personal db</td>
<td>Text Editor</td>
</tr>
<tr>
<td>Many appl.</td>
<td>Simple Editor</td>
</tr>
<tr>
<td>Multimedia retrieval</td>
<td>CAD system</td>
</tr>
<tr>
<td>Temporal data</td>
<td>Course planning</td>
</tr>
<tr>
<td>Measurements</td>
<td>Complex computations</td>
</tr>
<tr>
<td>Customized search</td>
<td></td>
</tr>
</tbody>
</table>

Kinds of DBMS support

<table>
<thead>
<tr>
<th>Query</th>
<th>No Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational DBMSs</td>
<td>File systems (scalable)</td>
</tr>
<tr>
<td>Object-Relational DBMSs</td>
<td>Storage managers</td>
</tr>
</tbody>
</table>

Object Stores (OO databases)

- First generation ODBs (around 1990)
  - Extend OO programming language with DBMS primitives
  - E.g. Smalltalk, C++, Java
  - Allow persistent data structures in C++ programs
  - Navigate through database using C++ primitives (as CODASYL)
  - An object store for C++

- Many products, e.g.:
  - Objectivity, Versant, ObjectStore
- Special embedded (C++/Java) OO Query language proposal: OQL
Object Stores (OO databases)

Pros and cons:
- Long transactions with checkin/checkout model
- Always same host language (C++/Java)
- High efficiency only for checked-out data
- Primitive ‘query languages’
- No methods in database (all code executes in client)
- Rudimentary data independence (no views)
- Limited concurrency
- Unsafe, database may crash
- Slow for many small transactions (e.g. ATM applications)

Orthogonal Persistence in Object Stores

Integrated with programming language
E.g. C++/Java with persistent objects (e.g. ObjectStore/Pjama)
```java
class PERSON { ... };
```
```c++
static PERSON p; // Local for execution
persistent PERSON p; // Exists between program executions
```

Orthogonal Persistence in Object Stores

Pointer swizzling:
Automatic conversion from disk addresses to pointers
References to data structures on disk (OIDs) look like regular
C++/Java pointers/references!
Navigational access style.
Fast when database cached in main-memory of client!
Preprocessed by OODBMS for convenient extension of C++
(JDK support in Pjama research project)

Object-Relational Databases

Second generation ODBs (around 1997)
Idea:
- Extend on RDBMS functionality
- Customized (abstract) data types
- Customized index structures
- Customized query optimizers
- Use declarative query language, SQL:99

Extensible DBMS technology:
- Object-orientation for abstract data types
- Data blades provide:
  - User definable index structures
  - Cost hints and for the query optimizer
Object-Relational Databases

**Pros and cons:**
- Support for high-level SQL queries, compatibility
- Views, logical data independence possible with queries
- Programming language independence
- Stored procedures, triggers, constraints
- High transaction performance by avoiding data shipping
- Overkill for application needing just a C++ object store
  Performance may suffer compared to OODBs for applications needing just an object store
- May be very difficult to extend index structures and query optimizers

Object-to-Relational Bridges

**Idea:**
Object-Store with relational database as back-end
Persistent objects in Java stored in relational databases
Interface stubs generated for easy Java programming
Some query language support

**Pros and cons:**
- No need to develop new storage manager
- Scalable search in back-end possible using JDBC/SQL
- Slower than JDBC
- No control over database schema!

**Products:** Hibernate, ObjectRelationalBridge (Apache), Castor, TopLink(Oracle)

OO/OR Comparison

**Object identity**
E.g. for structure sharing:
Unique OIDs maintained by DBMS
E.g. OR:
create Person instances :tore, :kalle, :ulla;
In OO: use OO programming (C++, Java) constructs.

**Complex objects**
Not only tables, numbers, strings
But sets, bags, lists, and arrays, i.e. non-1NF relations.
E.g. OR: set courses{:tore} = {:c1,:c2,:c3};
OO: use OO programming constructs in e.g. C++.
• Class Hierarchies as modelling tool (both OO/OR)
  Classification
  E.g. OR:
  ```
  create type Student under Person;
  Students are subsets of persons.
  ```
  Specialization
  Student subtype of Person with extra
  attributes University, Classes, ...
  ```
  create function University(Student) -> Charstring as...
  ```

• Computational completeness
  OR databases: SQL:99: Turing complete stored procedures
  language executed in database server
  OO Databases: C++/Java code executed in client

• Persistence
  OR databases: Embedded queries to access persistent objects
  OO databases: Transparent access to persistent objects
  by swizzling

• Secondary storage management
  OR databases: Indexes can be implemented by user
  (difficult!)

• Concurrency
  OO databases: Long transactions with checkin/checkout
  OR database: Normally short transactions

• Ad Hoc Query Facility
  OO Databases: Weak
  OR Databases: Very strong and extensible

• Data independence
  OO Databases: Very weak
  OR Databases: Strong, e.g. using views

• The ODMG standard proposal:
  R. Cattell, Ed.: The ODMG-93 Standard for Object Databases
  [http://www.odbms.org/odmg.html](http://www.odbms.org/odmg.html)

• The SQL-99 standard proposal:
  ISO standard