Object-Oriented Databases & Relations ▶ Objects

Robert Keller
CS 133
3/3/04
What is an OODB?

- OODB = Object-Oriented Database
- Some of the earliest data models were special cases:
  - Network or CODASYL model
  - Hierarchical model
- These models, being more focused, potentially provide for more efficient querying than relational models,
- at the expense of making queries more complex and less dynamic.
OOODB is like Object-Oriented Programming, except that:

- Objects are *persistent* (survive the current session)
- Transaction mechanism is available:
  - Changes only *committed* at the end of a transaction.

Essentially:

- Your C++ or Java program becomes a database program.
OO vs. Relational

- OO doesn’t use SQL or the equivalent
- Queries are typically “wired in”
- There is an emergent OQL (Object Query Language)
- OODB has closer ties to the Entity-Relationship model and UML (Unified Modeling Language).
- Another facet: CORBA standard (Common Object Request Broker)
3 Choices for Object Management ...

Java and C++
In-Memory
Object Networks

Serialize to disk
Store directly in an Object Database
Convert to relational model
Typically data will be brought to main memory from secondary memory the first time the data are accessed.

The code to transfer the data is *transparent* to the program.

The data will remain in main memory until committing or until the end of the session.
Examples of OODBMS
(Object-Oriented Database Management Systems)

- A somewhat aging list:
  - POET™ (http://www.poet.com/)
  - ObjectStore™ (http://www.odi.com/)
  - Objectivity™ (http://www.objectivity.com/)
  - Versant™ (http://www.versant.com/) merged with POET
  - Gemstone™ (http://www.gemstone.com/)
  - O2™ (http://www.ardentsoftware.com/)

- Standards are emerging, thanks to the ODMG (Object Data Management Group)
Use POET™ as “Typical” (but now obsolescent)
Use Java as the Language

- Data are stored on disk with symbolic name.
- Schema (through a schema file) determines the DB structure.
- Upon *opening* the database, the *root* of the structure on disk is *bound* to a reference *variable* in memory.
- Further bindings take place as the structure is navigated, using OOP.
POET/Java Examples

- `Database db = Database.open(dbName, Database.openReadWrite);`
- `Extent e = new Extent(trans, classname);`
  
```java
  while( e.hasMoreElements() )
  {
    ... e.nextElement() ...  
  }
```
POET/Java Examples

- db.add(myobject, trans);

- ObjectServices.delete(myobject);

reference

static method
Transactions

- All access (even read-only) is done within a transaction.

- Transaction trans = new Transaction(db);

  trans.begin();

  trans.checkpoint();

  trans.commit();
[schemata\poetOgreDict]
oneFile = true

[databases\poetOgreDB]
schema = poetOgreDict
location = SAME
oneFile = true

[classes\poetOgreObject]
persistent = true
hasextent = true
schema = poetOgreDict
Connecting Objects to Relations

- Model 1: Store object (reference) as \textit{values} in relations
  - 1 object (reference) = 1 attribute value
  - An object is of a specific data type. Thus a set of objects of the same type can be stored as a single relation.
  - Object’s methods are over-and-above the relational model.
Connecting Objects to Relations

- Model 2: Relations are **associations** in OOP
  - Objects are free-standing
  - Each association between classes is a relation.
Connecting Objects to Relations

- Model 3: Objects identified with tuples
  - Attributes become fields.
  - Values of attributes become values of fields.
  - A whole relation is a set of objects of the same class.
OGRE
(Objects Generated From Relations)

A transformation tool developed by Jeff Polakow ‘98 and Robert Keller sponsored by JPL
Genesis of OGRE

- pre-1996 JPL: “Grok”
- 1996 JPL CS Clinic Project: “Condor”
- 1997 JPL CS Clinic Project: “Ogre” (originally Oracle-Grok Interface)
Ogre Features (1)

- “Any” JDBC/ODBC-Compliant Relational Database \( \rightarrow \) POET OODB using Model 2.
- In principle, could be made to work with any OODB.
- Could also generate text or XML.
Ogre Features (2)

- GUI allows a naïve user to generate the OODB; essentially converts graphical queries into SQL to access RDB.

- The conversion alone may be of interest, aside from the connection with OODBs.
Ogre Architecture

Relational Databases
- Informix database
- Oracle database
- Sybase database

Object-Oriented Databases and other media
- POET database
- Screen image
- Text file

Symantec dbAnywhere

User

JDBC

Point & Click

SQL

Java
Loading an RDB into Ogre
Selecting Tables

```
<table>
<thead>
<tr>
<th>call_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog</td>
</tr>
<tr>
<td>cust_calls</td>
</tr>
<tr>
<td>customer</td>
</tr>
<tr>
<td>items</td>
</tr>
<tr>
<td>log_record</td>
</tr>
<tr>
<td>manufact</td>
</tr>
</tbody>
</table>
```

```
File

5: info: RDB stores7 loaded, Select Table(s), then click on canvas.
6: error: No schemas were selected.
7: info: Getting RDB table cust_calls from stores7 ...
8: info: Getting RDB table customer from stores7 ...
9: info: cust_calls loaded
10: info: customer loaded
```
Displaying Selected Relations

Relations

Attributes
From here we can generate objects. There will be two “types” of object, one for each relation.
“Universal” Object Type

- In the case of Ogre, objects are actually all the same Java class:
  - The principal field is an association list (attribute-value pairs).
  - This is used because the schema would otherwise be unknown until runtime.
- In principle, a static-compilation approach is possible.
Joining Relations to create Single Objects

Combined attributes still have unique names.
Joining Nuances

- unjoin
- rename
- hide
- mask
Viewing OODB
Generated SQL

```
11: info: SQL: Select cust_calls.call_dtime, cust_calls.user_id, cust_calls.call_code, cust_calls.call_date
12: info: Completed query: Select cust_calls.call_dtime, cust_calls.user_id, cust_calls.call_code, cust_calls.call_date
13: info: SQL: Select cust_calls.call_dtime, cust_calls.user_id, cust_calls.call_code, cust_calls.call_date
14: Populating ODB: .......
15: info: Population completed, committed 7 objects to database.
16: info: Completed query: Select cust_calls.call_dtime, cust_calls.user_id, cust_calls.call_code, cust_calls.call_date
```
Graphical Querying with Built-in Operators
Ogre Web Page

http://www.cs.hmc.edu/~keller/ogre/