Objects and Databases

CPS 296.1
Database and Programming Languages: Crossing the Chasm
Jun Yang
Duke University
January 26, 2010

Image from http://www.migrationheritage.nsw.gov.au/cms/wp-content/gallery/objectsthroughtime/objects/dunbar/objects.jpg

Announcements

- For Thursday
  - 1 paper about the experience of making a persistent PL
  - Review required
  - Bilgen and Ryan will lead the discussion
- For next week
  - Explore Java/Hibernate and Python/Django
    - Read online tutorials and documentation
    - Search for people's critiques
    - Perhaps try some coding yourself
  - Matt and Peter will lead the discussion

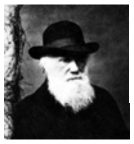
2

Overview

- Atkinson et al. "The Object-Oriented Database System Manifesto." *Deductive and Object-Oriented Databases* 1989
  - A group of researchers converging on a set of mandatory, optional, and open features for OODBMS
    - ➲ Did vendors follow their advice?
    - ➲ How close did ORDBMS get?
- Carey and DeWitt. "Of Objects and Databases: A Decade of Turmoil." *VLDB* 1996
  - 4 (+1) different ways of embracing objects
  - Past history, present status (as of 1996), and future predictions
    - ➲ How did their predictions pan out?
    - ➲ What remain the most important challenges as of 2010?

3

OODMBS Manifesto



- Backgrounds of authors
  - OODBMS (majority) + persistent PL
  - Academia (majority) + industry
    - But even Bancilhon started out in academia
- Motivation
  - A purely Darwinian approach to system building may lead to dominance by the first "good-enough" system instead of the fittest
  - There was much more confusion in the OODBMS landscape than the relational one
  - Get your definition/terminology straight!

Image from http://onegoodmove.org/1gm/1gmarchive/2005/02/happy_darwin_da.html

4

Mandatory features

- 13 features in basically two categories
- It should be a DBMS
  - 5 features: persistence, secondary storage management (i.e., large data), concurrency, recovery, ad hoc query facility
- It should be OO (consistent with the OO PLs)
  - 8 features: complex objects, object identity, encapsulation, types or classes, inheritance, overriding/overloading/late binding, extensibility, and computational completeness

5

DB-mandatory features

- Persistence, large data, CC, recovery, ad hoc query facility

Discussion points?

➲Requirement of an "ad hoc query facility" is rather weak
  - "A graphical browser could be sufficient"
  - No program access to the facility $\rightarrow$ burden on programmers

➲Eliminating the need to write additional operations for each UDT (under "ad hoc query facility") is hard
  - Okay at the query language level
  - But efficiency will suffer; e.g.:
    - Queries involving UDT for 3-d boxes will be slow without customized access methods

6

OO-mandatory features

- Complex objects, OID, encapsulation, types/classes, inheritance, overriding/loading/late binding, extensibility, completeness

Discussion points?

➲ Presenting the full extent as a table isn't always a good idea
  - E.g.: the same rectangle type can be used in different contexts

➲ Orthogonal object constructors: any constructor can apply to any object (Postgres didn't have this)

➲ It's reasonable to not extend the collection of constructors (tuples, sets, and lists are minimal)

➲ Differentiating is-part-of/general references is interesting

➲ They argue it's okay to "violate encapsulation" by allowing ad hoc queries to access fields without going through methods
  - IMO queryable fields have implicit getters; so no violation

7

Other features

Mandatory or optional?

- All DB-related: views and derived data, DB admin utilities, integrity constraints, schema evolution facility

Optional

- OO-related: multiple inheritance, type checking/inferencing
- DB-related: distribution, versions
- App-related: design transactions (long or nested)

Open choices

- Mostly PL/religion-related: programming paradigm, representation system, type system, uniformity
  - ➲ Authors are making a stronger statement by marking a feature as open as opposed to optional!

8

Discussion

➲ Was their advice any good?
  - To be fair, they just wanted to clarify, and said, "Thou shalt question the golden rules"
  - Could have been more focused
  - Could have pushed physical data independence further

➲ Did vendors follow their advice?

➲ How close did ORDBMS get?

Image from http://www.definitivejux.net/files/imagecache/container_full/files/news/advice.jpg

9

A decade of turmoil

Four approaches (mid-1980's to mid-1990's)

- Extended relational DBMS
  - Later dubbed OR, exemplified by Postgres
- Persistent OOPL
  - More on Thursday
- Object-oriented DBMS
  - Persistent OOPL + DB features (e.g., indexing, queries, versions)
- DBMS toolkits/components
  - One size cannot fit all
  - Provide tools for "rapidly" developing a domain-specific DBMS
  - EXODUS, GENESIS, DASDBS
  - Starburst (also seen as "developer-extended" relational)

10

Verdicts as of 1996

- Persistent PL and DBMS toolkits were practical dead-ends
- OODBMS failed to deliver
- ORDBMS flourished and appeared to be the winner
- OO client wrappers emerged as a new approach
  - Mostly language-specific, to help with impedance mismatch
  - Integration still imperfect: programmer need to write some SQL, and decide what business logic goes into DBMS
  - ➲ Hibernate and Django are recent examples
- Related efforts
  - CORBA: interoperable object RPC, but don't overdo it!
  - Java: safety makes it an ideal language for UDF
  - DB middleware: a uniform interface over multiple data sources

11

Reasoning behind verdicts

➲ Insights not covered by "What Goes Around Comes Around"?

- On DBMS toolkits
  - Too much work/expertise required to use these toolkits
  - Generalizability is hard—even with sacrifice of usability and performance, functionality is still incomplete
- On CORBA
  - Attempts at factoring object services (persistence, collection, indexing, transaction, etc.) and making each DB object a CORBA object will likely fail due to poor performance
- On OODBMS
  - While OODBMS was betting on "fat clients," "thin clients" talking database APIs like ODBC were becoming the norm

12

Prediction for 2006

- ORDBMS will provide "fully integrated" solutions
  - Truly OO types, as well as views, authorization, triggers, constraints on OO data
    - All standardized in SQL
  - An OO caching layer that supports queries and transactions, and intelligently decides where to execute them
  - OO client wrappers would be a first step
- OODBMS will remain only in niche markets

➲Did they pan out?
  - ORDBMS still has a long way to go
  - OO client wrappers remain popular
  - XML has created much diversion (or a good testbed?)

13

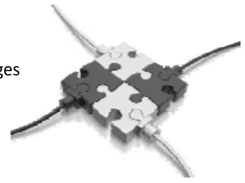
Challenges as of 1996

- ORDBMS
  - Catching up with relational: query processing, views, updates, authorization, triggers, constraints…
  - Extensible access methods in ORDBMS
- Client integration
  - Intelligent object cache, "cooperation hooks" provided by servers
- Parallelization
- Legacy/heterogeneous data sources; AKA information integration
  - Distributed query optimization, semi-structured data, ranked queries
- Standardization
  - Metadata about UDTs/UDFs, access method interface, client/server interface, new query language to shed old SQL baggage

14

Discussion

➲From server extensibility to integration/interoperability
  - Between client/server
  - Across multiple servers
  - Across data models and languages



➲What happened to ORDBMS in the past decade (beyond trying to incorporate XML)?

➲Domain-specific DBMS relevant again?
  - What's the lesson from 1986-1996?

➲What remain the most important challenges as of 2010?

Image from http://www.databaseguides.com/wp-content/uploads/2009/09/Data-Integration-Software-Option.jpg 15