Of Objects and Databases: A Decade of Turmoil
Michael J. Carey, David J. DeWitt

Discussion by: Shervin
Presentation by: Roland
Background

- Michael J. Carey
  - DBMS performance improvements, EXODUS
  - *OO DBMS – Shore-Project (1996)*
  - XML Technologies, SOA, EAI, XQuery

- David J. DeWitt
  - Parallel Database Systems, EXODUS
  - *OO DBMS – Shore-Project (1996)*

- SHORE Project
  - Improvement for: CAx Applications and Unix Filesystem-Support
  - Shore Storage Manager
    - EXODUS - enhancement
    - ARIES – logging
    - *Using state of the art OO-Techniques (1997)*

- Published on VLDB 10 year award for "Object and File Management in the EXODUS Extensible Database System" in 1986
Back to topic:

4 areas of research are covered

1. Extended relational database systems
2. Persistent programming languages
3. Object-oriented database systems
4. Database system toolkits/components
Extended relational database systems

- Evolutionary approach
- Opening up the type system of RDBMS
  - Allow: user defined abstract data types (ADTs)
- ADTs (as proposed in the Ingres Project):
  - Have to be implemented manually once (definition, function)
  - New ADT will be registered with the db system, can be used like an internal type
  - Can be used in queries / loaded dynamically
- Enhanced by Postgres Project:
  - Providing query optimizers with information about ADTs
  - Procedures are treated as data types
Persistent Programming Languages

- Persistence mechanism for type system of oo-programming languages
  - Persist inherent data
  - Make program executions atomic

- Desirable for applications that just need permanent data
  - loss of impedance mismatch
  - Only reasonable for applications that do not need to store huge amounts of data
  - A lot of issues have to solved (binding, type safety, garbage collection of serialized data, etc.)
OO-DBMS

- Approach of merging the best of both worlds:
  - Modern RDBMS
  - Object-oriented programming languages

- No agreement on standards for
  - Data model
  - Query language
  - Query model

- Three early approaches failed
Database System Toolkits/Components

- Basic idea: one DBMS can not serve all needs
- Solution: extensible DBMS
  - Based on set of kernel facilities
  - Tools for rapid development of domain specific DBMS
- Projects in this area:
  - EXODUS
  - GENESIS
  - DASDBS
  - Starburst (might be considered as extended RDBMS)
Discussion Question-1

- However Object-Oriented Database Systems led to many research results from the academic community, why could not they satisfy commercial/business data processing applications and have failed to live up with original commercial expectations (Please mention some specific reasons)? What are their shortcomings which avoid them to be compatible with commercial applications?

- In spite of weaknesses of relational database technology, why they were started to be adopted for use in serious enterprise-scale applications, instead of extended relational database systems, persistent programming language, database toolkit products, or OODBs?
Which approach survived?

- Interesting (research) results, but no further commercial success:
  - DBS toolkits and persistent programming languages
  - OO-DBMS

- New approach that seems to be important:
  - Generating language specific object wrappers for RDBMS

- Survivor:
  - Object-relational database systems
    - a.k.a. extended relational database systems

- Related developments: CORBA, OLE, Java, middleware
Why EXODUS failed

- **Direct reasons:**
  - To much expertise was required to use these systems
  - The design process has not been optimal
  - (nearly) no commercial users
  - Exodus Storage Manager added unwanted indirection
  - The artificial E programming language
  - Query optimizer afforded to much work of the developer

- **Indirect reasons:**
  - OO- and ORDBMS have been flexible enough to **not** justify the use of an extensible approach
OOTDBMSs: Wishlist

Long story short:

- RDBMS functionality (single user)
- mechanism to persist / initialize an object
- high degree of flexibility
Issues with OODBMS → Standards

- Language diversities
- Indexing → efficient handling of path expressions → queries over class hierarchies
- Assembling of objects
- Performance
- Transaction & recovery management
- Several Products (O2, ObjectStore, etc.)
- Object Database Management Group (ODMG)
  - Object Data Language (ODL)
  - Object Query Language (OQL)
  - C++ Interface
Object Relation Database Systems

- Adopt features from the OODB world to relational systems
- Announces itself as “third generation database system”

Scope:
- Subsume all RDBMS features
- Enhance with support for richer objects
- Openness for other systems

In detail (just the differences against OODBMS):
- Separation of datamodel/performance features; multiple language support, layered approach, SQL support
Discussion Question-2

- How do Object-Relational database systems overcome some shortcomings of OODB systems? Regarding these improvements, why aren’t they used in commercial applications, instead of Relational systems?

- How can Database researchers improve OODB systems to overcome weaknesses and problems which were mentioned as casualties of the first decade of doing research on database systems?
Predictions

- **Fully Integrated Solutions**
  - Full ADT and SQL support
  - Independent of programming language
  - Optimized Client support
  - One Object model on all application levels (!)

- **Server Functionality and Performance**
  - Industrial-strength solutions
  - Increasing performance
  - Achieved through a merge of the OQL and SQL
Predictions

- **Client Integration**
  - Fully integrated object query support
  - Clever caching mechanisms will be developed
  - Proper code execution on Server and Client side (Java 1996 - Applets)
  - Server Hooks for rich clients

- **Parallelization**
  - As in the relational world
  - Aiming for handling of huge ADTs to achieve load balancing and parallel solutions
Predictions

- **Legacy Data Sources**
  - Possible technological solution to integrate legacy and other data sources
  - Aiming for a middle ware solution in the sense of the “Enterprise Application Integration (EAI)”

- **Standards**
  - Aiming for standards for all needs
  - With all vendors/researchers agree on them
  - Setting up on data query standard including all the benefits of SQL and leaving out all weaknesses
Conclusion Discussion

- Regarding many researches on database systems which led to emersion of different database systems, why do we still use relational systems in practical applications?

- Do you think those researches achieve their goals to improve the performance and efficiency of database systems to produce such a system to be widely used in practice rather than academic area?

- If you don’t think so, explain your reasons about why they fail to achieve what they were supposed to.