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. 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 to less expressive 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, text, etc.*
The ANSI-SPARC Tree Level Schema Architecture

Acheives data independence

- Logical data independence:
  The capability to change conceptual schema without having to change interfaces between applications and views.
  E.g. Create new table, extend table, split table

- Physical data independence:
  The capability to change the internal schema without having to change applications and conceptual schema.
  E.g. add/drop indexes, change data formats, etc.

**NOTE:** Data Independence is very important since databases continuously change!
Database Manipulation

• Query language:
  Originally a QL could only specify database searches.
  Now the standard query language SQL is a general language for interactions with the database.

• Typical query language operations are:
  - Searching for records fulfilling certain selection conditions
  - Iterating over entire tables applying update operations
  - Schema definition and evolution operators
  - Object-Oriented Databases have create and delete objects

**PROBLEM:** Would like to be able to customize and extend query language for different application areas

Views

• Definition:
  A view is a mapping from the Conceptual Schema to a subset of the database as seen by a particular (group of) users.

  The external schema is defined by a set of views.

• Relational views:
  SQL is a closed query language that maps tables into tables
  => SQL allows very general views (derived tables) to be defined as single queries

  **NOTICE:** Views provide logical data independence.
Evolution of Database Technology

- 1968: Hierarchical (IMS)
  - Trees

- 1970: Network model (CODASYL)
  - Complex data structures

- 1980: Relational model (e.g. ORACLE)
  - Tables

- 1990: 1st Generation OODB (e.g. ObjectStore)
  - OO data structures

- 1997: Object Relational DBMS (e.g. SQL99)
  - Object model

New DBMS applications (for OODB)

- Classical DBMS:
  - 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

Problem areas:

- CASE Computer Aided Software Engineering
- CAD Computer Aided Design
- CAM Computer Aided Manufacturing
- OIS Office Information Systems
- Multi-media databases
- Scientific Applications
- Hypertext databases (WWW)

New DBMS applications (for OODB)

• New Needs for Engineering, Scientific databases, etc.:
  - 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
Problems with relational databases for new application areas

Problems with using RDBMSs for OO applications
- Complex mapping from OO conceptual model to relations
- Complex mapping => complex programs and queries
- Complex programs => maintenance problems
- Complex programs => reliability problems
- Complex queries => database query optimizer may be very slow
- Application vulnerable to schema changes
- Performance

Object Stores

- 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.:
  - ObjectStore, Objectivity, Poet, Versant, Ontos, Gemstone, Itasca, O2
- Special embedded (C++/Java) OO Query language proposal: OQL
Object Stores

Pros and cons:

- Long transactions with checkin/checkout model
- Always same 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 { ... };

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)

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*, SQL99
- 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

Complex objects

- Not only tables, numbers, strings
- But sets, bags, lists, and arrays, i.e. non-1NF relations.
  E.g. Courses(:tore) = {:c1,:c2,:c3}
OO/OR Comparison

• Extensibility
  User definable data types, e.g. create type Person,
  Create type Timepoint, etc..
  Operations on new datatypes,
  e.g. name(:tore), :t2 - :t1, :t2 > :t1, etc.
  Both OO and OR allow abstract datatypes through
  object-orientation
  OR databases: Allow extensions of physical representations
  and query processor

• Class Hierarchies as modelling tool (both OO/OR)
  Classification
    E.g. Student subtype of Person
  Shared properties
  Specialization
    Student subtype of Person with extra
    attributes University, Classes, ...
**OO/OR Comparison**

- **Computational completeness**
  - OR databases: Turing complete ‘query’ language: SQL99
  - Code executes on server
  - OO Databases: C++/Java code with embedded OQL statements executes in client (web server)

- **Persistence**
  - OO databases: Transparent access to persistent object by swizzling
  - OR databases: Embedded queries to access persistent objects

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

- **Concurrency**
  - OO databases: Good support for long transactions
  - OR database: Good support for short transactions

- **Ad Hoc Query Facility**
  - OO Databases: Weak
  - OR Databases: Very strong

- **Data independence**
  - OO Databases: Weak
  - OR Databases: Strong
OO/OR Comparison

• Views
  Important for data independence
  Query language required
  Only in OR databases!
• Schema evolution
  Relational DBs have it!
  Fully supported in OR databases, primitive in OO databases

Object-Oriented DBMS Standard

• The ODMG standard proposal:
  R. Cattell, Ed.: The ODMG-93 Standard for Object Databases
  Object Data Model
  Object Query Language: OQL (different model than SQL:99)
Object-Relational DBMS Standard

- The SQL-99 (SQL3) standard proposal:
  - ISO standard
  - Very large (1000 pages)
  - SQL-92 is subset
  - Much more than object-orientation included
  - Triggers, procedural language, OO, error handling, etc.
  - Certain parts, e.g. standards for procedures,
    error handling, triggers, already being included
  in the SQL-99 standard.