DATABASE TECHNOLOGY - 1DL124

Summer 2007

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-sommar07/
alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st07/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Introduction to Object-Oriented and Object-Relational Databases

Elmasri/Navathe ch 20, (21), 22
Padron-McCarthy/Risch ch 16

Kjell Orsborn

Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Outline of presentation

• Some General DBMS Concepts
  – limitations of traditional DBMSs
• History of DBMSs
• Object-Oriented Databases
• Object-Relational Databases
• Differences
• Standards
Database Design

- **Database Design:**
  - How to translate subset of reality into data representations in the database.

- **Schema:**
  - A description of properties of data in a database (i.e. a meta-database)

- **Data Model:**
  - A set of building blocks (data abstractions) to represent reality.
    - Each DBMS supports one Data Model.
    - The most common one is the Relational Data Model where data is represented in tables.
    - NOTICE: In some applied fields people use the word ‘Data Model’ instead of ‘Schema’

- **Conceptual Data Model:**
  - A very high level and user-oriented data model (often graphical).
    - CDM not necessarily representable in DBMS or computer!
    - Most common CDM is Entity-Relationship (ER) data model.
    - But also Extended ER models are common

- **Conceptual Schema Design**
  - Produce a DBMS independent Conceptual Schema in the Conceptual Data Model
Logical Database Design

• Logical Database Design:
  – How to translate Conceptual Schemas in the conceptual data model (e.g. ER-schemas) to a Conceptual Schema in the DBMS data model (e.g. relational tables)

• Logical Database Design for the Relational Data Model includes:
  – Key Identification: What attributes are used to identify rows in a table?
  – Normalization: Table decomposition to solve update problems, normal forms

• PROBLEM: Semantics may disappear or be blurred when data is translated to less expressive data model and normalized
Physical Database Design

• Physical Database Design:
  – Physical representation of the database schema optimized with respect to the access patterns of critical applications.

• Indexes:
  – permit fast matching of records in table satisfying certain search conditions.
  – The index structures are closely related to the internal physical representations of the DBMS.
  – Indexes can speed up execution considerably, as well as storing data usually accessed together in the same table.
  – Indexes permit the database to scale, i.e. the access times grow much slower than the database size.

• PROBLEM: New applications may require data and index structures that are not supported by the DBMS. (e.g. calendars, numerical data, geographical data, data exchange formats, etc.)
Data Independence

- **External View:**
  - Mapping Conceptual Schema --> subset of the database for a particular (group of) users.

- **Data Independence:**
  - The capability to change the database schema without having to change applications.
  - NOTE: Data Independence is very important since databases continuously change!

- **Logical Data Independence:**
  - The capability to change conceptual schema without having to change applications and interfaces to views.
  - E.g.: create a new table, add a column to a table, or split a table into two tables

- **Physical Data Independence:**
  - The capability to change the physical schema without having to change applications and logical schema (E.g. add/drop indexes, change data formats, etc.)

- **PROBLEM:** Application programs still often have data dependencies, e.g. to map relational database tables to application object structures.
Database Manipulation

• Query Language:
  – Originally a QL could only specify more or less complex database searches. Now the 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 other operations such as create and delete objects

• The user directly or indirectly calls SQL in the following ways:
  – By running an interpreter that interactively executes SQL commands
  – By running an application program that contains calls to Embedded SQL
  – By running a graphical Database Browser to navigate through the database. (The browser internally calls embedded SQL)

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

• View:
  – A view is a mapping from the Conceptual Schema to a subset of the database as seen by a particular (group of) users.
    • SQL is a closed query language that maps tables into tables => SQL allows very general views (derived tables) to be defined as single queries

• Views provide:
  – External schema
    • Each user is given a set of views that map to relevant parts of the database
  – Logical data independence
    • When schema is modified views mapping new to old schema can be defined
  – Encapsulation
    • Views hide details of physical table structure
  – Authorization
    • The DBA can assign different authorization privileges to views of different users

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

1960
Hierarchical
(IMS)
Trees

1970
Network model
(CODASYL)
Graph

1980
Relational model
(e.g. ORACLE)
Tables

1990
Object-oriented DBMS
(e.g. ObjectStore)
OO data structures

1997
Object-relational DBMS
(e.g. SQL:99)
Object model
New DBMS Applications (for OODBMSs)

• 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 Needs for Engineering, Scientific databases, etc.:
  – Extensibility (on all levels)
  – Better performance
  – Expressability (e.g. Object-Orientation needed)
  – Tight PL Interfaces
  – Long transactions (work in ‘sand box’)

Kjell Orsborn 8/7/07
New DBMS Applications (cont. ...)

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)
Classifying DBMS applications

- **Relational DBMS**
  - e.g. business data processing
- **Object-relational DBMS**
  - e.g. GIS
- **File systems**
  - e.g. text editor
- **Object-oriented DBMS**
  - e.g. ECAD

**Simple data**

**Complex data**

**Query**

**No Query**
Object-Oriented Databases

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-Oriented Databases

• First generation ODBs
• Extend OO programming language with DBMS primitives
  – E.g. C++, SmallTalk, Java
  – Allow persistent data structures in C++ programs
  – Navigate through database using C++ primitives (as CODASYL)
  – An object store for C++, SmallTalk, Java, etc.
• Several products out, e.g.:
  – Objectivity, Versant, ObjectStore, Gemstone, Poet, PJama, O₂
• Special embedded (C++/Java) OO query language proposal: OQL
Object-Oriented Databases

- Pros and cons:
  + Long transactions with checkin/checkout model (sand box)
  + Always same language (C++)
  + High efficiency (but only for checked-out data)
  - Primitive ‘query languages’ (now OQL standard proposed)
  - No methods in database (all code executes in client, no stored procedures)
  - Rudimentary data independence (no views)
  - Limited concurrency
  - Unsafe, database may crash
  - Slow for many small transactions (e.g. ATM applications)
  - May require extensive C++ or Java knowledge
Object-Oriented Databases

- Persistence
- Integrated with programming language:
  - E.g. C++ with persistent objects
    class PERSON { ... };

    {PERSON P; // Local within block... }
    static PERSON p; // Local for execution
    persistent PERSON p; // Exists between program executions

- Pointer swizzling:
  - Automatic conversion from disk addresses to MM addresses
  - References to data structures on disk (OIDs) look like regular C++ pointers!
  - 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

• Object-Relational DBMSs
• Idea:
  – Extend on RDBMS functionality
  – Customized (abstract) data types
  – Customized index structures
  – Customized query optimizers
  – Use declarative query languages, SQL extension (SQL99)
• Extensible DBMS
  – Object-orientation for abstract data types
  – Data blades (data cartridges, data extenders) are database server ‘plug-ins’ that provide:
    • User definable index structures
    • Cost hints and re-write rules for the query optimizer
Object-Relational Databases

- Pros and cons:
  + Migration path to SQL
  + Views, logical data independence possible
  + Programming language independence
  + Full DBMS functionality
  + Stored procedures, triggers, constraints
  + High transaction performance by avoiding data shipping
  + Easy to use declarative queries
  - 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

- Research prototypes: Iris (HP), Postgres (Berkeley), Starburst (IBM)
- Products: Informix, DB2, Oracle
Object-Oriented Databases

• Literature:
  – M.Stonebraker: *Object-relational DBMSs - The next great wave*, Morgan-Kaufmann 1996

• Object-Oriented Manifestos
  – First generation ODB Manifesto: *State-of-the-art OODBs* anno 1990
    • Atkinsson et al: *The OO Database System Manifesto* in W.Kim, J-M. Nicolas, S.Nishio (eds): *1st Intl. Conf. on Deductive and OO Databases*
      Early O₂
  – Object-relational DB Manifesto: *Requirements for next generation DBMSs* anno 1990
    • Stonebraker et. al.: *Third-generation Data Base System Manifesto*
Object-Oriented Databases

The Manifestos:

• Object identity
  – E.g. for structure sharing:
    Unique OIDs maintained by DBMS
    E.g. Parent(:tore) = :ulla, Parent(:kalle)=:ulla

• 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}

• Encapsulation
  – Simplicity
  – Modularity
  – Security
Object-Oriented Databases (manifesto cont. ...)

- Extensibility
  1. User-defined data types and operations on these new datatypes
    - e.g. datatypes: create type Person, create type Timepoint
    - e.g. operations. name(:tore), :t2 - :t1, :t2 > :t1, etc.
    - Both OO and OR allow abstract datatypes through object-orientation
  2. Extensions of physical representations (including indexes) and corresponding operations
    - OO/OR databases allow extensions of physical representations
    - OR databases allow definition of new indexes
  3. Extensions of query processor with optimization algorithms and cost models
    - OR databases allow extensions of query processing

- 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, …
Object-Oriented Databases (manifesto cont. ...)

• 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
Object-Oriented Databases (manifesto cont. ...)

• Data independence
  – OO Databases: weak
  – OR Databases: strong

• 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 Database Standards

• Object-Oriented DBMS Standard
  – The ODMG standard proposal:
    – Includes an Object Data Model
    – Object Query Language: OQL (different model than SQL99)

• Object-Relational DBMS Standards
  – The SQL99 (SQL3) standard proposal:
    • ISO-Final Draft International Standard (FDIS):
      ISO/IEC FDIS 9075-2 Database Language SQL
    – 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 new SQL-99 standard.
Data Exchange Formats

Purpose:

• Standardized formats for sending data between systems
  – examples: STEP/EXPRESS, PDF, HTML, XML, VRML, MIDI, MP3, etc.

• Engineering domain standard: STEP (standard for exchange of product data)
  – STEP is an industry wide ISO standard for exchange of mainly engineering (CAx etc.) data
  – separates meta-data (schema) and data as for databases
  – EXPRESS is data model in database terms: i.e. it is the language in which to define the schema.
  – STEP models are standardized schemas for different engineering application areas, e.g. AP209
  – The exchanged data follows specialized STEP schemas, e.g. PART 21 most common (XML based too, PART 29)
  – CAx vendors normally not able to handle EXPRESS schemas
  – Only PART 29 files following a specific schema, e.g. AP 209
Data Exchange Formats

- The STEP/EXPRESS and database community sometimes use the same terminology with different meanings:
  - **Data model:**
    - database world: schema language (i.e. EXPRESS is a data model)
    - STEP/EXPRESS world: here a particular schema definition written in EXPRESS
    - We therefore avoid the word data model to minimize confusion
  - **Multi-level schema architecture:**
    - database world: external - conceptual - internal schemas
    - STEP/EXPRESS world:
      - Application protocol, AP (c.f. external schema)
      - Integrated resources, IR (c.f. conceptual schema)
Data Exchange Formats

• The XML language
• Extension of HTML to be able to define *own tags* in web documents,
  - for example:
    ```xml
    <polygon>
    <line><start>1.2 1.3</start>
    </end>2.1 3.4</end>
    </line>
    <line><start>2.1 3.4</start>
    </end>4.6 4.2</end>
    </line>
    </polygon>
  ```
• Can also define DTD which is grammar for allowed tags in the documents *referencing* it
• DTDs are more or less well specified schemas
• On-going work to define real schema language for XML: SMLSchema
• XML not object-oriented - only nested structures
Extended Entity-Relationship Diagram
The ANSI/SPARC three-schema Architecture

- Achieves Data Independence