Course Notes on
A Short History of Database Technology

Three Eras of Database Technology

(1) **Prehistory**
- file systems
- hierarchical and network systems

(2) The **revolution**: relational database technology

(3) **Post-relational era**
- new organizations of data, more complex data
- influence of object technology
- more complex applications (e.g., distributed and web-based)

1

Traditional File-Based Approach

- File technology was the first attempt to **automate manual filing systems**
- Collection of applications that each define and manage their own files
- **File**: collection of records with structure linked to a specific application and language (Pascal's files of records, C++'s "struct" or "class" declarations, COBOL's data division)
- Data storage and retrieval must be coded explicitly in each application

2

- Example: management of student data in a university
  - ◇ student data needed by several applications (application, registration, finances, library management, grade reporting, ...)
  - ◇ without database technology, each application separately maintains data files and programs to manipulate those files
  - ◇ different formats can possibly coexist for the same data (e.g., length of names)
  - ◇ separate files can cause multiple (i.e., redundant) updates of the same data (e.g., to change an address)

Files versus Database

- A database is more self-descriptive on its content (schema) than files
- Data models capture more information and "know about" some inter-entity or inter-file relationships (e.g., by managing joins), while inter-file links must be coded in programs
- DBMSs offer a number of functions that need not be programmed in applications

3

- Capabilities of a modern DBMS include
  - maintain different views on the same data by different applications
  - control of multi-user access
  - distribution of data on different hardware and software platforms
  - access control
  - enforcement of integrity constraints
  - query processing and optimization
  - security and recovery

Problems with the File-Based Approach

- Tight integration between program and data files ⇒ lack of **data independence**:
  - ◇ file management is entirely application specific
  - ◇ physical storage structures visible in application code (e.g., ISAM, VSAM)
- **Labor intensive** (lower productivity of programmers)
  - ◇ no high-level query language
  - ◇ run-time performance can (and must) be programmed
  - ◇ DBMS functions must be programmed (consistency of redundant data in several files, backup, concurrency, etc)

4

- File-based data management is an **obsolete technology**, except for applications that do not require DBMS functions. Even for those problems, when efficiency is not crucial and size is not too big, it is often worth considering simple database software (like, e.g., ACCESS)
- File technology is also relevant when efficiency is very important and the operations on data relatively simple (e.g., movies on demand)

Hierarchical and Network Systems

- **Software systems without underlying organized discipline**
- **Lack of data independence**
  - ◇ traditional view of data structures as records (on disk storage) + links: mixture of physical and logical worlds
  - ◇ user view dependent on physical details irrelevant for information needs
  - ◇ modern view: **links = semantic relationships + physical access paths**
- **Navigational programming**
  - ◇ all database accesses imply procedural programming: no high-level query language
  - ◇ programming = "navigation" governed by physical access paths ⇒ long and complex programs
  - ◇ oriented towards one-record-at-a-time navigational programming ⇒ complex, necessarily-manual performance optimization

5

- The first general-purpose DBMSs were developed in the early 60's
  - ◇ based on a network (or graph-based) organisation, standardized as CODASYL ("Conference on Data Systems Languages"); examples of systems: IDS, IDMS, TOTAL, ADABAS, PHOLAS
  - ◇ hierarchical systems or tree-based; examples: IMS, System 2000
- At the time, they were unsystematic, adhoc; there was no real theoretical model about data organization (in particular, no notion of levels of data representation, of data independence, and no idea that nonprocedural data access was feasible)
- Early DBMSs encouraged users to view data much as it was stored
  - ◇ example of data dependency: physical location of fields within records visible in programs
  - ◇ other example of data dependency: pointers in the trees and graphs of hierarchies and networks
    - ∗ confusion between physical pointers (that provide efficient acces) and conceptual (or logical) pointers that express meaninful links between related pieces of information
    - ∗ the relational model established and demonstrated the virtues of data independence by suppressing pointers (an extreme solution!)
    - ∗ cleaner pointers came back with object technology

Relational Model

- Defined in 1970 (T. Codd)
- About 10 years of hesitation
- Robust RDBMSs built in the 80's
- **Historical compromise** between scientists, practitioners, vendors
- Most systems sold in the early 90's are relational

6

Novelty and Importance of the Relational Model

- **Simplicity** of basic concepts
- **Simple theoretical framework**
- Emergence of notion of **data model**
- **Data independence**: separate universe of application from that of implementation
- Emergence of **disciplined database design**
- Validation of a declarative, high-level approach to data management
- Construction of **powerful DBMS software**
- **New architectures**
- **Standards**

7

In short, the definition of the relational model and the advent of relational systems have founded modern database technology

- **Simplicity** of basic concepts: simple, abstract data structure, independent of physical considerations, sufficiently powerful and general as a realistic framework for numerous database applications in business data processing
- Simple basis for a **theoretical framework** that stimulated database research
  - ◇ **relational theory** based on well-established theories: mathematical set theory and first-order predicate logic
  - ◇ the DB field turns into a scientific discipline, the pre-relational era had drawn little interest from computer scientists
  - ◇ many interesting questions were raised for researchers and the field attracted a lot of good minds; progress was quick in the 70's
  - ◇ remark about theory in general: theoretical basis $\neq$ lack of practicality
    - ∗ theory = reliability, predictability
    - ∗ a strong formal underlying theory is **always** interesting
    - ∗ users need not master nor know about the whole theory
- Emergence of the notion of **data model**, quickly leading to semantic data models with more semantics than the relational model, and a clean a posteriori definition of network and hierarchic data models. Data models include the definition of **integrity** at the application-domain level, as a part of database design.
- **Data independence**: insulation of users from physical database design and access optimization; such a growing insulation of users from the lower layers of systems is a general trend in all informatics
- Emergence of a methodology for **disciplined database design**: a more semantic or conceptual view of information structures (e.g., entity-relationship) on top of the relational schema significantly simplifies the processes of databases analysis and design
- The design of **high-level interactive database languages**, based on set-at-a-time, nonprocedural operations, high-level operations (a first in the database world) demonstrated that a less procedural style (than that of programming with traditional programming languages) of data manipulation was not only practically feasible but also highly productive
- Construction of **efficient DBMS software**, in particular efficient query optimizers, transaction management for multiple users (concurrency control), reliability and failure recovery
- **New architectures**: database machines, client-server, distributed databases, client tools, 4GLs
- **SQL standard** = portability of applications

Relational Market

- Relational systems still dominate the user DBMS landscape
- Leading systems : Oracle, Informix (now disappeared), IBM (DB2), Sybase, Microsoft (SQL Server), MySQL
- Products of similar functionality and performance
- System performance, robustness have consistently strengthened (e.g., recovery, concurrency support, distribution)
- Numerous client tools (4GLs), users want to manage interfaces as forms, pictures, graphs, sounds, etc. not tables of numbers
- PC systems, competing with the larger DBMSs for lower-end applications
- Tool vendors have lost importance through extensions of main DBMS products

8

- Database market (Gartner, June 2003): Oracle (43 %), IBM (24 %), Microsoft (17 %)

Post-Relational Era

- **New organizations of data**
  - ◇ entity-relationship model
  - ◇ nonnormalized relations
  - ◇ object technology and object-relational systems
  - ◇ semi-structured and unstructured data (XML)
- **New functions**
  - ◇ distribution
  - ◇ heterogeneity (multidatabases, interoperability)
  - ◇ active databases (triggers) and deduction
  - ◇ ERP packages (application-oriented tasks common to many organizations)
  - ◇ data analysis (data warehouses and data mining)
- **More complex applications**: data-management technology enters application domains (e.g., design, geography, molecular biology, electronic commerce)

9

Evolution = Economics + Methodology

- **Economics**
  - ◇ name of the game 30 years ago: optimize the use of expensive hardware
  - ◇ analogous evolution in all domains of informatics (e.g., from machine language to high-level programming languages)
  - ◇ **human efficiency** has become the crucial quality factor
- **Methodological advances**
  - ◇ development of software engineering (after the diagnosis of "software crisis")
  - ◇ object technology
  - ◇ methods for software development (from structured methods to OO analysis and design)
  - ◇ continuous migration of program functions into DBMS functions

10

- **Economics**
  - ◇ reduction of hardware costs (memory, computing cycles, soon network bandwidth are essentially free at the margin)
  - ◇ 25 years ago, one hour of machine time ≈ one week of programmer time
  - ◇ today, one week of programmer time ≈ a personal computer
  - ◇ relational technology has provided huge overall productivity gains
- **Evolution of DBMS software**
  - ◇ development of efficient DBMS functions (e.g., relational query optimizers)
  - ◇ easier database programming
  - ◇ powerful and user-friendly front-end software (traditional database software is progressively becoming middleware)
- *If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost 100 USD, get one million miles to the gallon, and explode once a year, killing everybody inside.*

Weaknesses of Relational Technology

- Fixed collection of data types inadequate for complex data
- Poor conceptual modeling, e.g., of semantic relationships
- Separation between data and operations
- Insufficient transaction model
- Mismatch between SQL and usual programming languages
- Insufficient performance for data-analysis applications

11

Weaknesses of Relational Technology

- The first applications of DBMSs (those that relational technology addressed) had data composed of many small items, with a simple structure, and with many simple queries and modifications (e.g., airline reservation systems, banking systems, personnel management systems, sales and inventory management systems)
- Fixed collection of (built-in) data types (essentially alphanumeric) are inadequate for applications with complex data (e.g., design, WWW, multimedia, geography, biology)

- Poor conceptual modeling (e.g., semantic relationships like generalization, aggregation, materialization must be explicitly managed by user programs)
- Separation between data and operations (stored procedures are not integrated in the data model, there is no encapsulation)
- RDBMS have not fully exploited the relational model
  - ◇ the possibility of defining application-dependent domains has been reduced to using a fixed set of data types
  - ◇ a wider variety of integrity constraints could have been supported
- The transaction model of RDBMSs is inappropriate for long-duration transactions for interactive and cooperative design environments
- Application programming with RDBMSs is a combination of programs in usual procedural (tuple-at-a-time) programming languages and set-oriented queries in SQL: mismatch between SQL and traditional programming languages → 4GLs, OODBMSs
- The performance of RDBMS is insufficient for data-analysis applications (⇒ OLAP technology)

Contributions of OO Technology

- Objects can be used for both natural real-world modeling and software structuring
- Applications are less stable than data
- Extensible type system (i.e., domain-specific types) simplify application development
- Object identity helps object chaining and sharing
- Data abstraction, encapsulation, and modularity
  - ◇ some behavior naturally goes with data
  - ◇ stable public interface for objects
  - ◇ greater stability, easier evolution
- Generalization and inheritance, for reuse, easier adaptation
- Polymorphism, for simpler programming, easier evolution

Information-Management Context

- Relational technology "solved" business data processing (administrative data)
- Success of database technology ⇒ users demand it for domains other than business data processing
- Steady improvement in price/performance ratio of hardware and supporting architectures
- "Software crisis":
  - ◇ quality of application development is becoming a major concern
  - ◇ maintenance dominates overall development costs ⇒ reuse
  - ◇ application-development methodology is improving
- Open systems ⇒ variety of system architectures

13

Application Context

- Today's applications: objects + web + multimedia
- More complex data
  - ◇ time series, historical data (data warehouses, data mining ⇒ progress in decision-support systems)
  - ◇ multimedia (streaming audio and video)
  - ◇ weakly-structured data (XML)
  - ◇ complex data (e.g., molecular biology, geographic information systems)
- Web interfaces have made 4GLs and business forms obsolete
- Internet, emergence of electronic commerce

14

- Need for dynamic, interactive applications, with audio and video streams
- Convergence of data-modeling and document-management technologies:
  - ◇ originally, most web documents were manually composed (HTML)
  - ◇ DBMS increasingly include functions for making their contents displayable on the internet

Object Orientation for Database-Management Systems

- 2 approaches
  - ◇ **pure 00**: extend the concepts of object-oriented programming languages like Java or C++ to include persistence
  - ◇ **object-relational**: extend the relational model to include several common object-oriented concepts
- They differ in their answer to the question "how important is the relation?"
  - ◇ "not very" for the pure OO approach
  - ◇ "basically" for the object-relational approach

15

Object-Oriented Database-Management Systems (OODBMSs)

- Extension of object-oriented programming languages (OPLs) to management of persistent data
- Tight (seamless) integration between OPLs (C++, Smalltalk, Java) and DBMS
- OODBMSs manage (store, call, query) complex objects directly
- OQL language (extension of SQL)
- Performance thru caching large numbers of objects in address space of client program

16

Problems with OODBMSs

- Language-bound (closely tied to C++, Smalltalk, Java)
- Rediscover problems of tying database design close to application design
- Rediscover that declarative (SQL-like) approach is productive
- Lack of robustness for major DBMS functions for update-intensive OLTP applications (security, transaction processing, parallel processing)
- Vary widely in their syntax and functions (more than RDBMSs) in spite of ODMG standard

17

OODBMS Market

- Appeared in the early 90's, have now virtually disappeared
- Vendors: Object Design, Objectivity, Versant, Computer Associates (Jasmine)
- OODBMS have not displaced RDBMS and have not moved from niche markets
- Market development hindered by existing relational base and then by the development of object-relational systems

18

Object-Relational DBMSs

- Main goal: provide object capabilities while safeguarding RDBMS investments
- Technical goals
  - ◇ support of flexible and complex structures of the object world
  - ◇ offer query capabilities and performance of RDBMSs
- Differences with RDBMSs
  - ◇ extensible type system (application-dependent new types and functions)
  - ◇ better integration with programming languages than SQL-92
- Differences with OODBMSs
  - ◇ compatibility with RDBMSs and SQL

19

- RDBMS are evolving into object-relational systems
- RDBMS vendors have maintained their strong position (Oracle, Sybase, IBM, Microsoft)
- Roughly similar products
- Big business!

Problems with ORDBMS

- Hybrid approach
- Technological integration without new concepts
- Weak on inheritance and polymorphism
- No sound theoretical foundations

20

Web Databases

- The first web sites stored their data in operating systems files
- Now web pages are dynamically built from data queried from DBMSs
- Semi-structured data model
  - ◇ addresses the need to combine databases with other data sources (like web pages)
  - ◇ relax the necessity of a fixed schema for database data

21

Future of Database Management

- The quantity of available data is increasing enormously
- Networking drives multimedia databases, streaming audio and video, interactive video, huge digital libraries
- Applications with complex data structures (human genome, geographic databases)
- Exploit the available data (data warehousing and data mining for decision support)