

Three types of information systems:

- Information-Retrieval Systems (IR)
 - Search large bodies of information which are not specifically formatted as formal data bases.
 - Web search engine
 - Keyword search of a text base
 - Typically read-only
- Database Management Systems (DBMS)
 - Relatively small schema
 - Large body of homogeneous data
 - Minor or no deductive capability
 - Extensive formal update capability
 - Shared use for both read and write
- Knowledge-Base Systems (KBS)
 - Relatively small body of heterogeneous information
 - Significant deductive capability
 - Typical use: support of an intelligent application.

Key DBMS issues:

- Efficiency issues:
 - Databases can be very large. Efficient access must be provided despite the size.
- Simplicity issues:
 - Many potential users are not sophisticated programmers, and so simple means of access must be available.
 - Means of more sophisticated access must also be available.
- Multi-user issues:
 - Concurrency
 - Several users may have simultaneous access to the database.
 - Access via views
 - Each user has a limited “window” through which the appropriate part of the database is viewed.
 - Authorization
 - The access privileges of each user will be limited in a specific way.
- Robustness issues:
 - Deadlock must be avoided.
 - A means of recovery from crashes, with minimal loss of data, must be available.

Data Model Evolution:

Model	Devel.	Use	Properties	Analogy
File management	1950's – 1970's	1950's-	Low-level interaction. No data independence.	Assembly language
Navigational models	1950's – 1960's	1960's -	Some data independence, but the model invites dependence. Requires procedural queries.	Procedural languages
Relational model	1970's -	Late 1980's -	Simple, easy to use for non-experts. Strong data independence. Standard nonprocedural query language (SQL). Excellent implementations exist. Limited expressive capability.	Declarative languages
Object-oriented models	1980's -	1990's -	Powerful expressive capability, but require substantial expertise for use. Popular in niche applications. Standardization not imminent.	Object-oriented languages
Object-relational models	1990's	1990's -	Attempt to integrate the simplicity of the relational model with the advanced features of the object-oriented approach. A new standardized query language (SQL:1999) is available, with SQL:20xx on the way. Many “high-end” commercial relational systems embody object-relational features.	?
Semi-structured models	1990's	2000's -	Attempt to integrate data management with markup languages, principally via XML.	?

The course focuses on the relational model. Why?

- The relational model is very widely used.
- The relational model provides a flexible interface which has components appropriate for users at all levels.
- A standard query language, SQL, is used with virtually all commercial products. Thus, applications have a high degree of portability.
- The relational model provides strong data independence: the external product is relatively independent of the internal implementation.
- The relational model is dominant on single-user systems for microcomputers:
 - Multi platform FLOSS:
 - OpenOffice.org: Base
 - Kexi (free (as in beer) for Linux only)
 - MS Windows only \$\$\$:
 - Lotus SmartSuite: Approach
 - Microsoft Office: Access
 - dBase
- All have graphical interfaces, and provide programming-style queries as well.

- The relational model has also been dominant for multi-user database servers, including but not limited to UNIX systems.
- These system support concurrency.
- FLOSS systems:
 - PostgreSQL (used in this course)
 - MySQL (Now owned by Sun → Oracle)
 - MySQL Maria Engine (independent fork)
- Other systems which run under Linux/Unix (some with limited or academic versions at no cost, some multi platform):
 - Oracle
 - Interbase (Embarcadero)
 - Sybase
 - IBM Informix
 - IBM DB2
 - Mimer SQL (Uppsala)
- Other notable systems:
 - Microsoft SQL Server

In the past, this course had used Microsoft Access.

Since 2002, PostgreSQL has been used.

Why?

- The dialect of SQL which is supported under Access is much more limited than the dialects of comprehensive systems.
- PostgreSQL has matured greatly in the past decade.
- The Department of Computing Science has an SQL server, which is administered by the support staff.
- The course instructor is a strong advocate of FLOSS (Free/Libre and Open-Source Software).

The following system will also be used:

- Leap
 - A simple relational database system which uses the *relational algebra* as a query language.
 - Although not of commercial importance, use of this alternate query language is very beneficial pedagogically.
- Students are still free to use Microsoft Access, although it will not be discussed in class.
- All final versions of SQL assignments must run under PostgreSQL.

Database access models:

- SQL is the standard query language for the relational model.
- There are many *access models* which are built around SQL.
 - *Direct SQL*: Write and send SQL queries directly to the database system.
 - *Hosting SQL* within a programming language:
 - *Embedded SQL*: SQL statements are embedded in a host programming language, such as C. Generally requires preprocessing.
 - *Proprietary hosting languages*: (e.g., Oracle PL/SQL).
 - *Proprietary hosting systems*: (e.g., within Microsoft VBA).
 - *SQL / CLI ODBC*: A vendor- and OS-independent call-interface system (in principle) for SQL. Embedding may be in any of a variety of languages (C, C++ are the most common.)
- In this course, we will use both direct SQL and ODBC.

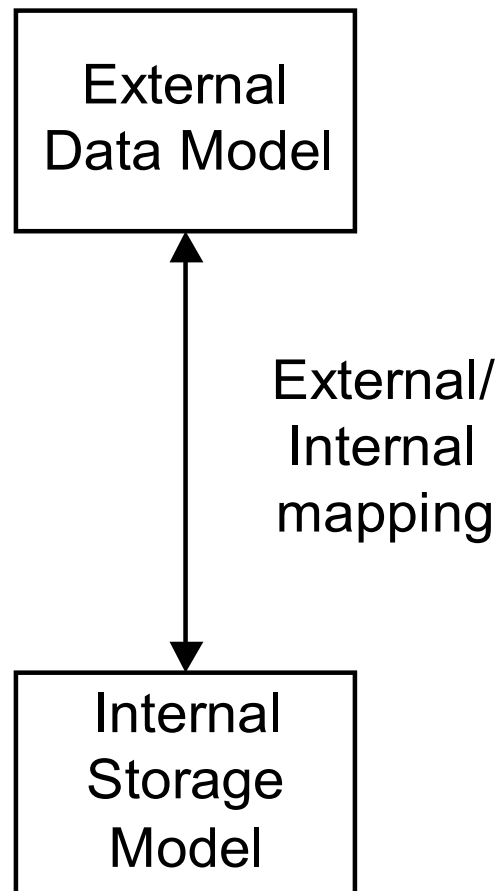
A Rough Course Outline:

- Introduction to DBMS's
- Knowledge Representation for DBMS's (10%)
 - Entity-relationship modelling
 - The relational model
- Query Processing and Constraints (45%)
 - Pure query languages
 - Relational algebra
 - Relational calculus
 - SQL
 - Basic use
 - Views
 - Database programming and the CLI/ODBC interface
 - Dependencies and normalization
- Implementation Issues (35%)
 - Physical database design
 - Database system architecture
 - Query optimization
 - Transaction processing and concurrency control
 - recovery
 - Security and authorization
- Special Topics (5%)
 - Object-oriented and object-relational approaches

Database System Architecture:

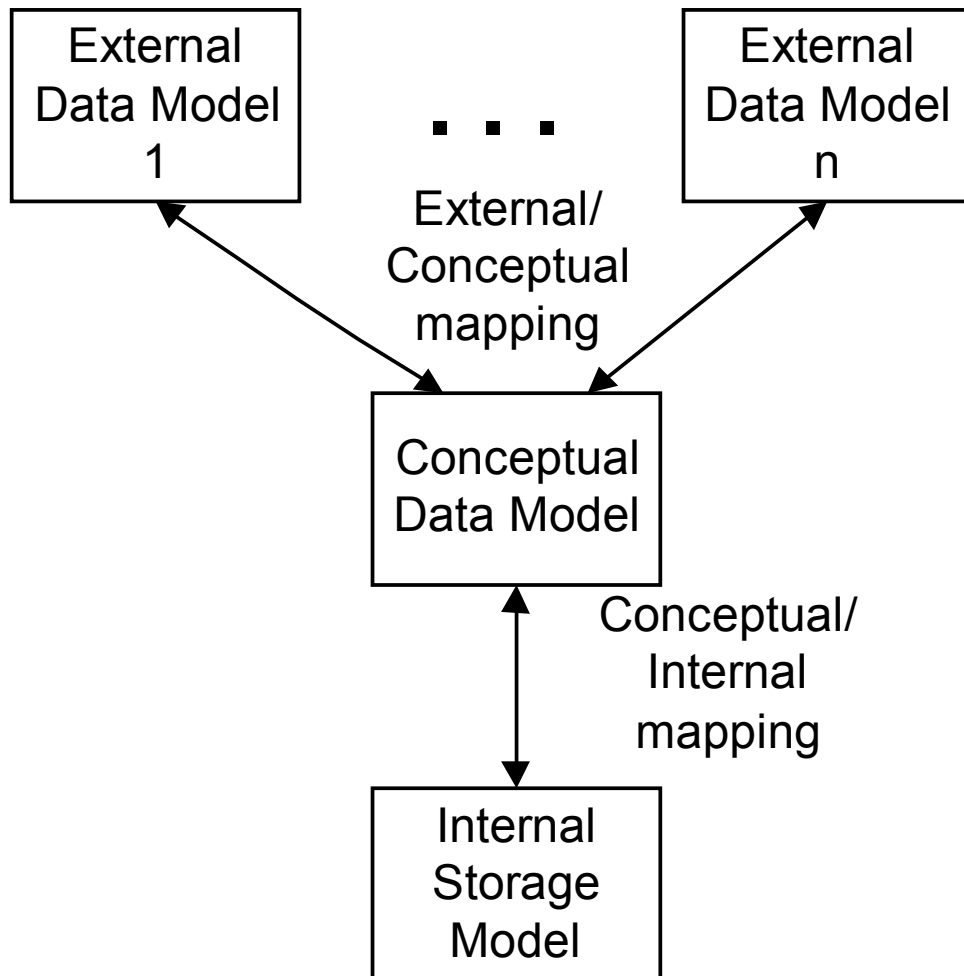
- Early approach: one-level
- The user interacted directly with the storage model.
- Analogy: assembly-language programming
- Disadvantages:
- Impossible to use for non-experts.
- Difficult to use and error-prone even for experts.
- Evolution of storage model, or migration to a new architecture, requires a total rebuild of all application programs.

A more modern approach: two-level



- Advantages:
 - Internal model and/or target architecture may be changed without requiring a rebuild of applications.
 - Analogy: A high-level programming language.
- Disadvantages:
 - There is a single external model for all.

The ANSI/SPARC three-level architecture:



- Advantages:
 - Provides two levels of independence:
 - The internal storage model is isolated from the conceptual component, as in the two-level architecture.
 - Many external views are possible.
 - The conceptual model may be re-designed without requiring rebuilds of application programs.

Data independence:

- Data independence refers to the idea that a more internal level of a database system may be re-engineered, or moved to a different architecture, without requiring a total rebuild of the more external layers.
- The ANSI/SPARC architecture provides two levels of data independence.
- It is often, however, something of an ideal, even with the systems of today.
- Usually, in a relational system, both the conceptual schema and the external schemata are relational.
- Still, the conceptual schema is often *designed* using a more general tool than the relational model.

A Note on Terminology:

- A *database* is a (usually structured) collection of data.
- A *database system or database management system (DBMS)* is a system for managing databases.
- Sometimes the word *database* is used as a synonym for *DBMS* but this is not correct.
- MySQL and PostgreSQL are DBMSs, not databases!
- Calling a DBMS a database is like calling JDK a Java program.