Fundamentals of Database Systems
COMPSCI 351

Instructor: Sebastian Link

The University of Auckland

Introduction

Databases

Definition (Database)

A **database** is a very large, integrated and commonly used collection of data

Databases

Definition (Database)

A **database** is a very large, integrated and commonly used collection of data

Example (Databases model real-world enterprises)

- entities (for example, Rihanna, COMPSCI351)
- relationships (for example, Rihanna is taking COMPSCI351)

Database (Management) Systems

Definition (Database Management System)

A **Database Management System** (DBMS) is a

- software package designed to
- store and manage databases

Database (Management) Systems

Definition (Database Management System)

A **Database Management System** (DBMS) is a

- software package designed to
- store and manage databases

Definition (Database System)

A **Database System** (DB) is a DBMS together with a database

Files vs DBMS

- Applications must handle large data sets
  - between main memory and secondary storage
  - for example, buffering, page-oriented access, 32/64-bit addressing, etc.

Files vs DBMS

- Applications must handle large data sets
  - between main memory and secondary storage
  - for example, buffering, page-oriented access, 32/64-bit addressing, etc.
- Special code for different queries

Files vs DBMS

- Applications must handle large data sets
  - between main memory and secondary storage
  - for example, buffering, page-oriented access, 32/64-bit addressing, etc.
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users

Files vs DBMS

- Applications must handle large data sets
  - between main memory and secondary storage
  - for example, buffering, page-oriented access, 32/64-bit addressing, etc.
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery

Files vs DBMS

- Applications must handle large data sets
  - between main memory and secondary storage
  - for example, buffering, page-oriented access, 32/64-bit addressing, etc.
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Why Use a DBMS?

- Data independence

Why Use a DBMS?

- Data independence
- Reduced application development time

Why Use a DBMS?

- Data independence
- Reduced application development time
- Data integrity and security

Why Use a DBMS?

- Data independence
- Reduced application development time
- Data integrity and security
- Uniform data administration

Why Use a DBMS?

- Data independence
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access

Why Use a DBMS?

- Data independence
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access
- Recovery from crashes

Why Use a DBMS?

- Data independence
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access
- Recovery from crashes
- Most important factors in designing DBMSs and databases:
  - Efficient access to data
  - Efficient updates of data
  - Usually, there are trade-offs between the two goals

Why Study Databases?

Shift from *computation* to *information*

- Always been true for corporate computing
- The Web made this point for personal computing
- More so true for scientific computing

Why Study Databases?

Shift from *computation* to *information*

- Always been true for corporate computing
- The Web made this point for personal computing
- More so true for scientific computing

Big Data (Volume, Variety, Velocity, Veracity)

- Digital libraries, life science projects
- Multimedia databases, social web (Facebook, Twitter, YouTube)
- Semantic annotations, sensor data, uncertain data

Why Study Databases?

Shift from *computation* to *information*

- Always been true for corporate computing
- The Web made this point for personal computing
- More so true for scientific computing

Big Data (Volume, Variety, Velocity, Veracity)

- Digital libraries, life science projects
- Multimedia databases, social web (Facebook, Twitter, YouTube)
- Semantic annotations, sensor data, uncertain data

DBMS: CompSci as a practical discipline

- AI, languages, logic, multimedia, OS, theory
- Yet traditional focus on real-world applications

Why Study Databases?

Shift from *computation* to *information*

- Always been true for corporate computing
- The Web made this point for personal computing
- More so true for scientific computing

Big Data (Volume, Variety, Velocity, Veracity)

- Digital libraries, life science projects
- Multimedia databases, social web (Facebook, Twitter, YouTube)
- Semantic annotations, sensor data, uncertain data

Need for DBMS exploding

DBMS: CompSci as a practical discipline

- AI, languages, logic, multimedia, OS, theory
- Yet traditional focus on real-world applications

Data Models

Definition (Data model)

A collection of concepts for describing data

Data Models

Definition (Data model)

A collection of concepts for describing data

Definition (Schema)

An abstract description of databases using the given data model

Data Models

Definition (Data model)

A collection of concepts for describing data

Definition (Schema)

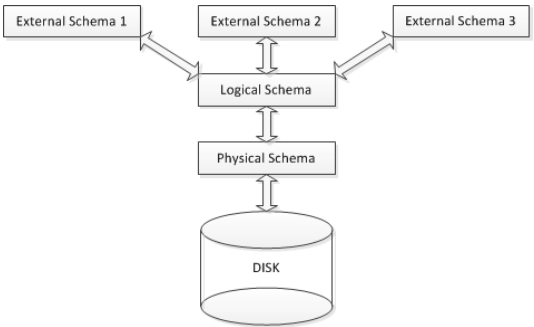
An abstract description of databases using the given data model

Example (Relational model of data)

- Most widely used model today
- Main concept: relation, basically a table with rows and columns
- Every relation has a schema, which describes the columns, or fields

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

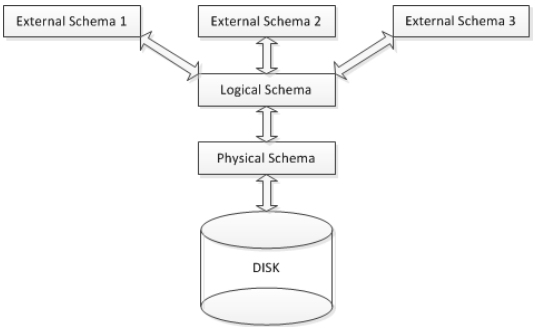


Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

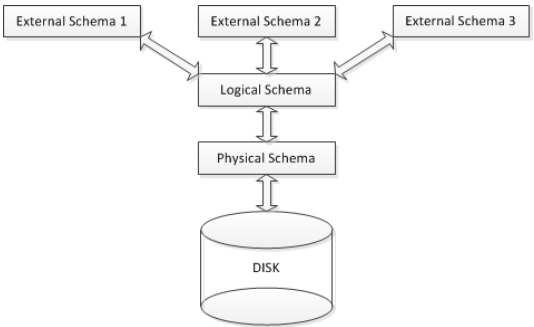
Views

describe how users see the data



Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

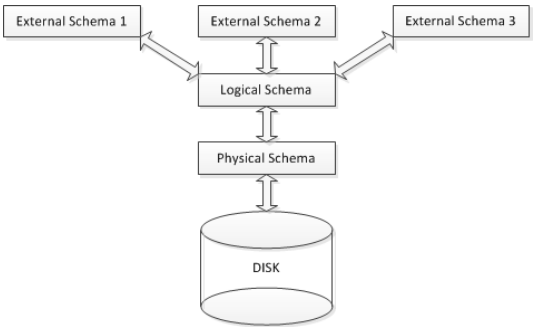


Logical schema

defines logical structure

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

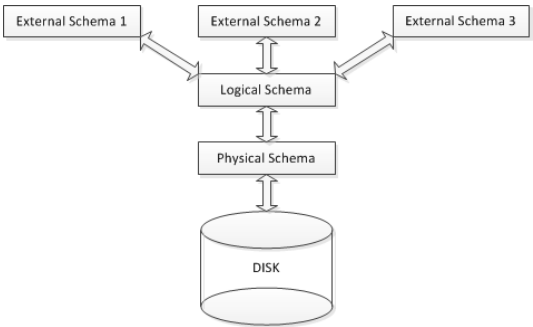


Physical schema

describes the files and indexes used

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

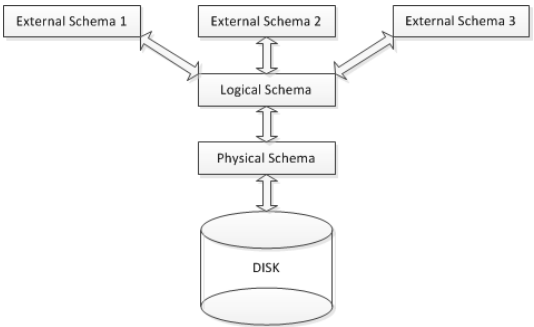


Data Definition Language (DDL)

defines database schema

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

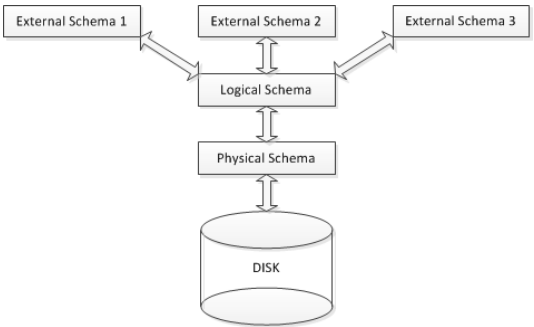


Data Manipulation Language (DML)

update data in database (insert, delete, modify)

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema



Query Language (QL)

access and retrieve data from database

Levels of Abstraction

Many views (external schemata), single logical schema and single physical schema

Views

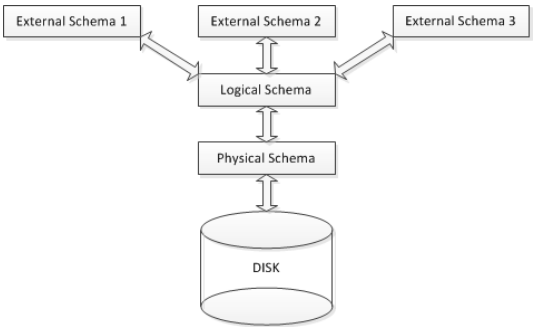
describe how users see the data

Logical schema

defines logical structure

Physical schema

describes the files and indexes used



Data Definition Language (DDL)

defines database schema

Data Manipulation Language (DML)

update data in database (insert, delete, modify)

Query Language (QL)

access and retrieve data from database

Example: University Database

Logical schema

- *Students*(sid:`string`, name:`string`, login:`string`, gpa:`real`)
- *Courses*(cid:`string`, cname:`string`, credits:`integer`)
- *Enrolled*(sid:`string`, cid:`string`, grade:`string`)

Example: University Database

External Schema (View)

- *Course_enrollment*(cid:`string`,enrollment:`integer`)
- *Course_averages*(cid:`string`,average_grade:`real`)

Logical schema

- *Students*(sid:`string`, name:`string`, login:`string`, gpa:`real`)
- *Courses*(cid:`string`, cname:`string`, credits:`integer`)
- *Enrolled*(sid:`string`, cid:`string`, grade:`string`)

Example: University Database

External Schema (View)

- *Course_enrollment*(cid:`string`,enrollment:`integer`)
- *Course_averages*(cid:`string`,average_grade:`real`)

Logical schema

- *Students*(sid:`string`, name:`string`, login:`string`, gpa:`real`)
- *Courses*(cid:`string`, cname:`string`, credits:`integer`)
- *Enrolled*(sid:`string`, cid:`string`, grade:`string`)

Physical schema

- Relations stored as unordered files
- Index on first column of *Students*, first of *Courses*, first two of *Enrolled*

Data Independence: A key benefit of using DBMSs

Applications insulated from how data is structured and stored

Data Independence: A key benefit of using DBMSs

Applications insulated from how data is structured and stored

Logical data independence

- external handling of data nearly independent from logical organization
- changes to the logical schema shall not affect the external schema
- only mapping shall be changed
- application shall only see the external schema (impossible in general)

Data Independence: A key benefit of using DBMSs

Applications insulated from how data is structured and stored

Logical data independence

- external handling of data nearly independent from logical organization
- changes to the logical schema shall not affect the external schema
- only mapping shall be changed
- application shall only see the external schema (impossible in general)

Physical data independence

- physical organization of data nearly independent from logical organization
- changes to physical schema have no implications on the logical layer
- can abstract from the realization of the DBMS storage organization
- can reason about data without worrying about physical realization
- can perform physical optimization / tuning

Concurrency Control

Concurrent execution of user programs is essential for good DBMS performance and business survival

Concurrency Control

Concurrent execution of user programs is essential for good DBMS performance and business survival

Disk accesses frequent, and relatively slow

important to keep CPU humming by working on several user programs concurrently

Concurrency Control

Concurrent execution of user programs is essential for good DBMS performance and business survival

Disk accesses frequent, and relatively slow

important to keep CPU humming by working on several user programs concurrently

Interleaving actions of different user programs can lead to inconsistency

e.g., cheque is cleared while account balance is being computed

Concurrency Control

Concurrent execution of user programs is essential for good DBMS performance and business survival

Disk accesses frequent, and relatively slow

important to keep CPU humming by working on several user programs concurrently

Interleaving actions of different user programs can lead to inconsistency

e.g., cheque is cleared while account balance is being computed

DBMS ensures such problems do not arise

users can pretend they are using a single-user system

Databases Make These Folks Happy...

End users

get answers to questions they ask

Databases Make These Folks Happy...

End users

get answers to questions they ask

DBMS vendors

see their products in use

Databases Make These Folks Happy...

End users

get answers to questions they ask

DBMS vendors

see their products in use

Application programmers

can retrieve and roll out information as required by application

Databases Make These Folks Happy...

End users

get answers to questions they ask

DBMS vendors

see their products in use

Application programmers

can retrieve and roll out information as required by application

Database administrators (DBA)

- Designs logical /physical schemas
- Handles security and authorization
- Data availability, crash recovery
- Database tuning as needs evolve

Databases Make These Folks Happy...

End users

get answers to questions they ask

DBMS vendors

see their products in use

Application programmers

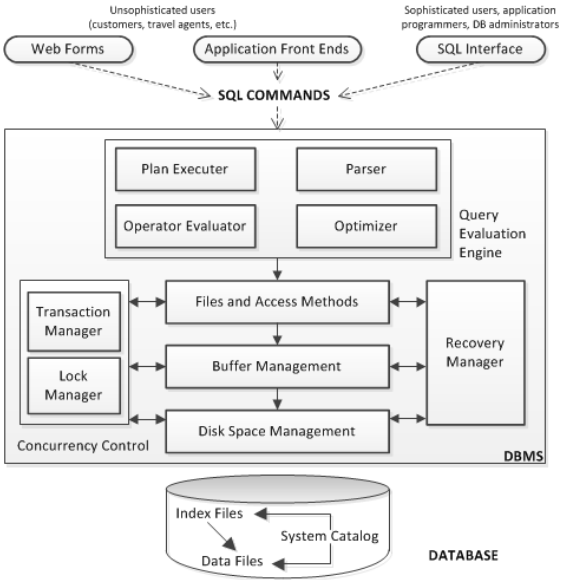
can retrieve and roll out information as required by application

Database administrators (DBA)

- Designs logical /physical schemas
- Handles security and authorization
- Data availability, crash recovery
- Database tuning as needs evolve

Understanding how a DBMS works is key

Layered Architecture of a DBMS



Historical Perspective

- First DBMS was built in the early 1960s
  - based on the network data model

Historical Perspective

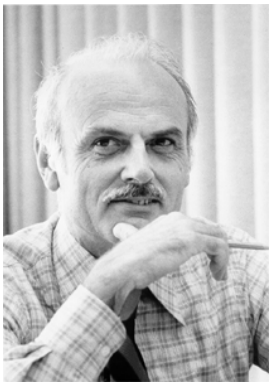
- First DBMS was built in the early 1960s
  - based on the network data model
- IBM developed IMS DBMS in late 1960s
  - based on the hierarchical data model

Historical Perspective

- First DBMS was built in the early 1960s
  - based on the network data model
- IBM developed IMS DBMS in late 1960s
  - based on the hierarchical data model
- SABRE, an airline reservation system
  - developed by American Airlines and IBM at same time

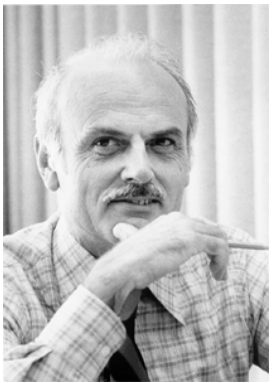
Historical Perspective

- First DBMS was built in the early 1960s
  - based on the network data model
- IBM developed IMS DBMS in late 1960s
  - based on the hierarchical data model
- SABRE, an airline reservation system
  - developed by American Airlines and IBM at same time
- Edgar Codd proposed the relational model in 1970s
  - providing a firm foundation for databases
  - won him the Turing Award in 1981



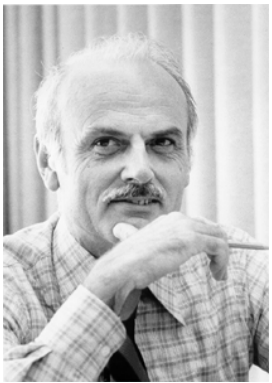
Historical Perspective

- First DBMS was built in the early 1960s
  - based on the network data model
- IBM developed IMS DBMS in late 1960s
  - based on the hierarchical data model
- SABRE, an airline reservation system
  - developed by American Airlines and IBM at same time
- Edgar Codd proposed the relational model in 1970s
  - providing a firm foundation for databases
  - won him the Turing Award in 1981
- 1980s: SQL developed as part of IBMs system R project



Historical Perspective

- First DBMS was built in the early 1960s
  - based on the network data model
- IBM developed IMS DBMS in late 1960s
  - based on the hierarchical data model
- SABRE, an airline reservation system
  - developed by American Airlines and IBM at same time
- Edgar Codd proposed the relational model in 1970s
  - providing a firm foundation for databases
  - won him the Turing Award in 1981
- 1980s: SQL developed as part of IBMs system R project
- 1990s: powerful query languages, richer data models (for example, to include images and text)



Historical Perspective Continued

- Emergence of Enterprise Resource Planning (ERP) and Material Requirements Planning (MRP) packages
  - add a substantial layer on top of usually a relational DBMS
  - e.g., Baan, Oracle, PeopleSoft, SAP, Siebel

Historical Perspective Continued

- Emergence of Enterprise Resource Planning (ERP) and Material Requirements Planning (MRP) packages
  - add a substantial layer on top of usually a relational DBMS
  - e.g., Baan, Oracle, PeopleSoft, SAP, Siebel
- Web data:
  - originally stored in files,
  - now more common to have an underlying database,
  - DBMS vendors are making DBMSs more web friendly

Historical Perspective Continued

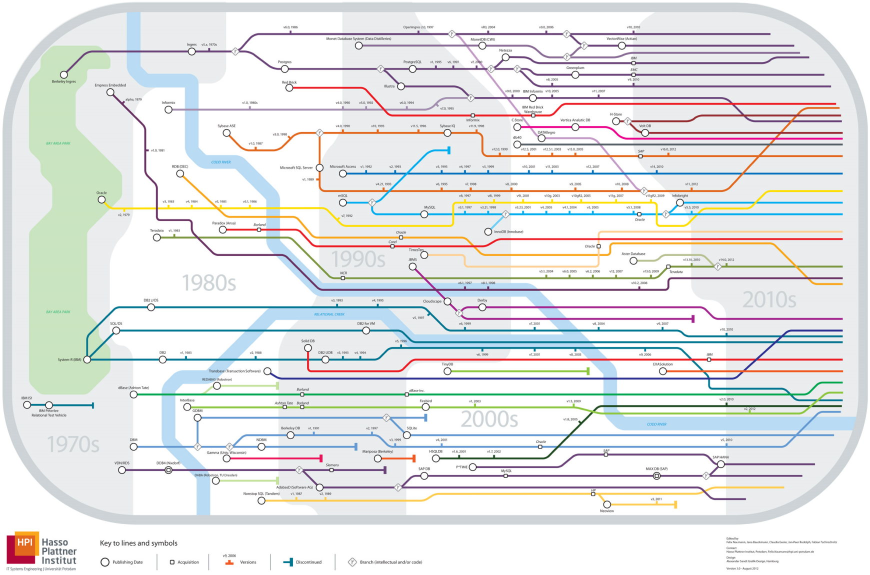
- Emergence of Enterprise Resource Planning (ERP) and Material Requirements Planning (MRP) packages
  - add a substantial layer on top of usually a relational DBMS
  - e.g., Baan, Oracle, PeopleSoft, SAP, Siebel
- Web data:
  - originally stored in files,
  - now more common to have an underlying database,
  - DBMS vendors are making DBMSs more web friendly
- Today field driven by e-commerce, multimedia databases, digital libraries, life sciences, Web X.0, consolidation of decision making processes and mining of massive data for useful business information

Historical Perspective Continued

- Emergence of Enterprise Resource Planning (ERP) and Material Requirements Planning (MRP) packages
  - add a substantial layer on top of usually a relational DBMS
  - e.g., Baan, Oracle, PeopleSoft, SAP, Siebel
- Web data:
  - originally stored in files,
  - now more common to have an underlying database,
  - DBMS vendors are making DBMSs more web friendly
- Today field driven by e-commerce, multimedia databases, digital libraries, life sciences, Web X.0, consolidation of decision making processes and mining of massive data for useful business information

> Relational databases may not be hot or sexy but for your important data, there is no substitute.
>
> Sean Doherty, Vice-President of EnterpriseDB

Genealogy of Relational Database Management Systems



Summary

- Value of data to an organization widely recognized
- Amount and type of data used is exploding
- Data can become a liability rather than an asset if not managed
- DBMS used to maintain and query large data sets
- Benefits include data integrity and security, quick application development, concurrent access and recovery from system crashes
- Overriding factors in database design are efficient access to and updates of data
- Levels of abstraction give data independence
- A DBMS typically has a layered architecture
- DBAs hold responsible jobs and are well-paid
- DBMS R&D broad and exciting with deep interplay between theory and practice