# Database Theory

Introduction

# DBMS - Introduction

- Database Management System (DBMS)
  - “Powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely” [UW:01]
- DBMS capabilities
  - Persistent storage
    - Large amount of data; exists independent of any processes
  - Programming interface
    - Easy access and modifications through a *query language*
  - Transaction management
    - Isolation, atomicity, and durability

# DBMS - Introduction

- Functionalities
  - Allows users to create new databases
    - Specification of a schema (logical structure of data)
      - Data-definition language (DDL)
  - Allows users to query data
    - Query and modify the data
      - Query language and data-manipulation language (DML), e.g. SQL
  - Support the storage of very large amounts of data
    - Secure it from accident or unauthorized use
  - Controls access to data from many users at once
    - Prevent to corrupt the data accidentally

# DBMS - Introduction

- DBMS vs. File System
  - Conventional *file systems* are inadequate as database systems!
  - Why?
  - They fail to support:
    - Guarantee that data cannot be lost
    - Efficient search and complex queries
    - Efficient modifications to small pieces of data
    - Create a schema (only directory structures)
    - Atomic and independent execution of transactions (concurrent access)

# Early DBMS

- Old DBMSs: IMS (hierarchy-based) or CODASYL (graph-based)
- Example: **Airline Reservation Systems**
  - Data
    - Reservations (flight no, seat, meal preference, …)
    - Flights (departure, destination, departure/arrival times, etc.)
    - Prices
  - Queries
    - Flights on a given date, time, city, seats available, prices
  - Modifications
    - Booking a flight, assigning a seat, …
  - Requirements
    - Concurrent access by many agents (make sure that no two agents assign the same seat simultaneously)
    - Avoid loss in case of a sudden system failure

# Relational Databases

- Based on the relational model
  - [E.F. Codd. A relational model for large shared data banks. Comm. ACM, 13:6, pp. 377-387, 1971]
- Data is organized as tables, called *relations*

Enrollment

| student | course | semester |
|---|---|---|
| Sjöberg | DA2032 | Fall 2007 |
| Jusufi | DA4104 | Spring 2008 |
| ... | ... | ... |

  - Columns: *attributes*
  - Rows: *tuples*
- Separates the logical view from the physical view (storage structure) of the data

# Relational Databases

- Querying a Database
  - Find all students who have taken DA2032 in Fall 2007
  - Structured Query Language – SQL

```
SELECT  student
FROM  Enrollment
WHERE  course = 'DA2032'  AND  semester = 'Fall 2007'
```

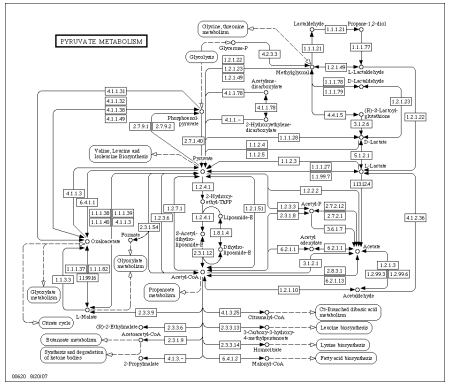
    - Answer: Sjöberg, Arlefur, Roos, …
  - So-called *Query Processor* figures out how to answer the query efficiently!

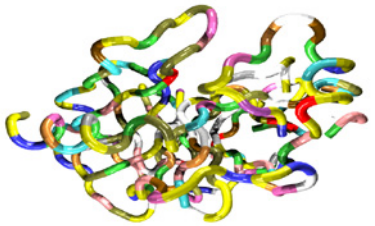
# DB in real world

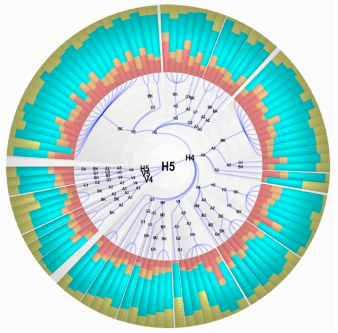
- Relational databases are a great success of theoretical ideas
- Most important DBMS companies are among the largest software companies in the world
  - Oracle
  - IBM (with DB2)
  - Microsoft (with SQL Server, Microsoft Access)
  - …
- Open-source DBMSs are similarly important
  - MySQL, PostgreSQL, …
- Rise of NOSQL databases
  - MongoDB, Neo4J, …

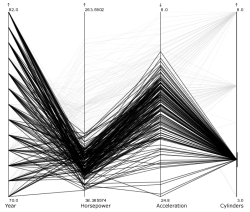
# Database Industry and Usage

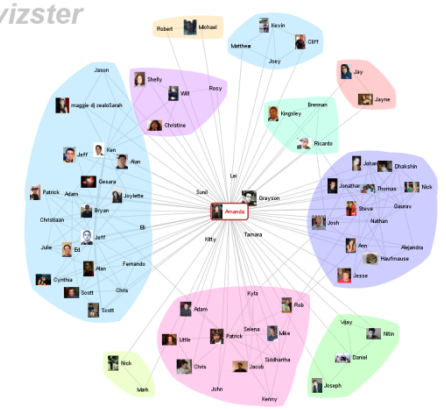
- Prominent application domains of DBMS
  - Information services on the web, administration, …
  - Bioinformatics
    - KEGG (biochemical pathways, …)
    - UniProt (protein sequences, …)
    - …
  - Source for Visualization and Visual Analytics
    - Experimental data
    - Statistical data
    - Homeland Security
    - …







# Evolution

- Smaller and smaller systems
  - In the past, DBMSs could only run on large computers (only large computers could store gigabytes of data)
  - Today, feasible to run a DBMS on a PC
- Bigger and bigger systems
  - Many applications need hundreds of gigabyte
    - Example: A retail chain may store terabytes ($10^{12}$)
  - Newer databases store images, audio, video, etc.
    - Example: Databases for satellite images store petabytes ($10^{15}$)
  - Modest size databases stored in *Secondary Storage* (arrays of disks)

# Evolution

- Dealing with large data
  - What distinguishes database systems from other software?
    - Database systems assume that data is too big to fit in main memory (*Primary Storage*) → data must be located on disk!
  - Today, we need sometimes more than disks!
  - *Tertiary Storage*
    - Capacity of many terabytes
    - Access takes seconds (ms for disks)
    - CD's, DVD's, etc. may be the storage medium
  - *Parallel Computing*
    - Satisfies the need of fast access to large data
    - Speeds up through parallel access to disks/devices
    - Needs efficient algorithms to intelligently break queries up

# Evolution

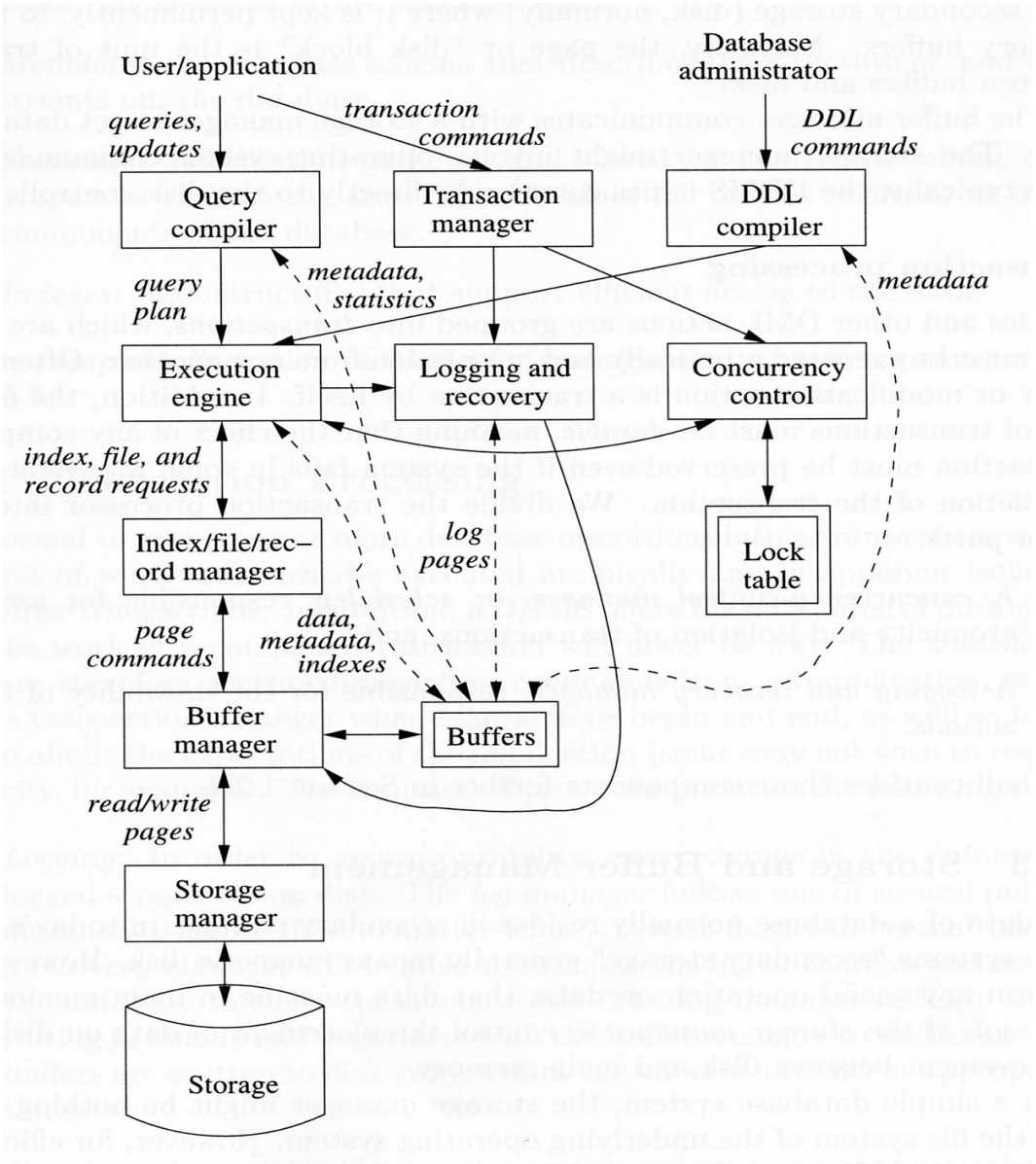
- Client-Server and Multi-Tier Architectures
  - Major database components are on the server, and the client is used to interface with the user
  - In multi-tier architectures, the client is an *application server,* which manages connections to the database, transactions, authorization, etc.
  - Several clients (e.g., web server) → application server → database server
- Multimedia Data
  - Video, audio, radar signals, satellite images, etc.
  - Challenges
    - Operations (queries, …) are not simple
    - Users should be able to introduce new functions (object-oriented DBMSs)
    - Query answers can be huge. DBMS must handle such cases

# Evolution

- Information Integration
  - To support information integration, a system should provide a unified view to the database user even if data comes from different sources, for example, different DBMSs of several subsidiary companies
  - Such sources, called *legacy databases*, often use different structures to represent information → We need a kind of meta-structure!
  - *Data Warehouses (DWs)* copy information from many legacy databases with an appropriate translation to a central database
    - There are several update strategies
    - DWs are reconstructed each night to reflect updates on legacy databases
- Among other things, this leads to the need of *data mining* techniques, i.e., the search for interesting and/or unusual patterns in data

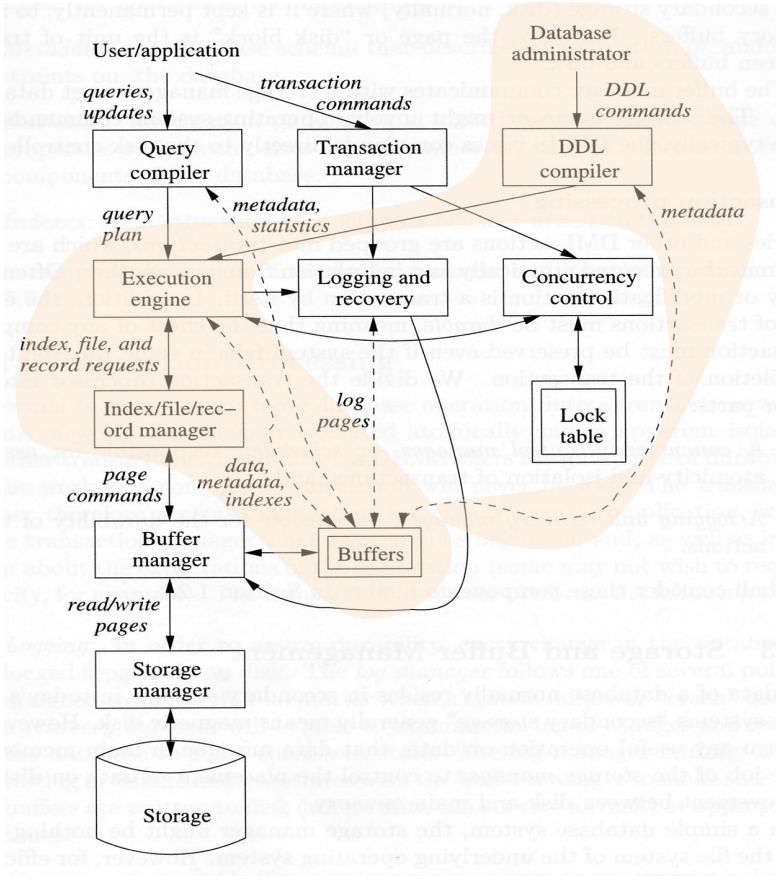
# Overview of DBMS

- DBMS Components
  - = system components
  - = data structures
  - = control & data flow
  - = data flow only



# DDL Storage and Buffer

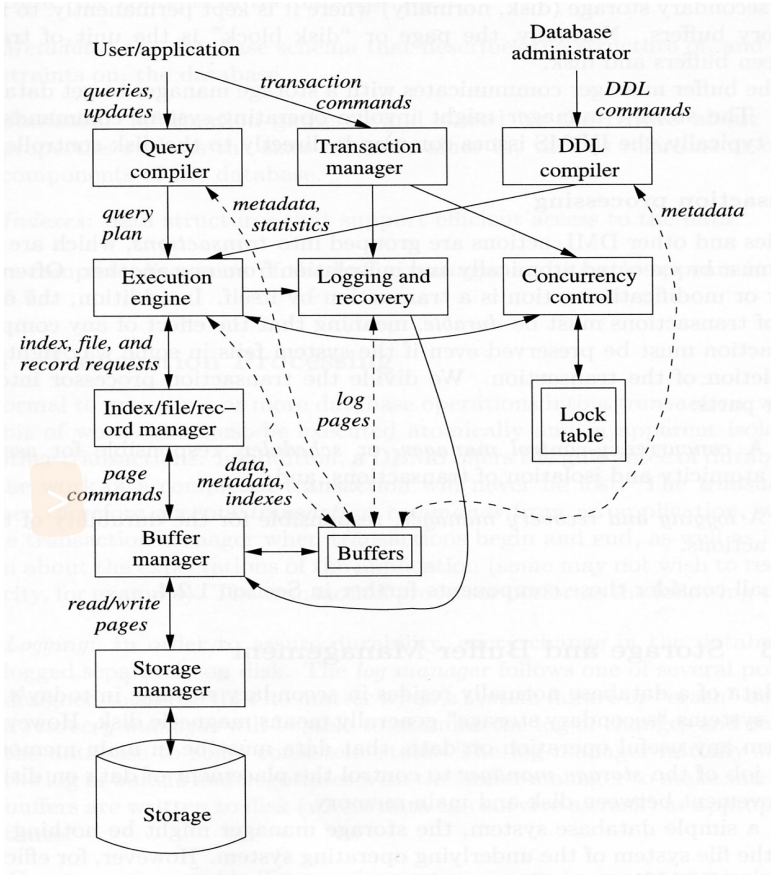
- DDL: Data Definition Language
  - Capable of altering the metadata (schema or structure/constraint information of the database)
  - Requires special authority (i.e., used by database administrator)



# DDL Storage and Buffer

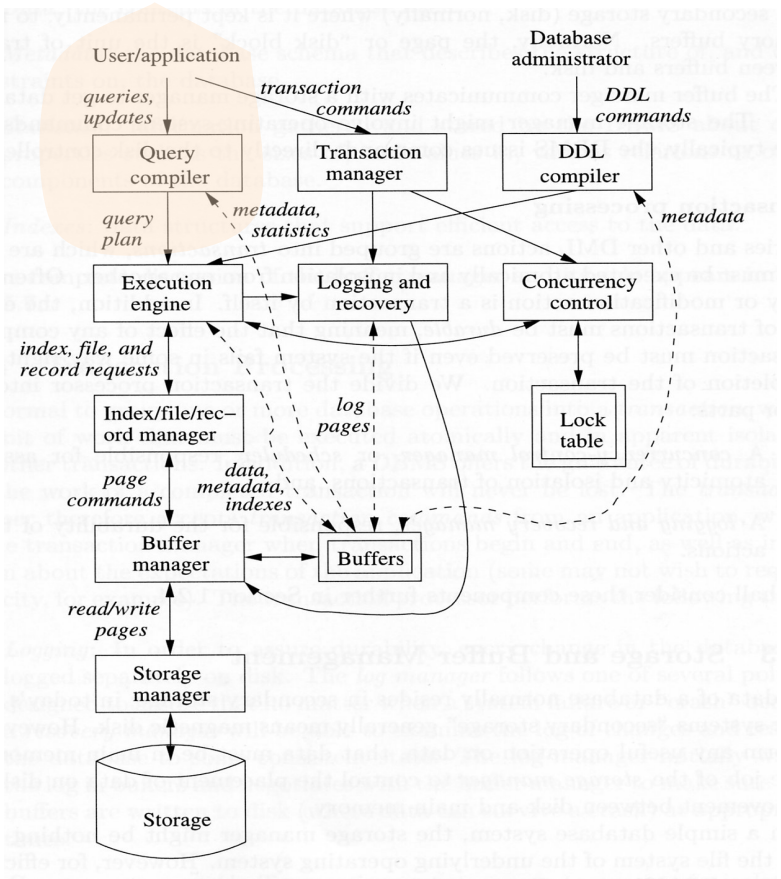
- DDL: Data Definition Language
  - Capable of altering the metadata (schema or structure/constraint information of the database)
  - Requires special authority (i.e., used by database administrator)

- Storage and Buffer Management
  - Storage manager controls the location of data on the disk and obtains the blocks containing a file on request from the buffer manager (BM)
  - BM partitions the available main memory into buffers (pages ↔ disk blocks)
  - Information that various components include: data, metadata, indexes (data structures that support efficient access to the data)



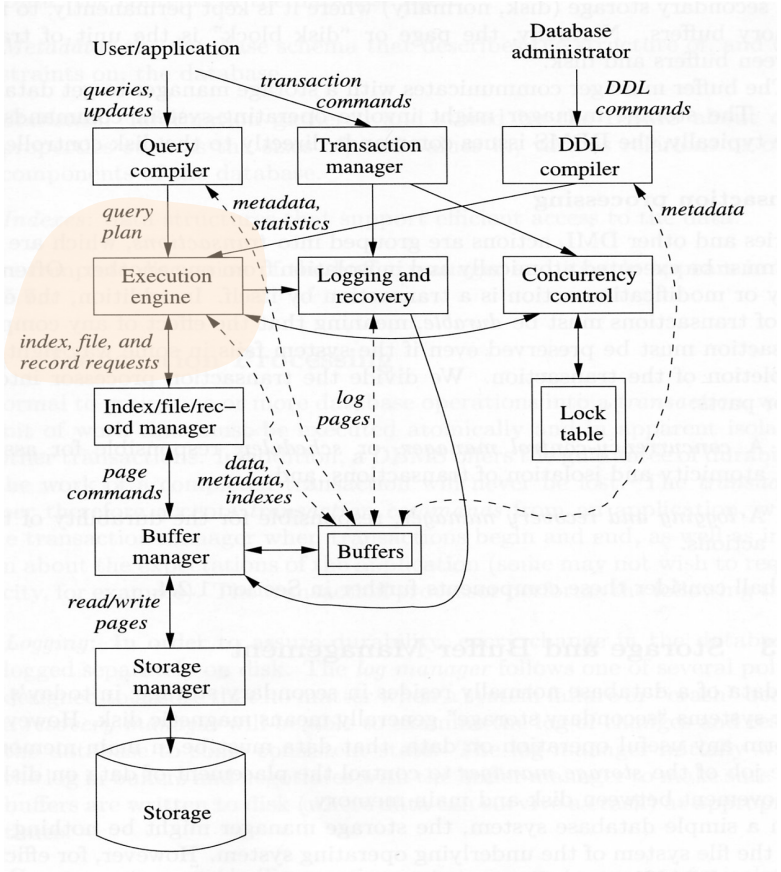
# Query Processing

- Query Compiler
  - *Parser* builds a tree structure from the textual form of the query
  - *Preprocessor* performs semantic checks on the query and translates the parse tree into algebraic operators representing the initial *query plan* (implementation of a relational algebra)
  - *Optimizer* transforms the initial query plan into the best available sequence of operations on actual data



# Query Processing

- Query Compiler
  - *Parser* builds a tree structure from the textual form of the query
  - *Preprocessor* performs semantic checks on the query and translates the parse tree into algebraic operators representing the initial *query plan* (implementation of a relational algebra)
  - *Optimizer* transforms the initial query plan into the best available sequence of operations on actual data
- Execution Engine
  - Executes the steps of the chosen query plan
  - Needs to interact with most of the other components of the DBMS



# Transaction Processing

- Group queries and/or DML actions into a transaction which must be executed atomically and in isolation from other transactions
- A DBMS guarantees *durability*: the work of a completed transaction will never be lost
- The *Transaction Manager* performs 3 tasks:
  - Logging
    - Every change in the database is logged on disk (through BM) to enable recovery in case of a crash
  - Concurrency Control assures
    - Atomicity (a transaction is performed either completely or not at all)
    - Isolation (transactions are executed as if there were no other concurrently executing transactions (uses locks))
  - Deadlock Resolution
    - "roll back" or "abort" some transaction