CSEN501 Database Systems
Lecture 1 – Introduction & Basic Concepts

Slides are based on the reference books:

Database System Concepts, 7th Ed., ©Silberschatz and Sudarshan

Fundamentals of Database Systems, 7th Ed., ©Elmasri and Navathe

Lecture Outline

- Course Logistics
- What is a Database? What is a Database Management System?
- Database-System Applications and Uses
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

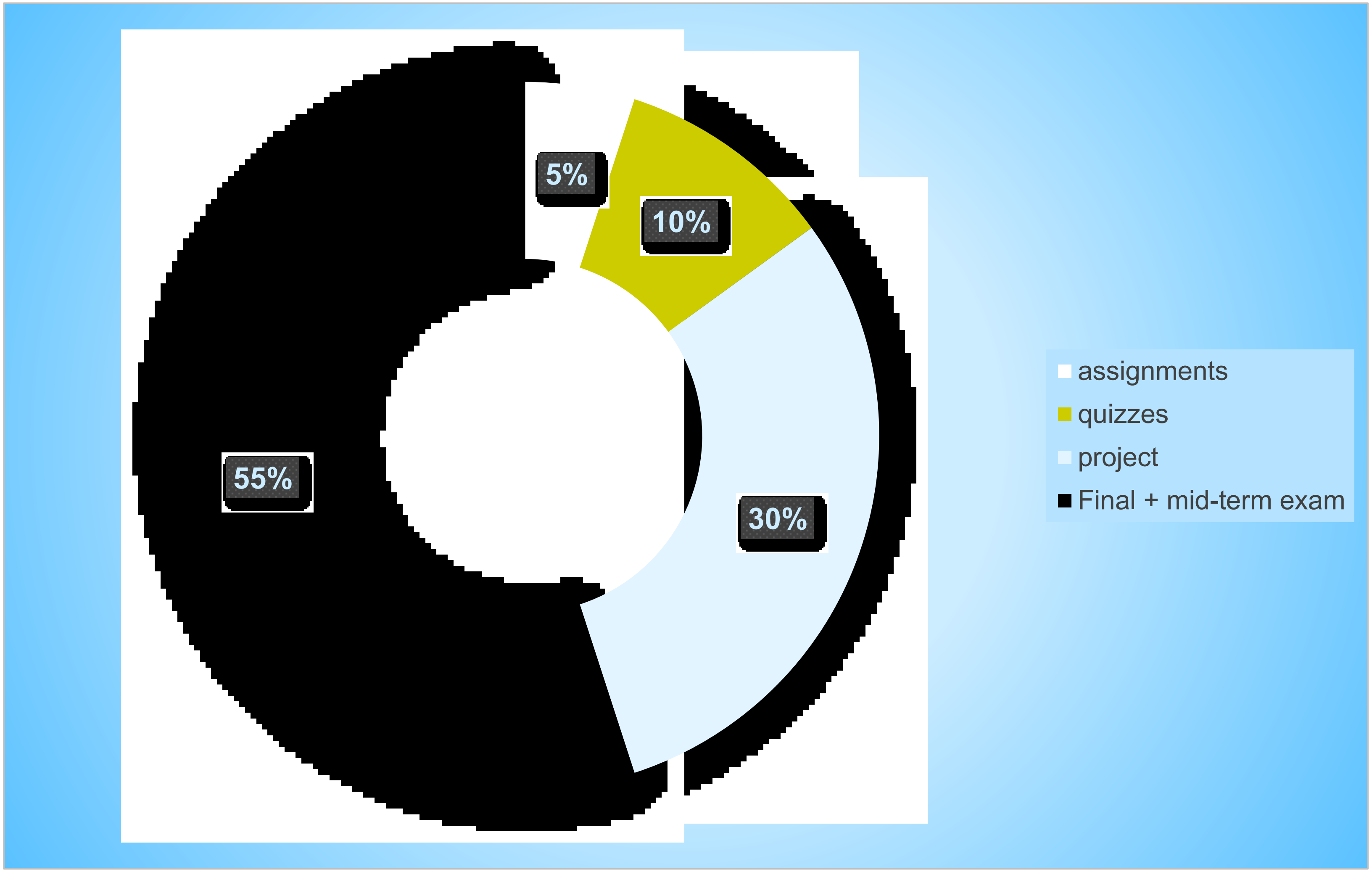
Course Structure

1. Lectures
2. Exercises and Homework
   - Practical Assignments
   - Work in teams, use feedback from tutors
3. Labs
   - Supervised lab Assignments
   - Work in teams
   - Piazza will be set up for questions and inquiries

Course Outline

1. Introduction and Basic Concepts
2. The Relational Model and The Entity Relationship Diagram
3. Relational Algebra
4. Relational Calculus
5. Database Design and Normalization
6. Introduction to SQL: From Beginner to Advanced
7. Query Processing and Optimization
8. Transaction Management
9. Storage Structures and Indexing (if time allows)

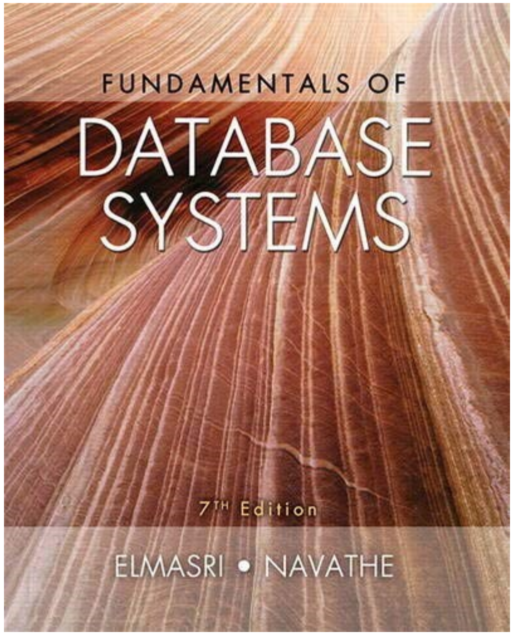
Grade Distribution



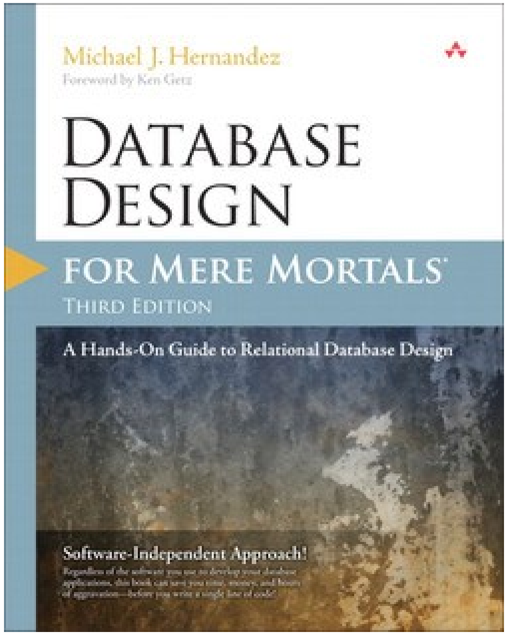
Textbooks (and Supplementary Resources)

Database System Concepts, 7th Ed.

Fundamentals of Database Systems



Database Design for Mere Mortals



and so many online SQL sources: SQL Cookbook, SQL Antipatterns, …

What is Data? Database? DBMS?

- **Data**: Known facts that can be recorded and have an implicit meaning; raw
- **Database**: a highly **organized**, **interrelated**, and **structured** set of data that models a **real world enterprise**
  - Controlled by a database management system (**DBMS**)
- **DBMS**
  - Set of programs to store, access, and manage the data
  - An environment that is both *convenient* and *efficient* to use
- **Database systems** are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time
- A modern database system is a complex software system whose task is to manage a large, complex collection of data (DBS = DBMS + Data)
- Databases touch all aspects of our lives

Database Applications Examples

- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
  - customer information, accounts, loans, and banking transactions
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: registration, grades

Database Applications Examples (Cont.)

- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Recent Developments

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
  - Facebook
  - Twitter
  - Linked-In
- All of the above constitutes data
- Search Engines, Google, Bing, Yahoo: collect their own repository of web pages for searching purposes
- New technologies are emerging to manage vast amounts of data generated on the web:
  - **Big data** storage systems involving large clusters of distributed computers
  - **NOSQL** (Not Only SQL) systems


Products DB
Accounts DB
amazon.com
Hello. Sign in to get personalized recommendations. New customer? Start here. Sell on Amazon, 30 days FREE*
Your Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards
Your Account
Help
Shop All Departments
Search
All Departments
GO
Cart
Wish List
Books
Movies, Music & Games
Digital Downloads
Kindle
Computers & Office
Electronics
Home, Garden & Pets
Grocery, Health & Beauty
Toys, Kids & Baby
Clothing, Shoes & Jewelry
Sports & Outdoors
Tools & Home Improvement
Automotive & Industrial
Shopping from Denmark? Visit amazon.co.uk
Shop now
The All-New Kindle
Smaller, Lighter, Faster with 50% Better Contrast
Order now: $139 Wi-Fi | $189 Free 3G+Wi-Fi

Why Do We Need Database Systems?

In the early days, data applications were built directly on top of file systems, which leads to:

- **Data redundancy and inconsistency**: data is stored in multiple file formats resulting in duplication of information in different files
- **Difficulty in accessing data**
  - Need to write a new program to carry out each new task
- **Data isolation**
  - Multiple files and formats
- **Integrity problems**
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- **Security problems**
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- **Relational model**
- **Entity-Relationship data model** (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- **Semi-structured data model (XML)**
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

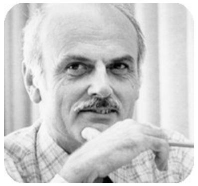
- All the data is stored in various tables
- Example of tabular data in the relational model

Columns

Rows

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table



Ted Codd
Turing Award 1981

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Levels of Abstraction

- **Physical level**: describes how a record (e.g., instructor) is stored/indexed
- **Logical level**: describes data stored in database, and the relationships among the data

  type *instructor* = **record**

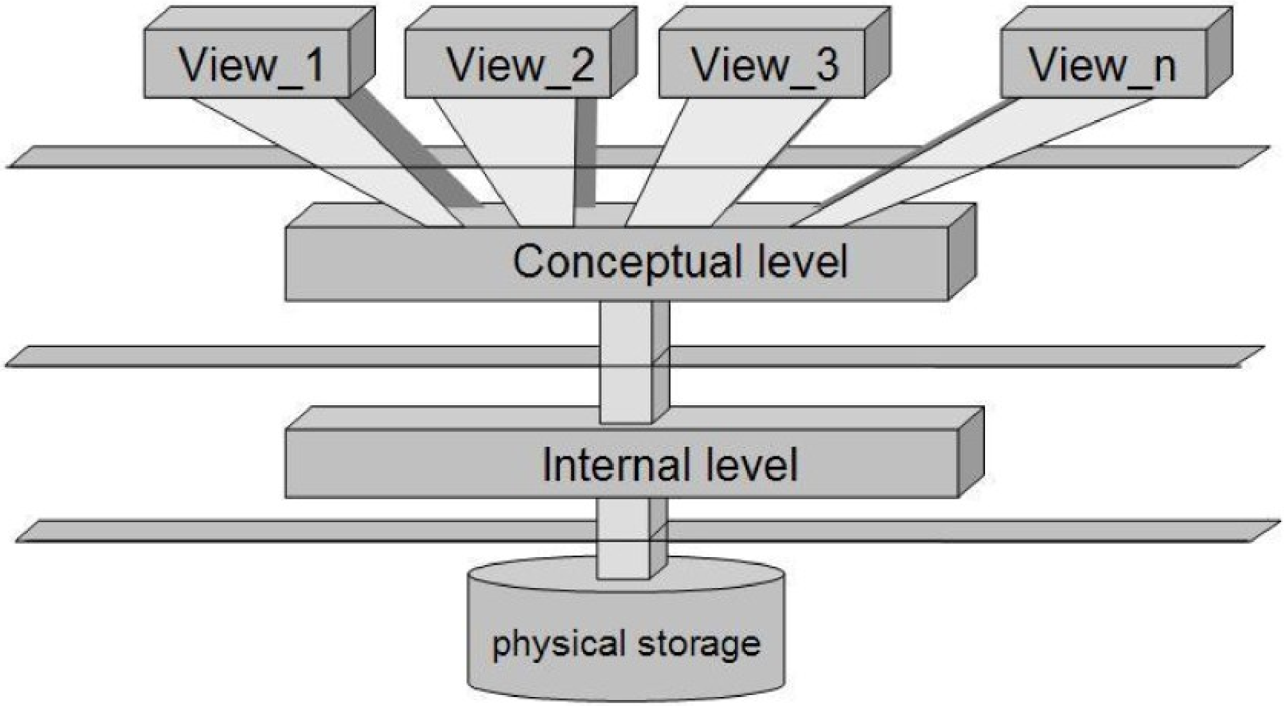
  ID : string;
  name : string;
  dept_name : string;
  salary : integer;

  end;

  - **Conceptual Level:** overall database design
  - **View level**: application programs that users use – views hide details of data types, and can also hide information (such as an employee's salary) for security purposes

View of Data

An architecture for a database system



Schemas and Instances

Similar to **types** and **variables** in programming languages

- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema** – the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Data Independence

- One of the most important benefits of DBMS
- A user of a relational database system should be able to use SQL to query the database without knowing about how precisely data is stored, e.g.

```
SELECT When; Where FROM Calendar WHERE Who = "Bill"
```

- There are two kinds of data independence:
  - Logical data independence protects the user from changes in the logical structure of the data – could completely reorganize the “schema” without changing how user would query it
  - Physical data independence protects the user from changes in the physical structure of data – could add an ***index*** on a column without changing how the user would write the query, but the query would execute faster (query optimization)

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example:
```
create table instructor (
    ID          char(5),
    name        varchar(20),
    dept_name   varchar(20),
    salary      numeric(8,2))
```

- DDL compiler generates a set of table templates stored in a **data dictionary**
- Data dictionary contains **metadata** (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as **query language**
- There are two types of data-manipulation language:
  - **Procedural DML** – require a user to specify what data are needed and how to get those data
  - **Declarative DML** – require a user to specify what data are needed without specifying how to get those data
- **Declarative DMLs** are usually easier to learn and use than **procedural DMLs**

SQL Query Language

- SQL query language is **nonprocedural**
  - A query takes as **input** several tables (possibly only one) and always **returns** a single table
- Example to find all instructors in Comp. Sci. dept

```
select name
from instructor
where dept_name = 'Comp. Sci.'
```

- SQL is **NOT** a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system
- The functional components of a database system can be divided into
  - The storage manager
  - The query processor component
  - The transaction management component

Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager
- The storage manager implements several data structures:
  - Data files – store the database itself
  - Data dictionary – stores metadata about the database schema
  - Indices – can provide fast access to data items

Query Processor

- The query processor components include:
  - DDL interpreter – interprets DDL statements and records the definitions in the data dictionary
  - DML compiler – translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands
    - The DML compiler performs **query optimization**; it picks the lowest cost evaluation plan from among the various alternatives
  - Query evaluation engine – executes low-level instructions generated by the DML compiler

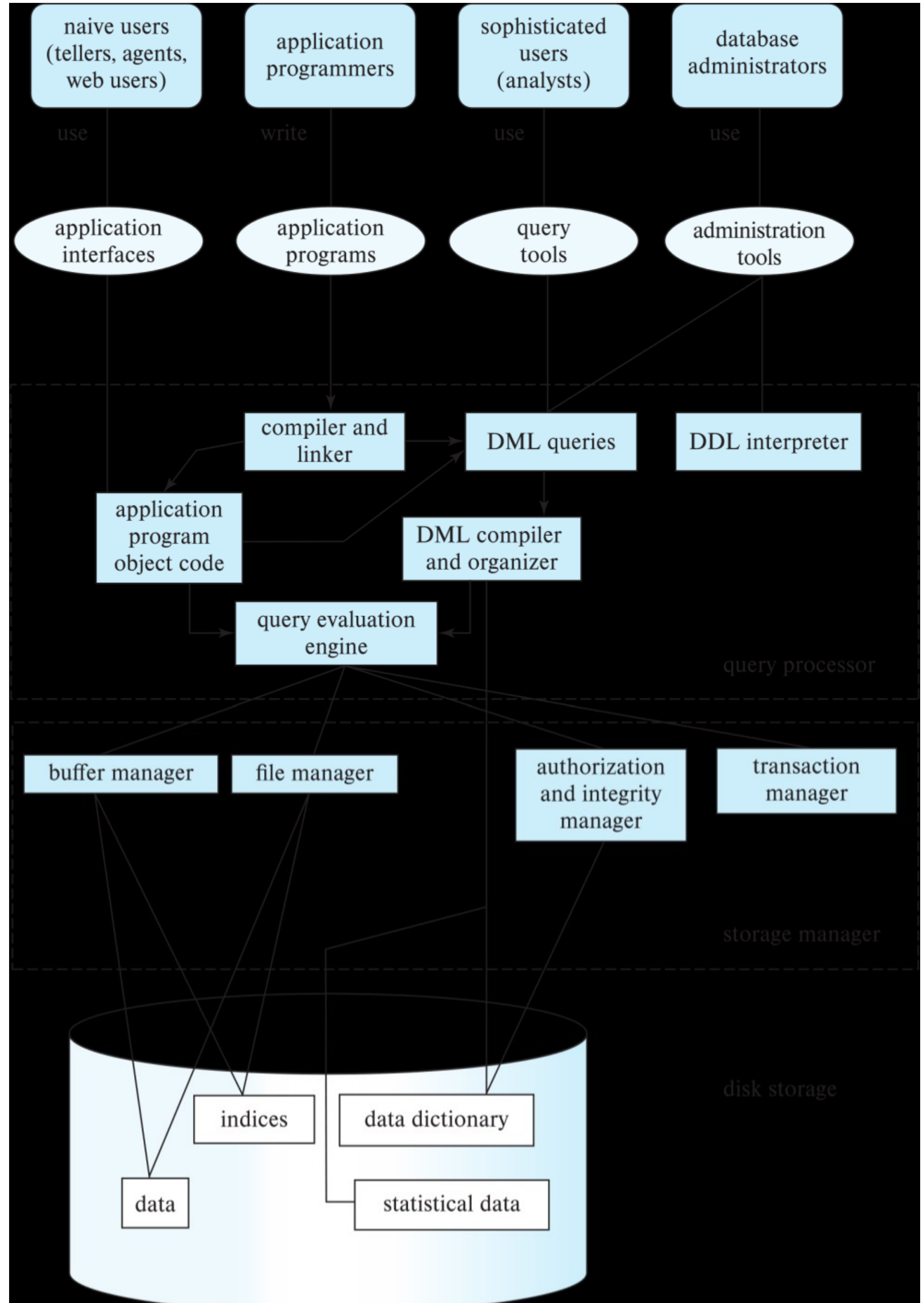
Transaction Manager

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-manager** ensures that the database remains in a **consistent** (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database

Database Architecture

- Centralized databases
  - One to a few cores, shared memory
- Client-server
  - One server machine executes work on behalf of multiple client machines
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity

Database Architecture
(A Deeper Look)



Database Administrator

A person who has central control over the system is called a **database administrator (DBA)**. Functions of a DBA include:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database

There is also the application programmers and the end users

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems (Cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Amazon
    - “NoSQL” systems
  - Big data analysis: beyond SQL
    - MapReduce and friends
- 2010s
  - SQL reloaded
    - SQL front-end to MapReduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Thank you