BLG 317E
DATABASE SYSTEMS

FALL 2023

ITU

Topics

- Data models
- Use of database systems
- Application development
- Database design

Weekly Schedule

Week	Lecture
1	Overview of Database Systems
2	Relational Model, Relational Algebra
3	Integrity Constraints, DML Queries
4	Joins, Nested Subqueries, and Set Operations
5	Views, Built-in Functions, Aggregations
6	Stored Procedures, Cursors, Triggers, App Development
7	MIDTERM EXAM

Week	Lecture
8	Database Design 1 – ER Diagrams
9	Database Design 2 - Normalization
10	Transactions and Concurrency
11	Non-Relational/NoSQL Databases
12	XML Databases, Graph Databases
13	Project Presentations and Demos
14	Project Presentations and Demos

Grading

- Midterm exam: 30%
- Project: 30%
- Final exam: 40%

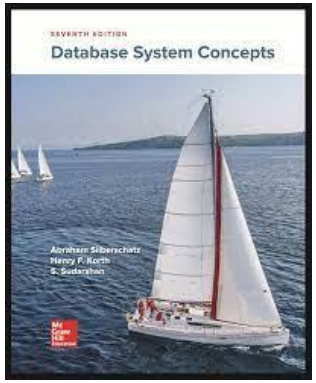
- VF:
  - AVG(midterm exam and project) < 30/100 → VF
- Minimum Passing Grade:
  - end-of-term total < 40/100 → FF

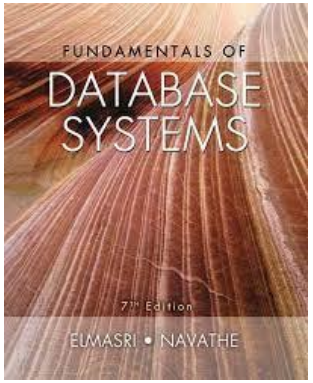
Projects

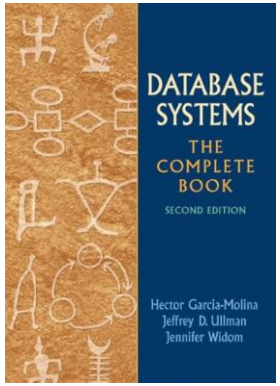
- As a team of 3-5 students (group size → min: 3, max: 5)
- Theme: Database-enabled web project
- Choose a public data set
- Tech Stack: Python, Flask, Relational Database
- Grading:
  - Implementation and Demo: 85%
  - Report: 15% (Cannot be 10% larger than your demo grade)
- Presentation: Bonus 10%
- GitHub repo
  - Weekly meaningful commits by each team member (starting week 5)
  - 10% deduction if no regular commits
- Progress review meeting with your TAs (week 9)
  - 10% bonus for significant progress
- All grades are individual-based (no group grade)

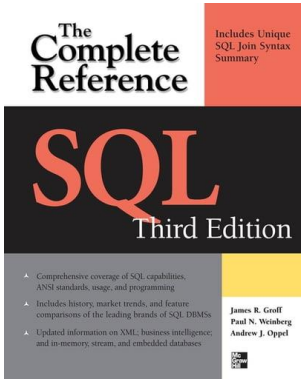
Recommended Reference Books

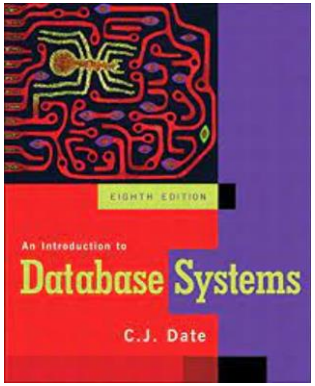
- Database System Concepts – Seventh Edition, McGraw-Hill, by Abraham Silberschatz, Henry F. Korth, S. Sudarshan, ISBN: 9780078022159, 2019.
- Fundamentals of Database Systems, R.Elmasri, S.B.Navathe, 7th ed., Pearson Publishing, 2016.
- Database Systems The Complete Book, H.Garcia-Molina, J.D.Ullman, J.Widom, 2nd ed, Pearson Prentice Hall Publishing, 2009.
- SQL The Complete Reference, J.R.Groff, P.N.Weinberg, A.J.Oppel, 3rd ed., McGraw-Hill Publishing, 2009.
- An Introduction to Database Systems, Chris J. Date, Addison-Wesley, ISBN 0-321-19784-4, 2004.











Course Logistics

- Ninova
  - Syllabus
  - Slides
  - Resources
  - Announcements
- Kepler
  - Grades
- ITU Email
  - Check your emails regularly
- Practice Session: Last hour of each week

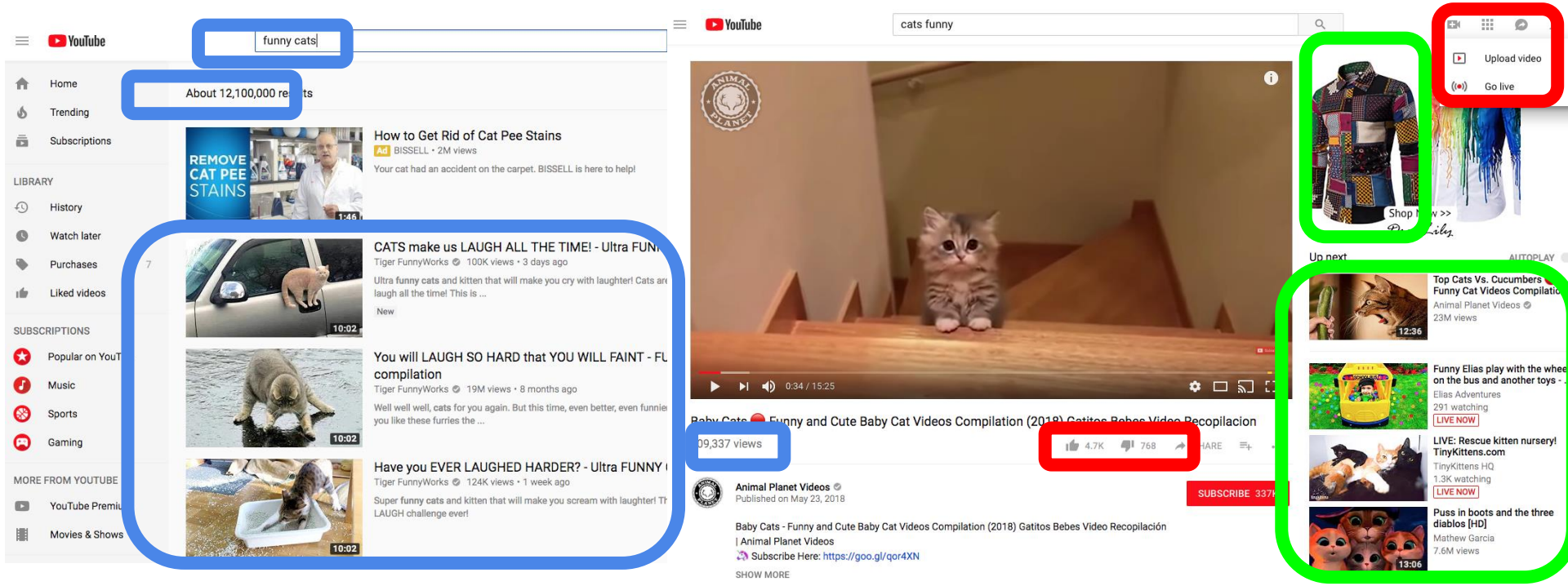
BLG 317E
DATABASE SYSTEMS

WEEK 1

Slides credit:
S. Shivakumar, CS 145, Stanford University
Database System Concepts, Silberschatz, Korth, Sudarshan

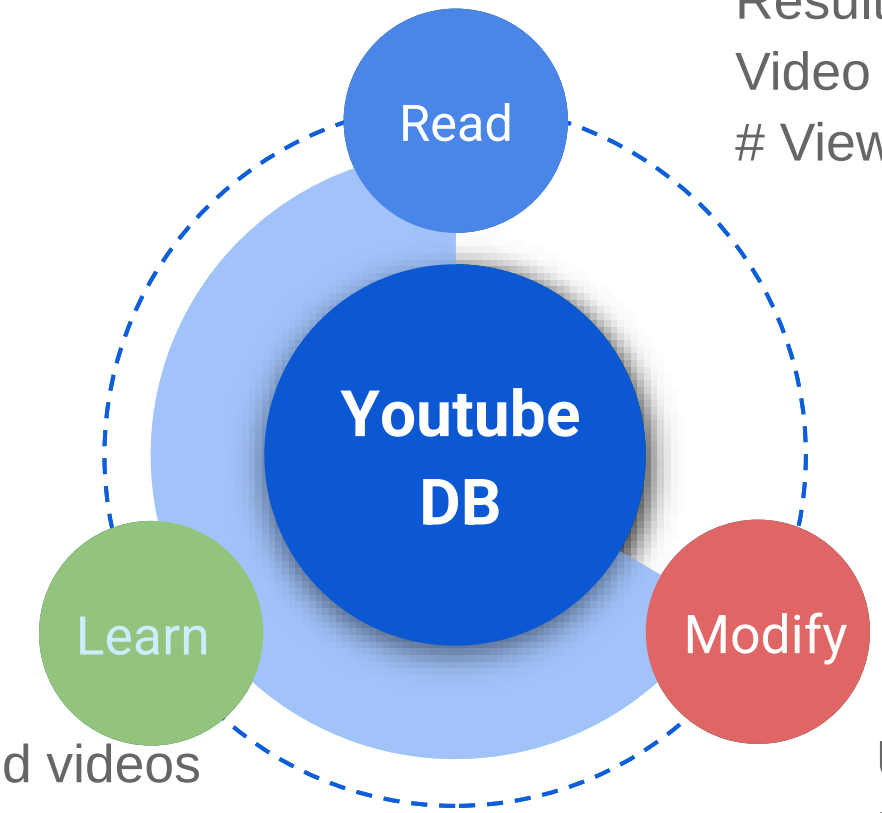
ITU

Example: Youtube DB



Example

Unpack Youtube DB

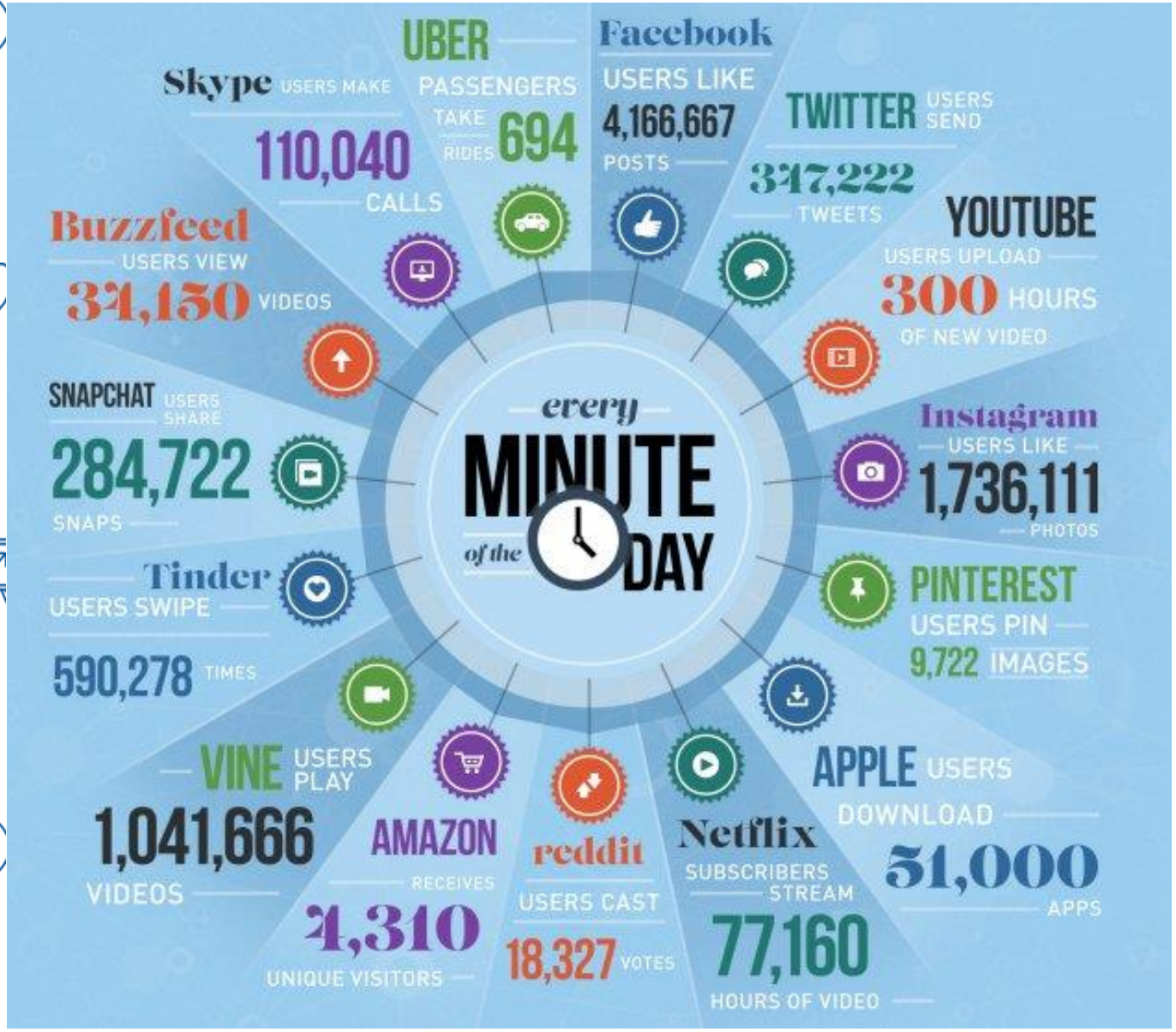


Result list
Video & description,
Views, Likes

Related videos
Relevant ads

Upload
Like, Review

Every minute on the Internet



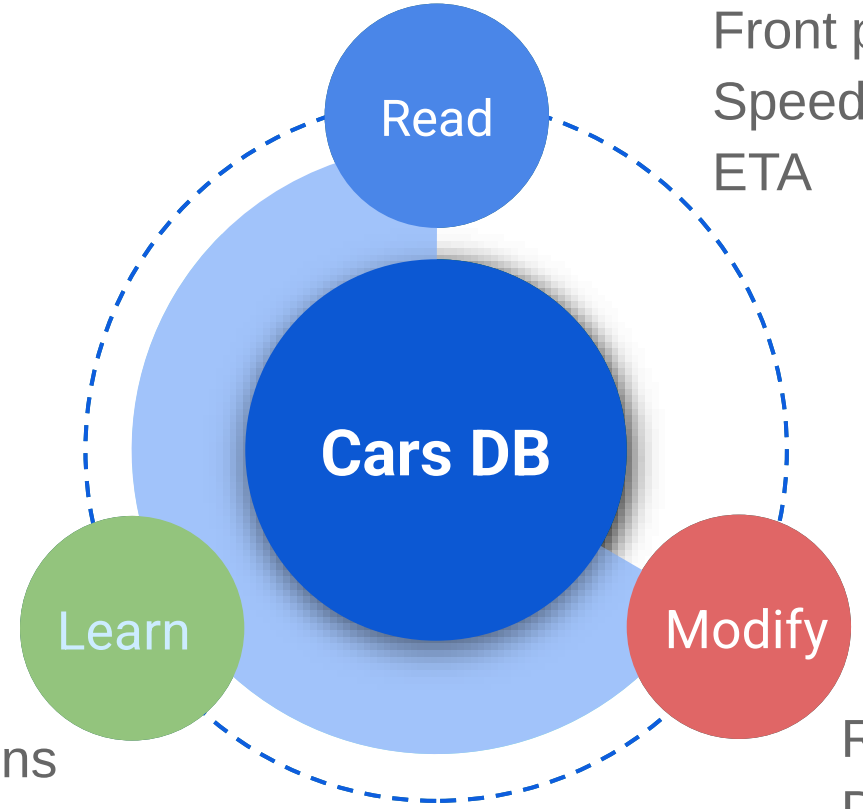
Example

Self Driving Cars


THE COMING FLOOD OF DATA IN AUTONOMOUS VEHICLES
RADAR
~10-100 KB
PER SECOND
SONAR
~10-100 KB
PER SECOND
GPS
~50KB
PER SECOND
CAMERAS
~20-40 MB
PER SECOND
AUTONOMOUS VEHICLES
4,000 GB
PER DAY... EACH DAY
LIDAR
~10-70 MB
PER SECOND
intel

Example

Unpack Cars DB



Front panel metrics
Speed, distance
ETA

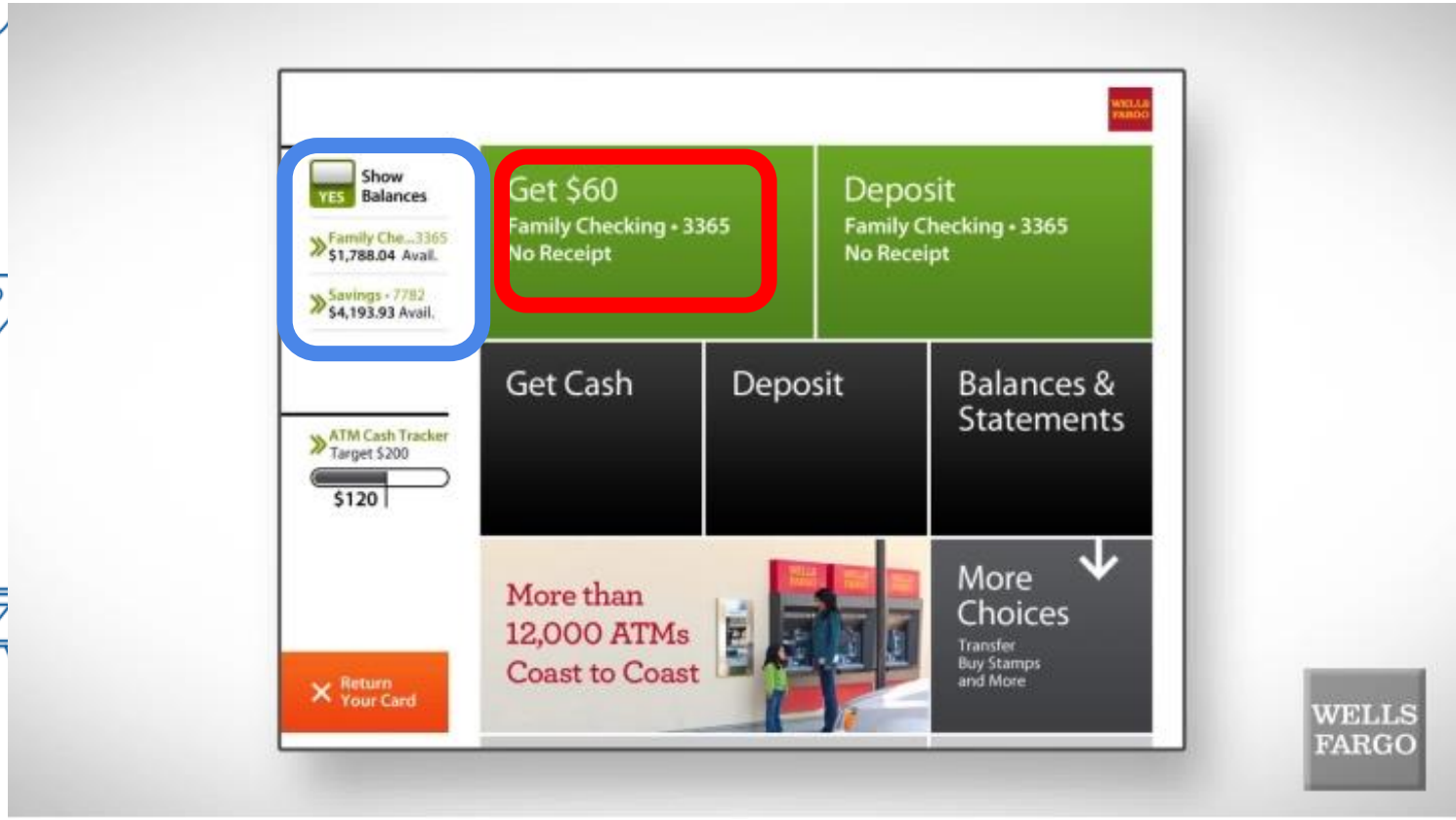
Collisions
Traffic signals

Road models
Drive models

Example

Unpack ATM DB:

Transaction



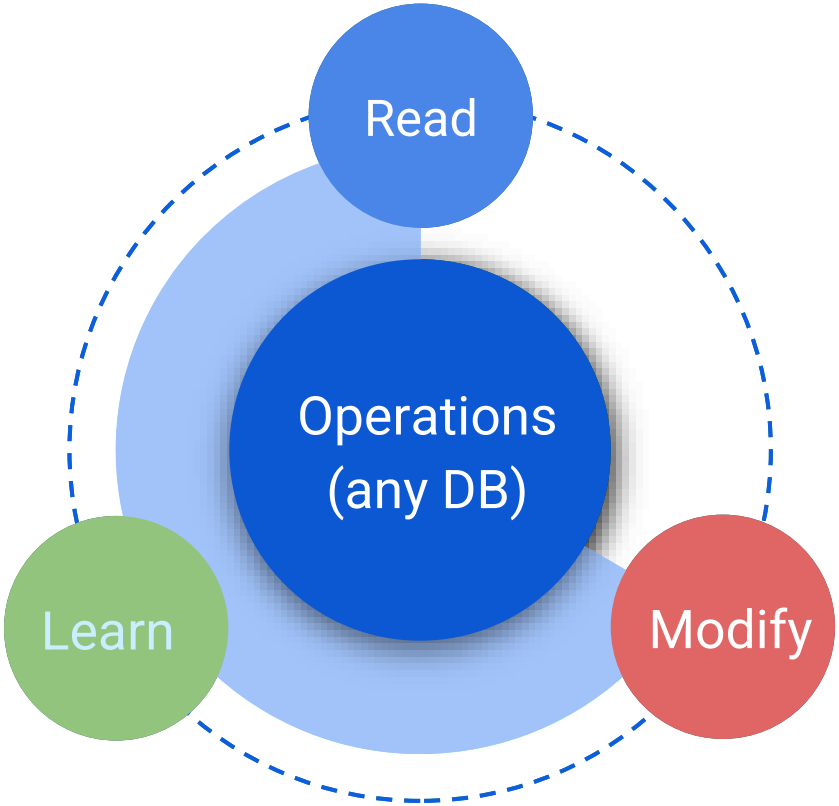
Read Balance
Give money
Update Balance

vs

Read Balance
Update Balance
Give money

Goals of Databases

Platform to store, manage data



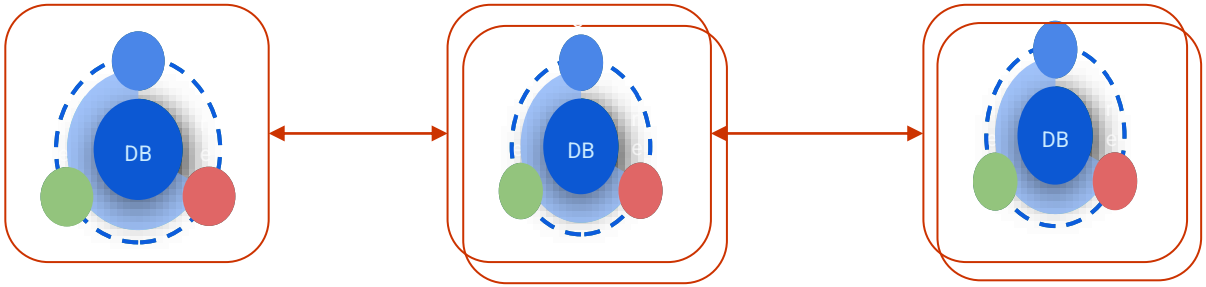
Supporting

- Scale
- Speed
- Stability
- Evolution
- Reliability
- Cost efficiency

Goals of Data systems

Tune for custom system (a la Lego blocks)

Connect one/many DBs for custom system



Store current data (e.g., lot of reads)

Optimize historical data (e.g., logs)

Run batch Workloads (e.g. training)

When to build a custom data system?

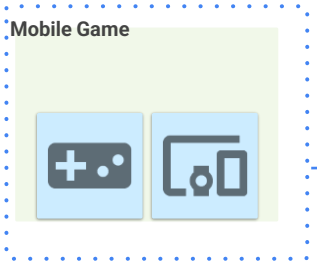
HOW?

FOR WHOM?

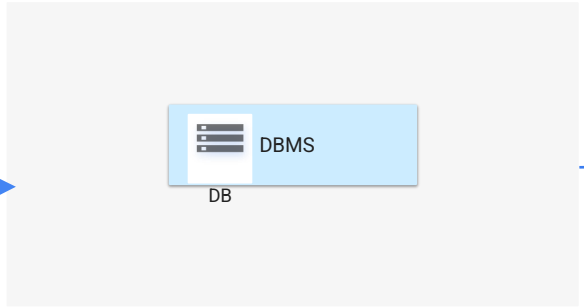
How?

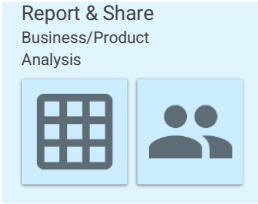
Example Game App

DB v0



Real-Time
User Events





Q1: 1000 users/sec?
Q2: Offline?
Q3: Support v1, v1' versions?

App designer

Q7: How to model/evolve game data?
Q8: How to scale to millions of users?
Q9: When machines die, restore game state gracefully?

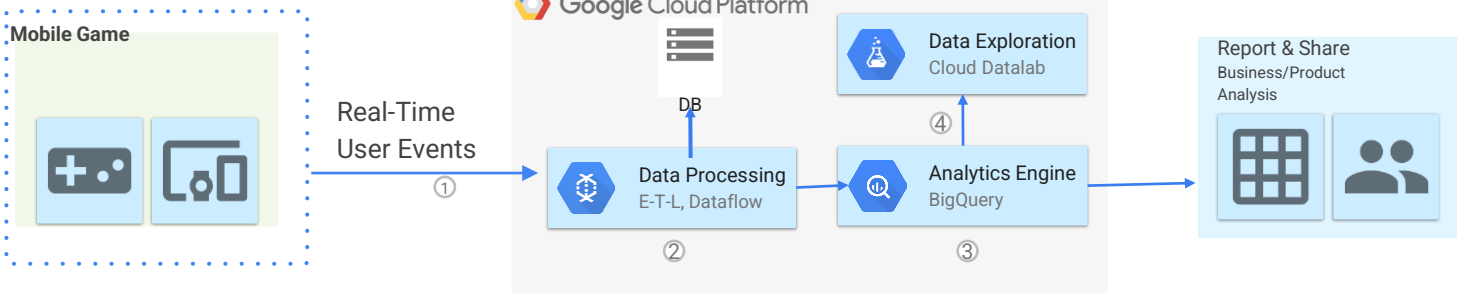
Systems designer

Q4: Which user cohorts?
Q5: Next features to build? Experiments to run?
Q6: Predict ads demand?

Product/Biz designer

How?

Example Game App



Data system "v1" on Cloud

1 Log user actions

2 Store in DB, after Extract-Transform-Load

3 Run queries in a peta scale analytics system

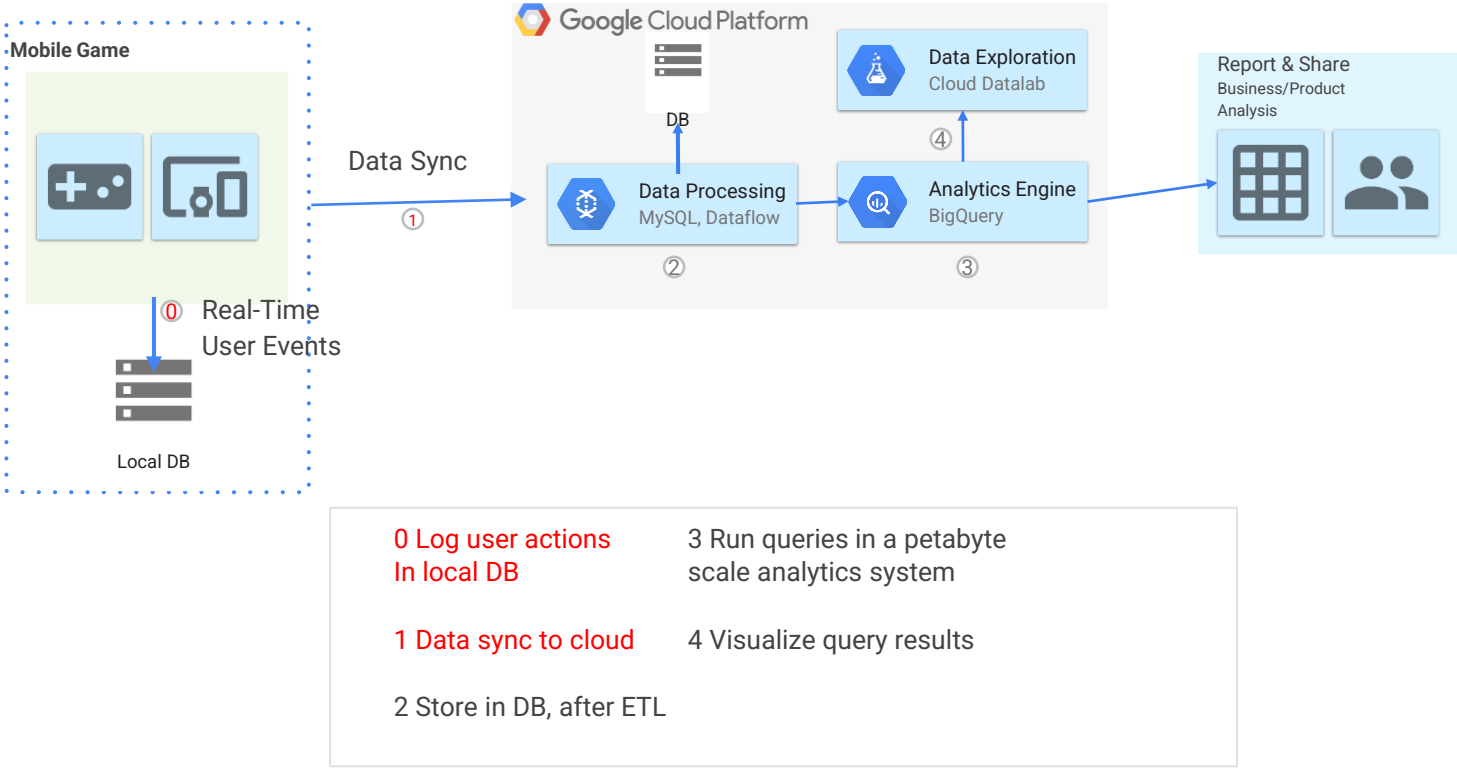
4 Visualize query results

How?

Example Game App

Data system

“v2” Cloud + Local



0 Log user actions In local DB

1 Data sync to cloud

2 Store in DB, after ETL

3 Run queries in a petabyte scale analytics system

4 Visualize query results

Definitions

- **Database**
  - A collection of related data on permanent storage.
  - Example: Students, Courses, and Enrollment information can be stored in a University database.
- **Database Management System (DBMS)**
  - A software system to facilitate the creation and maintenance of a database.
  - It is also called **Database Server**, or **Database Engine**.
- **Database System**
  - The DBMS software together with the data itself.
  - The application programs.

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

DBMS Category Usages

(Source: db-engines.com)

Database Model	Approximate Usage Percentage
Relational DBMS (SQL databases)	70%
Non-relational DBMS (NoSQL databases)	30%

Examples of Relational DBMSs

DBMS	Database Model	License Type
Oracle	Relational, Multi-model	Commercial
MySQL	Relational, Multi-model	Open Source
Microsoft SQL Server	Relational, Multi-model	Commercial
PostgreSQL	Relational, Multi-model	Open Source
IBM DB2	Relational, Multi-model	Commercial
SQLite	Relational	Open Source
Microsoft Access	Relational	Commercial

Examples of Relational DBMSs

- SQLite is used as an embedded database in many mobile applications (smartphone, tablet).
  - It is a serverless, single-user DBMS.
  - It can also be used as a personal DBMS on computers.
- Multi-model DBMS means it supports non-relational models also.
- PostgreSQL is an Object-Relational DBMS.
  - It supports object-oriented features such as table inheritance.
- Most DBMSs also have free editions.
  (Community Edition / Express Edition)

Examples of Non-relational DBMSs (NoSQL)

DBMS	Database Model
MongoDB	Document-store
Couchbase	Document-store
Redis	Key-value
Elasticsearch	Search engine
Amazon DynamoDB	Multi-model
Microsoft Azure Cosmos DB	Multi-model
Google Cloud Datastore	Document-store
Neo4j	Graph

Why data systems?

What about data structures from algorithms class?

Spreadsheets?

Text files?

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, 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

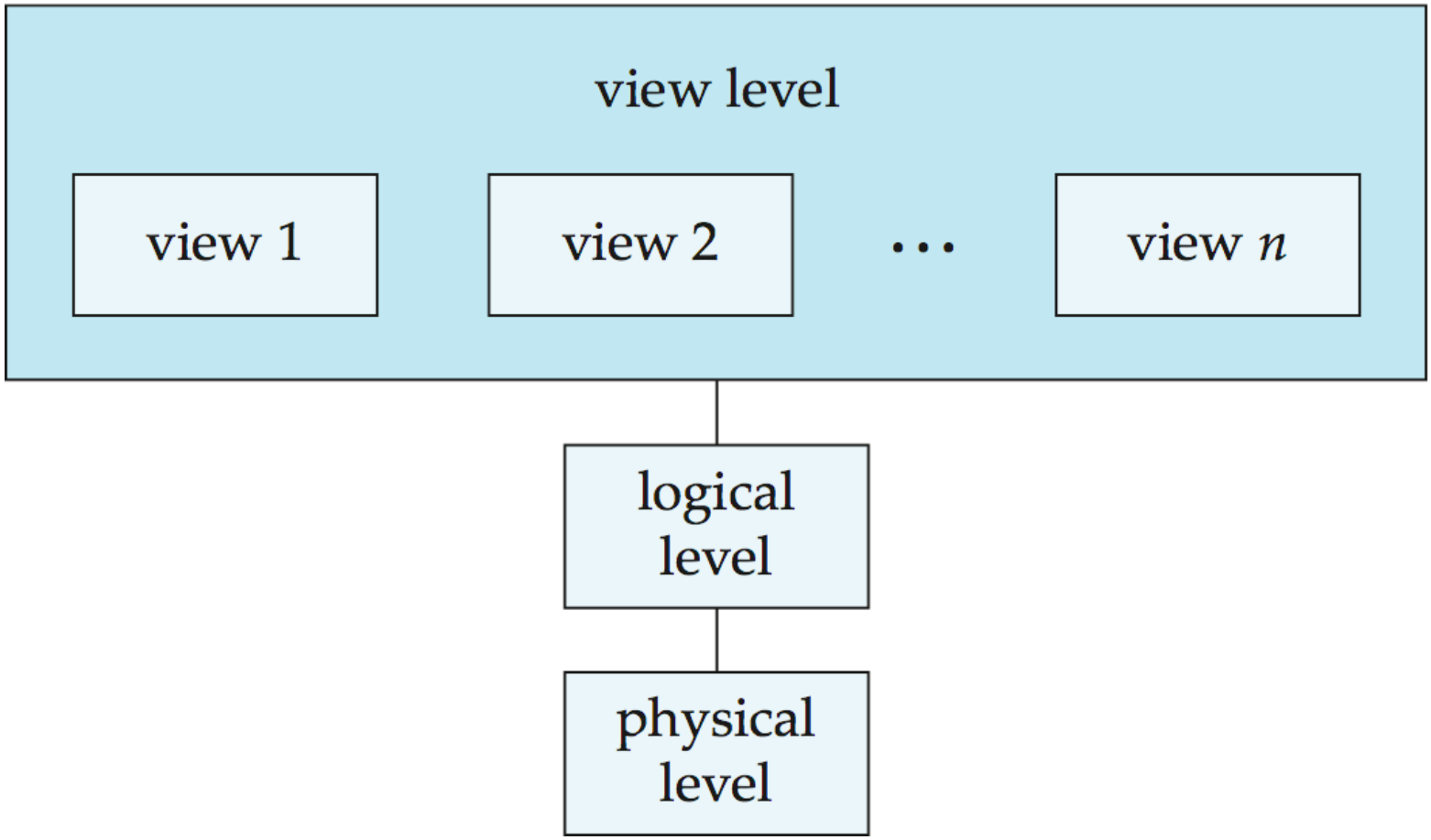
Drawbacks of using file systems to store data (Cont.)

- 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
    - Example: 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

Views of Data

An architecture for a database system



Levels of Abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored.
- **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;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

Instances and Schemas

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

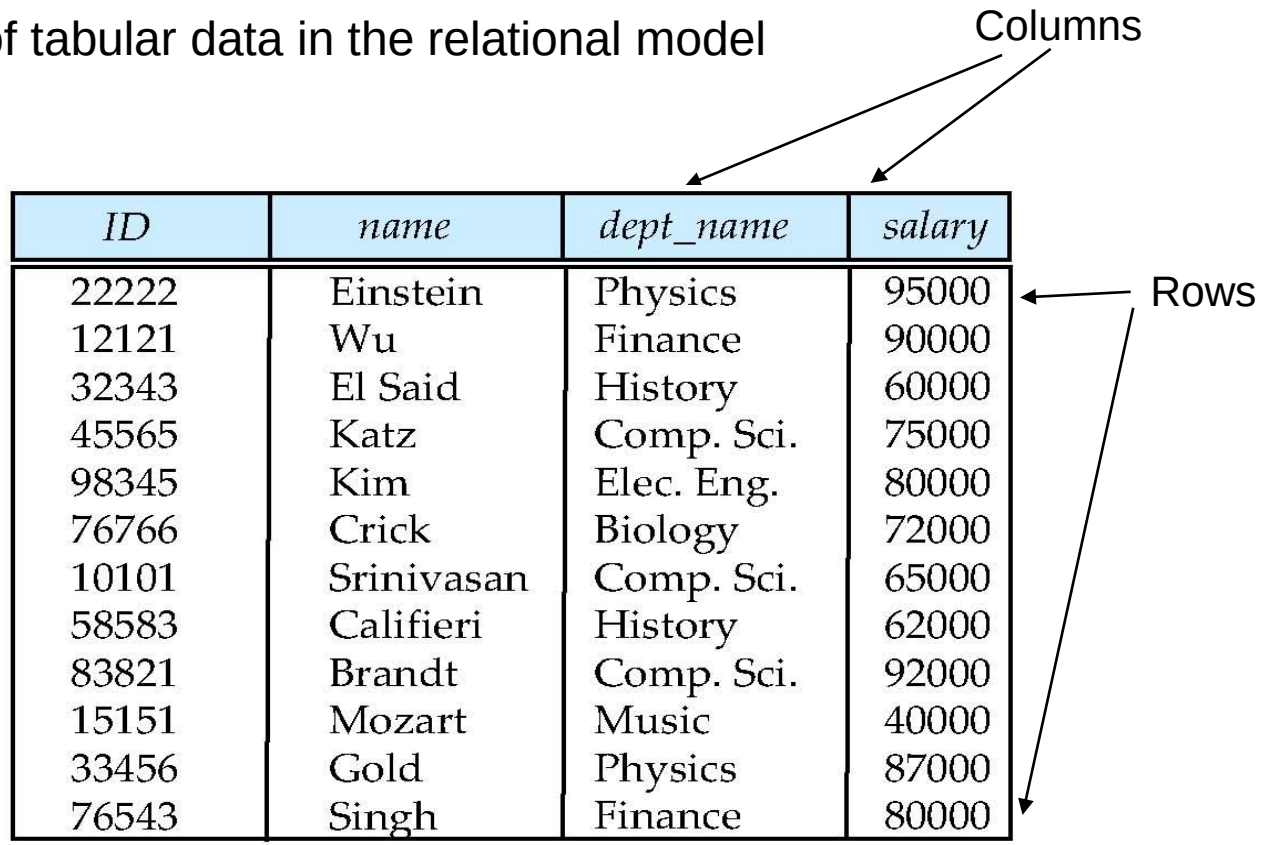
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

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)
- Semistructured 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



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

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

SQL

- The most widely used commercial language
- SQL is NOT as expressive as a programming 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 Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

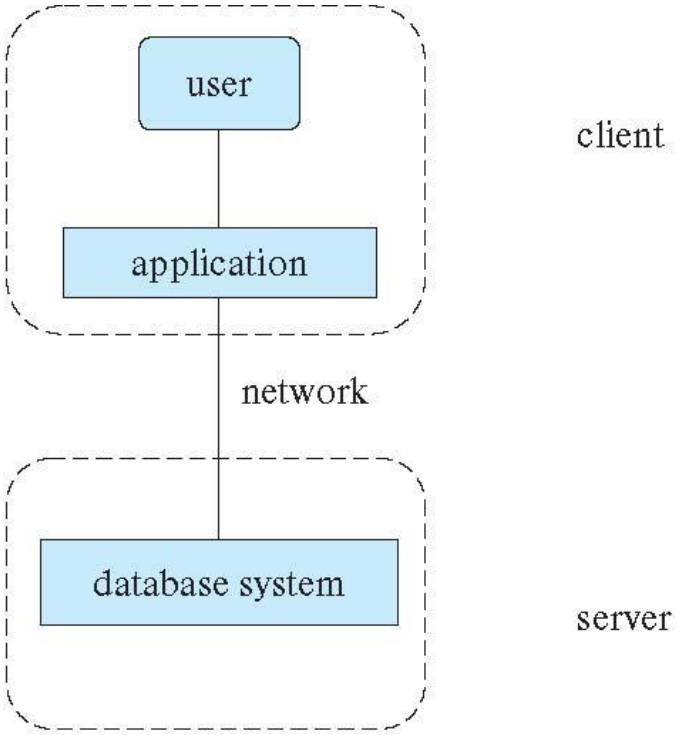
- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
  - Entity Relationship Model (Week 8)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Week 9)
    - Formalize what designs are bad, and test for them

Database Architectures

- Database systems structure
  - Centralized databases
    - One to a few cores, shared memory
  - Client-server,
    - One server machine executes work on behalf of multiple client machines.
    - Two tier, three tier, …, N-tier architectures

Two-tier Architecture

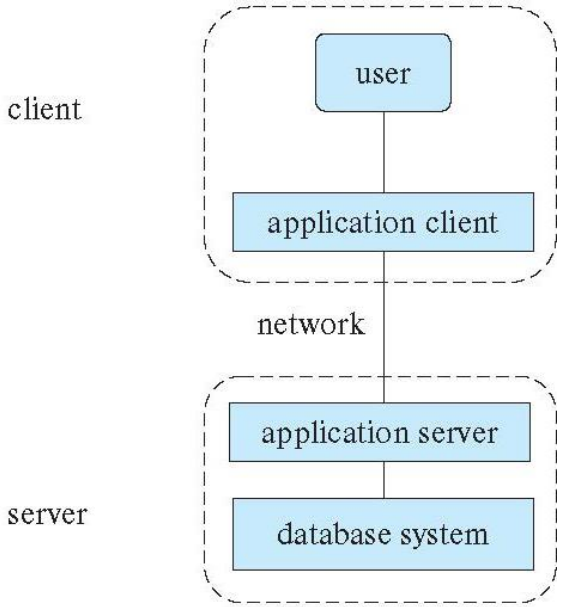
- Database applications are usually partitioned into two or three parts.
  - Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine



(a) Two-tier architecture

Three-tier Architecture

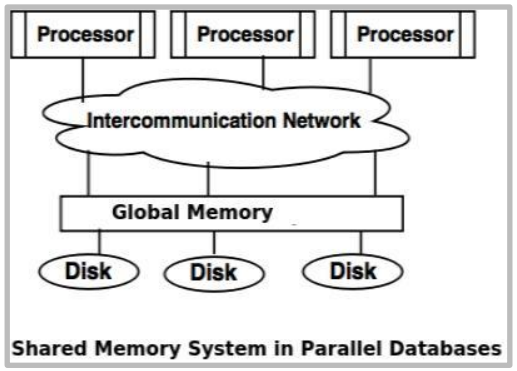
- Database applications are usually partitioned into two or three parts.
  - Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
    - The client end communicates with an application server, usually through a forms interface.
    - The application server in turn communicates with a database system to access data.

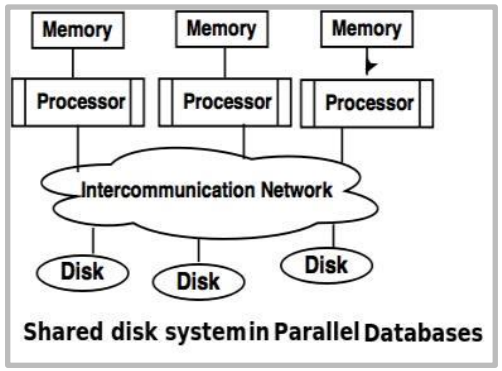


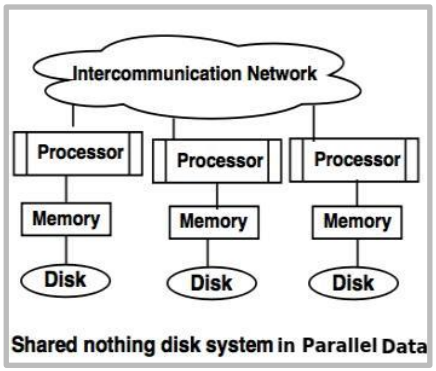
(b) Three-tier architecture

Database Architectures – Cont'd.

- Database systems structure
  - Parallel databases (multiple cores)
    - Shared memory
    - Shared disk
    - Shared nothing

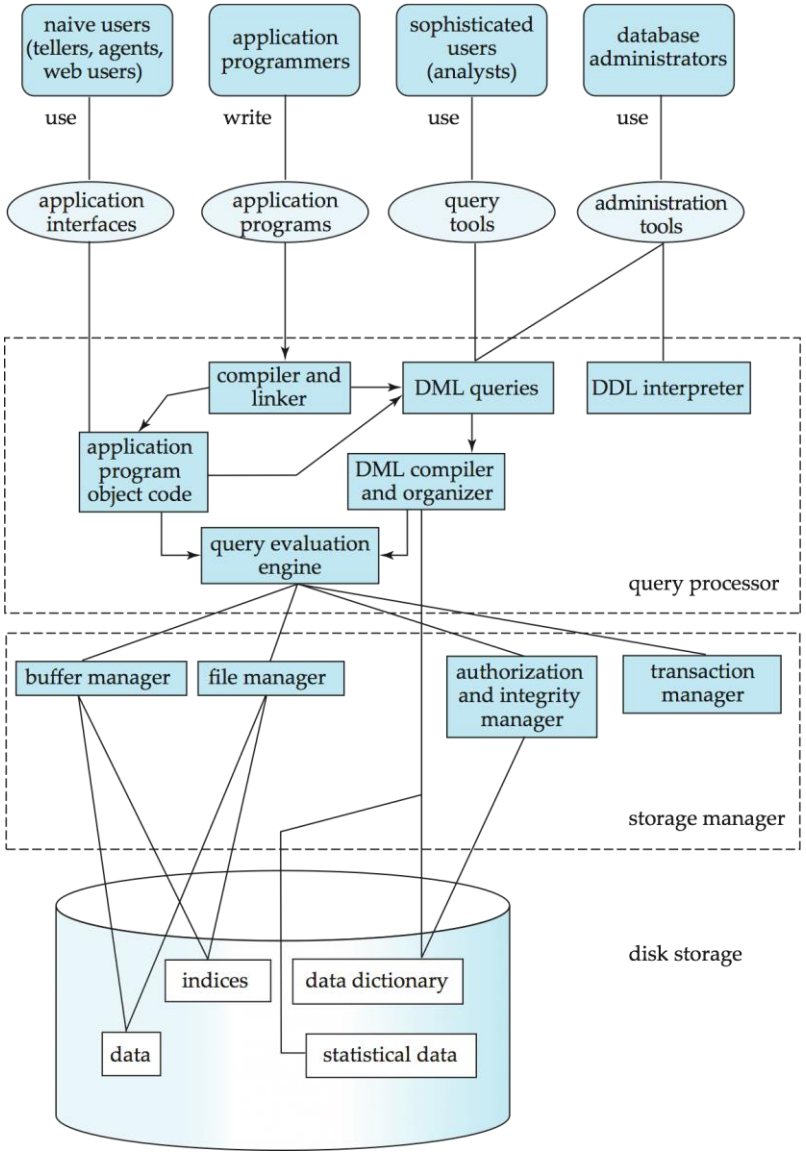






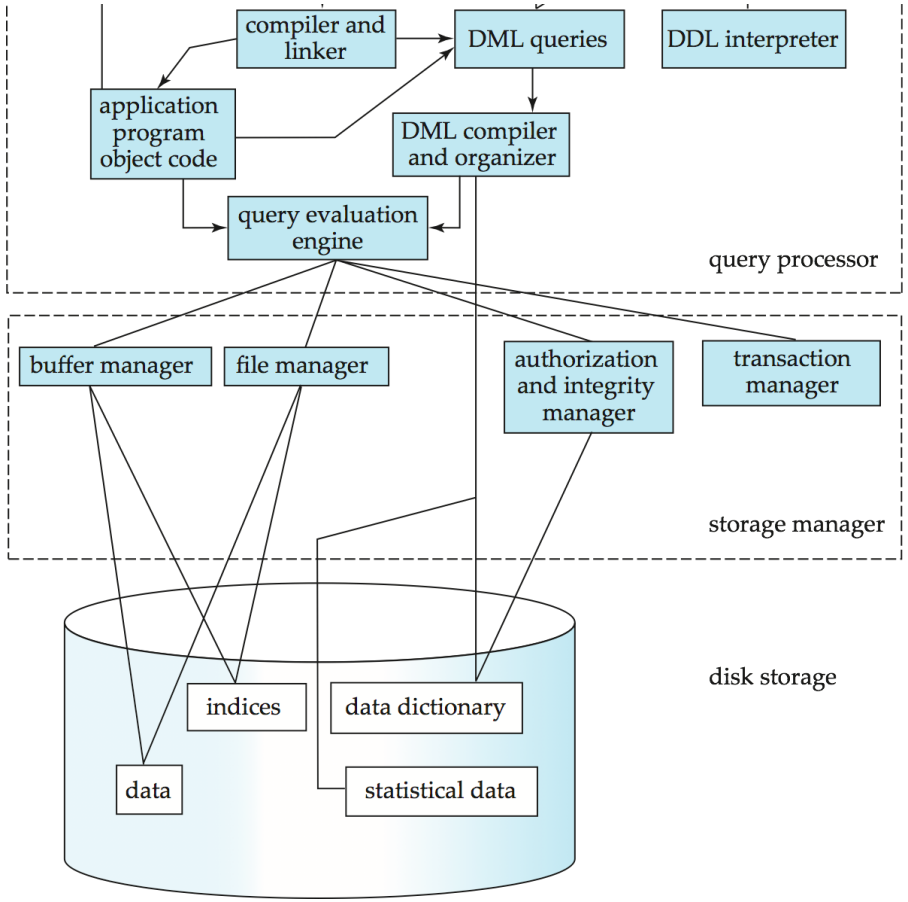
  - Distributed databases
    - Geographical distribution
    - Schema/data heterogeneity

Database System Internals – Full Picture



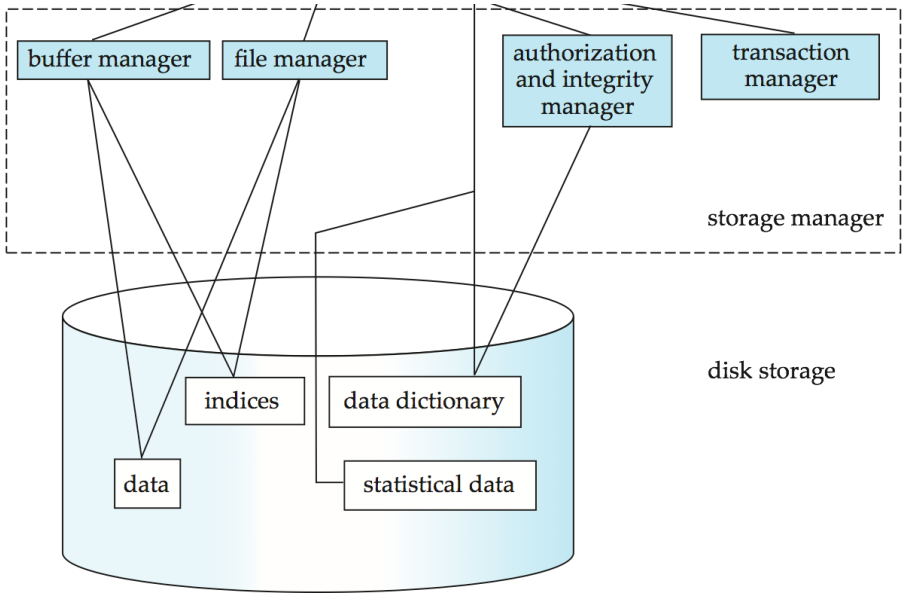
Database Engine

- 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.



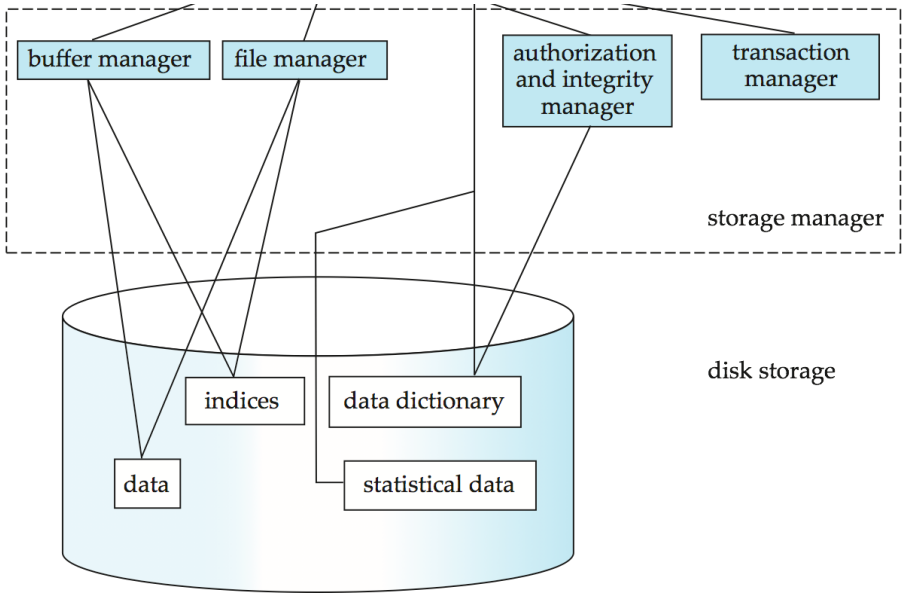
Storage Manager

- 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



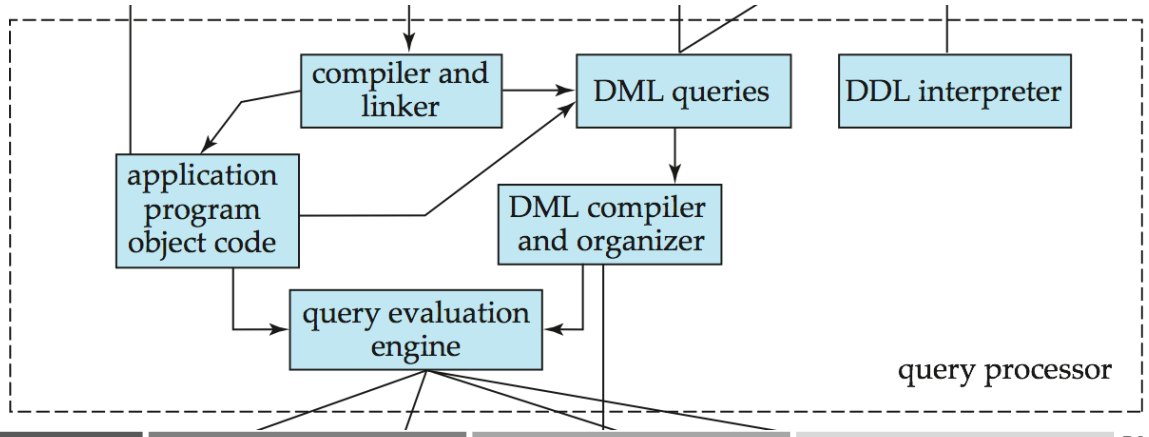
Storage Manager (Cont.)

- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.



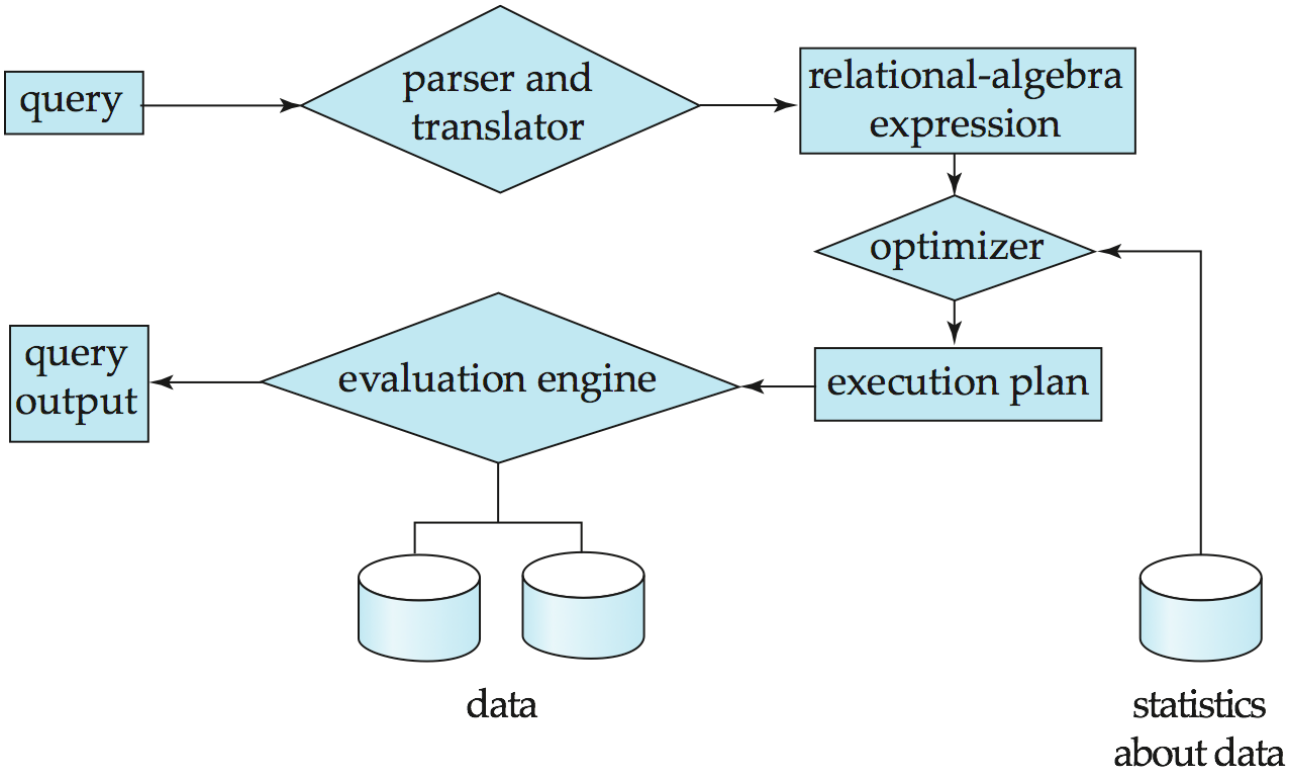
Query Processor

- 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; that is, it picks the lowest cost evaluation plan from among the various alternatives.
  - Query evaluation engine -- executes low-level instructions generated by the DML compiler.



Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



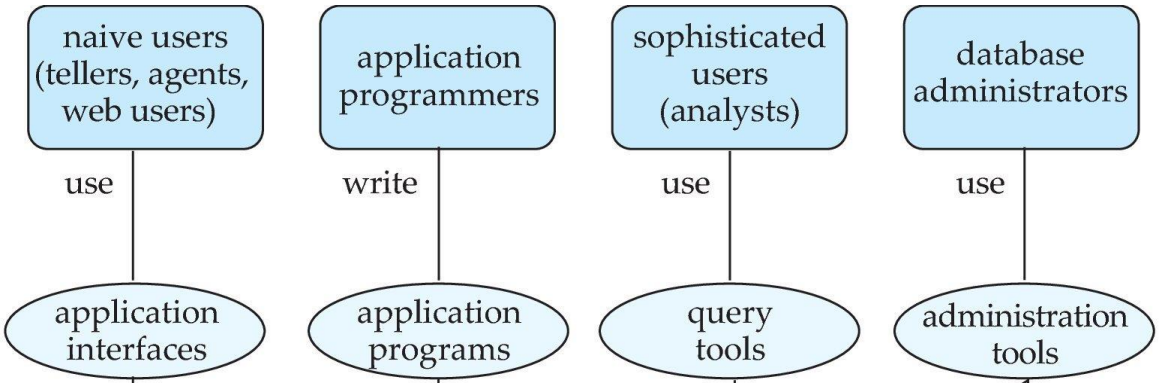
Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Database Users

There are four different types of database-system users

- Naive users -- unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- Application programmers -- are computer professionals who write application programs.
- Sophisticated users -- interact with the system without writing programs
  - using a database query language or by
  - using tools such as data analysis software.
- Database Administrators – see the next slide



Database Administrator

A person who has central control over the system is called a database administrator (DBA) whose functions are:

- 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 and ensuring that performance is not degraded by very expensive tasks submitted by some users

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** 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.

Transactions - Example

Transfer $3k from a10 to a20:

1 Debit $3k from a10
2 Credit $3k to a20

Crash before 1,
After 1 but before 2,
After 2.

Acct	Balance
a10	20,000
a20	15,000

Acct	Balance
a10	17,000
a20	18,000

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 begins Ingres prototype
  - High-performance (for the era) transaction processing

History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..