



Database Systems

Lecture01 – Introduction to DBMS



Beomseok Nam (남범석)
bnam@skku.edu



Class Organization

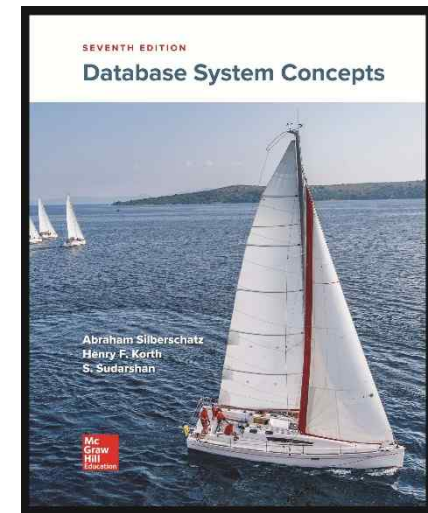
Instructor	Class code	Lecture hours	WebEx	Email
Nam, Beomseok	SWE3003	Tue 16:30 – 17:45 Thu 15:00 – 16:15	http://skku-ict.webex.com/meet/bnam	bnam@skku.edu

- Lecture Slides & QnA
 - iCampus
- Grading Policy: TBD
 - Midterm? Maybe, Final? Yes
 - SQL homework
 - DB application project
 - B-tree



Textbooks

- Database System Concepts (7th Ed.)
 - Avi Silberschatz
 - Henry F. Korth
 - S. Sudarshan





Chap 1. Introduction





Database Systems

- What is a DBMS (database management system)?
 - No clear definition
- What is Database?
 - Definition: A database is an **integrated collection of data** usually stored on secondary storage (disk).





Database Systems

- Ideally, all data is stored together in a uniform way without redundancy
 - **Example:** Information about a university:
 - Students
 - Courses
 - Faculty
 - Departments
 - Students taking courses
 - Faculty teaching courses
 - Faculty advising students
 - All the above examples can be organized using a DBMS
- Q: Why not use a simple file system to store this information?
 - There isn't enough **structure** in the information stored in files.



Reason for using a DBMS rather than a file system

- Reason for using a DBMS rather than a file system
 - Centralized control
 - The database administrator has complete control over the system and tunes the internal structure to achieve better performance
 - Uniform Access & Control of the data
 - Users don't have to write a different program for every single query
 - Control of redundancy – utilize structure
 - Don't store address many times
 - Control of integrity
 - No salary can be below \$5000.



Reason for using a DBMS rather than a file system (Cont'd)

- Reason for using a DBMS rather than a file system
 - Data Independence
 - Applications don't depend on data representation & storage organization
 - eg. Changing the storage structure from B-tree to hashing has no affect on the functionality of existing software
 - Concurrency control
 - Many users are allowed to access the same data at the same time
 - Recovery
 - Data is not lost when system fails for whatever reason



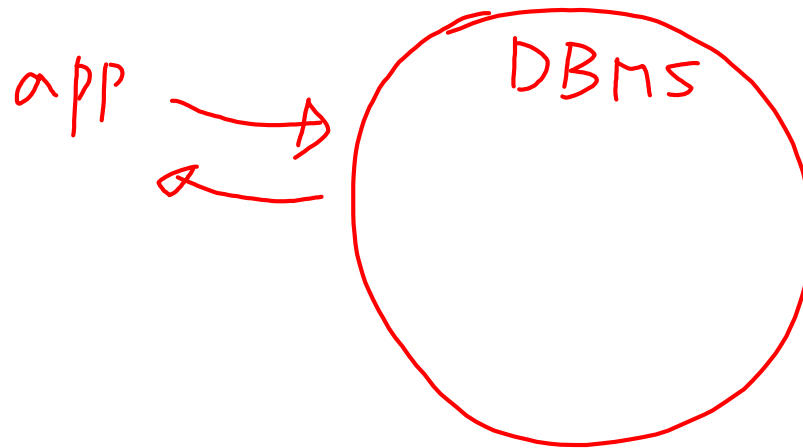
Reason for using a DBMS rather than a file system (Cont'd)

- Reason for using a DBMS rather than a file system
 - Security
 - The system takes care so that no user retrieves or updates any data in the database they are not authorized to access
 - eg. Seeing someone else's salary
 - Data consistency (goes partially with redundancy)
 - If many copies exist, one must update all copies otherwise inconsistencies develop and, whenever duplications do exist update all.
- DBMS can provide this and more!



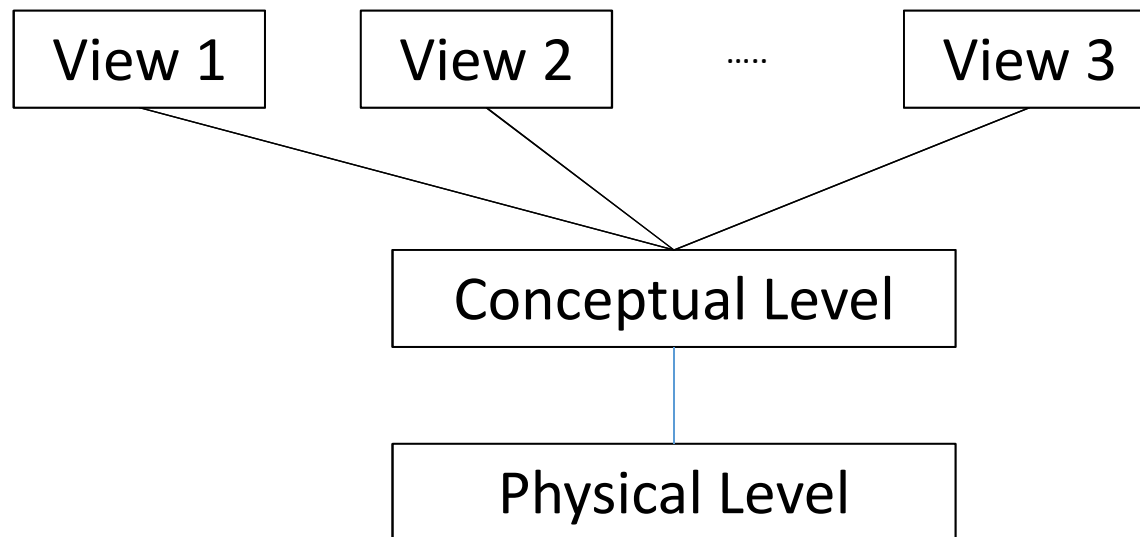
Data Abstraction

- A major purpose of a database system is to provide users with an abstract view of the data
 - However data must be retrieved efficiently.
- DBMS hides certain details of how data is stored and maintained
 - Note: many users of database systems are not computer trained, thus, the complexity of the system is hidden by several layers of abstraction



Data Abstraction

An architecture for a database system(abstractions)



Data Abstraction

■ Conceptual level

- eg.

Type employee = record

name:string;

street:string;

salary:integer;

end;

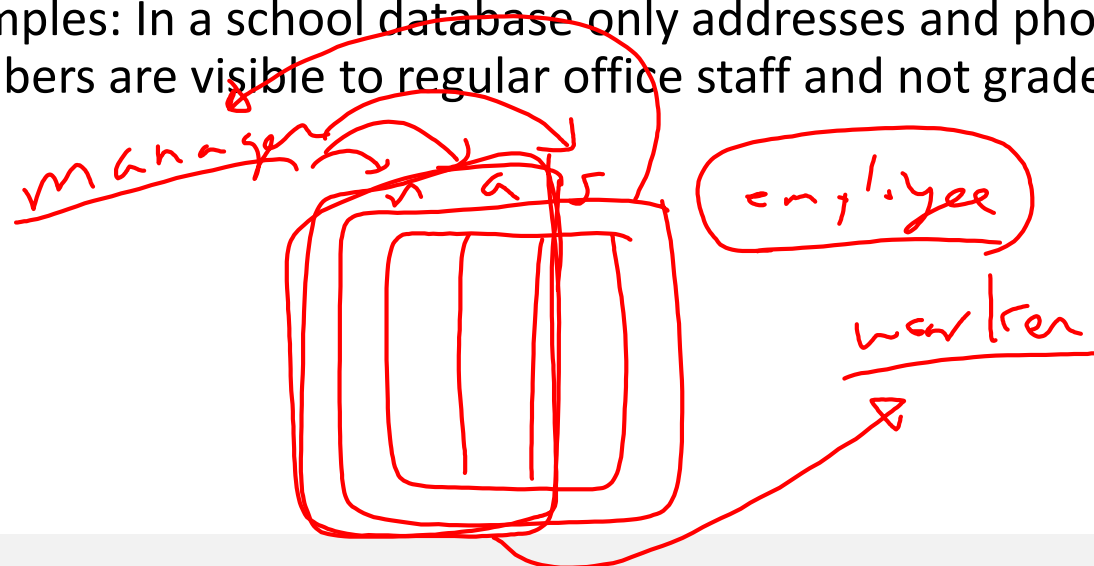
employer

- This describes in an abstracted way what an employee record looks like
 - it contains 3 fields of specified type
 - Does not specify how the record is stored
- Note: a particular application may have many records specified
 - eg.
customer record
account record
 - Note: conceptual level indicates how these records are related.



Data Abstraction

- Physical level
 - How and employee record is actually stored
 - eg. B-tree, Hash Table, etc
- View level (highest level of abstraction)
 - Only a portion of the database is visible to a user. The entire database is not visible. There may be many views.
 - Examples: In a school database only addresses and phone numbers are visible to regular office staff and not grades.



Data Models

- Q: When designing a database system, how do we begin? What is needed and helpful?
- A: A collection of tools for describing:
 - Data
 - Data relationships
 - Data semantics
 - Data constraints



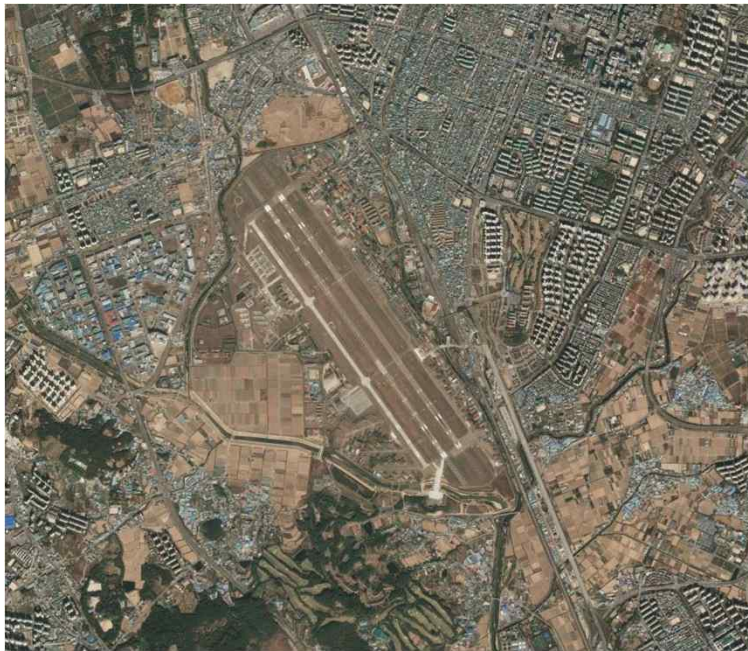


Why use models?

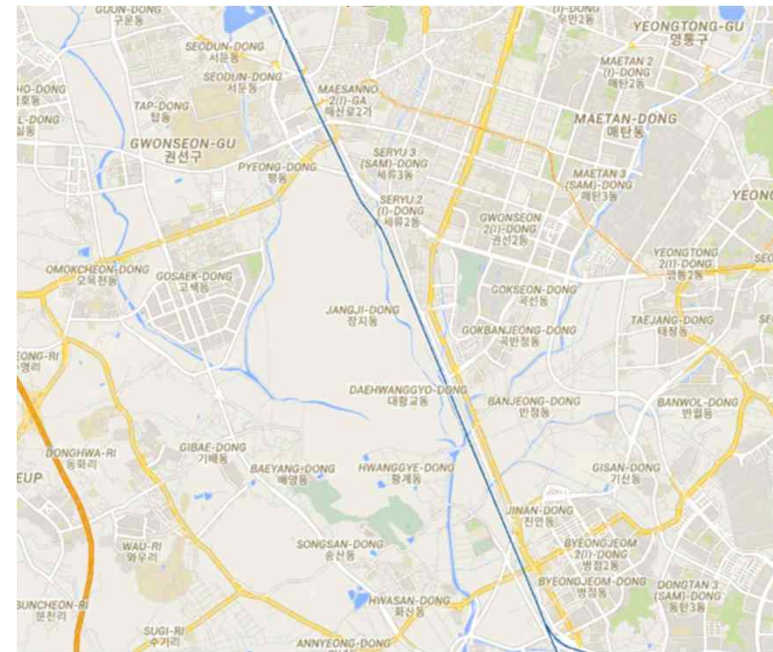
- Models can be useful when we want to examine or manage parts of the real world.
 - The costs of using a model are often considerably lower than the costs of using or experimenting with the real world itself.
 - Examples:
 - Airplane simulator
 - Nuclear power plant simulator
 - Molecular structure prediction & manipulation
 - Flood warning system
 - Model of U.S. economy
 - Model of a building
 - Map



Why use models?



Real World (Satellite Image)



Model (Map)





Why use models?

- Many important aspects of a model can be illustrated by the map example
 - Specify constraints to map matters on the colors of various parts of the map; rivers don't have country lines running through them etc.
 - A model is a means of easy communication
 - The users of a model must have a certain amount of knowledge in common
 - A model only emphasizes selected aspects of reality
 - Air Force Air Base is hidden in the map.
 - A model is described in some language
 - A model can be erroneous



Data Models

- Data Model : **high-level specification** how a data is structured and used
- Collection of tools for describing
 - Data
 - Data relationships
 - Data semantics
 - Data constraints
- **Relational model** (Entity-Relationship data model)
- Object-based data models
- Semi-structured data model (XML)
- Other old models:
 - Network model
 - Hierarchical model





Textbook Roadmap

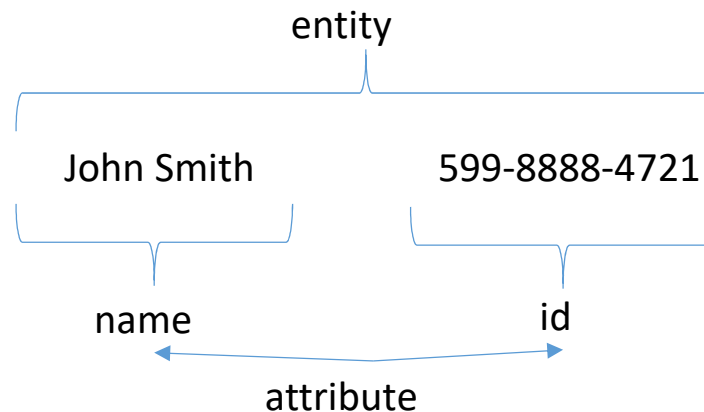
- Part I: Relational Database
- Part II: Database Design
- Part III: Data Storage and Querying
- Part IV: System Architecture
- and more..



Part I: Quick view of Relational Model

■ Relationship model

- Real world consists of objects(entities)
- Real world consists of relationships among the objects
- Def: entity
 - Is an object that is distinguishable from other objects by a specific set of attributes
 - eg.



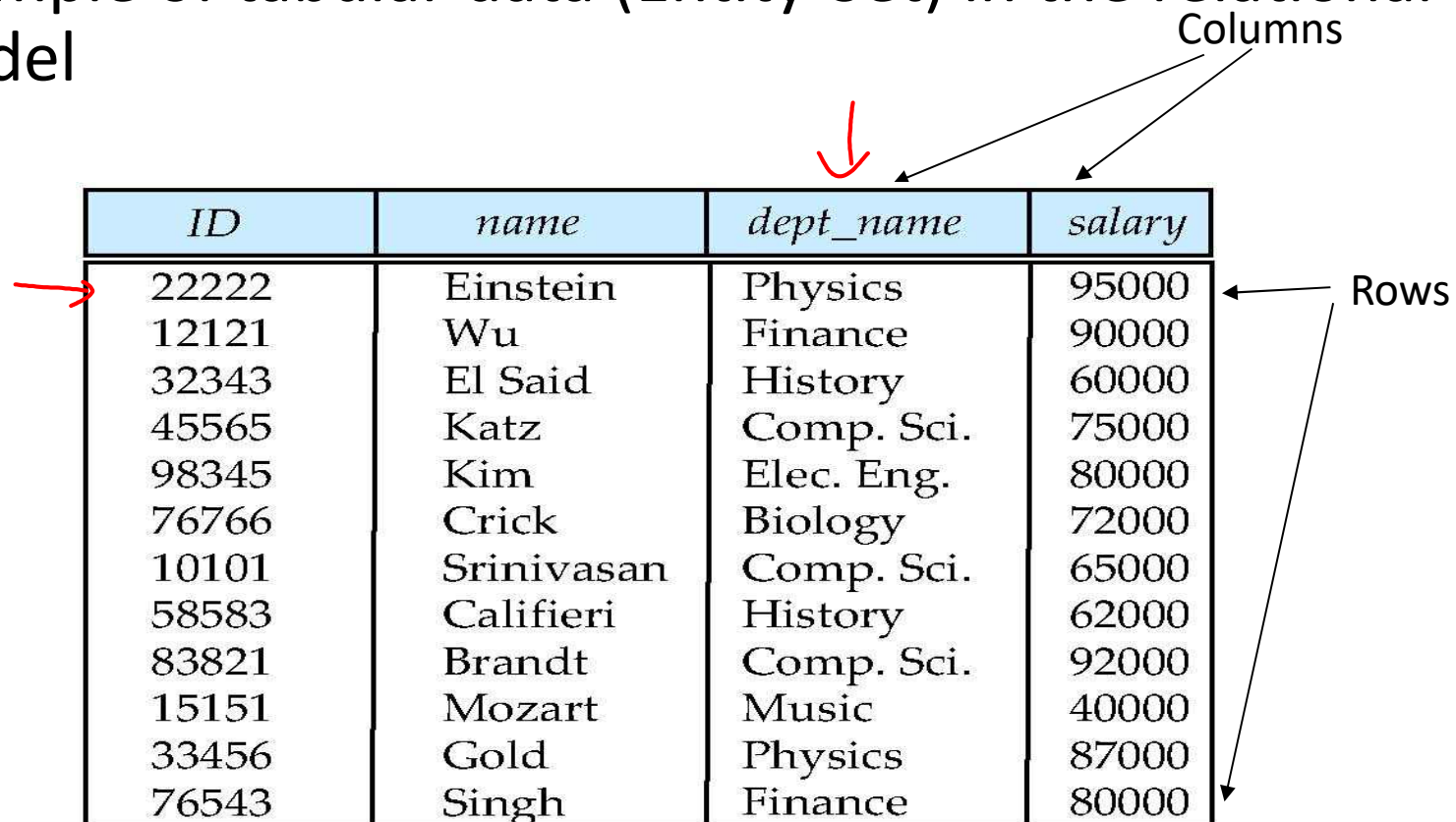
Part I: Quick view of Relational Model

- Entity-Relationship model (Cont'd)
 - Def: Relationship is an association among several entities
 - eg. CustAcct – associates a customer with each account they have
 - Def: Entity Set – set of all entities of the same type
 - Def: Relationship Set – set of all relationships of the same type
 - Def: Mapping Cardinalities – a constraint(?) database must conform to the number of entities to which another entity can be associated via a relationship set



Part I: Quick view of Relational Model

- Example of tabular data (Entity-Set) in the relational model





<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

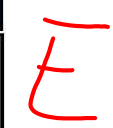
(a) The *instructor* table






A Sample Relational Database




<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	<u>Physics</u>	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



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



(b) The *department* table



Instances and Schemas

class A }

■ Schema

- Similar to types and variables in programming languages
- Logical Schema – the overall logical structure of the database
- Physical schema – the overall physical structure of the database

■ Instance – the actual content of the database at a particular time

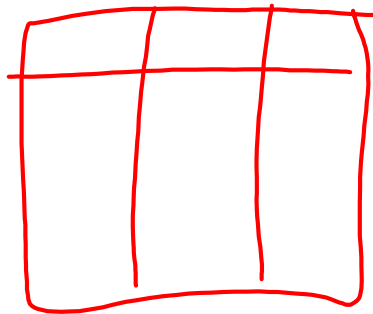
- Analogous to the value of a variable

A obj,



Physical Data Independence

- 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.



SQL

- **SQL**: Structured Query Language (Chapter 3-5)
 - pronounced as /ˈsiːkwəl/ or /ˈɛs kjuː ˈɛl/
 - It's often pronounced as /ˈsiːkwəl/ due to IBM's SEQUEL (Structured English Query Language)
 - consists of DML (Data Manipulation Language) and DDL (Data Definition Language).
- Recall that there are various types of programming languages
 - **Procedural** – user specifies what data is required and how to get those data
 - e.g.) C, C++, Java, Python, etc
 - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
 - e.g.) SQL, Prolog, etc



Data Definition Language (DDL)

- Used to define database schema

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

- Table definition, integrity constraints, authorization, etc. are stored in data dictionary (also called metadata catalog) in DBMS.



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 basically 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 easier to learn and use than procedural DMLs.



SQL Query Language

- SQL query language is nonprocedural.
 - A query takes input table(s) and always returns a single table.
 - Example to find all instructors in Comp. Sci. dept

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



Part II: Database Design

- Is there any problem with this design?

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





Design Approaches

■ Entity Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
 - Entity: a “thing” or “object” that is distinguishable from other objects
 - Described by a set of *attributes*
 - Relationship: an association among several entities
- Represented by an *entity-relationship (ER) diagram*:

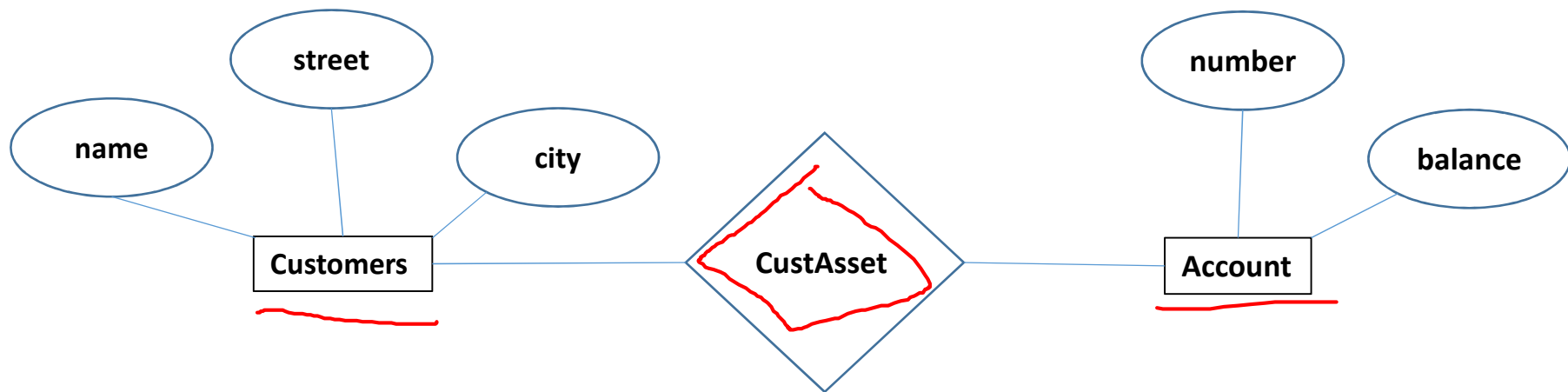
■ Normalization Theory

- Formalize what designs are bad, and test for them



Entity-Relationship model

- Graphical Representation of database via an E-R diagram



- Rectangles – represent entity sets
- Ellipses – represent entity attributes
- Diamonds – represent relationships among entity set
- Lines – link attributes to entity sets & entity sets to relationships



Part III and IV: Database Management Systems

- What's under the hood of DBMS?
 - Learn how to implement DBMS





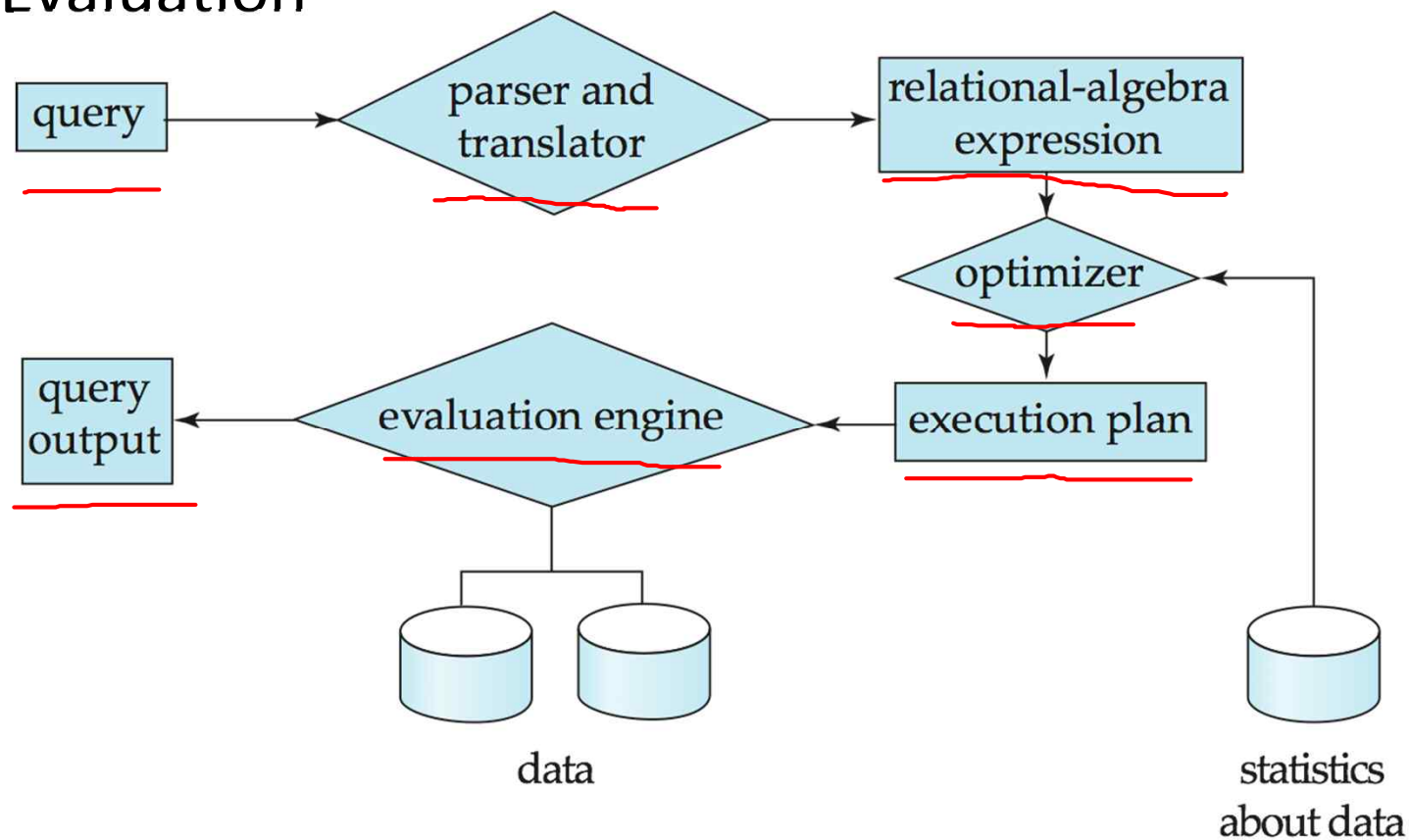
Storage Management

- **Storage manager** is 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 for the following tasks:
 - Interaction with the file manager
 - Efficient storing, retrieving and updating of data
- Issues:
 - Storage access (Ch. 10)
 - File organization (Ch. 10)
 - Indexing and hashing (Ch. 11)



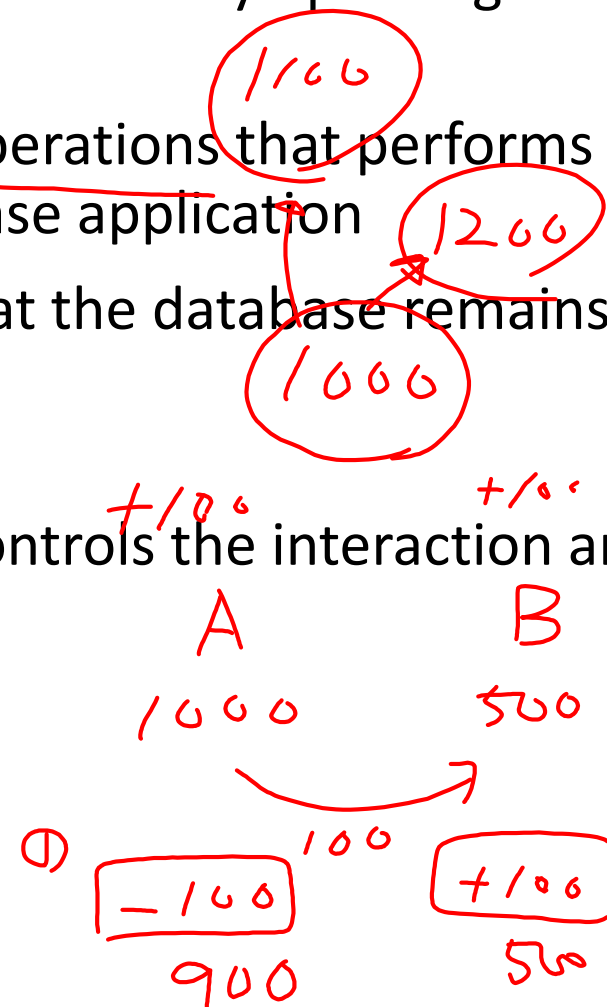
Query Processing

1. Parsing and translation (Ch. 12)
2. Optimization (Ch. 13)
3. Evaluation

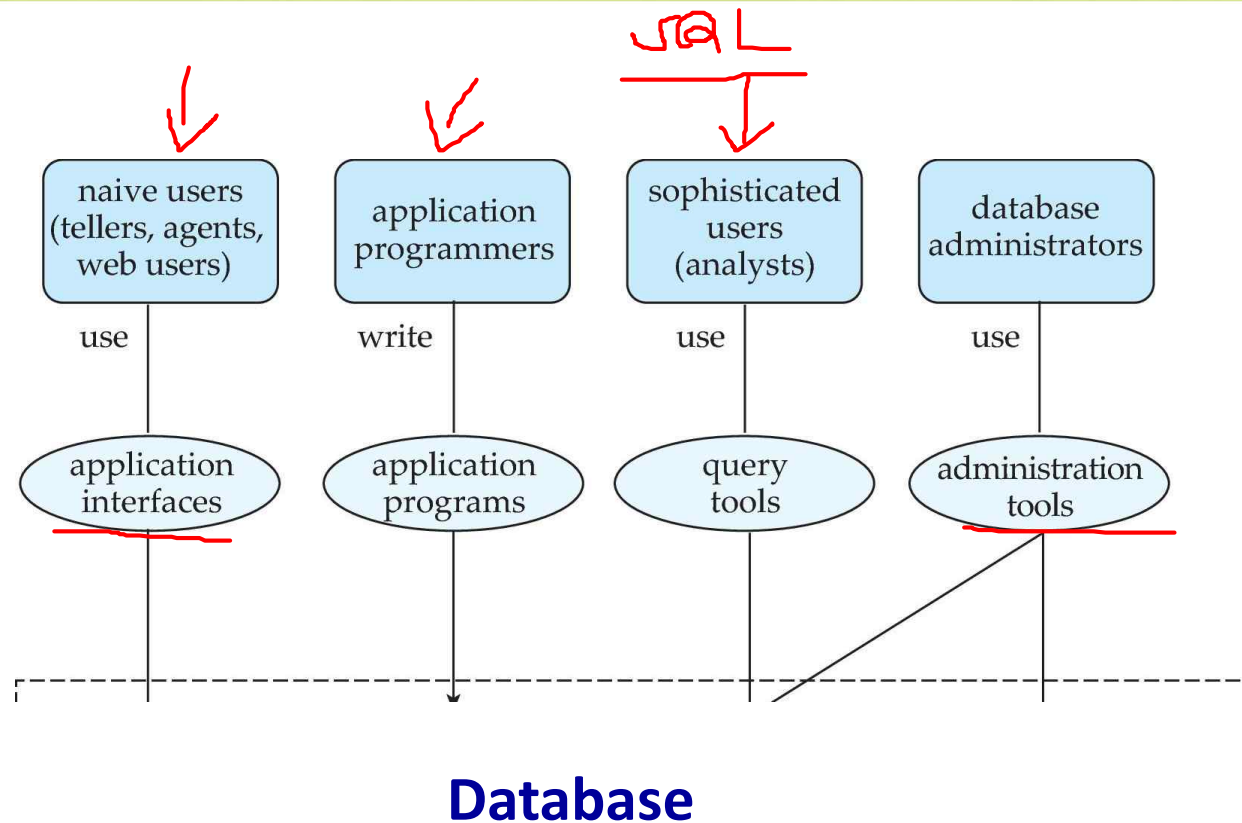


Transaction Management

- 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 manager** ensures that the database remains in a consistent state
 - Recovery scheme (Ch. 16)
- **Concurrency-control manager** controls the interaction among the concurrent transactions
 - Lock-based protocols (Ch. 15)



Database Users and Administrators



Database System Internals

