INFO 210: Database Management Systems

Topic 1

Introduction to databases
Overview of database design

supplementary material:
"A First Course in Database Systems" Ch. 1

Topic 1 outline

- Part 1: What is a DBMS?
- Part 2: Introducing data models
- Part 3: Introducing SQL
- Part 4: Sets
- Part 5: Summary

Part 1: What is a DBMS?

- A database is a large integrated collection of data
  - Models a real-world enterprise
  - Augments raw data with metadata, to give meaning to the data
- A Database Management System (DBMS) is a software package designed to store and manage databases
- Our focus: Relational Database Management Systems (RDBMS), centered around the relational model
  - Entities (e.g., students and courses)
  - Relationships (e.g., students taking courses)

The role of a DBMS

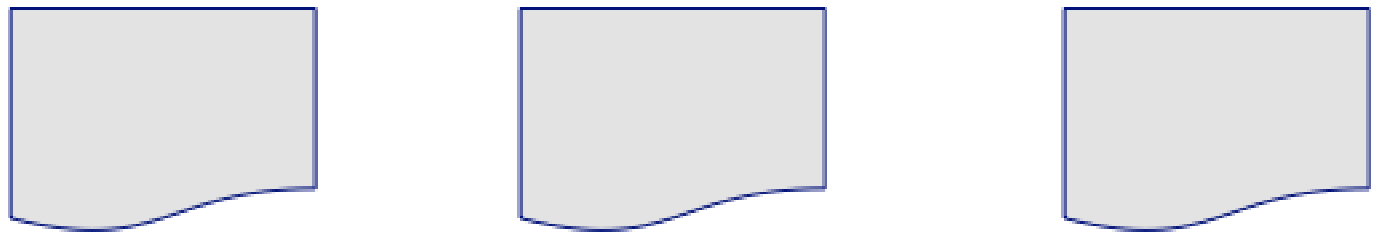
- Serves as an intermediary between the user and the database
- Enables the sharing of data
- Supports multiple alternative views of the data

A traditional database application

- Build a system to store and access information about
  - students
  - courses
  - professors
  - who takes what, who teaches what
- Functionality
  - record enrollment information
  - compute GPA for each student after each term
  - analyze student enrollment and performance in different courses, majors, departments etc

Can we do this without a DBMS?

- Sure we can! We can store data in files:
  - *students.txt*   *courses.txt*   *professors.txt*



A programmer can write a C or Java program to implement specific tasks.

Different users can execute these tasks (run the programs)

Without a DBMS

- Task: Enroll "Mary" in "INFO210"
  - write a Java program that does the following:

```
read *students.txt*
read *courses.txt*
find & update record "Mary"
find & update record "INFO210"
write *students.txt*
write *courses.txt*
```

Without a DBMS

- System crashes

read *students.txt*
read *courses.txt*
find & update record “Mary”
find & update record “INFO210
write *students.txt*
write *courses.txt*

crash

- Large data sets (say, 50GB)

- Simultaneous access by many users

Without a DBMS

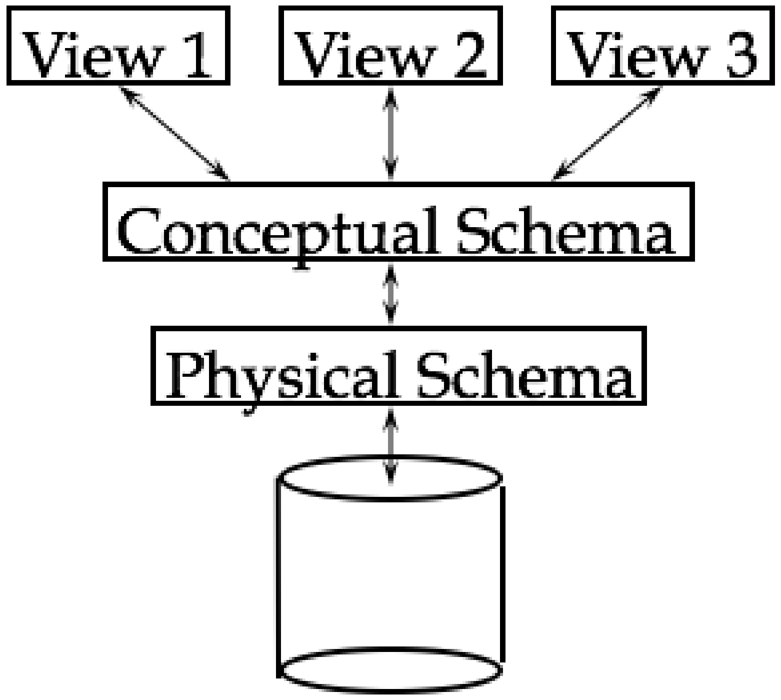
- Format of *students.txt* changes

```
read students.txt
read courses.txt
find & update record "Mary"
find & update record "INFO210"
write students.txt
write courses.txt
```

- Certain users should only be allowed to see parts of the file *courses.txt*

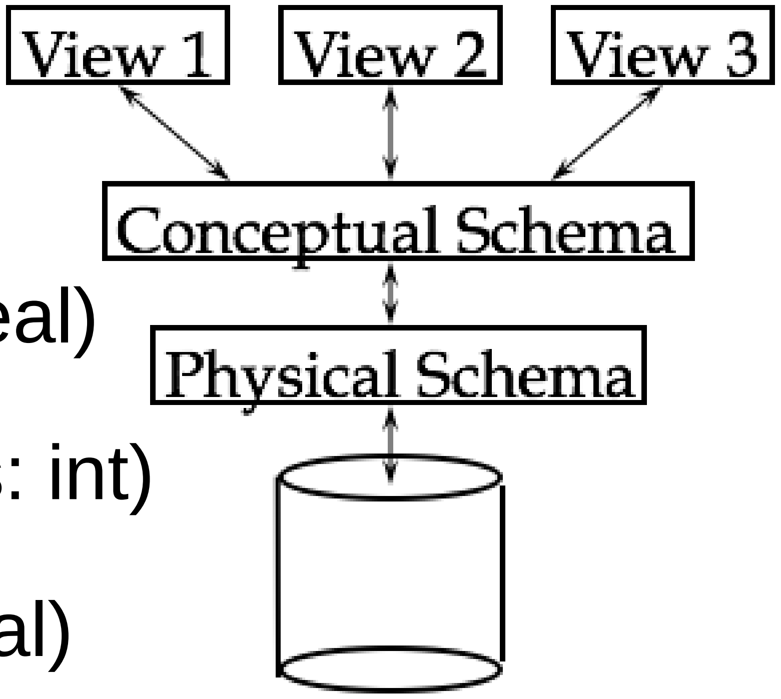
Enter a DBMS

- When in doubt - introduce a level (or levels) of abstraction!
- Many views (external schemas), one conceptual (logical) schema, one physical schema
  - Views describe how users see the data.
  - A conceptual schema defines the logical structure.
  - A physical schema describes the files and indexes used.



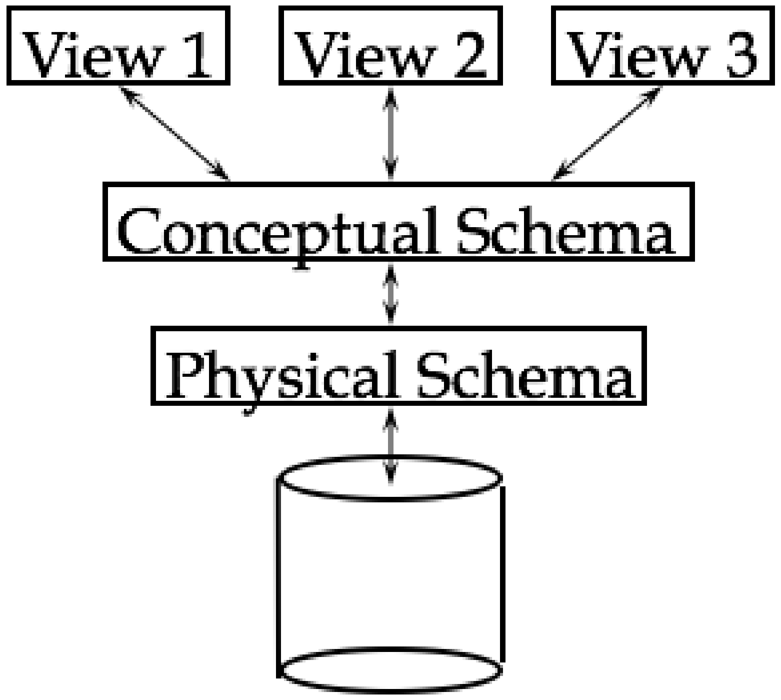
Example

- External schema (view)
  - Course_Info(cid: string, enrollment: int)
- Conceptual schema
  - Students (sid:string, name: string, gpa: real)
  - Courses (cid: string, name: string, credits: int)
  - Enrolled (sid: string, cid: string, grade: real)
- Physical schema
  - Courses, Enrolled stored as unordered files
  - Students stored sorted by sid



A key concept: data independence

- Applications (and users) are insulated from how data is structured and stored
- Logical data independence: protection from changes in logical structure of the data
- Physical data independence: protection from changes in physical structure of the data



So, why use a DBMS?

- Technical reasons
  - Data independence and efficient access
  - Reduced application development time
  - Data integrity and security
  - Uniform data administration
  - Concurrent access, recovery from crashes

- Business reasons
  - Reusing existing approaches makes data management more cost-effective!

A word of caution

- DBMS give us the tools to make data management more convenient, efficient, cost-effective
- But, like any technology, databases will only make our lives easier when used appropriately!

Part 2: Data models

- A data model is a collection of concepts for describing data.
- A schema is a description of a particular collection of data, using a given data model.

- We will look at two data models in this course
  - relational model - used to design logical and external schemas in relational databases
  - entity-relationship (ER) model - used for conceptual design

ER model basics

- Introduced by Chen in 1976.
- An entity is a real-world object distinguishable from other objects. Entities are described using a set of attributes.
- An attribute has a name and a datatype.
- An entity set is a collection of similar entities: all have the same set of attributes.
- A relationship is an association among 2 or more entities.
  - can be 1-to-1, 1-to-many, many-to-many
  - we can also express other interesting constraints on the relationships

The relational model

- Introduced by Edgar F. Codd in 1970 (Turing award)
- At the heart of relational database systems
  - the basic abstraction is a *relation* (a table)
  - *tuples* are stored in rows
  - *attributes* of tuples are stored in columns
  - conceptually, a relation is a set of tuples
- Why this model?
  - Simple yet powerful
  - Great for processing very large data sets in bulk

Some terminology

- The relational model is implemented (with some variations) by several RDBMS
  - IBM DB2, Oracle, Sybase, Microsoft SQL Server, mySQL, Postgress, ...
- Conceptually: relations are sets of tuples
- Reality: relations are implemented as bags of tuples (multisets).
- Keys allow us to unambiguously refer to tuples.

Relations

- Relations are used to describe two concepts
  - Entities, e.g., student, course, department
  - Relationships, e.g., student-enrolled-in-course, course-offered-by-department

Students

key

sid	name	GPA
1234	Joe	3.2
5678	Ann	4.0

Departments

did	name
1	INFO
2	MATH

Courses

cid	did	name	credits
110	1	HCI	3
210	1	Databases	3
312	2	Linear Algebra	3

Enrollment

did	cid	sid	term	grade
1	110	1234	SP11	A
2	312	1234	SP11	B
1	210	5678	FA12	A-
1	210	1234	FA12	B+

An alternative schema

Students

sid	name	GPA
1234	Joe	3.2
5678	Ann	4.0

Departments

did	name
1	INFO
2	MATH

Enrollment

did	sid	cid	**name**	**credits**	term	grade
1	1234	110	HCI	3	SP11	A
2	1234	312	Linear Algebra	3	SP11	B
1	5678	210	Databases	3	FA12	A-
1	1234	210	Databases	3	FA12	B+

Is this a well-designed schema?

I have one word for you: *normalization*!

Part 3: SQL

- SQL ("seekwel") stands for "Structured Query Language"
- Made up of 2 parts:
  - Data Definition Language (DDL) - used to create or modify a relational schema
  - Data Manipulation Language (DML) - used to retrieve or modify data in a schema
  - We call SQL statements - queries

Creating relations (DDL)

```
create table XXX(
  attribute dataType [keywords],
  …
);
```

```
create table Students (
  sid number primary key,
  name varchar(128) not null,
  gpa number
);

create table Departments (
  did number primary key,
  name varchar(128) not null
);

create table Courses (
  cid number,
  did number,
  name varchar(128) unique,
  credits number default 3,
  primary key (cid, did),
  foreign key (did) references Departments(did)
);
```

Students

key

sid	name	GPA
1234	Joe	3.2
5678	Ann	4.0

Primary keys in a relation make sure that tuples can be distinguished from one another

Courses

cid	did	name	credits
110	1	HCI	3
210	1	Databases	3
110	2	Linear Algebra	3

Foreign keys enforce **referential integrity**

Creating relations (DDL)

```
create table Enrollment (
  did number,
  cid number,
  sid number,
  term varchar(32),
  grade char(2),
  primary key (cid, did, sid),
  foreign key (cid, did) references Courses(cid, did),
  foreign key (sid) references Students(sid)
 );
```

Students

sid	name	GPA
1234	Joe	3.2
5678	Ann	4.0

Enrollment

did	cid	sid	term	grade
1	110	1234	SP11	A
2	312	1234	SP11	B
1	210	5678	FA12	A-
1	210	1234	FA12	B+

Courses

cid	did	name	credits
110	1	HCI	3
210	1	Databases	3
312	2	Linear Algebra	3

Dropping relations (DDL)

```
drop table XXX;
```

```
drop table Enrollment;
drop table Courses;
drop table Departments;
drop table Students;
```

$$\begin{tikzcd}
\text{Students} & & \text{Departments} \\
& & \text{Courses} \arrow[u] \\
& \text{Enrollment} \arrow[uul] \arrow[ur] &
\end{tikzcd}$$

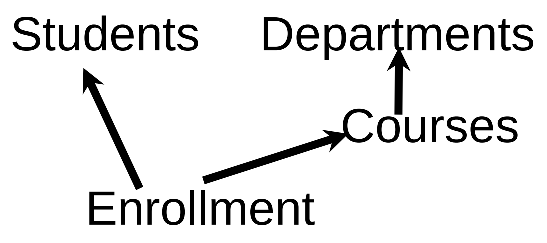
why dropped in this particular order?

is this the only possible order?

Dropping relations (DDL)

```
drop table XXX;
```

```
drop table Enrollment;
drop table Courses;
drop table Departments;
drop table Students;
```



$$\begin{tikzcd}
\text{Students} & & \text{Departments} \\
& & \text{Courses} \arrow[u] \\
& \text{Enrollment} \arrow[uul] \arrow[ur] &
\end{tikzcd}$$

why dropped in this particular order?

is this the only possible order?

Accessing data: queries (DML)

```
select  XXX
from XXX;
```

```
select *
from Students;

select *
from Students
where gpa > 3.0;

select name
from Students
where gpa > 3.0;

update Students
set name = 'Mike'
where sid = 4;
```

```
select *
from Enrollment
where grade is null;

select did
from Enrollment;

select distinct did
from Enrollment;
```

Accessing data: queries (DML)

this is a *join*

```
select *
from  Students, Enrollment
where Students.sid = Enrollment.sid
and   Enrollment.did = 100;
```

Accessing data: queries (DML)

these queries use *aggregation*

```
select count(*)
from Students;

select min(gpa), max(gpa)
from Students;

select term, count(*), avg(grade)
from Enrollment
group by term;
```

A key concept: SQL is declarative

- SQL is a declarative language: we say *what* we want to do, not *how* to do it
- This is important for two reasons:

1. usability
2. efficiency

Populating relations (DML)

```
insert into Students (sid, name, GPA) values (1, 'Jane', 4);

insert into Departments values (100, 'Math');

insert into Departments (name, did) values ('CIS', 200);

insert into Departments (name) values ('Italian');
```

do all of these work?

NB: SQL is not case-sensitive
NB: We use single quotes to quote strings

Part 4: Sets

- This course requires you to have a solid understanding of sets.
- Homework 1 has a sets section, it is your responsibility to refresh your memory on this material (or to learn it if you are not familiar with it)
- A quick refresher follows

Overview of sets

- A good overview at
- http://en.wikipedia.org/wiki/Set_theory#Basic_concepts
  - A set is an *unordered collection* of objects
  - An object belonging to a set is an *element* (or a *member*) of that set, written $a \in A$
  - We denote sets with capital letters, elements with lowercase letters
  - Examples of sets

Overview of sets (II)

- An empty set is a set that contains no elements. $A = \emptyset$

- We say that A is a *subset* of set B if all elements of A are also members of B. Then B is a *superset* of A. $A \subseteq B$ $B \supseteq A$

- A is a *proper subset* of B if A is a subset of B, and there exists at least one element $b \notin A$ such that $b \in B$ $A \subset B$ $B \supset A$

- Any set A a subset of itself. A proper subset?

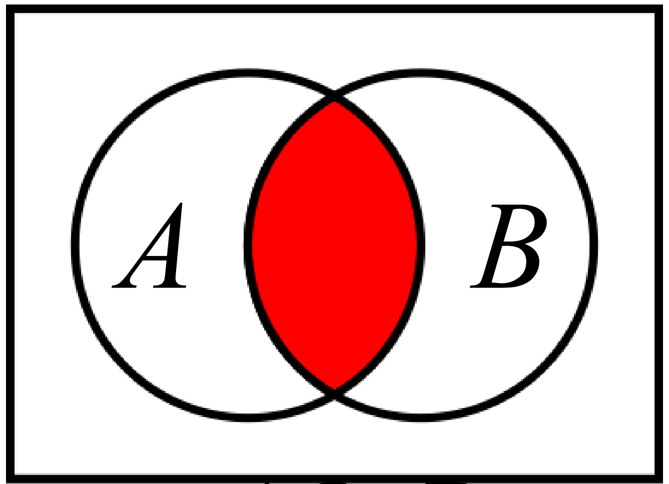
- $A = \emptyset$ is a subset of any set. A proper subset?

Overview of sets (III)

- The size of a set, denoted |A|, is the number of elements in the set
- The size of an empty set is 0
- Some sets are of infinite size, e.g., the set of all integers, all prime numbers, etc

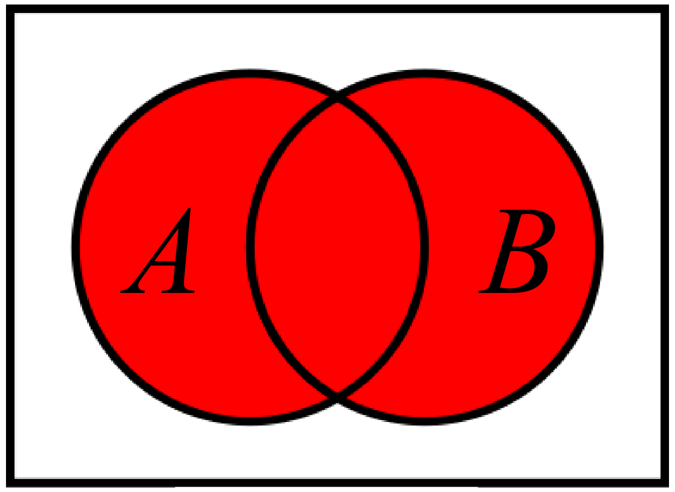
Set operations

Intersection



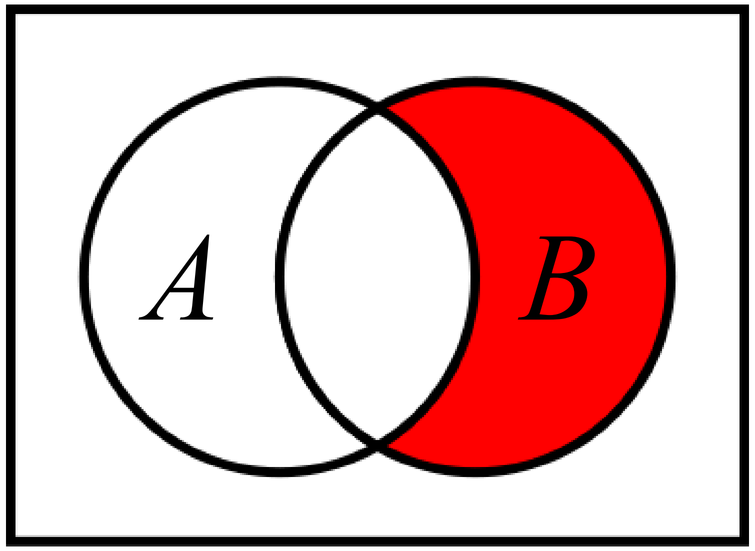
A Ç *B*

Union



$A \cup B$

Difference

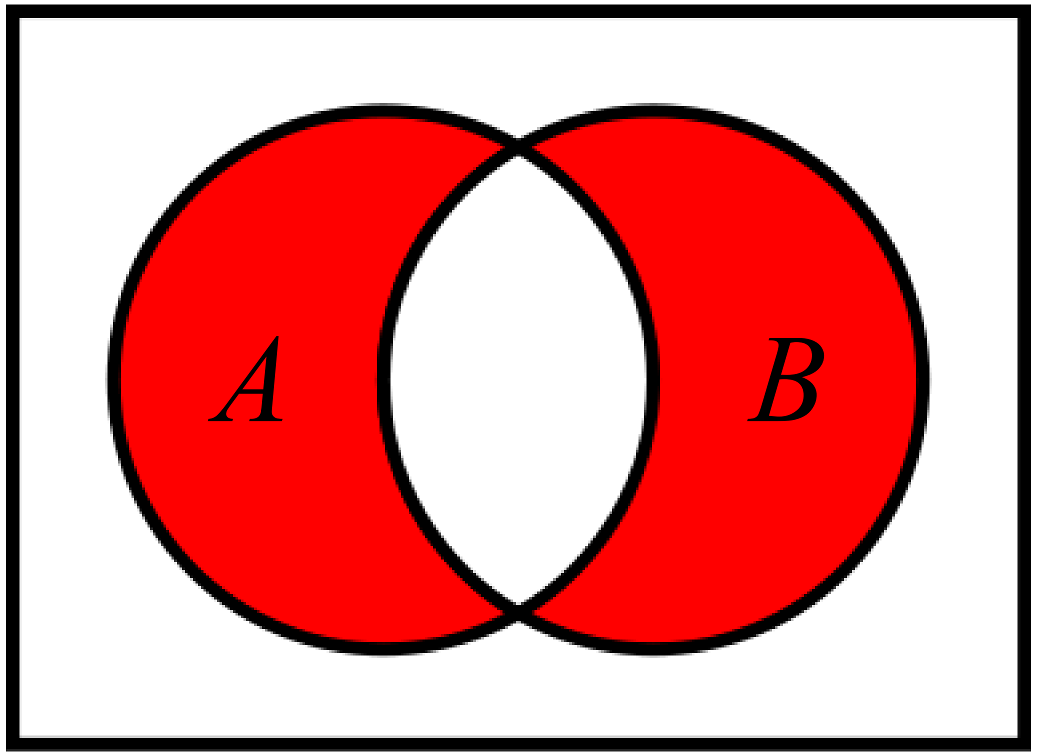


$B \setminus A$

Venn diagrams from
http://en.wikipedia.org/wiki/Venn_diagram

Set operations (II)

How can you express *symmetric difference* using other set operations?



Cartesian product

- Cartesian product of A and B, $A \times B$ denoted is a set of ordered pairs (a, b), where $a \in A, b \in B$

Example:

$$A = \{1, 2, 3\} \qquad B = \{3, 4\}$$

$$A \times B = \{(1,3), (1,4), (2,3), (2,4), (3,3), (3,4)\}$$

Part 5: Summary

- Take-home message: Databases are cool
- Differences between databases and file systems
- The relational model
- The ER model
- Example of a database schema
- Examples of SQL queries
- A key concept: data independence
- A key concept: SQL is declarative

Useful abbreviations

- DB - database
- DBMS - database management system
- RDBMS - relational database management system
- DBA - database administrator
- SQL - structured query language
    - DDL - data definition language
    - DML - data manipulation language
- ER - entity-relationship
- ERM - entity-relationship model