

# Database Architecture and Data Model

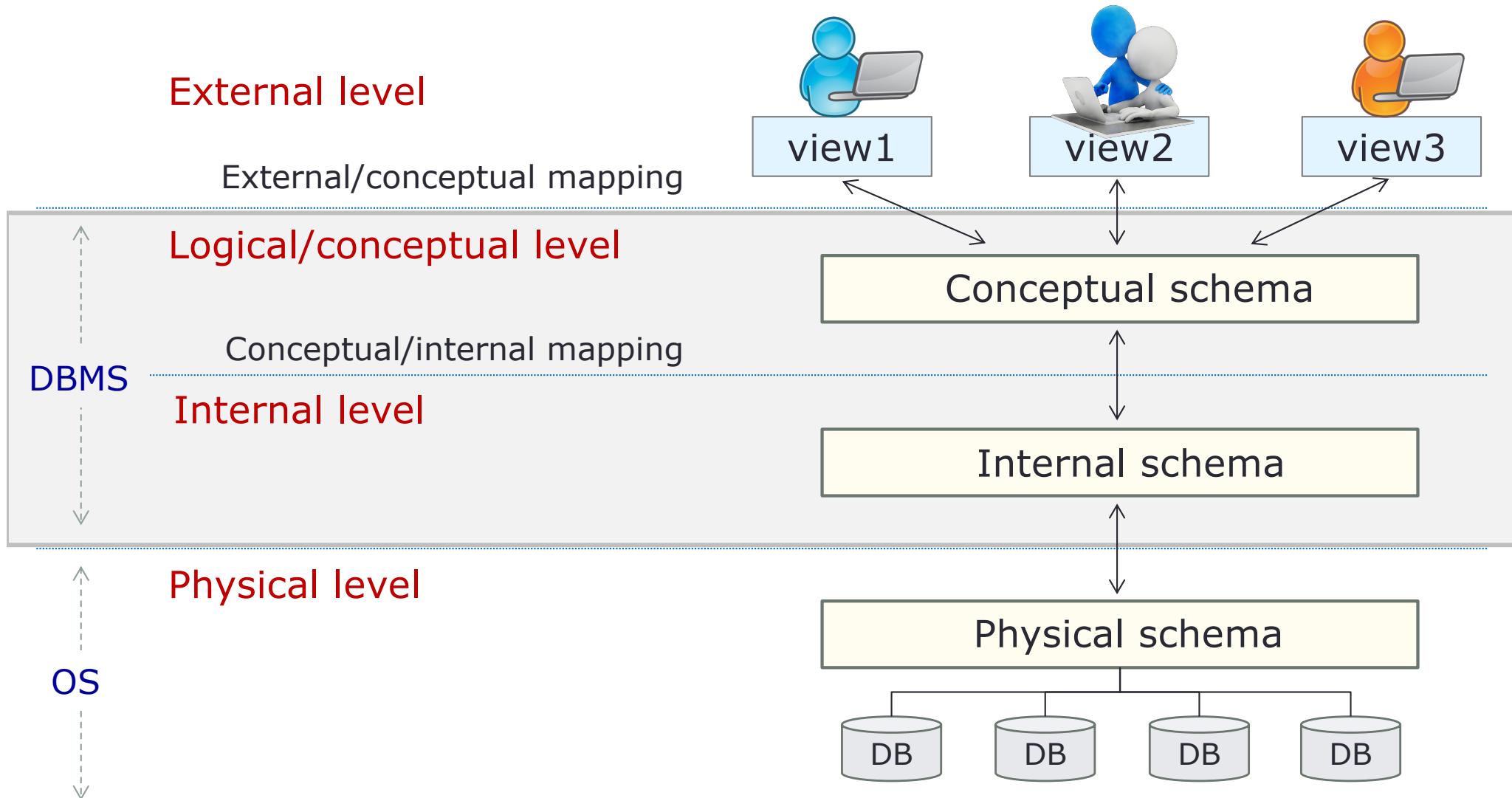
---

CS 4750  
**Database Systems**

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.2]  
[C.M. Ricardo and S.D. Urban, Database Illuminated, Ch. 2.6-2.7]

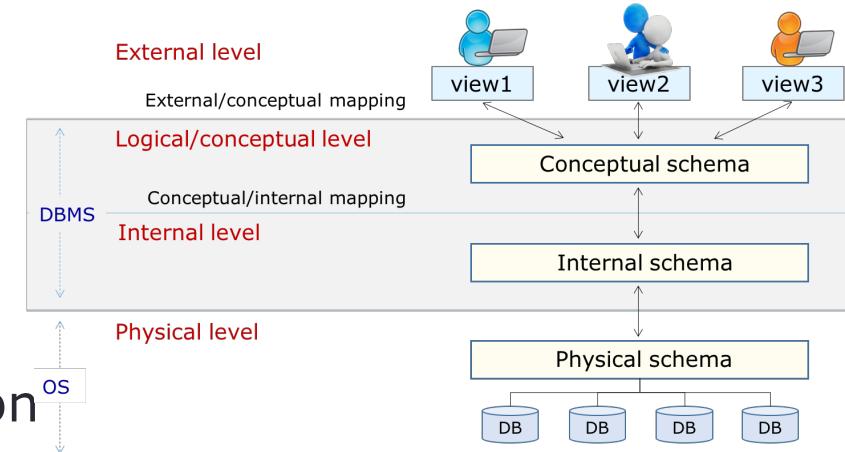
# Levels of Database Architecture

Databases are stored as files of records stored on disks



# External Level

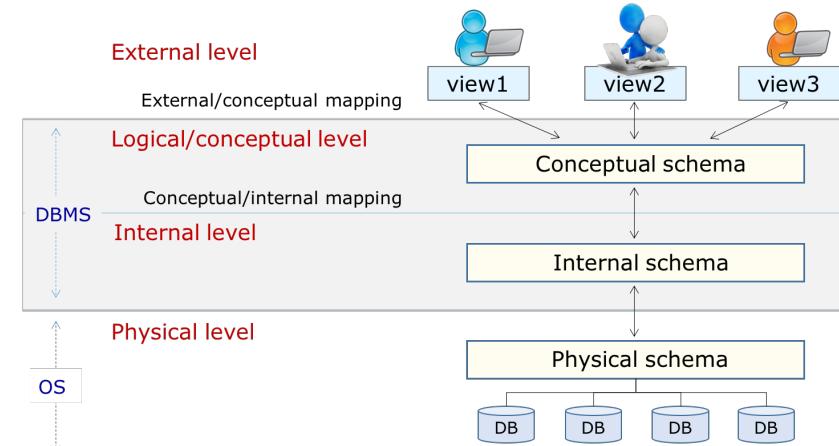
- Different users see different views



- Different views may consist of
  - Different contents, or
  - Same contents, different representation
- Some views may include **virtual data** (or **calculated data**)
  - Calculated age on demand
- **External view** – a collection of external records
- **External record** – a record as seen by a particular user
- **External schema** – describes an external view
- DBMS uses the external schema to create a user interface

# Logical Level

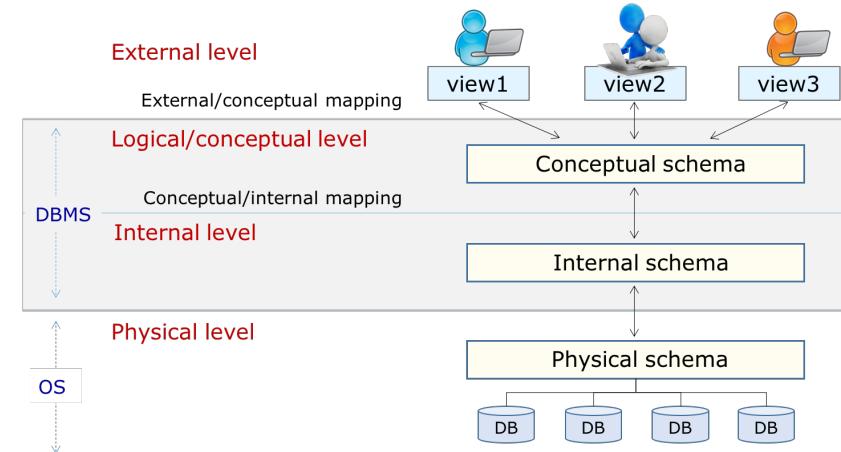
- Include description of all the data that is available to be shared



- **Logical schema** – a complete description of the information content of the database
- DBMS uses the logical schema to create logical record interface
- Logical record interface – **conceptual level** and **internal level** (defines what are visible / invisible to external level and physical level)
- **Logical model** – a collection of logical records (“comprehensive view of the user’s mini-world”)

# Internal and Physical Levels

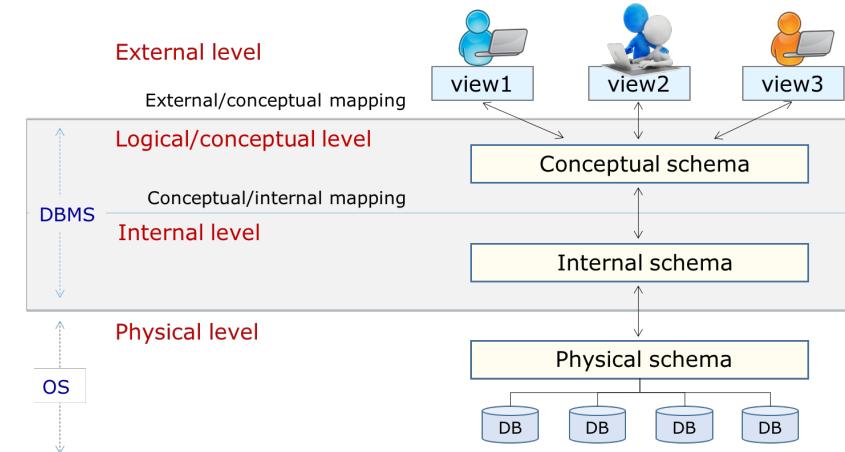
- **Internal level** deals with physical implementation of the database, responsible by DBMS
- **Physical level** is managed by operating system
- **Internal schema** – a complete description of the internal model
  - Describe how data are represented, how records are sequenced, what indexes exist, ...



# Data Independence

**Data independence** – upper levels are unaffected by changes to lower levels

- **Logical data independence** – changes to logical model do not impact external models
- **Physical data independence** – changes to internal model or physical do not impact logical model



[Ref: emoji by Ekarin Apirakthanakorn]

# How do we describe information?

Let's revisit our Brainstorm-Scheduler activity (start designing a database for an appointment scheduler).

How do we describe information?

PatientID	Name	Vaccine	Dose_number	Date
123	Humpty	Pfizer	3	08/20/2023
345	Dumpty	Pfizer	2	08/24/2023
567	Wacky	Moderna	1	07/17/2023

# Data Model

**Data model** = a collection of concepts or notations for describing the data in a database

Three parts of a data model:

**Structure, integrity, manipulation**

Definition of relations  
and contents

Constraints on  
the DB's contents

Actions that can be done  
on the DB's contents

Allow us to deal with data **conceptually** without  
having to think about implementation

Data  
independence

# Data Model

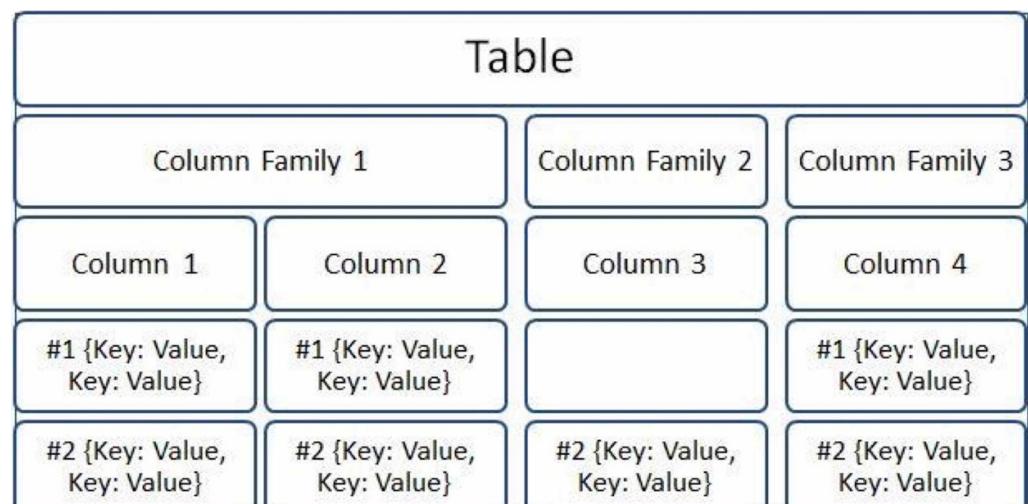
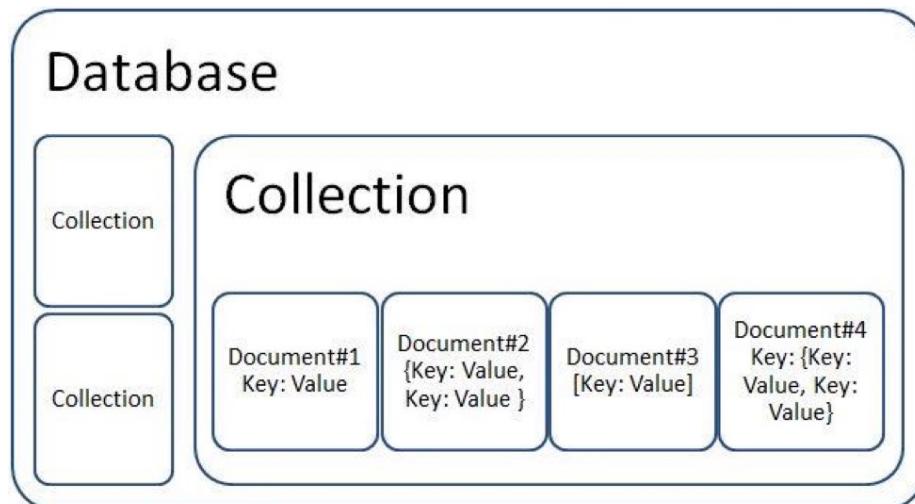
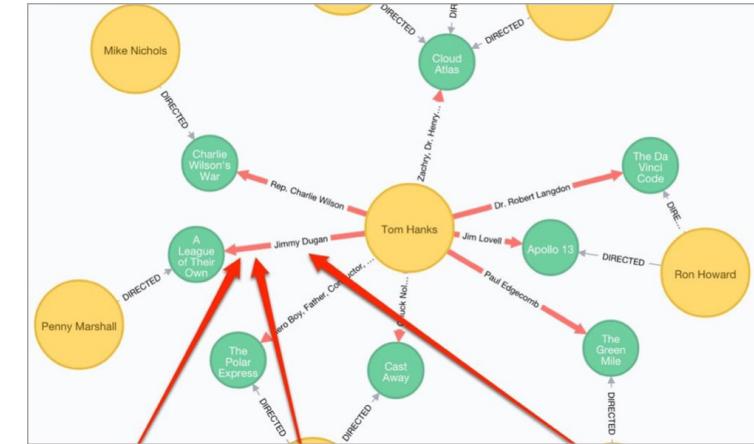
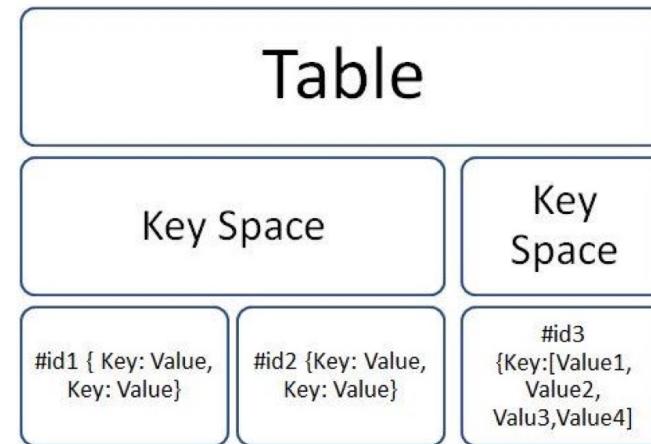
- Relational
  - Key/value
  - Graph
  - Document
  - Column-family
  - Array / matrix
  - Hierarchical
  - Network
- ← Most DBMSs

Title	Year	Length	Genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's World	1992	95	comedy

# Data Model

- Relational
- Key/value
- Graph
- Document
- Column-family

NoSQL



# Data Model

- Relational
- Key/value
- Graph
- Document
- Column-family
- Array / matrix
- Hierarchical
- Network

← Machine learning

← Obsolete / rare

	<i>Joe</i>	<i>Ann</i>	<i>Bob</i>
<i>Joe</i>	0	1	0
<i>Ann</i>	1	0	1
<i>Bob</i>	0	1	0

# Data Model

- Relational

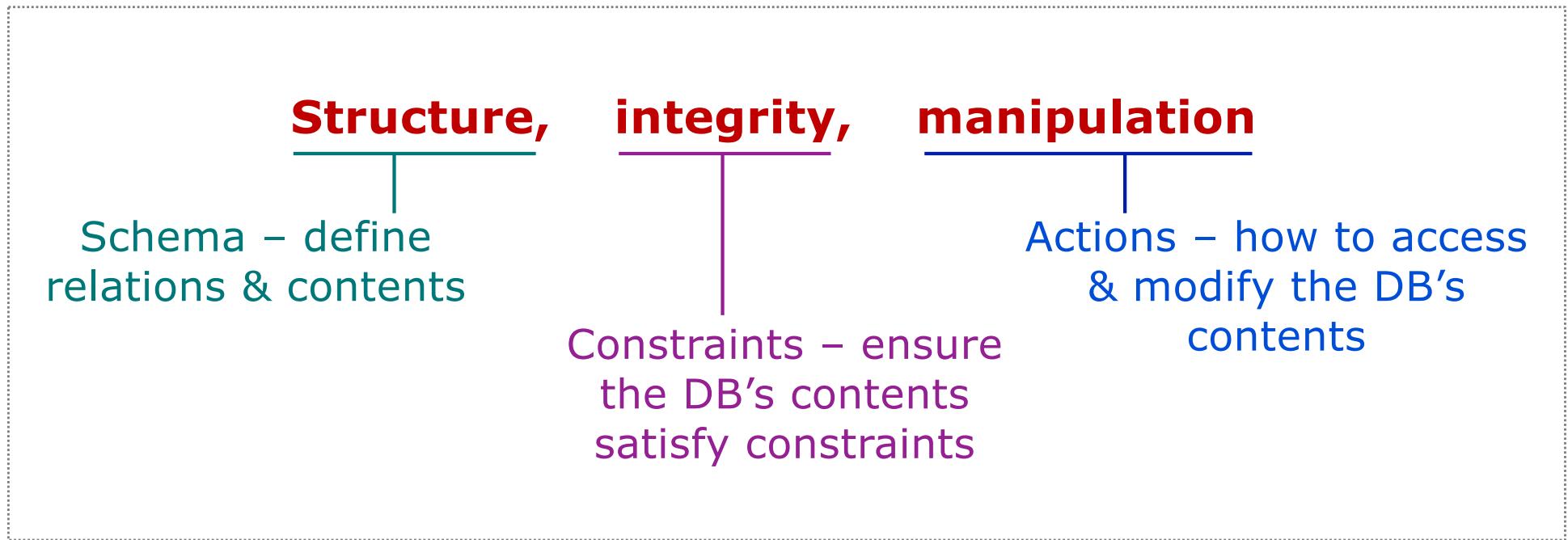
← This course

- Key/value
- Graph
- Document
- Column-family
- Array / matrix
- Hierarchical
- Network

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's World	1992	95	comedy

# Relational Model

Three parts:



Relational model gives us a single way to represent data as a two-dimensional table called a “**relation**”

# Relational Model

## Relational model

Relational Algebra

*relation name* ↗ movies ↓ *attribute*

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

← *tuple*

Relation = **set** of tuples (thus no duplicate), has no order

## Actual DB / SQL

SQL

*table name* ↗ movies ↓ *column*

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

← *row / record*

Table = **list** of rows (thus can have duplicates), has order

# Relational Model

The diagram illustrates a relational model. A blue arrow labeled "relation name" points to the word "movies" above a table. A green arrow labeled "attribute" points to the word "title" in the table header. A purple arrow labeled "tuple" points to the third row of the table.

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

**Relation** = unordered set of tuples that contain the relationship of attributes

n-ary relation = table with n columns

**Tuple** = set of attribute values ("domain") in the relation

- Values are (normally) **atomic**
- The special value **NULL** is a member of every domain

# Schema

**Schema** – logical design of the database (describe the tables),  
not generally change (note order of attributes)

movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

Write a **schema statement** for the above relation

schema      movies(title, year, length, genre)  
              movies(title:string, year:integer, length:integer, genre:string)

# Instance

**Instance** – data stored in the database at a given time (set of tuples), change from time to time

movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

tuple → Star wars, 1977, 124, sciFi ← instance

Write the selected tuple

tuple      (Star wars, 1977, 124, sciFi)

Write the selected instance

instance    {(Star wars, 1977, 124, sciFi),(Wayne's world, 1992, 95, comedy) }

# Recap: Schema and Instance

**Describe the difference between schema and instance**

- **Schema** – describe the relation (the attributes), doesn't change
- **Instance** – actual values of data in the relation at that moment, change constantly

# Keys of Relations

To specify which tuple, use the attribute values of a tuple to distinguish (uniquely identify the tuple)

movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

Make use of the real-world fact

Assumption:

- A movie may be remade in different years
- For each year, many movies may be made

Write a **schema statement** for the above relation, indicate the key of the relation

movies(title, year, length, genre)



# Super Key – Candidate Key - Primary Key

Super key – any attribute(s) that can uniquely identify a tuple

Candidate key – minimal super key

Primary key – a candidate key that is the most important key

Students\_info

computingID	SSN	name
mi1y	111-11-1111	Mickey
mi2e	222-22-2222	Minnie
do3d	333-33-3333	Donald
da4y	444-44-4444	Daisy
do5d	555-55-5555	Donald

Identify super key

computingID  
SSN  
computingID+SSN  
computingID+name  
SSN+name  
computingID+SSN+name

Start with assumptions

# Super Key – Candidate Key - Primary Key

Super key – any attribute(s) that can uniquely identify a tuple

Candidate key – minimal super key

Primary key – a candidate key that is the most important key

Students\_info

computingID	SSN	name
mi1y	111-11-1111	Mickey
mi2e	222-22-2222	Minnie
do3d	333-33-3333	Donald
da4y	444-44-4444	Daisy
do5d	555-55-5555	Donald

Identify candidate key

computingID  
SSN

# Super Key – Candidate Key - Primary Key

Super key – any attribute(s) that can uniquely identify a tuple

Candidate key – minimal super key

Primary key – a candidate key that is the most important key

Students\_info

computingID	SSN	name
mi1y	111-11-1111	Mickey
mi2e	222-22-2222	Minnie
do3d	333-33-3333	Donald
da4y	444-44-4444	Daisy
do5d	555-55-5555	Donald

Identify primary key

computingID

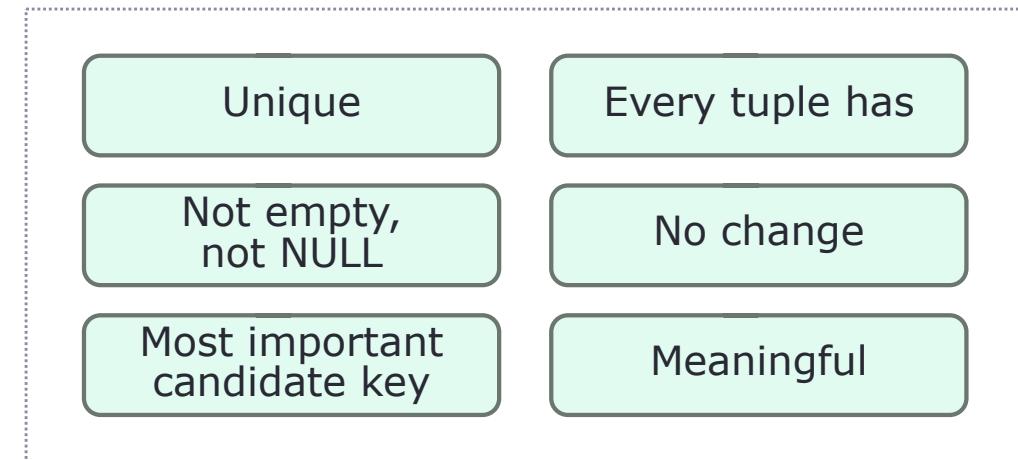
# Primary Keys

A relation's **primary key** uniquely identifies a single tuple

movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

The characteristics of primary keys



# Primary Keys

Some DBMSs automatically create an internal primary key if none is defined for the relation

Auto-generation of unique integer primary keys:

- **SEQUENCE** (SQL: 2003)

movies

<b>id</b>	<b>title</b>	<b>year</b>	<b>length</b>	<b>genre</b>
123	Gone with the wind	1939	231	drama
456	Star wars	1977	124	sciFi
789	Wayne's world	1992	95	comedy

- **AUTO\_INCREMENT** (MySQL)

# Foreign Keys

A **foreign key** specifies that an attribute from one relation has to map to a tuple in another relation

Attribute(s) that uniquely identify a row in another table

Artist(id, name, year, country)			
<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
123	Mickey	1992	USA
456	Minnie	1992	USA
789	Donald	1994	USA

## Primary key

Album(id, name, artists, year)

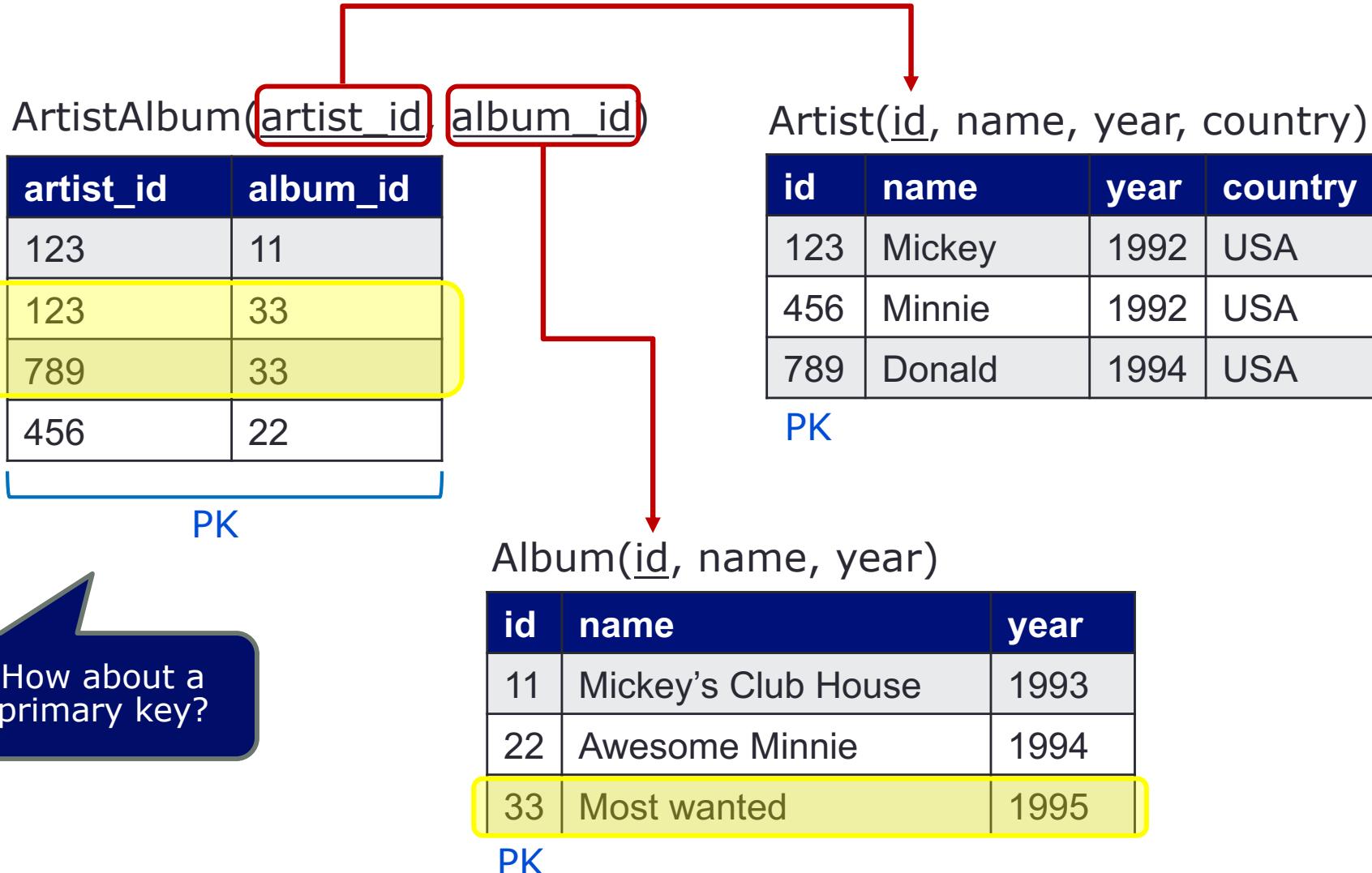
<b>id</b>	<b>name</b>	<b>artists</b>	<b>year</b>
11	Mickey's Club House	123	1993
22	Awesome Minnie	456	1994
33	Most wanted	789	1995

## Foreign key

## How about 2 artists (123 and 789)?

33	Most wanted	123	1995
33	Most wanted	789	1995

# Foreign Keys



# Recap: Keys

## List properties of primary keys

- (has to be a candidate key)
- Unique
- Not NULL
- Not empty
- Doesn't change
- Meaningful, given the domain (recommendation)
- Typically small in size (recommendation)

# Characteristics of Relational Model

- Originally defined with **Set semantics** (no duplicate tuples)
- Attributes are typed and **static** (INTEGER, FLOAT, ...)
- Tables are **flat**
- Attribute values are **atomic**
- Order of tuples doesn't matter

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
123	Mickey	1992	USA
456	Minnie	1992	USA
789	Donald	1994	USA

=

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
456	Minnie	1992	USA
789	Donald	1994	USA
123	Mickey	1992	USA

# Recap: Relational Model

Does the following table satisfy the characteristics of relational model?

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
123	Mickey	1992	USA
456	Minnie	1992	USA
789	Donald	1994	USA
567	Humpty	2016	USA
567	Humpty	2016	USA

No. Violates set semantics

# Recap: Relational Model (2)

Does the following table satisfy the characteristics of relational model?

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
123	Mickey	1992	USA
456	Minney	1992	USA
789	Donald	1994	USA
567	Humpty	banana	USA

No. Violates attribute type, assuming INT

# Recap: Relational Model (3)

Does the following table satisfy the characteristics of relational model?

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>						
			<b>country</b>						
123	Mickey	1992	<table border="1"><thead><tr><th><b>country</b></th><th><b>note</b></th></tr></thead><tbody><tr><td>USA</td><td>Some info</td></tr><tr><td>USA</td><td>Another info</td></tr></tbody></table>	<b>country</b>	<b>note</b>	USA	Some info	USA	Another info
<b>country</b>	<b>note</b>								
USA	Some info								
USA	Another info								
456	Minnie	1992	USA						
789	Donald	1994	USA						
567	Humpty	2016	USA						

No. No sub-tables allowed;  
must be flat and atomic

# Recap: Relational Model (4)

How are these data actually stored?

<b>id</b>	<b>name</b>	<b>year</b>	<b>country</b>
123	Mickey	1992	USA
456	Minnie	1992	USA
789	Donald	1994	USA
567	Humpty	2016	USA

Don't know. Don't care.

**“Physical Data Independence”**

# Wrap-Up

---

- Database architecture – separation of concerns
- Data independence – physical and logical
- Data model – schema, integrity, manipulation
- Relational model
- Key constraints – super key, candidate key, primary key, foreign key

## What's next?

- DB design – using E-R model
- Mapping E-R model to schemas

# (Additional) Practice Activity

---

This is an open-ended activity, consisting of 3 relations. For each relation, think about the data to be stored. You should also consider the type of data. Write a schema for each relation. Feel free to make any reasonable assumption.

Consider the following description. Write schemas that represent the data models for the given description.

# (Additional) Practice Activity (2)

## Relation: **Movies**

Each movie has a *title* and *year*; *title* and *year* together uniquely identify the movie. *Length* and *genre* are maintained for each movie. Each movie is associated with a *studioName* which tells us the studio that owns the movie, and *producerC#* which is an integer that represents the producer of the movie.

---

## Relation: **MovieStar**

This relation tells us something about stars. It maintains the *name* of the movie star, address, gender, and birthdate. The gender can be a single character (*M* or *F*). Birthday is of type “date,” which might be a character string of a special form.

---

## Relation: **StarsIn**

This relation connects movies to the stars of that movie, and likewise connects a stars to the movies in which they appeared. A star might appear in multiple movies in one year.