

# Introduction to relational databases

INFO/CS 2300:  
Intermediate Web Design and  
Programming

# Course overview

We started with PHP

Project 2 due today at 5pm

Then JavaScript

Homework 1 on Javascript due Tuesday

Ajax: Javascript and PHP

Friday's section and quiz is on JavaScript,  
jQuery, Ajax

The next few weeks - databases and SQL

# Homework 1: Using JavaScript / jQuery

We give you HTML and CSS for a page. You write the JavaScript / jQuery to change the the page in response to user input.

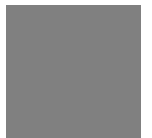
Will be released by this evening via Piazza.

Due Tuesday February 24th at 5 pm.

# P2: Limitations of file storage

What was difficult about storing data in a text file?

What challenges would emerge if there were thousands of items to store?



# P2: Limitations of file storage

## Steve's list

- Have to write the whole file to edit / delete
- Hold file in memory or frequent read/write
- Managing delimiters
- Enforcing data types
- Find requires looping through the whole thing
- Requiring that an item belong to a list

# P2: Limitations of file storage

## Class list

- Can't have multiple people using same file
- Or conditions
- Fuss with white space
- Rewriting the file requires rewriting all
- Finding an entry requires searching every line
- memory and performance issues
- order of reading writing matters

# Rollercoasters

Roller coaster	Type	Park	# of rides	Increment
Top Thrill Dragster	Steel	Cedar Point	1	Increment
El Toro	Wood	Six Flags Great Adventure	2	Increment
Leviathan	Steel	Canada's Wonderland	1	Increment
Intimidator 305	Steel	Kings Dominion	7	Increment
Mean Streak	Wood	Cedar Point	1	Increment
	Wood ▼			Add new

The data in the table is a database. But there are limitations when storing data in a text file.

- Type **options are hard coded** in the HTML
- Inefficient**: have to rewrite the file to increment rides
- delimiters are a nuisance**

# What is...?

A *database* is a collection of related data.

A *database management system* (DBMS, *database system*) is software used to manage a database efficiently, so that it will persist safely over long periods of time.



# A DBMS...

- Allows for **persistent storage** of very large amounts of information, and **efficient access** to it.
- Implements a “**query language**” (SQL) to store, access, and modify the data.
- Can **enforce** that the stored data is consistent in certain ways.
- **Manages access** by many users.

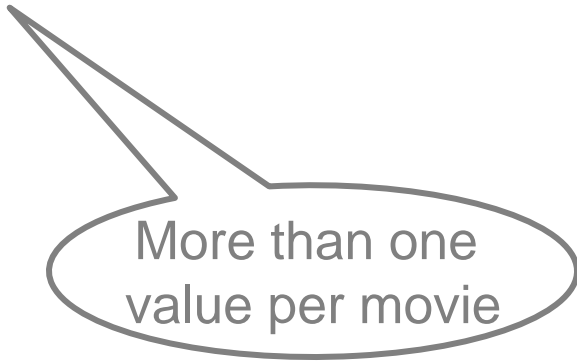
# A running example: Movies

Fields associated with a movie:

# A running example: Movies

Fields associated with a movie:

- year
- director
- title
- genre
- rating
- length
- ...
- actor / actress



More than one  
value per movie

# Relational databases

# Why “relational”?

The data is organized in *relations* (or *tables*).

The columns are called *attributes* or *fields*.

The rows are called *tuples* or *records*. Each row must be unique.

Each column / field has a set of allowable values called its *domain*.



domain for year is 4 digit integer

title	year	lead actor	lead actress
Sleepless in Seattle	1993	Tom Hanks	Meg Ryan
Holiday	1938	Cary Grant	Katherine Hepburn
Sabrina	1954	Humphrey Bogart	Audrey Hepburn

# Table schema

The table **schema** is a named set of attributes (fields) together with their associated domains (allowed values).

Field (attribute)	Type	Size	Allow null
title	VARCHAR	64	No
year	INT		No
lead_actor	VARCHAR	32	Yes
lead_actress	VARCHAR	32	Yes

# Issues with schema

Choosing a good one is not always straightforward.

title	year	lead actor	lead actress
The Philadelphia Story	1940	James Stewart Cary Grant	Katherine Hepburn

The data fits in the 32 character limit specified but...

Often better to have multiple tables (relations).

# Database schema

The associated relations (tables) form a *relational database*.

**title**                      **year**    **length**

Sleepless in Seattle	1993	105 mins.
Holiday	1938	95 mins.
Sabrina	1954	113 mins.

**name**                      **year**    **title**

Katherine Hepburn	1938	Holiday
Katherine Hepburn	1940	The Philadelphia Story
James Stewart	1940	The Philadelphia Story



# Constraining the data

# Unique rows


A table must have **uniquely identifiable rows** (records)

title	year	length
Harry Potter and the Sorcerer's Stone	2001	152
The Dark Knight	2008	152
Planet of the Apes	1968	112
Planet of the Apes	2001	119

How would we naturally talk about a record in this table?

# Primary Key

A *primary key* is the field(s) selected to uniquely identify records in the table.



title	year	length
Harry Potter and the Sorcerer's Stone	2001	152
The Dark Knight	2008	152
Planet of the Apes	1968	112
Planet of the Apes	2001	119

# Primary Key: Natural vs Surrogate

A **natural key** is made up of fields ( attributes ) that exist in the real world. ( title, year )

A **surrogate key** adds an artificial field ( movie\_id )

movie_id	title	year	length
1	Harry Potter and the Sorcerer's Stone	2001	152
2	The Dark Knight	2008	152
3	Planet of the Apes	1968	112
4	Planet of the Apes	2001	119

Sometimes **fields appear in more than one table**, implying a relationship between the two.

primary key

title	year	length
Sleepless in Seattle	1993	105 mins.
Holiday	1938	95 mins.
The Philadelphia Story	1940	112 mins.
Sabrina	1954	113 mins.
Planet of the Apes	1968	112
Planet of the Apes	2001	119

name	year	title
Katherine Hepburn	1938	Holiday
Katherine Hepburn	1940	The Philadelphia Story
James Stewart	1940	The Philadelphia Story
Charlton Heston	1968	Planet of the Apes
Charlton Heston	2001	Planet of the Apes

The same tables with a surrogate key for the movie table

movie_id	title	year	length
1	Sleepless in Seattle	1993	105
2	Holiday	1938	95
3	The Philadelphia Story	1940	112
4	Planet of the Apes	1968	112
5	Planet of the Apes	2001	119

name	movie_id
Katherine Hepburn	2
Katherine Hepburn	3
James Stewart	3
Charlton Heston	4
Charlton Heston	5

primary key

# Primary Key: Natural vs Surrogate

Natural Key	Surrogate Key
title, year	movie_id
Disadvantage: it can be cumbersome to work with multiple fields	Advantage: only one field needed
Advantage: meaning to humans	Disadvantage: no inherent meaning to humans
Disadvantage: If business rules change, key must too	Advantage: remains independent of business rules

# Foreign keys

If a field is (or fields are) a primary key in another table, we call this a *foreign key*. It *refers to* or *targets* the other table.

movie_id	title	year	length
2	Holiday	1938	95
3	The Philadelphia Story	1940	112
4	Planet of the Apes	1968	112
5	Planet of the Apes	2001	119

name	movie_id
Katherine Hepburn	2
Katherine Hepburn	3
James Stewart	3
Charlton Heston	4
Charlton Heston	5



# Why foreign keys?

Why might foreign keys be interesting?

movie_id	title	year	length
2	Holiday	1938	95
3	The Philadelphia Story	1940	112
4	Planet of the Apes	1968	112
5	Planet of the Apes	2001	119

name	movie_id
Katherine Hepburn	2
Katherine Hepburn	3
James Stewart	3
Charlton Heston	4
Charlton Heston	5

What is the **average length of a Katherine Hepburn movie?**

# Why else foreign keys?

movie_id	title	year	length
2	Holiday	1938	95
3	The Philadelphia Story	1940	112
4	Planet of the Apes	1968	112
5	Planet of the Apes	2001	119

name	movie_id
Katherine Hepburn	2
Katherine Hepburn	3
James Stewart	3
Charlton Heston	4
Charlton Heston	5

Can impose **constraints** such as preventing “Holiday” from being deleted from the Movie table or insisting that an actor's movie exist in the movie list. Remember P2?

# Nulls

Sometimes have “null” – i.e. no value -- for a field. Null can mean one of two things:

1. There **should be a value**, but we don't know what it is.

Title	Year	Length
Metropolis	1927	Null

2. There **isn't a value** for this particular record.

Title	Year	Lead actor	Lead actress
20,000 Leagues Under The Sea	1954	Kirk Douglas	Null

# Entity integrity

Primary keys cannot have null fields.  
Why?

Null isn't a value. It carries a meaning that is different from having a value.

Its against the rules.

# Referential integrity

If a foreign key exists in a table, either it **must match** the primary key a record in the related table **or** the foreign key value **must be null**. No bogus references allowed.

If the foreign key is **also** constrained to be **not null**, then **no orphan records**.

# Referential integrity

movie_id	title	year	length
1	Sleepless in Seattle	1993	105
2	Holiday	1938	95
3	The Philadelphia Story	1940	112
4	Sabrina	1954	113

name	movie_id
Katherine Hepburn	2
Katherine Hepburn	3
James Stewart	3

With a constraint set, Holiday 1938 can **only be deleted** from the movie table if there is **no reference to it** in the StarsIn table. Katherine Hepburn's "2" prevents this. Depending on the rules this could be handled with **null, delete or cascade delete**.

# General constraints

Can add other types of constraints to make sure the data has integrity.

E.g.

- date
- integer
- float

# Specifying schema

We often write down the schema for a relational database as follows:

Movies (Title, Year, Length)

StarsIn (Name, Title, Year)

where the **underline** indicates the **primary key** of the relation.

Sometimes we add **domain information**:

Movies (Title: **string**, Year: **integer**, Length: **integer**)



Click In...

# Click In!

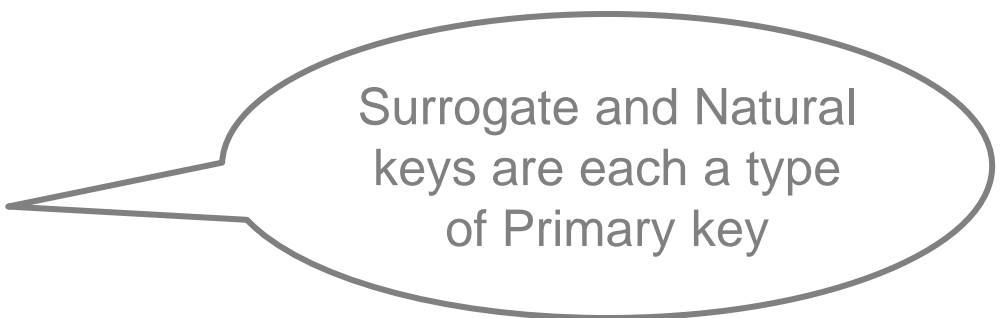
A \_\_\_\_\_ uniquely defines a record in a table.

1. Primary key
2. Surrogate key
3. Natural key
4. All of the above
5. Foreign key

# Click In!

A \_\_\_\_\_ uniquely defines a record in a table.

1. Primary key
2. Surrogate key
3. Natural key
4. All of the above
5. Foreign key



Surrogate and Natural keys are each a type of Primary key

# Click In!

Consider a rollercoaster DB schema:

Coaster(Name, Park, Type, Year).

What is a good primary key?

- A. Name
- B. Park, Year
- C. Name, Park
- D. Name, Type
- E. Name, Year

# Click In!

Consider a rollercoaster DB schema:

Coaster(Name, Park, Type, Year).

What is a good primary key?

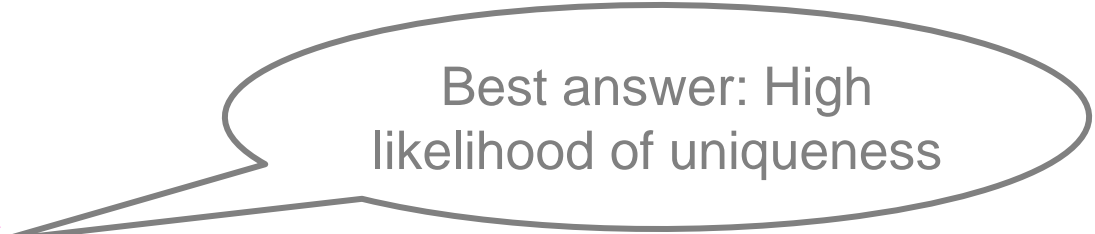
A. Name

B. Park, Year

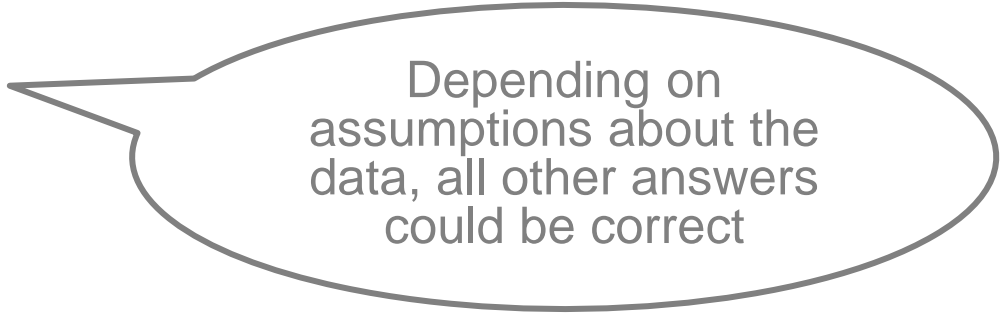
C. Name, Park

D. Name, Type

E. Name, Year



Best answer: High  
likelihood of uniqueness



Depending on  
assumptions about the  
data, all other answers  
could be correct

Movies ( movie\_id, title, year, length )  
StarsIn ( name, movie\_id )

The foreign key in this schema is:

- A. movie\_id in Movies
- B. movie\_id in StarsIn
- C. name and movie\_id in StarsIn
- D. title and year in Movies
- E. None of the above.

Movies ( movie\_id, title, year, length )  
StarsIn ( name, movie\_id )

The foreign key in this schema is:

A. movie\_id in Movies

B. movie\_id in StarsIn

C. name and movie\_id in StarsIn

D. title and year in Movies

E. None of the above.

Now you try...



# Cornell course registration

Suppose you are putting together a database that allows Cornell students to register for courses each semester. Give a sample relational database schema for this database.

What **tables** are needed?

What **fields** belong in each table?

What are the **primary keys** for each table?

What are **foreign keys**?



# Tables

Students

Courses

Registrations

# Tables, fields

Students( netid, first\_name, last\_name)

Courses( dept, number, time, semester)

Registrations( netid, dept, number, semester)

Students( netid, first\_name, last\_name)

Courses( course\_id, dept, number, time, semester)

Registrations( netid, course\_id)



Could also add  
registration\_id

# Tables, fields, primary

Students( netid, first\_name, last\_name)

Courses( dept, number, time, semester)

Registrations( netid, dept, number, semester)

Students( netid, first\_name, last\_name)

Courses( course\_id, dept, number, time, semester)

Registrations( netid, course\_id)

Or: Registrations( registration\_id, netid, course\_id)

# Tables, fields, primary, foreign

Students( netid, first\_name, last\_name)

Courses( dept, number, time, semester)

Registrations( netid, dept, number, semester)

Students( netid, first\_name, last\_name)

Courses( course\_id, dept, number, time, semester)

Registrations( netid, course\_id)

Or: Registrations( registration\_id, netid, course\_id)

# Reminders...

Get started on HW1.