Key elements of databases. Tables. SQL.
DRC Students

- If any student in the class requires a special accommodation for test taking or other assignment, please contact me
  - In person, or via email, ejw@cs.ucsc.edu
  - If you don’t contact me, I will not know you need this accommodation
  - The DRC office no longer sends notifications out about this
Midterm #1

- Next Wednesday, April 27, in class
- Study questions will be given in class on Monday
  - A list of questions of the type that would appear on the exam
- A review session will be held Tuesday afternoon, details TBD
- Test will be mostly short answer type questions, and questions similar to homework #2
- Closed book, closed note
- Will cover all material in class, up to and including next Monday’s lecture
Case studies in big data: Amazon

- Over 130 million active Amazon customer accounts
- Over 2 million active seller accounts
- Millions of daily interactions with company
- More than 42 terabytes of data
  - 1 Terabyte = 1,000,000,000,000 bytes, or $10^{12}$ bytes
- What if they used a paper filing system for customer accounts?
  - Assume each customer file takes $\frac{1}{8}$ of an inch
  - $130$ million * $0.125$ inches = $16.25$ million inches = $1,354,166$ linear feet of files
    - $= 256.5$ linear miles of files
  - If one person could manage 100 linear feet of files, you’d need $13,541$ people just to manage customer files
  - Of course, this is with no ability to search over them, learn from them, leverage them in promotions on the site, and with lots of errors, lost files, etc.
  - **Manual approaches don’t scale**
Case studies in big data: YouTube

- Approximately 145 million videos
  - Assume each video has a size of 1MB
    - Probably too low
  - 145,000,000,000,000 bytes of data
    = 145 Terabytes of video data
- 14.6 billion videos viewed per month
  - As of May, 2010, probably much larger now
- On top of this, there is all of the comment data, user profile data, etc.
- This business **could not exist** without large scale management of data and **inexpensive data storage**
What is a Database?

- Colloquially, in general use, the term database often means:
  - A large collection of data managed by a single organization
- Within computer science, a database typically means a program that provides the following services:
  - Storage of structured data
  - Access and update of this data
  - Searching of this data
  - ACID transaction guarantees
    - Atomic, Consistent, Isolation, Durability
    - Ensures data doesn’t get lost or overwritten
  - Administration of data
    - Access control, backups
  - … all of this at large scale
    - Terabytes and terabytes of data
Databases and organizations

- Any medium to large sized organization today **must** use databases
  - Businesses just do not scale without using them
  - Once a business gets larger than the “ma & pa” stage, it starts to need one or more databases to manage the business

- Databases are critical for such functions as:
  - Payroll
  - Inventory management
  - Sales
  - Order fulfillment
  - Customer management
  - Accounting
Organization of databases

- A database contains a set of tables
- A table contains a set of rows
  - Each row represents a set of related data items
  - Each data item belongs to a column
- Associated with a table is a schema
  - The schema describes the kind of data found in each column
  - These are basic data types, integer, float, string, date, etc.
## Database table

<table>
<thead>
<tr>
<th>Vid_ID</th>
<th>Video Title</th>
<th>Video Uploader</th>
<th># views</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ghengis Khan vs Easter Bunny</td>
<td>nicepeter</td>
<td>641698</td>
</tr>
<tr>
<td>2</td>
<td>Strangers, again</td>
<td>WongFuProductions</td>
<td>948102</td>
</tr>
<tr>
<td>3</td>
<td>Copa del Rey Aplastada</td>
<td>lunodos</td>
<td>2405925</td>
</tr>
<tr>
<td>4</td>
<td>Alien Found?</td>
<td>RayWilliamJohnson</td>
<td>741425</td>
</tr>
</tbody>
</table>

Each horizontal line is a **row**

Example: (2, “Strangers, again”, WongFuProductions, 948102)

Each vertical line is a **column**

Example: Video Title is a column
Unique Identifier

- Within a table, one of the columns typically is a **unique identifier**

- What is an identifier?
  - It is a number or string that is **guaranteed to be different** for each row
  - That is, the identifier for a given row is guaranteed to not be used for any other row
  - That is, the identifier is guaranteed to be unique across all rows
  - This permits each row to have data that is the same as data in other rows, but still be able to tell one row apart from another

- In databases, this unique identifier is known as a **key**
  - Each table typically has a key known as the **primary key**

- In class example
Use of keys

One advantage of keys is they allow one table to reference data in another table.

<table>
<thead>
<tr>
<th>Vid_ID</th>
<th>Video Title</th>
<th>Video Uploader</th>
<th># views</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ghengis Khan vs Easter Bunny</td>
<td>nicepeter</td>
<td>641698</td>
</tr>
<tr>
<td>2</td>
<td>Strangers, again</td>
<td>WongFuProductions</td>
<td>948102</td>
</tr>
<tr>
<td>3</td>
<td>Copa del Rey Aplastada</td>
<td>lunodos</td>
<td>2405925</td>
</tr>
<tr>
<td>4</td>
<td>Alien Found?</td>
<td>RayWilliamJohnson</td>
<td>741425</td>
</tr>
</tbody>
</table>

In the table above, don’t really want to store the string naming the video uploader. Instead, really want a reference to the table holding stats about each uploader:
### Use of keys (cont’d)

#### Video table

<table>
<thead>
<tr>
<th>Vid_ID</th>
<th>Video Title</th>
<th>Video Uploader</th>
<th># views</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ghengis Khan vs Easter Bunny</td>
<td>100</td>
<td>641698</td>
</tr>
<tr>
<td>2</td>
<td>Strangers, again</td>
<td>101</td>
<td>948102</td>
</tr>
<tr>
<td>3</td>
<td>Copa del Rey Aplastada</td>
<td>103</td>
<td>2405925</td>
</tr>
<tr>
<td>4</td>
<td>Alien Found?</td>
<td>102</td>
<td>741425</td>
</tr>
</tbody>
</table>

#### Uploaders table

<table>
<thead>
<tr>
<th>UL_ID</th>
<th>User Name</th>
<th># videos uploaded</th>
<th>Joined Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>nicepeter</td>
<td>5</td>
<td>12/25/2008</td>
</tr>
<tr>
<td>101</td>
<td>WongFuProductions</td>
<td>1</td>
<td>5/21/2007</td>
</tr>
<tr>
<td>102</td>
<td>RayWilliamJohnson</td>
<td>3</td>
<td>3/1/2010</td>
</tr>
<tr>
<td>103</td>
<td>lunodos</td>
<td>7</td>
<td>2/28/2009</td>
</tr>
</tbody>
</table>

Unique keys for video table: Vid_ID

Unique keys for uploaders table: UL_ID

A reference to this row:

- Video table
- Uploaders table
Part of creating a database is determining what kind of data each column holds

- This involves modeling a real world situation in data
- Usually, the same kinds of basic data types are available as we have seen so far in class
- Integer, float, string, boolean
- Also: dates, time, arbitrary precision numbers, bit fields and often blobs of text

When creating a table, you must determine which column is associate with which data type
## Schema example

<table>
<thead>
<tr>
<th>Vid_ID</th>
<th>Video Title</th>
<th>Video Uploader</th>
<th># views</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ghengis Khan vs Easter Bunny</td>
<td>nicepeter</td>
<td>641698</td>
</tr>
<tr>
<td>2</td>
<td>Strangers, again</td>
<td>WongFuProductions</td>
<td>948102</td>
</tr>
<tr>
<td>3</td>
<td>Copa del Rey Aplastada</td>
<td>lunodos</td>
<td>2405925</td>
</tr>
<tr>
<td>4</td>
<td>Alien Found?</td>
<td>RayWilliamJohnson</td>
<td>741425</td>
</tr>
</tbody>
</table>

In the table above:

```
CREATE TABLE Video_table(
    Vid_ID INT,            # integer
    Video_title VARCHAR(100),  # VARiable length CHARacter data (i.e., a string with fixed length)
    Video_uploader VARCHAR(40),
    num_views INT,
    PRIMARY KEY (Vid_ID),  # Use the Vid_ID as the unique identifier (the key) for this table
);```

A schema for a table and a class box for a class perform very similar functions

- They describe the **kind of data** for many **instances** of related clusters of data

```sql
CREATE TABLE Video_table(
    Vid_ID INT,
    Video_title VARCHAR(100),
    Video_uploader VARCHAR(40),
    num_views INT,
    PRIMARY KEY (Vid_ID)
);
```

<table>
<thead>
<tr>
<th>Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vid_ID: integer</td>
</tr>
<tr>
<td>Title: string</td>
</tr>
<tr>
<td>Uploader: string</td>
</tr>
<tr>
<td>Num_views: integer</td>
</tr>
</tbody>
</table>

Two representations of the same data, a database schema on the left, a class box on the right.
Finding the top videos on YouTube

- Let’s say you want a list of the top 10 most popular videos on YouTube, **manually**
  - You would need to have all 145 million videos printed out, along with the number of views
  - How would you do this?
Finding the top videos on YouTube

- Let’s say you want a list of the top 10 most popular videos on YouTube
  - Manually, you would need to have all 145 million videos printed out, along with the number of views
    - This would take a few days, but let’s ignore that for now
  - Assume one person could maybe look through 8,000 videos in a day and find the most viewed ones in their list
  - Would need 18,125 people to find the top 10 most popular videos on their lists
  - Would then need another 2-3 people just to collate these results together, and another 8-10 hours
  - If you work morning and night, it might be possible for 18,128 people to find the top 10 most popular videos in appx. 20 hours of work.
  - **Manual searching and sorting doesn’t scale**
SQL Query

- SQL is a standard language for searching a database

- Most important parts of a SQL query:
  - SELECT line
    - Lists the columns of data to see in the final result
  - FROM clause
    - Names the database to search
  - WHERE clause
    - Describes what properties the data must have
    - What data are we interested in?
  - ORDERBY clause
    - Once we have found data matching the WHERE clause, in what order should it be printed out?
SQL Query Example

- For finding the top YouTube videos, we might make a query like the following
  - Assumes the top videos have more than 1,000,000 views

```sql
SELECT Video_title, Num_views
FROM Video_table
WHERE Num_views > 1000000
ORDER BY Num_views DESCENDING
```

Sample output:

- Copa del Rey Aplastada 2,405,925
- Orphan Tears 2,110,871
- “Weird Al” Yankovic – Perform this Way 1,642,019
- Park BOM – Don’t Cry M/V [HD] 1,206,662

Sorted by Num_views from top to bottom (descending order)