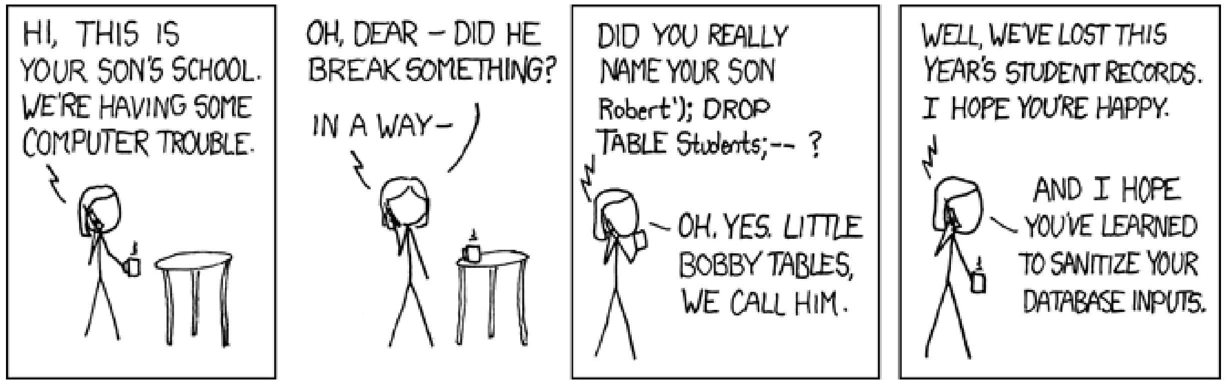
CSc 337


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

LECTURE 21: MULTI-TABLE SQL QUERIES (JOINS)

Querying databases in Node.js

You will need to install the node package called mysql.

```
npm install mysql
```

Connecting to a database

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: hostname,
  database: databasename,
  user: username,
  password: password,
  debug: "true"
});

con.connect(function(err) {
     if (err) throw err;
     console.log("Connected!");
});
```

Connecting to a Database Example

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "mysql.allisonobourn.com",
  database: "csc337world",
  user: "csc337traveler",
  password: "packmybags",
  debug: "true"
});

con.connect(function(err) {
     if (err) throw err;
     console.log("Connected!");
});
```

Querying a Database

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "mysql.allisonobourn.com",
  database: "csc337world",
  user: "csc337traveler",
  password: "packmybags",
  debug: "true"
});

con.connect(function(err) {
      if (err) throw err;
      console.log("Connected!");
      con.query("SELECT * FROM cities WHERE name='london'",
                                      function (err, result, fields) {
             if (err) throw err;
             console.log("Result: " + result[0]["name"]);
      });
});
```

Querying a Database Result

The result object returned by the query is a list of the rows that match the query.

Data for each column can be gotten by accessing the row at the column name.

`result[0]["name"]` from the last slide returns the name of the city in the first returned row.

Related tables and keys

id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

students

id	name
1234	Krabappel
5678	Hoover
9012	Obourn

teachers

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 154	9012
10004	Informatics 100	1234

courses

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

grades

- **primary key**: a column guaranteed to be unique for each record (e.g. Lisa Simpson's ID 888)
- **foreign key**: a column in table A storing a primary key value from table B
  - (e.g. records in `grades` with `student_id` of 888 are Lisa's grades)
- **normalizing**: splitting tables to improve structure / redundancy (linked by unique IDs)

Giving names to tables

```sql
SELECT s.name, g.*
FROM students s
JOIN grades g ON s.id = g.student_id
WHERE g.grade <= 'C';
```
SQL

name	student_id	course_id	grade
Bart	123	10001	B-
Bart	123	10002	C
Milhouse	456	10001	B+
Lisa	888	10002	A+
Lisa	888	10003	A+

- can give names to tables, like a variable name in Java
- to specify all columns from a table, write *table*`.*`
- (`grade` column sorts alphabetically, so grades C or better are ones `<=` it)

Querying multi-table databases

When we have larger datasets spread across multiple tables, we need queries that can answer high-level questions such as:

- What courses has Bart taken and gotten a B- or better?
- What courses have been taken by both Bart and Lisa?
- Who are all the teachers Bart has had?
- How many total students has Ms. Krabappel taught, and what are their names?

To do this, we'll have to **join** data from several tables in our SQL queries.

Joining with ON clauses

```
SELECT column(s)
FROM table1
JOIN table2 ON condition(s)
...
JOIN tableN ON condition(s);                                    SQL
```

```
SELECT *
FROM students
JOIN grades ON id = student_id;                                 SQL
```

- **join**: combines records from two or more tables if they satisfy certain conditions
- the ON clause specifies which records from each table are matched
- the rows are often linked by their **key** columns (id)

Join example

```
SELECT *
FROM students
JOIN grades ON id = student_id;                                    SQL
```

id	name	email	student_id	course_id	grade
123	Bart	bart@fox.com	**123**	10001	B-
123	Bart	bart@fox.com	**123**	10002	C
404	Ralph	ralph@fox.com	**404**	10004	D+
456	Milhouse	milhouse@fox.com	**456**	10001	B+
888	Lisa	lisa@fox.com	**888**	10002	A+
888	Lisa	lisa@fox.com	**888**	10003	A+

table.*column* can be used to disambiguate column names:

```
SELECT *
FROM students
JOIN grades ON students.id = grades.student_id;                    SQL
```

What's wrong with this?

```
SELECT name, id, course_id, grade
FROM students
JOIN grades ON id = 123
WHERE id = student_id;
```
SQL

name	id	course_id	grade
Bart	123	10001	B-
Bart	123	10002	C

- The above query produces the same rows as the previous one, but it is poor style. Why?

- The `JOIN ON` clause is poorly chosen. It doesn't really say what connects a `grades` record to a `students` record.
  - They are related when they are for a student with the same `id`.
  - Filtering out by a specific ID or name should be done in the `WHERE` clause, not `JOIN ON`.

A suboptimal query

Exercise: What courses have been taken by both Bart and Lisa?

```
SELECT bart.course_id
FROM grades bart
JOIN grades lisa ON lisa.course_id = bart.course_id
WHERE bart.student_id = 123
AND lisa.student_id = 888;                                           SQL
```

- problem: requires us to know Bart/Lisa's Student IDs, and only spits back course IDs, not names.

- Write a version of this query that gets us the course *names*, and only requires us to know Bart/Lisa's names, not their IDs.

Improved query

What courses have been taken by both Bart and Lisa?

```
SELECT DISTINCT c.name
FROM courses c
JOIN grades g1 ON g1.course_id = c.id
JOIN students bart ON g1.student_id = bart.id
JOIN grades g2 ON g2.course_id = c.id
JOIN students lisa ON g2.student_id = lisa.id
WHERE bart.name = 'Bart'
AND lisa.name = 'Lisa';                                             SQL
```

Practice queries

- What are the names of all teachers Bart has had?

```
SELECT DISTINCT t.name
FROM teachers t
JOIN courses c ON c.teacher_id = t.id
JOIN grades g ON g.course_id = c.id
JOIN students s ON s.id = g.student_id
WHERE s.name = 'Bart';                                              SQL
```

- How many total students has Ms. Krabappel taught, and what are their names?

```
SELECT DISTINCT s.name
FROM students s
JOIN grades g ON s.id = g.student_id
JOIN courses c ON g.course_id = c.id
JOIN teachers t ON t.id = c.teacher_id
WHERE t.name = 'Krabappel';                                         SQL
```

Designing a query

- Figure out the proper SQL queries in the following way:
  - Which table(s) contain the critical data? (`FROM`)
  - Which columns do I need in the result set? (`SELECT`)
  - How are tables connected (`JOIN`) and values filtered (`WHERE`)?
- Test on a small data set (`imdb_small`).
- Confirm on the real data set (`imdb`).
- Try out the queries first in the query tool.
- Write the PHP code to run those same queries.
  - Make sure to check for SQL errors at every step!!

Example imdb database

id	first_name	last_name	gender
433259	William	Shatner	M
797926	Britney	Spears	F
831289	Sigourney	Weaver	F
...			

actors

id	name	year	rank
112290	Fight Club	1999	8.5
209658	Meet the Parents	2000	7
210511	Memento	2000	8.7
...			

movies

actor_id	movie_id	role
433259	313398	Capt. James T. Kirk
433259	407323	Sgt. T.J. Hooker
797926	342189	Herself
...		

roles

movie_id	genre
209658	Comedy
313398	Action
313398	Sci-Fi
...	

movies_genres

id	first_name	last_name
24758	David	Fincher
66965	Jay	Roach
72723	William	Shatner
...		

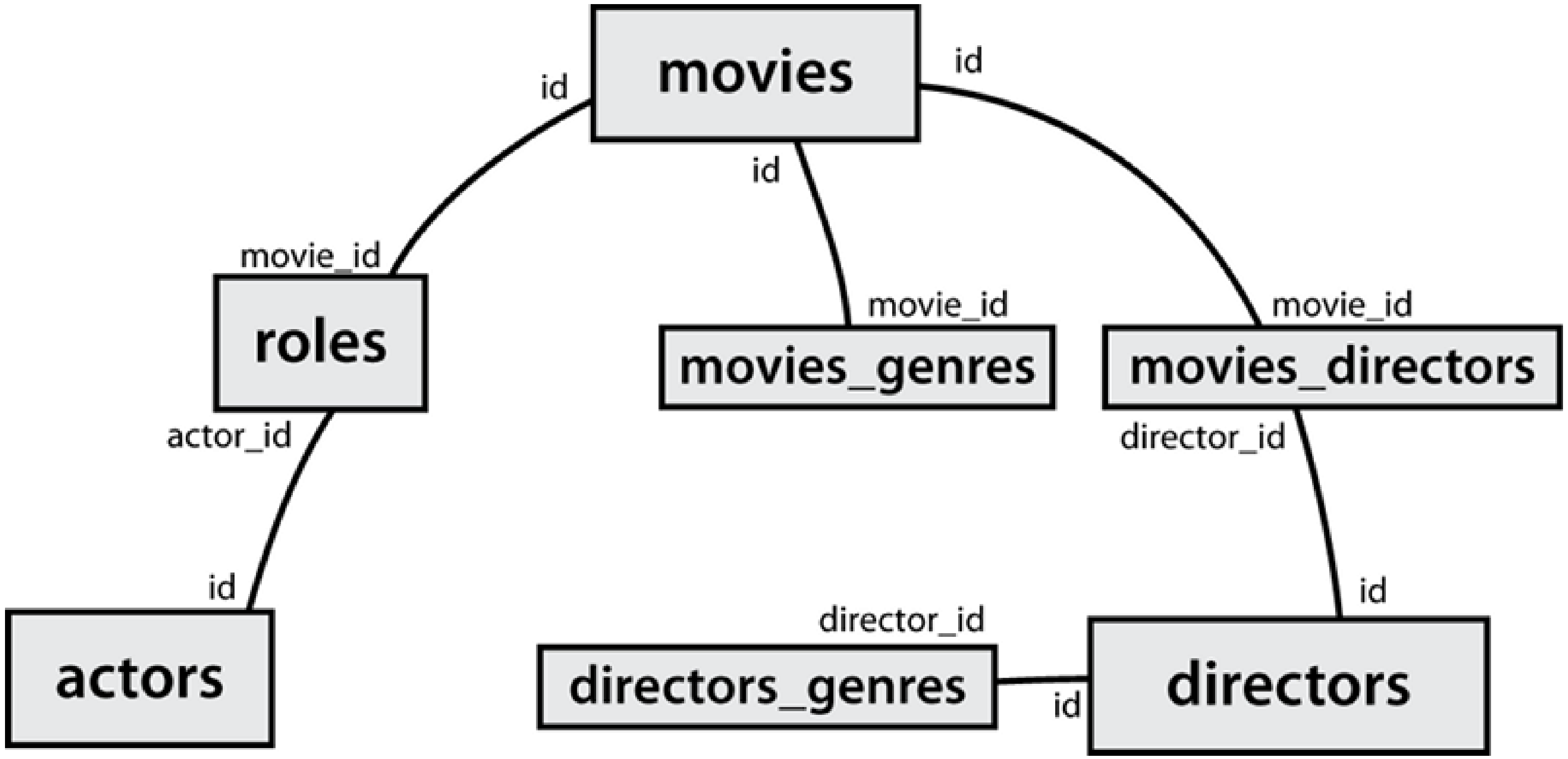
directors

director_id	movie_id
24758	112290
66965	209658
72723	313398
...	

movies_directors

- also available, `imdb_small` with fewer records (for testing queries)

IMDb table relationships / ids



IMDb practice queries

- What are the names of all movies released in 1995?
- How many people played a part in the movie "Lost in Translation"?
- What are the *names* of all the people who played a part in the movie "Lost in Translation"?
- Who directed the movie "Fight Club"?
- How many movies has Clint Eastwood directed?
- What are the *names* of all movies Clint Eastwood has directed?
- What are the names of all directors who have directed at least one horror film?
- What are the names of every actor who has appeared in a movie directed by Christopher Nolan?

HTML tables: <table>, <tr>, <td>

A 2D table of rows and columns of data (block element)

```html
<table>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```
HTML

```
1,1            1,2 okay
2,1 real wide  2,2
```
output

- `table` defines the overall table, `tr` each row, and `td` each cell's data
- tables are useful for displaying large row/column data sets
- NOTE: tables are sometimes used by novices for web page layout, but this is not proper semantic HTML and should be avoided

Table headers, captions: <th>, <caption>

```
<table>
  <caption>My important data</caption>
  <tr><th>Column 1</th><th>Column 2</th></tr>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```
HTML

My important data

Column 1	**Column 2**
1,1	1,2 okay
2,1 real wide	2,2

output

- `th` cells in a row are considered headers; by default, they appear bold
- a `caption` at the start of the table labels its meaning

Styling tables

```
table { border: 2px solid black; caption-side: bottom; }
tr { font-style: italic; }
td { background-color: yellow; text-align: center; width: 30%; }
```

Column 1	Column 2
1,1	1,2 okay
2,1 real wide	2,2

My important data

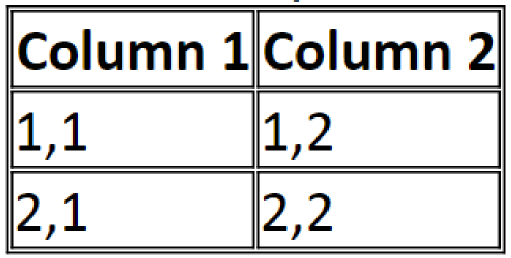
output

- all standard CSS styles can be applied to a table, row, or cell
- table specific CSS properties:
  - border-collapse, border-spacing, caption-side, empty-cells, table-layout

The border-collapse property

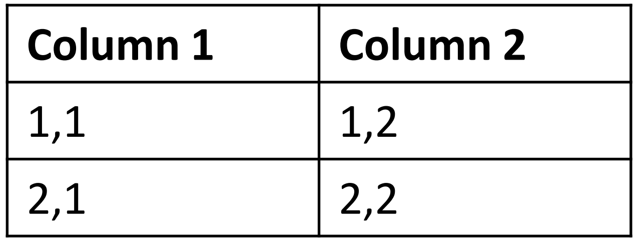
```css
table, td, th { border: 2px solid black; }
table { border-collapse: collapse; }                    CSS
```

Without `border-collapse`



Column 1	Column 2
1,1	1,2
2,1	2,2

With border-collapse



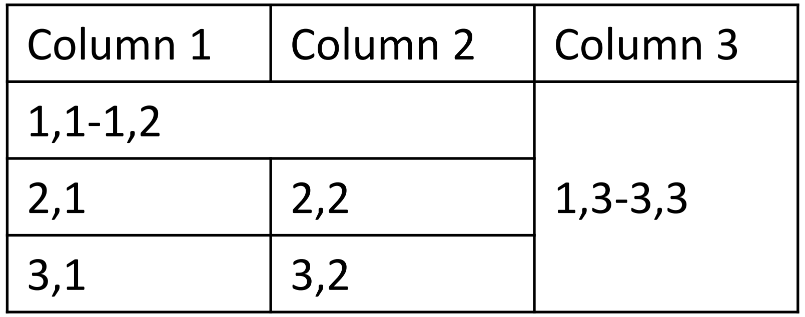
Column 1	Column 2
1,1	1,2
2,1	2,2

- by default, the overall table has a separate border from each cell inside
- the `border-collapse` property merges these borders into one

The rowspan and colspan attributes

```
<table>
  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td colspan="2">1,1-1,2</td>
    <td rowspan="3">1,3-3,3</td></tr>
  <tr><td>2,1</td><td>2,2</td></tr>
  <tr><td>3,1</td><td>3,2</td></tr>
</table>
```

HTML



Column 1	Column 2	Column 3
1,1-1,2		1,3-3,3
2,1	2,2	
3,1	3,2	

HTML

- `colspan` makes a cell occupy multiple columns; `rowspan` multiple rows
- `text-align` and `vertical-align` control where the text appears within a cell

Column styles: <col>, <colgroup>

```
<table>
  <col class="urgent" />
  <colgroup class="highlight" span="2"></colgroup>

  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td>1,1</td><td>1,2</td><td>1,3</td></tr>
  <tr><td>2,1</td><td>2,2</td><td>2,3</td></tr>
</table>
```
HTML

Column 1	Column 2	Column 3
1,1	1,2	1,3
2,1	2,2	2,3

output

- `col` tag can be used to define styles that apply to an entire column (self-closing)
- `colgroup` tag applies a style to a group of columns (NOT self-closing

Don't use tables for layout!

- (borderless) tables appear to be an easy way to achieve grid-like page layouts
  - many "newbie" web pages do this (including many UW CSE web pages...)
- but, a `table` has semantics; it should be used only to represent an actual table of data
- instead of tables, use `div`s, widths/margins, floats, etc. to perform layout

- tables should not be used for layout!
- tables should not be used for layout!!
- TABLES SHOULD NOT BE USED FOR LAYOUT!!!
- TABLES SHOULD NOT BE USED FOR LAYOUT!!!!

Designing a query

- Figure out the proper SQL queries in the following way:
    - Which table(s) contain the critical data? (`FROM`)
    - Which columns do I need in the result set? (`SELECT`)
    - How are tables connected (`JOIN`) and values filtered (`WHERE`)?
- Test on a small data set (`imdb_small`).
- Confirm on the real data set (`imdb`).
- Try out the queries first in the MySQL console.
- Write the Node.js code to run those same queries.
    - Make sure to check for SQL errors at every step!!