More SQL

INFO/CS 2300:
Intermediate Web Design and Programming

1 sheet on half wall

Grades – P2 and HW1

P2 regrade requests by Thursday 5 pm.

HW1 grades should be released by Friday.

Last Friday's quiz

window.onload

$(document).ready()

After everything has loaded

After HTML has loaded

Coming up

Homework 2

- Given queries in English, translate them into SQL.
- Due Tuesday March 8 at 5pm
- Don't share SQL

Friday quiz and activity:

databases / SQL

Click In!

Basic SQL

Foriegn keys: reference to other tables
Connect information from two tables using: JOIN

Selection

Recall the basic form of a SQL statement:

```
SELECT Fields
FROM Table
WHERE Condition;
```

title	year	length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
SELECT title, length
FROM movies
WHERE length > 150 AND title LIKE '%King%';
```

Inline calculation

We can modify output of table on the fly.

```
SELECT
  title,
  year,
  length/60 AS Hours
FROM movies;
```

INNER JOIN 2 tables

students(NetID, FirstName, LastName)

courses(CourseID, Dept, Number, Time, Semester)

registrations(NetID, CourseID)

```
SELECT
  students.NetID,
  students.FirstName,
FROM registrations
  INNER JOIN students
    ON registrations.NetID = students.NetID
WHERE registrations.CourseID = 12345;
```

INNER JOIN 3 tables

students(NetID, FirstName, LastName)

courses(CourseID, Dept, Number, Time, Semester)

registrations(NetID, CourseID)

```
SELECT
  students.NetID,
  students.FirstName,
  courses.Number
FROM registrations
  INNER JOIN students
    ON registrations.NetID = students.NetID
  INNER JOIN courses
    ON registrations.CourseID = courses.CourseID
WHERE courses.Dept = 'Computer Science';
```

Sailors, Boats and Reserves

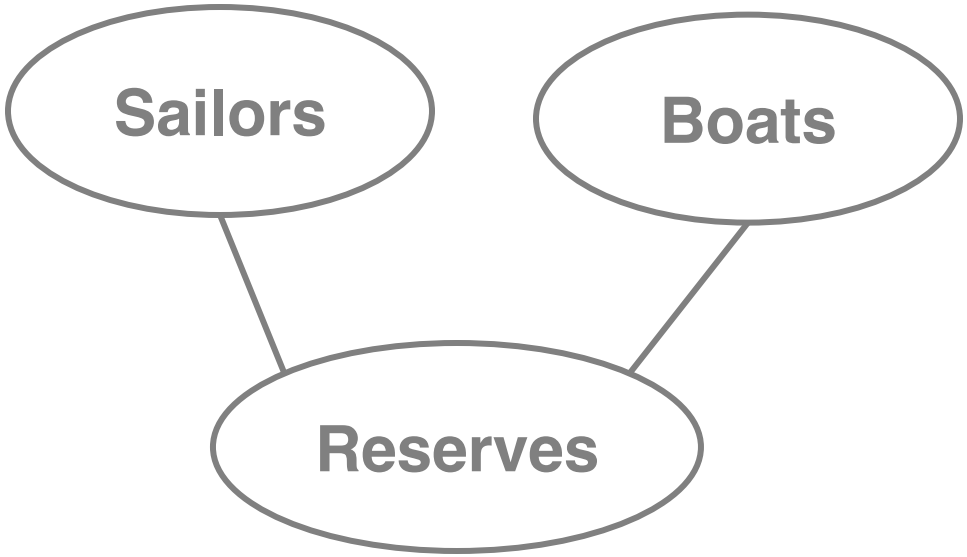
Sailors(
sailorId: integer,
sailorName: string,
rating: integer,
age: integer)

Boats(
boatId: integer,
boatName: string,
color: string)

Reserves(
sailorId: integer,
boatId: integer,
day: date)

What do we notice about this schema?

Reserves references back to Boats and Sailors



Exercises 1 - 3

```
2. SELECT sailorName
FROM Sailors INNER JOIN Reserves
ON Saiilors.sailorID = Reserves.sailorID
Where boatID = 103;

3. SELECT Boats.color
        FROM Reserves
        INNER JOIN Sailors
                ON Reserves.sailorID = Sailors.sailorID
        INNER JOIN Boats
                ON Reserves.boatID = Boats.boatID
WHERE sailorName = “Lubber”
```

INNER JOIN

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003
Hillary Swank	Million Dollar Baby	2004

Takes two fields to uniquely identify a record (.title and .year)

```
SELECT movies.title
FROM movies
   INNER JOIN starsIn
      ON movies.title = starsIn.title
      AND movies.year = starsIn.year
```

Title
Gladiator
A Beautiful Mind
The Return of the King
Million Dollar Baby

Outer joins

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003
Hillary Swank	Million Dollar Baby	2004

What if we want all the movies whether or not there is a record in StarsIn but we want the Name from StarsIn if there is one?

(Left) Outer joins

Start with table; left JOIN if the table isthere

```
SELECT
  movies.title,
  movies.year,
  movies.length,
  starsin.name
FROM movies
  LEFT OUTER JOIN starsin
    ON movies.title = starsin.title
     AND movies.year = starsin.year;
```

Ask for:

With Outer Joins the order of the tables matters

movies is going to JOIN starsin -> will include Chicago and stars

movies.title	movies.year	movies.length	starsin.name
A Beautiful Mind	2001	135	Russell Crowe
Chicago	2002	113	(null)
Gladiator	2000	155	Russell Crowe
Million Dollar Baby	2004	132	Hillary Swank
The Return of the King	2003	201	Viggo Mortensen

IS NULL

What movies don’t have an actor in the starsIn table?

```
SELECT
  movies.title,
FROM movies
  LEFT OUTER JOIN starsIn
    ON movies.year = starsin.year
     AND movies.title = starsin.title
WHERE starsIn.title IS NULL;
```

Looking for things that don’t have a match

movies.title
Chicago

FULL JOIN

title	year	length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113

name	title	year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Tom Hanks	Big	1988

```
SELECT *
FROM movies m
  FULL JOIN starsIn s
    ON m.year = s.year
     AND m.title = s.title;
```

All rows from both tables. Non matches are null.

m.title	m.year	m.length	s.name	s.title	s.year
A Beautiful Mind	2001	135	Russell Crowe	A Beautiful Mind	2001
Chicago	2002	113	(null)	(null)	(null)
Gladiator	2000	155	Russell Crowe	Gladiator	2000
(null)	(null)	(null)	Tom Hanks	Big	1988

Aggregation and Grouping

Aggregation

We can aggregate results of a given field across all records of a table.

SUM – sums a field with numerical value

AVG – averages a field with numerical values

MIN, MAX – produces minimum, maximum of a field (either for numbers or strings)

COUNT – counts the number of records

title	year	length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
SELECT MAX(length)
  FROM Movies;
```

MAX(length)
201

```
SELECT AVG(length) as average
  FROM Movies;
```

average
147.2

```
SELECT COUNT(*)
  FROM Movies
  WHERE Year >= 2002;
```

Count(*)
3

Grouping

```
SELECT AVG(length)
FROM Movies
WHERE year = 2000;
```

title	year	length
Gladiator	2000	155
Crouching Tiger, Hidden Dragon	2000	120
Moulin Rouge	2001	127
A Beautiful Mind	2001	135
Chicago	2002	113
Lost in Translation	2003	102
The Return of the King	2003	201

If we want to know the average length for each year, do we have to run a query for every year?

No

GROUP BY

Title	Year	Length
Gladiator	2000	155
Crouching Tiger, Hidden Dragon	2000	120
Moulin Rouge	2001	127
A Beautiful Mind	2001	135
Chicago	2002	113
Lost in Translation	2003	102
The Return of the King	2003	201

Averages length by year
No multiple queries

```
SELECT
  Year,
  AVG(Length) AS AvgLength
FROM Movies
GROUP BY Year;
```

Year	AvgLength
2000	137.5
2001	131
2002	113
2003	151.5

Grouping generalized

```
SELECT
  Field1, …,
  Fieldk,
  Aggregate1(B1) AS C1, …,
  AggregateN(Bn) AS Cn
FROM Table
GROUP BY Field1, …, Fieldk;
```

Aggregate = COUNT, MAX, LENGTH, etc.

Every field has to be aggregated or grouped by

You can group by more than one field

Each field either must be named in the GROUP BY clause or include an aggregate function.

Selecting groups

Title	Year	Length
Gladiator	2000	155
Crouching Tiger, Hidden Dragon	2000	120
Moulin Rouge	2001	127
A Beautiful Mind	2001	135
~~Chicago~~	~~2002~~	~~113~~
Lost in Translation	2003	102
The Return of the King	2003	201

Suppose we want average length by year but only for years with more than one movie.

Can't use "WHERE" because the criteria aren't known until after the grouping

Criteria for groups

Title	Year	Length
Gladiator	2000	155
Crouching Tiger, Hidden Dragon	2000	120
Moulin Rouge	2001	127
A Beautiful Mind	2001	135
~~Chicago~~	~~2002~~	~~113~~
Lost in Translation	2003	102
The Return of the King	2003	201

```
SELECT Year, AVG(Length) AS AvgLength
FROM Movies
GROUP BY Year
HAVING COUNT(Title) > 1;
```

Year	AvgLength
2000	137.5
2001	131
2003	151.5

Includes years that have at least more than one movie

HAVING vs WHERE

‘HAVING’ allows for conditions on *groups*.

‘WHERE’ allows for conditions on *rows*.

Sorting and Limits

Output in sorted order

```
SELECT *
FROM Movies
ORDER BY Length;

SELECT *
FROM Movies
ORDER BY Length, Year;

SELECT *
FROM Movies
ORDER BY Length DESC, Year;
```

Descending

Assume but not guaranteed: records are returned in random order unless ORDER BY is used.

Text
Text

Limit

```
SELECT *
FROM Movies
ORDER BY Length
LIMIT 3;
```

ORDER and LIMIT usually come together

It is particularly important to use ORDER BY when using LIMIT.

Click In!

Exercises 4 - 5

4. Find age of the youngest sailor for each rating level

```
SELECT rating, MIN(age) as MinAge
FROM Sailors
GROUP BY rating
```

5. Find the agerage age of sailors for each rating level that has at least two sailors

```
SELECT rating, AVG(age
FROM Sailors
GROUP BY raiting
HAVING COUNT(*) > 1
```

More SQL: Nested queries

When you know the rules but not the values

Nested Query

Nested query in WHERE clause: generates one value

title	year	length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

List the title and year of movies that are longer than average.

Nested query in the WHERE clause

```
SELECT title, year
FROM movies
WHERE length > (SELECT AVG(length)
                FROM movies);
```

Nested values IN

Using a subquery to generate a set of values.

```
SELECT name
FROM starsIn
WHERE ( title, year ) IN
     (SELECT title, year
       FROM movies
       WHERE
         length > (SELECT AVG(length)
                     FROM movies)
     );
```

Temporary Table

Nested query in FROM clause

Using a subquery to generate the table for the FROM statement.

Nested query in the FROM clause

```
SELECT Title, Year
FROM (SELECT *
        FROM StarsIn
        WHERE Name <> 'Russell Crowe')
    AS Temp
  INNER JOIN Movies
    ON Temp.Title = Movies.Title
      AND Temp.Year = Movies.Year;
```

actors NOT russel crowe

Multi-level nesting

```
SELECT name
FROM starsIn
INNER JOIN
  (SELECT title, year
    FROM movies
    WHERE length > (SELECT AVG(length)
                    FROM movies)
  ) AS longMovies
    ON starsIn.title = longMovies.title
      AND starsIn.year = longMovies.year;
```

SOME and ALL

For numeric values, we can compare against SOME or ALL of the values from a subquery.

```
SELECT Title, Year
FROM Movies
WHERE Length >= ALL (SELECT Length
                     FROM Movies);
```

What does this do?

```
SELECT Title, Year
FROM Movies
WHERE Length > SOME (SELECT Length
                     FROM Movies);
```

EXISTS

What movies are remakes?
(released in more than one year)

E.g.

```
SELECT DISTINCT Title
FROM Movies m
WHERE EXISTS (SELECT *
              FROM Movies o
              WHERE o.Title = m.Title AND
                    o.Year <> m.Year);
```

information from the "outer" query used in the "inner" query

Exercises 6 - 9

Review

- Group By allows aggregate functions to run on groups of rows
- We can use the results of SQL queries as part of another SQL query.