CSE 414: Section 2
A SeQueL to SQL

April 5th, 2018
Administrivia

WQ2 due TUESDAY!!! (Tuesday, Apr 10th at 11:59 PM)

HW2 due Tuesday, Apr10th at 11:59 PM

Last day to turn in HW1 (with late days)
Git Demo

How to add git remote upstream?
SQL 3-Valued Logic

SQL has 3-valued logic

- **FALSE = 0**
  
  [ex] price < 25 is FALSE when price = 99

- **UNKNOWN = 0.5**
  
  [ex] price < 25 is UNKNOWN when price = NULL

- **TRUE = 1**
  
  [ex] price < 25 is TRUE when price = 19
SQL 3-Valued Logic (con’t)

Formal definitions:
- $C_1 \text{ AND } C_2$ means $\min(C_1, C_2)$
- $C_1 \text{ OR } C_2$ means $\max(C_1, C_2)$
- $\text{NOT } C$ means $1-C$

The rule for \textit{SELECT} ... \textit{FROM} ... \textit{WHERE} $C$ is the following:
- if $C = \text{TRUE}$ then include the row in the output
- if $C = \text{FALSE}$ or $C = \text{unknown}$ then do not include it
First, create the table.
Then, import the data.

```
.mode csv
.import ./population.csv Population
.import ./gdp.csv GDP
.import ./airport.csv Airport
.import /path/to/file NameOfTable
```
Aliasing

- Good style for renaming attribute operations to more intuitive labels
- Essential for self joins (ex: FROM [table] AS T1, [table] AS T2)
- You can alias without “AS” in the FROM clause (i.e. “AS” keyword can be omitted)

```sql
SELECT [attribute] AS [attribute_name]
FROM [table] AS [table_name]
... [table_name].[attribute_name] ...```
Aggregates

- Computes aggregated values for a set of tuples.

**COUNT(attribute)** - counts the number of tuples
**SUM(attribute)**
**MIN/MAX(attribute)**
**AVG(attribute)**
...


Filters

**LIMIT** *number* - limits the amount of tuples returned

[ex] SELECT * FROM table LIMIT 1;

**DISTINCT** - only returns different values (gets rid of duplicates)

[ex] SELECT DISTINCT column_name FROM table;
Grouping and Ordering

GROUP BY [attribute], ..., [attribute_n]

HAVING [predicate] - operates on groups

ORDER BY
SQL Query Evaluation Order

FWGHOS

(From, Where, Group By, Having, Order By, Select)
Joining

Inner vs. Outer Self Joins
Join Semantics

- Think as “nested loops”.
- NOT the most efficient implementation on a large database! (we will talk about other ways to join later in the course)
  - Hash Join
  - Sort-Merge Join

For more information and different types of joins see:
Nested Loop Semantics

SELECT $x_1.a_1$, ..., $x_n.a_n$
FROM $x_1$, ..., $x_n$
WHERE $<\text{cond}>$

for each tuple in $x_1$:

...  
  for each tuple in $x_n$:
    if $<\text{cond}>$(x_1, ..., x_n):
      output($x_1.a_1$, ..., $x_n.a_n$)