CSE 344: Section 2
A SeQueL to SQL

Jun 28th, 2018

Administrivia

WQ2 due **Friday, June 29th** at 11:00 PM

HW2 due **Wednesday, July 4th** at 11:00 PM

SQL 3-Valued Logic

SQL has 3-valued logic

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

C1 AND C2 means min(C1,C2)

C1 OR C2 means max(C1,C2)

NOT C means means 1-C

(F, false; U, unknown; T, true)

NOT(A)

A	¬A
F	T
U	U
T	F

AND(A, B)

A ∧ B		B		
		F	U	T
A	F	F	F	F
	U	F	U	U
	T	F	U	T

OR(A, B)

A ∨ B		B		
		F	U	T
A	F	F	U	T
	U	U	U	T
	T	T	T	T

SQL 3-Valued Logic (con't)

Formal definitions:

C1 AND C2 means min(C1,C2)

C1 OR C2 means max(C1,C2)

NOT C means means 1-C

The rule for SELECT ... FROM ... WHERE C is the following:

if C = TRUE then include the row in the output

if C = FALSE or C = unknown then do not include it

Importing Files

First, make the table.

Then, import the data.

```
.mode csv
    .import ./class.csv Class
    .import ./instructor.csv Instructor
    .import ./teaches.csv Teaches
```

Aliasing

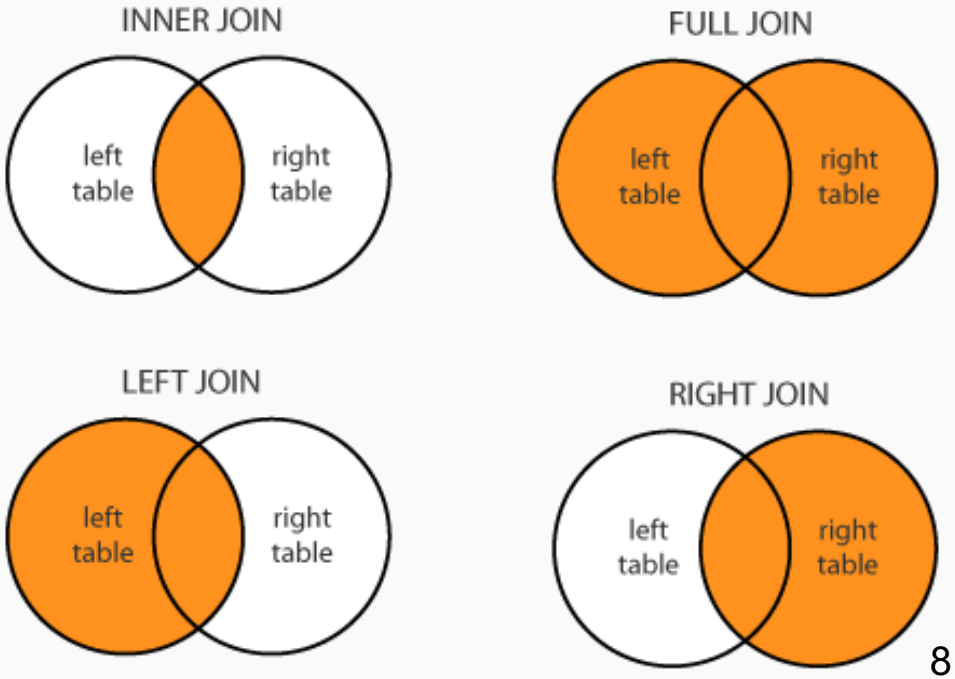
- Good style for renaming attribute operations to more intuitive labels
- Essential for self joins (ex: `FROM [table] AS T1, [table] AS T2`)
- You can alias without "AS" in the FROM clause (i.e. "AS" keyword can be omitted)

```
SELECT [attribute] AS [attribute_name]
FROM [table] AS [table_name]
… [table_name].[attribute_name] …
```

Joining

Inner vs. Outer

Self Joins



Given tables A and B,

A:

a1	a2
1	7
2	5
3	3
4	6

B:

b1	b2
3	2
4	1
5	0
6	3

Write down the output of each of the following queries:

```
SELECT * FROM A INNER JOIN B ON A.a1 = B.b1;
```

a1	a2	b1	b2

Given tables A and B,

A:

a1	a2
1	7
2	5
3	3
4	6

B:

b1	b2
3	2
4	1
5	0
6	3

Write down the output of each of the following queries:

```
SELECT * FROM A LEFT OUTER JOIN B ON A.a1 = B.b1;
```

a1	a2	b1	b2

Given tables A and B,

A:

a1	a2
1	7
2	5
3	3
4	6

B:

b1	b2
3	2
4	1
5	0
6	3

Write down the output of each of the following queries:

```
SELECT * FROM A FULL OUTER JOIN B ON A.a1 = B.b1;
```

a1	a2	b1	b2

For more information and different types of joins see:
https://blogs.msdn.microsoft.com/craigfr/2006/08/16/summary-of-join-properties/

Join Semantics

- For now, we are primarily focusing on “nested loops” semantics
- 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

Nested Loop Semantics

```
SELECT x_1.a_1, …, x_n.a_n
FROM x_1, …, x_n
WHERE <cond>

for each tuple in x_1:
    …
        for each tuple in x_n:
            if <cond>(x_1, …, x_n):
                output(x_1.a_1, …, x_n.a_n)
```

Aggregates

- Aggregates will make the query return a single tuple.

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

```
CREATE TABLE Movies (
    id int,
    name varchar(30),
    budget int,
    gross int,
    rating int,
    year int,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE Actors (
    id int,
    name varchar(30),
    age int,
    PRIMARY KEY (id)
);

CREATE TABLE ActsIn (
    mid int,
    aid int,
    FOREIGN KEY (mid) REFERENCES Movies (id),
    FOREIGN KEY (aid) REFERENCES Actors (id)
);
```

Write queries to answer the following:

(a) For each movie, find the number of actors who acted in it, ordered by descending number of actors. Make sure to include movies with no actors!

```
CREATE TABLE Movies (
    id int,
    name varchar(30),
    budget int,
    gross int,
    rating int,
    year int,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE Actors (
    id int,
    name varchar(30),
    age int,
    PRIMARY KEY (id)
);

CREATE TABLE ActsIn (
    mid int,
    aid int,
    FOREIGN KEY (mid) REFERENCES Movies (id),
    FOREIGN KEY (aid) REFERENCES Actors (id)
);
```

Write queries to answer the following:

(b) What is the number of movies and the average rating of all the movies that the actor "Kit Harington" has appeared in?

```
CREATE TABLE Movies (
    id int,
    name varchar(30),
    budget int,
    gross int,
    rating int,
    year int,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE Actors (
    id int,
    name varchar(30),
    age int,
    PRIMARY KEY (id)
);

CREATE TABLE ActsIn (
    mid int,
    aid int,
    FOREIGN KEY (mid) REFERENCES Movies (id),
    FOREIGN KEY (aid) REFERENCES Actors (id)
);
```

Write queries to answer the following:

(c) What is the age of the youngest actor who has appeared in a movie that grossed over $1,000,000,000?

SQL Query Evaluation Order

FWGHOS

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