Welcome to CS 186, Section 2!

TA: Bryan Munar

OH: Mondays 11-12pm and Thursdays 2:30-3:30pm (651 Soda)

DISC: Tuesdays 11-12am (136 Barrows) and Wednesdays 10-11am (130 Wheeler)

Announcements and Such

- Project due FRIDAY (9/11) !! Any questions, come to office hours

- How do you like the pdfs? Want me to make more if a topic allows?

Review of External Sorting and Hashing?

Discussion 2: All About SQL

Overview:

1. Single Table SQL
2. Worksheet exercises
3. Querying Multiple Relations
4. Worksheet exercises

(A lot of the slides based on lecture!)

Relational Tables

- *Schema* is fixed:
  - attribute names, *atomic* types
  - `students(name text, gpa float, dept text)`

- *Instance* can change
  - a *multi*set of "rows" ("tuples")
    - `{('Bob Snob', 3.3,'CS'),`
      `('Bob Snob', 3.3,'CS'),`
      `('Mary Contrary', 3.8, 'CS')}`

Basic Single-Table Queries


```
SELECT [DISTINCT] <column expression list>
   FROM <single table>
[WHERE <predicate>]
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>];
```

Let’s start with the basics of SQL!

The brackets mean that you can add that part into a SQL query.

The red portions are what we’re focusing on per slide.

Basic Single-Table Queries


```
SELECT [DISTINCT] <column expression list>
   FROM <single table>
[WHERE <predicate>]
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>] ;
```

Simplest version is straightforward

- Produce all tuples in the table that satisfy the predicate
- Output the expressions in the SELECT list
- Expression can be a column reference, or an arithmetic expression over column refs

Basic Single-Table Queries


```
SELECT S.name, S.gpa
  FROM students AS S
 WHERE S.dept = 'CS'
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>] ;
```

Simplest version is straightforward

- Produce all tuples in the table that satisfy the predicate
- Output the expressions in the SELECT list
- Expression can be a column reference, or an arithmetic expression over column refs

SELECT DISTINCT


```
SELECT DISTINCT S.name, S.gpa
  FROM students S
 WHERE S.dept = 'CS'
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>] ;
```

DISTINCT flag specifies removal of duplicates before output

Removed the “AS” from FROM clause --- it’s optional

ORDER BY


```
SELECT DISTINCT S.name, S.gpa, S.age*2 AS a2
  FROM Students S
 WHERE S.dept = 'CS'
[GROUP BY <column list>
 [HAVING <predicate>] ]
 ORDER BY S.gpa, S.name, a2;
```

ORDER BY clause specifies output to be sorted

- ***Lexicographic*** ordering (left to right)

Obviously must refer to columns in the output

- Note the AS clause for naming output columns!

ORDER BY


```
SELECT DISTINCT  S.name, S.gpa
  FROM Students S
 WHERE S.dept = 'CS'
[GROUP BY <column list>
 [HAVING <predicate>] ]
 ORDER BY S.gpa DESC, S.name ASC;
```

Ascending order by default, but can be overridden

- DESC flag for descending, ASC for ascending
- Can mix and match, lexicographically

AGGREGATES


```
SELECT [DISTINCT] AVG(S.gpa)
   FROM Students S
  WHERE S.dept = 'CS'
[GROUP BY <column list>
  [HAVING <predicate>] ]
[ORDER BY <column list>] ;
```

Before producing output, compute a summary
(a.k.a. an *aggregate*) of some arithmetic expression

Produces 1 row of output

- with one column in this case

Other aggregates: SUM, COUNT, MAX, MIN

Note: can use DISTINCT *inside* the agg function

- SELECT COUNT(DISTINCT S.name) FROM Students S
- vs. SELECT DISTINCT COUNT (S.name) FROM Students S;

GROUP BY


```
SELECT [DISTINCT] AVG(S.gpa), S.dept
  FROM Students S
[WHERE <predicate>]
 GROUP BY S.dept
  [HAVING <predicate>]
[ORDER BY <column list>] ;
```

Partition table into groups with same GROUP BY column values

- Can group by a list of columns

Produce an aggregate result per group

- Cardinality of output = # of distinct group values

Note: can put grouping columns in SELECT list

- For aggregate queries, SELECT list can contain aggs and GROUP BY columns only!
- What would it mean if we said SELECT S.name, AVG(S.gpa) above??

HAVING


```
SELECT [DISTINCT] AVG(S.gpa), S.dept
  FROM Students S
[WHERE <predicate>]
 GROUP BY S.dept
  HAVING COUNT(*) > 5
[ORDER BY <column list>] ;
```

The HAVING predicate is applied *after* grouping and aggregation

- Hence can contain anything that could go in the SELECT list
- I.e. aggs or GROUP BY columns

HAVING can only be used in aggregate queries

(It's an optional clause for GROUP BY)

String Comparisons


```
SELECT S.sname
FROM   Sailors S
WHERE  S.sname LIKE 'B_%B'
```

‘_’ stands for any one character and ‘%’ stands for 0 or more arbitrary characters.

Most DBMSs now support standard regex as well

Let’s do an example!!

```
SELECT year_released, COUNT(*)
FROM Albums
WHERE year_released < 2000
GROUP BY year_released;
```

thanks Michelle!

```
SELECT year_released, COUNT(*)
```
Output total # of albums in each release year

```
FROM Albums
```
Query on albums table

```
WHERE year_released < 2000
```
Only include albums released before 2000

```
GROUP BY year_released;
```
Group by year it was released

Do questions #1-4

Querying Multiple Relations


```
SELECT S.sname
FROM   Sailors AS S, Reserves AS R
WHERE  S.sid=R.sid AND R.bid=102
```

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13

Joins


```
SELECT (column_list)
FROM  table_name
  [INNER  |  {LEFT  | RIGHT  |  FULL } {OUTER}] JOIN table_name
    ON qualification_list
WHERE …
```

INNER is default

Inner/Natural Joins


```
SELECT s.sid, s.sname, r.bid
FROM Sailors s, Reserves r
WHERE s.sid = r.sid
```

```
SELECT s.sid, s.sname, r.bid
FROM Sailors s INNER JOIN Reserves r
ON s.sid = r.sid
```

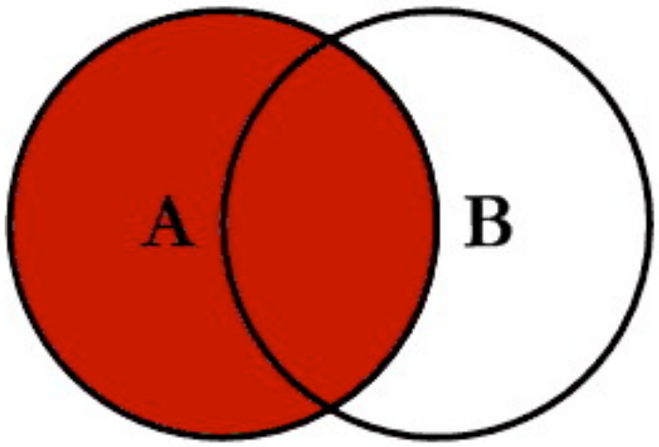
```
SELECT s.sid, s.sname, r.bid
FROM Sailors s NATURAL JOIN Reserves r
```

all 3 are equivalent!

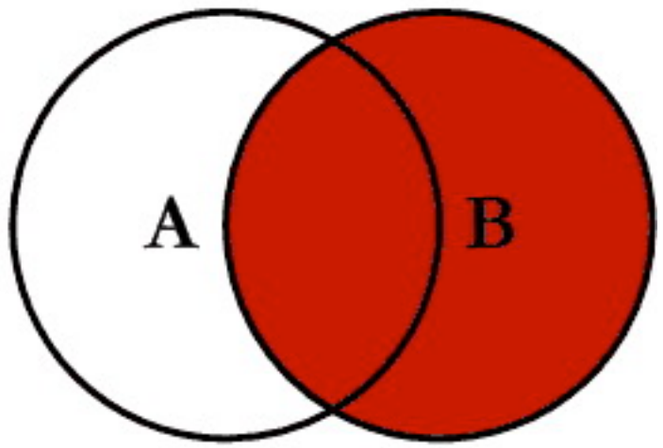
"NATURAL" means equi-join for each pair of attributes with the same name

SQL JOINS

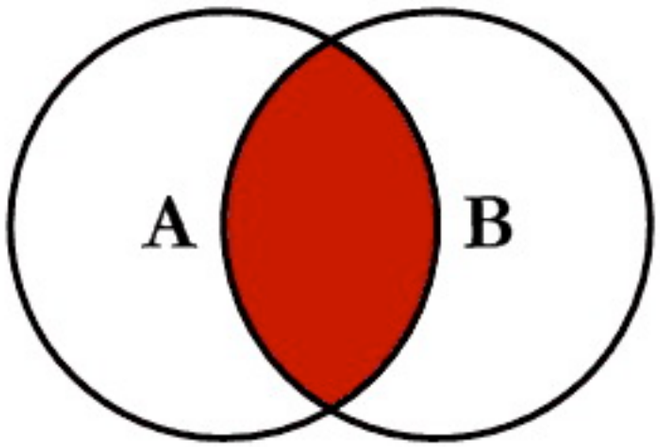
IMPORTANT



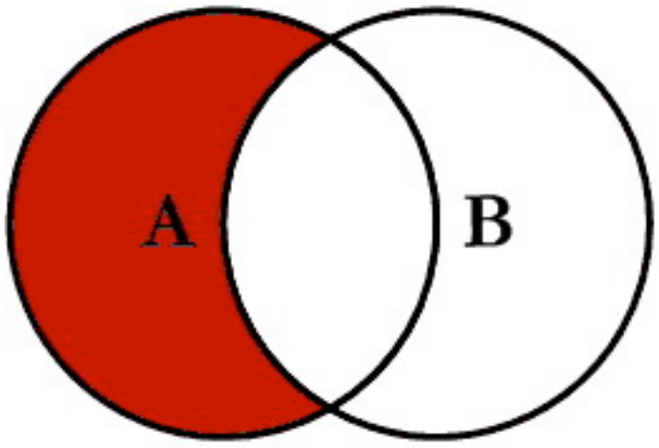

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



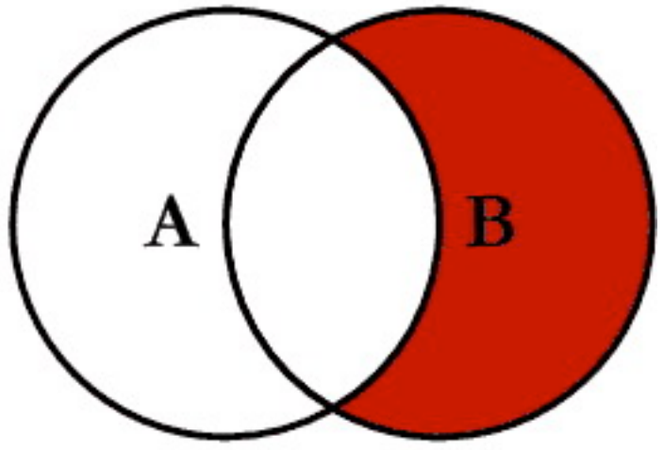

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



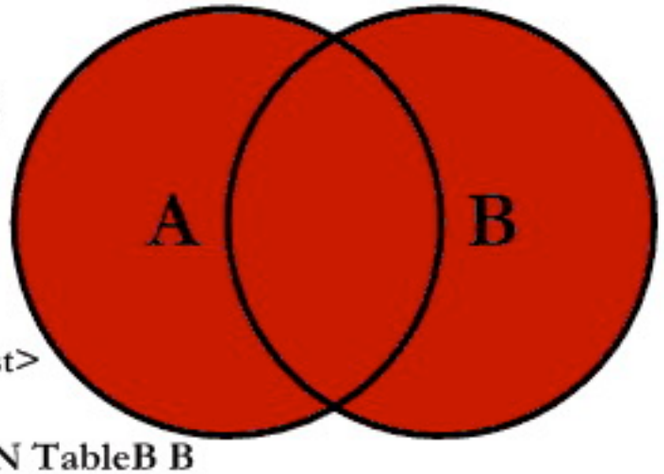

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



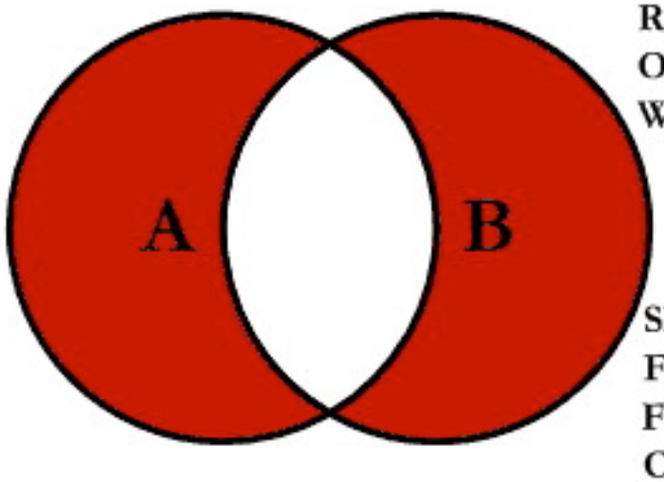

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Do questions #4-8