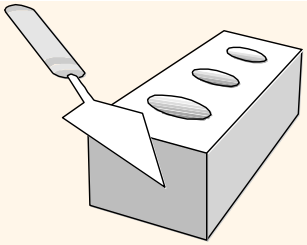
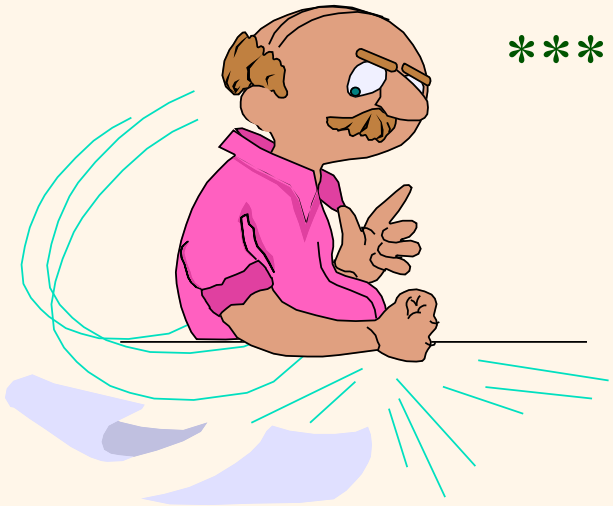


Introduction to Data Management

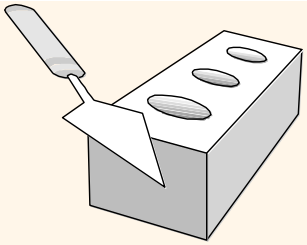
**** The "Flipped" Edition ****



Lecture #15
(SQL IV)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

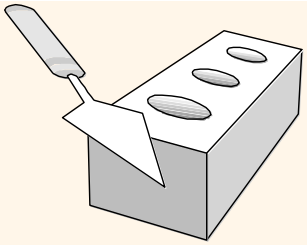


- Roadmap reminder:

Relational Algebra	Ch. 2.5-2.7
Relational Calculus	Wikipedia: Tuple relational calculus
SQL Basics (SPJ and Nested Queries)	Ch. 3.3-3.5
SQL Analytics: Aggregation, Nulls, and Outer Joins	Ch. 3.6-3.9, 4.1
Advanced SQL: Constraints, Triggers, Views, and Security	Ch. 4.2, 4.4-4.5, 4.7
Midterm Exam 2	***Mon, Nov 15*** (during lecture time)

- HW #4 is due at 4PM today
  - Hopefully you've had a nice RelaXing week!
- HW #5 will be out this evening (still in "*Friday mode*")
  - First in our series of ***SQL-based*** HW assignments!
- See the wiki attachments area for data to play with
  - SQLLectureData.txt

Aside: Revisiting ÷ and ∀ in SQL



```
SELECT S.sname FROM Sailors S
WHERE NOT EXISTS (SELECT B.bid FROM Boats B
                  WHERE NOT EXISTS  (SELECT  R.bid FROM Reserves R
                                     WHERE R.bid=B.bid
                                       AND R.sid=S.sid))
```

Sailors S such that ...

*there is **no** boat B that ...*

*S has **not** reserved!*

```
SELECT S.sname FROM Sailors S
WHERE (SELECT COUNT(DISTINCT R.bid) FROM Reserves R
       WHERE R.sid = S.sid) =
       (SELECT COUNT(B.bid) FROM Boats B)
```

Sailors S such that ...

*S's **reserved** boat count matches ...*

*the **total** boat count!*

Ex: Sailors ***and Reserves*** *w/Nulls*

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

*Nulls w/**Aggregates***

	sid	sname	rating	age
1	22	Dustin	7	45.0
2	29	Brutus	1	33.0
3	31	Lubber	8	55.5
4	32	Andy	~~8~~	25.5
5	58	Rusty	10	35.0
6	64	Horatio	~~7~~	35.0
7	71	Zorba	~~10~~	16.0
8	74	Horatio	9	35.0
9	85	Art	4	25.5
10	95	Bob	3	63.5
11	101	Joan	~~3~~	NULL
12	107	Johannes	NULL ← ?	35.0

SELECT COUNT(rating)
FROM Sailors (11)

SELECT
COUNT(DISTINCT rating)
FROM Sailors (7)

SELECT SUM(rating),
COUNT(rating),
AVG(rating)
FROM Sailors
(70, 11, 6.3636)

(Useful, but <u>logically</u> "wrong"!)

*Nulls w/Aggregates & **Grouping***

sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

SELECT COUNT(DISTINCT bid)
FROM Reserves

(4)

SELECT bid, COUNT(*)
FROM Reserves
GROUP BY bid

bid	COUNT(*)
NULL	2
101	2
102	3
103	4
104	2

(!)

Some "dangling" tuple examples

*Nulls w/**Joins** → Inner vs. **Outer** Joins*

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Inner vs. Outer Joins in SQL

```
SELECT DISTINCT s.sname, r.date
FROM Sailors s, Reserves r
WHERE s.sid = r.sid
ORDER BY s.sname
```

	sname character varying (45)	date date
1	Horatio	1998-09-05
2	Horatio	1998-09-08
3	Dustin	1998-10-07
4	Dustin	1998-10-08
5	Dustin	1998-10-10
6	Lubber	1998-11-06
7	Lubber	1998-11-10
8	Lubber	1998-11-12

Inner vs. Outer Joins in SQL (2)

SELECT DISTINCT s.sname, r.date
FROM Sailors s INNER JOIN Reserves r ON s.sid = r.sid
ORDER BY r.date

	sname character varying (45)	date date
1	Horatio	1998-09-05
2	Horatio	1998-09-08
3	Dustin	1998-10-07
4	Dustin	1998-10-08
5	Dustin	1998-10-10
6	Lubber	1998-11-06
7	Lubber	1998-11-10
8	Lubber	1998-11-12

("INNER" is optional, and will be the default type of JOIN assumed if one isn't specified)

Inner vs. ***Outer*** *Joins in SQL (3)*

(1) SELECT DISTINCT s.sname, r.date
FROM Sailors s LEFT OUTER JOIN Reserves r ON s.sid = r.sid
ORDER BY r.date

(2) SELECT DISTINCT s.sname, r.date
FROM Reserves r RIGHT OUTER JOIN Sailors s ON s.sid = r.sid
ORDER BY r.date

	sname character varying (45)	date date
1	Horatio	1998-09-05
2	Horatio	1998-09-08
3	Dustin	1998-10-07
4	Dustin	1998-10-08
5	Dustin	1998-10-10
6	Lubber	1998-11-06
7	Lubber	1998-11-10
8	Lubber	1998-11-12
9	Andy	[null]
10	Art	[null]
11	Bob	[null]
12	Brutus	[null]
13	Joan	[null]
14	Johannes	[null]
15	Rusty	[null]
16	Zorba	[null]

- Variations on a theme:
  - JOIN (or INNER JOIN)
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
  - CROSS JOIN ×

  (Varies from RDBMS to RDBMS)

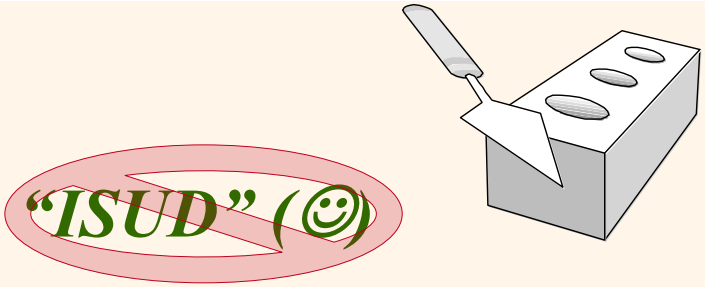
An Algebra Side Note...

- As a side note:
  - The underlying operations are also part of the <u>extended</u> relational algebra, which adds...
    - Outer joins (left, right, and full): Sailors ⟕ Reserves
    - Ordering (sorting): τ age desc (Sailors)
    - Grouping (w/aggregates):

      γ age; avg(rating)->avgrtg (Sailors)
    - ....
- You can explore these on the RelaX site if you want
  - ***And*** you can learn more in the in-class video recording from Friday, Oct. 15th – this was bonus material that I covered!

*Updates: Oh **CRUD***!*

*(***C***reate, **R***etrieve, **U***pdate, **D***elete)*



- Can add one or more tuples using INSERT:

```
INSERT INTO Sailors (sid, sname, rating, age)
VALUES (200, 'Dr. Mike', 10, 64.5)
```

Schema!

```
INSERT INTO Sailors (sid, sname, rating, age)
SELECT ... (your favorite SQL query goes here!) ...
```

- Can remove all tuples satisfying any SQL query condition using DELETE:

```
DELETE FROM Sailors S
WHERE S.age = (SELECT MAX (age) FROM Sailors)
```

*Updates: Oh **CRUD**!*
(Cont.)

- Can change one or more tuples using UPDATE:

```
UPDATE Sailors
SET  sname = 'King Arthur',
     rating = rating + 1
WHERE sname = 'Art';
```

- A few things to note:
  - LHS of SET is column name, RHS is (any) expression
  - WHERE predicate is any SQL condition, which again means SQL subqueries are available as a tool, e.g., to search for targets based on multiple tables' content

That's It for SQL Basics & Analytics!