More SQL: Null, Outer Joins

Today's Database

Sailors

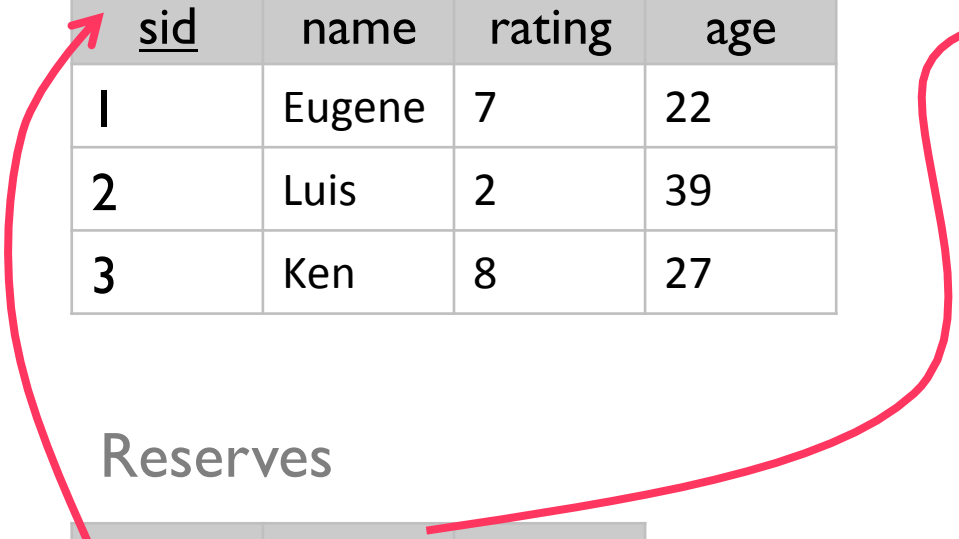
sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Boats

bid	name	color
101	Legacy	red
102	Melon	blue
103	Mars	red

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14
2	103	9/15



Is Reserves table correct?
Day should be part of key

Today's Database

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14
2	103	9/15

PRIMARY KEY (sid, bid)

Sailor can only reserve a boat (e.g. 102) **once**

PRIMARY KEY (sid, bid, day)

Boat (e.g. 102) reserved by 2 sailors on same day

Today's Database

PRIMARY KEY (sid, bid, day)
  Boat (e.g. 102) reserved by 2 sailors on same day

+ UNIQUE (bid, day)? Works!

PRIMARY KEY (sid, bid, day) + UNIQUE (bid, day) =
    PRIMARY KEY(bid, day) + sid NOT NULL

HW1 bugs

Missing CHECK constraints

```
Prof(
  type text,
  check(text = ‘junior’ or text = ‘senior’),
)
```

UNION, INTERSECT, EXCEPT

Algebra: ∪, ∩, -

Combine results from two queries:

SELECT [query1] UNION SELECT [query2]

By default: *distinct results*! (set semantics)
(operator) ALL: Keep duplicates: multi-set

sid of Sailors that reserved red or blue boat

```
SELECT   DISTINCT R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND
         (B.color = ‘red’ OR B.color = ‘blue’)
```

OR

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘red’
UNION
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red or blue boat

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND
         (B.color = ‘red’ OR B.color = ‘blue’)
```

OR

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘red’
UNION ALL
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND
         (B.color = ‘red’ AND B.color = ‘blue’)
```

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘red’
INTERSECT
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R1.sid
FROM     Boats B1, Reserves R1
WHERE
         B1.bid = R1.bid AND

         B1.color = ‘red’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R1.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE
         B1.bid = R1.bid AND

         B1.color = ‘red’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R1.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE
         B1.bid = R1.bid AND
         B2.bid = R2.bid AND
         B1.color = ‘red’ AND B2.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R1.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE    R1.sid = R2.sid AND
         B1.bid = R1.bid AND
         B2.bid = R2.bid AND
         B1.color = ‘red’ AND B2.color = ‘blue’
```

sids of sailors that haven't reserved a boat

```
SELECT   S.sid
FROM     Sailors S

EXCEPT

SELECT   S.sid
FROM     Sailors S, Reserves R
WHERE    S.sid = R.sid
```

Nested Queries

```
SELECT  S.sid
FROM    Sailors S
WHERE   S.sid IN (SELECT   R.sid
                  FROM     Reserves R
                  WHERE    R.bid = 101)
```

Many clauses can contain SQL queries

WHERE, FROM, HAVING, SELECT

Conceptual model:

```
for each Sailors tuple
   run the subquery and evaluate qualification
```

Nested Query vs Join

```
SELECT  S.sid
FROM    Sailors S
WHERE   S.sid IN (SELECT   R.sid
                  FROM     Reserves R
                  WHERE    R.bid = 101)
```

```
SELECT  S.sid
FROM    Sailors S, Reserves R
WHERE   S.sid = R.sid AND R.bid = 101
```

What if a student reserved a boat more than once?

Nested: No duplicates

As checks per row in Sailors

Join: Duplicates

Grabs all from cross product

SET Comparison Operators

x IN *r*: True if value *x* appears in *r*

EXISTS *r*: True if relation *r* is not empty (NOT EXISTS)

x (operator) ANY r: True if x (operator) is true for any row in r

E.g. x IN r is equivalent to x = ANY r

x (operator) ALL r: True if x (operator) is true for all rows in r

E.g. x NOT IN r is equivalent to x <> ALL r

Reference outer table in nested query

```
SELECT   S.sid
FROM     Sailors S
WHERE    EXISTS (SELECT   *
                 FROM     Reserves R
                 WHERE    R.bid = 101 AND
                          S.sid = R.sid)
```

Outer table referenced in nested query

Conceptual model:

```
for each Sailors tuple
   run the subquery and evaluate qualification
```

Sailors whose rating is greater than any sailor named “Bobby”

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ANY (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = ‘Bobby’)
```

How are these different?

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ANY (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = ‘Bobby’)
```

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ALL (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = ‘Bobby’)
```

Rewrite INTERSECT using IN

```
SELECT S.sid
FROM   Sailors S
WHERE  S.rating > 2

INTERSECT

SELECT R.sid
FROM   Reserves R
```

```
SELECT S.sid
FROM   Sailors S
WHERE  S.rating > 2 AND
       S.sid IN (
    SELECT R.sid
    FROM   Reserves R
)
```

Similar trick for EXCEPT → NOT IN

What if want *names* instead of sids?

Names are not unique!

NULL

Field values sometimes unknown or inapplicable

SQL provides a special value *null* for such situations.

The presence of null complicates many issues e.g.,

Is age = null true or false?

Is null = null true or false?

Is null = 8 OR 1 = 1 true or false?

Special syntax "IS NULL" and "IS NOT NULL"

3 Valued Logic (true, false, unknown)

How does WHERE remove rows?

if qualification doesn't evaluate to true

New operators (in particular, outer joins) possible/needed.

NULL

(null > 0) = null

(null + 1) = null

(null = 0) = null

(null AND true) = null

null is null = true

Some truth tables

AND	T	F	NULL
T	T	F	NULL
F	F	F	F
NULL	NULL	F	NULL

OR	T	F	NULL
T	T	T	T
F	T	F	NULL
NULL	T	NULL	NULL

JOINS

```
SELECT [DISTINCT] target_list
FROM table_name
    [INNER | {LEFT |RIGHT | FULL } {OUTER}] JOIN table_name
    ON qualification_list
WHERE ...
```

INNER is default

Difference in how to deal with NULL values

PostgreSQL documentation:

http://www.postgresql.org/docs/9.4/static/tutorial-join.html

Inner/Natural Join

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

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

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

All Equivalent!

Natural Join means equi-join for each pair of attrs with same name

Sailor names and their reserved boat ids

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

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

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

Result

sid	name	bid
1	Eugene	102
2	Luis	102

Sailor names and their reserved boat ids

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

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

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

Result

sid	name	bid
1	Eugene	102
2	Luis	102

Prefer INNER JOIN over NATURAL JOIN. Why?

Sailor names and their reserved boat ids

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

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

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

Result

sid	name	bid
1	Eugene	102
2	Luis	102

Notice: No result for Ken!

Left Outer Join (or No Results for Ken)

Returns all matched rows *and all unmatched rows from table on left of join clause*

(*at least one* row for each row in left table)

```
SELECT s.sid, s.name, r.bid
FROM   Sailors s LEFT OUTER JOIN Reserves r
ON     s.sid = r.sid
```

All sailors & bid for boat in their reservations

Bid set to NULL if no reservation

Left Outer Join

```
SELECT  s.sid, s.name, r.bid
FROM    Sailors s LEFT OUTER JOIN Reserves r
ON      s.sid = r.sid
```

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

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

Result

sid	name	bid
1	Eugene	102
2	Luis	102
3	Ken	NULL

Can Left Outer Join be expressed with Cross-Product?

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

sid	bid	day

```
Sailors x Reserves
```

Result

sid	name	bid

```
Sailors s LEFT OUTER JOIN Reserves r
ON  s.sid = r.sid
```

Result

sid	name	bid
1	Eugene	NULL
2	Luis	NULL
3	Ken	NULL

Can Left Outer Join be expressed with Cross-Product?

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

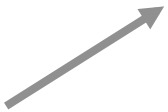
Reserves

sid	bid	day

$$\text{Sailors} \bowtie \text{Reserves}$$

$$\cup$$

$$(\text{Sailors} - (\text{Sailors} \bowtie \text{Reserves})) \times \{(\text{null}, \ldots)\}$$



How to compute this with a query?

Right Outer Join

Same as LEFT OUTER JOIN, but guarantees result for rows in table on right side of JOIN

```
SELECT  s.sid, s.name, r.bid
FROM    Reserves r RIGHT OUTER JOIN Sailors S
ON      s.sid = r.sid
```

FULL OUTER JOIN

Returns all matched *or* unmatched rows from both sides of JOIN

```
SELECT s.sid, s.name, r.bid
FROM   Sailors s FULL OUTER JOIN Reserves r
ON     s.sid = r.sid
```

FULL OUTER JOIN

```
SELECT  s.sid, s.name, r.bid
FROM    Sailors s Full OUTER JOIN Reserves r
ON      s.sid = r.sid
```

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
4	109	9/20

Result

sid	name	bid
1	Eugene	102
2	Luis	102
3	Ken	NULL
NULL	NULL	109

Why is sid NULL?