SQL: Structured Query Language

Part II

You may lose points in the exam!

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

If you want to **join tables**:
Don’t forget the condition!

Ok! Let’s continue with join queries

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

List the name of the Sailors with at least one reservation.

How about this one:

List the name of the Sailors who reserved the boat 102.

Find names of sailors who've reserved boat 102

```
SELECT S.sname
FROM   Sailors S, Reserves 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	$\boxed{102}$	9/12
2	$\boxed{102}$	9/13
2	101	10/20
3	104	11/20

A Note on Range Variables

- Really needed only if ambiguity could arise.
- The previous query can also be written as:

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

It is good style,however, to use range variables always!

About Range Variables

- Another example: List pairs of sailors where the first sailor is older than the second.
  - same table used multiple times in FROM ("self-join")

```
SELECT  X.sname, X.age, Y.sname, Y.age
FROM    Sailors X, Sailors Y
WHERE   X.age > Y.age
```

Sailors

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

X.sname	X.age	Y.sname	Y.age
Jim	39	Fred	22
Jim	39	Nancy	27
Nancy	27	Fred	22

Arithmetic Expressions

```sql
SELECT S.age, S.age-5 AS age1, 2*S.age AS age2
FROM   Sailors S
WHERE  S.sname = 'dustin'
```

```sql
SELECT S1.sname AS name1, S2.sname AS name2
FROM   Sailors S1, Sailors S2
WHERE  2*S1.rating = S2.rating - 1
```

String Comparisons

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sname LIKE ‘A_%A'
```

ANA
ANNA
AYLA
ALINA
…

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

Most DBMSs now support standard regex as well

Find sid's of sailors who've reserved a red or a green boat

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND
         (B.color='red' OR
          B.color='green')
```

sid
3
3
2

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red or a green boat

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND
         (B.color='red' OR
          B.color='green')
```

sid
3
3
2

R.sid	R.bid	R.day	B.bid	B.bname	B.color
1	103	12/9/2015	103	Santa Maria	blue
2	102	13/9/2015	102	Pinta	green
2	103	1/1/2020	103	Santa Maria	blue
3	101	1/1/2020	101	Nina	red
3	102	5/1/2020	102	Pinta	green

Find sid's of sailors who've reserved a red or a green boat

… or:

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND
       B.color='red'
UNION
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='green'
```

sid
3

UNION

sid
3
2

sid
3
2

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red or a green boat

… or:

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

UNION, EXCEPT, INTERSECT eliminate DUPLICATES!

UNION ALL, EXCEPT ALL, INTERSECT ALL keep DUPLICATES!

Find sid's of sailors who've reserved a red and a green boat

```
SELECT R.sid
FROM   Boats B,Reserves R
WHERE  R.bid=B.bid AND
   (B.color='red' AND B.color='green')
```

Find sid’s of sailors who’ve reserved a red and a green boat

```
SELECT S.sid
FROM   Boats B, Reserves R
WHERE     R.bid=B.bid
          AND B.color='red'
INTERSECT
SELECT S.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='green'
```

sid
3

INTERSECT

sid
3
2

sid
3

Reserves

sid	bid
3	101
3	102
2	102

Boats

bid	color
101	red
102	green

(smaller R table to fit to slide)

R1.sid	R1.bid	R2.sid	R2.bid	B1.bid	B1.color	B2.bid	B2.color
3	101	3	101	101	red	101	red
3	101	3	102	101	red	102	green
3	102	3	101	102	green	101	red
3	102	3	102	102	green	102	green
2	102	2	102	102	green	102	green

Find sid's of sailors who've reserved a red and a green boat

- Or we could use a self-join:

```
SELECT R1.sid
FROM   Boats B1, Reserves R1,
       Boats B2, Reserves R2
WHERE R1.sid=R2.sid
        AND R1.bid=B1.bid
        AND R2.bid=B2.bid
        AND B1.color='red'
        AND B2.color='green'
```

DISTINCT would it help here?

Find sid and names of sailors who have not reserved boat#102

```
SELECT S.sid, S.sname
FROM   Sailors S

EXCEPT

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

sid	sname
1	Fred
2	Jim
3	Nancy

sid	sname
3	Nancy

sid	sname
1	Fred
2	Jim

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

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Find sid ~~and names~~ of sailors who have not reserved boat#102

```
SELECT S.sid, ~~S.sname~~
FROM   Sailors S

EXCEPT

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

$\longrightarrow$ Can we use Reserves instead?

When is it ok?

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27
4	Mary	1	17

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries: IN

IN: Allows us to test whether a value is in a given set of elements (usually generated by another SQL query)

Find names of sailors who've ids 1 or 2 or 3 or 4 or 5

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN (1, 2, 3, 4, 5)
```

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
9	Mike	4	57
4	Mary	1	17
22	Jake	10	57
3	Nancy	8	27

sname
Fred
Jim
Mary
Nancy

Nested Queries: IN

IN: Allows us to test whether a value is in a given set of elements (usually generated by another SQL query)

Find names of sailors who've reserved boat #102:

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN
              (SELECT R.sid
               FROM   Reserves R
               WHERE  R.bid=102)
```

sid
1
2

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

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Find sid's of sailors who've reserved a red and a green boat

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='red'
       AND R.sid IN
                    (SELECT R2.sid
                     FROM Boats B2,Reserves R2
                     WHERE R2.bid=B2.bid
                           AND B2.color='green‘)
```

sid
2
3

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red **and** a green boat

- Or, we could use IN
  - INTERSECT can be **re-written** using IN

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='red'
       AND R.sid IN (SELECT R2.sid
                     FROM Boats B2,Reserves R2
                     WHERE R2.bid=B2.bid
                           AND B2.color='green‘)
```

Nested Queries: NOT IN

*Find names of sailors who've **not** reserved boat #102:*

```
SELECT   S.sname
FROM     Sailors S
WHERE    S.sid NOT IN
    (SELECT  R.sid
     FROM    Reserves R
     WHERE  R.bid=102)
```

sid
1
2

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

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries: NOT IN

*Find names of sailors who've **not** reserved boat #102:*

```
SELECT   S.sname
FROM     Sailors S
WHERE    S.sid NOT IN
    (SELECT  R.sid
     FROM    Reserves R
     WHERE  R.bid=102)
```

- **EXCEPT can be re-written using NOT IN**

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

- Subquery must be recomputed for each Sailors tuple.
  - Think of subquery as a function call that runs a query

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
        FROM  Reserves R
        WHERE R.bid=102 AND S.sid=R.sid)
```

sid	bid	day

sid	bid	day
2	102	9/13

sid	bid	day
1	102	9/12

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

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries with Correlation

Finds sailors with at most one reservation for boat #102

```
SELECT  S.sname
FROM    Sailors S
WHERE UNIQUE
      (SELECT  R.bid
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

UNIQUE checks for duplicate tuples. When applied to a subquery, it is **TRUE**:

- if no row appers twice,
- if the answer is empty set

In the subquery, why do we have to replace * by *R.bid*?

Nested Queries with Correlation

Finds sailors with at most one reservation for boat #102

```
SELECT   S.sname
FROM     Sailors S
WHERE UNIQUE
       (SELECT   R.bid
         FROM   Reserves R
         WHERE R.bid=102 AND S.sid=R.sid)
```

bid

bid
102

bid
102
102

sname
Fred
Nancy

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

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20
2	102	11/20

In the subquery, why do we **have to** replace * by *R.bid*?