

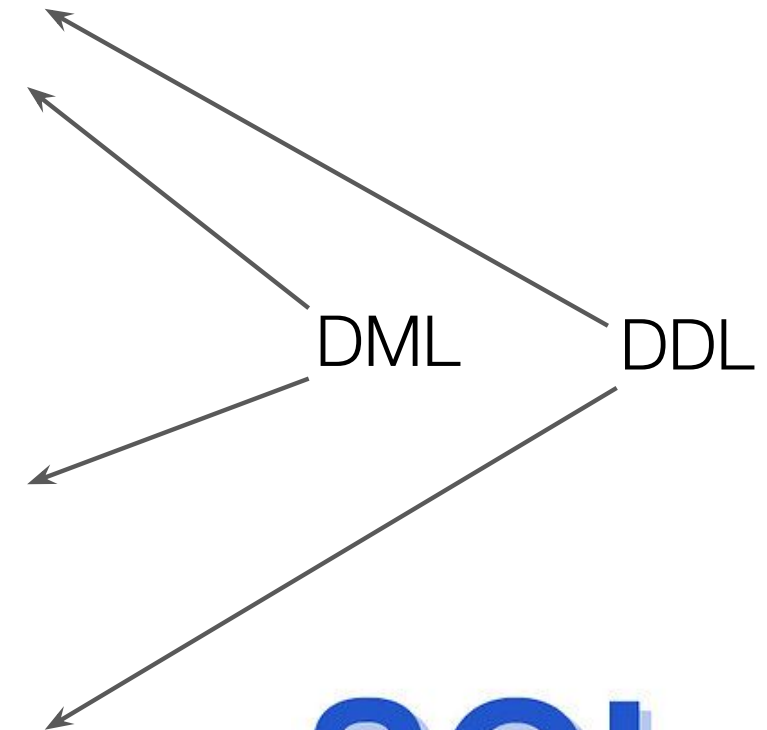


SQL Queries

Chapter 5

Structured Query Language

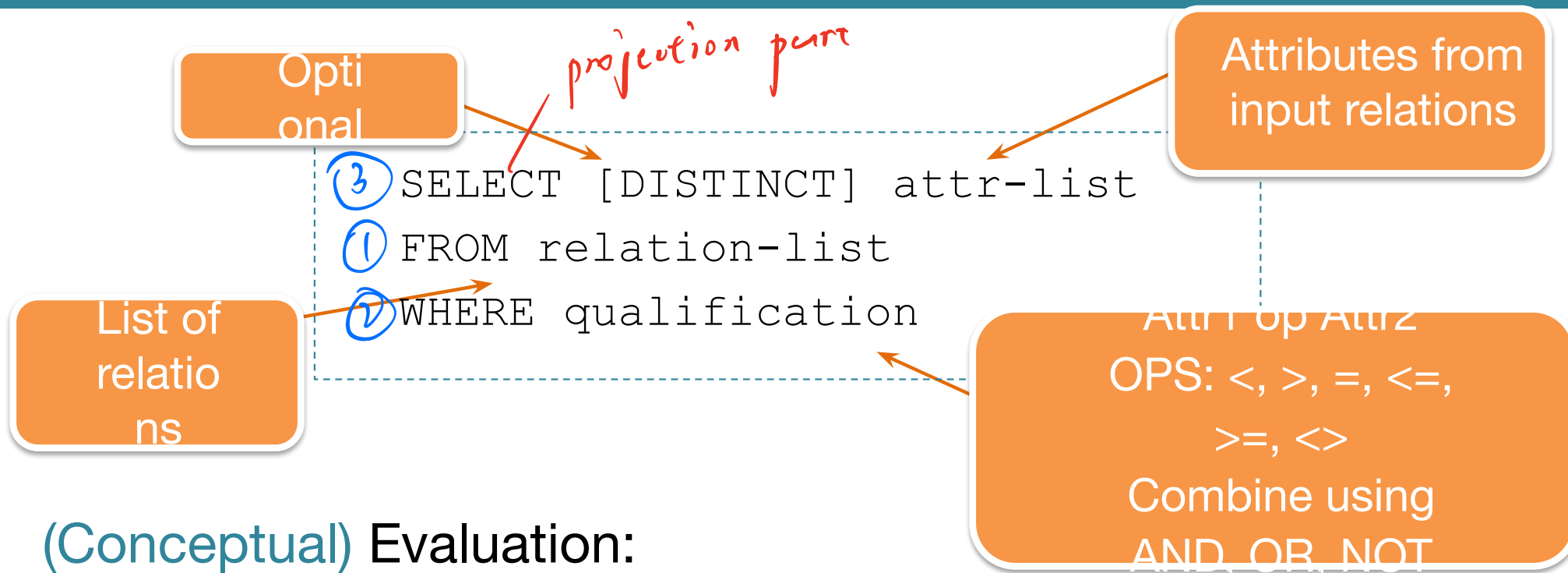
- | | | |
|--------------------|--------|-----------------|
| • Create a Table | Create | (Delete/Modify) |
| • Add new records | Insert | |
| • Retrieve records | Select | |
| • Update records | Update | |
| • Delete records | Delete | |
| | | |
| • Create a View | Create | |
| • Update a View | Update | |



SQL Query Language

- Implements relational algebra...
Select, Project, Join, Set operators “Relationally complete”
- And much more...
 - Correlated subqueries
 - Ordering of results
 - Aggregate queries (e.g., SUM, MAX, AVG)
 - Three-valued logic for NULL values
 - Etc.

Basic SQL Query



(Conceptual) Evaluation:

1. Take cross-product of relation-list
2. Select rows satisfying qualification
3. Project columns in attr-list
(eliminate duplicates only if DISTINCT)

Optimizer
chooses
efficient plan!

Example of Basic Query:

Cross-product syntax:

```
SELECT S.sname  
FROM Sailors S, Reserves R  
WHERE S.sid = R.sid AND  
R.bid = 103;
```

Reserves

sid	bid	rday
22	101	10/10
58	103	11/12

Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	35
31	Lubber	8	55

like natural join

Reserves x Sailors

sid	bid	rday	sid	sname	rating	age
22	101	10/10	22	Dustin	7	45
22	101	10/10	58	Rusty	10	35
22	101	10/10	31	Lubber	8	55
58	103	11/12	22	Dustin	7	45
58	103	11/12	58	Rusty	10	35
58	103	11/12	31	Lubber	8	55

Question??



Sailors

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	Dustin	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND
R.bid = 103;
```

The result of executing this query will be:

- A. {Dustin, Lubber} *select distinct*
- B. {Dustin, Dustin, Lubber}
- C. {Dustin, Lubber, Dustin}

no guaranteed order

Eliminating Duplicates



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

Another Example



- Schema:

Sailors (**sid**, sname, rating, age)

Boats (**bid**, bname, color)

Reserves (**sid**, **bid**, rday)

Sailors				Reserves			Boats		
sid	sname	rating	age	sid	bid	rday	bid	bname	color
25	Rustin	7	45.5	25	101	12/01/85	101	Interlock	blue
26	Rustin	1	35.5	22	102	12/01/85	102	Interlock	red
31	L. Biewer	8	37.5	22	101	12/01/84	101	Interlock	green
35	Arny	8	27.5	22	104	12/01/84	104	Marine	red
57	Helen	10	37.5	21	102	11/01/84			
64	Helen	7	35.5	21	101	11/01/85			
71	Charles	10	38.0	21	104	11/01/84			
74	Helen	8	35.0	24	101	9/01/8			
85	Art	3	25.5	24	102	9/01/8			
95	Bob	3	21.5	24	108	9/01/8			

- Find the colors of boats reserved by any sailor named Rusty

```
SELECT B.color
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
      S.sname = 'Rusty';
```


Note on Range Variables



Needed when same relation appears twice in FROM clause

```
SELECT S1.sname, S2.sname  
FROM Sailors S1, Sailors S2  
WHERE S1.age > S2.age;
```

What does this
Query compute?



It is considered good style to use range variables

Another Example



- Find pairs of sailors where the first one has half the rating of the second one:



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

Sailors

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	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Incrementing the result



- ① Compute increments for the ratings of persons who have sailed two different boats on the same day:

```
SELECT S.sname, S.rating+1 AS rating
FROM Sailors S, Reserves R1, Reserves R2
WHERE S.sid=R1.sid AND S.sid=R2.sid AND
      R1.day=R2.day AND R1.bid<>R2.bid;
```

define the new column you created, (apply ONLY of the output)

Sailors

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	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

INNER Joins



The join we just saw is also called an INNER JOIN
(We will see outer joins shortly)

Join syntax:

(default inner join)

```
SELECT S.sname
FROM Sailors S JOIN Reserves R ON S.sid = R.sid
WHERE R.bid = 103;
```

Eqvt. Inner join syntax:

```
SELECT S.sname
FROM Sailors S INNER JOIN Reserves R ON
    S.sid = R.sid
WHERE R.bid = 103;
```

ORDER BY Clause



Helps sort the result for presentation

Attribute(s) in **ORDER BY** clause (must be) in **SELECT** list

Find the names
and ages of all sailors,
in increasing order of age

```
SELECT S.sname, S.age  
FROM Sailors S  
ORDER BY S.age [ASC]
```

Find the names
and ages of all sailors,
in decreasing order of age

```
SELECT S.sname, S.age  
FROM Sailors S  
ORDER BY S.age DESC
```

↓
Case thing in query (ordered by options: in select clause)

ORDER BY Clause



```
SELECT S.sname, S.age, S.rating  
FROM Sailors S  
ORDER BY S.age ASC, S.rating DESC
```

What does this
query
compute?

tie-breaker

Find the names, ages, and rankings of all sailors.

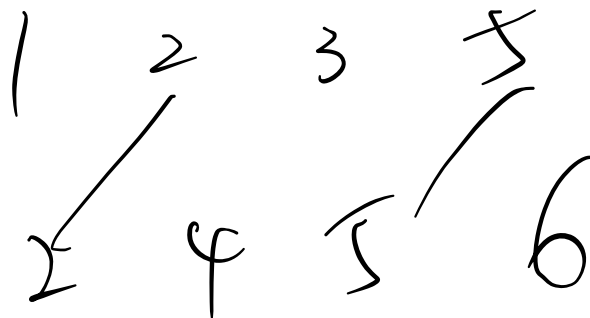
Sort the result in increasing order of age.

If there is a tie, sort those tuples in decreasing order of rating.

Set Operators

~~opposite~~
to
select
default

- UNION ^{default} (eliminates duplicates)
- UNION ALL (keeps duplicates)
- INTERSECT — ^{multiset intersection logic}
(duplicates allowed)
- EXCEPT or MINUS (set difference)



Union Example



Find names of sailors who have reserved a red or a green boat.

Try without UNION



Sailors

sld	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	3	25.5
95	Bob	3	63.5

Reserves

sld	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Union Example



Find names of sailors who have reserved a red or a green boat.

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



```
SELECT  S.sname
FROM    Sailors S, Reserves R, Boats B
WHERE   S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
UNION
SELECT  S.sname
FROM    Sailors S, Reserves R, Boats B
WHERE   S.sid = R.sid and R.bid = B.bid AND B.color = 'green';
```

Question??



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

What is wrong with the
above query?

- A. Extra parentheses on the last line should not be there
- ☒ B. A boat cannot be multi-colored.
A boat with red and green stripes would not satisfy last line
- C. Both A and B above
- D. Neither. There is nothing wrong.



Intersect



Find names of sailors who have reserved a red and a green boat.

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



Intersect



Find names of sailors who have reserved a red and a green boat.

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

*we need 2
instances
of this*

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



Set Difference Example

Find tuples in A that are not in B

```
SELECT * FROM A  
MINUS  
SELECT * FROM B;
```

MINUS and EXCEPT are synonyms

Set Difference Example



Find sids of sailors who have reserved red, but not green boats.

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

~~EXCEPT~~

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

Simpler?



Sailors

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	Rusly	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Set Difference Example



Find sids of sailors who have reserved red, but not green boats.

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

Sailors

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	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/8/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

More Set Comparison Operators

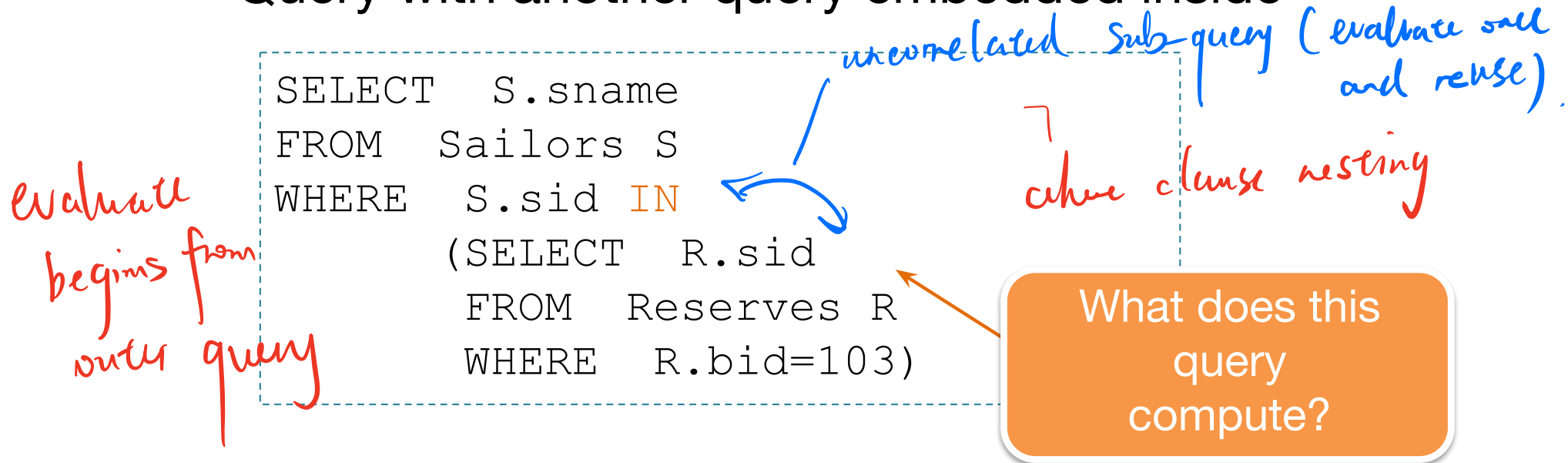
- Set comparisons:
 - $attr \text{ IN } R$: true if R contains $attr$
 - $\text{EXISTS } R$: true if R is not an empty relation
 - $\text{UNIQUE } R$: true if no duplicates in R
 - You can use **NOT** with these, e.g. NOT EXISTS
- Also available ANY or ALL: (op is $<$, \leq , $>$, \geq , $=$, \neq)
 - $attr > \text{ANY } R$: some element of R satisfies the condition that $attr >$ that element
 - $attr < \text{ALL } R$: all elements of R satisfy the condition that $attr <$ element

⇒ booleans
in
conditions

Nested Queries



Query with another query embedded inside



Conceptual evaluation:

For each row of Sailors, evaluate the subquery over reserves.

To find sailors who have not reserved 103, use NOT IN.

Over-Use of Nesting



- Common error by novice SQL programmers
- Query optimizers not as good at optimizing queries across nesting boundaries
- Try hard first to write non-nested



```
SELECT  DISTINCT S.sname  
FROM    Sailors S, Reserves R  
WHERE   S.sid = R.sid AND R.bid = 103;
```

Example



- Q1: What does this query compute?



```
SELECT S.sid  
FROM Sailors S  
WHERE S.rating > ANY (SELECT S2.rating  
                      FROM Sailors S2  
                      WHERE S2.name = 'John');
```

existential quantification

- Q2: Rewrite the query without using a nested query

Example



- Q1: What does this query compute?



```
SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                      FROM Sailors S2
                      WHERE S2.name = 'John');
```

*extensive
can be avoidable
most of time*

- Q2: Rewrite the query without using a nested query

```
SELECT DISTINCT S.sid
FROM Sailors S, Sailors S2
WHERE S.rating > S2.rating AND S2.name = 'John';
```

Example



Find sailors (all their info) whose rating is greater than that of all sailors called Horatio:

```
SELECT  *
FROM    Sailors S
WHERE   S.rating > ALL (SELECT  S2.rating
                        FROM    Sailors S2
                        WHERE   S2.sname='Horatio')
```

Question??



```
SELECT  *
FROM    Sailors S
WHERE    S.rating > ALL (SELECT  S2.rating
                        FROM    Sailors S2
                        WHERE    S2.sname='Horatio')
```

every one passes (with an arrow pointing from the text to the `> ALL` condition)

no one qualified (with an arrow pointing from the text to the `ALL` keyword)

What if there is no sailor with sname Horatio? Then,
The "> ALL" condition is

- A. True for every sailor in S
- B. False for every sailor in S
- C. Undefined

Aggregate Operators



```
SELECT COUNT (*)  
FROM Sailors S
```

```
SELECT COUNT  
  (DISTINCT S.name)  
FROM Sailors S
```

```
SELECT AVG(S.age)  
FROM Sailors S  
WHERE S.rating=10
```

rating = 10

```
SELECT S.sname  
FROM Sailors S  
WHERE S.rating = (SELECT MAX(S2.rating)  
                  FROM Sailors S2)
```

*usually
do
with
one
column*

COUNT (*)

(# rows in table)

COUNT ([DISTINCT] A)

SUM ([DISTINCT] A)

AVG ([DISTINCT] A)

MAX (A) *Can use Distinct*

MIN (A) *Can use Distinct*

*single column**

*ignore
NULL*

```
SELECT AVG(DISTINCT S.age)  
FROM Sailors S  
WHERE S.rating=10
```

Sailors

sld	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	3	25.5
95	Bob	3	63.5

unrelated

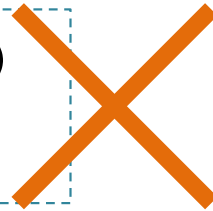
*★ nested query only useful when
it can't be done after*

Aggregate Query - Example



Find name and age of the oldest sailor(s)

```
SELECT  S.sname, MAX (S.age)
FROM    Sailors S
```



⇒ output: name of all sailors, paired with max age

```
SELECT  S.sname, S.age
FROM    Sailors S
WHERE   S.age = (SELECT MAX (S2.age)
                  FROM    Sailors S2)
```

How many tuples
in the result?

Does not work in

sqlite3

```
SELECT  S.sname, S.age
FROM    Sailors S
WHERE   S.age >= ALL (SELECT  S2.age
                       FROM    Sailors S2)
```

= MAX

GROUP BY



Conceptual evaluation

- Partition data into groups according to some criterion
- Evaluate the aggregate for each group

Example:

For each rating level, find the age of the youngest sailor

```
SELECT  MIN (S.age), S.rating  
FROM    Sailors S  
GROUP BY S.rating
```

How many tuples
in the result?



GROUP BY and HAVING

join first, where

do big join

qualify rows

qualify buckets

```
① SELECT [DISTINCT] target-list  
② FROM relation-list  
③ WHERE qualification  
④ GROUP BY grouping-list  
⑤ HAVING group-qualification
```

Target-list contains:
1. Attribute names
(subset of grouping-list)
2. Aggregate operations
e.g. min(age)

Conceptual Evaluation:

1. Eliminate tuples that don't satisfy qualification
2. Partition remaining data into groups
3. Eliminate groups according to group-qualification
4. Evaluate aggregate operation(s) for each group

Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 **such** sailors



```
SELECT  S.rating, MIN (S.age)
FROM    Sailors S
WHERE   S.age >= 18
GROUP BY  S.rating
HAVING    COUNT (*) >= 2
```

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
29	Brutus	1	33.0
58	Rusty	10	35.0



Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 **such** sailors

```
SELECT  S.rating, MIN (S.age)
FROM    Sailors S
WHERE   S.age  $\geq$  18
GROUP BY S.rating
HAVING  COUNT (*)  $\geq$  2
```

sid	sname	rating	age
29	Brutus	1	33.0
22	Dustin	7	45.0
64	Horatio	7	35.0
31	Lubber	8	55.5
58	Rusty	10	35.0

rating	age
7	35.0

Answer
relation

Grouped Sailors

NULL Values in SQL

- NULL represents 'unknown' or 'inapplicable'
- Query evaluation complications
 - Q: Is (rating > 10) true when rating is NULL?
 - A: Condition evaluates to 'unknown' (not T or F)
- What about AND, OR connectives?
 - Need 3-valued logic
- WHERE clause eliminates rows that **don't evaluate to true**

p	q	p AND q	p OR q
T	T	T	T
T	F	F	T
T	U	U	T
F	T	F	T
F	F	F	F
F	U	F	U
U	T	U	T
U	F	F	U
U	U	U	U

throw that row away

Question??



Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	NULL
31	Lubber	8	55

What does this query return?

- A. {Dustin, Rusty, Lubber}
- B. {Dustin, Lubber} ✓
- C. {Lubber}
- D. Error

```
SELECT sname
FROM sailors
WHERE age > 45 OR age <= 45
```

F
U
T

OR

T
U
F

T
U
T

Question??



Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	NULL
31	Lubber	8	55

```
SELECT AVG(age)
FROM sailors
```

ignore NULLS

What does this query return?

- A. 50
- B. NULL
- C. Error
- D. A range of possible values based on domain constraint on age

Outer Joins



Sailors

sid	sname	rating	age
22	dustin	7	45.0
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/99

even if rusty made NO reserves:

```
SELECT S.sid, R.bid
FROM Sailors S NATURAL LEFT [OUTER]
JOIN Reserves R
```

special case of inner join.

replace ON.

return tuples
from left table
that not participates
in join

Result

sid	bid
22	101
58	null

Similarly:

- Right Outer Join
- Full Outer Join

left join right join

Note: OUTER is default, when using LEFT, RIGHT, or FULL

More Outer Joins

```
SELECT S.sid, R.bid
FROM Sailors S RIGHT[OUTER] JOIN
Reserves R ON S.sid=R.sid;
```

Sailors

sid	sname	rating	age
58	Rusty	10	35
31	Lubber	8	55

Result

sid	bid
null	101
58	103

Reserves

sid	bid	rday
22	101	10/10
58	103	11/12

if full outer join \Rightarrow 31 | null

JOIN Syntax with Multiple Tables

Sailors

sid	sname	rating	age
22	dustin	7	45.0
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/99

```
SELECT S.sname, B.bname
FROM Sailors S JOIN Reserves R ON (S.sid = R.sid)
      JOIN Boats B ON (R.bid = B.bid)
WHERE S.name = 'dustin';
```

Similarly:

- RIGHT [OUTER] JOIN ON...
- LEFT [OUTER] JOIN on...
- FULL [OUTER] JOIN ON...
- NATURAL JOINS (outer and inner)

For each red boat, find the number of reservations for this boat*



```
SELECT B.bid, COUNT (*) AS scount
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='red'
GROUP BY B.bid
```

```
SELECT B.bid, COUNT (*) AS scount
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
```

```
GROUP BY B.bid
HAVING B.color = 'red'
```

*boat color is
not a property of bucket,*

Would this work?
Note: one color per bid



Subtle Errors



Find the sid of sailors who have reserved exactly one boat

```
SELECT S1.sid FROM Sailors S1
MINUS
SELECT R1.sid
FROM Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid AND R1.bid=B1.bid
      AND R2.bid=B2.bid AND R1.bid <> R2.bid;
```

There is a subtle error in the above



Sailors

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	Arl	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/6/98
74	103	9/6/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Swiss with a
boat.

Error Fixed



Find the sid of sailors who have reserved exactly one boat

```
SELECT R3.sid FROM Reserves R3
MINUS
SELECT R1.sid
FROM Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid AND R1.bid=B1.bid
      AND R2.bid=B2.bid AND R1.bid <> R2.bid;
```

Error Fixed: Another Solution



Find the sid of sailors who have reserved exactly one boat

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid AND B.bid = R.bid
GROUP BY S.sid
HAVING COUNT(*) = 1;
```

Question??



Find names of sailors who have reserved a red and a green boat

name not a key.

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

Which of these is the one FALSE statement?

- A. The SQL query is legal (meets SQL standard spec.)
- B. The SQL query correctly expresses the query in English above.
- C. The first 3 lines find names of sailors who have reserved a red boat.
- D. There could be a sailor named Dustin (sid 22) who reserved a red boat and another sailor named Dustin (sid 37) who reserved a green boat.

Intersect on Non-Key



Find the names of sailors who have reserved a red and a green boat

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

Fix it: (1) by using a view
and (2) without a view.



Error Fixed



Find the names of sailors who have reserved a red and a green boat

```
CREATE VIEW RedGreenSailors AS
SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green';

SELECT S.sname
FROM Sailors S, RedGreenSailors R
WHERE S.sid = R.sid;
DROP VIEW RedGreenSailors;
```

Error Fixed: Another Solution



Find the names of sailors who have
reserved a red and a green boat

Get rid of the VIEW

from clause nesting
nested

```
SELECT S.sname
FROM Sailors S,
  (SELECT S.sid
   FROM Sailors S, Reserves R, Boats B
   WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
  ) INTERSECT
  (SELECT S.sid
   FROM Sailors S, Reserves R, Boats B
   WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green')
RedGreenSailors
WHERE S.sid = RedGreenSailors.sid;
```

Yet Another Solution



Find the names of sailors who have
reserved a red and a green boat
Use WHERE nesting rather than FROM nesting

```
SELECT S.sname
FROM Sailors S,
WHERE S.sid IN (SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green')
```

where clause nesting

Another sol:
See Q8 in
Ch. 5, p.150

Find the age of the youngest sailor with age > 18, for each rating with at least 2 sailors (of any age)



*rather than
count those who
are only
>18*

```
SELECT  S.rating, MIN (S.age) AS MINAGE
FROM    Sailors S
WHERE   S.age > 18
GROUP BY S.rating
HAVING  1 < (SELECT COUNT (*) FROM Sailors S2
             WHERE S2.rating=S.rating)
```

- Subquery in the **HAVING** clause
- Compare this with the query where we considered only ratings with 2 sailors over 18!

Find the age of the youngest sailor with age > 18, for each rating with at least 2 sailors (of any age)



```
SELECT  S.rating, MIN (S.age) AS MINAGE
FROM    Sailors S
WHERE   S.age > 18
GROUP BY S.rating
HAVING  1 < (SELECT COUNT (*) FROM Sailors S2
            WHERE S2.rating=S.rating)
```

*table with single. col single row
⇒ single value*

*straight
Comparison
allowed*

- Compare this with the query where we considered only ratings with 2 sailors over 18!

```
SELECT  S.rating, MIN (S.age)
FROM    Sailors S
WHERE   S.age >= 18
GROUP BY S.rating
HAVING  COUNT (*) >= 2
```

Find ratings for which the average age is the minimum of the average age over all ratings*



Aggregate operations
cannot be nested!

WRONG: →

```
SELECT  S.rating
FROM    Sailors S
WHERE   AVG(S.age) =
        (SELECT  MIN (AVG (S2.age))
         FROM    Sailors S2
         GROUP BY S2.rating)
```

all rating group

Correct solution

```
SELECT  T.rating, T.avgage
FROM    (SELECT  S.rating, AVG (S.age) AS avgage
         FROM    Sailors S
         GROUP BY S.rating) T
WHERE   T.avgage = (SELECT  MIN (T.avgage) FROM T);
```

Meets SQL/92 standard, but some products may not support this!

Solution Using Views



If previous solution does not work, you can define T as a view

```
CREATE VIEW AVG_AGE_BY_RATING AS
SELECT S.rating, AVG(S.age) AS avgage
FROM Sailors S
GROUP BY S.rating;

SELECT T.rating, T.avgage
FROM AVG_AGE_BY_RATING T
WHERE T.avgage= (SELECT MIN(A.avgage)
                 FROM AVG_AGE_BY_RATING A);
```