Database Management Systems
L5

Umass Boston
Summer 2023
Cristina Maier

Some slides are based on “Database Management System, 3rd ed., by Ramakrishnan and Gehrke

Topics

- Introduction to DBMS
- Relational Data Model
- *Relational Algebra*
- Conceptual Design: the Entity-Relationship Model
- Structured Query Language (SQL)
- Application Development (Java, Python)
- Schema Refinement and Normal Forms
- Database Security and Authorization
- Some NoSQL topics (If time permitted)

Example

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0
60	andy	10	60.0
61	mary	8	25.0

SELECT rating, MIN(age)
FROM Sailors
GROUP BY rating;

rating	MIN(age)
7	45.0
8	25.0
10	35.0

SELECT rating, MIN(age)
FROM Sailors
GROUP BY sname;

?

Example(cont.)

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0
60	andy	10	60.0
61	mary	8	25.0

SELECT rating, MIN(age)
FROM Sailors
GROUP BY rating;

rating	MIN(age)
7	45.0
8	25.0
10	35.0

SELECT rating, MIN(age)
FROM Sailors
GROUP BY sname;

Invalid query!
Why?

Example with HAVING

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0
60	andy	10	60.0
61	mary	8	25.0

```
SELECT rating, AVG(age)
FROM Sailors
GROUP BY rating;
```

rating	AVG(age)
7	45.0
8	40
10	46.66

```
SELECT rating, AVG(age)
FROM Sailors
GROUP BY rating
HAVING AVG(age)>=45;
```

rating	AVG(age)
7	45.0
10	46.66

More Group qualification functions

- So far we have seen group qualification based on a property of the group
  - e.g.: aggregate function computed for each group
- Recent SQL standard allows group qualification based on a property of individual records
  - EVERY(condition):True if condition holds for every group tuple
  - ANY(condition): True if condition holds for any group tuple
  - *HAVING .. ANY .., HAVING … EVERY … are not currently supported by Oracle*

Example

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, MIN(s.age)
FROM Sailors3 s
GROUP BY s.rating;
```

rating	MIN(age)
7	45.0
8	25.0
10	35.0

```
SELECT s.rating, MIN(s.age)
FROM Sailors3 s
GROUP BY s.rating
HAVING  EVERY (s.salary>=20000);
```

rating	MIN(age)
7	45.0
8	25.0

Group with rating=10 is discarded because not all records from the that group have a salary >= 20000. Record with sid=38 does not satisfy the condition.

Next

- More Group By Exercises
- Nested Queries

Example 1

Find the min age for each rating. Include only data with salaries greater than 25000.

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, MIN(s.age)
FROM Sailors3 s
WHERE s.salary > 25000
GROUP BY s.rating
```

WHERE salary >25000

sid	sname	rating	age	salary
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~20000~~
31	lubber	8	55.0	30000
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~15000~~
59	rusty	10	45.0	40000
~~60~~	~~andy~~	~~10~~	~~60.0~~	~~25000~~
61	mary	8	25.0	30000

The GROUP by will be applied to

sid	sname	rating	age	salary
31	lubber	8	55.0	30000
59	rusty	10	45.0	40000
61	mary	8	25.0	30000

Example 1 (cont.)

Find the min age for each rating. Include only data with salaries greater than 25000.

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, MIN(s.age)
FROM Sailors3 s
WHERE s.salary > 25000
GROUP BY s.rating
```

GROUP BY s.rating

rating	MIN(s.age)
8	25.0
10	45.0

Result is:

rating	MIN(s.age)
8	25.0
10	45.0

Example 2

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an avg Salary <29000

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating >7
GROUP BY s.rating
HAVING AVG(s.salary) < 29000;
```

Example 2

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an avg Salary <29000

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING AVG(s.salary)<29000
```

WHERE s.rating >7

sid	sname	rating	age	salary
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~20000~~
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

The GROUP by will be applied to

sid	sname	rating	age	salary
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Example 2 (cont.)

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an avg Salary <29000

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING AVG(s.salary)<29000
```

The GROUP by will be applied to

sid	sname	rating	age	salary
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

GROUP BY s.rating

rating	AVG(salary)
8	30000
10	26666.66

Example 2 (cont.)

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an avg Salary <29000

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING AVG(s.salary)<29000
```

HAVING AVG(s.salary)>29000

rating	AVG(salary)
~~8~~	~~30000~~
10	25666.66

Result is:

rating	AVG(salary)
10	26666.66

Example 3
(The aggregate from HAVING does not need to be present in the target list)

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Find the AVG salary for each rating
greater than 7.
Keep only the ratings with an MIN Salary >20000

SQL Query?

Example 3
(The aggregate from HAVING does not to be present in the target list)

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an MIN Salary >20000

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING MIN(s.salary)>20000;
```

WHERE s.rating >7

sid	sname	rating	age	salary
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~20000~~
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

The GROUP by will be applied to

sid	sname	rating	age	salary
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Example 3 (cont.)

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an MIN Salary >20000.

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING MIN(s.salary)>20000;
```

The GROUP by will be applied to

sid	sname	rating	age	salary
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

rating	MIN(salary)	AVG(salary)
8	30000	30000
10	15000	25666.66

Example 3 (cont.)

Sailors3

sid	sname	rating	age	salary
22	dustin	7	45.0	20000
31	lubber	8	55.0	30000
58	rusty	10	35.0	15000
59	rusty	10	45.0	40000
60	andy	10	60.0	25000
61	mary	8	25.0	30000

Find the AVG salary for each rating
greater than 7.
Keep only the ones with an MIN Salary >20000.

```
SELECT s.rating, AVG(s.salary)
FROM Sailors3 s
WHERE s.rating > 7
GROUP BY s.rating
HAVING MIN(s.salary)>20000;
```

HAVING MIN(salary)> 20000

rating	MIN(salary)	AVG(salary)
8	30000	30000
~~10~~	~~15000~~	~~25666.66~~

Result:
(Not that it has the attributes
and aggregates from target-list.)

rating	AVG(salary)
8	30000

SQL

- Create, alter, delete tables
- Insert Statement
- Basic Select Statement Structure
- LIKE, AS keywords
- Dates, Text case sensitivity
- Count
- Set Operations
- Aggregates
- Nested Queries

Nested Queries

- An SQL Query can be used to help evaluate another query
- E.g.: a condition might need to be evaluated on a computed relation, not on one readily available
- Multiple levels of nesting are possible
- Semantics are similar to those of nest loops

Connecting queries and sub-queries

- The result of one query can be:
  - A relation
    - E.g.: SELECT s.sname, s.rating FROM Sailors s;
  - A scalar value (1×1 table)
    - In this case, this result can be used in locations where a constant can be placed
    - E.g.: SELECT AVG(s.rating) from Sailors s;
- Where do subqueries appear?
  - Most often in the WHERE clause of a parent query
  - In FROM clause follow by range variable
  - In HAVING clause

Examples Database Instance

sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0

reserves

sid	bid	day
22	101	10/10/22
58	101	10/11/22
22	102	10/20/22

boats

bid	name	color
101	interlake	red
102	clipper	green

Subqueries that return a scalar

- Also referred to as subqueries that return a constant
- *If the subquery returns zero or more values, an error will occur. Very fragile!*
- Example: Select all sailors who have the smallest rating

```
SELECT * from sailors s
WHERE s.rating = (SELECT min(s2.rating)
                  FROM sailors s2);
```

Conditions involving relations

- Test that a relation satisfies some condition:
  - WHERE EXISTS (SELECT …) returns True if the subquery result is not empty
  - WHERE UNIQUE (SELECT …) returns True if the subquery result has no duplicates
    - E.g: Select all sailors for which there is a reservation for boat 101.

```
SELECT *
FROM Sailors s
WHERE EXISTS (SELECT *
                FROM Reserves r
                WHERE r.bid=101 and s.sid=r.sid);
```

  - Subquery is correlated with parent query

Conditions involving relations and tuples

- Typically have some sort of set-operations semantics
  - WHERE field IN (SELECT …)
  - WHERE field op ANY (SELECT …)
  - WHERE field op ALL (SELECT …)
- E.g.: select all sailors who have at least one reservation

```
SELECT * from Sailors s
WHERE s.sid IN ( SELECT r.sid
                 FROM Reserves r);
```

Conditions involving relations and tuples

- Typically have some sort of set-operations semantics
  - WHERE field IN (SELECT …)
  - WHERE field op ANY (SELECT …)
  - WHERE field op ALL (SELECT …)
- E.g.: select all sailors who have no reservation

```
SELECT * from Sailors s
WHERE s.sid NOT IN ( SELECT r.sid
                     FROM Reserves r);
```

Conditions involving relations and tuples

- Typically have some sort of set-operations semantics
  - WHERE field IN (SELECT …)
  - WHERE field op ANY (SELECT …)
  - WHERE field op ALL (SELECT …)
- E.g.: select all sailors whose rating is higher than the rating of any sailor who reserved boat 101

```
SELECT * from Sailors s
WHERE s.rating > ANY ( SELECT s2.rating
                       FROM Sailors s2,Reserves r
                       WHERE s2.sid=r.sid and r.bid=101);
```

Conditions involving relations and tuples

- Typically have some sort of set-operations semantics
  - WHERE field IN (SELECT …)
  - WHERE field op ANY (SELECT …)
  - WHERE field op ALL (SELECT …)
- E.g.: select all sailors with the oldest age

```
SELECT * from Sailors s
WHERE s.age >= ALL ( SELECT s2.age
                     FROM Sailors s2);
```

Same query could be written with aggregates

- E.g.: select all sailors with the oldest age

```
SELECT * from Sailors s
WHERE s.age = (SELECT max(s2.age)
               FROM Sailors s2);
```

Oracle Practice Session

- Connect to SqlPlus
- PracticeSessionSQL3.sql

SQL

- Create, alter, delete tables
- Insert Statement
- Basic Select Statement Structure
- LIKE, AS keywords
- Dates, Text case sensitivity
- Count
- Set Operations
- Aggregates
- Nested Queries
- SQL Division

Examples Database Instance

sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0

reserves

sid	bid	day
22	101	10/10/22
58	101	10/11/22
22	102	10/20/22

boats

bid	name	color
101	interlake	red
102	clipper	green

Recap.: Conditions involving relations

- Test that a relation satisfies some condition:
  - WHERE EXISTS (SELECT …) returns True if the subquery result is not empty
  - WHERE UNIQUE (SELECT …) returns True if the subquery result has no duplicates
    - E.g: Select all sailors for which there is a reservation for boat 101.

```
SELECT *
FROM Sailors s
WHERE EXISTS (SELECT *
                FROM Reserves r
                WHERE r.bid=101 and s.sid=r.sid);
```

  - Subquery is correlated with parent query

Recap.: Conditions involving relations and tuples

- Typically have some sort of set-operations semantics
  - WHERE field IN (SELECT …)
  - WHERE field op ANY (SELECT …)
  - WHERE field op ALL (SELECT …)
- E.g.: select all sailors who have at least one reservation

```
SELECT * from Sailors s
WHERE s.sid IN ( SELECT r.sid
                 FROM Reserves r);
```

Subqueries in the FROM clause

- Find the names of sailors who reserved boat interlake. Return the sailors names and the boat name.

```
SELECT SQ.sname, SQ.bname
FROM (SELECT S.sname, B.name as bname
        FROM Sailors S, Boats B, Reserves R
        WHERE S.sid=R.sid and R.bid=B.bid) SQ
WHERE SQ.bname='interlake';
```

HAVING with Subquery

- Find the age of the youngest sailor older than 18, for each rating with at least two sailors.

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

- NOTE: HAVING clause always executes after WHERE clause

Example
Incorrect solution

- sailors(sid:int, sname: string, rating:int, age:, salary:real)
- boats(bid:int, name:string, color:string, manufacturer:string, prod_year:date)
- reserves (sid:int, bid:int, day:date)
- Find the sailors' ids who reserved both red and green boats
- The solution given below is incorrect !!!

```
SELECT s.sid
FROM sailors s, reserves r, boats b
WHERE s.sid = r.sid AND b.bid =r.bid AND
          b.color='red' AND b.color='green';
```

Rewriting INTERSECT queries using IN

- Find the sailors' ids who reserved both red and green boats

- Correct Solution with INTERSECT:

```
(SELECT r.sid
FROM Reserves r, Boats b
WHERE r.bid=b.bid AND b.color='red')
INTERSECT
(SELECT r.sid
FROM Reserves r, Boats b
WHERE r.bid = b.bid AND b.color='green');
```

Rewriting INTERSECT queries using IN

- Find the sailors' ids who reserved both red and green boats

- Solution without INTERSECT:

```
SELECT DISTINCT r.sid
FROM Reserves r, Boats b
WHERE r.bid=b.bid AND
        b.color='red' AND r.sid IN ( SELECT r2.sid
                                     FROM Reserves r2, Boats b2
                                     WHERE r2.bid=b2.bid AND b2.color='green');
```

Another Example

- sailors(sid:int, sname: string, rating:int, age:, salary:real)
- boats(bid:int, name:string, color:string, manufacturer:string, prod_year:date)
- reserves (sid:int, bid:int, day:date)
- Find the boat ids and names that are booked at least 3 times

```
SELECT b.bid, b.name
FROM boats b
Where 3 <= (select count(*)
                FROM reserves r2
                 Where r2.bid=b.bid);
```

Another Example

- sailors(sid:int, sname: string, rating:int, age:, salary:real)
- boats(bid:int, name:string, color:string, manufacturer:string, prod_year:date)
- reserves (sid:int, bid:int, day:date)
- Find the boat ids and names that are booked by at least 3 unique sailors

```
SELECT b.bid, b.name
FROM boats b
Where 3 <= (select count(DISTINCT r2.sid)
                FROM reserves r2
                 Where r2.bid=b.bid);
```

Nested Queries Summary

- Nested queries returning a constant
  - Typically the constant is compared with other values in the WHERE clause
  - … WHERE field= (SELECT x FROM…)…
- Nested queries returning a relation
  - In WHERE Clause
    - …WHERE (EXISTS/UNIQUE) (SELECT…)…
    - … WHERE field IN (SELECT col FROM ….)…
    - …WHERE field op ANY|ALL (SELECT col FROM…)….
  - In FROM clause, followed by range variable
    - E.g: ..FROM Sailors, (SELECT bid FROM Boats..) BIDS ….
  - In HAVING clause
    - … HAVING field/const op (SELECT ….) ….

Questions?