Database Management Systems
L4

Umass Boston
Summer 2023
Cristina Maier

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

SQL Practice Session

- Connect to Oracle DB using SqlPlus
- Run queries from PracticeSessionSQL1.sql file

Topics

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

Note on Oracle Database

- DatabaseName
  - DatabaseSchemaA
    - table1
    - table2..etc.
  - DatabaseSchemaB
  - DatabaseSchema C …etc
- Students who enrolled to CS630 class on the cs portal and created the <firstName>_<lastName>_<studentIT>_<section> folder according to the instructions from the email received an username to connect to the DB. Each of such student has his/her own database schema. When you create tables, those tables are created in your own schema.

Note on Database Schemas

- Some DBMS allow multiple database schemas to be created inside a database. (E.g. Oracle, PostgreSQL)

- MySQL allows only one database schema to be created for a database

  - databasename

    - Table1

    - Table2 …etc.

Basic structure of SELECT queries

- SELECT [DISTINCT] target-list

FROM relation-list

[WHERE conditions];

- **relation-list** : list of relation names
- **target-list**: list of attributes from relations in relation-list. **It can also contain expressions**.
- **conditions**: condition expressions connected by AND, OR
  - Condition is of forms:
    - **attribute** op **constant**
    - **attributeA** op **attributeB**
    - **op** can be: =, <=, >=, <>, or string operators
  - Later we will discuss more about this when we talk about nested queries
- **DISTINCT**: if present, eliminates duplicates from the result

Note on Case Sensitivity

- Usually in DBMS implementations the names of tables and names columns are not cases sensitive (e.g. Oracle, PostgreSQL)

- The case of text stored in the values from records matters!!!

Sorting the result

- SELECT [DISTINCT] target-list

FROM relation-list

[WHERE conditions] ORDER BY column-list;

- The resulted relation will be sorted by the columns present in the column-list
- If the optional DESC is present after a column, then the sorting by that column is done in descending order. If DESC is not present after a column, the sorting by that column is done in ascending order
- If sorting by multiple columns, DESC has effect only on the column right before it. For example ***ORDER BY col1, col2*** sorts by both column in ascending order. If we want to sort by each column in DESC order then we need to write ***ORDER BY col1 DESC, col2 DESC***
- Column-list might contain column indexes, which start from 1 (indexes calculated based on columns from the resulted relation)
- Best practice is to use the column names. But in some cases column indexes are necessary!!! - we'll see an example on that

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

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

SELECT * FROM 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

SELECT * FROM sailors ORDER BY age;

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

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

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

SELECT * FROM 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

SELECT * FROM sailors ORDER BY rating DESC;

sid	sname	rating	age
58	rusty	10	35
59	rusty	10	45
31	lubber	8	55.0
22	dustin	7	45

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

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

SELECT sname,rating,age FROM sailors;

sname	rating	age
dustin	7	45.0
lubber	8	55.0
rusty	10	35.0
rusty	10	45.0

SELECT sname, rating, age FROM sailors ORDER BY 2 DESC;

sname	rating	age
rusty	10	35.0
rusty	10	45.0
lubber	8	55.0
dustin	7	45.0

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

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

SELECT sname, rating, age FROM sailors;

sname	rating	age
dustin	7	45.0
lubber	8	55.0
rusty	10	35.0
rusty	10	45.0

SELECT sname, rating, age FROM sailors
ORDER BY 2 DESC ,3 DESC;

sname	rating	age
rusty	10	45.0
rusty	10	35.0
lubber	8	55.0
dustin	7	45

SELECT sname, rating, age FROM sailors ORDER BY 2 DESC;

sname	rating	age
rusty	10	35
rusty	10	45
lubber	8	55.0
dustin	7	45

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

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

SELECT * FROM sailors, reserves WHERE sailors.sid=reserves.sid;

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/22
22	dustin	7	45.0	22	102	10/20/22
58	rusty	10	35.0	58	101	10/11/22

Example when the index of the column is needed

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

SELECT * FROM sailors, reserves WHERE sailors.sid=reserves.sid ORDER BY sid;

Will return an error, because column sid is ambiguous !
In this case, after the SORT BY you must specify
the index of column (not the name)!

Example when the index of the column is needed

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

SELECT * FROM sailors, reserves WHERE sailors.sid=reserves.sid ORDER BY 1;

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/22
22	dustin	7	45.0	22	102	10/20/22
58	rusty	10	35.0	58	101	10/11/22

SQL

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

String Operations

- LIKE keyword
- Can be used for string matching
- % stands for 0 or more characters
- _ stands for one character

Examples with LIKE keyword

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

SELECT * FROM sailors WHERE sname LIKE 'dustin';

sid	sname	rating	age
22	dustin	7	45.0

SELECT * FROM sailors WHERE sname='dustin';

sid	sname	rating	age
22	dustin	7	45.0

Examples with LIKE keyword

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

SELECT * FROM sailors WHERE sname LIKE '%st%';

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

SELECT * FROM sailors WHERE sname LIKE '%ty';

sid	sname	rating	age
58	rusty	10	35.0
59	rusty	10	45.0

Examples with LIKE keyword

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	justin	9	25.0

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

SELECT * FROM sailors WHERE sname LIKE '_ustin';

sid	sname	rating	age
22	dustin	7	45.0
60	justin	9	25.0

SELECT * FROM sailors WHERE sname = '_ustin'; ?

SELECT * FROM sailors WHERE sname LIKE '_tin'; ?

AS keyword

- Can be used
  - to rename a field in the result
  - to give a name to expressions from the target-list

Examples

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

SELECT sid, sname, rating-1 AS nrating FROM sailors ORDER BY age;

SELECT sid, sname AS name FROM sailors;

sid	name
22	dustin
31	lubber
58	rusty
59	rusty

sid	sname	nrating
58	rusty	9
22	dustin	6
59	rusty	9
31	lubber	7

select sid, sname, rating-1 AS nrating from sailors order by nrating;

sid	sname	nrating
22	dustin	6
31	lubber	7
58	rusty	9
59	rusty	9

Counting

- COUNT([DISTINCT] column)
- COUNT(*)
  - Used to count the number of records from the result
  - When DISTINCT is used, it removes the duplicates.
  - Note that DISTINCT cannot precede *
  - Later we will discuss how to use COUNT with GROUP BY expressions

COUNT Examples

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	justin	9	25.0

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

SELECT COUNT(*) FROM sailors;

count(*)
5

SELECT COUNT(DISTINCT rating) FROM sailors;

count(distinct ratings)
4

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find all sailors who have a rating greater than 8

```
SELECT * FROM sailors
WHERE rating>8;
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find the names of sailors who reserved a red boat

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find the names of sailors who reserved a 'red' boat

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

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find the how many reservations we have for 'green' boats

```
SELECT count(*)
FROM reserves r, boats b
WHERE r.bid=b.bid and b.color= 'green';
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find the how many unique 'green' boats were ever reserved

```
SELECT count(DISTINCT b.bid)
FROM reserves r, boats b
WHERE r.bid=b.bid AND b.color= 'green';
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)
- Find the names and ages of sailors whose name starts with 'a'.

```
SELECT sname, age
FROM sailors
WHERE sname LIKE 'a%';
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)

- Find the names, ages and ratings of sailors whose name contains string 'ar'. Sort the result by rating in descending order.

SELECT sname, age, rating

FROM sailors

WHERE sname like '%ar%' order by rating desc;

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)

- Find all sailors named 'mary'. The case of the letters should not matter.

```
SELECT * FROM sailors
where upper(sname)=upper('mary');
```

- (another solution)

```
SELECT * FROM sailors
WHERE lower(sname)=lower('mary');
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)

- Find all boats reservations made for days before Oct 11, 2022. Return the id and name of the sailor, the boat name and color and the date of reservation.

```
SELECT s.sid, s.sname, b.name, b.color, r.day
FROM sailors s, reserves r, boats b
   WHERE s.sid = r.sid AND r.bid= b.bid
                AND  r.day<TO_DATE('10/11/2022','mm/dd/yyyy');
```

Given the following database schema

- sailors (sid: int, sname: string, rating:int, age: real)
- boats(bid: int, name:string, color: string)
- reserves(sid: int, bid:int, day: date)

- Find the name of the sailors who are older than 30 and reserved a white or green boat. The returned names should only use uppercase letters.

```
SELECT upper(sname)
FROM sailors s, reserves r, boats b
WHERE s.sid=r.sid AND r.bid=b.bid AND s.age>30 AND
        (b.color='white' or b.color='green');
```

SQL

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

Set Operators

- UNION
- INTERSECT
- MINUS
  - Or EXCEPT (but EXCEPT not in Oracle)
- Note: UNION, INTERSECT, MINUS, EXCEPT eliminate duplicates
- UNION ALL retains the duplicates
  - Some DBMS also support INTERSECT ALL, EXCEPT ALL (Oracle does not!)

Example UNION

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

Sailors2

sid	sname	rating	age
20	joe	9	24.0
22	dustin	7	45.0
31	lubber	8	55.0
60	andy	10	60.0

```
(SELECT * FROM Sailors)
UNION
(SELECT * FROM Sailors2);
```

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0
20	joe	9	24.0
60	andy	10	60.0

Example INTERSECT

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

Sailors2

sid	sname	rating	age
20	joe	9	24.0
22	dustin	7	45.0
31	lubber	8	55.0
60	andy	10	60.0

```
SELECT * FROM Sailors
INTERSECT
SELECT * FROM Sailors2;
```

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0

Example MINUS (Set difference)

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

Sailors2

sid	sname	rating	age
20	joe	9	24.0
22	dustin	7	45.0
31	lubber	8	55.0
60	andy	10	60.0

```
SELECT * FROM Sailors
MINUS
SELECT * FROM Sailors2;
```

sid	sname	rating	age
58	rusty	10	35.0
59	rusty	10	45.0

Example
UNION ALL

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

Sailors2

sid	sname	rating	age
20	joe	9	24.0
22	dustin	7	45.0
31	lubber	8	55.0
60	andy	10	60.0

```
SELECT * FROM Sailors
UNION ALL
SELECT * FROM Sailors2;
```

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0
59	rusty	10	45.0
20	joe	9	24.0
22	dustin	7	45.0
31	lubber	8	55.0
60	andy	10	60.0

Practice Queries Oracle

- Practice with more queries at home:
  - Connect to Oracle DB
  - PracticeSessionSQL2.sql file

Aggregates

- COUNT(*)
- COUNT([DISTINCT] column)
- SUM([DISTINCT] column)
- AVG([DISTINCT] column)
- MIN([DISTINCT] column)
- MAX([DISTINCT] column)

Examples

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

SELECT AVG(rating) FROM sailors;

8.75

SELECT AVG(DISTINCT rating) FROM sailors;

8.33

SELECT max(rating) FROM sailors;

10

SELECT min(age) FROM sailors;

35.0

SELECT COUNT(*) FROM sailors;

4

SELECT COUNT(rating) FROM sailors;

4

SELECT COUNT(DISTINCT rating) FROM sailors;

3

Examples

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

```
SELECT SUM(rating)
FROM sailors
WHERE sname='rusty';
```

20

```
SELECT AVG(rating)
FROM sailors
WHERE age >40;
```

8.33

Important Note!!!!
Common Mistake

- You cannot have both aggregates and non-aggregates in SELECT

- SELECT sname, min(rating) from Sailors;

  - Illegal query! It will return an error

- The only exception is when the GROUP BY clause is present (we will go over this next)

Aggregates and groups

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

What if we want to find the youngest age for each rating?

GROUP BY , HAVING

SELECT [DISTINCT] target-list

FROM relation-list

[WHERE conditions]

GROUP BY group-list

[HAVING] group-qualification;

- *group-list*: a set of attributes (i.e. columns) to group by
- *target-list*: can contain aggregate(s). It can also contain attributes that are present in the group-list
- *group-qualification*: is applied to each group (Note that WHERE is applied to each row). It checks the conditions from group-qualification. If the condition is not satisfied for a group, that group will not be part of the result
    - An attribute in the group-qualification that is not an aggregate operation must appear in grouping-list (unless EVERY or ANY are used (we'll go over this later))

Conceptual evaluation

1. Compute cross-product of relation-list
2. Discard records that fail qualification-list conditions (WHERE clause)
3. Remaining records are partitioned into groups by the value of columns in the group-list
4. Discard groups that fail group-qualification
    - Expressions in the group-qualification must have a single value per group
    - An attribute in the group-qualification that is not an argument of an aggregate operation must appear in grouping-list (unless EVERY or ANY used (we'll go over this later))
5. Result will have a single record per qualifying group

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?

Questions?