ADITYA ENGINEERING COLLEGE (A)

DBMS
Basic SQL - 2

By

Dr. S Rama Sree
Professor in CSE & Dean(Academics)
Aditya Engineering College(A)
Surampalem.

CONTENTS

Queries

Sailors(sid: integer, sname: string, rating: integer, age: real);
Boats(bid: integer, bname: string, color: string);
Reserves(sid: integer, bid: integer, day: date).

Sailors

Sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Ravi	9	40
85	Art	3	25.5
95	Bob	3	63.5

Boats

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

Reserves

sid	bid	day
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-8
22	104	1998-10-7
31	102	1998-11-10
31	103	1998-11-6
31	104	1998-11-12
64	101	1998-9-5
64	102	1998-9-8
74	103	1998-9-8

Queries

1) Find the names of sailors who have reserved boat no 103

Select s.sname from sailors s, reserves r

Where s.sid=r.sid and r.bid=103;

Assume these instances:

sid	*bid*	*day*
22	101	1998-10-10
58	103	1998-11-12

sid	*sname*	*rating*	*age*
22	Dustin	7	45
31	Lubber	8	55.5
58	Rusty	10	35

Queries

The first step is to compute the cross product:

sid	*sname*	*rating*	*age*	*sid*	*bid*	*day*
22	Dustin	7	45	22	101	1998-10-10
22	Dustin	7	45	58	103	1998-11-12
31	Lubber	8	55.5	22	101	1998-10-10
31	Lubber	8	55.5	58	103	1998-11-12
58	Rusty	10	35	22	101	1998-10-10
58	Rusty	10	35	58	103	1998-11-12

Now we apply qualification:

```
S.sid = R.sid AND R.bid = 103
```

sid	*sname*	*rating*	*age*	*sid*	*bid*	*day*
58	Rusty	10	35	58	103	1998-11-12

Finally, we do projection:

sname
Rusty

We can also use

Select sname from Sailors, Reserves where Sailors.sid=Reserves.sid and bid=103;

Queries

2) Find the names of sailors who have reserved a red.

Select R.sid from Boats B, Reserves R where B.bid=R.bid AND B.color=‘red’

3) Find the names and ages of all sailors

Select distinct s.sname, s.age from Sailors S

4) Find all the sailors with rating above 7

Select S.sid ,S.sname, S.rating ,S.age from Sailors S where S.rating>7;

5) Find the colors of boats reserved by Lubber

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

6) Find the ages of sailors whose name begins and ends with B and has at least three characters.

SELECT S.age

FROM Sailors S WHERE S.sname LIKE 'B_%B';

UNION, INTERSECT AND EXCEPT

- SQL provides three set-manipulation constructs that extend the basic query under the names UNION, INTERSECT, and EXCEPT.

7) Find the names of sailors who have reserved a red or 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' OR B.color = 'green')

8) Find the names of sailor's who have reserved both a red and a green boat

SELECT S.sname FROM Sailors S, Reserves R1, Boats BI, 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';

- These 2 queries are difficult to understand and hence we use the UNION, INTERSECT constructs.

Queries

7) Find the names of sailors who have reserved a red or 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'
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';

o/p: Dustin
Lubber
Ravi
horatio

Queries

8) Find the names of sailors who have reserved both a red and 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';

o/p:

Dustin

lubber

Queries

9) Find the sids of sailors who have reserved red boat but not a green boat.

```
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';
```

O/p: 64 {dustin,lubber,horatio}—red {dustin,lubber,ravi}—green

Or in simpler way

```
SELECT R.sid from Boats B, Reserves R where R.bid = B.bid AND B.color = 'red
EXCEPT
SELECT R2.sid from Boats B2, Reserves R2 where R2.bid = B2.bid AND B2.color = 'red
```

Nested Queries

- A nested query is a query that has another query embedded within it; the embedded query is called a subquery. The embedded query can of course be a nested query itself; thus queries that have very deeply nested structures are possible.
- A subquery typically appears within the WHERE clause of a query. Subqueries can sometimes appear in the FROM clause or the HAVING clause.

1) Find the names of sailors who have reserved boat number103.

SELECT s.sname FROM Sailors S

WHERE S.sid IN (SELECT R.sid FROM Reserves R WHERE R.bid = 103)

6) Find the names of sailors who have not reserved boat number 103.

SELECT s.sname FROM Sailors S

WHERE S.sid NOT IN (SELECT R.sid FROM Reserves R WHERE R.bid = 103)

Nested Queries

- Find the names of sailors who have reserved a red boat.

```
SELECT S.sname from Sailors S where
    S.sid IN ( SELECT R.sid FROM Reserves R
            WHERE R. bid IN (SELECT B.bid FROM Boats B
                    WHERE B.color = 'red');
```

- Find the names of sailors who have not reserved a red boat.

```
SELECT S.sname from Sailors S where
    S.sid NOT IN ( SELECT R.sid FROM Reserves R
            WHERE R. bid IN (SELECT B.bid FROM Boats B
                    WHERE B.color = 'red');
```

Correlated Nested Queries

In correlated Nested Queries, the inner subquery could depend on the row currently being examined in the outer query.

1) Find the names of sailors who have reserved boat number 103.

```
SELECT s.sname FROM Sailors S
WHERE EXISTS (SELECT * FROM Reserves R WHERE R.bid = 103
AND R.sid = S.sid)
```

8) Find the names of sailors who have not reserved boat number 103.

```
SELECT s.sname FROM Sailors S
WHERE NOT EXISTS (SELECT * FROM Reserves R WHERE R.bid =
103 AND R.sid = S.sid)
```

Set-Comparison Operators

- We have already seen the set-comparison operators EXISTS, IN, and UNIQUE, along with their negated versions.
- SQL also supports op ANY and op ALL, where op is one of the arithmetic comparison operators {<=, =, <>, >=, >}

9) Find the sailors with the highest rating

SELECT s.sid FROM Sailors S

WHERE S.rating >= ALL (SELECT S2.rating FROM Sailors S2)

10) Find sailors whose rating is better than some sailor called Horatio

SELECT s.sid FROM Sailors S

WHERE S.rating > ANY (SELECT S2.rating FROM Sailors S2 WHERE S2.sname = 'Horatio')

Aggregate Operators

SQL supports five aggregate operations, which can be applied on any column

1. COUNT ([DISTINCT] A): The number of (unique) values in the A column.
2. SUM ([DISTINCT] A): The sum of all (unique) values in the A column.
3. AVG ([DISTINCT] A): The average of all (unique) values in the A column.
4. MAX (A): The maximum value in the A column.
5. MIN (A): The minimum value in the A column

- Find the average age of all sailors.

  SELECT AVG (S.age) FROM Sailors S;

- Find the average age of sailors with a rating of 10.

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

- Find the name and age of the oldest sailor.
- Consider the following attempt to answer this query:

```
SELECT S.sname, MAX (S.age) FROM Sailors S; // wrong statement

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

- Count the number of sailors.

```
SELECT COUNT (*) from Sailors S;
```

- Count the number of different sailor names.

```
SELECT COUNT ( DISTINCT S.sname ) FROM Sailors S;
```

- Find the names of sailors who are older than the oldest sailor with a rating of 10.

```
SELECT S.sname from Sailors S
        where S.age > ( SELECT MAX ( S2.age ) FROM Sailors S2
                        WHERE S2.rating = 10 );
```

(or)

```
SELECT S.sname FROM Sailors S
        WHERE S.age > ALL ( SELECT S2.age from Sailors S2
                            where S2.rating = 10 );
```

The GROUP BY and HAVING Clause

- The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups.
- This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the HAVING clause which is optional.

Syntax:

SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification

- The `GROUP BY` satement is often used with aggregate functions (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) to group the result-set by one or more columns.

- Find the age of the youngest sailor for each rating level

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

- Find the age of the youngest sailor who is eligible to vote (at least 18 years old) for each rating level with atleast 2 sailors

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

ORDER BY

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

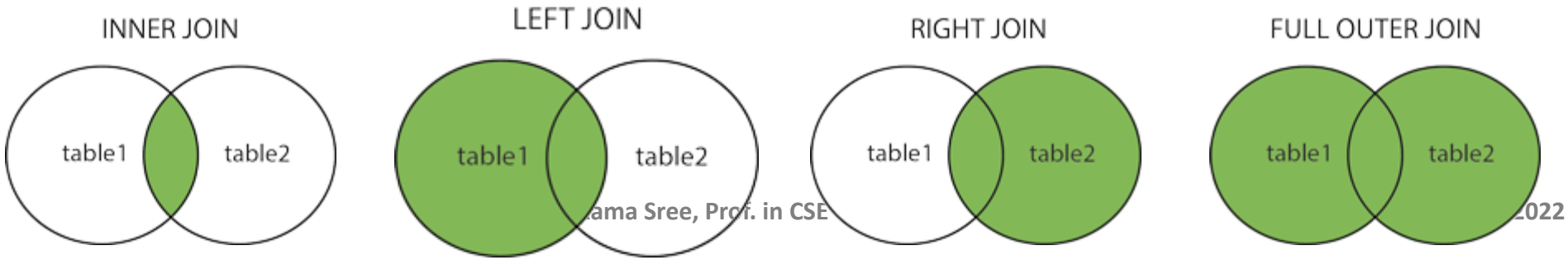
```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- Ex: select * from Students order by name desc;

  select * fom Students order by name;

Joins

- In relational databases, data is spread over multiple tables. Sometimes we may want data from two or more tables.
- A join is an operation which combines results from two or more tables. Joins are of different types, and they are
- **Cartesian Join** : The CARTESIAN JOIN is also known as CROSS JOIN. In a CARTESIAN JOIN there is a join for each row of one table to every row of another table. In the absence of a WHERE condition the CARTESIAN JOIN will behave like a CARTESIAN PRODUCT . i.e., the number of rows in the result-set is the product of the number of rows of the two tables. In the presence of WHERE condition this JOIN will function like a INNER JOIN.
- **Inner Join or Equi Join** : Returns records that have matching values in both tables
- **Full Outer Join** : Returns all records from both the tables
- **Left-outer Join:** Returns all records from the left table, and the matched records from the right table
- **Right-outer Join** : Returns all records from the right table, and the matched records from the left table
- **Self Join :**In SELF JOIN a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.



STUDENTS

SID	NAME	AGE
1	aaa	20
2	bbb	21
3	ccc	21
	ddd	22
5	eee	22

ENROLLED

STUDID	CID
2	101
3	102
	102

- **Cartesian Join** :
  select * from students s ,enrolled e; // 15 rows

- **Self Join:** select * from students s1, students s2; //25 rows

SID	NAME	AGE	STUDID	CID
1	aaa	20	2	101
2	bbb	21	2	101
3	ccc	21	2	101
	ddd	22	2	101
5	eee	22	2	101
1	aaa	20	3	102
2	bbb	21	3	102
3	ccc	21	3	102
	ddd	22	3	102
5	eee	22	3	102
1	aaa	20		102
2	bbb	21		102
3	ccc	21		102
	ddd	22		102
5	eee	22		102

15 rows

- **Inner Join or Equi Join** :

select * from students s, enrolled e where s.sid=e.studid;

SID	NAME	AGE	STUDID	CID
2	bbb	21	2	101
3	ccc	21	3	102

- **Full Outer Join** : select * from students s full outer join enrolled e on s.sid=e.studid;

SID	NAME	AGE	STUDID	CID
1	aaa	20		
2	bbb	21	2	101
3	ccc	21	3	102
	ddd	22		
5	eee	22		
				102

- **Left-outer Join:**

select * from students s left outer join enrolled e on s.sid=e.studid;

SID	NAME	AGE	STUDID	CID
2	bbb	21	2	101
3	ccc	21	3	102
1	aaa	20		
	ddd	22		
5	eee	22		

- **Right-outer Join** : select * from J1 s right outer join J2 e on s.sid=e.studid;

SID	NAME	AGE	STUDID	CID
2	bbb	21	2	101
3	ccc	21	3	102
				102

VIEWS

- A view is a customized representation of data from one or more tables. The tables that the view is referencing are known as base tables.
- A view can be considered as a stored query or a virtual table that contains ros and columns.
- Only the query is stored in the Oracle data dictionary; the actual data is not copied anywhere. This means that creating views does not take any storage space, other than the space in dictionary.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.
- A view can be created using the **CREATE VIEW** statement.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

- A view can be deleted using the Drop View statement.

```
DROP VIEW view_name;
```

- **HORIZONTAL VIEW:** It restricts a user's access to only selected rows of a table.

QUERY:

```
CREATE  VIEW  HOR_V_EMP
AS  SELECT * FROM  EMP
WHERE  EMPNO  IN (7902, 7566, 7654);
```

- **VERTICAL VIEW:** It restricts a user's access to only selected columns of a table.

QUERY:

```
CREATE  VIEW  VER_V_EMP
AS  SELECT EMPNO, ENAME, SAL  FROM  EMP;
```

Updatable View: A view which can be updated. Values can be inserted, updated or deleted.

- Ex: **create view sview as select sid, name, gpa from students; insert into sview values(3,'ccc',8.4);**

SID	NAME	GPA
1	aaa	5.9
2	bbb	7.9
3	ccc	8.4

Delete from sview where sid=2;

SID	NAME	GPA
1	aaa	5.9
3	ccc	8.4

Non-Updateable Views: A view which cannot be updated. Insertions, updations, deletions cannot be done.

- Ex: **create or replace view nusview as select distinct sid,name from stu;**

SQL> select * from nusview;

SID	NAME
1	aa
2	bb
3	cc
4	cc

- **SQL> insert into nusview values(5,'ff');**

Output: ERROR at line 1: ORA-01732: data manipulation operation not legal on this view

- If you use SELECT DISTINCT, Aggregate operators,Group BY, Join in your view definition, it makes your view not updatable.
- The set operators UNION, MINUS, and INTERSECT also result in non-updatable views.

References

Text Books:

- Database Management Systems, 3/e, Raghurama Krishnan, Johannes Gehrke, TMH.
- Database System Concepts,5/e, Silberschatz, Korth, TMH.

Reference Books:

- Introduction to Database Systems, 8/e C J Date, PEA.
- Database Management System, 6/e Ramez Elmasri, Shamkant B. Navathe, PEA
- Database Principles Fundamentals of Design Implementation and Management, Corlos Coronel, Steven Morris, Peter Robb, Cengage Learning.

Web Links:

- https://nptel.ac.in/courses/106/105/106105175/
- https://www.geeksforgeeks.org/introduction-to-nosql/
- https://beginnersbook.com/2015/05/normalization-in-dbms/


Thank
You