SQL
(Simple Query Language)

1

Query Components

- A query can contain the following clauses
  - select
  - from
  - where
  - group by
  - having
  - order by
- Only select and from are required
- Order of clauses is always as above

2

Basic SQL Query

```
SELECT [Distinct] target-list
  FROM relation-list
 WHERE condition;
```

- **relation-list**: A list of relation names (possibly with a range-variable after each name)
- **target-list**: A list of fields onto which the query projects
- **condition**: A Boolean condition
- **DISTINCT**: Optional keyword to delete duplicates

3

Basic SQL Query

```
SELECT [Distinct] target-list
  FROM relation-list
 WHERE condition;
```

- **This is confusing!** The "SELECT" clause defines the projection. Selection is defined by the WHERE clause

4

Basic SQL Query

```
SELECT [Distinct] target-list
  FROM relation-list
 WHERE condition;
```

Evaluation:

1. Compute the cross product of the tables in *from-list*.
2. Delete all rows that do not satisfy *condition*.
3. Delete all columns that do not appear in *target-list*.
4. If **Distinct** is specified eliminate duplicate rows.

5

Basic SQL Query

```
SELECT Distinct A1, ,An
  FROM R1, ,Rm
 WHERE C;
```

$$\pi_{A_1, \ldots, A_n} (\sigma_C (R_1 \times \ldots \times R_m))$$

6

Query Without WHERE

```
SELECT Distinct A1, , An
  FROM R1, , Rm
```

$\pi_{A_1, , A_n} (R_1 \times \times R_m)$

7

Query Without Projection

```
SELECT Distinct *
  FROM R1, , Rm
 WHERE C;
```

$(\sigma_C (R_1 \times \times R_m))$

8

Query Without Projection, Without WHERE

```
SELECT Distinct *
  FROM R1, , Rm
```

$R_1 \times \times R_m$

9

Example Tables Used

Sailors			
sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Boats		
bid	bname	color
101	Nancy	red
103	Gloria	green

Reserves		
sid	bid	day
22	101	10/10/96
58	103	11/12/96

10

What Are You Asking?

```
SELECT DISTINCT sname, age
  FROM Sailors
 WHERE rating>7;
```

- What does this compute?
- Write it in algebra
- When would the result be different if we removed distinct?

11

Sailors Who Reserved Boat 103

```
SELECT DISTINCT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

$\pi_{\text{sname}}(\sigma_{\textit{Sailors.sid = Reserves.sid} \wedge \textit{bid = 103}} (\textit{Sailors x Reserves}))$

12

Sailors x Reserves

Sailors				Reserves		
sid	sname	rating	age	sid	bid	day
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

13

$\sigma_{Sailors.sid = Reserves.sid \wedge bid = 103}$

Sailors				Reserves		
sid	sname	rating	age	sid	bid	day
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

14

π_{sname}

Sailors				Reserves		
sid	sname	rating	age	sid	bid	day
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

15

Range Variables

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

- Range variables are good style.
- They are necessary if the same relation appears twice in the FROM clause
- Similar to Renaming in Relational Algebra

16

What does this return?

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

17

A Few SELECT Options

- Select all columns:

```
SELECT *
  FROM Sailors S;
```

- Rename selected columns:

```
SELECT S.sname AS Sailors_Name
  FROM Sailors S;
```

- Applying functions (e.g., Mathematical manipulations)

```
SELECT (age-5)*2
  FROM Sailors S;
```

18

The WHERE Clause

- Numerical and string comparison:
  != ,<>,=, <, >, >=, <=, between(**between** *val1* **AND** *val2*)
  String comparison is according to the alphabetical order!
- Logical components: AND, OR
- Null verification: IS NULL, IS NOT NULL
- Example:

```
SELECT sname
  FROM Sailors
WHERE age>=40 AND rating IS NOT NULL ;
```

19

The LIKE Operator

- A pattern matching operator
- Basic format: colname LIKE pattern
  - Example:

```
SELECT  sid
  FROM  Sailors
WHERE  sname LIKE R_%y ;
```

_ is a single character

% is 0 or more characters

20

What is this?

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

? Would adding DISTINCT give a different result?

21

Are any of these the same?

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

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

```
SELECT R.sid
  FROM Reserves R
 WHERE R.sid;
```

22

Sailors who've reserved two different boats

23

What does this return?

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

24

Color of Boats Reserved by Bob

25

Order Of the Result

- The ORDER BY clause can be used to sort results by one or more columns
- The default sorting is in ascending order
- Can specify ASC or DESC

26

Example

```
  SELECT sname, rating, age
    FROM Sailors S
   WHERE age > 50
ORDER BY rating ASC, age DESC
```

27

Other Relational Algebra Operators

- So far, we have seen selection, projection and Cartesian product
- How do we do operators UNION and MINUS?

28

Three SET Operators

- [Query] UNION [Query]
- [Query] MINUS [Query]
- [Query] INTERSECT [QUERY]
- Note: The operators remove duplicates by default!
- How would you express intersect in Relational Algebra?

Note that the standard is EXECPT

29

What does this return?

```
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')
```

? What would happen if we replaced *or* by *and*?

30

Sailors who've reserved red *or* green boat

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

What would happen if we wrote MINUS? Or INTERSECT?

31

Sailors who've reserved red *and* green boat

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

32

Multiset (Bag) Operators

- SQL standard includes 3 bag operators:
  - UNION ALL
  - INTERSECT ALL
  - MINUS ALL
- Oracle supports only UNION ALL. Does not remove duplicates when performing UNION

33

Example

```
   SELECT   DISTINCT sname
     FROM   Sailors S
UNION ALL
   SELECT   DISTINCT sname
     FROM   Sailors S
```

34

Nested Queries

35

Nested Queries

Names of sailors who've reserved boat 103:

```
SELECT S.sname
  FROM Sailors S
 WHERE S.sid IN (SELECT R.sid
                   FROM Reserves R
                  WHERE R.bid = 103);
```

The SELECT, FROM and WHERE clauses can have sub-queries. Conceptually, they are computed using nested loops.

What would happen if we wrote NOT IN?

36

Another Example

```
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'))
```

37

Rewrite the Previous Query Using MINUS

38

Correlated Nested Queries

Names of sailors who've reserved boat 103:

```
SELECT S.sid
  FROM Sailors S
 WHERE EXISTS (SELECT *
                 FROM Reserves R
                WHERE R.bid = 103 and
                      S.sid = R.sid);
```

What would happen if we wrote NOT EXISTS?

39

Set-Comparison Queries

Sailors who are not the youngest:

```
SELECT *
  FROM Sailors S1
 WHERE S1.age > ANY (SELECT S2.age
                       FROM Sailors S2);
```

We can also use *op* **ALL** (*op* is >, <, =, >=, <=, or <>).

40