#### SQL (Simple Query Language)

## 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

#### 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

#### 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

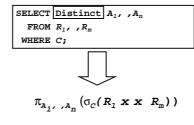
#### Basic SQL Query

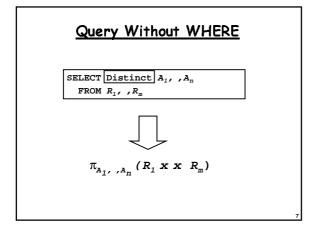
SELECT [Distinct] target-list FROM relation-list WHERE condition;

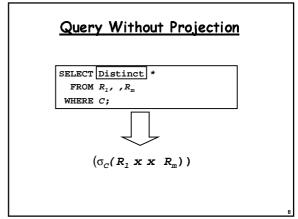
#### Evaluation:

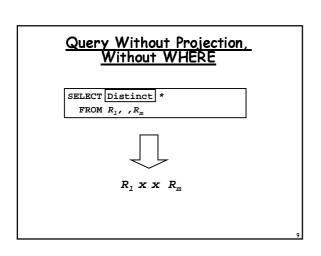
- 1. Compute the cross product of the tables in from-list.
- Delete all rows that do not satisfy *condition*.
   Delete all columns that do not appear in *target-list*.
- 4. If Distinct is specified eliminate duplicate rows.

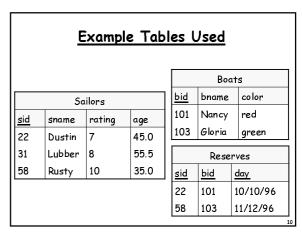
#### Basic SQL Query











## What Are You Asking?

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

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

#### Sailors Who Reserved Boat 103

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

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

#### 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



	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		

#### $\pi_{\text{sname}}$

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	

## 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

#### What does this return?

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

### 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;

3

#### The WHERE Clause

- Numerical and string comparison:
   !=,<>,=,<,>,>=,<=, between(between val1 AND val2)</li>
   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;

### 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

#### What is this?

SELECT S.sid

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

?

Would adding DISTINCT give a different result?

## 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;

Sailors who've reserved two different boats

#### 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

#### 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

## Other Relational Algebra Operators

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

#### Three SET Operators

- · [Query] UNION [Query]
- Note that the standard is EXECPT
- · [Query] MINUS [Query] / is EXECPT
- · [Query] INTERSECT [QUERY]
- Note: The operators remove duplicates by default!
- How would you express intersect in Relational Algebra?

#### 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?

## 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
What would happen if we wrote
MINUS? Or INTERSECT?
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid and R.bid = B.bid
and B.color = green;
```

## 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;
```

#### 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

#### **Example**

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

#### **Nested Queries**

#### **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 subqueries. Conceptually, they are computed using nested loops.

What would happen if we wrote NOT IN?

## 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'))
```

# Rewrite the Previous Query Using MINUS

38

## Correlated Nested Queries

Names of sailors who ve reserved boat 103:

```
What would happen if
we wrote NOT EXISTS?

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

### 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