



Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
 - Compute the cross-product of *relation-list*.
 - Discard resulting tuples if they fail *qualifications*.
 - Delete attributes that are not in *target-list*.
 - If **DISTINCT** is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute *the same answers*.



Simple Example

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

| (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 |



result



A Note on Range Variables

- Really needed only if the same relation appears twice in the FROM clause. The previous query can also be written as:

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

OR

```
SELECT sname  
FROM   Sailors, Reserves  
WHERE  Sailors.sid=Reserves.sid  
       AND bid=103
```

*It is good style,
however, to use
range variables
always!*



Find sailors who've reserved at least one boat

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

- Would adding DISTINCT to this query make a difference?
- What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause? Would adding DISTINCT to this variant of the query make a difference?



Expressions and Strings

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2  
FROM Sailors S  
WHERE S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*
- **AS** and **=** are two ways to name fields in result.
- **LIKE** is used for string matching. **'_'** stands for any one character and **'%'** stands for 0 or more arbitrary characters.