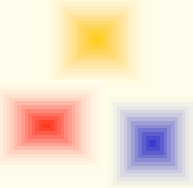# **Expressions and Strings**

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.

Find sid's of sailors who've reserved a red <u>or</u> a green boat

- **UNION**: Can be used to compute the union of any two *union-compatible* sets of tuples (which are themselves the result of SQL queries).

- If we replace OR by AND in the first version, what do we get?

- Also available: EXCEPT (What do we get if we replace UNION by EXCEPT?)

SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
   AND (B.color='red' OR B.color='green')

SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
            AND B.color='red'
UNION
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
            AND B.color='green'

Find sid's of sailors who've reserved a red <u>and</u> a green boat

- INTERSECT: Can be used to compute the intersection of any two *union-compatible* sets of tuples.

- Included in the SQL/92 standard, but some systems don't support it.

- Contrast symmetry of the UNION and INTERSECT queries with how much the other versions differ.

SELECT S.sid
FROM Sailors S, Boats B1, Reserves R1,
          Boats B2, Reserves R2
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')

SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
          AND B.color='red'
INTERSECT
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
          AND B.color='green'
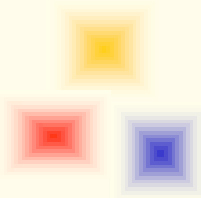
Nested Queries

Find 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)

- A very powerful feature of SQL: a WHERE clause can itself contain an SQL query! (Actually, so can FROM and HAVING clauses.)
- To find sailors who've *not* reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a _nested loops_ evaluation: *For each Sailors tuple, check the qualification by computing the subquery.*