ICS 321 Fall 2010
The Database Language SQL (ii)

Asst. Prof. Lipyeow Lim

Information & Computer Science Department

University of Hawaii at Manoa

UNION, INTERSECT & EXCEPT

- Set-manipulation constructs for result sets of SQL queries that are *union-compatible*
- Can simplify some complicated SQL queries
- Consider Q5: Find the names of sailors who have reserved a red or a green boat

SELECT S1.sname
FROM   Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid
           AND R1.bid=B1.bid
           AND (B1.color=`red' OR B1.color=`green')

Q6: Find the names of sailors who have reserved both a red and a green boat

SELECT S1.sname
FROM   Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid
          AND R1.bid=B1.bid
          AND (B1.color=`red'
                  ~~OR~~ AND B1.color=`green')

SELECT S1.sname
FROM   Sailors S1, Reserves R1, Boats B1,
          Reserves R2, Boats B2
WHERE S1.sid=R1.sid AND R1.bid=B1.bid
          AND S1.sid=R2.sid AND R2.bid=B2.bid
          AND B1.color=`red' AND B2.color=`green'

Q6 with INTERSECT : Find the names of sailors who have reserved both a red and a green boat

SELECT S1.sname
FROM   Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid AND R1.bid=B1.bid
              AND B1.color=`red'

INTERSECT

SELECT S2.sname
FROM   Sailors S2, Reserves R2, Boats B2
WHERE S2.sid=R2.sid AND R2.bid=B2.bid
              AND B2.color=`green'

Q6 Nested: Find the names of sailors who have reserved both a red and a green boat

SELECT S3.sname
FROM   Sailors S3
WHERE S3.sid **IN** (
         SELECT S1.sid
         FROM   Sailors S1, Reserves R1, Boats B1
         WHERE S1.sid=R1.sid AND R1.bid=B1.bid
                        AND B1.color=`red'
         INTERSECT
         SELECT S2.sid
         FROM   Sailors S2, Reserves R2, Boats B2
         WHERE S2.sid=R2.sid AND R2.bid=B2.bid
                        AND B2.color=`green')

Q5 with UNION : Find the names of sailors who have reserved a red or a green boat

SELECT S1.sname
FROM   Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid AND R1.bid=B1.bid
            AND B1.color=`red'

UNION

SELECT S2.sname
FROM   Sailors S2, Reserves R2, Boats B2
WHERE S2.sid=R2.sid AND R2.bid=B2.bid
            AND B2.color=`green'

Q19: Find the sids of sailors who have reserved red boats but not green boats

SELECT S1.sid
FROM   Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid AND R1.bid=B1.bid
          AND B1.color=`red'

EXCEPT

SELECT S2.sid
FROM   Sailors S2, Reserves R2, Boats B2
WHERE S2.sid=R2.sid AND R2.bid=B2.bid
          AND B2.color=`green'

Find the sid of sailors who have reserved exactly one boat

SELECT S1.sid
FROM   Sailors S1
EXCEPT
SELECT R1.sid
FROM   Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid **AND** R1.bid=B1.bid
      AND R2.bid=B2.bid **AND** R1.bid≠R2.bid

SELECT R3.sid
FROM   Reserves R3
EXCEPT
SELECT R1.sid
FROM   Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid **AND** R1.bid=B1.bid
      AND R2.bid=B2.bid **AND** R1.bid≠R2.bid

Nested Queries

Q1 : Find the names of sailors who have reserved boat 103

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

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

- A *nested query* is a query that has another query, called a *subquery,* embedded within it.
- Subqueries can appear in WHERE, FROM, HAVING clauses

Conceptual Evaluation Strategy for Nested Queries

1. Compute the cross-product of *relation-list*.
   - ❑ If there is a subquery, recursively (re-)compute the subquery using this conceptual evaluation strategy
   - ❑ Compute the cross-product over the results of the subquery.
2. Discard resulting tuples if they fail *qualifications*.
   - ❑ If there is a subquery, recursively (re-)compute the subquery using this conceptual evaluation strategy
   - ❑ Evaluate the qualification condition that depends on the subquery
3. Delete attributes that are not in *target-list*.
4. If DISTINCT is specified, eliminate duplicate rows.

Q2: 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'))

- Unravel the nesting from the innermost subquery

Q21: 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

Q1: Find the names of sailors who've reserved boat #103

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

- EXISTS is another set comparison operator, like *IN*.

- If UNIQUE is used, and * is replaced by R.bid, finds sailors with at most one reservation for boat #103. (UNIQUE checks for duplicate tuples; * denotes all attributes. Why do we have to replace * by R.bid?)

- Illustrates why, in general, subquery must be re-computed for each Sailors tuple.

Set Comparison Operators: ANY

- Q22: Find sailors whose rating is better than some sailor called Horatio.

SELECT S1.sid
FROM   Sailors S1
WHERE S1.rating > **ANY** (**SELECT** S2.rating
                            FROM Sailors S2
                            WHERE S2.name=`Horatio')

- Subquery must return a row that makes the comparison true, in order for S1.rating>ANY to return true

Set Comparison Operators: ALL

- Q23: Find sailors whose rating is better than every sailor.

SELECT S1.sid
FROM   Sailors S1
WHERE S1.rating > **ALL** (**SELECT** S2.rating
                                         FROM Sailors S2
                                         WHERE S2.name=`Horatio')

- Subquery must return a row that makes the comparison true, in order for S1.rating>ANY to return true

Rewriting INTERSECT Queries using IN

- Q6: Find sid's of sailors who've reserved both a red and a green boat.

SELECT S1.sid
FROM Sailors S1, Boats B1, Reserves R1
WHERE S1.sid=R1.sid **AND** R1.bid=B1.bid
    AND B1.color='red'
    AND S1.sid **IN** (**SELECT** S2.sid
                    FROM Sailors S2, Boats B2,
                        Reserves R2
                    WHERE S2.sid=R2.sid
                        AND R2.bid=B2.bid
                        AND B2.color=`green')

Q9: Find the names of sailors who have reserved all boats

SELECT S.sname
FROM   Sailors S
WHERE **NOT EXISTS** ((**SELECT** B.bid
                                    FROM Boats B)

                     EXCEPT

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

Q9: Find the names of sailors who have reserved all boats (without EXCEPT)

SELECT S.sname
FROM   Sailors S
WHERE NOT EXISTS ((**SELECT** B.bid
                               FROM Boats B)
                        WHERE NOT EXISTS
                                (**SELECT** R.bid
                                   FROM Reserves R
                                   WHERE R.bid=B.bid
                                       AND R.sid=S.sid))