SQL
Null Values, Join, and Sorting

R &G - Chapter 5

Exercise from Tutorial Session (Cont.)

- **Schema**
  - Student (snum, sname, major, level, age)
  - Class (cname, room, fid)
  - Enrolled (snum, cname)
  - Faculty (fid, fname, deptid)

Q5: Find the name of faculty members *who teach in every room in which* some class is taught

```
SELECT F.fname
FROM Faculty F
WHERE NOT EXISTS (
        ( SELECT C.room FROM Class C )
        EXCEPT
        (SELECT C1.room FROM Class C1
        WHERE C1.fid = F.fid )
        )
```

IS THIS SOLUTION CORRECT?

Q5: Possible Scenarios

- SELECT C.room FROM Class C returns A= {R1, R1, R2}
- For faculty F1, SELECT C1.room FROM Class C1 WHERE C1.fid = F.fid returns B={R1, R2},
- A-B = {R1}
- The query will not return F1 as part of the answer, which is WRONG!

- **Lesson 1**: be careful of duplicate elimination, especially for set operations.

- How to fix the problem?

Q5: Correlations between Main and Sub-queries

- Is this solution correct (note: no correlation between main and subquery)?

Solution NC:

```
SELECT F.fname
FROM Faculty F
WHERE NOT EXISTS (
        ( SELECT DISTINCT C.room FROM Class C )
        EXCEPT
        (SELECT DISTINCT C1.room
         FROM Class C1, Faculty F1
        WHERE C1.fid = F1.fid )
        )
```

Q5: Correlations between Main and Sub-queries

```
Solution NC:
SELECT F.fname
FROM Faculty F
WHERE NOT EXISTS (
        ( SELECT DISTINCT C.room
        FROM Class C )
        EXCEPT
        (SELECT DISTINCT C1.room
         FROM Class C1, Faculty F1
        WHERE C1.fid = F1.fid )
        )
```

Lesson 2: Always keep correlations between main&sub query.

- All faculty records are checked in the same way by the subquery
  - if each Class record has a non-empty fid (i.e., subquery returns empty set), all faculty names will be output;
  - otherwise, zero faculty names will be output)

Today's Topics

- Null values
- Joins
- Sorting

Null Values

- Field values in a tuple are sometimes *unknown* or *inapplicable*.
  - SQL provides a special value *null* for such situations.
- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not *null*.
  - Is *rating>8* true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
- We need a 3-valued logic (true, false and *unknown*).

Impact of Null Values on SQL

- Definition of duplicates
  - Two values are equal if they are the same or they are both Null
  - In MySQL, Null values are present in SELECT DISTINCT result.
- WHERE clause eliminates rows that are evaluated on NULL values
- COUNT(*) return all values including NULL
- COUNT, SUM, AVG, MIN, MAX on specific columns discard NULL
  - Special case: if all values are null, then these operators return NULL

Today's Topics

- Null values
- Joins
- Sorting

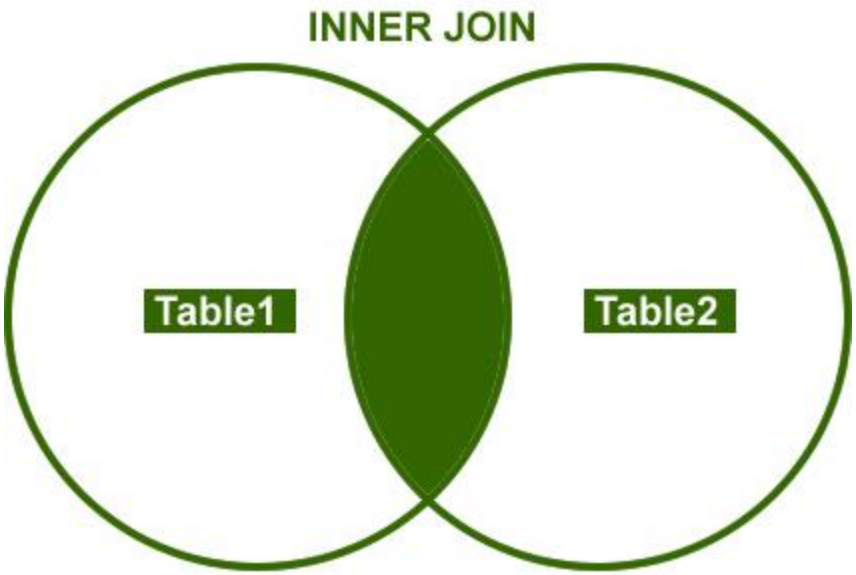
Inner/Outer Joins

```
SELECT (column_list)
FROM  table_name
  [INNER | {LEFT |RIGHT | FULL } OUTER] JOIN table_name
    ON qualification_list
WHERE …
```

Explicit join semantics needed unless it is an INNER join (INNER is default)

Inner Join

- Only the rows that match the search conditions are returned.
- Same as equal-join



Inner Join Example

```
SELECT s.sid, s.name, r.bid
FROM Sailors s INNER JOIN Reserves r
ON s.sid = r.sid
```

- Returns only those sailors who have reserved boats

```
SELECT s.sid, s.name, r.bid
FROM Sailors s INNER JOIN Reserves r
ON s.sid = r.sid
```

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/96
95	103	11/12/96

s.sid	s.name	r.bid
22	Dustin	101
95	Bob	103

Inner Join VS. Natural Join

- The difference is the number of columns that are to be returned.

R1

Sid	Bid	day
22	101	10/10/96
58	103	11/12/96

S1

Sid	Sname	Rating	Age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.5

Natural Join

Sid	Sname	Rating	Age	Bid	day
22	Dustin	7	45.0	101	10/10/96
58	Rusty	10	35.0	103	11/12/96

Inner Join on R1.sid = S1.sid

R1.Sid	Sname	Rating	Age	Bid	day	S1.sid
22	Dustin	7	45.0	101	10/10/96	22
58	Rusty	10	35.0	103	11/12/96	58

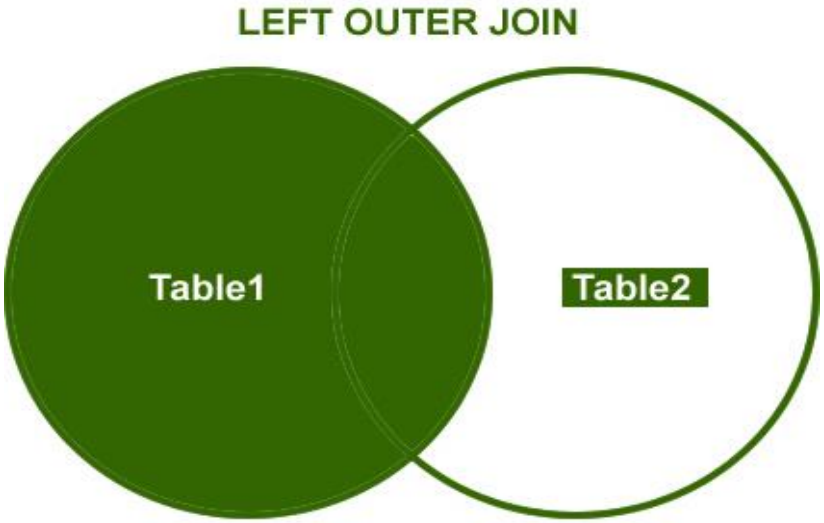
Left Outer Join

Left Outer Join returns:

all matched rows (same as inner join) **+**

All unmatched rows from the table on the left of the join clause

(use nulls in fields of non-matching tuples)



Left Outer Join Example

```
SELECT s.sid, s.name, r.bid
FROM Sailors s LEFT OUTER JOIN Reserves r
ON s.sid = r.sid
```

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/96
95	103	11/12/96

s.sid	s.name	r.bid
22	Dustin	101
95	Bob	103
31	Lubber	

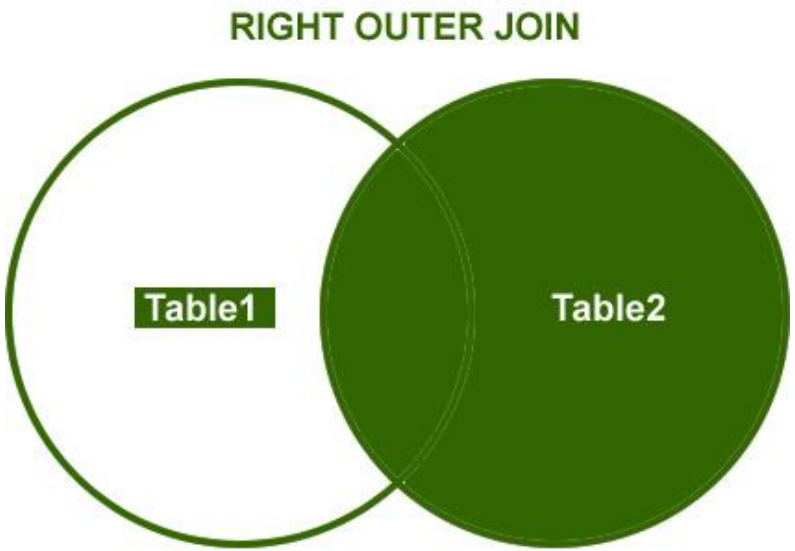
Returns all sailors & information on whether they have reserved boats

Right Outer Join

Right Outer Join returns:

All matched rows (same as inner join) **+**

All unmatched rows from the table on the right of the join clause (use nulls in fields of non-matching tuples)



Right Outer Join Example

SELECT r.sid, b.bid, b.name

FROM Reserves r RIGHT OUTER JOIN Boats b

ON r.bid = b.bid

sid	bid	day
22	101	10/10/96
95	103	11/12/96

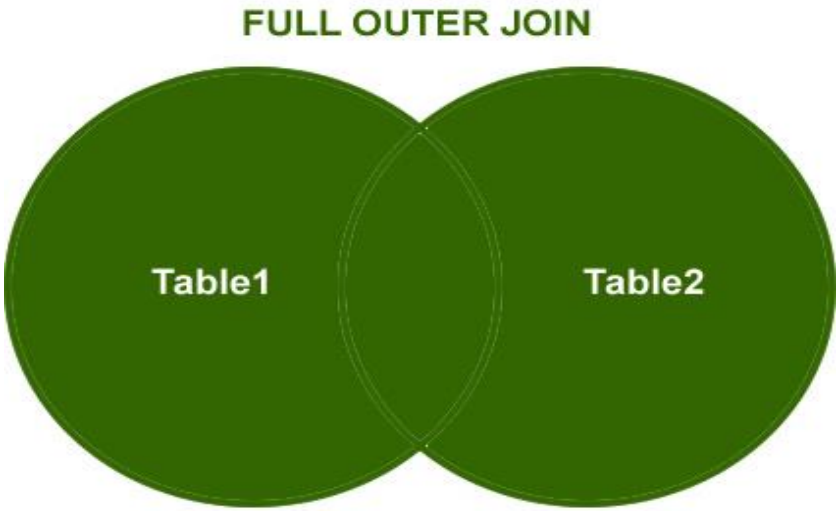
bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

Returns all boats & information on which ones are reserved.

Full Outer Join

Full Outer Join returns all (matched or unmatched) rows from the tables on both sides of the join clause



Full Outer Join Example

SELECT r.sid, b.bid, b.name

FROM Reserves r FULL OUTER JOIN Boats b

ON r.bid = b.bid

sid	bid	day
22	101	10/10/96
95	103	11/12/96

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

Why is full outer join the same as the right outer join for this example?

Sorting the Results of a Query

- ORDER BY *column* [ASC | DESC] (By default: Ascending order)

```
SELECT  S.rating, S.sname, S.age
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
ORDER BY  S.rating, S.sname;
```

(sorted by *rating* in ascending order, with a secondary sort by sname in ascending order.)

Sorting the Results of a Query (Cont.)

- Can order by any attribute in SELECT clause, including expressions or aggregation:

```
SELECT  S.sid, COUNT (*) AS redrescnt
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
GROUP BY S.sid
ORDER BY  redrescnt DESC;
```

Sorting the Results of a Query (Cont.)

- Different orders on different attributes (columns)

```
SELECT  S.rating, S.sname, S.age
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
ORDER BY  S.rating ASC, S.sname DESC;
```

(sorted by *rating* in ascending order, with a secondary sort by *sname* in descending order.)

Online Demo

http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_orderby