Relational Database Management Systems for Epidemiologists:

SQL Part III

Outline

- Subqueries
  - SELECT, FROM, WHERE or HAVING clause of another SELECT statement

Subquery

- A subquery is a SELECT statement embedded in another SQL statement.
- In general, you can use a subquery anywhere an expression is allowed, but your DBMS may place specific restrictions on where subqueries appear.
  - NOTE: MySQL 4.0 and earlier versions do NOT support subqueries.

Qualifying Column Names

- RECALL: Column names must be unique within a table but can be reused in more than one table.
- To identify a column in a query that involves multiple tables, use its qualified name.
- A qualified name is a table name followed by a dot (“.”) and the name of the column in the table.
- Example: the column called “ptid” in the “patients” table is referred as “*patients.ptid*”.

Qualifying Column Name Example 1

```
SELECT patients.ptid, fname, lname,
    FROM patients, submitter
    WHERE patients.ptid=submitter.ptid;
```

Qualifying Column Name Example 2

```
SELECT patients.ptid, fname, lname,
    FROM patients p, submitter s
    WHERE p.ptid=s.ptid;
```

- Alias is a single, unquoted word that contains only letters, digits, or underscores; don't use spaces, punctuation, or special characters.

Comments on JOINS

- Reading in Chapter 7 outlines major comments regarding JOINS (read it carefully).
- Two ways to join tables:
  - Use a JOIN statement
  - Use a WHERE clause, specifying the condition of the join.

Equivalent Example 1

```
SELECT patients.ptid, fname, lname,
    FROM patients
    INNER JOIN submitter
    ON patients.ptid=submitter.ptid;
```

Inner Join

- An inner join:
  - Uses a comparison operator (=, <>, <, <=, >, >=) to match rows from different tables based on values in common columns.
  - Returns only joined rows that satisfy the join condition (or conditions); that is, only rows that match.

Two Tables

```
SELECT columns
   FROM table1
   INNER JOIN table2
      ON join_condition(s);
```

Three Tables

```
SELECT columns
   FROM table1
   INNER JOIN table2
      ON join_condition1
   INNER JOIN table3
      ON join_condition2;
```

Inner Join Syntax

- In general

SELECT *columns*

FROM *table1*

INNER JOIN *table2*

ON *table1.column1 op table2.column2*

INNER JOIN *table3*

ON table2.column2 op table3.column3;

- MS Access

SELECT *columns*

FROM *table1*

INNER JOIN (*table2*

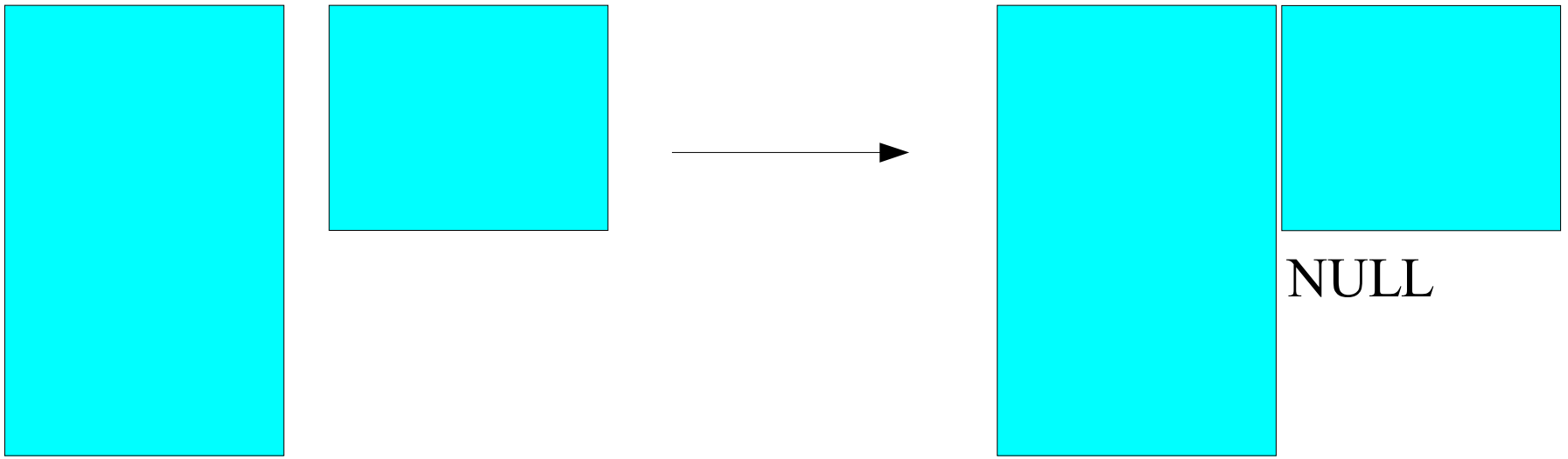
INNER JOIN *table3*

ON *table2.column2 op table3.column3)*

ON *table1.column1 op table2.column2;*

Left Outer Join

- The result of a left outer join includes all rows from the left table specified in the LEFT OUTER JOIN clause, not just the rows in which the joined columns match.

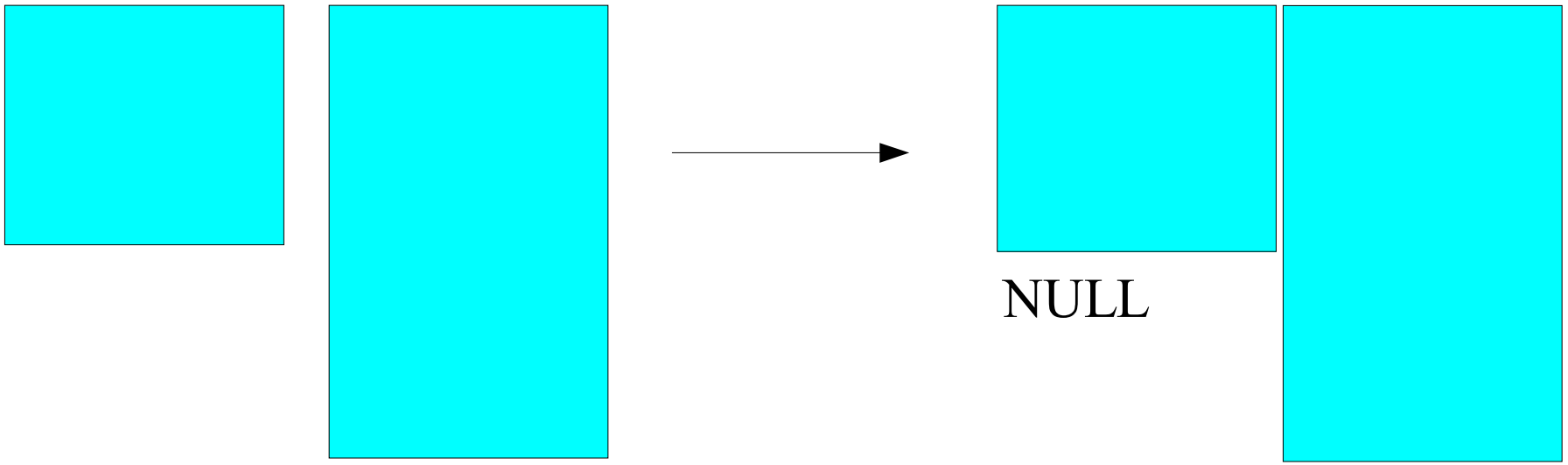


LEFT OUTER JOIN Syntax

```
SELECT patients.ptid,
       submitter.fname,
       submitter.lname
    FROM patients
    LEFT OUTER JOIN submitter
    ON patients.ptid=submitter.ptid;
```

Right Outer Join

- The result of a right outer join includes all rows from the right table specified in the RIGHT OUTER JOIN clause, not just the rows in which the joined columns match.



RIGHT OUTER JOIN Syntax

```
SELECT patients.ptid,
       submitter.fname,
       submitter.lname
    FROM submitter
    RIGHT OUTER JOIN patients
    ON patients.ptid=submitter.ptid;
```

Full Outer Join

- A full outer join is a combination of left and right outer joins.
- It returns all rows in both the left and right tables.
- NOTE: full outer joins are *not* supported by MS Access or MySQL!
  - Use UNION ALL command to complete a full outer join.

UNION Syntax

```
SELECT column1, column2
    FROM table1
    LEFT OUTER JOIN table2
        ON table1.column1=table2.column2
UNION ALL
SELECT column1, column2
    FROM table1
    RIGHT OUTER JOIN table2
        ON table1.column1=table2.column2
    WHERE table1.column1 IS NULL;
```

Cross Join

- Cross joins return all possible combinations of rows of two tables; each row from the first table is combined with all rows from the second table.
- To do a cross join:
  - Omit the ON clause if you're using JOIN syntax
  - Omit the WHERE clause if you're using WHERE syntax.

Natural Join

- A natural join compares all the columns in one table with corresponding columns that have the same name in the other table for equality.
- The same can be achieved with ON clause in JOIN syntax (or a WHERE clause in WHERE syntax).
- I would suggest being explicit about your joins.

Self-Join

- You can join a table to itself by using two different aliases for the same table (see chapter).
- However, you need to have a reflexive relationship, ie, primary key/foreign key relationship from a column or combination of columns in a table to other columns in that same table.
- Self-joins are rarely used.

Execution Sequence of a Query

- The DBMS uses a logical sequence to execute join:

  (1) Applies the join condition in the JOIN clause.

  (2) Applies the join conditions and search conditions in the WHERE clause.

  (3) Groups rows according tot he GROUP BY clause.

  (4) Applies the search conditions in the HAVING clause to the groups.

  (5) Sorts the result according to the ORDER BY clause.