CSE460/560 DATA MODELS AND QUERY LANGUAGES

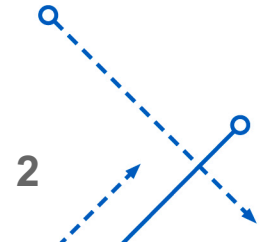
Structured Query Language (SQL)

Cheng-En Chuang
(Slides Adopted from Jan Chomicki and Ning Deng)

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

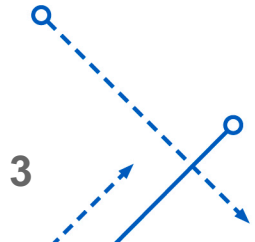
Outline

1. Join
2. Nested Queries
3. Aggregation



Outline

1. **Join**
2. Nested Queries
3. Aggregation



Joins

- R_1 [INNER] JOIN R_2 ON C
  - Where C is a join condition: theta-join
- R_1 NATURAL JOIN R_2
  - Natural join
- Outer Joins
  - Keep tuples in the result even no matching tuples
  - Types
    - R_1 LEFT [OUTER] JOIN R_2 ON C
    - R_1 RIGHT [OUTER] JOIN R_2 ON C
    - R_1 FULL [OUTER] JOIN R_2 ON C

Joins

- R_1 [INNER] JOIN R_2 ON C
- Outer Joins
  - R_1 LEFT [OUTER] JOIN R_2 ON C
  - R_1 RIGHT [OUTER] JOIN R_2 ON C
  - R_1 FULL [OUTER] JOIN R_2 ON C

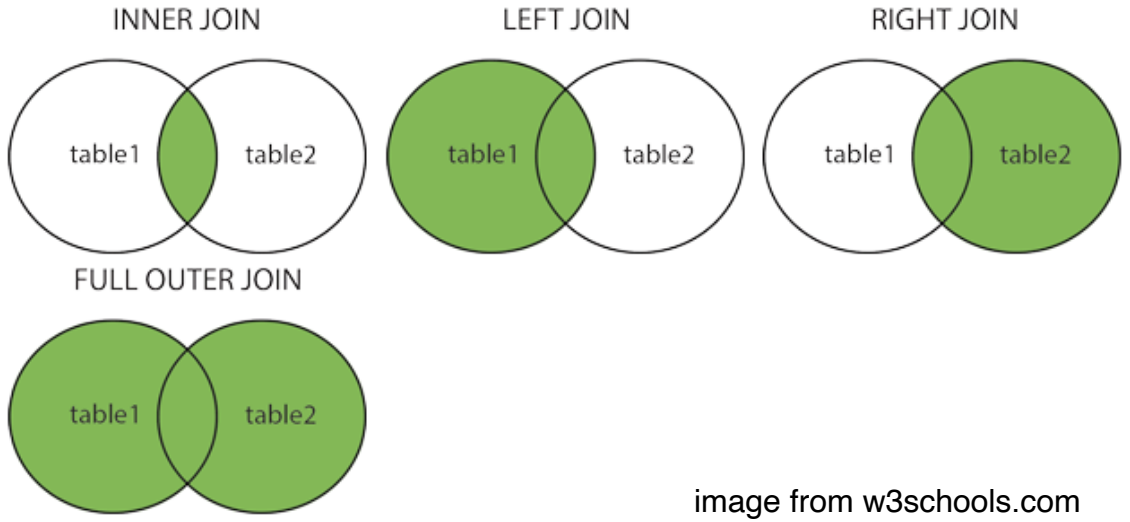


image from w3schools.com

Joins

R1

A	B
a1	b1
a2	b2

R2

B	C
b1	c1
b3	c3

Inner Join

A	R1.B	R2.B	C
a1	b1	b1	c1

Left Join

A	R1.B	R2.B	C
a1	b1	b1	c1
a2	b2	null	null

Right Join

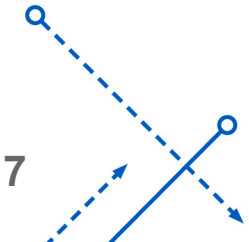
A	R1.B	R2.B	C
a1	b1	b1	c1
null	null	b3	c3

Full Join

A	R1.B	R2.B	C
a1	b1	b1	c1
null	null	b3	c3
a2	b2	null	null

Outline

1. Join
2. **Nested Queries**
3. Aggregation



Nested Queries – WHERE-Nesting

- Subquery
  - A query Q can appear as a **subquery** in WHERE clause
    - a IN Q: for set membership $a \in Q$
    - a NOT IN Q: for the negation of set membership $a \notin Q$
    - a θ ANY Q: a is in the relationship θ to some elements of Q
      - $\Theta \in \{=, <, <=, >=, >, <>\}$
    - a θ ALL Q: a is in the relationship θ to all elements of Q
    - EXISTS Q : Q is nonempty
    - NOT EXISTS Q : Q is empty
- Note
  - Subqueries can contain columns from enclosing queries
  - Multiple occurrences of the same column name are disambiguated
    - By choosing the closest enclosing FROM clause

Nested Queries – IN and NOT IN

- Find students who is not a president of any club
  - SELECT S.sid
    FROM Student S
    WHERE S.Firstname IN (
        SELECT C.President
        FROM Club C);

Nested Queries – ANY and ALL

- Students with the highest GPA
  - ```
    SELECT *
    FROM Student S
    WHERE S.GPA >= ALL (
            SELECT S2.GPA
            FROM Student S2
            WHERE S.sid <> S2.Sid
    );
    ```
    ```

Nested Queries – EXISTS and NOT EXISTS

- Students who is a president of one of clubs
  - EXISTS is true if the nested query returns at least one result
  - SELECT S.Firstname
    FROM Student S
    WHERE EXISTS (
        SELECT *
        FROM Club C
        WHERE S.Firstname = C.President);

Nested Queries – FROM-Nesting

- A subquery can also be constructed in the FROM clause
  - SELECT *
    FROM Student S,
    (SELECT C.President
    FROM Club C
    WHERE C.Name='C2') C2President
    WHERE S.Firstname=C2President.President;

Outline

1. Join
2. Nested Queries
3. **Aggregation**

Aggregation

- Instead of return all tuples, the result can be aggregated, respect to some columns
  - COUNT(A): the number of all values in the column A
  - SUM(A): the sum of all values in the column A
  - AVG(A) the average of all values in the column A
  - MAX(A): the maximum value in the column A
  - MIN(A): the minimum value in the column A
- Note
  - DISTINCT A: consider only distinct values of A
  - COUNT(*): counting tuples
  - Aggregate functions are NOT expressible in relational algebra

Aggregate – Counting Tuples

- Counting
  - COUNT(A) and COUNT(*)
  - Returns 0 if it is an empty relation
- Other aggregate functions
  - The result is null (even for SUM)

Aggregate – Grouping

- GROUP BY
  - GROUP BY clause
    - $GROUP\ BY\ A_1, \ldots, A_n$
  - Assembles the tuples in the result of the query into groups
    - with identical values in the columns $A_1, \ldots, A_n$
  - Example
    - SELECT E.ClassID, AVG(E.Grade)
      FROM Enrollment E
      GROUP BY E.ClassId;
- Note: The SELECT list of a query with GROUP BY can contain only
  - The columns mentioned in GROUP BY
    - E.ClassId
  - The result of an aggregate function
    - AVG(E.Grade)

Having

- We can't apply the condition in WHERE clause to the result of GROUP BY
  - Need HAVING
- The clause
  - HAVING C
- Keeps only those groups that satisfy the condition C
- Example
  - SELECT E.ClassId, AVG(E.Grade)
    FROM Enrollment E
    GROUP BY E.ClassId
    HAVING AVG(E.Grade) >3.3

Recommended Reading

Database Systems: The Complete Book
Chapter 6.1 – 6.5