Database Design and Development

DATABASE DEVELOPMENT – SQL DML – JOINS

Specifying Joined Tables in the FROM Clause of SQL

Joined table

- It was incorporated into SQL to permit users to specify a table resulting from a join operation in the FROM clause of a query.
- JOIN may also be called INNER JOIN

fname	Lname	address
John	Smith	731 Fondren, Houston,TX
Franklin	Wong	638 Voss, Houston,TX
Joyce	English	5631 Rice, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX

Join Example

Retrieve the name and address of all employees who work for the ‘Research’ department.

```
SELECT      employee.fname,employee.lname,
            employee.address
FROM        employee
JOIN        department
            ON department.number =
               employee.dept_number
WHERE       department.name = 'Research';
```

Multiway JOIN in the FROM clause

Can nest JOIN specifications for a multiway join:

For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date

```
SELECT    project.number, project.dept_number,
          employee.lname, employee.address,
          employee.birthdate
FROM      project
JOIN      department ON project.dept_number = department.number
JOIN      employee ON department.manager_SSN = employee.SSN
WHERE     project.location='Stafford';
```

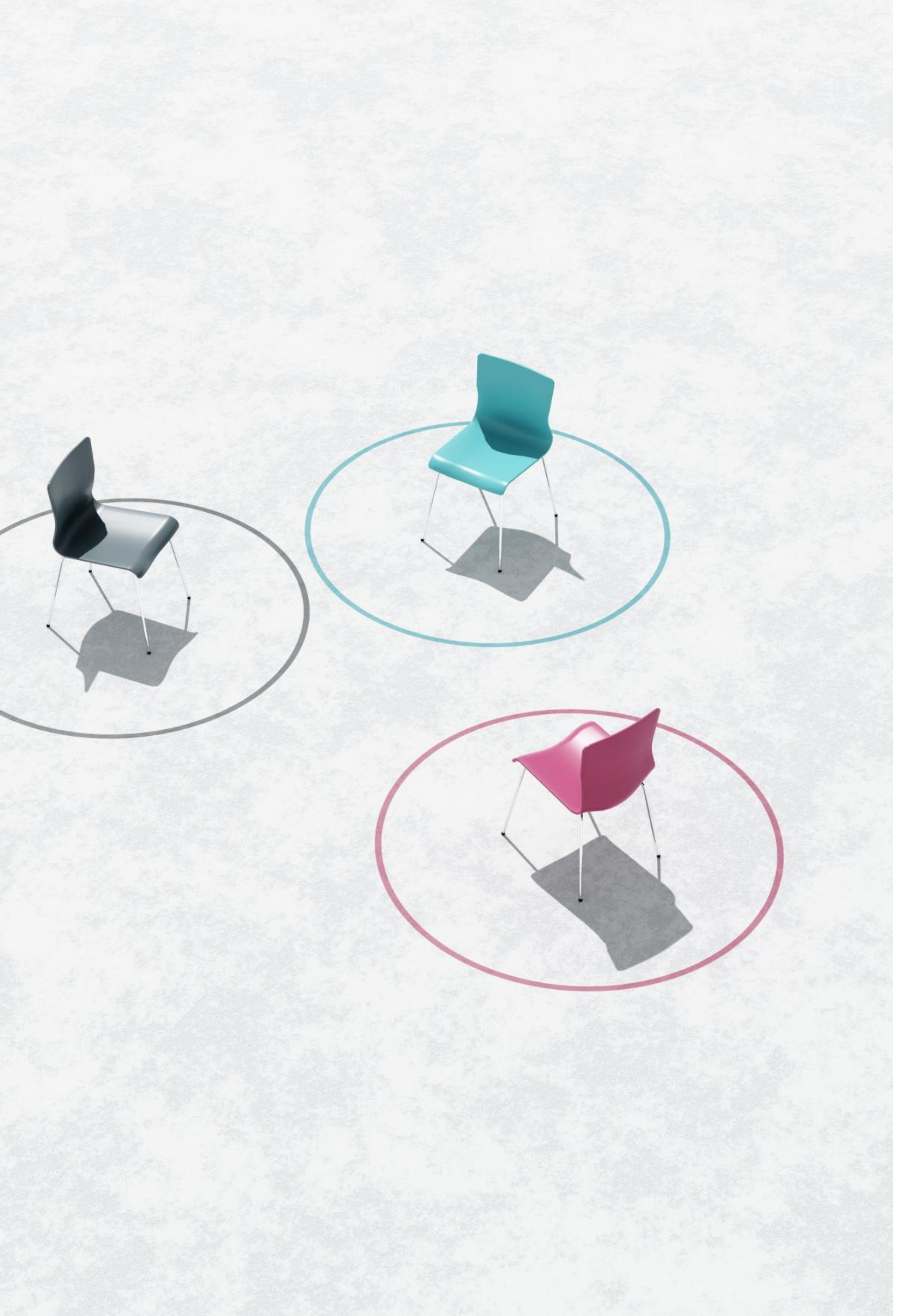
Ambiguous Attribute Names

Same name can be used for two (or more) attributes in different relations

- As long as the attributes are in different relations
- Must **qualify** the attribute name with the relation's name to prevent ambiguity
- Examples:
  - dept_number attribute in project and employee relations
  - number and name attributes in project and department relations

For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
SELECT    project.number, project.dept_number,
          employee.lname,  employee.address,
          employee.birthdate

FROM      project

JOIN      department

          ON project.dept_number = department.number

JOIN      employee

          ON department.manager_SSN = employee.SSN

WHERE     project.location = 'Stafford'
```



INNER and OUTER Joins

INNER JOIN (versus OUTER JOIN)

- Default type of join in a joined table
- Tuple is included in the result only if a matching tuple exists in the other relation

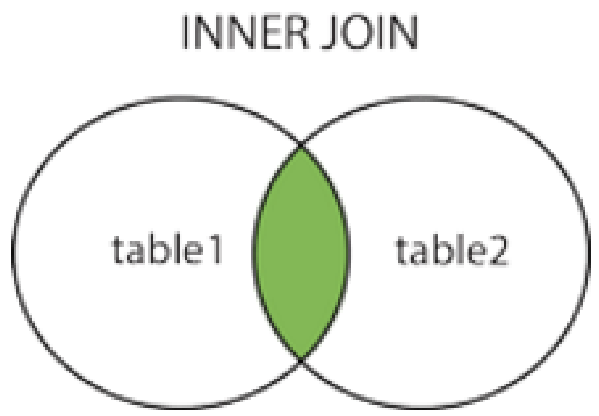
LEFT OUTER JOIN

- Every tuple in left table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of right table

RIGHT OUTER JOIN

- Every tuple in right table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of left table

Inner Join



Syntax:

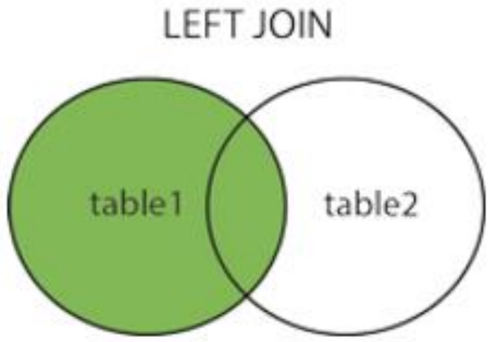
```
SELECT          column_name(s)
FROM            table1 JOIN table2
                ON table1.column_name = table2.column_name;
```

Example:

```
SELECT          *
FROM            employee JOIN department
                ON employee.dept_number = department.number;
```

Note: JOIN is the same as INNER JOIN.

Left Outer Join



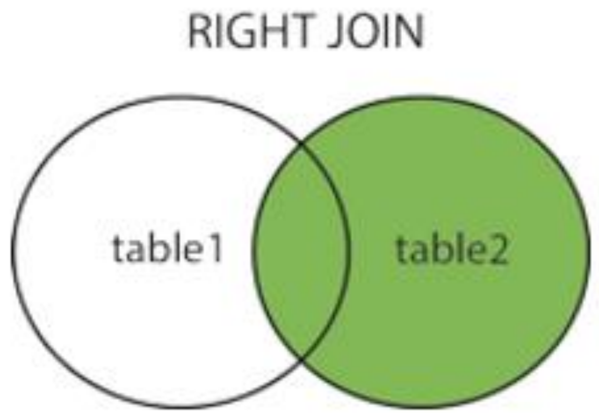
Syntax:

```
SELECT          column_name(s)
FROM            table1 LEFT JOIN table2
                ON table1.column_name = table2.column_name;
```

Example:

```
SELECT          *
FROM            employee LEFT JOIN department
                ON employee.dept_number = department.number;
```

Note: LEFT JOIN is the same as LEFT OUTER JOIN.



Right Outer Join

Syntax:

```
SELECT          column_name(s)
FROM            table1 RIGHT JOIN table2
                ON table1.column_name = table2.column_name;
```

Example:

```
SELECT          *
FROM            employee RIGHT JOIN department
                ON employee.SSN = department.manager_SSN;
```

Note: RIGHT JOIN is the same as RIGHT OUTER JOIN.

Aliasing / Tuple Variables

Aliases or **tuple variables**

- Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:
- Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables.

- For each employee, retrieve the employee's first and last name and his or her department name.

```
SELECT      e.fname,e.lname,
            d.name
FROM        employee AS e
JOIN        department AS d
ON          e.dept_number =
            d.number;
```

Renaming of Attributes in SQL

Use qualifier AS followed by desired new name

- Rename any attribute that appears in the result of a query

For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
SELECT     e.fname AS Employee_First_Name,
           e.lname AS Employee_Last_Name,
           s.fname AS Supervisor_First_Name,
           s.lname AS Supervisor_Last_Name
FROM       employee AS e
JOIN       employee AS s
           ON e.supervisor_SSN = s.SSN;
```

Arithmetic Operations

Standard arithmetic operators:

- Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**

Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.

```
SELECT      e.fname, e.lname,
            1.1 * e.salary AS increased_sal
FROM        employee AS e
JOIN        employee_project AS w
            ON e.SSN=w.employee_SSN
JOIN        project AS p
            ON w.project_number=p.number
WHERE       p.name='ProductX';
```

Ordering of Query Results

Use **ORDER BY** clause

- Keyword **DESC** to see result in a descending order of values
- Keyword **ASC** to specify ascending order explicitly
- Typically placed at the end of the query

```
SELECT      *
FROM        employee e
JOIN        department d
ON          e.dept_number =
            d.number
ORDER BY    d.name ASC,
            e.salary DESC,
            e.fname ASC;
```

JOIN with Aggregate Functions

GROUP BY may be applied to the result of two relations

For each project, retrieve the project number, the project name, and the number of employees who work on that project.

```
SELECT        p.number, p.name, COUNT(*)
 FROM         project p
 JOIN         employee_project ep ON
              p.number = ep.project_number
 GROUP BY     p.number, p.name;
```

JOIN with Aggregate Functions

Retrieve the total number of employees in the 'Research' department.

```
SELECT      COUNT(*)

FROM        employee

JOIN        department
            ON dept_number = number

WHERE       name =‘Research’;
```

Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT  SUM(salary), MAX(salary),

        MIN(salary), AVG(salary)

FROM    employee

JOIN    department
        ON dept_number = number

WHERE   name = 'Research';
```

HAVING Clauses

HAVING clause

- Provides a condition to select or reject an entire group:

For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT   p.number, p.name, COUNT(*)
FROM     project p
JOIN     employee_project ep
         ON p.number = ep.project_number
GROUP BY p.number, p.name
HAVING   COUNT(*) > 2;
```

References

Elmasri, R., & Navathe, S. (2017). Fundamentals of database systems (Vol. 7). Pearson

Nugent, D. (2017). Higher Nationals in Computing Core Textbook. Pearson Education Custom Content.