CS354: Database

Recap: Basic SQL Retrieval Query

A SQL query can consist of several clauses, but only **SELECT** and **FROM** are mandatory

SELECT <attribute list>
FROM <table list>
[WHERE <condition on the tables (join or selection)>]
[ORDER BY <attribute list>]
[LIMIT <number of tuples>]

Subquery

- Subquery: A parenthesized **SELECT-FROM-WHERE** statement which results in a relation of tuples
- Syntax:
  (**SELECT**-command)
- Usage
  - Inside **WHERE** clause (nested query)
  - Inside **FROM** clause (temporal relation)

Nested Query

- Nested query is when a subquery is specified within the **WHERE** clause of another query, called the outer query

- Syntax:

```
SELECT ...
FROM ...
WHERE ... (SELECT ...
           FROM ...        Nested Query
           WHERE ...)
```

Nested Query (2)

- Forms of nested query:
  - Set membership: **IN** and **NOT IN**
  - Set comparison:
    compareOp ANY or **compareOp ALL**
  - Test for empty relation: **EXIST**
- In theory, nesting can be arbitrarily deep but in practice the number of levels is limited

Example Query: Nested Query

Retrieve the name and address of all employees who work for the 'Research' department

- Soln #1:
```
SELECT fname, lname
FROM   employee, department
WHERE dno = dnumber
    AND dname = 'Research';
```

- Soln #2:
```
SELECT fname, lname
FROM EMPLOYEE
WHERE dno IN (SELECT pnumber
                FROM department
                WHERE dname = 'Research')
```

Example Query: Nested Query (2)

Find fname, lname of employees that do not have any dependents

```
SELECT fname, lname
FROM   employee
WHERE  ssn NOT IN (SELECT essn
                   FROM   dependent);
```

Correlated Nested Queries

- Correlated: inner query (query in the **WHERE** clause) uses one or more attributes from relation(s) specified in the outer query
- Uncorrelated: inner query is a stand-alone query that can be executed independently from the outer query
- Example Syntax:

```
SELECT ...
FROM   R1
WHERE  attr1 IN (SELECT attr2
                 FROM   R2
                 WHERE R2.attr3 = R1.attr4)
```

Example Query: Correlated Nested Query

Retrieve the name of each employee who has a dependent with the same name as the employee

```
SELECT   e.fname, e.lname
FROM     employee AS e
WHERE    e.ssn IN (SELECT essn
                   FROM    dependent
                   WHERE   essn = e.ssn
                   AND     e.fname = name);
```

Correlated Nested Query Execution

- FOR (each tuple X in the outer query) DO {
  Execute inner query using attribute value of tuple X
  }

- Example:

```
SELECT fname, lname, salary, uno
FROM   employee a
WHERE salary >= ALL (SELECT salary
                     FROM   employee a
                     WHERE  b.dno = a.dno)
```

Correlated Nested Query Execution (2)

FName	LName	DNo	Salary
John	Smith	4	50,000
James	Bond	4	80,000
Jane	Brown	3	60,000
Jennifer	Wallace	5	30,000
James	Borg	1	55,000
Joyce	English	5	25,000
Alicia	Wong	4	70,000

- Outer tuple a = | John | Smith | 4 | 50,000 |
  WHERE 50,000 >= ALL (SELECT salary FROM employee b where b.dno = 4)
  => FALSE

Correlated Nested Query Execution (2)

FName	LName	DNo	Salary
John	Smith	4	50,000
James	Bond	4	80,000
Jane	Brown	3	60,000
Jennifer	Wallace	5	30,000
James	Borg	1	55,000
Joyce	English	5	25,000
Alicia	Wong	4	70,000

- Outer tuple a =

James	Bond	4	80,000

  WHERE 80,000 >= ALL (SELECT salary FROM employee b where b.dno = 4)
  => TRUE (select tuple)

Correlated Nested Query Execution (2)

FName	LName	DNo	Salary
James	Bond	4	80,000
Jane	Brown	3	60,000
Jennifer	Wallace	5	30,000
James	Borg	1	55,000

```
SELECT fname, lname, salary, uno
FROM   employee a
WHERE salary >= ALL (SELECT salary FROM    employee a
                     WHERE  b.dno = a.dno)
```

Return name, salary, and department number of employees whose salary is the highest of all employees in his/her department

Correlated Nested Query Scope

Scoping rules defines where a name is visible

- Each nesting level constitutes a new inner scope
- Names of relations and their attributes in outer query are visible in the inner query but not the converse
- Attribute name specified inside an inner query is associated with nearest relation

Example: Scoping Nested Queries

```
SELECT <attribute list from R1 and/or R2>
FROM     R1, R2
WHERE  <conditions from R1 and/or R2> AND
        (SELECT <attribute list from R1, R2, R3 and/or R4>
         FROM     R3, R4
         WHERE <conditions from R1, R2, R3, and/or R4>)
```

- Attributes of R1 and R2 are visible in the inner query
- Attributes of R3 and R4 are not visible in the outer query

Example: Scoping Nested Queries (2)

```
SELECT <attribute list from R1 and/or R2>
FROM    R1, R2
WHERE  <conditions from R1 and/or R2> AND
       (SELECT  x
        FROM    R3, R4
        WHERE <conditions from R1, R2, R3, and/or R4>)
```

- If R3 or R4 contains the attribute name x, then x refers to that attribute in R3 or Rf
- If R3 and R4 does not contain the attribute name x, then x in the inner query refers to the attribute in R1 or R2

SQL Query: EXISTS

- Checks whether the result of a correlated nested query is empty (contains no tuples) or not

- Example: Retrieve the names of employees who have no dependents

```
SELECT fname, lname
FROM    employee
WHERE  NOT EXISTS (SELECT *
                   FROM   dependent
                   WHERE ssn = essn);
```

SQL Query: Aggregate Functions

- **COUNT**, **SUM**, **MAX**, **MIN**, **AVG** can be used in the **SELECT** clause

- Example: Find the sum, maximum, minimum, and average salary among all employees in the Research department

```
SELECT SUM(salary), MAX(salary)
           MIN(salary), AVG(salary)
FROM     employee, department
WHERE   dno = dnumber AND dname = 'Research'
```

SQL Query: Aggregate Functions (2)

- Name given to the selected aggregate function attribute is the same as the function call
  - **SELECT MAX(salary), MIN(salary), AVG(salary)**
    FROM employee;

max(salary)	min(salary)	avg(salary)

- Rename selected attributes with AS alias clause inside the SELECT clause
  - **SELECT MAX(salary) AS max, MIN(salary) AS min, AVG(salary) AS average**
    FROM employee;

SQL Example: Aggregate Function

Retrieve the names of all employees who have two or more dependents

SQL Query: GROUP BY

- Apply aggregate functions to subgroups of tuples in a relation
  - Corresponds to grouping and aggregate function in RA
  - Grouping attributes: attributes used to group the tuples
  - Function is applied to each subgroup independently
- Syntax:

```
SELECT    <attribute list>
FROM      <table list>
WHERE     <condition on the tables>
GROUP BY <grouping attributes>
```

GROUP BY Execution

A query with **GROUP BY** clause is processed as follows:

1. Select the tuples that satisfies the **WHERE** condition
2. Selected tuples from (1) are grouped based on their value in the grouping attributes
3. One or more set functions is applied to the group

SQL Example: GROUP BY

For each department, retrieve the department number, the number of employees in the department, and their average salary

```
SELECT    dno, count(*), avg(salary)
FROM      employee
GROUP BY dno
```

SQL Query: GROUP BY details

- What happens if we do not include certain grouping attributes in the **SELECT** clause?

- What happens if we include an attribute in the **SELECT** clause that is not in the group by attribute list?

SQL Query: HAVING

- **HAVING** clause specifies a selection condition on groups (rather than individual tuples)

- Filters out groups that do not satisfy the group condition

- Syntax:

```
SELECT      <attribute list>
FROM        <table list>
WHERE       <condition on the tables>
GROUP BY    <grouping attributes>
HAVING      <group condition>
```

SQL Query: HAVING Details

- Group condition is a condition on a set of tuples —> must use a grouping attribute inside the **HAVING** clause
- Process order:
    1. Select tuples that satisfy the **WHERE** condition
    2. Selected tuples from (1) are grouped based on their value in the grouping attributes
    3. Filter groups so only those satisfying the condition are left
    4. Set functions in the **SELECT** clause are applied to these groups

SQL Example: HAVING

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

```
SELECT      pnumber, pname, COUNT(*)
FROM        project, works_on
WHERE       pnumber = pno
GROUP BY    pnumber, pname
HAVING      COUNT(*) > 2;
```

SQL Example: HAVING (2)

For each department with at least 2 employees, find the department name, and the number of employees in that department that earns greater than $40K

```
SELECT      dname, COUNT(ssn)
FROM        department, employee
WHERE       dnumber = dno
   AND      salary > 40000
GROUP BY    dname
HAVING      COUNT(ssn) > 2;
```

Is this right? What does it return?

SQL Example: HAVING (2)

- Previous query only counts the number of departments that have at least 2 employees that earn more then $40K.

- ```
  SELECT   dname, COUNT(ssn)
  FROM     employee, department
  WHERE    dno = dnumber
     AND   dno IN ( SELECT      dno
                    FROM        employee
                    GROUP BY    dno
                    HAVING      COUNT(ssn) >= 2)
    AND     salary > 40000
  GROUP BY dname
  ```
  ```

Summary of SQL Queries

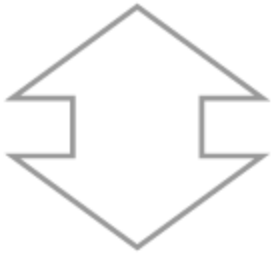
```
SELECT      [DISTINCT] <attribute list>
FROM         <table list>
[WHERE       <condition on the tables>]
[GROUP BY <grouping attributes>]
[HAVING      <group condition>]
[ORDER BY  <attribute list>   ASC | DESC]
[LIMIT       <number of tuples>]
```

This has every possible clause of a SQL command included

Query Formulation Techniques: INTERSECT

How to compute the intersection of two sets when the system does not support INTERSECT (e.g., MySQL)?

x IN (set1 INTERSECT set2)



(x IN set1) AND (x IN set2)

SQL Example: INTERSECT

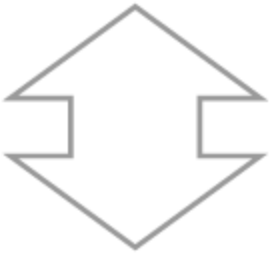
Find fname and lname of employees who work on some project controlled by the 'Research' department and also on some project controlled by the 'Administration' department

```
SELECT  fname, lname
FROM    employee
WHERE  ssn IN ( SELECT essn
                FROM    works_on, project, department
                WHERE  pno = pnumber
                    AND dnum = dnumber AND dname = 'Research')
   AND   ssn IN  ( SELECT essn
                   FROM    works_on, project, department
                   WHERE  pno = pnumber AND dnum = dnumber
                      AND dname = 'Administration');
```

Query Formulation Techniques: DIFFERENCE

How to compute the difference of two sets when SQL doesn't support set difference?

x IN (set1 - set2)



(x IN set1) AND (x NOT IN set2)

SQL Example: DIFFERENCE

Find SSN of employees in the 'Research' department who has no dependents

```
SELECT  ssn
FROM    employee
WHERE   ssn IN ( SELECT ssn
                 FROM   employee, department
                 WHERE dno = dnumber
                    AND dname = 'Research')
    AND   ssn NOT IN ( SELECT essn
                       FROM dependent)
```

Query Formulation Techniques: Superset

How to formulate set1 is a superset (contains) of another set, set2?

set1 CONTAINS set2 <=> set2 - set1 = EMPTY

```
SELECT ...
FROM ...
WHERE NOT EXISTS (SELECT *
                  FROM    <table>
                  WHERE x IN set2
                     AND  x NOT IN set1)
```

Query Formulation Techniques: Subset

How to formulate set1 is a subset (part of) of another set, set2?

set1 SUBSET set2 <=> set1 - set2 = EMPTY

```
SELECT ...
FROM ...
WHERE NOT EXISTS (SELECT *
                  FROM    <table>
                  WHERE x IN set1
                     AND  x NOT IN set2)
```

Query Formulation Techniques: Division

- How to compute the division between two relations?
- Example: Find lname of all employees who work on all projects controlled by department number
  - RA:

$$H1 = \pi_{\text{pnumber}}(\text{PROJECT} \bowtie_{\text{dnum=dnumber}} \sigma_{\text{dname='Research'}}(\text{DEPARTMENT}))$$
$$H2 = \pi_{\text{essn,pno}}(\text{WORKS\_ON})$$
$$H3 = H2 \div H1$$
$$\text{Answer} = \pi_{\text{fname,lname}}(\text{EMPLOYEE} \bowtie_{\text{ssn = ssn}} H3)$$

  - SQL: Use **NOT EXISTS** and set difference
    (can think of it as set of projects worked on employee contains set of projects controlled by department 4)

SQL Example: DIVISION

Find lname of all employees who work on all projects controlled by department number

```
SELECT  fname, lname
FROM    employee
WHERE   NOT EXISTS
        ( SELECT pnumber
          FROM   project
          WHERE  pnumber IN (SELECT pnumber
                             FROM   project
                             WHERE  dnum = 4)
             AND   pnumber NOT IN (SELECT dno
                                   FROM   works_on
                                   WHERE essn = ssn));
```

projects worked on by employee

project controlled by Research

Query Formulation Techniques: Only

- How to compute queries that ask for only?
- Example: Find the names of projects that are worked on by only employees in the ‘Research’ department?
- Formulate the solution using a subset condition:
  - Employees working on project p are a subset of employees in the Research department

SQL Example: Only

Find the names of projects that are worked on by only employees in the 'Research' department?

```
SELECT  pname
FROM    project
WHERE   NOT EXISTS
        ( SELECT ssn
          FROM   employee
          WHERE  ssn IN (SELECT essn
                         FROM   works_on
                         WHERE  pno = pnumber)
            AND  ssn NOT IN (SELECT ssn
                             FROM   employee, department
                             WHERE dno = dnumber
                               AND dname = 'Research'));
```

employees working on project p

employees from research department

Query Formulation Techniques: Most Number of

- How to compute queries that ask for the most number of some attribute?

- Example: Find the name of the departments with most number of employees?

Ans: Use nested query with the max function

```
SELECT      dname
FROM        department, employee
WHERE       dno = dnumber
GROUP BY dname
HAVING      COUNT(ssn) = (SELECT MAX(COUNT(ssn))
                          FROM    employee GROUP BY dno);
```

SQL Practice (1)

Find the name of the departments with 2 or more male employees

SQL Practice (2)

Find the name of employees who have more than two dependents and work on more than 2 projects

SQL Practice (3)

Find the name of the employees with the most number of dependents

SQL Practice (4)

Find fname and lame of employees who works on all projects that are worked on by John Smith

SQL Nested & Complex Queries: Recap

- Nested Queries
- Aggregate Functions
- SQL Grouping
  - GROUP BY
  - HAVING
- Query Formulation Techniques