Complex Queries

Database Systems (CSCI 440)

Fall 2014

Patrick Donnelly

Montana State University

MONTANA STATE UNIVERSITY

Administrivia

Homework #1:

due Friday, September 26, 2014

Reading:

Chapter 5.1

NULL Value

NULL

- Unknown value
- Unavailable or withheld value
- Not applicable attribute

Each individual NULL value considered to be different from every other NULL value.

NULL Value

NULL

- Unknown value
- Unavailable or withheld value
- Not applicable attribute

Each individual NULL value considered to be different from every other NULL value.

SQL allows queries that check whether an attribute value is NULL

IS or IS NOT NULL

Query 18. Retrieve the names of all employees who do not have supervisors.

```
Q18:  SELECT  Fname, Lname
      FROM    EMPLOYEE
      WHERE   Super_ssn IS NULL;
```

Three-Valued Logic

SQL uses a three-valued logic: `TRUE`, `FALSE`, and `UNKNOWN`

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Subqueries

Definition

Subqueries or **Nested queries** are complete select-from-where blocks within `WHERE` clause of another query, the **Outer Query**.

`IN`

The comparison `IN` operator compares value v with a set (or multiset) of values V and evaluates to `TRUE` if v is one of the elements in V.

Subquery Example

```
Q4A:  SELECT   DISTINCT Pnumber
      FROM     PROJECT
      WHERE    Pnumber IN
               ( SELECT     Pnumber
                 FROM       PROJECT, DEPARTMENT, EMPLOYEE
                 WHERE      Dnum=Dnumber AND
                            Mgr_ssn=Ssn AND Lname='Smith' )
               OR
               Pnumber IN
               ( SELECT     Pno
                 FROM       WORKS_ON, EMPLOYEE
                 WHERE      Essn=Ssn AND Lname='Smith' );
```

`IN` Operator

The `IN` operator allows you to specify multiple values in a `WHERE` clause and use the tuples of values in comparisons by placing them within parentheses.

```
SELECT    DISTINCT Essn
FROM      WORKS_ON
WHERE     (Pno, Hours) IN ( SELECT    Pno, Hours
                            FROM      WORKS_ON
                            WHERE     Essn='123456789' );
```

ANY and SOME Operators

ANY Operator

The comparison ANY operator returns TRUE if the value v is equal to some value in the set V and is hence equivalent to IN.

ANY is combined with $=, >, >=, <, <=$, and $<>$.

SOME Operator

The comparison ANY operator is equivalent to ANY and IN.

ALL Operator

ANY Operator

The comparison ALL operator returns TRUE if the comparison is true for all values in the set V.

ALL is combined with $=, >, >=, <, <=, <>$, and with other aggregate operators.

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL  ( SELECT   Salary
                          FROM     EMPLOYEE
                          WHERE    Dno=5 );
```

Aliases

Definition

Aliases are tuple variables for all tables referenced in SQL query. Aliases help avoid potential errors and ambiguities.

`AS` Qualifier

Use qualifier `AS` followed by desired new name to rename any attribute that appears in the result of a query.

This allows explicit use of values in WHERE clause.

Alias Examples

```
Q8A:   SELECT   E.Lname AS Employee_name, S.Lname AS Supervisor_name
       FROM     EMPLOYEE AS E, EMPLOYEE AS S
       WHERE    E.Super_ssn=S.Ssn;
```

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

```
Q16:   SELECT   E.Fname, E.Lname
       FROM     EMPLOYEE AS E
       WHERE    E.Ssn IN ( SELECT   Essn
                           FROM     DEPENDENT AS D
                           WHERE    E.Fname=D.Dependent_name
                           AND E.Sex=D.Sex );
```

Types of Queries

Subqueries differ based on placement:

Definition

Nested Query: subquery appears in the `WHERE` clause of the SQL.

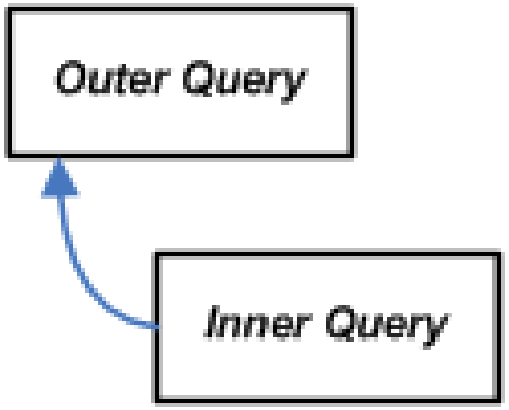
Definition

Inline View: subquery appears in the `FROM` clause of the SQL.

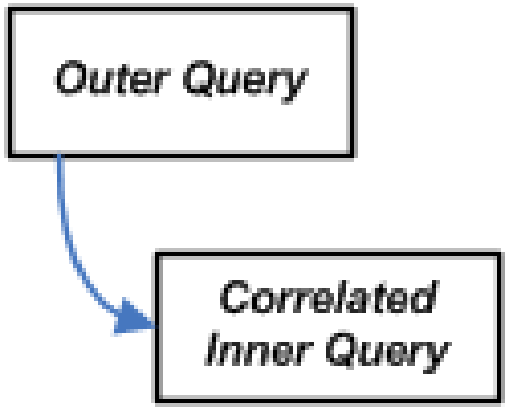
Definition

Scalar Subquery: subquery appears in the `SELECT` clause of SQL.

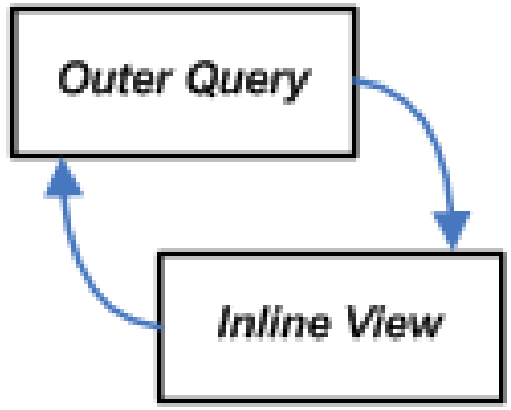
Types of Queries



Ordinary Subquery



Correlated Subquery



Inline View

Correlated Nested Queries

Subqueries also differ in the way the subquery is parsed:

- Simple Subquery
- Correlated Subquery

Correlated Nested Queries

Subqueries also differ in the way the subquery is parsed:

- Simple Subquery
- Correlated Subquery

Definition

A nested query is said to be **correlated** if a condition in the `WHERE` references some attribute of a relation declared in the outer query.

```
SELECT employee_number, name
    FROM employees AS Bob
    WHERE salary > (
        SELECT AVG(salary)
            FROM employees
            WHERE department = Bob.department);
```

Existence and Uniqueness

`EXISTS` and `NOT EXISTS` function

The comparison `EXISTS` function checks whether the result of a correlated nested query is empty or not and is typically used in conjunction with a correlated nested query.

`UNIQUE` function

The SQL function `UNIQUE(Q)` returns `TRUE` if there are no duplicate tuples in the result of query Q.

Grouping Results

Grouping creates subgroups of tuples before summarizing and allows applying function to each such group independently.

`GROUP BY` Function

The `GROUP BY` allows partitioning relation into subsets of tuples by specifying the grouping attributes.

If `NULL`s exist in grouping attribute, separate group is reated for all tuples with a `NULL` value in grouping attribute.

HAVING Clause

The HAVING clause provides a grouping condition on the summary information.

Query 28. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

```
Q28:  SELECT   Dnumber, COUNT (*)
      FROM     DEPARTMENT, EMPLOYEE
      WHERE    Dnumber=Dno AND Salary>40000 AND
               ( SELECT       Dno
                 FROM         EMPLOYEE
                 GROUP BY Dno
                 HAVING       COUNT (*) > 5)
```

Ordering of Query Results

ORDER BY

The ORDER BY clause:

- Keyword **DESC** to see result in a descending order of values
- Keyword **ASC** to specify ascending order explicitly

Example

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

SELECT <attribute list>
FROM <table list>
[**WHERE** <condition>]
[**ORDER BY** <attribute list>];

Query Template

SELECT <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>];

Aggregate Functions in SQL

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary.

Built-in aggregate functions:

- `COUNT` - number of values
- `SUM` - sum of values
- `MAX` - maximum value
- `MIN` - minimum value
- `AVG` - average value

These functions can be used in the `SELECT` clause or in a `HAVING` clause.

Aggregate Functions in SQL

NULL values discarded when aggregate functions are applied to a particular column.

Query 20. 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.

```
Q20:  SELECT  SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
      FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
      WHERE   Dname='Research';
```

Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).

```
Q21:  SELECT  COUNT (*)
      FROM    EMPLOYEE;

Q22:  SELECT  COUNT (*)
      FROM    EMPLOYEE, DEPARTMENT
      WHERE   DNO=DNUMBER AND DNAME='Research';
```