IT 775
Database Technology
SQL-DML

Select Statement
SubQueries

Nested Subqueries

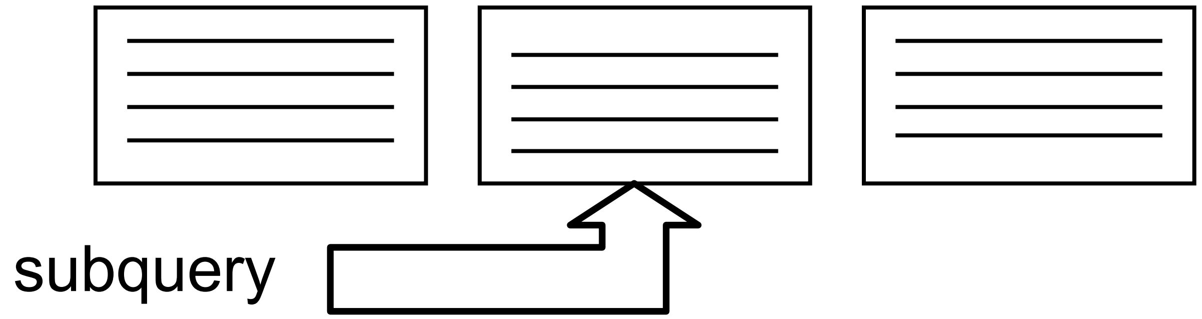
- inner query returns information to an outer query
- subquery is a parenthesized select statement embedded in another select statement
- subquery can occur in the WHERE clause of outer query

```
SELECT columns FROM tables
WHERE predicate
   ( SELECT columns FROM tables WHERE cond-expr )
```

- subquery can occur in the FROM clause of outer query

```
SELECT columns
  FROM table JOIN ( SELECT cols FROM tbls WHERE pred )
WHERE predicate
```

- or both



Rules for Subqueries

each subquery must be enclosed in parenthesis

ORDER BY clause can appear only at the outermost level

subqueries can nest to any level

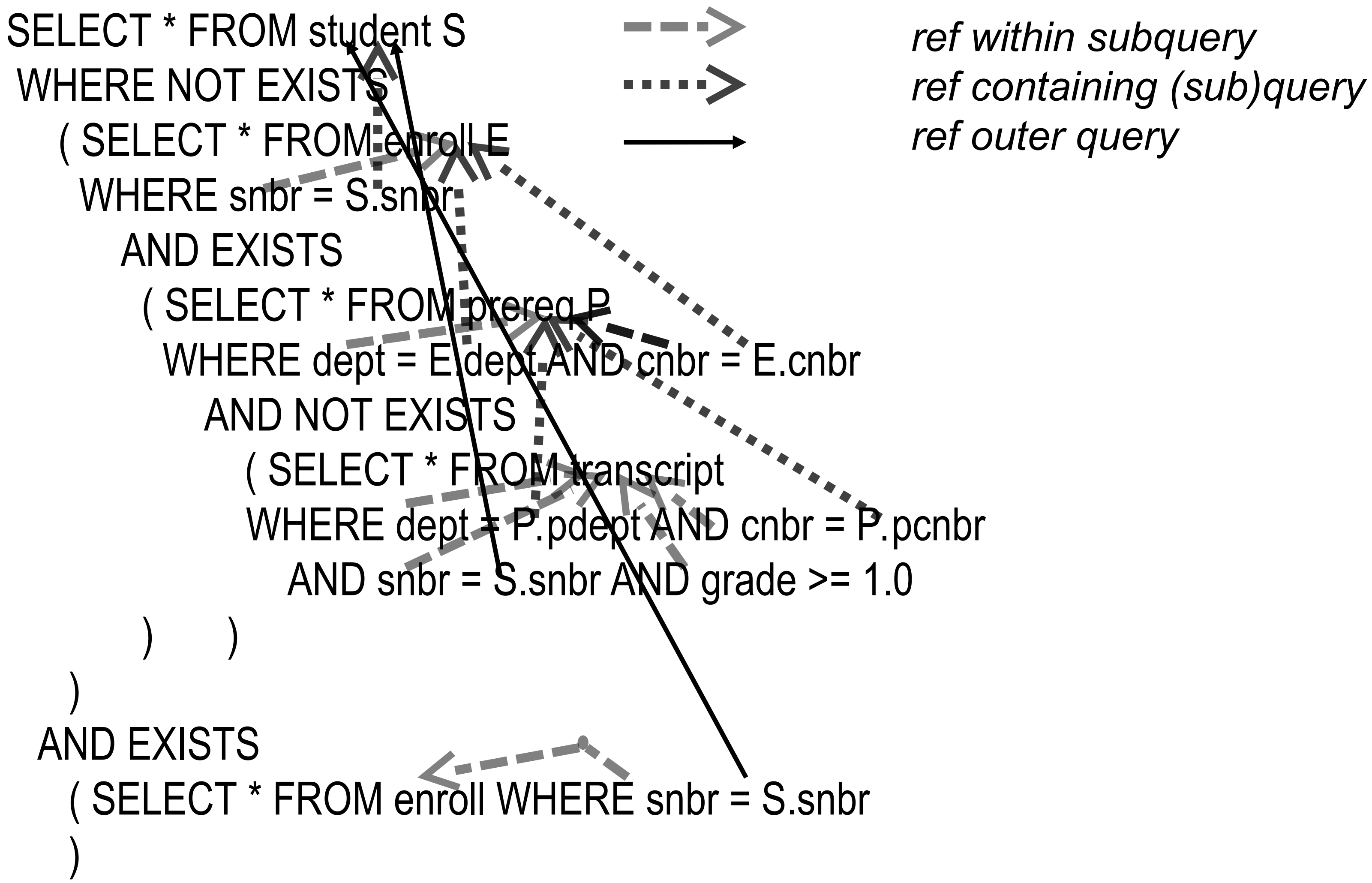
subquery can reference relation tables and attribute names in containing (outer) queries

subquery cannot reference relation tables and attributes names in contained (inner) clauses

ambiguous unqualified attribute references (no relation name or range variable) refer to innermost inclusion level

see example next

Subquery Example



```
SELECT * FROM student S
 WHERE NOT EXISTS
    ( SELECT * FROM enroll E
      WHERE snbr = S.snbr
         AND EXISTS
           ( SELECT * FROM prereq P
             WHERE dept = E.dept AND cnbr = E.cnbr
                AND NOT EXISTS
                   ( SELECT * FROM transcript
                   WHERE dept = P.pdept AND cnbr = P.pcnbr
                      AND snbr = S.snbr AND grade >= 1.0
           )     )
    )
  AND EXISTS
    ( SELECT * FROM enroll WHERE snbr = S.snbr
    )
```

Subquery Result Forms & Usage

subquery can return:

one column of values (degenerate table)

- single row –– ***scalar*** subquery
- multiple rows –– ***set-valued*** subquery

multiple columns of values (full table)

- ***existence*** subquery only
  - returns true if at least one row qualifies
  - returns false if subquery result is empty
- set-valued subquery + row constructor
  - tuple is treated as an element in a set.

subquery usage

FROM clause – arbitrary table result – aliased

WHERE/HAVING clause – acceptable result: f(comparator)

• = <> < > BETWEEN	scalar result only
• IN ALL ANY	set result only (single column)
• EXISTS, NOT EXISTS	full table result

Scalar Subquery Result Usage

table of 1 column and 1 row –– i.e., single value

can be used directly in conditional expressions

if value is ordinal, it can be an operand in a comparison expr

Example: students whose gpa exceeds that of Joe College

```
SELECT sname FROM student
WHERE gpa > ( SELECT gpa FROM student
                  WHERE sname = 'joe college'
                )
```

Example: students in the same major as Joe College

```
SELECT sname FROM student
WHERE major = ( SELECT major FROM student
                  WHERE sname = 'joe college'
                )
```

Statistical Subquery Scalar-Valued

- students with above average gpa values
  - stat subquery version

```
SELECT sname, gpa FROM student
WHERE gpa > ( SELECT avg( gpa ) FROM student );
```

- students with above average in-major gpa values

```
SELECT sname, major, gpa FROM student S
WHERE gpa >
  ( SELECT avg( gpa ) FROM student WHERE major = S.major );
```

note: horrible efficiency –– this correlated subquery reevaluates the average gpa of a major for each student in that major

(unless optimizer rescues us)

Set–Valued Subquery Result Usage

must be in the form of a table of 1 column

usually more than 1 row –– i.e., multi–valued

used in set comparison expressions

expr θ SOME (set-valued subquery)

expr θ ANY (set-valued subquery)

expr θ ALL (set-valued subquery)

expr [NOT] IN (set-valued subquery)

e.g., CEPS students with 3.0 or better gpa

```
SELECT sname FROM student
WHERE gpa >= 3.0
   AND major IN ( SELECT dname FROM department
                  WHERE college = 'ceps')
```

subquery evaluates to a set of departments

∀ θ is a comparison operator, i.e., =, <, …

Statistical Subquery Set Valued

- compute students' in–major gpas

```
SELECT sname, student.snbr,
           SUM( cr * grade ) / sum( cr ) AS majgpa
  FROM ( student JOIN transcript
            ON student.snbr = transcript.snbr AND major = dept
         )
         NATURAL JOIN course
GROUP BY sname, student.snbr;
```

- department with highest average gpa
  - multi-valued subquery

```
SELECT major, avg( gpa ) FROM student
 GROUP BY major
HAVING avg( gpa ) >= ALL
  ( SELECT AVG(gpa) FROM student
    GROUP BY major
  );
```

Existence Subquery Result Usage

preceded by an EXISTS or NOT EXISTS quantifier

determines whether a nested subquery qualifies any rows
row count is all that matters, not values
select attribute (often *) since actual values are not pulled out

EXISTS
succeeds when inner query produces non-empty result
- at least one row

fails when it produces no rows –– empty result

NOT EXISTS
succeeds when inner query produces no rows
- empty table with 0 rows

fails when it produces non-empty table

Subquery Evaluation

query evaluation proceeds inside out

like parenthesized subexpressions

subquery evaluation is either correlated or uncorrelated

uncorrelated

inner select does not reference outer select

inner & outer select evaluate independently

- inner select evaluates once at start
- outer select evaluates starts after inner result available
- efficient

correlated

inner select references outer select attribute, table, or alias

inner & outer select evaluation is interdependent, ∴

- inner select evaluates once for each outer select row
- powerful, but can be inefficient

Uncorrelated Multi–valued Subquery

CEPS students with 3.0 or better gpa – three variations

```
SELECT sname FROM student
WHERE gpa >= 3.0
   AND major IN ( SELECT dname FROM department WHERE college = ‘science' )

SELECT sname FROM student
WHERE gpa >= 3.0
   AND major = SOME ( SELECT dname FROM department WHERE college =
    ‘science' )

SELECT sname FROM student
WHERE gpa >= 3.0
    AND major = ANY ( SELECT dname FROM department WHERE college = ‘science'
    )
```

student with the highest gpa

```
SELECT sname FROM student
WHERE gpa >= ALL ( SELECT gpa FROM student)
```

Uncorrelated Multi–valued Subqueries

- departments with no majors –– two variations

```
SELECT dname FROM department
WHERE dname NOT IN ( SELECT major FROM student )

SELECT dname FROM department
WHERE dname <> ALL ( SELECT major FROM student )
```

- list departments with majors –– two variations

```
SELECT dname FROM department
WHERE dname IN ( SELECT major FROM student )

SELECT dname FROM department
WHERE dname = ANY ( SELECT major FROM student )
```

- cs students whose advisor is from outside 'cs'

```
SELECT sname FROM student
WHERE major = 'cs' AND advisor NOT IN
                        ( SELECT name FROM faculty WHERE dept = 'cs' )
```

join is a better way for this

Correlated Multi–valued Subquery

- students whose advisors are from their respective majors

  SELECT sname FROM student WHERE advisor IN
  (SELECT name FROM faculty WHERE dname = *major*)

  SELECT sname FROM student WHERE advisor =
  ANY (SELECT name FROM faculty WHERE dname = major)

  - subquery references attribute major from the outer query
  - the appropriate department to check varies with each student

- students whose advisor is from outside their major

  SELECT sname FROM student WHERE advisor NOT IN
  (SELECT name FROM faculty WHERE dname = *major*)

  SELECT sname FROM student WHERE advisor <> ALL
  (SELECT name FROM faculty WHERE dname = major)

  - inefficient, subquery recomputes department faculty for each major

Correlated Multi–valued Subquery - Aliases

- students taking at least one course in their respective major

```
SELECT sname FROM student S
WHERE S.major IN
     ( SELECT dept FROM enroll WHERE snbr = S.snbr )
```

  - snbr refers to table enroll

S.snbr refers to table student

- student(s) in each major with the highest gpa in the major

```
SELECT sname FROM student S
WHERE gpa >= ALL
   ( SELECT gpa FROM student WHERE major = S.major )
```

inner & outer both range over same table
need alias to distinguish references
unqualified major refers to inner subquery
qualified major refers to the outer select

Existence Subquery Patterns

SELECT subject
WHERE EXISTS (SELECT case(s) that qualify subject)

- students taking at least 1 major course (used earlier in join)

```
SELECT sname FROM student WHERE EXISTS
        ( SELECT * FROM enroll WHERE snbr = student.snbr AND dept =
  major );
```

SELECT subject
WHERE NOT EXISTS (SELECT case(s) that disqualify subject)

- students taking no major courses

```
SELECT sname FROM student WHERE NOT EXISTS
        ( SELECT * FROM enroll WHERE snbr = student.snbr AND dept =
  major );
```

- students taking only major courses (no non-major courses)

```
SELECT sname FROM student WHERE NOT EXISTS
        ( SELECT * FROM enroll WHERE snbr = student.snbr AND dept <>
  major );
```

- students not enrolled in any courses

```
SELECT sname FROM student WHERE NOT EXISTS
  ( SELECT * FROM enroll WHERE snbr = student.snbr )
```

Nested Existence Subqueries

```
SELECT subject (that satisfies all requirements)
WHERE NOT EXISTS
                SELECT requirements the subject must meet
                WHERE NOT EXISTS
                     SELECT case where subject fails the requirement
```

principle: double negative ⇒ positive

if there is no failed prerequisite, all prerequisites are met

Example: course enrollments where student satisfies all prereqs -- note pattern: (.. not exists .. (not exists ..))

```
SELECT sname, dept, cnbr FROM student S JOIN enroll E ON S.snbr = E.snbr
WHERE NOT EXISTS                  succeeds when student has
                 ( SELECT * FROM prereq                 completed all prerequisites
                   WHERE dept = E.dept
                     AND cnbr = E.cnbr
                     AND NOT EXISTS          succeeds when student has
                 ( SELECT * FROM transcript              not completed a prerequisite
                            WHERE snbr = S.snbr                  fails if a prereq is
  completed
          AND dept = pdept AND cnbr = pcnbr
                             AND grade >= 1.0
                 )          );
```

Similar Existence Queries

- students who satisfy all prereqs for all enrollments

```
SELECT sname FROM student S
WHERE NOT EXISTS
    ( SELECT * FROM enroll E
                                JOIN prereq P ON P.dept = E.dept AND P.cnbr = E.cnbr
                  WHERE E.snbr = S.snbr AND NOT EXISTS
                               ( SELECT * FROM transcript
                                 WHERE snbr = S.snbr AND grade >= 1.0
                           AND dept = pdept AND cnbr = pcnbr
                 )                 );
```

- students who satisfy their degree plan (same structure)

```
SELECT       snbr, sname FROM student S
WHERE        NOT EXISTS
             ( SELECT * FROM degrplan D
     WHERE id = S.snbr AND NOT EXISTS
        ( SELECT * FROM transcript
                  WHERE snbr = D.id AND grade >= 1.0
                     AND dept = D.dname AND cnbr = D.cnbr
    )           );
```

Existence Subqueries –– (exists (not exists))

- enrollments where student skipped a prereq

```
SELECT sname, dept, cnbr
  FROM student JOIN enroll ON student.snbr = enroll.snbr
WHERE EXISTS                                        succeeds when student has
                  ( SELECT * FROM prereq                      not completed a prerequisite
                    WHERE dept = enroll.dept
                      AND cnbr = enroll.cnbr
                      AND NOT EXISTS                               succeeds when student has
                               ( SELECT * FROM transcript     not completed a prerequisite
                          WHERE snbr = student.snbr                fails if a prereq is completed
                                                    AND dept = pdept
                                     AND cnbr = pcnbr
                                     AND grade >= 1.0
                                  )
     );
```

Existence Subqueries –– (not exists)

- enrollments & prereq that student skipped

```
SELECT sname, enroll.dept, enroll.cnbr, pdept, pcnbr
  FROM student
          JOIN enroll ON student.snbr = enroll.snbr
          JOIN prereq ON enroll.dept = prereq.dept
                        AND enroll.cnbr = prereq.cnbr
WHERE    NOT EXISTS
         ( SELECT * FROM transcript
           WHERE snbr = student.snbr
               AND dept = pdept
               AND cnbr = pcnbr
               AND grade >= 1.0
         );
```

Existence Subqueries – (not exists (exists))

- enrollments where student satisfies no prerequisites

```
SELECT sname, dept, cnbr
 FROM student
        JOIN enroll ON student.snbr = enroll.snbr
WHERE NOT EXISTS                              succeeds when student has
( SELECT * FROM prereq                                  not completed a prerequisite
 WHERE dept = enroll.dept
    AND cnbr = enroll.cnbr
    AND EXISTS                                          succeeds when student has
     ( SELECT * FROM transcript                      completed a prerequisite
       WHERE snbr = student.snbr                        fails if a prereq is uncompleted
                AND dept = pdept
                AND cnbr = pcnbr
                AND grade >= 1.0
     )
);
```

Existence Subqueries –– four levels

- students who satisfy all prerequisites for all enrollments
  - … (NOT EXISTS (EXISTS (NOT EXISTS …))) …

```
SELECT sname
 FROM student S
WHERE NOT EXISTS                                              fails when student has an
( SELECT ∗ FROM student                                     enrollment but has not
                           JOIN enroll ON student.id = enroll.id
   WHERE student.id = S.id                                    satisfied some prereq
       AND EXISTS                                             succeeds when student has
        ( SELECT ∗ FROM prereq                                not completed all prereq
              WHERE dept = enroll.dept
                       AND cnbr = enroll.cnbr
                       AND NOT EXISTS                                    fails when student has
        ( SELECT ∗ FROM transcript                            completed the prereq
                                 WHERE id = student.id                   succeeds otherwise
                                        AND dept = pdept
                                        AND cnbr = pcnbr
                                   AND grade >= 1.0
                 )
        )
);
```

Subquery vs. Join

- use the more intuitive form
  - i.e., which syntax better conveys the nature of the query
  - join often more intuitive for an existing relationship
    - Like a FK/PK
  - subquery often more intuitive for
    - creating an ad-hoc relationship
    - passing an aggregate value to the outer query
    - isolating parts of a long complex query from each other
- inner joins often more efficient than correlated subqueries
  - most vendors make considerable effort to optimize joins
  - especially true for defined relationships such as PK/FK
- result columns must come from table(s) in the outer query
  - join often helps put more tables in the outer query

Subquery vs. Join: Obvious Relationship

- join often more intuitive for existing relationships (FK/PK)
  - query to list names of students enrolled in cs courses

```
SELECT sname
  FROM student S
         JOIN enroll E
           ON E.snbr = S.snbr
WHERE E.dept = 'cs‘;
```

```
SELECT sname
  FROM student S
WHERE S.snbr IN
      ( SELECT snbr
          FROM enroll
         WHERE dept = 'cs' );
```

  - query to list students and their teachers this semester

```
SELECT DISTINCT sname, name
  FROM student S
         JOIN enroll E
              ON S.snbr = E.snbr
         JOIN instr I
              ON E.dept = I.dname
              AND E.cnbr = I.cnbr
              AND E.sect = I.sect;
```

```
SELECT DISTINCT sname, name
  FROM student S, instr E
 WHERE EXISTS
       ( SELECT * FROM enroll
         WHERE snbr = S.snbr
           AND dept = I.dname
           AND cnbr = I.cnbr
           AND sect = I.sect );
```

Subquery vs. Join: Statistical Summary

- subquery value for the outer query

```
SELECT sname FROM student      join version ???
WHERE gpa >=
   ( SELECT AVG( gpa ) FROM student );
```

- subquery for a complex query

```
SELECT * FROM student S          join version ???
 WHERE NOT EXISTS
    ( SELECT * FROM enroll E
      WHERE snbr = S.snbr AND EXISTS
         ( SELECT * FROM prereq P
           WHERE dept = E.dept AND cnbr = E.cnbr
              AND NOT EXISTS
               ( SELECT * FROM transcript
                 WHERE dept = P.pdept AND cnbr = P.pcnbr
                    AND snbr = S.snbr AND grade >= 1.0
     )     )      )
   AND EXISTS
     ( SELECT * FROM enroll WHERE snbr = S.snbr );
```

Bottom Line

- if straightforward join expression is available,
  it is probably better than a subquery
  - more intuitive
  - more efficient
- otherwise, subquery expression is probably more intuitive