CSC 453 Database Systems Lecture

Tanu Malik
College of CDM
DePaul University

Mid-term Exam

- Date: Oct 15th (in-class) OL: Oct 13th-Oct 16th
- Syllabus: Everything upto October 8th
- Guidance: Study guide
- Time: 135 minutes
- Venue: Class
- Format: Paper-based, multiple-choice, multi-select questions
- Allowed: 1 sheet of A4 size typed/handwritten notes
- Partial grading

If you are waking up late

- Course Website:
  - http://facsrv.cs.depaul.edu/~tmalik1/teaching/csc453/index.html
  - FAQs

Quiz Errata

- Consider a social network of users in which each user has a 'friend' relationship with other users.
- Errata: None of them are correct.

```
Create table SNUser(
userid integer
);
Create table Friend(
Userid integer,
friendid integer,
foreign key (userid) references SNUser(userid),
foreign key (friendid) references SNUser(userid)
);
```

HW 1 PartC Q10 Errata

- ~~Find the details of a) the most costly trip, b) the cheapest trip taken by either the air, rail, or two separate queries.~~
- Find the details of a) the most costly trip, b) the cheapest trip taken by either the air, rail, or or car. Write two separate queries.
  - You must write one query for part (a) and one for part (b)
- In addition, bonus points will be awarded if you have written your two queries using the following SQL keywords: WHERE, SELECT, JOIN, EXCEPT/MINUS, RENAME, UNION, INTERSECT.
  - To receive bonus points do not use ALL, ANY, DISTINCT, GROUP BY HAVING, MAX, MIN.

Last time

- SQL
  - Basic SQL on single table
  - Basic SQL on two tables

Today

- SQL
  - Review
  - Basic SQL on two tables
  - Subqueries
    - Nested subqueries
    - Correlated subqueries
  - Update, Delete statements
- Creating Views
  - Temporary Tables
- Recursion

SQL Review

Queries on a single table

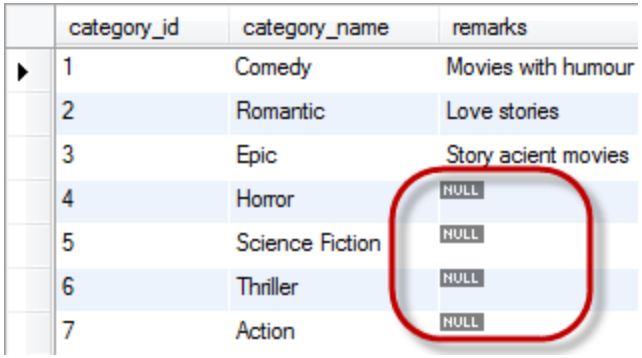
- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY

SQL Review

Find how many students joined the University in 2008?

SQL NULLs Review

- Given this data, what will the following query return



category_id	category_name	remarks
1	Comedy	Movies with humour
2	Romantic	Love stories
3	Epic	Story acient movies
4	Horror	NULL
5	Science Fiction	NULL
6	Thriller	NULL
7	Action	NULL

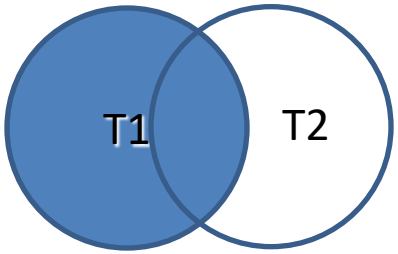
- `Select count(*) from Movies`
- `Select count(remarks) from Movies`

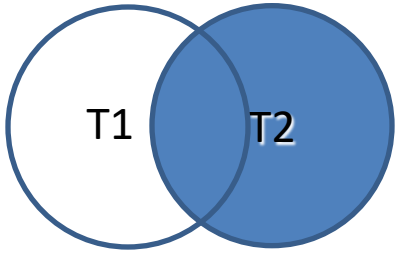
SQL: Review

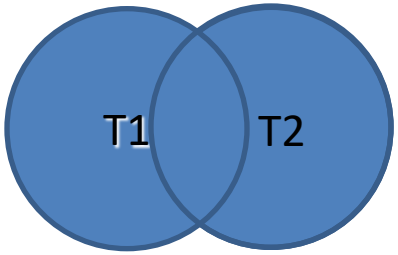
Queries on two tables

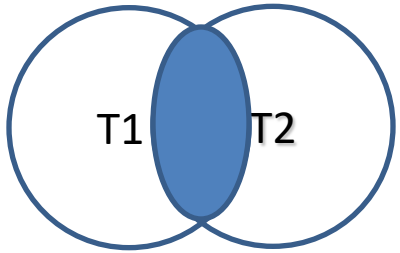
- Join
- Cross Join
- Outer join (Left, Right, Full)

Joins









Another SQL Query

- List all students who are enrolled in courses.
- List all students who are not enrolled in a course.

	LASTNAME	FIRSTNAME	SID	SSN	CAREER	PROGRAM	CITY	STARTED	STUDENTID	COURSEID	QUARTER	YEAR
1	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1020	Fall	2012
2	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1092	Fall	2012
3	Patel	Prakash	75234	(null)	UGRD	COMP-SCI	Chicago	2011	75234	3201	Winter	2012
4	Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2009	8871	1092	Fall	2013
5	Johnson	Peter	32105	123456789	UGRD	COMP-SCI	Chicago	2010	(null)	(null)	(null)	(null)
6	Brennigan	Marcus	90421	987654321	UGRD	COMP-GAM	Evanston	2010	(null)	(null)	(null)	(null)
7	Snowdon	Jennifer	93321	321321321	GRD	COMP-SCI	Springfield	2012	(null)	(null)	(null)	(null)
8	Patel	Deepa	14662	(null)	GRD	COMP-SCI	Evanston	2013	(null)	(null)	(null)	(null)
9	Starck	Jason	19992	789789789	UGRD	INFO-SYS	Springfield	2009	(null)	(null)	(null)	(null)

DIFFERENCE Operation

	LASTNAME	FIRSTNAME	SID	SSN	CAREER	PROGRAM	CITY	STARTED	STUDENTID	COURSEID	QUARTER	YEAR
1	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1020	Fall	2012
2	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1092	Fall	2012
3	Patel	Prakash	75234	(null)	UGRD	COMP-SCI	Chicago	2011	75234	3201	Winter	2012
4	Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2009	8871	1092	Fall	2013
5	Johnson	Peter	32105	123456789	UGRD	COMP-SCI	Chicago	2010	(null)	(null)	(null)	(null)
6	Brennigan	Marcus	90421	987654321	UGRD	COMP-GAM	Evanston	2010	(null)	(null)	(null)	(null)
7	Snowdon	Jennifer	93321	321321321	GRD	COMP-SCI	Springfield	2012	(null)	(null)	(null)	(null)
8	Patel	Deepa	14662	(null)	GRD	COMP-SCI	Evanston	2013	(null)	(null)	(null)	(null)
9	Starck	Jason	19992	789789789	UGRD	INFO-SYS	Springfield	2009	(null)	(null)	(null)	(null)

Subtract

	LASTNAME	FIRSTNAME	SID	SSN	CAREER	PROGRAM	CITY	STARTED	STUDENTID	COURSEID	QUARTER	YEAR
1	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1020	Fall	2012
2	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009	11035	1092	Fall	2012
3	Patel	Prakash	75234	(null)	UGRD	COMP-SCI	Chicago	2011	75234	3201	Winter	2012
4	Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2009	8871	1092	Fall	2013

SQL: EXCEPT/MINUS

- The SQL **EXCEPT** clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.

- SELECT column1 [, column2]
  FROM table1 [, table2]
  [WHERE condition]
  EXCEPT
  SELECT column1 [, column2]
  FROM table1 [, table2]
  [WHERE condition]

SQL

```
SELECT sid
FROM student left outer join course
MINUS
SELECT studentid
FROM enrolled
```

- OR

```
SELECT sid
FROM student left outer join course where courseid is null
```

- But courseid must not be null in Enrolled

Practice

- List student members of DeFrag and HerCTI.

UNION Operation

Students1

Number	Surname	Age
7274	Robinson	18
7432	O'Malley	19
9824	Darkes	28

Students2

Number	Surname	Age
9297	O'Malley	39
7432	O'Malley	19
9824	Darkes	28

Number	Surname	Age
9297	O'Malley	39
7432	O'Malley	19
9824	Darkes	28
7274	Robinson	18

SQL: UNION

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order
- SELECT *column_name(s)* FROM *table1*
  UNION
  SELECT *column_name(s)* FROM *table2*;

Mixed Practice

- List students that are members of both DeFrag and HerCTI groups.

INTERSECT Operation

Students 1

Number	Surname	Age
7274	Robinson	18
7432	O'Malley	19
9824	Darkes	28

Students2

Number	Surname	Age
9297	O'Malley	39
7432	O'Malley	19
9824	Darkes	28

Number	Surname	Age
7432	O'Malley	19
9824	Darkes	28

SQL: Intersect

- Combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.
- SELECT column1 [, column2]

  FROM table1 [, table2]

  [WHERE condition]

  INTERSECT

  SELECT column1 [, column2]

  FROM table1 [, table2]

  [WHERE condition]

Mixed Practice

- We require that all gaming students are members of DeFrag; list students that violate this rule.

Joins for comparison: Self Join

- Joins the table to itself

```
SELECT …
TABLE1 as Alias1,  TABLE as Alias2
ON Alias1.AttributeA = Alias2.AttributeB;
```


25

Self-Join

- List students enrolled in two or more courses
- List pair of students taking the same course.

SQL Queries

- General form of a query:

  1. SELECT *list of attributes to report*
  3. FROM *list of tables* [CROSS JOIN| [LEFT|REIGHT] OUTER JOIN]
  2. [WHERE *tuple condition*]
  5. [GROUP BY *list of grouping attributes*]
  6. [HAVING *group condition*]
  4. [ORDER BY *list of ordering attributes*] ;

     [UNION|INTERSECT|EXCEPT]

- Result is an ordered set of ordered tuples

Subqueries

Subqueries

- The result of one query may be needed by another to compute its result.
- There are two kinds of subqueries
  - **Nested subqueries in which result of a inner subquery is used by outer subquery to compute result**
  - Correlated query in which each tuple of outer query runs aa loop for the inner subqery.

Nested Subquery Example

- Find out students who started the earliest.

```
SELECT min(started)
FROM student

SELECT student.*
FROM student
WHERE started = ‘2003’
```

- This is not a robust way of querying

Nested Subquery Example

- Nest Q1 inside Q2

Outer query

```
SELECT S1.*
FROM student S1
WHERE started = (SELECT
min(S2.started)
FROM student S2)
```

Subquery OR
Inner query

Writing Nested Subqueries

- A subquery is nested (using parentheses) within an outer query
- Outer query uses the result of the subquery, which can be either single value or a table
- Outer query checks if a tuple in the outer query is within the inner query single value or table

Subquery check

- Different ways of checking:
  - Within the inner query set
  - Not within the inner query set
  - Against all members of the inner query set
  - Against any one member of the inner query set
  - Does the set exists

Set Membership

- (6 in {2,4,6}) = ?
- (5 in {2,4,6}) = ?
- (5 not in {2,4,6}) = ?

Set Comparison

- Attribute <comp> ANY/ALL (Subquery result)
- (5 < ANY {0,4,6}) = FALSE
- (5 < ALL {0,2,4}) = TRUE
- (5 = ANY {0,5}) = TRUE
- (5 != ANY {0,5}) = ?
- (5 = ALL {4,5}) = FALSE
- (5 != ALL {4,6}) = ?

Break

SQL: IN

- IN checks membership in a set
- The set may be specified by a subquery or declared in the query
- NOT IN check non-membership

The IN Operator

Conditions can contain IN for “element of”

- SELECT LastName, FirstName
FROM student
WHERE started IN (2010, 2013, 2014);

- SELECT LastName, FirstName
FROM student
WHERE started NOT IN (2010, 2013, 2014);

- SELECT Department, CourseName
FROM Course
WHERE Department IN ('CSC' , 'IT', 'IS');

Nesting Queries with IN

```
SELECT LastName, FirstName, SID
FROM student
WHERE SID IN
    (SELECT PresidentID
     FROM studentgroup)
```

Examples

List all students who are members of HerCTI.

Presidents who are members of their groups.

NOT IN: Set Complement

Example: Students who did not enroll in 2013.

```
SELECT LastName, FirstName, SID
FROM student
WHERE sid NOT IN (SELECT studentID
                  FROM enrolled
                  WHERE year = 2013);
```

Example ALL/ANY

List student who started first

```
SELECT LastName, FirstName, SID
FROM student
WHERE started >= ALL (SELECT started
                      FROM student);
```

List students if there was another student who started before them

```
SELECT LastName, FirstName, SID
FROM student
WHERE started >= ANY (SELECT started
                      FROM student);
```

Pop Quiz

```
SELECT LastName, FirstName, SID
FROM student
WHERE started >= ALL (SELECT started
                      FROM student);

SELECT LastName, FirstName, SID
FROM student
WHERE started = (SELECT max(started)
                 FROM student);
```

Exists

- Tests if a set is nonempty
- Lists students if they ever enrolled in a course

```
SELECT LastName, FirstName, sid
FROM student
WHERE EXISTS (SELECT *
              FROM enrolled
             WHERE sid = studentID);
```

- List students if they never enrolled in a course

```
SELECT LastName, FirstName, sid
FROM student
WHERE NOT EXISTS (SELECT *
                  FROM enrolled
                 WHERE sid = studentID);
```

Subqueries

- The result of one query may be needed by another to compute its result
- The subquery can appear in a SELECT, WHERE, and HAVING clause when they are guaranteed to return a single value result
- Subquery can appear in a FROM or WHERE clause when they return a table.

Returning a Single Value Result

When a single value is returned, it can be used like any other value on the right-hand side of a WHERE or HAVING condition

```
SELECT * FROM Student
 WHERE Started <
 (SELECT Max(Started) FROM Student);
```

Returning a Table in FROM clause

- List departments in which the average enrollment in courses is below 2

```
SELECT department, avg(cnt)
FROM
(SELECT department, courseid, count(studentid) as cnt FROM
    enrolled, course
   WHERE coursed = cid
    GROUP BY courseid, department) t
GROUP BY department
HAVING avg(cnt) < 2
```

Subqueries

- The result of one query may be needed by another to compute its result.
- There are two kinds of subqueries
  - Nested subqueries in which result of a inner subquery is used by outer subquery to compute result
  - **Correlated query in which each tuple of outer query runs aa loop for the inner subqery.**

Example

- Find the student who started earliest from ‘Chicago’

```
SELECT Student.*
FROM Student
WHERE started = (SELECT
min(started)
FROM student
WHERE city = ‘Chicago’)
```

- And then for ‘Naperville’, ’Evanston’, etc..

Example

```
SELECT Student.*
FROM Student
WHERE started = (SELECT min(started)
FROM student
WHERE city = ‘Chicago’)
UNION
SELECT Student.*
FROM Student
WHERE started = (SELECT min(started)
FROM student
WHERE city = ‘Naperville’)
UNION
(SELECT Student.*
FROM Student
WHERE started = (SELECT min(started)
FROM student
WHERE city = ‘Evanston’))
```

Correlated Subquery

```
SELECT Student.*
FROM Student S1
WHERE started =
(SELECT min(started)
FROM student S2
WHERE S2.city = S1.city)
```

Correlated Subqueries

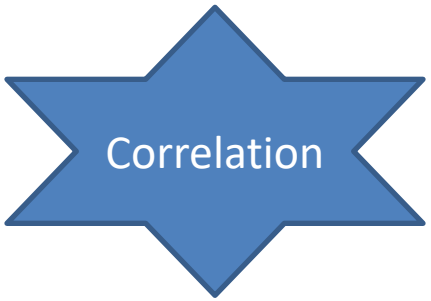
A subquery may refer to attributes of the table in the outer query

- The subquery will be evaluated repeatedly, once for each tuple in the table in the outer query
- Attributes from outer query table must be qualified with the table name if they appear in the subquery
- If the tables in the outer query and subquery are the same, must create an alias for the outer query table

Returning a Single Value Result

- On the LHS of a WHERE clause

```
SELECT LastName, FirstName, SID
FROM student
WHERE (SELECT count(*)
       FROM enrolled
       WHERE SID = StudentID) >= 2;
```



- As part of SELECT clause

```
SELECT LastName, FirstName, SID,
       (SELECT count(*)
        FROM enrolled
           WHERE SID = StudentID) AS EnrCrs
FROM student;
```

Practice

List members of HerCTI that are not enrolled in courses.

Courses not offered in 2013 (i.e. no record of anybody being enrolled).

Practice

- List the oldest studentgroup
- List students belonging to the first studentgroup

Practice

- List student groups that have both graduate and undergraduate members.
- List courses that have a unique number
- For all departments list the highest course number used by that department

Practice

- List students who are members of all studentgroups
- List students who have taken courses in all departments
- List students who have enrolled in courses every year that courses were offered