CS 540
Database Management Systems

Review of Relational Model and SQL

Oregon State University

Review problems: people betting on OSU football games

Out(<u>game</u>, <u>outcome</u>)   Bets(<u>who</u>, <u>outcome</u>, <u>game</u>, amt)

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

Some games have not been played yet, e.g., *Arizona*.

OSU Oregon State UNIVERSITY

Problem 1

List the completed games that nobody bet on.

game	outcome
USC	W
UCLA	L
Stanford	W

game

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

Problem 1

List the completed games that nobody bet on.

```
(Select Game
 From Out)
Except
 (Select Game
  From Bets)
```

Problem 2

Who bet the most money on a single game?

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

who	amt
Kevin	210

Problem 2

Who bet the most money on a single game?

```
Select Who, Amt
From Bets
Where Amt >= All
                (Select Amt
                 From Bets)
```

Problem 3

List the games that all bettors agree on.

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

game
Stanford
Arizona
UO
USC

OSU Oregon State UNIVERSITY

Problem 3

List the games that all bettors agree on.

```
(Select game
 From Bets)
 Except
(Select Bets1.game
 From Bets Bets1, Bets Bets2
 Where (Bets1.game = Bets2.game)
   And (Bets1.outcome <> Bets2.outcome))
```

Problem 4

For each game, the number of people betting on OSU to win and the number betting on OSU to lose.

game	outcome
USC	W
UCLA	L
Stanford	W

game	outcome	num
Stanford	W	1
UO	L	1
UCLA	W	1
UCLA	L	1
USC	W	1
Arizona	L	1

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

OSU Oregon State UNIVERSITY

Problem 4

For each game, the number of people betting on OSU to win and the number betting on OSU to lose.

```
Select game, outcome, Count(who) As num
From Bets
Group By game, outcome
```

Problem 5

Find the people who have made two or more bets on OSU to lose.

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

who
Kevin

Oregon State University

Problem 5

Find the people who have made two or more bets on OSU to lose.

```
Select who
From Bets
Where outcome = 'L'
Group By who
Having Count(outcome) >= 2
```

Problem 6

Who bet the most money overall?

game	outcome
USC	W
UCLA	L
Stanford	W

who	sumAmt
John	450

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

OSU Oregon State University

Problem 6

Who bet the most money overall?

```
Select who, Sum(amt) As sumAmt
From Bets
Group By who
Having Sum(amt) >= ALL
                    (Select Sum(amt)
                       From Bets
                       Group By who)
```

Problem 7

Who has bet on every game?

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

Who

Problem 7

Who has bet on every game?

```
Create View AllGames As
(Select  game From Out) Union
(Select game From Bets)

Select who
From Bets
Group By who
Having Count(Distinct game) =
(Select Count(Distinct game)
 From AllGames)
```

Problem 8

What games have won the most money for the people who bet on OSU to win?

game	outcome
USC	W
UCLA	L
Stanford	W

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

game
USC

Oregon State UNIVERSITY

Problem 8

What games have won the most money for the people who bet on OSU to win?

```
Create View Success-Win As
 (Select Bets.game, Sum(Bets.amt) As SumAmt
  From Bets, Out
  Where out.game = Bets.game And
        Bets.outcome = 'W' And Out.outcome = 'W'
         Group By Game)

Select Distinct game
From Success-Win
Where SumAmt >= All
                  (Select SumAmt
                   From Success-Win)
```

Problem 9

List the people who won some money so far.

game	outcome
USC	W
UCLA	L
Stanford	W

who
John
Kevin

who	outcome	game	amt
John	W	USC	200
John	W	UCLA	100
John	L	Arizona	150
Kevin	L	UO	210
Kevin	L	UCLA	50
Kevin	W	Stanford	120

Oregon State University

Problem 9

List the people who won some money so far.

```
Create View Success As
(Select who, Sum(amt) As pAmt
 From Bets, Out
 Where Out.game = Bets.game And Bets.outcome = Out.outcome
 Group By who)

Create View Failure As
(Select who, Sum(amt) As nAmt
 From Bets, Out
 Where Out.game = Bets.game And Bets.outcome <> Out.outcome
 Group By who)
```

Problem 9

List the people who won some money so far.

```
(Select Success.who
 From Success, Failure
 Where Success.who = Failure.who And
       Success.pAmt > Failure.nAmt )

Union
 (Select who
   From Success
    Where who not in
                    (Select who
                     From Failure) )
```

Query equivalency and containment

- Interesting and long-standing problems in query optimization.
- Queries q_1 and q_2 are *equivalent* if and only if for every database instance I, $q_1(I) = q_2(I)$
  - Shown as $q_1 \equiv q_2$
- Query q_1 is contained in q_2 if and only if for every database instance I, $q_1(I) \subseteq q_2(I)$
  - Shown as $q_1 \subseteq q_2$

Conjunctive queries (CQ)

- One datalog rule.
- SELECT-DISTINCT-FROM-WHERE.
- Select/project/join (σ, Π, ∞) fragment of RA.
- Existential/ conjunctive fragment of RC

- There is not any comparison operator ($<$, $\neq$, …) in CQ.
  - If used the family is called $CQ^{<}$, $CQ^{\neq}$, …

CQ examples

Movie(<u>mid</u>, title, year, total-gross)
Actor(<u>aid</u>, name, b-year)
Plays(<u>mid</u>, <u>aid</u>)

Actors who played in "LTR".

```
Q7(y):- Actor(x, y, z), Plays(t, x),
        Movie(t, 'LTR', w, f).
```

Non-CQ: Actors who played in some movies with only one actor.

Containment examples

Is $q_1 \subseteq q_2$?

```
q1(x):- R(x,y),R(y,z),R(z,w).
q2(x):- R(x,y),R(y,z).
```

```
q1(x):-R(x,y),R(y,'Joe').
q2(x):-R(x,y),R(y,z).
```

```
q1(x):- R(x,y),R(y,z),R(z,x).
q2(x):- R(x,y),R(y,x).
```

Oregon State University

Containment examples

Is $q_1 \subseteq q_2$?

```
q1(x):- R(x,y),R(y,y).
q2(x):- R(x,y),R(y,z),R(z,t).
```

Variables in a query

- The set of variables in q is shown as $var(q)$

- Example

  ```
  q1(x):- R(x,y),R(y,y)
  ```

  $var(q_1) = \{x,y\}$

Homomorphism

- A **homomorphism** $h : q_2 \rightarrow q_1$ is a function from var(q_2) to var(q_1) s.t. for each atom R(x, y, …) in q_1 there is an atom R(h(x), h(y), …) in q_2.
- h leaves the *constants* in q_2 *intact.*

- **Example**

```
q1(x):- R(x,y),R(y,z),R(z,w).
q2(x):- R(x,y),R(y,z).
```

We treat head variables, 'x', as constants, i.e., the same in q_1 and q_2.

$$h : q_2 \rightarrow q_1 \ : h(y) = y, h(z) = z.$$

Homomorphism Theorem

- Given CQs q_1 and q_2, we have $q_1 \subseteq q_2$ if and only if there exists a homomorphism $h : q_2 \rightarrow q_1$.

- Example:

```
q1(x):-R(x,y),R(y,z),R(z,w).
q2(x):-R(x,y),R(y,z).
```

  – Since $h : q_2 \rightarrow q_1$ is a homomorphism, we have $q_1 \subseteq q_2$.

Homomorphism examples

```
q1(x):-R(x,y),R(y,'Joe').
q2(x):-R(x,y),R(y,z).
```

$$h: q_2 \rightarrow q_1 : h(y) = y, h(z) = 'Joe' \Rightarrow q_1 \subseteq q_2$$

```
q1(x):- R(x,y),R(y,z),R(z,x).
q2(x):- R(x,y),R(y,x).
```

There is no homomorphism: $q_1 \not\subset q_2$

Homomorphism examples

Is $q_1 \subseteq q_2$?

```
q1(x):- R(x,y),R(y,y).
q2(x):- R(x,y),R(y,z),R(z,t).
```

$$h : q_2 \rightarrow q_1 : h(y) = y, h(z) = y, h(t) = y \Rightarrow q_1 \subseteq q_2$$

Oregon State University

Proof of Homomorphism Theorem

- Rules based form of a CQ

```
q(u) :- R_1(u_1),…,R_n(u_n).
```

  – u_i is shorthand for (x, y,…, z).

- **Valuation *v*** is a total function from a set of variables to the domain and identity on constants.

Proof of Homomorphism Theorem

- Recall that the set of variables in q is var(q)
  - Example.

    ```
    q1(x):- R(x,y),R(y,y).
    ```

    $var(q_1) = \{x,y\}$

- The result of q over database I is

  $q(I) = \{v(u) \mid v \text{ is a valuation over } var(q)\}$

- Also called the *image* of I under q

Proof of Homomorphism Theorem

- We have $q_1 \subseteq q_2$ if and only if there is a homomorphism $h : q_2 \rightarrow q_1$
- **Proof:**

```
q1(u):- R1(u1),…,Rn(un).
q2(u):- S1(w1),…,Sn(wn).
```

– If $h : q_2 \rightarrow q_1$ then $q_1 \subseteq q_2$

For each tuple t in q_1(I), there is a valuation v that maps variables in q_1 to I such that $v(u) = t$.

Thus, $h(v(u))$ maps variables in q_2 to I and $h(v(u)) = t$ where t *is in* q_2(I).

– If $q_1 \subseteq q_2$ then $h : q_2 \rightarrow q_1$

(Alice Book page 117)

Checking containment

- Check if there exists a homomorphism between queries.
- The problem is NP-complete, proved by reducing from 3-SAT.
- Since the size of queries are relatively small, the process is sufficiently fast.

Query minimization

- A conjunctive query q is minimal if for every other conjunctive query q', if $q' \equiv q$, q' has at least as many atoms as q.

- Example:

```
q1(x):- R(x,z),R(x,z).
q2(x):- R(x,z).
```

Query minimization algorithm

1. Remove an atom from q. Let's call new query q'.
2. We have $q \subseteq q'$.
3. Check to see if $q' \subseteq q$, if it is then remove atom permanently.

- Example:

```
q1(x):- R(x,z),R(z,t),R(x,w).
q2(x):- R(x,z),R(z,t).
```

We have a homomorphism from q_1 to q_2.

Larger families: UCQ

Movie(mid, title, year, total-gross)
Actor(aid, name, b-year)
Plays(mid, aid)

- CQ with union

Movies that were produced in 1998 or made more than $2,000.

```
Q1(y):- Movie(x,y,1998,z).
Q1(y):- Movie(x,y,z,t), t > 2000.
```

- We can extend homomorphism theorem for UCQs.

Homomorphism Theorem for UCQ

- Given UCQs $q_1 \cup \ldots \cup q_n$ and $q'_1 \cup \ldots \cup q'_m$, we have $q_1 \cup \ldots \cup q_n \subseteq q'_1 \cup \ldots \cup q'_m$ if and only if for every $i \leq n$ there is a $j \leq m$, such that $q_i \subseteq q_j$

- Thus, we can use apply homomorphism theorem to each CQ in a UCQ to check the containment.

- Containment checking for UCQs is NP-complete.
  – No worries, query size is usually small.

Oregon State University

Larger families: relational queries

- Containment checking for relational queries is undecidable.

- Proved using finite satisfiability problem:
  - Given a query, is there any (finite) database where the query as at least one answer.

Is SQL Sufficient?

- Using IsParent(parent,child), find grand children of a given person.

- Now find all descendants of a given person.

- It is **not** possible to write this query in standard SQL!
  - We can prove it.

Recursive queries in SQL

- SQL standards have recursive SQL
  - most database systems do not implement that
- Database systems usually support limited recursion
  - MySQL recursive cte, Oracle's connected by, …
- They define within a single query
  - a base case (base query)
  - a recursion step
- Systems limit the type of queries used for recursion
  - not group by/ aggregation function
  - to keep the plan for normal queries fast.

OSU Oregon State UNIVERSITY

Recursive queries in MySQL

- Common table expression (CTE)
  – relation variable within the scope of a single query

```
WITH cte (col1, col2) AS
(
    SELECT 1, 2
    UNION ALL
    SELECT 3, 4
)
SELECT col1, col2 FROM cte;
```

Oregon State University

Recursive queries in MySQL

- Common table expression (CTE)

```
WITH
   cte1 AS (SELECT a, b FROM table1),
   cte2 AS (SELECT c, d FROM table2)
SELECT b, d
   FROM cte1 JOIN cte2
   WHERE cte1.a = cte2.c;
```

- used similar to virtual view in the query
  - the query plan may be more efficient as each CTE is executed only once and used multiple times.

Oregon State University

Recursive queries in MySQL

- Recursive CTE

```
WITH RECURSIVE cte (n) AS
(
    SELECT 1
    UNION ALL
    SELECT n + 1 FROM cte WHERE n < 5
)
SELECT * FROM cte;
```

base case (base query) ?

recursion step ?

Recursive queries in MySQL

- Using *Employee(id, name, manager_id)* produce the organizational chart of the management chain.

```
WITH RECURSIVE employee_paths (id, name, path) AS
(
    SELECT id, name, CAST(id AS CHAR(200))
    FROM employees WHERE manager_id IS NULL
    UNION ALL
    SELECT e.id, e.name, CONCAT(ep.path, ',', e.id)
    FROM employee_paths AS ep
    JOIN employees AS e ON ep.id = e.manager_id
)
SELECT * FROM employee_paths ORDER BY path;
```

Oregon State University