NYU | TANDON SCHOOL OF ENGINEERING

CS6083: Principles of Database Systems

Writing Queries in SQL, RA, and RC

Topics:

- How to write queries using SQL, RA, RC
- How to move between SQL, RA, and RC
- Using nested subqueries
- Using set operations

Example Schema: Actor-Movie DB

Actor (aid, aname)

Movie (mid, mname, budget, gross)

ActedIn (aid, mid, starring, wage)

aid refs aid in actor, mid refs mid in movie

aid	aname

Actor

aid	mid	s	w

ActedIn

mid	mname	b	g

Movie

Domain-Relational Calculus:

- Query: ID of the actor with name "Spacey"

Domain-Relational Calculus:

- Query: ID of the actor with name “Spacey”

{ <aid> | Ex an (<aid, an> in ACTOR and
an = “Spacey”) }

Note: we use
Ex “the exists”
Al “for all”
in “element of”
and, or, not

Domain-Relational Calculus:

- Query: ID of the actor with name “Spacey”

{ <aid> | Ex an (<aid, an> in ACTOR and
an = “Spacey”) }

or in shorter form

{ <aid> | <aid, “Spacey”> in ACTOR }

Another Query:

- ID and name of actors appearing in T2

" ID AND NAME OF ACTORS WHO APPEARED IN T2 "

$$\{ \langle aid, an \rangle \mid \langle aid, an \rangle \in ACTOR \wedge \exists\, mid, s, w (\langle aid, mid, s, w \rangle \in ACTED\_IN \wedge \exists\, b, g (\langle mid, \text{"T2"}, b, g \rangle \in MOVIE)) \}$$

Another Query:

- ID and name of actors appearing in T2
- How about “and where wage > $500”?
- Or how about actors not appearing in T2?

COMPARE TO RELATIONAL ALGEBRA:

$$\pi_{AID,\,ANAME}\left(\sigma_{MNAME = \text{"T2"}}\left(ACTOR \bowtie \left(ACTED\_IN \bowtie MOVIE\right)\right)\right)$$

$$\text{JOIN} \quad \text{VS.} \quad \exists x,y\left(\langle x, y, \ldots\rangle \in \ldots\right)$$

" ID OF ACTOR WHO HAS APPEARED
IN EVERY MOVIE "

" ID OF ACTOR WHO HAS APPEARED IN EVERY MOVIE "

$$\{\langle aid\rangle \mid \nexists\, mid, mn, b, g\,\big(\langle mid, mn, b, g\rangle \in MOVIE \wedge \nexists\, s, w\,(\langle aid, mid, s, w\rangle \in ACTED\_IN)\big)\}$$

OR

$$\{\langle aid\rangle \mid \forall\, mid, mn, b, g\,\big(\langle mid, mn, b, g\rangle \in MOVIE \Rightarrow \exists\, s, w\,(\langle aid, mid, s, w\rangle \in ACTED\_IN)\big)\}$$

RELATIONAL ALGEBRA :

RELATIONAL ALGEBRA:

$$\left(\pi_{aid,\,mid}(\text{ACTED\_IN})\right) \div \left(\pi_{MID}(\text{MOVIE})\right)$$

Set Operations in SQL:

- Union, Intersect, Except

- In, not in, some, all, exists, unique

  (used in nested queries)

- “Actors who appeared in Star Wars I and II”

Set Operations in SQL:

- "Actors who appeared in Star Wars I and II"

```
( SELECT A.name
  FROM Actor A, ActedIn AI, Movie M
  WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
               M.title = "Star Wars I" )
INTERSECT
( SELECT A.name
  FROM Actor A, ActedIn AI, Movie M
  WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
               M.title = "Star Wars II" )
```

Set Operations in SQL:

- "Actors who appeared in Star Wars I and II"

```
( SELECT A.name
  FROM Actor A, ActedIn AI, Movie M
  WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
               M.title = "Star Wars I" )
INTERSECT
( SELECT A.name
  FROM Actor A, ActedIn AI, Movie M
  WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
               M.title = "Star Wars II" )
```

How does result change if use UNION or EXCEPT?

Set Operations in SQL:

- **Another solution:**

```
SELECT A.name
FROM Actor A, ActedIn AI, Movie M
WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
              M.title = “Star Wars I” AND A.aid IN

  ( SELECT A.aid
    FROM Actor A, ActedIn AI, Movie M
    WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
                  M.title = “Star Wars II” )
```

NYU | TANDON SCHOOL OF ENGINEERING

Set Operations in SQL:

- **Another solution:**

```
SELECT A.name
FROM Actor A, ActedIn AI, Movie M
WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
             M.title = “Star Wars I” AND A.aid IN

  ( SELECT A.aid
    FROM Actor A, ActedIn AI, Movie M
    WHERE  A.aid = AI.aid  AND  AI.mid = M.mid AND
                 M.title = “Star Wars II” )
```

What if we use “NOT IN”?

Using Nested Query:

- "Actors who received the highest wage ever"

```
SELECT A.name
FROM  Actor A, ActedIn AI
WHERE A.aid = AI.aid AND
            AI.wage >= ALL  ( SELECT AI.wage
                              FROM ActedIn AI )
```

How about if we use "SOME" ?

Another Solution:

- "Actors who received the highest wage ever"

```
SELECT A.name
FROM  Actor A, ActedIn AI
WHERE A.aid = AI.aid AND
            AI.wage = ( SELECT MAX(AI.wage)
                        FROM ActedIn AI )
```

Yet Another Solution:

- "Actors who received the highest wage ever"

```
SELECT A.name
FROM  Actor A, ActedIn AI
WHERE A.aid = AI.aid AND
             NOT EXIST ( SELECT AI.wage)
                         FROM ActedIn AI2
                         WHERE AI2.wage > AI.wage )
```

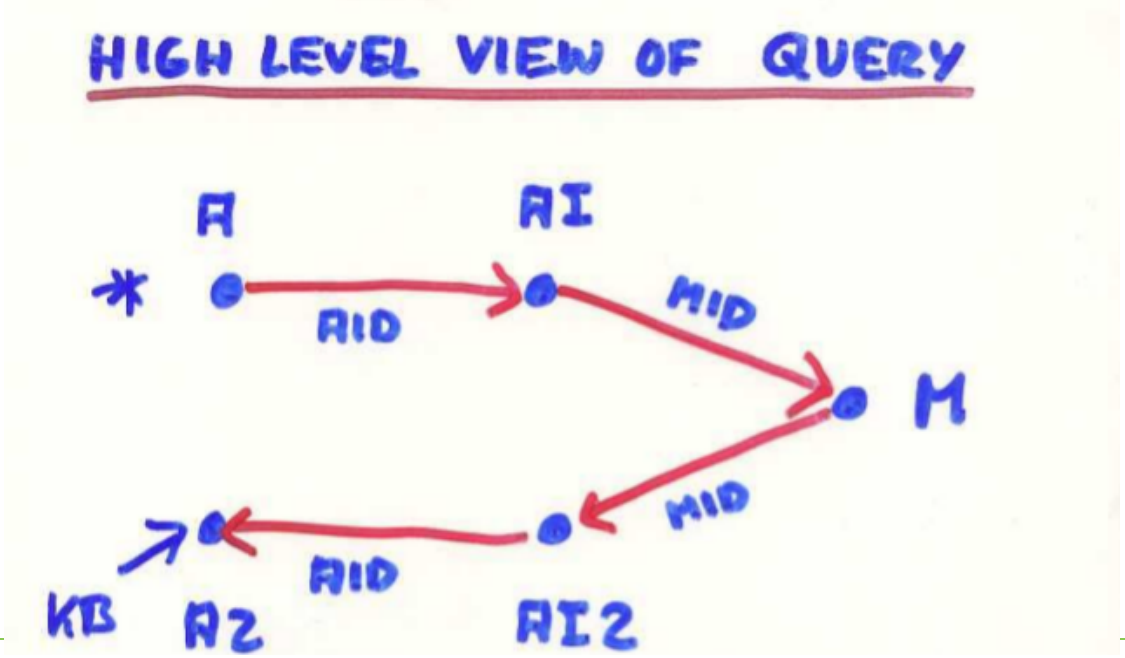
Why is this not a good way to write this query?

Query Graphs, RC, RA, and SQL:

- "Actors who acted in a movie with Kevin Bacon where budget was > 1 million"

Query Graphs, RC, RA, and SQL:

- "Actors who acted in a movie with Kevin Bacon where budget was > 1 million"



"IN A"

"FROM A TO AI

$$\langle NAME \mid \exists AID \Big((NAME, AID) \in A \wedge \exists MID$$

"FROM AI TO M"

$$\Big((AID, MID) \in AI \wedge \exists MNAME, B, G$$

$$\Big((MNAME, MID, B, G) \in M \wedge B > 1000000 \wedge$$

"FROM M TO AI2"

"FROM AI2 TO A2

$$\exists AID2 \Big(\big((AID2, MID) \in AI \wedge \exists NAME2$$

$$\big((NAME2, AID2) \in A \wedge NAME2 = \text{"K.BACON"} \big) \big) \Big) \Big) \Big) \rangle$$

Query Graphs, RC, RA, and SQL:

- "Actors who acted in a movie with Kevin Bacon where budget was > 1 million"

  - how about using RA?

  - how about using SQL?

Query Graphs, RC, RA, and SQL:

- "Actors who acted in a movie with Kevin Bacon where budget was > 1 million"

  - how about actors who acted with an actor who has acted with Kevin Bacon?

  - how about distance 3, 4, infinity?

  - transitive closure limitation