CSE 344

JULY 2ND

DATALOG

ADMINISTRATIVE MINUTIAE

- No class Wednesday

- HW2 Due Wednesday

RELATIONAL ALGEBRA

Set-at-a-time algebra, which manipulates relations

In SQL we say ***what*** we want

In RA we can express ***how*** to get it

Every DBMS implementations converts a SQL query to RA in order to execute it

An RA expression is called a ***query plan***

BASICS

- Relations and attributes
- Functions that are applied to relations
  - Return relations
  - Can be composed together
  - Often displayed using a tree rather than linearly
  - Use Greek symbols as shorthand:
    - union $\cup$ and difference –
    - selection σ
    - projection π
    - cartesian product $\times$
    - natural join $\bowtie$

NATURAL JOIN

$$R1 \bowtie R2$$

Meaning: $R1 \bowtie R2 = \Pi_A(\sigma_\theta(R1 \times R2))$

- Returns a relation where all attribute names are unambiguous

Where:

- Selection σ_θ checks equality of all common attributes (i.e., attributes with same names)
- Projection Π_A eliminates duplicate common attributes

NATURAL JOIN EXAMPLE

R

A	B
X	Y
X	Z
Y	Z
Z	V

S

B	C
Z	U
V	W
Z	V

R ⋈ S =
$\Pi_{ABC}(\sigma_{R.B=S.B}(R \times S))$

A	B	C
X	Z	U
X	Z	V
Y	Z	U
Y	Z	V
Z	V	W

NATURAL JOIN EXAMPLE 2

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
Alice	54	98125
Bob	20	98120

P $\bowtie$ V

age	zip	disease	name
54	98125	heart	Alice
20	98120	flu	Bob

AnonPatient (age, zip, disease)
Voters (name, age, zip)

THETA JOIN

A join that involves a predicate

$$R1 \bowtie_{\theta} R2 = \sigma_{\theta} (R1 \times R2)$$

Here θ can be any condition

No projection in this case!

For our voters/patients example:

$$P \bowtie_{P.zip = V.zip \text{ and } P.age >= V.age -1 \text{ and } P.age <= V.age +1} V$$

EQUIJOIN

A theta join where θ is an equality predicate

$$R1 \bowtie_{\theta} R2 \quad = \sigma_{\theta} (R1 \times R2)$$

By far the most used variant of join in practice

What is the relationship with natural join?

EQUIJOIN EXAMPLE

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age} V$

P.age	P.zip	P.disease	V.name	V.age	V.zip
54	98125	heart	p1	54	98125
20	98120	flu	p2	20	98120

JOIN SUMMARY

Theta-join: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$

- Join of R and S with a join condition θ
- Cross-product followed by selection θ
- No projection

Equijoin: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$

- Join condition θ consists only of equalities
- No projection

Natural join: $R \bowtie S = \pi_A(\sigma_{\theta}(R \times S))$

- Equality on **all** fields with same name in R and in S
- Projection π_A drops all redundant attributes

MORE JOINS

Outer join

- Include tuples with no matches in the output
- Use NULL values for missing attributes
- Does not eliminate duplicate columns

Variants

- Left outer join
- Right outer join
- Full outer join

SOME EXAMPLES

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)
```

Name of supplier of parts with size greater than 10

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10} (Part))$

Name of supplier of red parts or parts with size greater than 10

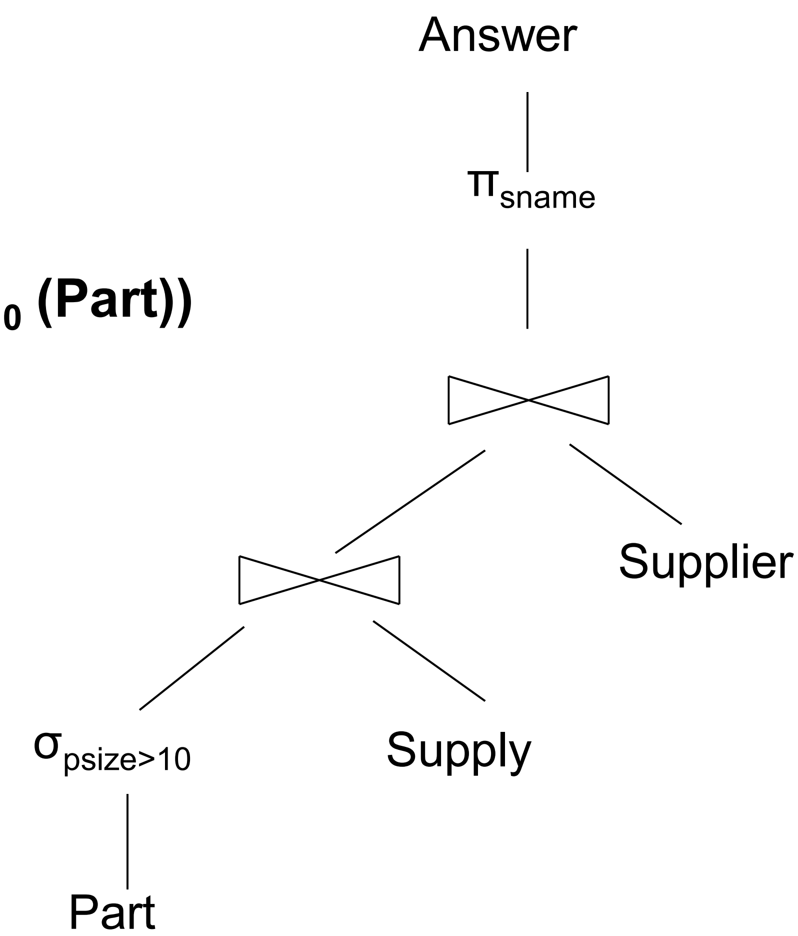
$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10} (Part) \cup \sigma_{pcolor='red'} (Part)))$

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10 \vee pcolor='red'} (Part)))$

Can be represented as trees as well

REPRESENTING RA QUERIES AS TREES

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}))$



RELATIONAL ALGEBRA OPERATORS

Union ∪, ~~intersection ∩,~~ difference -

Selection σ

Projection π

Cartesian product X, join ⋈

(Rename ρ)

RA

Duplicate elimination δ

Grouping and aggregation γ

Sorting τ

Extended RA

All operators take in 1 or more relations as inputs and return another relation

EXTENDED RA: OPERATORS ON BAGS

Duplicate elimination δ

Grouping γ

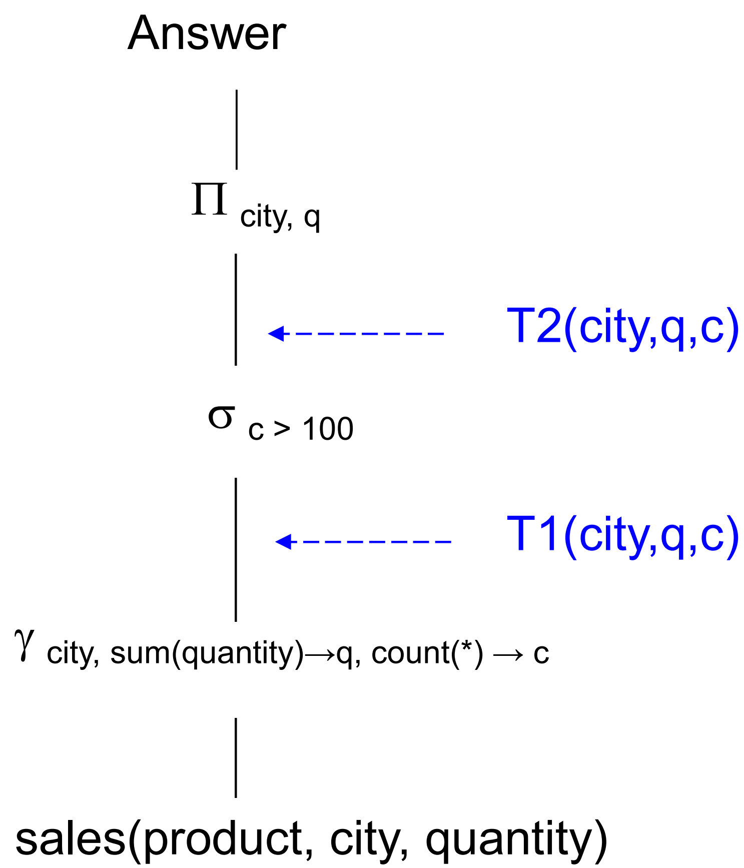
- Takes in relation and a list of grouping operations (e.g., aggregates). Returns a new relation.

Sorting τ

- Takes in a relation, a list of attributes to sort on, and an order. Returns a new relation.

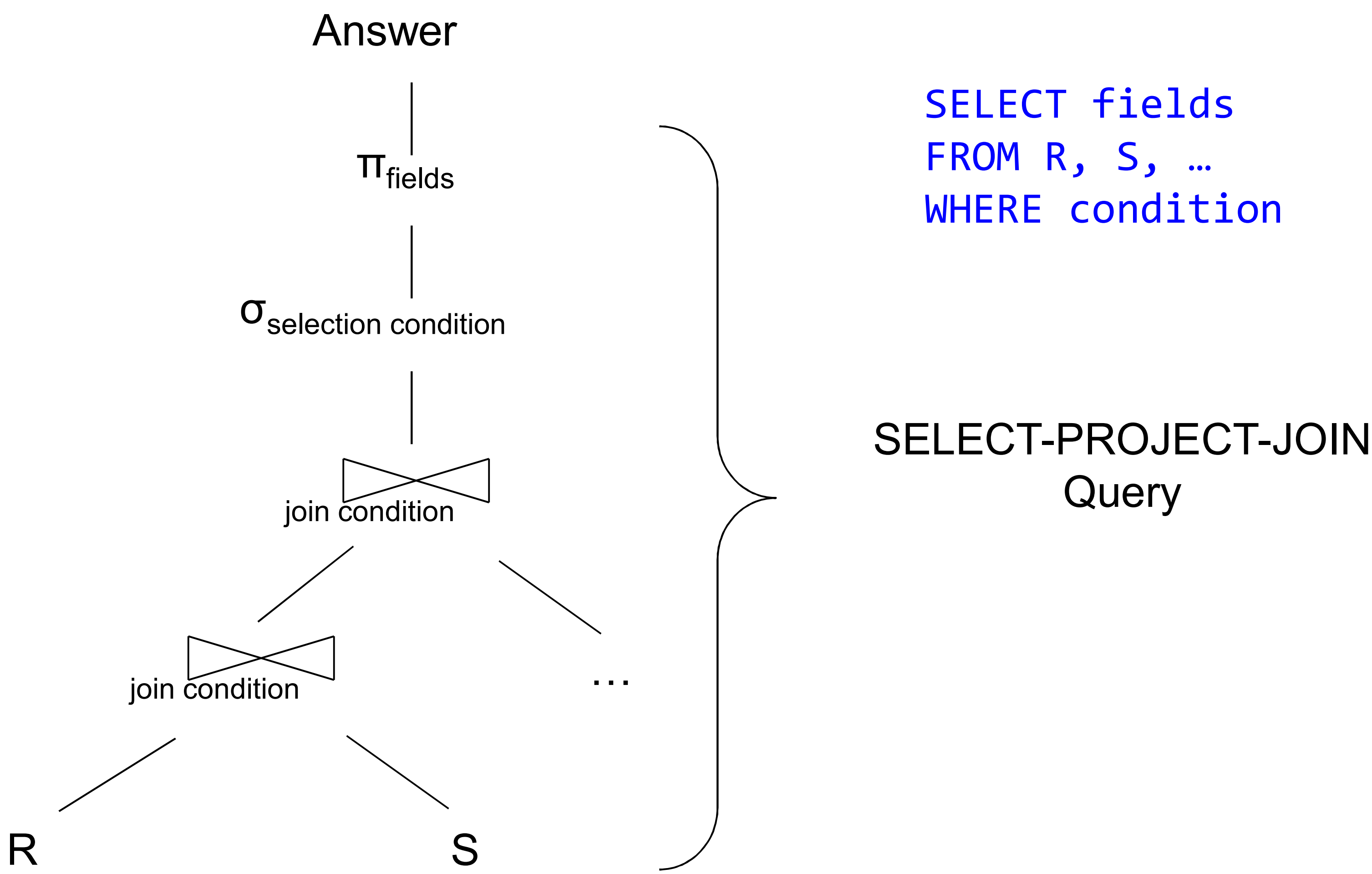
USING EXTENDED RA OPERATORS

```
SELECT city, sum(quantity)
FROM sales
GROUP BY city
HAVING count(*) > 100
```



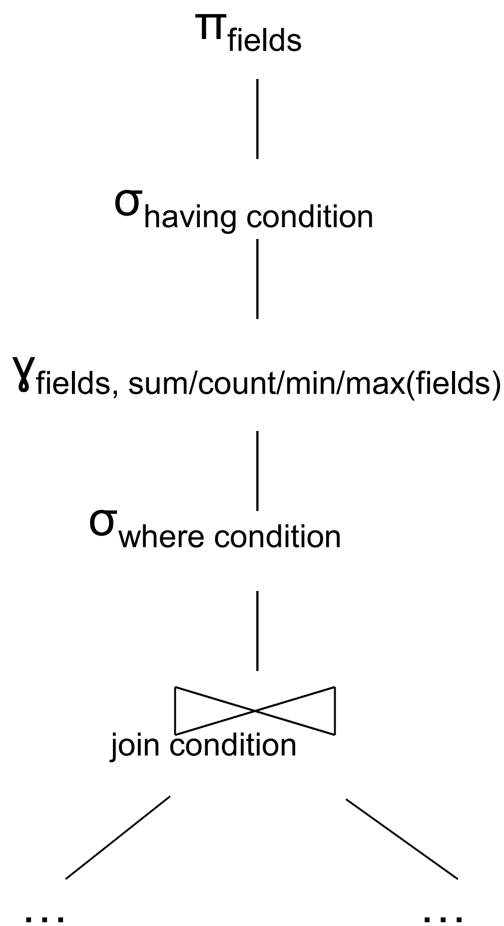
T1, T2 = temporary tables

TYPICAL PLAN FOR A QUERY (1/2)




```
SELECT fields
FROM R, S, …
WHERE condition
```

TYPICAL PLAN FOR A QUERY (1/2)




```
SELECT fields
FROM R, S, …
WHERE condition
GROUP BY fields
HAVING condition
```

HOW ABOUT SUBQUERIES?

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = ‘WA’
  and not exists
  (SELECT *
   FROM Supply P
   WHERE P.sno = Q.sno
         and P.price > 100)
```

HOW ABOUT SUBQUERIES?

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
  and not exists
  (SELECT *
   FROM Supply P
   WHERE P.sno = Q.sno
         and P.price > 100)
```

Correlation !

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

HOW ABOUT SUBQUERIES?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
  and not exists
  (SELECT *
   FROM Supply P
   WHERE P.sno = Q.sno
         and P.price > 100)
```

De-Correlation

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
  and Q.sno not in
  (SELECT P.sno
   FROM Supply P
   WHERE P.price > 100)
```

HOW ABOUT SUBQUERIES?

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

Un-nesting

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'WA')
    EXCEPT
 (SELECT P.sno
   FROM Supply P
   WHERE P.price > 100)
```

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
  and Q.sno not in
  (SELECT P.sno
   FROM Supply P
   WHERE P.price > 100)
```

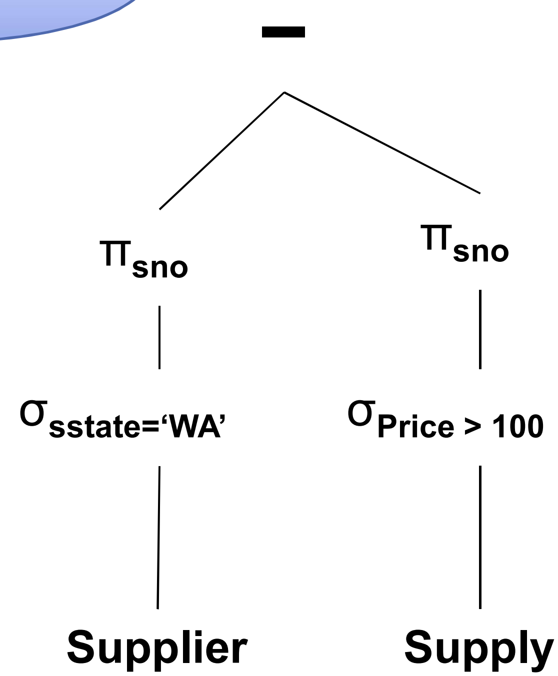
EXCEPT = set difference

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

HOW ABOUT SUBQUERIES?

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'WA')
    EXCEPT
 (SELECT P.sno
   FROM Supply P
   WHERE P.price > 100)
```

Finally…



SUMMARY OF RA AND SQL

SQL = a declarative language where we say *what* data we want to retrieve

RA = an algebra where we say *how* we want to retrieve the data

Theorem: SQL and RA can express exactly the same class of queries

RDBMS translate SQL → RA, then optimize RA

RELATIONAL ALGEBRA TAKEAWAYS

- **For a given query, be able write the equivalent relational algebra expression**
- **Given a relational algebra expression, write the equivalent query**
- **Understand what each are trying to get semantically**

SUMMARY OF RA AND SQL

SQL (and RA) cannot express ALL queries that we could write in, say, Java

Example:

- Parent(p,c):   find all descendants of ‘Alice’
- No RA query can compute this!
- This is called a *recursive query*

***Datalog* is an extension that can compute recursive queries**

WHAT IS DATALOG?

Another query language for relational model

- Designed in the 80's
- Simple, concise, elegant
- Extends relational queries with <u>*recursion*</u>

Relies on a logical framework for "record" selection

DATALOG: FACTS AND RULES

Facts = tuples in the database

Rules = queries

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

Schema

DATALOG: FACTS AND RULES

Facts = tuples in the database

Rules = queries

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

Find Movies made in 1940

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
             Movie(x,y,'1940').
```

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
              Movie(x,y,'1940').
```

Find Actors who acted in Movies made in 1940

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
             Movie(x,y,'1940').
```

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                         Casts(z,x2), Movie(x2,y2,1940)
```

DATALOG: FACTS AND RULES

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
             Movie(x,y,'1940').
```

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                         Casts(z,x2), Movie(x2,y2,1940)
```

Find Actors who acted in a Movie in 1940 and in one in 1910

DATALOG: FACTS AND RULES

Facts = tuples in the database

Rules = queries

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

```
Q1(y) :-  Movie(x,y,z), z='1940'.
```

```
Q2(f, l) :-  Actor(z,f,l), Casts(z,x),
               Movie(x,y,'1940').
```

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                    Casts(z,x2), Movie(x2,y2,1940)
```

Extensional Database Predicates = EDB = Actor, Casts, Movie

Intensional Database Predicates = IDB = Q1, Q2, Q3