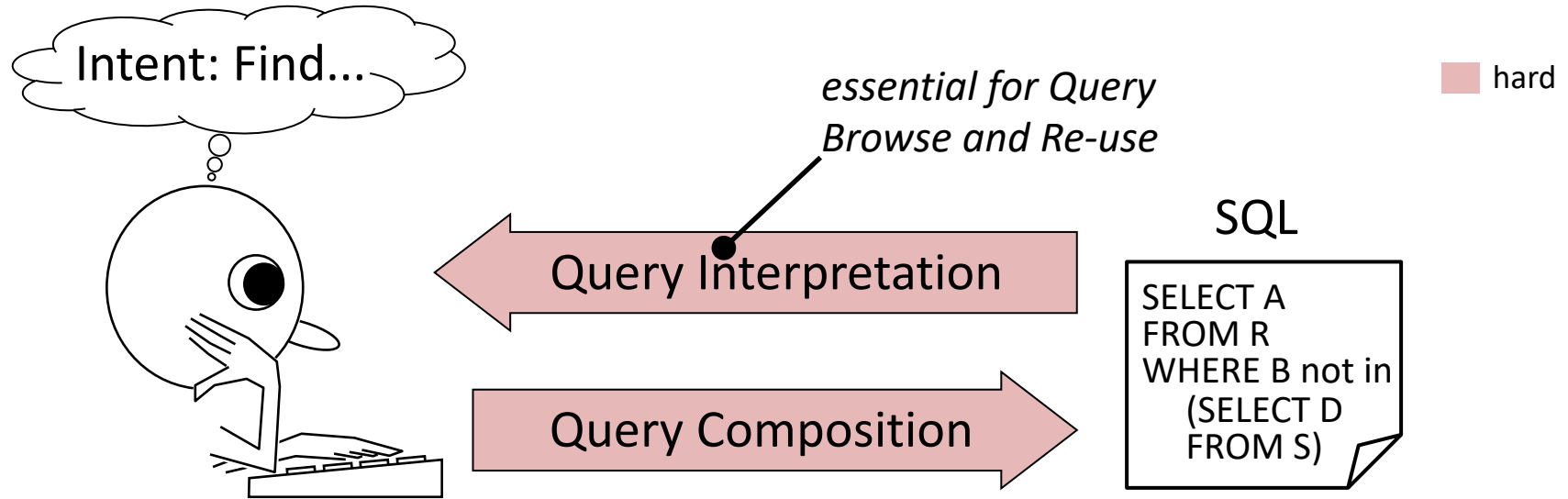


Databases will visualize queries too

Wolfgang Gatterbauer

Aug 30, 2011 (VLDB'11)

Interactions between Users and Queries are hard



Recent work on **Query Management**:
Idea: Re-use and adapt existing queries

CQMS [Khoussainova+ \[CIDR'09\]](#)
SQL QuerIE [Chatzopoulou+ \[SSDBM'09\]](#)
SQLshare [Howe+ \[MS eSc WS'10\]](#)
DBease [Li+ \[CIDR'11\]](#)

Problem:

Query Interpretation is hard too!

even used for testing purposes,
e.g., on www.gradiance.com

Browsing & Understanding existing Queries is hard

hard

5

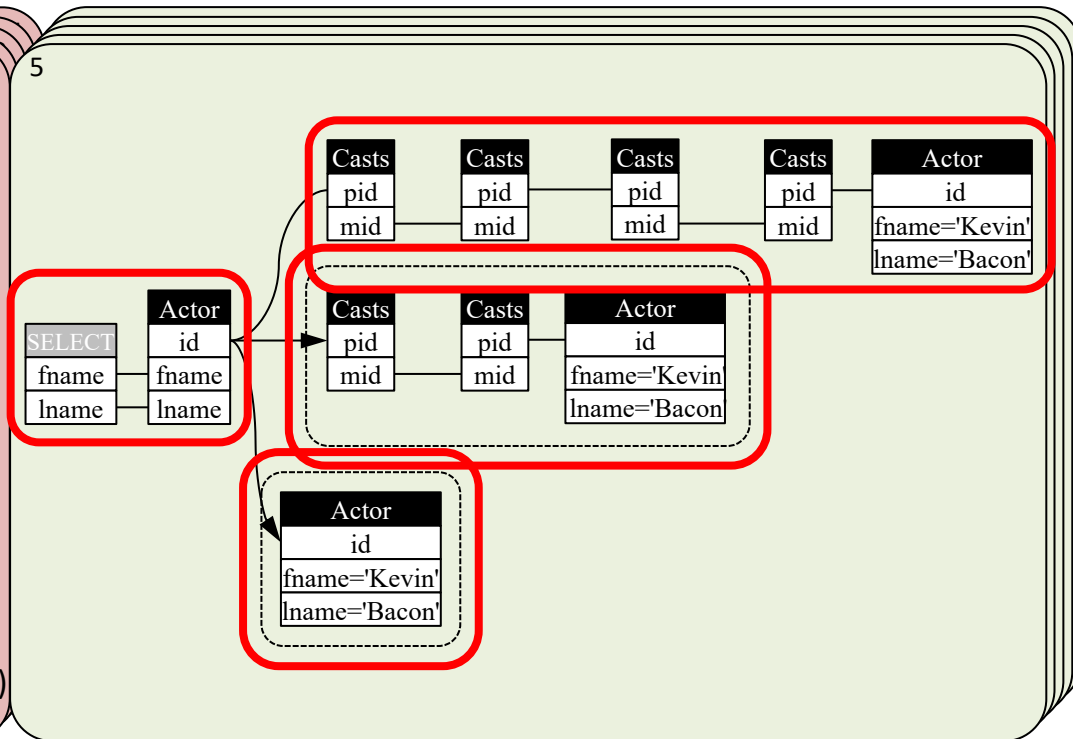
```
select distinct a3.fname, a3.lname
from Actor a0, Casts c0, Casts c1, Casts c2, Casts c3,
     Actor a3
where a0.fname = 'Kevin' and a0.lname = 'Bacon'
and c0.pid = a0.id and c0.mid = c1.mid
and c1.pid = c2.pid and c2.mid = c3.mid
and c3.pid = a3.id
and not exists (select xc1.pid
                from Actor xa0, Casts xc0, Casts xc1
                where xa0.fname = 'Kevin' and xa0.lname = 'Bacon'
                and xa0.id = xc0.pid and xc0.mid = xc1.mid
                and xc1.pid = a3.id)
and not exists (select ya0.id
                from Actor ya0
                where ya0.fname = 'Kevin' and ya0.lname = 'Bacon')
```

Can Query Visualization help?

easy
hard

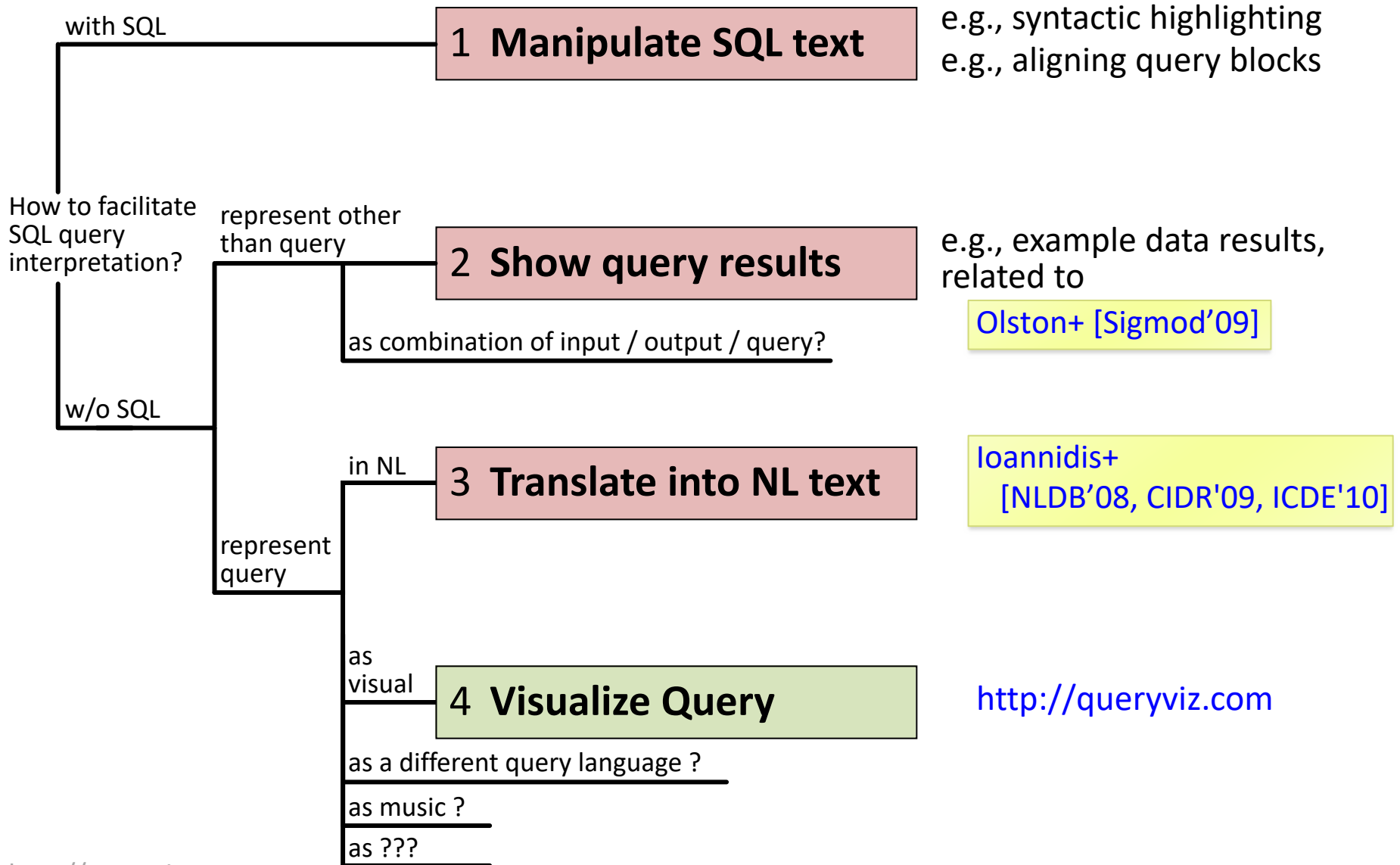
5

```
select distinct a3.fname, a3.lname
from Actor a0, Casts c0, Casts c1, Casts c2, Casts c3,
      Actor a3
where a0.fname = 'Kevin' and a0.lname = 'Bacon'
and c0.pid = a0.id and c0.mid = c1.mid
and c1.pid = c2.pid and c2.mid = c3.mid
and c3.pid = a3.id
and not exists (select xc1.pid
                from Actor xa0, Casts xc0, Casts xc1
                where xa0.fname = 'Kevin' and xa0.lname = 'Bacon'
                and xa0.id = xc0.pid and xc0.mid = xc1.mid
                and xc1.pid = a3.id)
and not exists (select ya0.id
                from Actor ya0
                where ya0.fname = 'Kevin' and ya0.lname = 'Bacon')
```



Query Intent: "Find all actors with Bacon number 2."

Four principal ways for Query Interpretation

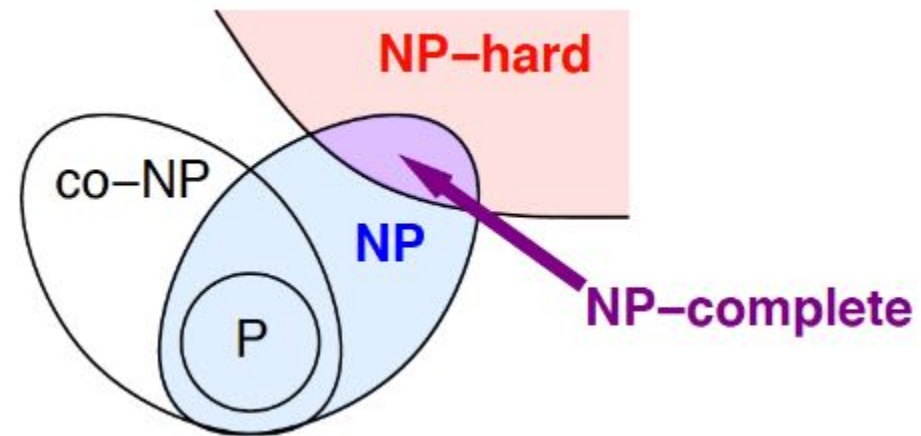


"A picture is worth 1000 words"

Text

"... **P** is the set of problems that can be solved quickly... **NP** is the set of decision problems where we can verify a YES answer quickly if we have the solution in front of us... A problem is **NP-hard** if a polynomial-time algorithm for would imply a polynomial-time algorithm for every problem in **NP**... If the answer to a problem in **co-NP** is NO, then there is a proof of this fact that can be checked in polynomial time...a problem is **NP-complete** if it is both NP-hard and an element of NP."

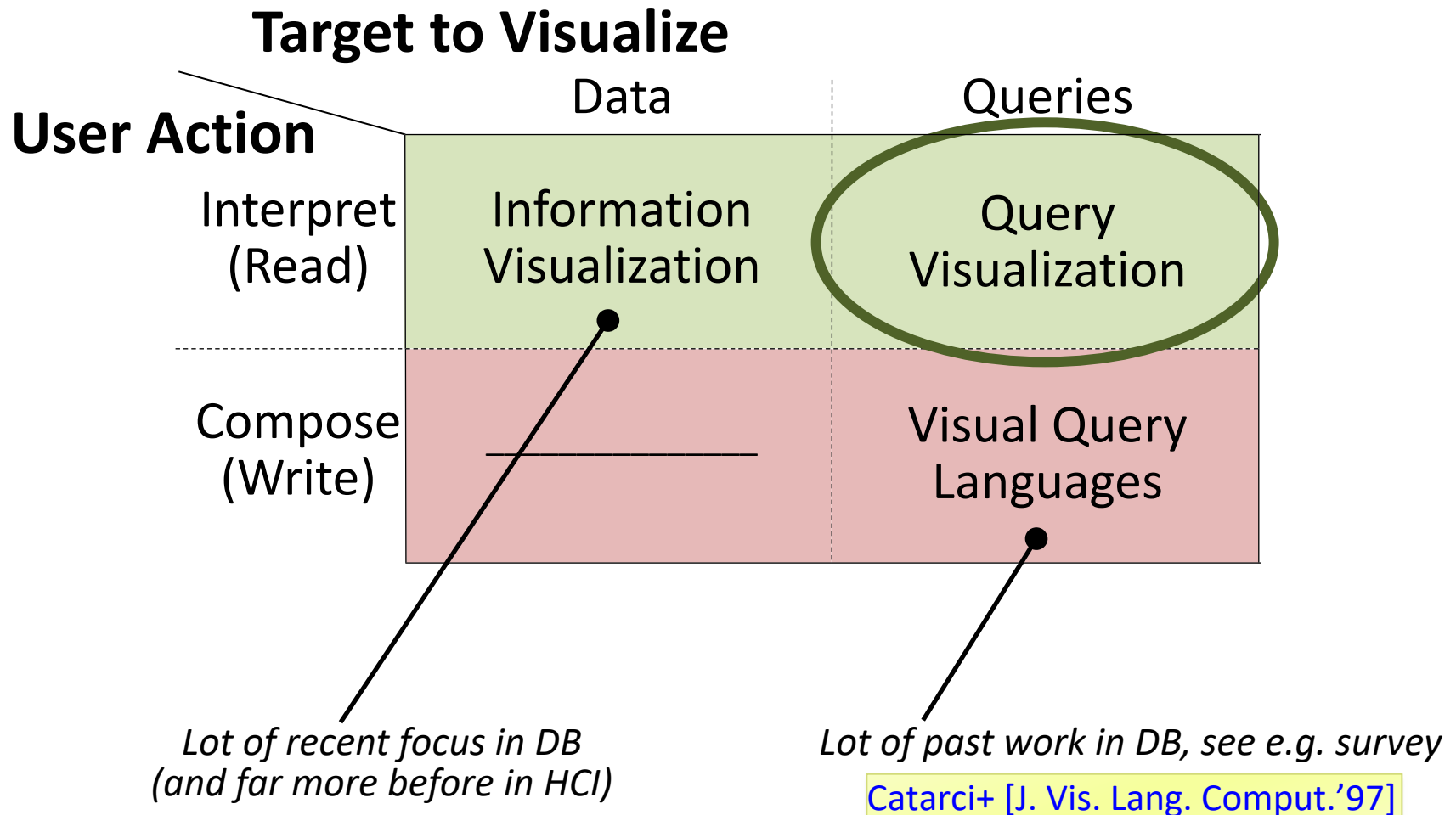
Visual



"...what we think the world looks like" according to [Erickson \[lecture notes'09\]](#)

Query Visualization vs. Visual Query Languages

easy
hard



Visual Interpretation vs. Visual Composition


easy
hard

		Communication Medium	
		Text	Visual (graphics)
User Action	Interpret (Read)	Sequential	Parallel
	Compose (Write)	Sequential	Sequential

Suggestion (why VQL have not found wide adoption):
Visual composition is an inherently sequential process.
In contrast, Query Visualization is different and can use the full bandwidth of human visual perception.

The Challenge

Find the appropriate visual alphabet which

- (i) allows users to **quickly understand** a query's intent,*
 - (ii) can be **easily learned** by users, and*
 - (iii) can **express** a large fraction of SQL.*
- 
- goals*

Additionally, find

- (iv) **automatic translations** from SQL to the visualization.*

Agenda

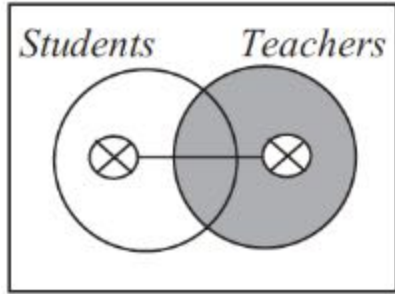
1) Why Query Visualization?

2) The Development of QueryViz

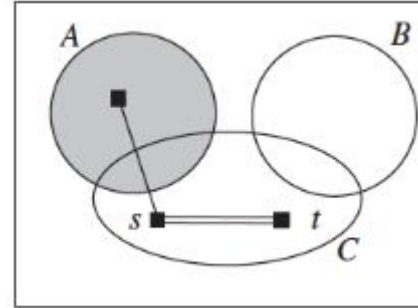
(joint work with Jonathan Danaparamita)

Inspiration from Diagrammatic Reasoning

Diagrammatic reasoning systems [Howse \[ICCS'08\]](#)



"There is an element that is a student or a teacher but not both, and the set of Teachers is empty."



"s represents an individual in the set $C - A \cap B$ iff s represents the same individual as t."

Inspired by Euler graphs, Venn diagrams, existential graphs by C. Sanders Pierce

Idea: use topological properties, such as enclosure, to represent logical expressions and set-theoretic relationships

First-Order Logic (FOL) representation of SQL

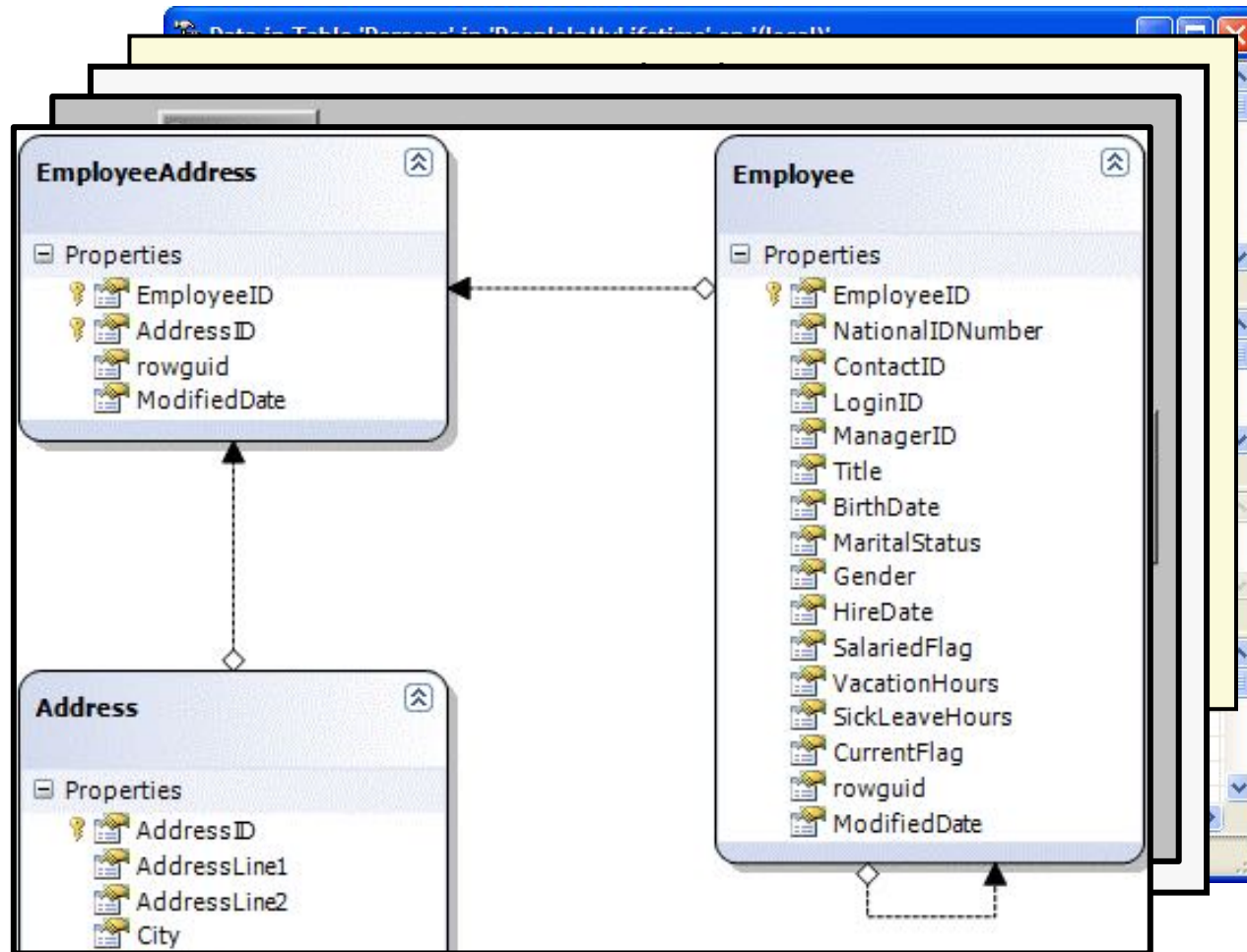
```
select a
from R
where b not IN
  (select d
   from S)
```

SQL \rightarrow FOL

$a: \exists b. \exists c. [R(a,b,c) \wedge \neg(\exists d. \exists e. [S(d,e) \wedge b=d])]$

Design decision 1: start from FOL representation of SQL queries and represent it with topological properties

DB schemas as familiar visual construct



Design decision 2: start from known visual UML metaphors for DB schemas

Incremental Complexity

Design decision 3: gradually extend known visual metaphors for CQs

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, price)

```
select  F.person
from    Frequents F, Likes L, Serves S
where   F.person = L.person
and     F.bar = S.bar
and     L.drink = S.drink
```

Unlike SQL: no aliases needed; schema implicit

Unlike Datalog: no anonymous variables shown

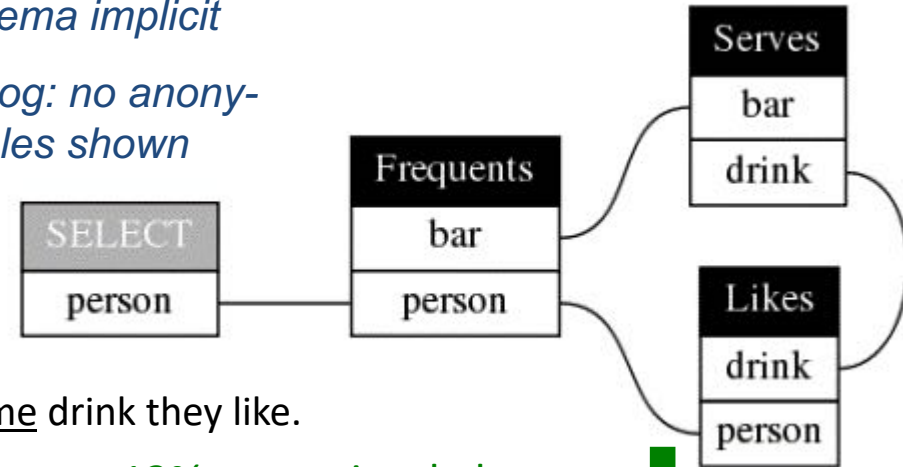
$Q(x) :- \text{Frequents}(x, y), \text{Serves}(y, z, _), \text{Likes}(x, z)$

Q: Find persons that frequent some bar that serves some drink they like.

+167% more SQL text

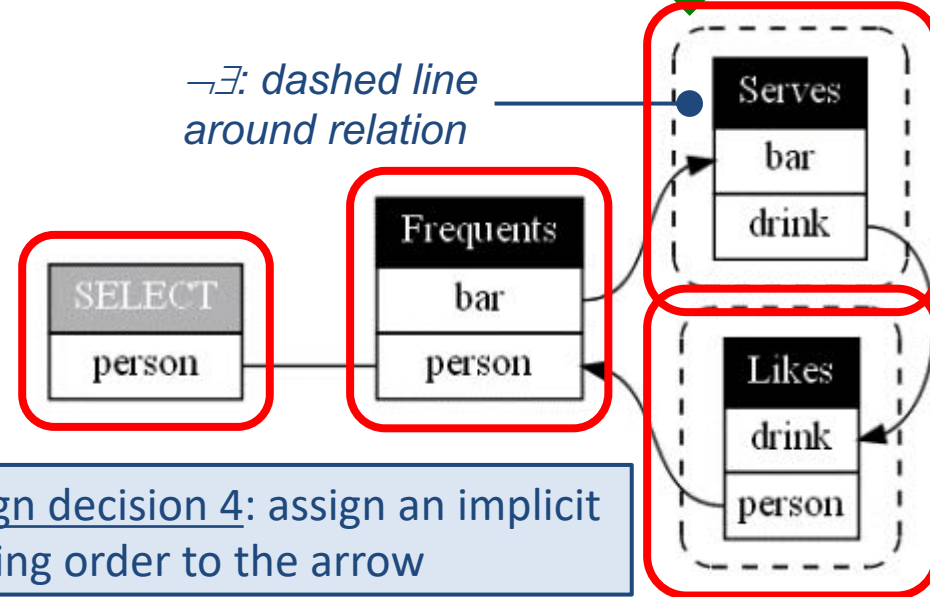
```
select  F.person
from    Frequents F
where   not exists
        (select  S.drink
         from    Serves S
         where   S.bar = F.bar
         and     not exists
                 (select  L.drink
                  from    Likes L
                  where   L.person = F.person
                  and     S.drink = L.drink))
```

Q: Find persons that frequent some bar that serves only drinks they like.



+13% more visual elements

$\neg\exists$: dashed line around relation



Design decision 4: assign an implicit reading order to the arrow

Logical transformations

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, price)

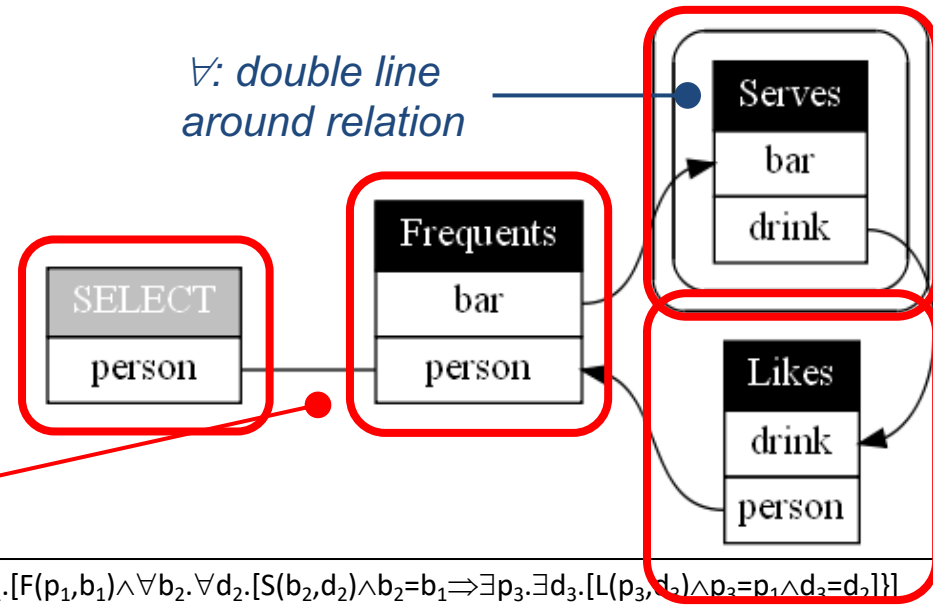
Design decision 5: limited logical transformation can further simplify representation

Q: Find persons that frequent a bar so that they like all drinks served.

Q: Find persons that frequent some bar so that there is no drink served that the person does not like.

```
select  F.person
from    Frequents F
where   not exists
        (select  S.drink
         from    Serves S
         where   S.bar = F.bar
         and     not exists
                 (select  L.drink
                  from    Likes L
                  where   L.person = F.person
                  and     S.drink = L.drink))
```

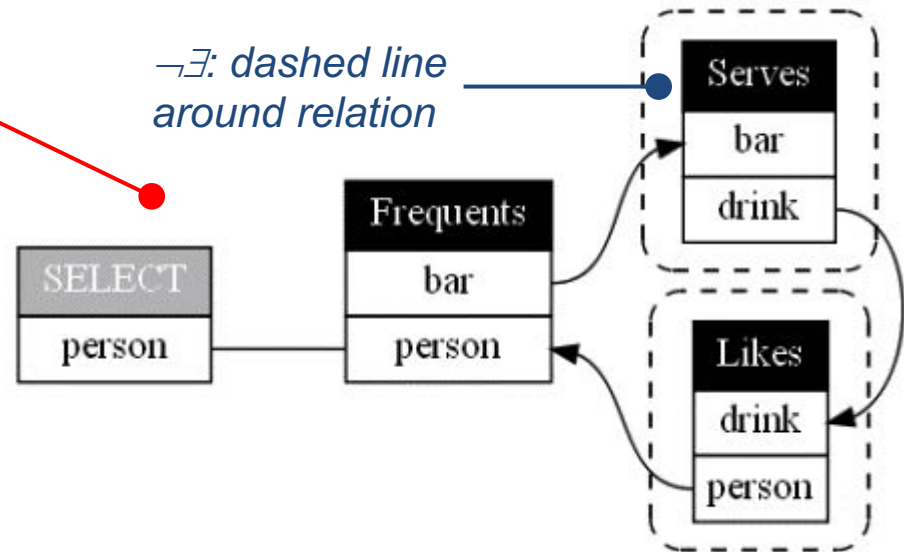
\forall : double line around relation



$p_1: \exists b_1. [F(p_1, b_1) \wedge \forall b_2. \forall d_2. [S(b_2, d_2) \wedge b_2 = b_1 \Rightarrow \exists p_3. \exists d_3. [L(p_3, d_3) \wedge p_3 = p_1 \wedge d_3 = d_2]]]$

$p_1: \exists b_1. [F(p_1, b_1) \wedge \neg (\exists b_2. \exists d_2. [S(b_2, d_2) \wedge b_2 = b_1 \wedge \neg (\exists p_3. \exists d_3. [L(p_3, d_3) \wedge p_3 = p_1 \wedge d_3 = d_2]])]$

$\neg \exists$: dashed line around relation



Q: Find persons that frequent some bar that serves only drinks they like.

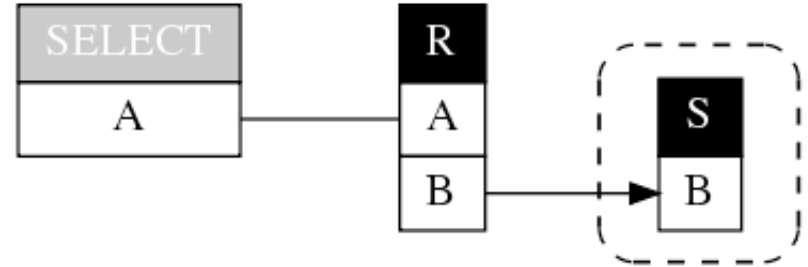
QueryViz for Query Intent, not Debugging

Discontinuity with NULL values

```
select R.A
from R
where not exists
      (select *
       from S
       where S.B = R.B)
```

```
select R.A
from R
where R.B not IN
      (select S.B
       from S)
```

Empty result if S.B contains NULL

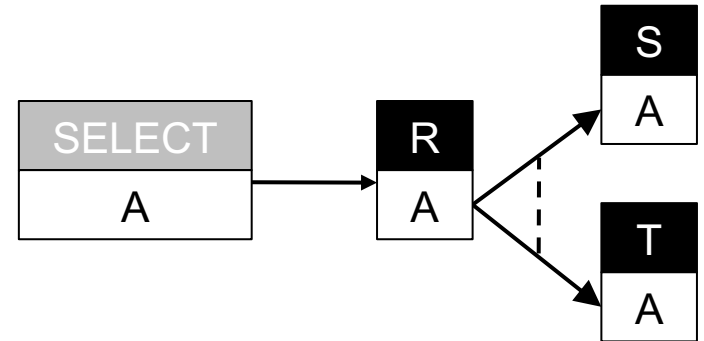


Discontinuity with empty tables

```
select R.a
from R, S
where R.a=S.a
or exists
      (select *
       from T
       where R.a=T.a)
```

```
select R.a
from R, S, T
where R.a=S.a
or R.a=T.a
```

Empty result if T is empty



Design decision 6: minimum visual complexity ?
possible overloading and ambiguity just as in NL

Input: Schema

Input Query

Output: Visualization

Danaparamita+ [EDBT'11]

Your Input

Specify or choose a pre-defined schema [help](#)

Employee and Department

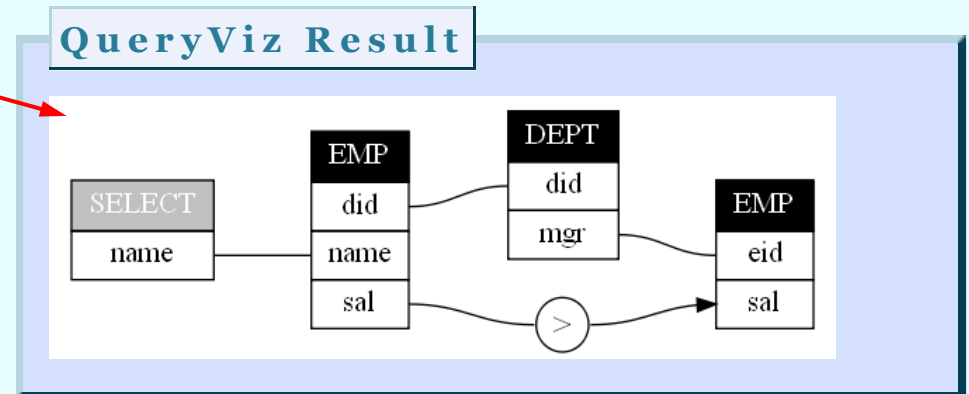
EMP(eid,name,sal,did)
DEPT(did,dname,mgr)

Specify or choose an SQL Query [help](#)

Query 8

SELECT e1.name
FROM EMP e1, EMP e2, DEPT d
WHERE e1.did = d.did
AND d.mgr = e2.eid
AND e1.sal > e2.sal

Submit

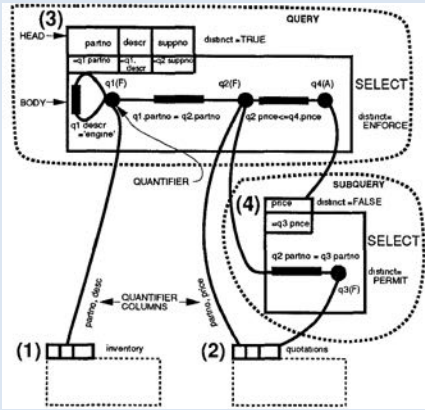


Wide Open Questions

- 1. How to visualize outer joins, sorting, arithmetic expressions, etc.?*
- 2. What is the appropriate level of abstraction? (intent vs. debugging)*
- 3. What are the appropriate basic visual metaphors?*
- 4. Can we visualize at different granularities? ("zooming in")*
- 5. How can we visualize query fragments?*
- 6. How to adapt visualizations to audiences? ("one size fit all")*
- 7. How to optimally place the visual elements?*
- 8. How to standardize evaluation of alternative approaches?
("TPC-H for speed of Query Interpretation" via user studies)*

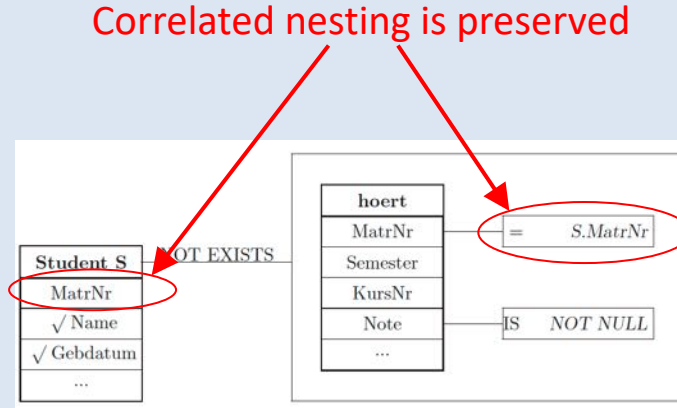
Wide Open Questions

1. *How to visualize outer joins, sorting, arithmetic expressions, etc.?*
2. *What is the appropriate level of abstraction? (intent vs. debugging)*



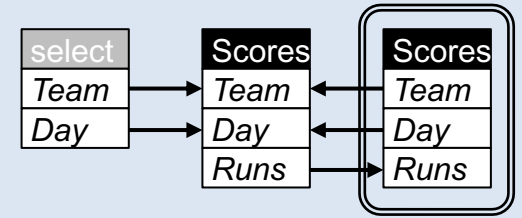
Query Graph Model

Pirahesh et al. [Sigmod'92]



Most VQL* such as Visual SQL

Jaakkola & Thalheim [ER WS'03]



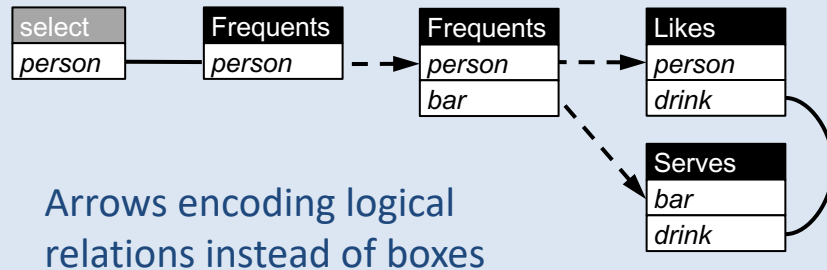
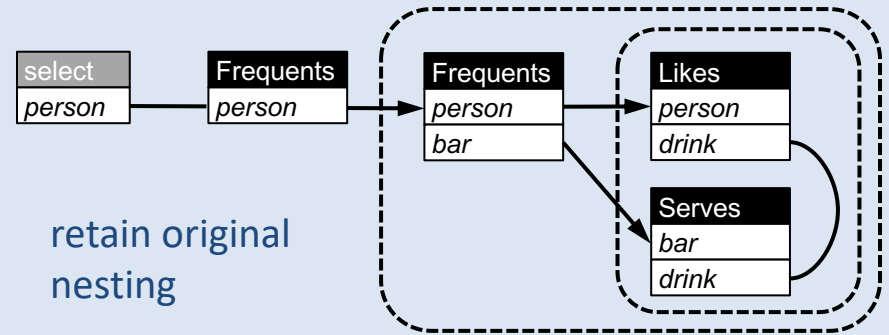
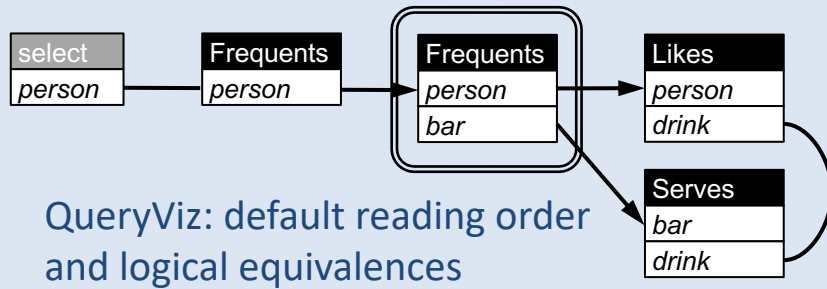
Query Intent

→
more abstract

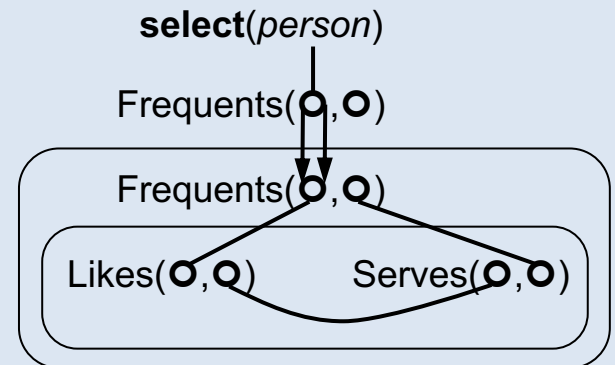
* Note that VQL (Visual Query Languages) do not provide the reverse functionality of query visualization

Wide Open Questions

1. How to visualize outer joins, sorting, arithmetic expressions, etc.?
2. What is the appropriate level of abstraction? (intent vs. debugging)
3. What are the appropriate basic visual metaphors?



Something completely different



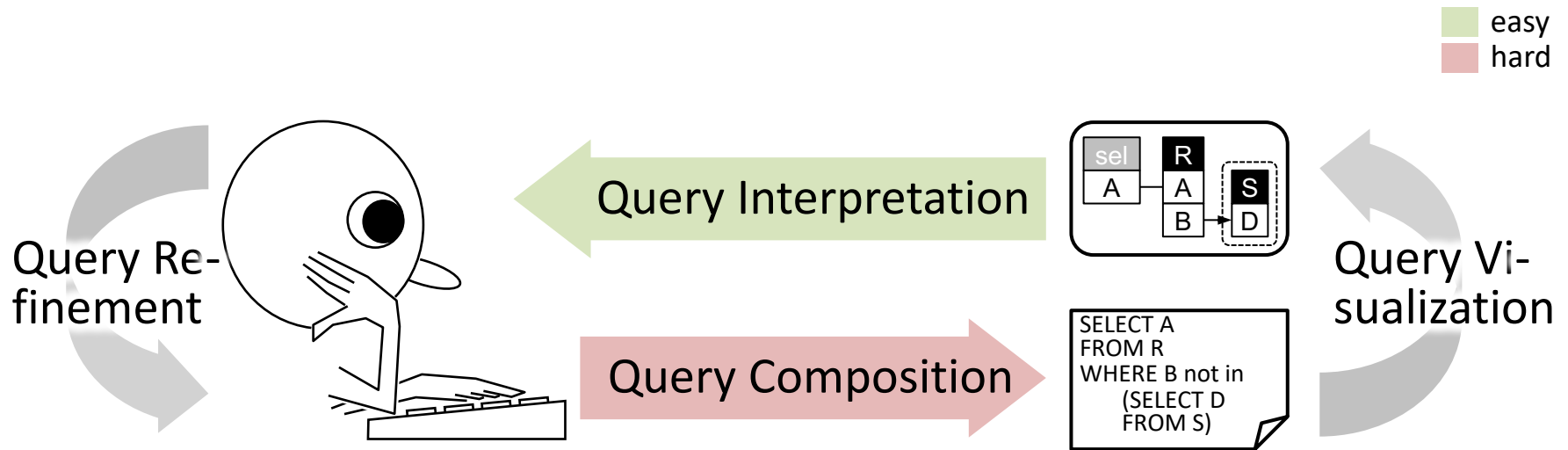
Wide Open Questions

- 1. How to visualize outer joins, sorting, arithmetic expressions, etc.?*
- 2. What is the appropriate level of abstraction? (intent vs. debugging)*
- 3. What are the appropriate basic visual metaphors?*
- 4. Can we visualize at different granularities? ("zooming in")*
- 5. How can we visualize query fragments?*
- 6. How to adapt visualizations to audiences? ("one size fit all")*
- 7. How to optimally place the visual elements?*
- 8. How to standardize evaluation of alternative approaches?
("TPC-H for speed of Query Interpretation" via user studies)*

The Vision in a Nutshell

Q Visualization can facilitate **Q Composition** through

- (i) faster **Q Interpretation** and thus Q Re-use, and
- (ii) a visual understanding of **SQL design patterns**.



☐ *"Databases will visualize queries too"*