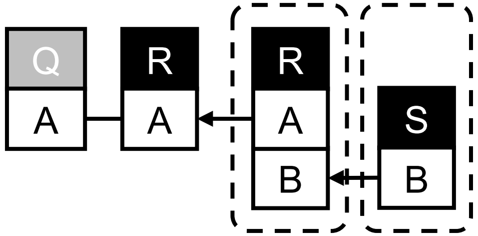


Last Updated 10/5/2023

PLEASE ask questions!
Thank you ☺

Visual Representations of Relational Queries

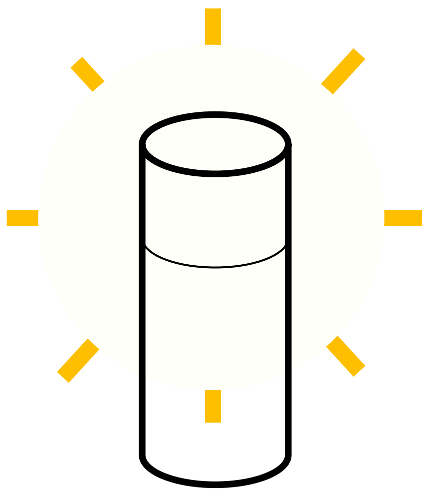
Wolfgang Gatterbauer

August 31, 2023

Thanks everyone for having left excellent comments via the anonymous feedback form linked from the tutorial web page. I updated the slides based on comments. Please do still keep the comments coming ☺

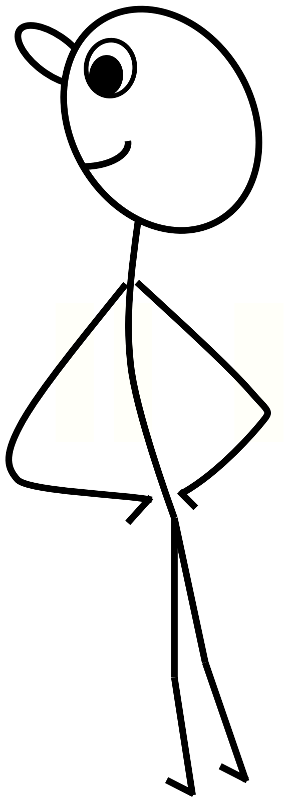
Slides (after I get more feedback) will be posted on the tutorial web page:
https://northeastern-datalab.github.io/visual-query-representation-tutorial/

1. How do you know the voice assistant understood you correctly?



Q: Find me companies in Vancouver.

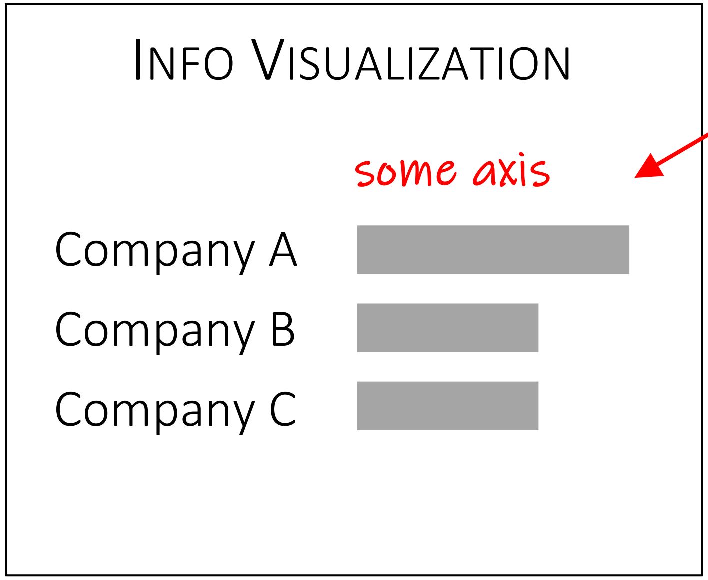
A: Company A, Company B, Company C



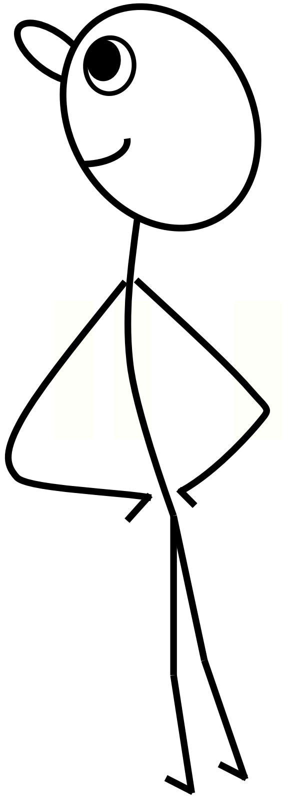
Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf

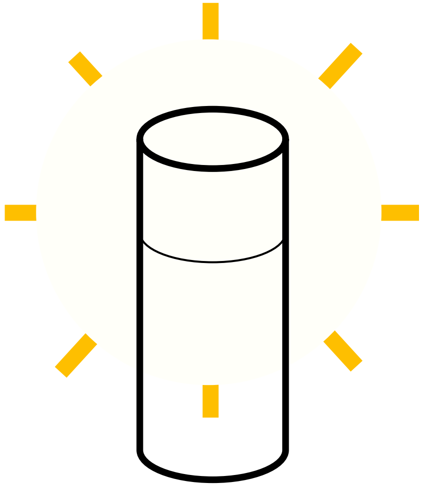
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

1. How do you know the voice assistant understood you correctly?



Visualize the answer



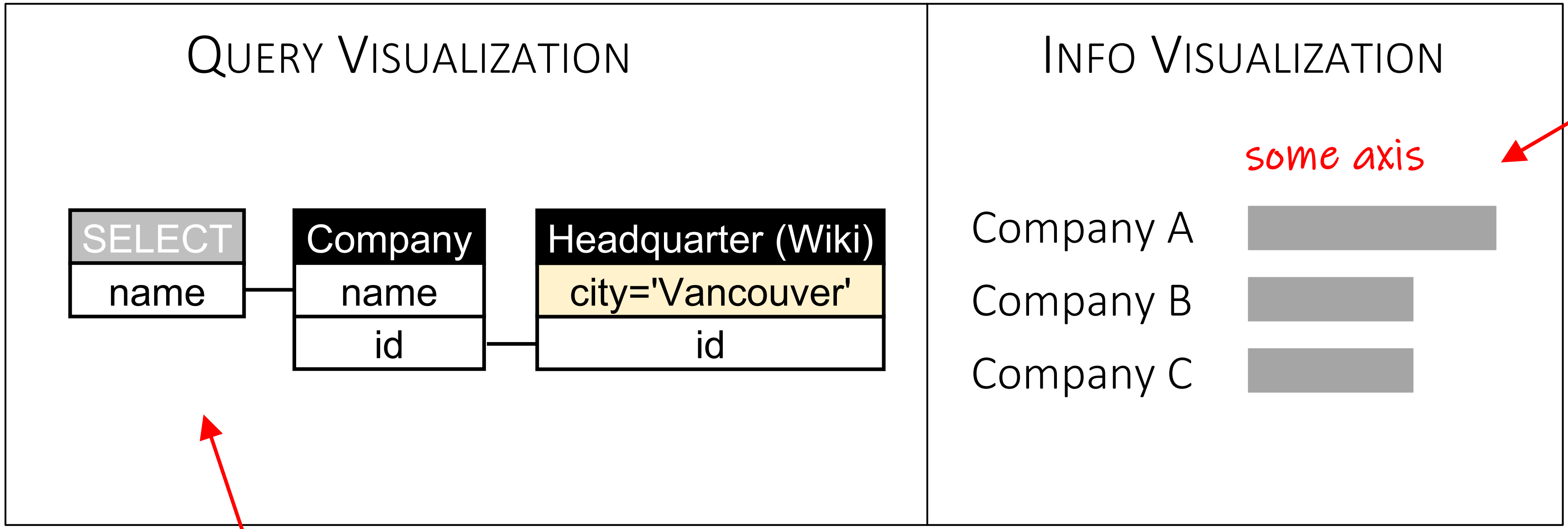


Q: Find me companies in Vancouver.

A: Company A, Company B, Company C

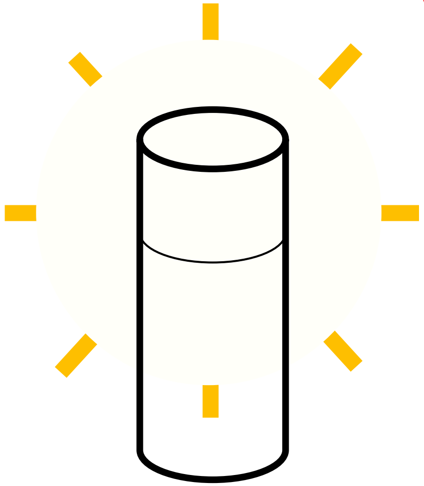
Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

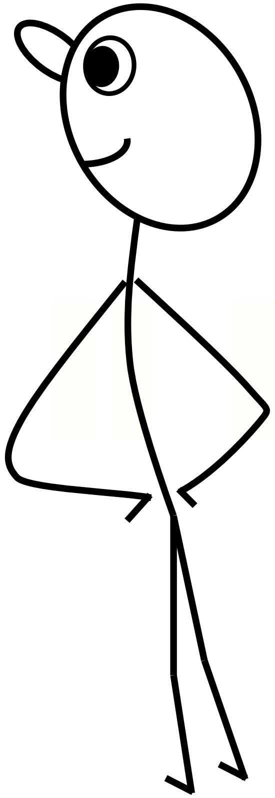
1. How do you know the voice assistant understood you correctly?



Visualize the answer

! Visualize the query (what the assistant understood, and & how sources are used)



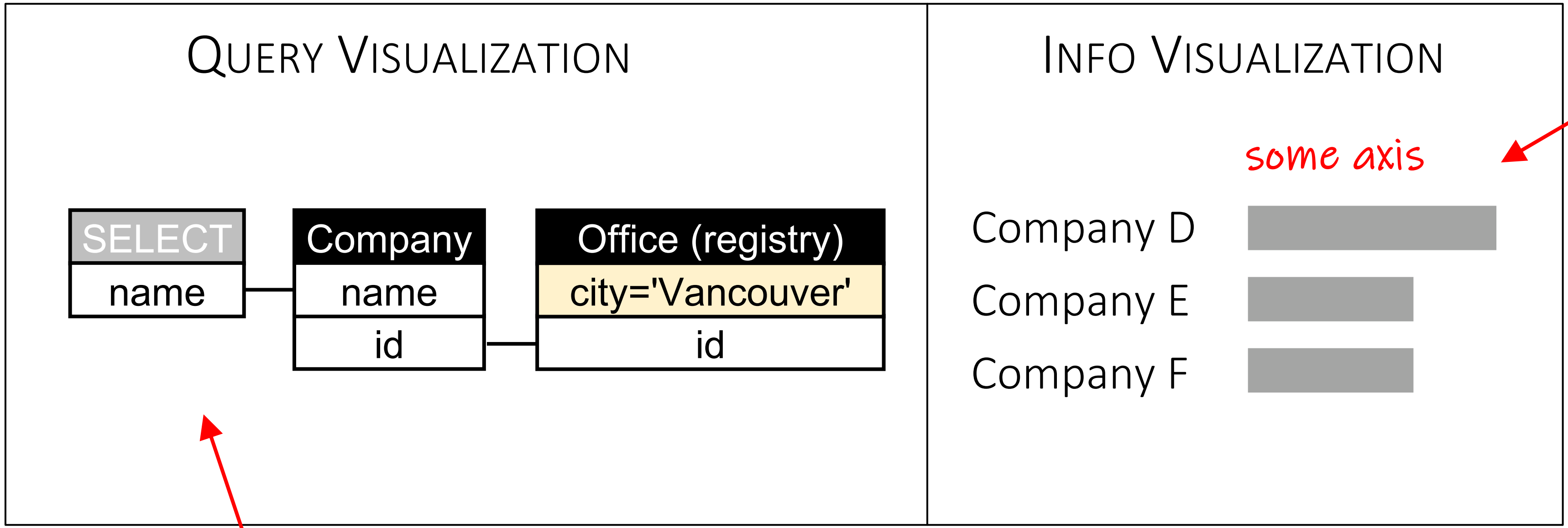


Q: Find me companies in Vancouver.

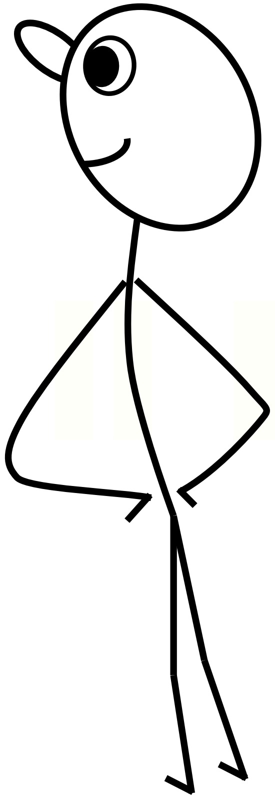
A: Company A, Company B, Company C

Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

1. How do you know the voice assistant understood you correctly?

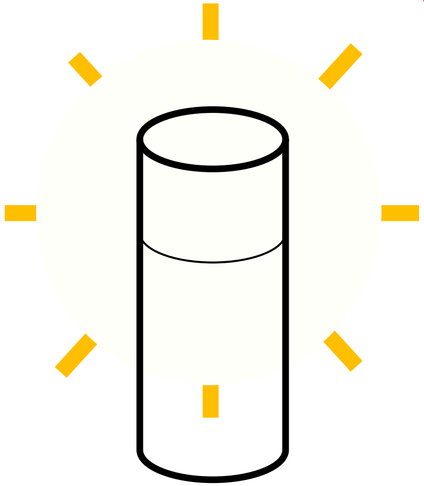


Visualize the answer



! Visualize the query

(Do you mean companies with headquarter or office in Vancouver?) ?

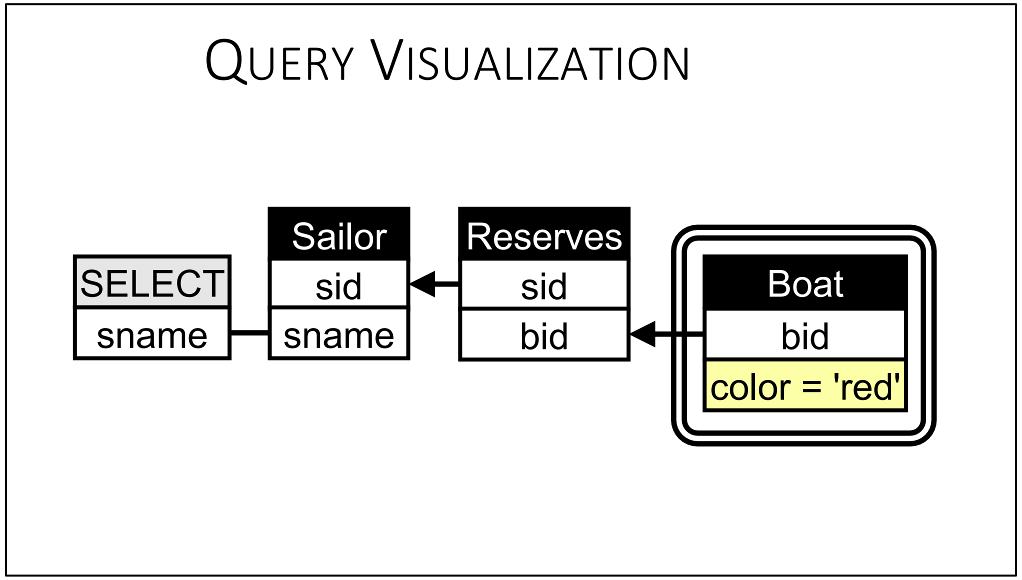


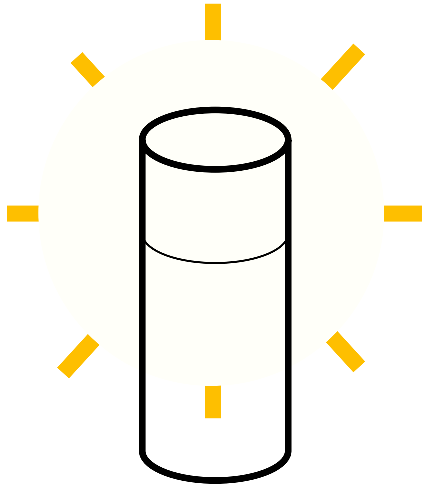
Q: Find me companies in Vancouver.

A: Company D, Company E, Company F

Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

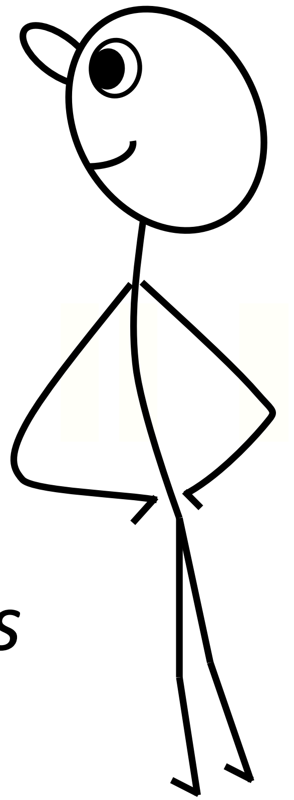
1. How do you know the voice assistant understood you correctly?





This will be our running example (for didactic reasons)

Q: Find sailors who reserved all red boats



Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

2. Helping students see SQL patterns

Sailor (<u>sid</u>, sname, rating, bdate)
Reserves (<u>sid, bid, day</u>)
Boat (<u>bid</u>, bname, color, pdate)

```
select S.sname
from Sailor S
where not exists (
    select *
    from Reserves R, Boat B
    where R.sid = S.sid
    and R.bid = B.bid
    and B.color = 'red')
```

```
select S.sname
from Sailor S
where not exists (
    select *
    from Reserves R
    where R.sid = S.sid
    and not exists (
        select *
        from Boat B
        where B.color = 'red'
        and R.bid = B.bid))
```

```
select S.sname
from Sailor S
where not exists (
    select *
    from Boat B
    where B.color = 'red'
    and not exists (
        select *
        from Reserves R
        where R.bid = B.bid
        and R.sid = S.sid))
```

Example taken from: https://queryvis.com/example.html

2. Helping students see SQL patterns

Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)

	Sailors who **have not** reserved a red boat	Sailors who reserved **only** red boats	Sailors who reserved **all** red boats

SQL	select S.sname
	from Sailor S
	where not exists (
	    select *
	    from Reserves R, Boat B
	    where R.sid = S.sid
	    and R.bid = B.bid
	    and B.color = 'red')
		select S.sname
		from Sailor S
		where not exists (
		    select *
		    from Reserves R
		    where R.sid = S.sid
		    and not exists (
		        select *
		        from Boat B
		        where B.color = 'red'
		        and R.bid = B.bid))
			select S.sname
			from Sailor S
			where not exists (
			    select *
			    from Boat B
			    where B.color = 'red'
			    and not exists (
			        select *
			        from Reserves R
			        where R.bid = B.bid
			        and R.sid = S.sid))

Example taken from: https://queryvis.com/example.html

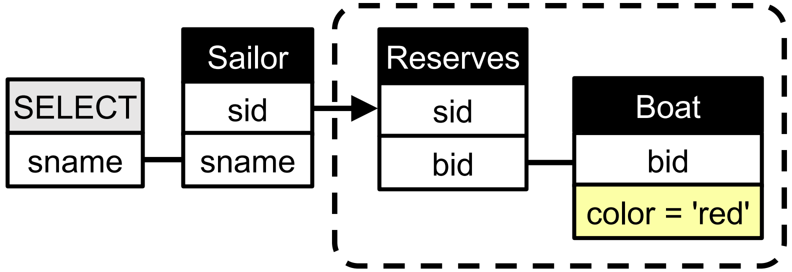
2. Helping students see SQL patterns

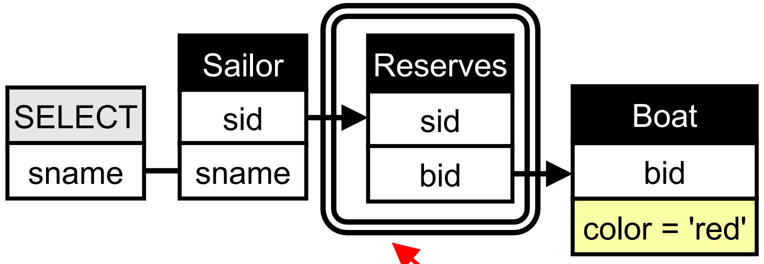
Sailor (<u>sid</u>, sname, rating, bdate)
Reserves (<u>sid, bid, day</u>)
Boat (<u>bid</u>, bname, color, pdate)

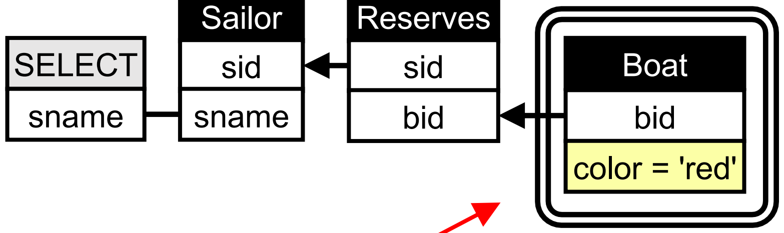
	Sailors who **have not** reserved a red boat	Sailors who reserved **only** red boats	Sailors who reserved **all** red boats
SQL	select S.sname from Sailor S where not exists (select * from Reserves R, Boat B where R.sid = S.sid and R.bid = B.bid and B.color = 'red')	select S.sname from Sailor S where not exists (select * from Reserves R where R.sid = S.sid and not exists (select * from Boat B where B.color = 'red' and R.bid = B.bid))	select S.sname from Sailor S where not exists (select * from Boat B where B.color = 'red' and not exists (select * from Reserves R where R.bid = B.bid and R.sid = S.sid))

dashed box = not exists

QV







double lined box = for all

Example taken from: https://queryvis.com/example.html
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)

	not	**only**	**all**
Sailors renting boats	have not reserved a red boat	reserved only red boats	reserved all red boats

Example taken from: https://queryvis.com/example.html
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Student (sid, sname)
Takes (sid, cid, semester)
Course (cid, cname, department)

Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)

	not	**only**	**all**
Sailors renting boats	have not reserved a red boat	reserved only red boats	reserved all red boats
Students taking classes	took no art class	took only art classes	took all art classes

Example taken from: https://queryvis.com/example.html

Actor (aid, aname)
Plays (aid, mid, role)
Movie (mid, mname, director)

Student (sid, sname)
Takes (sid, cid, semester)
Course (cid, cname, department)

Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)

	not	**only**	**all**
Sailors renting boats	have not reserved a red boat	reserved only red boats	reserved all red boats
Students taking classes	took no art class	took only art classes	took all art classes
Actors playing in movies	did not play in a Hitchcock movie	played only Hitchcock movies	played in all Hitchcock movies

Example taken from: https://queryvis.com/example.html

Actor (aid, aname)
Plays (aid, mid, role)
Movie (mid, mname, director)

Student (sid, sname)
Takes (sid, cid, semester)
Course (cid, cname, department)

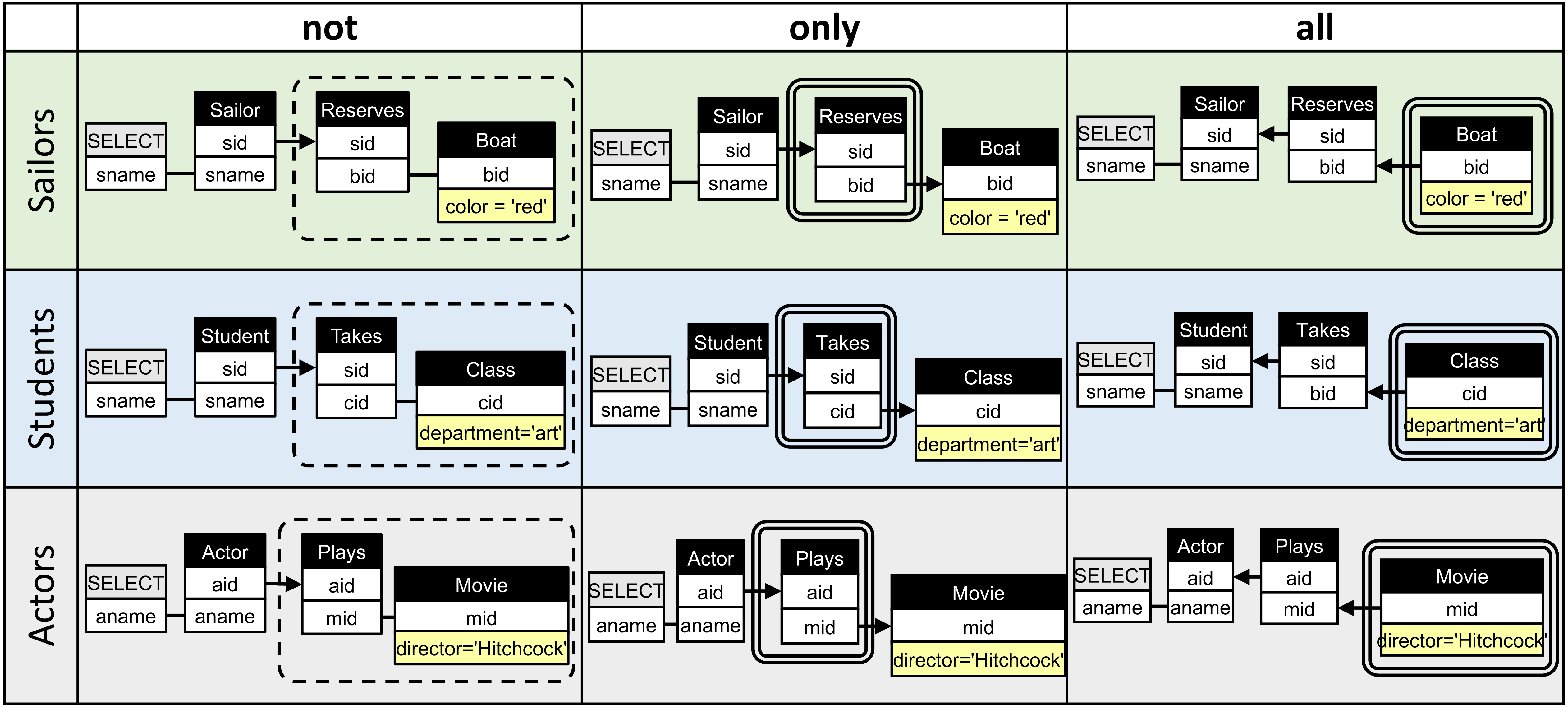
Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)

	not	only	all
Sailors	select S.sname from Sailor S where not exists (select * from Reserves R, Boat B where R.sid = S.sid and R.bid = B.bid and B.color = 'red')	select S.sname from Sailor S where not exists (select * from Reserves R where R.sid = S.sid and not exists (select * from Boat B where B.color = 'red' and B.bid = R.bid))	select S.sname from Sailor S where not exists (select * from Boat B where B.color = 'red' and not exists (select * from Reserves R where R.bid = B.bid and R.sid = S.sid))
Students	select S.sname from Student S where not exists (select * from Takes T, Class C where T.sid = S.sid and C.cid = T.cid and C.department ='art')	select S.sname from Student S where not exists (select * from Takes T where T.sid = S.sid and not exists (select * from Class C where C.department = 'art' and C.cid= T.cid))	select S.sname from Student S where not exists (select * from Class C where C.department= 'art' and not exists (select * from Takes T where T.cid= C.cid and T.sid= S.sid))
Actors	select A.aname from Actor A where not exists (select * from Plays P, Movie M where P.aid = A.aid and M.mid = P.mid and M.director = 'Hitchcock')	select A.aname from Actor A where not exists (select * from Plays P where P.aid = A.aid and not exists (select * from Movie M where M.director = 'Hitchcock' and M.mid = P.mid))	select A.aname from Actor A where not exists (select * from Movie M where M.director = 'Hitchcock' and not exists (select * from Plays P where P.mid = M.mid and P.aid = A.aid))

Actor (aid, aname)
Plays (aid, mid, role)
Movie (mid, mname, director)

Student (sid, sname)
Takes (sid, cid, semester)
Course (cid, cname, department)

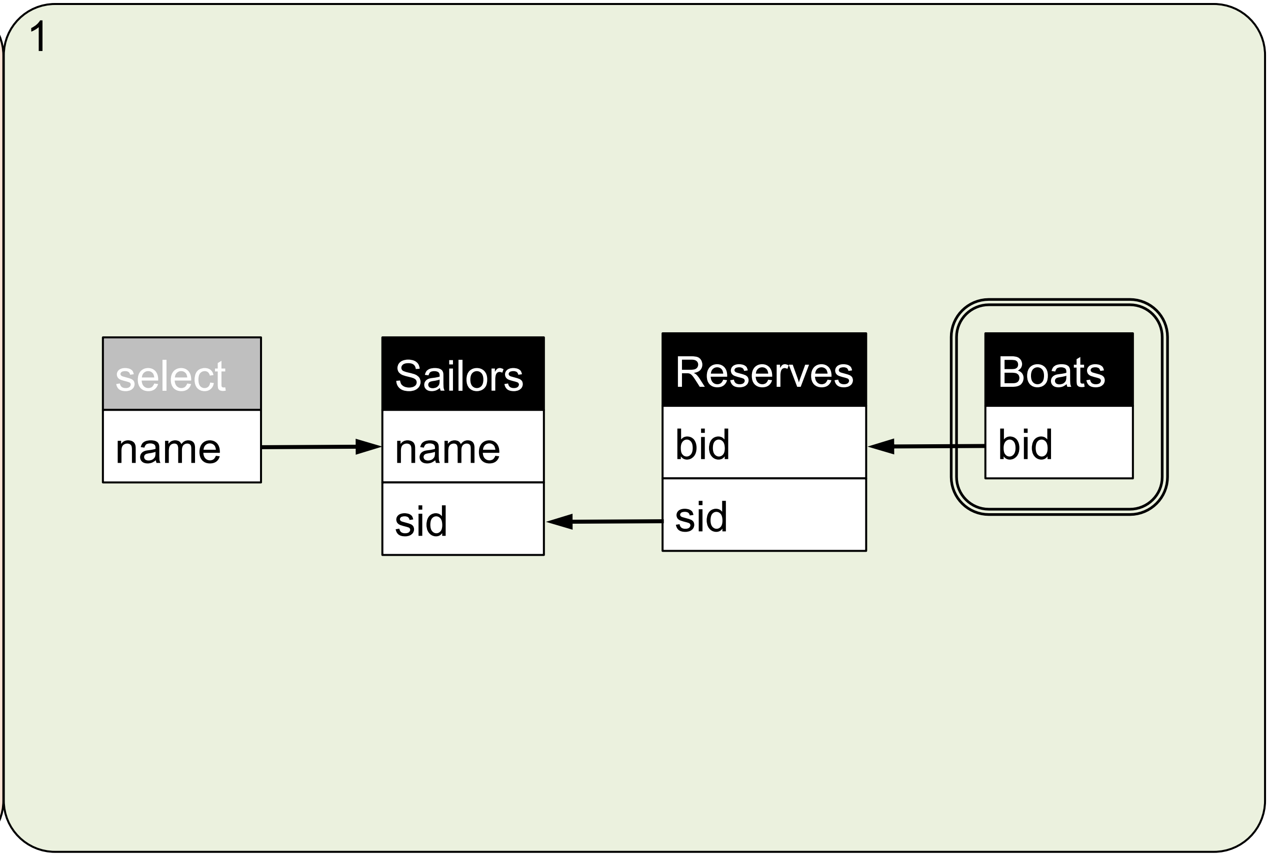
Sailor (sid, sname, rating, bdate)
Reserves (sid, bid, day)
Boat (bid, bname, color, pdate)



3. Browsing & understanding existing Queries

1

```
select S.sname
from Sailors S
where not exists
    (select B.bid
    from Boats B
    where not exists
        (select R.bid
        from Reserves R
        where R.bid = B.bid
        and R.sid = S.sid))
```

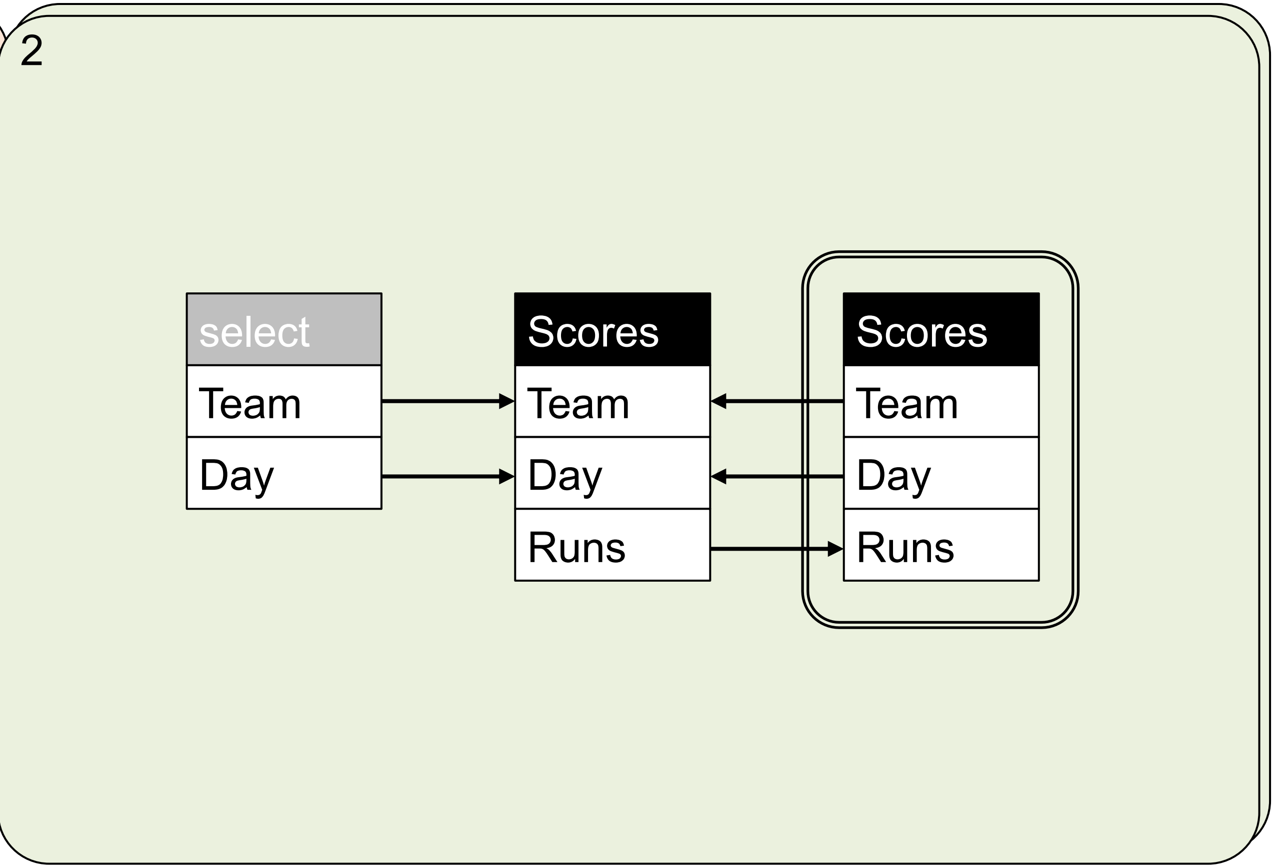


Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

3. Browsing & understanding existing Queries

2

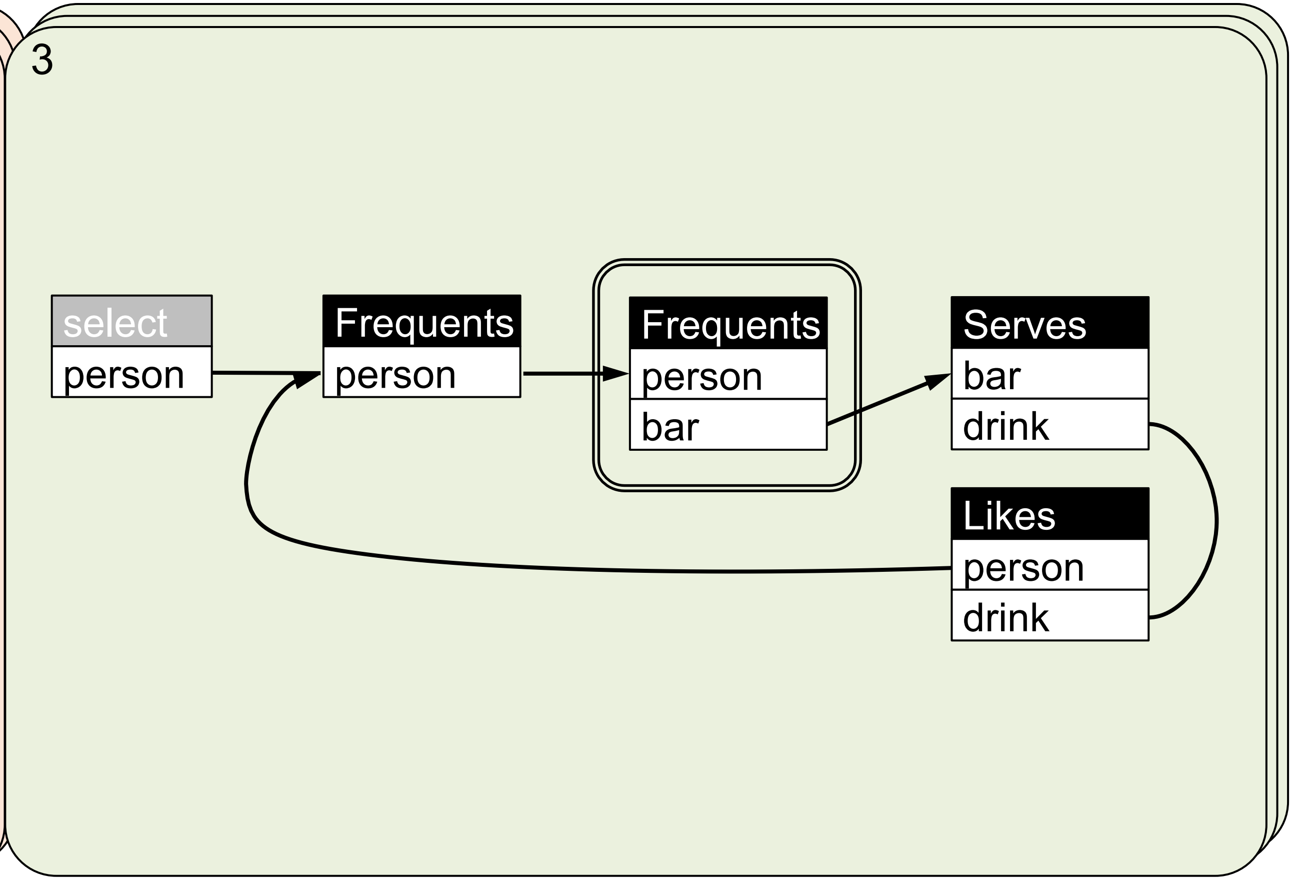
```
select Team, Day
from Scores S1
where not exists
      (select *
      from Scores S2
      where S1.Runs = S2.Runs
      and (S1.Team <> S2.Team
      or S1.Day <> S2.Day))
```



3. Browsing & understanding existing Queries

3

```
select F1.person
from Frequents F1
where not exists
    (select F2.bar
    from Frequents F2
    where F2.person = F1.person
    and not exists
        (select S3.drink
        from Serves S3, Likes L4
        where L4.person = F1.person
        and L4.drink = S3.drink
        and S3.bar = F2.bar))
```

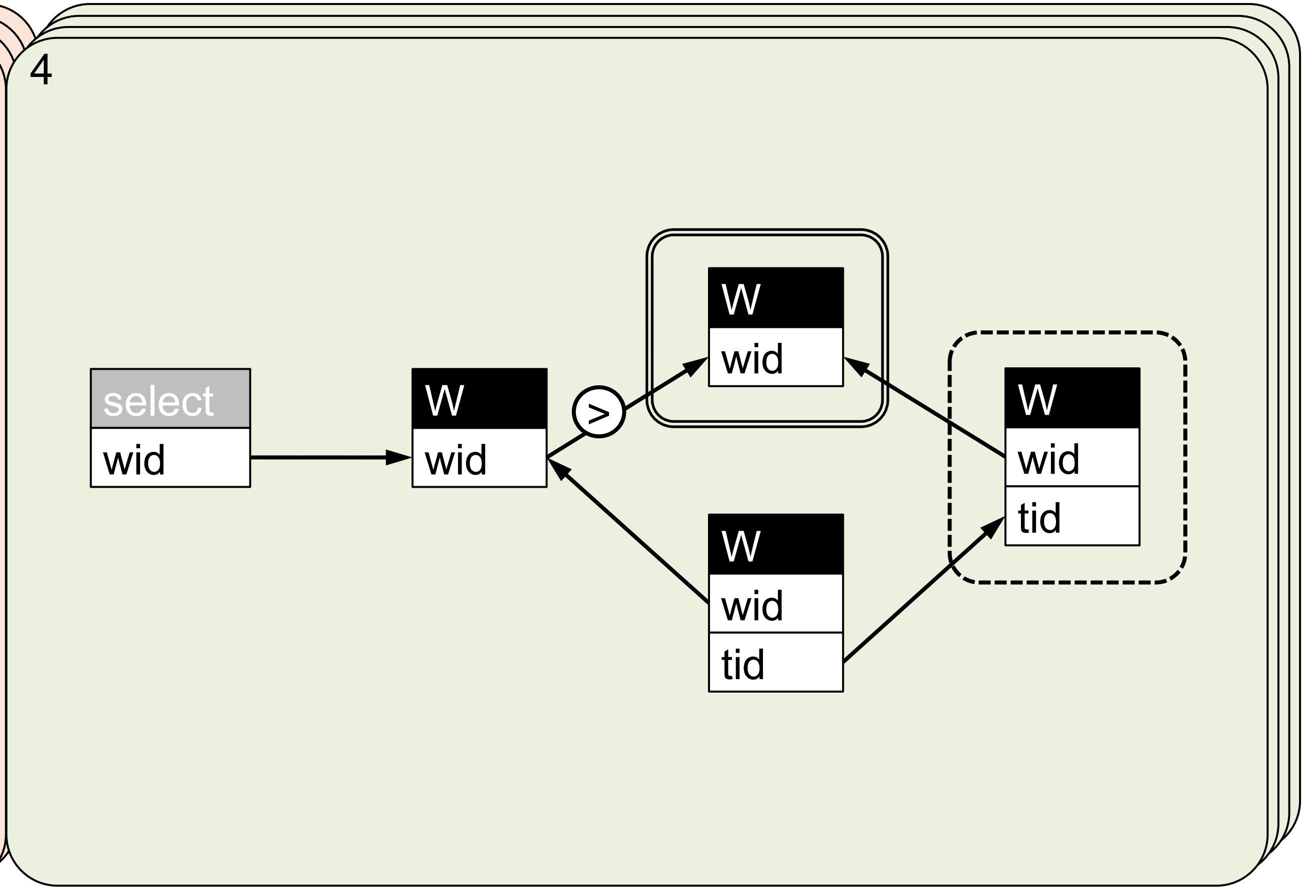


Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

3. Browsing & understanding existing Queries

4

```
select W1.wid
from Worlds W1
where not exists
      (select *
      from Worlds W2
      where W2.wid < W1.wid
      and not exists
            (select *
            from Worlds W3
            where W3.wid = W1.wid
            and not exists
                  (select *
                  from Worlds W4
                  where W4.wid = W2.wid
                  and W4.tid = W3.tid)))
```



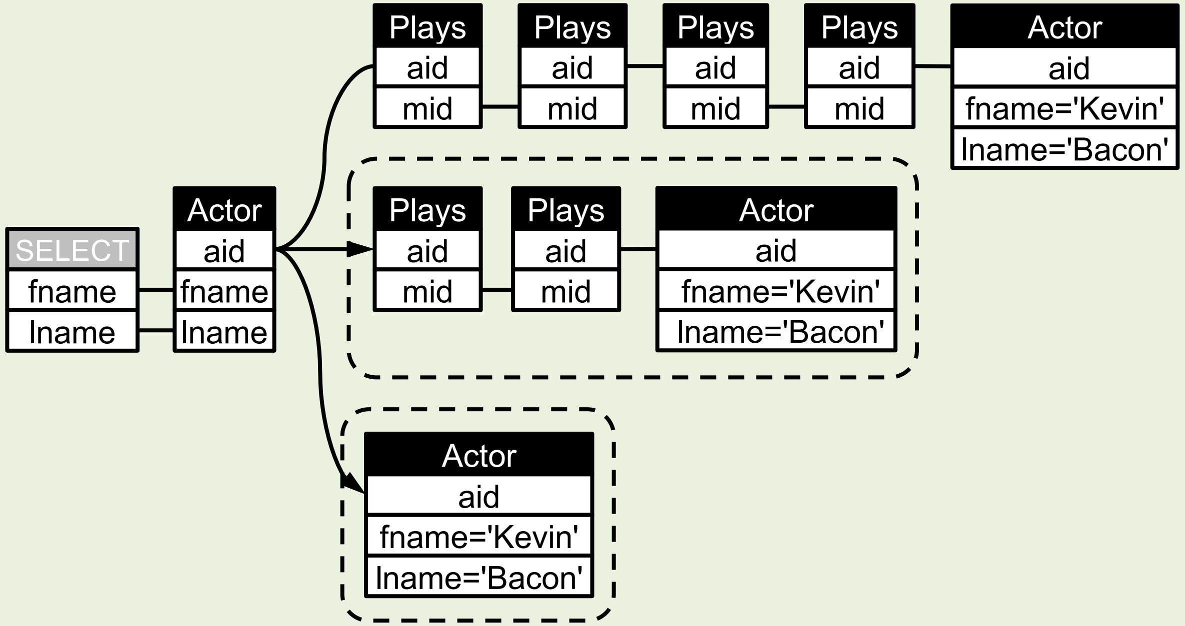
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

3. Browsing & understanding existing Queries

5

```
select distinct a3.fname, a3.lname
from Actor a0, Plays p1, Plays p2, Plays p3, Plays p4,
        Actor a5
where a0.fname = 'Kevin' and a0.lname = 'Bacon'
and a0.aid = p1.aid and p1.mid = p2.mid
and p2.aid = p3.aid and p3.mid = p4.mid
and p4.aid = a5.aid
and not exists (select *
   from Actor a6, Plays p7, Plays p8
   where a6.fname = 'Kevin' and a6.lname = 'Bacon'
   and a6.aid = p7.aid and p7.mid = p8.mid
   and p8.aid = a5.aid)
and not exists (select *
   from Actor a9
   where a9.fname = 'Kevin' and a9.lname = 'Bacon'
   and a9.aid = a5.aid)
```

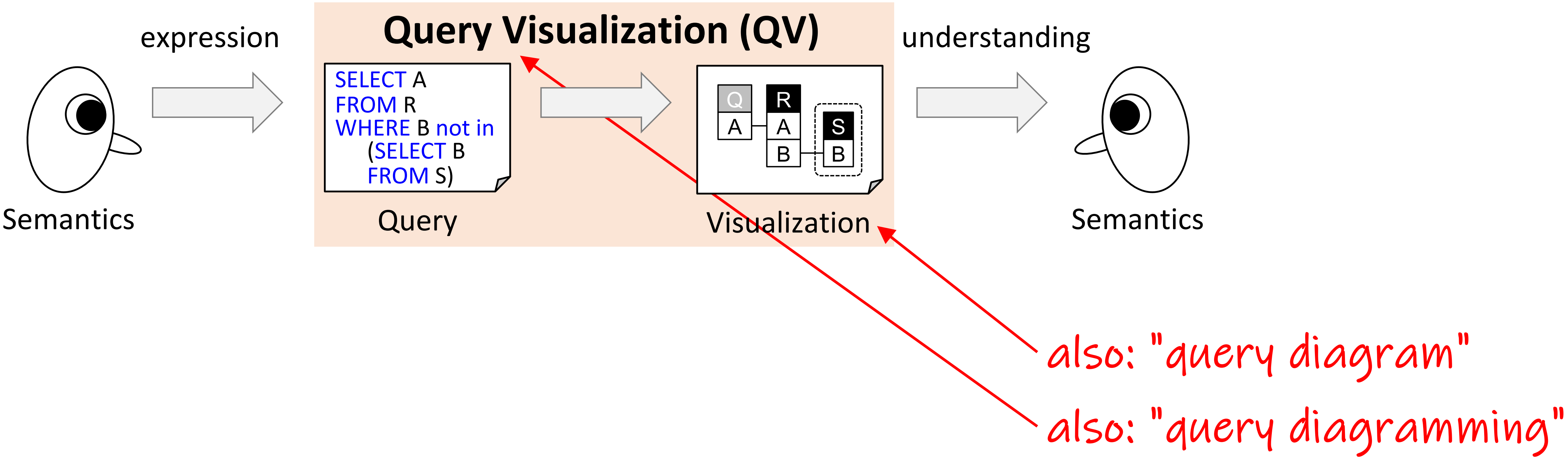
5



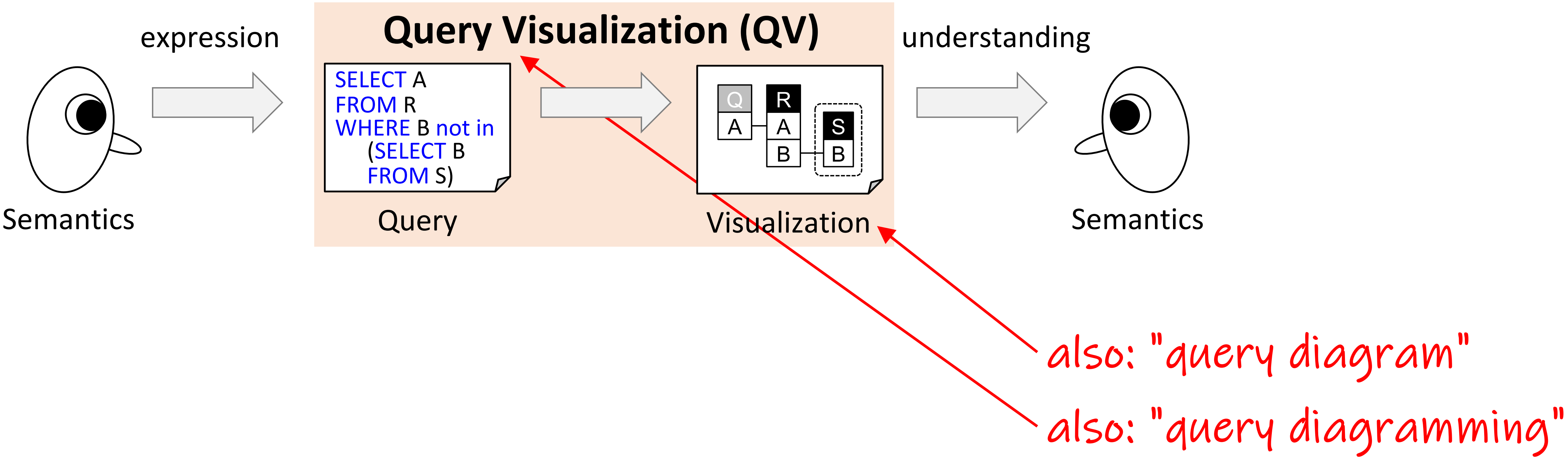
Query Intent: "Q: Find all actors with Bacon number 2."

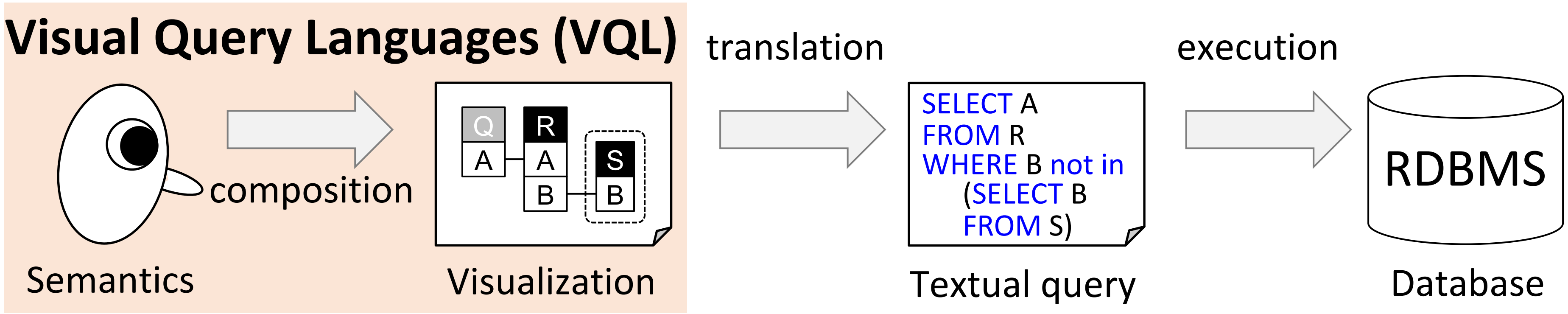
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

"Query Visualization" (for understanding) ≠ "Visual Query Languages" (for composition)

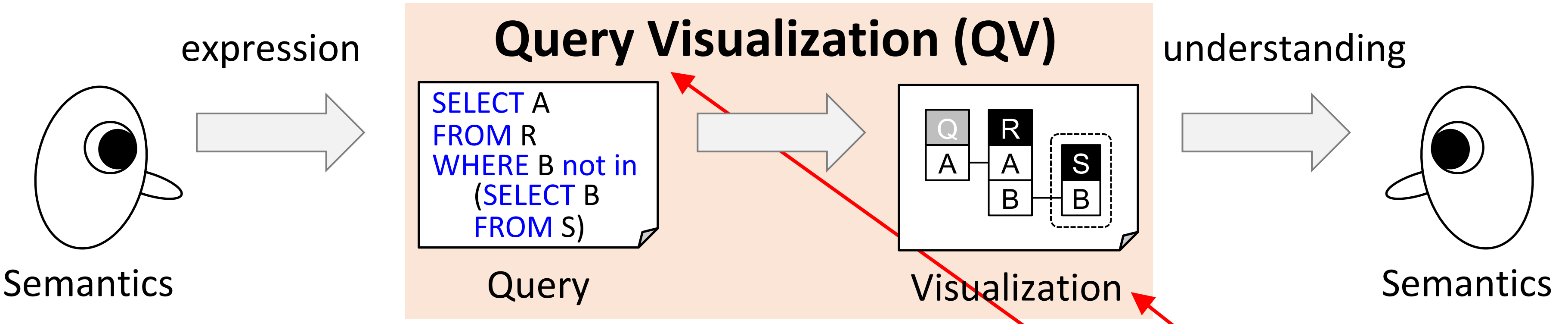


"Query Visualization" (for understanding) ≠ "Visual Query Languages" (for composition)





"Query Visualization" (for understanding) ≠ "Visual Query Languages" (for composition)

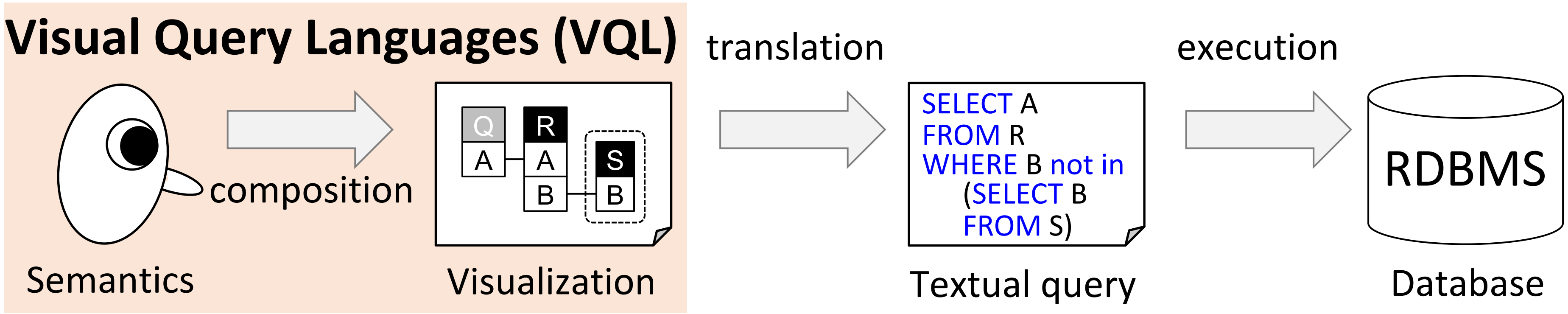


A visual query representation is for: query reading / understanding / explanation

also: "query diagram"

Both applications use "visual representations",
but different objectives lead to different criteria!

also: "query diagramming"

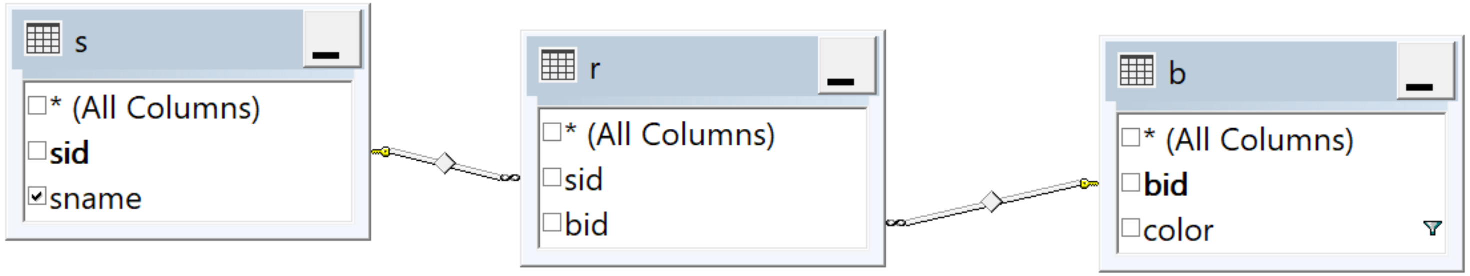


A visual query representation is used for: query writing /composition

What about visual active query builders? Say SSMS?

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor s, Reserves r, Boat b
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

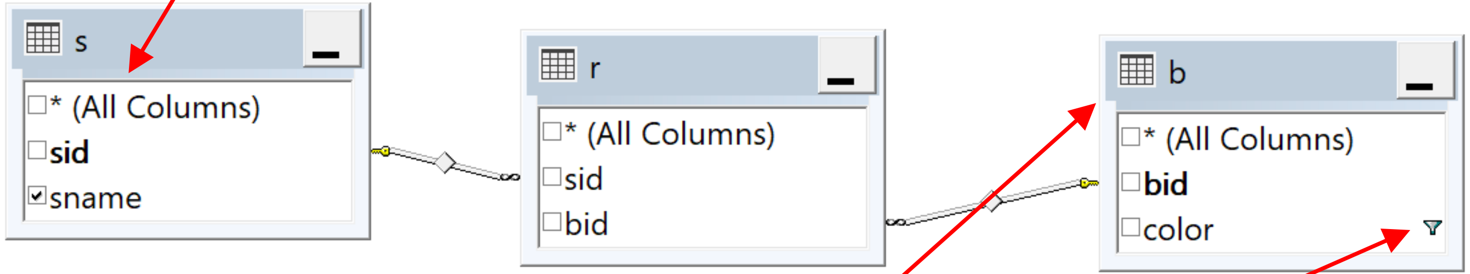


What about visual active query builders? Say SSMS?

Q: "Find sailors who reserved a red boat."

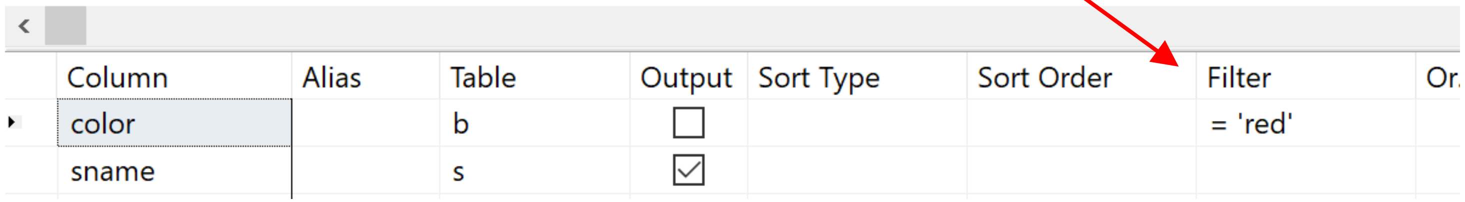
```
select S.sname
from Sailor s, Reserves r, Boat b
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Visual elements that are relevant for composition but not understanding



Other important information is not shown (e.g. the filter symbol hides actual predicate)

Additional window reveals selection predicate



Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or.
color		b	☐			= 'red'	
sname		s	☑				

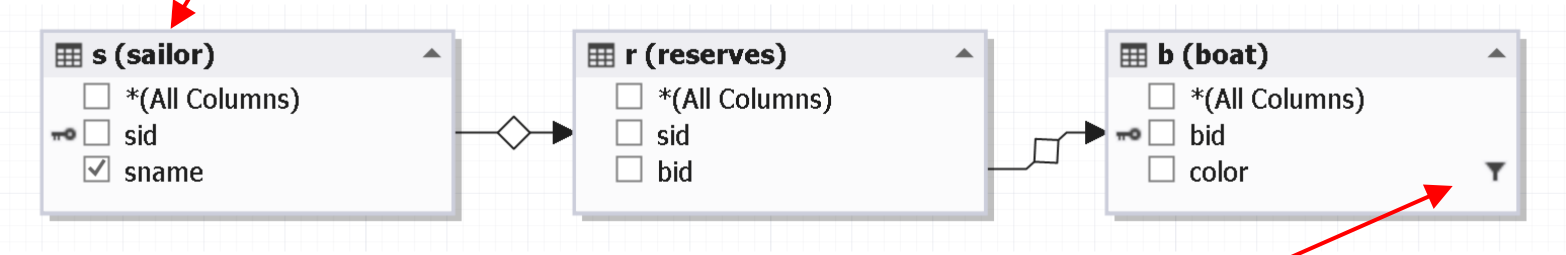
Screenshot acknowledgement: Jiahui Zhang (4/2020), SSMS stands for SQL Server Management Studio (https://en.wikipedia.org/wiki/SQL_Server_Management_Studio)

What about visual active query builders? Or dbForge?

Q: "Find sailors who reserved a red boat."

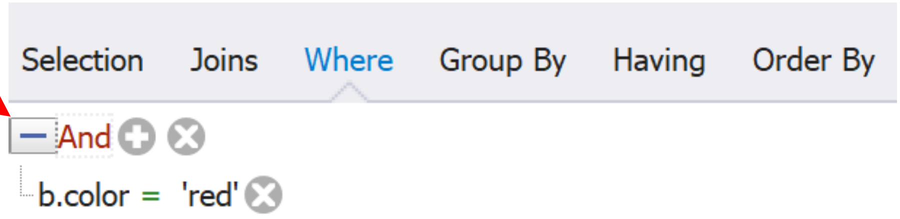
```
select S.sname
from Sailor s, Reserves r, Boat b
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Unnecessary visual elements shown that are relevant for composition but not understanding



Other important information is not shown (e.g. the filter symbol hides actual predicate)

Additional window reveals selection predicate



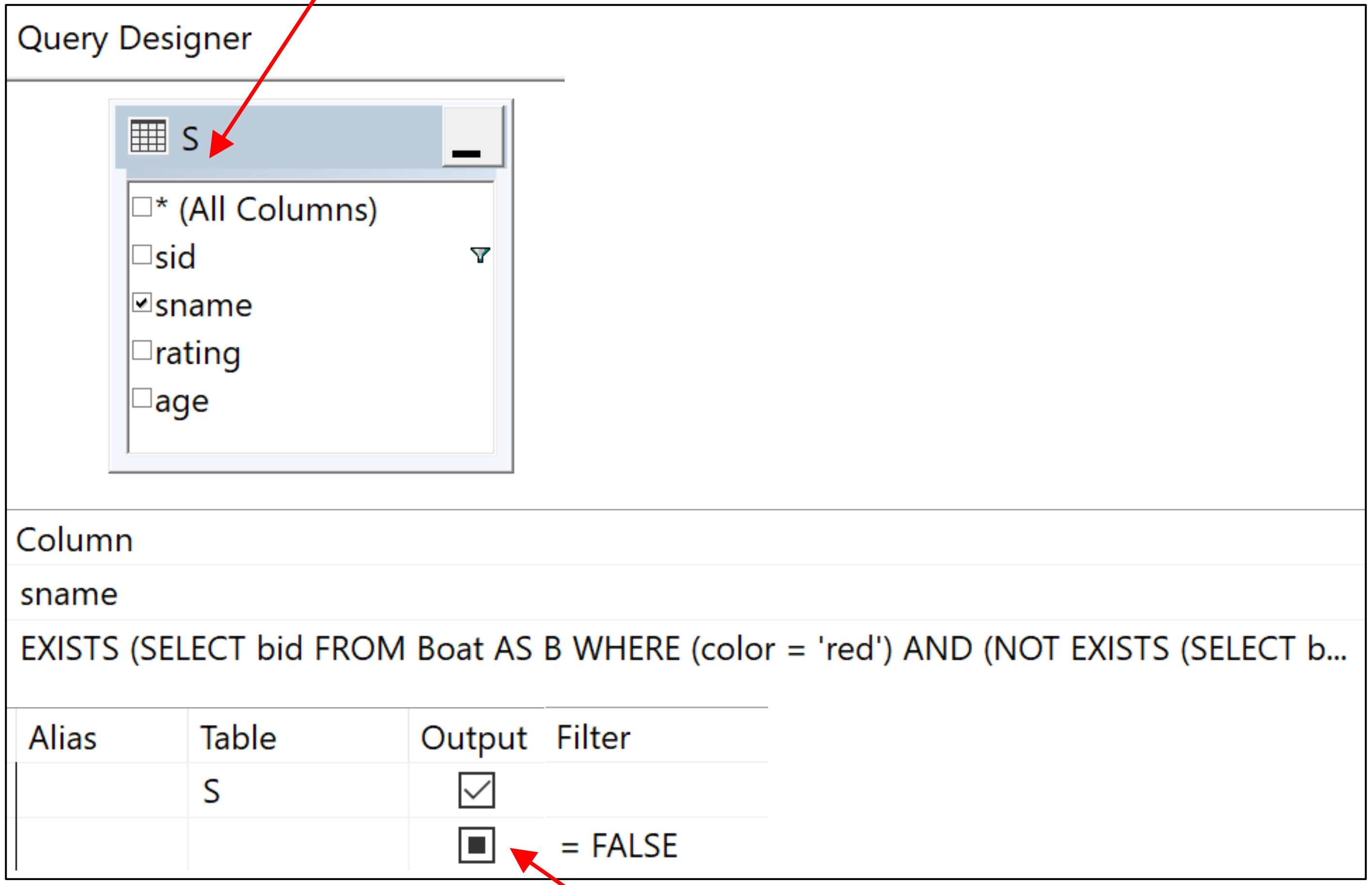
Screenshot acknowledgement: Jiahui Zhang (4/2020), dbForge: https://www.devart.com/dbforge/

What about visual active query builders? Back to SSMS...

Q: "Find sailors who have reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

SSMS does not render the nested blocks



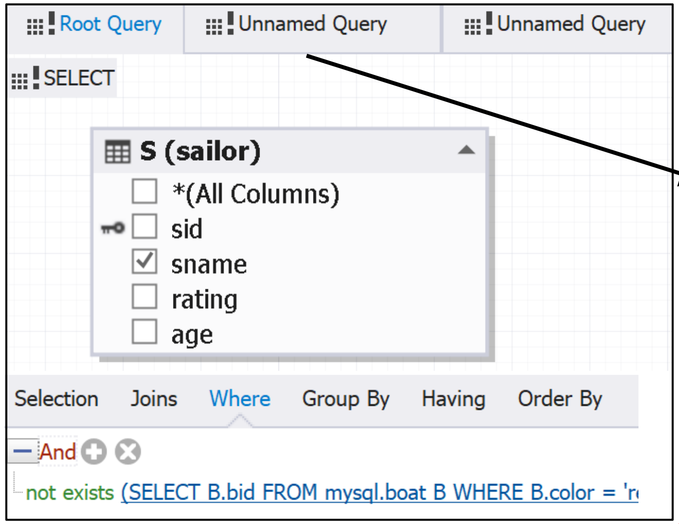
"Not exists" is treated as an expression with a "false" value in the filter

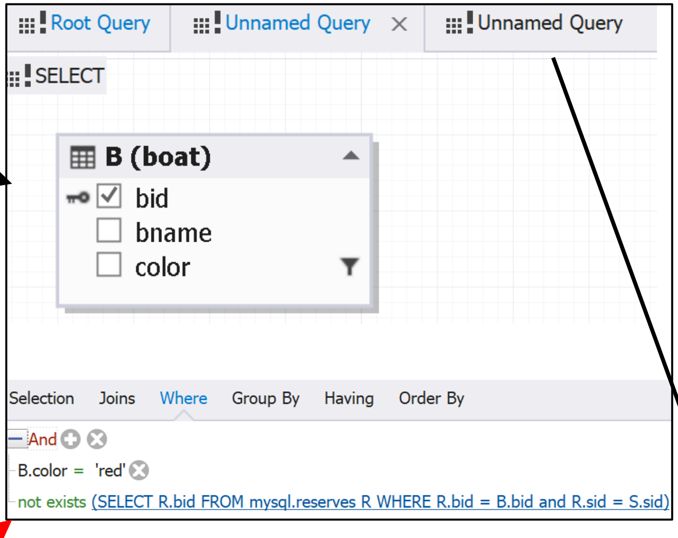
What about visual active query builders? dbForge...

Q: "Find sailors who have reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

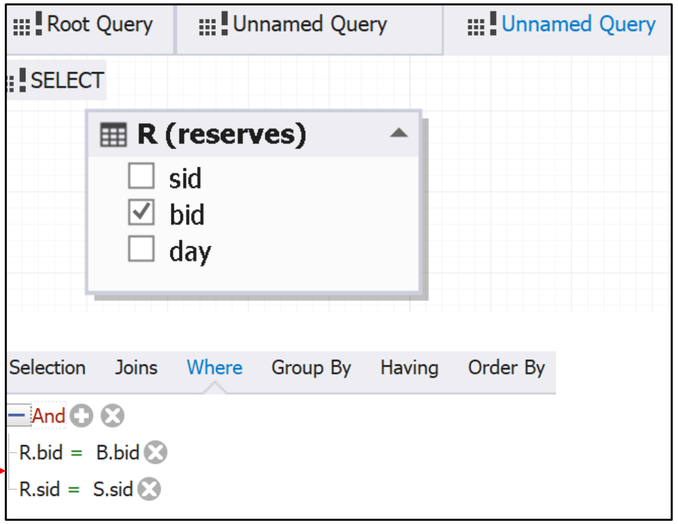
Individual query blocks shown in separate windows





dbForge does not use 'Expression' + 'Criteria' to represent 'NOT' + 'EXISTS'. It directly shows 'Not Exists' as a part of 'Where' clause

dbForge keeps join conditions in WHERE clause as an expression



Screenshot acknowledgement: Jiahui Zhang (4/2020), dbForge: https://www.devart.com/dbforge/

DEFINITION (Query Visualization): The term "query visualization" refers to both

i. a graphical representation of a query and

   (alternatively: "query diagram")

ii. the process of transforming a given query into a graphical representation.

   (alternatively: "query diagramming")

The goal of query visualization is to help users more quickly understand the intent of a query, as well as its relational query pattern.

Source: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Intended Agenda today

Please leave feedback ☺

1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. Early diagrammatic representations
5. Visual Query Representations for Databases
6. Various Open Challenges

What does it mean for a QL to be "visual"?

What is actually a "visual" representation?

- Many attempts on defining an exact notion of "visual" (it's *not easy*)
- In general, authorities acknowledge a spectrum between TEXT and DIAGRAMS

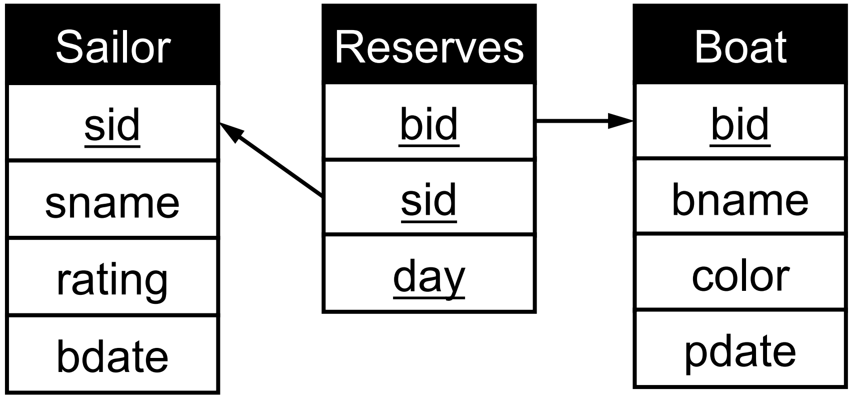
TEXTUAL		DIAGRAMMATIC
symbolic, linguistic, linear, sentential	⟺	visual, graphical, non-symbolic, schematic, as picture, two-dimensional

The exact boundary b/w text and "visual" is not clear-cut!
(And as we will see, "visual" gets interpreted very differently)
Next: We try to develop an intuition for "practical visualization"

Let's explore the boundary between text and diagram

Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

FK Reserves.sid references Sailor
FK Reserves.bid references Boat

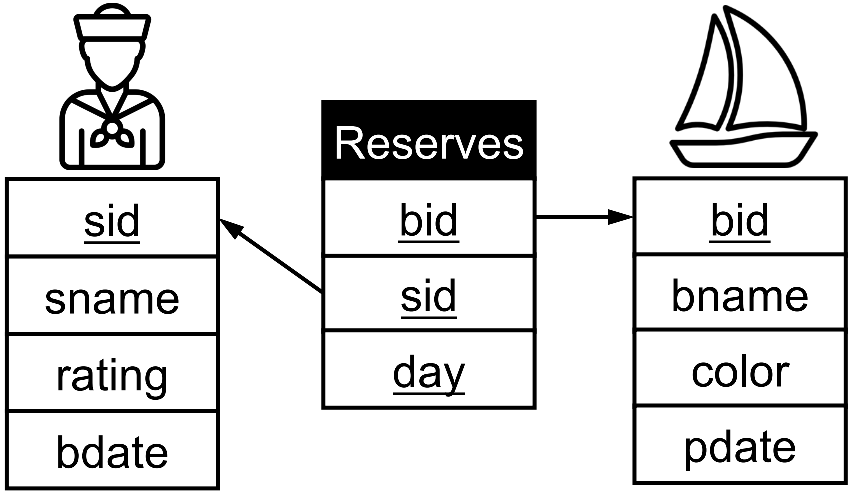


UML diagram of relational schema.

Let's explore the boundary between text and diagram

Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

FK Reserves.sid references Sailor
FK Reserves.bid references Boat



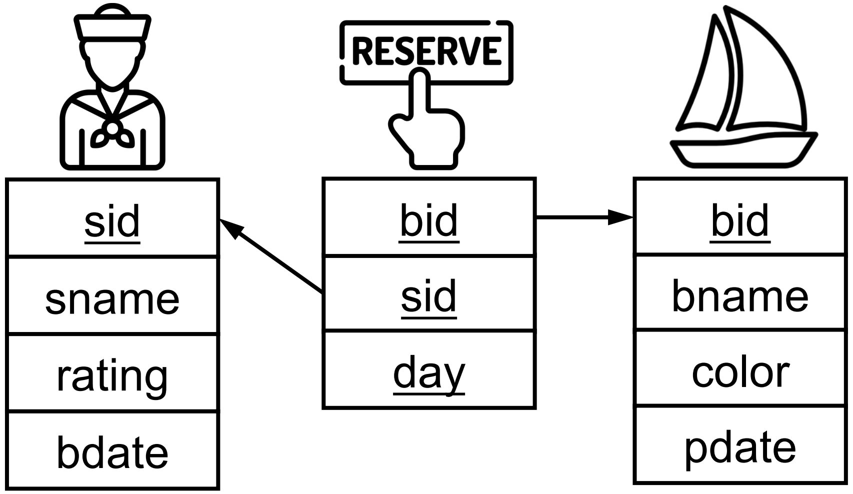
UML diagram of relational schema.

But no need for icons!

Let's explore the boundary between text and diagram

Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

FK Reserves.sid references Sailor
FK Reserves.bid references Boat



UML diagram of relational schema.

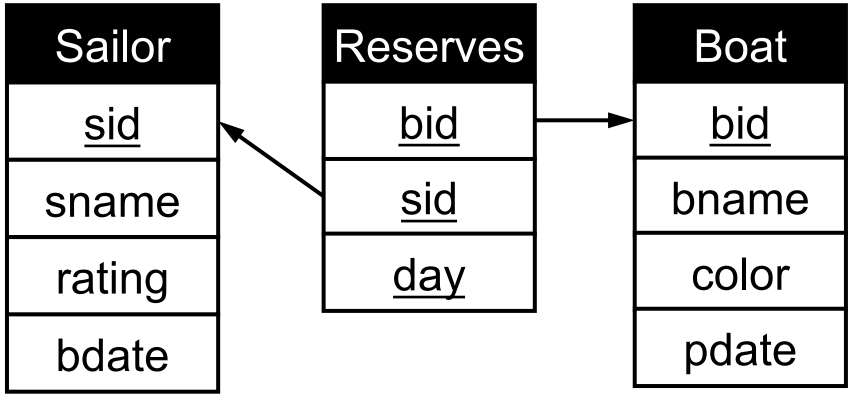
But no need for icons!
(Actually which icons?)

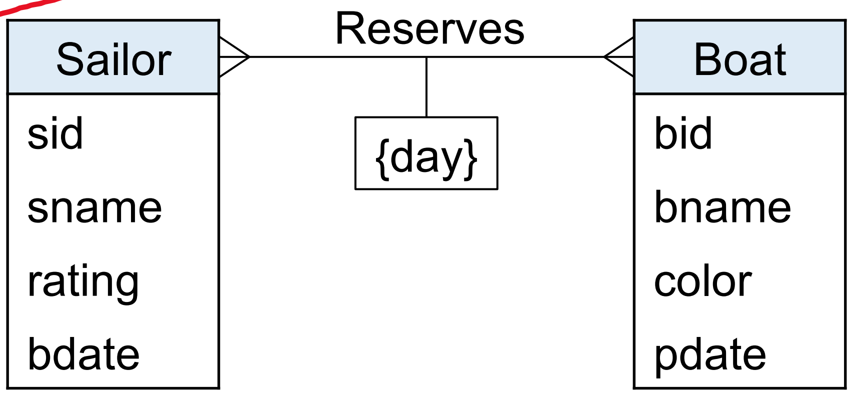
Icons by flaticon.com
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

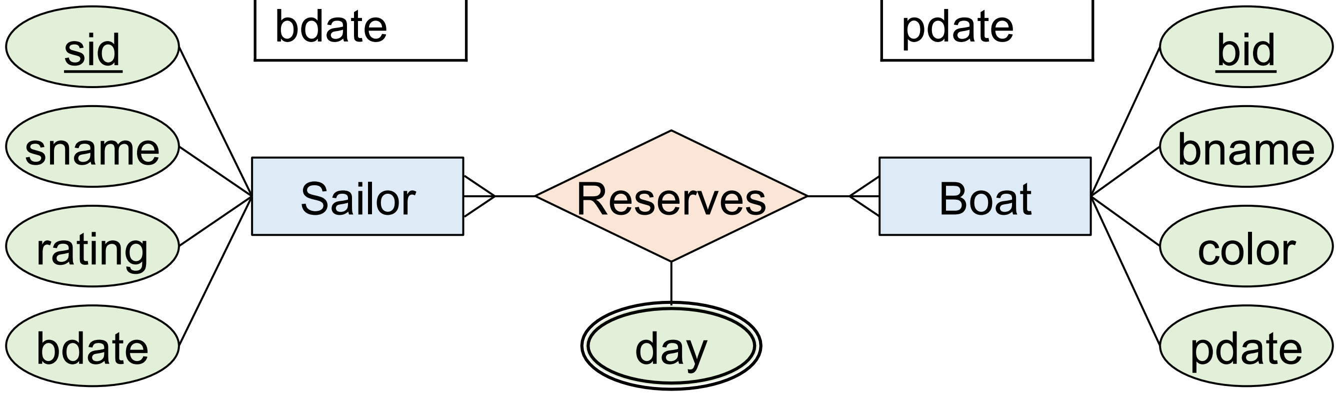
Let's explore the boundary between text and diagram

Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

FK Reserves.sid references Sailor
FK Reserves.bid references Boat







UML diagram of relational schema.

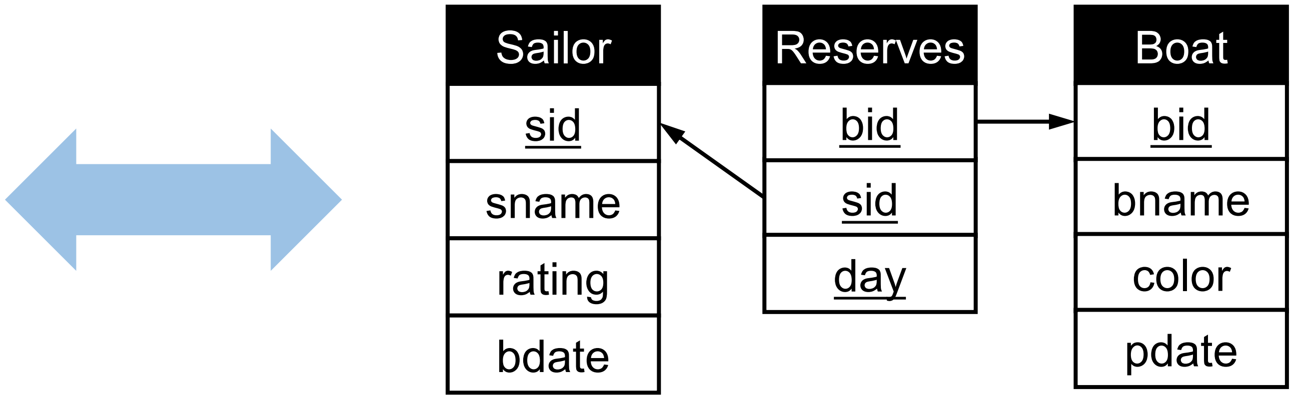
But no need for icons! (Actually which icons?)

ER diagrams also use text

Text vs diagrammatic representations

Sailor(<u>sid</u>, sname, rating, bdate)
Reserves(<u>sid, bid, day</u>)
Boat(<u>bid</u>, bname, color, pdate)

FK Reserves.sid references Sailor
FK Reserves.bid references Boat



Observations:

1. We prefer text as names/labels for atomic elements (tables, attributes)

2. We prefer to "visualize" relationships (the "structure") between these elements.

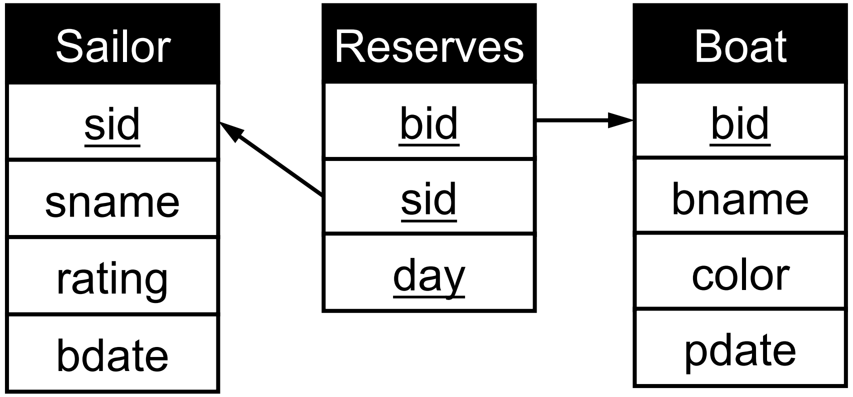
Text vs diagrammatic representations

Sailor(<u>sid</u>, sname, rating, bdate)

Reserves(<u>sid, bid, day</u>)

Boat(<u>bid</u>, bname, color, pdate)

?

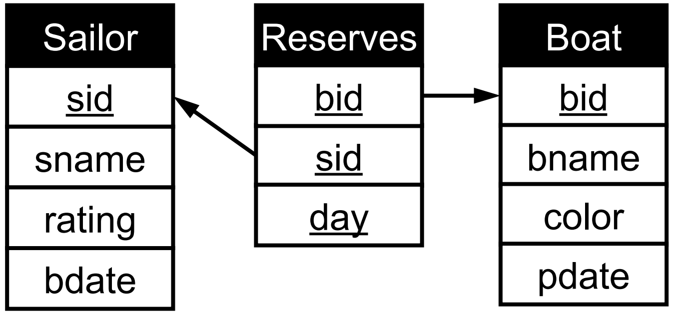


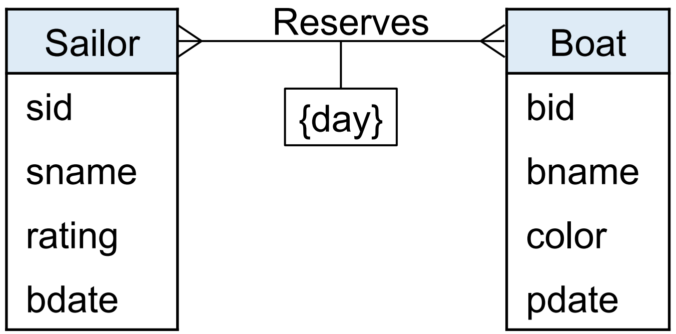
Observations:

1. We prefer text as names/labels for atomic elements (tables, attributes)

2. We prefer to "visualize" relationships (the "structure") between these elements.

There seems to be a sweet spot for visualizing relations

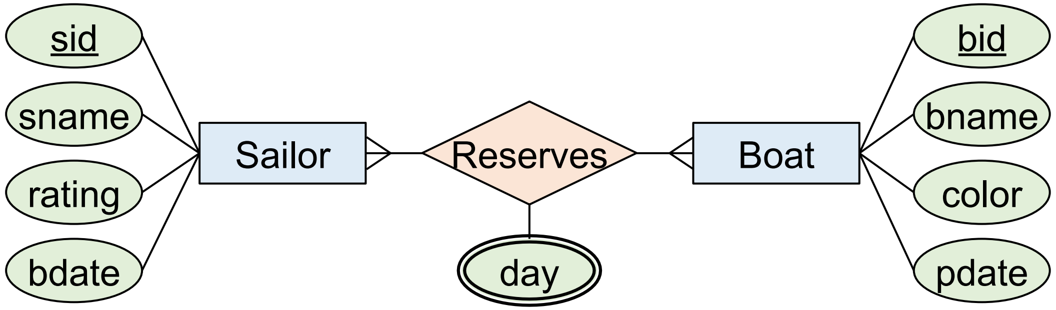




Sailor(sid, sname, rating, bdate)

Reserves(sid, bid, day)

Boat(bid, bname, color, pdate)

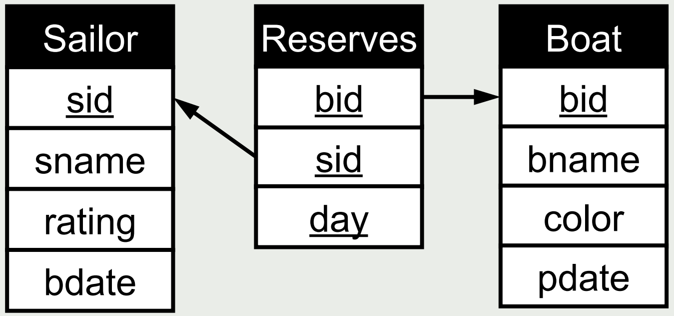


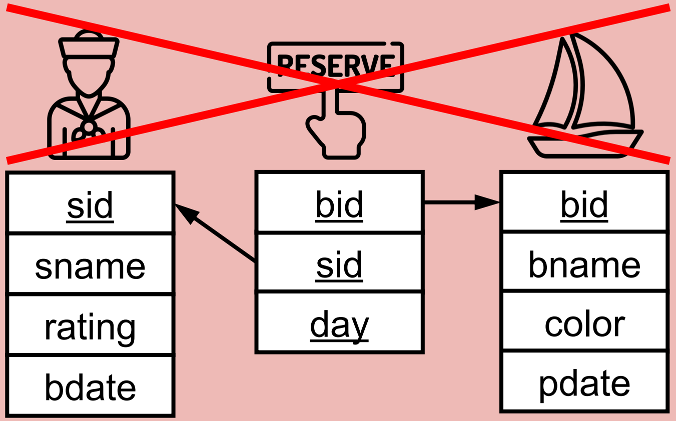
1. We prefer text as names/labels for atomic elements (tables, attributes)
2. We prefer to "visualize" relationships (the "structure") between these elements.

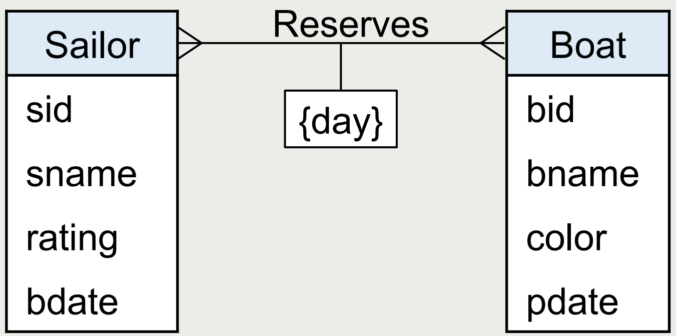
There seems to be a sweet spot for visualizing relations

Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

~~FK Reserves.sid references Sailor~~
~~FK Reserves.bid references Boat~~



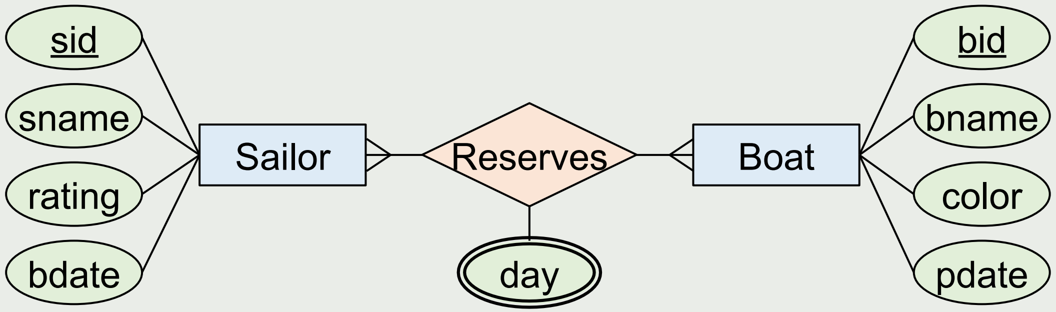




Sailor(sid, sname, rating, bdate)

Reserves(sid, bid, day)

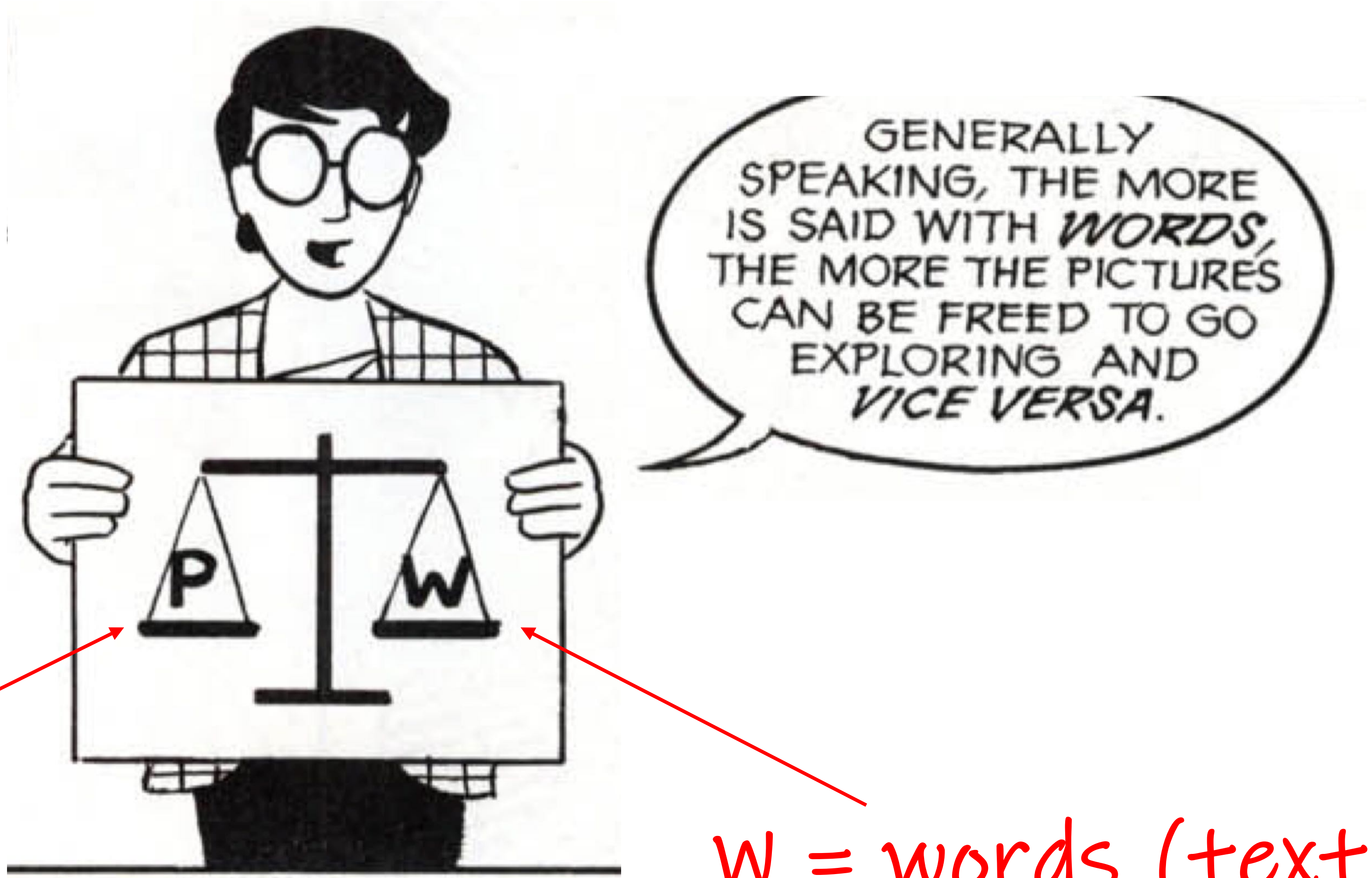
Boat(bid, bname, color, pdate)



1. We prefer text as names/labels for atomic elements (tables, attributes)

2. We prefer to "visualize" relationships (the "structure") between these elements.

Text (= words) vs. pictures in Comics



P = pictures

W = words (text)

Scott McCloud. Understanding comics: the invisible art. 1993. https://en.wikipedia.org/wiki/Understanding_Comics
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Text vs. visualization for "information visualization"

Show It or Tell It?
Text, Visualization, and Their Combination

Marti A. Hearst
hearst@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

- How much text should appear on a visualization?
- What should it say?
- Where should it be placed?
- And how do the visual and the language components interact?

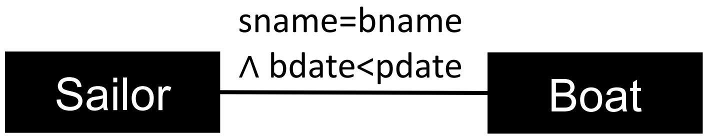
Marti Hearst. Show It or Tell It? VIZ 2022 keynote. Also upcoming CACM 2023 paper. https://people.ischool.berkeley.edu/~hearst/papers/hearst_CACM2023.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

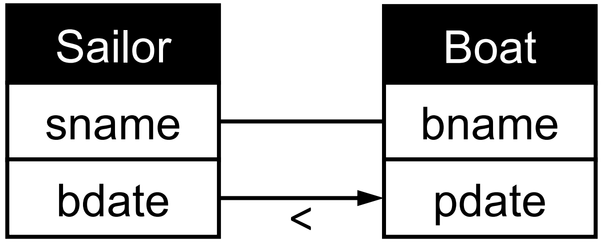
What "structures" / relationships do we actually have in relational queries?

Textual vs diagrammatic representations

Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."

```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```

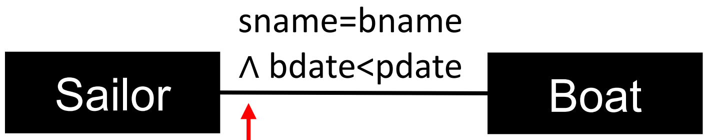




Textual vs diagrammatic representations

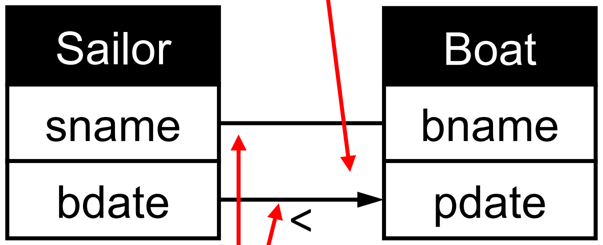
Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."

```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```



a new syntactic device "∧"
for conjunction, used in text

arrowhead shows:
"bdate < pdate"

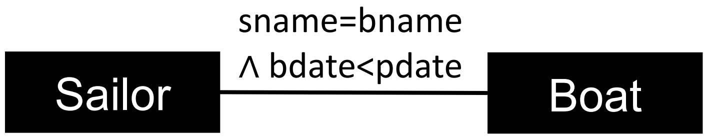


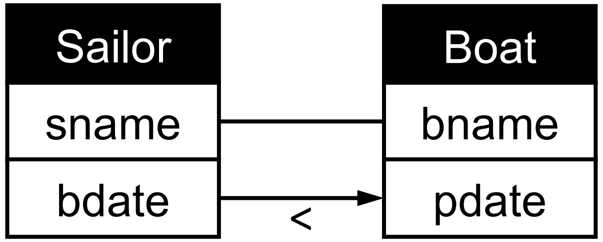
juxtaposition of conjunctive information
(we perceive them independently)

Textual vs diagrammatic representations

Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."

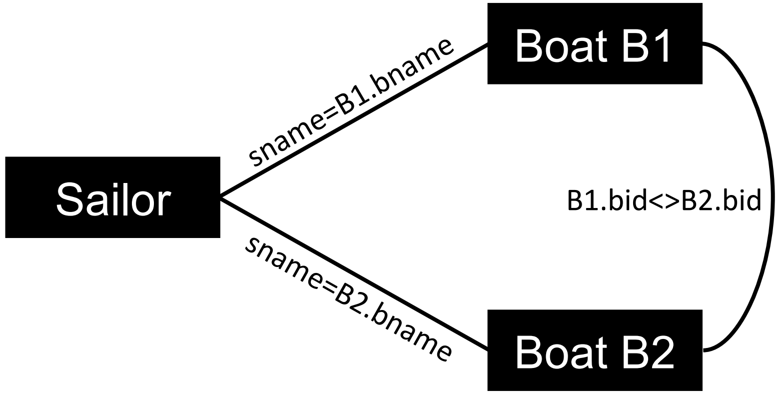
```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```

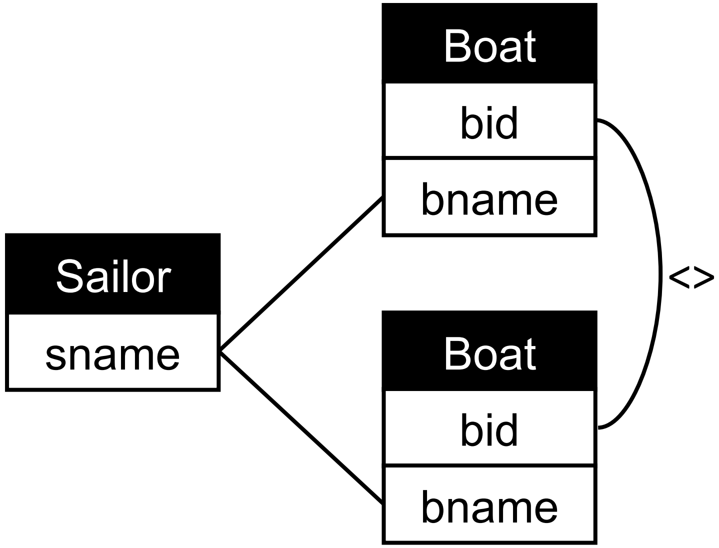




Q: "There is a sailor who shares
the same name with 2 different boats."

```
select exists
  (select *
  from Sailor, Boat B1, Boat B2
  where sname=B1.bname
  and sname=B2.bname
  and B1.bid<>B2.bid)
```

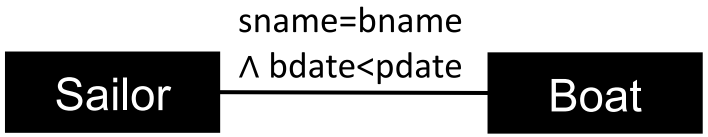


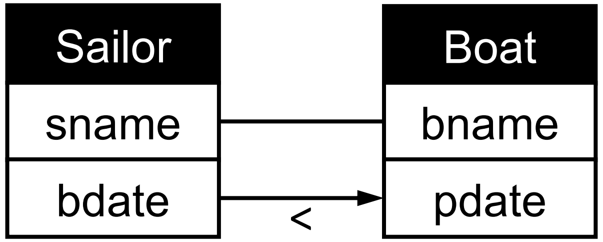


Textual vs diagrammatic representations

*Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."*

```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```

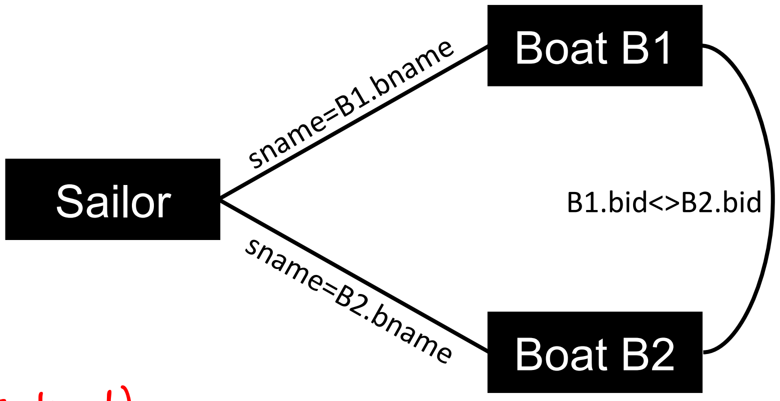




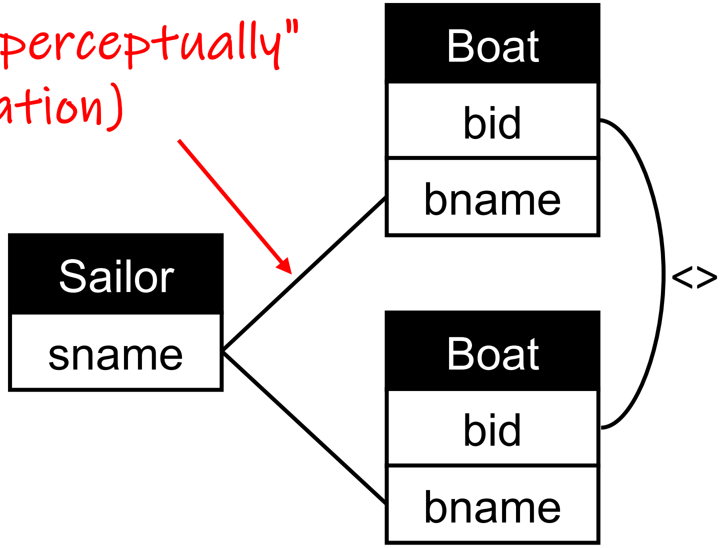
*Q: "There is a sailor who shares
the same name with 2 different boats."*

```
select exists
  (select *
  from Sailor, Boat B1, Boat B2
  where sname=B1.bname
  and sname=B2.bname
  and B1.bid<>B2.bid)
```

⇒ B1.bname=B2.bname
(the two boats share the names too!)



New fact can be inferred "perceptually"
from a path (w/o manipulation)

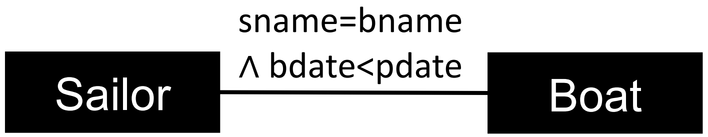


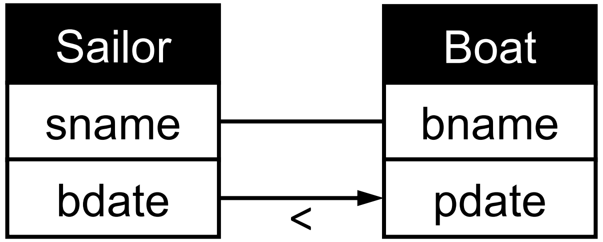
Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Textual vs diagrammatic representations

Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."

```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```

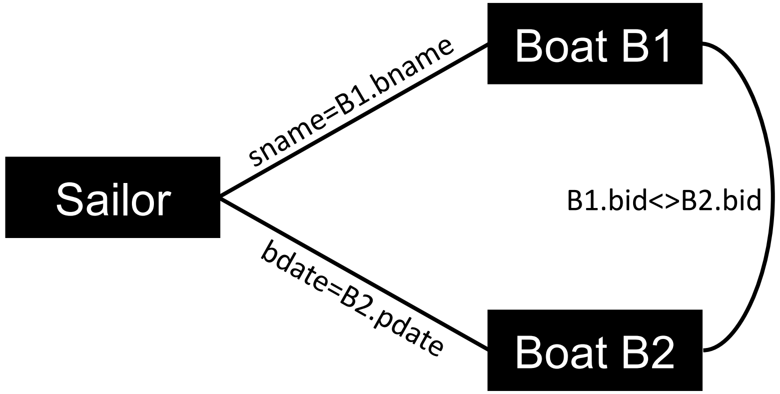




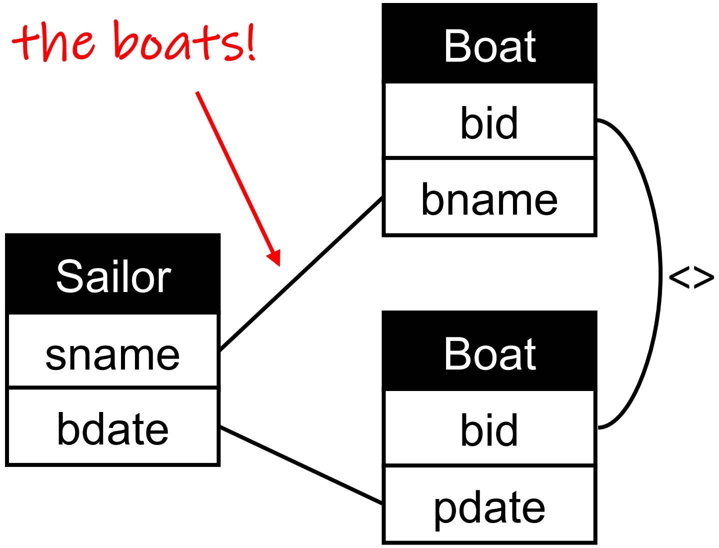
Q: "There is a sailor who shares the name with one boat
and the birthday with the purchase day of another boat."

```
select exists
  (select *
  from Sailor, Boat B1, Boat B2
  where sname=B1.bname
  and bdate=B2.pdate
  and B1.bid<>B2.bid)
```

⇒ ~~B1.bname=B2.bname~~



No path between the boats!



"Textual" vs "Diagrammatic" Representations

TEXTUAL

symbolic, linguistic, linear, sentential

DIAGRAMMATIC

visual, graphical, non-symbolic, schematic, as picture, two-dimensional

"*Diagram: a simplified drawing showing the appearance, structure, or workings of something; a schematic representation.*" [Oxford languages]

"*Diagram: a graphic design that explains rather than represents; especially: a drawing that shows arrangement and relations (as of parts)*" [Merriam-Webster]

"*Logic diagram: a two-dimensional geometric figure with spatial relations that are isomorphic with the structure of a logical statement*" [Gardner, 1958, p. 28]

"The relationships established between two sets of elements constitute a diagram." [Bertin, 1981, p. 129]

[Oxford languages]: https://www.google.com/search?q=diagram, [Merriam-Webster]: https://www.merriam-webster.com/dictionary/diagram,
[Gardner, 1958]: Martin Gardner, Logic machines and diagrams, McGraw-Hill 1958. https://archive.org/details/logicmachinesdia227gard/mode/2up ,
[Bertin, 1981]: Jacques Bertin. Graphics and graphic information-processing. de Gruyter. 1981

Now we have a shared notion of "diagrams"
Next: What are desiderata for QV?

We call those "principles". But they are not meant to be irrevocable axioms, but rather Intuitive objectives, whose formulation help us develop a shared vocabulary to discuss various approaches. They can be revisited when needed.

Algebraic Visualization Design [Kindlmann, Scheidegger 2014]

$$\Delta D \longrightarrow \Delta V$$

Goal: describe how changes in data lead to changes in the visualization

Data ⟶ Representation ⟶ Visualization

Key insight: visualizations don't act on data itself, but on *representations of data*

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Algebraic Visualization Design [Kindlmann, Scheidegger 2014]

$$\Delta D \longrightarrow \Delta V$$

Goal: describe how changes in data lead to changes in the visualization

Data ⟶ Representation ⟶ Visualization

Key insight: visualizations don't act on data itself, but on *representations of data*

$$\begin{tikzcd}
D \arrow[r, "r_1"] \arrow[d, "\alpha"'] & R \arrow[r, "v"] \arrow[d, "h"] & V \arrow[d, "\omega"] \\
D \arrow[r, "r_2"] & R \arrow[r, "v"] & V
\end{tikzcd}$$

Formulate 3 "algebraic" design principles in the language of a commutative diagram.

Vertical arrows represent transformations. "no difference" expressed as identity transformation, e.g. $\alpha = I$ for $\alpha(D) = D$

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data $\longrightarrow$ Representation
$\longrightarrow$ Visualization

P1

$$
\begin{tikzcd}
 & R \arrow[r, "v"'] \arrow[dd, "h"] & V \arrow[dd, "\omega"] \\
D \arrow[ur, "r_1"] \arrow[dr, "r_2"'] & & \\
 & R \arrow[r, "v"'] & V
\end{tikzcd}
$$

P2

$$
\begin{tikzcd}
D \arrow[dr, "v"] \arrow[dd, "\alpha"'] & \\
 & V \\
D \arrow[ur, "v"'] &
\end{tikzcd}
$$

P3

$$
\begin{tikzcd}
D \arrow[r, "v"'] \arrow[d, "\alpha"'] & V \arrow[d, "\omega"] \\
D \arrow[r, "v"'] & V
\end{tikzcd}
$$

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data $\longrightarrow$ Representation
$\longrightarrow$ Visualization

P1
$$
\begin{tikzcd}
 & R \arrow[r, "v"'] \arrow[d, "h"] & V \arrow[d, "\omega"] \\
D \arrow[ur, "r_1"] \arrow[r, "r_2"'] & R \arrow[r, "v"'] & V
\end{tikzcd}
$$

P1: **PRINCIPLE OF REPRESENTATION INVARIANCE**

A different representation, for the same data, does not lead to a different visualization. ($\alpha = I_D \Rightarrow \omega = I_V$)

Reminiscent of logical data independence (however you normalize, you get the same information)

P2
$$
\begin{tikzcd}
D \arrow[dr, "v"] \arrow[dd, "\alpha"'] & \\
 & V \\
D \arrow[ur, "v"'] &
\end{tikzcd}
$$

P3
$$
\begin{tikzcd}
D \arrow[r, "v"'] \arrow[d, "\alpha"'] & V \arrow[d, "\omega"] \\
D \arrow[r, "v"'] & V
\end{tikzcd}
$$

Example of P1: Representation Invariance for InfoViz

Vis methods use **computational** representation

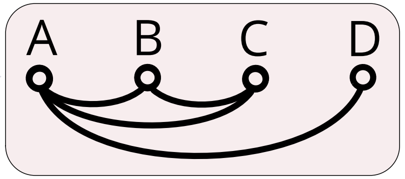
Data → Representation → Visual

Underlying thing of interest → How we can measure or store it on computer

"Show data variation, not design variation" [Tufte 1983]

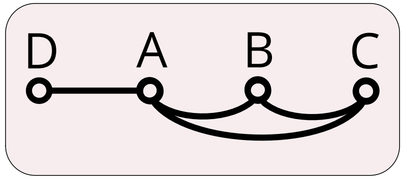
(a graph)

V=(A,B,C,D);
E=(A-B,B-C,A-C,A-D)



not equal: bug?

V=(D,A,B,C);
E=(A-B,B-C,A-C,A-D)



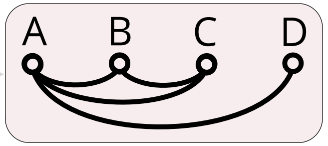
Invariance example: Graph layout

Representation: **lists** of verts, edges

Data: a graph on 4 vertices

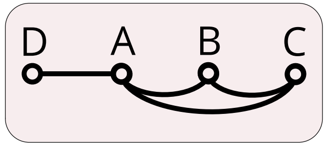
D (α=1)

R $\xrightarrow{v}$



h: permute vert list

R $\xrightarrow{v}$



ω**≠1**: layout depends on vertex ordering

Invariance example: Graph layout

Representation: **lists** of verts, edges

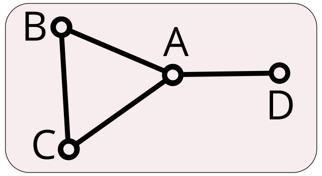
Data: a graph on 4 vertices

D (α=1)

R, R

h: permute vert list

v, v



ω**=1**: with **order-invariant** layout

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data $\longrightarrow$ Representation
$\longrightarrow$ Visualization

P1
$$\begin{tikzcd} & R \arrow[r, "v"'] \arrow[d, "h"] & V \arrow[d, "\omega"] \\ D \arrow[ur, "r_1"] \arrow[dr, "r_2"'] & R \arrow[r, "v"'] & V \\ & & \end{tikzcd}$$

P1: **PRINCIPLE OF REPRESENTATION INVARIANCE**

A different representation, for the same data, does not lead to a different visualization. ($\alpha = I_D \Rightarrow \omega = I_V$)

P2
$$\begin{tikzcd} D \arrow[dr, "v"] \arrow[dd, "\alpha"'] & \\ & V \\ D \arrow[ur, "v"'] & \end{tikzcd}$$

P2: **UNAMBIGUOUS DATA DEPICTION PRINCIPLE**

"An interesting α applied to the data should induce a non-trivial ω." ($\omega = I_V \Rightarrow \alpha = I_D$)

P3
$$\begin{tikzcd} D \arrow[r, "v"'] \arrow[d, "\alpha"'] & V \arrow[d, "\omega"] \\ D \arrow[r, "v"'] & V \end{tikzcd}$$

P3: **CORRESPONDENCE PRINCIPLE** ("congruence")
"ω somehow makes sense, given α." ($\alpha \cong \omega$)
(also: noticeable, "meaningful" changes)

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data ⟶ Representation
⟶ Visualization

We will adapt these 3 "principles", originally meant for InfoViz, instead to Query Visualization

P1
$$\begin{tikzcd} & R \arrow[r, "v"'] \arrow[d, "h"] & V \arrow[d, "\omega"] \\ D \arrow[ur, "r_1"] \arrow[r, "r_2"'] & R \arrow[r, "v"'] & V \end{tikzcd}$$

P1: **Principle of representation Invariance**

A different representation, for the same data, does not lead to a different visualization. ($\alpha = I_D \Rightarrow \omega = I_V$)

→ 4 bijection principles for Query Visualization

P2
$$\begin{tikzcd} D \arrow[dr, "v"] \arrow[d, "\alpha"'] & \\ D \arrow[r, "v"'] & V \end{tikzcd}$$

P2: **Unambiguous data depiction principle**

"An interesting α applied to the data should induce a non-trivial ω." ($\omega = I_V \Rightarrow \alpha = I_D$)

P3
$$\begin{tikzcd} D \arrow[r, "v"'] \arrow[d, "\alpha"'] & V \arrow[d, "\omega"] \\ D \arrow[r, "v"'] & V \end{tikzcd}$$

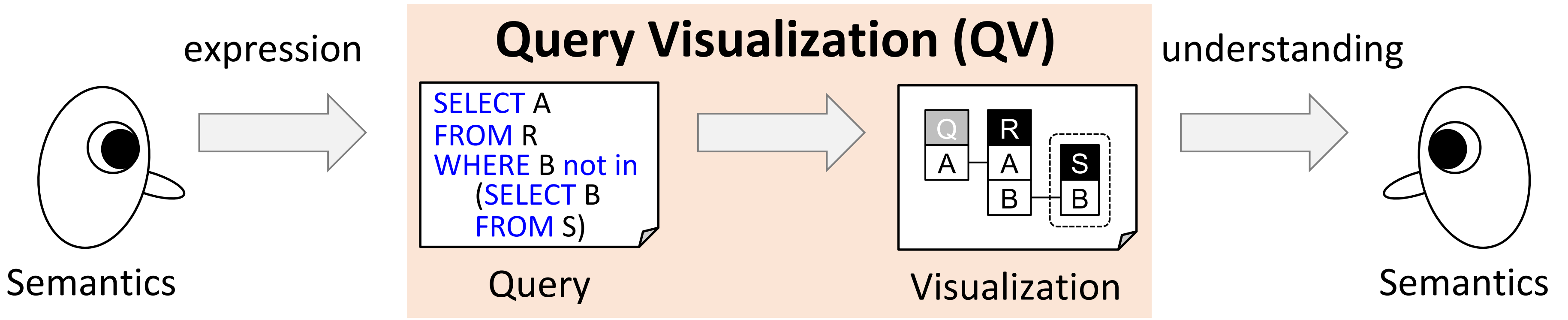
→ 7 correspondence principles for Query Visualiz.

P3: **Correspondence principle** ("congruence")

"ω somehow makes sense, given α." ($\alpha \cong \omega$)

(also: noticeable, "meaningful" changes)

An Algebraic Framework for Query Visualization

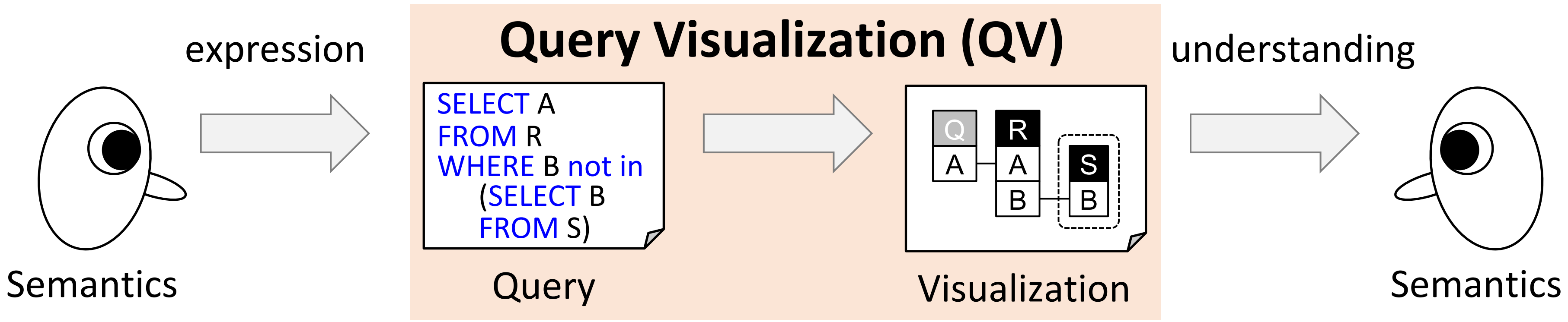


Query as Text

Visual Query Representation

$$\begin{tikzcd} q \arrow[r, "r"] \arrow[d, "\Delta q"] & v \arrow[d, "\Delta v"] \\ q' \arrow[r, "r"] & v' \end{tikzcd}$$

An Algebraic Framework for Query Visualization



Semantics	Query as Text	Visual Query Representation

$$\begin{tikzcd} s \arrow[r, "e_1"] \arrow[d, "\Delta s"] & q \arrow[r, "r"] \arrow[d, "\Delta q"] & v \arrow[d, "\Delta v"] \\ s' \arrow[r, "e_2"] & q' \arrow[r, "r"] & v' \end{tikzcd}$$

4 Bijection principles of Query Visualization

P1. Representation invariance

$$\begin{tikzcd} & q \arrow[r, "r"] \arrow[dd, "\Delta q"] & v \arrow[dd, "\cancel{\Delta v}"] \\ s \arrow[ur, "e"] \arrow[dr] & & \\ & q' \arrow[r] & v' \end{tikzcd}$$

P2. Unambiguous

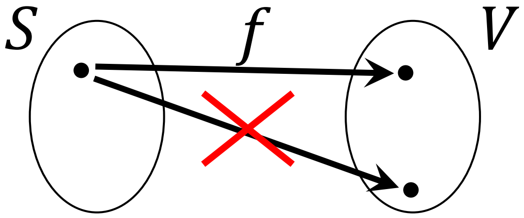
$$\begin{tikzcd} s \arrow[r] \arrow[dd, "\cancel{\Delta s}"] & q \arrow[dr] \arrow[dd] & \\ & & v \\ s' \arrow[r] & q' \arrow[ur] & \end{tikzcd}$$

4 Bijection principles of Query Visualization

More on this controversial principle later (thus "seven")

P1. REPRESENTATION INVARIANCE

$$\begin{tikzcd} & q \arrow[r, "r"] \arrow[d, "\Delta q"] & v \arrow[d, "\cancel{\Delta v}"] \\ s \arrow[ur, "e"] \arrow[r] & q' \arrow[r] & v' \end{tikzcd}$$

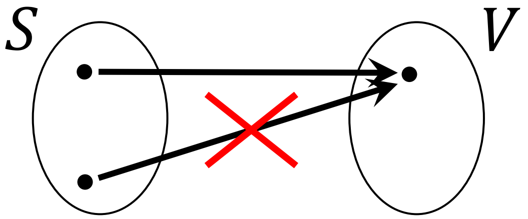


FUNCTION (instead of Relation)

P7:
Visualizations abstract away from syntax details ("syntactic invariance"): Each query semantics gets mapped to exactly one visualization ΔS=0 ⇒ ΔV=0

P2. UNAMBIGUOUS

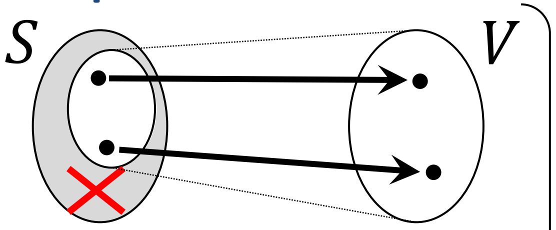
$$\begin{tikzcd} s \arrow[r] \arrow[d, "\cancel{\Delta s}"] & q \arrow[d] \arrow[dr] & \\ s' \arrow[r] & q' \arrow[r] & v \end{tikzcd}$$



INJECTIVE (preserve distinction)

P2:
Visualizations are unambiguous: Every visualization represents at most one query semantics

4 Bijection principles of Query Visualization

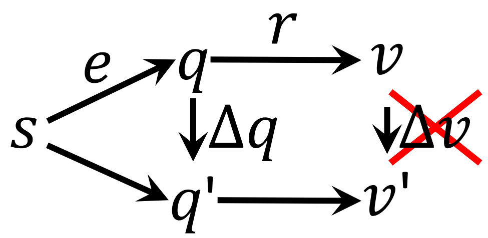


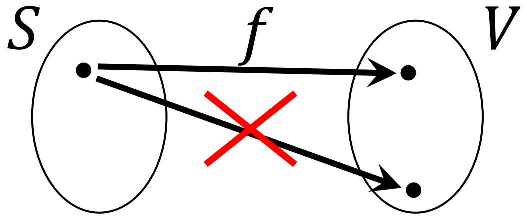
TOTAL (COMPLETE, instead of Partial)

P1:
The visual query representation is <u>relationally complete</u>: every query semantics can be represented

TOTAL FUNCTION

P1. REPRESENTATION INVARIANCE

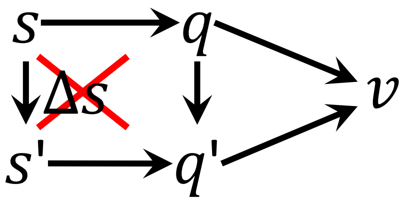


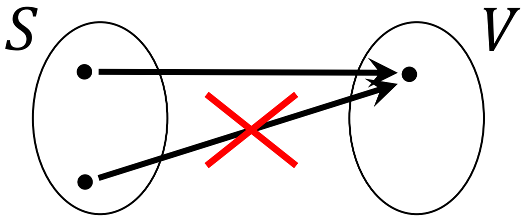


FUNCTION (instead of Relation)

P7:
Visualizations abstract away from syntax details ("<u>syntactic invariance</u>"): Each query semantics gets mapped to exactly one visualization $\Delta S=0 \Rightarrow \Delta V=0$

P2. UNAMBIGUOUS





INJECTIVE (preserve distinction)

P2:
Visualizations are <u>unambiguous</u>: Every visualization represents at most one query semantics

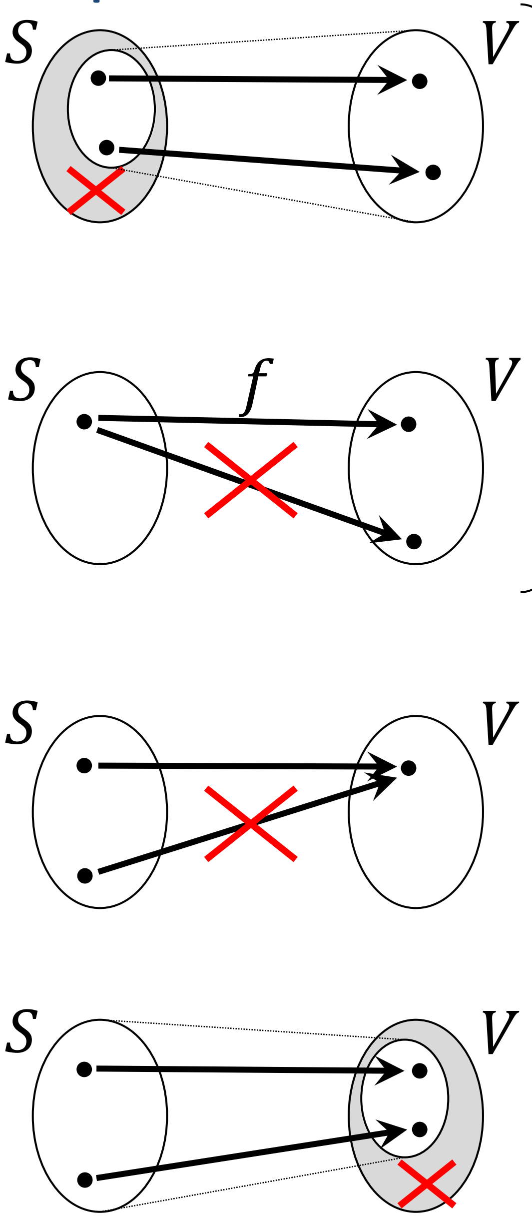
4 Bijection principles of Query Visualization

P1. Representation invariance

$$\begin{tikzcd} & q \arrow[r, "r"] \arrow[dd, "\Delta q"] & v \arrow[dd, "\cancel{\Delta v}"] \\ s \arrow[ur, "e"] \arrow[dr] & & \\ & q' \arrow[r] & v' \end{tikzcd}$$

P2. Unambiguous

$$\begin{tikzcd} s \arrow[r] \arrow[dd, "\cancel{\Delta s}"] & q \arrow[dr] \arrow[dd] & \\ & & v \\ s' \arrow[r] & q' \arrow[ur] & \end{tikzcd}$$



Total (Complete, instead of Partial)

Total Function

Function (instead of Relation)

Injective (preserve distinction)

Surjective (covering)

P1:
The visual query representation is <u>relationally complete</u>: every query semantics can be represented

P7:
Visualizations abstract away from syntax details ("<u>syntactic invariance</u>"): Each query semantics gets mapped to exactly one visualization ΔS=0 ⇒ ΔV=0

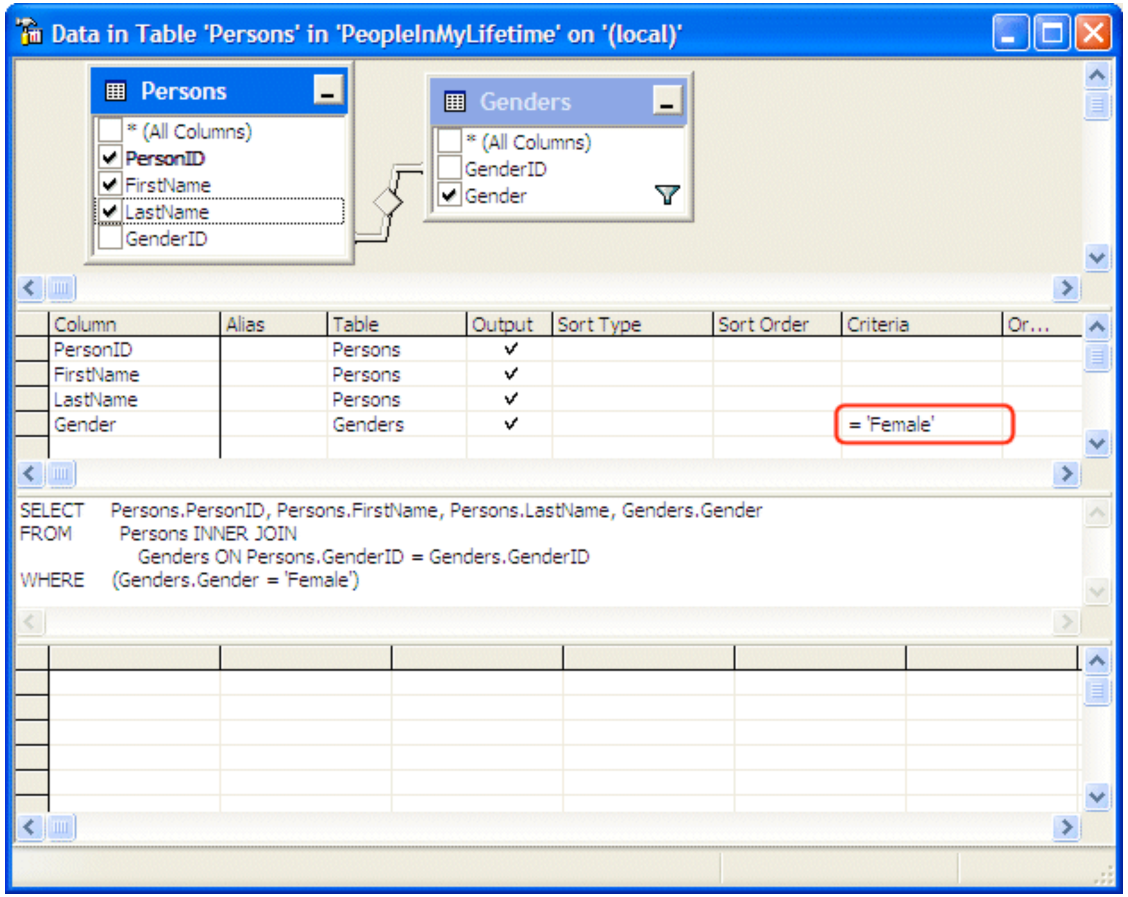
P2:
Visualizations are <u>unambiguous</u>: Every visualization represents at most one query semantics

P3:
Visualizations are <u>sound</u>: Every valid visualization has some valid interpretation (query semantics)

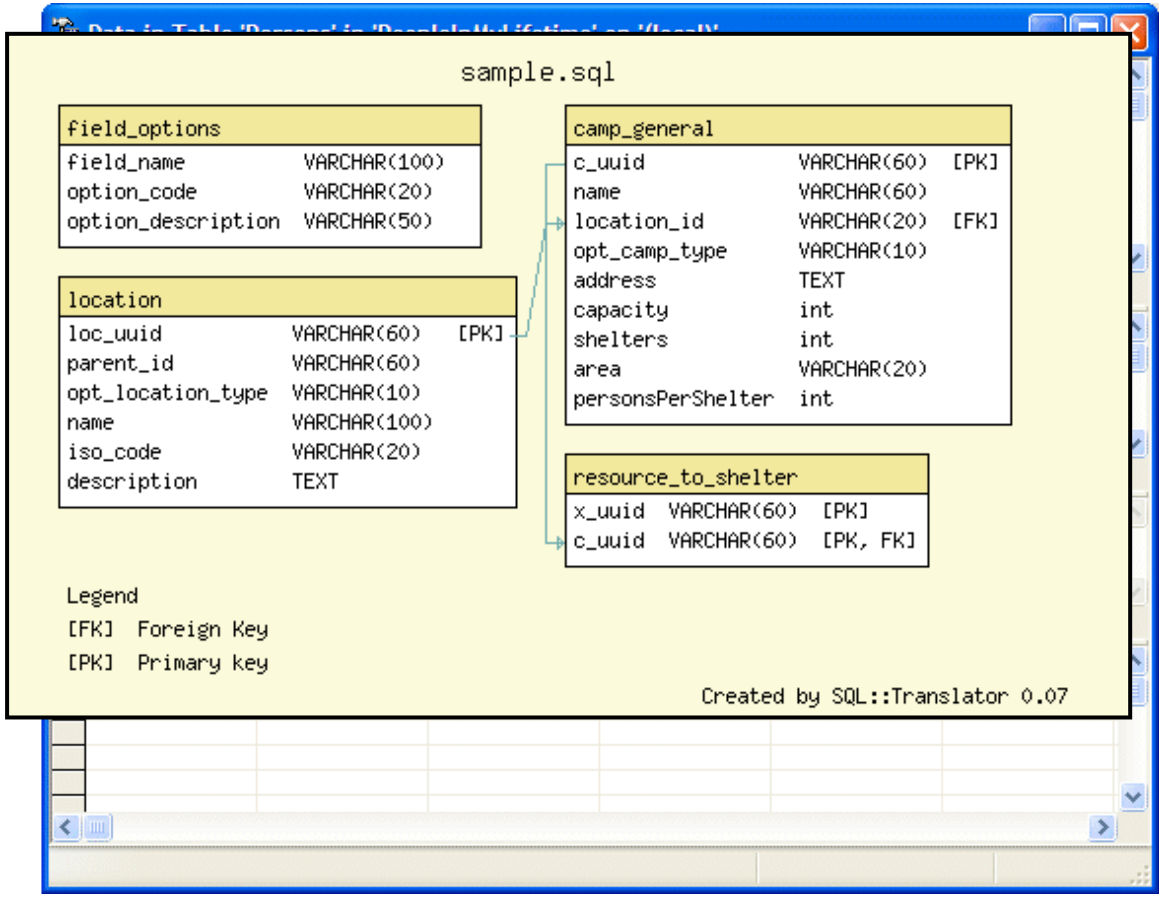
Those were the 4 bijection principles

Next: 7 additional "correspondence" principles

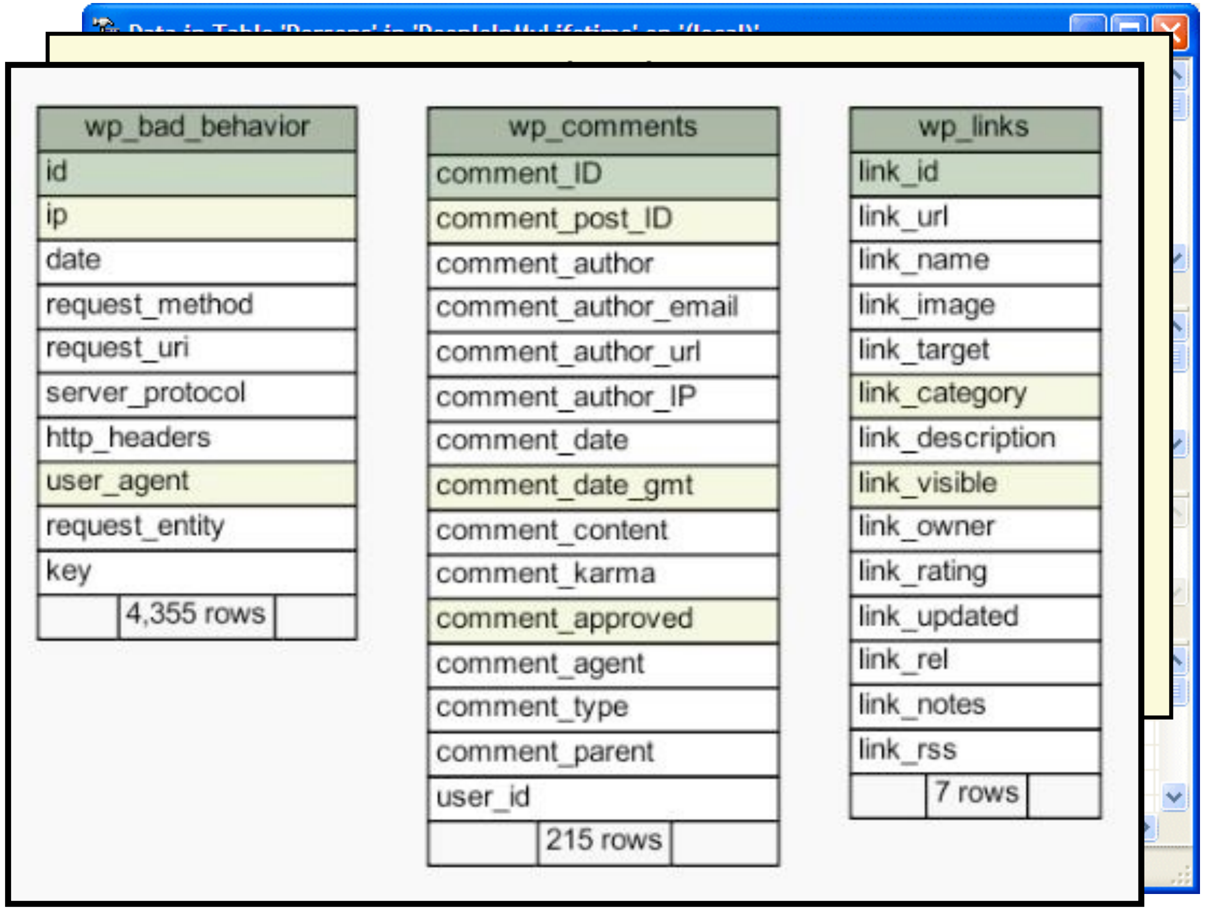
P4: Existing visual metaphors as starting point



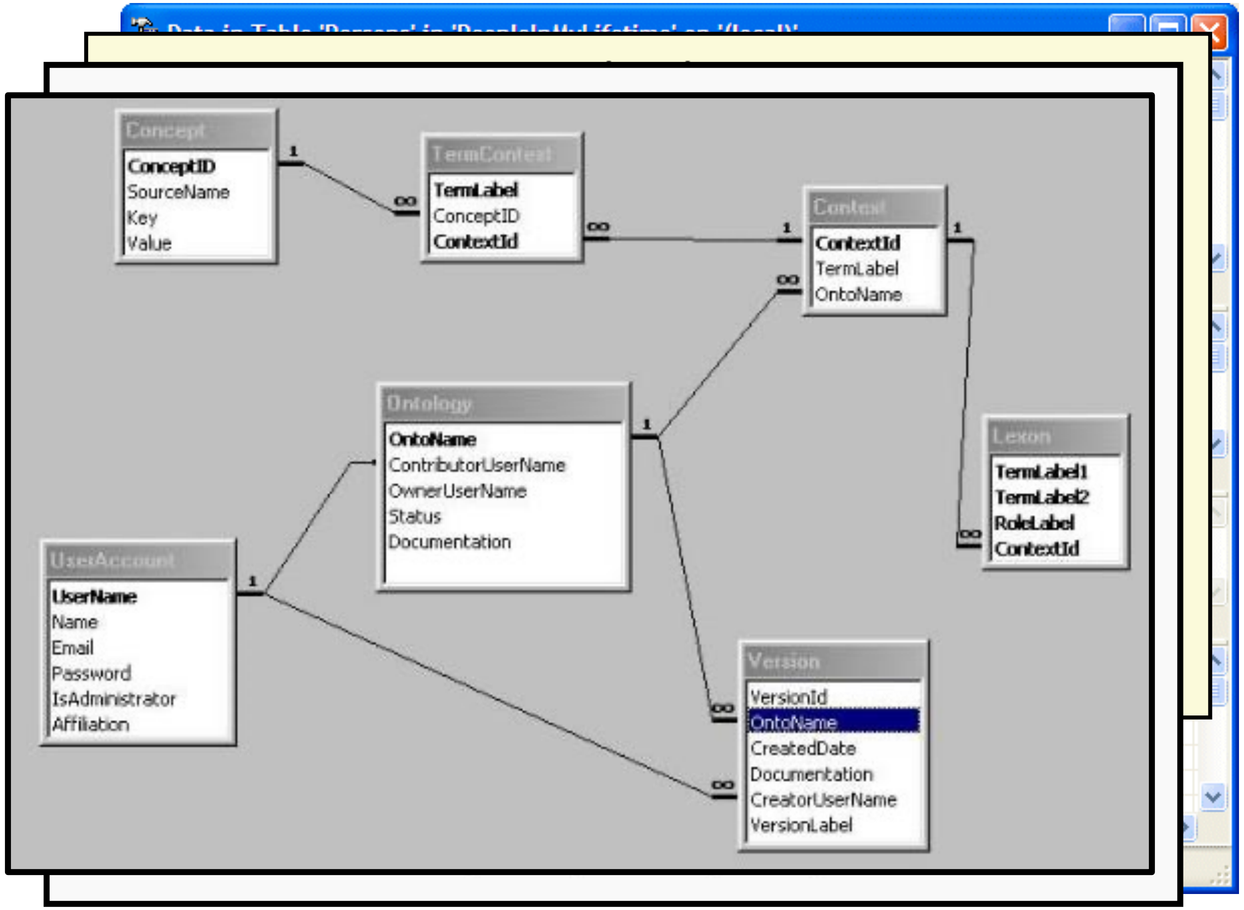
P4: Existing visual metaphors as starting point



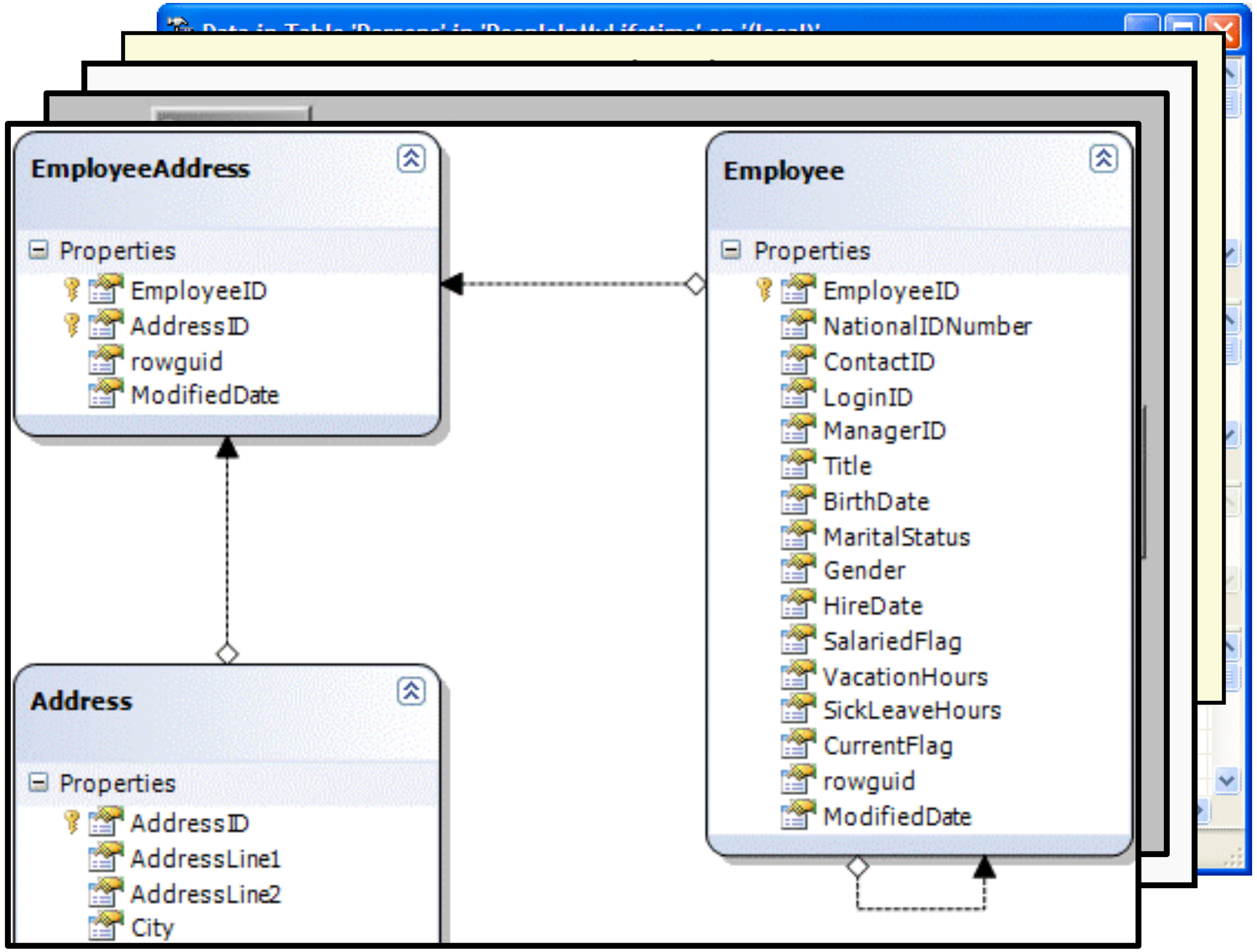
P4: Existing visual metaphors as starting point



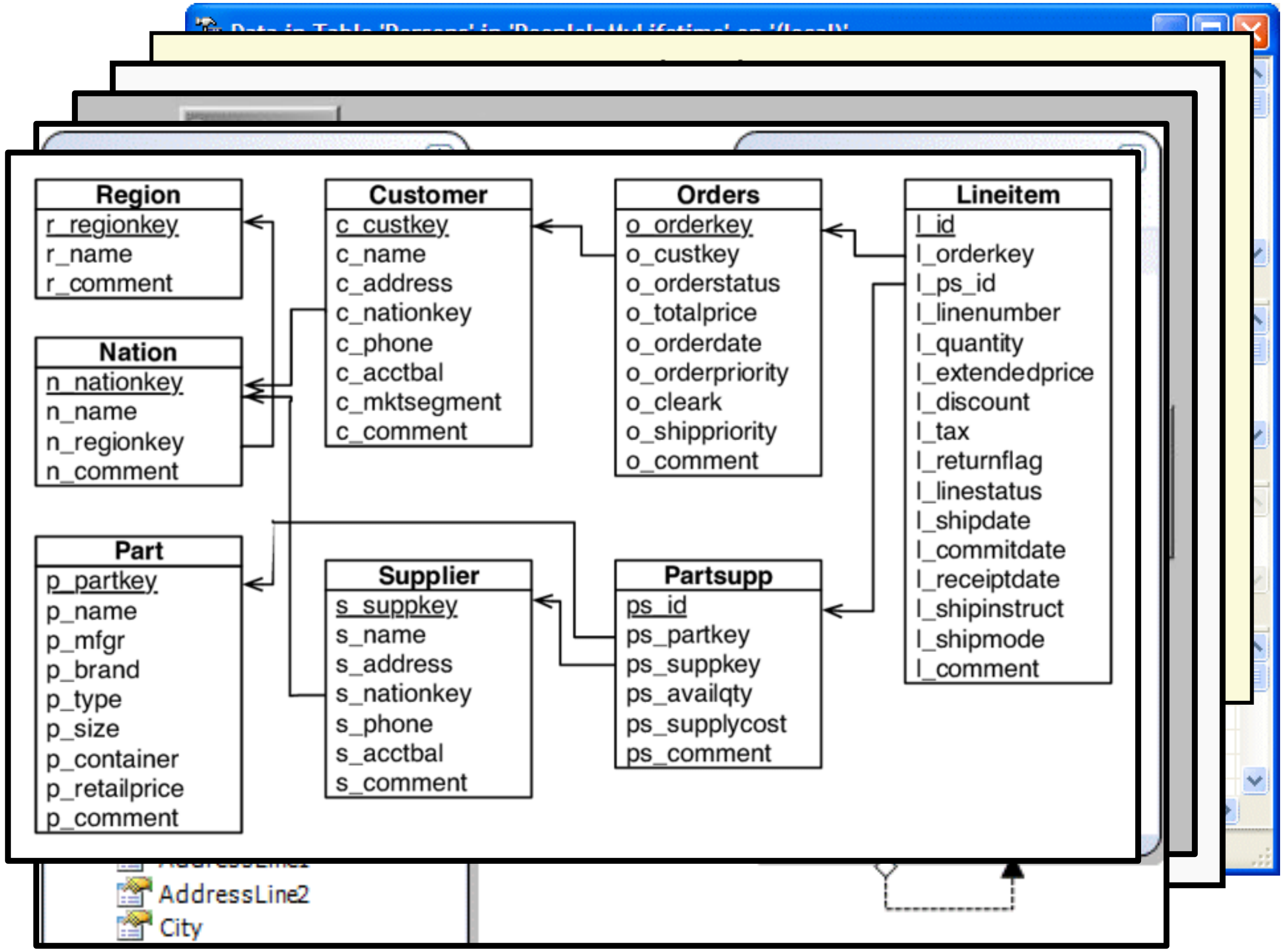
P4: Existing visual metaphors as starting point



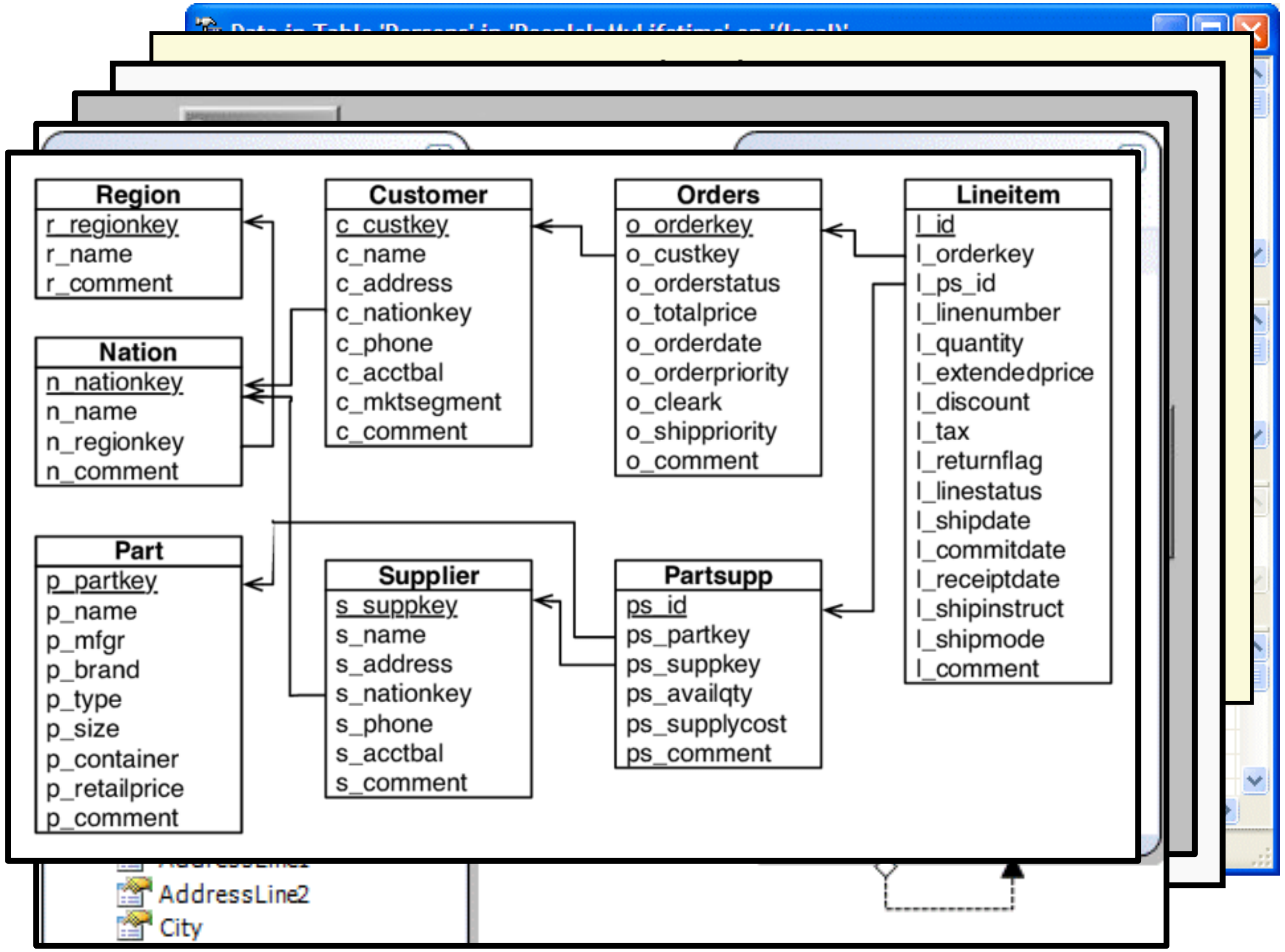
P4: Existing visual metaphors as starting point

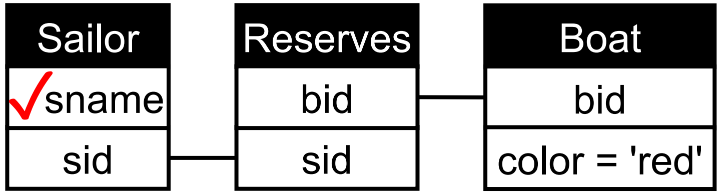


P4: Existing visual metaphors as starting point



P4: Existing visual metaphors as starting point





Q: *"Find sailors who reserved a red boat."*

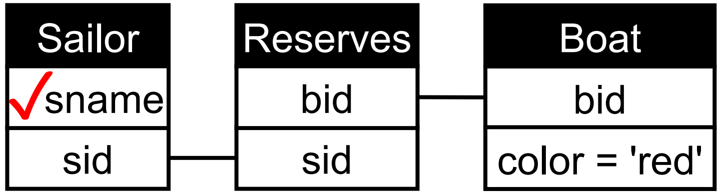
P4: Start from known visual UML metaphors for relational schemas

Conjunctive queries resemble schema notation with FK/PK constraints

P5: Compositionality of the relational model

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor, ∃r∈Reserves, ∃b∈Boat [
q.sname=s.sname ∧ r.sid=s.sid ∧
b.bid=r.bid ∧ b.color='red']}

Datalog

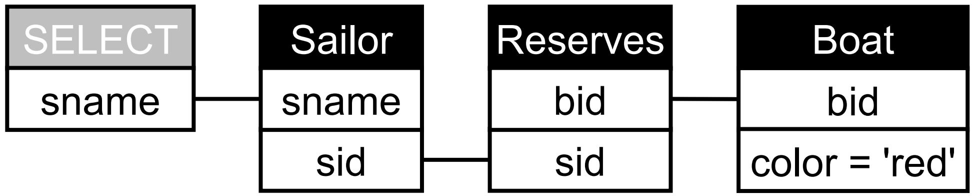
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P5: Compositionality of the relational model

Q: "Find sailors who reserved a red boat."

P5: Show the output relation

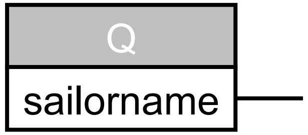
```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



Relational queries are compositional:
- Input are relations (tables)
- Output are tables

TRC (Tuple Relational Calculus)

$\{$q.sname $\mid \exists s \in$ Sailor, $\exists r \in$ Reserves, $\exists b \in$ Boat [
q.sname=s.sname $\wedge$ r.sid=s.sid $\wedge$
b.bid=r.bid $\wedge$ b.color='red']$\}$



Explicit output table also allow renaming of tables and attributes

Datalog

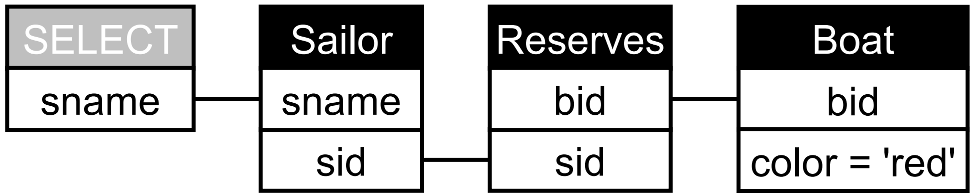
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P5: Compositionality of the relational model

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

P5: Show the output relation

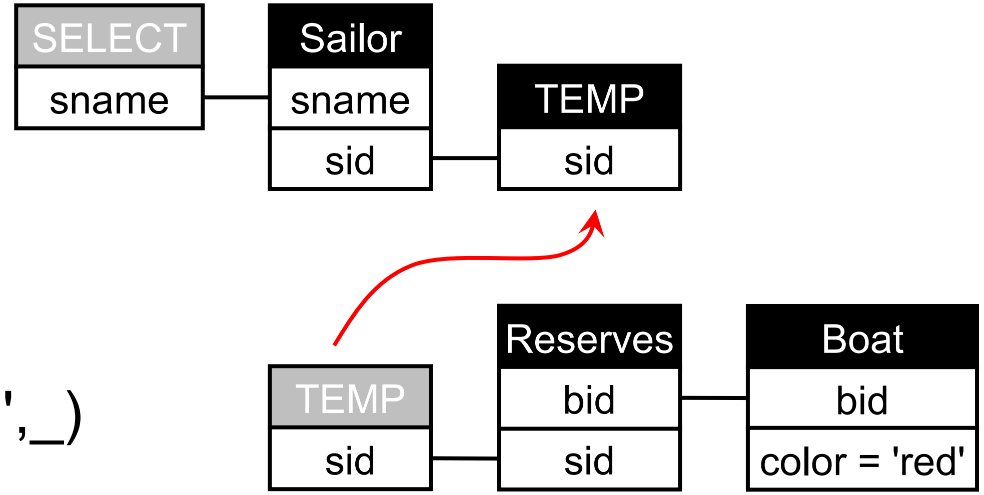


Relational composition:

- One may want to use/define intermediate relations

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor, ∃r∈Reserves, ∃b∈Boat [
q.sname=s.sname ∧ r.sid=s.sid ∧
b.bid=r.bid ∧ b.color='red']}



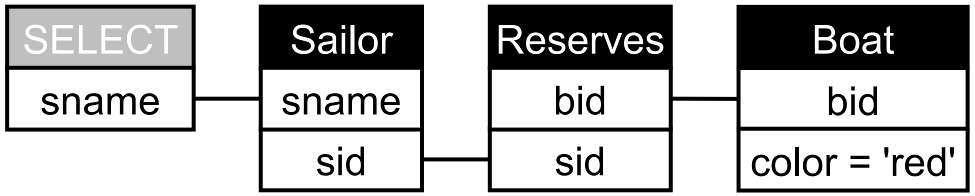
Datalog

Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P6: Progressive visual complexity

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



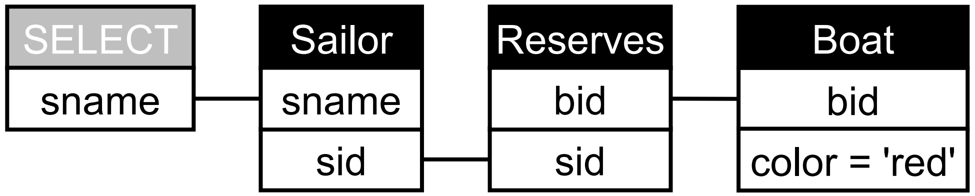
Q: "Find sailors who reserved only red boats."

?

P6: Progressive visual complexity

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



+67% more SQL text

Q: "Find sailors who reserved only red boats."

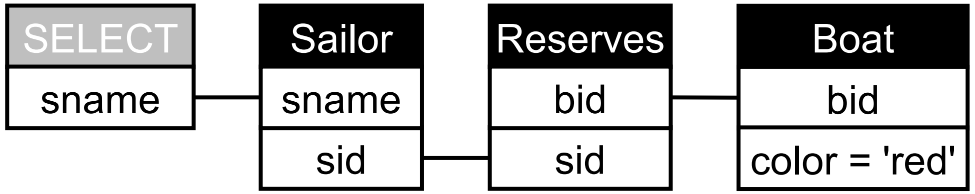
```
select distinct S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

P6: Progressive visual complexity

P6: Conjunctive queries are simplest, then gradually add visual metaphors

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

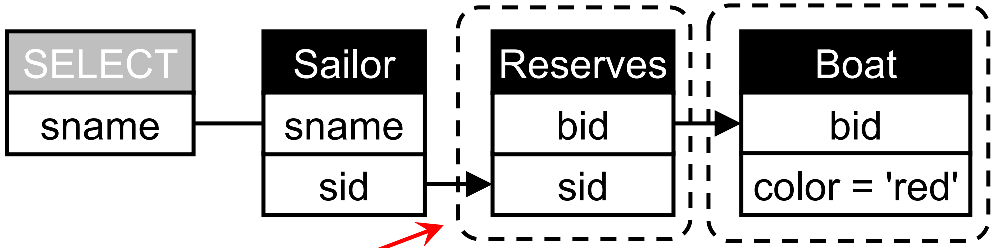


+67% more SQL text

Q: "Find sailors who reserved only red boats."

```
select distinct S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

+13% more "visual symbols"



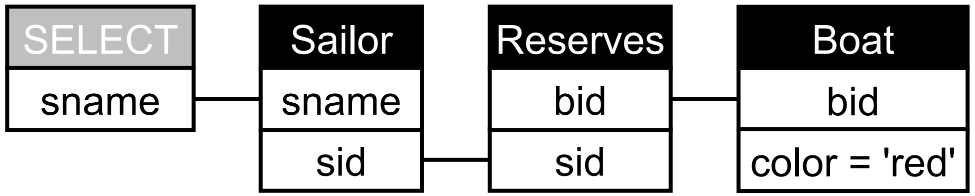
interprete query block with dashed block as negated

P6: Progressive visual complexity

P6: Conjunctive queries are simplest, then gradually add visual metaphors

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

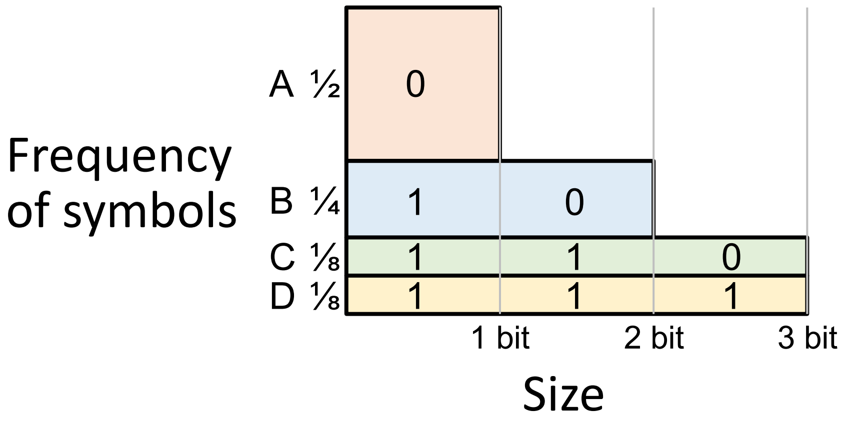


+67% more SQL text

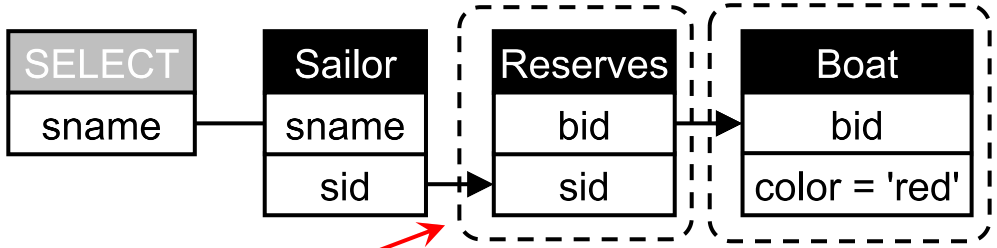
Q: "Find sailors who reserved only red boats."

```
select distinct S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

Variable-length entropy codes



+13% more "visual symbols"



interprete query block with dashed block as negated

P7: Abstract away from syntax details

Are these two SQL queries identical?

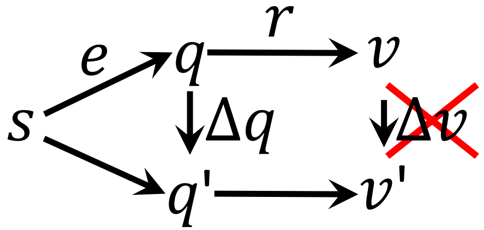
```
select distinct S.sname
from Sailor S, Reserves R
where S.sid=R.sid
```

```
select distinct S.sname
from Sailor S
where exists (
   select S.sname
   from Reserves R
   where S.sid=R.sid)
```

Example queries from "Databases will visualize queries too", by G, VLDB 2011. https://doi.org/10.14778/3402755.3402805

P7: Abstract away from syntax details

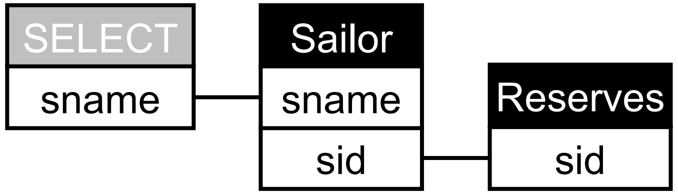
P7. SYNTACTIC INVARIANCE



Q: "Find sailors who made some reservation."

```
select distinct S.sname
from Sailor S, Reserves R
where S.sid=R.sid
```

```
select distinct S.sname
from Sailor S
where exists (
   select S.sname
   from Reserves R
   where S.sid=R.sid)
```



These two SQL queries are identical!

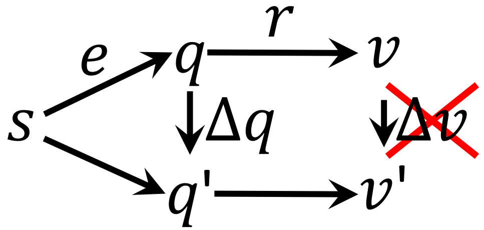
<u>P7</u>: Ignore peculiarities of SQL and focus on common logical core of relational queries

<u>TRC (Tuple Relational Calculus)</u>

{q(sname) | ∃s∈Sailor, ∃r∈Reserves [q.sname=s.sname ∧ r.sid=s.sid]}
{q(sname) | ∃s∈Sailor [q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid]]}

P7: Abstract away from syntax details

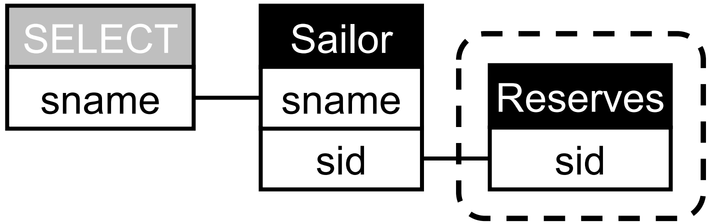
P7. SYNTACTIC INVARIANCE



Q: "Find sailors who made no reservation."

```
select distinct sname
from Sailor
where sid not in (
   select sid
   from Reserves R)
```

```
select distinct S.sname
from Sailor S
where not exists (
   select S.sname
   from Reserves R
   where S.sid=R.sid)
```



These two SQL queries are also identical
(if R.sid contains no NULL value ...)

P7: Ignore peculiarities of SQL and focus on common logical core of relational queries

TRC (Tuple Relational Calculus)

{q(sname) | ∃s∈Sailor [q.sname=s.sname ∧ ¬(∃r∈Reserves[r.sid=s.sid])]}

Example queries from "Databases will visualize queries too", by G, VLDB 2011. https://doi.org/10.14778/3402755.3402805
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

P8: Expose (and not hide) relational patterns

Reserves
sid
bid

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))
  and not exists
    (select *
    from Reserves R5
    where R5.sid= R1.sid
    and not exists
      (select *
      from Reserves R6
      where R6.sid = R2.sid
      and R6.bid= R5.bid)))
```

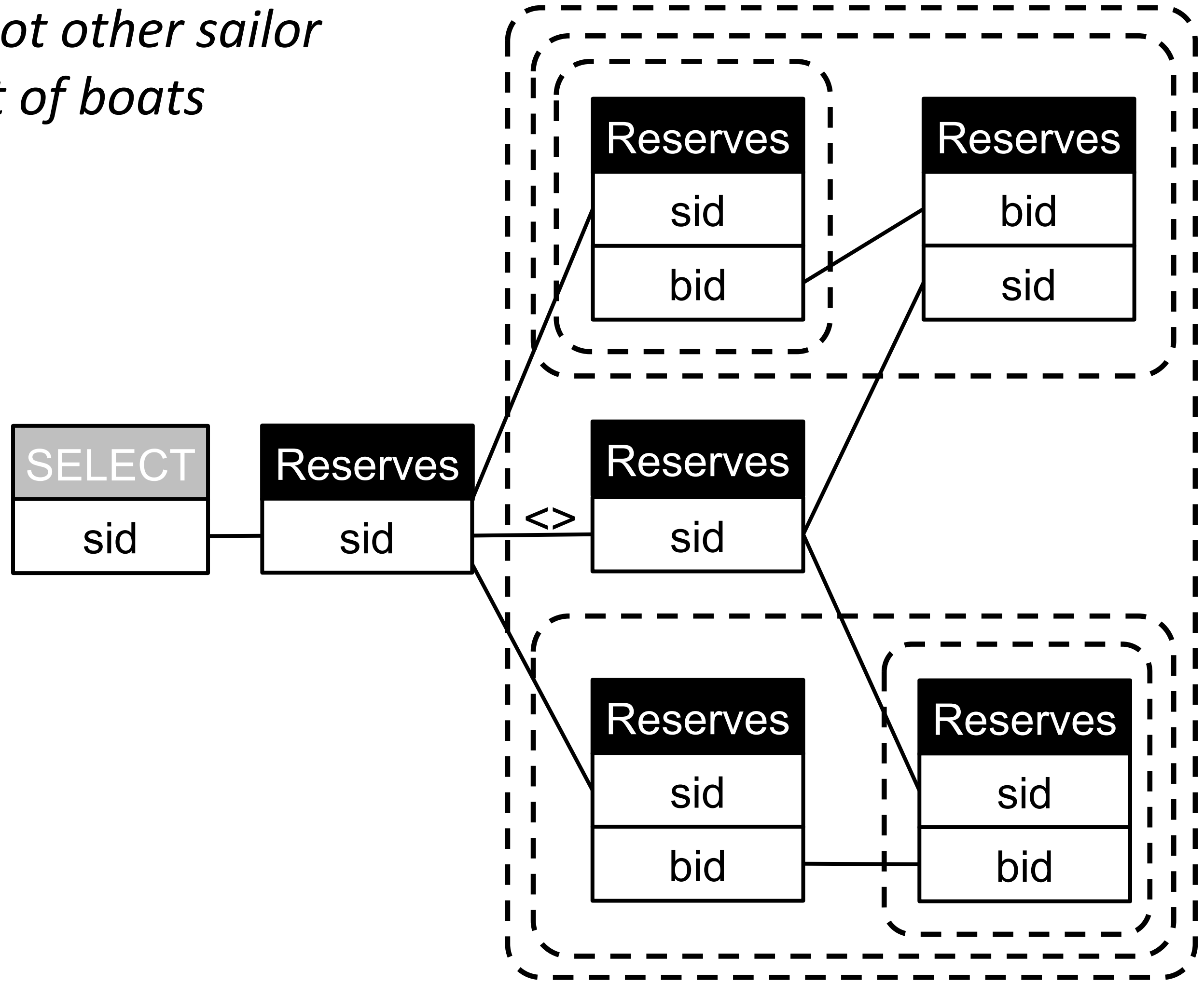
?

P8: Expose (and not hide) relational patterns

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))
  and not exists
    (select *
    from Reserves R5
    where R5.sid= R1.sid
    and not exists
      (select *
      from Reserves R6
      where R6.sid = R2.sid
      and R6.bid= R5.bid)))
```

Q: "*Find sailors with a unique set of reserved boats*"

= *Find sailors s.t. there is not other sailor that reserved the same set of boats*

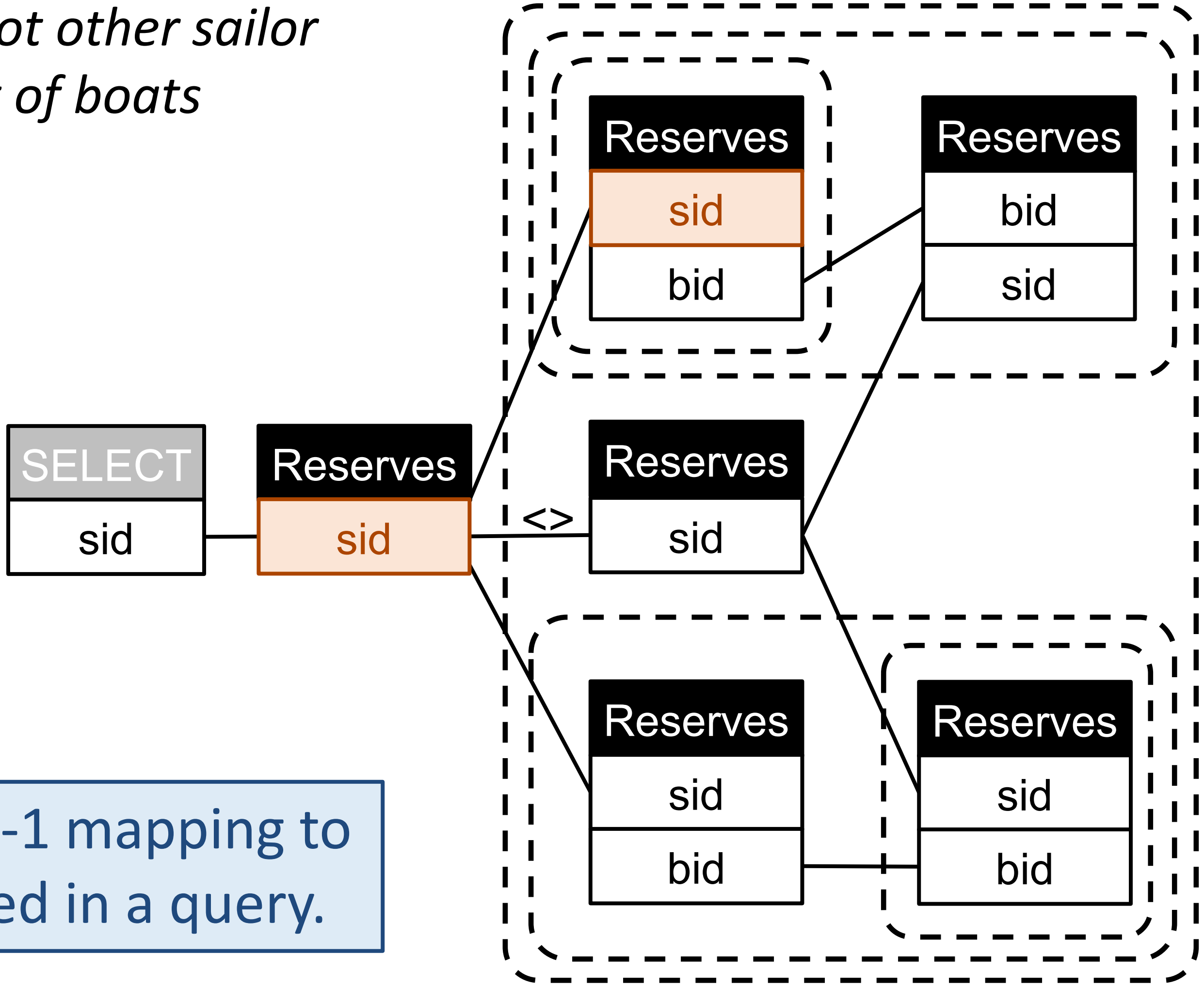


P8: Expose (and not hide) relational patterns

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))
  and not exists
    (select *
    from Reserves R5
    where R5.sid= R1.sid
    and not exists
      (select *
      from Reserves R6
      where R6.sid = R2.sid
      and R6.bid= R5.bid)))
```

Q: "*Find sailors with a unique set of reserved boats*"

= *Find sailors s.t. there is not other sailor that reserved the same set of boats*



P8: Preserve a 1-1 mapping to the relations used in a query.

For a formal definition of "relational query patterns" see "Relational Diagrams: a pattern-preserving diagrammatic representation of non-disjunctive Relational Queries". https://arxiv.org/pdf/2203.07284

P8: Expose (and not hide) relational patterns

Are these two SQL queries identical?

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2, Reserves R3
  where R2.sid = R3.sid
  and R2.bid<R3.bid
  and R2.sid=R1.sid)
```

```
select sid
from Reserves
group by sid
having count(distinct bid)=1
```

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

P8: Expose (and not hide) relational patterns

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2, Reserves R3
  where R2.sid = R3.sid
  and R2.bid<R3.bid
  and R2.sid=R1.sid)
```

```
select sid
from Reserves
group by sid
having count(distinct bid)=1
```

These two SQL queries give the same answers, but arguably use very different patterns (that goes beyond syntax). The underlying logic differs.

P8: Preserve a 1-1 mapping to the relations used in a query.

Contrast this principle with P7: "Abstract away from syntax details"

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

P9: Minimal visual complexity

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```

SQL requires aliases (which implies an inconvenient indirection to the database schema)

```
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)
```

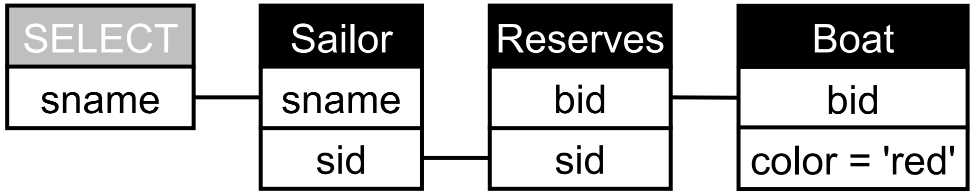
Datalog needs to show all attributes due to positional encoding (though it allows the use of "anonymous variables", via underscores)

P9: Minimal visual complexity

P9: Obey some kind of minimality criteria: only show information relevant for a query

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```



SQL requires aliases (which implies an inconvenient indirection to the database schema)

Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

Datalog needs to show all attributes due to positional encoding (though it allows the use of "anonymous variables", via underscores)

Only use as much "ink" as necessary

$$\text{Data-ink ratio} = \frac{\text{data-ink}}{\text{total ink used to print the graphic}}$$

$$= \text{proportion of a graphic's ink devoted to the non-redundant display of data-information}$$

P10: Output-oriented reading order

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```

TRC (Tuple Relational Calculus)

$\{q.sname \mid \exists s \in Sailor, \exists r \in Reserves, \exists b \in Boat\ [$
$\quad q.sname = s.sname \wedge r.sid = s.sid \wedge$
$\quad b.bid = r.bid \wedge b.color = \text{'red'}]\}$

Datalog

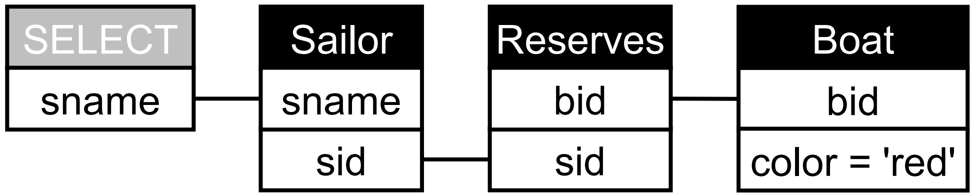
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P10: Output-oriented reading order

P10: use an output-oriented reading order (as in SQL, Datalog, calculus)

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```



Start with the output on the left!

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor, ∃r∈Reserves, ∃b∈Boat [
q.sname=s.sname ∧ r.sid=s.sid ∧
b.bid=r.bid ∧ b.color='red']}

Datalog

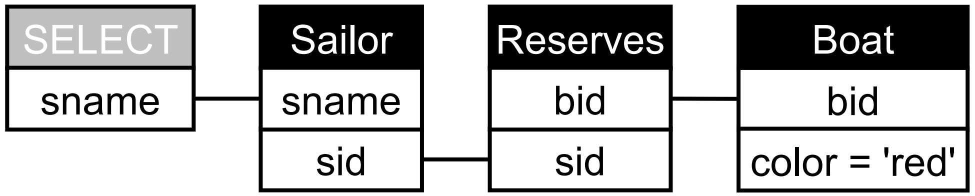
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P10: Output-oriented reading order

P10: use an output-oriented reading order (as in SQL, Datalog, calculus)

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```



Start with the output on the left!

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor, ∃r∈Reserves, ∃b∈Boat [
q.sname=s.sname ∧ r.sid=s.sid ∧
b.bid=r.bid ∧ b.color='red']}

Notice that this is notably different from typical workflow visualizations!

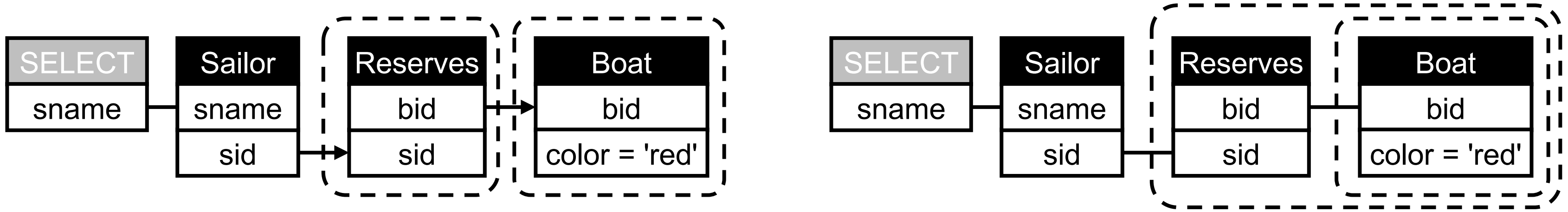
Datalog

Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P11: Logic-based visual transformations

Q: "Find sailors who reserved only red boats."

```
select distinct S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid
   and not exists
      (select *
      from Boat B
      where R.bid=B.bid
      and color = 'red'))
```



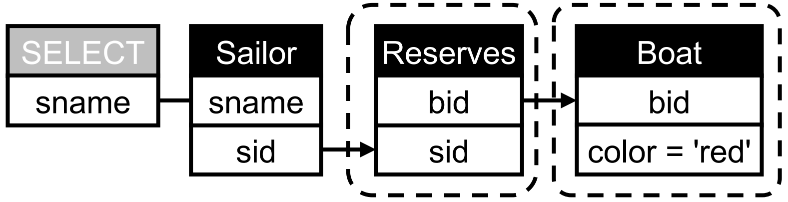
$\{q.n \mid \exists s \in \text{Sailor}[q.n=s.n \wedge \neg(\exists r \in \text{Reserves}[r.s=s.s \wedge \neg(\exists b \in \text{Boat}[b.b=r.b \wedge b.c=\text{'red'}])])]\}$

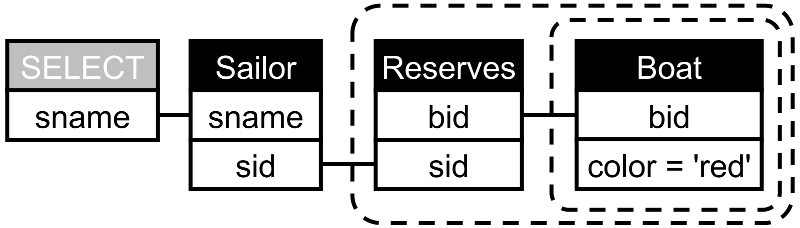
P11: Logic-based visual transformations

P11: Allow limited visual transformations that help reading nested negations

Q: "Find sailors who reserved only red boats."

```
select distinct S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid
   and not exists
      (select *
      from Boat B
      where R.bid=B.bid
      and color = 'red'))
```

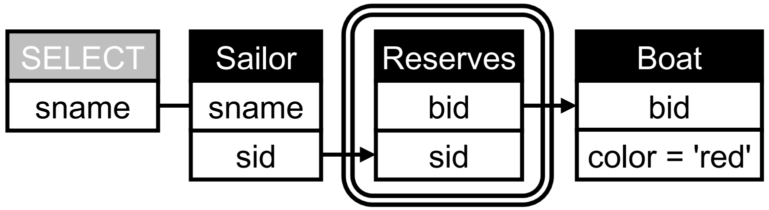


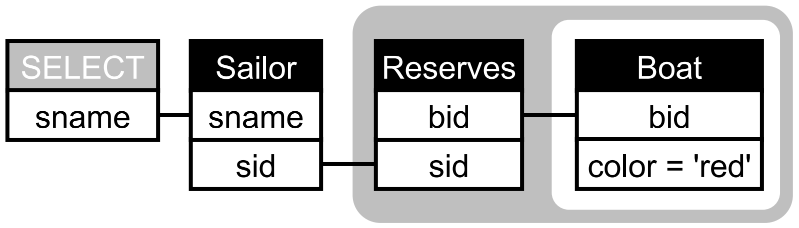


$\{q.n \mid \exists s \in Sailor[q.n=s.n \wedge \neg(\exists r \in Reserves[r.s=s.s \wedge \neg(\exists b \in Boat[b.b=r.b \wedge b.c='red'])])]\}$

$\{q.n \mid \exists s \in Sailor[q.n=s.n \wedge (\forall r \in Reserves[r.s=s.s \rightarrow (\exists b \in Boat[b.b=r.b \wedge b.c='red'])])]\}$

Double negation in logic allows a rewriting and replacing with universal quantification





Double lines represent for all ∀

Human perception can "manipulate" shaded areas without new symbols

Intended Agenda today

Please leave feedback ☺

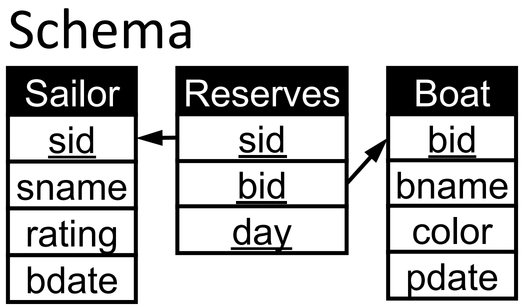
1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. Early diagrammatic representations
5. Visual Query Representations for Databases
6. Various Open Challenges

- One database schema
- 4 queries = 4 slides
- 5 Query Languages (QLs)

Example query 1

Q: "Find boats that are red or blue."

```
select distinct bname
from Boat
where color = 'red'
or color = 'blue'
```



TRC (Tuple Relational Calculus)

{q.bname | ∃b∈Boat [q.bname=b.bname ∧ (b.color='red' ∨ b.color='blue')]}

DRC (Domain Relational Calculus)

{(x) | ∃y,z,u [Boat(z,x,y,u) ∧ (y='red' ∨ y='blue')]}

{(x) | ∃y [Boat(_,x,y,_) ∧ (y='red' ∨ y='blue')]}

Anonymous variables are possible in both DRC and Datalog

Datalog

Q(x) :- Boat(_,x,'red',_)
Q(x) :- Boat(_,x,'blue',_)

Q(x) :- Boat(_,x,y,_), (y="red"; y="blue")

Disjunctions in Datalog are not standard but used in some Datalog implementations like Souffle (see https://souffle-lang.github.io/rules#disjunction)

Relational Algebra

$\sigma_{\text{color='red'} \vee \text{color='blue}} B$

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

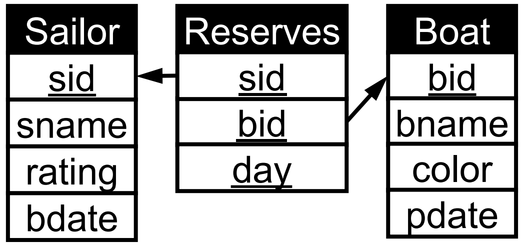
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Example query 2

Q: "Find sailors who reserved a red boat."

```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid = R.sid
and B.bid = R.bid
and color = 'red'
```

Schema



TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor, ∃r∈Reserves,
∃b∈Boat [q.sname=s.sname ∧
r.sid=s.sid ∧ b.bid=r.bid ∧ b.color='red']}

DRC (Domain Relational Calculus)

{(x) | ∃v,z,w,y,t,u,s [Sailor(v,x,z,w) ∧
Reserves(v,y,t) ∧
Boat(y,u,'red',s)}

Datalog

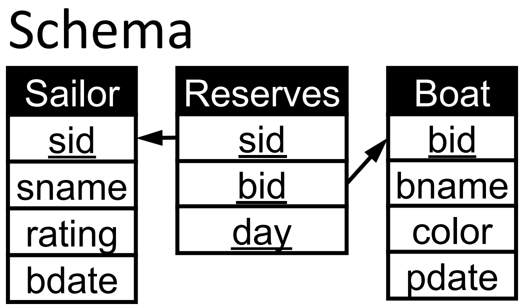
Q(x) :- Sailor(y,x,_,_), Reserves(y,z,_), Boat(z,_,'red',_)

Relational Algebra

$$\pi_{\text{sname}}(S \bowtie R \bowtie \sigma_{\text{color='red'}} B)$$

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Example query 3



Q: "Find sailors who reserved only red boats."

```
select distinct S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid = R.sid
    and not exists
        (select *
        from from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor [q.sname=s.sname ∧
¬(∃r∈Reserves [r.sid=s.sid ∧
¬(∃b∈Boat [b.bid=r.bid ∧ b.color='red'])])]}

DRC (Domain Relational Calculus)

{(x) | ∃v,z,w [Sailor(v,x,z,w) ∧
¬(∃y,t [Reserves(v,y,t) ∧
¬(∃u,s [Boat(y,u,'red',s)])])]}

Non-recursive Datalog with negation

RedBoat(z) :- Boat(z,_,'red',_)
BadSid(y) :- Reserves(y,z,_), not RedBoat(z)
Q(x) :- Sailor(y,x,_,_), not BadSid(y)

Datalog requires an intermediate view due to safety

BadSids: sailors who reserved some non-red boat.

Relational Algebra

$$\pi_{sname}(S \bowtie (\pi_{sid} S - \pi_{sid}(R \bowtie \sigma_{color='red'} B)))$$

$$\pi_{sname}(S \,\overline{\ltimes}\, \qquad \pi_{sid}(R \bowtie \sigma_{color='red'} B)))$$

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

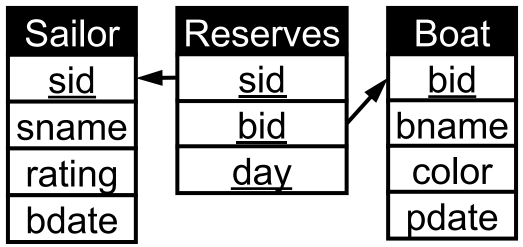
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Example query 4

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Boat B
   where color = 'red'
   and not exists
      (select *
      from Reserves R
      where S.sid=R.sid
      and R.bid=B.bid))
```

Schema



TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor [q.sname=s.sname ∧
¬(∃b∈Boat [b.color='red' ∧
¬(∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid])])]}

DRC (Domain Relational Calculus)

{(x) | ∃v,z,w [Sailor(v,x,z,w) ∧
¬(∃y,u,s [Boat(y,u,'red',s) ∧
¬(∃t [Reserves(v,y,t)])])]}

BadSids: sailors who have not reserved all red boats.

Non-recursive Datalog with negation

I(y,z) :- Reserves(y,z,_)
BadSid(y) :- Sailor(y,_,_,_), Boat(z,_,'red',_), not I(y,z,)
Q(x) :- Sailor(y,x,_,_), not BadSid(y)

Datalog requires another Sailor relation!

Relational Algebra

Notice the cross product!

$$\pi_{sname}(S\bowtie(\pi_{sid}S-\pi_{sid}((\pi_{sid}S \times \pi_{bid}\sigma_{color='red'}B) - \pi_{sid,bid}R)))$$

$$\pi_{sname}(S\,\overline{\ltimes}\,\pi_{sid}((S \times \sigma_{color='red'}B)\,\overline{\ltimes}\,\pi_{sid,bid}R)))$$

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Intended Agenda today

1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. Early diagrammatic representations
5. Visual Query Representations for Databases
6. Various Open Challenges

Skipped due to time constraints

Intended Agenda today

1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. Early diagrammatic representations
5. Visual Query Representations for Databases
6. Various Open Challenges

Visual Query Representations in the Database Literature

Please leave feedback on the tutorial page ☺

How the selection of work we see was made:
1) if highly cited or influential, or
   2a) it represents an didactically interesting type of visualizations, and
   2b) documentation was easy enough to find and use

DISCLAIMER 1: I may be missing relevant work. If you think I did, please let me know.

DISCLAIMER 2: my best effort to understand the visual representation implied by an approach, based on available information (it's surprisingly hard to create visualizations for new queries, based on paper write-ups). I may have gotten things wrong. If you spot an error, let me know where, and how it can be fixed.

DISCLAIMER 3: query composition has its own challenges separate from visualization. Thus the focus of some tools may not have been on the "visual alphabet" and have contributions other than our focus today: the visual representation.

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

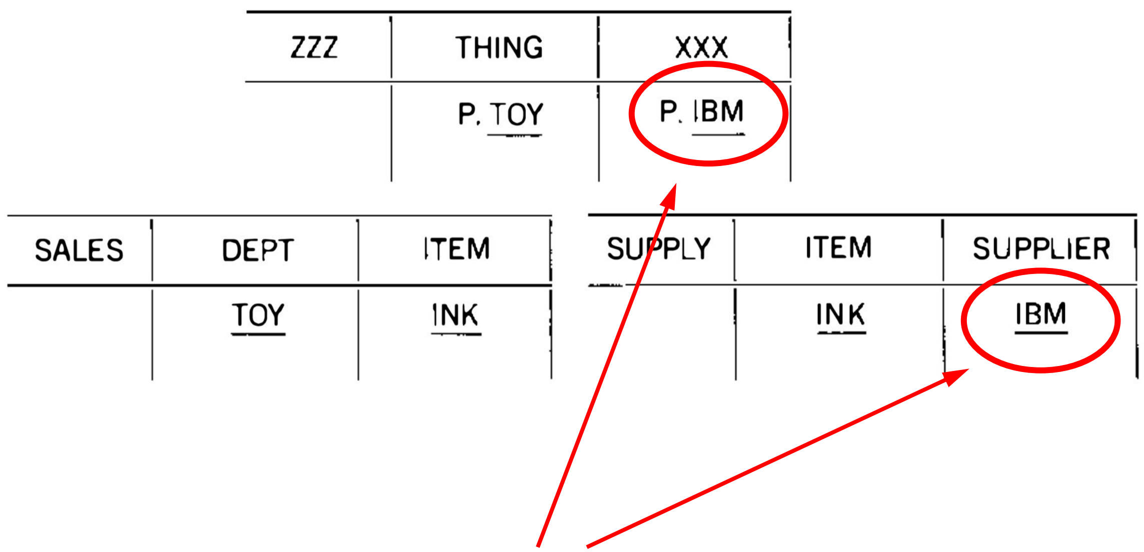
QBE (1977)
(Query-By-Example)

Zloof. Query-by-Example: A Data Base Language. IBM Systems Journal 16(4). 1977. https://doi.org/10.1147/sj.164.0324
Ramakrishnan, Gehrke. Database management systems, 2nd ed, 2000. Section 6. https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf
Elmasri, Navathe. Fundamentals of Database Systems, 7th ed, 2015. Appendix C. https://dl.acm.org/doi/10.5555/2842853
Various extensions are nicely compared in "Ozsoyoglu, Wang. Example-Based Graphical Database Query Languages. Computer, 1993. https://doi.org/10.1109/2.211893 "

QBE (Query-By-Example)

- Developed in 1970s and thus one of the first "**graphical**" query languages developed for database systems.
- Influential for later interactive query composition tools, in particular "Example-Based Graphical Database Query Languages"

Figure 19 Retrieval of collected output from multiple tables



- User specify queries using two-dimensional forms. The "examples" are actually variables, motivated from DRC (domain relational calculus).
- "Query-By-Form" would be more appropriate.

- Called by the creator Zloof as "the first visual programming language". But is QBE really "visual", i.e. is it diagrammatic?

Picture from: Zloof. Query-by-Example: A Data Base Language. IBM Systems Journal 16(4). 1977. https://doi.org/10.1147/sj.164.0324
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBE (Query-By-Example)

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```

Boat	bid	bname	color	pdate
		P	!=red	

101	Interlake	blue	4/10/13
103	Clipper	green	4/10/13

Original visualization in 1977 paper, inspired by the way relations were written out. Also used by [Gehrke Ramakrishnan'00]

Relational schema provides a template

Influenced by DRC and similar to Datalog

Boat

bid	bname	color	pdate
	P.	!=red	

More modern variant, based on UML notations. Also used by [Elmasri Navathe'15]

P(x) :- Boat(_x,y,_), y!='red'

P. = "Print" = part of output

QBE (Query-By-Example)

Q: "Find boats that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```

Boat

bid	bname	color	pdate
	P.	red	
	P.	blue	

Conditions in distinct rows are connected by "OR"

P(x) :- Boat(_,x,'red',_)
P(x) :- Boat(_,x,'blue',_)

QBE (Query-By-Example)

Q: "Find boats that are red or blue and purchased before 1980."

```
select bname
from Boat
where (color = 'red'
or color = 'blue')
and pdate < 1980
```

Boat

bid	bname	color	pdate
	P.	red	<1980
	P.	blue	<1980

Conditions in the same row are connected by "AND"

Conditions in distinct rows are connected by "OR"

P(x) :- Boat(_,x,'red',z), z<1980
P(x) :- Boat(_,x,'blue',z), z<1980

QBE (Query-By-Example)

Q: "Find boats that are red or blue and purchased before 1980."

```
select bname
from Boat
where (color = 'red'
or color = 'blue')
and pdate < 1980
```

Variables (of arbitrary name) are expressed with underscores

Boat

bid	bname	color	pdate
	P.	_c	_d

A "conditions box" allows to express more complicated conditions in a strictly linguistic way

Conditions

_d<1980 and (_c=red or _c=blue)

P(x) :- Boat(_,x,'red',z), z<1980
P(x) :- Boat(_,x,'blue',z), z<1980

QBE (Query-By-Example)

Q: "Find boats that are red or blue and purchased before 1980."

Question to the audience: which of those two query expressions is "visual", which one is not?

?

```
select bname
from Boat
where (color = 'red'
or color = 'blue')
and pdate < 1980
```

Disjunctions in Datalog are not standard but used in some Datalog implementations like Souffle (see https://souffle-lang.github.io/rules#disjunction)

Boat

bid	bname	color	pdate
	P.	_c	_d

Conditions

_d<1980 and (_c=red or _c=blue)

P(x) :- Boat(x,_,y,z), z<1980, (y=red; y=blue)

QBE (Query-By-Example)

Q: "Find boats that are red or blue and purchased before 1980."

```
select bname
from Boat
where (color = 'red'
or color = 'blue')
and pdate < 1980
```

Question to the audience:
which of those two query expressions is "visual", which one is not?

What about now?

?

Disjunctions in Datalog are not standard but used in some Datalog implementations like Souffle (see https://souffle-lang.github.io/rules#disjunction)
We now chose more "readable" variable names

Boat

bid	bname	color	pdate
	P.	_c	_d

Conditions

_d<1980 and (_c=red or _c=blue)

P(bname) :- Boat(bname,_,color,pdate), pdate<1980, (color=red; color=blue)

QBE (Query-By-Example)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Sailor

sid	sname	rating	bdate
_y	P.		

Reserves

sid	bid	day
_y	_x	

Boat

bid	bname	color	pdate
_x		red	

Q(z) :- Sailor(y,z,_,_), Reserves(y,x,_), Boat(x,_,'red',_)

QBE (Query-By-Example)

Q: "Find sailors and red boats they reserved."

```
select S.sname, B.bname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Sailor

sid	sname	rating	bdate
_y	_z		

Reserves

sid	bid	day
_y	_x	

NewOutputTable

sname	bname
P._z	P._w

Boat

bid	bname	color	pdate
_x	_w	red	

Q(z,w) :- Sailor(y,z,_,_), Reserves(y,x,_), Boat(x,w,'red',_)

QBE (Query-By-Example)

Double negations need to create an intermediate table, just like in Datalog

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

Symbol for negation

Boat

	bid	bname	color	pdate
¬	_y		red	

Reserves

sid	bid	day
_x	_y	

BadSid

	sid
I.	_x

Insert into temporary table "BadSid" sailors who reserved at least one non-red boat

Sailor

sid	sname	rating	bdate
_z	P.		

BadSid

	sid
¬	_z

RedBoat(y) :- Boat(y,_,'red',_)
BadSid(x) :- Reserves(x,y,_), not RedBoat(y)
Q(w) :- Sailor(z,w,_,_), not BadSid(z)

QBE (Query-By-Example)

Double negations need to create an intermediate table, just like in Datalog

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

Symbol for negation

Boat

	bid	bname	color	pdate
¬	_y		red	

Reserves

sid	bid	day
_x	_y	

BadSid

	sid
I.	_x

Insert into temporary table "BadSid" sailors who reserved at least one non-red boat

Sailor

sid	sname	rating	bdate
_z	P.		

BadSid

	sid
¬	_z

Datalog's safety conditions do not allow negation of (anonymous) variables that are not guarded. Thus we need an intermdiate table "RedBoat". But for anonymous variables that could be a simple syntactic extension...

RedBoat(y) :- Boat(y,_,'red',_)
BadSid(x) :- Reserves(x,y,_), not RedBoat(y)
Q(w) :- Sailor(z,w,_,_), not BadSid(z)

BadSid(x) :- Reserves(x,y,_), not Boat(y,~~_~~,'red',~~_~~)

QBE (Query-By-Example)

Double negations need to create an intermediate table, just like in Datalog

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

RedBoat(y) :- Boat(y,_,'red',_)
BadSid(x) :- Reserves(x,y,_), not RedBoat(y)
Q(w) :- Sailor(z,w,_,_), not BadSid(z)

Boat

	bid	bname	color	pdate
¬	_y		red	

Reserves

sid	bid	day
_x	_y	

BadSid

	sid
I.	_x
¬	_z

Sailor

sid	sname	rating	bdate
_z	P.		

A possible way to make it more succinct (not 100% sure if possible in current implementations)

QBE (Query-By-Example)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

Sailor

sid	sname	rating	bdate
_x			

Reserves

	sid	bid	day
¬	_x	_y	

Boat

bid	bname	color	pdate
_y		red	

BadSid

	sid
I.	_x

Sailor

sid	sname	rating	bdate
_z	P.		

BadSid

	sid
¬	_z

I(x,y) :- Reserves(x,y,_)
BadSid(x) :- Sailor(x,_,_,_), Boat(y,_,'red',_), not I(x,y)
Q(w) :- Sailor(z,w,_,_), not BadSid(z)

BadSid: sailors who have not reserved at least one red boat

QBE (Query-By-Example)

?

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Boat B
   where color = 'red'
   and not exists
      (select *
      from Reserves R
      where S.sid=R.sid
      and R.bid=B.bid))
```

Sailor

sid	sname	rating	bdate
_x			
_z	P.		

Reserves

	sid	bid	day
¬	_x	_y	

Boat

bid	bname	color	pdate
_y		red	

BadSid

	sid
I.	_x
¬	_z

Not clear if any currently available system supports the query in this more compact representation (using every table only once). But it looks even more complicated when reusing the tables (what is the order in which to read?)

I(x,y) :- Reserves(x,y,_)
BadSid(x) :- Sailor(x,_,_,_), Boat(y,_,'red',_), not I(x,y)
Q(w) :- Sailor(z,w,_,_), not BadSid(z)

QBE (Query-By-Example)

Which query is "visual" and which is not ?

1:

I(sid,bid) :- Reserves(sid,bid,_)

BadSid(sid) :- Sailor(sid,_,_,_), Boat(bid,_,'red',_),
not I(sid,bid)

Q(sname) :- Sailor(sid,sname,_,_),
not BadSid(sid)

2:

Sailor

sid	sname	rating	bdate
_x			

Reserves

	sid	bid	day
¬	_x	_y	

Boat

bid	bname	color	pdate
_y		red	

BadSid

	sid
I.	_x

Sailor

sid	sname	rating	bdate
_z	P.		

BadSid

	sid
¬	_z

Comparing various approaches from database literature

5. Visual Query Representations for Databases
   1. QBE (1977): Query-By-Example
   2. QBD (1990): Query By Diagram
   3. TableTalk (1991)
   4. OO-VQL (1993): "Object-Oriented" VQL
   5. DFQL (1994): DataFlow QL
   6. Visual SQL (2003)
   7. QueryVis (2011)
   8. Dataplay (2012)
   9. SIEUFERD (2016)
   10. SQLVis (2021)

Skipped due to time constraints

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

QBD (1990) (Query By Diagram)

Angelaccio, Catarci, Santucci. QBD*: a graphical query language with recursion. IEEE TSE 1990. https://doi.org/10.1109/32.60295
Angelaccio, Catarci, Santucci. Query by Diagram: a fully visual query system. JVLC 1990. https://doi.org/10.1016/S1045-926X(05)80009-6
Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307
Catarci, Santucci. Query by diagram : a graphical environment for querying databases. SIGMOD demo 1994. https://doi.org/10.1145/191839.191976
Catarci, Costabile, Levialdi, Batini. Visual query systems for databases: a survey. JVLC 1997. https://doi.org/10.1006/jvlc.1997.0037

QBD (Query-By-Diagram)

- Based on an ER model of the data, thus separates entities and relationships
- User navigates the ERD and creates "bridges" b/w entities when specifying the query
- Describes a mapping of the RA (relational algebra) operators to labels on edges
- Our focus is here just the visual metaphors as possibly applied to relations directly

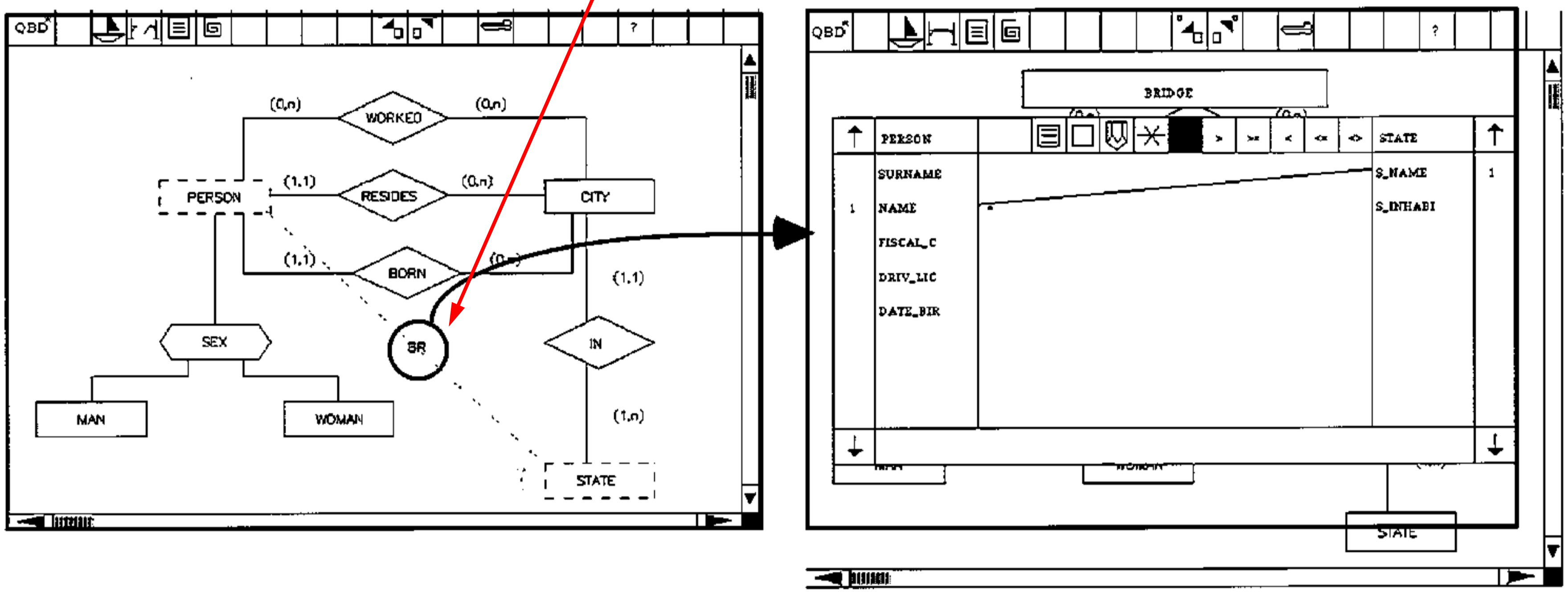


Figure source: "Catarci, Costabile, Levialdi, Batini. Visual query systems for databases: a survey. JVLC 1997. https://doi.org/10.1006/jvlc.1997.0037"
citing "Angelaccio, Catarci, Santucci. Query by Diagram*: a fully visual query system. JVLC 1990. https://doi.org/10.1016/S1045-926X(05)80009-6"

QBD (Query-By-Diagram)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Returned attributes and selections are not displayed in the main panel

Relational Algebra

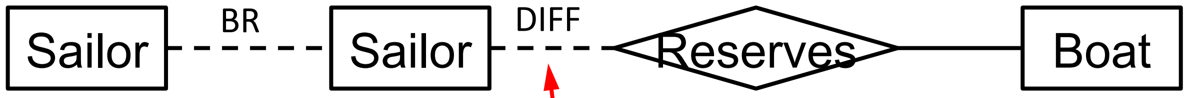
$$\pi_{\text{sname}}(S \bowtie R \bowtie \sigma_{\text{color='red'}} B)$$

QBD (Query-By-Diagram)

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid
   and not exists
      (select *
      from Boat B
      where R.bid=B.bid
      and color = 'red'))
```

A "bridge" connects entities that are not connected in the original ERD



Set difference as algebraic operator is shown by a labeled edge

Relational Algebra

$$\pi_{sname}(S \bowtie (\pi_{sid} S - (\pi_{sid}(R \bowtie \sigma_{color='red'} B))))$$

Figure drawn based on best understanding of "Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307"
and "Angelaccio, Catarci, Santucci. QBD*: a graphical query language with recursion. IEEE TSE 1990. https://doi.org/10.1109/32.60295 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

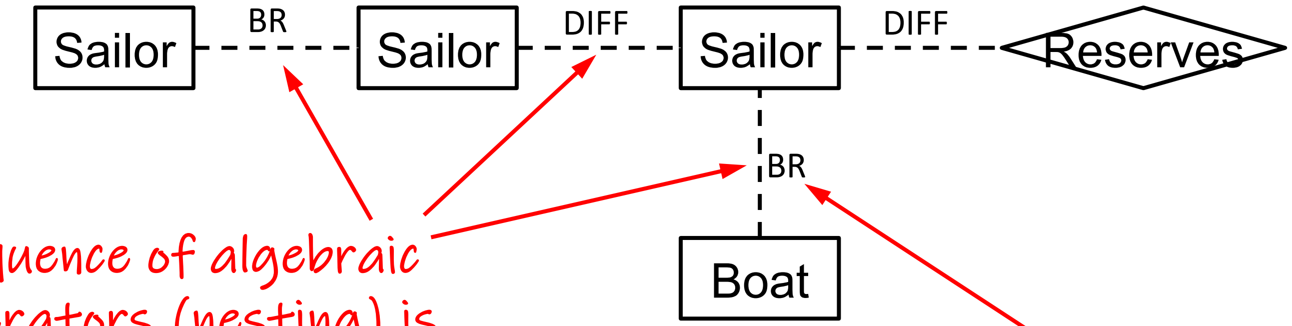
QBD (Query-By-Diagram)

?

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

QBD is originally an ER model, so "Reserves" would be a relationship. For RA it needs to be treated as table



Sequence of algebraic operators (nesting) is ambiguous

The Cross Product is treated as join thus connects two entities via a "bridge"

Relational Algebra

$$\pi_{\text{sname}}(S \bowtie (\pi_{\text{sid}} S - \pi_{\text{sid}}((\pi_{\text{sid}} S \times \pi_{\text{bid}} \sigma_{\text{color='red'}} B) - \pi_{\text{sid,bid}} R)))$$

Figure drawn based on best understanding of "Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307"
and "Angelaccio, Catarci, Santucci. QBD*: a graphical query language with recursion. IEEE TSE 1990. https://doi.org/10.1109/32.60295 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

TableTalk (1991)

Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6

Tabletalk

- A flowchart inspired visual representation based on blocks

The program is "run" top-down, and displays results of the query at the bottom

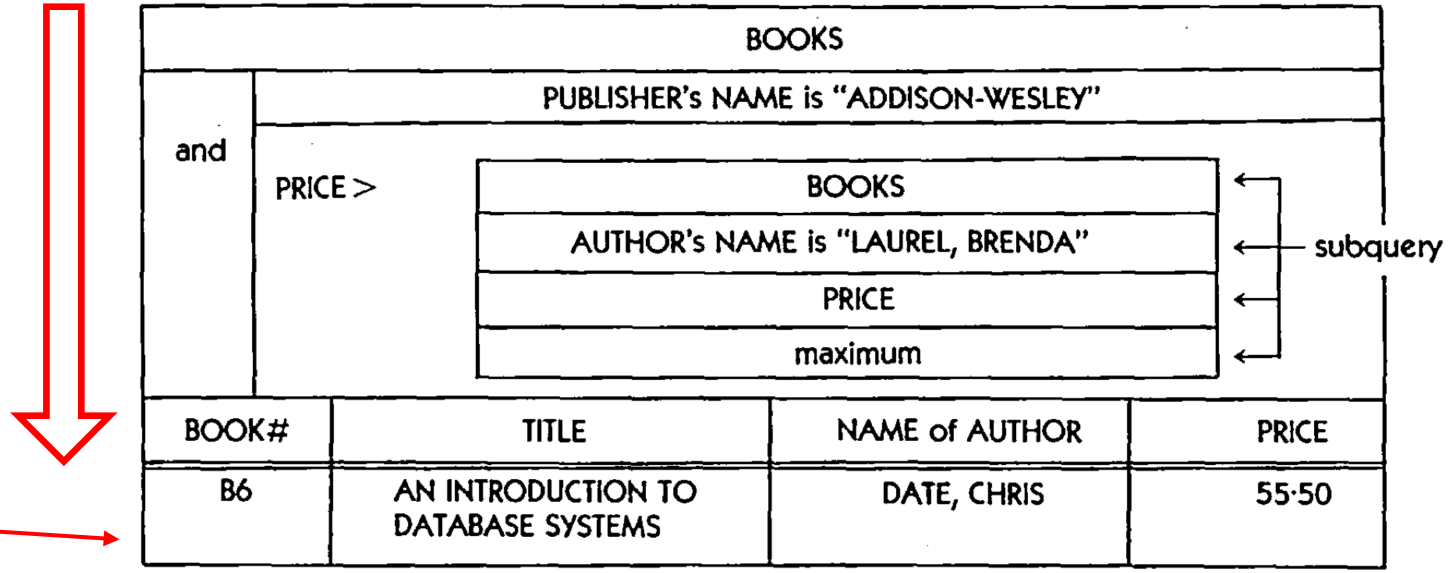


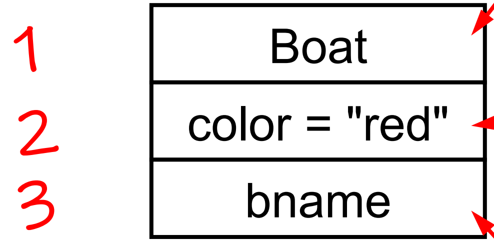
Figure source: "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

Tabletalk

Q: "Find boats that are not red."

```
select bname
from Boat
where color = 'red'
```

row #:



class row (= table containing objects of interest)

predicate row: specifies a logical condition

scalar transformer

sequence processing language with rows and tiles (is read top down)

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

Tabletalk

Q: "Find boats that are red or blue."

```
select bname, color
from Boat
where color = 'red'
or color = 'blue'
```

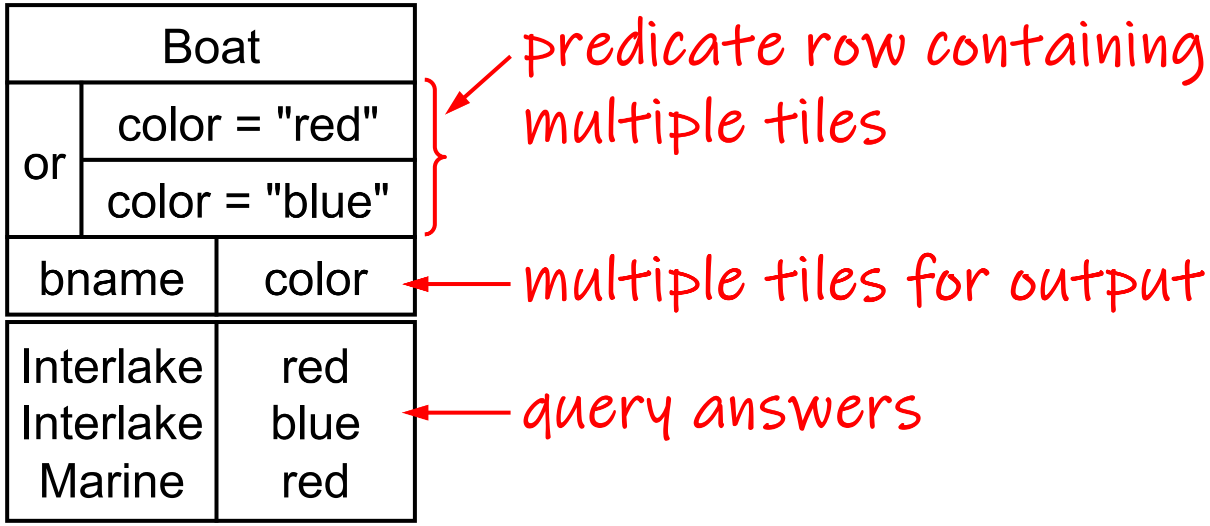
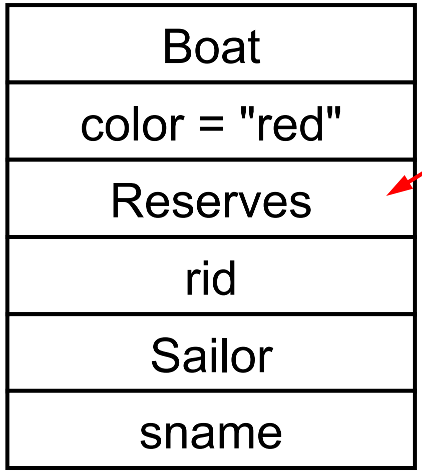


Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

Tabletalk

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



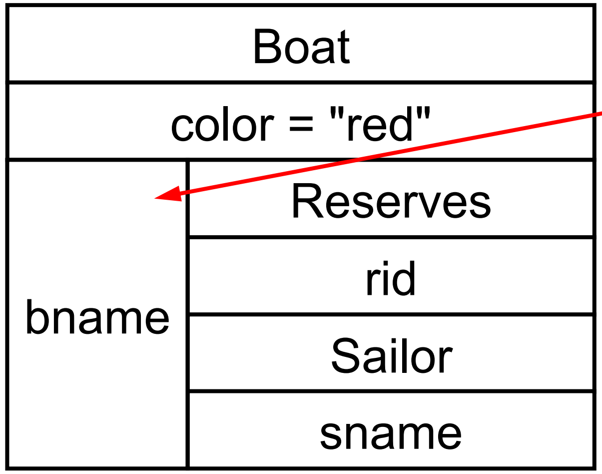
"object transformer"
(assumes FK-PK constraints)
"transforms the primordial objects of the main processing sequence into new primordial objects whose class is the codomain class of the object attribute."

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Tabletalk

Q: "Find sailors who reserved a red boat."

```
select S.sname, B.bname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



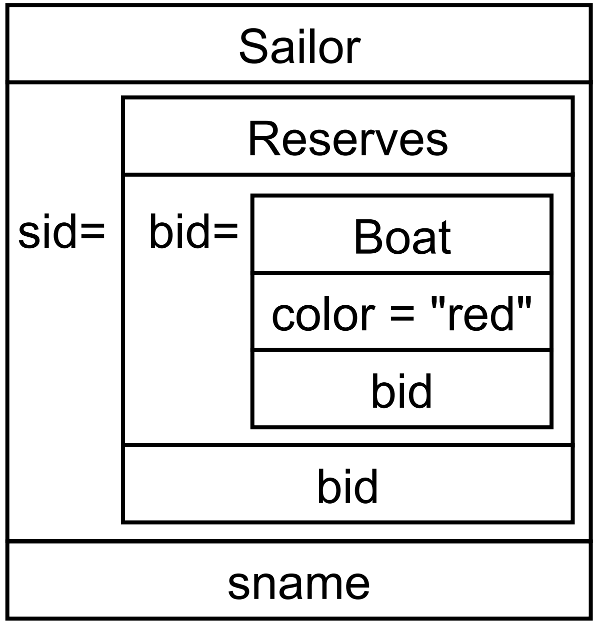
apply "product" to not "forget"

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Tabletalk

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
where exists
    (select *
    from Boat B
    where color = 'red'
    and exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```



Nested queries are evaluated inside out

Tabletalk

?

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

To express nested correlated queries, we first have to unnest and then follow a dataflow strategy.

This requires a cross-product and difference, similar to Datalog and QBE.

But paper does not discuss a difference or other non-monotone operator...

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

"OO-VQL" (1993)
"Object-Oriented VQL"

Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

OO-VQL (Object-oriented VQL)

- Developed for an object-oriented data model, thus separates entities and relationships
- We again focus here on the visual metaphors as possibly applied to relations directly

- Notice that "VQL" is ambiguous: The term has been used multiple times in the literature for proposed visual languages, even for the term Visual Query Language. So we use here OO-VQL for "Object-Oriented VQL"

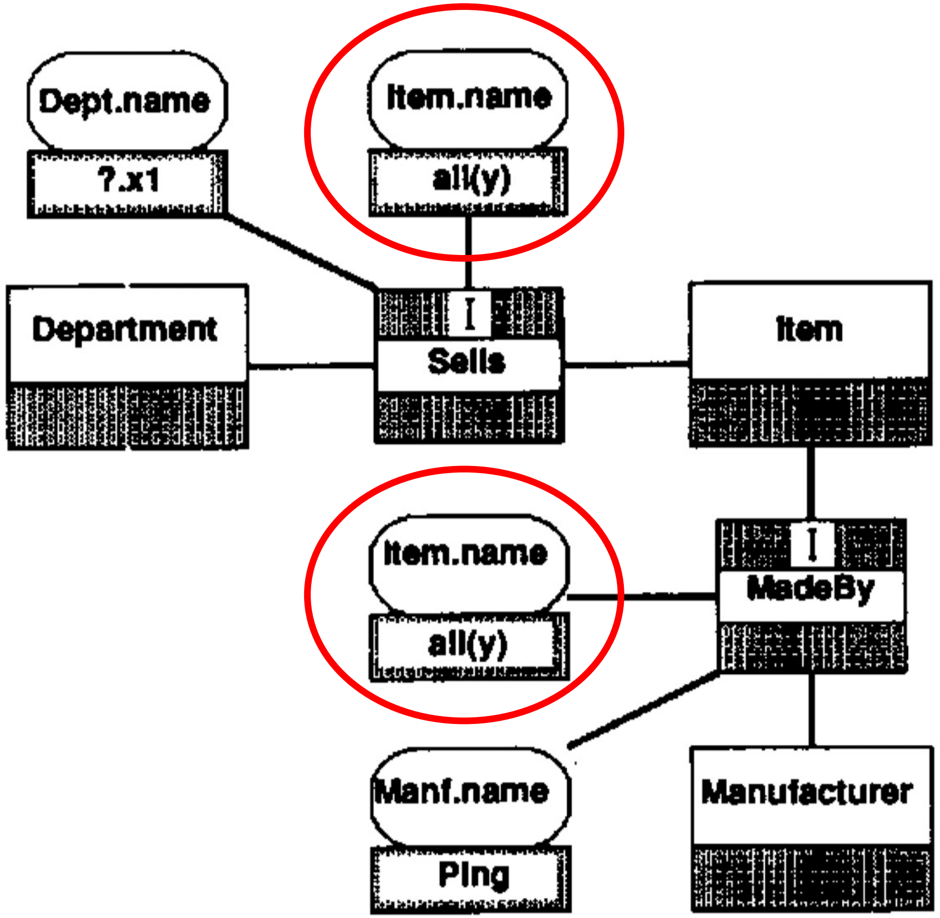
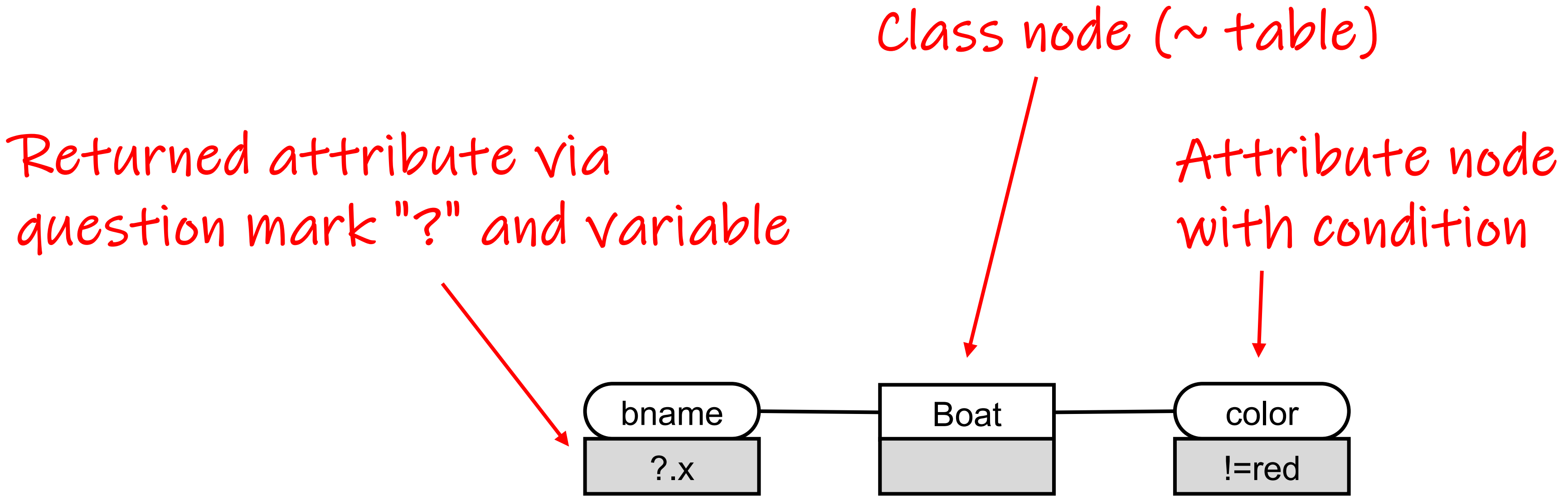


Figure source: Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

OO-VQL (Object-oriented VQL)

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```



DRC (Domain Relational Calculus)

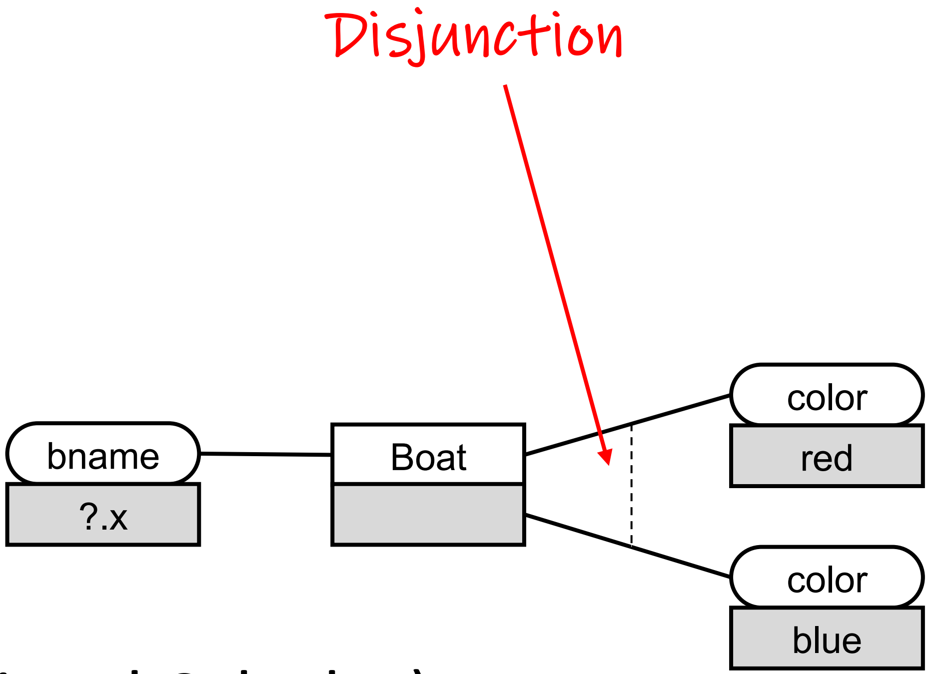
{(x) | Sailor(_,x,y,_) ∧ (y!='red')}

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q: "Find boats that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```



DRC (Domain Relational Calculus)

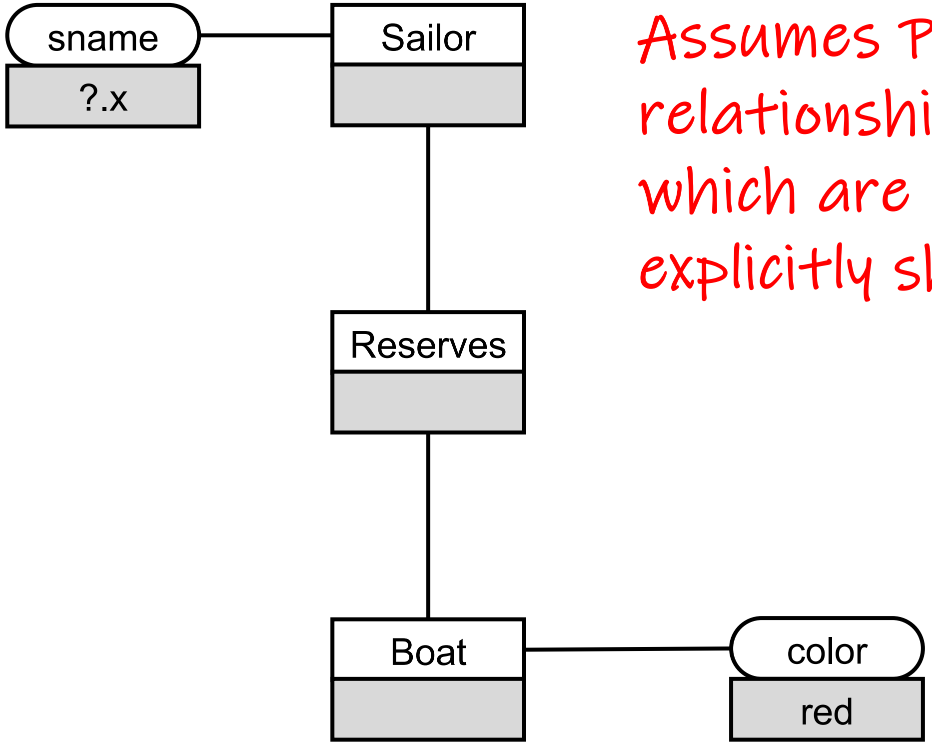
{(x) | Sailor(_,x,y,_) ∧
(y='red' ∨ y='red')}

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



Assumes PK-FK relationships, which are not explicitly shown

DRC (Domain Relational Calculus)

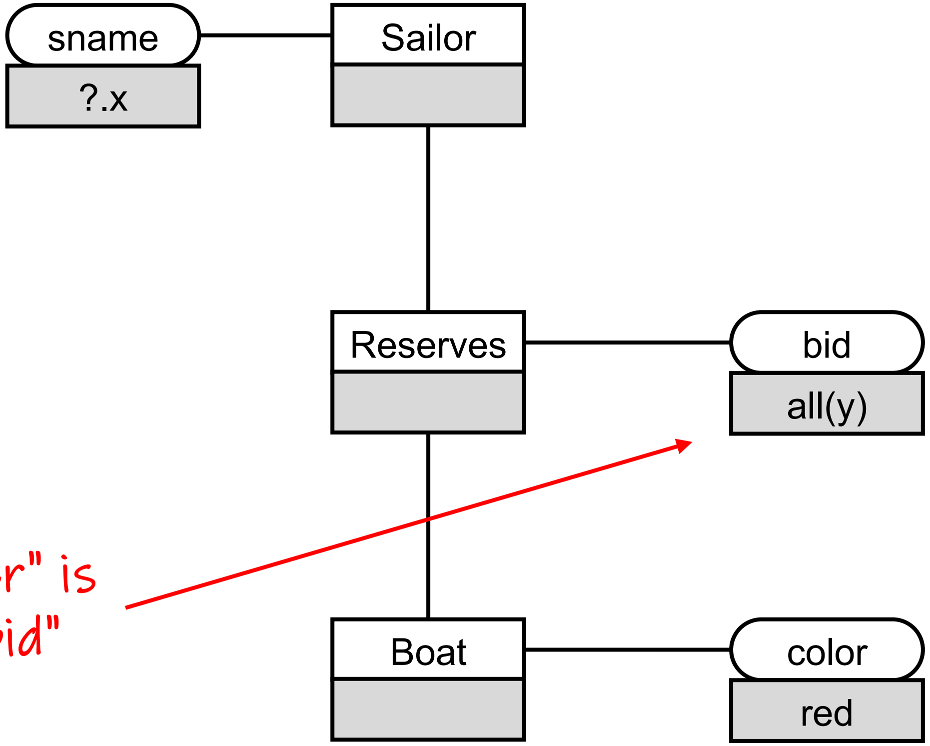
$$\{(x) \mid \exists v[\text{Sailor}(v,x,\_,\_) \wedge \exists y[\text{Reserves}(v,y,\_) \wedge \text{Boat}(y,\_,\text{'red'},\_)]]$$

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```



A "universal quantifier" is added to attribute "bid"

DRC (Domain Relational Calculus)

$$\{(x) \mid \exists v[\text{Sailor}(v,x,\_,\_) \wedge (\forall y[\text{Reserves}(v,y,\_) \rightarrow (\text{Boat}(y,\_,\text{'red'},\_))])]\}$$

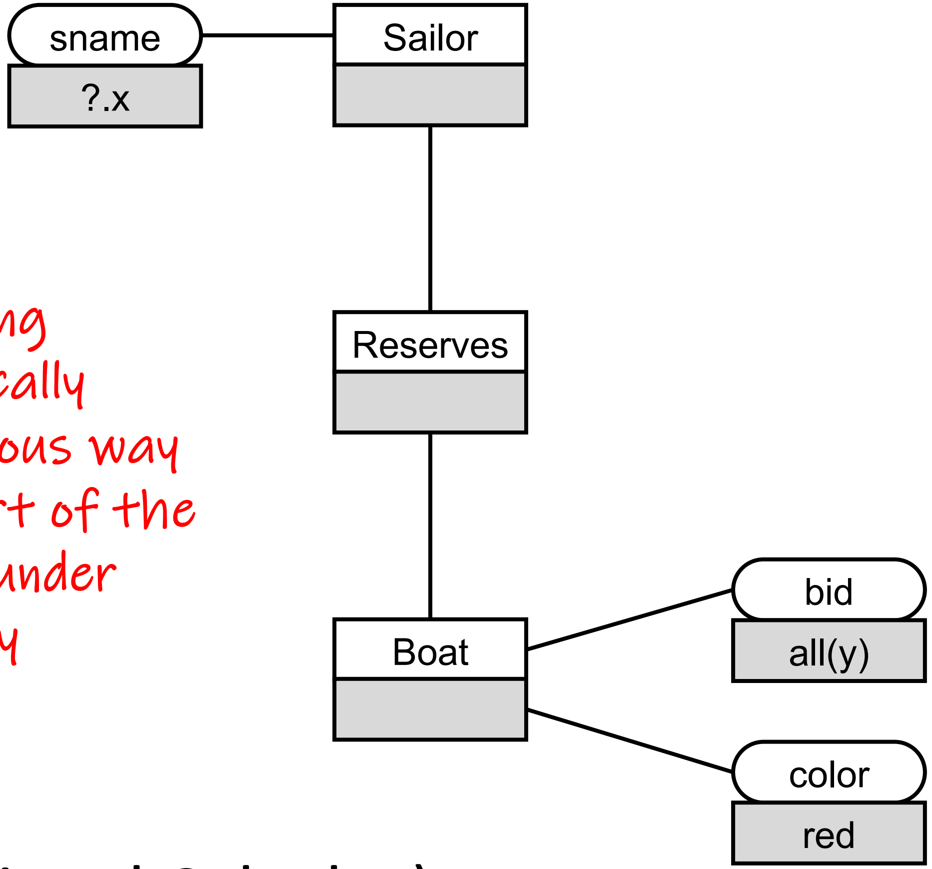
OO-VQL (Object-oriented VQL)

?

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

Visualization can become ambiguous because scoping cannot always be graphically represented in unambiguous way (e.g. is Reserves still part of the universal quantification under Boat, or is it existentially quantified under Boat?)

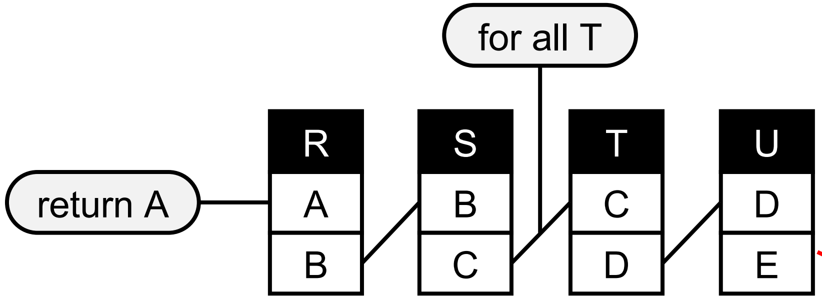


DRC (Domain Relational Calculus)

$$\{(x) \mid \exists v[\text{Sailor}(v,x,\_,\_) \wedge (\forall y[\text{Boat}(y,\_,\text{'red'},\_) \rightarrow (\text{Reserves}(v,y,\_))])]\}$$

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

When the quantifier scope becomes ambiguous



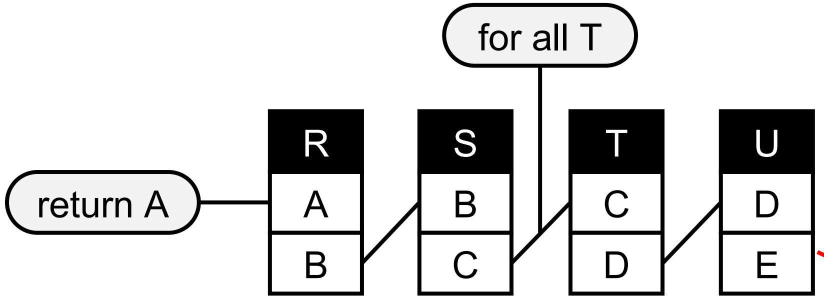
This figure allows
2 interpretations

interpretation 2

interpretation 1

?

When the quantifier scope becomes ambiguous



This figure allows
2 interpretations

interpretation 1

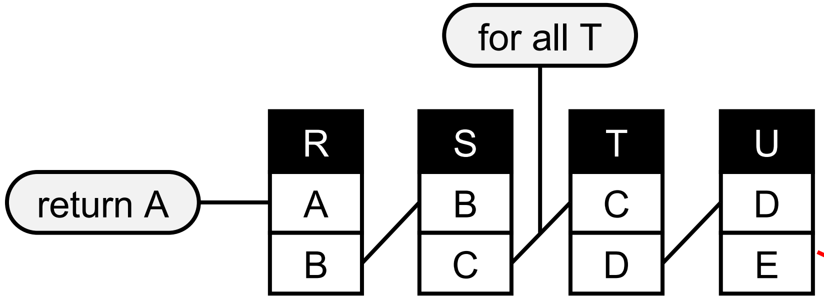
$\{(a) \mid \exists b[R(a,b)$
$(\forall c[S(b,c) \rightarrow$
$(\exists d,e[T(c,d) \wedge U(d,e)])])]\}$

interpretation 2

$\{(a) \mid \exists b,d,e[R(a,b) \wedge U(d,e)$
$(\forall c[T(c,d) \rightarrow$
$(S(b,c))])]\}$

?

When the quantifier scope becomes ambiguous



This figure allows
2 interpretations

interpretation 1

interpretation 2

```
select distinct R.A
from R
where not exists
    (select *
    from S
    where R.B=S.B
    and not exists
        (select *
        from T,U
        where S.C=T.C
        and T.D=U.D))
```

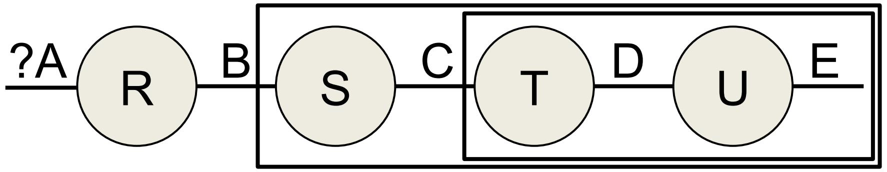
$$\{(a) \mid \exists b[R(a,b)\ (\forall c[S(b,c) \rightarrow (\exists d,e[T(c,d) \land U(d,e)])])]\}$$

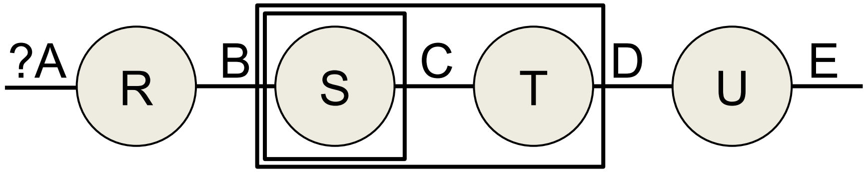
$$\{q(A) \mid \exists r \in R[q.A=r.A \land \neg(\exists s \in S[r.B=s.B \land \neg(\exists t \in T, \exists u \in U[s.C=t.C \land t.D=u.D])])]\}$$

```
select distinct R.A
from R, U
where not exists
    (select *
    from T
    where T.D=U.D
    and not exists
        (select *
        from S
        where R.B=S.B
        and S.C=T.C))
```

$$\{(a) \mid \exists b,d,e[R(a,b) \land U(d,e)\ (\forall c[T(c,d) \rightarrow (S(b,c))])]\}$$

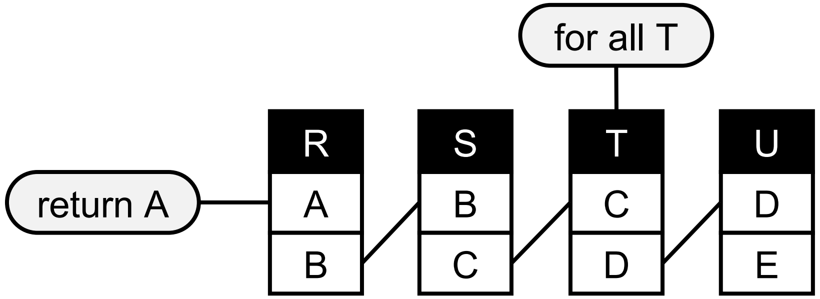
$$\{q(A) \mid \exists r \in R, u \in U[q.A=r.A\ \neg(\exists t \in T[t.D=u.D \land \neg(\exists s \in S[r.B=s.B \land t.D=u.D])])]\}$$





See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

When the quantifier scope becomes ambiguous



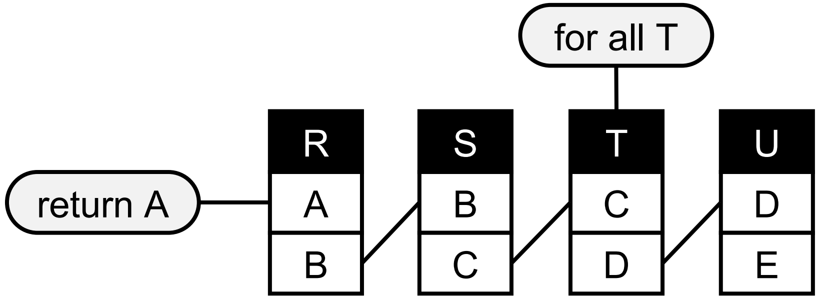
This figure also allows 2 interpretations

interpretation 3

interpretation 2

?

When the quantifier scope becomes ambiguous



This figure also allows 2 interpretations

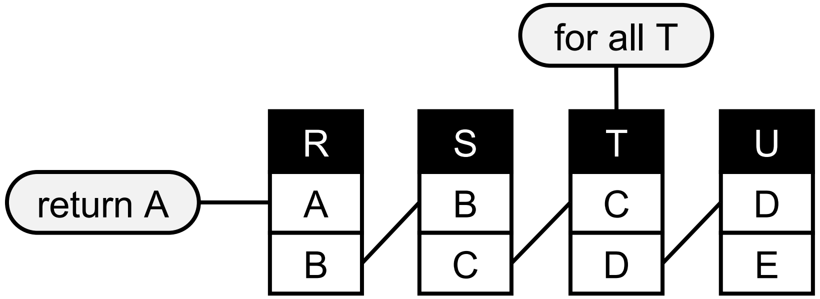
interpretation 3

interpretation 2

$$\{q(A) \mid \exists r \in R, s \in S[q.A=r.A \land r.B=s.B \land \neg(\exists t \in T[s.C=t.C \land \neg(\exists u \in U[t.D=u.D])])]\}$$

$$\{q(A) \mid \exists r \in R, u \in U[q.A=r.A \; \neg(\exists t \in T[t.D=u.D \land \neg(\exists s \in S[r.B=s.B \land t.D=u.D])])]\}$$

When the quantifier scope becomes ambiguous



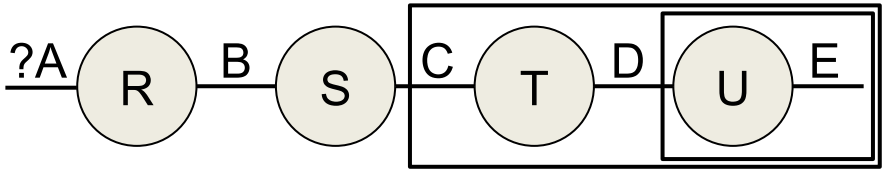
This figure also allows 2 interpretations

interpretation 3

```
select distinct R.A
from R, S
where R.B=S.B
and not exists
    (select *
    from T
    where S.C=T.C
    and not exists
        (select *
        from U
        where T.D=U.D))
```

$$\{(a) \mid \exists b,c[R(a,b) \land S(b,c) \land (\forall d[T(c,d) \rightarrow (\exists e[U(d,e)])])]\}$$

$$\{q(A) \mid \exists r\in R, s\in S[q.A=r.A \land r.B=s.B \land \neg(\exists t\in T[s.C=t.C \land \neg(\exists u\in U[t.D=u.D])])]\}$$

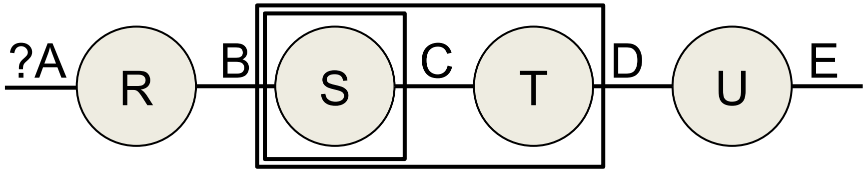


interpretation 2

```
select distinct R.A
from R, U
where not exists
    (select *
    from T
    where T.D=U.D
    and not exists
        (select *
        from S
        where R.B=S.B
        and S.C=T.C))
```

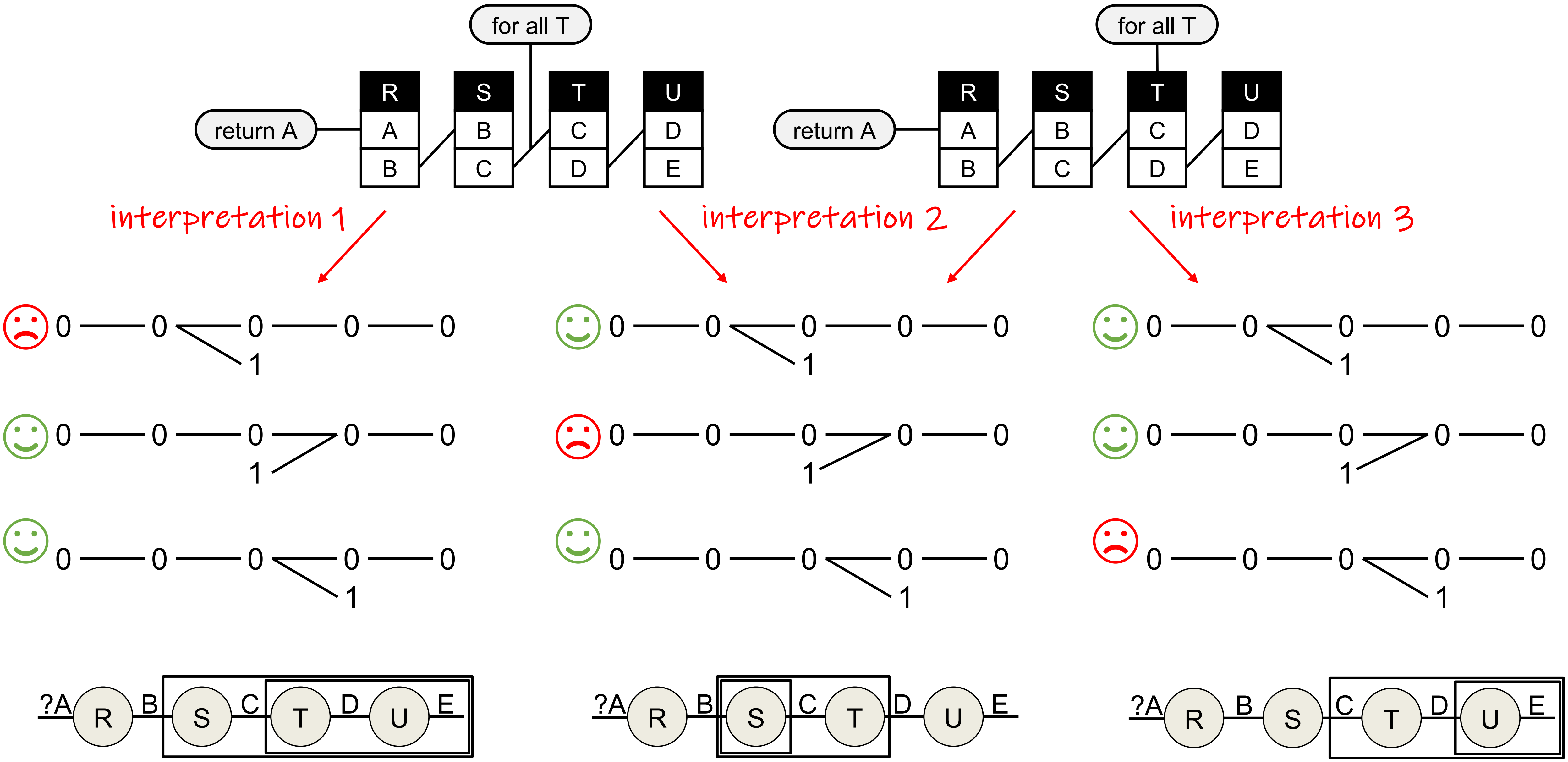
$$\{(a) \mid \exists b,d,e[R(a,b) \land U(d,e) \; (\forall c[T(c,d) \rightarrow (S(b,c))])]\}$$

$$\{q(A) \mid \exists r\in R, u\in U[q.A=r.A \; \neg(\exists t\in T[t.D=u.D \land \neg(\exists s\in S[r.B=s.B \land t.D=u.D])])]\}$$



See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

When the quantifier scope becomes ambiguous



See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Comparing various approaches from database literature

5. Visual Query Representations for Databases
   1. QBE (1977): Query-By-Example
   2. QBD (1990): Query By Diagram
   3. TableTalk (1991)
   4. OO-VQL (1993): "Object-Oriented" VQL
   5. DFQL (1994): DataFlow QL
   6. Visual SQL (2003)
   7. QueryVis (2011)
   8. Dataplay (2012)
   9. SIEUFERD (2016)
   10. SQLVis (2021)

DFQL (1994)
DataFlow QL

Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1
Girsang. The comparison of SQL, QBE, and DFQL as query languages for relational databases, Master thesis, Naval Postgraduate School, 1994. https://core.ac.uk/download/pdf/36723678.pdf

DFQL (DataFlow Query Language)

- Example visual representation that is relationally complete by <u>mapping its visual symbols to the operators of RA</u>
- Actually many Visual Query Languages (VQLs) become "visual" by introducing visual representations of RA operators in a dataflow

- Here, <u>nodes represent the operations</u>, instead of edges as in QBD
- We chose DFQL as representative since it has a nice documentation

We will ignore the "Display" operator that sends output to the screen

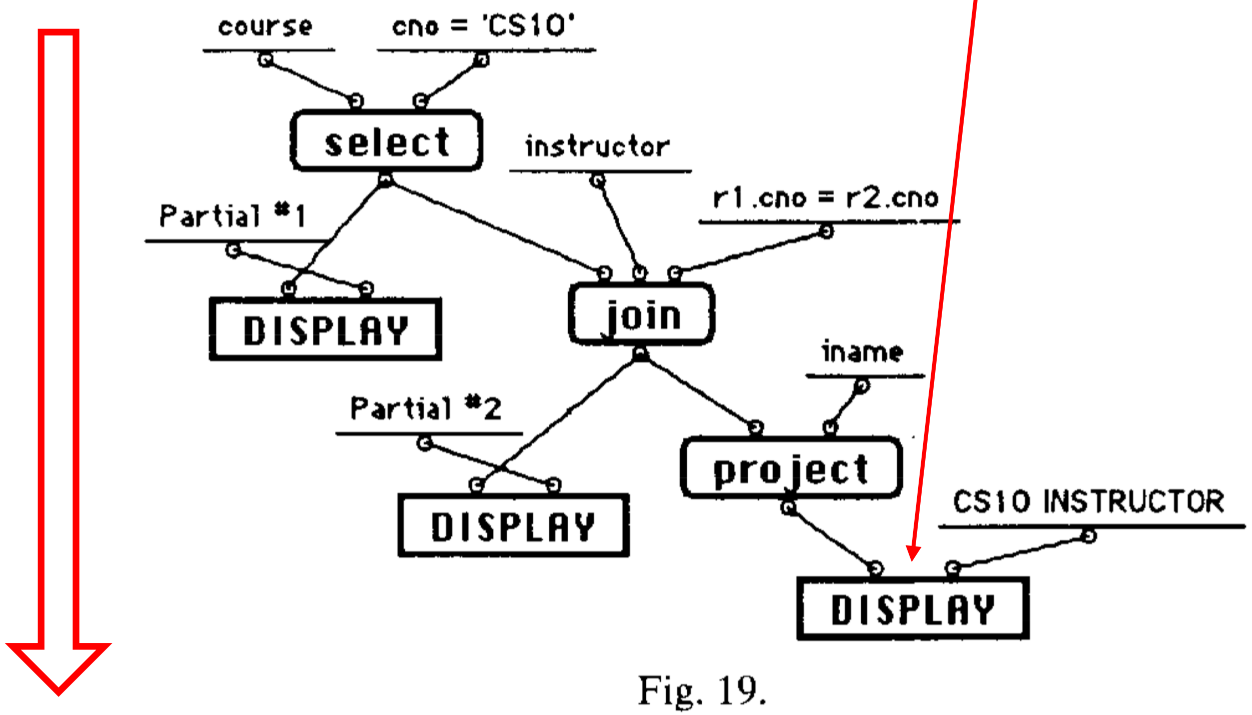


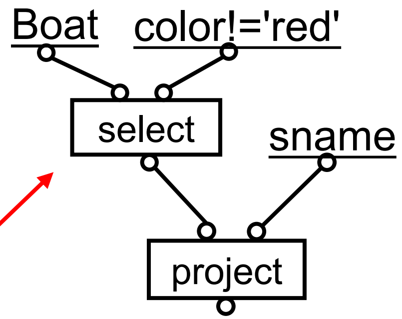
Fig. 19.

Figure source: "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "

DFQL (DataFlow Query Language)

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```



boxes model the operators
of RA (relational Algebra)
in a top down "dataflow"

Relational Algebra

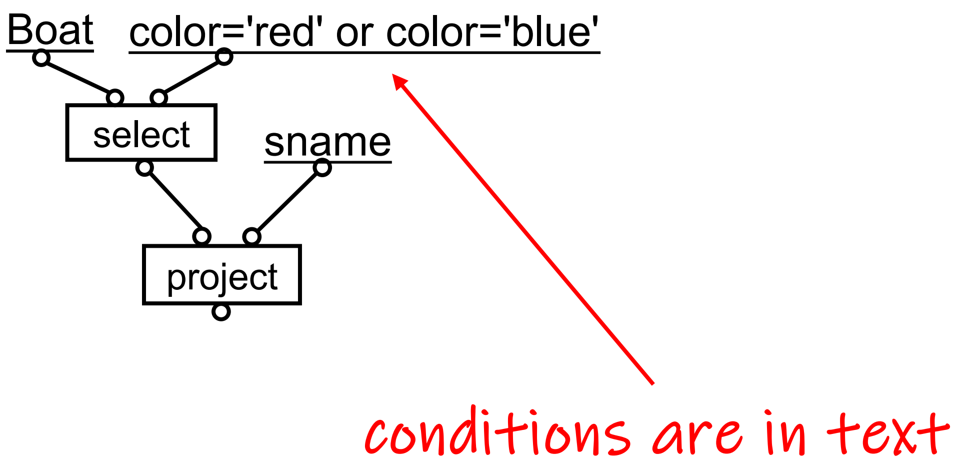
$$\pi_{\text{sname}}(\sigma_{\text{color=!'red'}}\text{B})$$

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

Q: "Find boats that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```



Relational Algebra

$$\pi_{\text{sname}}(\sigma_{\text{color='red'} \vee \text{z='blue}}B)$$

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

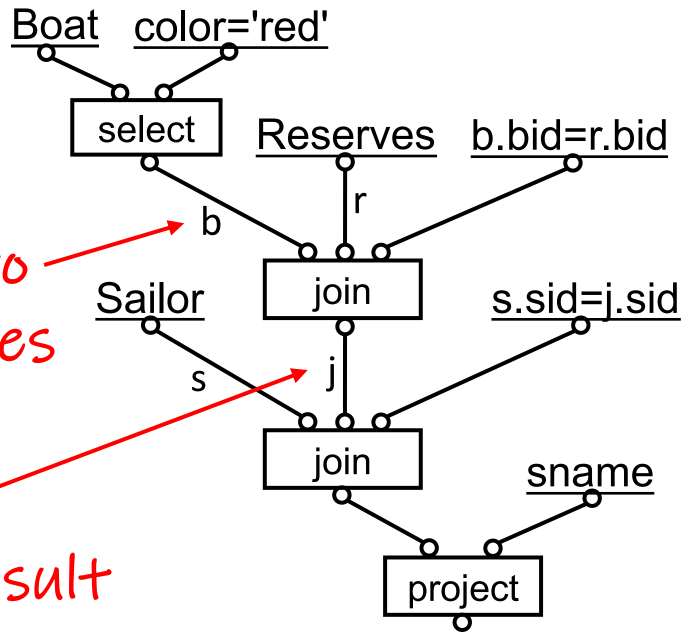
DFQL (DataFlow Query Language)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

alias on edges to avoid ambiguities

new alias for intermediate result



Relational Algebra

$$\pi_{\text{sname}}(S \bowtie R \bowtie \sigma_{\text{color='red'}} B)$$

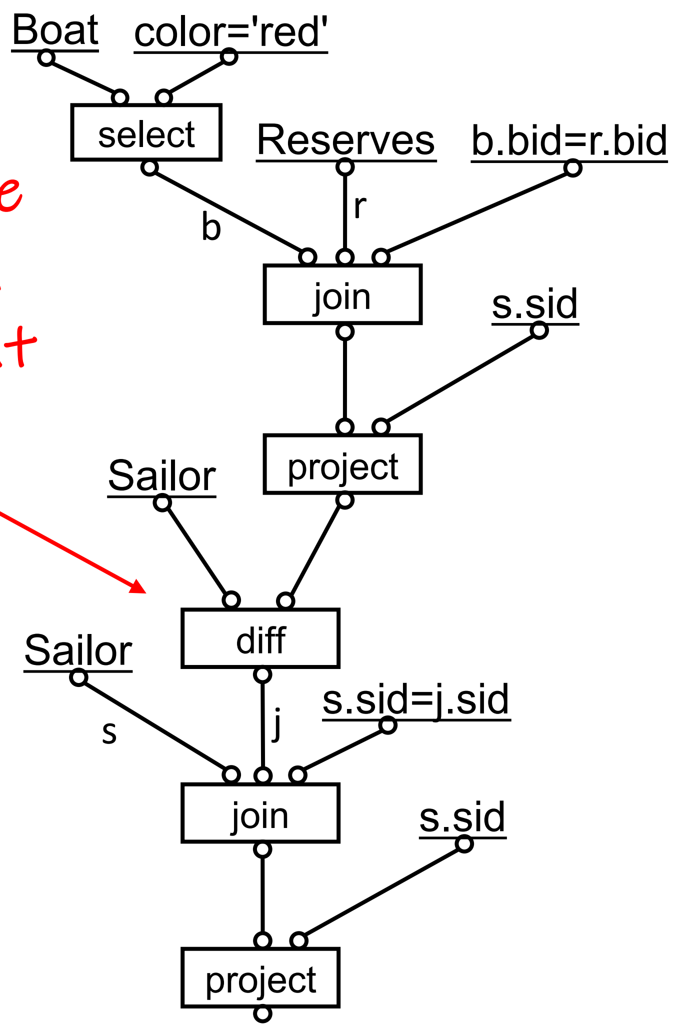
Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

The diff operator is same as binary minus (-) in RA. Notice that order of input matters!



Relational Algebra

$$\pi_{sname}(S \bowtie (\pi_{sid} S - (\pi_{sid}(R \bowtie \sigma_{color='red'} B))))$$

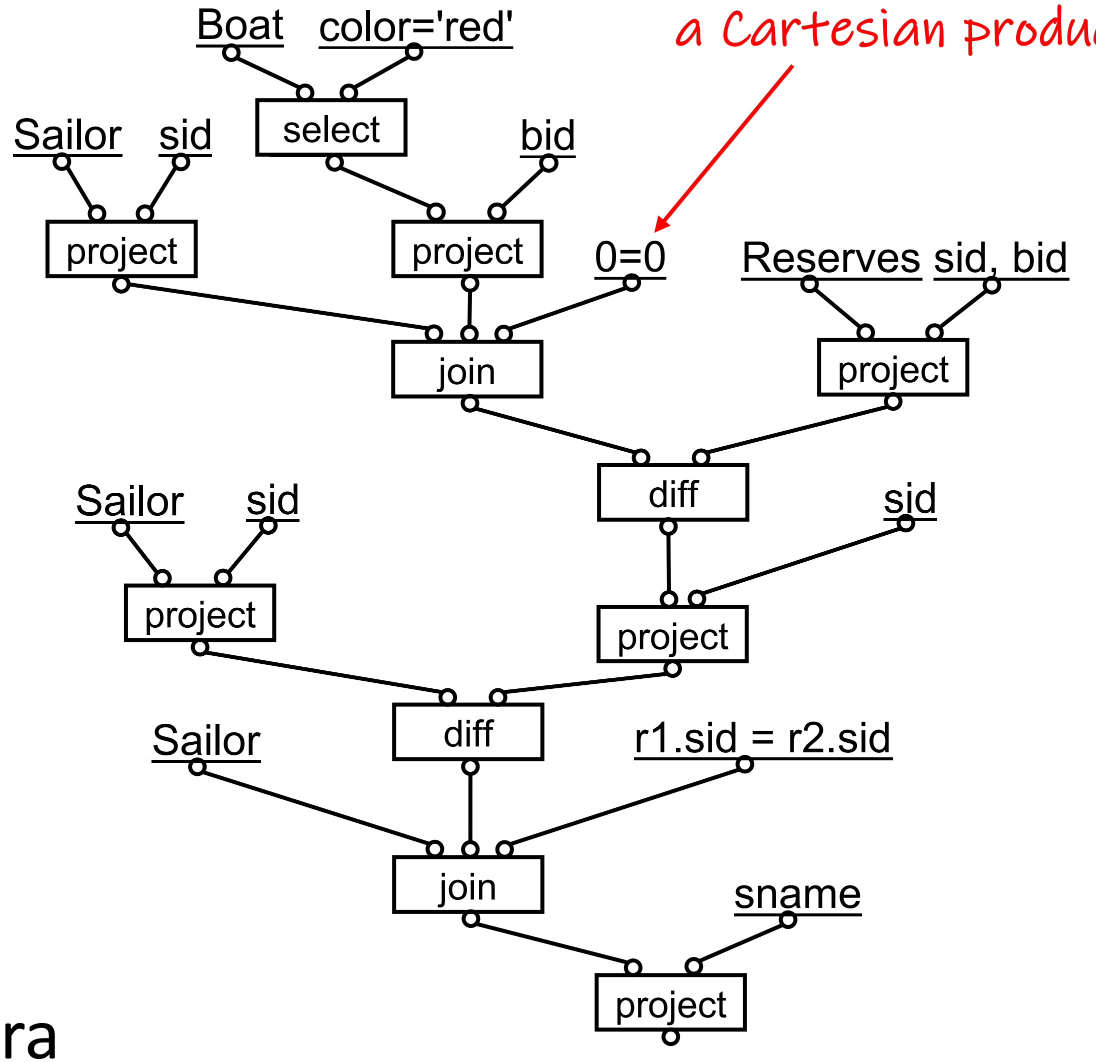
Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

The tautology "0 = 0" in join operator is needed for a Cartesian product



Relational Algebra

$$\pi_{\text{sname}}(S \bowtie (\pi_{\text{sid}} S - \pi_{\text{sid}}((\pi_{\text{sid}} S \times \pi_{\text{bid}} \sigma_{\text{color='red'}} B) - \pi_{\text{sid,bid}} R)))$$

Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

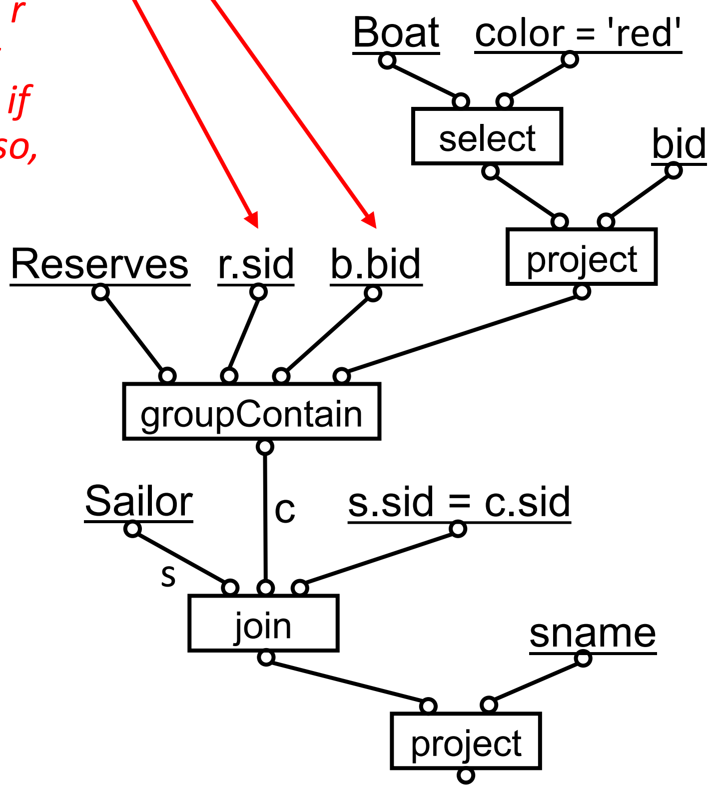
Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where
    (select R.bid
    from Reserves R
    where S.sid=R.sid)
    CONTAINS
    (select B.bid
    from Boat B
    where color = 'red')
```

The "groupContain" operators takes the RESERVES relation r and the second relation b (with red boats) and groups the tuples in r according to the grouping attribute r.sid. It then compares with attributes b.bid to see if one r.sid has all the b.bid values from b. If so, the sid is selected

order left/right is crucial here

Invented SQL operator similar to relational division (but does not match exactly)



Relational Algebra

$$\pi_{sname}(S \bowtie (\pi_{sid,rid} R \div \pi_{bid} \sigma_{color='red'} B)$$

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

Visual SQL (2003)

Jaakkola, Thalheim. Visual SQL -- high-quality ER-based query treatment. ER workshops, 2003. https://doi.org/10.1007/978-3-540-39597-3_13
Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, Cottbus, 2003.
Thalheim. Visual SQL as the alternative to Linear SQL. Talk slides. 2013. Originally available online at:
https://www.is.informatik.uni-kiel.de/fileadmin/arbeitsgruppen/is_engineering/visualsql/HERM.VisualSQL.Talk2013.pdf (4/2020)

Visual SQL

- The goal of Visual SQL was primarily query specification (but an implementation also supports to some extent the reverse functionality)
- The tool focus on representing syntactic details and some details are not shown visually
- Notice that the notation uses a more visually familiar UML notation

- Notice that the original website has since disappeared and other unrelated tools can now be found under the name "VisualSQL"

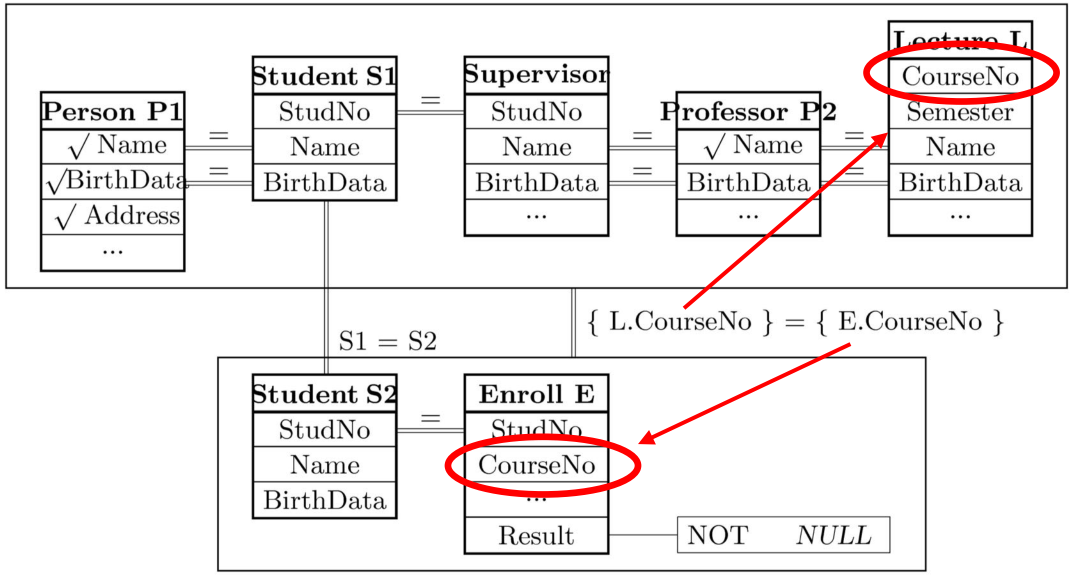
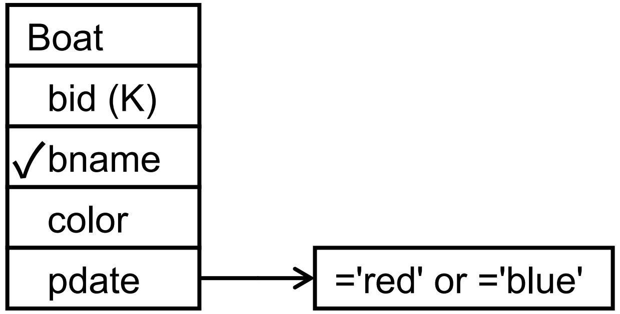


Fig. 1. Visual SQL Involving Equality On Two Visual SQL Subqueries

Visual SQL

Q: "Find boats
that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```

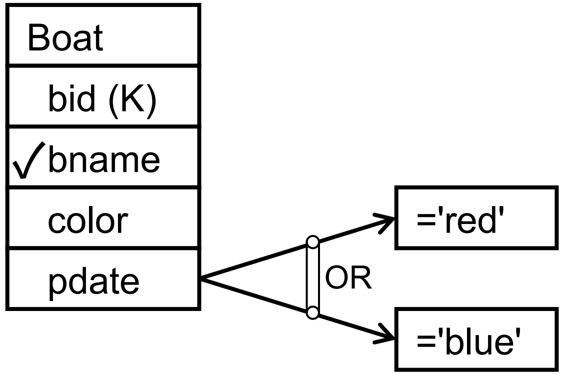


Disjunctions are explicitly expressed inside a box

Visual SQL

Q: "Find boats that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```



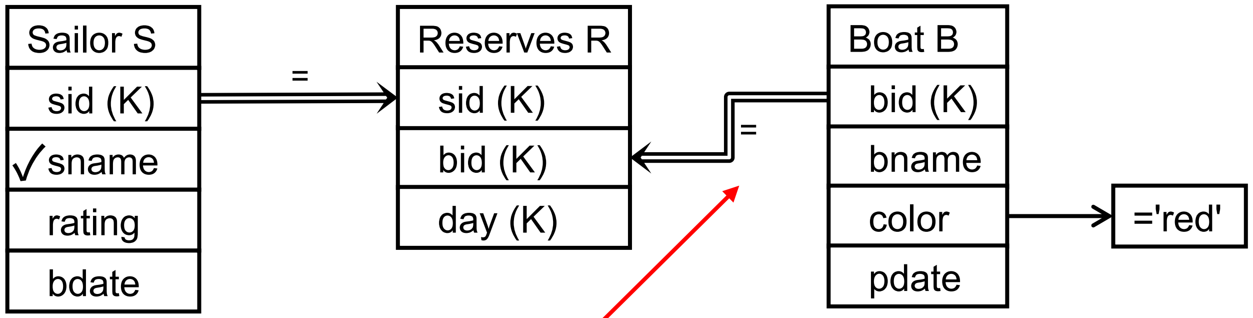
An alternative suggestion was to connect two alternative lines with an "OR" labeled connector

Figure drawn based on Figure 14 in "Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003."

Visual SQL

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



Joins use double-lined arcs. Direction is determined by the predicate

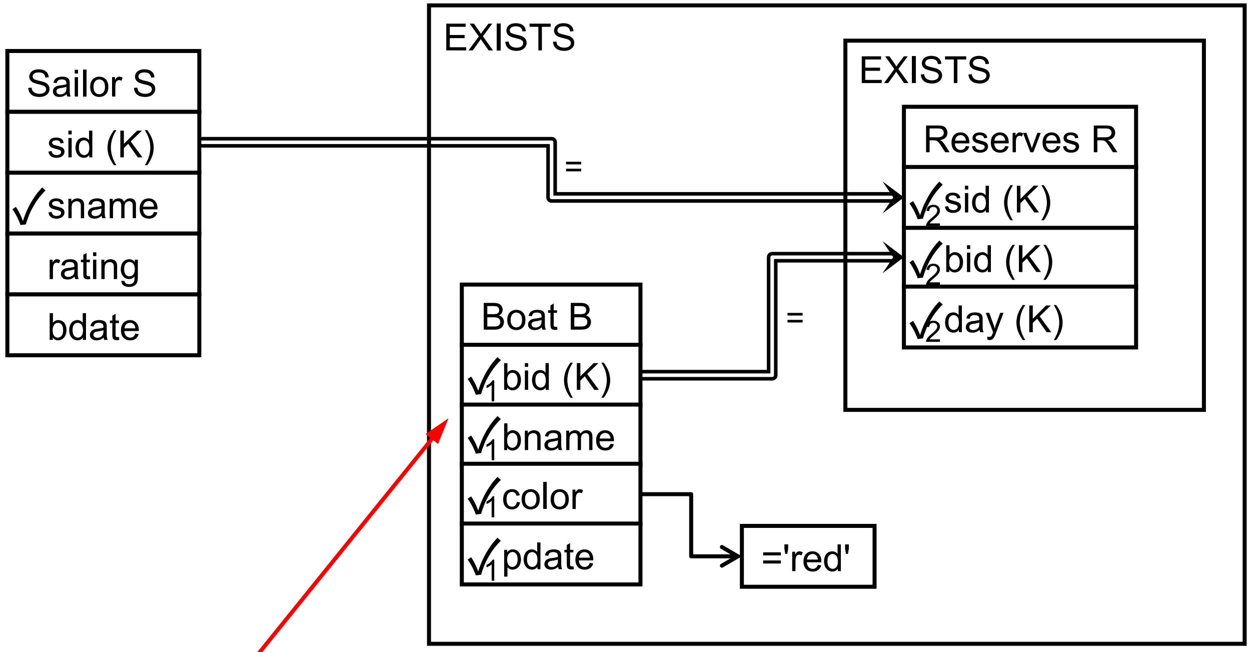
... B.bid=R.bid ...

Visual SQL

Nested queries preserve the scope of nestings

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
where exists
    (select *
    from Boat B
    where color = 'red'
    and exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```



Check-marks are labeled with indices likely based on when the subquery appears in the SQL text.
SELECT * leads to all attributes being checked.

Figure drawn based on a result by Jiahui Zhang visualized with a code available on the website of University of Kiel around 4/2020
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

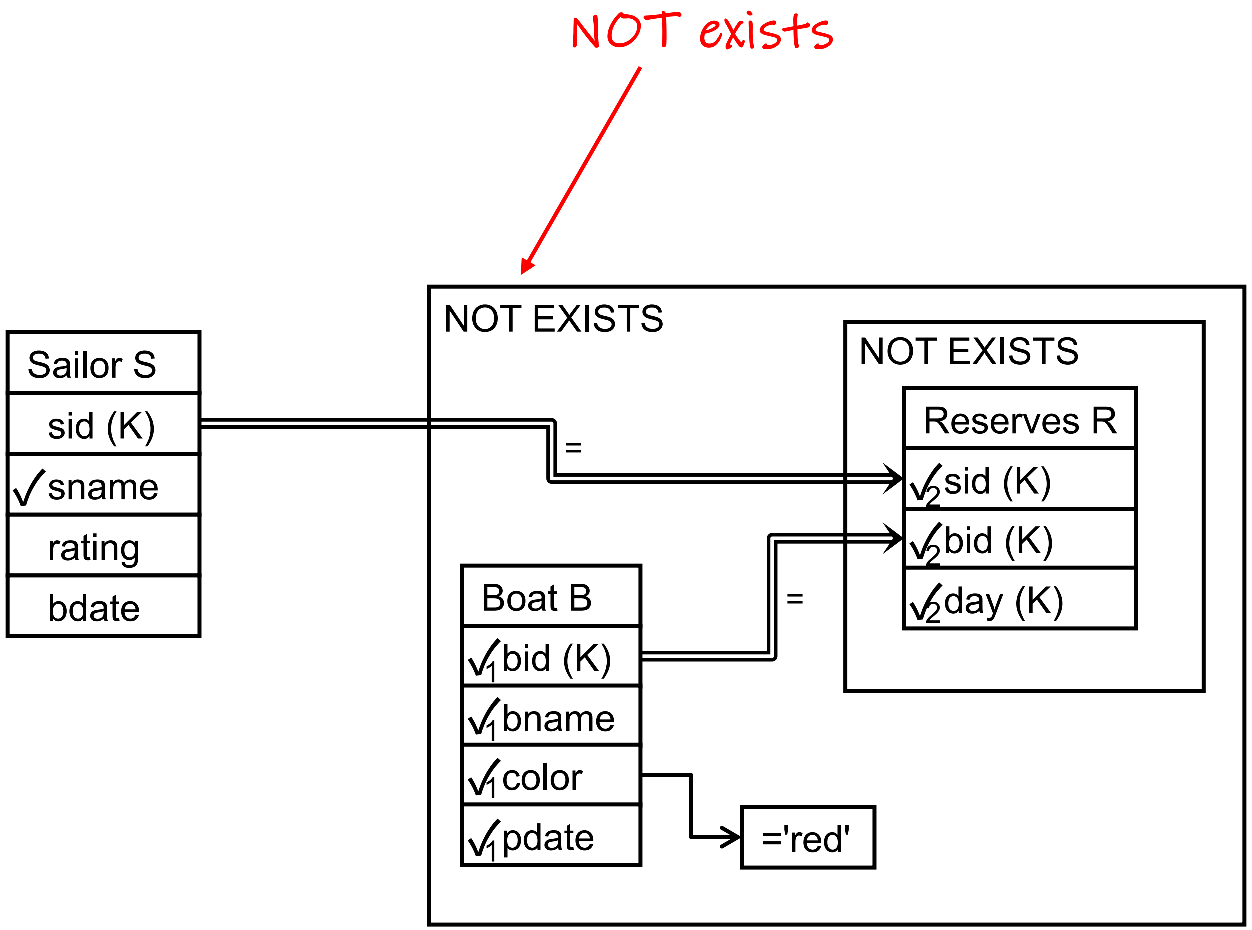


Figure drawn based on a result by Jiahui Zhang visualized with a code available on the website of University of Kiel around 4/2020

Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

An alternative proposal that visually represents "NOT EXISTS" as connection between a table and the subquery

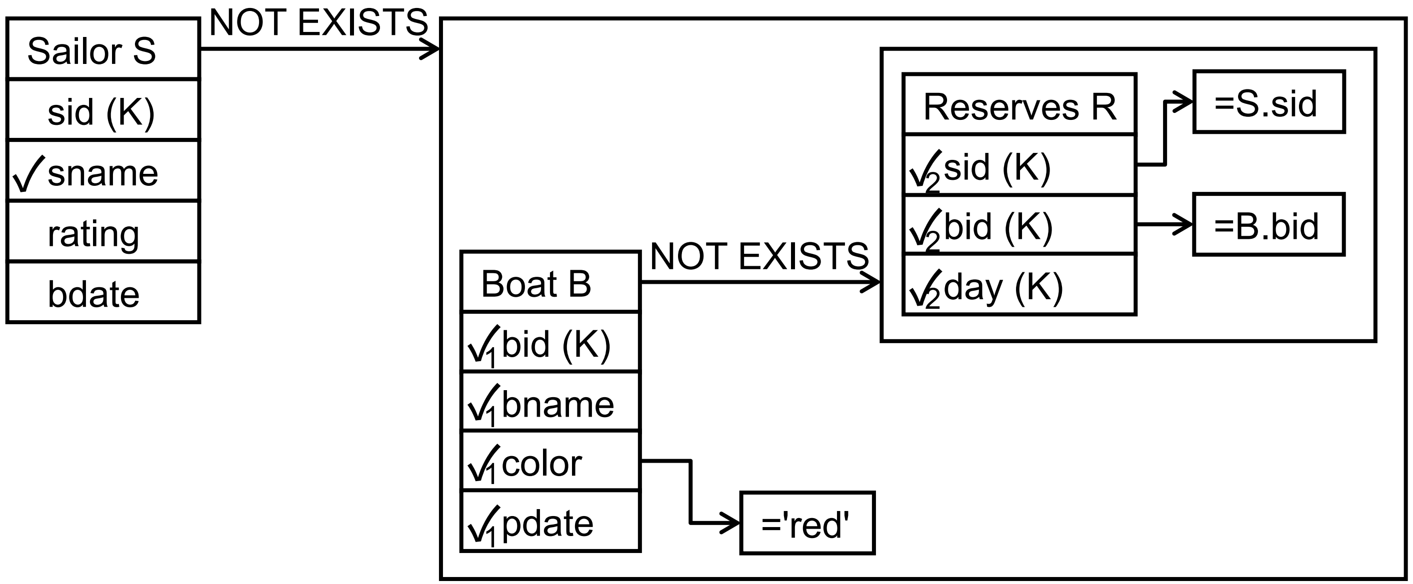


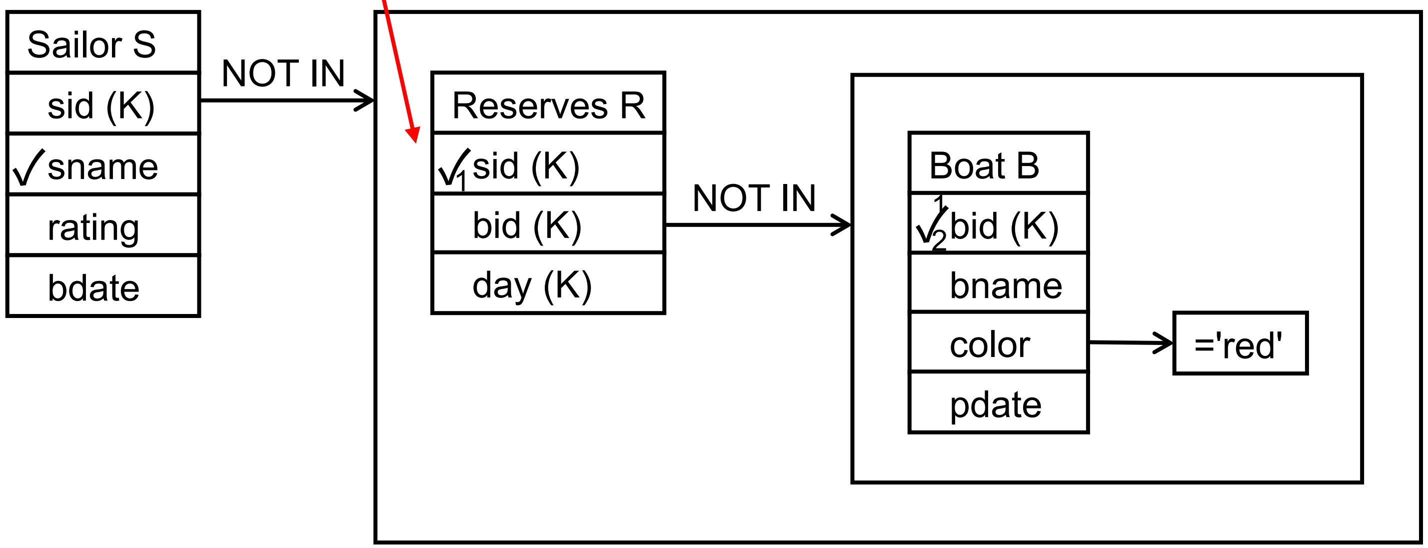
Figure drawn based on Figure 49 in "Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003."

Visual SQL

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where S.sid not in
    (select R.sid
    from Reserves R
    where R.bid not in
        (select B.bid
        from Boat B
        where color = 'red'))
```

"NOT IN" is represented as connection between an attribute and the subquery. The projected attribute in the subquery is important.



Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

QueryVis (2011) (also QueryViz)

Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440
Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805 ,
presentation slides: https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf ,
video: https://www.youtube.com/watch?v=kVFnQRGAQls&list=PL_72ERGKF6DR4R0Cowx-LnnnqLXRf4ZjB
Leventidis, Zhang, Dunne, Gatterbauer, Jagadish, Riedewald. QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster. SIGMOD. 2020. https://doi.org/10.1145/3318464.3389767

QueryVis (formerly QueryViz)

- The goal of QueryVis is the reverse functionality of VQLs and to visualize existing SQL queries with simple, easy-to-read diagrams
- Inspired by diagrammatic reasoning systems, uses topological properties, such as enclosure, to represent logical expressions and set-theoretic relationships
- The EDBT'11 demo takes a SQL query as input and returns a query visualization. It has been online since then (with interruptions): http://demo.queryvis.com

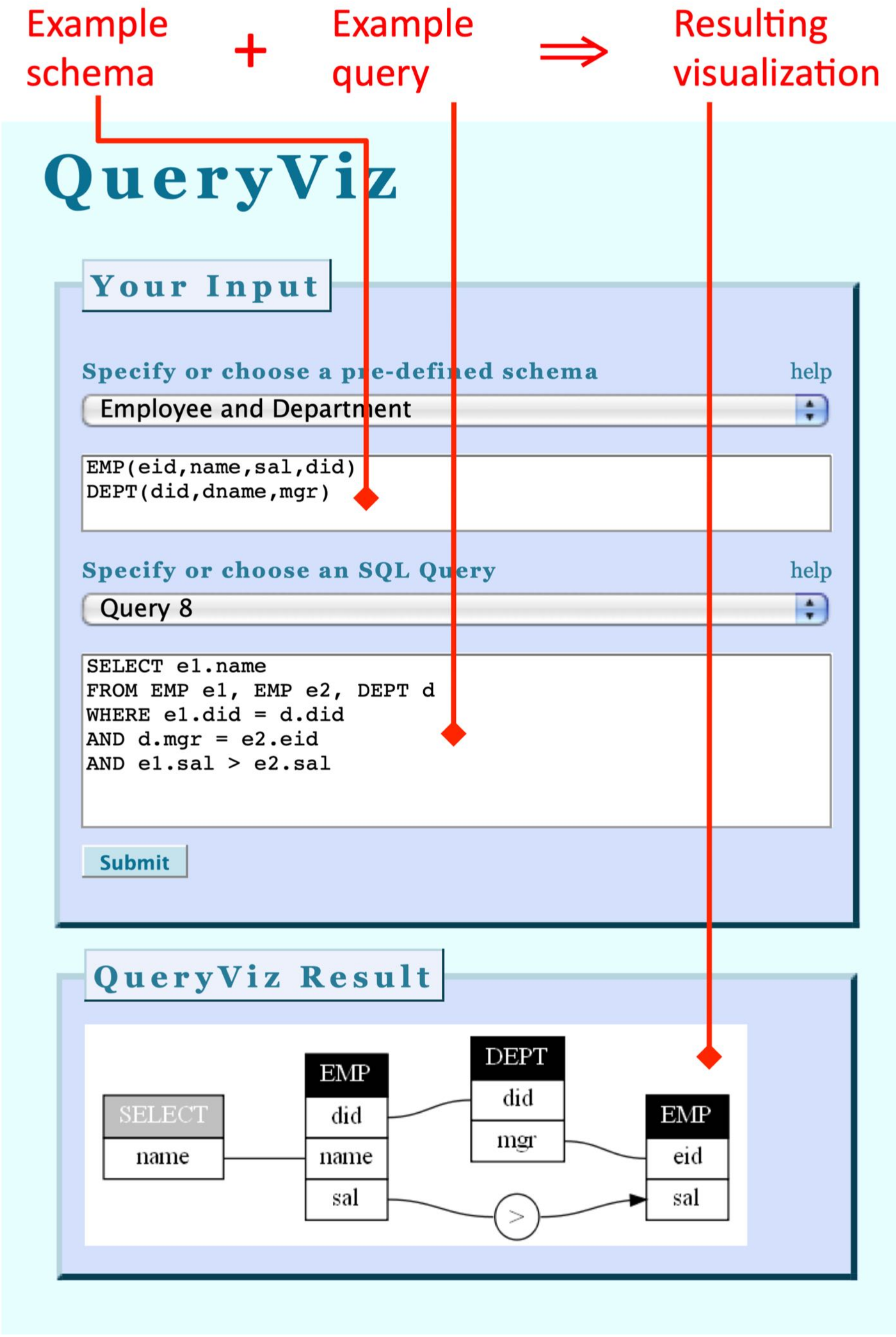
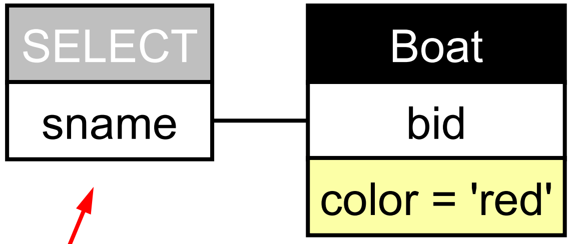


Figure source: Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440

QueryVis (formerly QueryViz)

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```



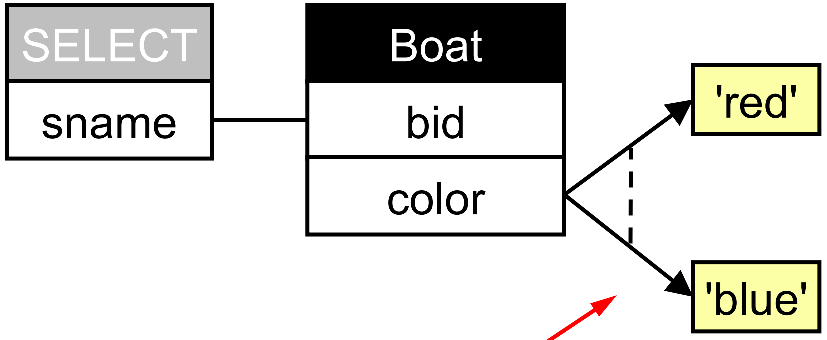
An output table is explicitly models (allows e.g. renaming of attributes)

Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find boats that are red or blue."

```sql
select bname
from Boat
where color = 'red'
or color = 'blue'
```



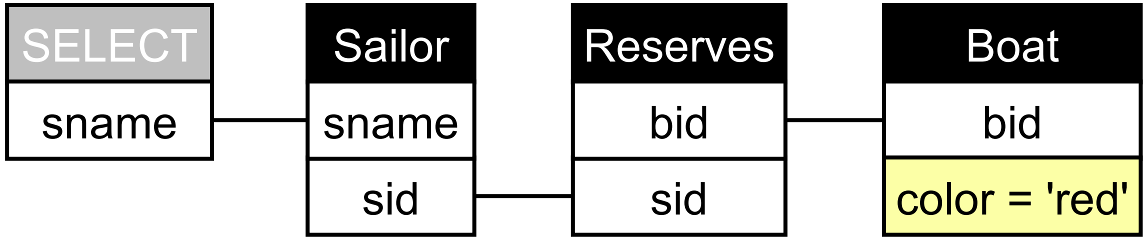
Disjunction was never implemented in online QueryVis.
Also, this is incomplete for more general disjunctions
(more on this later)

Figure drawn based on the presentations for "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805": https://www.youtube.com/watch?v=kVFnQRGAQls&list=PL_72ERGKF6DR4R0Cowx-LnnnqLXRf4ZjB
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



TRC (Tuple Relational Calculus)

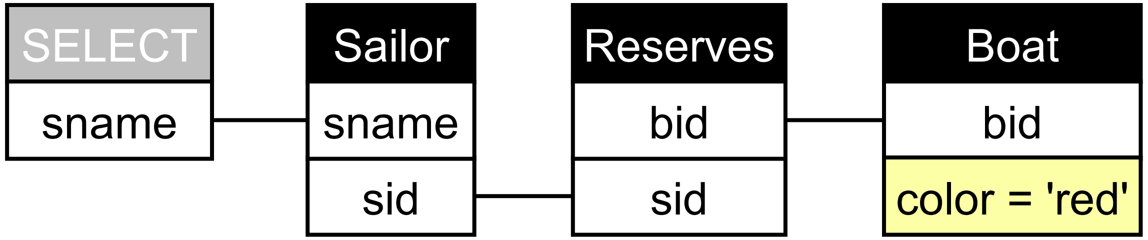
$\{q.sname \mid \exists s \in Sailor, \exists r \in Reserves, \exists b \in Boat[q.sname = s.sname \wedge r.sid = s.sid \wedge b.bid = r.bid \wedge b.color = \text{'red'}]\}$

Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
where exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and exists
        (select *
        from from Boat B
        where R.bid=B.bid
        and color = 'red'))
```



QueryVis does not focus on the way the query is written but assumes the existential quantifiers pushed up as much as possible.

TRC (Tuple Relational Calculus)

$$\{q.sname \mid \exists s \in Sailor, \exists r \in Reserves, \exists b \in Boat[q.sname = s.sname \wedge r.sid = s.sid \wedge b.bid = r.bid \wedge b.color = \text{'red'}]\}$$

$$\{q.sname \mid \exists s \in Sailor[q.sname = s.sname \wedge \exists r \in Reserves[r.sid = s.sid \wedge \exists b \in Boat[b.bid = r.bid \wedge b.color = \text{'red'}]]]\}$$

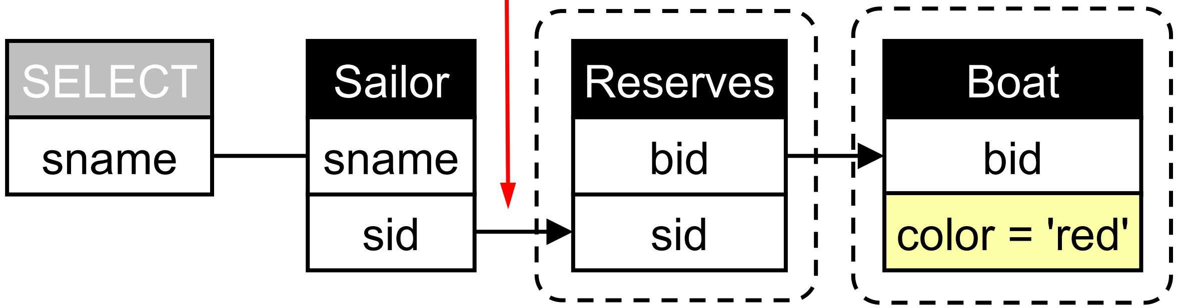
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from from Boat B
        where R.bid=B.bid
        and color = 'red'))
```

Arrows encode the reading order



Dashed lines represent "not exists" (∄)

TRC (Tuple Relational Calculus)

$\{q.sname \mid \exists s \in Sailor[q.sname=s.sname \wedge \neg(\exists r \in Reserves[r.sid=s.sid \wedge \neg(\exists b \in Boat[b.bid=r.bid \wedge b.color='red'])])]\}$

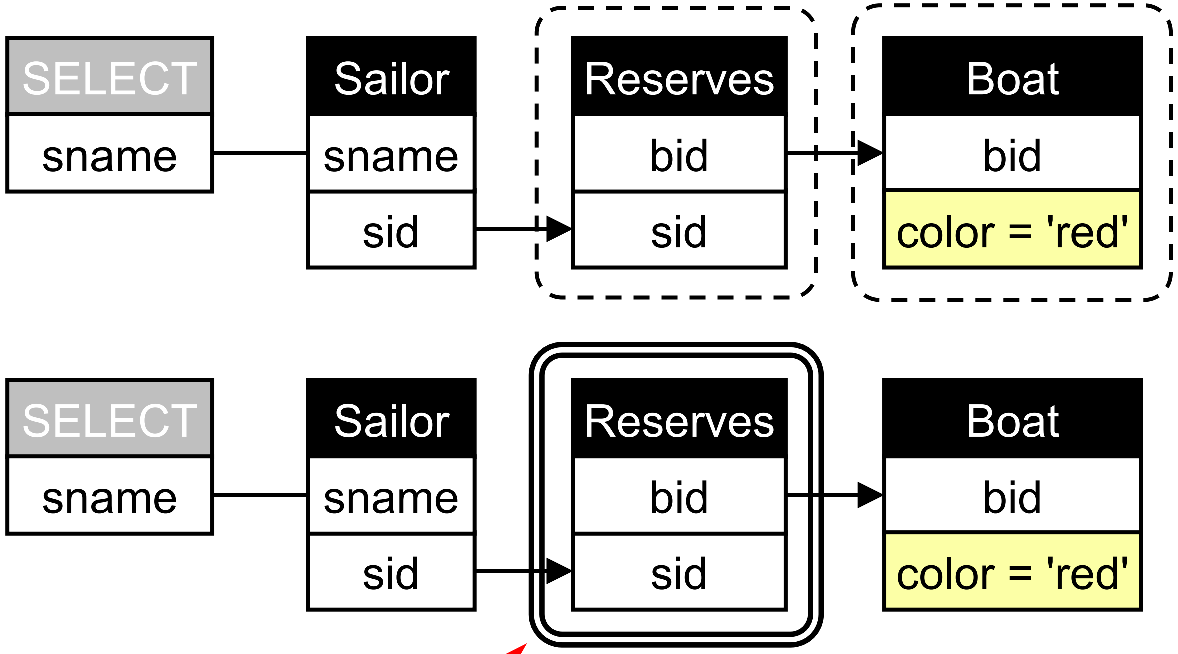
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved only red boats."

The theory of QueryVis (but not the online demo) allows a rewriting and replacing double negation with universal quantification

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from from Boat B
        where R.bid=B.bid
        and color = 'red'))
```



Double lines represent for all ∀

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ¬(∃r∈Reserves[r.sid=s.sid ∧ ¬(∃b∈Boat[b.bid=r.bid ∧ b.color='red'])])]}

{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ (∀r∈Reserves[r.sid=s.sid → (∃b∈Boat[b.bid=r.bid ∧ b.color='red'])])]}

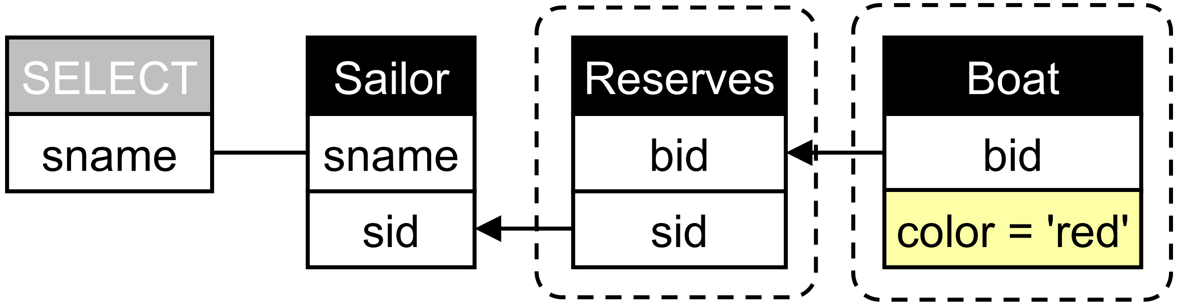
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

Correlated nested queries pose no problem.
But notice the changed arrow direction!



TRC (Tuple Relational Calculus)

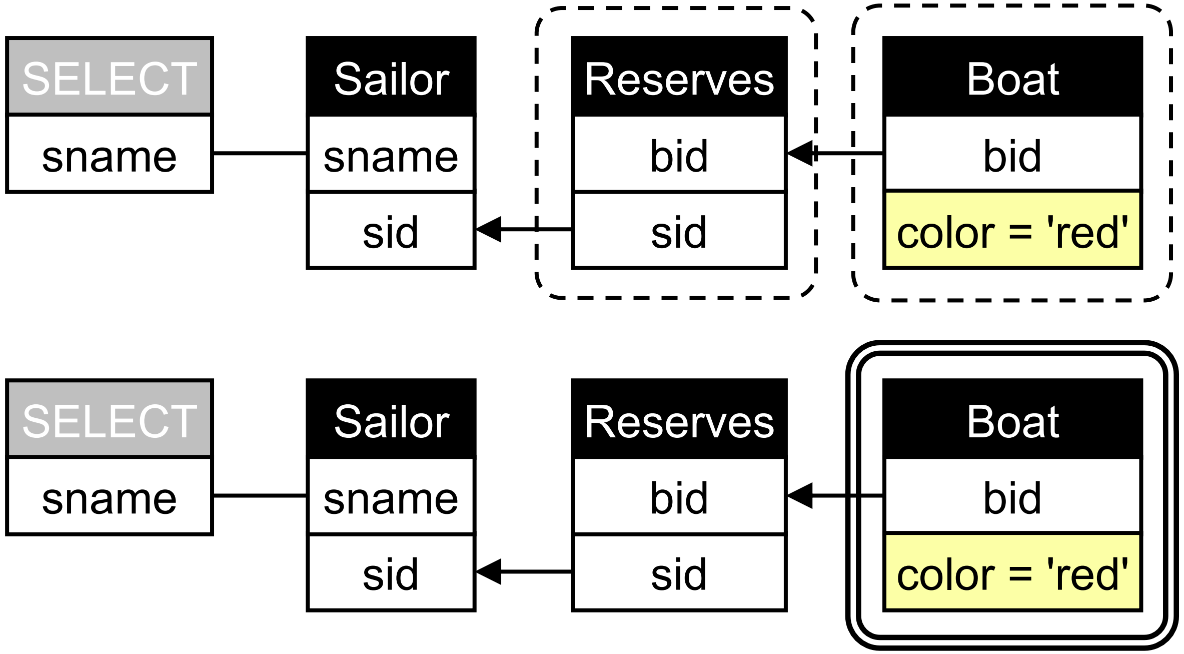
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ¬(∃b∈Boat[b.color='red' ∧ ¬(∃r∈Reserves[b.bid=r.bid ∧ r.sid=s.sid])])]}

Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly QueryViz)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```



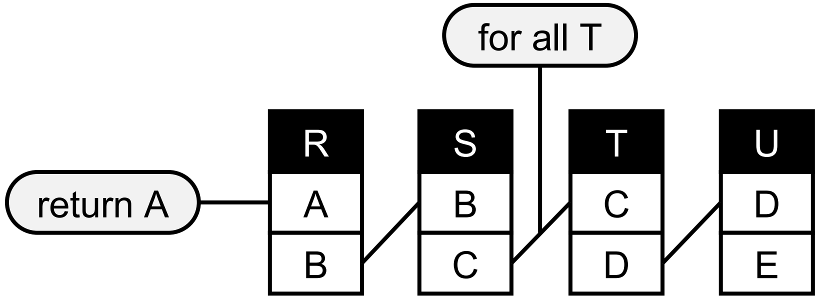
TRC (Tuple Relational Calculus)

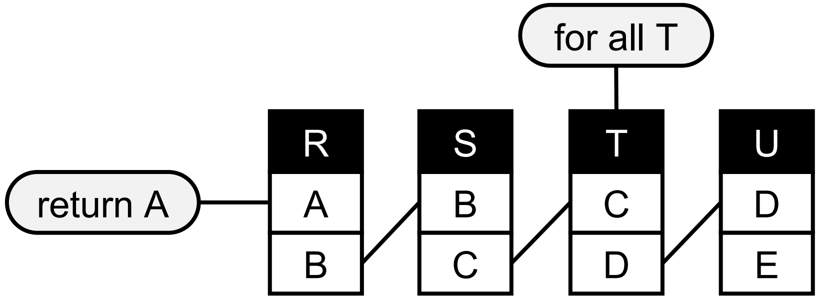
$\{q.sname \mid \exists s \in Sailor[q.sname = s.sname \wedge \neg(\exists b \in Boat[b.color = 'red' \wedge \neg(\exists r \in Reserves[b.bid = r.bid \wedge r.sid = s.sid])])]\}$

$\{q.sname \mid \exists s \in Sailor[q.sname = s.sname \wedge (\forall b \in Boat[b.color = 'red' \rightarrow (\exists r \in Reserves[b.bid = r.bid \wedge r.sid = s.sid])])]\}$

Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

The quantifier example from OO-VQS is not ambiguous

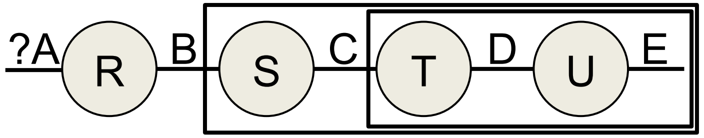


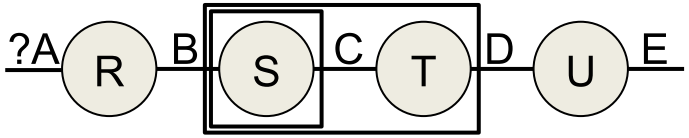


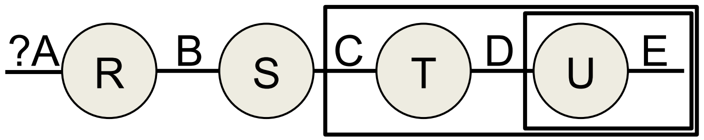
interpretation 1

interpretation 2

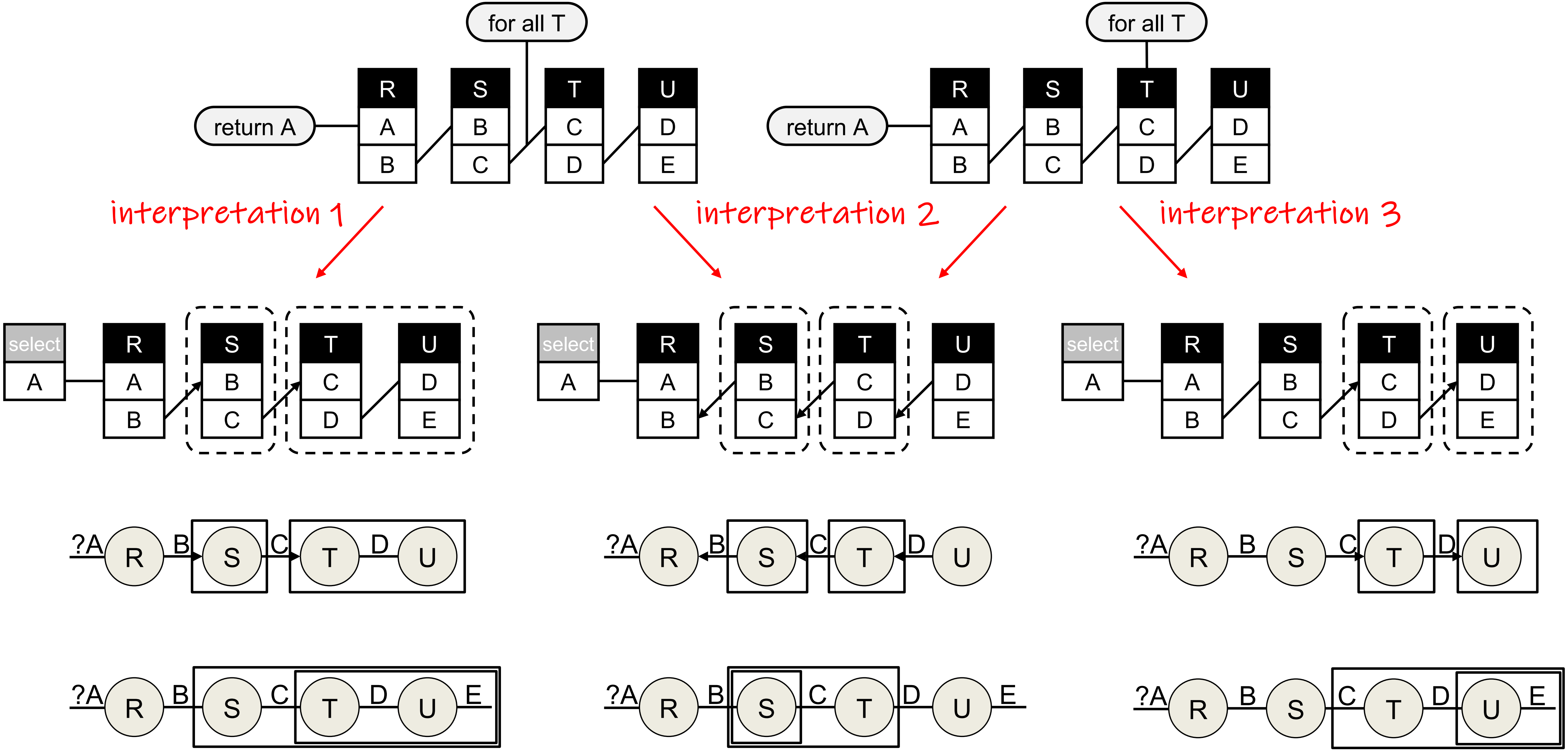
interpretation 3







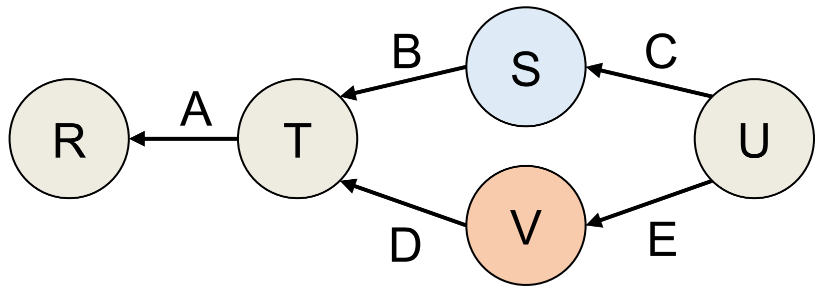
The quantifier example from OO-VQS is not ambiguous



See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

With nesting depth 4, arrows also become ambiguous

One visualization:

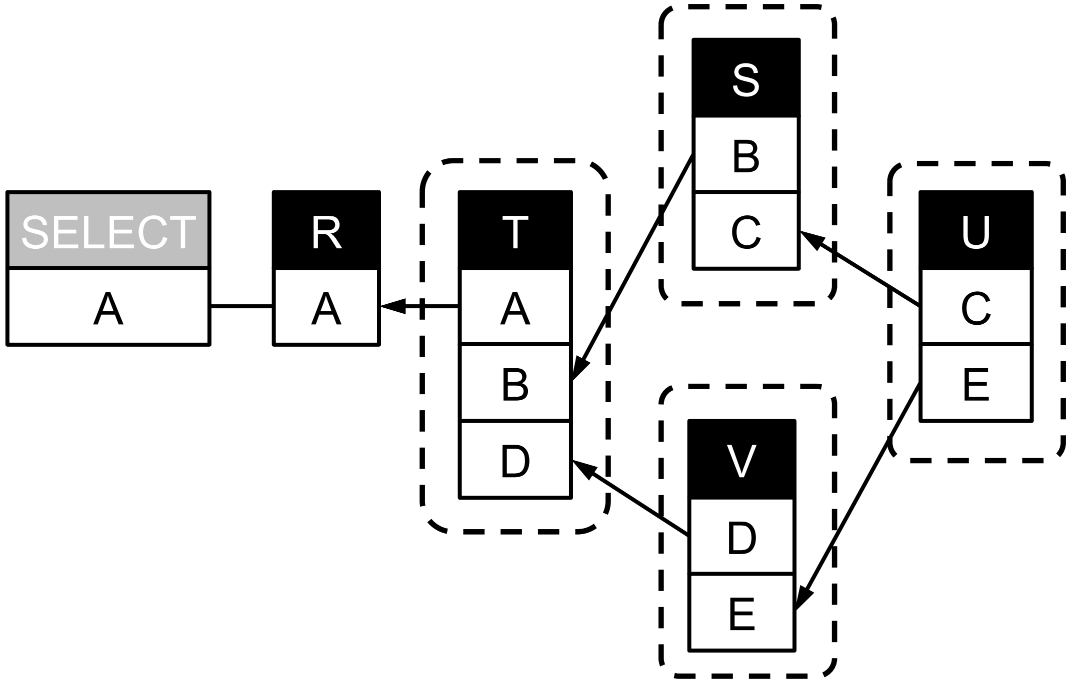


Two interpretations:

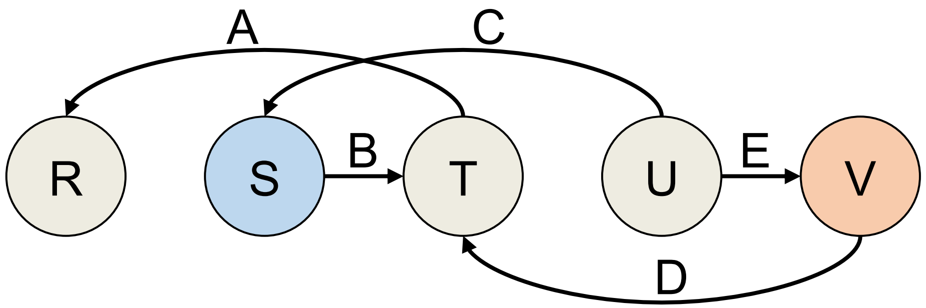
Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

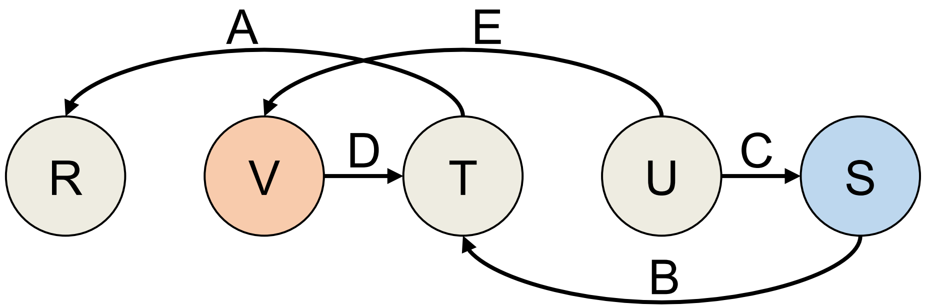
With nesting depth 4, arrows also become ambiguous

```
select distinct *
from R
where not exists
    (select *
    from S
    where not exists
        (select *
        from T
        where T.A = R.A
        and T.B = S.B
        and not exists
            (select *
            from U
            where U.C = S.C
            and not exists
                (select *
                from V
                where V.D = T.D
                and V.E = U.E))))
```




```
select distinct *
from R
where not exists
    (select *
    from V
    where not exists
        (select *
        from T
        where T.A = R.A
        and T.D = V.D
        and not exists
            (select *
            from U
            where U.E = V.E
            and not exists
                (select *
                from S
                where S.B = T.B
                and S.C = U.C))))
```





Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

Dataplay (2012)

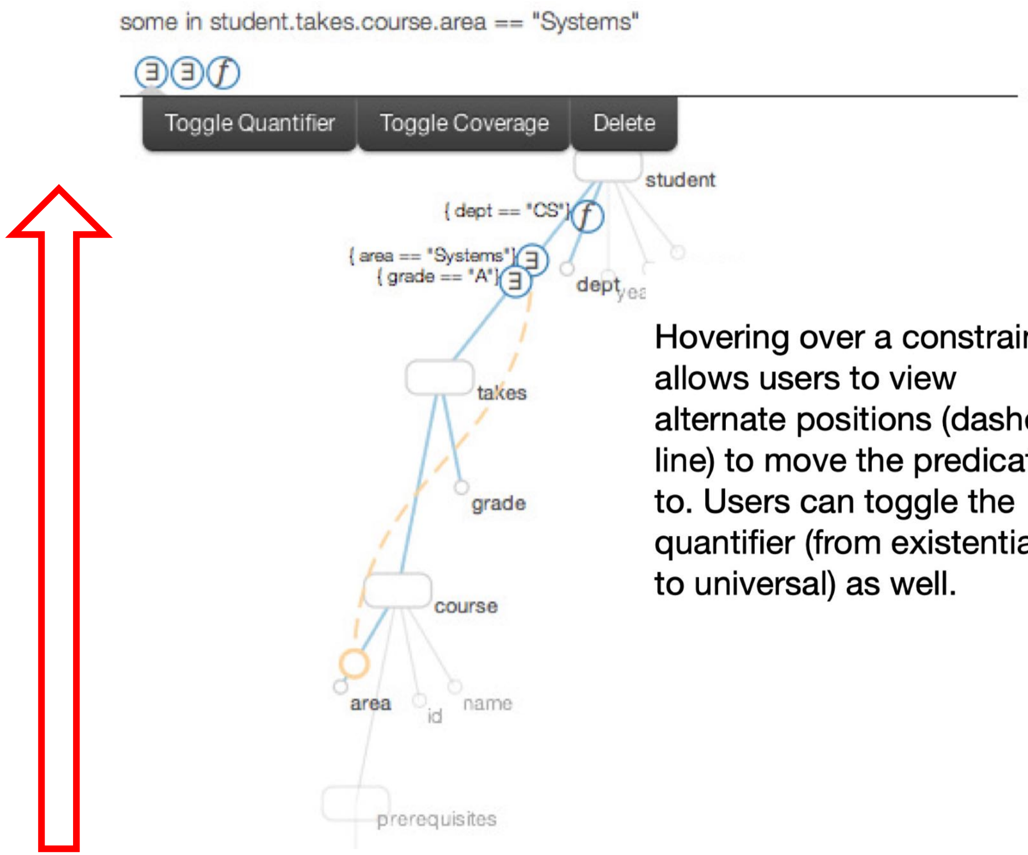
Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144
Abouzied, Hellerstein, Silberschatz. Playful Query Specification with DataPlay. VLDB demo 2012. https://doi.org/10.14778/2367502.2367542
Video tutorial: https://vimeo.com/45918228

Dataplay

- Dataplay allows users to interactively explore data starting from ERD-inspired tree representation of the database schema (similar in spirit to OO-VQL [Mohan, Kashyap'93])
- Goal is a representation that doesn't change too much with regard to simple syntactic changes like "find sailors who reserved a red boat" vs. "only red boats"

- Example of a system where the navigation path is implicitly expressed by transforming the query schema into a tree and tables appear in the selected order.
- The visualization is interpreted bottom-up
- Shows the query, and also the results

- Our focus here is on the visual abstractions and some of the advanced interactions and innovations (like hoovering, brushing) are not reflected in the visualization



Hovering over a constraint allows users to view alternate positions (dashed line) to move the predicate to. Users can toggle the quantifier (from existential to universal) as well.

Dataplay

Q: "Find boats that are not red."

```
select bid
from Boat
where color = 'red'
```

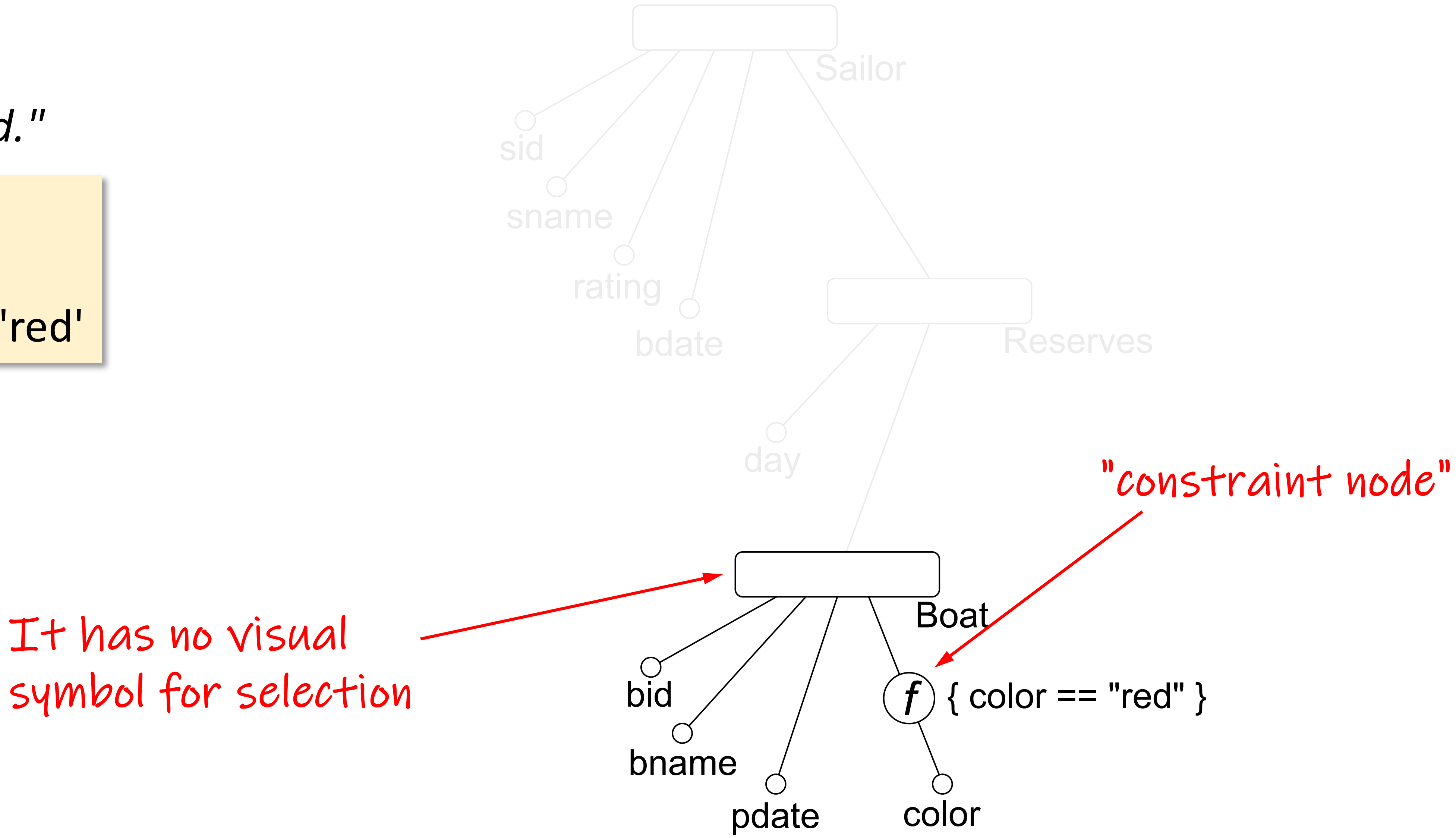


Figure drawn based on "Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144

Dataplay

Q: "Find boats that are red or blue."

```
select bid
from Boat
where color = 'red'
or color = 'blue'
```

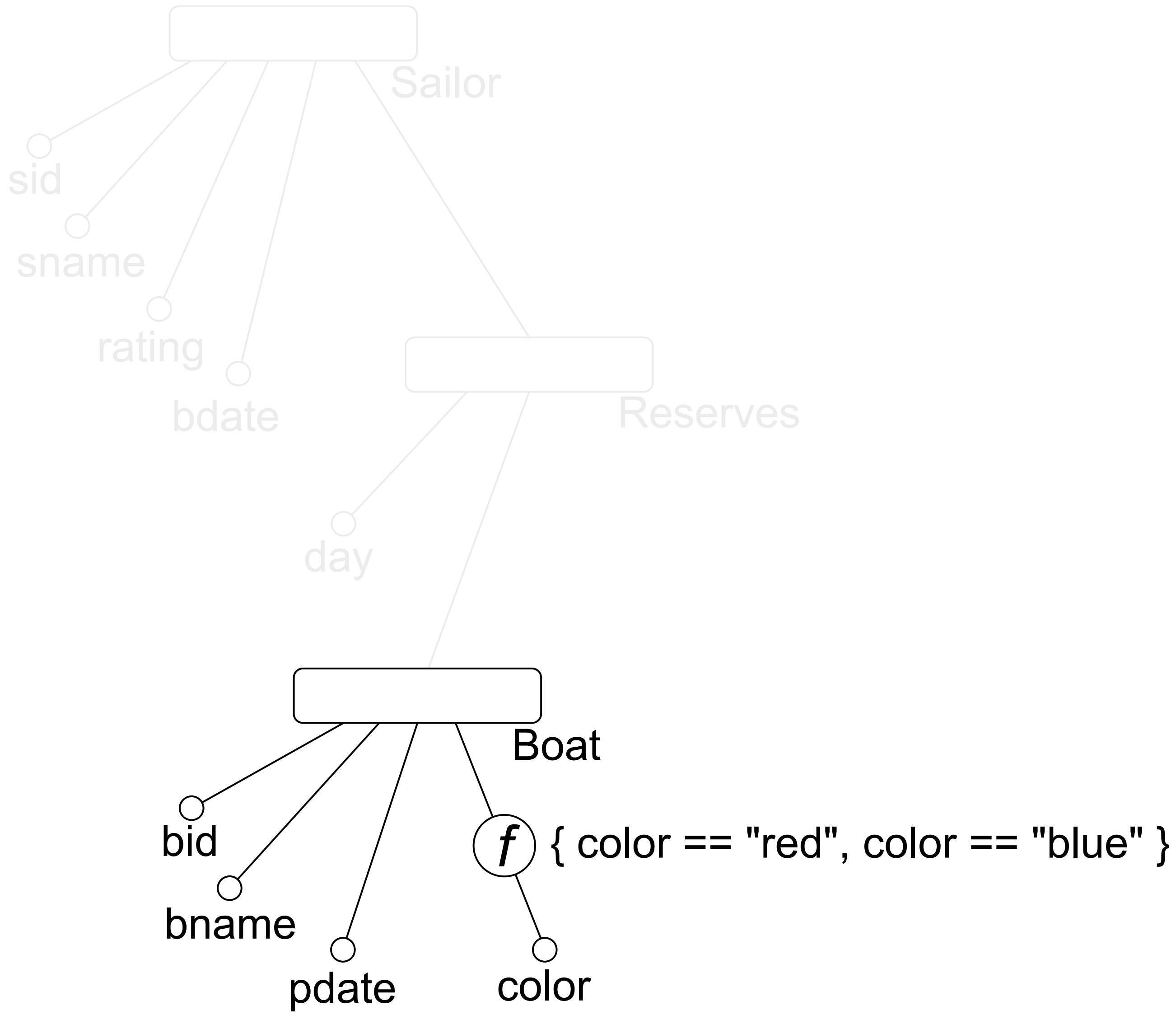


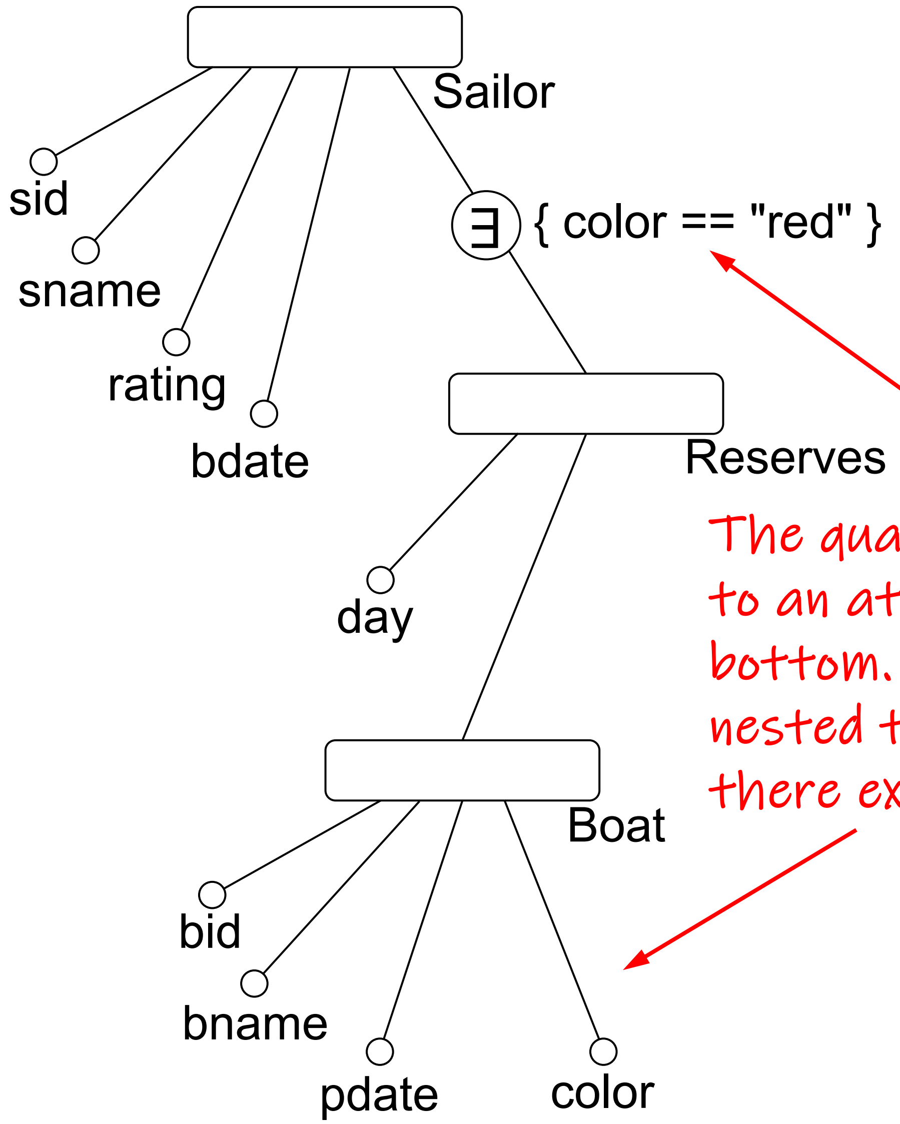
Figure drawn based on "Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144

Dataplay

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

The query tree can be interpreted as an evaluation tree but now bottom-up



The quantifier on top refers to an attribute at the bottom. It is read as: in the nested tuples for a sailor, there exists a red boat

TRC (Tuple Relational Calculus)

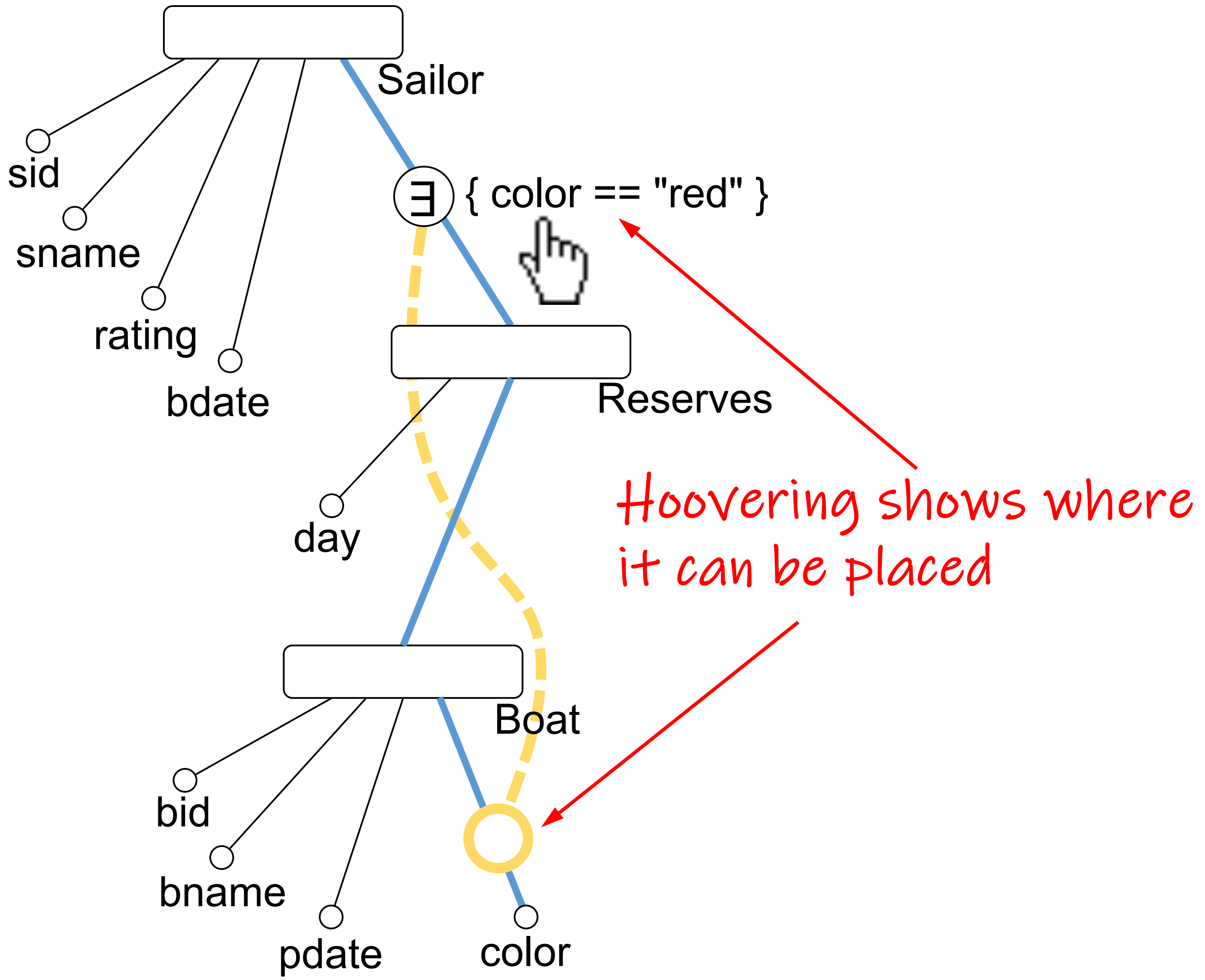
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

Figure drawn based on "Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144

Dataplay

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



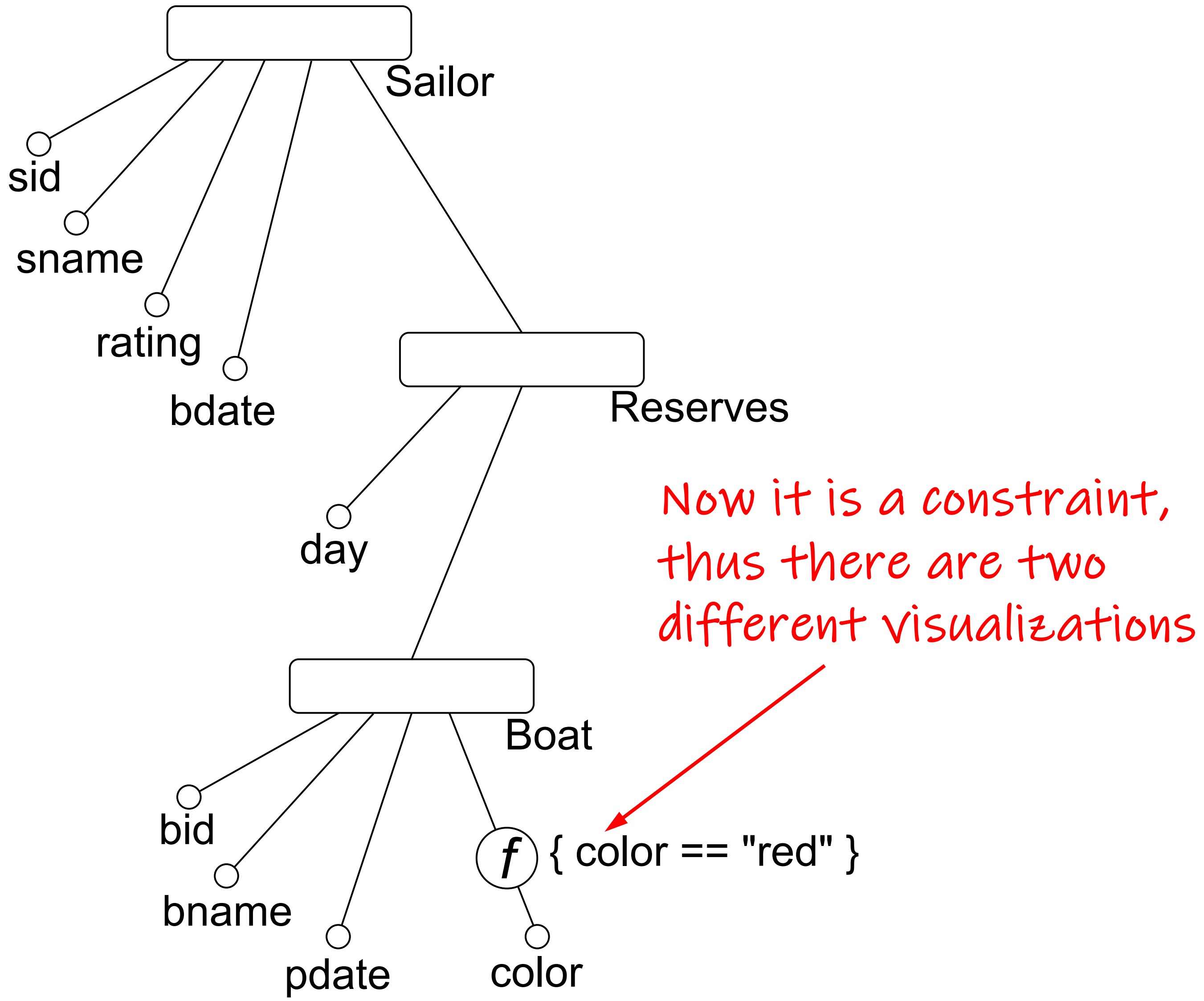
TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

Dataplay

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



TRC (Tuple Relational Calculus)

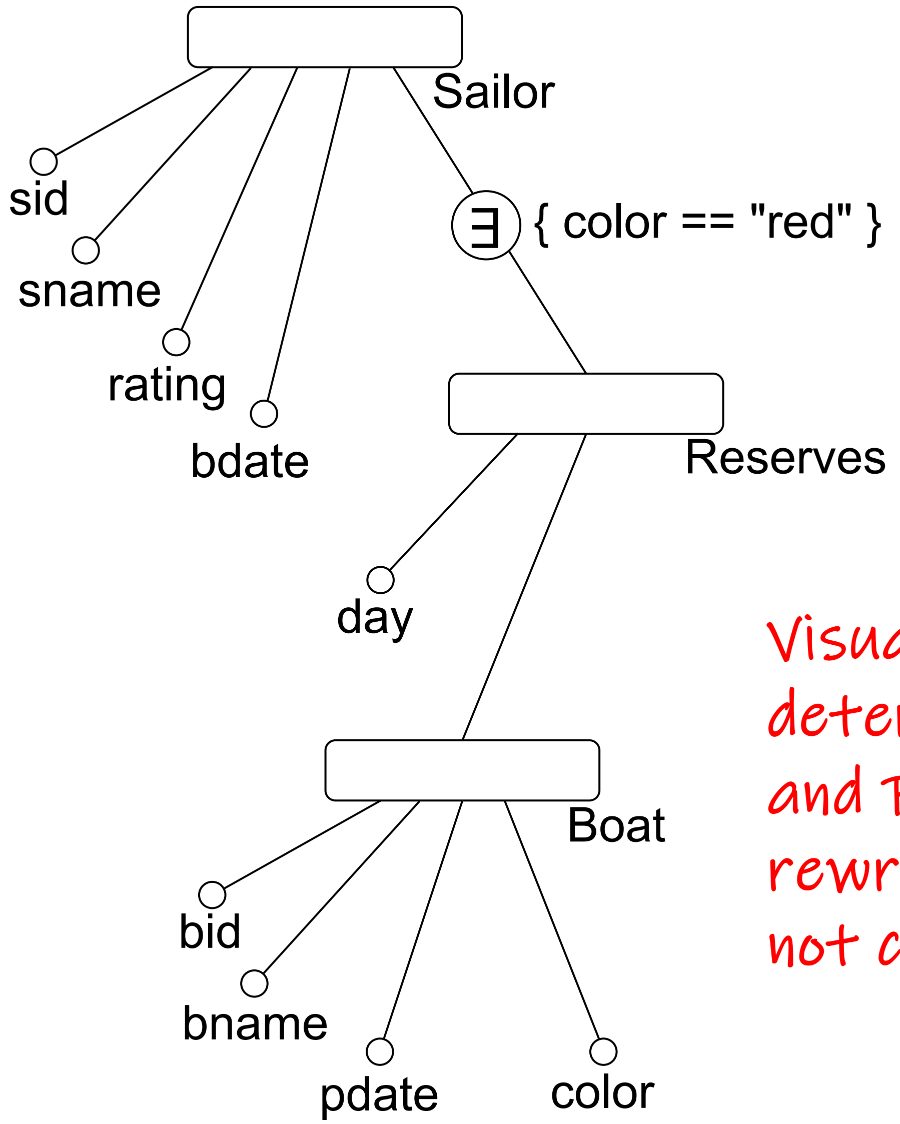
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

Figure drawn based on "Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144

Dataplay

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
where exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```



Visualization determined by schema and PK-FKs. Thus rewrite of query does not change the tree.

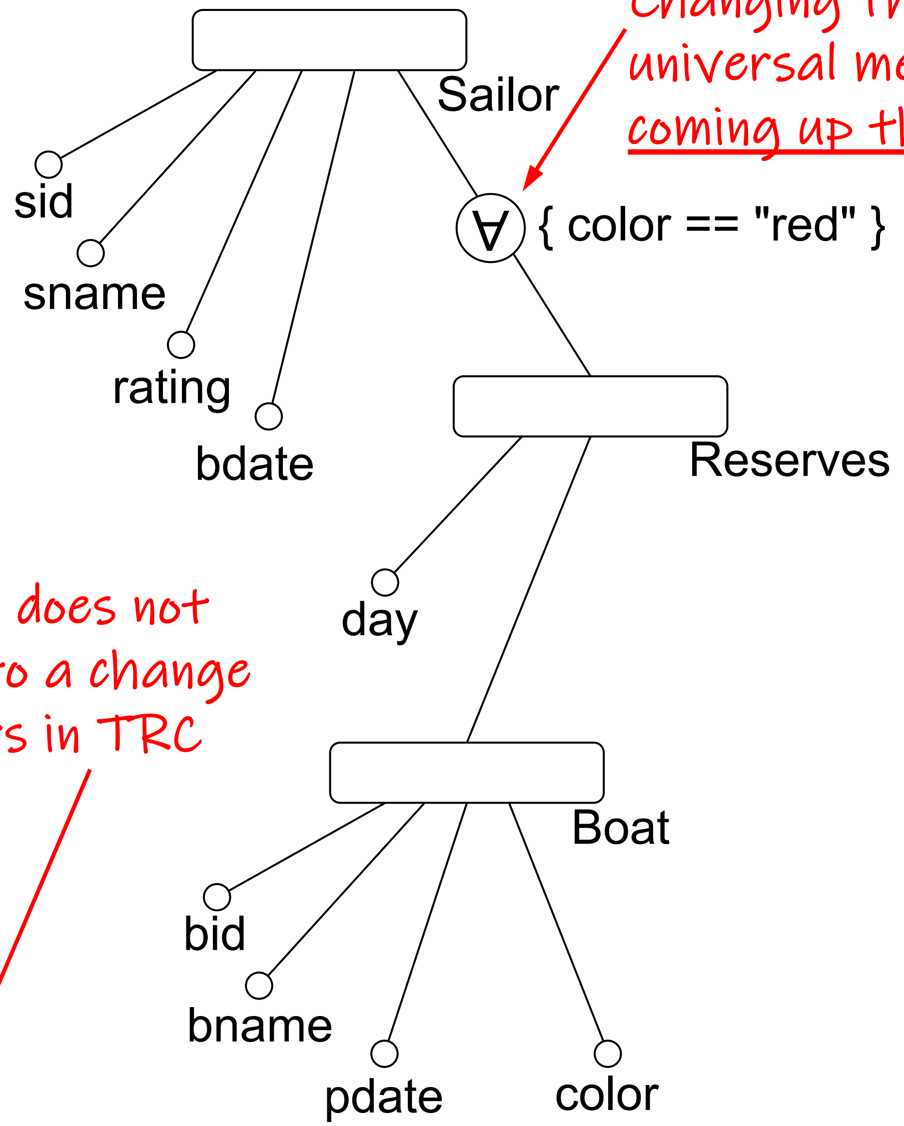
TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

Dataplay

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R
    where S.sid=R.sid
    and not exists
        (select *
        from Boat B
        where R.bid=B.bid
        and color = 'red'))
```



Changing the quantifier to universal means that <u>all tuples coming up the tree need to be red</u>

This change does not correspond to a change in quantifiers in TRC

TRC (Tuple Relational Calculus)

$\{q.sname \mid \exists s \in Sailor[q.sname = s.sname \wedge (\forall r \in Reserves[r.sid = s.sid \rightarrow (\exists b \in Boat[b.bid = r.bid \wedge b.color = \text{'red'}])])]\}$

Dataplay

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Boat B
   where color = 'red'
   and not exists
      (select *
      from Reserves R
      where S.sid=R.sid
      and R.bid=B.bid))
```

- Cannot visualize correlated nested queries because the nesting hierarchy (and thus order of quantifiers) is predetermined
- However, Dataplay can express the query *for a fixed database instance* by interaction. As example:
  - create a bar chart of boat colors
  - brush bar for red (this creates a predicate expression of (bid='123', bid='236', bid='789', ...))
  - Add this predicate expression to an exists quantifier
  - cover the quantifier this becomes exists bid='123' AND exists bid='236' AND ...
  - The resulting query asks for sailors such there exist reservations for all these boats (which is an explicit list of the red boats in the current database)

TRC (Tuple Relational Calculus)

{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ (∀b∈Boat[b.color='red' → (∃r∈Reserves[b.bid=r.bid ∧ r.sid=s.sid])])]}

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

SIEUFERD (2016)

Schema-Independent End-User Front-End for Relational Databases

Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
SIEUFERD: A Visual Query System. https://www.youtube.com/watch?v=W6xmqcb8hFQ

SIEUFERD (2016)

The crow's foot (one-to-many relationship between tables) is not necessary for understanding tables and we don't show it subsequently

- SIEUFERD is a direct manipulation spreadsheet-like interface that lets users manipulate the actual data. The interface consists of a "result header", optional popups, and the result area. We focus here only on the result header, which "encodes the structure of the query"

- The paper claims to be SQL-92-complete. The most interesting aspect for us here will be modeling negation with left joins and filters on null values

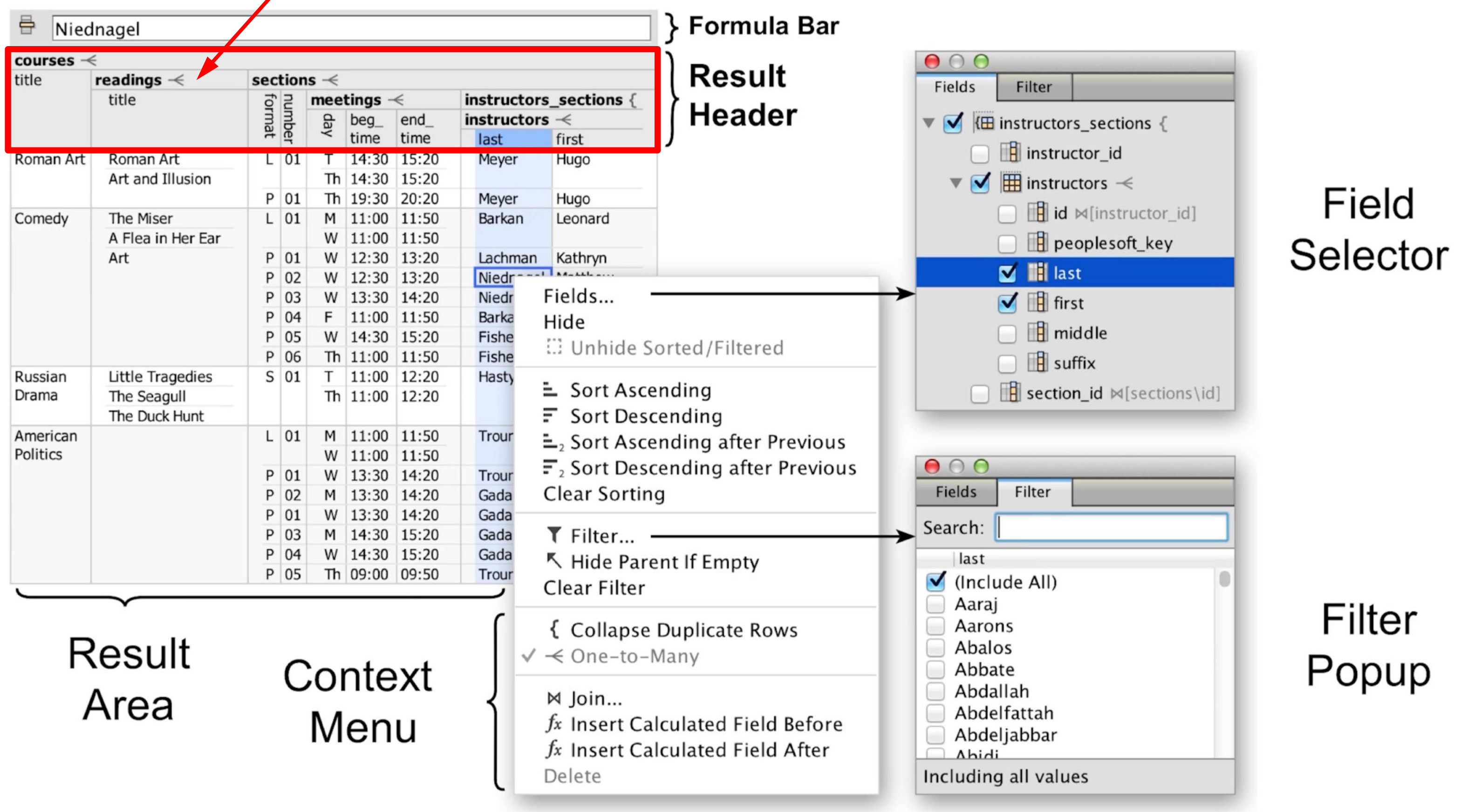


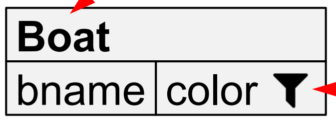
Figure source: https://www.youtube.com/watch?v=W6xmqcb8hFQ

SIEUFERD (2016)

Q: "Find boats that are not red."

```
select bid
from Boat
where color != 'red'
```

Relation names are bold, (attributes are not bold)



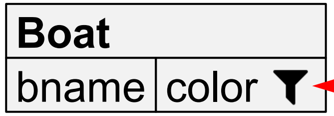
Funnel icon (▼) indicates filter: actual filter condition is only shown in a separate "filter popup" window. In that popup window, all colors other than red are chosen.

All attributes that are visible are returned. Other attributes can be hidden in the "result header", but then the query logic is not visible

SIEUFERD (2016)

Q: "Find boats that are red or blue."

```
select bid
from Boat
where color = 'red'
or color = 'blue'
```



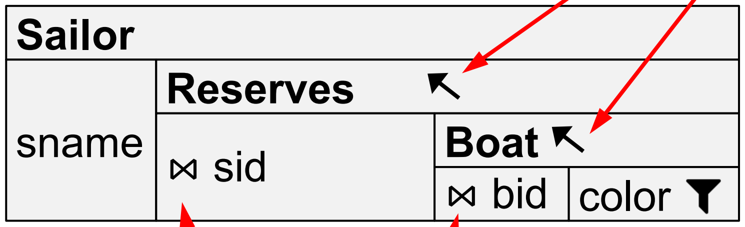
Header does not change: actual filter condition is only shown in a separate "filter pop" up window where colors "red" and "blue" are chosen

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
join Reserves R
on S.sid=R.sid
join Boat B
on R.bid=B.bid
where color = 'red'
```



"Nest equijoins" are by default treated like left joins (all tuples on the left of the join are shown even if there is no match on the right). To remove tuples on the left-hand side of the operator, a special "hide parent if empty" setting is required and indicated by the arrow-towards-root icon (↖)

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

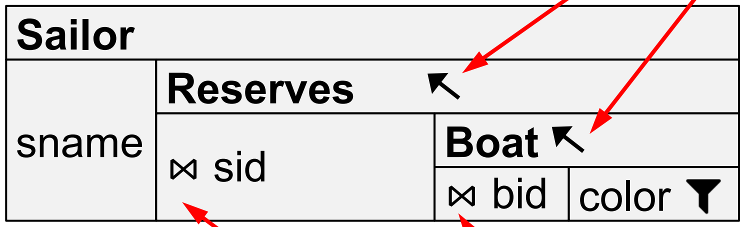
Join conditions with attributes in outer scope are not visible and only shown in a separate "field selector" popup window. Both attributes Reserves.sid and Boat. bid could also be hidden.

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```



"Nest equijoins" are by default treated like left joins (all tuples on the left of the join are shown even if there is no match on the right). To remove tuples on the left-hand side of the operator, a special "hide parent if empty" setting is required and indicated by the arrow-towards-root icon (↖)

Intuitively, SIEUFERD expresses negation via left joins. We see here on the right a SQL variant that most closely mirrors that join-based semantics

```
select S.sname
from Sailor S
join Reserves R
on S.sid=R.sid
join Boat B
on R.bid=B.bid
where color = 'red'
```

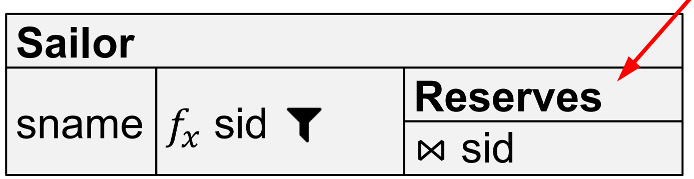
Join conditions with attributes in outer scope are not visible and only shown in a separate "field selector" popup window. Both attributes Reserves.sid and Boat.bid could also be hidden.

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

Q: "Find sailors who reserved no boat."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid)
```



Since we need to use a left join the "hide parent if empty" setting is not used, thus no arrow-towards-root icon (↖)

Intuitively, SIEUFERD expresses negation via left joins. We see here on the right a SQL variant that most closely mirrors that join-based semantics

```
select S.sname
from Sailor S
left join Reserves R
on S.sid = R.sid
where R.sid is null
```

Since the "is null" condition needs to be applied to the result of the left join (not the reserves table directly), we need to add a "reference formula" under the sailor relation that repeats the values of Reserves.sid *after* the join. And then apply a filter condition "is null"

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

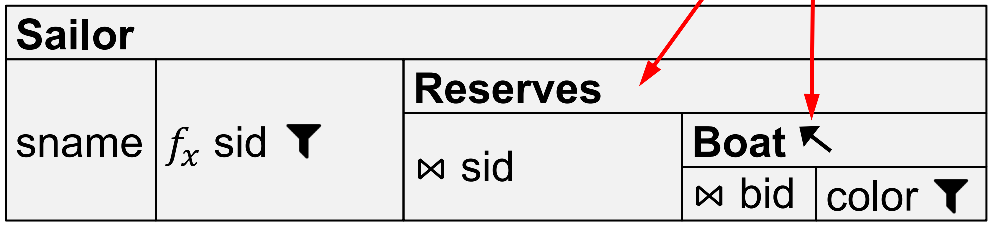
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

We need to apply the "arrow-towards-root" icon ("hide parent if empty") for Boat, but must not apply it for Reserves! See SQL explanation below

Q: "Find sailors who reserved no red boat."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Reserves R, Boat B
    where S.sid=R.sid
    and R.bid=B.bid
    and B.color='red')
```




```
select S.sname
from Sailor S
left join (Reserves R
    join Boat B
    on R.bid = B.bid
    and color = 'red')
on S.sid = R.sid
where R.sid is null
```

The query needs to propagate the filter from Boats to Reserves (thus *no* left join) forming the right part of a left join from Sailor. This operation is *not* associative, thus the required parenthesis here (and "hide parent if empty" above)

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

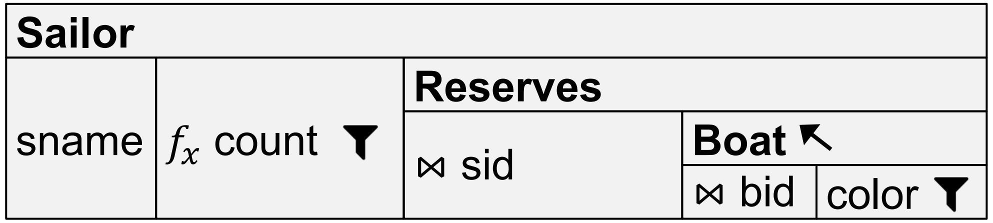
Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

Q: "Find sailors who reserved only red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid
   and not exists
      (select *
      from Boat B
      where R.bid=B.bid
      and color = 'red'))
```



An alternative syntax using aggregation and COUNT=0

```
select S.sname
from Sailor S
left join (Reserves R
   join Boat B
   on R.bid=B.bid
   and color != 'red')
on S.sid = R.sid
group by S.sid, S.sname
having count(R.sid) = 0
```

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

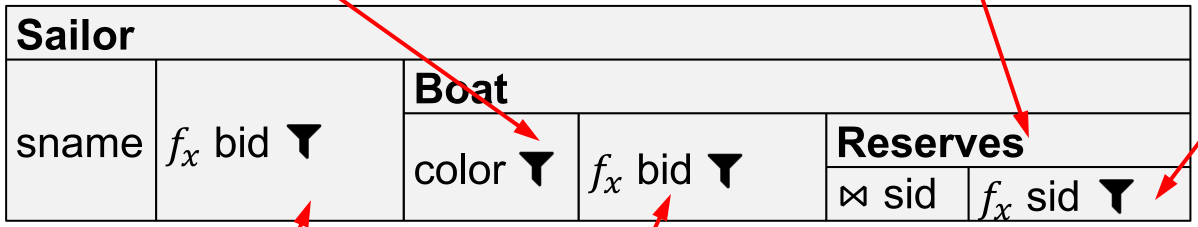
Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Boat B
   where color = 'red'
   and not exists
      (select *
      from Reserves R
      where S.sid=R.sid
      and R.bid=B.bid))
```

1. Reference to Sailor.sid / filter for Sailor.sid=Reserves.sid is true

2. Filter for color='red'

3. Left Join, thus no "arrow-towards-root"



4. Reference to Reserves.bid / then filter "is null"

5. Red boats that are not reserved by each Sailor

6. References to Boat.bid / then filter "is null"

Notice how the query header by itself does not allow a user to understand a query's semantic.

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

SQLVis (2021)

Miedema. Towards successful interaction between humans and databases. Master thesis. Eindhoven University of Technology. 2019. https://research.tue.nl/en/studentTheses/towards-successful-interaction-between-humans-and-databases
Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. https://doi.org/10.1109/VL/HCC51201.2021.9576431
SQLVis library: https://github.com/Giraphne/sqlvis (8/2023)

SQLVis

- The Goal of SQLVis is to represent SQL queries visually while a user is composing a query.
- It is thus closely capturing the actual syntax of SQL

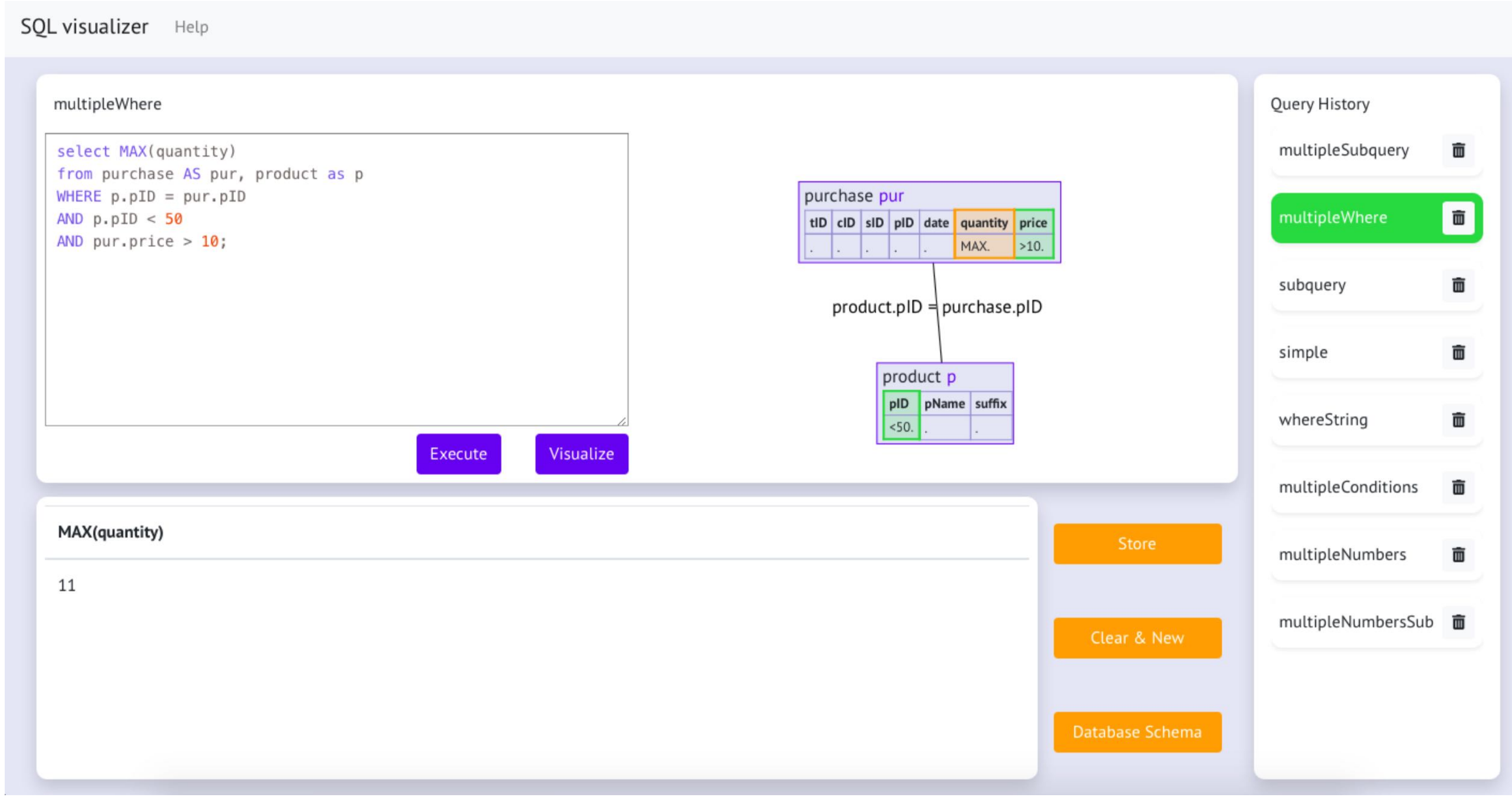


Figure source: "Miedema. Towards successful interaction between humans and databases. Master thesis. Eindhoven University of Technology. 2019. https://research.tue.nl/en/studentTheses/towards-successful-interaction-between-humans-and-databases "

SQLVis: one table

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```

Tables are by default "collapsed".

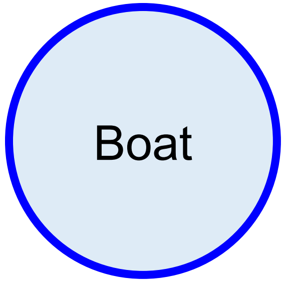


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 3. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SQLVis: one table

Q: "Find boats that are not red."

```
select bname
from Boat
where color != 'red'
```

Tables are by default "collapsed".

The expanded view shows all attributes horizontally, in a tabular form inspired by QBE

Returned attributes in orange (alias not possible)

Selections in green.

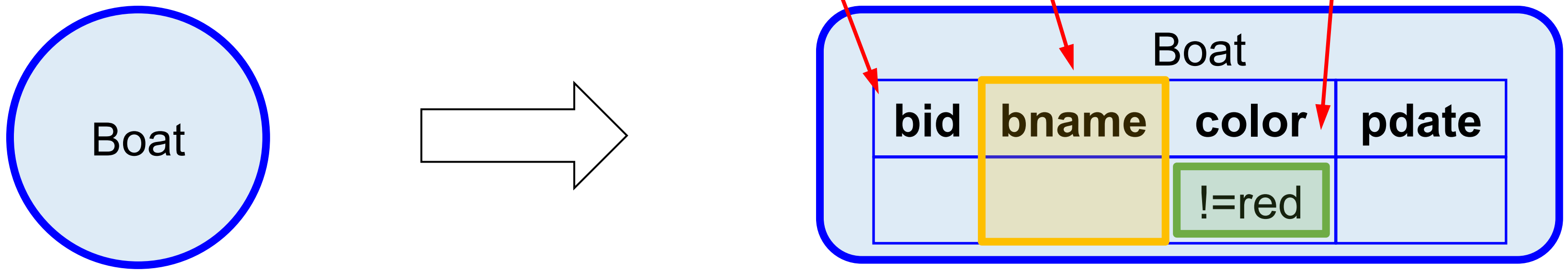


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 3. https://doi.org/10.1109/VL/HCC51201.2021.9576431

SQLVis: one table

Q: "Find boats that are red or blue."

```
select bname
from Boat
where color = 'red'
or color = 'blue'
```

Disjunctions are written in different rows (like for QBE)

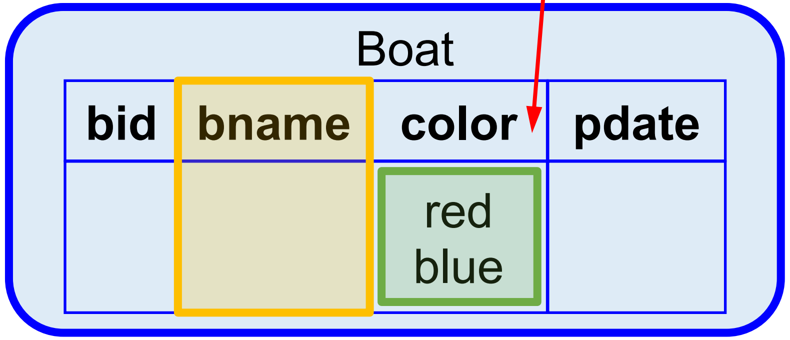


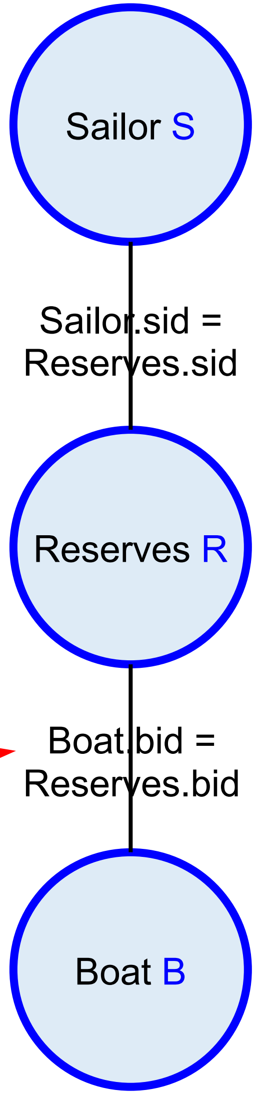
Figure drawn based on https://github.com/Giraphne/sqlvis (8/2023)

SQLVis: multiple tables

Q: "Find sailors who reserved a red boat."

```sql
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Join conditions shown as atomic label (text) on line between tables



SQLVis: multiple tables

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

Join conditions shown as atomic label (text) on line between tables

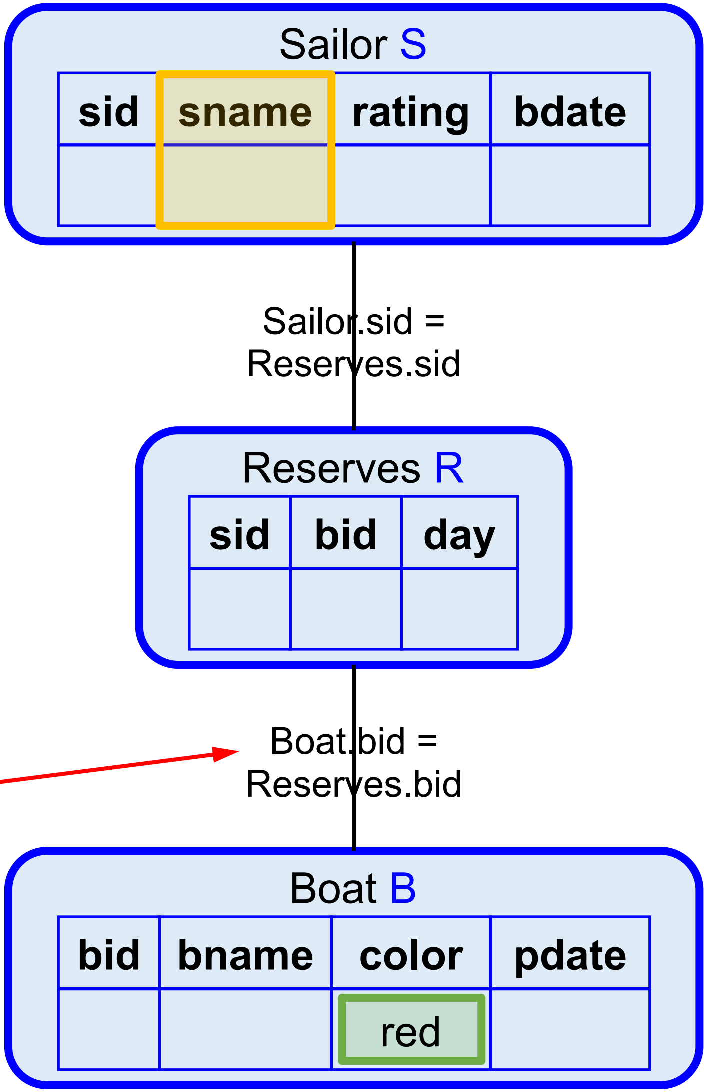


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SQLVis: multiple tables

Q: "Find sailors who reserved a red boat."

```
select S.sname
from Sailor S
where exists
    (select *
    from Boat B
    where color = 'red'
    and exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

boxes with different saturation drawn for subqueries

Join conditions shown as atomic label (text) on line between tables

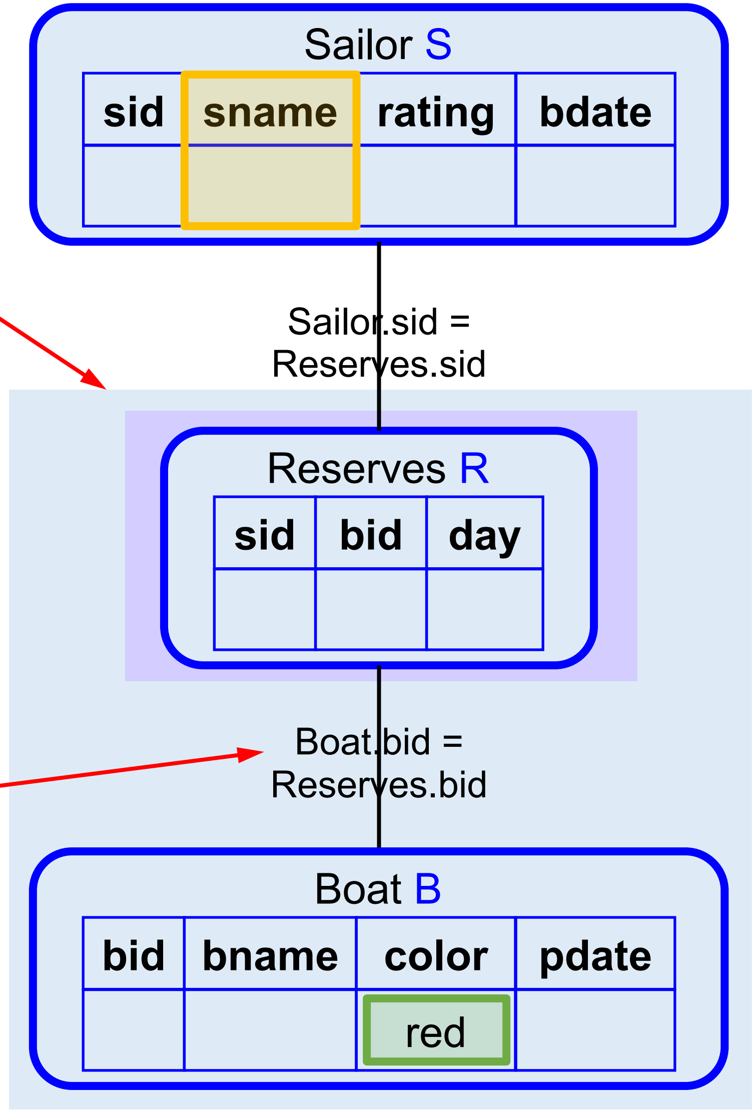


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SQLVis: multiple tables

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```

boxes with different saturation drawn for subqueries

negation for negated subqueries are expressed in words

Join conditions shown as atomic label (text) on line between tables

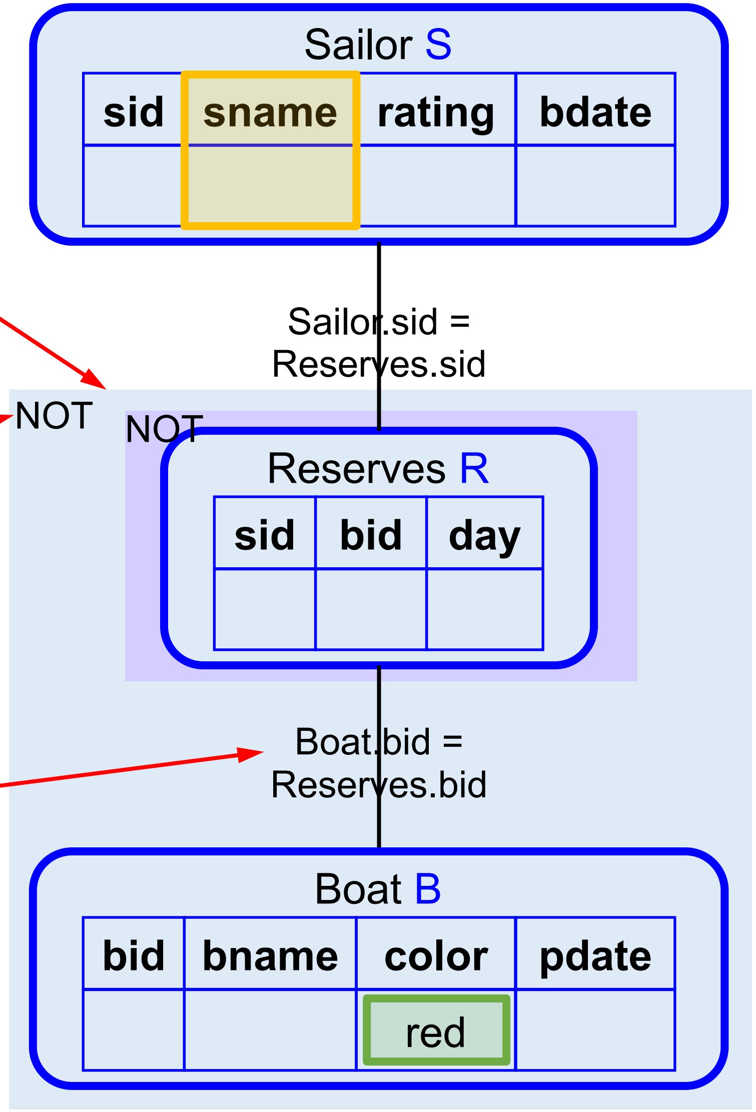


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Intended Agenda today

1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. Early diagrammatic representations
5. Visual Query Representations for Databases
6. Various Open Challenges

1: Showing Disjunctions diagrammatically is hard

Text:

Boat
bid
color = 'red' or 'blue'

Diagram: ?

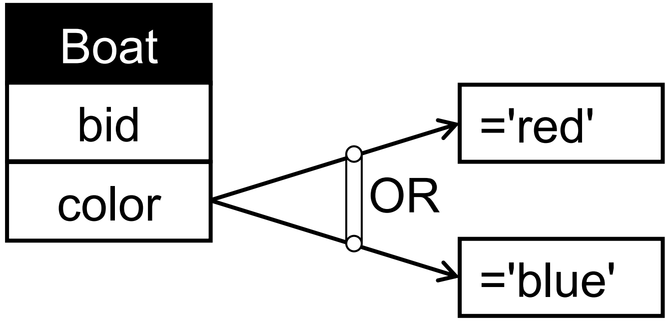
1: Showing Disjunctions diagrammatically is hard

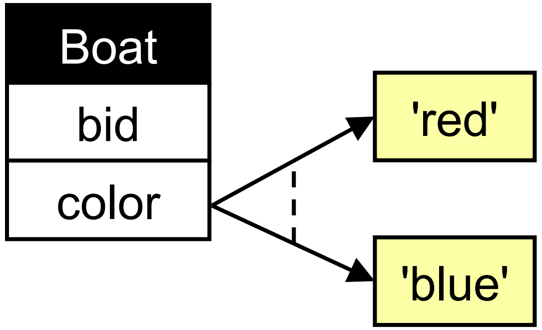
Text:

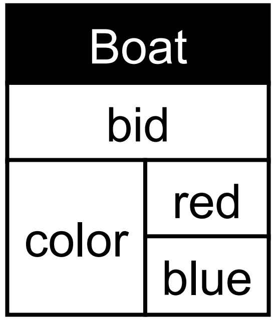
Boat
bid
color = 'red' or 'blue'

But it can get far more complicated ☹

Diagram:







1: Showing Disjunctions diagrammatically is hard

R(A,B,C)
S(A)

How to visually represent arbitrary Boolean formulas?

```
select exists
    (select *
    from R
    where (R.B = 0
        and exists
            (select *
            from S
            where R.A = S.A))
        and exists
    or (R.B = 1
        and R.C = 2))
```

?

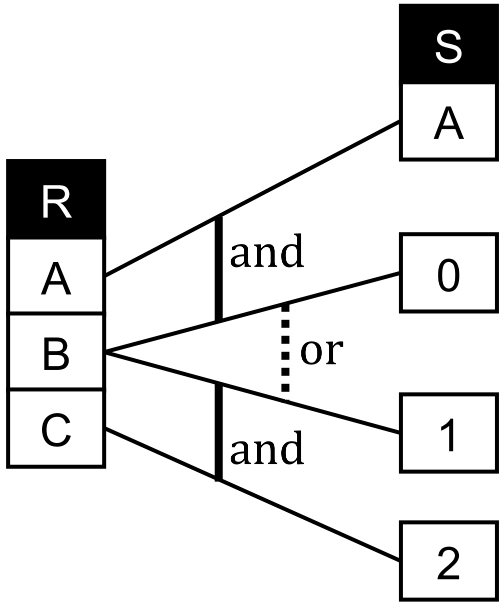
1: Showing Disjunctions diagrammatically is hard

R(A,B,C)
S(A)

How to visually represent arbitrary Boolean formulas?

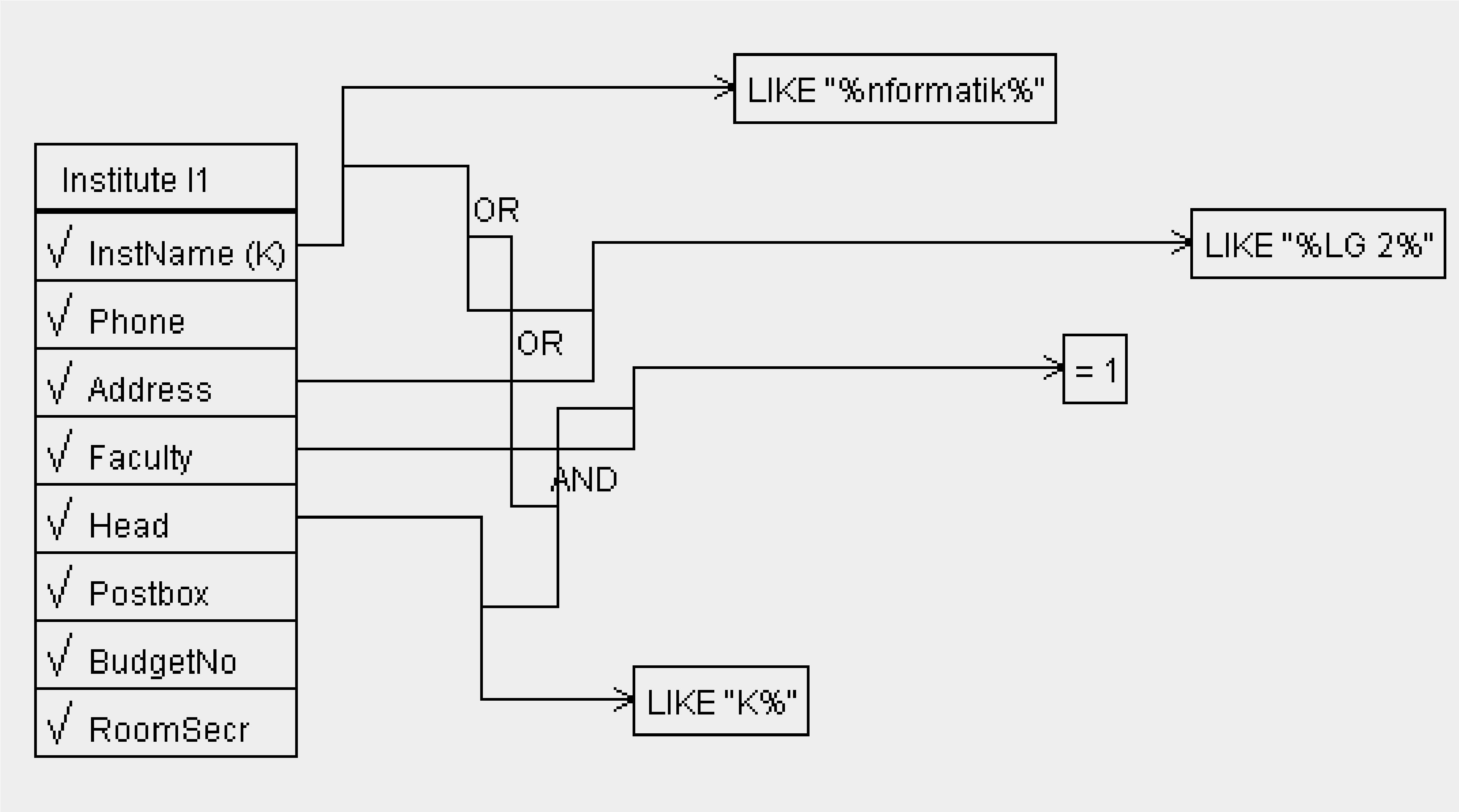
```
select exists
    (select *
    from R
    where (R.B = 0
        and exists
            (select *
            from S
            where R.A = S.A))
        and exists
    or (R.B = 1
        and R.C = 2))
```

?



quickly becomes ambiguous

1: Showing Disjunctions diagrammatically is hard

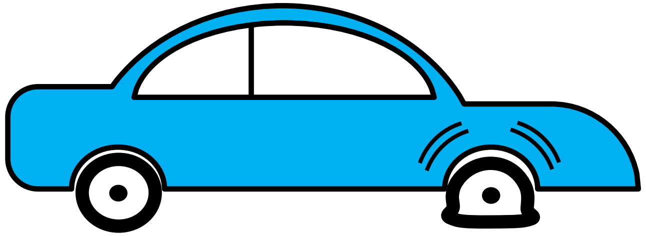


There is no simple way to represent Boolean formulas.

Figure from "Thalheim. Visual SQL as the Alternative to Linear SQL, Talk slides, 2013."

1: Why is visualizing disjunctive information harder?

I see a car that is blue
AND that has a flat tire

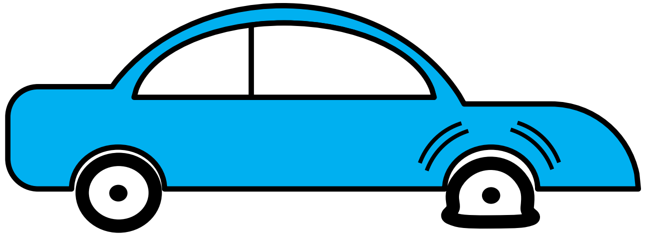


I see a car that is blue
OR that has a flat tire

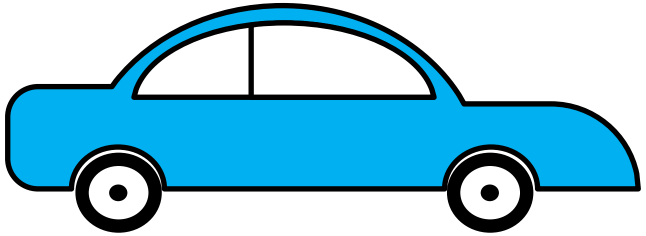
?

1: Why is visualizing disjunctive information harder?

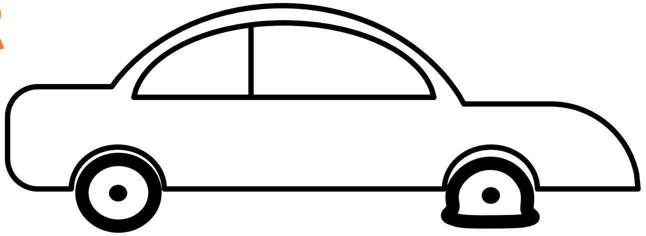
I see a car that is blue
AND that has a flat tire



I see a car that is blue
OR that has a flat tire

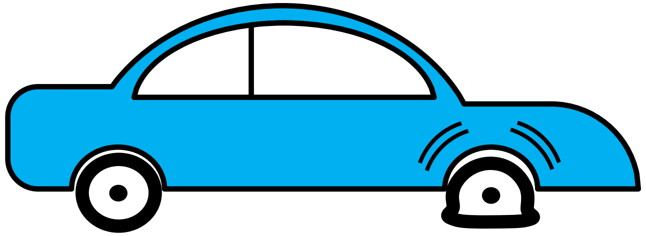


OR



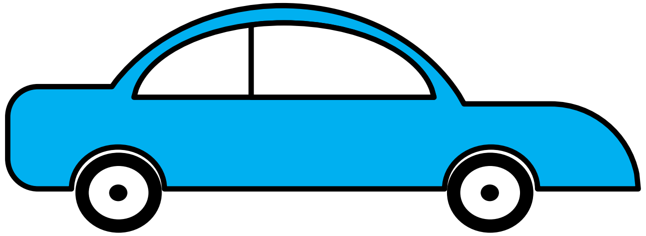
1: Why is visualizing disjunctive information harder?

I see a car that is blue
AND that has a flat tire

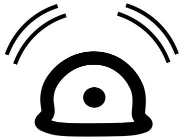


color="blue"
AND tire="flat"

I see a car that is blue
OR that has a flat tire



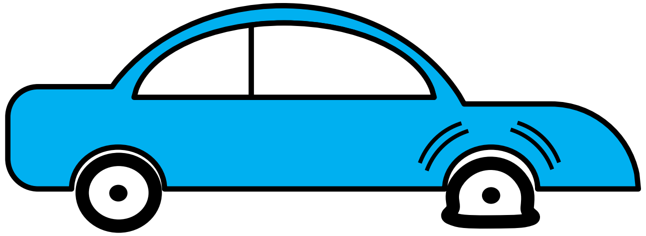
OR



color="blue" OR tire="flat"

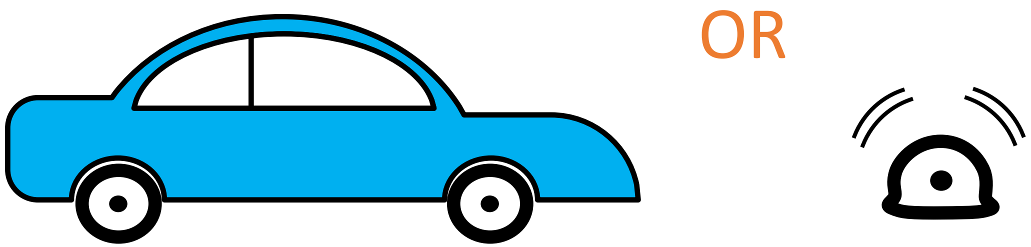
1: Why is visualizing disjunctive information harder?

I see a car that is blue
AND that has a flat tire



$\sigma_{\text{color="blue" AND tire="flat}}(\dots)$

I see a car that is blue
OR that has a flat tire

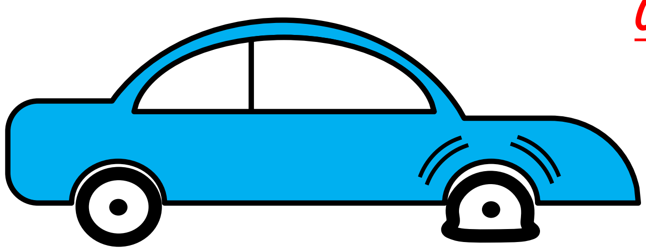


$\sigma_{\text{color="blue" OR tire="flat}}(\dots)$

1: Why is visualizing disjunctive information harder?

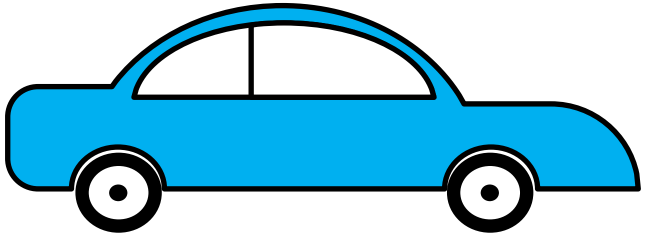
"A situation only displays <u>conjunctive</u> information."*

I see a car that is blue
AND that has a flat tire

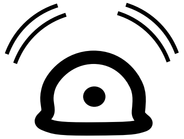


$\sigma_{\text{color="blue"}}(\sigma_{\text{tire="flat}}(\ldots))$

I see a car that is blue
OR that has a flat tire



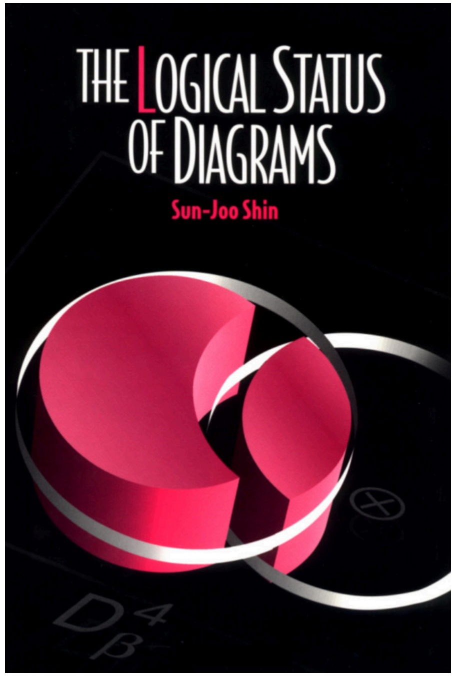
OR



$\sigma_{\text{color="blue" OR tire="flat}}(\ldots)$

* Sun-Joo Shin, "The logical status of diagrams", Cambridge university press 1994. https://doi.org/10.1017/CBO9780511574696

1: Why is visualizing disjunctive information harder? Diagrammatic reasoning systems and their expressiveness



Diagrams are widely used in reasoning about problems in physics, mathematics, and logic, but have traditionally been considered to be only heuristic tools and not valid elements of mathematical proofs. This book challenges this prejudice against visualization in the history of logic and mathematics and provides a formal foundation for work on natural reasoning in a visual mode.

The author presents Venn diagrams as a formal system of representation equipped with its own syntax and semantics and specifies rules of transformation that make this system sound and complete. The system is then extended to the equivalent of a first-order monadic language. The soundness of these diagrammatic systems refutes the contention that graphical representation is misleading in reasoning. The validity of the transformation rules ensures that the correct application of the rules will not lead to fallacies. The book concludes with a discussion of some fundamental differences between graphical systems and linguistic systems.

This groundbreaking work will have important influence on research in logic, philosophy, and knowledge representation.

objects. Conjunctive information is more naturally represented by diagrams than by linguistic formulæ. For example, a single Venn diagram can

Still, not all relations can be viewed as membership or inclusion. Shin has been careful throughout her book to restrict herself to monadic systems. Relations per se (polyadic predicates) are not considered. And while it may be true that the formation of a system (such as Venn-II) that is provably both sound and complete would help mitigate the prejudice

perception. In her discussion of perception she shows that disjunctive information is not representable in *any* system. In doing so she relies on

Source: Sun-Joo Shin, "The logical status of diagrams", Cambridge university press 1994. https://doi.org/10.1017/CBO9780511574696

2: A theory on visual minimality

How to measure "visual minimality"?

And what objective should we actually minimize?
(alphabet size, time-to-learn a representation, time-to-solve a problem, "visual happiness", ...)

3. Other extensions beyond FOL for SQL

aggregates

```sql
select  avg(price)
from  Car
where  maker='Toyota'
```

groupings

```sql
select  product, sum(quantity)
from  Purchase
where  price > 1
group by  product
```

?

3. Other extensions beyond FOL for SQL

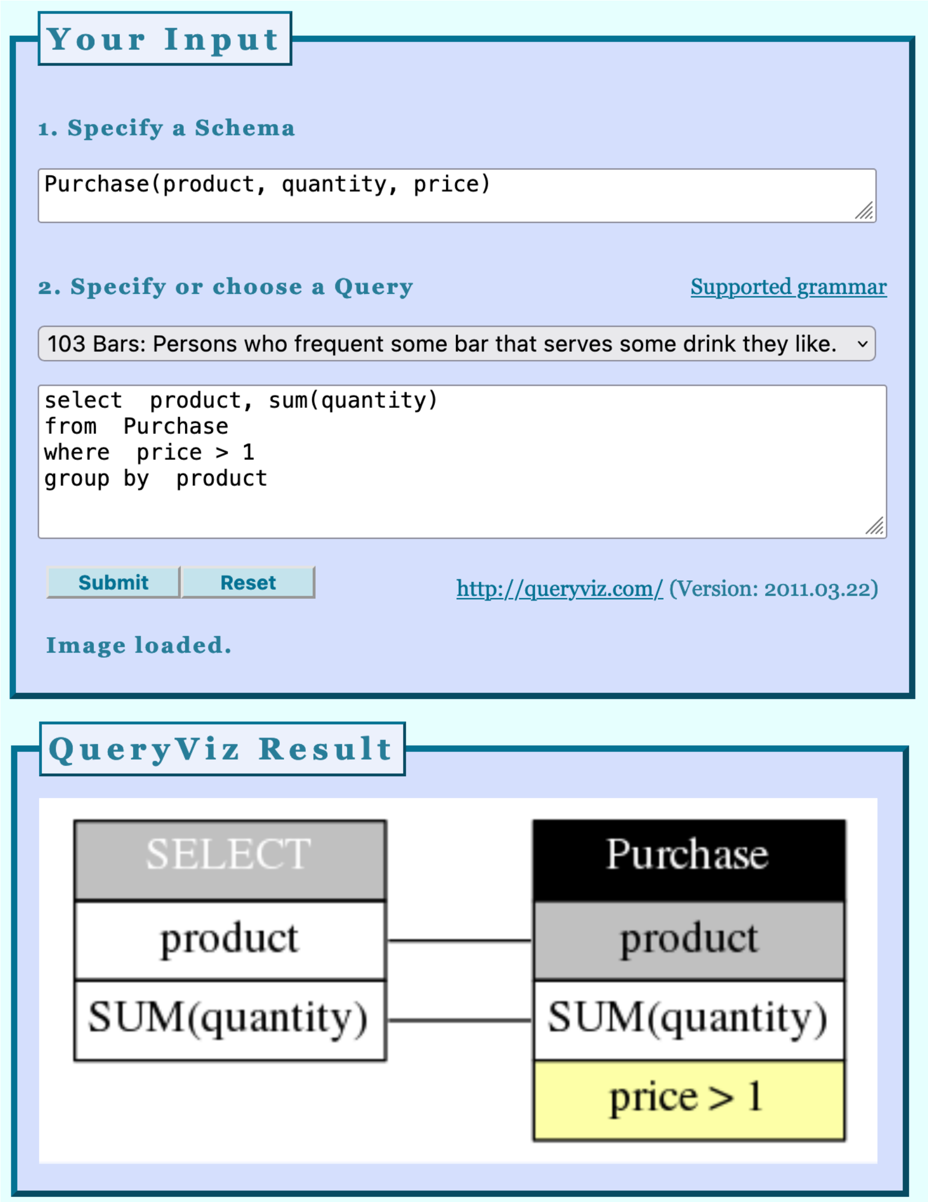
aggregates

```
select  avg(price)
from  Car
where  maker='Toyota'
```

QueryVis has actually been supporting simple groupings and aggregates (again, it can get quickly more complicated, think disjunctions but more complicated, a general solution still open,...)

groupings

```
select  product, sum(quantity)
from  Purchase
where  price > 1
group by  product
```



3. Other extensions beyond FOL for SQL

aggregates

```
select  avg(price)
from  Car
where  maker='Toyota'
```

arithmetic predicates

```
select  sum(price * quantity)
from  Purchase
```

groupings

```
select  product, sum(quantity)
from  Purchase
where  price > 1
group by  product
```

- bag semantics
- outer joins
- null values
- recursion

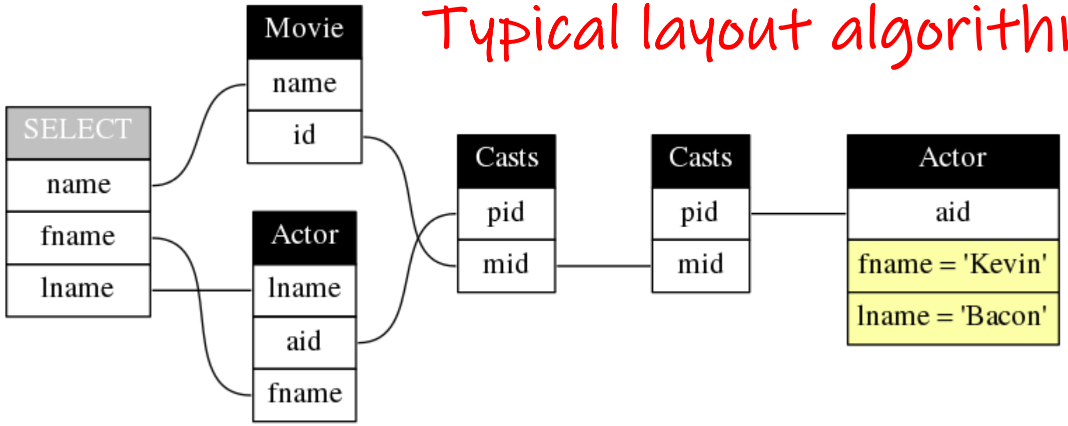
?

4. We need a principled notion of relational patterns

What are "relational patterns"?

For first steps towards "relational patterns" see: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "

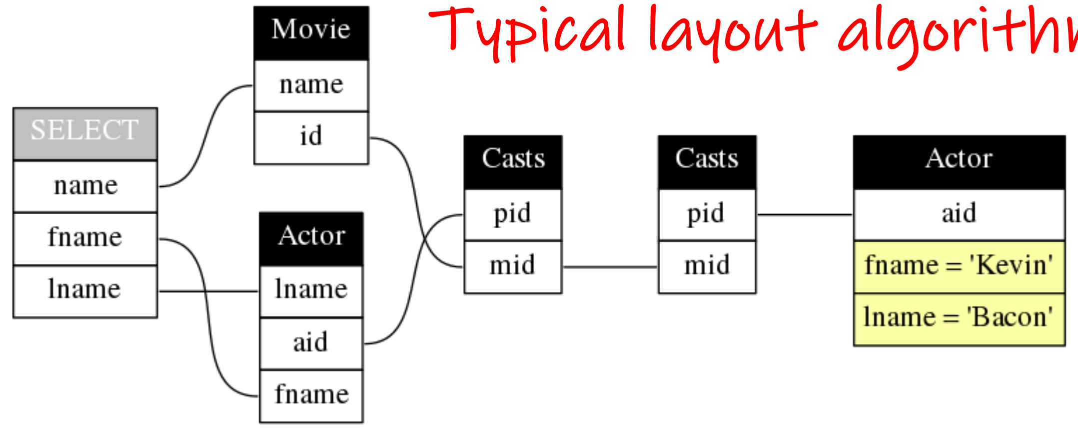
5. "Nice" layouts: Automatic layout algorithms



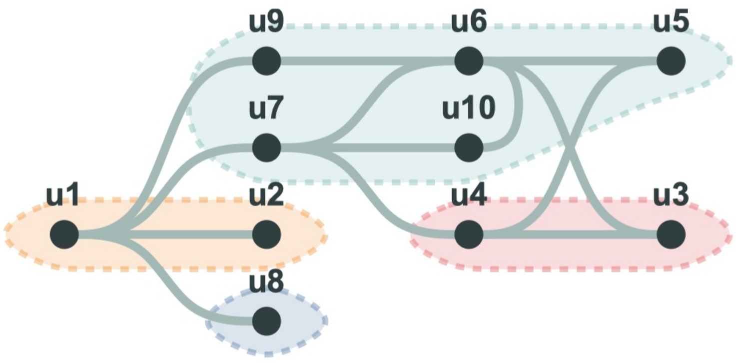
Typical layout algorithms cannot handle hierarchies well

?

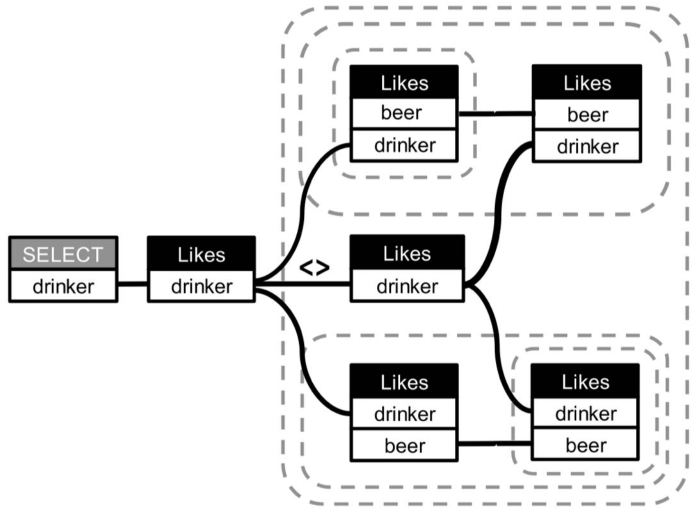
5. "Nice" layouts: Automatic layout algorithms

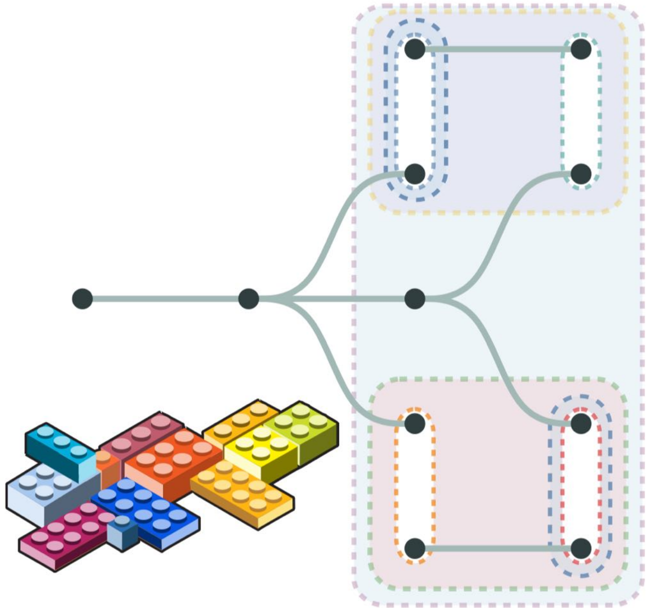


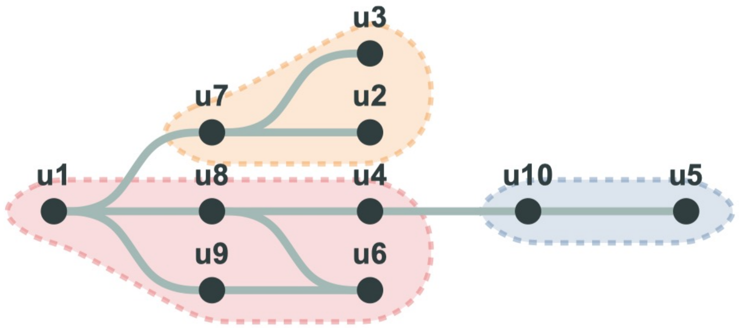
Typical layout algorithms cannot handle hierarchies well



How do we design principled layered node-link visualizations?



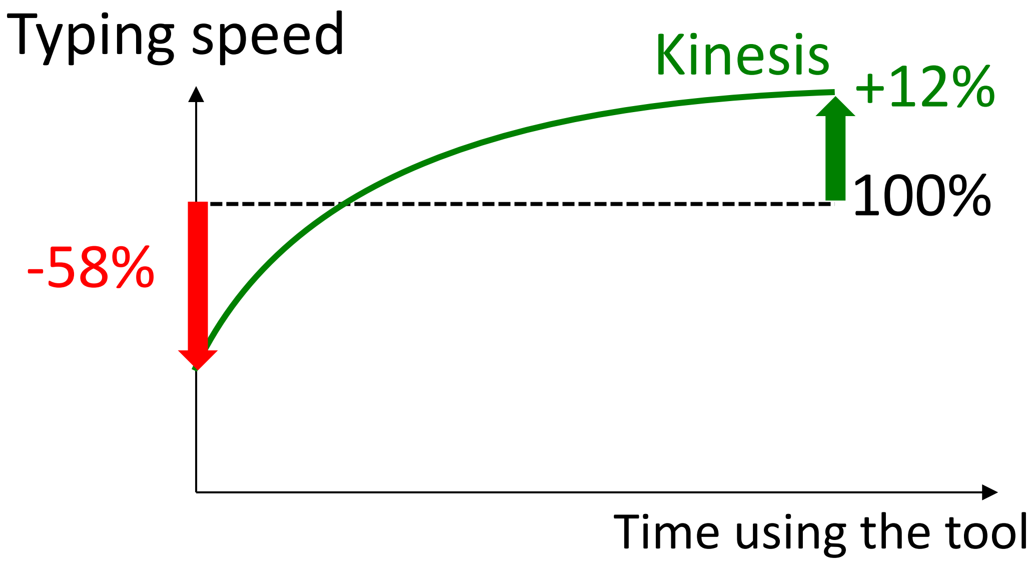




Source: STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations, IEEE TVC 2021, https://doi.org/10.1109/TVCG.2021.3114756
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

6. Principled User Studies (preregistered, beyond students)

How to design principled, reproducible, longitudinal user studies?

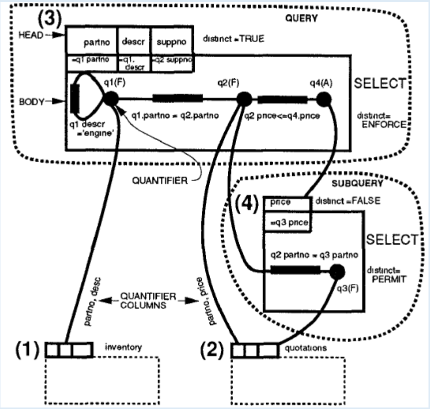


[*] Self-test and test with first-time user (in 2011): 3 repetitions, 2-minute typing test from http://hi-games.net/typing-test/

7. Revisiting P7: syntactic invariance is maybe too strong

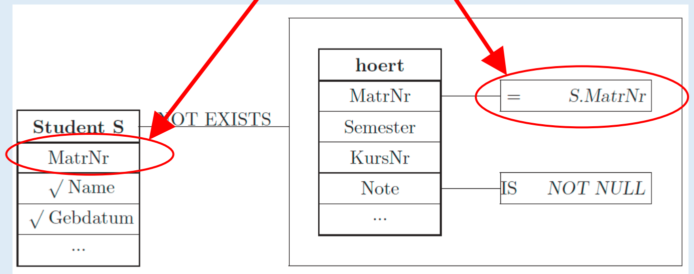
What is the appropriate level of abstraction? (intent vs. debugging)

Correlated nesting is preserved



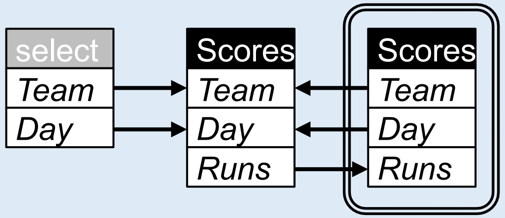
Query Graph Model

Pirahesh et al. [Sigmod'92]



Most VQL* such as Visual SQL

Jaakkola & Thalheim [ER WS'03]



http://queryviz.com

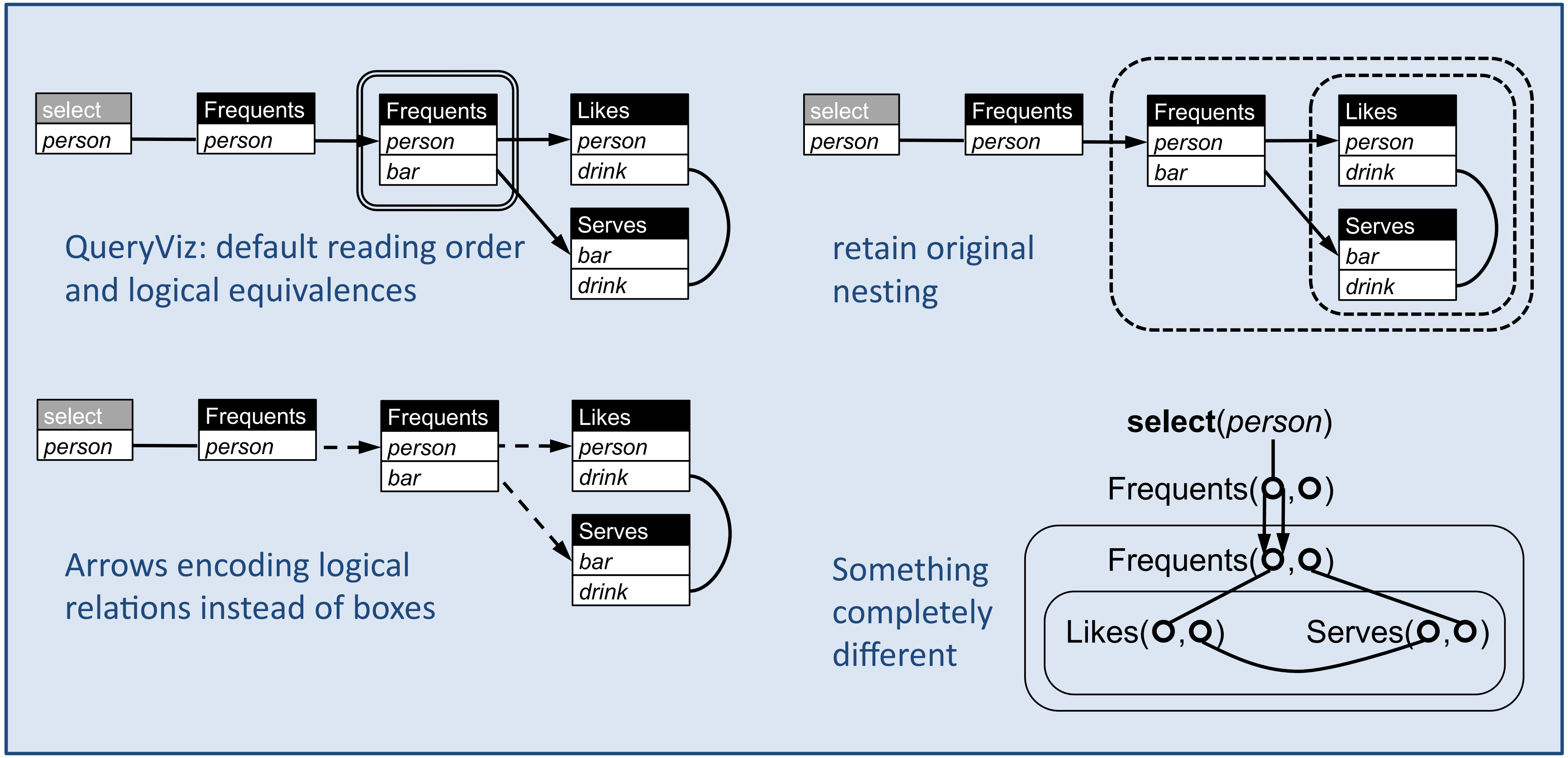
Query Plan **SQL Query** **Query Intent**

more abstract

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

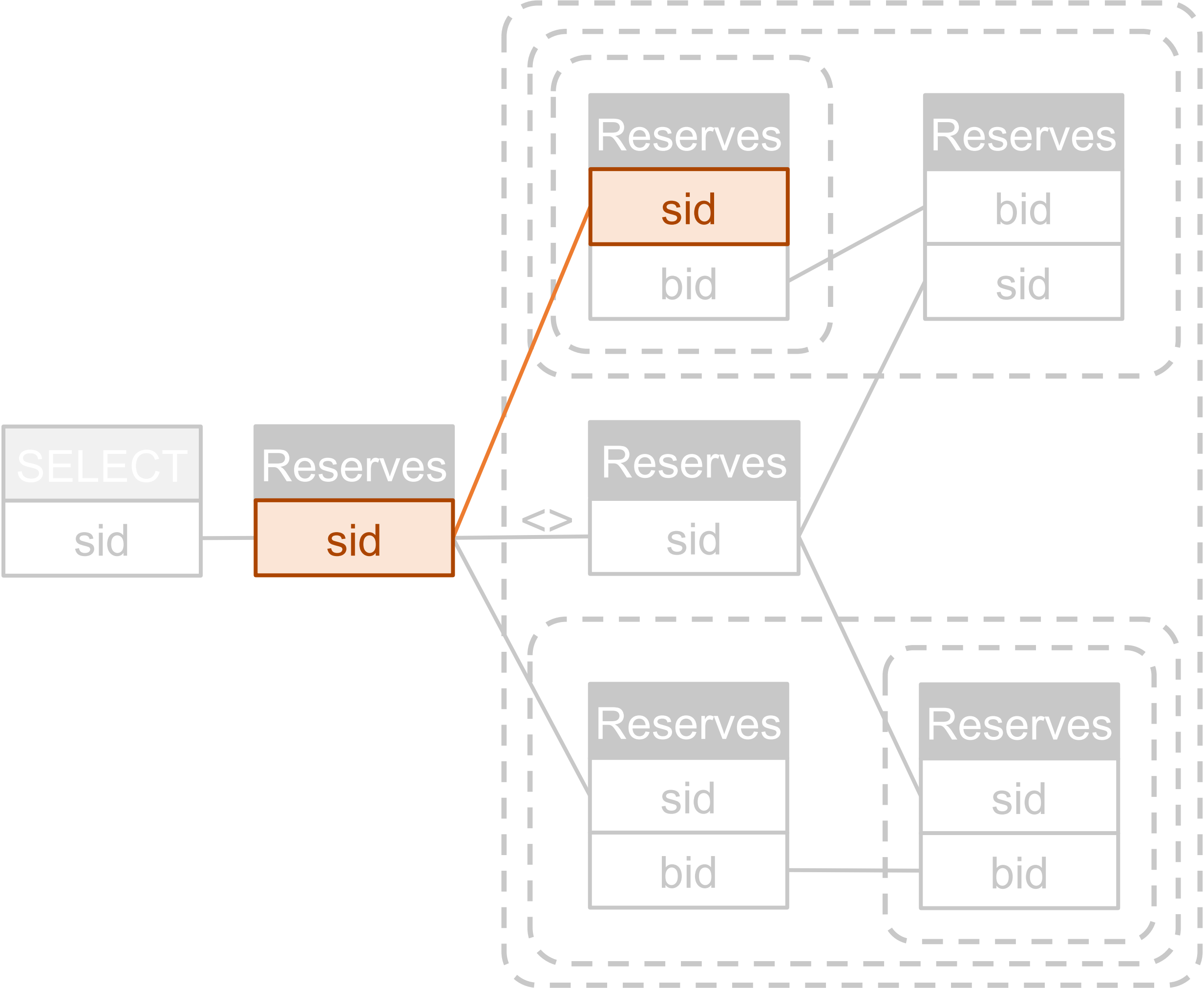
Source: Gatterbauer. Databases will visualize queries too. VLDB 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

8. What are other "better" visual metaphors?



9. What about interaction with diagrams?

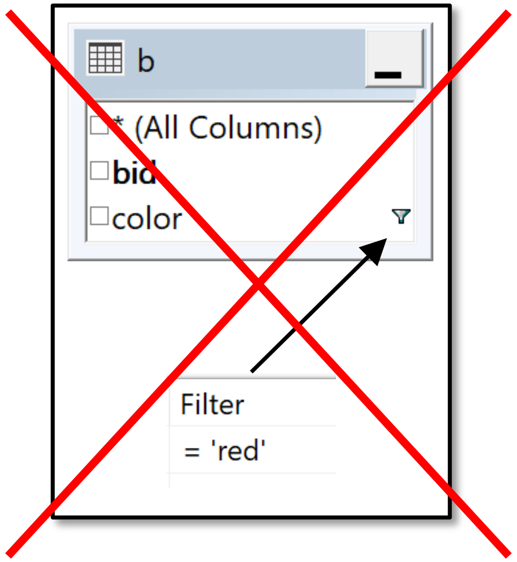
```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))
  and not exists
    (select *
    from Reserves R5
    where R5.sid= R1.sid
    and not exists
      (select *
      from Reserves R6
      where R6.sid = R2.sid
      and R6.bid= R5.bid)))
```

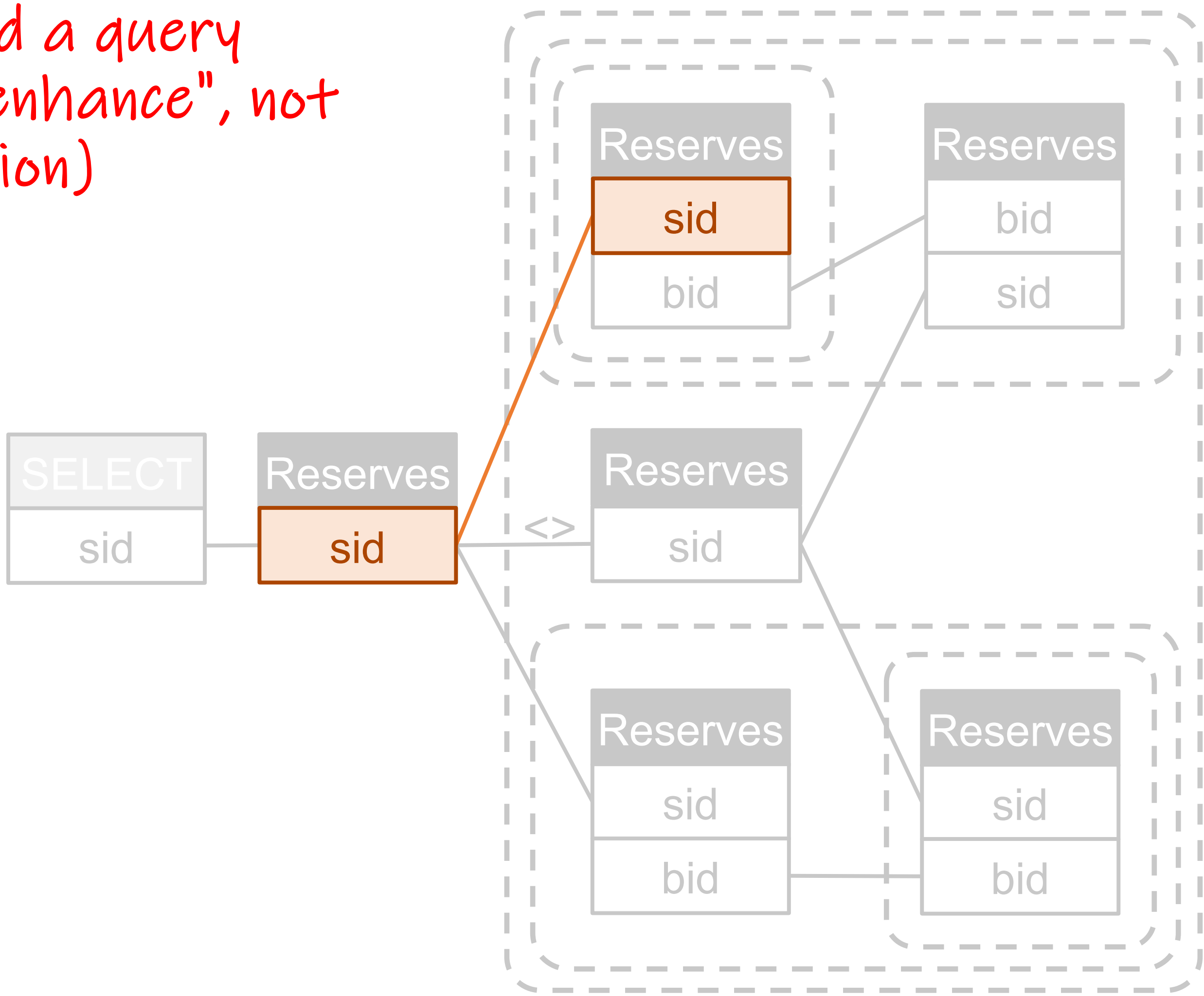


9. What about interaction with diagrams?

```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))
  and not exists
    (select *
    from Reserves R5
    where R5.sid= R1.sid
    and not exists
      (select *
      from Reserves R6
      where R6.sid = R2.sid
      and R6.bid= R5.bid)))
```

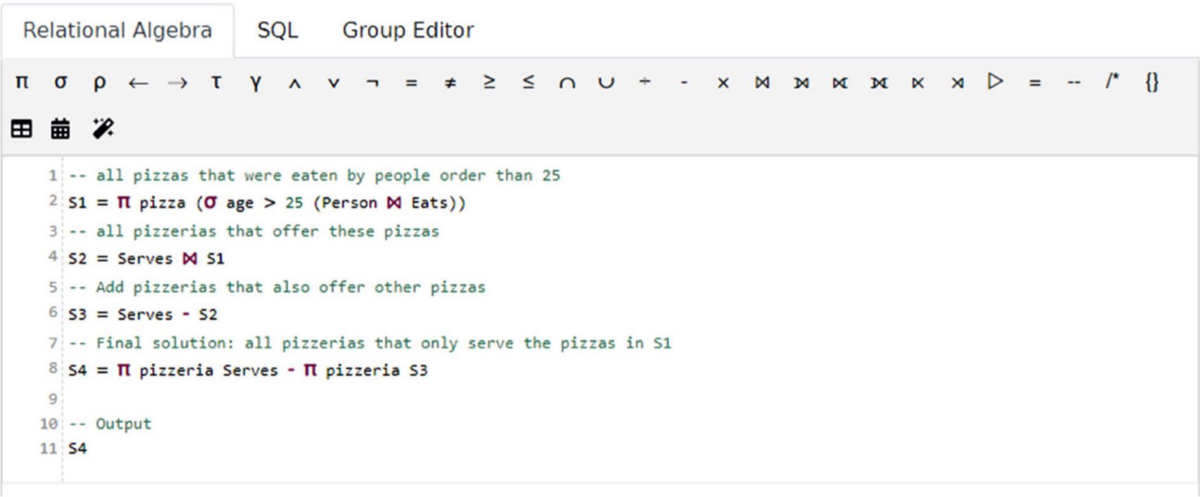
But recall: we don't want interaction <u>required</u> to understand a query (interactions should "enhance", not replace visual perception)





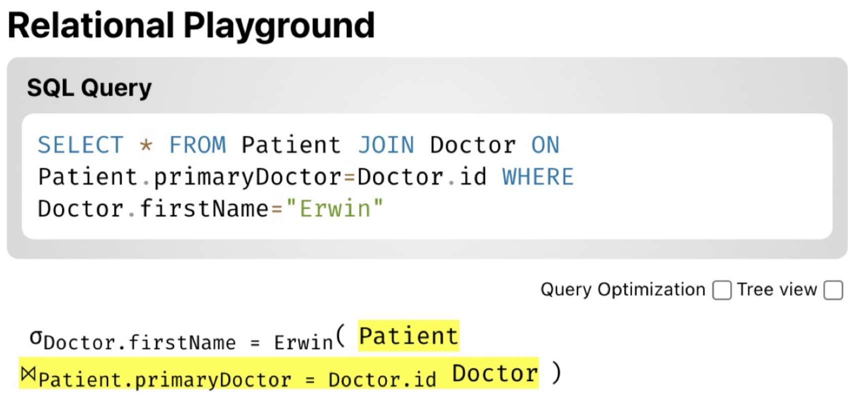
10. An ecosystem of tools that can translate between languages

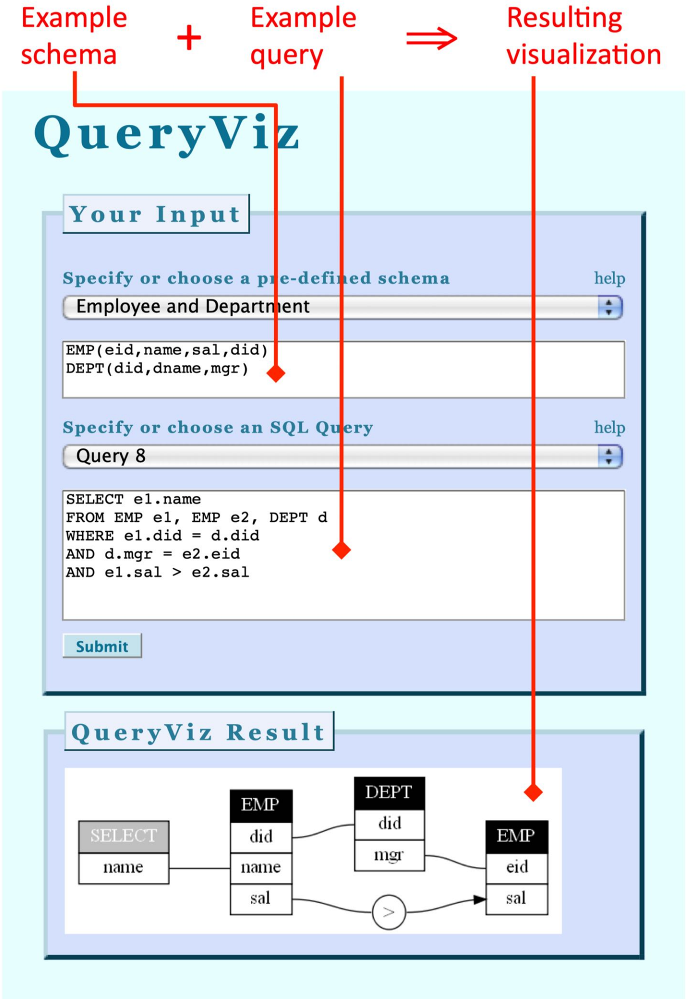
Relax: relational algebra calculator (https://dbis-uibk.github.io/relax/):



Online tools for translating b/w various query languages

Relational Playground by Michael Mior (https://relationalplayground.com/):





QueryVis: http://demo.queryvis.com

Screenshot sources: "Specht, Kessler, Mayerl, Tschuggnall. RelaX -- Interaktive Relationale Algebra in der Lehre. Datenbank Spektrum 2021. https://doi.org/10.1007/s13222-021-00367-x " , "Mior. Relational Playground -- Teaching the Duality of Relational Algebra and SQL, DataEd@Sigmod. 2023. https://doi.org/10.1145/3596673.3596978", Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Some take-aways from today

- Visualizations of Relational Expressions have been investigated for > 100 years
- The inverse functionality of query visualization (≠ VQLs from the 1990s) has not gotten much attention, yet new reasons to revisit
- Many (unresolved) issues lie in the actual details
- Solving those need rigorous and principled approaches

Thanks for your attention
and for leaving feedback ☺

Backup

Excerpts from original literature that were used to derive the illustrations in section 5 of this tutorial

Comparing various approaches from database literature

5. Visual Query Representations for Databases
    1. QBE (1977): Query-By-Example
    2. QBD (1990): Query By Diagram
    3. TableTalk (1991)
    4. OO-VQL (1993): "Object-Oriented" VQL
    5. DFQL (1994): DataFlow QL
    6. Visual SQL (2003)
    7. QueryVis (2011)
    8. Dataplay (2012)
    9. SIEUFERD (2016)
    10. SQLVis (2021)

QBE (1977) (Query-By-Example) Backup

QBE (Query-By-Example)

Retrieval using a negation. Print the departments that sell an item not supplied by the Pencraft Company. This query is shown in Figure 17. Here the not (¬) operator is applied against the entire query expression in the SUPPLY table. This query may be paraphrased as follows. Print department names for items <u>INK</u> such that it is not the case that PENCRAFT supplies <u>INK</u>. In other words, the system is to look for (<u>INK</u>, PENCRAFT) throughout the entire table, and only if it does not find that entry is the corresponding department printed. This query is different from the following one.

Figure 17 Retrieval using a negation

SALES	DEPT	ITEM
	P.	<u>INK</u>

SUPPLY	ITEM	SUPPLIER
¬	<u>INK</u>	PENCRAFT

Retrieval using a negation. Print the departments that sell items supplied by a supplier other than the Pencraft Company. This query is illustrated by Figure 18. Here the system retrieves data in the SUPPLY table with suppliers different from Pencraft, and then retrieves the relevant departments. Note that (<u>INK</u>, PENCRAFT) might also exist.

Figure 18 Retrieval using a negation

SALES	DEPT	ITEM
	P.	<u>INK</u>

SUPPLY	ITEM	SUPPLIER
	<u>INK</u>	¬ PEN CRAFT

QBE (Query-By-Example)

Retrieval of collected output from multiple tables. Print out each department with its corresponding suppliers. Since the output must be a new table, the user must generate a third table skeleton, and fill it in with examples mapped from the two existing tables that satisfy the stipulation of the query. Since it is a user-created table—and, therefore, does not correspond to stored data—the user can fill in the required descriptive headings or leave them blank. This is shown in Figure 19.

Figure 19 Retrieval of collected output from multiple tables

ZZZ	THING	XXX
	P. TOY	P. IBM

SALES	DEPT	ITEM
	TOY	INK

SUPPLY	ITEM	SUPPLIER
	INK	IBM

QBE (Query-By-Example)

Print the names of employees whose salary is between $10000 and $15000, provided it is not $13000, as shown in Figure 22. The use of the same example element JONES in all three rows implies that these three conditions are ANDed on the employee JONES.

Print the names of employees whose salary is either $10000 or $13000 or $16000. This is illustrated in Figure 23. Different example elements are used in each row, so that the three lines express independent queries. The output is the union of the three sets of answers. (In this example, the P.'s alone would have been sufficient.)

Figure 22 Implicit AND operation

EMP	NAME	SAL
	P. JONES	>10000
	JONES	<15000
	JONES	¬ 13000

Figure 23 Implicit OR operation

EMP	NAME	SAL
	P. JONES	10000
	P. LEWIS	13000
	P. HENRY	16000

Figure 24 AND operation using condition box

EMP	NAME	SAL
	P.	S1

CONDITIONS
S1 = (>10000 & <15000 & ¬13000)

Figure 25 OR operation using condition box

EMP	NAME	SAL
	P.	S1

CONDITIONS
S1 = (10000 | 13000 | 16000)

QBE (Query-By-Example)

We can print the names of sailors who do *not* have a reservation by using the ¬ command in the relation name column:

Sailors	*sid*	*sname*	*rating*	*age*
	_Id	P._S		

Reserves	*sid*	*bid*	*day*
¬	_Id		

The use of ¬ in the relation-name column gives us a limited form of the set-difference operator of relational algebra. For example, we can easily modify the previous query to find sailors who are not (both) younger than 30 and rated higher than 4:

Sailors	*sid*	*sname*	*rating*	*age*
	_Id	P._S		

Sailors	*sid*	*sname*	*rating*	*age*
¬	_Id		> 4	< 30

QBE (Query-By-Example)

Q: "Print the names of sailors who are younger than 30 or older than 20."

Sailors	*sid*	*sname*	*rating*	*age*
		P.		< 30
		P.		> 20

Q: "Print the names of sailors who are both younger than 30 and older than 20."

Sailors	*sid*	*sname*	*rating*	*age*
	_Id	P.		< 30
	_Id			> 20

Source: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000). Section 6. https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBE (Query-By-Example)

Find sailors who have reserved all boats.

$$\{\langle I, N, T, A\rangle \mid \langle I, N, T, A\rangle \in Sailors \land \neg\exists\langle B, BN, C\rangle \in Boats$$
$$(\neg\exists\langle Ir, Br, D\rangle \in Reserves(I = Ir \land Br = B))\}$$

One way to achieve such control is to break the query into several parts by using temporary relations or views. As we saw in Chapter 4, we can accomplish division in two logical steps: first, identify *disqualified* candidates, and then remove this set from the set of all candidates. In the query at hand, we have to first identify the set of *sid*s (called, say, BadSids) of sailors who have not reserved some boat (i.e., for each such sailor, we can find a boat not reserved by that sailor), and then we have to remove BadSids from the set of *sid*s of all sailors. This process will identify the set of sailors who've reserved all boats. The view BadSids can be defined as follows:

Sailors	*sid*	*sname*	*rating*	*age*
	_Id			

Reserves	*sid*	*bid*	*day*
¬	_Id	_B	

Boats	*bid*	*bname*	*color*
	_B		

BadSids	*sid*
I.	_Id

Given the view BadSids, it is a simple matter to find sailors whose *sid*s are not in this view.

Source: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000). Section 6. https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBE (Query-By-Example)

(a) **WORKS_ON**

Essn	Pno	Hours
P._ES	1	
P._ES	2	

(b) **WORKS_ON**

Essn	Pno	Hours
P._EX	1	
P._EY	2	

CONDITIONS

_EX = _EY

Figure C.4
Specifying EMPLOYEES who work on both projects. (a) Incorrect specification of an AND condition. (b) Correct specification.

Now consider query Q0C: *List the social security numbers of employees who work on both project 1 and project 2*; this cannot be specified as in Figure C.4(a), which lists those who work on *either* project 1 or project 2. The example variable _ES will bind itself to Essn values in <–, 1, –> tuples *as well as* to those in <–, 2, –> tuples. Figure C.4(b) shows how to specify Q0C correctly, where the condition (_EX = _EY) in the box makes the _EX and _EY variables bind only to identical Essn values.

Source: Elmasri, Navathe. Fundamentals of Database Systems, 7th ed, 2015. Appendix C. https://dl.acm.org/doi/10.5555/2842853
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBE (Query-By-Example)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
P.		P.	_SX						

DEPENDENT

	Essn	Dependent_name	Sex	Bdate	Relationship
¬	_SX				

Figure C.7
Illustrating negation by the query Q6.

Query 6. Retrieve the names of employees who have no dependents.

```
Q6:  SELECT   Fname, Lname
     FROM     EMPLOYEE
     WHERE    NOT EXISTS ( SELECT   *
                           FROM     DEPENDENT
                           WHERE    Ssn = Essn );
```

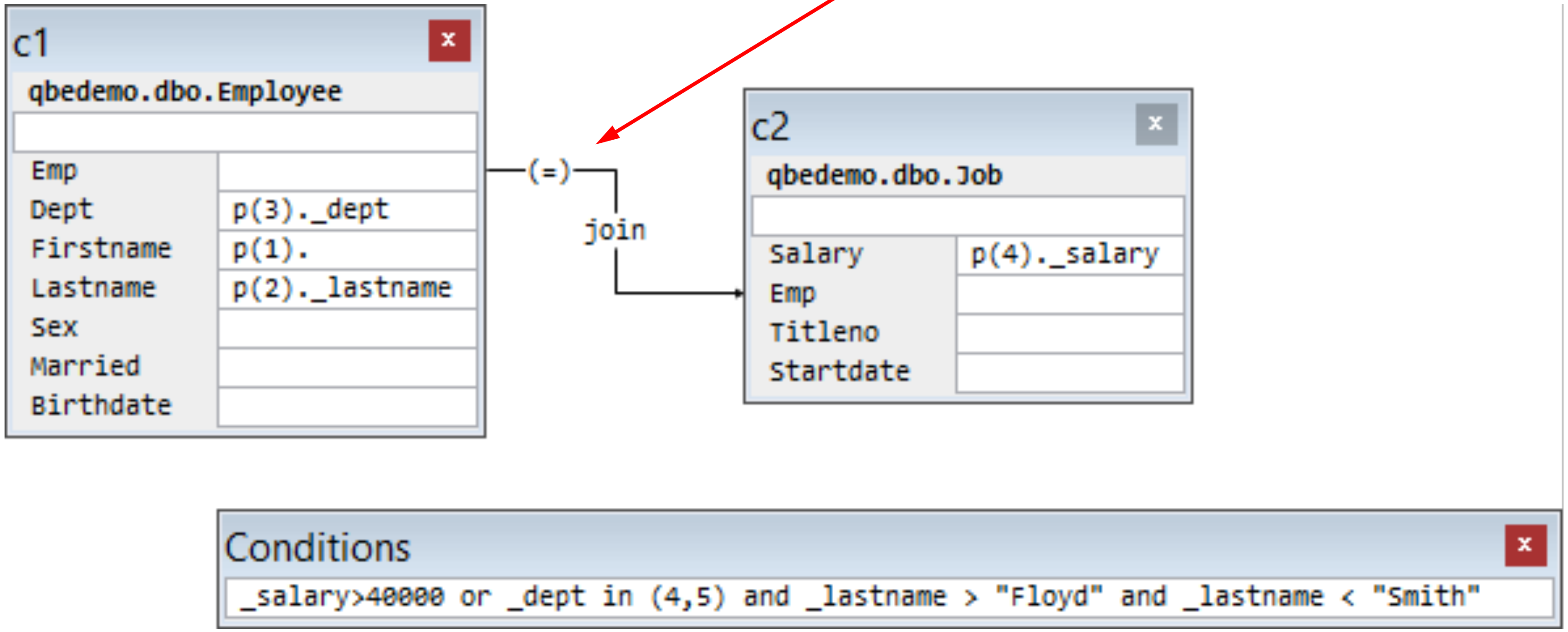
Q6: $\{q, s \mid (\exists t)(\text{EMPLOYEE}(qrstuvwxyz) \textbf{ AND } (\textbf{NOT}(\exists l)(\text{DEPENDENT}(lmnop) \textbf{ AND } t{=}l)))\}$

Source: Elmasri, Navathe. Fundamentals of Database Systems, 7th ed, 2015. Appendix C. https://dl.acm.org/doi/10.5555/2842853
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBE variant: Catalyst One Sysdeco

Q: "List employees who earn more than 40,000. In the same list, we want to include employees who work in departments 4 and 5 and have a surname in the interval between Floyd and Smith, irrespective of how much they earn."

Using a tool called "visual query editor" which combines ideas from QBE with a visual join syntax.



Source: http://www.sysdeco.no/documentation/17/querybuilder/index.html#!WordDocuments/orvisualqueryconditionboxoperator.htm

QBD (1990)
(Query By Diagram)
Backup

QBD (Query-By-Diagram)

At this point the query session can begin. To retrieve the desired pilots the user selects in sequence the entities PILOT and PASSENG(er), creating an effective bridge with the condition PILOT.surname=PASSENG(er).surname. The window mechanism used to put conditions on attributes is shown in Figure 8 (the image comes from a previous version of QBD*: the window mechanism in the Windows environment is under development).

The new relationship, represented by a dotted line and the letters BR, is shown in Figure 9.

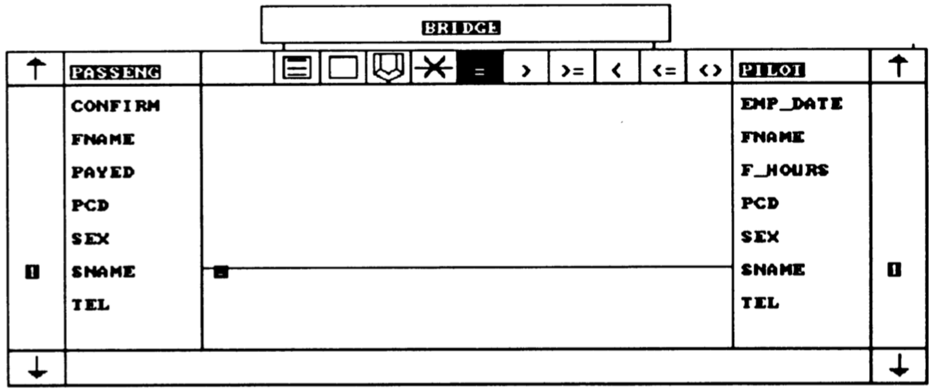


Figure 8. The windows mechanism for the 'bridge'

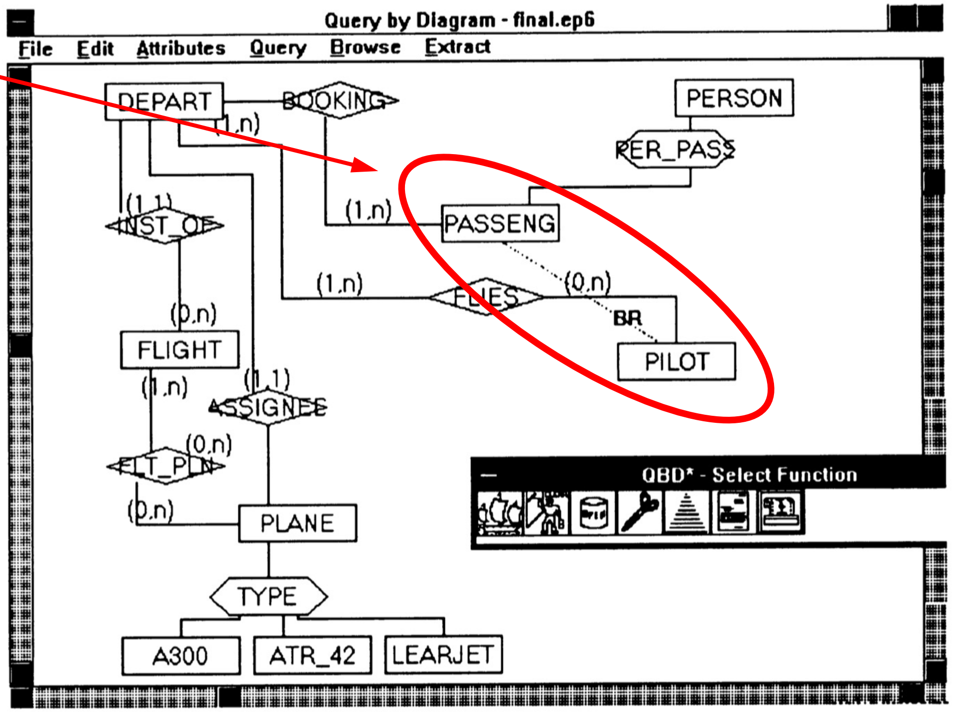


Figure 9. The new relationship

Source: Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QBD (Query-By-Diagram)

Relationship model. For example, in QBD* a query primitive is available that allows joining entities not explicitly linked in the schema [4].

These "bridges" are also necessary for cross joins

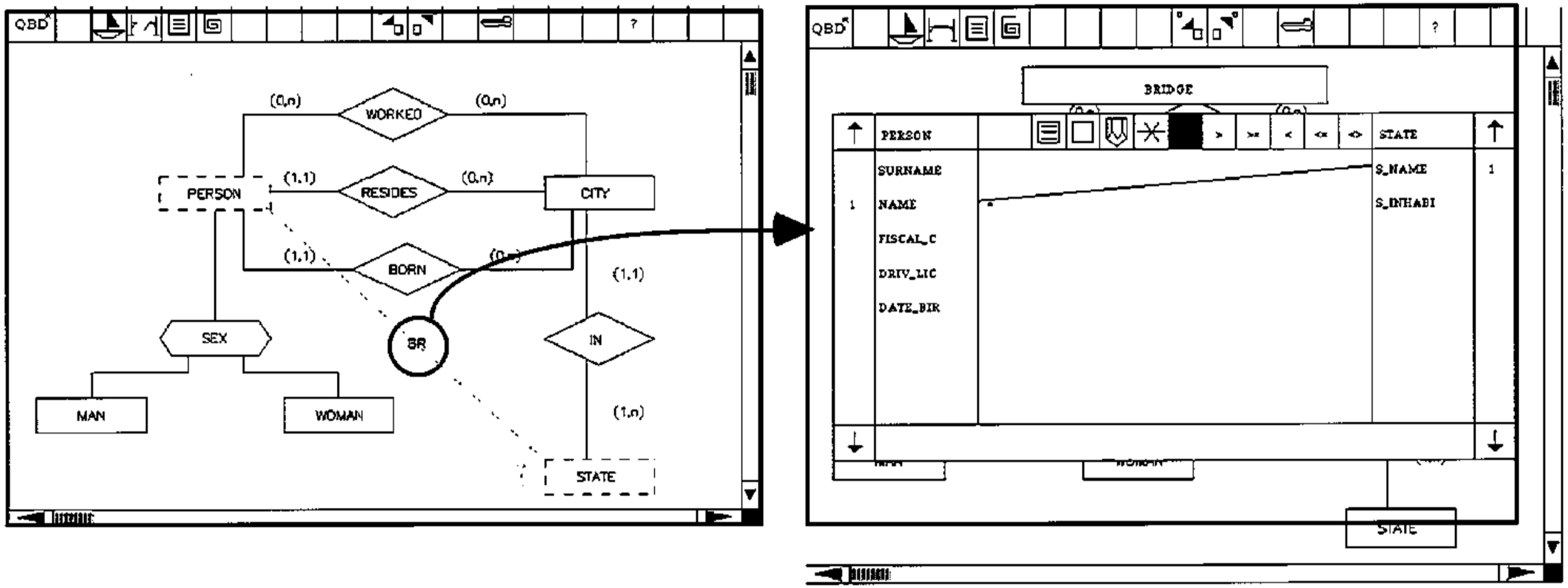


Figure 11. Unconnected path in QBD* (from Angelaccio *et al.* [4])

Source: "Catarci, Costabile, Levialdi, Batini. Visual query systems for databases: a survey. JVLC 1997. https://doi.org/10.1006/jvlc.1997.0037"
citing "Angelaccio, Catarci, Santucci. Query by Diagram*: a fully visual query system. JVLC 1990. https://doi.org/10.1016/S1045-926X(05)80009-6"

TableTalk (1991)
Backup

Tabletalk (1991)

EXAMPLE 1. Give the titles, author names and author photos for all books whose price is over $40·00.

row #:			
1	BOOKS		
2	PRICE > 40·0		
3	TITLE	AUTHOR	
		NAME	PHOTO
	AN INTRODUCTION TO DATABASE SYSTEMS	DATE, CHRIS	
	THE ART OF HUMAN-COMPUTER INTERFACE DESIGN	LAUREL, BRENDA	

Source: "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

Tabletalk (1991)

EXAMPLE 2. Give the book numbers, titles, order numbers, and customers for those orders for books published by Addison-Wesley that are out of stock.

BOOKS			
and	IN_STOCK = 0		
	NAME of PUBLISHER = "ADDISON-WESLEY"		
		ORDERS	
BOOK#	TITLE	ORDER#	CUSTOMER
B5	A GUIDE TO DB2	ORD5	C5
		ORD6	C2
B6	AN INTRODUCTION TO DATABASE SYSTEMS	ORD6	C5

Source: "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"

"Object-oriented VQL"
(1993)
Backup

OO-VQL (Object-oriented VQL)

InletNeedle (idNumber, diameter, length, weight)

Non-graphical variables
for returned predicates

Q2: Find the idNumbers of all the InletNeedles that have diameter greater than 0.25 and have length greater than 12.5.

$$x1\text{: (InletNeedle, } (x1,\ > 0.25,\ > 12.5,\ \_))$$

Q3: Find the idNumbers of all the InletNeedles that have either diameter greater than 0.25 or have length greater than 12.5.

$$x1\text{: (InletNeedle, } (x1,\ > 0.25,\ \_,\ \_)) \vee \text{(InletNeedle, } (x1,\ \_,\ 12.5,\ \_))$$

Disjunction

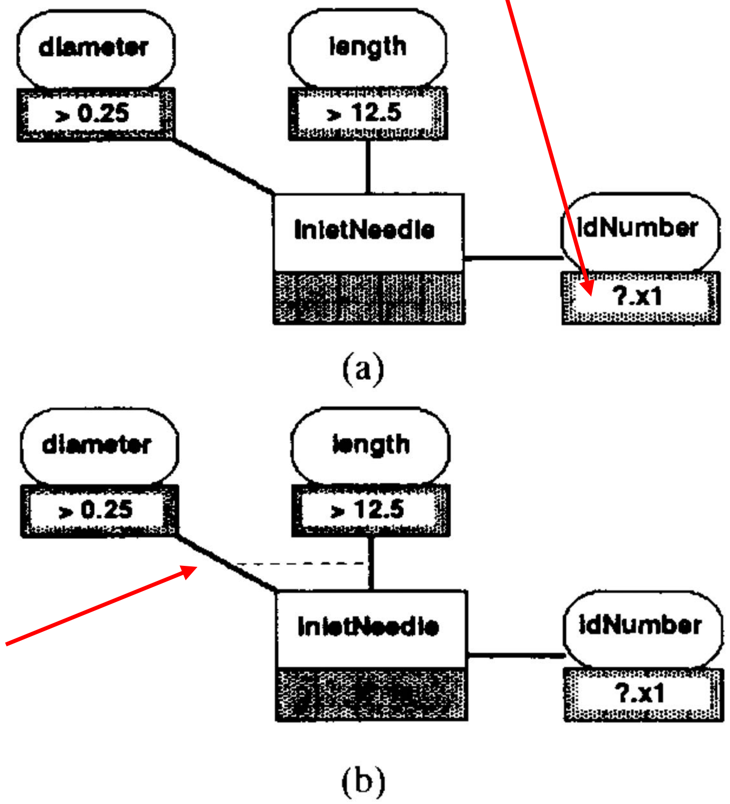


Fig. 6. Conjunctive and disjunctive queries.

OO-VQL (Object-oriented VQL)

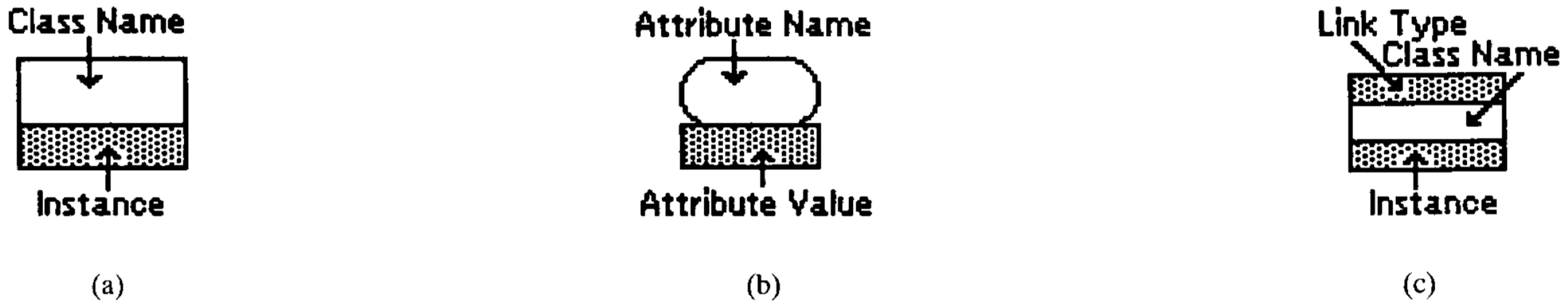


Fig. 3. Visual query primitives.

Figures from: Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

OO-VQL (Object-oriented VQL)

InletNeedle (idNumber, diameter, length, weight)
Manufacturer (name, streetAddress, city)
MadeBy [InletNeedle.idNumber, Manufacturer.name]

Q4: Find the names of those manufacturers who manufacture InletNeedles with diameters greater than 0.25.

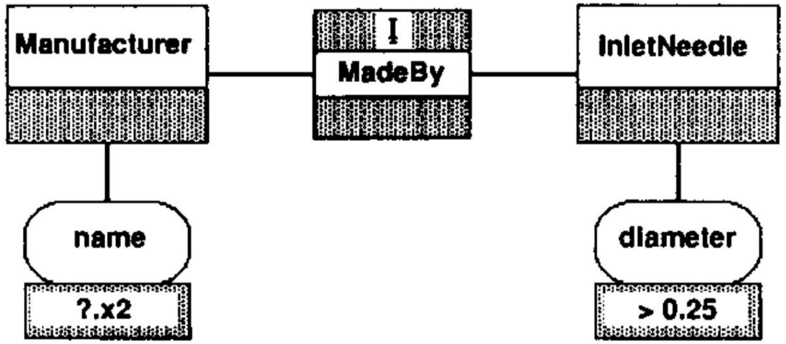


Fig. 7. An associative "Join" query using the "Interaction" link.

$$x2\text{: (I, MadeBy, } (x1, x2))$$
$$[(\text{InletNeedle}, (x1, > 0.25, \_, \_)),$$
$$(\text{Manufacturer}, (x2, \_, \_))]$$

Figures from: Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

OO-VQL (Object-oriented VQL)

Department [name, size, location]
Sells [Department.name, item.name]
Item [name, color, size, weight]
Supplies [Supplier.name, item.name]
Supplier [name, street, city, zip]

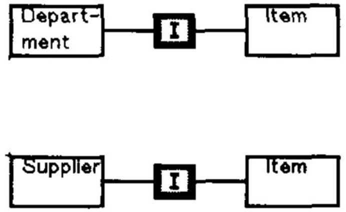


Fig. 8. Portions of two separate schemas.

Q5: Find the suppliers that supply an item sold by the TOY department.

Not clear why the same item object cannot be used for both relationships

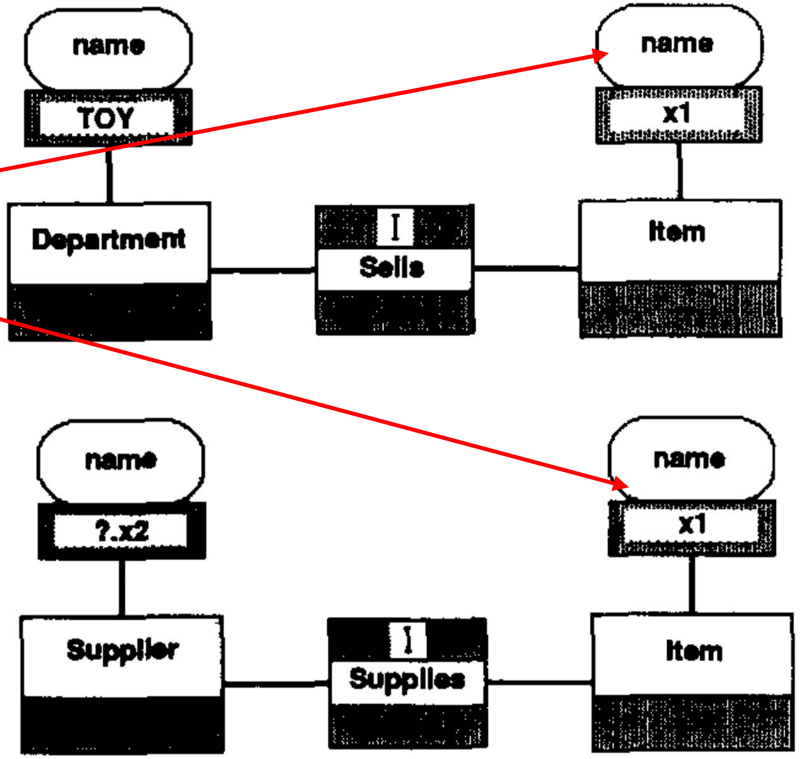


Fig. 9. A "Join" query that links unconnected schemas.

$$x2\text{: } (\text{I, Sells}, (\_, x1))[(\text{Department}, (\text{Toy}, \_, \_)), (\text{Item}, \_)] \vee (\text{I, Supplies}, (\_, x1))[(\text{Supplier}, (x2, \_, \_, \_)), (\mathit{Item}, \_)]$$

Disjunction here seems to be an error

Figures from: Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

OO-VQL (Object-oriented VQL)

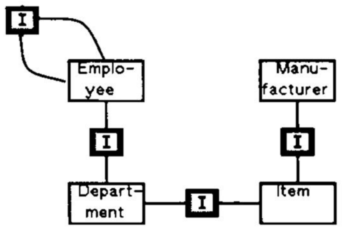


Fig. 10. A department store schema as represented in SSONET.

Q6: Find the names of those employees that earn more than their managers.

Non-graphical variables for bult-in predicates

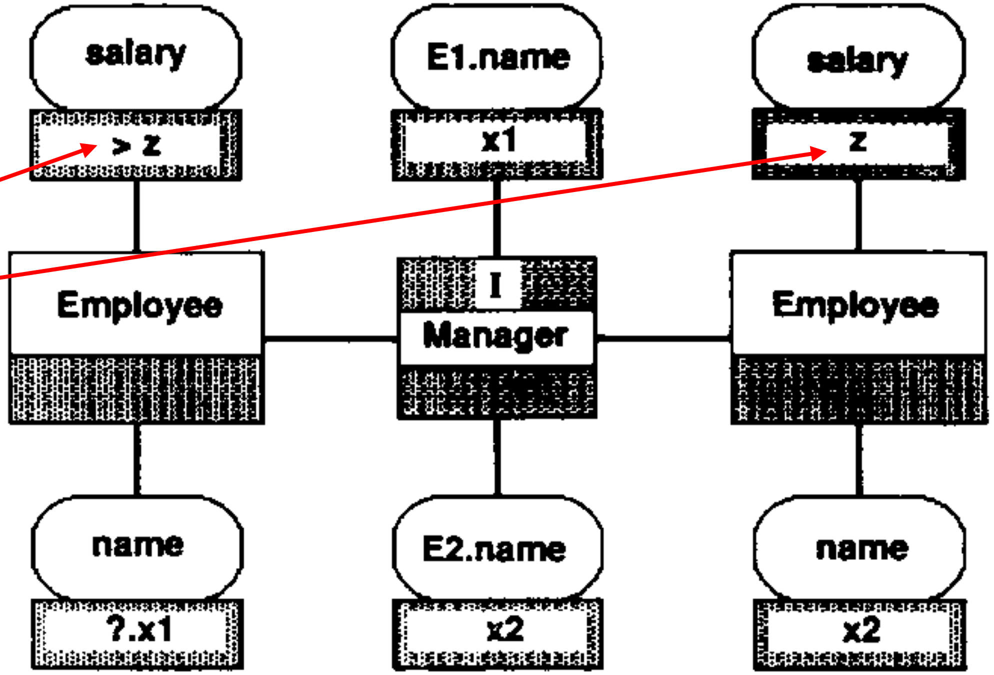


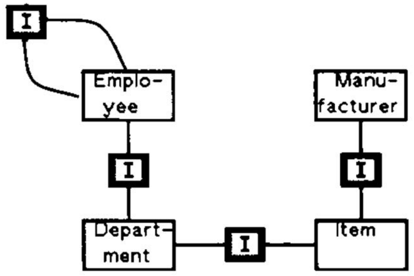
Fig. 11. A query involving a reflexive relationship.

$x1$: (I, ManagedBy, $(x1, x2)$)

[(Employee, $(x1, > z, \_, \_)$), (Employee, $(x2, z, \_, \_)$)]

OO-VQL (Object-oriented VQL)

Department [name, size, location]
Item [name, color, size, weight]
Manufacturer [name, street, city, zip]



MadeBy(mname, ipname)
Sells(dname, iname)

Q7: Find those departments that sell all the items that are manufactured by the manufacturer "Ping"

This expression seems unsafe and would likely be written in today's notation as follows:

{(x) | Department(x,_,_) ∧
(∀v[MadeBy(v, 'Ping') →
(Sells(x,v)))}

It is not evident how this diagram distinguishes between "that sells all items by Ping" vs. "s.t. that all sold items are by Ping"

$$x1\text{: } \forall y(\mathrm{I},\ \mathrm{MadeBy},\ (y,\ \mathrm{Ping}))[(\mathrm{Item},\ \_),\ (\mathrm{Manufacturer},\ \_)] \rightarrow (\mathrm{I},\ \mathrm{Sells},\ (x1,\ y))[(\mathrm{Department},\ \_),\ (\mathrm{Item},\ \_)]$$

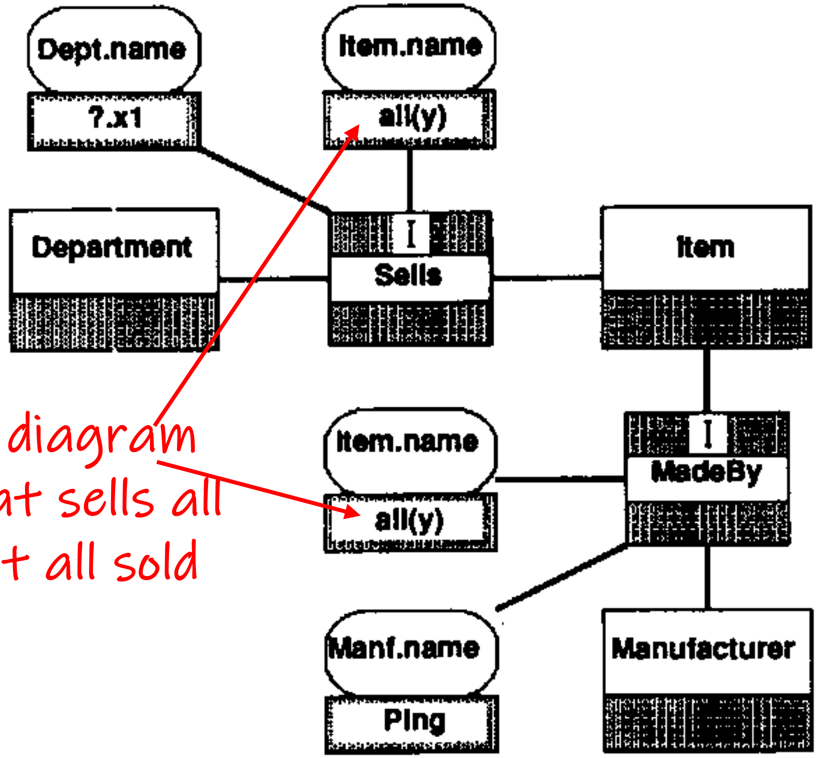


Fig. 12. A query that uses the "all" operator.

DFQL (1994)
DataFlow QL
BACKUP

DFQL (DataFlow Query Language)

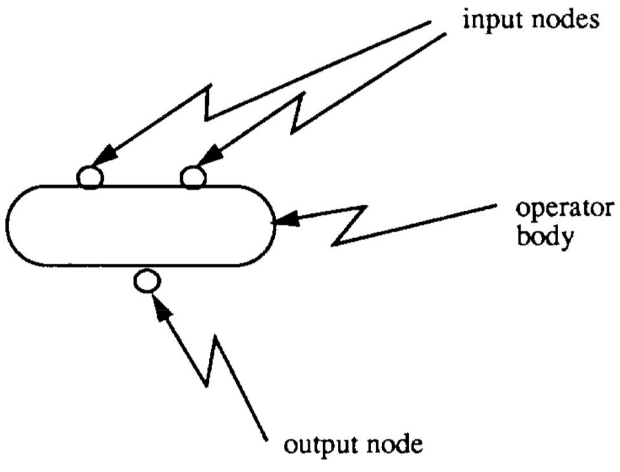


Fig. 3.

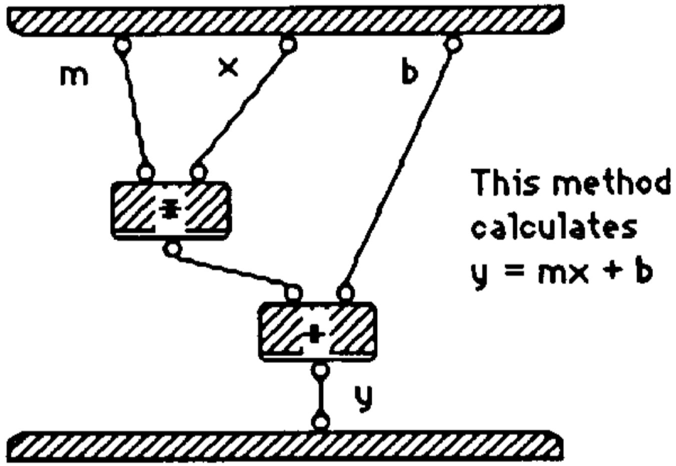


Fig. 2.

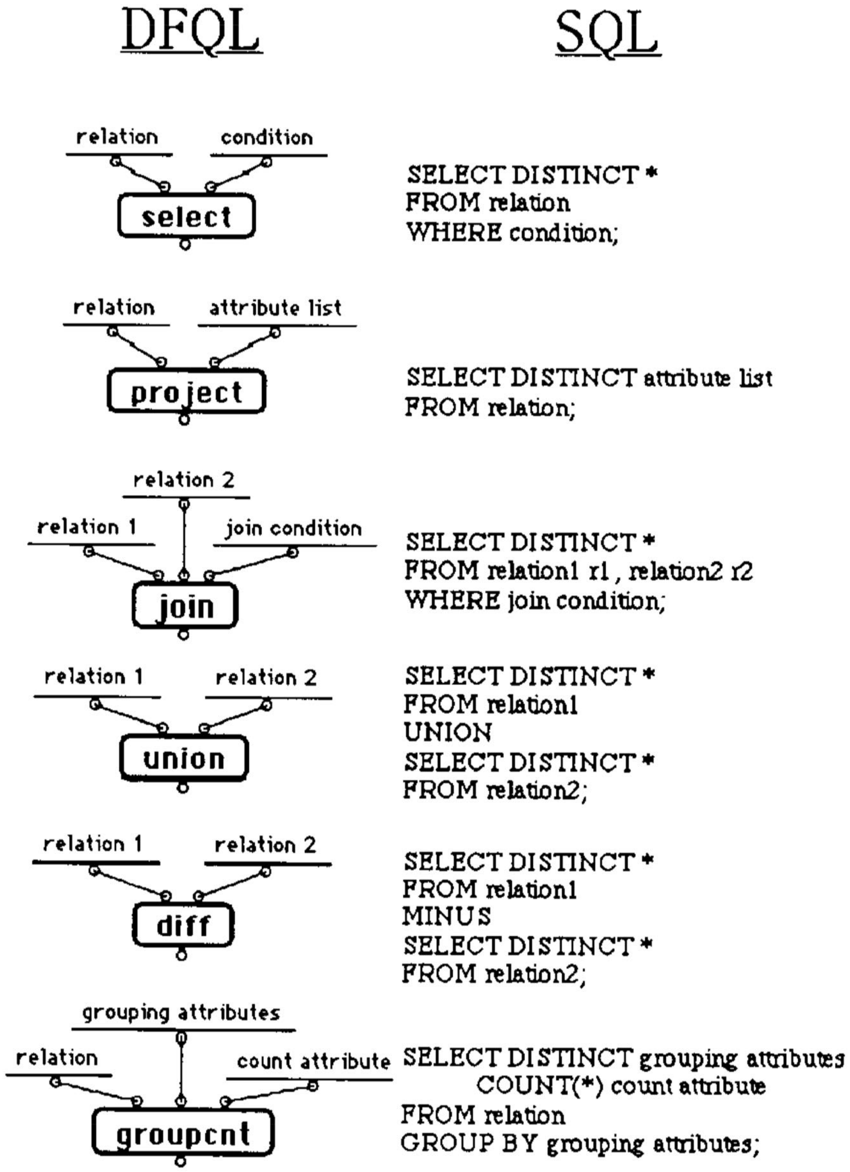


Fig. 4.

DFQL (DataFlow Query Language)

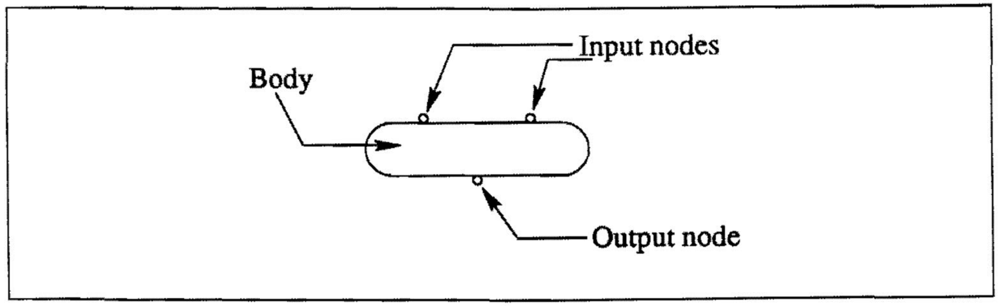
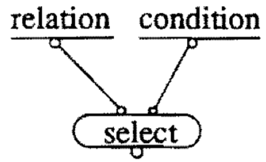
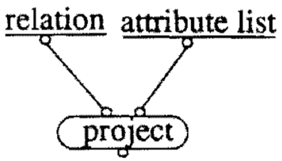


Figure 2.2: Operator Construction

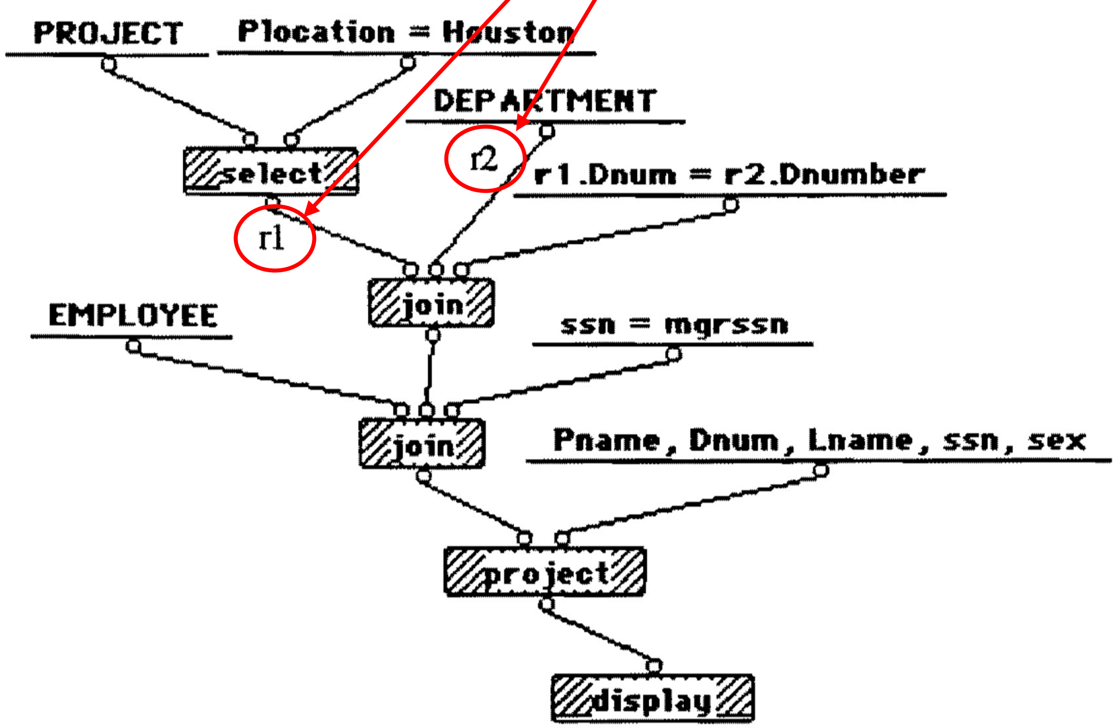
TABLE 2.1: BASIC DFQL OPERATORS AND THEIR SQL EQUIVALENTS

SQL	**Description**	**SQL Equivalent**
 **SELECT**	Implements the relational algebra *selection* operator. The algebraic notation is: $\sigma_{<condition>}(<relation>)$. It retrieves tuples from the relation which fits the specified condition. There are no duplicate tuples in the result.	**SELECT DISTINCT *** **FROM** relation **WHERE** condition
 **PROJECT**	Implements the relational algebra *projection* operator. The algebraic notation is: $\pi_{<attribute\ list>}(<relation>)$. The attributes list, separated by commas contains the names of attributes to be retrieved from the relation . The *project* operator eliminates duplicate tuples from the result.	**SELECT DISTINCT** attribute list **FROM** relation

DFQL (DataFlow Query Language)

Query 3: "For every project located in Houston, list the project name, the controlling department number, and department manager's last name, ssn, and sex."

alias on edges to avoid ambiguities



SELECT PNAME, DNUM, LNAME, SSN, SEX

FROM EMPLOYEE, DEPARTMENT, PROJECT

WHERE MGRSSN = SSN **AND** DNUM = DNUMBER

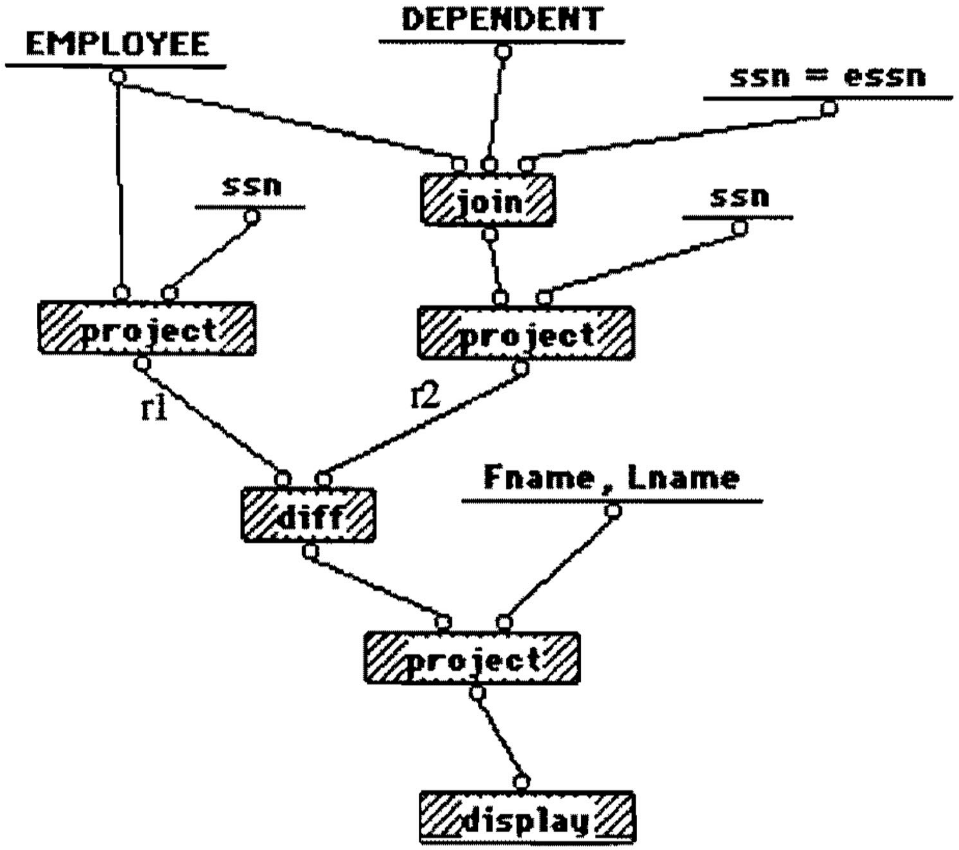
AND PLOCATION = 'Houston'

Girsang. The comparison of SQL, QBE, and DFQL as query languages for relational databases, Master thesis, Naval Postgraduate School, 1994. https://core.ac.uk/download/pdf/36723678.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

Query 5: "For each department retrieve the first names and the last names of employees who have no dependents."

```
SELECT   DNO FNAME, LNAME
FROM     EMPLOYEE
WHERE    NOT EXISTS (SELECT *
                     FROM    DEPENDENT
                     WHER    SSN = ESSN)
GROUP BY DNO
```

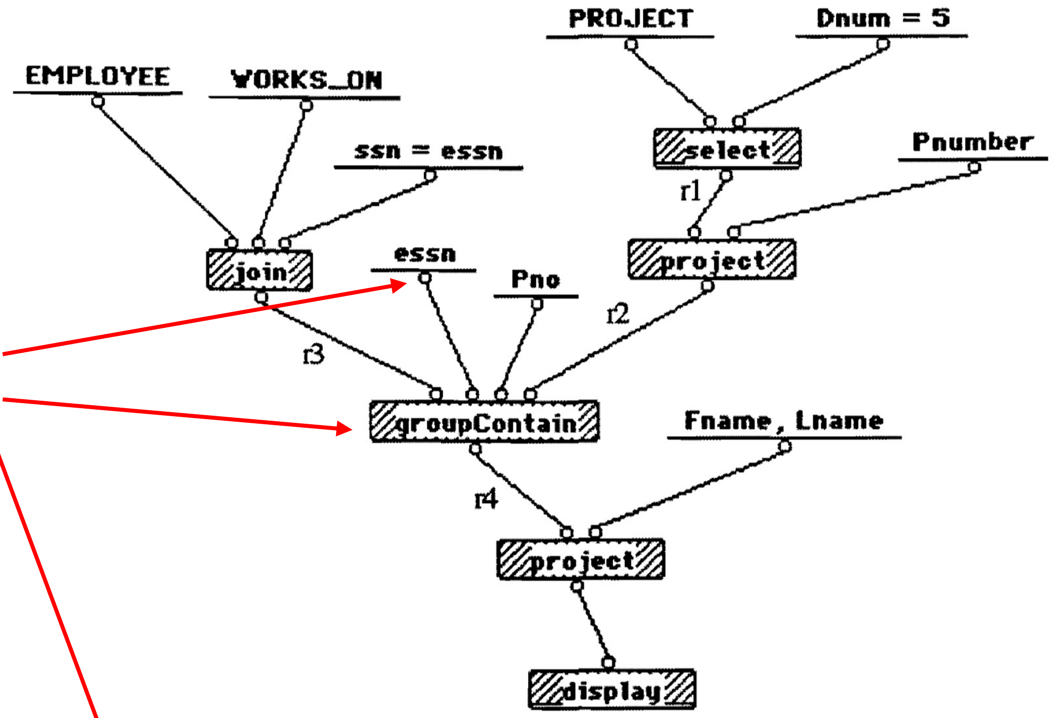


Girsang. The comparison of SQL, QBE, and DFQL as query languages for relational databases, Master thesis, Naval Postgraduate School, 1994. https://core.ac.uk/download/pdf/36723678.pdf
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

DFQL (DataFlow Query Language)

Query 9: "Retrieve the first name and last name of each employee who works on all the projects managed by department number 5."

Based on the description of the operator, there seems to be an error in the visualization: the "groupContain" only returns the essn's, and a subsequent join with Employee is necessary




```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    NOT EXISTS
          (SELECT *
           FROM    WORKS_ON B
           WHERE   (B.PNO IN (SELECT  PNUMBER
                              FROM    PROJECT
                              WHERE   DNUM = 5))
               AND
               NOT EXIST (SELECT *
                          FROM    WORKS_ON C
                          WHERE   C. ESSN = SSN
                              AND C. PNO = B. PNO))
```

```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE (SELECT  PNO
       FROM    WORKS_ON
       WHERE  SSN = ESSN)
       CONTAINS
       (SELECT  PNUMBER
        FROM    PROJECT
        WHERE  DNUM = '5')
```

The *groupContain* operator takes the WORKS_ON relation (r1) and the second relation (r2) and groups the tuples according to the grouping attribute essn. It then compares attribute Pno to see if one essn has all the Pno values contained in r2. If so, the essn is selected.

1. *GroupContain* operator is a part of *Group Set Comparation*. *GroupSet Comparation* also provides *GroupEqual* and *GroupContainBy* operators. These operators are discussed in class notes of Dr. C. Thomas Wu, Computer Science Department, Naval Postgraduate School, Monterey, CA.

Notice that the CONTAINS comparison operator in this query is similar in function to the DIVISION operation of the relational algebra [Elma 89].

Visual SQL (2003)
BACKUP

Visual SQL

Q: "Return ("successless") students who no enrollment with null as result."

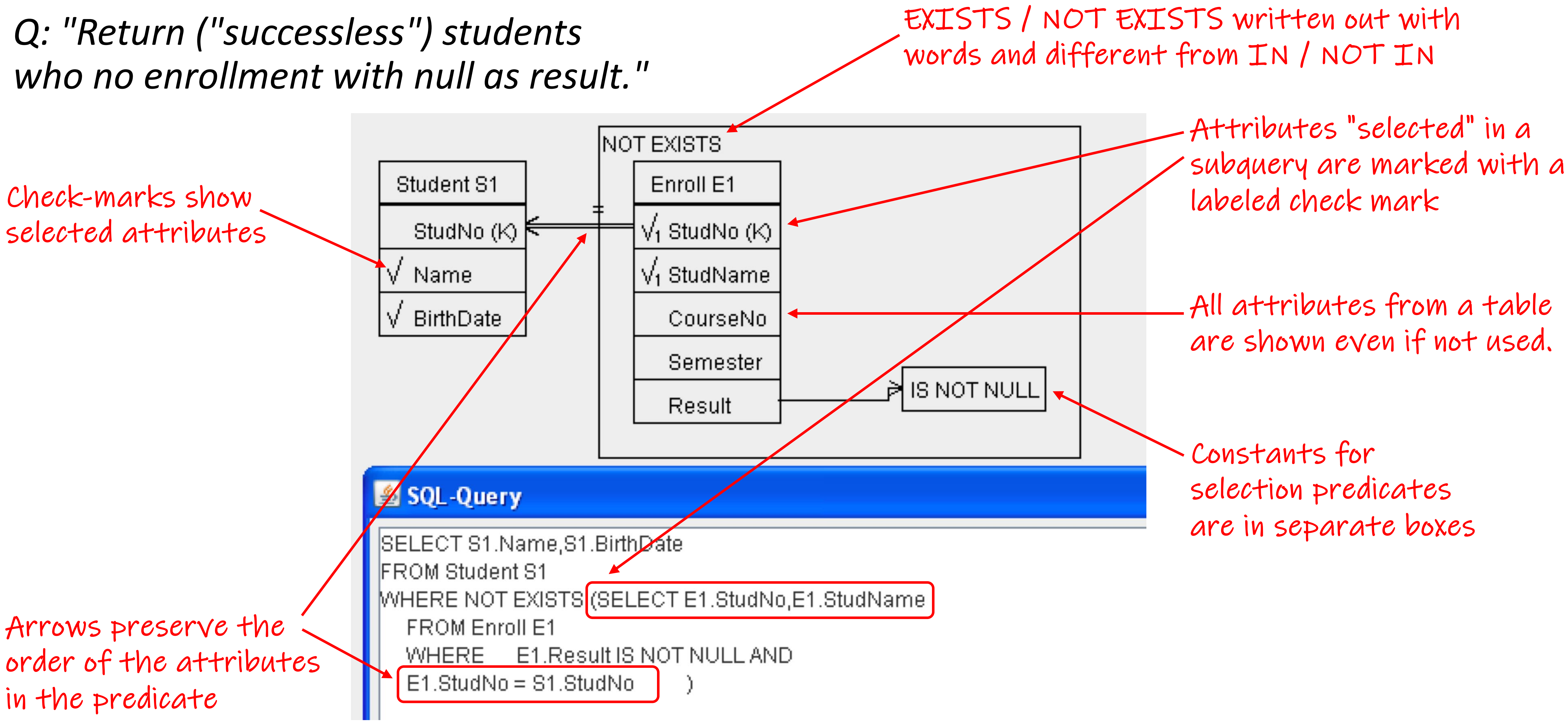


Figure from "Thalheim. Visual SQL as the Alternative to Linear SQL, Talk slides, 2013."

Visual SQL

Provide data on students who have successfully completed those and only those courses which have successfully been given or which are currently given by the student's supervisor?

```
SELECT  P1.Name, P1.BirthData, P1. Address,
        P2.Name AS "Name of supervisor"
  FROM  Person P1, Professor P2, Student S1,  Supervisor, Lecture L,
        Enroll E
 WHERE  P1.Name = Student.Name AND P1.BirthData = Student.BirthData
        AND S1.StudNo = E.StudNo
        AND E.Result NOT NULL
        AND S1.StudNo = Supervisor.StudNo
        AND Supervisor.Name = Professor.Name
        AND Supervisor.BirthData = Professor.BirthData
        AND P2.Name = Professor.Name AND P2.BirthData = P2.BirthData
        AND L.Name = Professor.Name AND L.BirthData = Professor.BirthData
        AND
         L.CourseNo
         IN
            (SELECT E2.CourseNo
              FROM  Enroll E2
              WHERE
                 S1.StudNo = E2.StudNo  AND
                 E2.Result NOT NULL
            )
        AND
         E.CourseNo
         IN
            (SELECT  L2.CourseNo
              FROM  Lecture L2
              WHERE
                 L2.Name = P2.Name AND
                 L2.BirthData = P2.BirthData
            );
```

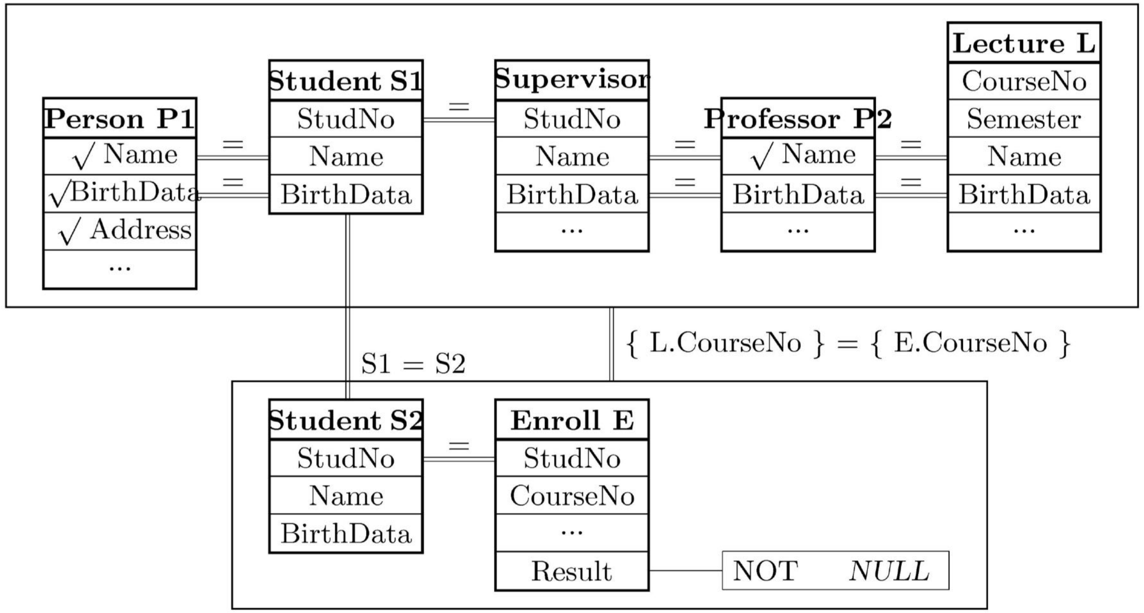


Fig. 1. Visual SQL Involving Equality On Two Visual SQL Subqueries

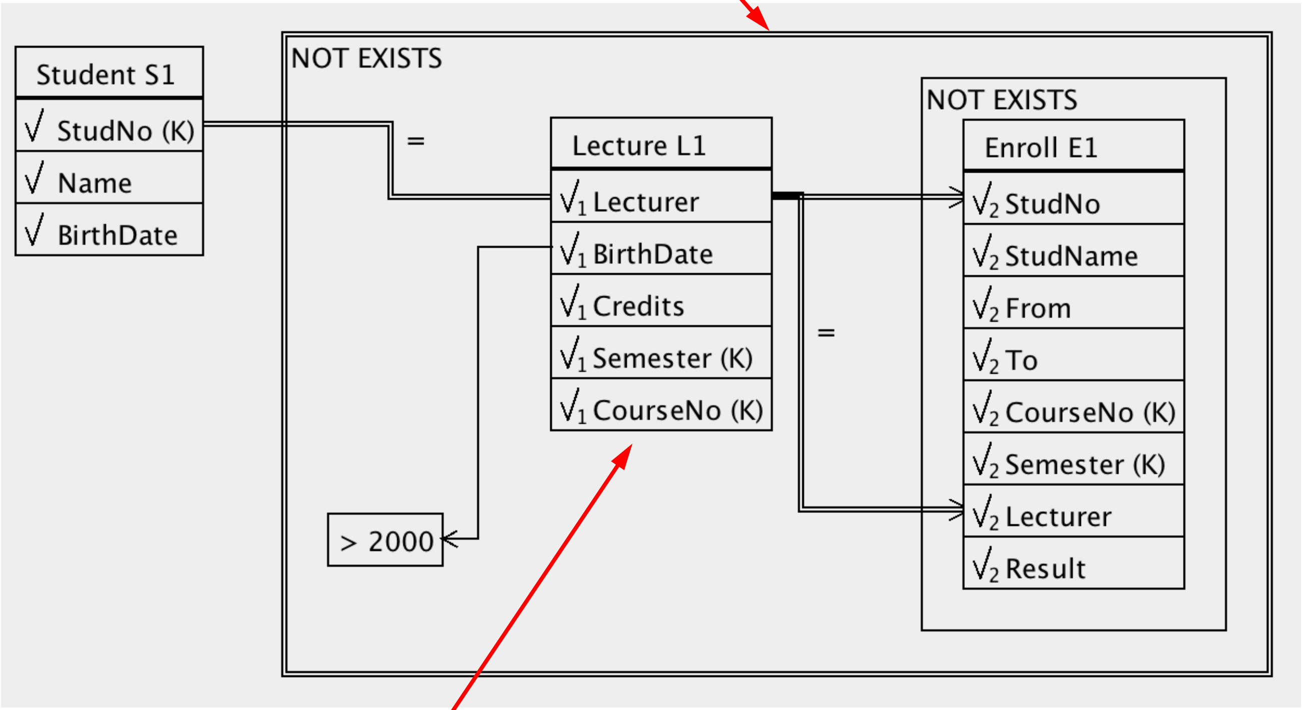
The visualization contains fewer tables and the translation between the visualization and SQL is not explained in the paper.

Source: Jaakkola, Thalheim. Visual SQL -- high-quality ER-based query treatment. ER workshops, 2003. https://doi.org/10.1007/978-3-540-39597-3_13
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

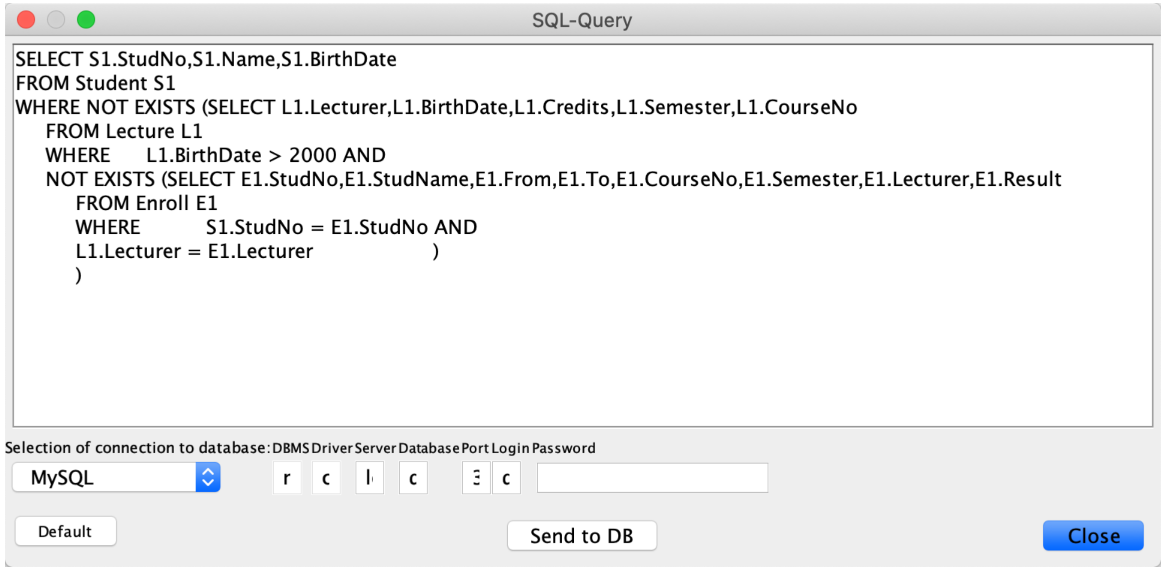
Visual SQL

Q: "Find students who have taken classes from all lecturer who were born after 2000"

double-line is assumed a bug and not part of the motivation



outline of L1 is overlapping with the connection between S1 and E1 and is better moved to the right or below)



Figures drawn by Jiahui Zhang around 4/2020 with a code at the time still available on the website of University of Kiel
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Subqueries connected other than "(not) exists" are connected at the whole nesting level of the subquery.

This line likely should start at S1.MatrNr. to mean:

... and S1.MatrNR not in
(select S2.Matr.Nr from...

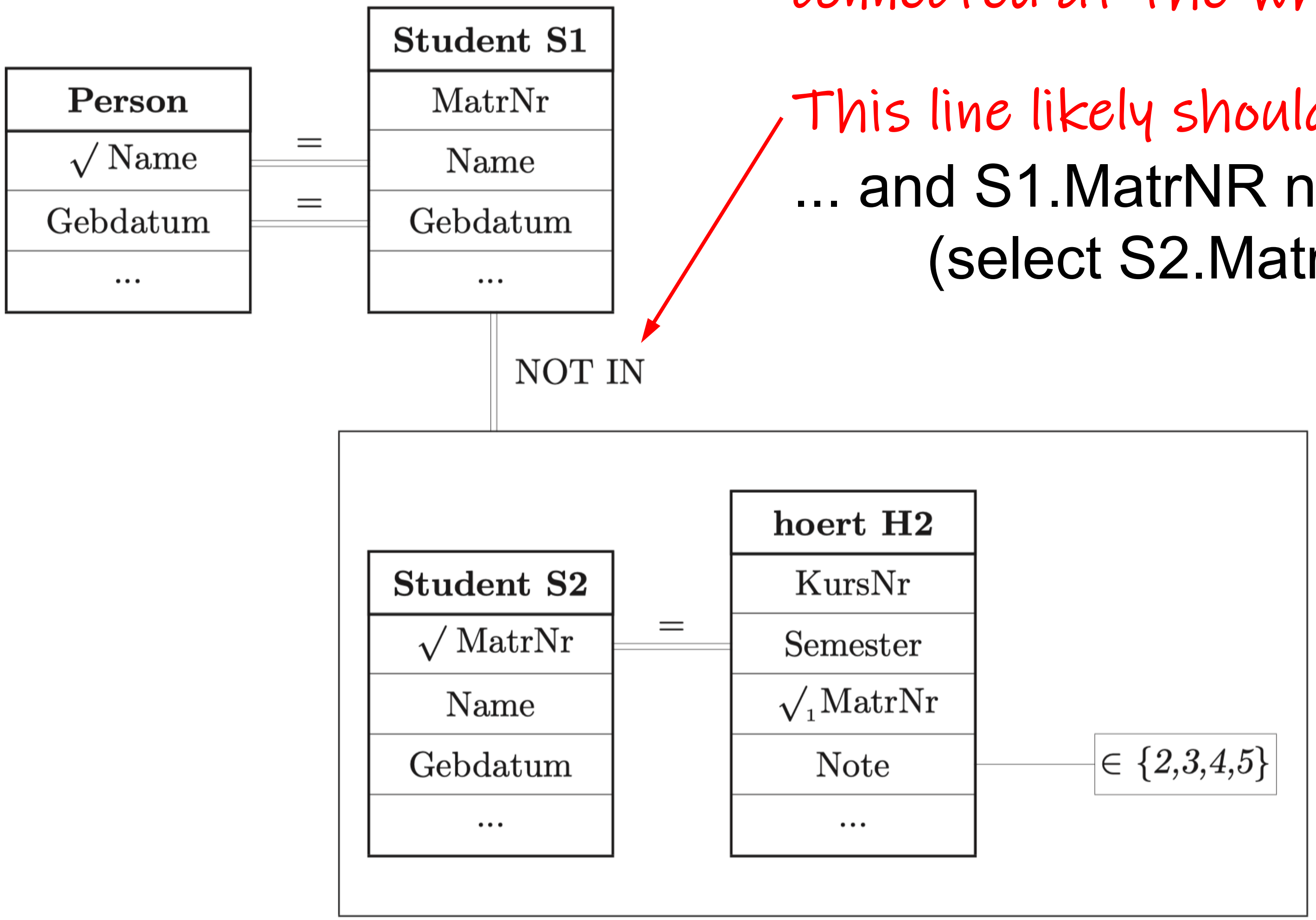


Bild 40: Teilanfrage mit Differenzbildung

Source: Figure 40 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Complicated Boolean expressions are hard to visualize (by nature)

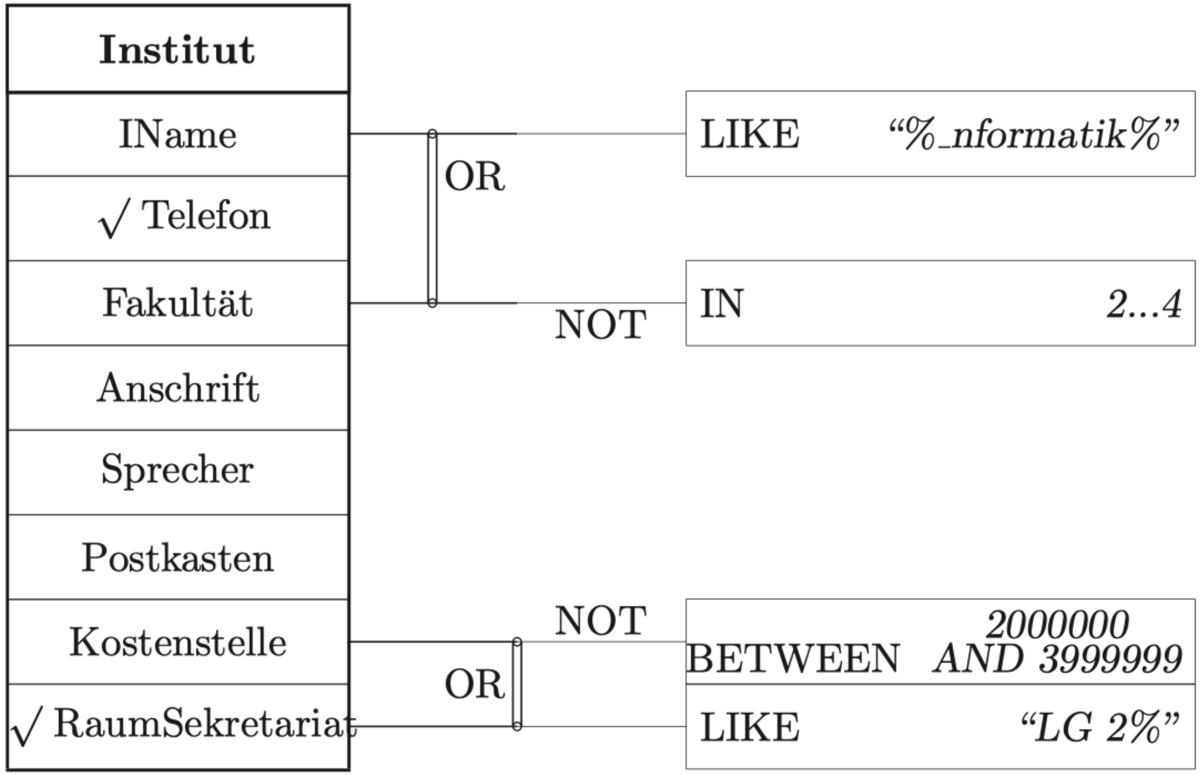


Bild 20: Auswahl in Klassen mit AND, OR und NOT

```
SELECT   Telefon, RaumSekretariat
  FROM   Institut
  WHERE    (IName LIKE "%_nformatik%"  OR    Fakultaet  NOT  IN (2,3,4))
      AND  (RaumSekretariat LIKE  "LG 2%"    OR
                Kostenstelle NOT BETWEEN 2000000 AND 3999999);
```

Source: Figure 20 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Complicated Boolean expressions are hard to visualize (by nature)

Here the nesting sequence is captured by indices

... X or Y or (A and B)...

or is higher nested, thus index 1

and then has index 2

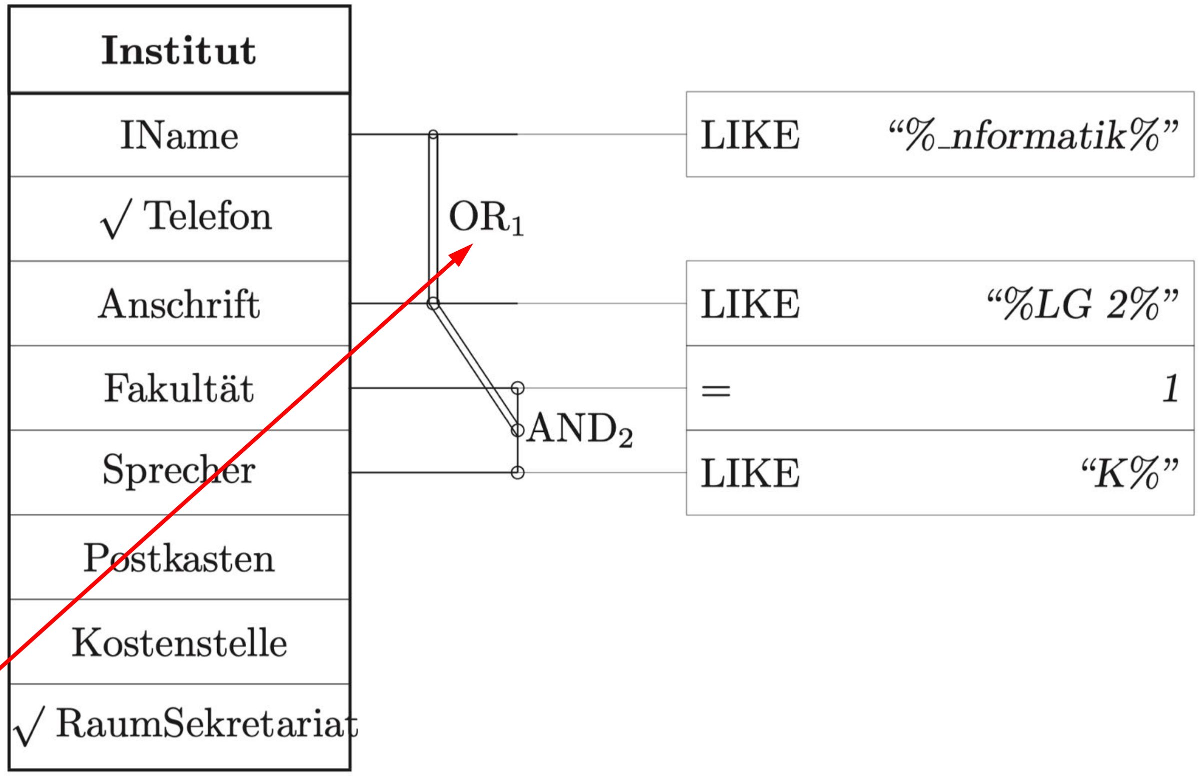


Bild 19: Auswahl in Klassen mit AND und OR

```
SELECT   Telefon, RaumSekretariat
  FROM   Institut
  WHERE  Anschrift LIKE "%LG 2%"    OR    IName LIKE ‘‘%_nformatik\%’’
         OR   (Fakultaet = 1   AND   Sprecher LIKE "K%");
```

Source: Figure 19 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

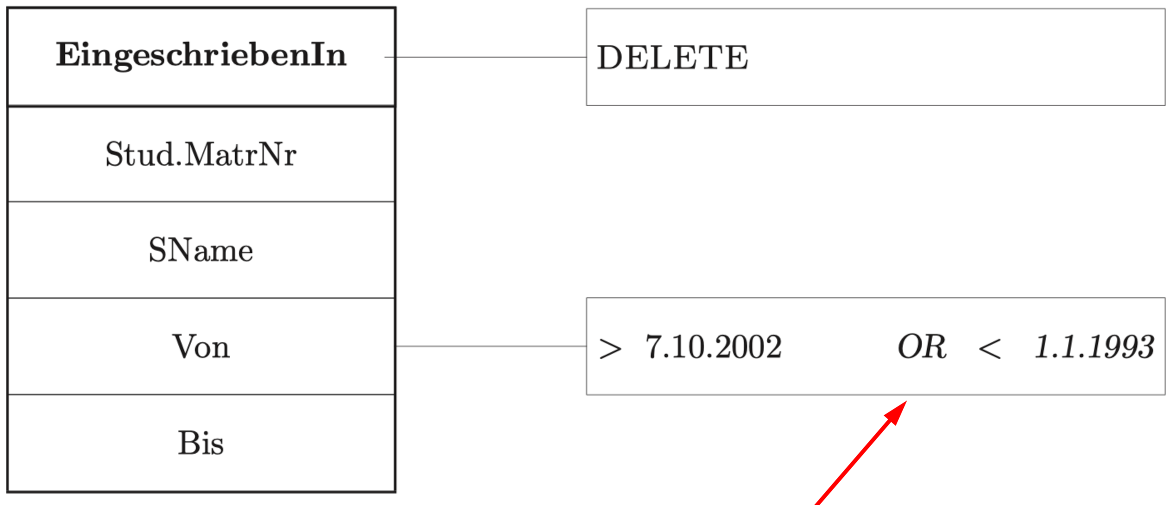


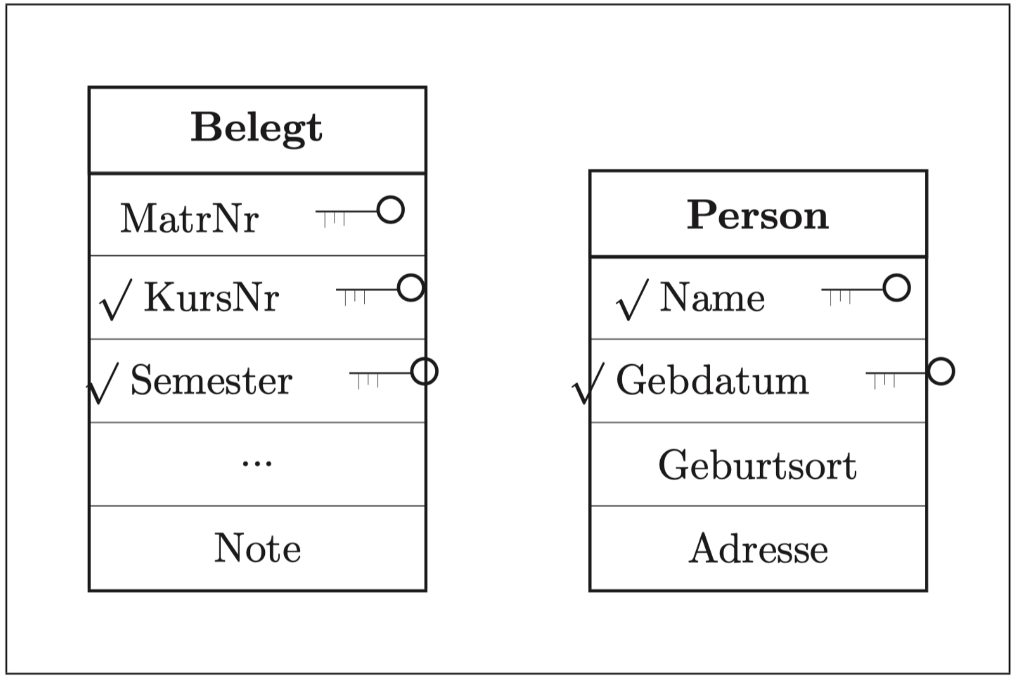
Bild 14: Streichen von Objekten mit Bedingung

Disjunction of values
on one predicate

```
DELETE FROM Eingeschrieben
    WHERE  Von > 7.10.2002  OR Von < 1.1.1993;
```

Source: Figure 14 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL



Additional box is used for Cartesian product

Bild 28: Kartesisches Produkt mit Projektion

```
SELECT   KursNr, Semester, Name, Gebdatum
  FROM      Belegt, Person;
```

Source: Figure 28 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.

Visual SQL

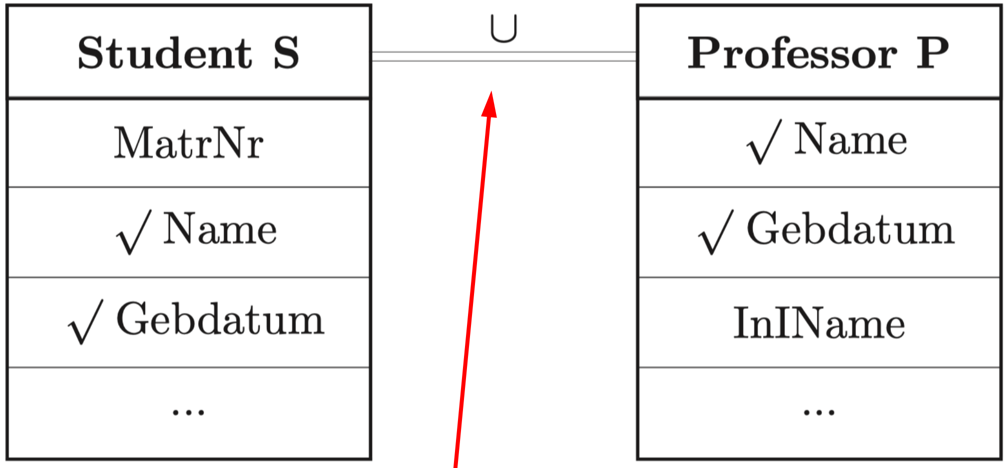


Bild 47: Vereinigung von zwei vollständig typengleichen Relationen

Union

```
SELECT      S.Name, S.Gebdatum
  FROM      Student S
UNION
SELECT      P.Name, P.Gebdatum
  FROM      Professor P;
```

Visual SQL

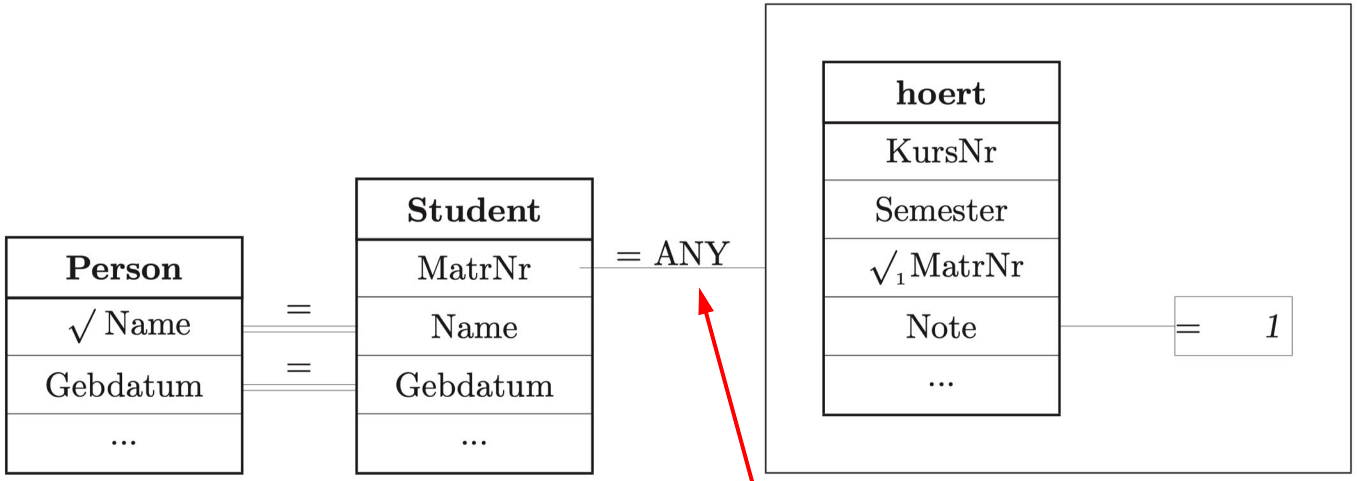


Bild 38: Teilanfrage mit Vergleich

Subquery with ANY

```
SELECT   Name
  FROM   Person, Student
  WHERE  Person.Name = Student.Name  AND Person.Gebdatum = Student.Gebdatum AND
         MatrNr  = ANY
                   (SELECT S1.MatrNr
                    FROM hoert S1
                    WHERE Note = 1);
```

Source: Figure 38 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Visual SQL

Subquery with ALL

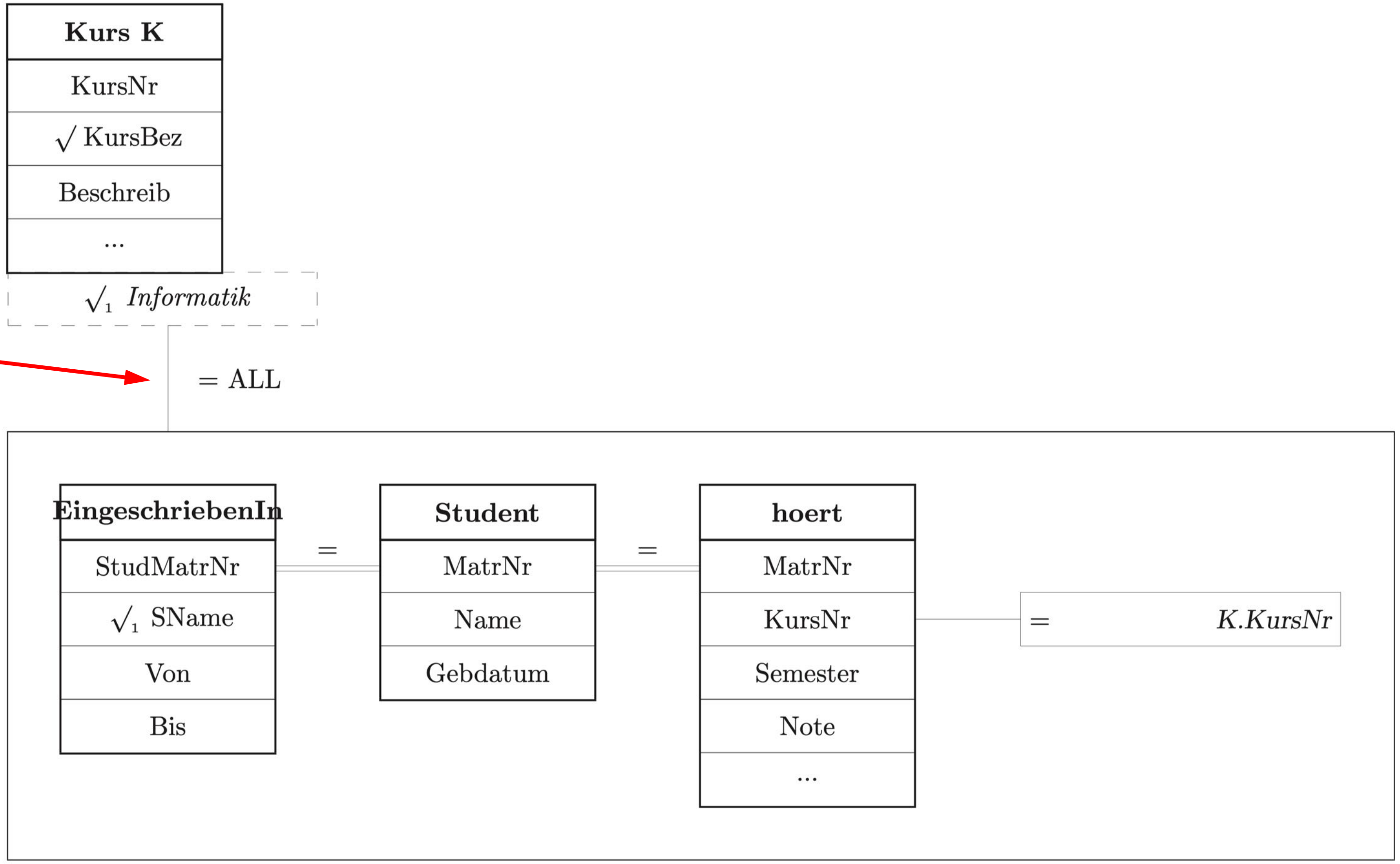


Bild 52: Korrelierte Teilanfragen mit ALL

```
SELECT     K.KursBez
  FROM     Kurs K
  WHERE    "Informatik" = ALL
              (SELECT   SName
                 FROM   hoert, Student, EingeschriebenIn
                 WHERE  K.KursNr = hoert.KursNr
                        AND  Student.MatrNr = hoert.MatrNr
                        AND EingeschriebenIn.StudMatrNr = Student.MatrNr );
```

Source: Figure 52 in Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003.
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (2011) (also QueryViz) Backup

QueryVis (formerly known as QueryViz)

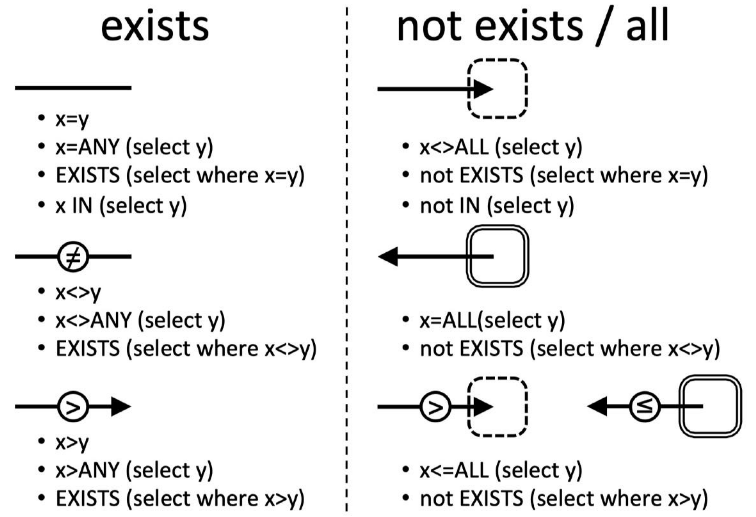


Figure 5: Lines with optional direction and comparison operators, together with bounding boxes in two line styles that group together relations suffice to express the most important syntactic constructs of nested SQL queries.

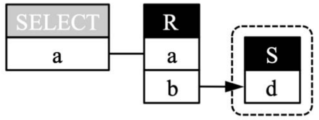
R(*a*,*b*,*c*), S(*d*,*e*)

q_2
```
select  a
from    R
where   b <> ALL
        (select  d
         from    S)
```

q_3
```
select  a
from    R
where   b not IN
        (select  d
         from    S)
```

q_4
```
select  a
from    R
where   not exists
        (select  *
         from    S
         where   b=d)
```

V_2



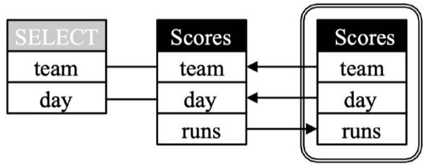
$$q_2: \; a : \exists b.\exists c.(R(a,b,c) \wedge \forall d.\forall e.(S(d,e) \Rightarrow d \neq b))$$
$$q_3: \; a : \exists b.\exists c.(R(a,b,c) \wedge b \notin \{d \,|\, \exists e.S(d,e)\})$$
$$q_4: \; a : \exists b.\exists c.(R(a,b,c) \wedge \neg\exists d.\exists e.(S(d,e) \wedge d = b))$$
$$V_2: \; a : \exists b.(R(a,b,\_) \wedge \neg(\exists d.(S(d,\_) \wedge d = b)))$$

Figure 4: Schema with three equivalent queries (q_1 to q_3), their common QueryViz representation V_2 and their respective translations into FOL.

Scores(*team*,*day*,*opponent*,*runs*)

q_5
```
select  S1.team, S1.day
from    Scores S1
where   not EXISTS
        (select  *
         from    Scores S2
         where   S1.runs=S2.runs
         and     (S1.team<>S2.team OR
                  S1.day<>S2.day))
```

V_5



$$q_5: \; t,d : \exists o.\exists r.(S(t,d,o,r) \wedge \neg(\exists t_2.\exists d_2.\exists o_2.\exists r_2.$$
$$(S(t_2,d_2,o_2,r_2) \wedge r_2 = r \wedge (t_2 \neq t \vee d_2 \neq d)))$$
$$V_5: \; t,d : \exists r.(S(t,d,\_,r) \wedge \forall t_2.\forall d_2.\forall r_2.$$
$$(S(t_2,d_2,\_,r_2) \wedge r_2 = r \Rightarrow t_2 = t \wedge d_2 = d))$$

Figure 6: Query for "teams and days on which the team had a run that was neither repeated on another day nor by another team," its QueryViz representation, and their respective translations into FOL.

Figure source: Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

QueryVis (formerly known as QueryViz)

```
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
```

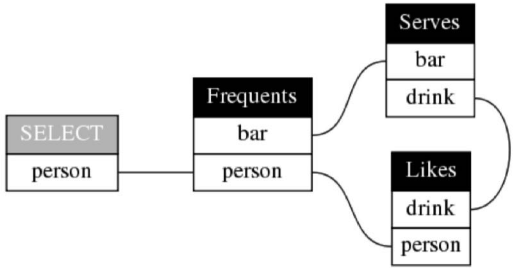


Figure 5: Visualizing a conjunctive query closely follows an all-familiar UML notation. ***Q: Find persons who frequent some bar that serves some drink they like.*** **There is nothing really new here.**

```
SELECT  F.person
FROM    Frequents F
WHERE   not exists
        (SELECT  *
         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))
```

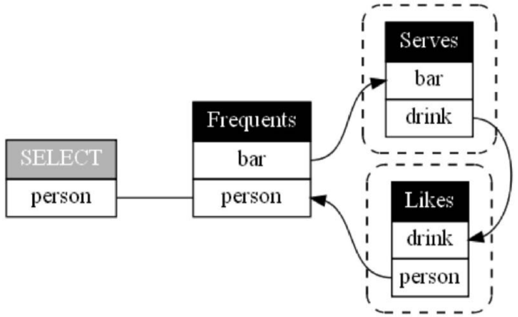


Figure 6: Visualizing a nested query still follows familiar UML notations, but now adds visual metaphors for $\nexists$ (dashed box) and reading order (arrows). ***Q: Find persons who frequent some bar that serves only drinks they like ≡ … some bar that serves no drink that is not liked by them.***

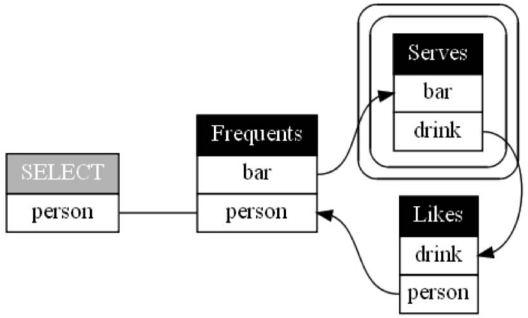


Figure 7: The visualization from Fig. 6 can be further simplified by using another visual metaphor for $\forall$ (double-lined box), a logical and intuitive operator that does not exist in SQL. ***Q: Find persons who frequent some bar that serves only drinks they like ≡ … some bar so that all drinks served are liked by them.***

QueryVis (formerly known as QueryViz)

```
SELECT  A
FROM    R
WHERE   B not in
        (SELECT  D
        FROM     S)
```

```
SELECT  A
FROM    R
WHERE   not exists
        (SELECT  *
        FROM     S
        WHERE    B=D)
```

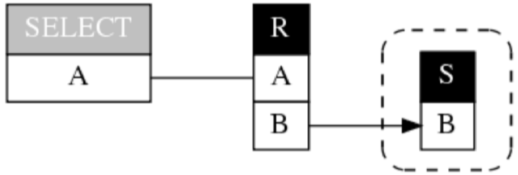


Figure 4: Two queries which are equivalent *except* if the column $S.b$ contains NULL values. Ignoring this one case, they are equivalent. Hence, the *query intent* can be shown by the same representation.

Figure source: "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805 "

QueryVis (formerly known as QueryViz)

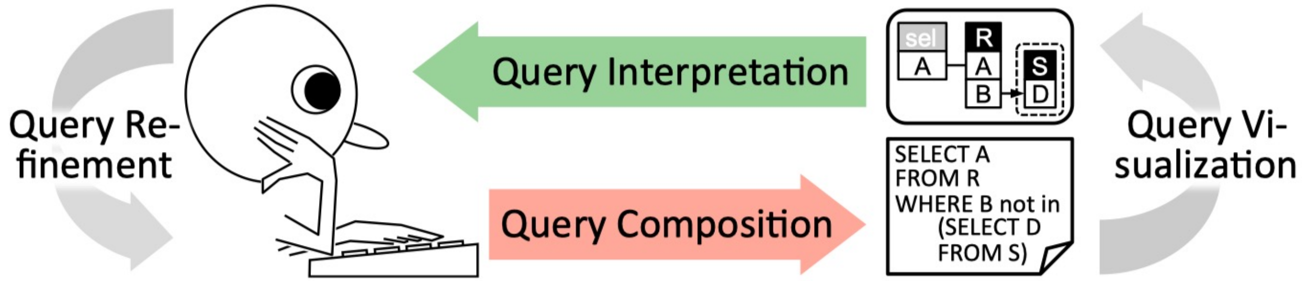


Figure 8: The vision: In the near future, *DBMSs will visualize queries too*, and not just data (as in information and scientific data visualization). This feature will allow iterative query refinement and will enhance the usability of databases.

Figure source: "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805 "

Dataplay (2012)
Backup

Dataplay

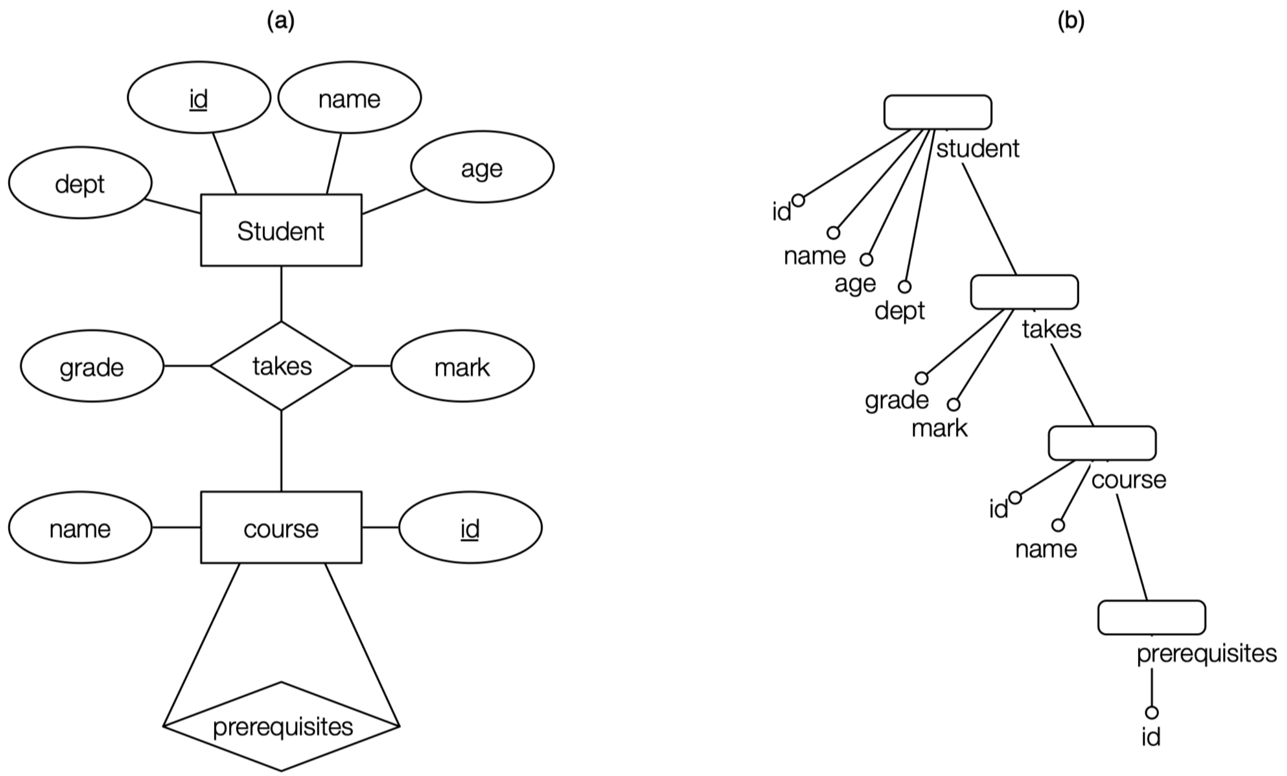


Figure 1: (a) ER-diagram of a school database; (b) The nested-UR schema with student as pivot.

Other observations:

1. Every query includes all tables from the schema. Thus querying one table (like red boats) requires showing all tables
2. Self-joins (one table appearing multiple times) seems not to be handled
3. Cyclic schemas are represented by using a spanning tree and a text-based join (reuse of variables)

Dataplay

Figure 3 illustrates two QTs: one for finding students with at least one A and one for finding straight-A students. Since the nesting hierarchy for a query is predetermined, users do not need to specify how to group tuples for quantification. More importantly, modifying the quantifier type is localized to simply changing the symbol from ∃ to ∀ on the constraint node; the structure of the QT is preserved.

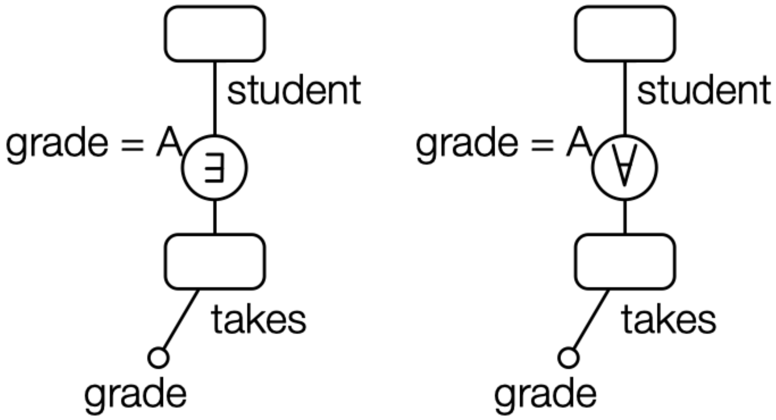


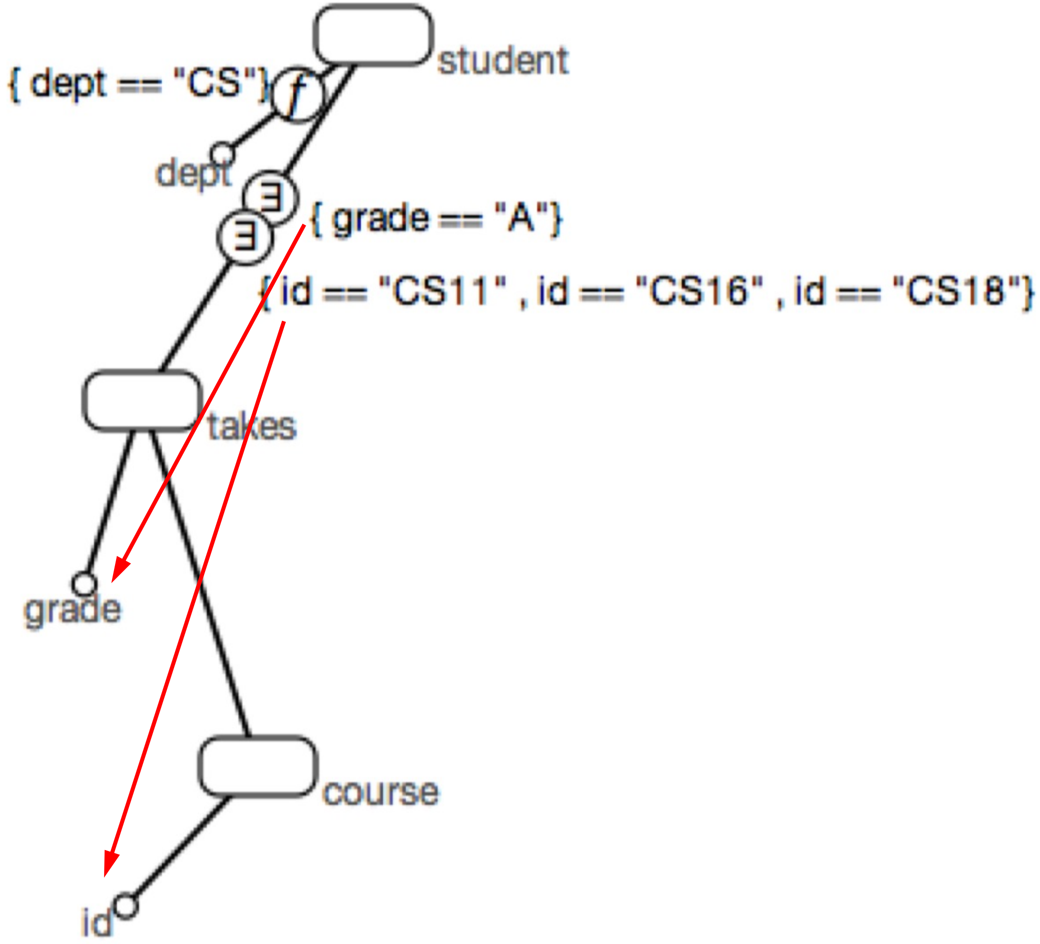
Figure 3: (a) Students with some A's (b) Straight-A's

Other observations:

1. Predetermined nesting hierarchy prevents correlated nested queries to be expressed (e.g. students who have taken all CS classes)

Source: Abouzied, Hellerstein, Silberschatz. Playful query specification with DataPlay. VLDB demo 2012. https://doi.org/10.14778/2367502.2367542
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

Dataplay



Other observations:

1. Ambiguity if multiple tables contain an attribute named "id" (e.g. course id or prerequisite id)
2. All quantifiers are shown at the root of the tree, even if attributes appear in leaves. "Location" of attributes in the schema and their use can be separated (non-local)

SIEUFERD (2016)
Backup

SIEUFERD (2016)

Filter. Using the filter popup (Figure 2), a filter can be defined on any field, indicated by the filter icon (▼). Filters on relation fields restrict the set of tuples retrieved in that relation, while filters on primitive fields restrict the tuples of the parent relation. In the following example, the MEETINGS relation is filtered to show only tuples for which the DAY is W:

courses ≺								
title	max _enroll	**readings** ≺ author_ name	title	**sections** ≺↖ type	num	**meetings** ≺↖ day ▼	start	end
Comedy	99	Moliere	The Miser	L	01	W	11:00	11:50
		Feydeau	A Flea in Her Ear	P	01	W	12:30	13:20
				P	02	W	12:30	13:20
		Reza	Art	P	03	W	13:30	14:20
				P	05	W	14:30	15:20
American Politics	78			L	01	W	11:00	11:50
				P	01	W	13:30	14:20
				P	01	W	13:30	14:20
				P	04	W	14:30	15:20
Judicial Politics	24	Rosenberg, Gerald	The Hollow Hope	L	01	W	11:00	11:50
				P	02	W	13:30	14:20
		Lazarus	Closed	P	04	W	11:00	11:50

Filter icon (▼) indicates filter. Here the meetings relation is filtered to show only tuples with day = W (actual filter condition only shown in separate "filter popup" window)

"Nest equijoins" are by default treated like left joins (all tuples on the left of the join are shown even if there is no match on the right). To remove tuples on the left-hand side of the operator, a special "hide parent if empty" setting is required and indicated by the arrow-towards-root icon (↖)

Notice that filters are applied to the relation, not the result of a join. This is important for modeling negation

SIEUFERD (2016)

Flat joins. Traditional flat joins can be expressed by referencing a descendant relation from a formula without enclosing the reference in an aggregate function. In the following example, each course title is repeated once for each distinct author name in the reading list, because the AUTHOR REFERENCE field in the COURSES relation references the READINGS relation without the use of an aggregate function:

courses			readings	
title	exam_ type	author reference *fx*	author_name	title
Roman Art	Other	Gombrich	Gombrich	Art and Illusion
Roman Art	Other	Ramage	Ramage	Roman Art
Comedy	Final	Feydeau	Feydeau	A Flea in Her Ear
Comedy	Final	=[author_name]		The Miser
Comedy	Final	Reza	Reza	Art
Russian Drama	Other	Chekhov	Chekhov	The Seagull
Russian Drama	Other	Pushkin	Pushkin	Little Tragedies
Russian Drama	Other	Vampilov	Vampilov	The Duck Hunt
American Politics	Final			
Junior Seminars	Other	Pierre Loti	Pierre Loti	India
Judicial	Final	Lazarus	Lazarus, Edward	Closed Chambers

In order to apply a filter *after* a left join, a "reference formula" needs to be added that represents a "reference" to an attribute in the right side of the left join

The actual behavior is that of a left join, with a null value being returned for the course AMERICAN POLITICS, which has no readings in its reading list. To express an inner join instead, the HIDE

Filters and aggregate functions. When an aggregate function references a relation with a filter applied to it, the filter is evaluated before the aggregate.

It is equally valid to define a filter on the output side of an aggregate, e.g. on TITLE or TOTAL DURATION in the example above.

SIEUFERD (2016)

- *Set difference.* Here, we can filter for null values generated by a left join. If $e = e_a - e_b$, with $n = N(e)$, then $t(e) = t(\pi_{\langle e_a[1]\rangle \to e[1],\dots,\langle e_a[n]\rangle \to e[n]}(\sigma_{\langle M \text{ IS NULL}\rangle}(e_a \ltimes_C e'_b)))$ where $\ltimes$ is a left outer join, $C = \langle e_a[1] = e'_b[1] \wedge \dots \wedge e_a[n] = e'_b[n]\rangle$, and e'_b adds an arbitrary non-nullable attribute M to e_b, e.g. $e'_b = \pi_{\langle e_b[1]\rangle \to e'_b[1],\dots,\langle e_b[n]\rangle \to e'_b[n],\langle 42\rangle \to M}(e_b)$. Another approach would be to COUNT values in e_b and filter for zero.

Negation (set difference) is modeled via a left join. Since filters are by applied to the relation, not the result of a join, one needs to add a "reference formula" to the relation on the left side of the join, followed by a filter

Here the filter "is null" needs to be applied to the relation resulting from the join, not the relation on the right side

SIEUFERD (2016)

The following is a simple query that instantiates the table called COURSES and displays a selection of its fields:

courses					
id	area_id	title	may_pdf	may_audit	exam_type
56	2	Roman Art	N	Y	Other
177	2	Comedy	Y	Y	Final
845	2	Russian Drama	N	N	Other
1795	4	American Politics	Y	Y	Final
2566		Junior Seminars	N	N	Other
3921	4	Judicial Politics	Y	Y	Final

Crow's foot is shown for single table.
The symbol is not needed for understanding a query, and was also removed from later systems by the authors.

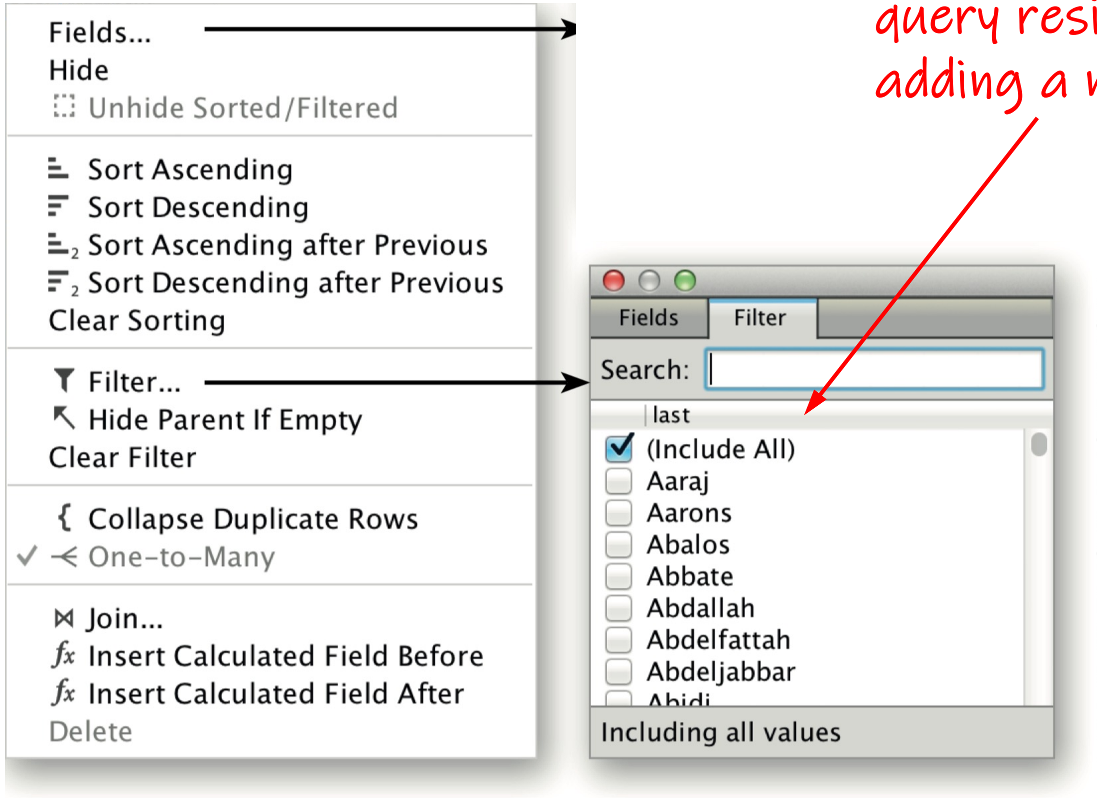
Figure source: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SIEUFERD (2016)

Negation of a filter color = 'red' to instead color != 'red' requires choosing all other values. But it is achieved with a sequence of:

- first choosing "include all",
- then unselecting "red"

This translates into a filter "color <> 'red' ", which makes the query resilient to changes in data: thus it still works after adding a new color, say "frog-green"

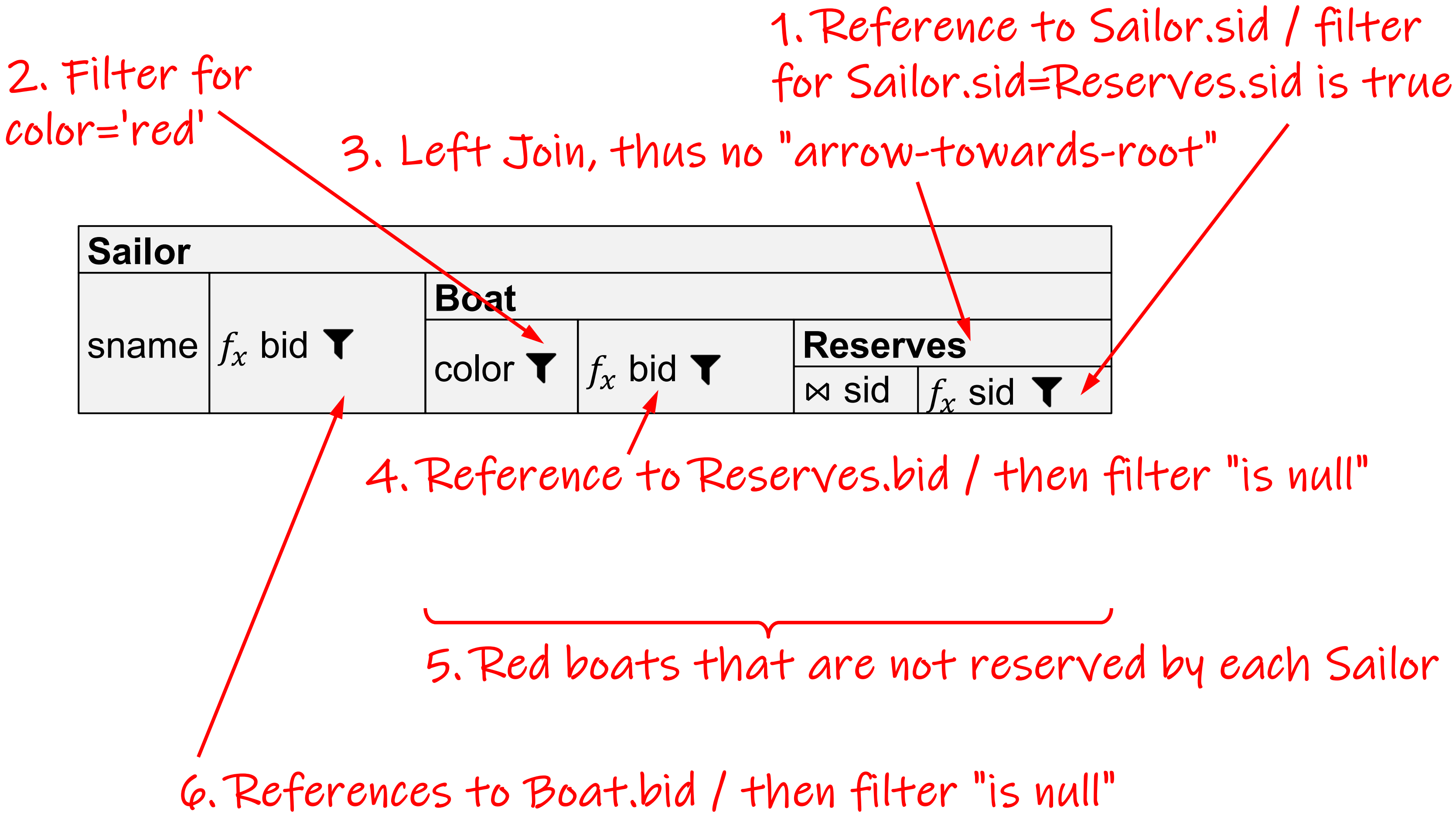


Filter popup: Allows the user to associate a filter with the currently selected field. The list of values available to filter on is generated automatically using a separate database query. Filters may be associated with either primitive fields or relation fields.

SIEUFERD (2016)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
   (select *
   from Boat B
   where color = 'red'
   and not exists
      (select *
      from Reserves R
      where S.sid=R.sid
      and R.bid=B.bid))
```

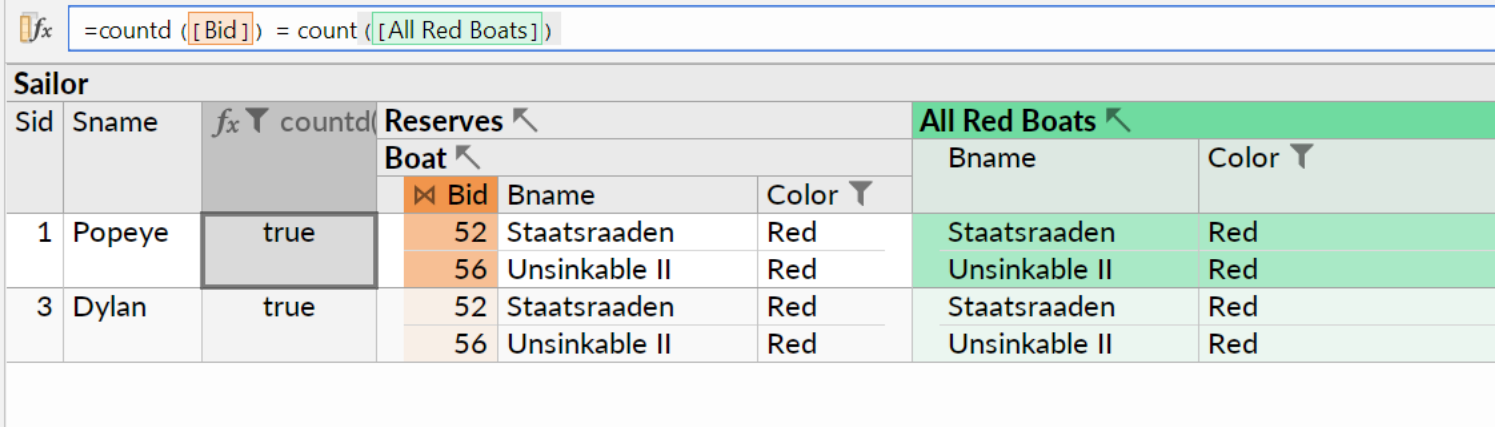


Notice how the query header by itself does not allow a user to understand a query's semantic.

SIEUFERD (2016)

Q: "Find sailors who reserved all red boats."

```
select S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and R.bid=B.bid))
```




```
select S.sname
from Sailor S, Boat B, Reserves R
where R.sid=S.sid
and R.bid=B.bid
and color = 'red'
group by S.sid
having count(distinct B.bid) =
    (select count(B2.bid)
    from boat B2
    where B2.color='red')
```

The preferred way of expressing universally quantified queries in SIEUFERD is via GROUP BY and COUNTING

Figure source: Personal communication with Eirik Bakke

SQLVis (2021)
Backup

SQLVis

```sql
SELECT *
FROM store
WHERE city != London;
```

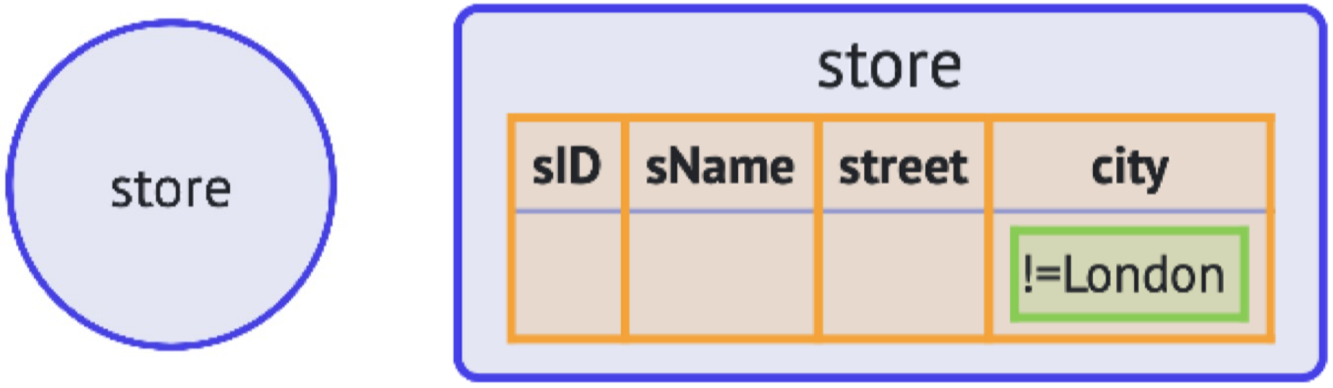
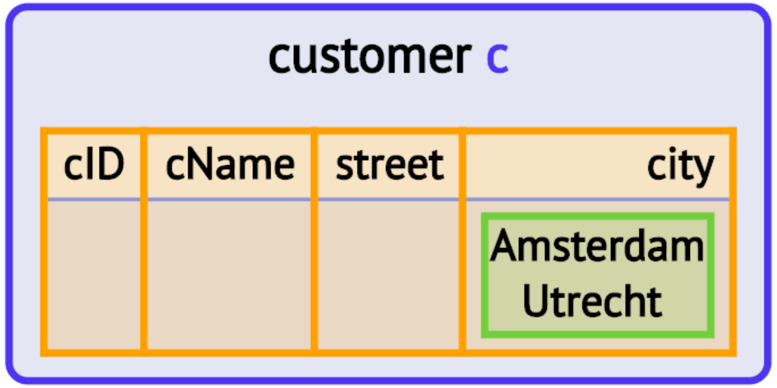


Fig. 3: SQLVis representation for a simple query (collapsed on the left, expanded in the middle).

Source: Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 3. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries, VLDB tutorial 2023. https://northeastern-datalab.github.io/visual-query-representation-tutorial/

SQLVis

```
SELECT *
FROM customer AS c
WHERE city = "Amsterdam" OR city = "Utrecht";
```



SQLVis

```sql
SELECT c.cName
FROM customer AS c, purchase AS p
WHERE p.cID = c.cID;
```

SQLVis

All complex SQL queries contain subqueries and other types of nesting. To visualize these subqueries in the most intuitive way, SQLVis draws boxes around these subqueries as suggested by Thalheim [38]. To distinguish between different subqueries on the same level, and nested subqueries on different levels, SQLVis draws each level of nesting in a different saturation (see Figure 4). The use of these different colors helps to give an immediate overview of the level of nesting in the query. In case a subquery is negated, for example by using NOT EXISTS or NOT IN, SQLVis displays this negation in words. Other options, such as a different background color or a border for the box led to a very cluttered representation.

```
SELECT s.sID, s.sName
FROM store AS s
WHERE NOT EXISTS (
  SELECT p.pID
  FROM product AS p
  WHERE NOT EXISTS (
    SELECT *
    FROM inventory AS i
    WHERE s.sID = i.sID
    AND i.pID = p.pID))
```

(b) A query to find stores with all items in stock.