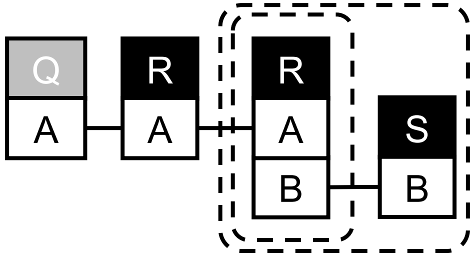


Last Updated June 3, 2024

A Comprehensive Tutorial on over 100 years of Diagrammatic Representations of Logical Statements and Relational Queries

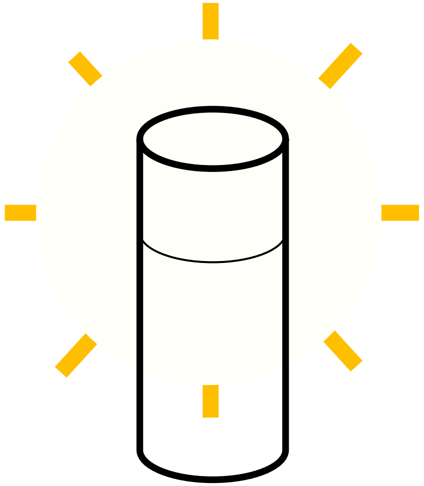
Wolfgang Gatterbauer
(w.gatterbauer@northeastern.edu)

May 17, 2024
(ICDE'24)

PLEASE ask questions and let me know if you spot any errors (we cover a lot of material and I must have made some errors…)
Thank you ☺

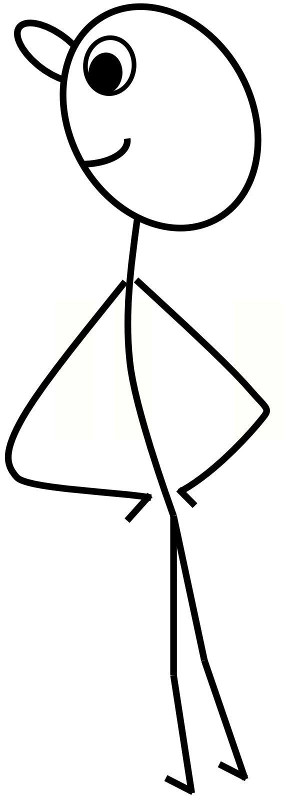
An anonymous feedback form is linked from the tutorial web page:
https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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



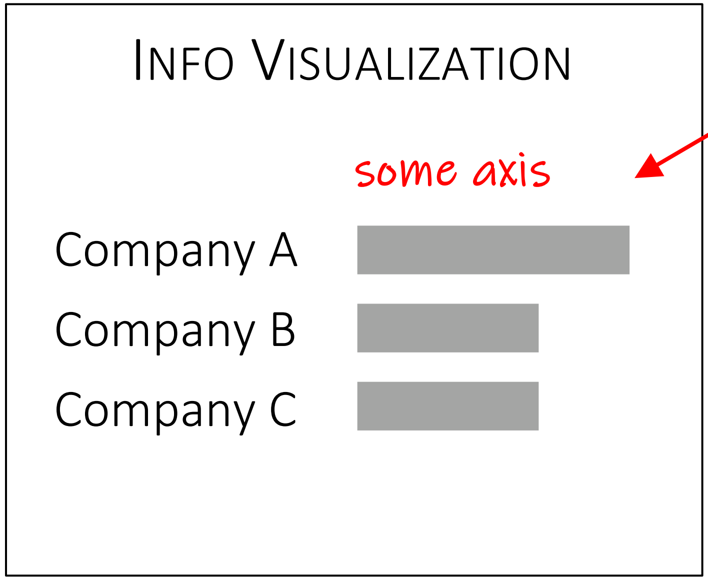
Q: Find me companies in Utrecht.

A: Company A, Company B, Company C

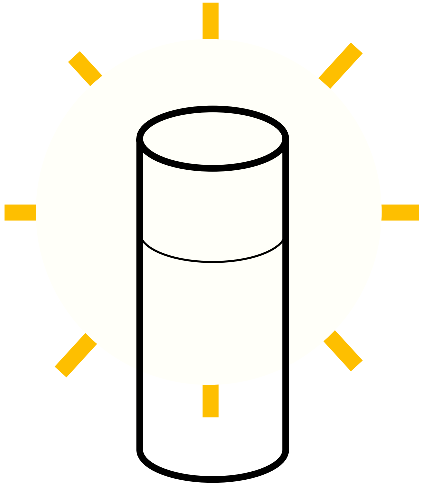


Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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

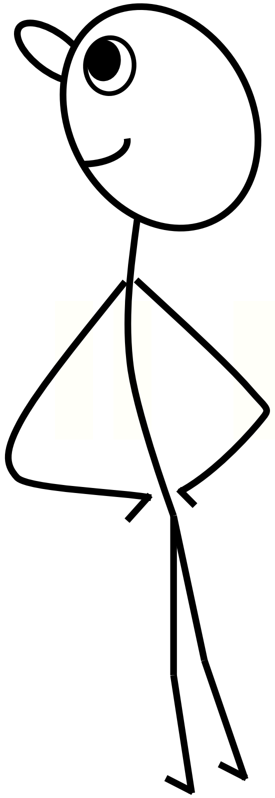


Visualize the answer



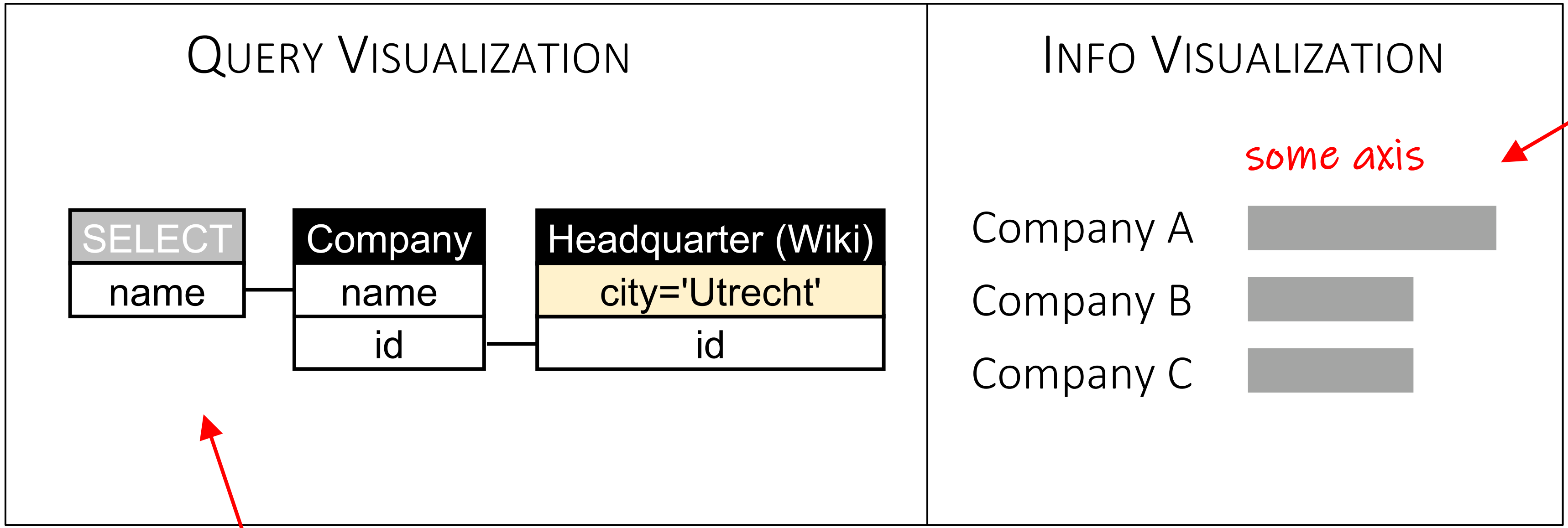
Q: Find me companies in Utrecht.

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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



QUERY VISUALIZATION

SELECT
name

Company
name
id

Headquarter (Wiki)
city='Utrecht'
id

INFO VISUALIZATION

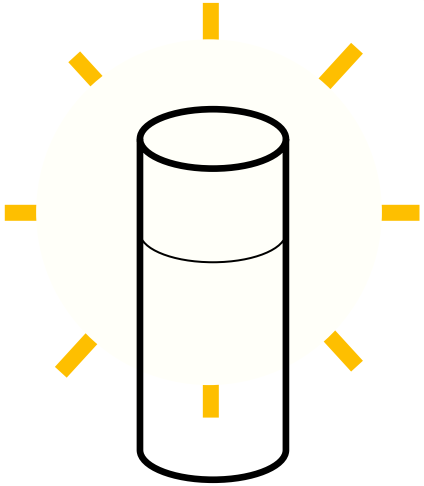
some axis

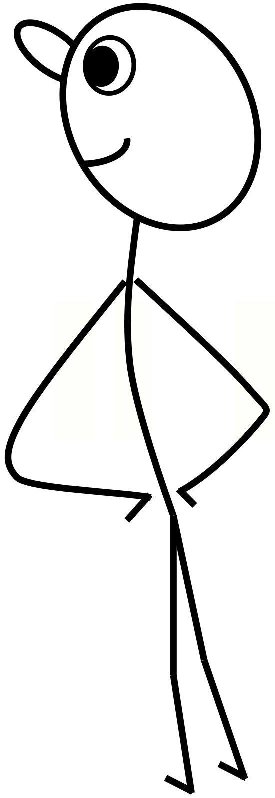
Company A
Company B
Company C

Visualize
the answer

! Visualize
the query

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



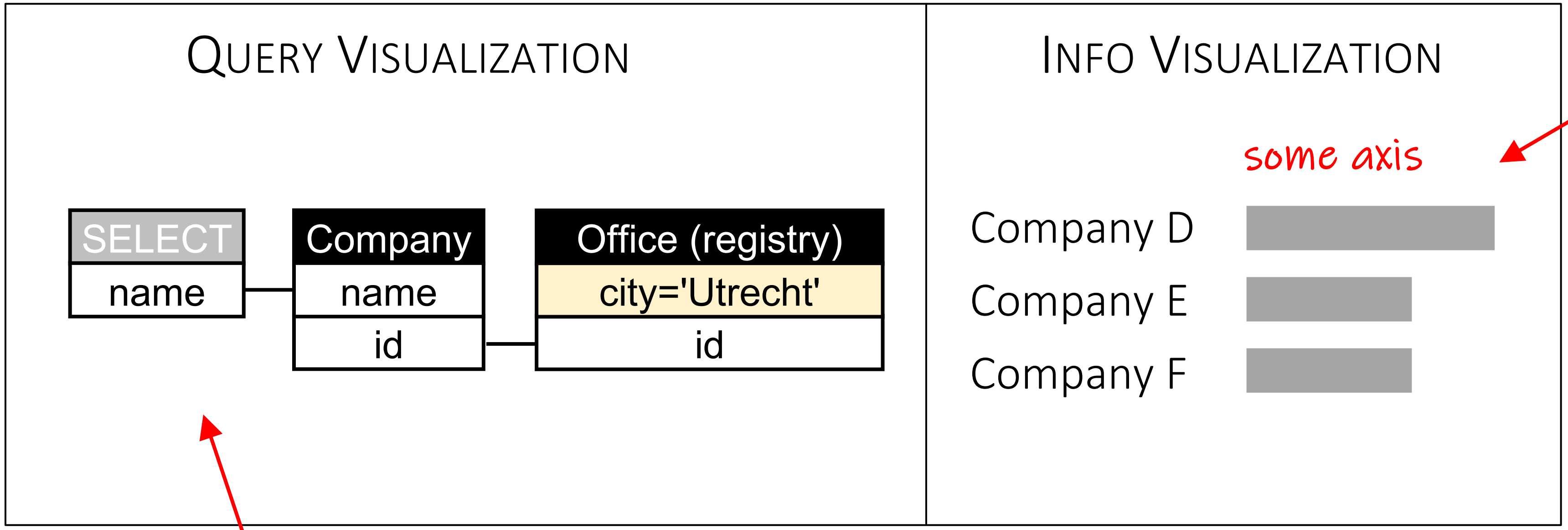


Q: Find me companies in Utrecht.

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

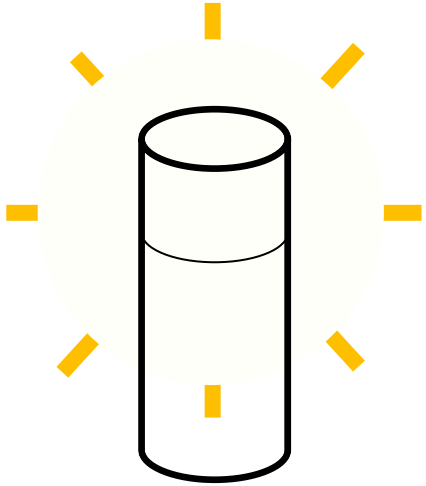
1. How do you know your voice assistant understood you correctly?

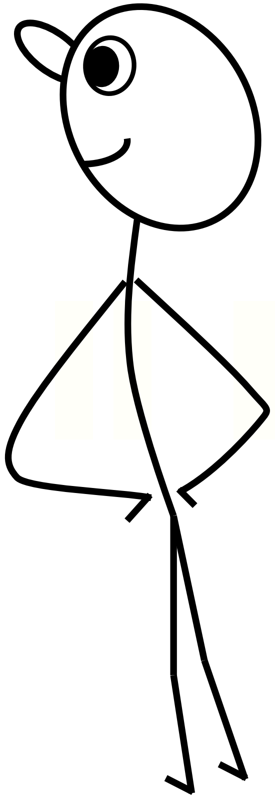


Visualize the answer

! Visualize the query

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





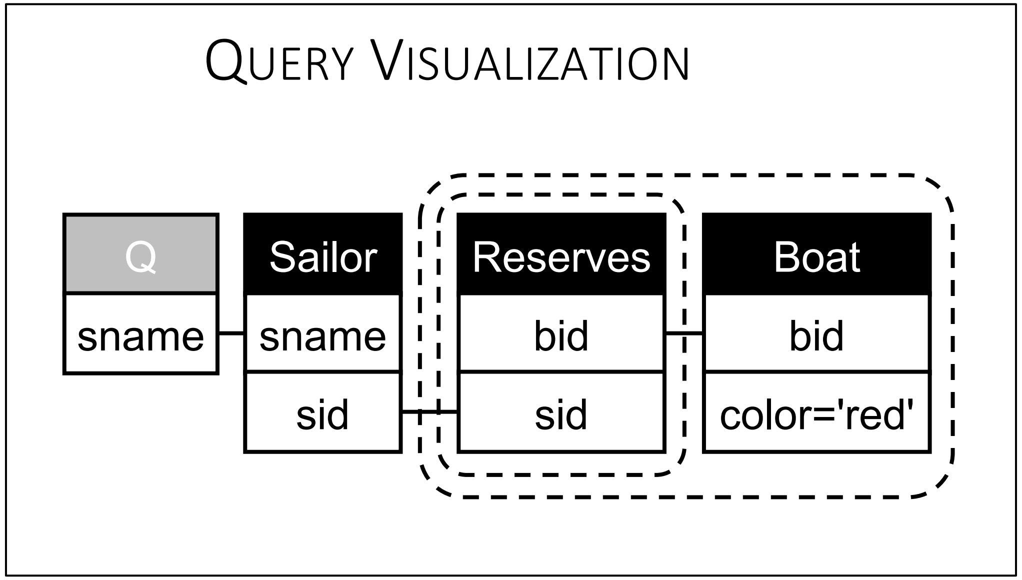
Q: Find me companies in Utrecht.

A: Company D, Company E, Company F

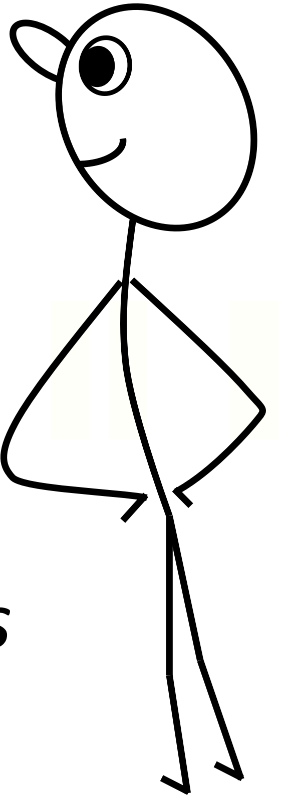
Example adapted from: Principles of Query Visualization. IEEE Debull. 2022. http://sites.computer.org/debull/A22sept/p47.pdf

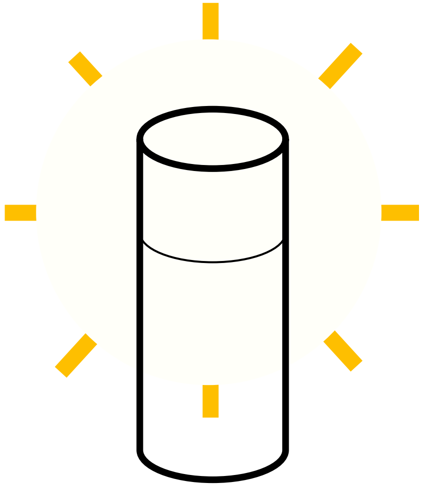
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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



Can a diagrammatic representation of the query help the user quickly "debug" its translation, i.e. check it was understood 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

2. Query Debugging of Student-Submitted Homeworks

Reference solution:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1
where L.drinker = 'Eve'
and L.beer = S1.beer
and not exists
    (select *
    from Serves S2
    where S2.beer = S1.beer
    and S2.price > S1.price)
```

Homework problem:

Q1: For each beer that Eve likes, find the bar serving it at the highest price.

As TA you need to decide what is wrong with this query:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1, Serves S2
where L.drinker = 'Eve'
and L.beer = S1.beer
and S1.beer = S2.beer
and S1.price > S2.price
```

?

Example given by Jun Yang in a talk on building software tools for helping students learn SQL at a DATAlab seminar https://db.khoury.northeastern.edu/activities (5/2024)
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

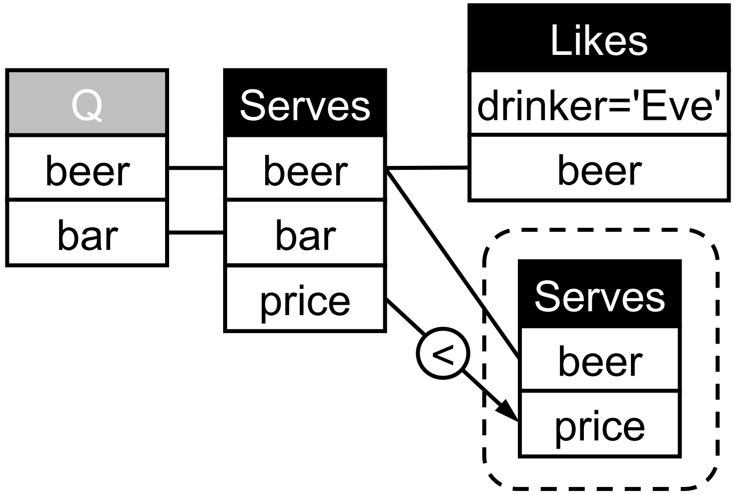
2. Query Debugging of Student-Submitted Homeworks

Reference solution:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1
where L.drinker = 'Eve'
and L.beer = S1.beer
and not exists
    (select *
    from Serves S2
    where S2.beer = S1.beer
    and S2.price > S1.price)
```

Homework problem:

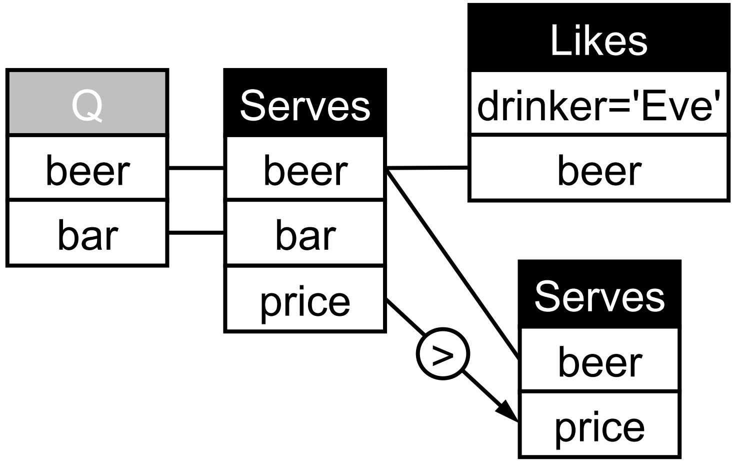
Q1: For each beer that Eve likes, find the bar serving it at the highest price.



As TA you need to decide what is wrong with this query:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1, Serves S2
where L.drinker = 'Eve'
and L.beer = S1.beer
and S1.beer = S2.beer
and S1.price > S2.price
```

?



Example given by Jun Yang in a talk on building software tools for helping students learn SQL at a DATAlab seminar https://db.khoury.northeastern.edu/activities (5/2024)
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

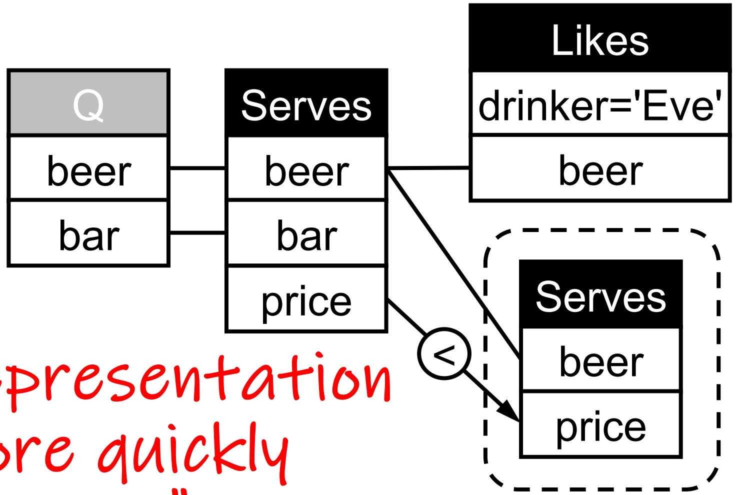
2. Query Debugging of Student-Submitted Homeworks

Reference solution:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1
where L.drinker = 'Eve'
and L.beer = S1.beer
and not exists
    (select *
    from Serves S2
    where S2.beer = S1.beer
    and S2.price > S1.price)
```

Homework problem:

Q1: For each beer that Eve likes, find the bar serving it at the highest price.

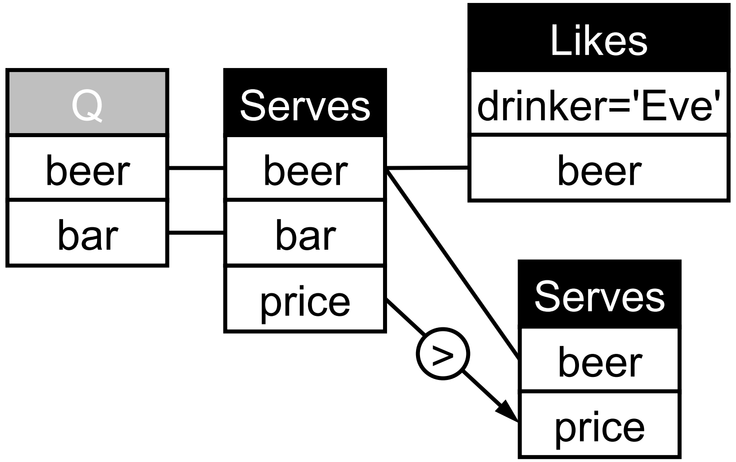


Again: Can a diagrammatic representation of the query help the user more quickly understand its intent, its "pattern"?

As TA you need to decide what is wrong with this query:

```
select distinct S1.beer, S1.bar
from Likes L, Serves S1, Serves S2
where L.drinker = 'Eve'
and L.beer = S1.beer
and S1.beer = S2.beer
and S1.price > S2.price
```

Q2: For each beer that Eve likes, find the bar not serving it at the lowest price.



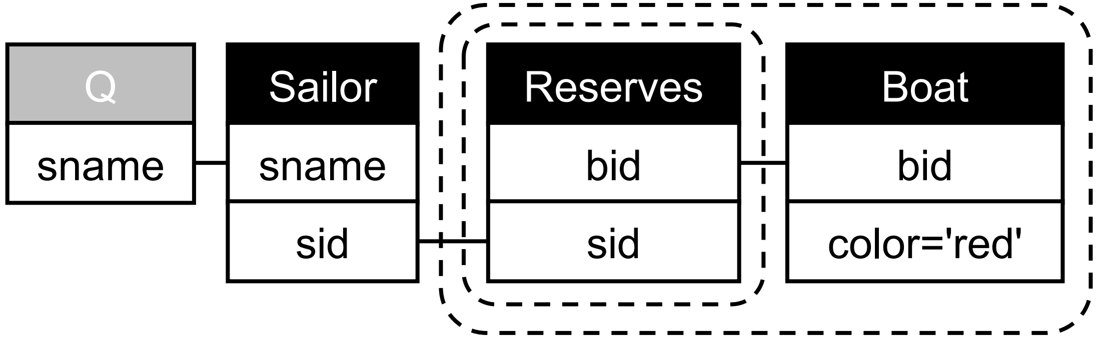
Example given by Jun Yang in a talk on building software tools for helping students learn SQL at a DATAlab seminar https://db.khoury.northeastern.edu/activities (5/2024)
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

1

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

1



Query Intent: ?

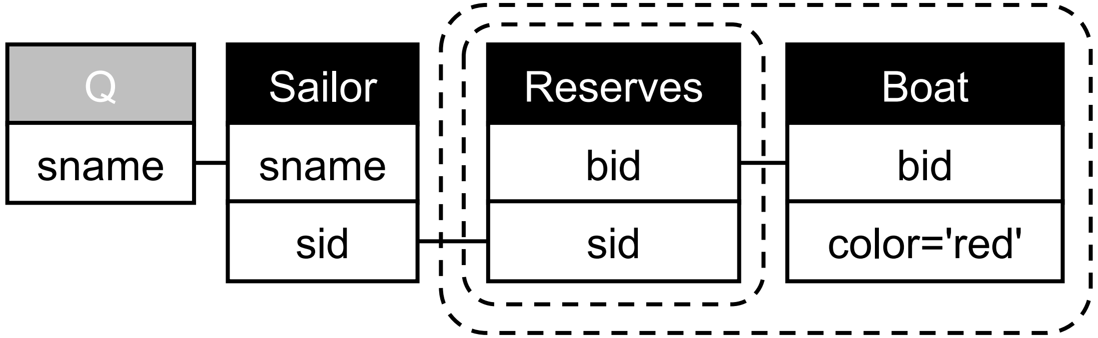
Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

1

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

1



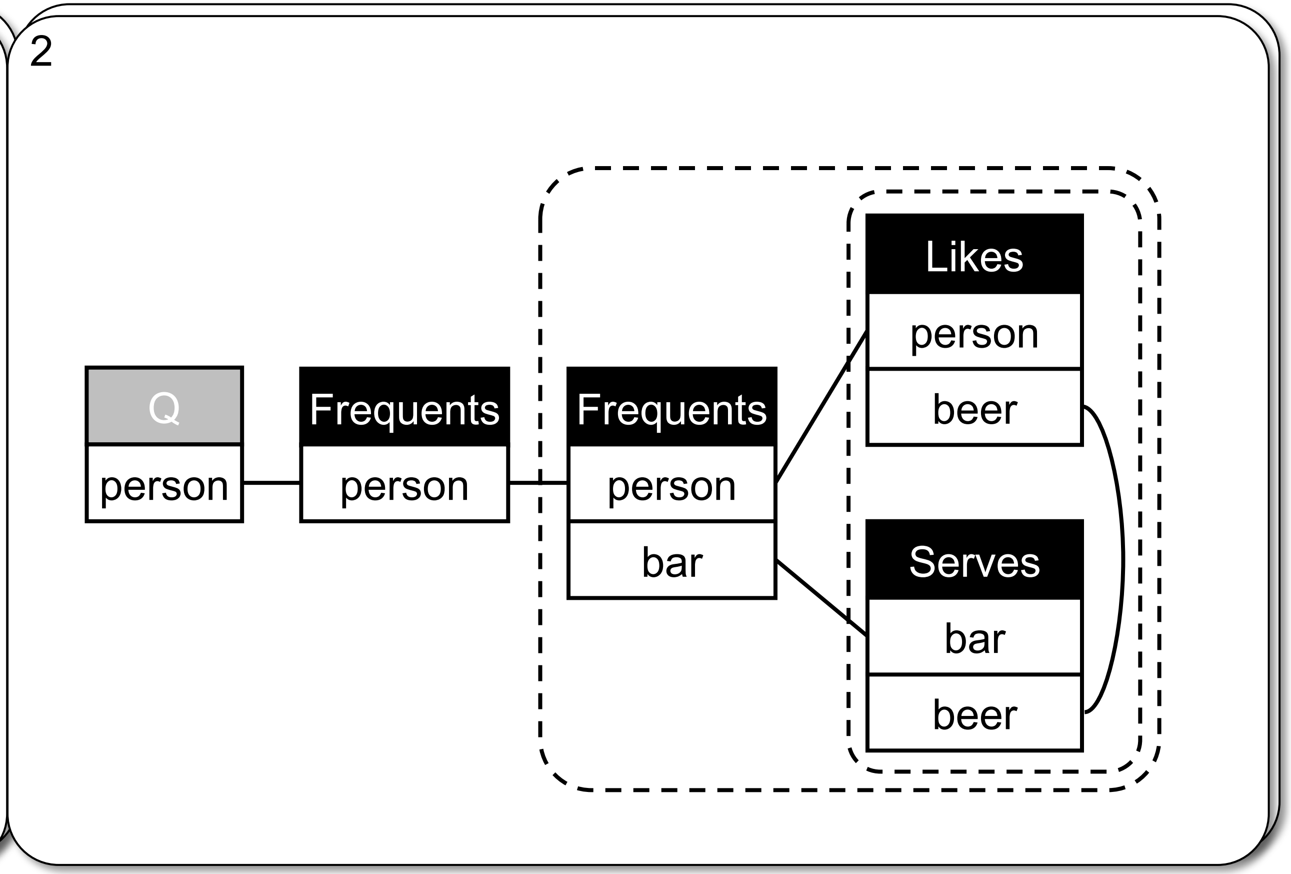
Query Intent: "Find persons who reserved all red boats."

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

2

```
select distinct 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 = F2.person
        and L4.drink = S3.drink
        and S3.bar = F2.bar))
```



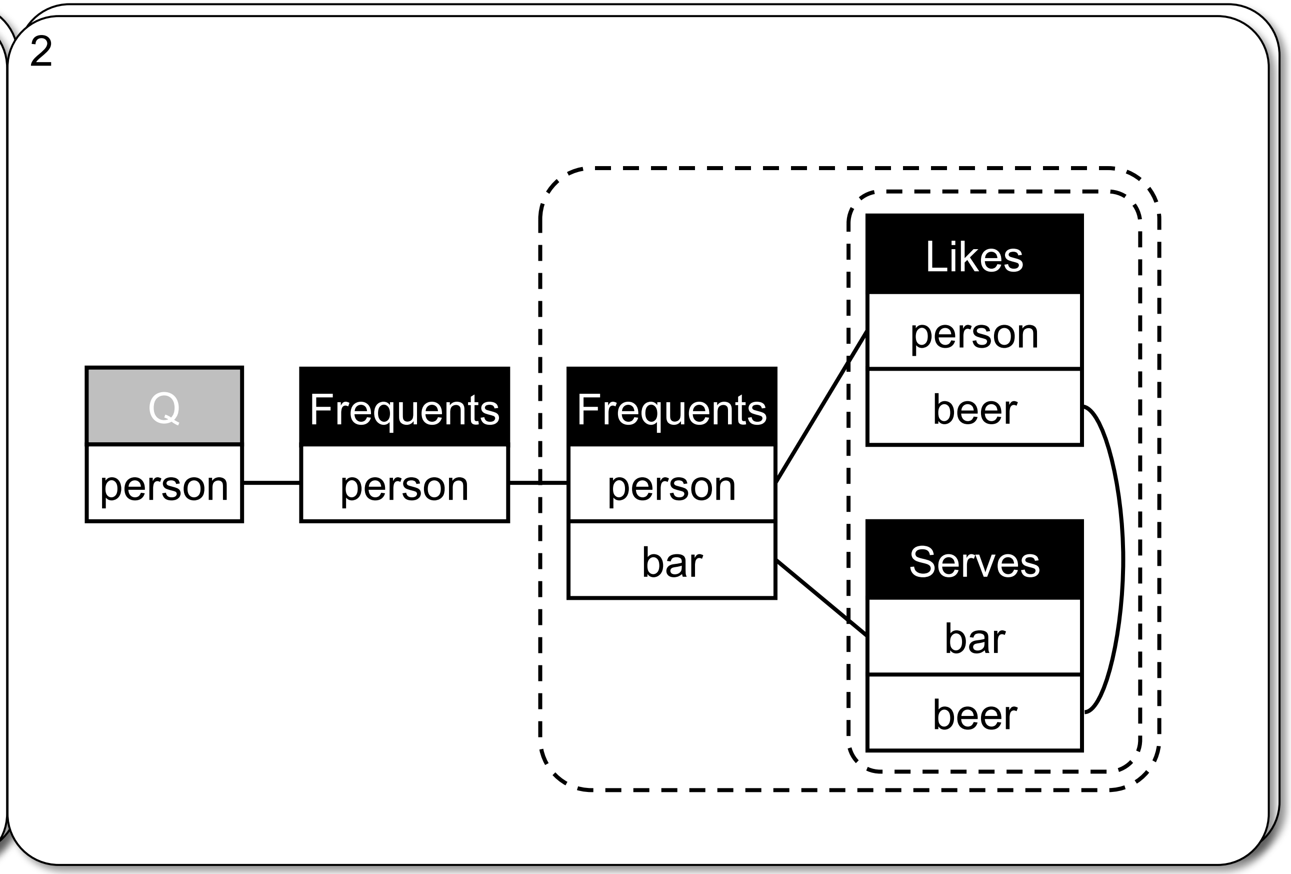
Query Intent: ?

Database to run SQL queries is available as schema 331 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

2

```
select distinct 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 = F2.person
        and L4.drink = S3.drink
        and S3.bar = F2.bar))
```



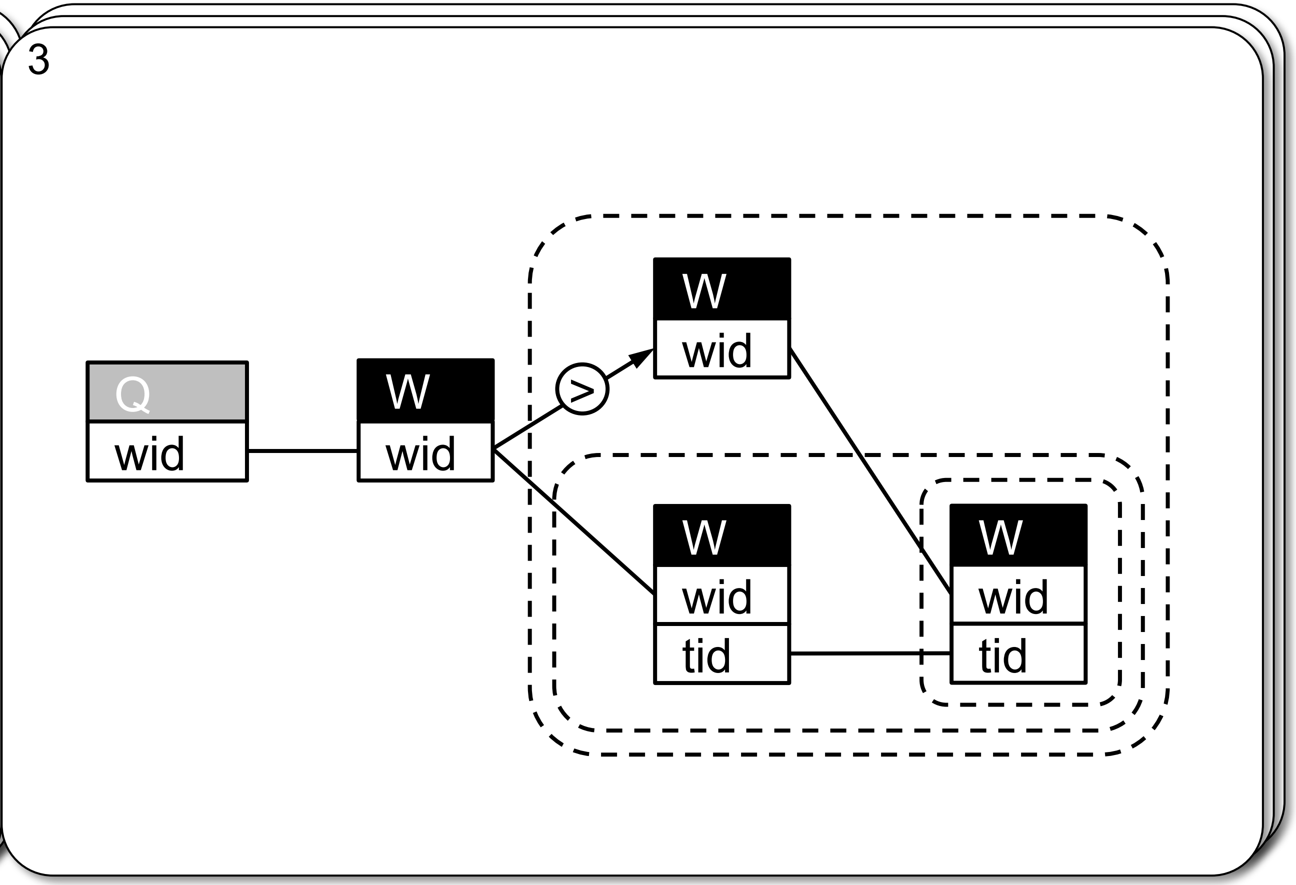
Query Intent: "Find persons who frequent only bars that serve some beer they like."

Database to run SQL queries is available as schema 331 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

3

```
select distinct 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)))
```

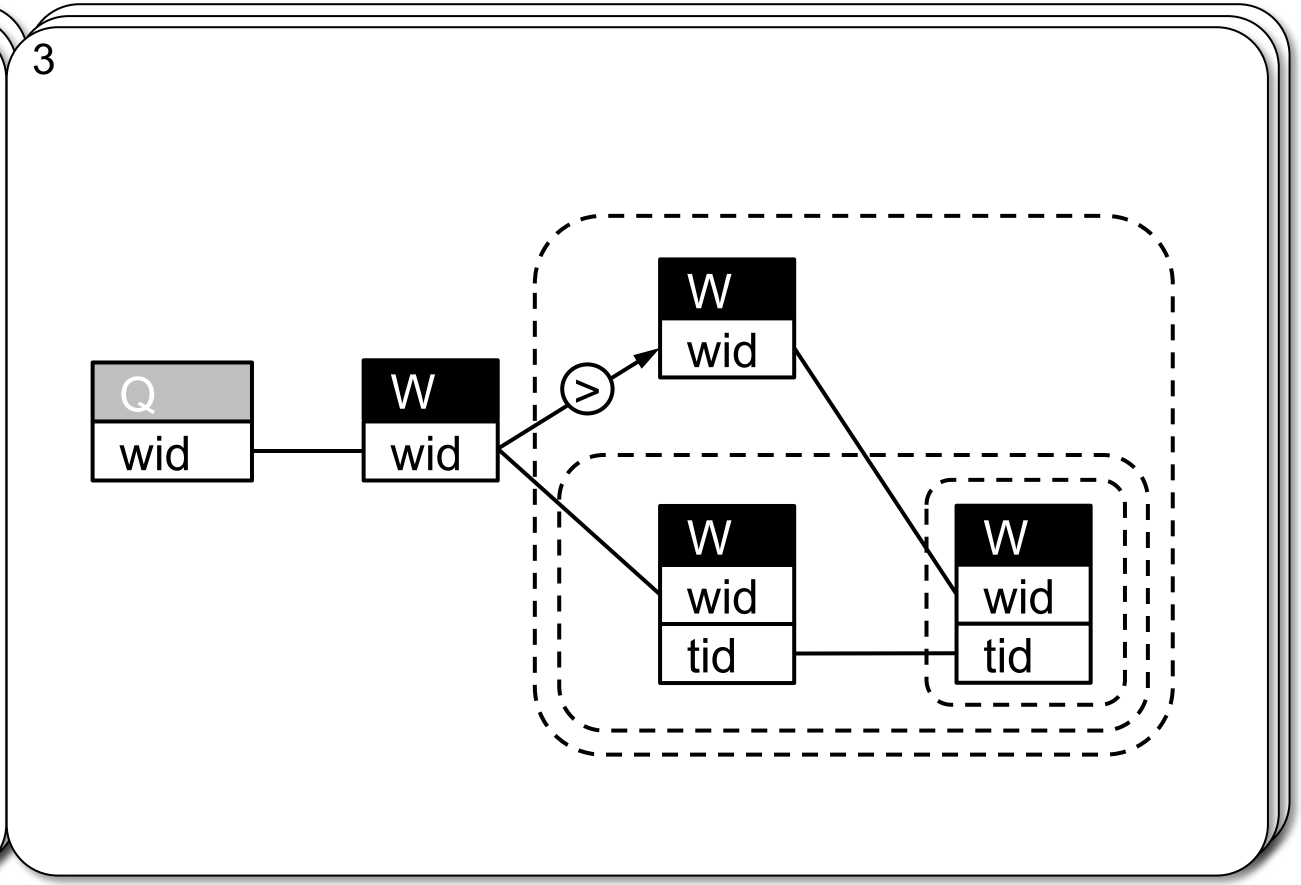


Query Intent: ?

Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

```
select distinct 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)))
```



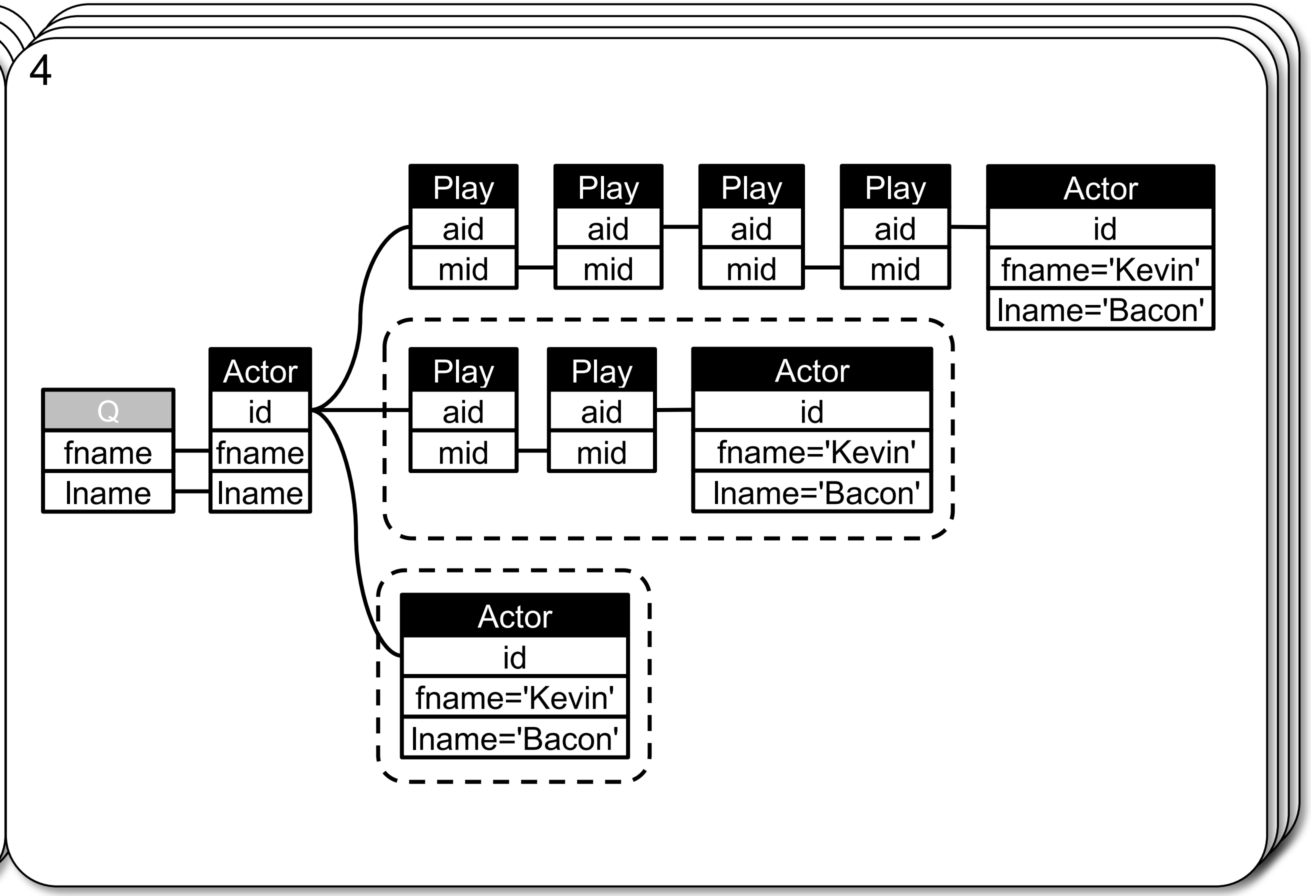
<u>Query Intent</u>: "Find worlds for which no earlier world contains all its tuples"

Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

4

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



Query Intent: ?

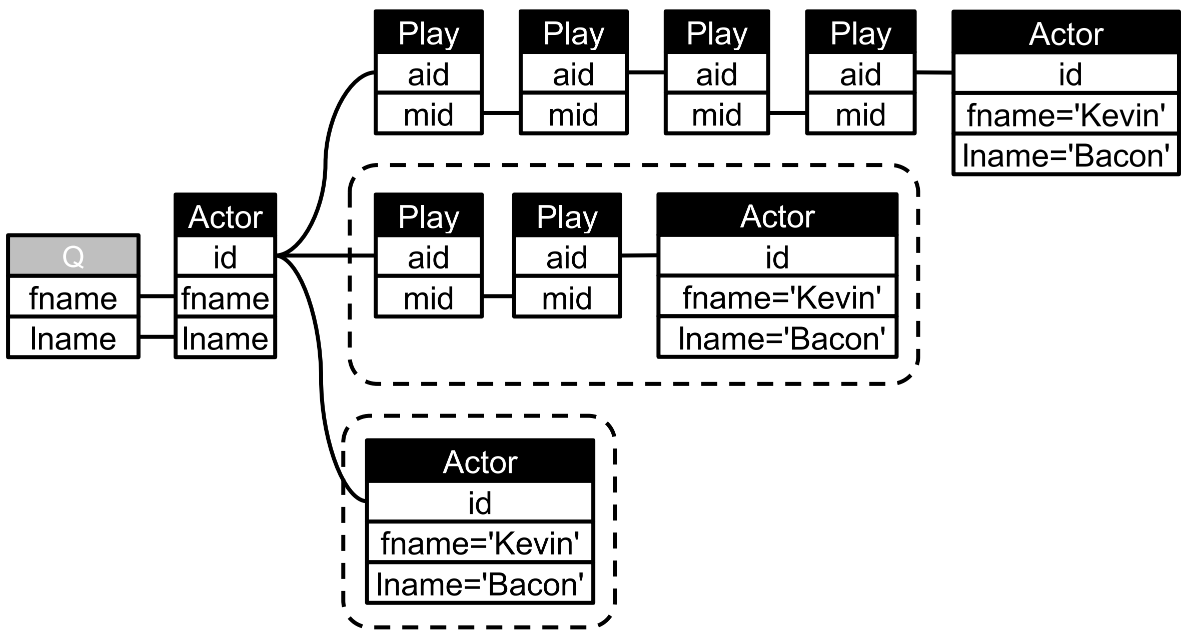
Database to run SQL queries is available as schema 300 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Browsing & understanding existing Queries

4

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

4



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

Database to run SQL queries is available as schema 300 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Example taken from: Databases will Visualize Queries too. PVLDB vision 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Why logical diagrams [Gardner'58]

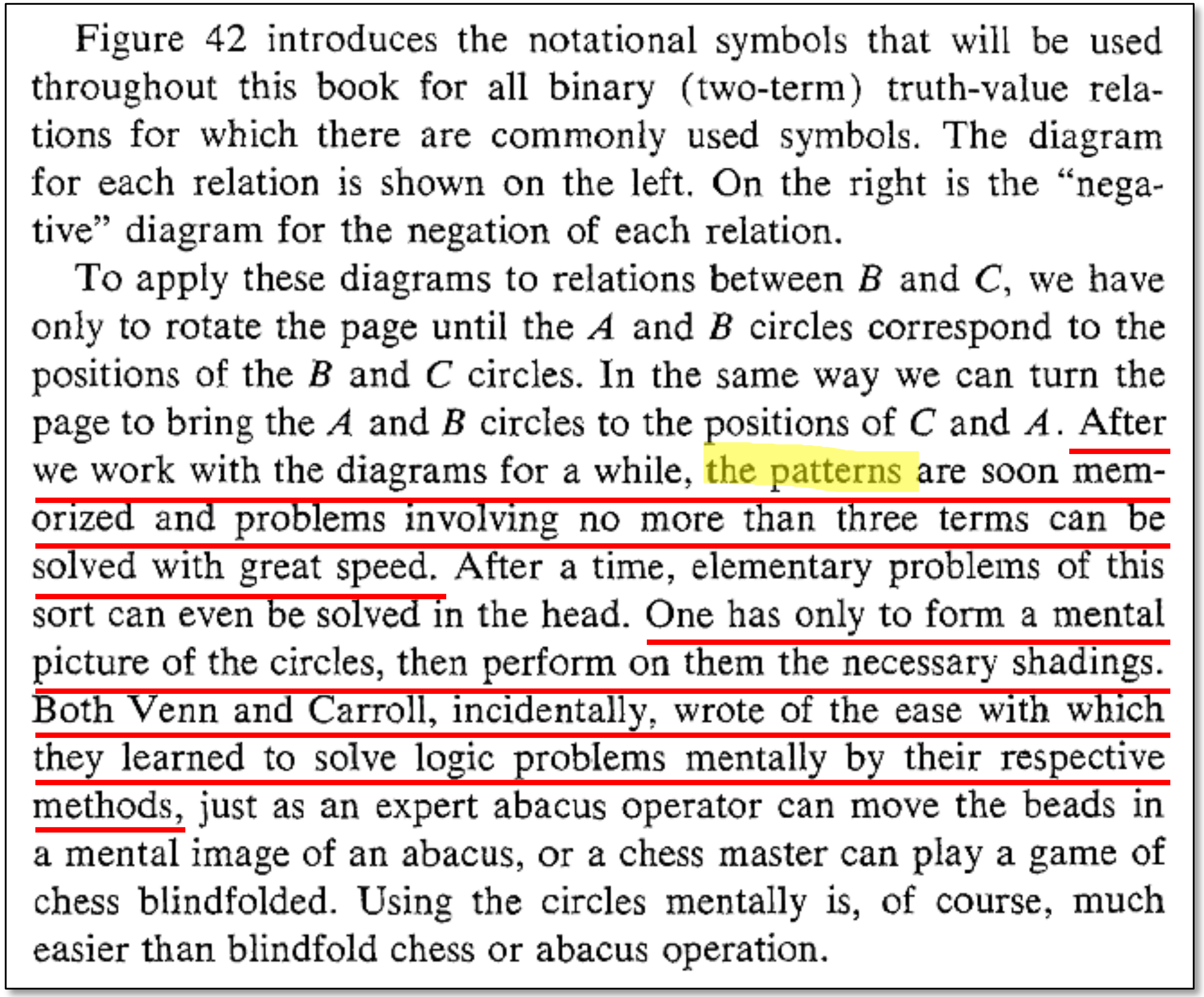
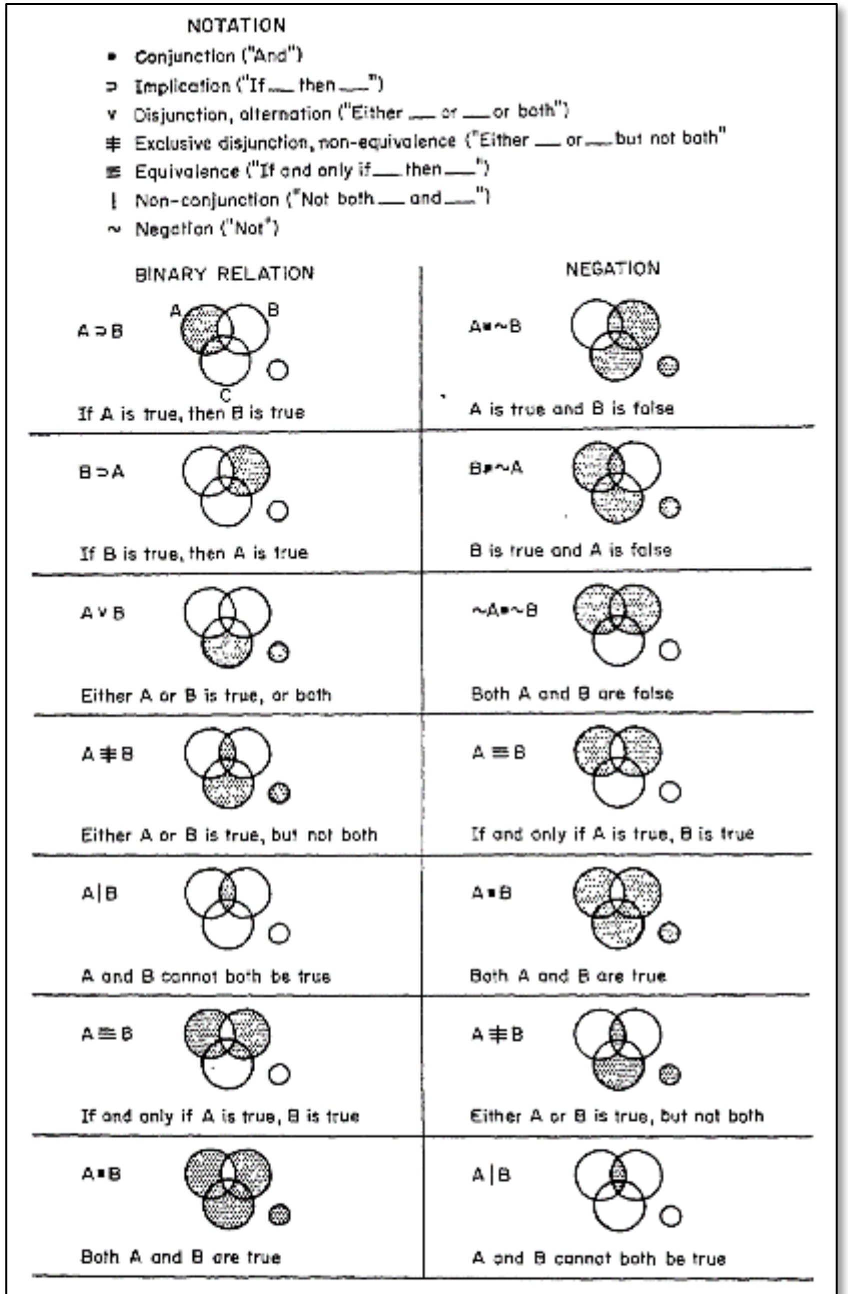


Figure 42 introduces the notational symbols that will be used throughout this book for all binary (two-term) truth-value relations for which there are commonly used symbols. The diagram for each relation is shown on the left. On the right is the "negative" diagram for the negation of each relation.

To apply these diagrams to relations between *B* and *C*, we have only to rotate the page until the *A* and *B* circles correspond to the positions of the *B* and *C* circles. In the same way we can turn the page to bring the *A* and *B* circles to the positions of *C* and *A*. After we work with the diagrams for a while, the patterns are soon memorized and problems involving no more than three terms can be solved with great speed. After a time, elementary problems of this sort can even be solved in the head. One has only to form a mental picture of the circles, then perform on them the necessary shadings. Both Venn and Carroll, incidentally, wrote of the ease with which they learned to solve logic problems mentally by their respective methods, just as an expert abacus operator can move the beads in a mental image of an abacus, or a chess master can play a game of chess blindfolded. Using the circles mentally is, of course, much easier than blindfold chess or abacus operation.



NOTATION

- • Conjunction ("And")
- ⊃ Implication ("If __ then __")
- ∨ Disjunction, alternation ("Either __ or __ or both")
- ≢ Exclusive disjunction, non-equivalence ("Either __ or __ but not both"
- ≡ Equivalence ("If and only if __ then __")
- | Non-conjunction ("Not both __ and __")
- ~ Negation ("Not")

Figure 42.

Gardner. Logic machines and diagrams, 1958. https://archive.org/details/logicmachinesdia00mart

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

4. Patterns

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

	some	**not any**	**not all**	**all**
Sailors renting boats	have reserved some red boat	have not reserved any red boat	reserved not all red boats	reserved all red boats

Example taken from: https://relationaldiagrams.com/

4. Patterns

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

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

	some	not any	not all	all
Sailors renting boats	have reserved some red boat	have not reserved any red boat	reserved not all red boats	reserved all red boats
Students taking classes	took some art class	took no art class	took not all art classes	took all art classes

Example taken from: https://relationaldiagrams.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

4. Patterns

Actor (id, lname)
Play (aid, mid, role)
Movie (id, name, dir)

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

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

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

Example taken from: https://relationaldiagrams.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

4. Patterns

Actor (id, lname)
Play (aid, mid, role)
Movie (id, name, dir)

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

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

Sailors

some

```sql
SELECT DISTINCT S.sname
FROM Sailor S
WHERE EXISTS(
   SELECT *
   FROM Reserves R
   WHERE R.sid = S.sid
   AND EXISTS(
      SELECT *
      FROM Boat B
      WHERE B.color = 'red'
      AND B.bid = R.bid))
```

not any

```sql
SELECT DISTINCT S.sname
FROM Sailor S
WHERE 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))
```

not all

```sql
SELECT DISTINCT S.sname
FROM Sailor S
WHERE NOT EXISTS(
   SELECT *
   FROM Reserves R
   WHERE R.sid = S.sid
   AND EXISTS(
      SELECT *
      FROM Boat B
      WHERE B.color = 'red'
      AND B.bid = R.bid))
```

all

```sql
SELECT DISTINCT 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

some

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE EXISTS(
   SELECT *
   FROM Takes T
   WHERE T.sid = S.sid
   AND EXISTS(
      SELECT *
      FROM Class C
      WHERE C.depart = 'art'
      AND C.cid = T.cid))
```

not any

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE EXISTS(
   SELECT *
   FROM Class C
   WHERE C.depart = 'art'
   AND NOT EXISTS(
      SELECT *
      FROM Takes T
      WHERE T.cid = C.cid
      AND T.sid = S.sid))
```

not all

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE NOT EXISTS(
   SELECT *
   FROM Takes T
   WHERE T.sid = S.sid
   AND EXISTS(
      SELECT *
      FROM Class C
      WHERE C.depart = 'art'
      AND C.cid = T.cid))
```

all

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE NOT EXISTS(
   SELECT *
   FROM Class C
   WHERE C.depart = 'art'
   AND NOT EXISTS(
      SELECT *
      FROM Takes T
      WHERE T.cid = C.cid
      AND T.sid = S.sid))
```

Actors

some

```sql
SELECT DISTINCT A.lname
FROM Actor A
WHERE EXISTS(
   SELECT *
   FROM Play P
   WHERE P.aid = A.aid
   AND EXISTS(
      SELECT *
      FROM Movie M
      WHERE M.dir = 'Hitchcock'
      AND M.id = P.mid))
```

not any

```sql
SELECT DISTINCT A.lname
FROM Actor A
WHERE EXISTS(
   SELECT *
   FROM Movie M
   WHERE M.dir = 'Hitchcock'
   AND NOT EXISTS(
      SELECT *
      FROM Play P
      WHERE P.mid = M.id
      AND P.aid = A.aid))
```

not all

```sql
SELECT DISTINCT A.lname
FROM Actor A
WHERE NOT EXISTS(
   SELECT *
   FROM Play P
   WHERE P.aid = A.aid
   AND EXISTS(
      SELECT *
      FROM Movie M
      WHERE M.dir = 'Hitchcock'
      AND M.id = P.mid))
```

all

```sql
SELECT DISTINCT A.lname
FROM Actor A
WHERE NOT EXISTS(
   SELECT *
   FROM Movie M
   WHERE M.dir = 'Hitchcock'
   AND NOT EXISTS(
      SELECT *
      FROM Play P
      WHERE P.mid = M.id
      AND P.aid = A.aid))
```

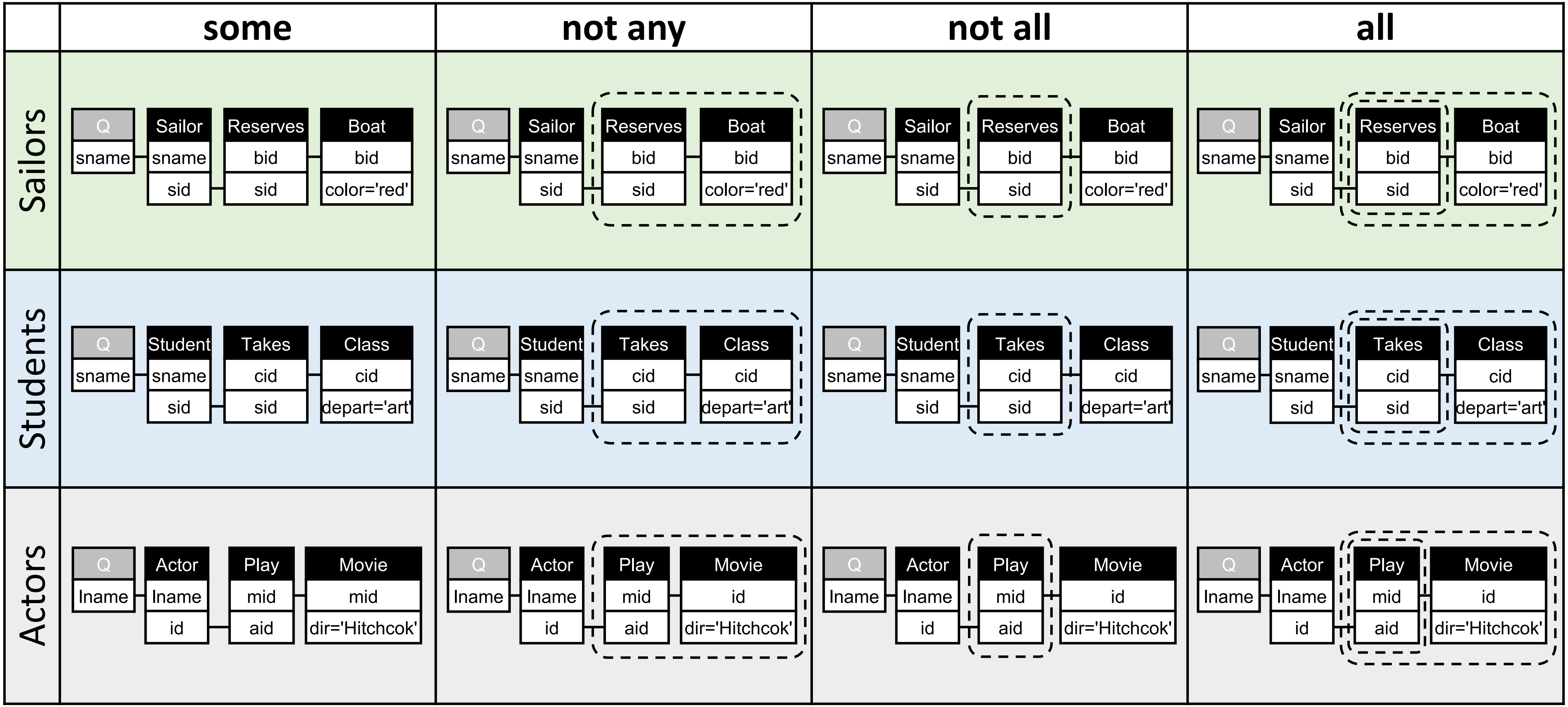
Example taken from: https://relationaldiagrams.com/ , Database to run Sailor SQL available as schema 300 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

4. Patterns

Actor (<u>id</u>, lname)
Play (<u>aid, mid, role</u>)
Movie (<u>id</u>, name, dir)

Student (<u>sid</u>, sname)
Takes (<u>sid, cid, semester</u>)
Course (<u>cid</u>, cname, depart)

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

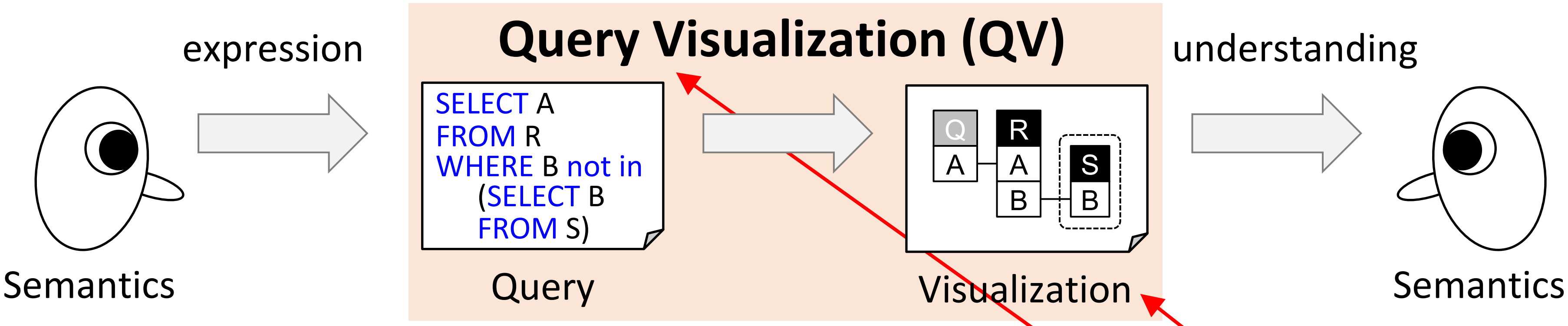


Example taken from: https://relationaldiagrams.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Don't we already have visual query languages, and interactive query builders?

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



Query Visualization (QV)

```
SELECT A
FROM R
WHERE B not in
      (SELECT B
      FROM S)
```

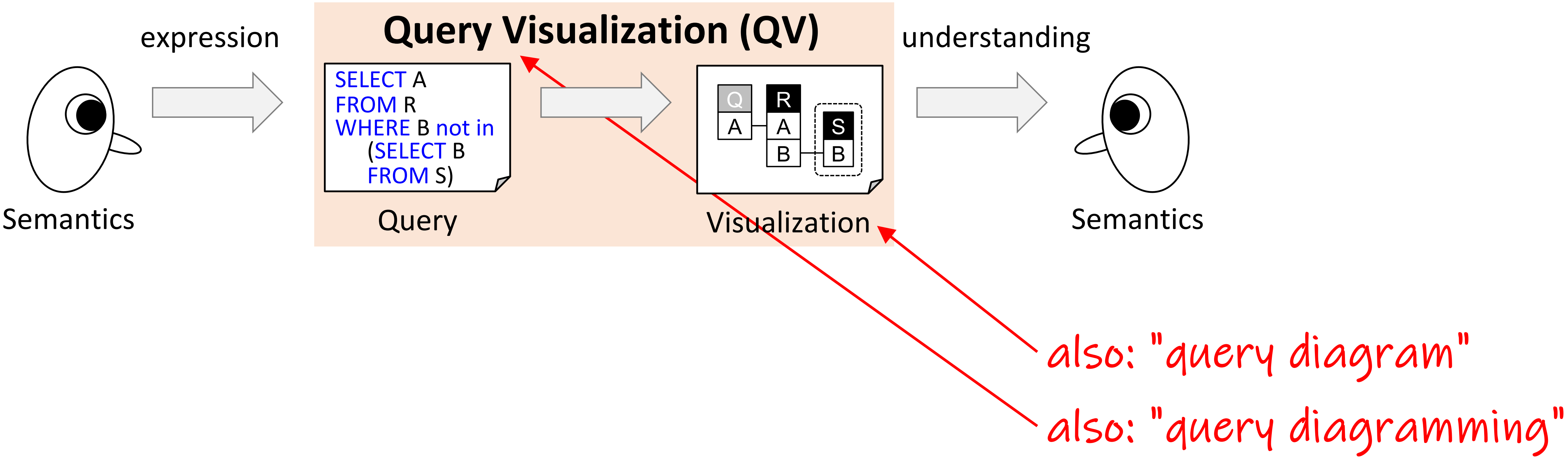
Query

Visualization

also: "query diagram"

also: "query diagramming"

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



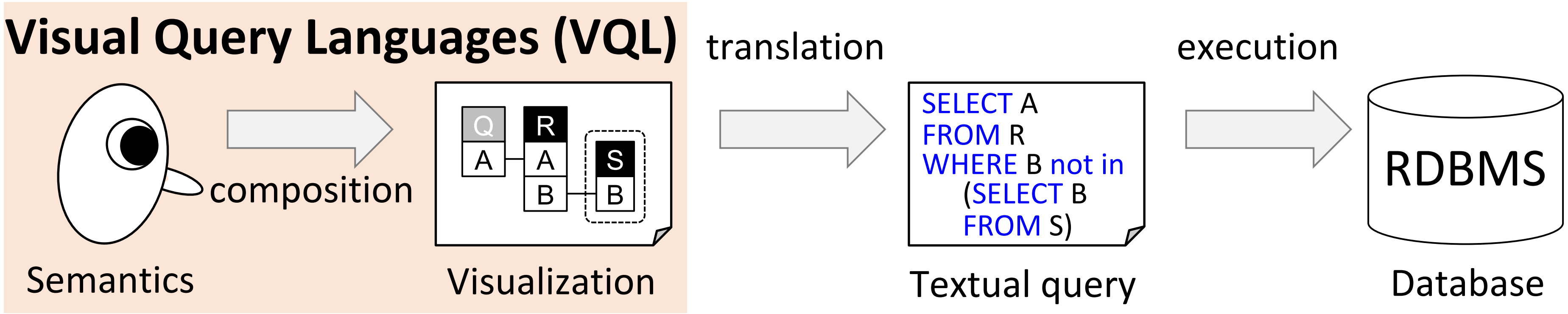
Query Visualization (QV)

Semantics → expression → Query → Visualization → understanding → Semantics

```
SELECT A
FROM R
WHERE B not in
    (SELECT B
    FROM S)
```

also: "query diagram"

also: "query diagramming"

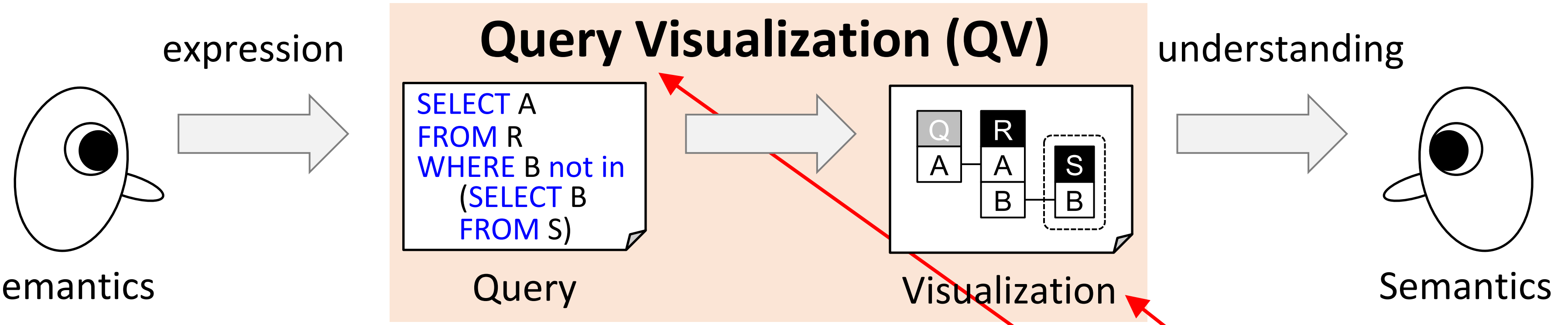


Visual Query Languages (VQL)

Semantics → composition → Visualization → translation → Textual query → execution → Database

```
SELECT A
FROM R
WHERE B not in
    (SELECT B
    FROM S)
```

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



Query Visualization (QV)

expression

understanding

```
SELECT A
FROM R
WHERE B not in
      (SELECT B
      FROM S)
```

Semantics

Query

Visualization

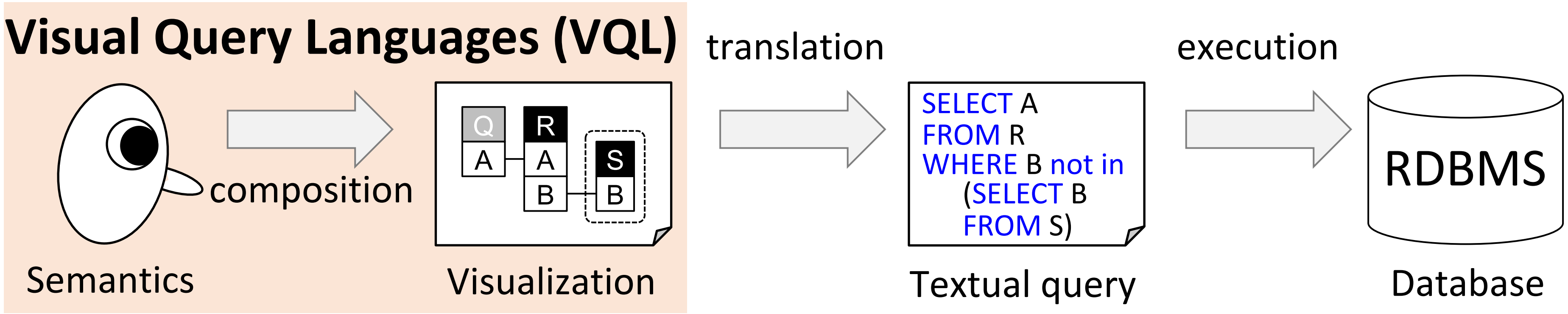
Semantics

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

also: "query diagram"

also: "query diagramming"

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



Visual Query Languages (VQL)

composition

translation

execution

```
SELECT A
FROM R
WHERE B not in
      (SELECT B
      FROM S)
```

RDBMS

Semantics

Visualization

Textual query

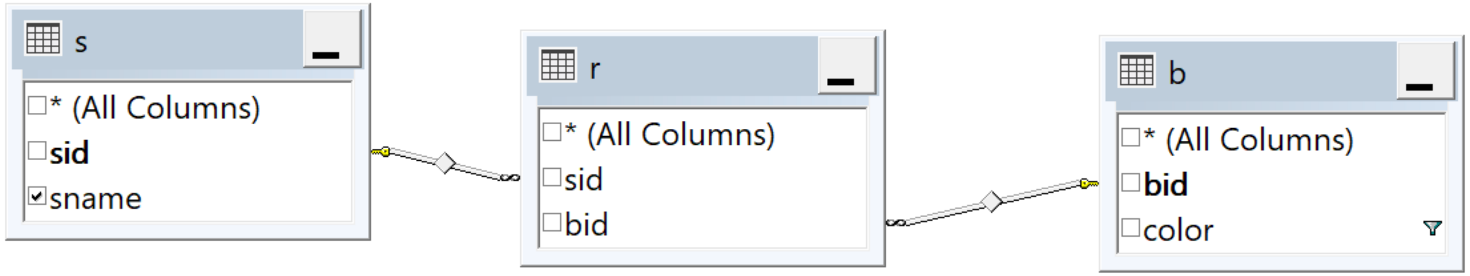
Database

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

What about visual active query builders? Say SSMS?

Q2: "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'
```



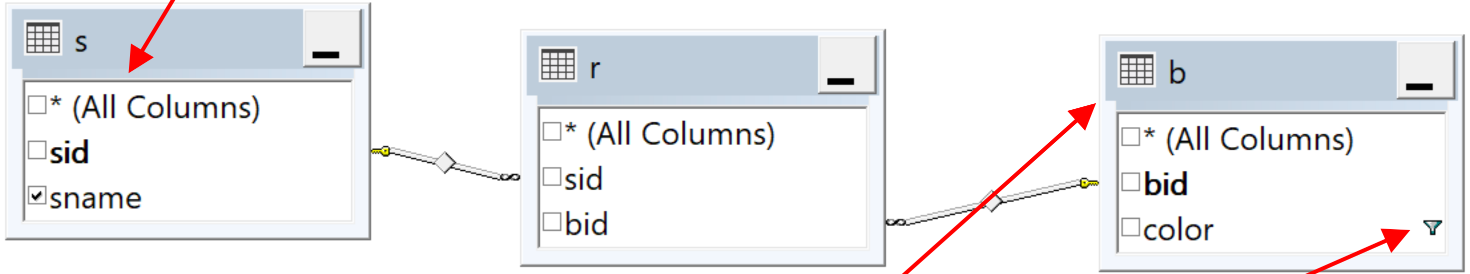
Screenshot acknowledgement: Jiahui Zhang (4/2020), SSMS stands for SQL Server Management Studio (https://en.wikipedia.org/wiki/SQL_Server_Management_Studio)

What about visual active query builders? Say SSMS?

Q2: "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



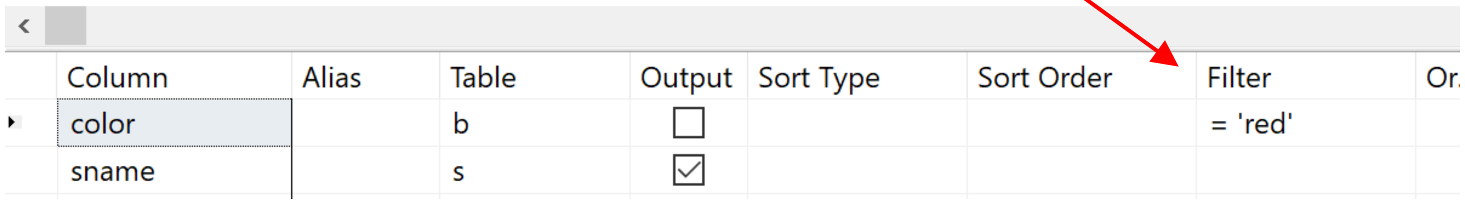
s
☐* (All Columns)
☐**sid**
☑sname

r
☐* (All Columns)
☐sid
☐bid

b
☐* (All Columns)
☐**bid**
☐color

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	☑				

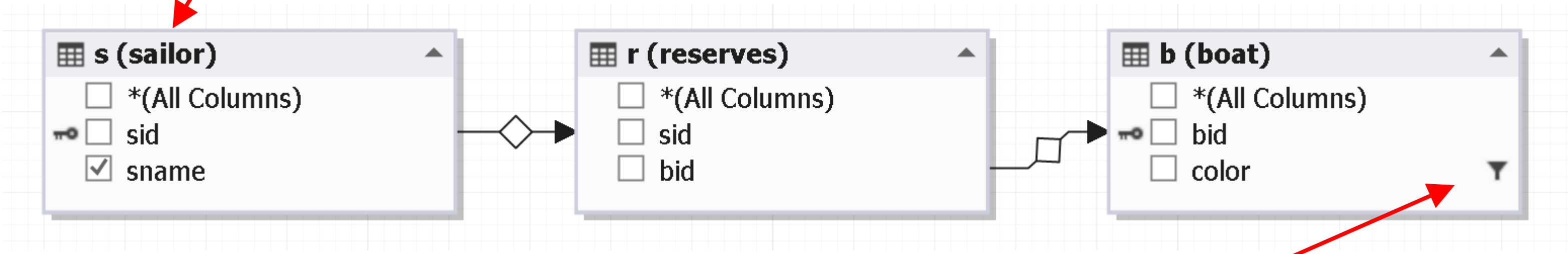
Screenshot acknowledgement: Jiahui Zhang (4/2020), SSMS stands for SQL Server Management Studio (https://en.wikipedia.org/wiki/SQL_Server_Management_Studio)

What about visual active query builders? Or dbForge?

Q2: "Find sailors who reserved a red boat."

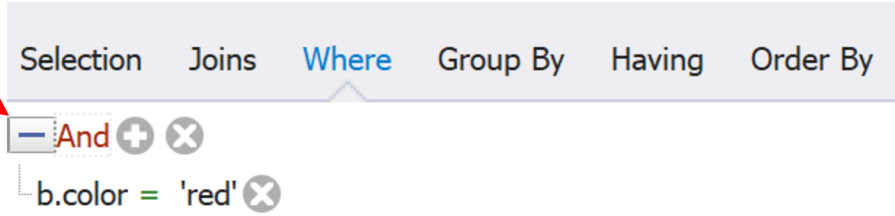
```
select S.sname
from Sailor s, Reserves r, Boat b
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

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



b.color = 'red'

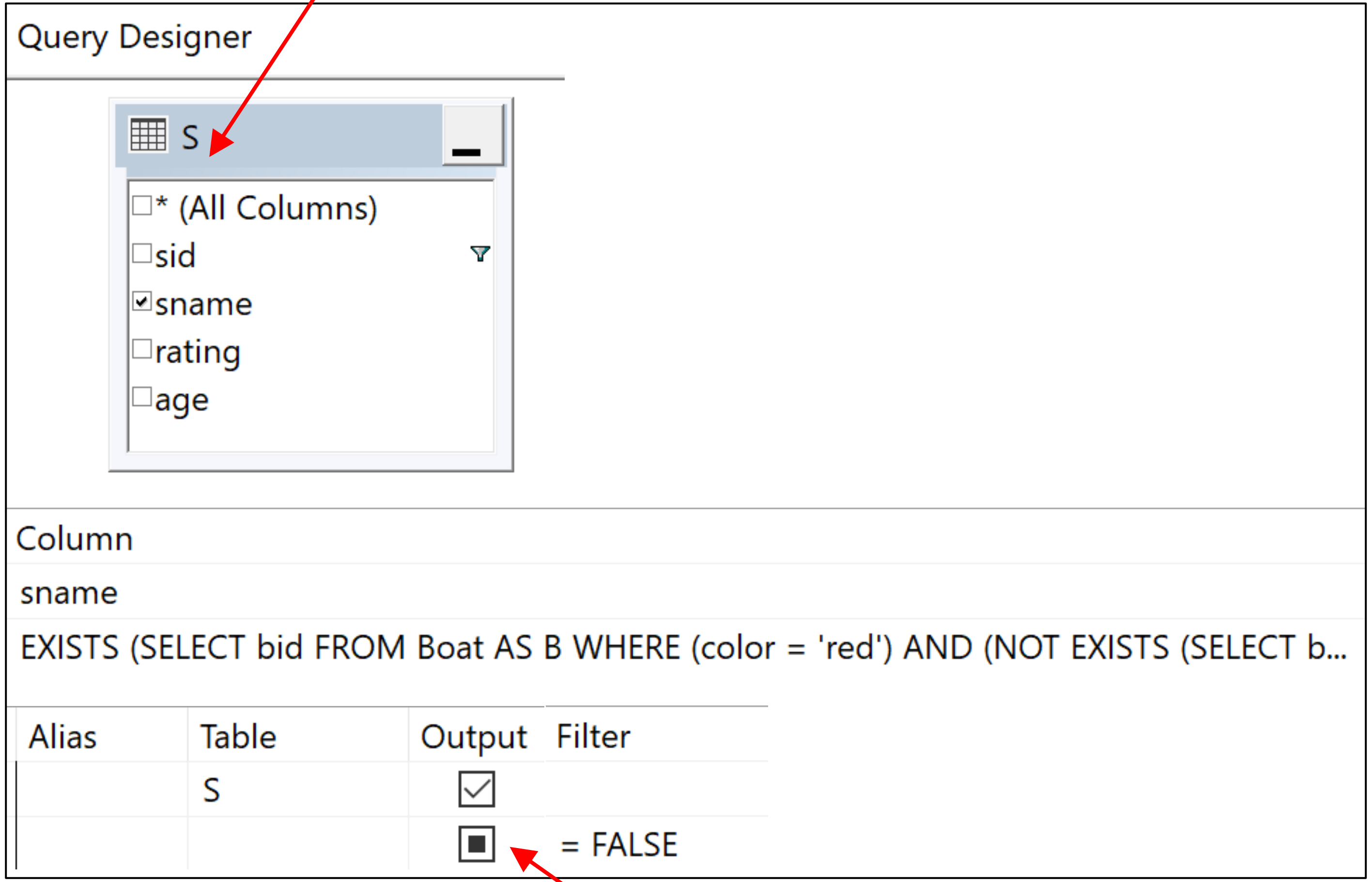
Screenshot acknowledgement: Jiahui Zhang (4/2020), dbForge: https://www.devart.com/dbforge/

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

Q4: "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))
```

SSMS does not render the nested blocks



Query Designer

S

☐ * (All Columns)
☐ sid
☑ sname
☐ rating
☐ age

Column
sname
EXISTS (SELECT bid FROM Boat AS B WHERE (color = 'red') AND (NOT EXISTS (SELECT b...

Alias	Table	Output	Filter
	S	☑	
		■	= FALSE

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

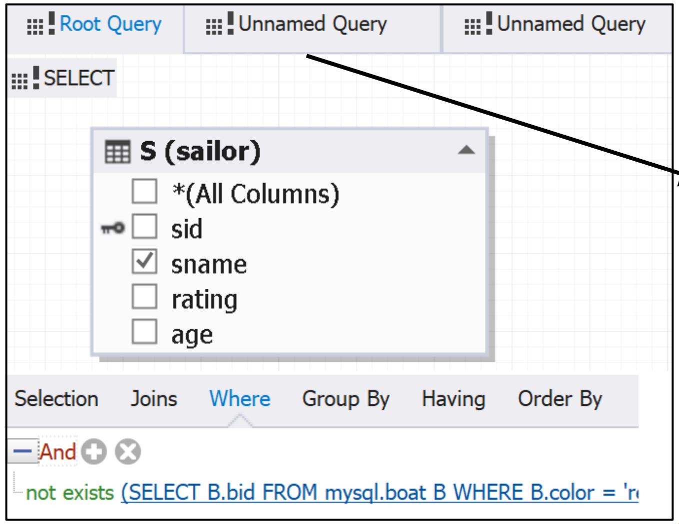
Screenshot acknowledgement: Jiahui Zhang (4/2020), SSMS stands for SQL Server Management Studio (https://en.wikipedia.org/wiki/SQL_Server_Management_Studio)

What about visual active query builders? dbForge...

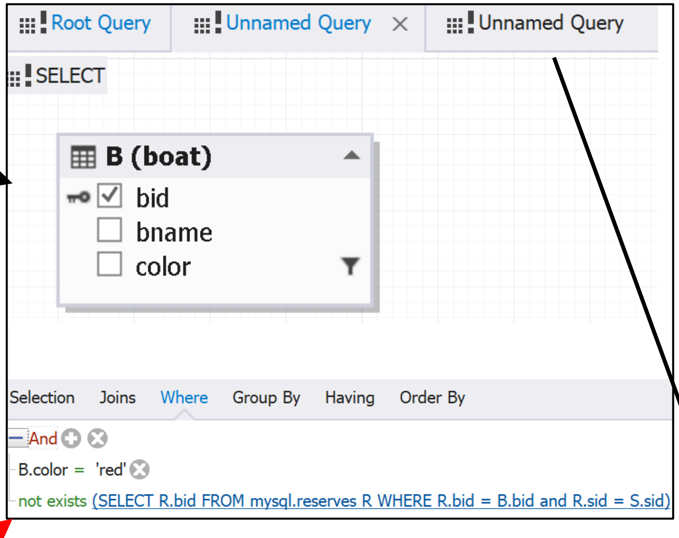
Q4: "Find sailors who reserved all red boats."

```
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))
```

Individual query blocks shown in separate windows



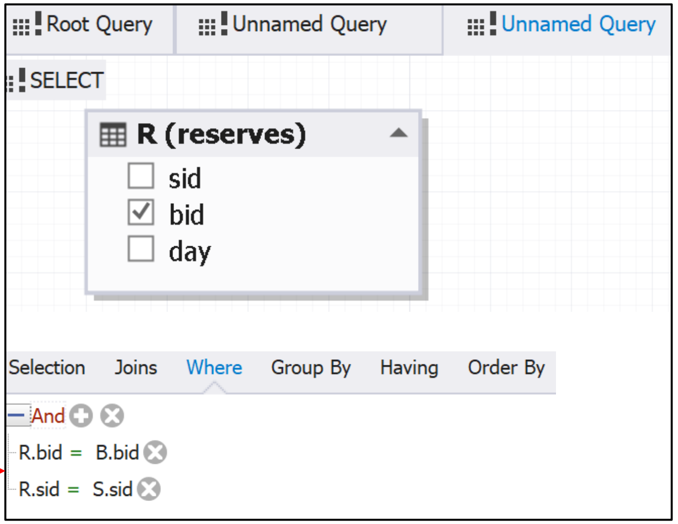
not exists (SELECT B.bid FROM mysql.boat B WHERE B.color = 'r



B.color = 'red'

not exists (SELECT R.bid FROM mysql.reserves R WHERE R.bid = B.bid and R.sid = S.sid)

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



R.bid = B.bid

R.sid = S.sid

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

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 (from DB community)
6. Lessons Learned and 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

symbolic, linguistic, linear, sentential

DIAGRAMMATIC

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

The exact boundary b/w text and "visual" is not clear-cut!

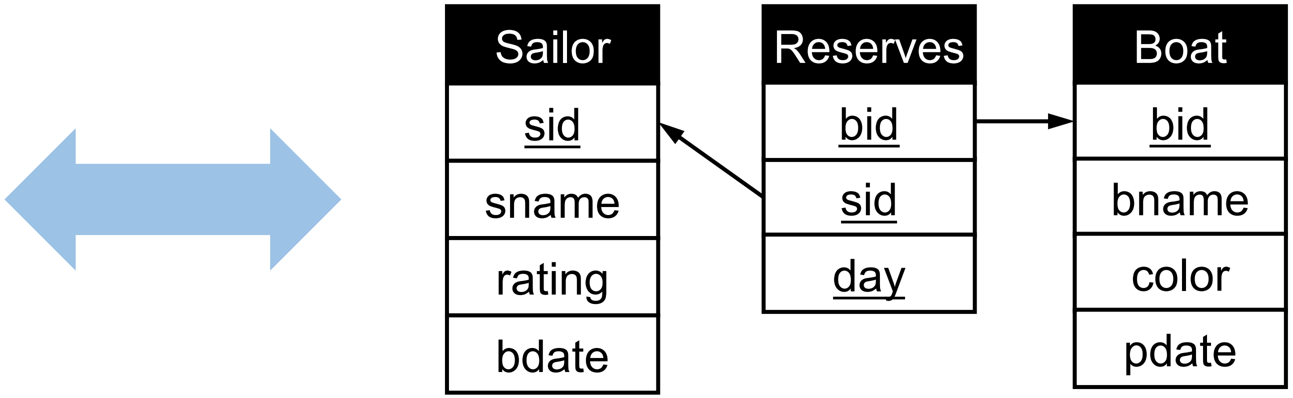
(And as we will see, "visual" gets interpreted very differently)

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

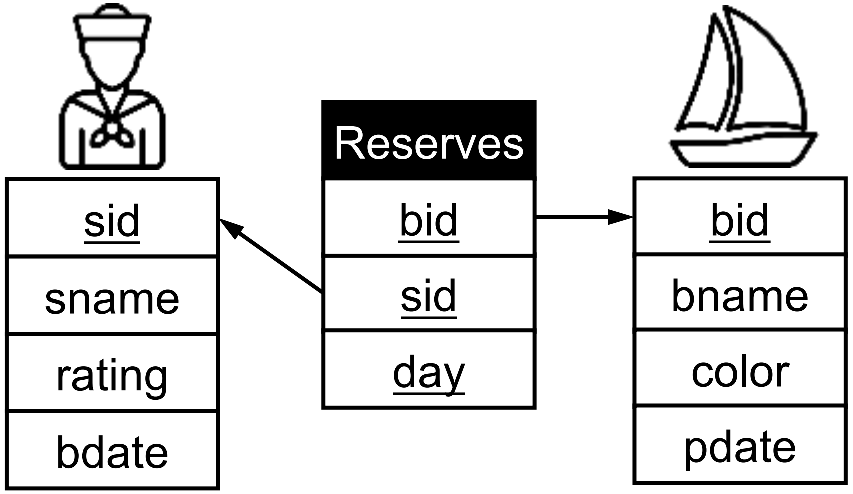


UML diagram of
relational schema.

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



sid
sname
rating
bdate

Reserves
bid
sid
day

bid
bname
color
pdate

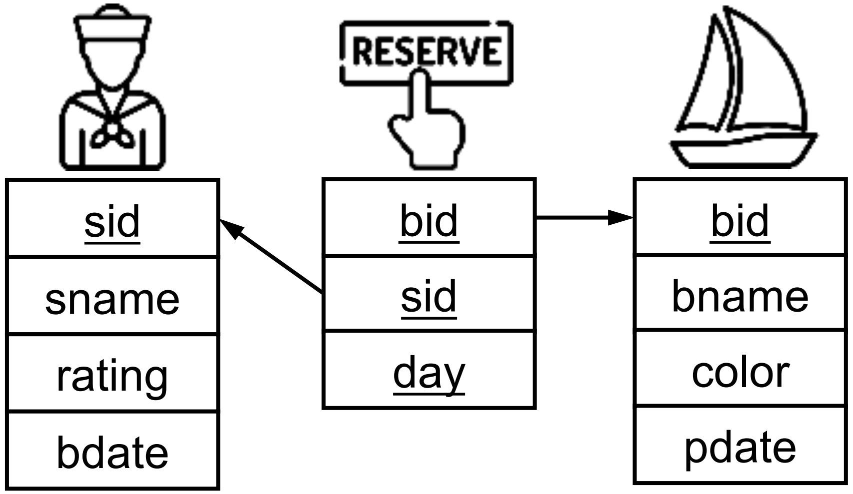
UML diagram of
relational schema.

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.

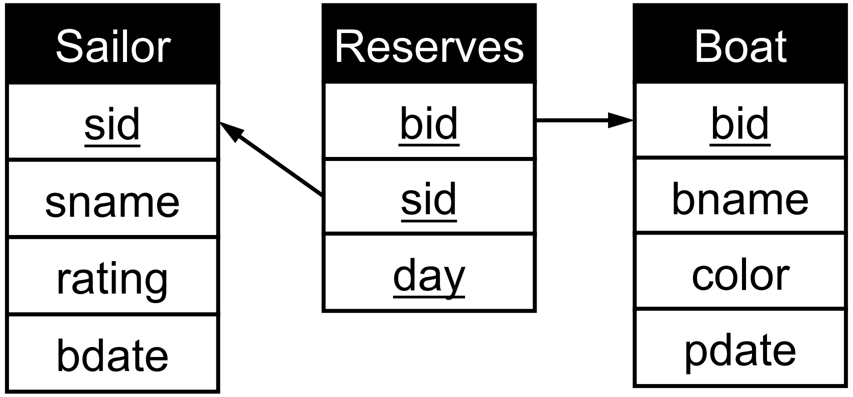
But no need for icons!
(Actually which icons?)

Icons by flaticon.com

Let's explore the boundary between text and diagram

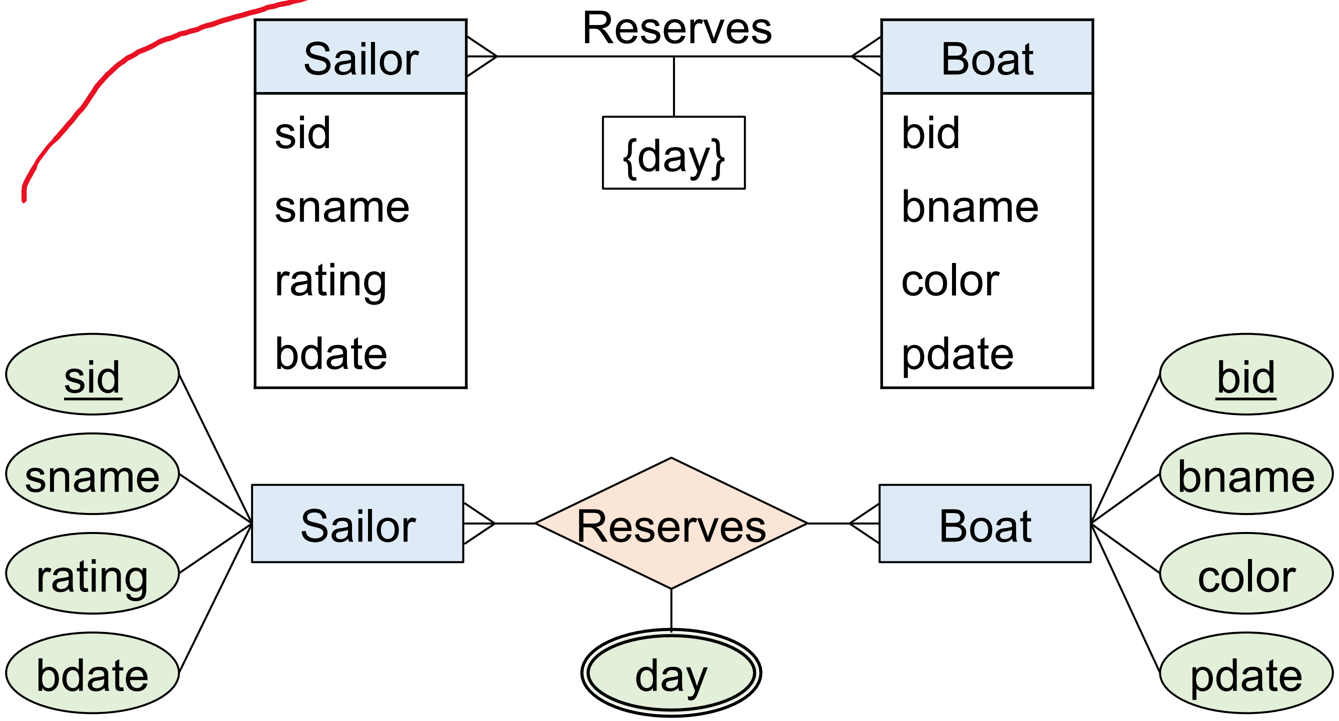
Sailor(sid, sname, rating, bdate)
Reserves(sid, bid, day)
Boat(bid, bname, color, pdate)

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



UML diagram of relational schema.

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

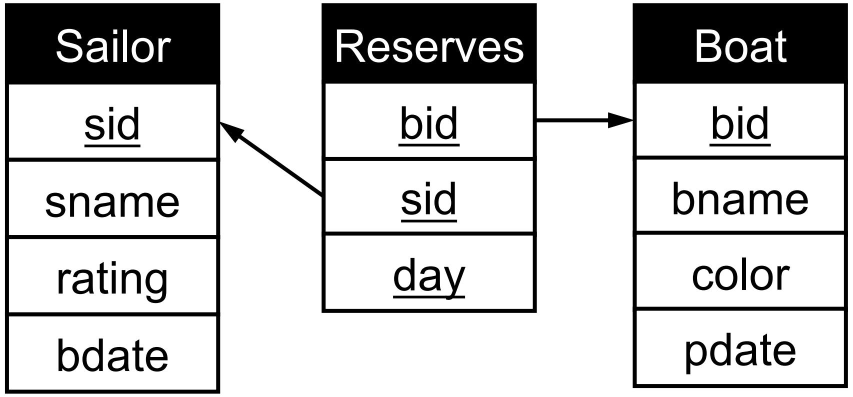


ER diagrams also use text

Text vs diagrammatic representations

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

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

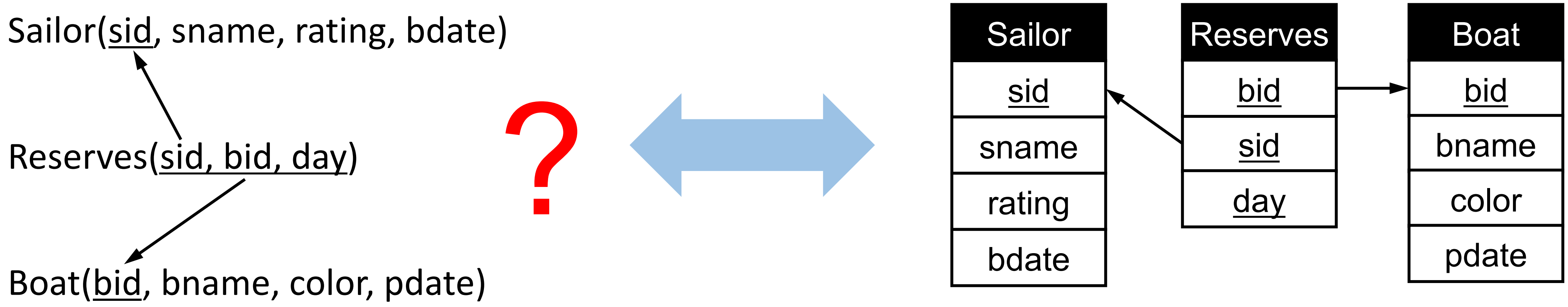


Observations:

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(sid, sname, rating, bdate)

Reserves(sid, bid, day)

Boat(bid, bname, color, pdate)

?

Sailor
sid
sname
rating
bdate

Reserves
bid
sid
day

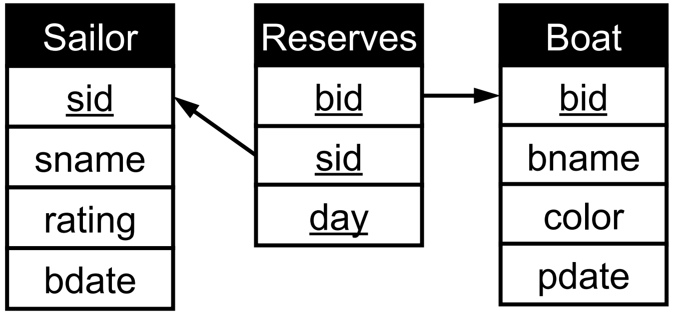
Boat
bid
bname
color
pdate

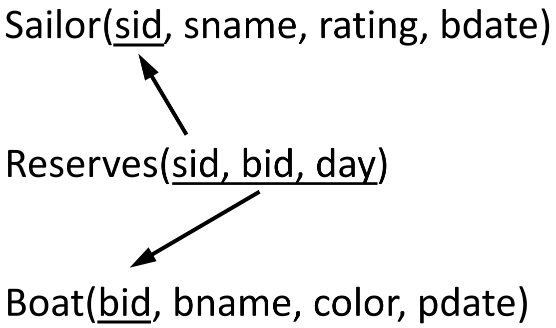
Observations:

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

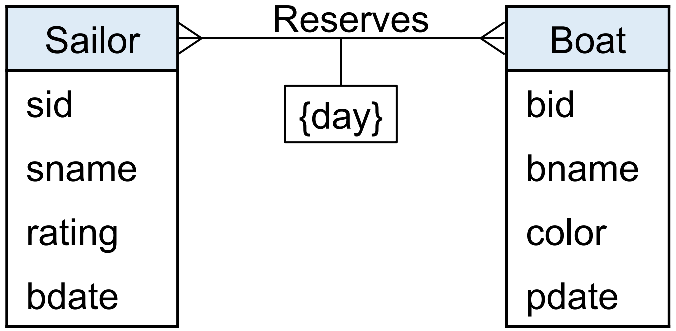


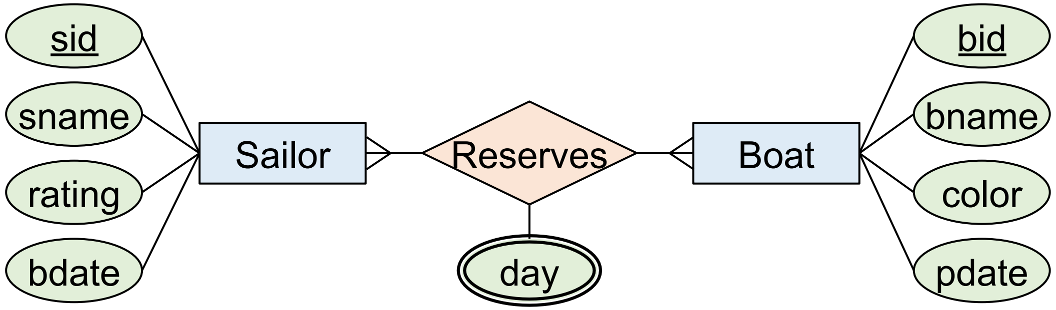


Sailor(sid, sname, rating, bdate)

Reserves(sid, bid, day)

Boat(bid, bname, color, pdate)





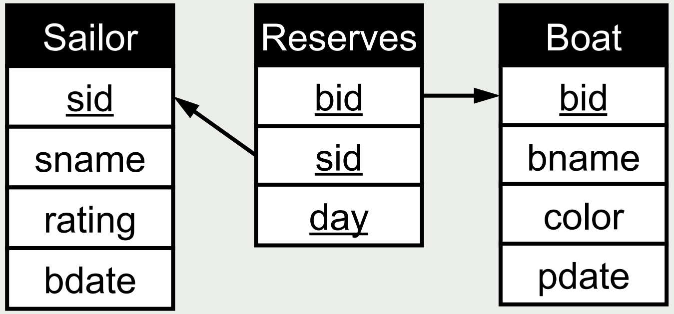
1. We prefer text as names/labels for atomic elements (tables, attributes)

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)

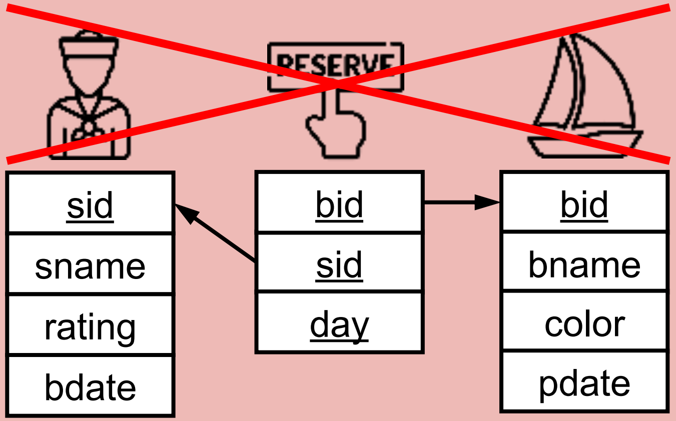
~~FK Reserves.sid references Sailor~~
~~FK Reserves.bid references Boat~~



Sailor
sid
sname
rating
bdate

Reserves
bid
sid
day

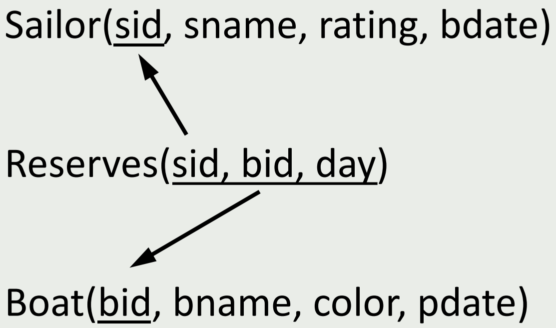
Boat
bid
bname
color
pdate



sid
sname
rating
bdate

bid
sid
day

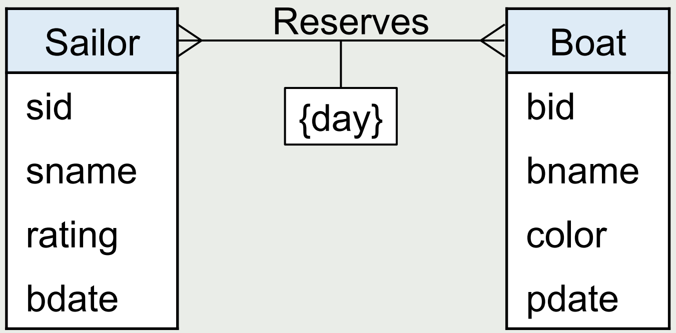
bid
bname
color
pdate



Sailor(sid, sname, rating, bdate)

Reserves(sid, bid, day)

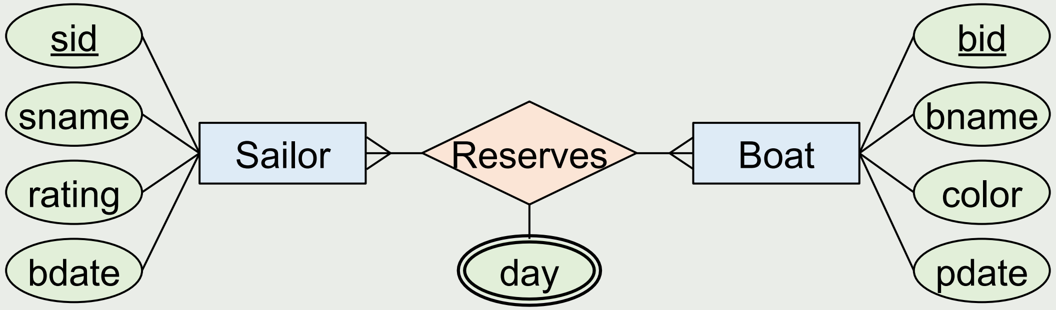
Boat(bid, bname, color, pdate)



Sailor
sid
sname
rating
bdate

Reserves
{day}

Boat
bid
bname
color
pdate

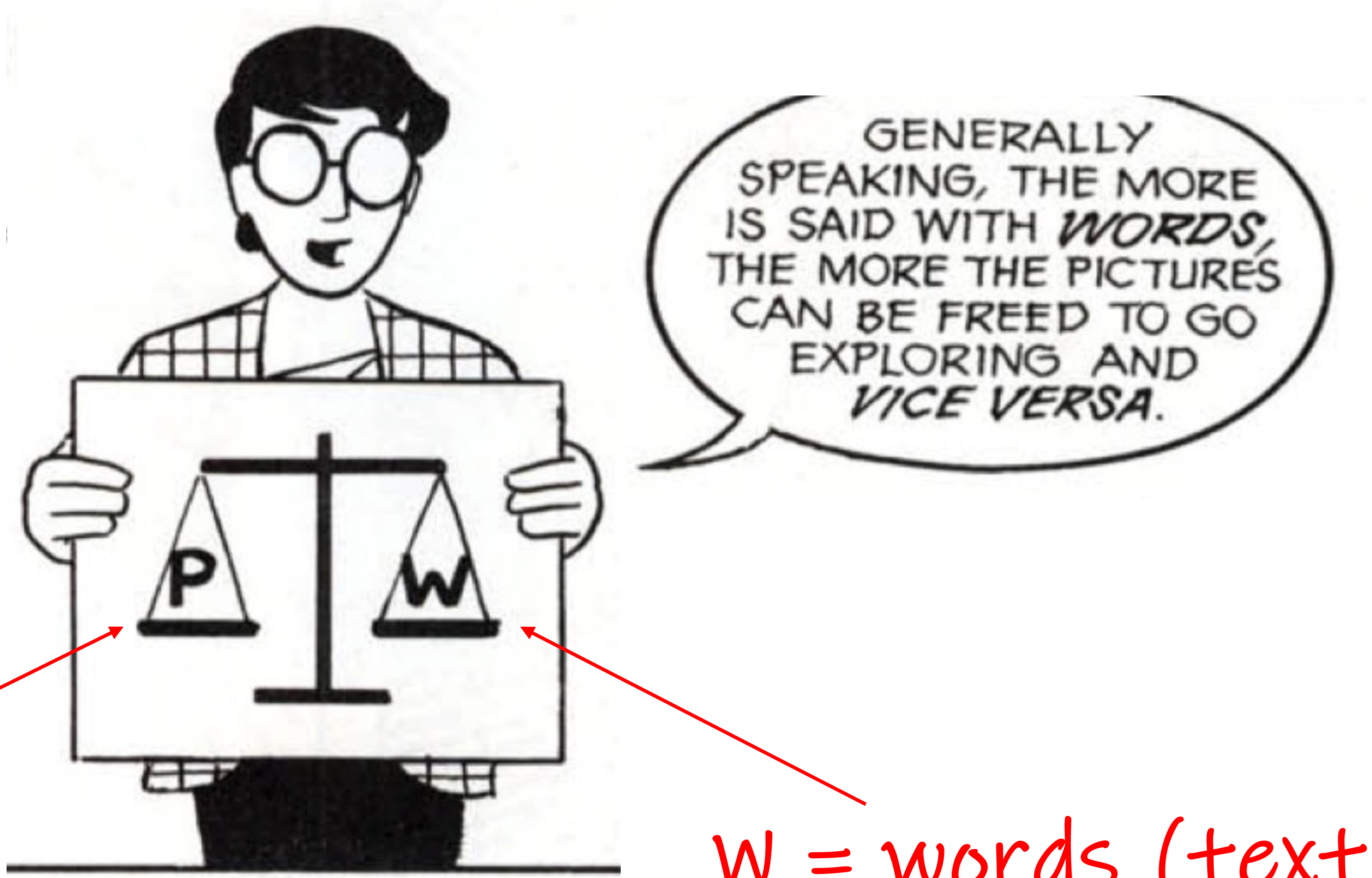


sid
sname
rating
bdate
Sailor
Reserves
day
Boat
bid
bname
color
pdate

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

What important "structures" (relationships) do we actually have in relational queries?

Answer: the <u>join structures</u>!

Textual vs diagrammatic representations

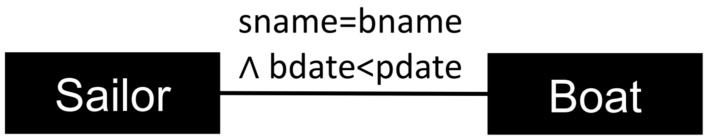
Schema

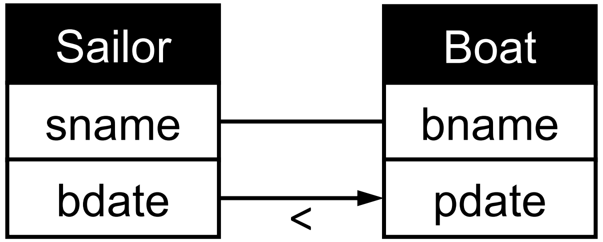
Sailor
sid
sname
rating
bdate

Boat
bid
bname
color
pdate

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

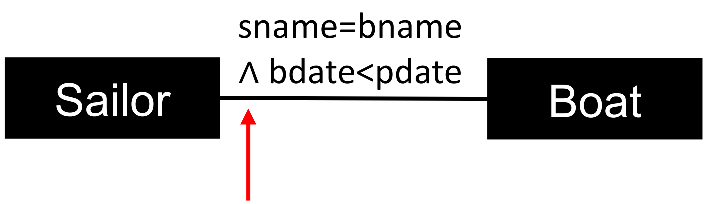
Schema

Sailor
sid
sname
rating
bdate

Boat
bid
bname
color
pdate

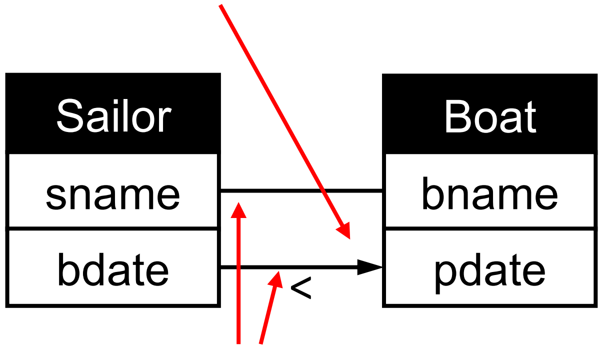
Q: "There is a sailor with the same name as a boat,
and that boat was purchased before the sailor was born."

```
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"

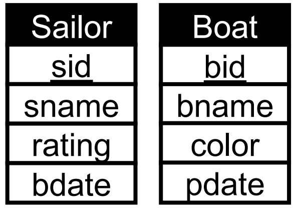


juxtaposition of conjunctive information
(we perceive them independently)

Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Textual vs diagrammatic representations

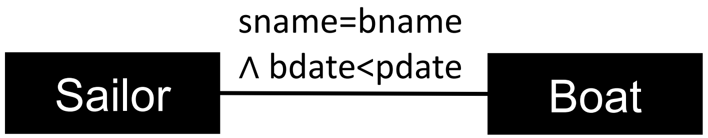
Schema

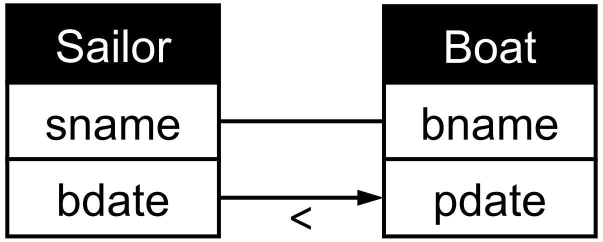

Sailor
sid
sname
rating
bdate

Boat
bid
bname
color
pdate

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

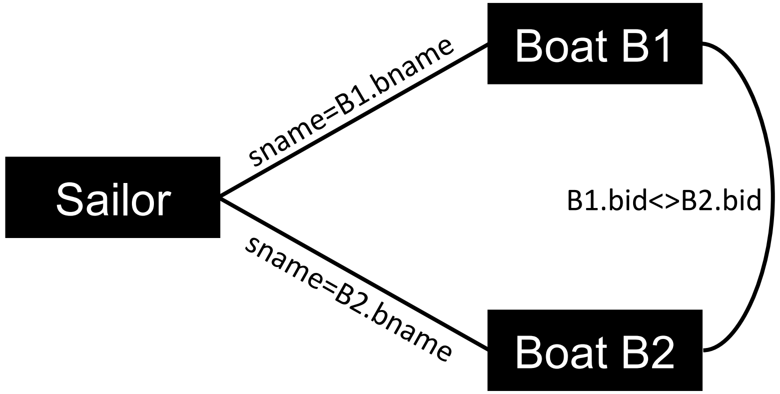
```
select exists
  (select *
  from Sailor, Boat
  where sname=bname
  and bdate<pdate)
```

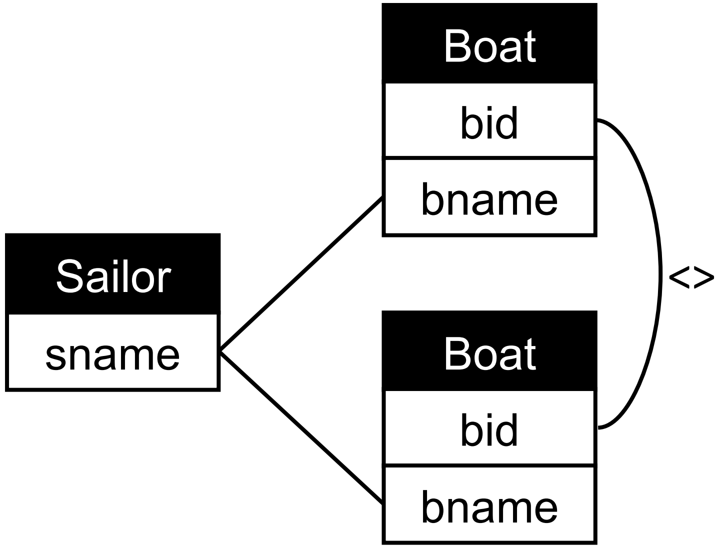

Sailor
sname=bname
∧ bdate<pdate
Boat


Sailor
sname
bdate
Boat
bname
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)
```


Sailor
sname=B1.bname
Boat B1
sname=B2.bname
Boat B2
B1.bid<>B2.bid

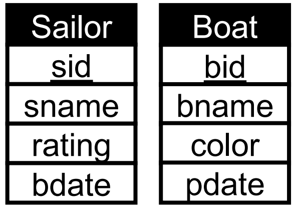

Sailor
sname
Boat
bid
bname
Boat
bid
bname
<>

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

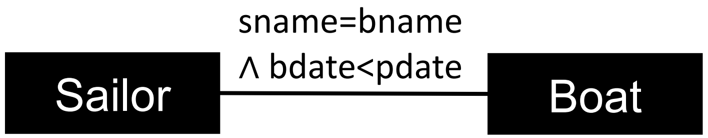
Textual vs diagrammatic representations

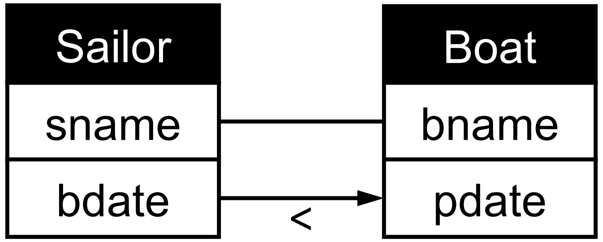
Schema



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)
```



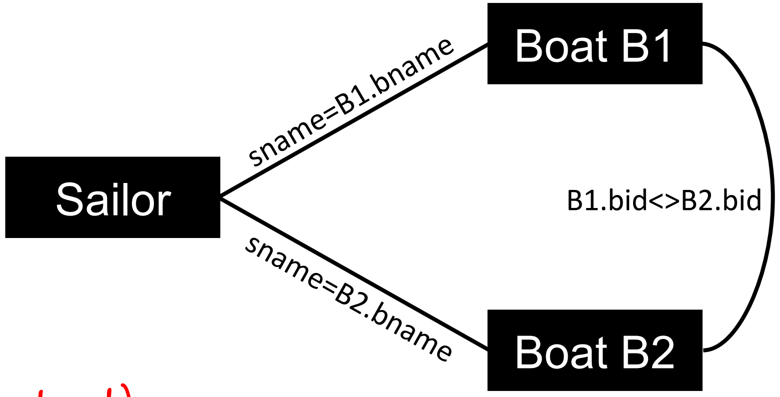


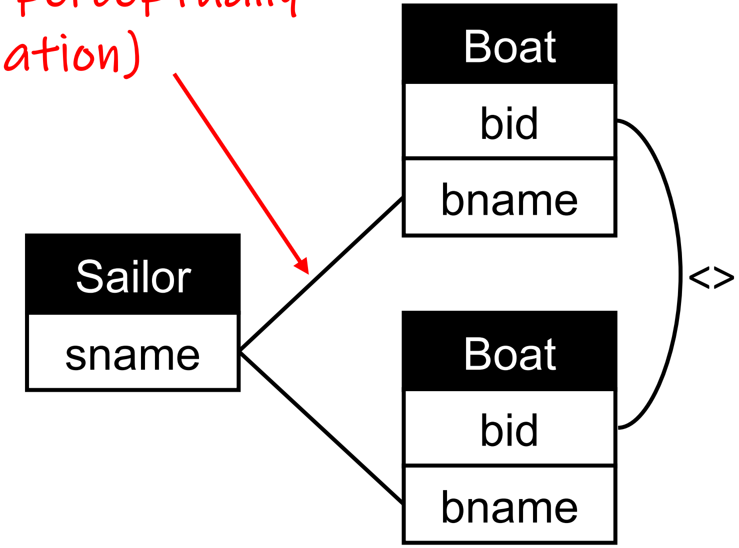
Q: "There is a sailor who shares the same name with 2 different boats."

```
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)

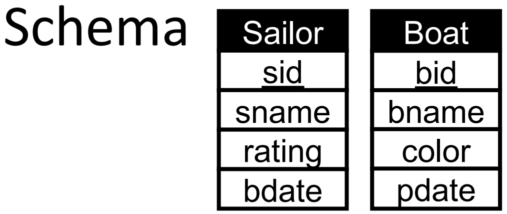




Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

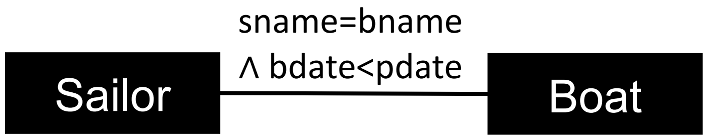
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

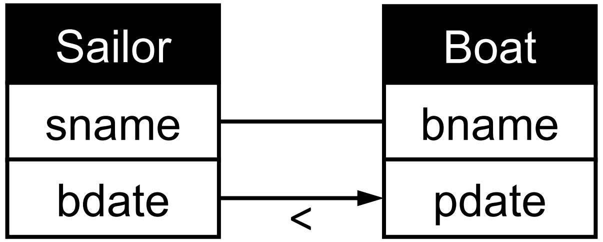
Textual vs diagrammatic representations



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)
```

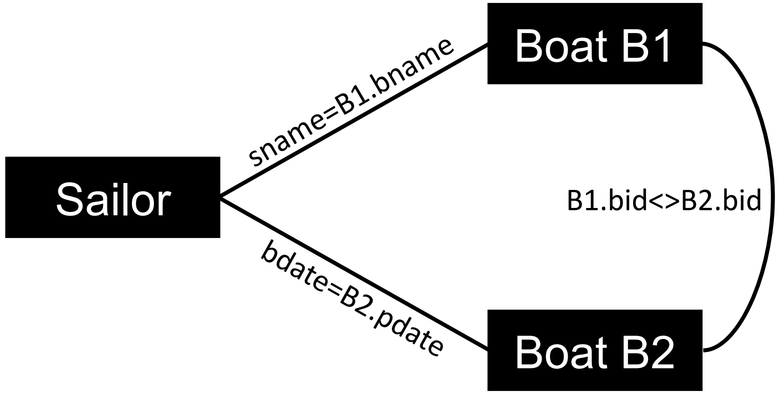




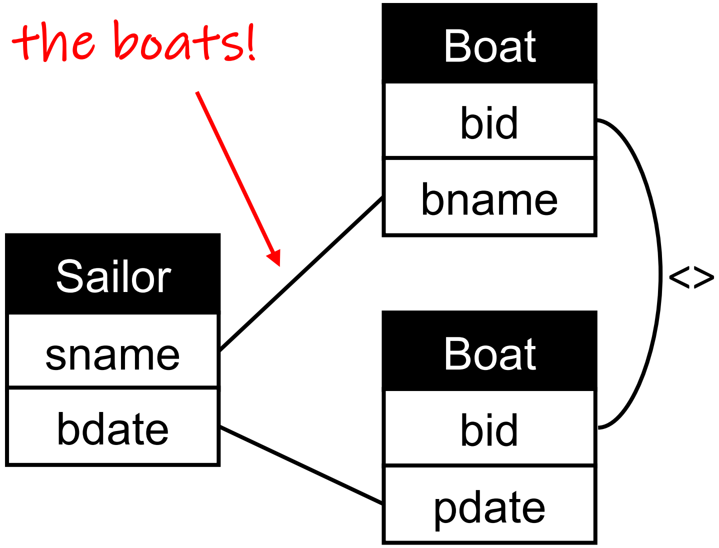
Q: "There is a sailor who shares the name with one boat and the birthday with the purchase day of another boat."

```
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!



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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

"Textual" vs "Diagrammatic" Representations

"structure" / "relation" / "relationship" = join structure !

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]

$$\Delta D \xrightarrow{\qquad\qquad\qquad} \Delta V$$

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

Algebraic Visualization Design [Kindlmann, Scheidegger 2014]

$$\begin{tikzcd} \Delta D \arrow[r] & \Delta V \end{tikzcd}$$

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 $\alpha(D) = D$

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data ⟶ Representation
⟶ 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

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

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data ⟶ Representation
⟶ 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

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

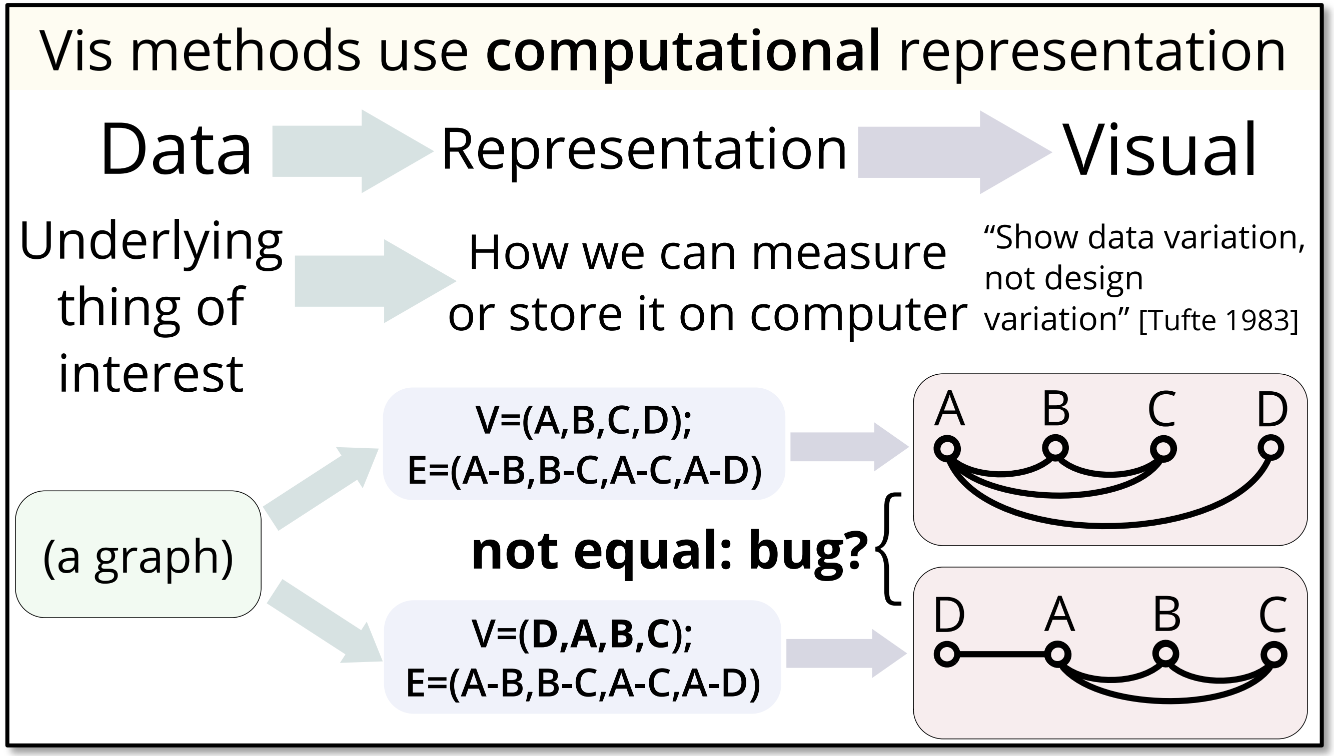
P1: **PRINCIPLE OF REPRESENTATION INVARIANCE**

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)

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325

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

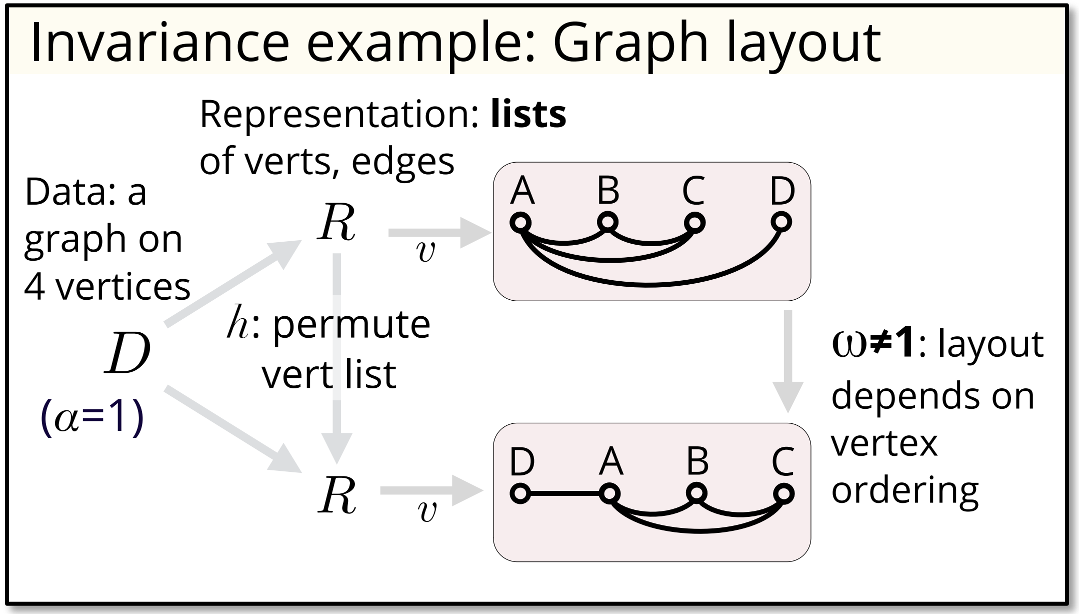
"Show data variation, not design variation" [Tufte 1983]

(a graph)

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

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

not equal: bug?



Invariance example: Graph layout

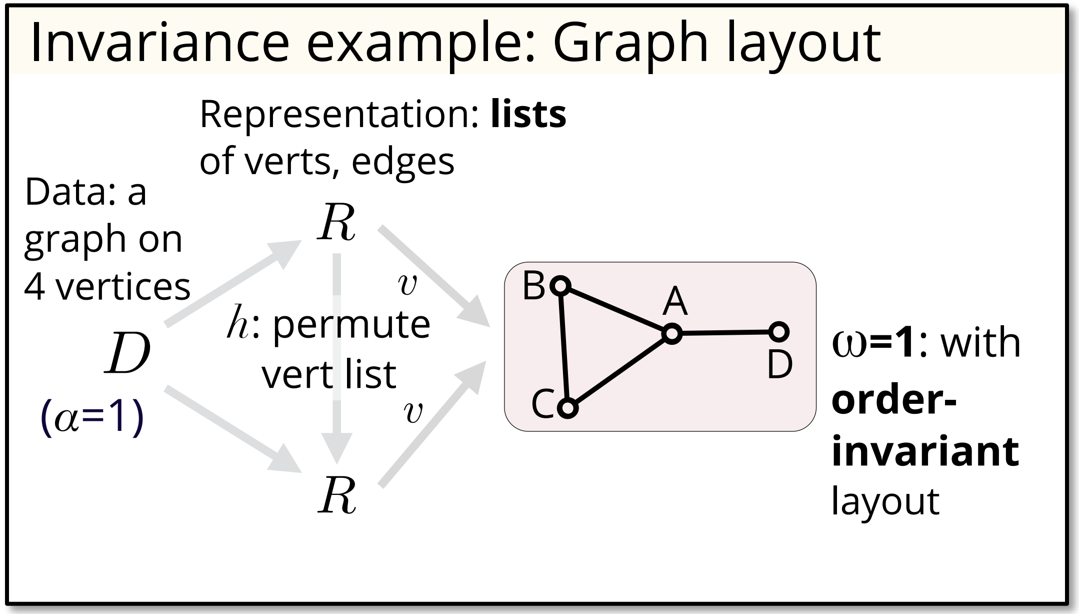
Representation: **lists** of verts, edges

Data: a graph on 4 vertices

D (α=1)

h: permute vert list

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)

h: permute vert list

ω=1: with **order-invariant** layout

Source: Slides by Kindlmann, 2016: Introduction to an Algebraic Process for Visualization Design. https://people.cs.uchicago.edu/~glk/talks/pdf/Kindlmann-IntroAlgebraicVis-Geilo-2016a.pdf

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3 algebraic visualization principles by [Kindlmann, Scheidegger 2014]

Data ⟶ Representation
⟶ Visualization

P1

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

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)

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325

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

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

P1: **PRINCIPLE OF REPRESENTATION INVARIANCE**

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

→ 4 bijection principles for Query Visualization

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." ($\omega = I_V \Rightarrow \alpha = I_D$)

→ 7 correspondence principles for Query Visualiz.

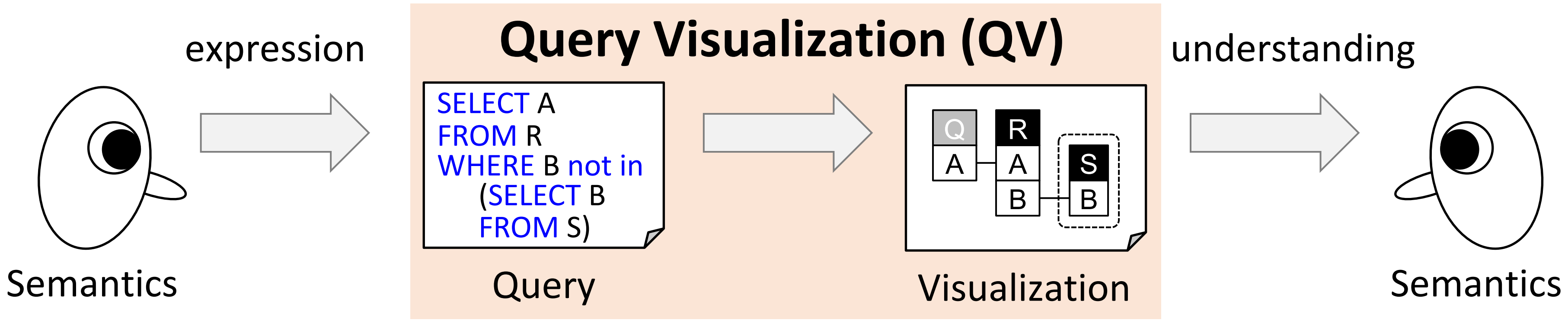
P3

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

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

Conceptual framework: Kindlmann, Scheidegger. An Algebraic Process for Visualization Design. TVCG 2014. https://doi.org/10.1109/TVCG.2014.2346325
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

An Algebraic Framework for Query Visualization



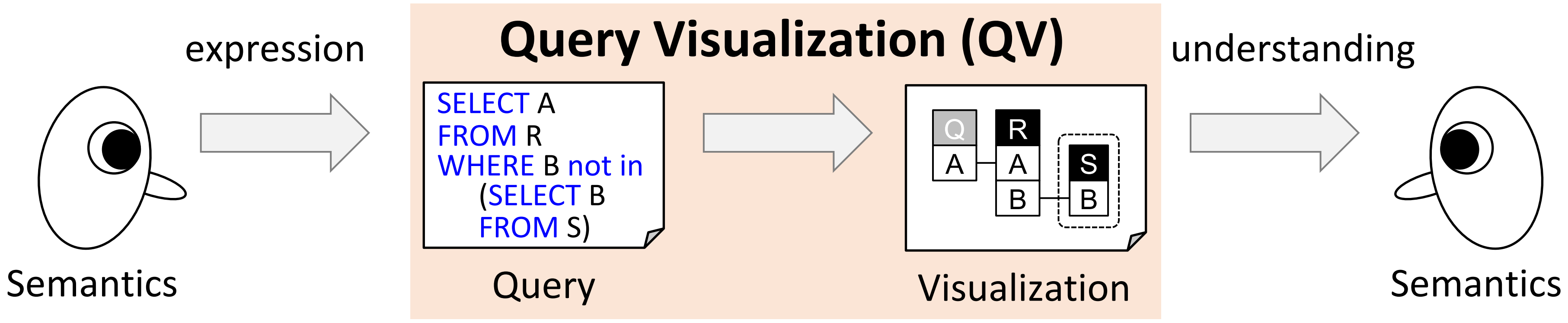
```
SELECT A
FROM R
WHERE B not in
    (SELECT B
    FROM S)
```

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

expression

Query Visualization (QV)

```
SELECT A
FROM R
WHERE B not in
    (SELECT B
    FROM S)
```

Query

Visualization

understanding

Semantics

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

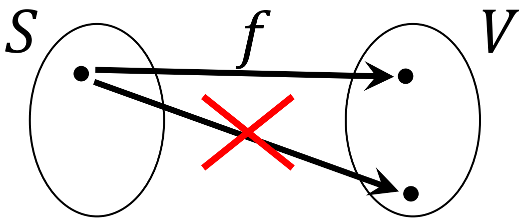
$$\begin{tikzcd}
s \arrow[r] \arrow[dd, "\cancel{\Delta s}"] & q \arrow[dr] \arrow[dd] & \\
& & v \\
s' \arrow[r] & q' \arrow[ur] &
\end{tikzcd}$$

4 Bijection principles of Query Visualization

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

P1. REPRESENTATION INVARIANCE

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

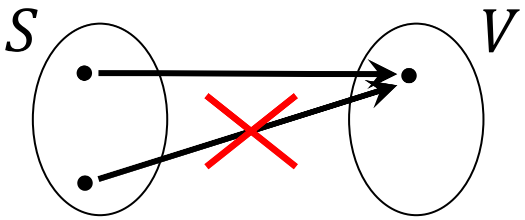


FUNCTION (instead of Relation)

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

P2. UNAMBIGUOUS

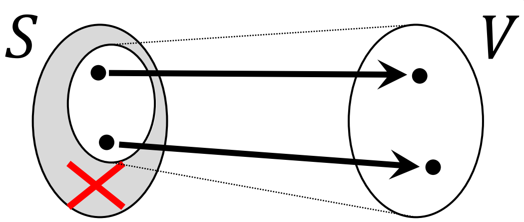
$$\begin{tikzcd} s \arrow[r] \arrow[dd, "\cancel{\Delta s}"] & q \arrow[dd] \arrow[dr] & \\ & & v \\ s' \arrow[r] & q' \arrow[ur] & \end{tikzcd}$$



INJECTIVE (preserve distinction)

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

4 Bijection principles of Query Visualization



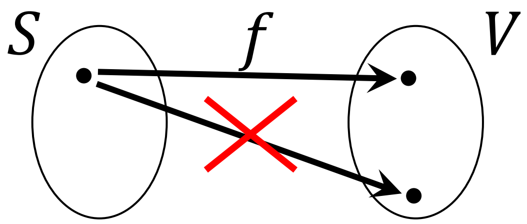
TOTAL (COMPLETE, instead of Partial)

P1:
The visual query representation is relationally complete: every query semantics can be represented

TOTAL FUNCTION

P1. REPRESENTATION INVARIANCE

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

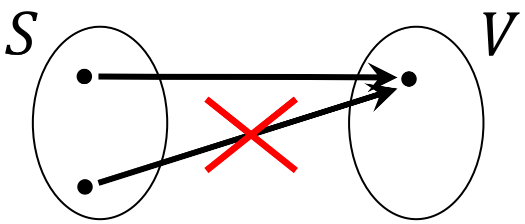


FUNCTION (instead of Relation)

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

P2. UNAMBIGUOUS

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

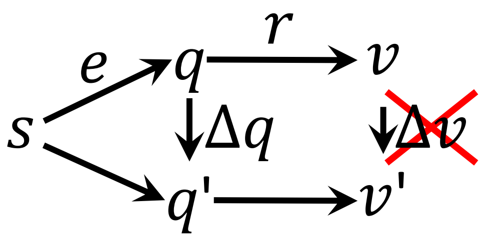


INJECTIVE (preserve distinction)

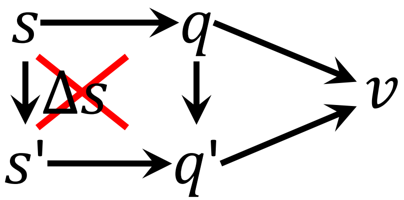
P2:
Visualizations are unambiguous: Every visualization represents at most one query semantics

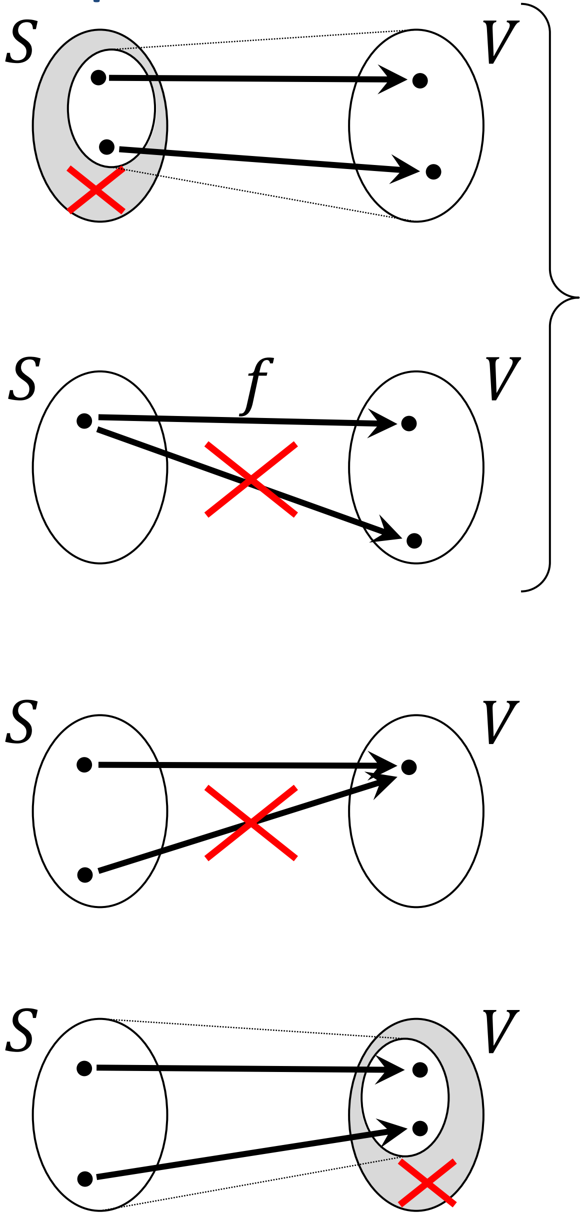
4 Bijection principles of Query Visualization

P1. REPRESENTATION INVARIANCE



P2. UNAMBIGUOUS





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 $\Delta S=0 \Rightarrow \Delta V=0$

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

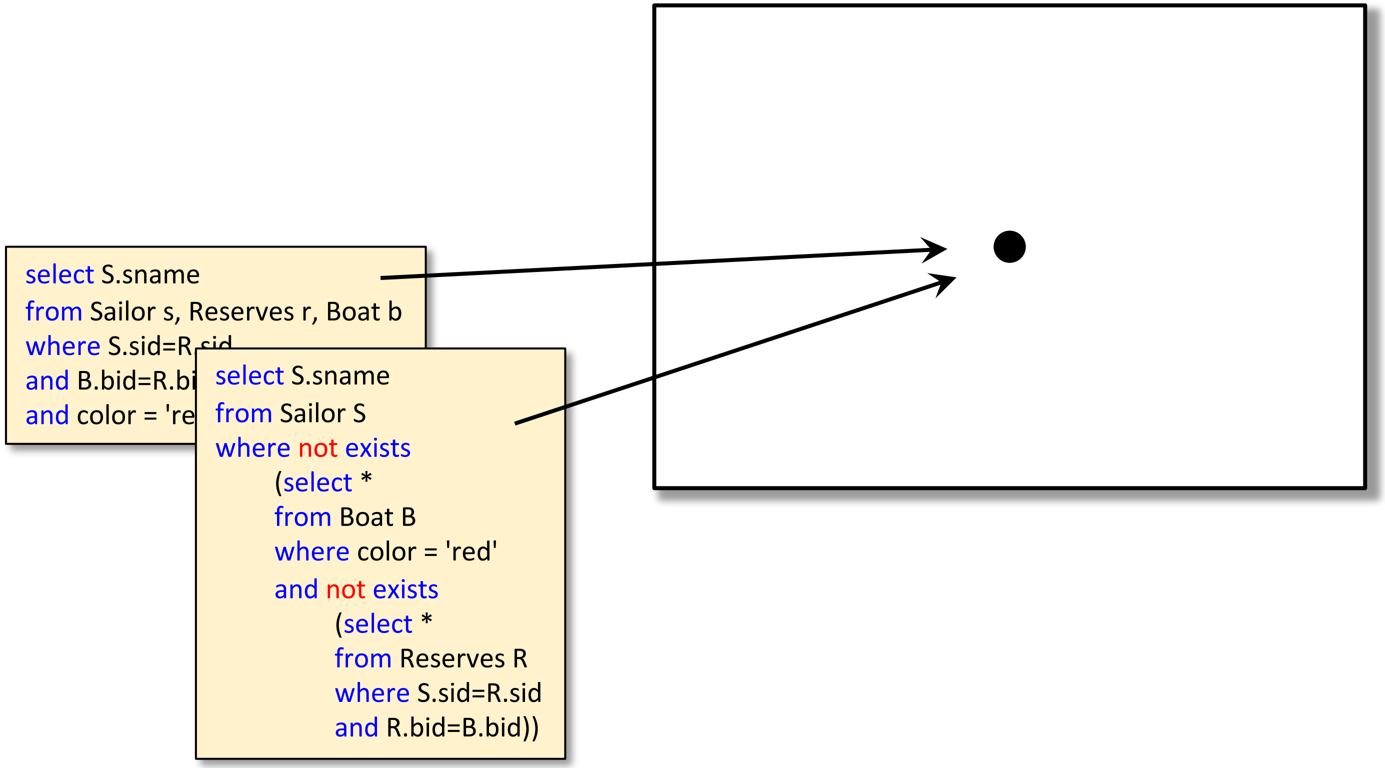
P3:
Visualizations are <u>sound</u>: Every valid visualization has some valid interpretation (query semantics)

Exercise: Discuss following new "query visualization"

Take any query and represent it by a centered dot in the plane

(You can make it "interactive": a clicking on the dot reveals the SQL query)

Which of the 4 principles to the right does this fulfill ? ?



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

```
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))
```

? 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

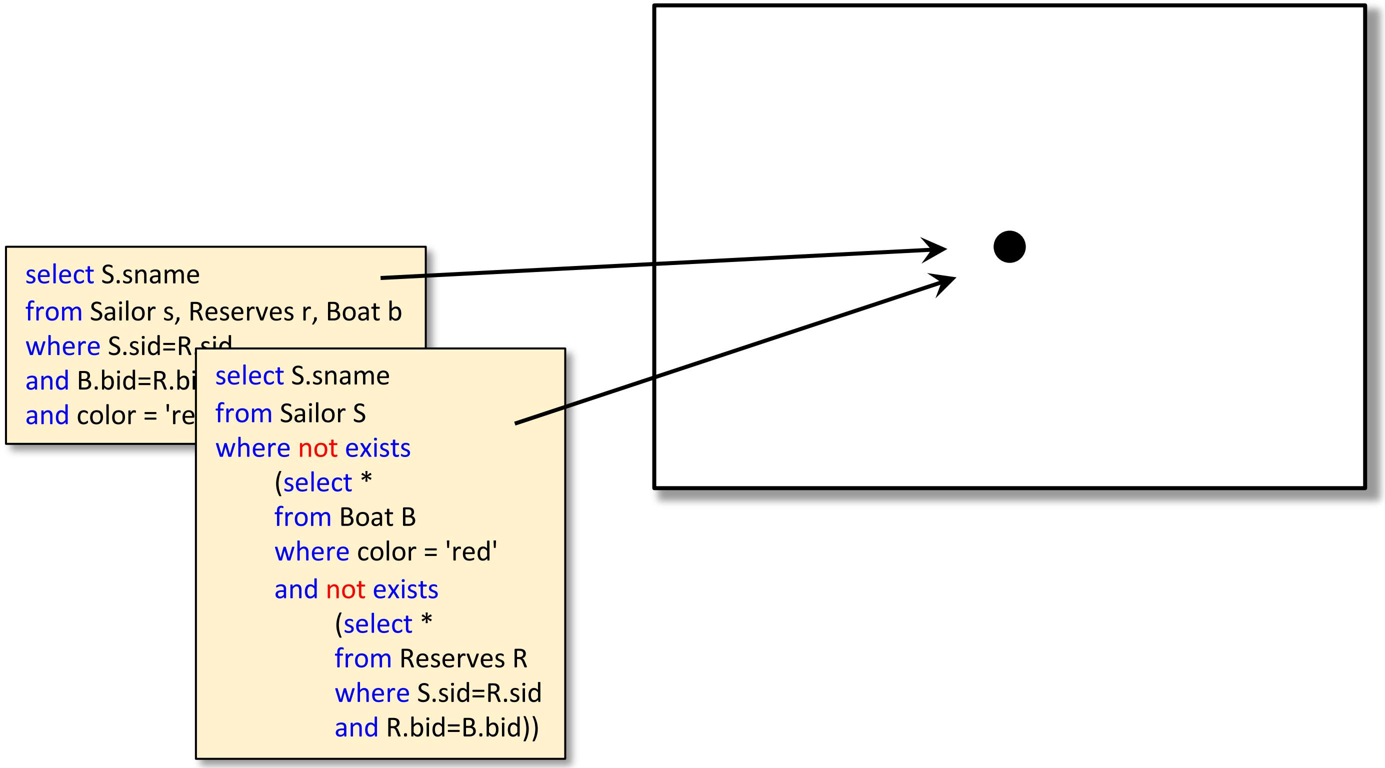
? P3:
Visualizations are <u>sound</u>: Every valid visualization has some valid interpretation (query semantics)

Exercise: Discuss following new "query visualization"

Take any query and represent it by a centered dot in the plane

(You can make it "interactive": a clicking on the dot reveals the SQL query)

Which of the 4 principles to the right does this fulfill ?



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

```
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))
```

🙂 P1:
The visual query representation is relationally complete: every query semantics can be represented

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

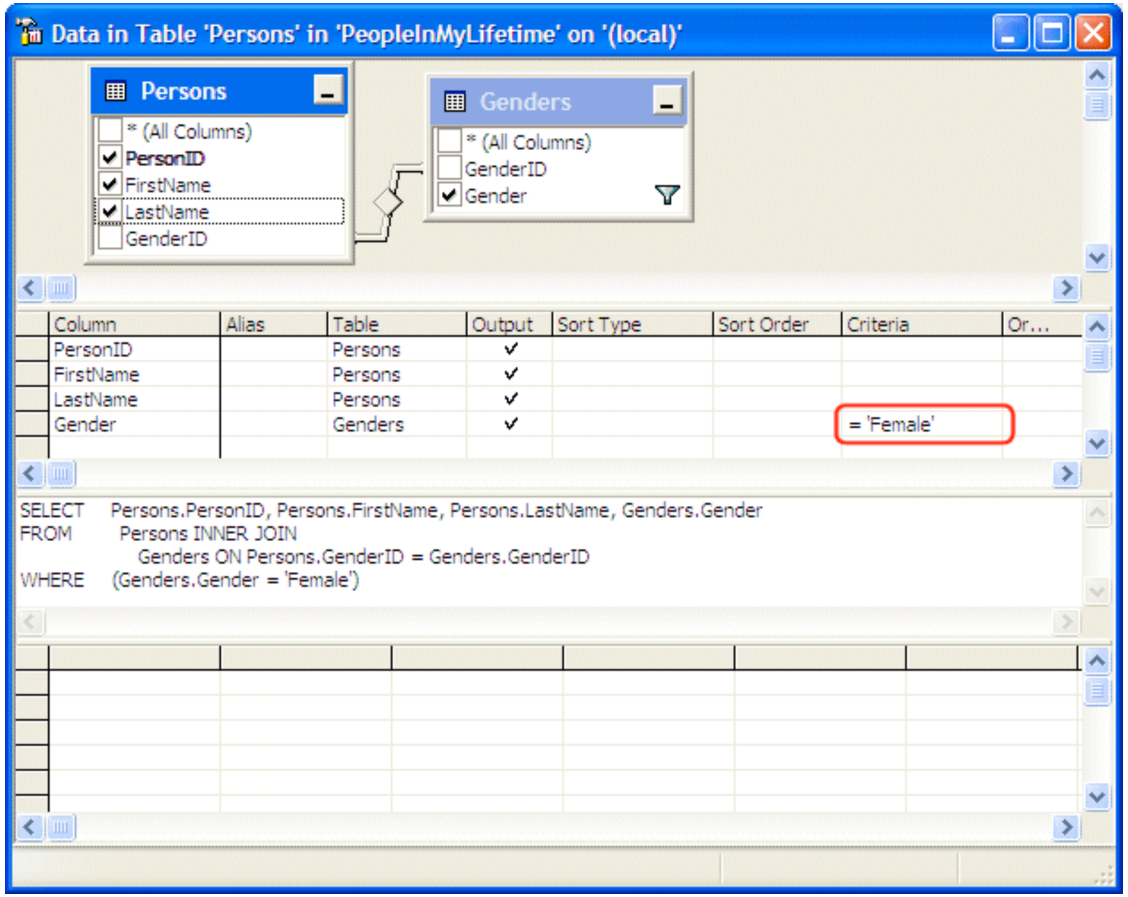
☹ P2:
Visualizations are unambiguous: Every visualization represents at most one query semantics

🙂 P3:
Visualizations are sound: Every valid visualization (*here: every single dot!*) has some valid interpretation (query semantics)

Those were the 4 bijection principles

Next: 7 additional "correspondence" principles

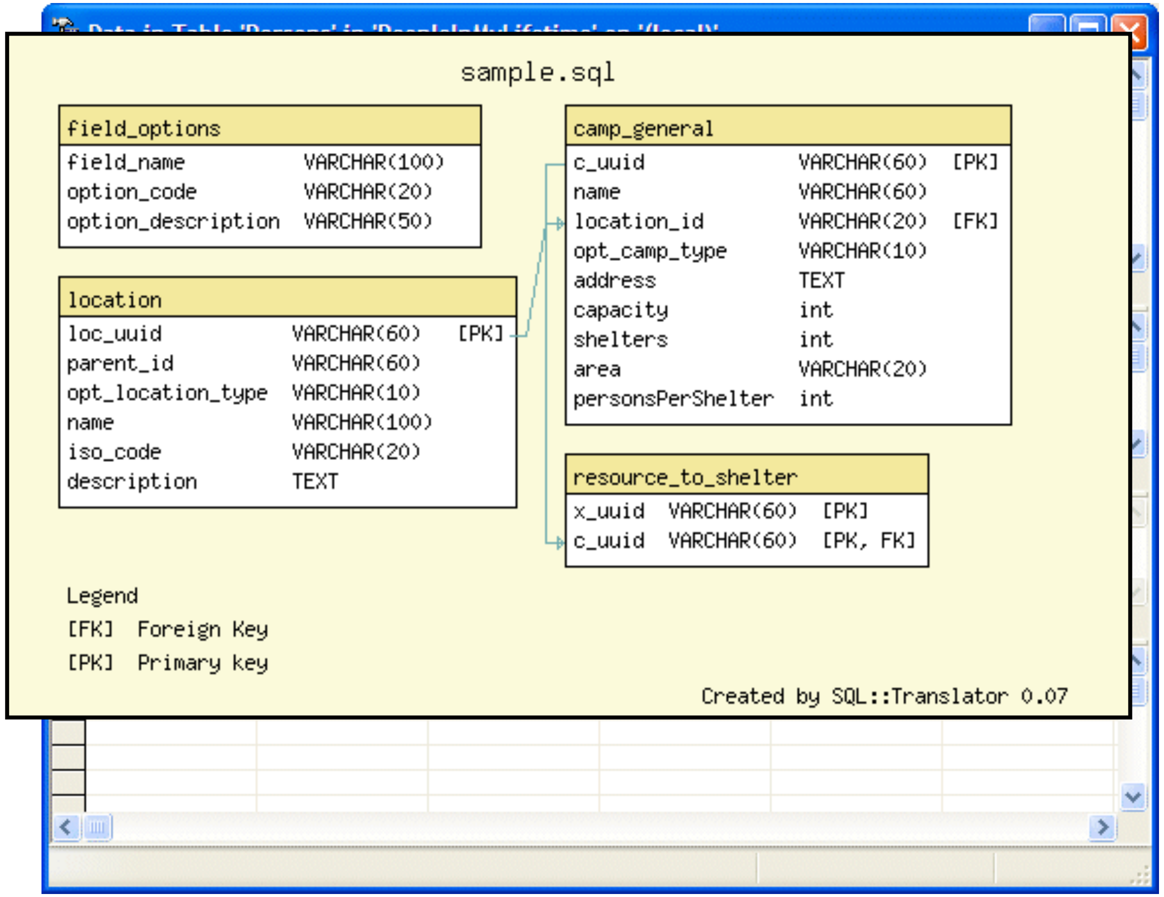
P4: Existing visual metaphors as starting point



Column	Alias	Table	Output	Sort Type	Sort Order	Criteria	Or...
PersonID		Persons	✓				
FirstName		Persons	✓				
LastName		Persons	✓				
Gender		Genders	✓			= 'Female'	

```
SELECT    Persons.PersonID, Persons.FirstName, Persons.LastName, Genders.Gender
FROM        Persons INNER JOIN
                 Genders ON Persons.GenderID = Genders.GenderID
WHERE    (Genders.Gender = 'Female')
```

P4: Existing visual metaphors as starting point



sample.sql

field_options		
field_name	VARCHAR(100)	
option_code	VARCHAR(20)	
option_description	VARCHAR(50)	

location		
loc_uuid	VARCHAR(60)	[PK]
parent_id	VARCHAR(60)	
opt_location_type	VARCHAR(10)	
name	VARCHAR(100)	
iso_code	VARCHAR(20)	
description	TEXT	

camp_general		
c_uuid	VARCHAR(60)	[PK]
name	VARCHAR(60)	
location_id	VARCHAR(20)	[FK]
opt_camp_type	VARCHAR(10)	
address	TEXT	
capacity	int	
shelters	int	
area	VARCHAR(20)	
personsPerShelter	int	

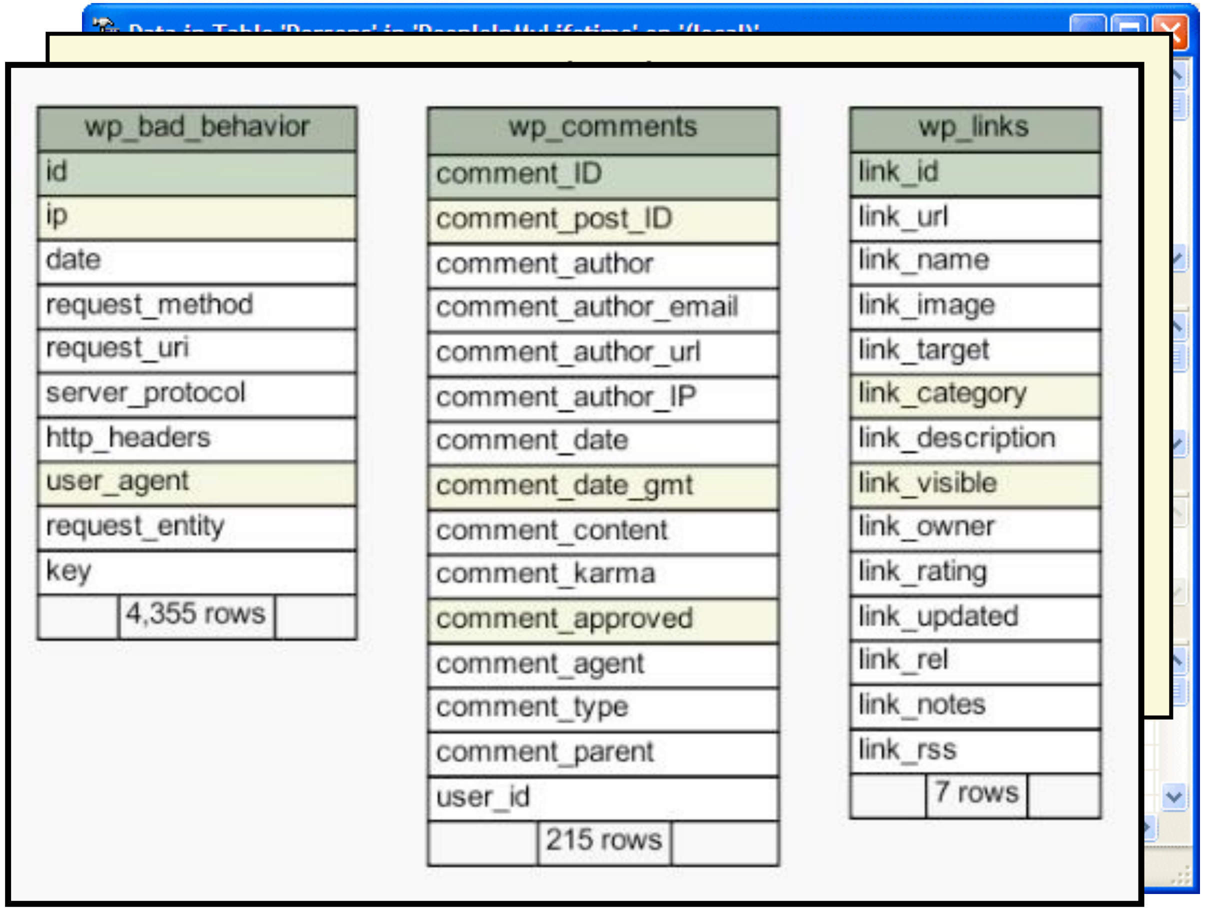
resource_to_shelter		
x_uuid	VARCHAR(60)	[PK]
c_uuid	VARCHAR(60)	[PK, FK]

Legend

[FK]	Foreign Key
[PK]	Primary key

Created by SQL::Translator 0.07

P4: Existing visual metaphors as starting point

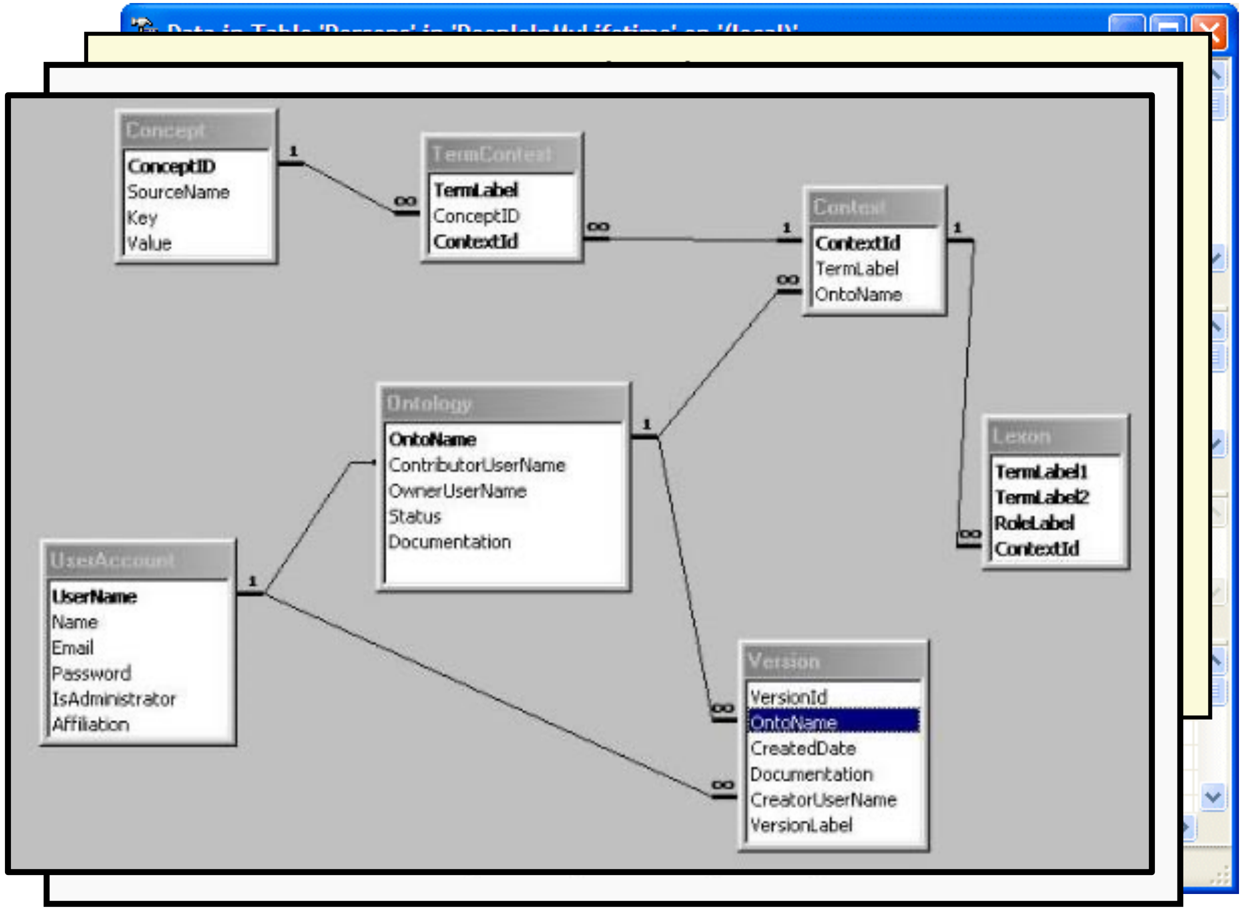


wp_bad_behavior
id
ip
date
request_method
request_uri
server_protocol
http_headers
user_agent
request_entity
key
4,355 rows

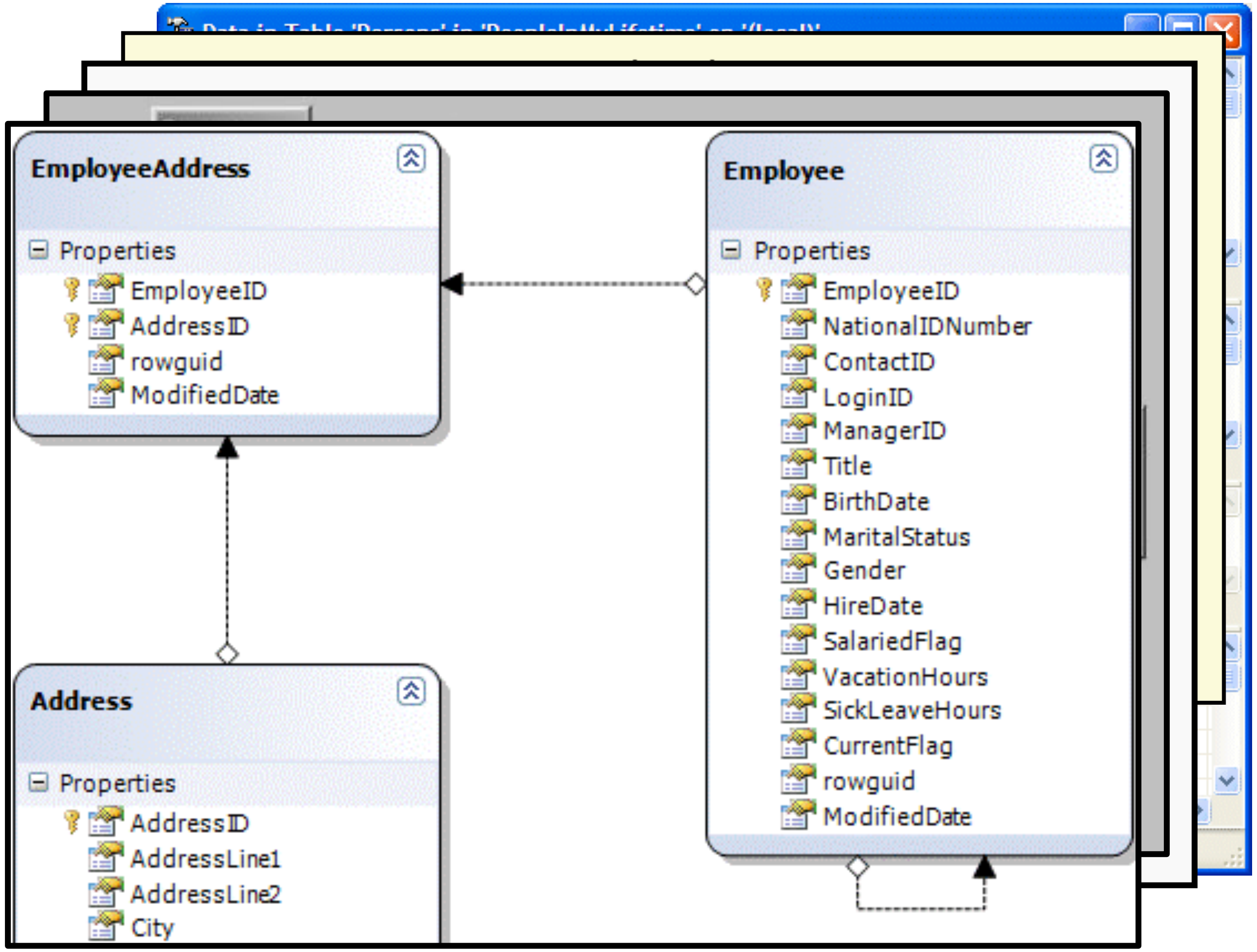
wp_comments
comment_ID
comment_post_ID
comment_author
comment_author_email
comment_author_url
comment_author_IP
comment_date
comment_date_gmt
comment_content
comment_karma
comment_approved
comment_agent
comment_type
comment_parent
user_id
215 rows

wp_links
link_id
link_url
link_name
link_image
link_target
link_category
link_description
link_visible
link_owner
link_rating
link_updated
link_rel
link_notes
link_rss
7 rows

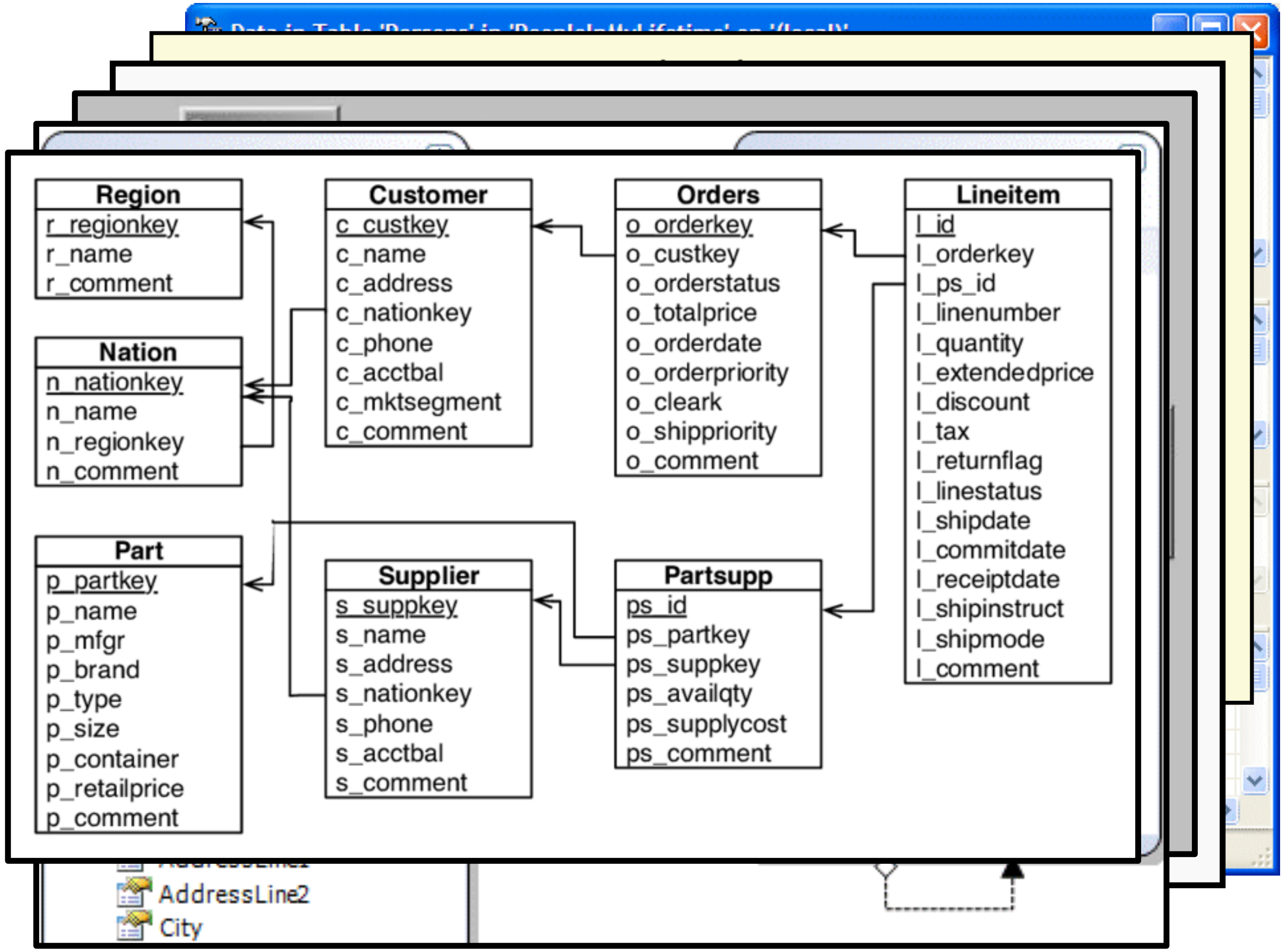
P4: Existing visual metaphors as starting point



P4: Existing visual metaphors as starting point



P4: Existing visual metaphors as starting point



Region
r_regionkey
r_name
r_comment

Nation
n_nationkey
n_name
n_regionkey
n_comment

Part
p_partkey
p_name
p_mfgr
p_brand
p_type
p_size
p_container
p_retailprice
p_comment

Customer
c_custkey
c_name
c_address
c_nationkey
c_phone
c_acctbal
c_mktsegment
c_comment

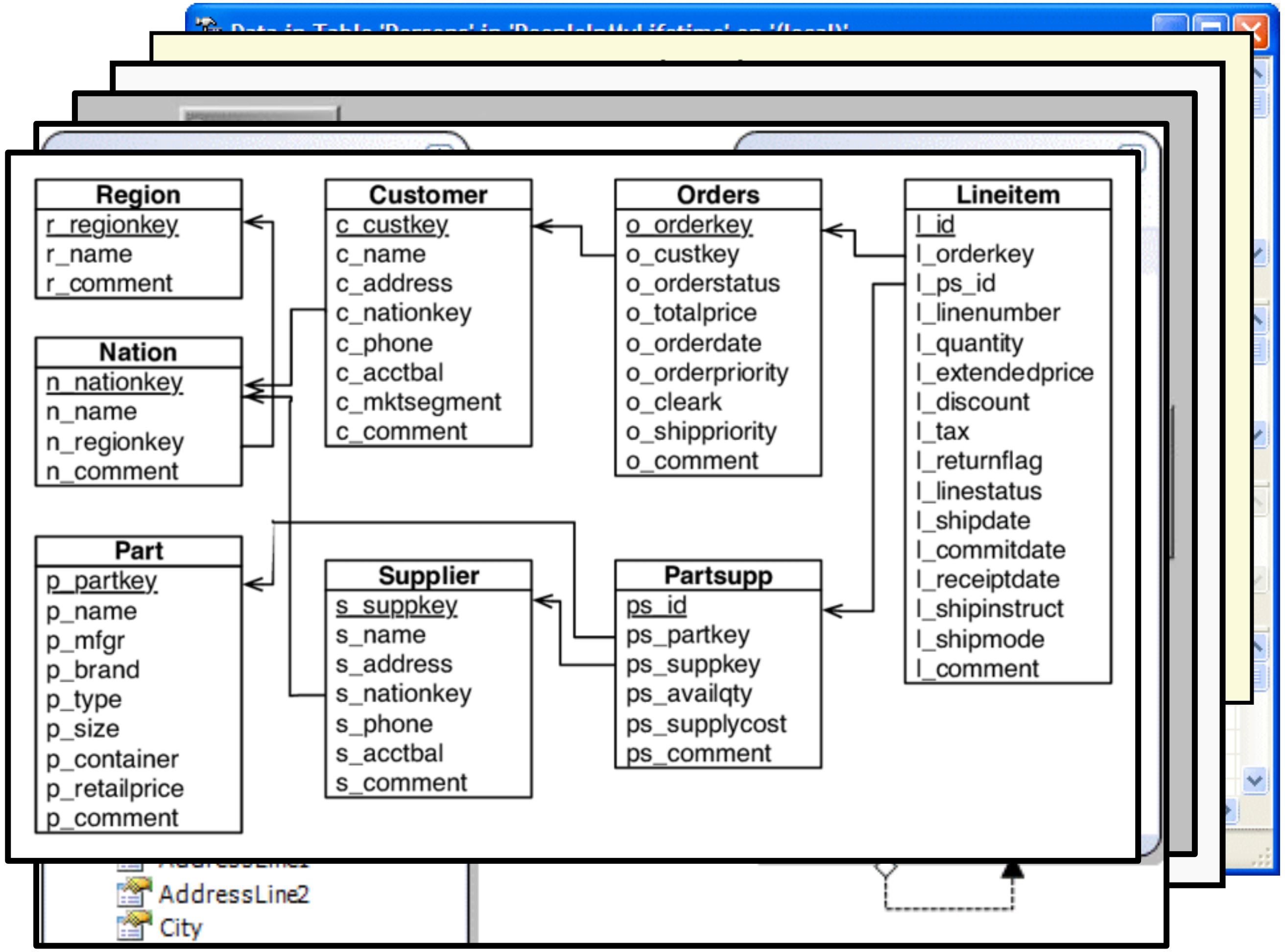
Supplier
s_suppkey
s_name
s_address
s_nationkey
s_phone
s_acctbal
s_comment

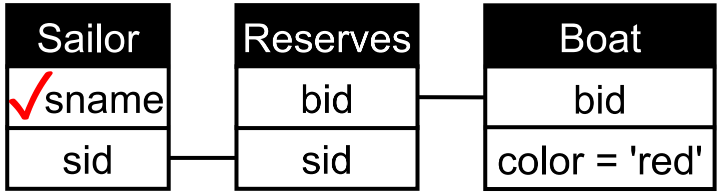
Orders
o_orderkey
o_custkey
o_orderstatus
o_totalprice
o_orderdate
o_orderpriority
o_cleark
o_shippriority
o_comment

Partsupp
ps_id
ps_partkey
ps_suppkey
ps_availqty
ps_supplycost
ps_comment

Lineitem
l_id
l_orderkey
l_ps_id
l_linenumber
l_quantity
l_extendedprice
l_discount
l_tax
l_returnflag
l_linestatus
l_shipdate
l_commitdate
l_receiptdate
l_shipinstruct
l_shipmode
l_comment

P4: Existing visual metaphors as starting point





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

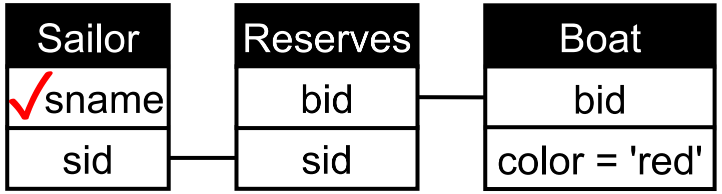
P4: Start from known visual UML metaphors for relational schemas

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',_)

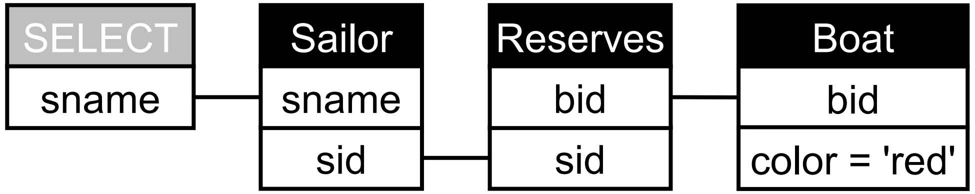
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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'
```

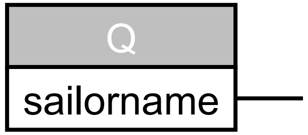


Relational queries are compositional:

- Input are relations (tables)
- Output are tables

TRC (Tuple Relational Calculus)

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



Explicit output table also allow renaming of tables and attributes

Datalog

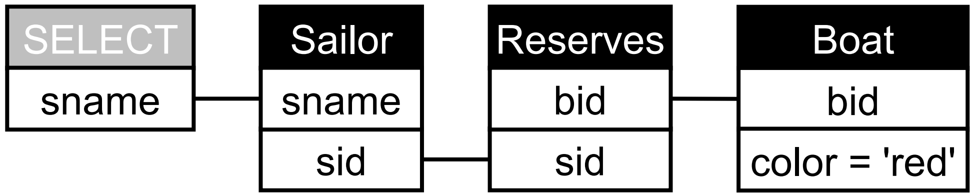
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

P5: Compositionality of the relational model

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

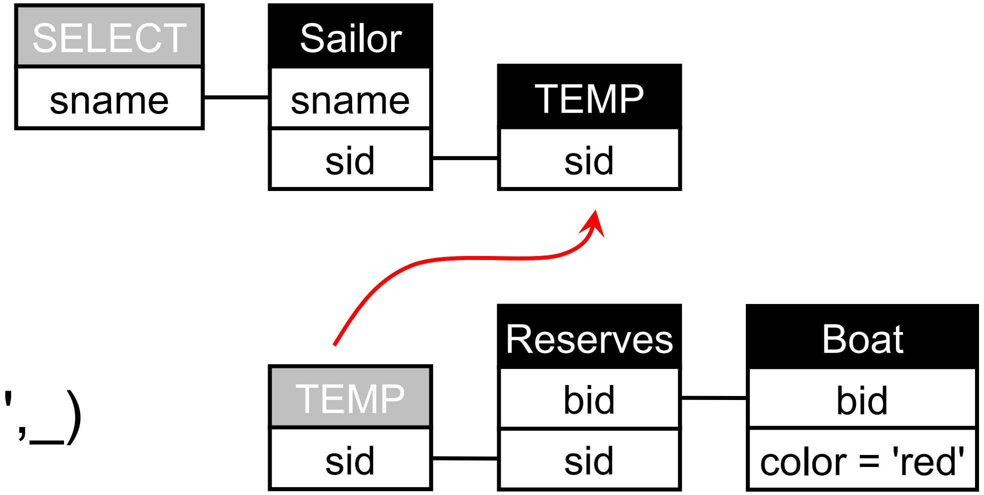
```
select distinct S.sname
from Sailor S, Reserves R, Boat B
where S.sid=R.sid
and B.bid=R.bid
and color = 'red'
```

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']}

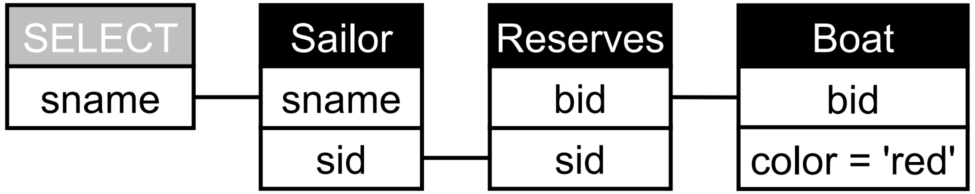
Datalog

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'
```



SELECT
sname

Sailor
sname
sid

Reserves
bid
sid

Boat
bid
color = 'red'

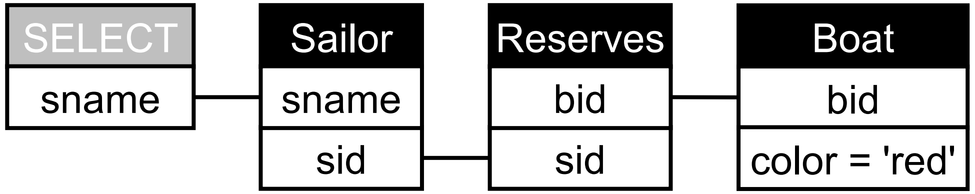
Q: "Find sailors who reserved only red boats."

?

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'))
```

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

P6: Progressive visual complexity

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

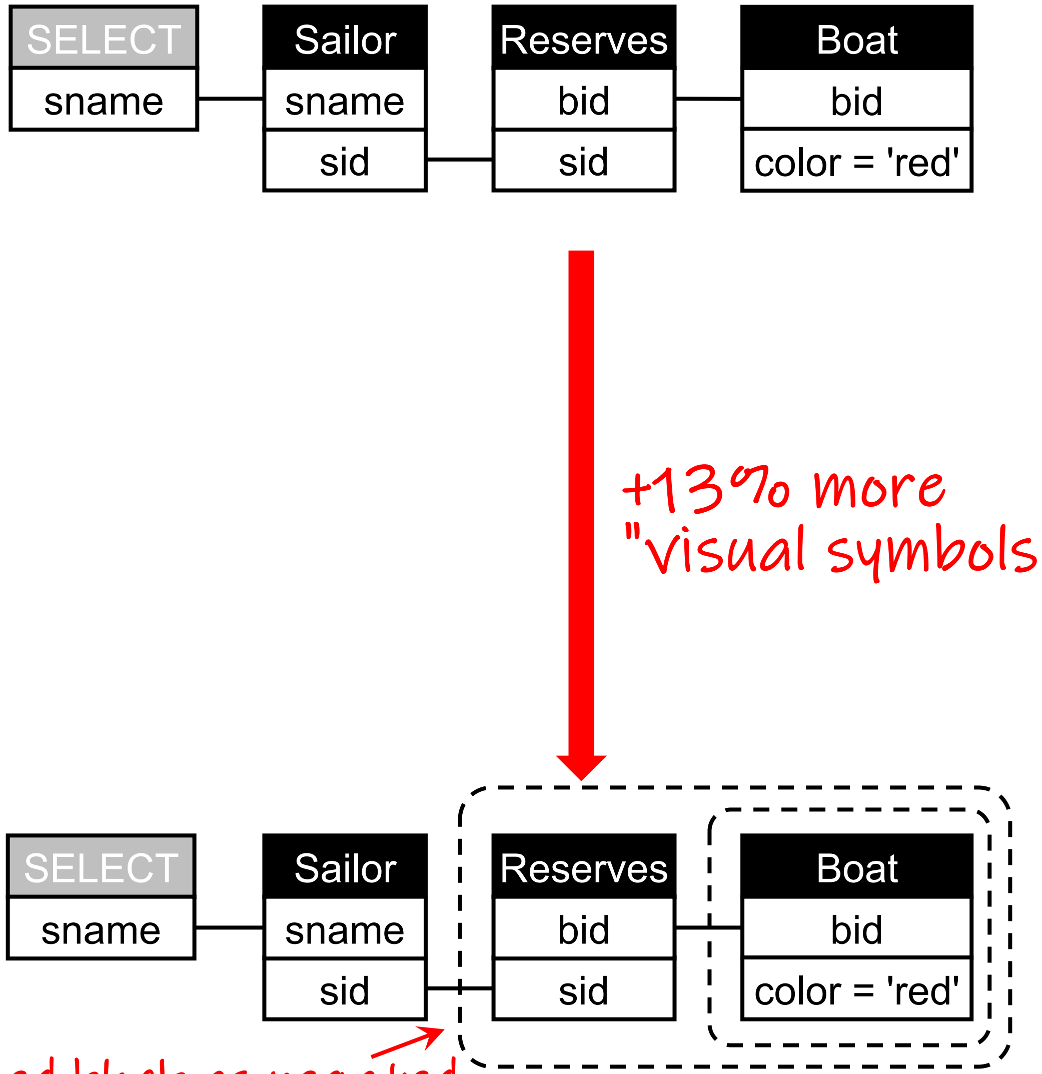
Q: "Find sailors who reserved a red boat."

```
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"

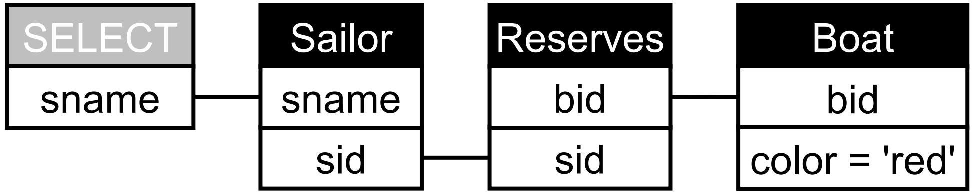
interprete query block with dashed block as negated

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'
```

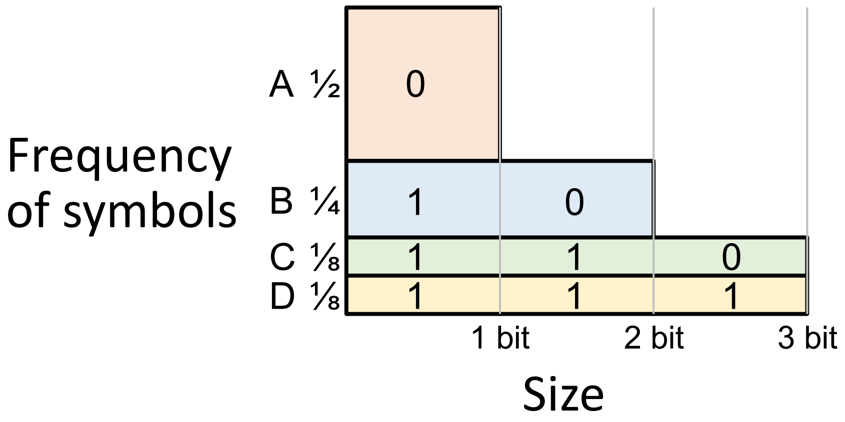


+67% more SQL text

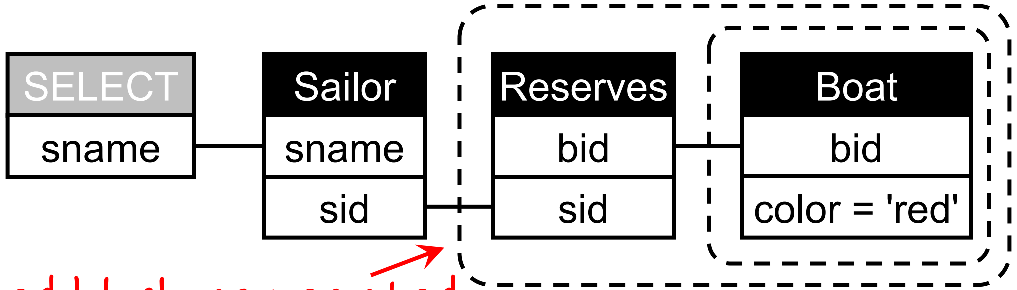
Q: "Find sailors who reserved only red boats."

```
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

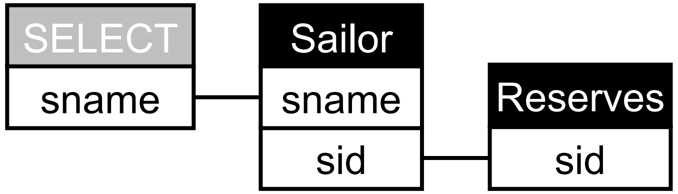
P7. SYNTACTIC INVARIANCE

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

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

TRC (Tuple Relational Calculus)

{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]]}

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

P7: Abstract away from syntax details

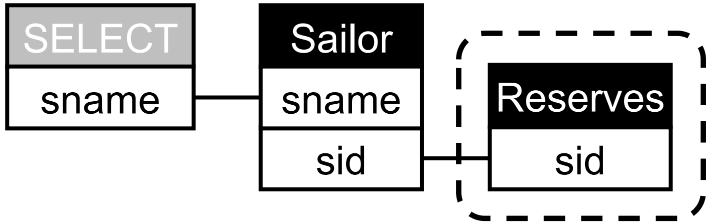
P7. SYNTACTIC INVARIANCE

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

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)

$$\{q(\text{sname}) \mid \exists s \in \text{Sailor} [q.\text{sname}=s.\text{sname} \wedge \neg(\exists r \in \text{Reserves}[r.\text{sid}=s.\text{sid}])]\}$$

Example queries from "Databases will visualize queries too", by G, VLDB 2011. https://doi.org/10.14778/3402755.3402805
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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)))
```

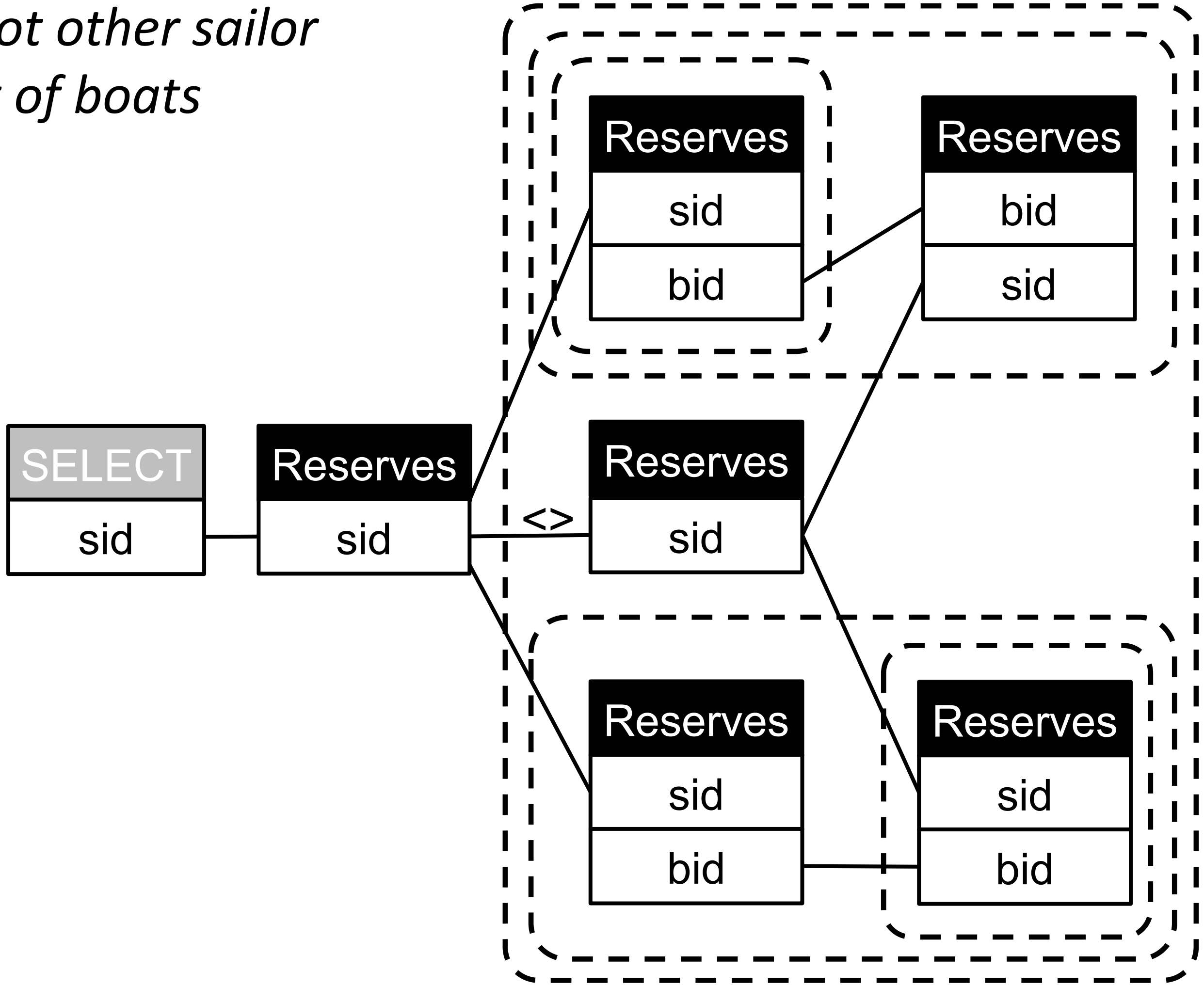
?

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*



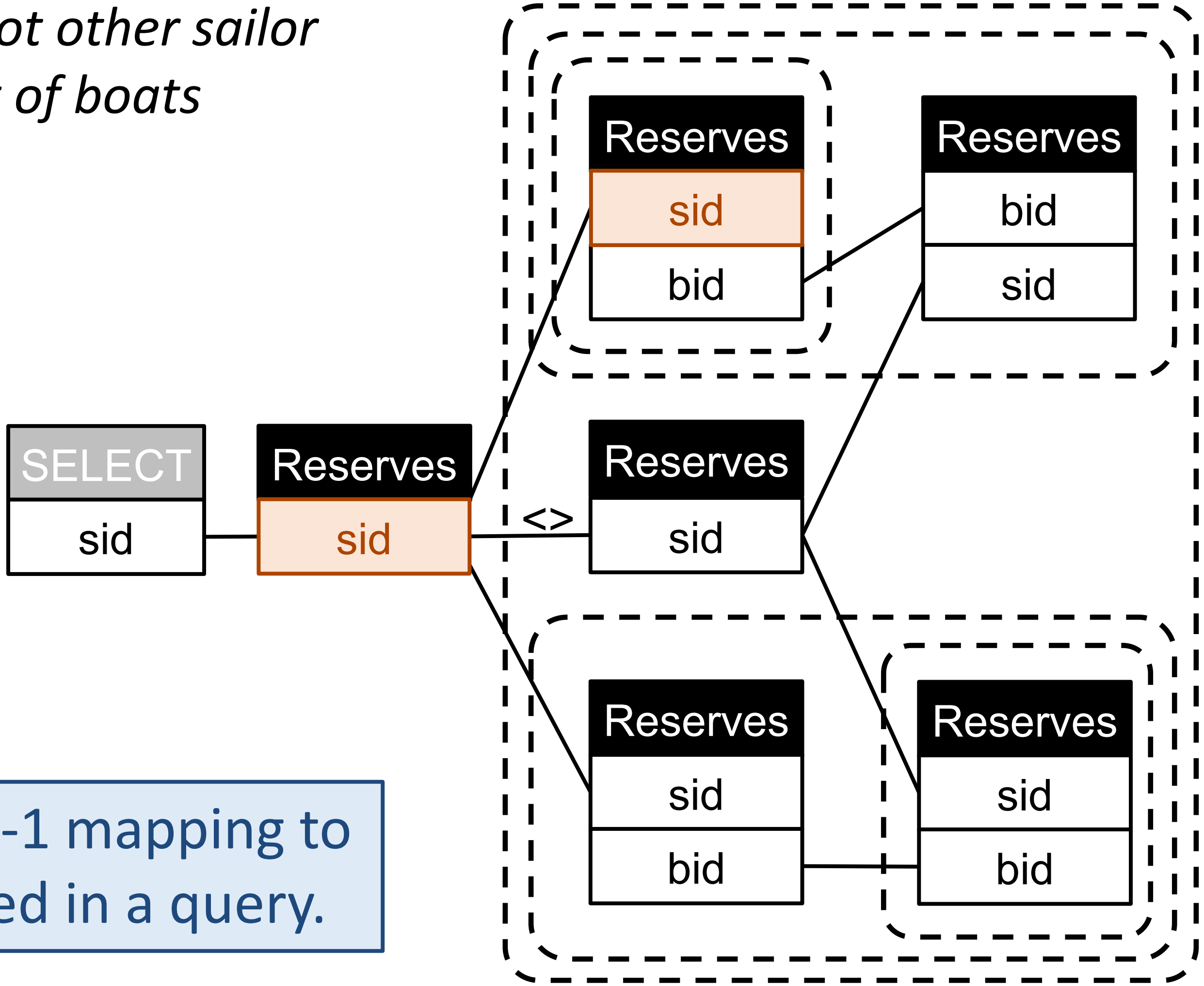
For a formal definition of "relational query patterns" see "Relational Diagrams: a pattern-preserving diagrammatic representation of non-disjunctive Relational Queries". https://arxiv.org/pdf/2203.07284

P8: Expose (and not hide) relational patterns

```
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

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
```

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/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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 for self-joins (i.e. repeated appearances of the same input table), which implies an inconvenient indirection to the database schema

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

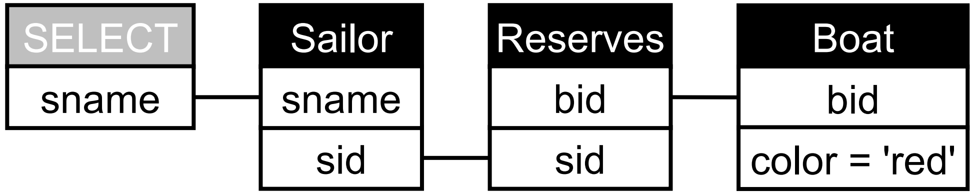
Datalog needs to show all attributes due to positional encoding (though it allows the use of "anonymous variables", via underscores)

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'
```



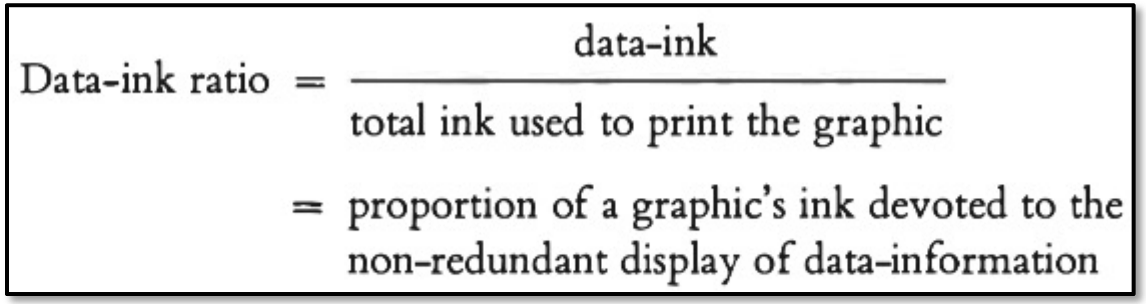
- no aliases needed
- no need to show unused predicates

SQL requires aliases for self-joins (i.e. repeated appearances of the same input table), 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



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

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

"Data-ink ratio" screenshot from "Tufte. The visual display of quantitative information, 2nd ed, 2001. https://www.edwardtufte.com/tufte/books_vdqi "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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

Datalog

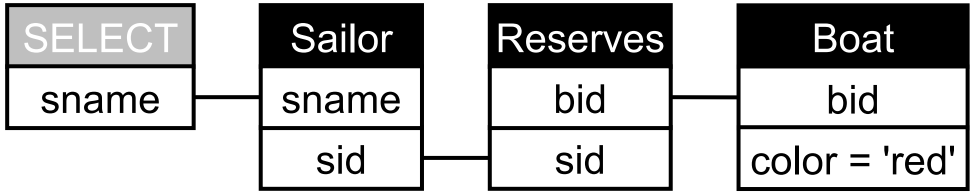
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

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']}

Datalog

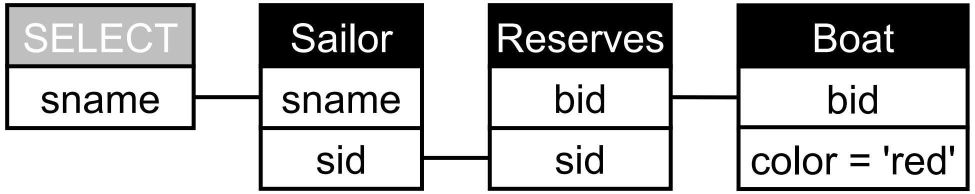
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)

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)

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

Notice that this is notably different from typical workflow visualizations!

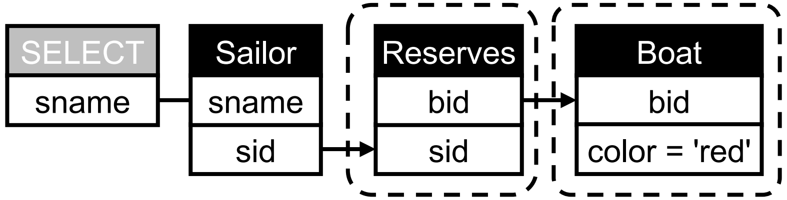
Datalog

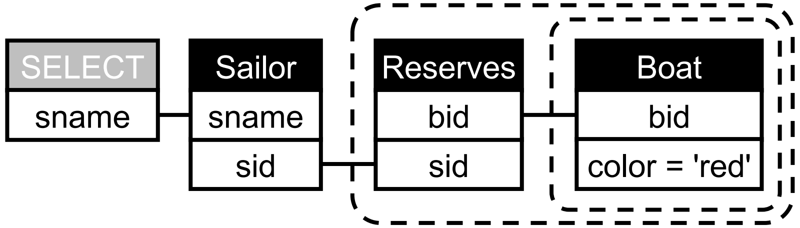
```
Q(x) :- Sailor(y,x,_,_),Reserves(y,z,_), Boat(z,_,'red',_)
```

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'))
```





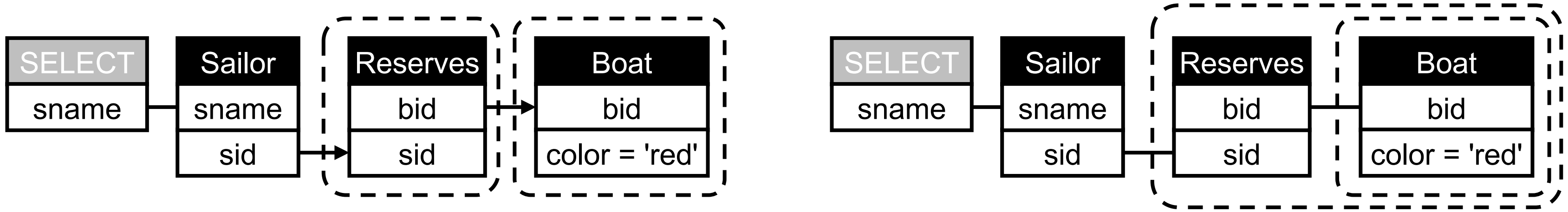
$$\{q.n \mid \exists s \in \text{Sailor}[q.n = s.n \wedge \neg(\exists r \in \text{Reserves}[r.s = s.s \wedge \neg(\exists b \in \text{Boat}[b.b = r.b \wedge b.c = \text{'red'}])])]\}$$

P11: Logic-based visual transformations

P11: Allow limited visual transformations that help reading nested negations

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

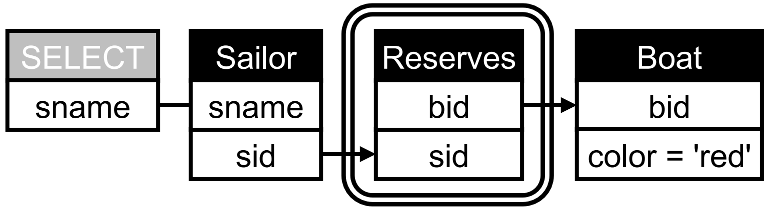
```
select distinct S.sname
from Sailor S
where not exists
   (select *
   from Reserves R
   where S.sid=R.sid
   and not exists
      (select *
      from Boat B
      where R.bid=B.bid
      and color = 'red'))
```

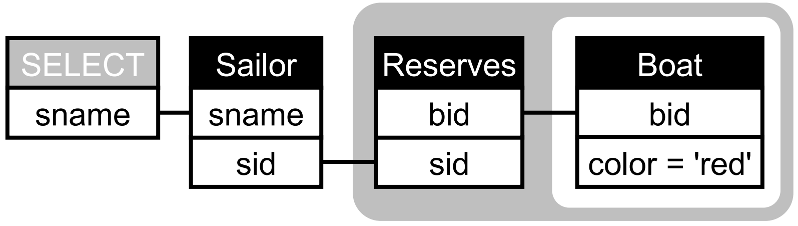


$$\{q.n \mid \exists s \in \text{Sailor}[q.n=s.n \wedge \neg(\exists r \in \text{Reserves}[r.s=s.s \wedge \neg(\exists b \in \text{Boat}[b.b=r.b \wedge b.c=\text{'red'}])])]\}$$

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

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





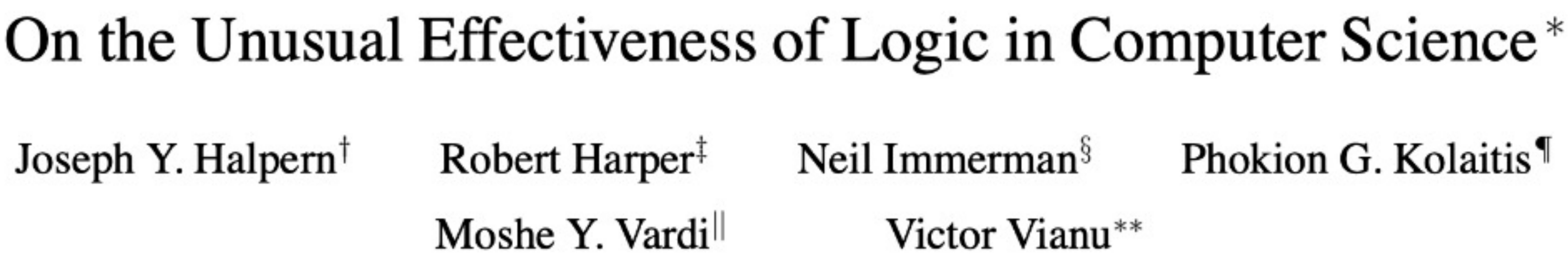
Double lines represent for all ∀

Human perception can "manipulate" shaded areas without new symbols

Intended Agenda today

Please leave feedback ☺

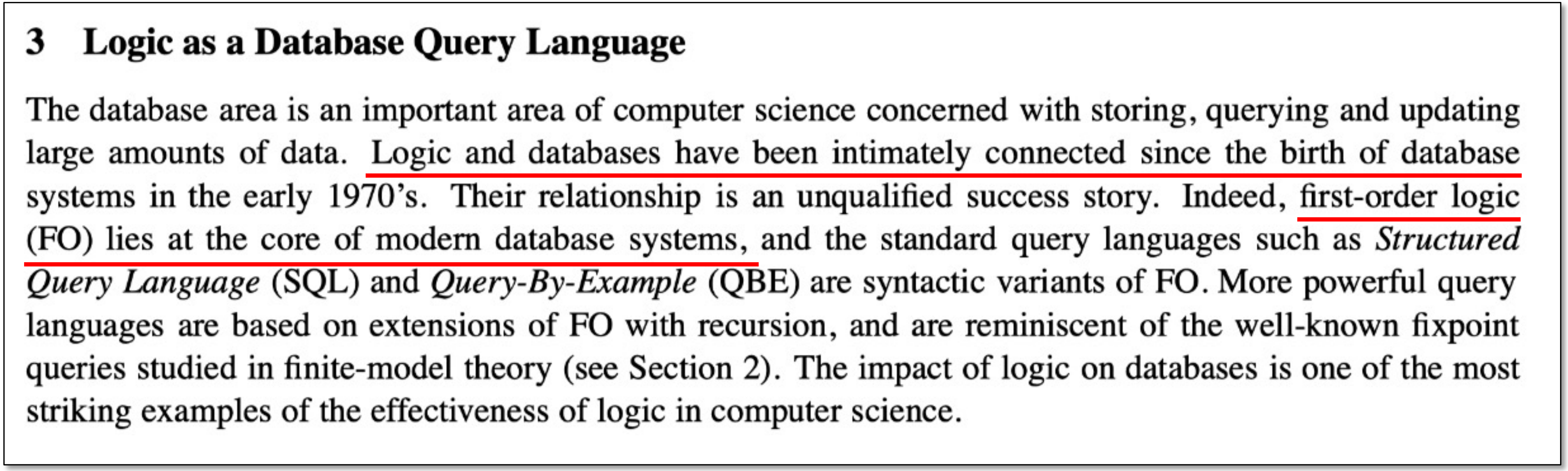
1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. (Early) Diagrammatic representations
5. Visual Query Representations (from DB community)
6. Lessons Learned and Open Challenges



On the Unusual Effectiveness of Logic in Computer Science[*]

Joseph Y. Halpern[†] Robert Harper[‡] Neil Immerman[§] Phokion G. Kolaitis[¶]
Moshe Y. Vardi[||] Victor Vianu[**]

THE
BULLETIN OF
SYMBOLIC
LOGIC



3 Logic as a Database Query Language

The database area is an important area of computer science concerned with storing, querying and updating large amounts of data. Logic and databases have been intimately connected since the birth of database systems in the early 1970's. Their relationship is an unqualified success story. Indeed, first-order logic (FO) lies at the core of modern database systems, and the standard query languages such as *Structured Query Language* (SQL) and *Query-By-Example* (QBE) are syntactic variants of FO. More powerful query languages are based on extensions of FO with recursion, and are reminiscent of the well-known fixpoint queries studied in finite-model theory (see Section 2). The impact of logic on databases is one of the most striking examples of the effectiveness of logic in computer science.

"On the Unusual Effectiveness of Logic in Computer Science" by Halpern, Harper, Immerman, Kolaitis, Vardi, Bulletin of Symbolic Logic, 2001. https://doi.org/10.2307/2687775

Towards a principled comparison of visual metaphors

- Our goal is to highlight similarities and differences of various visual query representations.
- To achieve a unified comparison of visual formalisms, we will represent a consistent yet diverse set of queries from the fragment of first order logic (FOL) with equalities.
- We give these queries in different textual QLs since various visual formalisms align more naturally with different textual QL:
  - Relational Algebra (RA)
  - Relational Calculus (RC): Domain RC (DRC) vs. Tuple RC (TRC)
  - Datalog
  - SQL, restricted to a limited fragment and set semantics

Part 3: Logical Foundations of Relational Query Languages

Sailor queries (for Part 5)

- 1 practical database schema (sailors-reserve-boats)
- 5 queries (with variants) in 5 slides
- Each query in 5 Query Languages (QLs)

Monadic predicate calculus = unary predicates (for Part 4)

- 1 abstract schema (R, S)
- 4 queries in 4 slides
- Each statement in 5 textual and 3 visual QLs

Query 1

We focus on set semantics (which is the semantics in logics and algebra)

Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

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

Q1a: "Find boats that are red."

Q1b: "Find boats that are not red."

Q1c: "Find boats that are red or blue and are purchased before 1980."

Comparison predicate

TRC (Tuple Relational Calculus)

$$\{q(\text{bname}) \mid \exists b \in \text{Boat}\,[q.\text{bname}=b.\text{bname} \wedge (b.\text{color}=\text{'red'} \vee b.\text{color}=\text{'blue'})]\}$$

DRC (Domain Relational Calculus)

$$\{(x) \mid \exists y,z,u\,[\text{Boat}(z,x,y,u) \wedge (y=\text{'red'} \vee y=\text{'blue'})\}$$

$$\{(x) \mid \exists y\,[\text{Boat}(\_,x,y,\_) \wedge (y=\text{'red'} \vee y=\text{'blue'})\}$$

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

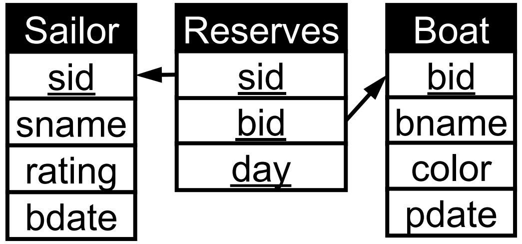
$$\sigma_{\text{color='red'} \vee \text{color='blue}} B$$

Query 2

Q2: "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)

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

DRC (Domain Relational Calculus)

$$\{(x) \mid \exists v,z,w,y,t,u,s\ [\text{Sailor}(v,x,z,w) \wedge \text{Reserves}(v,y,t) \wedge \text{Boat}(y,u,'\text{red}',s)\}$$

Datalog

Q2a: "Find sailors and red boats they reserved."

$$Q(x) \text{ :- } \text{Sailor}(y,x,\_,\_),\ \text{Reserves}(y,z,\_),\ \text{Boat}(z,\_,'\text{red}',\_)$$

Relational Algebra

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

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

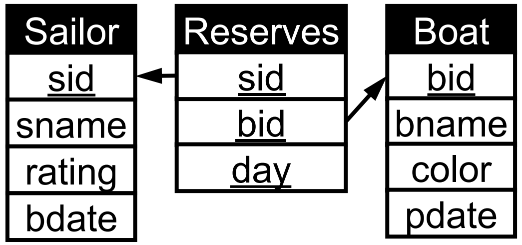
Query 3

Q3: "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'))
```

Q3a: "Find sailors who reserved no boat."
Q3b: "Find sailors who reserved no red boat."

Schema



TRC (Tuple Relational Calculus)

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

DRC (Domain Relational Calculus)

$\{(x) \mid \exists v,z,w\ [Sailor(v,x,z,w) \wedge$
$\quad \neg(\exists y,t\ [Reserves(v,y,t) \wedge$
$\quad \neg(\exists u,s\ [Boat(y,u,\text{'red'},s)])])]\}$

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 conditions

BadSids: sailors who reserved some non-red boat

Relational Algebra

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

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

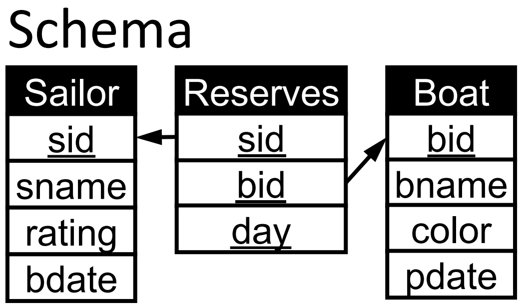
Antijoin operator (also anti-semijoin)

Query 4

Q4: "Find sailors who reserved all red boats."

```
select distinct 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

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

Datalog requires another Sailor relation!

Relational Algebra

Notice the cross product!

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

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

Query 5

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

TRC (Tuple Relational Calculus)

$$\{q(\text{bid},\text{bname}) \mid \exists b \in \text{RedBoat}\,[q.\text{bid}=b.\text{bid} \wedge q.\text{bname}=b.\text{bname}] \\ \vee \exists b \in \text{BlueBoat}\,[q.\text{bid}=b.\text{bid} \wedge q.\text{bname}=b.\text{bname}]\}$$

DRC (Domain Relational Calculus)

$$\{(x,y) \mid \exists z\,[\text{RedBoat}(x,y,z) \vee \text{BlueBoat}(x,y,z)]\}$$

Non-recursive Datalog with negation

```
Q(x,y) :- RedBoat(x,y,_)
Q(x,y) :- BlueBoat(x,y,_)
```

Relational Algebra

Algebra requires the union operator

$$\pi_{\text{bid, bname}}(\text{RedBoat} \cup \text{BlueBoat})$$

Schema

RedBoat
<u>bid</u>
bname
pdate

BlueBoat
<u>bid</u>
bname
pdate

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 3: Logical Foundations of Relational Query Languages

Sailor queries (for Part 5)

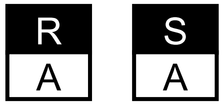
- 1 practical database schema (sailors-reserve-boats)
- 5 queries (with variants) in 5 slides
- Each query in 5 Query Languages (QLs)

Monadic predicate calculus = unary predicates (for Part 4)

- 1 abstract schema (R, S)
- 4 queries in 4 slides
- Each statement in 5 textual and 3 visual QLs

Statement 6: Monadic FOL (= only unary predicates)

Schema



S6: "Some R is not S."

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

"Statements" are Boolean queries (the answer is true or false).

"Monadic" means that all predicates are unary

TRC (Tuple Relational Calculus)

$\exists r \in R\, [\neg(\exists s \in S\, [r.A = s.A])]$

DRC (Domain Relational Calculus)

$\exists x\, [R(x) \wedge \neg S(x)]$

Datalog w/ inequalities

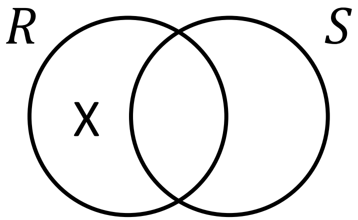
Q() :- R(x), not S(x)

Relational Algebra

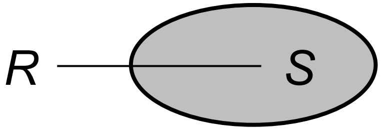
$\pi_{\emptyset}(R - S)$

We will discuss these visual formalisms in detail later

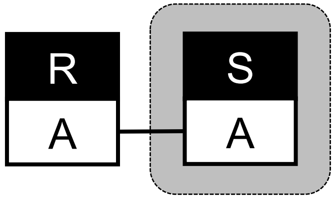
Venn-Peirce diagram



Beta Existential Graph



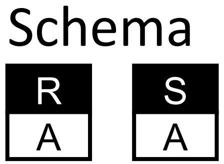
Relational Diagram



Database to run SQL queries is available as schema 371 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Statement 7: Monadic FOL with equalities



S7: "There is exactly one R."

```
select exists
    (select *
    from R
    where not exists
        (select *
        from R R2
        where R.A <> R2.A))
```

TRC (Tuple Relational Calculus)

$\exists r \in R\ [\neg(\exists r2 \in R\ [r.A \neq r2.A])]$

DRC (Domain Relational Calculus)

$\exists x\ [R(x) \wedge \neg(\exists x[R(y)\ \wedge x \neq y])]$

$\exists x\ [R(x) \wedge \quad \forall x[R(y) \rightarrow x=y]]$

Datalog w/ inequalities

```
I(x) :- R(x), R(y), x!=y
Q() :- R(x), not I(x)
```

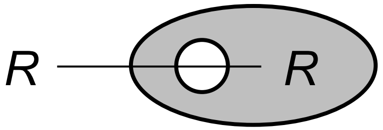
Relational Algebra

$\pi_{\emptyset}(R - \pi_A(\sigma_{A \neq B}(R \times \rho_{A \rightarrow B}(R))))$

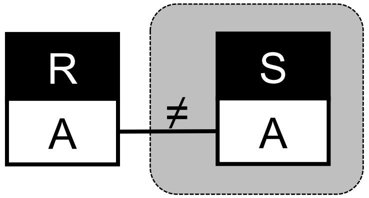
Venn-Peirce diagram

Venn diagrams can't handle (in)equalities

Beta Existential Graph



Relational Diagram

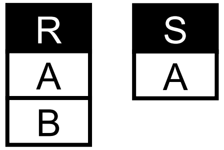


Database to run SQL queries is available as schema 371 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Statement 8: Polyadic FOL

Schema



S8: "There is some R.B that occurs with some R.A that is not in S."

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

TRC (Tuple Relational Calculus)

$\exists r \in R\ [\neg(\exists s \in S\ [r.A = s.A])]$

DRC (Domain Relational Calculus)

$\exists x,y\ [R(y,x) \land \neg S(x)]$

Datalog w/ inequalities

Q() :- R(y,x), not S(x)

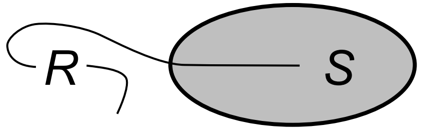
Relational Algebra

$\pi_{\emptyset}(\pi_A(R) - S)$

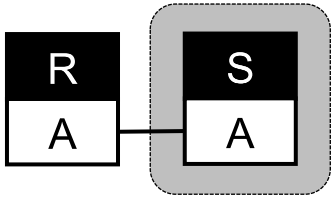
Venn-Peirce diagram

Venn diagrams can't handle polyadic logic

Beta Existential Graph



Relational Diagram

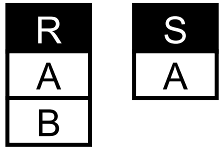


Database to run SQL queries is available as schema 372 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Query 9: Polyadic FOL w/ free variables (= queries)

Schema



Q9: "Find R.B that has some R.A that is not in S."

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

TRC (Tuple Relational Calculus)

{q(B) | ∃r∈R[q.B=r.B ∧ ¬(∃s∈S[r.A=s.A])]}

DRC (Domain Relational Calculus)

{(y) | ∃x [R(x,y) ∧ ¬S(x)]}

Datalog w/ inequalities

Q(y) :- R(y, x), not S(x)

Relational Algebra

$$\mathrm{R} - (S \times \pi_{\mathrm{B}} R)$$

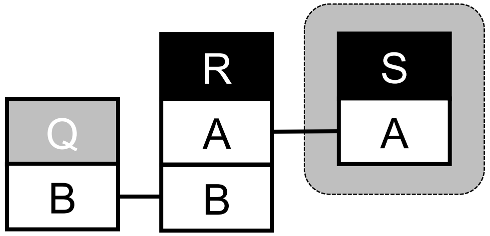
Venn-Peirce diagram

Venn diagrams can't handle polyadic logic, nor free variables

Beta Existential Graph

Beta EGs can't handle free variables

Relational Diagram



Database to run SQL queries is available as schema 372 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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 (from DB community)
6. Lessons Learned and Open Challenges

Part 4: (Early) diagrammatic representations

- We next look at various visual representations for logical statements that were proposed from *outside* the database community, sometimes referred to as the "diagrammatic reasoning" community.
- The attempt to express simple statements (a famous example: "All men are mortal") and to reason based on visual representations predates the database community (and even the formal development of first-order logic) by centuries. But we are trying to bring a "database perspective."

Part 4: Early Diagrammatic Representations

Monadic predicate calculus (Boolean queries & unary predicates)

1. Euler Circles (1768)
2. Venn Diagrams (1880)
3. Venn-Peirce Diagrams (~1896)
4. Venn-Peirce-Shin Diagrams (1995)

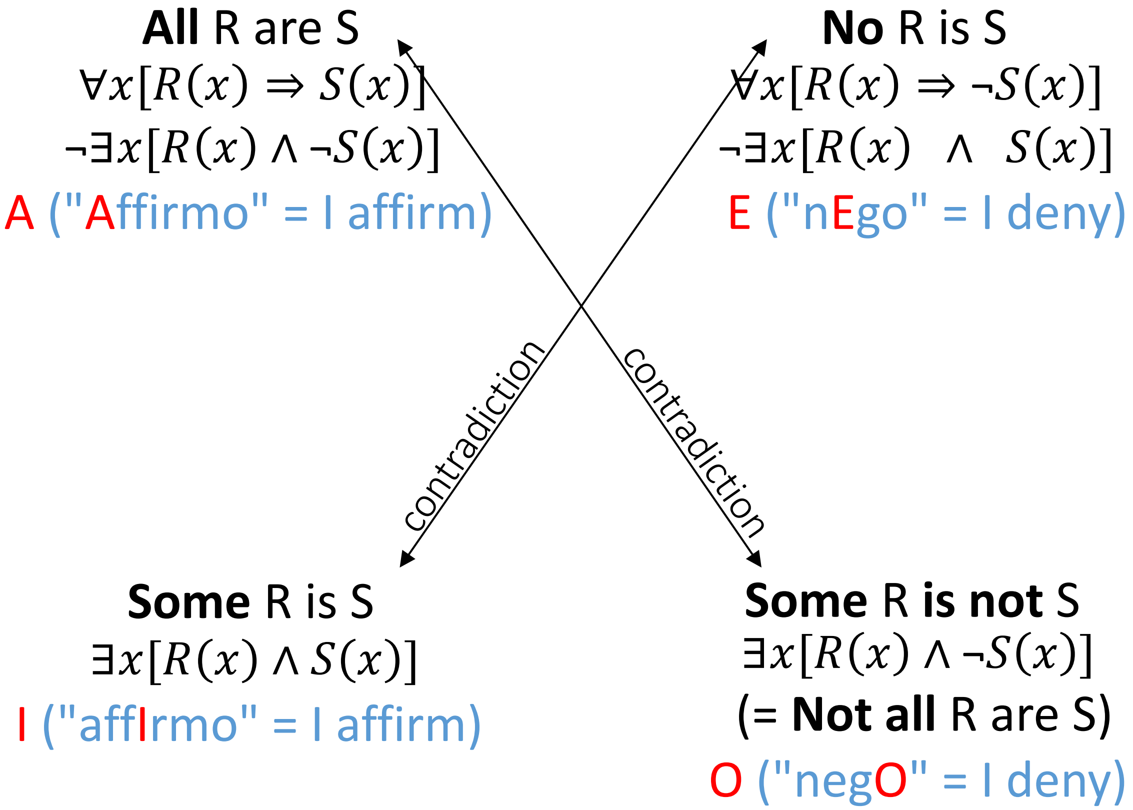
Polyadic predicate calculus (allowing predicates with arities ≥ 2)

7. Peirce Beta Existential Graphs (~1909)
8. Conceptual graphs (1976)
9. String diagrams (2024)

4 categorical propositions / square of opposition

S... subject
P... predicate

Universal propositions



All R are S
$\forall x[R(x) \Rightarrow S(x)]$
$\neg\exists x[R(x) \wedge \neg S(x)]$
A ("Affirmo" = I affirm)

No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$
$\neg\exists x[R(x) \wedge S(x)]$
E ("nEgo" = I deny)

Some R is S
$\exists x[R(x) \wedge S(x)]$
I ("affIrmo" = I affirm)

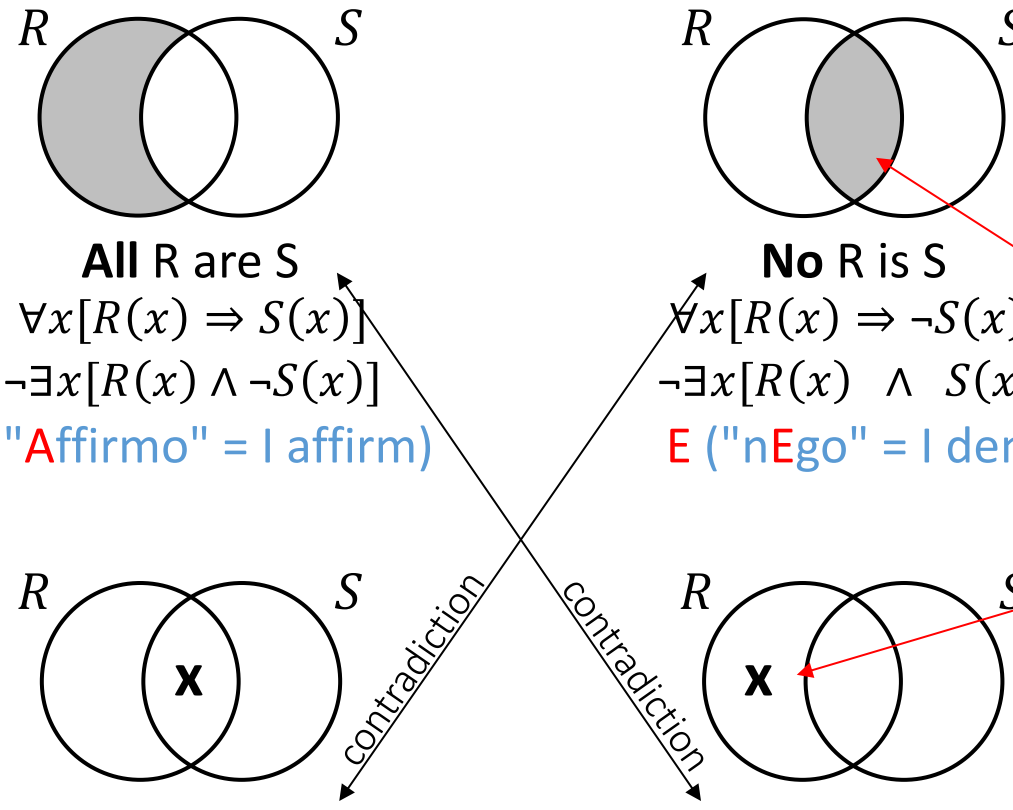
Some R **is not** S
$\exists x[R(x) \wedge \neg S(x)]$
(= **Not all** R are S)
O ("negO" = I deny)

Particular propositions

Abbreviations from Latin. For more details see: https://en.wikipedia.org/wiki/Square_of_opposition

4 categorical propositions / square of opposition

S… subject
P… predicate



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

$\neg\exists x[R(x) \land \neg S(x)]$

A ("Affirmo" = I affirm)

No R is S

$\forall x[R(x) \Rightarrow \neg S(x)]$

$\neg\exists x[R(x) \ \land \ S(x)]$

E ("nEgo" = I deny)

Some R is S

$\exists x[R(x) \land S(x)]$

I ("affIrmo" = I affirm)

Some R **is not** S

$\exists x[R(x) \land \neg S(x)]$

(= **Not all** R are S)

O ("negO" = I deny)

Universal propositions

Particular propositions

Venn-Peirce-Shin: shaded areas are empty, "x" shows that something exists

Abbreviations from Latin. For more details see: https://en.wikipedia.org/wiki/Square_of_opposition

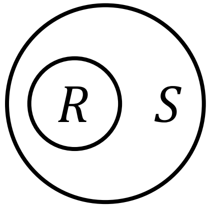
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Euler Circles (1768)

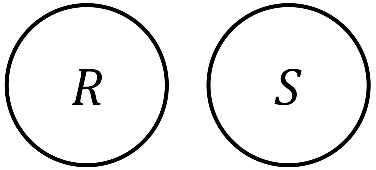
Sources used:

- Euler, Lettres à une princesse d'Allemagne sur divers sujets de physique et de philosophie, 1770. https://www.e-rara.ch/zut/content/zoom/2380256, https://doi.org/10.3931/e-rara-8642
- "Letters to a German Princess, On Different Subjects in Physics and Philosophy (French: Lettres à une princesse d'Allemagne sur divers sujets de physique et de philosophie) were a series of 234 letters written by the mathematician Leonhard Euler between 1760 and 1762 addressed to Friederike Charlotte of Brandenburg-Schwedt and her younger sister Louise ... The first two volumes of the 234 letters originally written in French appeared in print in Saint Petersburg in 1768 and the third in Frankfurt in 1774. The letters were later reprinted in Paris with the first volume in 1787, the second in 1788 and the third in 1789" https://en.wikipedia.org/wiki/Letters_to_a_German_Princess
- Shin. The logical status of diagrams. 1995. https://doi.org/10.1017/CBO9780511574696

Euler: 4 categorical sentences



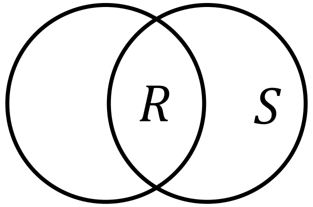
A **All** R are S
$\forall x[R(x) \Rightarrow S(x)]$



No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

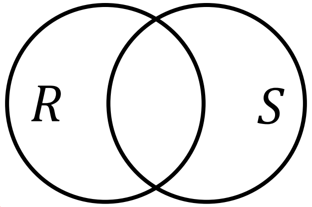
Euler's 1st problem: A and O are contradictory.
But that contradiction is not visible in an obvious way

contradiction



Some R is S
$\exists x[R(x) \wedge S(x)]$

ambiguous!

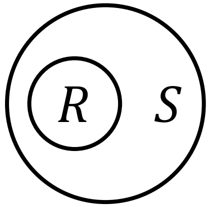


O **Some** R **is not** S
$\exists x[R(x) \wedge \neg S(x)]$

Euler's 2nd problem: "unclear representation of existential statements": A part of concept R is located outside of concept S (but also the other way around)

Problems listed in: [Shin'95]. The logical status of diagrams. 1995. Section 2. https://doi.org/10.1017/CBO9780511574696

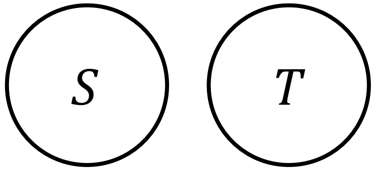
Euler: Application for syllogistic reasoning



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

$\wedge$



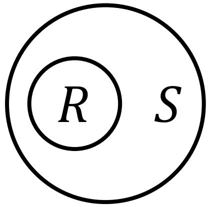
No S is T

$\forall x[S(x) \Rightarrow \neg T(x)]$

$=$?

Problems listed in: [Shin'95]. The logical status of diagrams. 1995. Section 2. https://doi.org/10.1017/CBO9780511574696

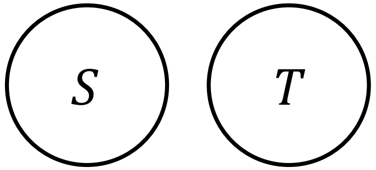
Euler: Application for syllogistic reasoning



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

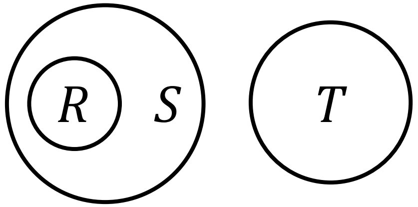
$\wedge$



No S is T

$\forall x[S(x) \Rightarrow \neg T(x)]$

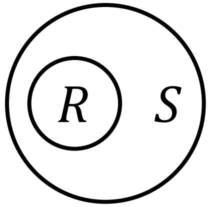
$=$



No R is T

$\forall x[R(x) \Rightarrow \neg T(x)]$

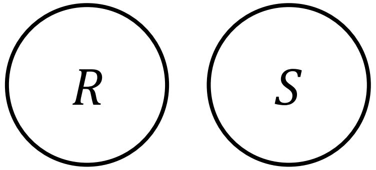
☺



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

$\wedge$



No R is S

$\forall x[R(x) \Rightarrow \neg S(x)]$

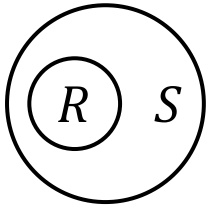
$=$

?

Problems listed in: [Shin'95]. The logical status of diagrams. 1995. Section 2. https://doi.org/10.1017/CBO9780511574696

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

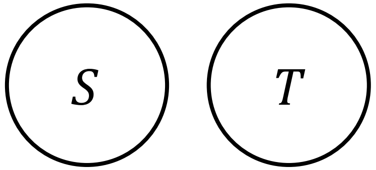
Euler: Application for syllogistic reasoning



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

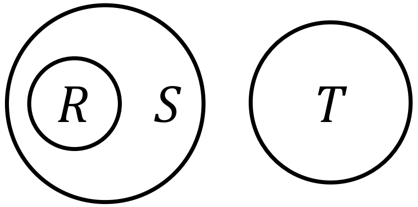
$\land$



No S is T

$\forall x[S(x) \Rightarrow \neg T(x)]$

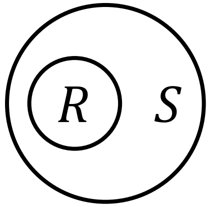
$=$



No R is T

$\forall x[R(x) \Rightarrow \neg T(x)]$

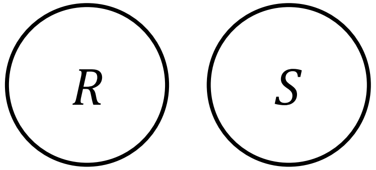
☺



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

$\land$



No R is S

$\forall x[R(x) \Rightarrow \neg S(x)]$

$=$

Euler's 3rd problem: "compatible pieces of information are not always representable in one diagram" (Notice both propositions can be true at the same time if R is empty)

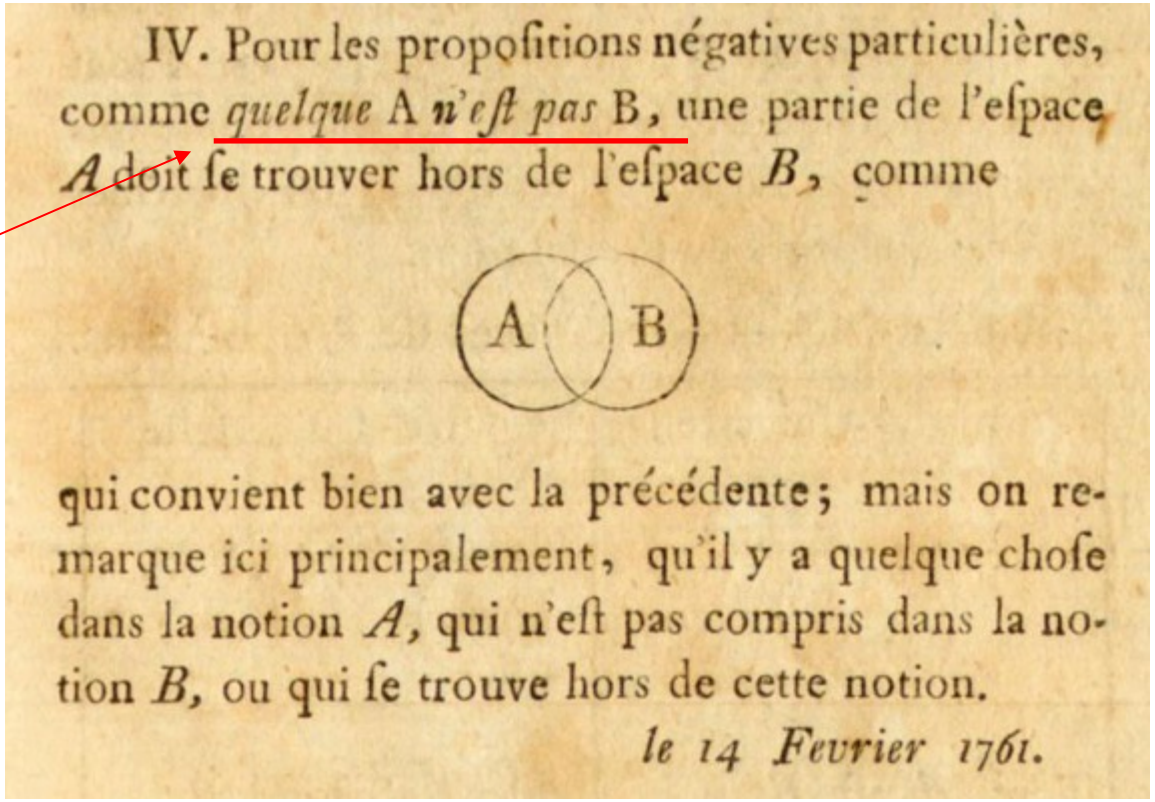
☹

Problems listed in: [Shin'95]. The logical status of diagrams. 1995. Section 2. https://doi.org/10.1017/CBO9780511574696

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Euler (1770)

Some A is not B



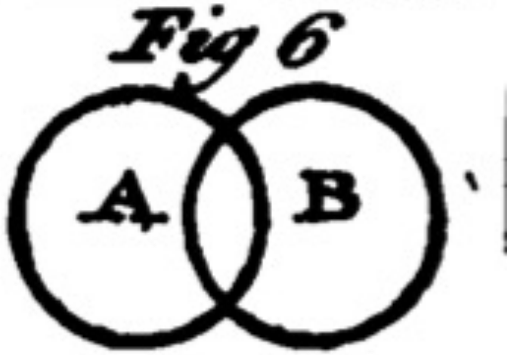
IV. Pour les propoſitions négatives particulières, comme *quelque* A *n'eſt pas* B, une partie de l'eſpace *A* doit ſe trouver hors de l'eſpace *B*, comme

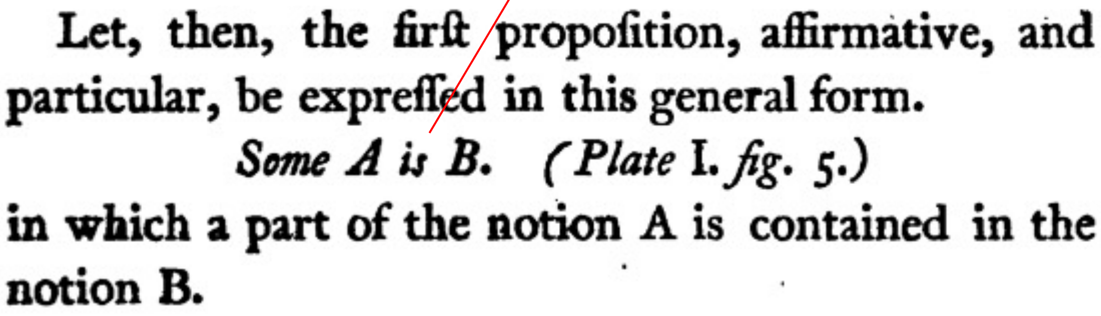
qui convient bien avec la précédente; mais on remarque ici principalement, qu'il y a quelque choſe dans la notion *A*, qui n'eſt pas compris dans la notion *B*, ou qui ſe trouve hors de cette notion.

le 14 Fevrier 1761.

Euler, Lettres à une princesse d'Allemagne sur divers sujets de physique et de philosophie, 1770. https://www.e-rara.ch/zut/content/zoom/2380256, https://doi.org/10.3931/e-rara-8642
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Euler, Hunter (1802)

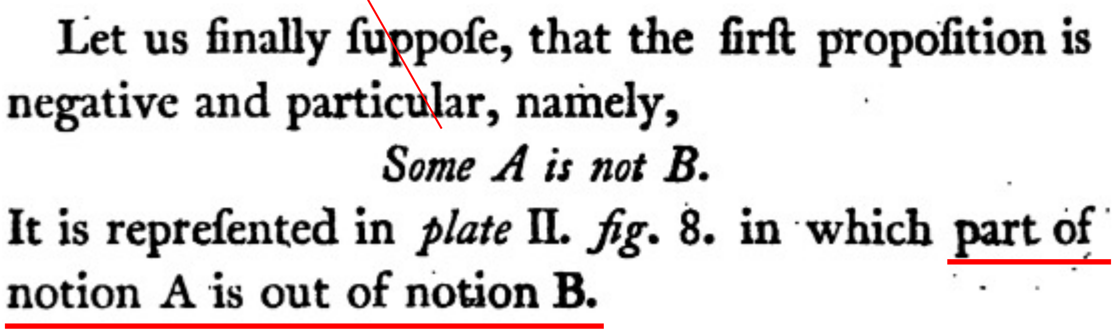




Let, then, the firſt propoſition, affirmative, and particular, be expreſſed in this general form.

Some A is B. (*Plate* I. *fig.* 5.)

in which a part of the notion A is contained in the notion B.



Let us finally ſuppoſe, that the firſt propoſition is negative and particular, namely,

Some A is not B.

It is repreſented in *plate* II. *fig.* 8. in which part of notion A is out of notion B.

Fig 6 / Fig 8: there seems to be an indexing error (?)

Euler (Hunter), Letters of Euler on different subjects in physics and philosophy, addressed to a German princess (book), 1802. p403-405 https://archive.org/details/letterseulertoa00eulegoog/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn Diagrams (1880)

Sources used:

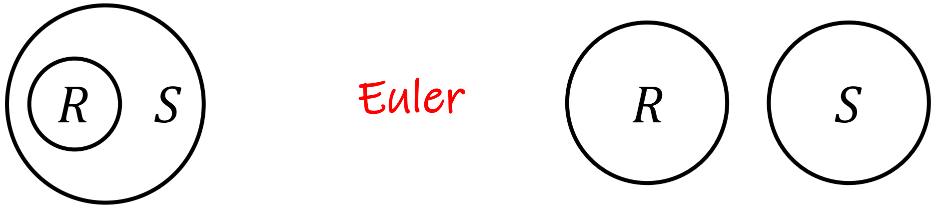
- Venn. On the diagrammatic and mechanical representation of propositions and reasonings, 1880. https://doi.org/10.1080/14786448008626877
- Venn. Symbolic logic, Macmillan, 1881. https://archive.org/details/symboliclogic00venniala/
- Shin. The logical status of diagrams. 1995. https://doi.org/10.1017/CBO9780511574696

Venn

All R are S
$\forall x[R(x) \Rightarrow S(x)]$

No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

VENN manly concerned with representing multiple pieces of information conjunctively in one diagram (Euler's 3rd problem)



Euler

Key problem with Euler: does not allow partial information / uncertainty of information

Idea Venn: start with primary diagram

Venn

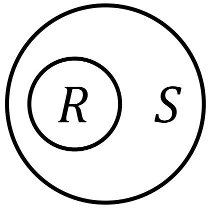
Shading as new syntactic device:
"All, then, that we have to do is to scratch out that subdivision (that is not possible)"

Venn. On the diagrammatic and mechanical representation of propositions and reasonings, 1880. https://doi.org/10.1080/14786448008626877
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn: showing concurrent information

All R are S

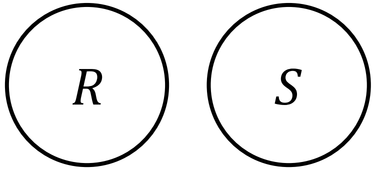
$\forall x[R(x) \Rightarrow S(x)]$



$\wedge$

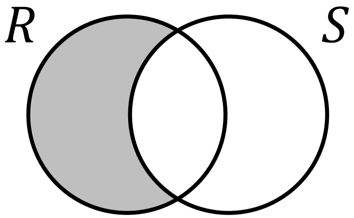
No R is S

$\forall x[R(x) \Rightarrow \neg S(x)]$

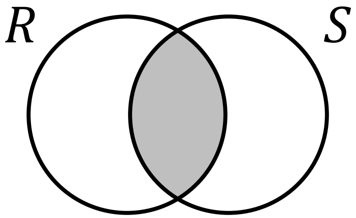


$=$

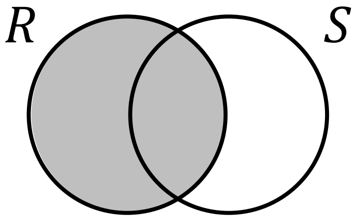
☹



$\wedge$



$=$

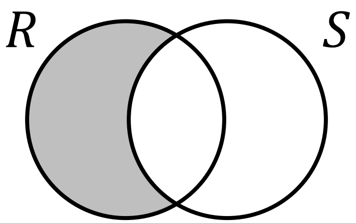


$\neg\exists x[R(x)]$

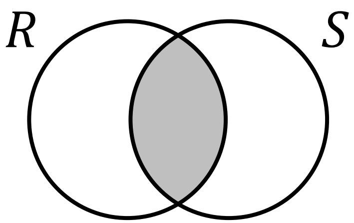
☺

Venn. On the diagrammatic and mechanical representation of propositions and reasonings, 1880. https://doi.org/10.1080/14786448008626877
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn: 4 categorical sentences

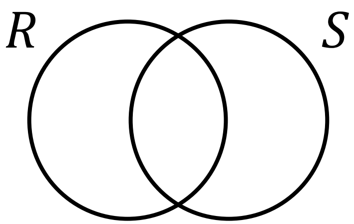


All R are S
$\forall x[R(x) \Rightarrow S(x)]$

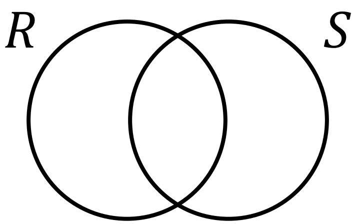


No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

VENN overcame Euler's 3rd problem: representing multiple pieces of non-contradicting information conjunctively in one diagram



Some R is S
$\exists x[R(x) \land S(x)]$

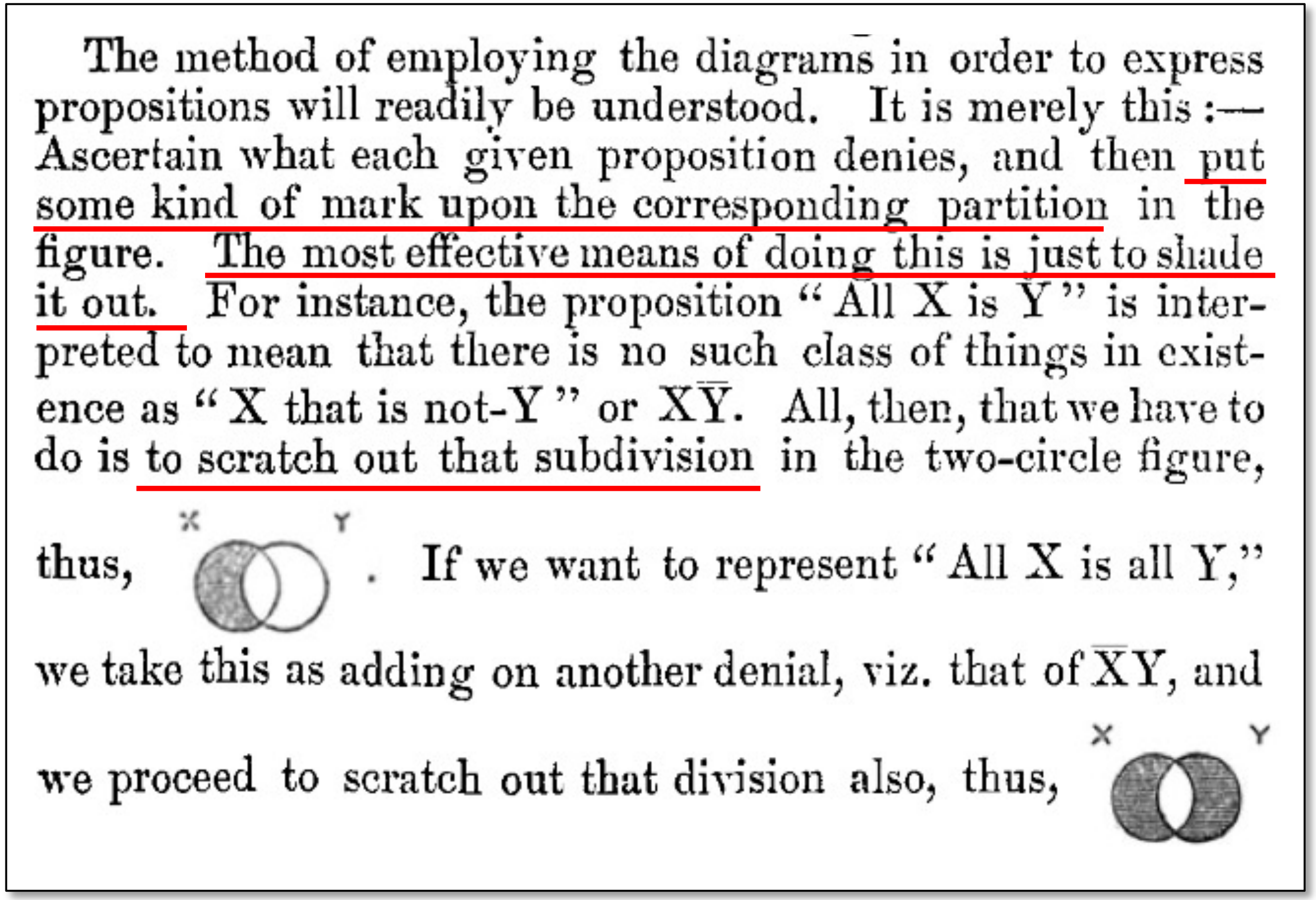


Some R **is not** S
$\exists x[R(x) \land \neg S(x)]$

However, VENN did not overcome Euler's 1st problem: ambiguous representation of existential statements

(next: how Pierce solved that)

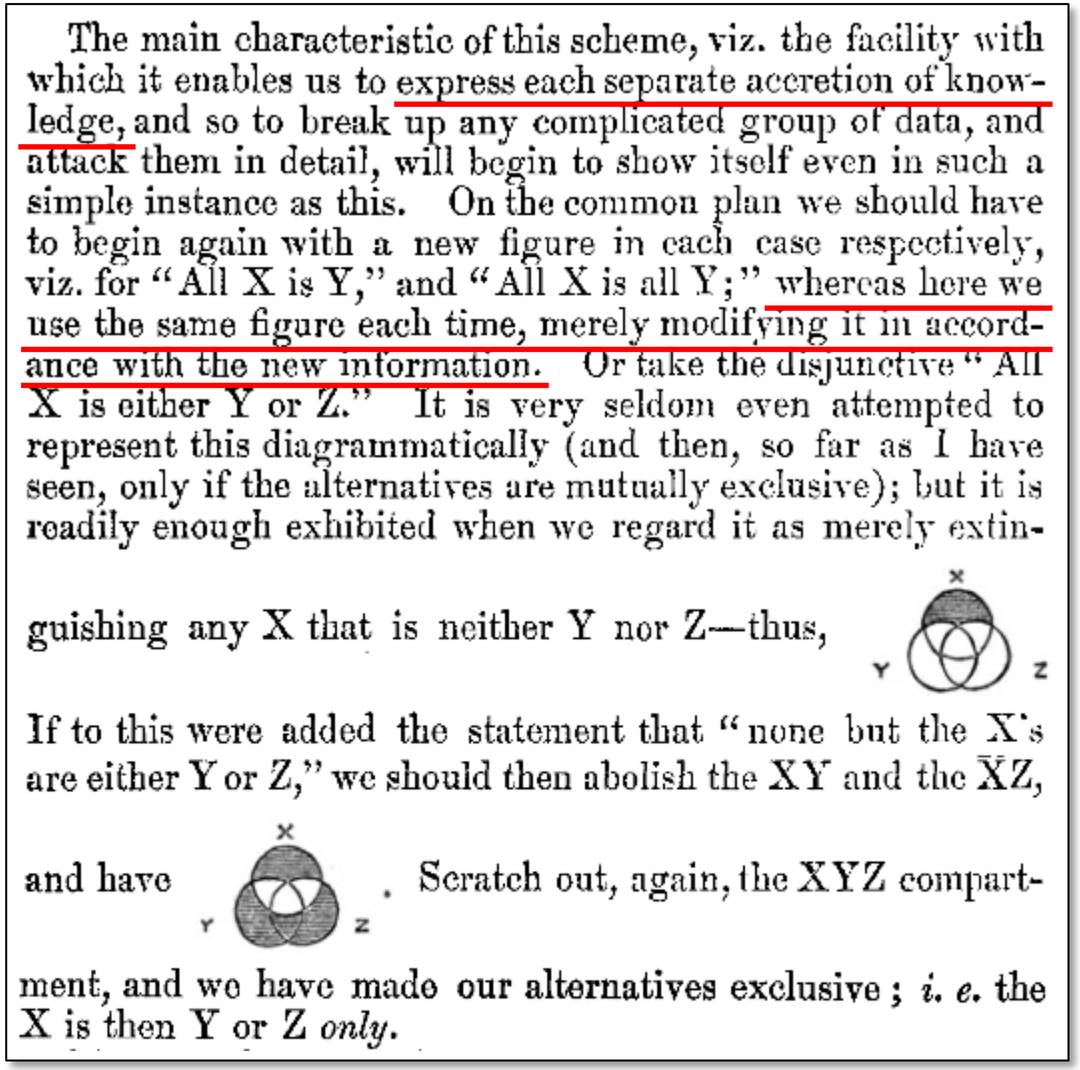
Venn (1880)



The method of employing the diagrams in order to express propositions will readily be understood. It is merely this :— Ascertain what each given proposition denies, and then put some kind of mark upon the corresponding partition in the figure. The most effective means of doing this is just to shade it out. For instance, the proposition " All X is Y " is interpreted to mean that there is no such class of things in existence as " X that is not-Y " or X$\bar{Y}$. All, then, that we have to do is to scratch out that subdivision in the two-circle figure, thus, . If we want to represent " All X is all Y," we take this as adding on another denial, viz. that of $\bar{X}$Y, and we proceed to scratch out that division also, thus,

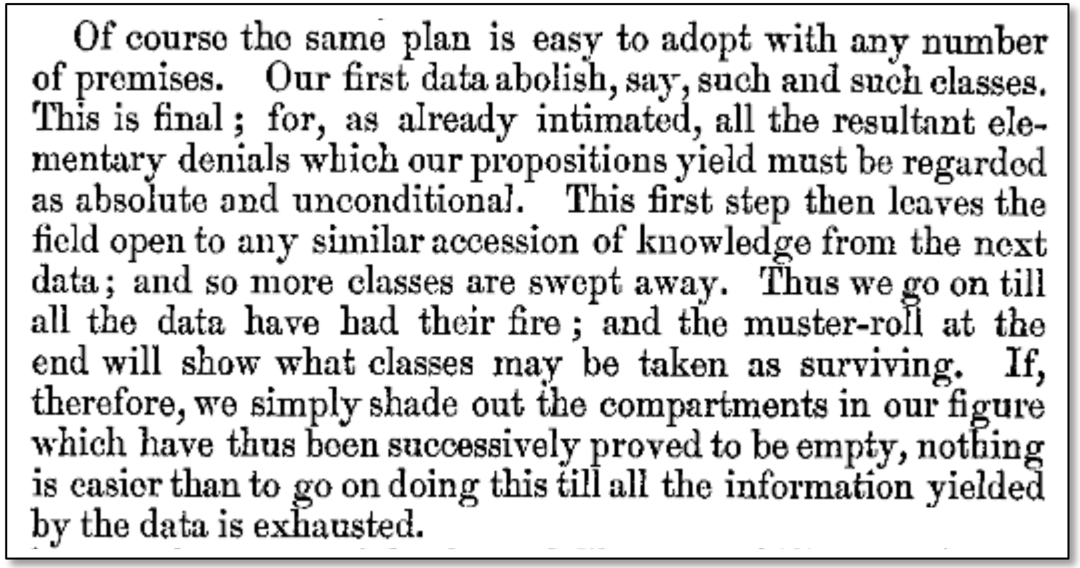
Venn. On the diagrammatic and mechanical representation of propositions and reasonings, 1880. https://doi.org/10.1080/14786448008626877

Venn (1880)



The main characteristic of this scheme, viz. the facility with which it enables us to express each separate accretion of knowledge, and so to break up any complicated group of data, and attack them in detail, will begin to show itself even in such a simple instance as this. On the common plan we should have to begin again with a new figure in each case respectively, viz. for "All X is Y," and "All X is all Y;" whereas here we use the same figure each time, merely modifying it in accordance with the new information. Or take the disjunctive "All X is either Y or Z." It is very seldom even attempted to represent this diagrammatically (and then, so far as I have seen, only if the alternatives are mutually exclusive); but it is readily enough exhibited when we regard it as merely extinguishing any X that is neither Y nor Z—thus,

If to this were added the statement that "none but the X's are either Y or Z," we should then abolish the XY and the $\bar{X}Z$, and have . Scratch out, again, the XYZ compartment, and we have made our alternatives exclusive; *i. e.* the X is then Y or Z *only*.



Of course the same plan is easy to adopt with any number of premises. Our first data abolish, say, such and such classes. This is final; for, as already intimated, all the resultant elementary denials which our propositions yield must be regarded as absolute and unconditional. This first step then leaves the field open to any similar accession of knowledge from the next data; and so more classes are swept away. Thus we go on till all the data have had their fire; and the muster-roll at the end will show what classes may be taken as surviving. If, therefore, we simply shade out the compartments in our figure which have thus been successively proved to be empty, nothing is easier than to go on doing this till all the information yielded by the data is exhausted.

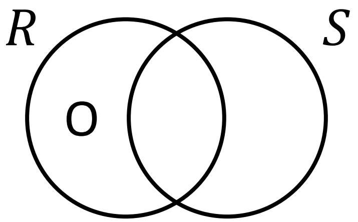
Venn. On the diagrammatic and mechanical representation of propositions and reasonings, 1880. https://doi.org/10.1080/14786448008626877
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce (~1896)

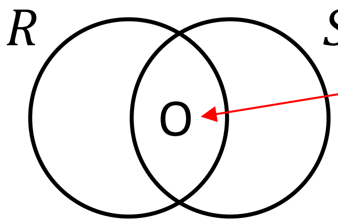
Sources used:

- Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraphs 4.350 – 4.371. https://archive.org/details/collectedpaperso0000unse_r5j9/
- Shin. The logical status of diagrams. 1995. https://doi.org/10.1017/CBO9780511574696

Venn-Peirce: 4 categorical sentences

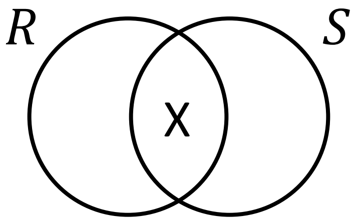


All R are S
$\forall x[R(x) \Rightarrow S(x)]$

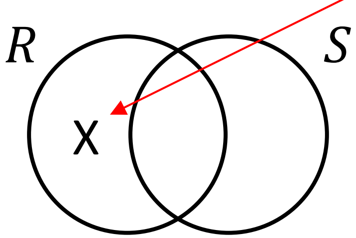


No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

1) "O" instead of shading: This change gives an "anchor" to the symbol "O", which becomes important later for disjunctive information ("O" stands for zero)



Some R is S
$\exists x[R(x) \wedge S(x)]$

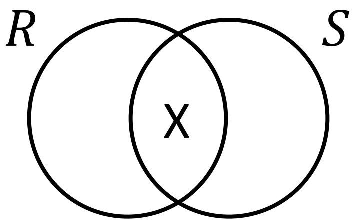


Some R **is not** S
$\exists x[R(x) \wedge \neg S(x)]$

2) "X" for exists: This visual device overcomes Euler's and Venn's unclear representation of existential statements ("X" stands for there is something)

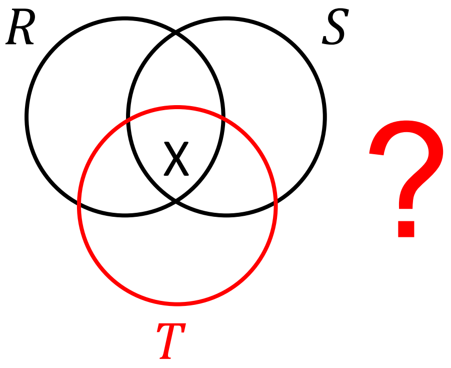
Venn-Peirce: adding predicates

How can we express the same information from the left (not more and not less) if we just add an additional predicate T?



Some R is S

$\exists x[R(x) \wedge S(x)]$

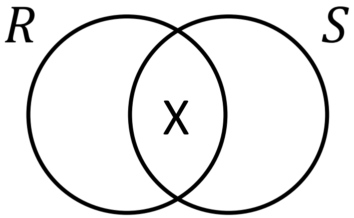


Some R is S

$\exists x[R(x) \wedge S(x)]$

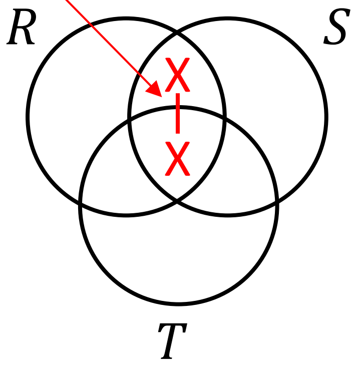
Venn-Peirce: adding predicates via disjunctions

A line connecting anchors as syntactic device to express disjunctive information. Thus connected assertions are made alternatively.



Some R is S

$\exists x[R(x) \land S(x)]$

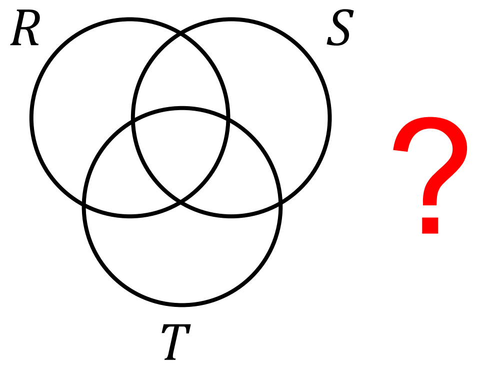


Some R is S

$\exists x[R(x) \land S(x)]$

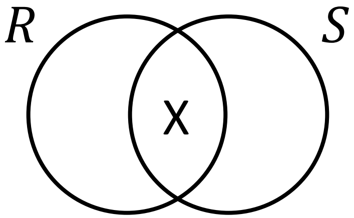
$\exists x[R(x) \land S(x) \land \neg T(x)]$
$\lor \exists y[R(y) \land S(y) \land T(y)]$

Venn-Peirce: conjunctions



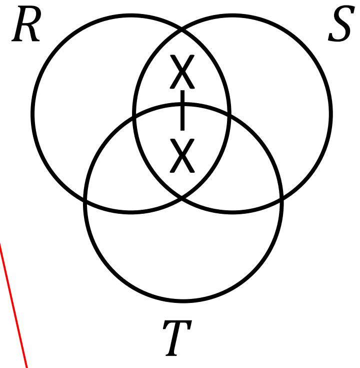
Some R is S, and **some** R is T

$\underbrace{\exists x[R(x) \wedge S(x)]} \wedge \exists z[R(z) \wedge T(z)]$



Some R is S

$\exists x[R(x) \wedge S(x)]$

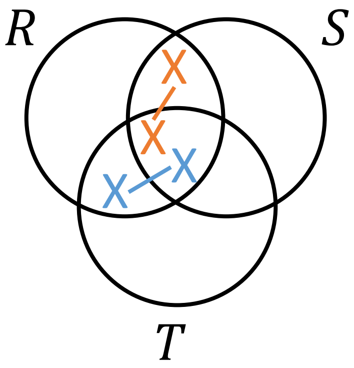


Some R is S

$\exists x[R(x) \wedge S(x)]$

Venn-Peirce: conjunctions = juxtaposition

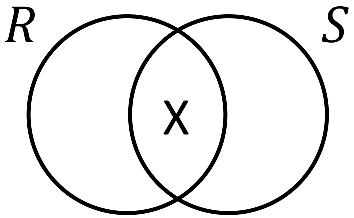
$\exists x[R(x) \land S(x) \land \neg T(x)]$
$\lor \exists y[R(y) \land S(y) \land T(y)]$



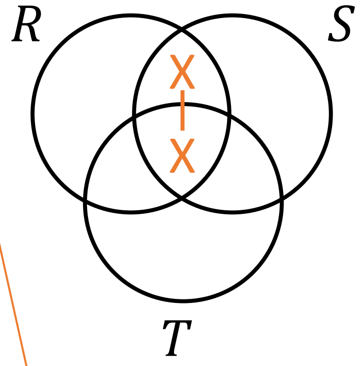
$\exists z[R(z) \land S(z) \land \neg T(z)]$
$\lor \exists v[R(v) \land S(v) \land T(v)]$

Some R is S, and **some** R is T
$\underbrace{\exists x[R(x) \land S(x)]}_{} \land \exists z[R(z) \land T(z)]$

Conjunction is just juxtaposition!

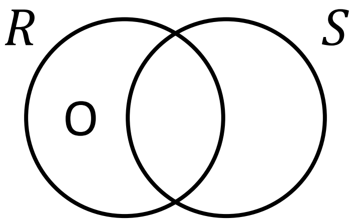


Some R is S
$\exists x[R(x) \land S(x)]$



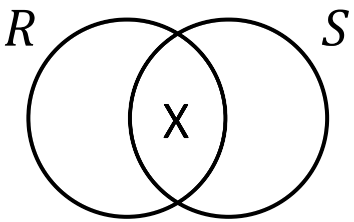
Some R is S
$\exists x[R(x) \land S(x)]$

Venn-Peirce: disjunction



All R are S

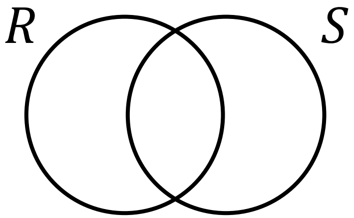
$\forall x[R(x) \Rightarrow S(x)]$



Some R is S

$\exists x[R(x) \wedge S(x)]$

disjunction



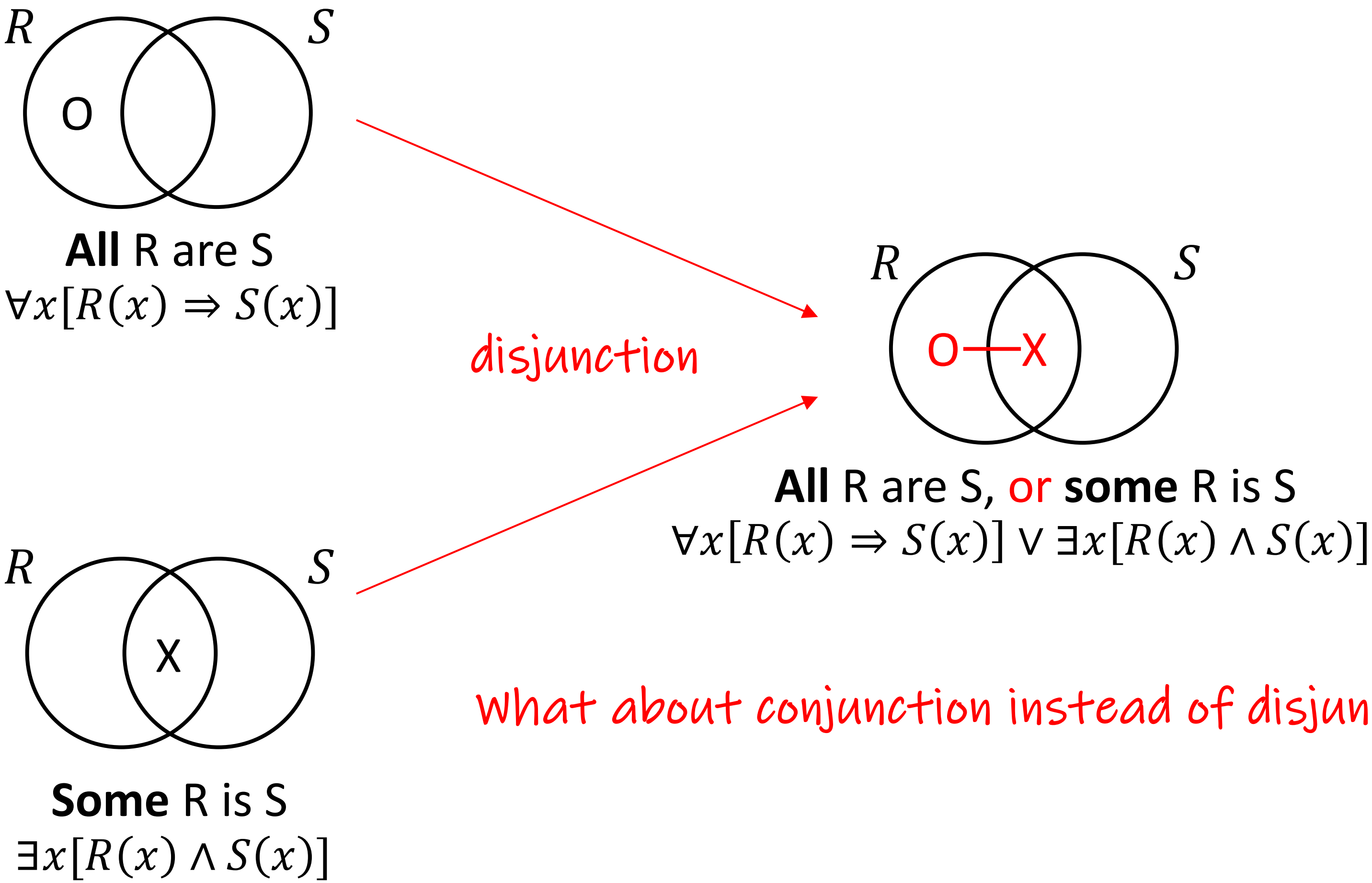
?

All R are S, or **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \vee \exists x[R(x) \wedge S(x)]$

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce: disjunction



All R are S

$\forall x[R(x) \Rightarrow S(x)]$

disjunction

All R are S, or **some** R is S

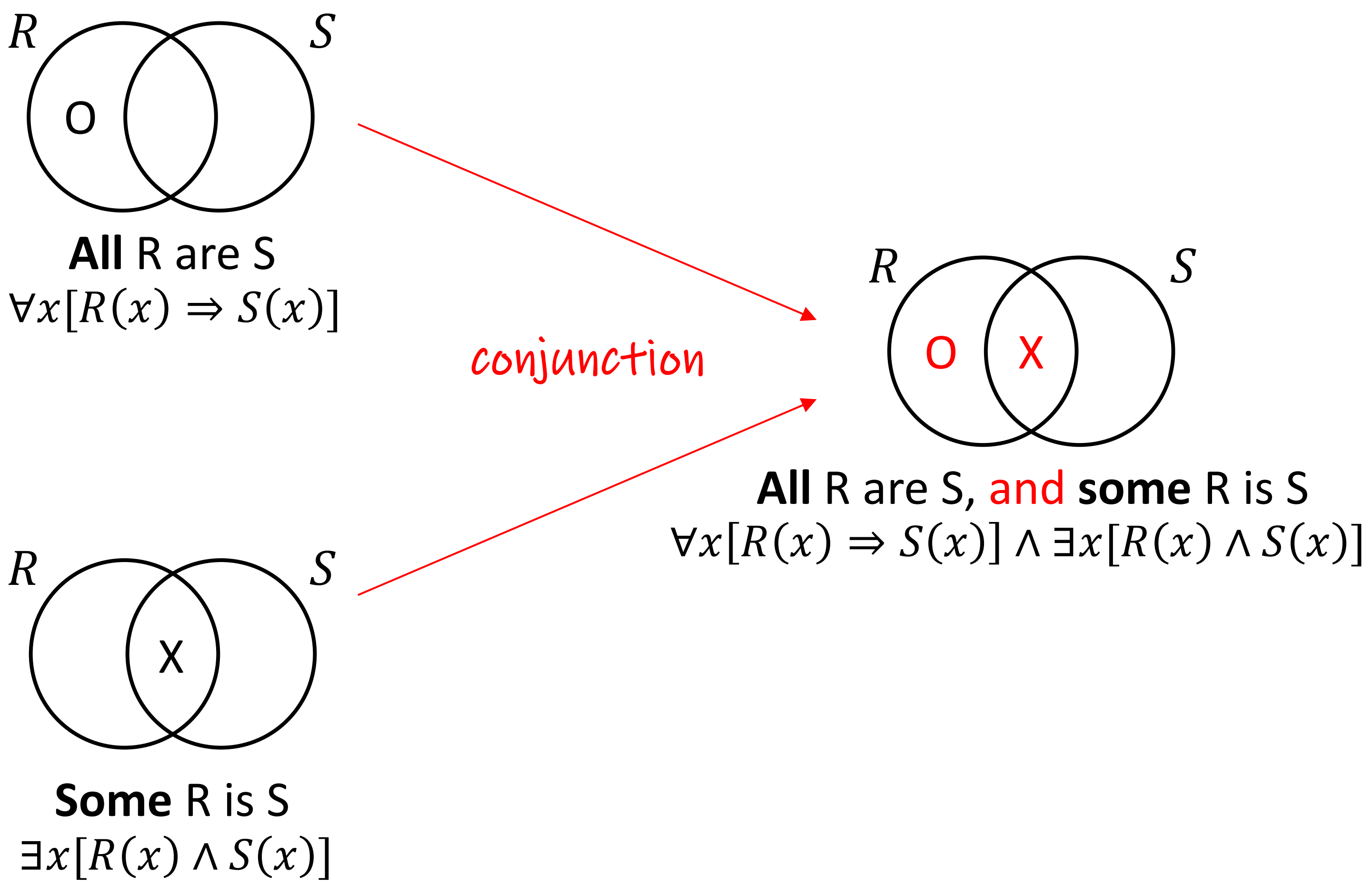
$\forall x[R(x) \Rightarrow S(x)] \lor \exists x[R(x) \land S(x)]$

Some R is S

$\exists x[R(x) \land S(x)]$

What about conjunction instead of disjunction ?

Venn-Peirce: conjunction

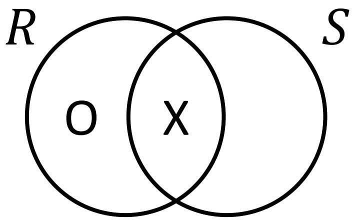


All R are S
$\forall x[R(x) \Rightarrow S(x)]$

Some R is S
$\exists x[R(x) \wedge S(x)]$

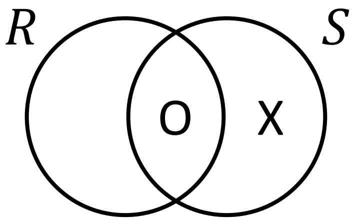
All R are S, and **some** R is S
$\forall x[R(x) \Rightarrow S(x)] \wedge \exists x[R(x) \wedge S(x)]$

Venn-Peirce: disjunction of conjunctions



All R are S, and **some** R is S

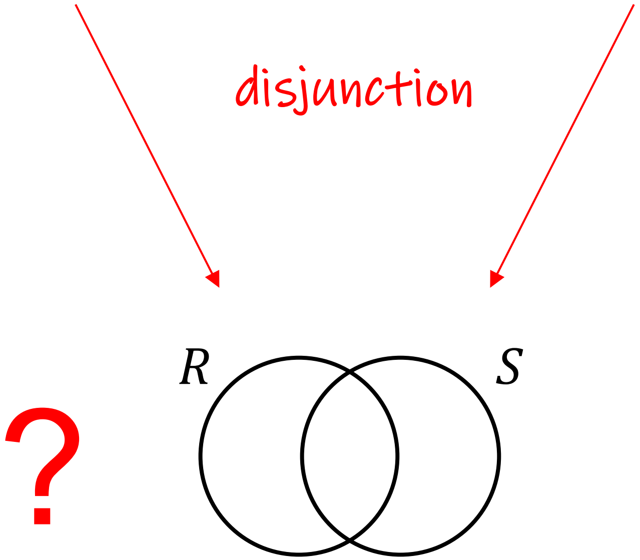
$\forall x[R(x) \Rightarrow S(x)] \wedge \exists x[R(x) \wedge S(x)]$



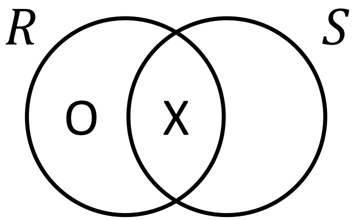
No R is S, and **some** S **is not** R

$\forall x[R(x) \Rightarrow \neg S(x)] \ \wedge \exists x[S(x) \wedge \neg R(x)]$

$(\forall x[R(x) \Rightarrow \ \ S(x)] \wedge \exists x[R(x) \wedge \ \ S(x)]) \ \vee$
$(\forall x[R(x) \Rightarrow \neg S(x)] \wedge \exists x[S(x) \wedge \neg R(x)])$

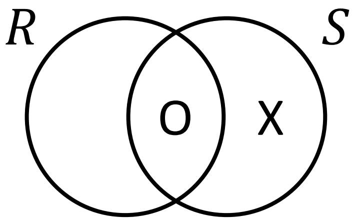


Venn-Peirce: disjunction of conjunctions



All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]$



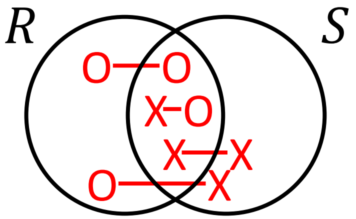
No R is S, and **some** S **is not** R

$\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)]$

disjunction

~~DNF~~

$(\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]) \lor$
$(\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)])$



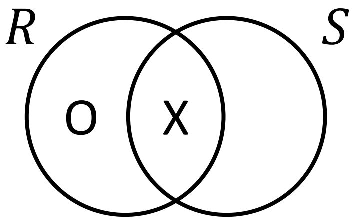
CNF

$(\forall x[R(x) \Rightarrow S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$
$(\forall x[R(x) \Rightarrow S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$

"It is only disjunctions of conjunctions that cause some inconvenience" (Peirce [~1896])

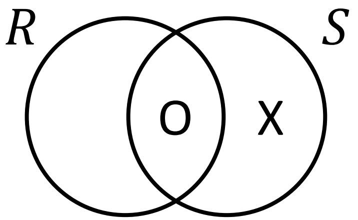
Citation from "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/ "

Venn-Peirce: disjunction of conjunctions



All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]$



No R is S, and **some** S **is not** R

$\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)]$

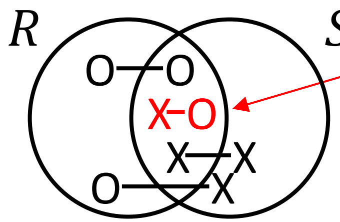
disjunction

~~DNF~~

$(\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]) \lor$
$(\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)])$

CNF

$(\forall x[R(x) \Rightarrow S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$
$(\forall x[R(x) \Rightarrow S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$

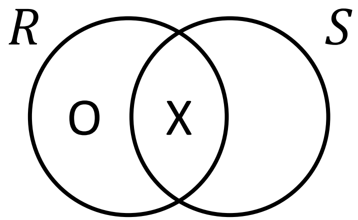


Next we can directly manipulate the diagram ("diagrammatic reasoning")

"It is only disjunctions of conjunctions that cause some inconvenience" (Peirce [~1896])

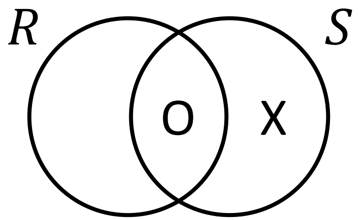
Citation from "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/ "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce: disjunction of conjunctions



All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]$



No R is S, and **some** S **is not** R

$\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)]$

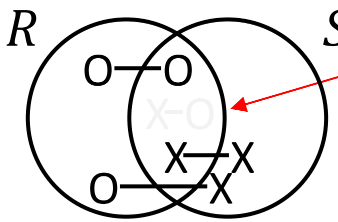
disjunction

~~DNF~~

$(\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]) \lor$
$(\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)])$

CNF

$(\forall x[R(x) \Rightarrow S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \forall x[R(x) \Rightarrow \neg S(x)]) \land$
$(\exists x[R(x) \land S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$
$(\forall x[R(x) \Rightarrow S(x)] \lor \exists x[S(x) \land \neg R(x)]) \land$



Next we can directly manipulate the diagram ("diagrammatic reasoning")

"It is only disjunctions of conjunctions that cause some inconvenience" (Peirce [~1896])

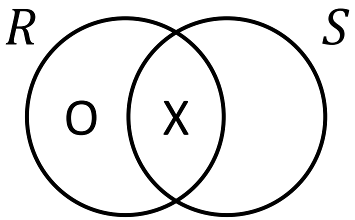
"There is, however, a very easy and very useful way of avoiding this" (Peirce [~1896])

?

Citation from "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/ "

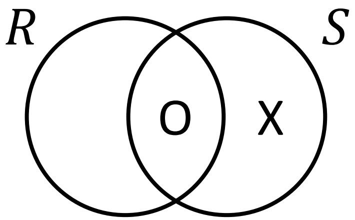
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce: disjunction of conjunctions in compartments



All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \wedge \exists x[R(x) \wedge S(x)]$



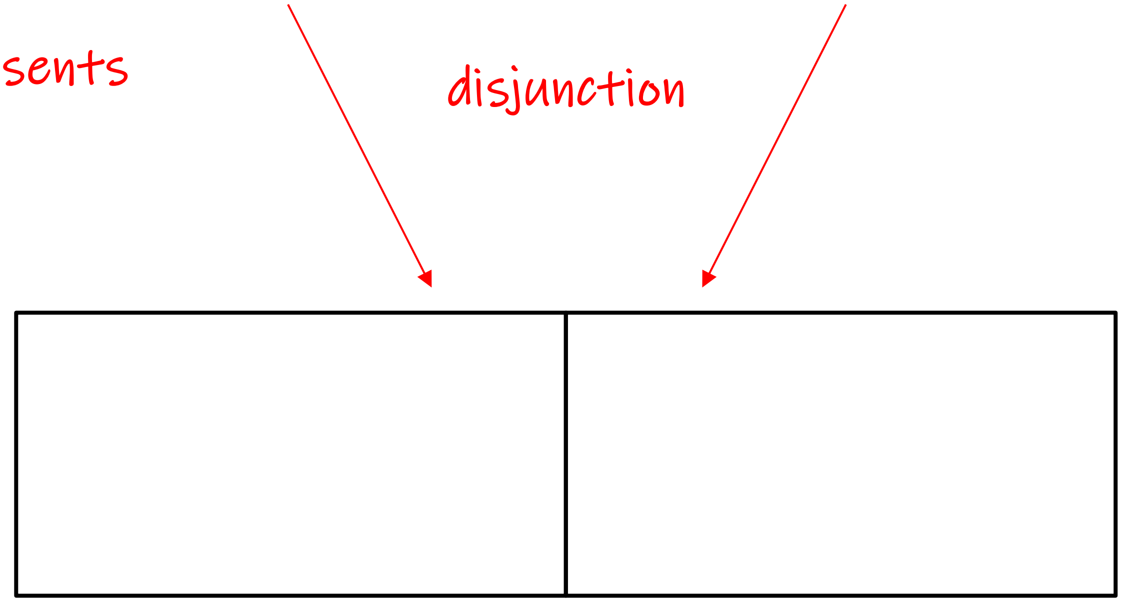
No R is S, and **some** S **is not** R

$\forall x[R(x) \Rightarrow \neg S(x)] \wedge \exists x[S(x) \wedge \neg R(x)]$

- Each "compartment" represents conjunctive information.

DNF

disjunction



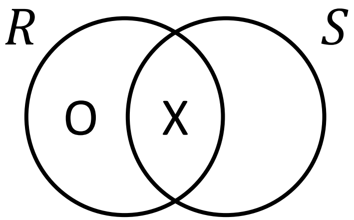
~~CNF~~

"It is only disjunctions of conjunctions that cause some inconvenience" (Peirce [~1896])

"There is, however, a very easy and very useful way of avoiding this"

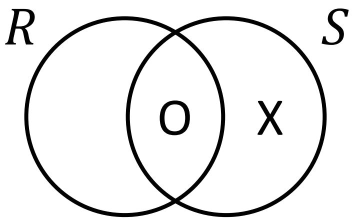
Citation from "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/ "

Venn-Peirce: disjunction of conjunctions in compartments



All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \land \exists x[R(x) \land S(x)]$



No R is S, and **some** S **is not** R

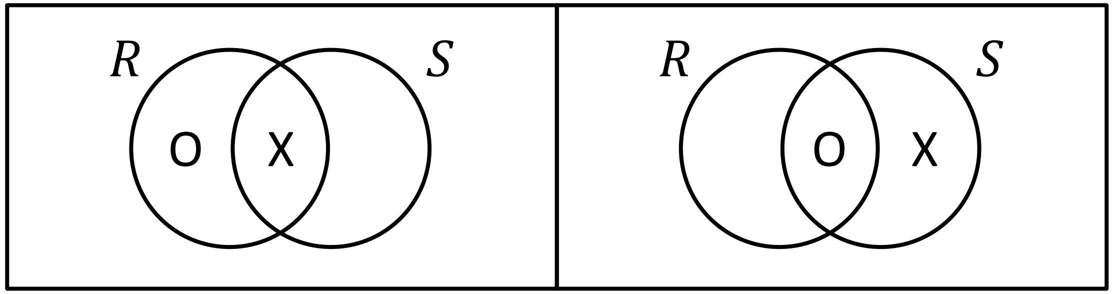
$\forall x[R(x) \Rightarrow \neg S(x)] \land \exists x[S(x) \land \neg R(x)]$

DNF

- Each "compartment" represents conjunctive information.
- Different "compartments" represent different cases (disjunctions)

disjunction

~~CNF~~



"It is only disjunctions of conjunctions that cause some inconvenience" (Peirce [~1896])

"There is, however, a very easy and very useful way of avoiding this"

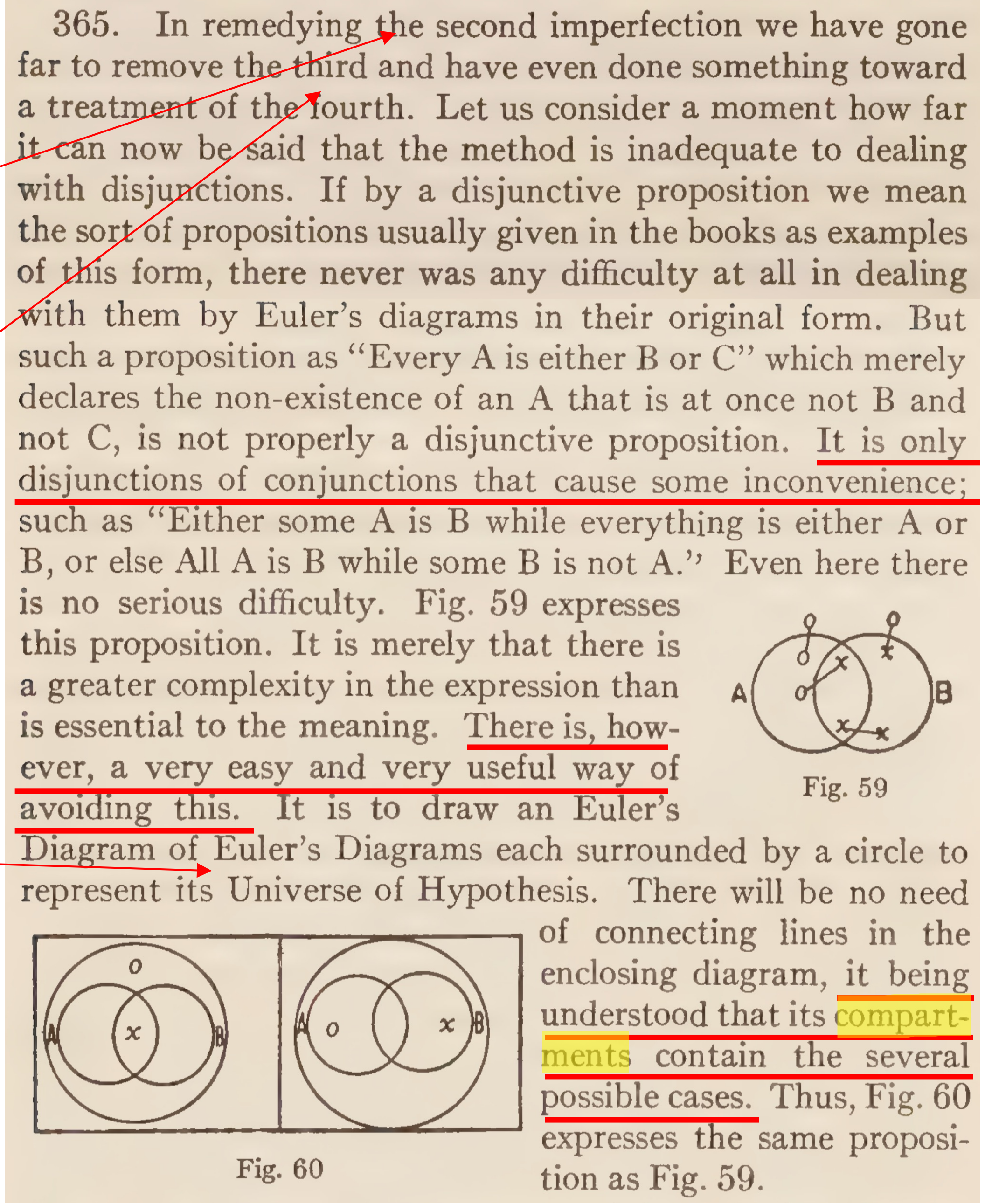
Citations from "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/ "

Venn-Peirce [Peirce 1896]

2nd imperfection: Euler cannot affirm the existence of objects. Remedied with (x)

3rd imperfection: Euler cannot express that one or another of several alternative states of things occurs.

Notice a slight inconsistency with the Euler's diagram reference: two disjoint circles would represent mutual exclusive statements (e.g. R disjoint with S means "no R is S"). Here, in contrast, compartments represent non-exclusive cases (in contrast with a "possible world semantics" in which worlds are mutually exclusive)



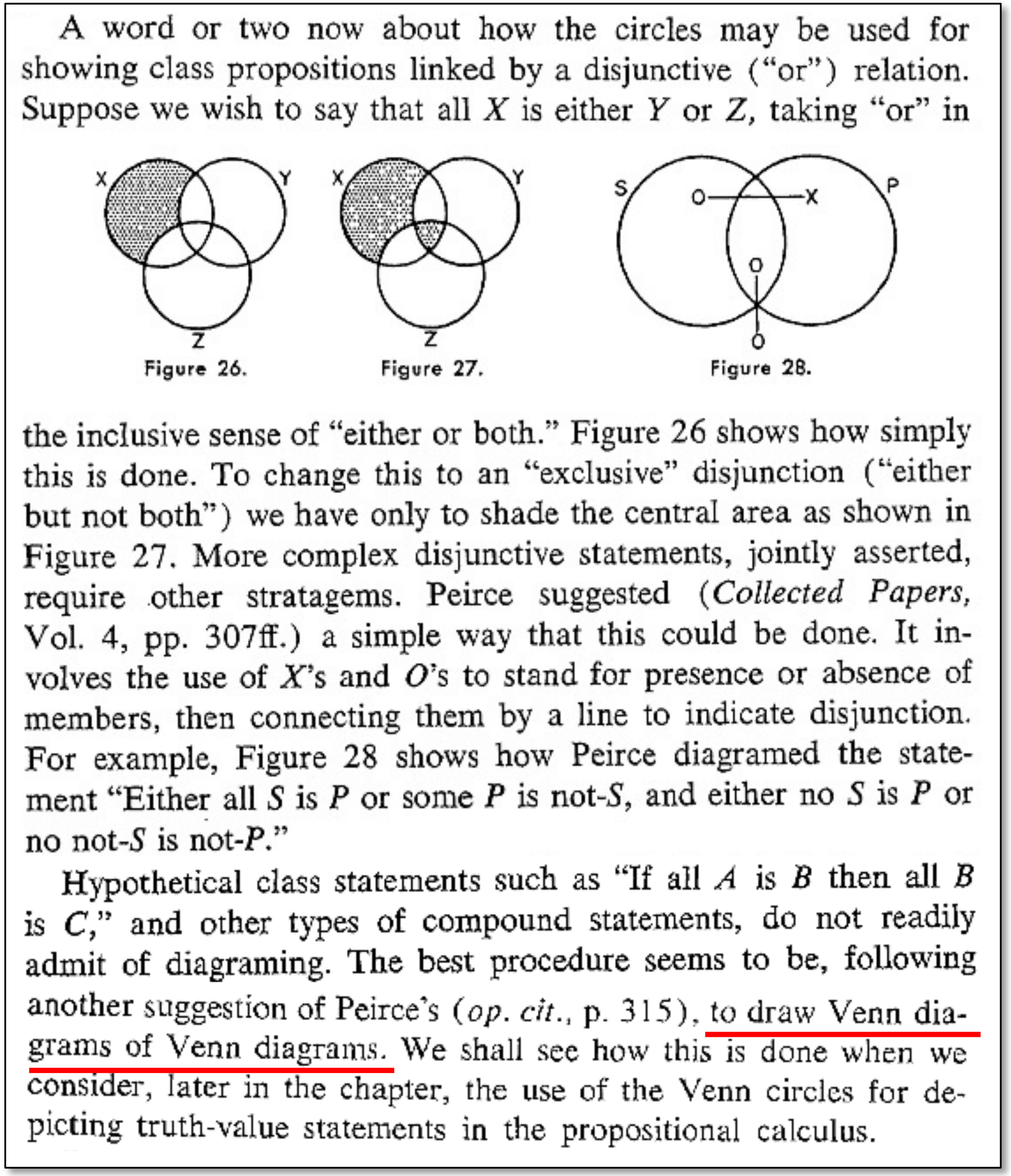
365. In remedying the second imperfection we have gone far to remove the third and have even done something toward a treatment of the fourth. Let us consider a moment how far it can now be said that the method is inadequate to dealing with disjunctions. If by a disjunctive proposition we mean the sort of propositions usually given in the books as examples of this form, there never was any difficulty at all in dealing with them by Euler's diagrams in their original form. But such a proposition as "Every A is either B or C" which merely declares the non-existence of an A that is at once not B and not C, is not properly a disjunctive proposition. It is only disjunctions of conjunctions that cause some inconvenience; such as "Either some A is B while everything is either A or B, or else All A is B while some B is not A." Even here there is no serious difficulty. Fig. 59 expresses this proposition. It is merely that there is a greater complexity in the expression than is essential to the meaning. There is, however, a very easy and very useful way of avoiding this. It is to draw an Euler's Diagram of Euler's Diagrams each surrounded by a circle to represent its Universe of Hypothesis. There will be no need of connecting lines in the enclosing diagram, it being understood that its compartments contain the several possible cases. Thus, Fig. 60 expresses the same proposition as Fig. 59.

Fig. 59

Fig. 60

Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.365. https://archive.org/details/collectedpaperso0000unse_r5j9/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce [Gardner 1958]



A word or two now about how the circles may be used for showing class propositions linked by a disjunctive ("or") relation. Suppose we wish to say that all *X* is either *Y* or *Z*, taking "or" in

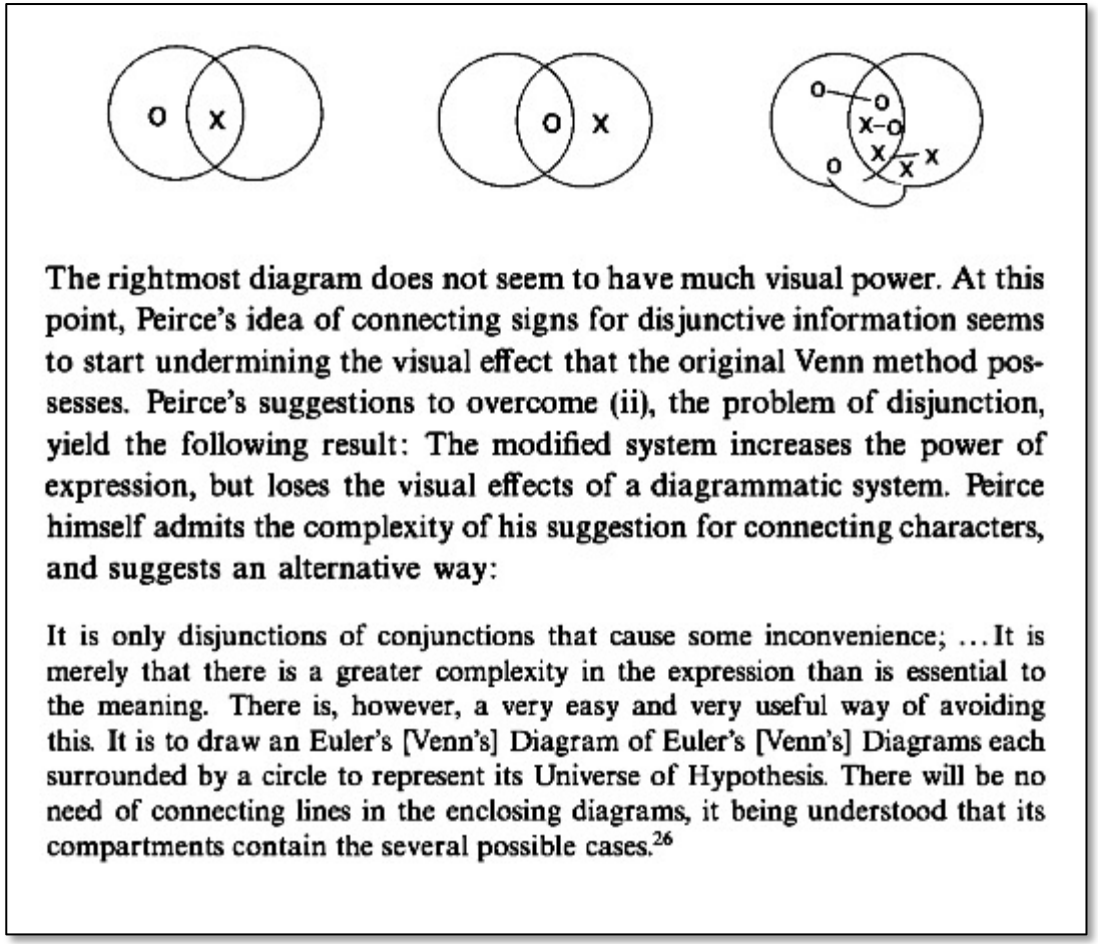
Figure 26. Figure 27. Figure 28.

the inclusive sense of "either or both." Figure 26 shows how simply this is done. To change this to an "exclusive" disjunction ("either but not both") we have only to shade the central area as shown in Figure 27. More complex disjunctive statements, jointly asserted, require other stratagems. Peirce suggested (*Collected Papers*, Vol. 4, pp. 307ff.) a simple way that this could be done. It involves the use of *X*'s and *O*'s to stand for presence or absence of members, then connecting them by a line to indicate disjunction. For example, Figure 28 shows how Peirce diagramed the statement "Either all *S* is *P* or some *P* is not-*S*, and either no *S* is *P* or no not-*S* is not-*P*."

Hypothetical class statements such as "If all *A* is *B* then all *B* is *C*," and other types of compound statements, do not readily admit of diagraming. The best procedure seems to be, following another suggestion of Peirce's (*op. cit.*, p. 315), to draw Venn diagrams of Venn diagrams. We shall see how this is done when we consider, later in the chapter, the use of the Venn circles for depicting truth-value statements in the propositional calculus.

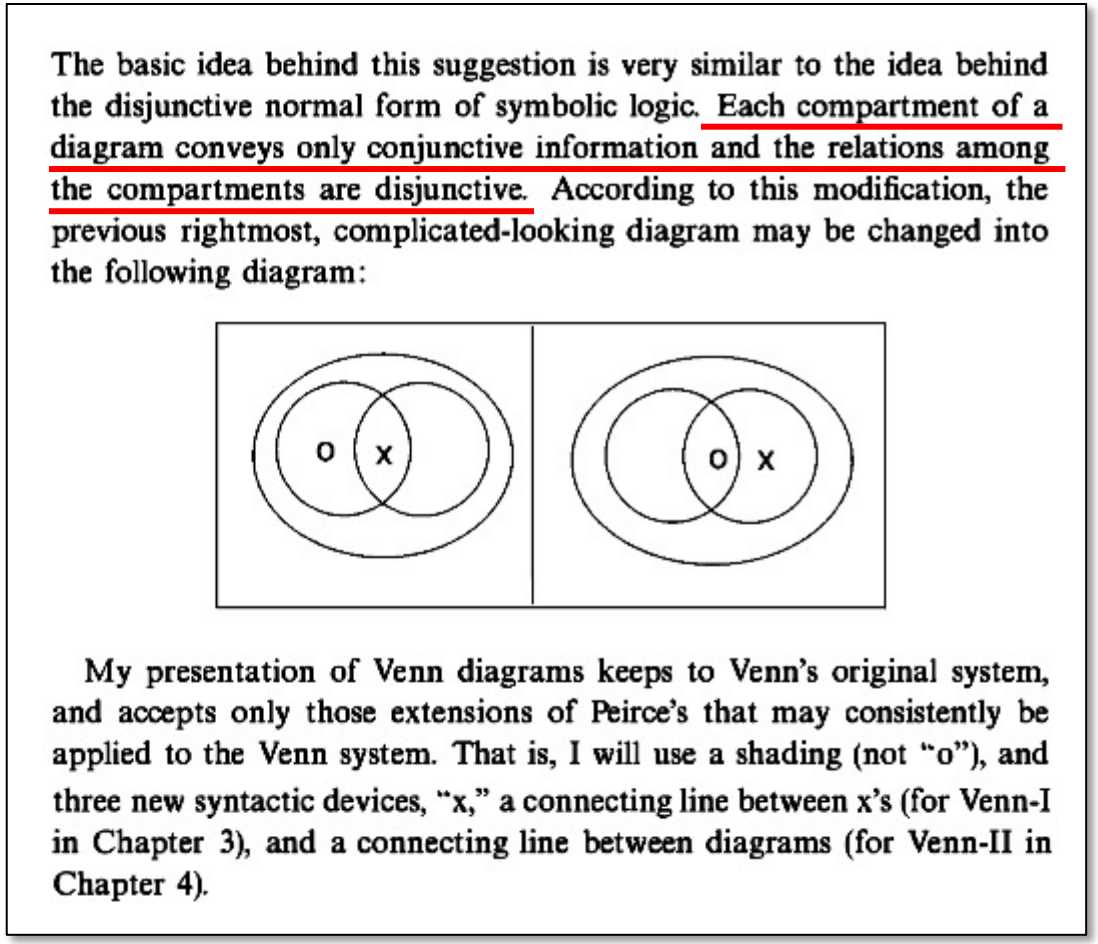
1958 - Gardner - Logic machines and diagrams (book). https://archive.org/details/logicmachinesdia00mart

Venn-Peirce [Shin 1995]



The rightmost diagram does not seem to have much visual power. At this point, Peirce's idea of connecting signs for disjunctive information seems to start undermining the visual effect that the original Venn method possesses. Peirce's suggestions to overcome (ii), the problem of disjunction, yield the following result: The modified system increases the power of expression, but loses the visual effects of a diagrammatic system. Peirce himself admits the complexity of his suggestion for connecting characters, and suggests an alternative way:

> It is only disjunctions of conjunctions that cause some inconvenience; …It is merely that there is a greater complexity in the expression than is essential to the meaning. There is, however, a very easy and very useful way of avoiding this. It is to draw an Euler's [Venn's] Diagram of Euler's [Venn's] Diagrams each surrounded by a circle to represent its Universe of Hypothesis. There will be no need of connecting lines in the enclosing diagrams, it being understood that its compartments contain the several possible cases.[26]



The basic idea behind this suggestion is very similar to the idea behind the disjunctive normal form of symbolic logic. Each compartment of a diagram conveys only conjunctive information and the relations among the compartments are disjunctive. According to this modification, the previous rightmost, complicated-looking diagram may be changed into the following diagram:

My presentation of Venn diagrams keeps to Venn's original system, and accepts only those extensions of Peirce's that may consistently be applied to the Venn system. That is, I will use a shading (not "o"), and three new syntactic devices, "x," a connecting line between x's (for Venn-I in Chapter 3), and a connecting line between diagrams (for Venn-II in Chapter 4).

Shin. The logical status of diagrams. 1995. Section 2.3. https://doi.org/10.1017/CBO9780511574696

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

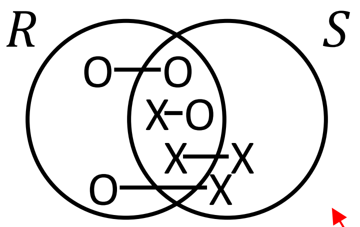
Venn-Peirce-Shin [Venn-II] (1995)

Sources used:

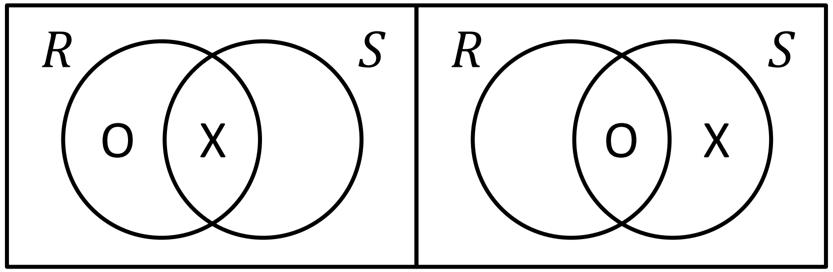
- Shin. The logical status of diagrams. 1995. https://doi.org/10.1017/CBO9780511574696

Venn-Peirce-Shin (Venn-II)

Venn-Peirce (disjunctions w/ lines between anchors)



Venn-Peirce (disjunctions via compartments)



Venn-Peirce-Shin ("Venn-II")

?

(**All** R are S, and **some** R is S) or (**No** R is S, and **some** S **is not** R)

Notice: disjunctions between <u>all anchors,</u> thus both X's and O's

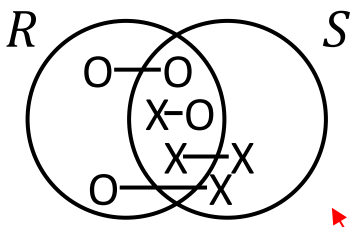
Notice: X and O anchors are not used for disjunctions

Shin finds that disjunctions <u>between universal statements</u> (in addition to existential statements) "increases the power of expression, but loses the visual effect(iveness) of a diagrammatic system." [Shin'95]

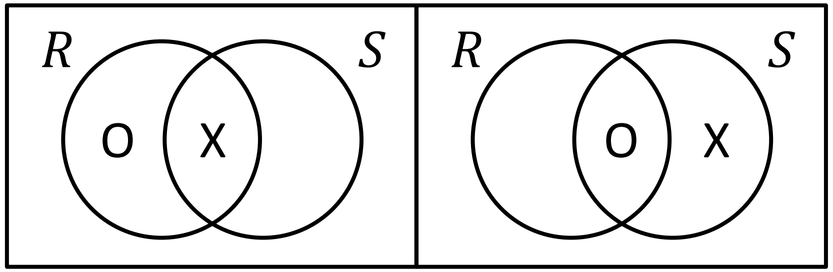
Shin. The logical status of diagrams. 1995. Section 2.3. https://doi.org/10.1017/CBO9780511574696

Venn-Peirce-Shin (Venn-II)

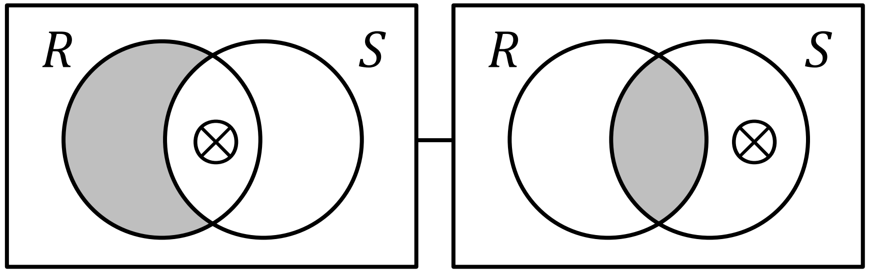
Venn-Peirce (disjunctions w/ lines between anchors)



Venn-Peirce (disjunctions via compartments)



Venn-Peirce-Shin ("Venn-II")



(**All** R are S, and **some** R is S) or (**No** R is S, and **some** S **is not** R)

Notice: disjunctions between all anchors, thus both X's and O's

Notice: X and O anchors are not used for disjunctions

- replaces "O" with shading
- replaces "X" with "⊗"
- allows "X-sequences" (disjunctions between existential statements "X")
- disjunctions between universal statements (now via shading, formerly via "O") are now handled via lines b/w compartments

Shin finds that disjunctions between universal statements (in addition to existential statements) "increases the power of expression, but loses the visual effect(iveness) of a diagrammatic system." [Shin'95]

Shin proved this diagrammatic system to be equivalent in expressiveness as monadic FOL without equality. "Without equality" means it cannot express: "There is exactly one R".

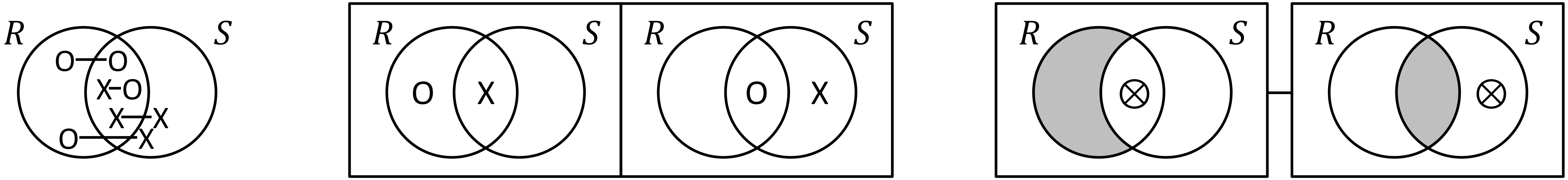
Shin. The logical status of diagrams. 1995. Section 2.3. https://doi.org/10.1017/CBO9780511574696
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Venn-Peirce-Shin (Venn-II)

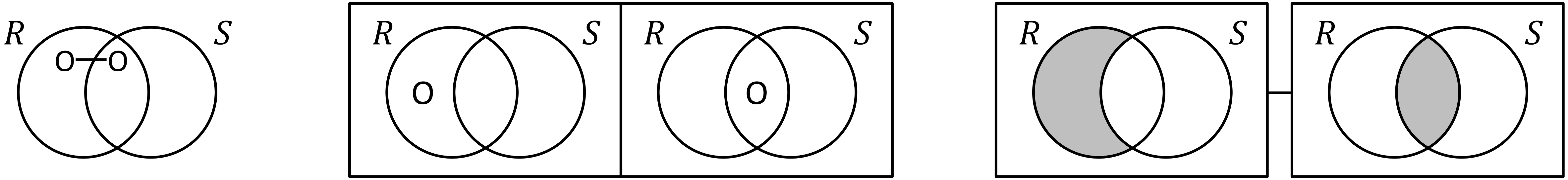
Venn-Peirce (disjunctions w/ lines between anchors)

Venn-Peirce (disjunctions via compartments)

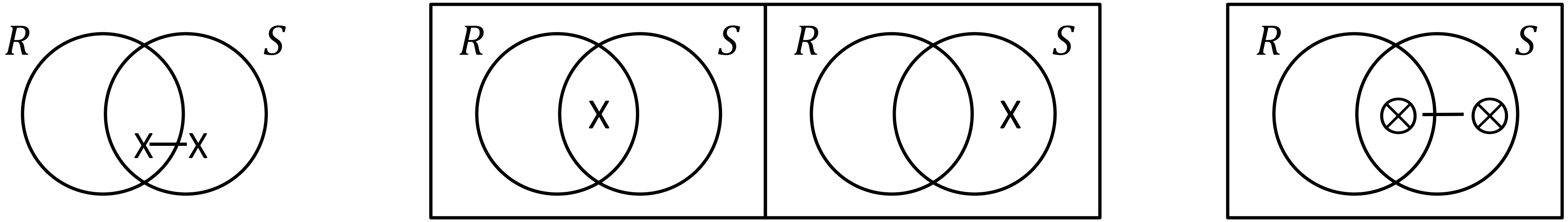
Venn-Peirce-Shin ("Venn-II")



(**All** R are S, and **some** R is S) or (**No** R is S, and **some** S **is not** R)



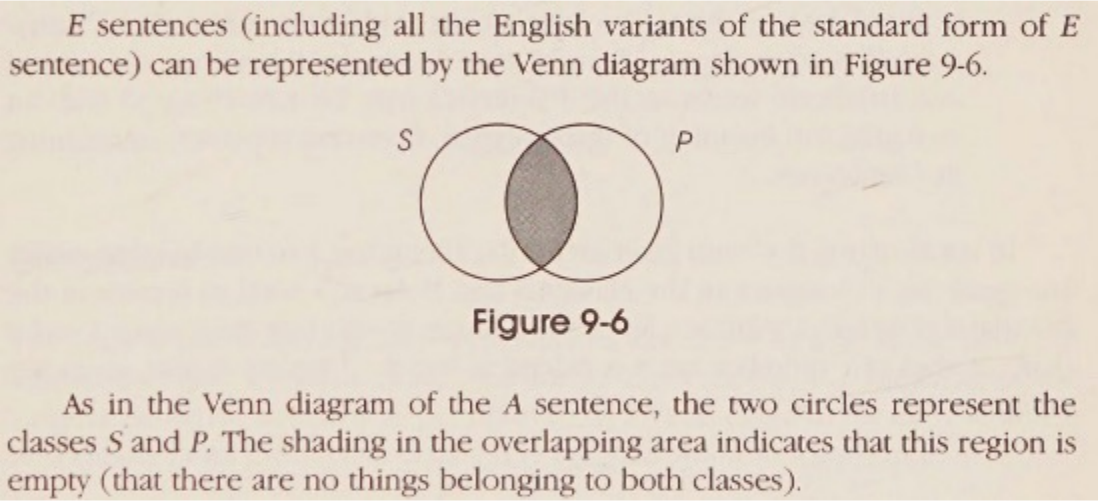
(**All** R are S) or (**no** R is S)



Goal: both full logical expressiveness and visual effectiveness

(**Some** R is S) or (**some** S **is not** R)

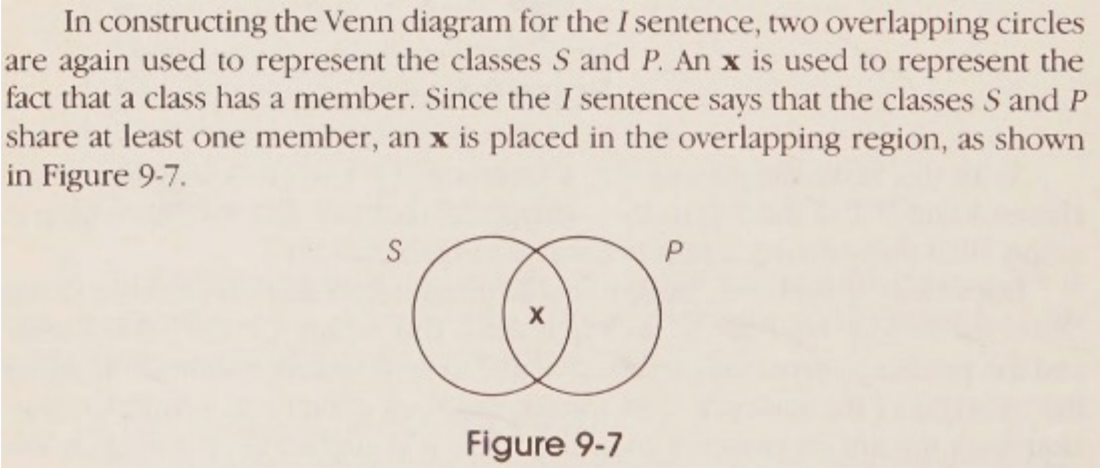
Venn-Peirce-Shin [Salmon 1984]



E sentences (including all the English variants of the standard form of *E* sentence) can be represented by the Venn diagram shown in Figure 9-6.

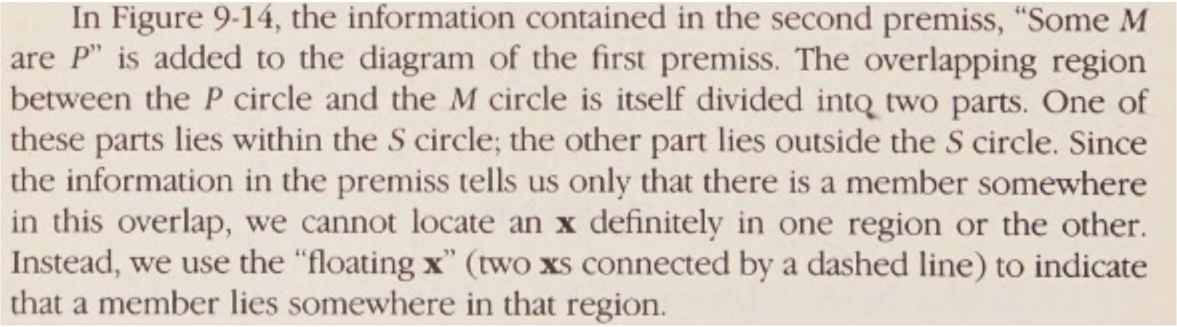
Figure 9-6

As in the Venn diagram of the *A* sentence, the two circles represent the classes *S* and *P*. The shading in the overlapping area indicates that this region is empty (that there are no things belonging to both classes).



In constructing the Venn diagram for the *I* sentence, two overlapping circles are again used to represent the classes *S* and *P*. An **x** is used to represent the fact that a class has a member. Since the *I* sentence says that the classes *S* and *P* share at least one member, an **x** is placed in the overlapping region, as shown in Figure 9-7.

Figure 9-7



In Figure 9-14, the information contained in the second premiss, "Some *M* are *P*" is added to the diagram of the first premiss. The overlapping region between the *P* circle and the *M* circle is itself divided into two parts. One of these parts lies within the *S* circle; the other part lies outside the *S* circle. Since the information in the premiss tells us only that there is a member somewhere in this overlap, we cannot locate an **x** definitely in one region or the other. Instead, we use the "floating **x**" (two **x**s connected by a dashed line) to indicate that a member lies somewhere in that region.

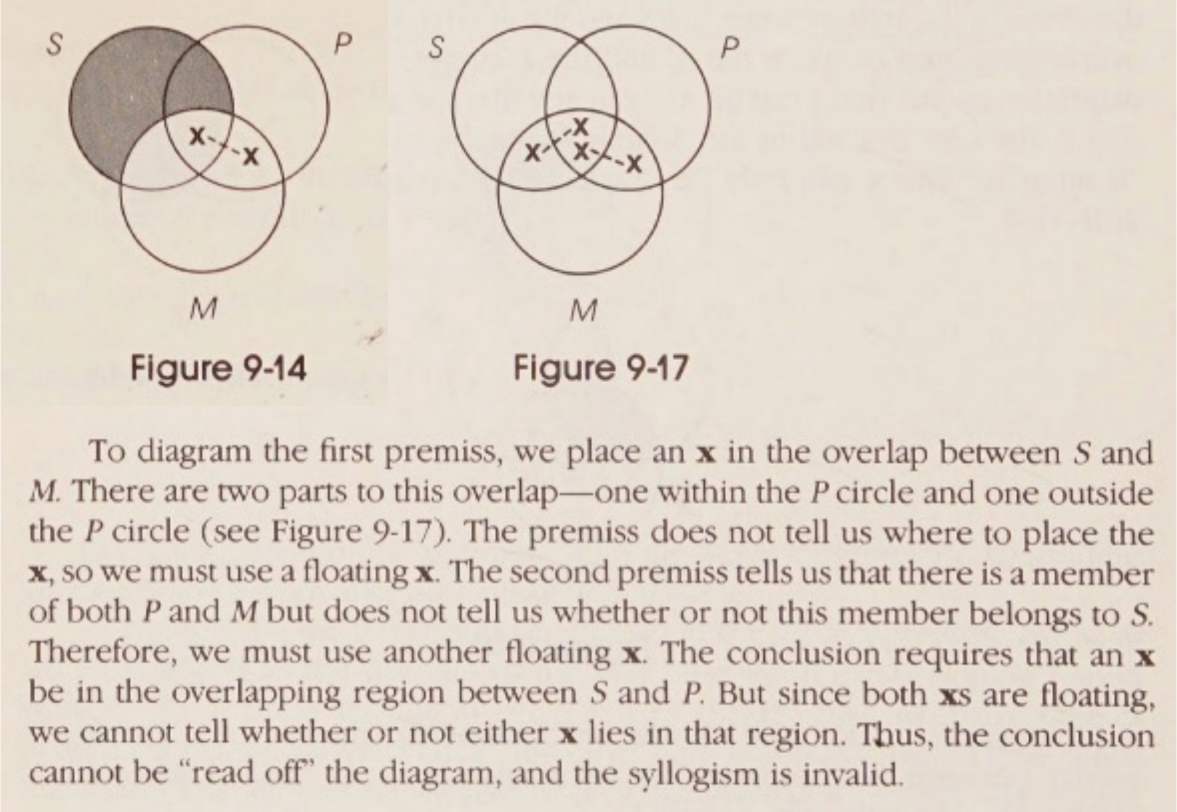


Figure 9-14

Figure 9-17

To diagram the first premiss, we place an **x** in the overlap between *S* and *M*. There are two parts to this overlap—one within the *P* circle and one outside the *P* circle (see Figure 9-17). The premiss does not tell us where to place the **x**, so we must use a floating **x**. The second premiss tells us that there is a member of both *P* and *M* but does not tell us whether or not this member belongs to *S*. Therefore, we must use another floating **x**. The conclusion requires that an **x** be in the overlapping region between *S* and *P*. But since both **x**s are floating, we cannot tell whether or not either **x** lies in that region. Thus, the conclusion cannot be "read off" the diagram, and the syllogism is invalid.

Salmon. Introduction to Logic and Critical Thinking, 1st ed, 1984. https://archive.org/details/introductiontolo0000salm_v3j4
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 4: Early Diagrammatic Representations

Monadic predicate calculus (Boolean queries & unary predicates)

1. Euler Circles (1768)
2. Venn Diagrams (1880)
3. Venn-Peirce Diagrams (~1896)
4. Venn-Peirce-Shin Diagrams (1995)

Polyadic predicate calculus (allowing predicates with arities ≥ 2)

7. Peirce Beta Existential Graphs (~1909)
8. Conceptual graphs (1976)
9. String diagrams (2024)

Beta Existential Graphs (Beta EGs) by Peirce (~1909)

Sources used:

- Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.621. https://archive.org/details/collectedpaperso0000unse_r5j9/
- Zeman. The Graphical Logic of Charles S. Peirce, Ph.D. dissertation, University of Chicago. 1964. https://isidore.co/CalibreLibrary/Zeman,%20Joseph%20Jay/The%20Graphical%20Logic%20of%20C.%20S.%20Peirce%20(4481)/
- Roberts. The existential graphs of Charles S. Peirce. De Gruyter Mouton. 1973. https://doi.org/10.1515/9783110226225
- Roberts. The existential graphs. Computers & Mathematics with Applications. 1992. https://doi.org/10.1016/0898-1221(92)90127-4
- Sun-Joo Shin. The Iconic Logic of Peirce’s Graphs. The MIT Press. 2002. https://doi.org/10.7551/mitpress/3633.001.0001

Beta Existential Graphs (EGs)

Schema

	Boat
1	bid
2	bname
3	color
4	pdate

S: "There is a boat."

∃x,y,z,u [Boat(x,y,z,u)]

2. Quantified variables from DRC are represented by "lines of identities" on the "sheet of assertion"

1. Supports sentence (no free variables), no queries

?

Beta Existential Graphs (EGs)

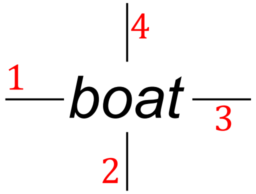
Schema

	Boat
1	bid
2	bname
3	color
4	pdate

S: "There is a boat."

∃x,y,z,u [Boat(x,y,z,u)]

1. Supports sentence (no free variables), no queries



2. Quantified variables from DRC are represented by "lines of identities" on the "sheet of assertion"

3. Requires an order on the attributes of each predicate (we use counter-clockwise)

S: "There is a red boat."

∃x,y,u [Boat(x,y,'red',u)] = ∃x,y,z,u [Boat(x,y,z,u) ∧ Red(z)]

?

4. EGs do not support constants, we thus use unary predicates. As alternative, we could have used: ∃x,y,u [Boat(x,y,u) ∧ Red(x)]

Beta Existential Graphs (EGs)

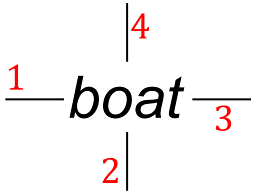
Schema

	Boat
1	<u>bid</u>
2	bname
3	color
4	pdate

S: "There is a boat."

∃x,y,z,u [Boat(x,y,z,u)]

1. Supports sentence (no free variables), no queries



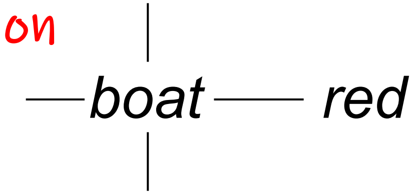
2. Quantified variables from DRC are represented by "lines of identities" on the "sheet of assertion"

3. Requires an order on the attributes of each predicate (we use counter-clockwise)

S: "There is a red boat."

∃x,y,u [Boat(x,y,'red',u)] = ∃x,y,z,u [Boat(x,y,z,u) ∧ Red(z)]

5. juxtaposition = conjunction



4. EGs do not support constants, we thus use unary predicates. As alternative, we could have used: ∃x,y,u [Boat(x,y,u) ∧ Red(x)]

S: "There is a boat that is not red."

∃x,y,z,u [Boat(x,y,z,u) ∧ ¬(∃w [Red(w) ∧ z=w])]

?

6. A statement is denied by enclosing it in closed curve, called a "cut"

Beta Existential Graphs (EGs)

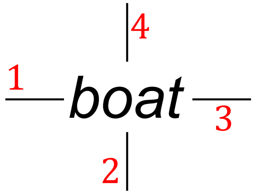
Schema

	Boat
1	bid
2	bname
3	color
4	pdate

S: "There is a boat."

∃x,y,z,u [Boat(x,y,z,u)]

1. Supports sentence (no free variables), no queries



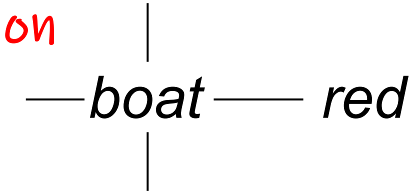
2. Quantified variables from DRC are represented by "lines of identities" on the "sheet of assertion"

3. Requires an order on the attributes of each predicate (we use counter-clockwise)

S: "There is a red boat."

∃x,y,u [Boat(x,y,'red',u)] = ∃x,y,z,u [Boat(x,y,z,u) ∧ Red(z)]

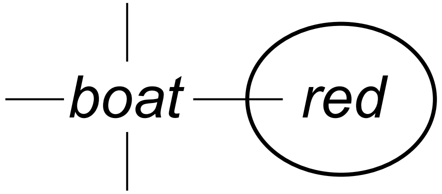
5. juxtaposition = conjunction



4. EGs do not support constants, we thus use unary predicates. As alternative, we could have used: ∃x,y,u [Boat(x,y,u) ∧ Red(x)]

S: "There is a boat that is not red."

∃x,y,z,u [Boat(x,y,z,u) ∧ ¬(∃w [Red(w) ∧ z=w])]



6. A statement is denied by enclosing it in closed curve, called a "cut"

Beta Existential Graphs (EGs)

Schema

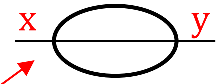
	Boat
1	<u>bid</u>
2	bname
3	color
4	pdate

S: "There is exactly one boat."

$\exists x,y,z,w\ [\text{Boat}(x,y,z,w) \wedge \neg(\exists s,t,u,v\ [\text{Boat}(s,t,u,v) \wedge x \neq s])]$

$\exists x,y,z,w\ [\text{Boat}(x,y,z,w) \wedge \forall s,t,u,v\ [\text{Boat}(s,t,u,v) \rightarrow x = s]]$

?



7. EGs can express equality. "There exist two things (x and y) and they are different" (the line between them is negated)

Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

Beta Existential Graphs (EGs)

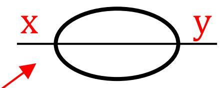
Schema

	Boat
1	bid
2	bname
3	color
4	pdate

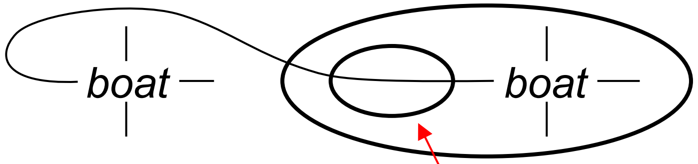
S: "There is exactly one boat."

$\exists x,y,z,w\ [\mathrm{Boat}(x,y,z,w) \land \neg(\exists s,t,u,v\ [\mathrm{Boat}(s,t,u,v) \land\ x \neq s])]$

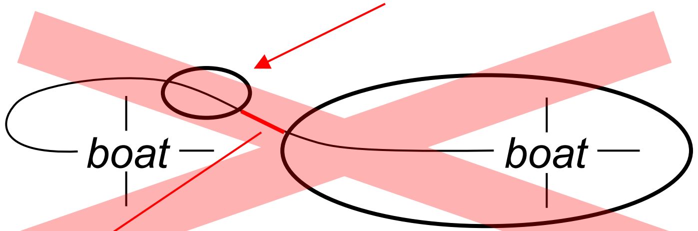
$\exists x,y,z,w\ [\mathrm{Boat}(x,y,z,w) \land\quad \forall s,t,u,v\ [\mathrm{Boat}(s,t,u,v) \rightarrow x = s]]$



7. EGs can express equality. "There exist two things (x and y) and they are different" (the line between them is negated)



8. The placement of the cuts matters! The EG below could erroneously be seen as equivalent way to write above statement (I got this wrong myself earlier). But a more careful analysis shows that it has a different semantic interpretation.



This line segment and variable s are not guarded

$\exists x,y,z,w,s\ [\mathrm{Boat}(x,y,z,w) \land x \neq s \land \neg(\exists t,u,v\ [\mathrm{Boat}(s,t,u,v)])]$

Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

Beta Existential Graphs (EGs)

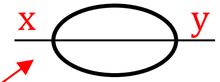
Schema

Boat	
1	bid
2	bname
3	color
4	pdate

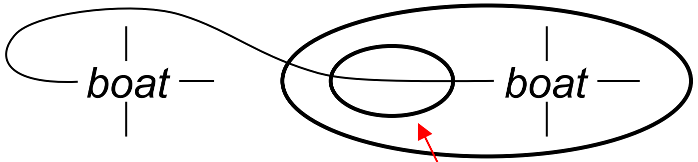
S: "There is exactly one boat."

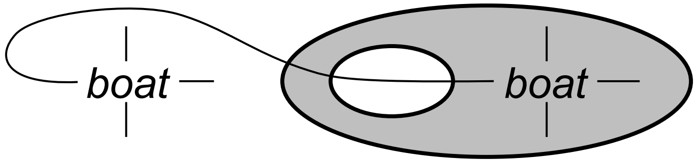
$\exists x,y,z,w\ [\mathrm{Boat}(x,y,z,w) \land \neg(\exists s,t,u,v\ [\mathrm{Boat}(s,t,u,v) \land\ x \neq s])]$

$\exists x,y,z,w\ [\mathrm{Boat}(x,y,z,w) \land \quad \forall s,t,u,v\ [\mathrm{Boat}(s,t,u,v) \rightarrow x = s]]$



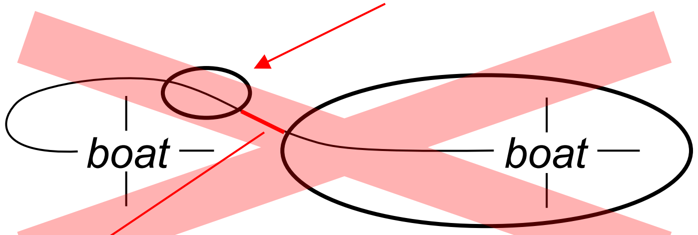
7. EGs can express equality. "There exist two things (x and y) and they are different" (the line between them is negated)

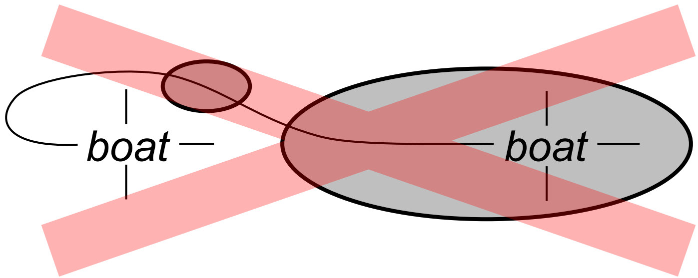




8. The placement of the cuts matters! The EG below could erroneously be seen as equivalent way to write above statement (I got this wrong myself earlier). But a more careful analysis shows that it has a different semantic interpretation.

9. "Peirce shading": shade the oddly enclosed areas. Makes it often easier to read these diagrams.





This line segment and variable s are not guarded

$\exists x,y,z,w,s\ [\mathrm{Boat}(x,y,z,w) \land x \neq s \land \neg(\exists t,u,v\ [\mathrm{Boat}(s,t,u,v)])]$

Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

"Peirce shading": Alternating shaded areas

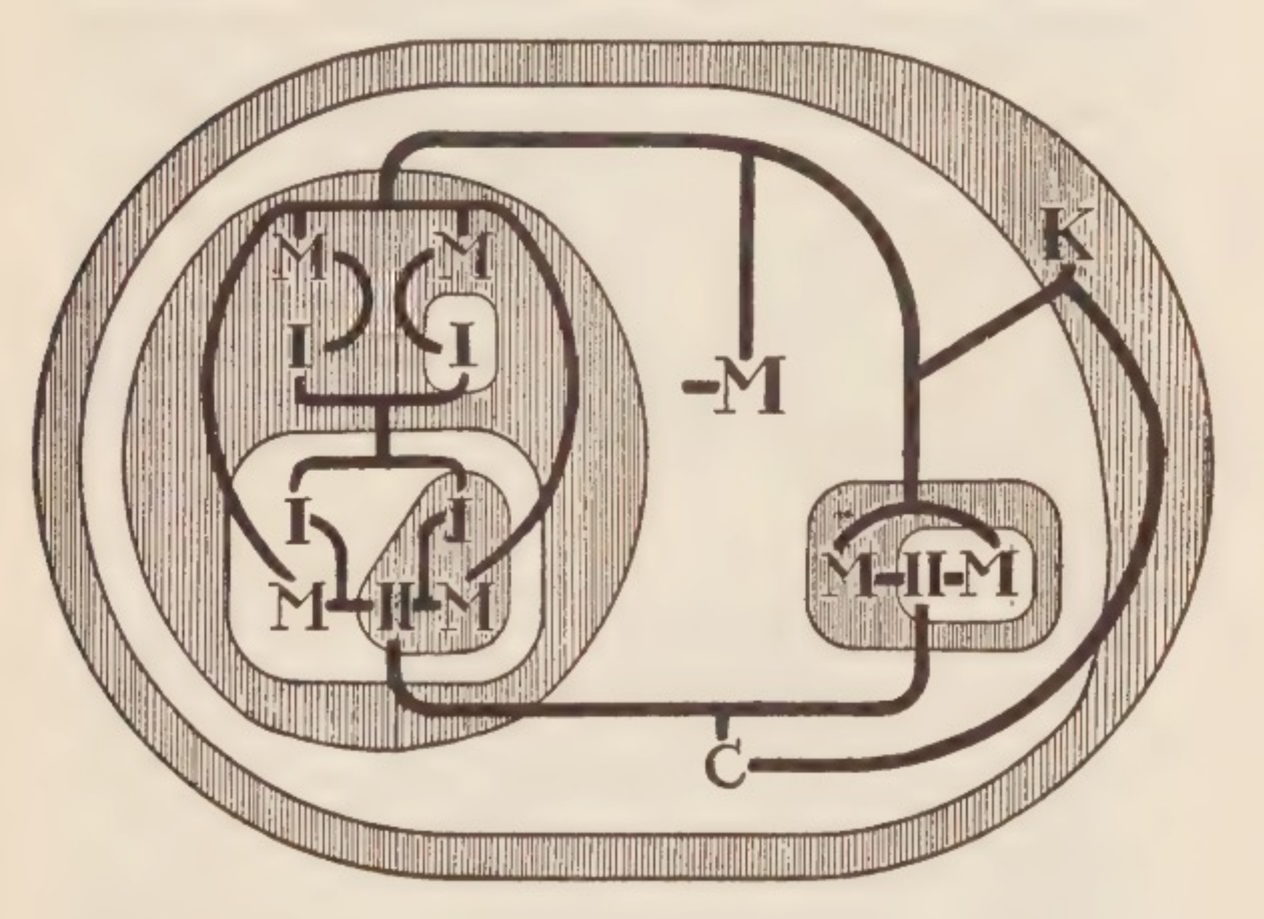
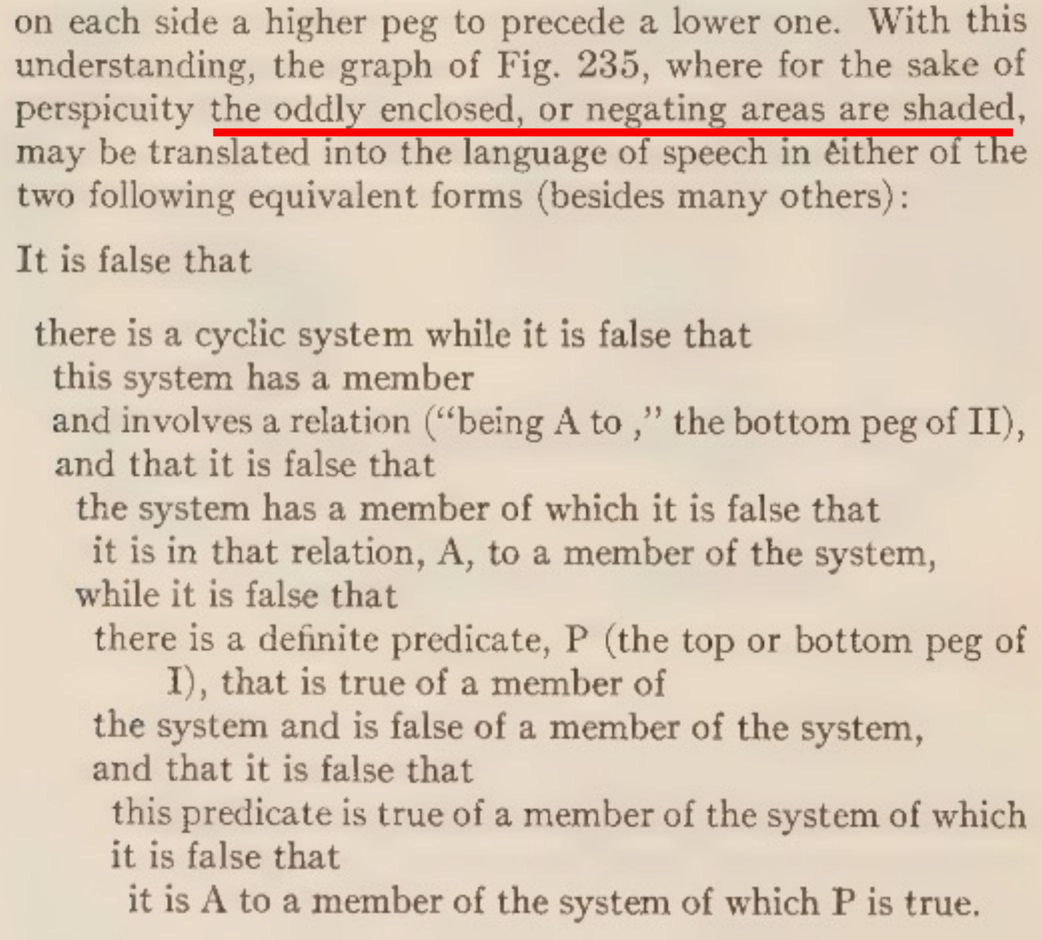


Fig. 235



on each side a higher peg to precede a lower one. With this understanding, the graph of Fig. 235, where for the sake of perspicuity the oddly enclosed, or negating areas are shaded, may be translated into the language of speech in either of the two following equivalent forms (besides many others):

It is false that

there is a cyclic system while it is false that
  this system has a member
  and involves a relation ("being A to ," the bottom peg of II),
  and that it is false that
    the system has a member of which it is false that
      it is in that relation, A, to a member of the system,
    while it is false that
      there is a definite predicate, P (the top or bottom peg of I), that is true of a member of
      the system and is false of a member of the system,
      and that it is false that
        this predicate is true of a member of the system of which
        it is false that
          it is A to a member of the system of which P is true.

Reading from the outside inward also called "endoporeutic" reading by Peirce

Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.621. https://archive.org/details/collectedpaperso0000unse_r5j9/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Beta Existential Graphs (EGs)

Schema

	Boat
1	bid
2	bname
3	color
4	pdate

S: "There are exactly two boats."

$$\exists x,y,z,s,t,p,r\ [\text{Boat}(x,y,\text{'red'},z) \land \text{Boat}(s,t,p,r) \land\ x \neq s \land$$
$$\neg(\exists u,v,w,\ [\text{Boat}(u,v,\text{'red'},w) \land u \neq x \land u \neq s])]$$

$$\exists x,y,z,s,t,p,r\ [\text{Boat}(x,y,\text{'red'},z) \land \text{Boat}(s,t,p,r) \land\ x \neq s \land$$
$$\forall u,v,w,\ [\text{Boat}(u,v,\text{'red'},w) \longrightarrow (u=x \lor u=s])]$$

?

Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

Beta Existential Graphs (EGs)

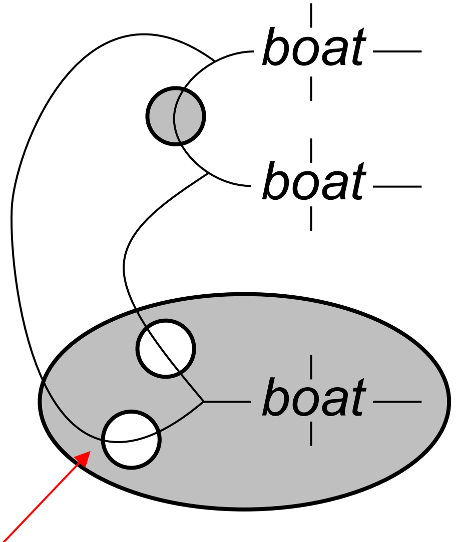
Schema

Boat	
1	bid
2	bname
3	color
4	pdate

S: "There are exactly two boats."

∃x,y,z,s,t,p,r [Boat(x,y,'red',z) ∧ Boat(s,t,p,r) ∧ x≠s ∧
¬(∃u,v,w, [Boat(u,v,'red',w) ∧ u≠x ∧ u≠s])]

∃x,y,z,s,t,p,r [Boat(x,y,'red',z) ∧ Boat(s,t,p,r) ∧ x≠s ∧
∀u,v,w, [Boat(u,v,'red',w) → (u=x ∨ u=s])]

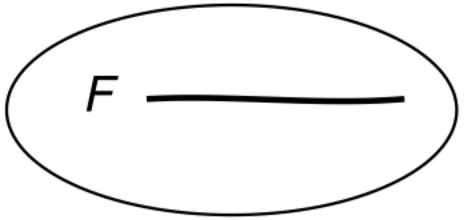


The placement of the two cuts *inside* the larger one is important; placing them outside instead would give an incorrect semantics.

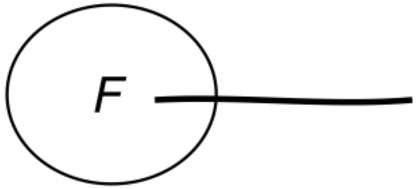
Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

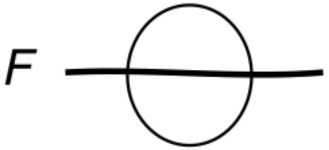
Beta EGs: Lines of Identity & "Loose ends"



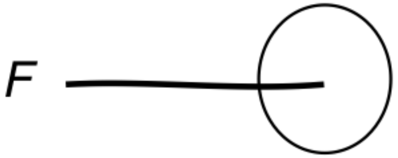
"It is not the case that something is F"



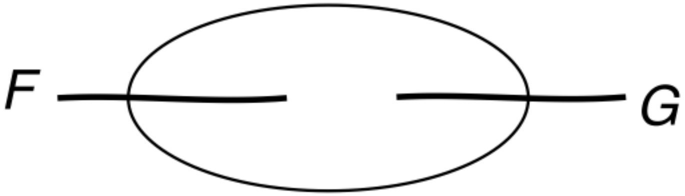
"Something is not F"



"Something is F, and there is at least one more thing"



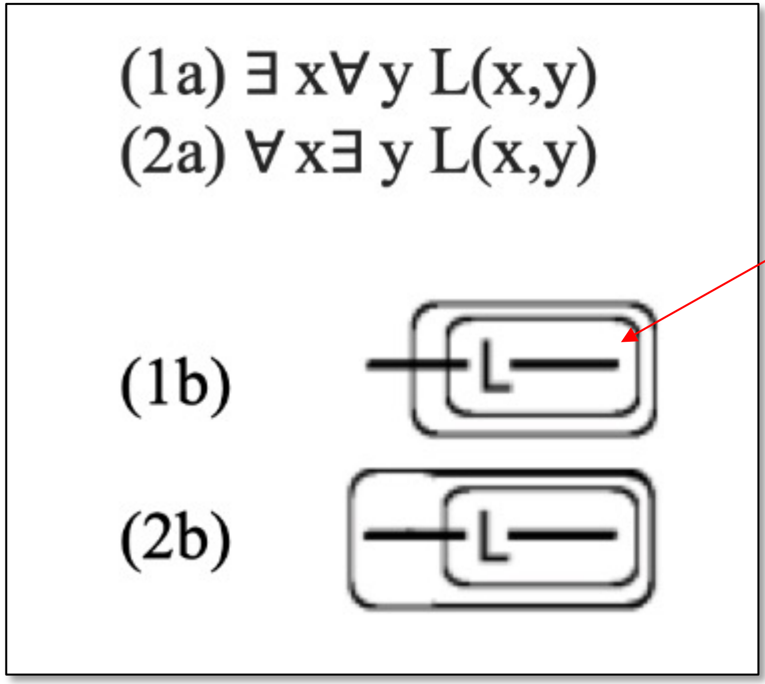
"Some F is not identical with itself."



"Some F or some G is not identical with itself."

Sun-Joo Shin. The Iconic Logic of Peirce's Graphs. The MIT Press. 2002. chapter 3, p.42. https://doi.org/10.7551/mitpress/3633.001.0001
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Beta EGs: Lines of Identity & "Loose ends"

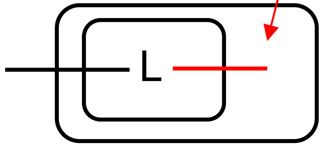


(1a) ∃ x∀ y L(x,y)

(2a) ∀ x∃ y L(x,y)

The end point of these lines matters a lot. One appears in the wrong enclosure.

It should rather end in nesting depth 1, instead of 2



Pietarinen, Bellucci. Two Dogmas of Diagrammatic Reasoning: A View from Existential Graphs, book chapter 2017. https://doi.org/10.4324/9781315444642
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

The "lines of identity" are complicated to interpret

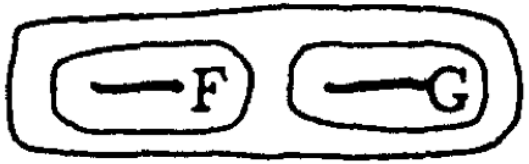


Figure 65.

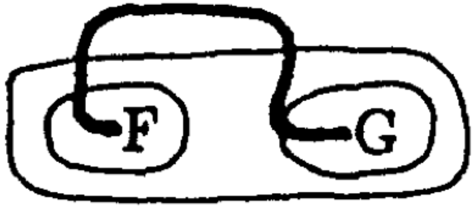
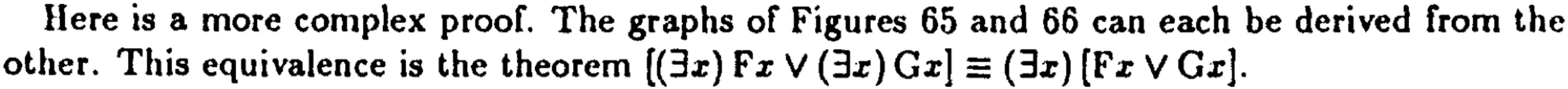


Figure 66.



Here is a more complex proof. The graphs of Figures 65 and 66 can each be derived from the other. This equivalence is the theorem $[(\exists x)\,Fx \vee (\exists x)\,Gx] \equiv (\exists x)\,[Fx \vee Gx]$.

There is extensive secondary literature that elaborates on how routing the lines affects the semantics. These interpretations rely on visual inference rules (= diagrammatic reasoning) and are pretty hairy.

Roberts. The existential graphs. Computers & Mathematics with Applications. 1992. https://doi.org/10.1016/0898-1221(92)90127-4
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

The "lines of identity" are complicated to interpret

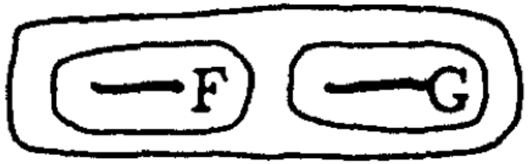


Figure 65.

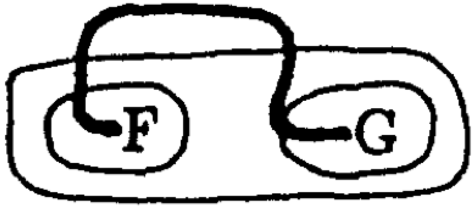
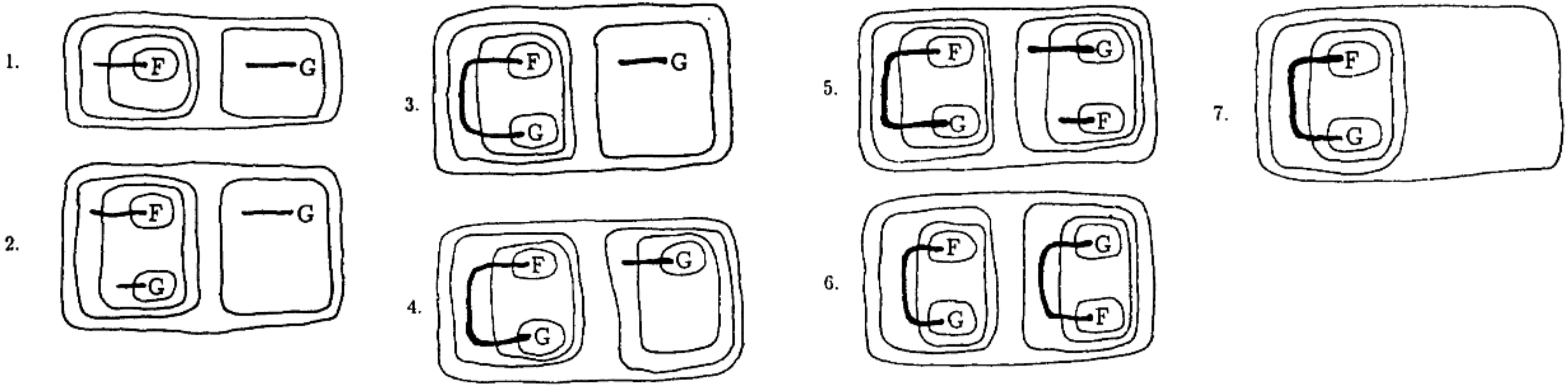


Figure 66.

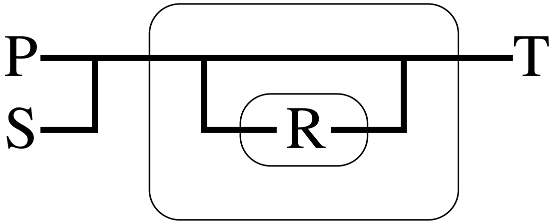
The inference from Figure 66 to Figure 65 is straightforward, but the converse inference requires subtlety.



Roberts. The existential graphs. Computers & Mathematics with Applications. 1992. https://doi.org/10.1016/0898-1221(92)90127-4

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

How to interpret more complicated graphs?



?

Dau. Fixing Shin's Reading Algorithm for Peirce's Existential Graphs. Diagrams 2006. https://doi.org/10.1007/11783183_10

Shin. The Iconic Logic of Peirce's Graphs. 2002. Chapter 3. https://doi.org/10.7551/mitpress/3633.001.0001

How to interpret more complicated graphs?

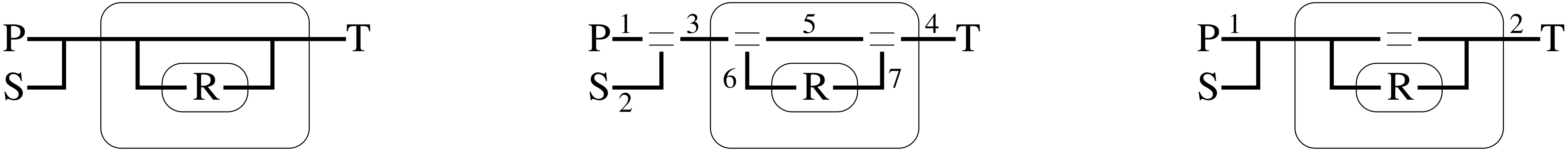


Figure from Dau [2006] discussing an example beta-EG whose interpretation provided by Shin [2002] is incorrect, together with two alternative interpretations of splitting the Lines of Identity (LI's) in order to interpret this beta-EGs correctly. The details of the arguments are not important here, only that there is a lot of disagreement as to how interpret LI's correctly.

Take-away: <u>Beta EGs are complicated to read and interpret correctly, mainly because of the use of lines for existential quantification.</u>

Dau. Fixing Shin's Reading Algorithm for Peirce's Existential Graphs. Diagrams 2006. https://doi.org/10.1007/11783183_10

Shin. The Iconic Logic of Peirce's Graphs. 2002. Chapter 3. https://doi.org/10.7551/mitpress/3633.001.0001

Beta Existential Graphs

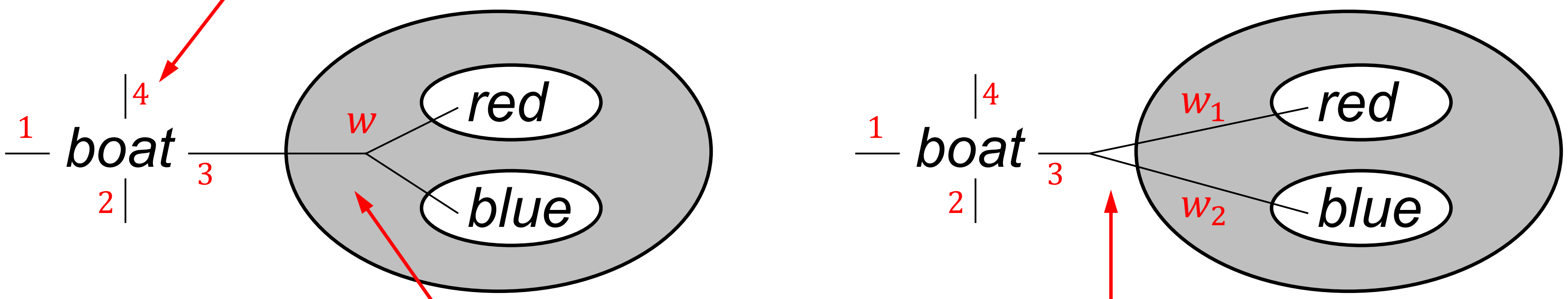
Schema

	Boat
1	bid
2	bname
3	color
4	pdate

~Q1: "There is a boat that is red or blue."

```
select exists
   (select *
   from Boat
   where color = 'red'
   or color = 'blue')
```

The literature is not explicit about how to determine the order of the attributes. My convention here is a counter-clockwise order.



Comparison predicates such as "pdate < 1980" are not supported

The exact placement of this fork (inside the shaded area) makes a slight difference in the interpretation of the line segments. Yet both turn out to be logically equivalent.

left: $\{(x) \mid \exists y,z,u\ [\text{Boat}(z,x,y,u) \land \neg(\exists w\ [w=y \land \neg(w=\text{'red'}) \land \neg(w=\text{'blue'})])]\}$

right: $\{(x) \mid \exists y,z,u\ [\text{Boat}(z,x,y,u) \land \neg(\exists w_1,w_2\ [w_1=y \land w_2=y \land \neg(w_1=\text{'red'}) \land \neg(w_2=\text{'blue'})])]\}$

Disjunction of selection predicates can be represented via De Morgan (double-negation and conjunction)

DRC (Domain Relational Calculus)

$\exists x,y,z,u\ [\text{Boat}(x,y,z,u) \land (z=\text{'red'} \lor z=\text{'blue'})$

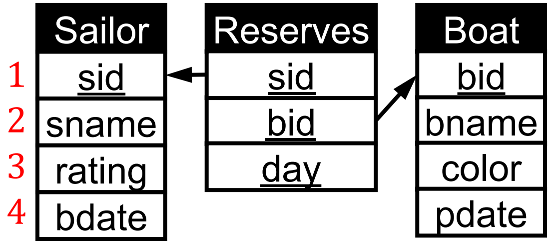
$\exists x,y,z,u\ [\text{Boat}(x,y,z,u) \land \neg(\neg(z=\text{'red'}) \land \neg(z=\text{'blue'}))]$

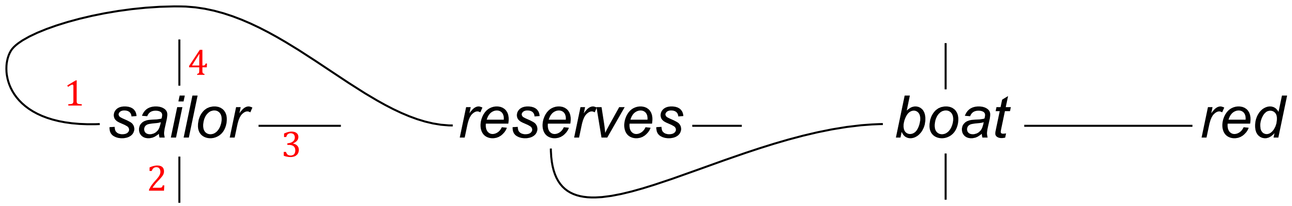
Beta Existential Graphs

~Q2: "There is a sailor who reserved a red boat."

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

Schema



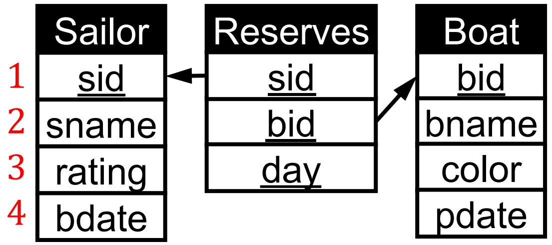


DRC (Domain Relational Calculus)

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

Beta Existential Graphs

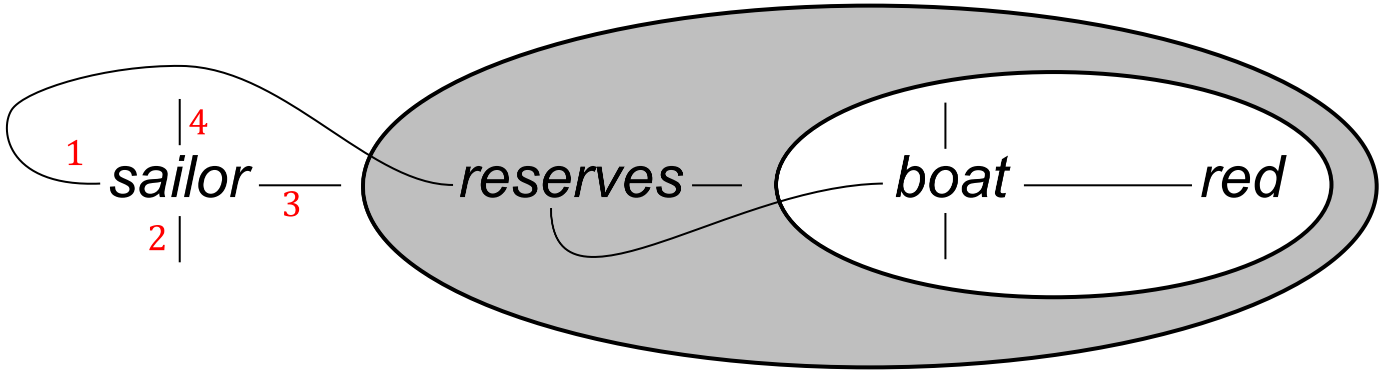
Schema



Sailor	Reserves	Boat
1 sid	sid	bid
2 sname	bid	bname
3 rating	day	color
4 bdate		pdate

~Q3: "There is a sailor who reserved only red boats."

```
select exists
   (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')))
```



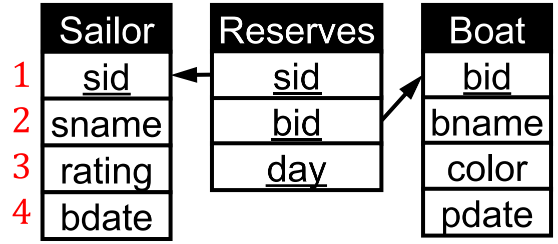
DRC (Domain Relational Calculus)

$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \land \neg(\exists y,t\ [\text{Reserves}(v,y,t) \land \neg(\exists u,s\ [\text{Boat}(y,u,\text{'red'},s)])])]$

$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \land \quad \forall y,t\ [\text{Reserves}(v,y,t) \rightarrow (\exists u,s\ [\text{Boat}(y,u,\text{'red'},s)])])]$

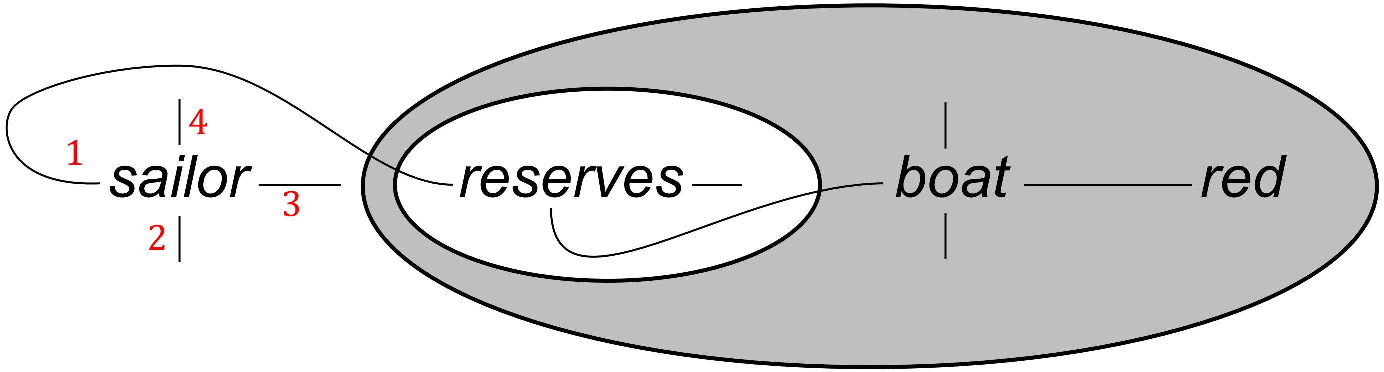
Beta Existential Graphs

Schema



~Q4: "There is a sailor who reserved all red boats."

```
select exists
   ( 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))
```



DRC (Domain Relational Calculus)

$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \neg(\exists y,u,s\ [\text{Boat}(y,u,\text{'red'},s) \wedge \neg(\exists t\ [\text{Reserves}(v,y,t)])])]$

$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \quad \forall y,u,s\ [\text{Boat}(y,u,\text{'red'},s) \rightarrow (\ \exists t\ [\text{Reserves}(v,y,t)])])]$

Beta Existential Graphs

Schema

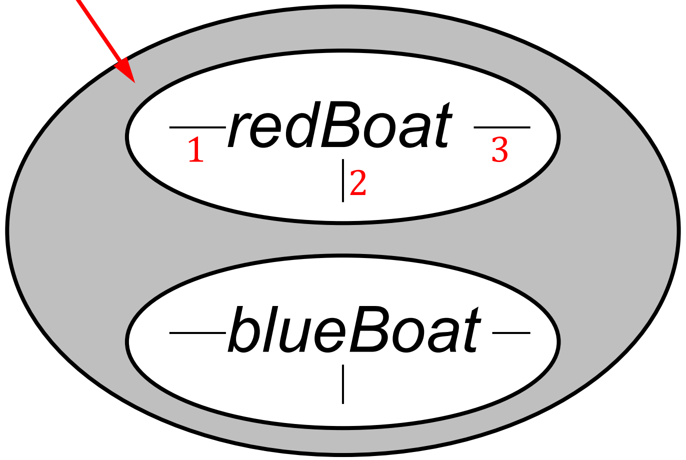
	RedBoat
1	bid
2	bname
3	pdate

BlueBoat
bid
bname
pdate

~Q5: "There is a boat that is red or blue."

```
select exists
   (select *
   from RedBoat R
   union
   select *
   from BlueBoat B)
```

Union of table can be represented via De Morgan (double-negation and conjunction). However, notice that the resulting expression is not safe (free variables x,y, z are not "guarded", i.e. not bound to an element from the active domain)



DRC (Domain Relational Calculus)

$$\exists x \exists y \exists z\ [\text{RedBoat}(x,y,z)\ \vee\ \text{BlueBoat}(x,y,z)]$$

$$\neg(\neg(\exists x \exists y \exists z\ [\text{RedBoat}(x,y,z)])\ \wedge\ \neg(\exists z\ [\text{BlueBoat}(x,y,z)]))$$

An accessible overview of issues involving safety is: Topor, Safety and Domain Independence, Encyclopedia of Database Systems. 2009 https://doi.org/10.1007/978-0-387-39940-9_1255

Conceptual graphs
(1976)

Sources used:

- Sowa. “Conceptual graphs for a data base interface,” IBM Journal of Research and Development, 1976. https://doi.org/10.1147/rd.204.0336
- Sowa. “Conceptual Structures: Information Processing in Mind and Machine,” Addison-Wesley, 1984. https://dl.acm.org/doi/book/10.5555/4569
- Sowa. “Knowledge Representation: Logical, Philosophical, and Computational Foundations,” Brooks/Cole, 2000. https://dl.acm.org/doi/book/10.5555/318183
- Sowa. “Conceptual graphs,” in Handbook of Knowledge Representation,” 2008. https://doi.org/10.1016/S1574-6526(07)03005-2
- Sowa. “From existential graphs to conceptual graphs,” IJCSSA, 2013. https://doi.org/10.4018/ijcssa.2013010103
- Sowa. “Reasoning with diagrams and images,” Journal of Applied Logics. 2018. https://www.collegepublications.co.uk/ifcolog/?00025
- Dau. “The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic,” 2003. https://doi.org/10.1007/b94030
- Chein, Mugnier. “Graph-based knowledge representation,” 2009. https://doi.org/10.1007/978-1-84800-286-9

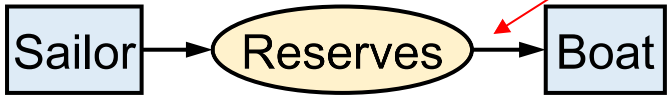
Conceptual Graphs and their translation to calculus

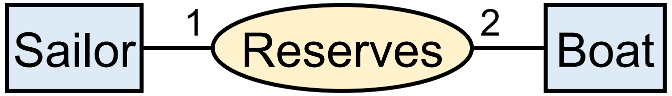
Conceptual graphs have 2 types of nodes (similar to ER diagrams):

- concept nodes (rectangular): represent quantification (instead of line), entities (and attributes)
- relation nodes (oval): show relationships (of arbitrary arity) between entities

entities of type "Sailor"

arcs link concepts and relations (arcs "belong" to concepts, arrowhead marks first argument for binary relation, sequence numbers are added for higher arity relations)





Can think of binary relations in a relational database with typed columns

Translation to predicate calculus:

$$\exists x, \exists y[\text{Sailor}(x) \land \text{Boat}(y) \land \text{Reserves}(x, y)]$$

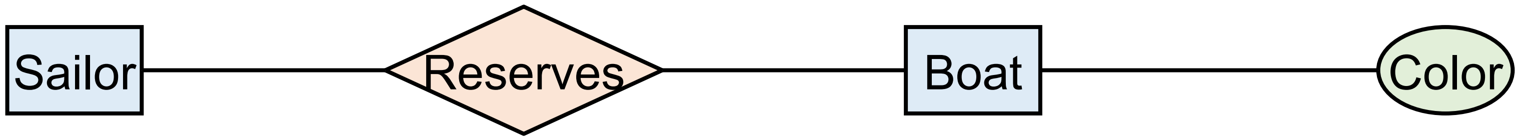
Concepts are existentially quantified variables

$$\exists x{:}\,\text{Sailor}, \exists y{:}\,\text{Boat}[\text{Reserves}(x, y)]$$

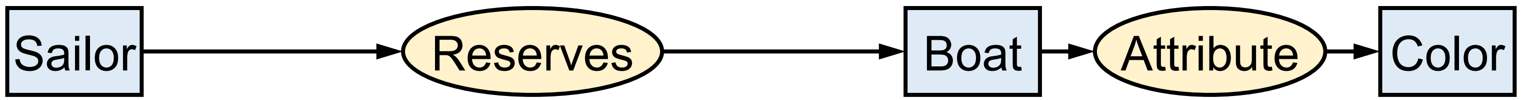
Concepts are typed variables in "Typed predicate calculus"

Colors are not part of formal definitions, but used in presentations by John Sowa ("Introduction to Common Logic," 2011. www.jfsowa.com/talks/clintro.pdf) and make them easier to read
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Conceptual Graphs vs. ERD attributes and relations

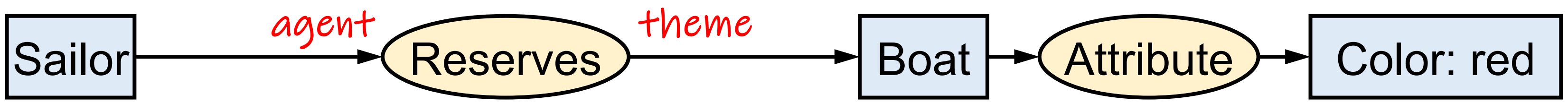


ER diagrams show "entity sets" and may include attributes



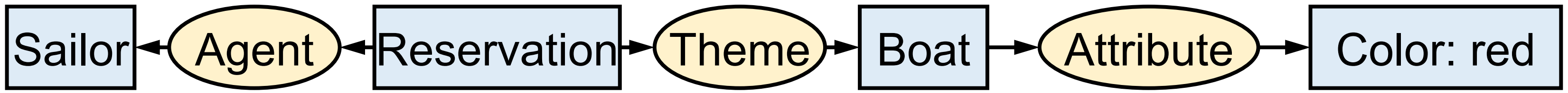
$\exists x\colon \text{Sailor}, \exists y\colon \text{Boat}, \exists z\colon \text{Color}[\text{Reserves}(x, y) \wedge \text{Attribute}(y, z)]$

CGs show statements about "entities" (no attributes)



$\exists x\colon \text{Sailor}, \exists y\colon \text{Boat}[\text{Reserves}(x, y) \wedge \text{Color}(\text{red}) \wedge \text{Attribute}(y, \text{red})]$

"Individual" concept (instead of "generic")



Former relation now represented as as concept, linked to its participants by "case relations"

$\exists x\colon \text{Sailor}, \exists y\colon \text{Boat}, \exists z\colon \text{Reservation}[\text{Agent}(z, x) \wedge \text{Theme}(z, y) \wedge \text{Color}(\text{red}) \wedge \text{Attribute}(x, \text{red})]$

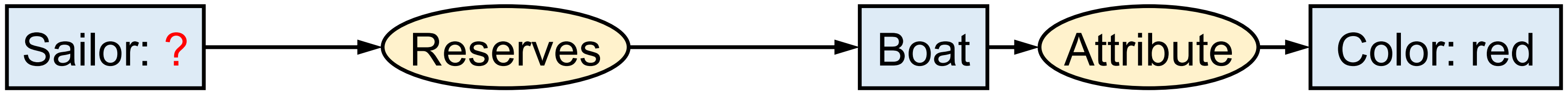
After: Figure 1.9, Section 1.4 from Sowa. "Knowledge Representation: Logical, Philosophical, and Computational Foundations," 2000. https://dl.acm.org/doi/book/10.5555/318183
and Sect 5.1 from Sowa. "Conceptual graphs," in Handbook of Knowledge Representation, 2008. https://doi.org/10.1016/S1574-6526(07)03005-2

Conceptual Graphs vs. ERD attributes and relations

A <u>query graph</u> is a conceptual graph that contains one or more concepts with a question mark in the referent field, as in [Sailor: ?]

Sailor → Reserves → Boat → Attribute → Color: red

$\exists x\text{: Sailor}, \exists y\text{: Boat}[\text{Reserves}(x, y) \land \text{Color}(\text{red}) \land \text{Attribute}(y, \text{red})]$

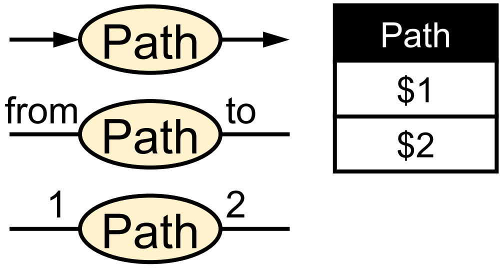


Sailor: ? → Reserves → Boat → Attribute → Color: red

$\{x \mid x\text{: Sailor}, \exists y\text{: Boat}[\text{Reserves}(x, y) \land \text{Color}(\text{red}) \land \text{Attribute}(y, \text{red})]\}$

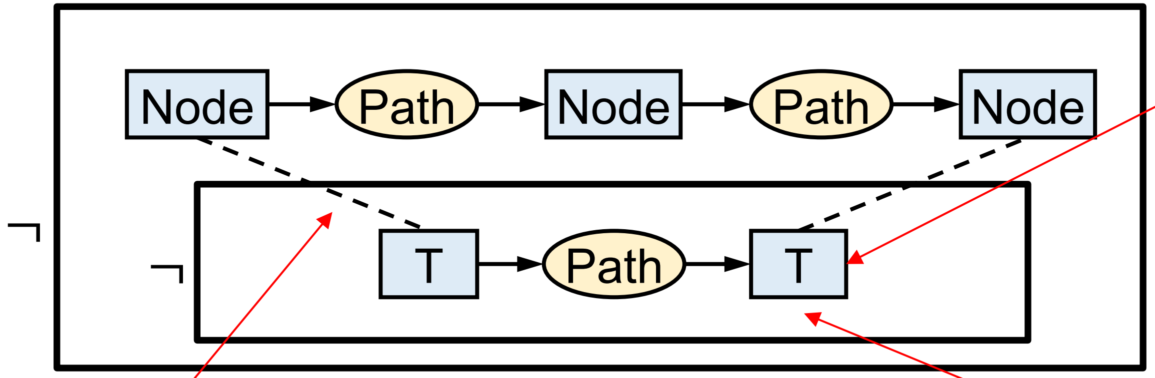
After: Figure 1.9, Section 1.4 from Sowa. “Knowledge Representation: Logical, Philosophical, and Computational Foundations,” 2000. https://dl.acm.org/doi/book/10.5555/318183
and Sect 5.1 from Sowa. “Conceptual graphs,” in Handbook of Knowledge Representation, 2008. https://doi.org/10.1016/S1574-6526(07)03005-2

Conceptual Graphs: negation



"Paths are transitive."

Concept boxes with negation symbol "¬"



T (top) = "universal type" (details a bit complicated with minor errors even in [Sowa'84], the universal type is the greatest type in a type hierarchy).

1. "Coreference" link (dotted line): both endpoints refer to same entity

2. Relations cannot be shown without concepts (concepts provide the required quantification). CGs within each "context" need to be a valid proposition (also within negated context). Thus the required quantification by **T** in the nesting

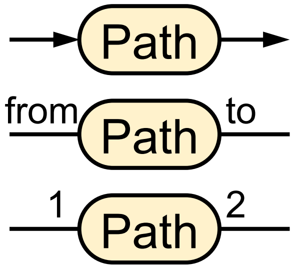
Translation to (domain relational) calculus:

$$\neg(\exists x\colon \text{Node}, \exists y\colon \text{Node}, \exists z\colon \text{Node}[\text{Path}(x,y) \wedge \text{Path}(y,z) \wedge \neg(\text{Path}(x,z))])$$

Meaning: "The relation 'Path' is transitive on entities of type 'Node'."

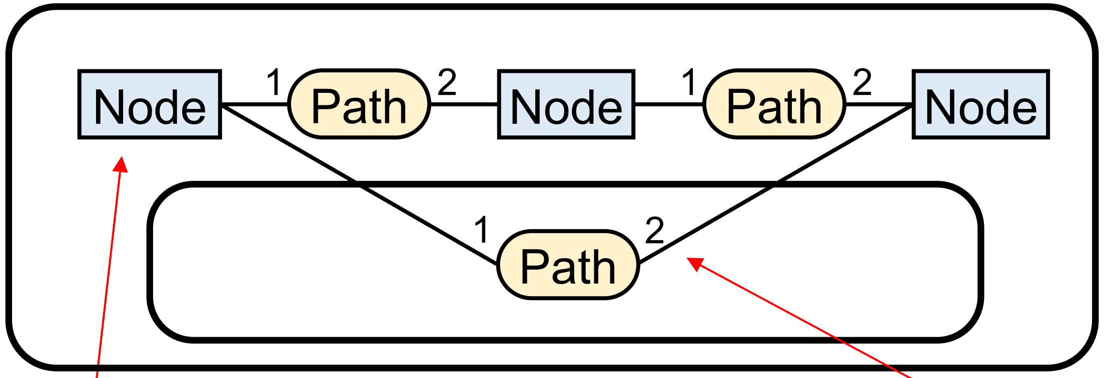
Based on "Chein, Mugnier. “Graph-based knowledge representation,” 2009. https://doi.org/10.1007/978-1-84800-286-9"

Conceptual Graphs (Dau variant)



Path
$1
$2

"Paths are transitive."



"The definition of Sowa does not suffice to derive a mathematical definition from it." [Dau'03]
Dau streamlines various details for the purpose of FOL, leading to a more precise and simpler notation, called "concept graphs w/ cuts". We instead call them "conceptual graphs (Dau variant)"

Dau's variant still requires concept nodes *in addition* to relation nodes

1. Does not require any coreference links anymore.
2. Neither do graphs inside "cuts" (instead of "negative contexts") need to valid and thus locally quantified concepts for all relations

Translation to (domain relational) calculus:

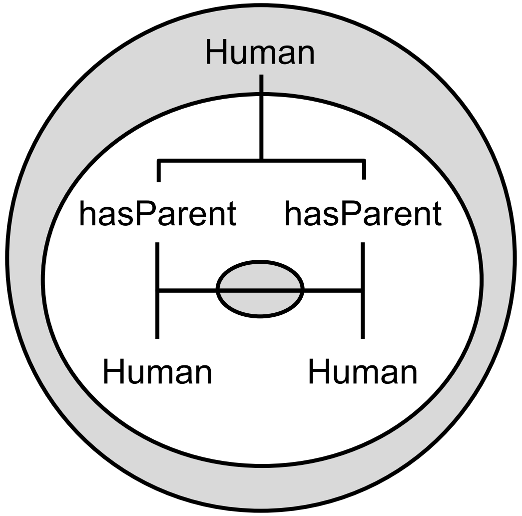
$$\neg(\exists x{:}\,\text{Node}, \exists y{:}\,\text{Node}, \exists z{:}\,\text{Node}[\text{Path}(x,y) \land \text{Path}(y,z) \land \neg(\text{Path}(x,z))])$$

Meaning: "The relation 'Path' is transitive on entities of type 'Node'."

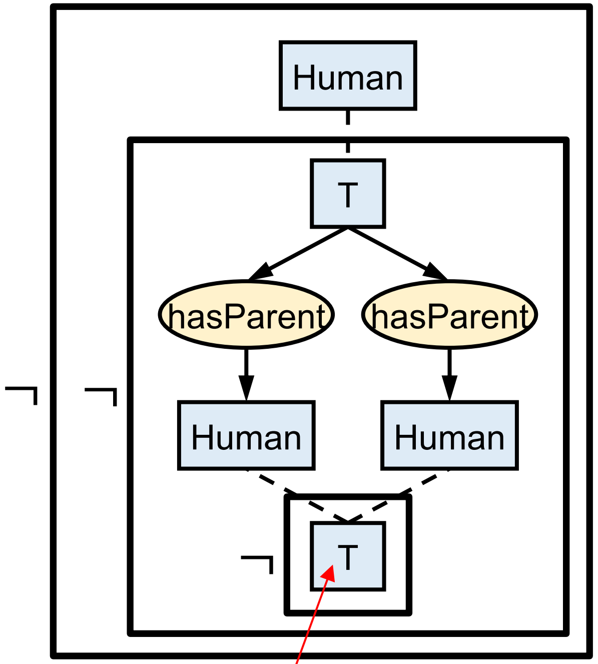
Based on "Chein, Mugnier. "Graph-based knowledge representation," 2009. https://doi.org/10.1007/978-1-84800-286-9",
discussing "Dau. "The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic," 2003. https://doi.org/10.1007/b94030 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Beta Existential Graph

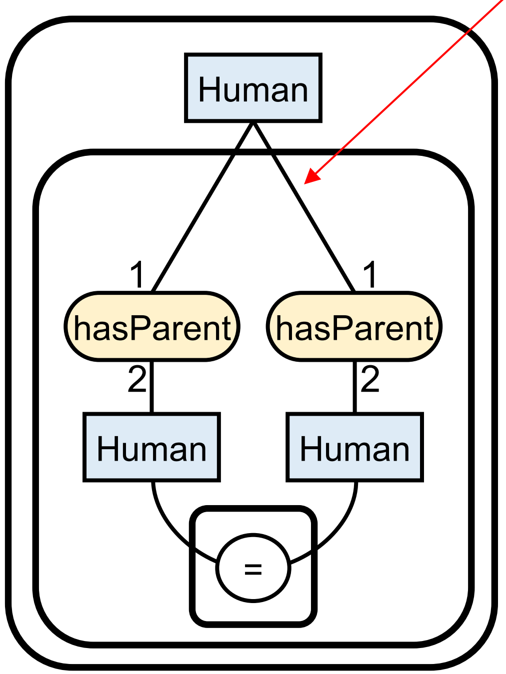
"Every Human has at least two parents who are Human."



Conceptual Graph



Conceptual Graphs (Dau variant)



Lines are used to connect arguments of predicates, not for existential quantification. Relations (like "hasParent") do not need to have concept nodes within their scopes

To express that y ≠ z, we have to say "No entity exists that is equal to both y and z."

Translation:

DRC: $\neg(\exists x[\text{Human}(x) \wedge \neg(\exists y, \exists z[\text{hasParent}(x, y) \wedge \text{Human}(y) \wedge \text{hasParent}(x, z) \wedge \text{Human}(z) \wedge \neg(y = z)])])$

Based on "Chein, Mugnier. "Graph-based knowledge representation," 2009. https://doi.org/10.1007/978-1-84800-286-9"

String Diagrams NPR_{Σ} (2024)

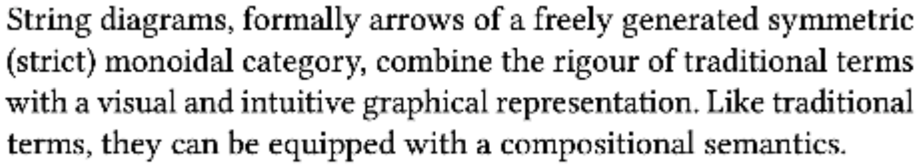
Neo-Peircean Relations w/ signature Σ

Sources used:

- Bonchi, Haydon, Di Giorgio, Sobocinski. Diagrammatic Algebra of First Order Logic, 2024. https://arxiv.org/abs/2401.07055
- Haydon, Sobocinski. Compositional Diagrammatic First-Order Logic, Diagrams 2020. https://doi.org/10.1007/978-3-030-54249-8_32
- Bonchi, Seeber, Sobocinski. Graphical Conjunctive Queries, CSL 2018. https://doi.org/10.4230/LIPIcs.CSL.2018.13
- Personal communication with the authors

String Diagrams NPR$_\Sigma$

- "Neo-Peircean Relations" (NPR$_\Sigma$) is variant of "string diagrams" that redefines Peirce's β Existential Graphs (EGs) in "category theory" (the details of which are hard to follow)

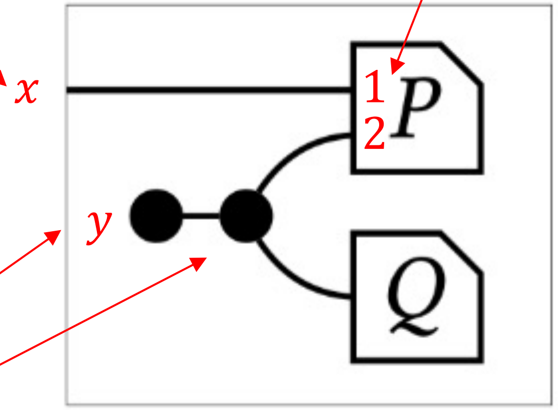


> String diagrams, formally arrows of a freely generated symmetric (strict) monoidal category, combine the rigour of traditional terms with a visual and intuitive graphical representation. Like traditional terms, they can be equipped with a compositional semantics.

- In contrast to Peirce's β-EGs, they can express queries due to an elegant visual distinction between 1) **<u>free variables</u>** (a line w/o marker) and 2) **<u>bound variables</u>** (a line with a **<u>bullet as marker</u>**), which seems graphically inspired by the "free hooks" notation of [Burch'91]
- They are 3) interpreted left-to-right, 4) use bullets also whenever a line splits into segments, and 5) use "<u>photographic negatives</u>" (alternatively shaded areas that also change foreground colors) for negation

For all practical purposes, the arguments for predicates can be thought of as on the side only, and their order top-down.

The default for a line without marker, reaching the end of the bordered Canvas, is that it is free



A bullet at the end of a line means the variable is bound

The outer box (the limits of Peirce's "sheet of assertion") seems not strictly necessary, but simplifies the reading

$$\{(x) \mid \exists y[P(x,y) \wedge Q(y)]\}$$

Picture from "Bonchi, Haydon, Di Giorgio, Sobocinski. Diagrammatic Algebra of First Order Logic, 2024. https://arxiv.org/abs/2401.07055", red annotations are original interpretations.

[Burch'91]: Burch, A Peircean reduction thesis -- the foundations of topological logic. Texas Tech University Press 1991.

String Diagrams NPR_{Σ}

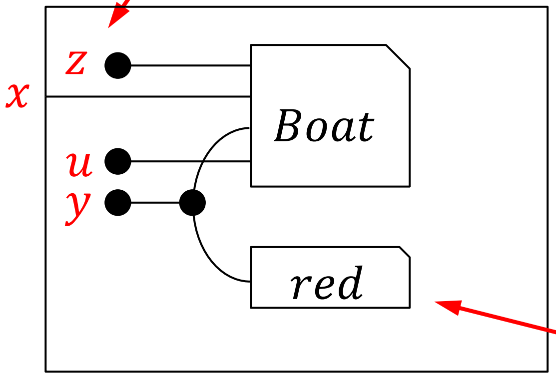
Schema

Boat
bid
bname
color
pdate

Q1a: "Find boats that are red."

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

Marking bounded variables with bullet markers (and free variables without) allows a visual distinction necessary to represent relational queries



Constants (like 'red') are treated as unary predicates

DRC (Domain Relational Calculus)

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

Comparison predicates ("before 1980", "pdate<1980") are not supported.

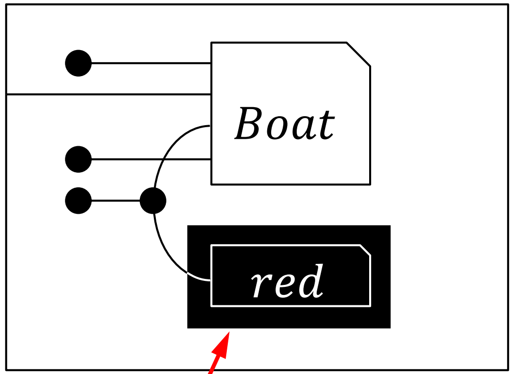
String Diagrams NPR$_{\Sigma}$

Schema

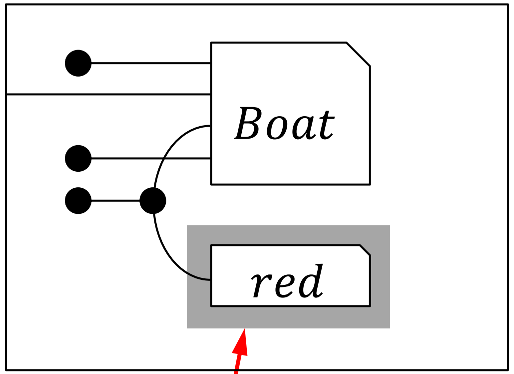
Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

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



Negation is shown by switching colors black / white (notably both background and foreground) called "photograph negative"



This is the closest visual alternative had we applied Peirce's originally suggested "shading" of the background for negation, yet not changing the foreground.

DRC (Domain Relational Calculus)

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

"Peirce shading" as suggested in "Peirce. Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. Paragraph 4.621. https://archive.org/details/collectedpaperso0000unse_r5j9/ "

String Diagrams NPR$_\Sigma$

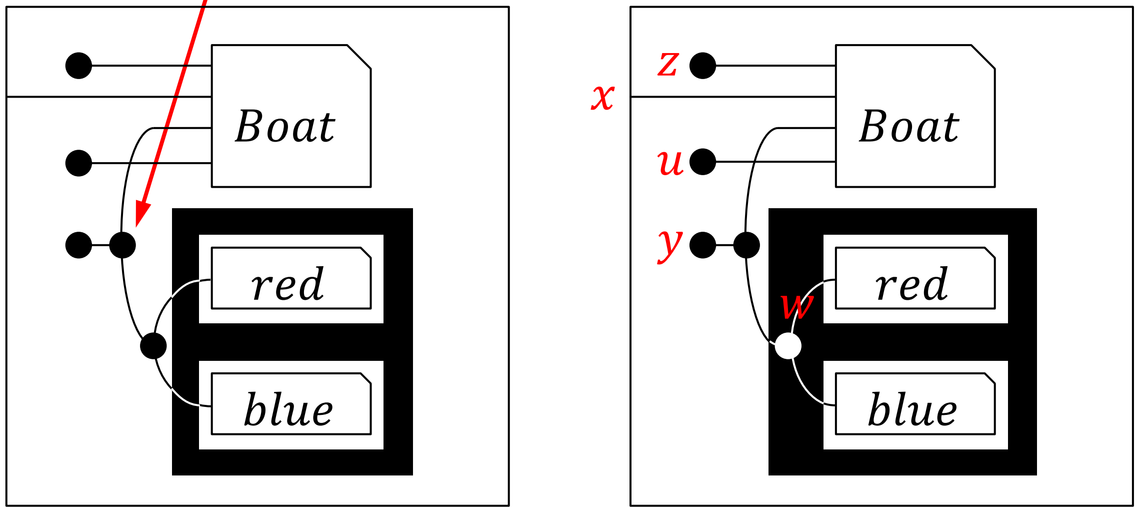
Because of considerations from category theory, every internal bullet can have maximally 3 outgoing lines in a strict syntactical sense (e.g. 1 incoming left and 2 outgoing right, but not 1 incoming left and 3 outgoing right). For practical visual purposes, this enforcement would not matter.

Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

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



Comparison predicates such as "pdate < 1980" are not supported

Disjunction of selection predicates can be represented via De Morgan (double-negation and conjunction)

These two diagram are logically identical, but have a slightly different "literal" interpretation. The right can be read as:

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

DRC (Domain Relational Calculus)

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

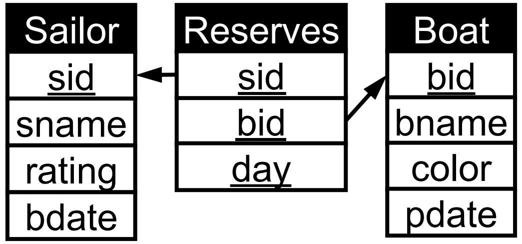
{(x) | ∃y,z,u [Boat(z,x,y,u) ∧ ¬(¬(y='red') ∧ ¬(y='blue'))]}

String Diagrams NPR$_\Sigma$

Q2: "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'

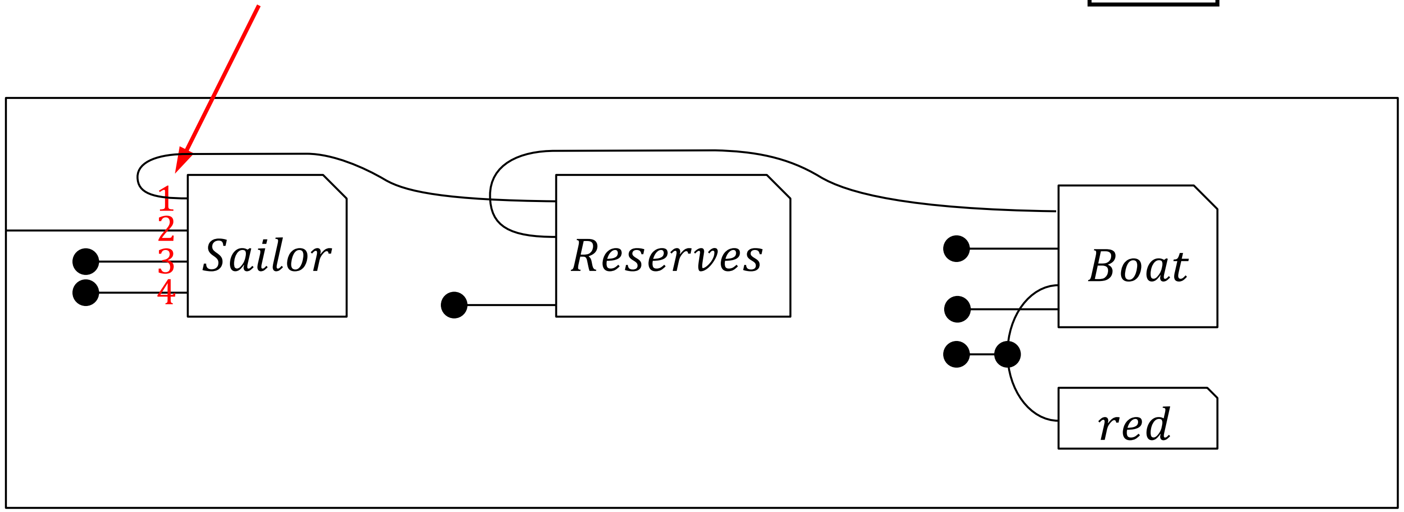
Schema



Sailor
<u>sid</u>
sname
rating
bdate

Reserves
<u>sid</u>
<u>bid</u>
<u>day</u>

Boat
<u>bid</u>
bname
color
pdate



Lines from arguments of predicates follow the order from the relational schema

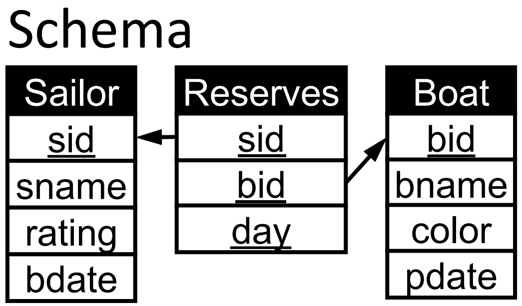
DRC (Domain Relational Calculus)

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

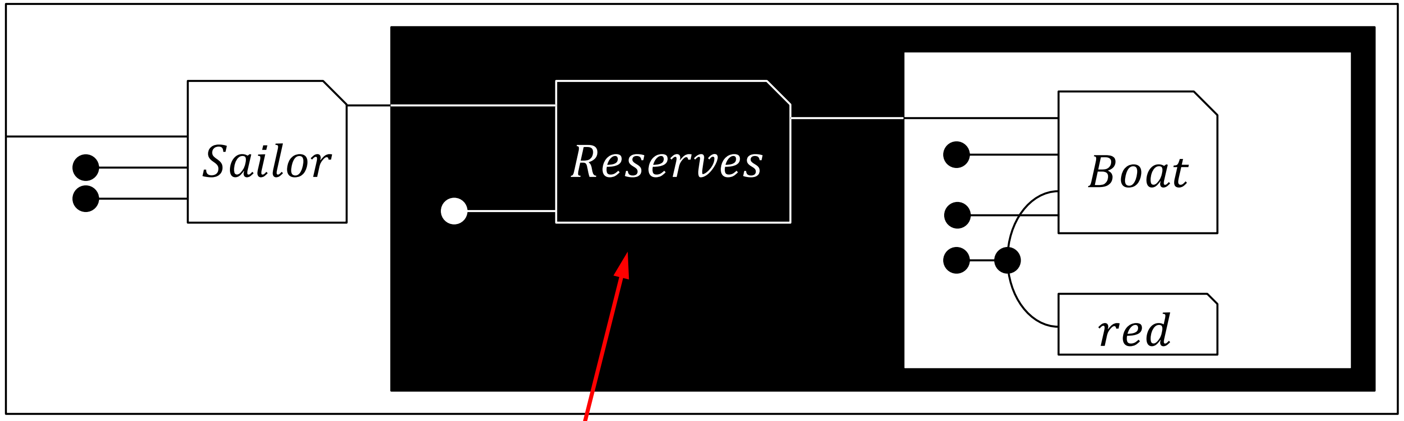
String Diagrams NPR_{Σ}

Q3: "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'))
```



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



Negation is shown by using Peirce's "cuts" and switching colors between black / white (notably both background and foreground)

DRC (Domain Relational Calculus)

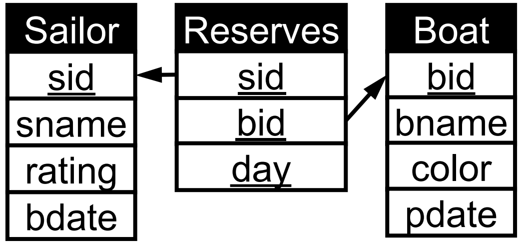
{(x) | ∃v,z,w [Sailor(v,x,z,w) ∧ ¬(∃y,t [Reserves(v,y,t) ∧ ¬(∃u,s [Boat(y,u,'red',s)])])]}

String Diagrams NPR_{Σ}

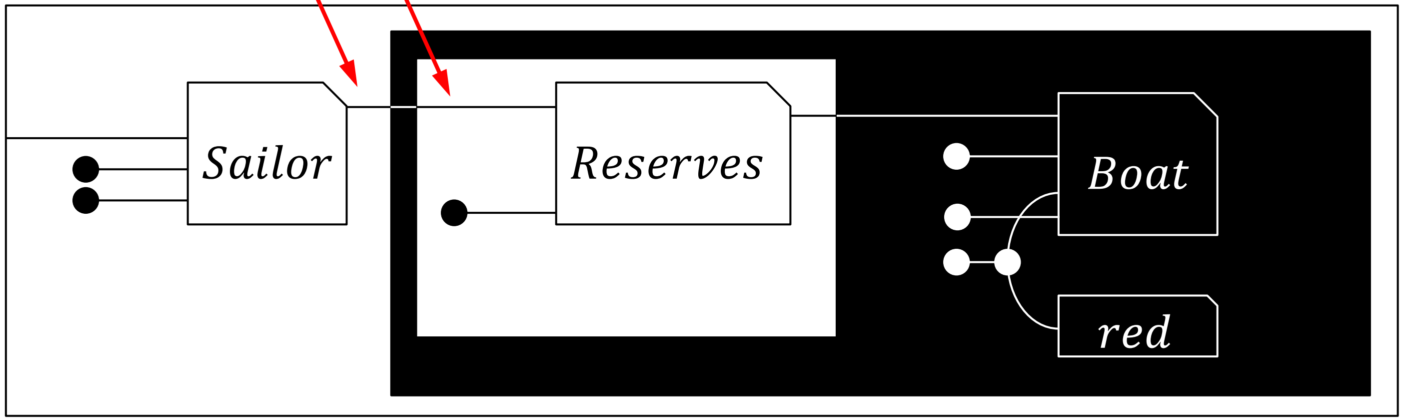
Q4: "Find sailors who reserved all red boats."

```
select distinct 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))
```

Schema



Line segments change color between black and white based on background



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)])])]}

String Diagrams NPR$_\Sigma$

Schema

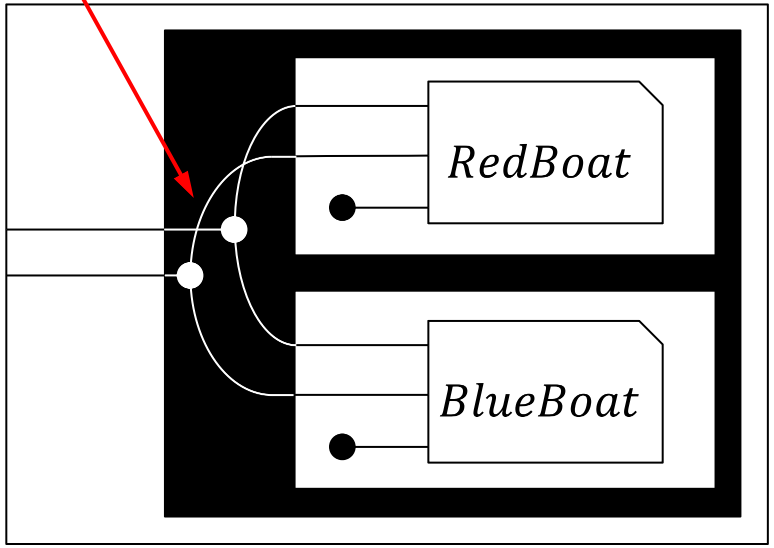
RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

Union of table can be represented via De Morgan (double-negation and conjunction). However, notice that the resulting expression is not safe (free variables x and y are not "guarded", i.e. not bound to an element from the active domain)



DRC (Domain Relational Calculus)

$\{(x,y) \mid \exists z\, [RedBoat(x,y,z) \vee BlueBoat(x,y,z)]\}$

$\{(x,y) \mid \neg(\neg(\exists z\, [RedBoat(x,y,z)]) \wedge \neg(\exists z\, [BlueBoat(x,y,z)]))\}$

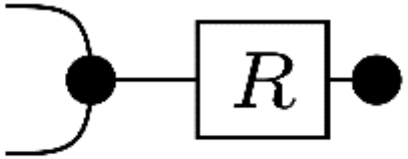
An accessible overview of issues involving safety is: Topor, Safety and Domain Independence, Encyclopedia of Database Systems. 2009 https://doi.org/10.1007/978-0-387-39940-9_1255

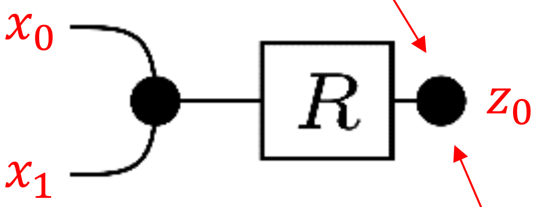
BACKUP
String Diagrams

String Diagrams NPR$_\Sigma$

In contrast to Peirce's β-EGs, the default for a line without bullet is to be free

The left side of predicates have "arities", the right side has "co-arities", a notion from category theory that appears to depend on the actual query, not just the schema. Here, R.$2 is the co-arity





$$\phi = \exists z_0 \colon (x_0 = x_1) \wedge R(x_0, z_0)$$

notation of paper

A bullet as arrowhead of a line binds a variable

$$\{(x_0, x_1) \mid \exists z_0\, [R(x_0, z_0) \wedge x_0 = x_1]\}$$

notation of this survey

free

bound

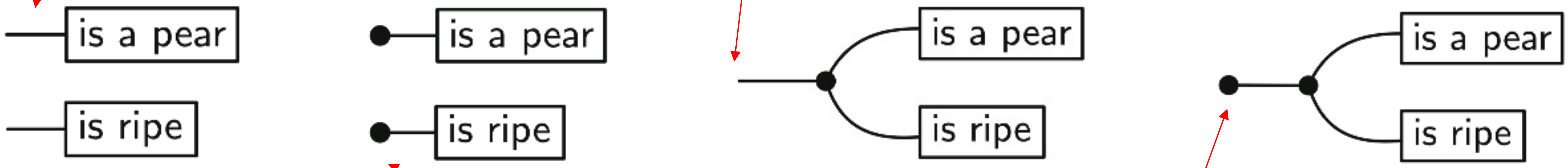
Picture from "Bonchi, Seeber, Sobocinski. Graphical Conjunctive Queries. CSL 2018. https://doi.org/10.4230/LIPIcs.CSL.2018.13", red annotations are original interpretations

String Diagrams NPR_Σ

Free variables: lines are dangling ("hooks are unfilled")

$\{(x, y) \mid \text{is\_a\_pear}(x) \land \text{is\_ripe}(y)\}$

$\{x \mid \text{is\_a\_pear}(x) \land \text{is\_ripe}(x)\}$



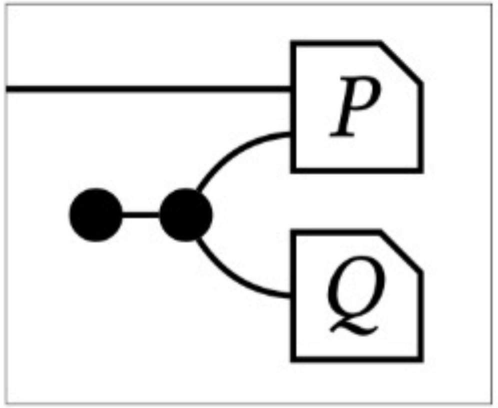
$\exists x\, [\text{is\_a\_pear}(x)] \land \exists y\, [\text{is\_ripe}(y)]$

$\exists x\, [\text{is\_a\_pear}(x) \land \text{is\_ripe}(x)]$

Bound variables: lines are capped off with a bullet as arrowhead

Picture from " Haydon, Sobocinski. Compositional Diagrammatic First-Order Logic, Diagrams 2020. https://doi.org/10.1007/978-3-030-54249-8_32"

String Diagrams NPR$_\Sigma$



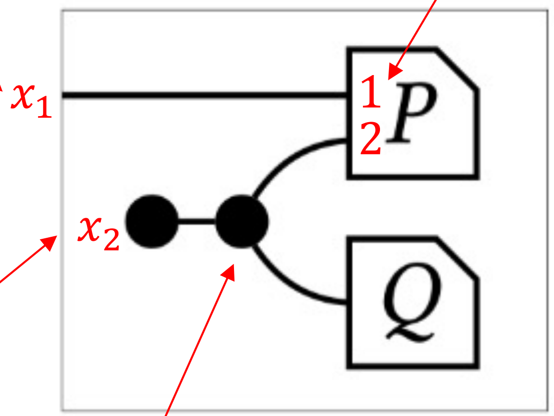
FOL formula $\exists x_2.P(x_1, x_2) \wedge Q(x_2)$

notation of paper

Outer bounding box appears not necessary

The order of arguments for predicates seems top down.

Free variable



Bound variable w/ bullets at line end

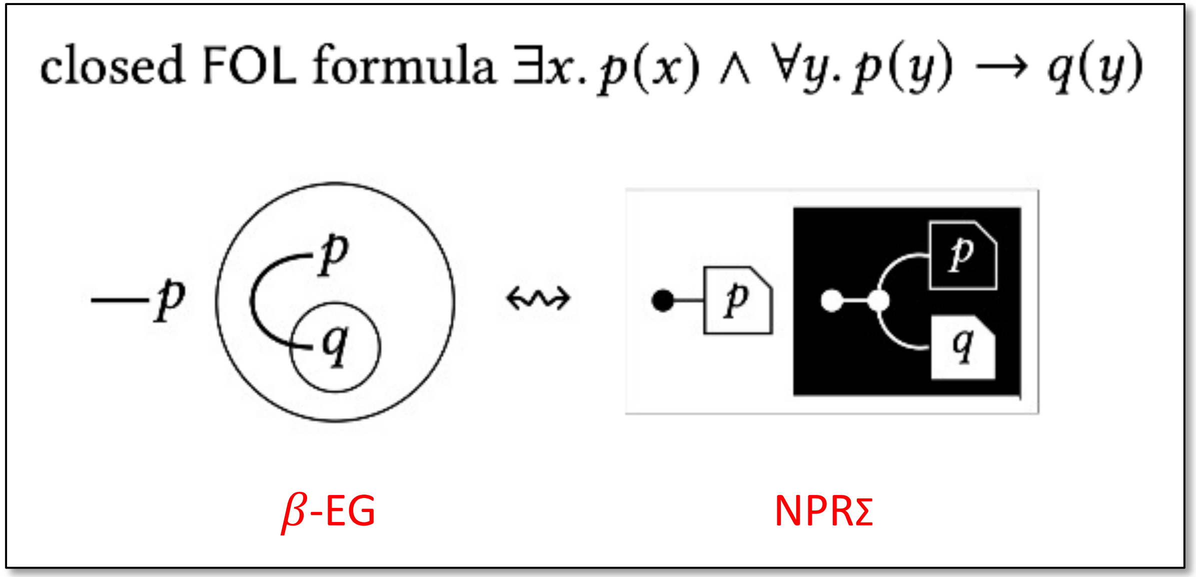
Bullets are also used for splitting / connecting line segments

$$\{(x_1) \mid \exists x_2[P(x_1, x_2) \wedge Q(x_2)]\}$$

notation of this survey

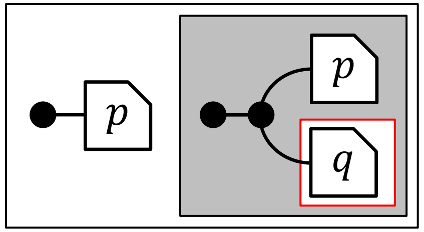
Picture from "Bonchi, Haydon, Di Giorgio, Sobocinski. Diagrammatic Algebra of First Order Logic, 2024. https://arxiv.org/abs/2401.07055", red annotations are original interpretations

Connection of String Diagrams NPR_Σ to Peirce's β-EGs

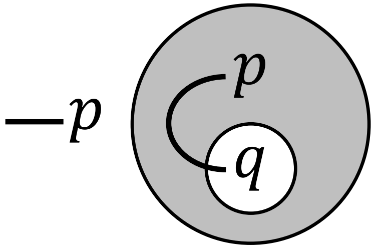

closed FOL formula $\exists x.\, p(x) \land \forall y.\, p(y) \to q(y)$

β-EG

NPRΣ



NPR_Σ with "Peirce shading" which would separates the depiction of the predicates and variables (the foreground is always in black) from the alternatively shaded and nested bounding boxes (notice here the red bounding box missing on the left)



β-EG with shading proposed by Peirce

Figure 8 from "Bonchi, Haydon, Di Giorgio, Sobocinski. Diagrammatic Algebra of First Order Logic, 2024. https://arxiv.org/abs/2401.07055"

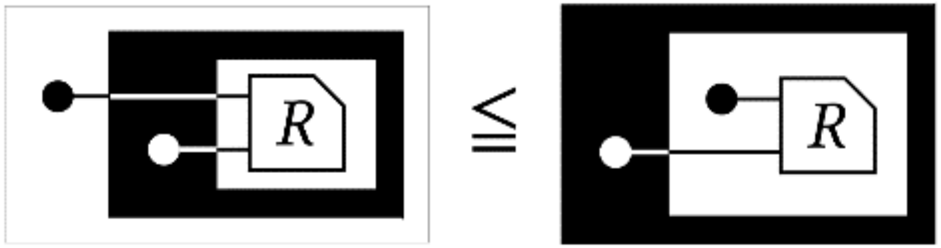
"Peirce shading": See paragraph 4.621 in Collected Papers of Charles Sanders Peirce. Volumes 3 and 4. 1933. https://archive.org/details/collectedpaperso0000unse_r5j9/

Contrast "Diagrammatic representation" from "Diagrammatic reasoning"

Our main focus in "diagrammatic representation (of relational queries)" is to help users quickly and unambiguously understand the semantics of a query. In contrast, the focus in "diagrammatic reasoning" is to support proofs in a visual formalism.

Example: $\exists x. \forall y. R(x, y) \models \forall y. \exists x. R(x, y)$

$\exists x. \forall y. R(x, y)$



$\forall y. \exists x. R(x, y)$

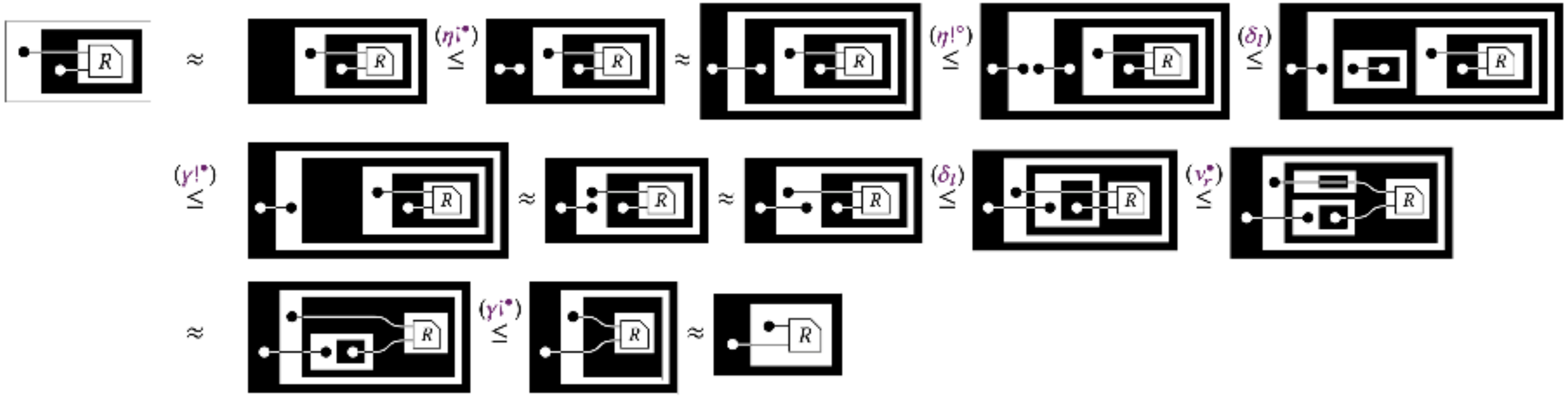


Figure 10: Completely axiomatic proof of (1).

Figures from "Bonchi, Haydon, Di Giorgio, Sobocinski. Diagrammatic Algebra of First Order Logic, 2024. https://arxiv.org/abs/2401.07055"

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

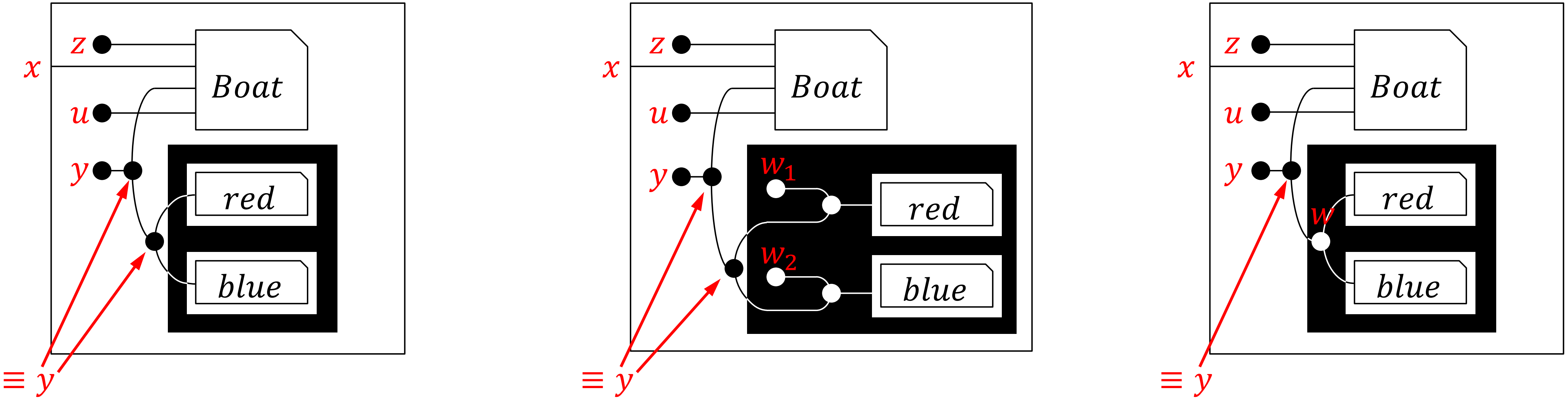
String Diagrams NPR$_\Sigma$

Three equivalent representations with slightly different "literal" interpretation

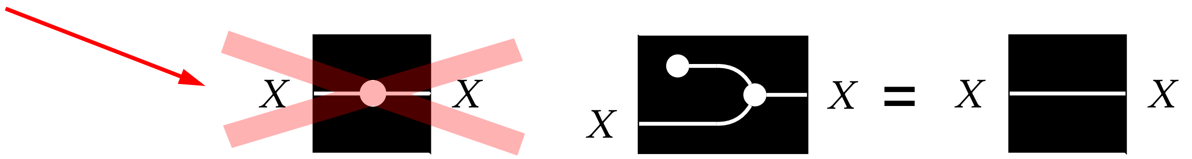
Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."



The strict syntax does not allow to have a dot attached to a line on the left and a line on the right (but intuitively, that's the interpretation):



DRC (Domain Relational Calculus)

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

middle: {(x) | ∃y,z,u [Boat(z,x,y,u) ∧ ¬(∃w_1,w_2 [w_1=y ∧ w_2=y ∧ ¬(w_1='red') ∧ ¬(w_2='blue')])]}

right: {(x) | ∃y,z,u [Boat(z,x,y,u) ∧ ¬(∃w [w=y ∧ ¬(w='red') ∧ ¬(w='blue')])]}

Based on personal communication with the authors

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 (from DB community)
6. Lessons Learned and Open Challenges

Part 5: Visual query representations (from DB community)

- We next look at various visual representations for relational queries that were proposed from *inside* the database community.
- Many of these formalisms were proposed as "Visual Query Language" and thus as alternative to SQL for specifying queries.
- Our perspective is thus often different from the original authors' goals: here, we are only interested the visual formalisms.

Part 5: Visual query representations (from DB community)

Please leave feedback
on the tutorial page ☺

How the selection of work was made:
1) if highly cited or influential, or
  2a) it represents a 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: It's 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 I can fix.

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 the tools have other interesting contributions (like the interactive exploration, or spreadsheet-type interface). Our focus today is uniquely on the visual representation.

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

QBE (1977)
(Query-By-Example)

Sources used:

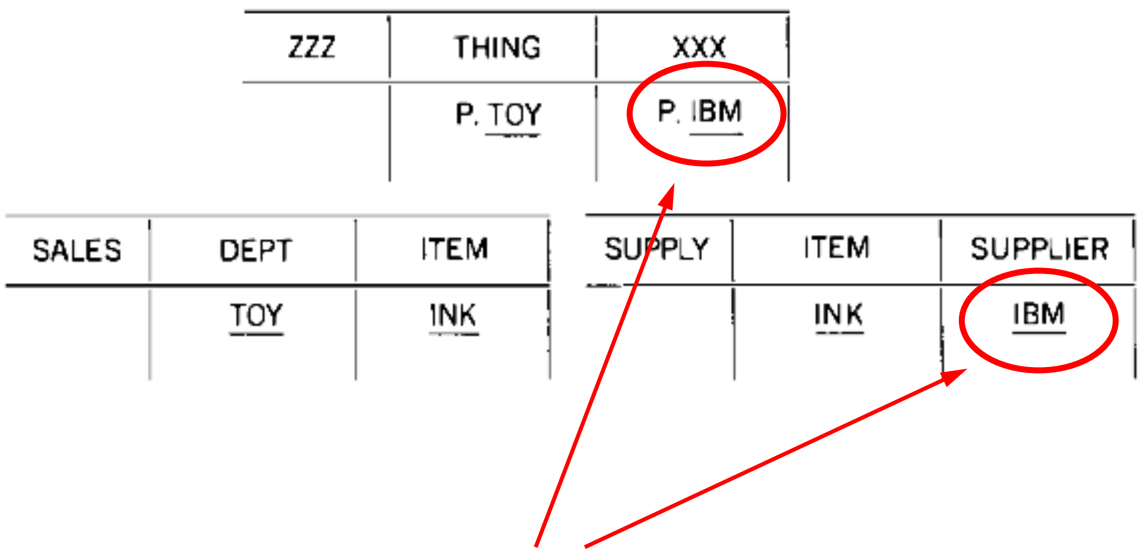
- Zloof. Query-by-Example: A Data Base Language. IBM Systems Journal 16(4). 1977. https://doi.org/10.1147/sj.164.0324
- Ramakrishnan, Gehrke. Database management systems, 2nd ed, 2000. Section 6. https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf
- Elmasri, Navathe. Fundamentals of Database Systems, 7th ed, 2015. Appendix C. https://dl.acm.org/doi/10.5555/2842853

A nice comparison of various extensions is "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



ZZZ	THING	XXX
	P. TOY	P. IBM

SALES	DEPT	ITEM
	TOY	INK

SUPPLY	ITEM	SUPPLIER
	INK	IBM

- 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 than "Query-By-Example"

- Described by the creator Zloof as "the first visual programming language". But is QBE really "visual", i.e. is it diagrammatic? (I would rather call it a 2D variant of Datalog)

Picture from: Zloof. Query-by-Example: A Data Base Language. IBM Systems Journal 16(4). 1977. https://doi.org/10.1147/sj.164.0324

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

```
select distinct 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]

Actual database table

Relational schema provides a template

Boat

bid	bname	color	pdate
	P.	!=red	

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

Influenced by DRC and similar to Datalog

P. = "Print" = part of output

Datalog

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

```
select distinct 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"

Datalog

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

QBE (Query-By-Example)

Schema

Boat
<u>bid</u>
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

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

Intuitively, QBE represents the selection predicates in DNF

```
... where (color = 'red' and pdate < 1980)
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"

Datalog

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

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

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

Boat

bid	bname	color	pdate
	P.	_c	_d

Conditions

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

Variables (of arbitrary name) are expressed with underscores

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

Datalog

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

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

```
select distinct 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?

?

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)

Datalog (Souffle syntax)

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

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

```
select distinct 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)

Datalog (Souffle syntax)

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

QBE (Query-By-Example)

Schema

Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

```
select distinct 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?

I believe QBE should not be called a "visual programming language" (any more than Datalog); and *we* should stop continuing this historical mis-classification in our undergraduate database classes!

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

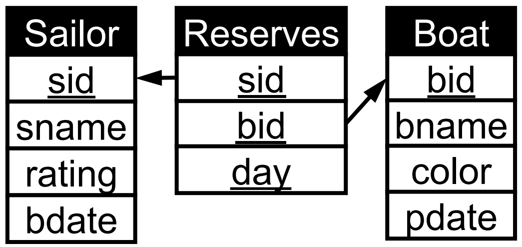
_d<1980 and (_c=red or _c=blue)

Datalog (Souffle syntax)

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

QBE (Query-By-Example)

Schema



Sailor
sid
sname
rating
bdate

Reserves
sid
bid
day

Boat
bid
bname
color
pdate

Q2: "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'
```

Sailor

sid	sname	rating	bdate
_y	P.		

Reserves

sid	bid	day
_y	_x	

Boat

bid	bname	color	pdate
_x		red	

The "join structure" is not visualized any different from Datalog!

Datalog

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

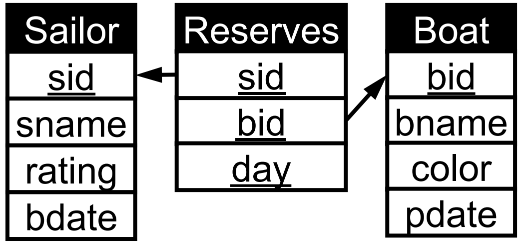
QBE (Query-By-Example)

Q2a: "Find sailors and red boats they reserved."

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

The "join structure" is not visualized any different from Datalog!

Schema



Sailor
sid
sname
rating
bdate

Reserves
sid
bid
day

Boat
bid
bname
color
pdate

Sailor

sid	sname	rating	bdate
_y	_z		

Reserves

sid	bid	day
_y	_x	

Boat

bid	bname	color	pdate
_x	_w	red	

Output

sname	bname
P._z	P._w

We need a separate output table in order to collect attributes from different input tables

Datalog

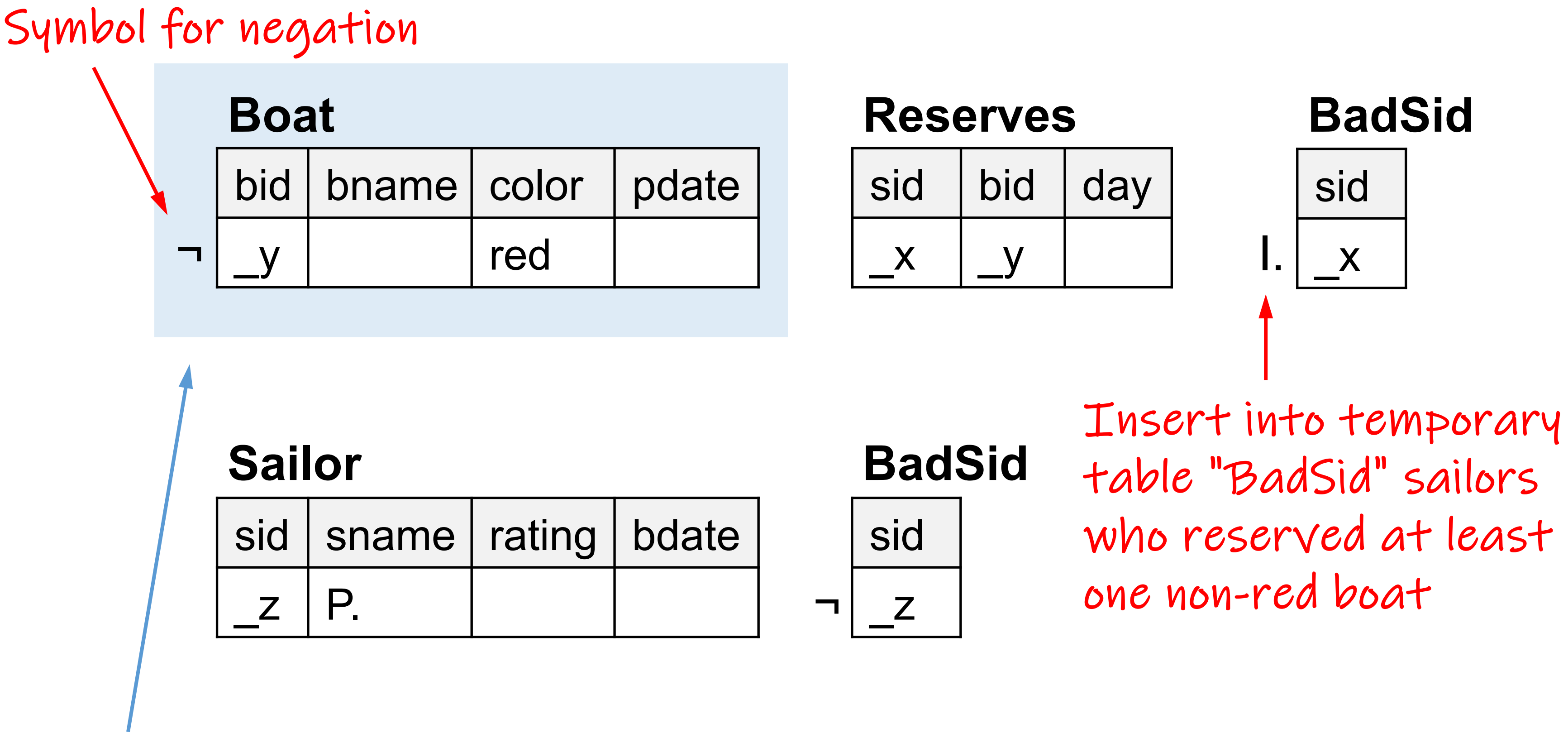
Q(z,w) :- Sailor(y,z,_,_), Reserves(y,x,_), Boat(x,w,'red',_)

QBE (Query-By-Example)

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

Q3: "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'))
```



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

QBE seems to allow a specification that would be unsafe in Datalog: "..., not Boat(y,_,'red',_)". Datalog needs to use an extra intermediate relation "RedBoat"

Datalog

```
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

Q3: "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'))
```

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...

Datalog

```
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

Q3: "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'))
```

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 supported in any current implementations)

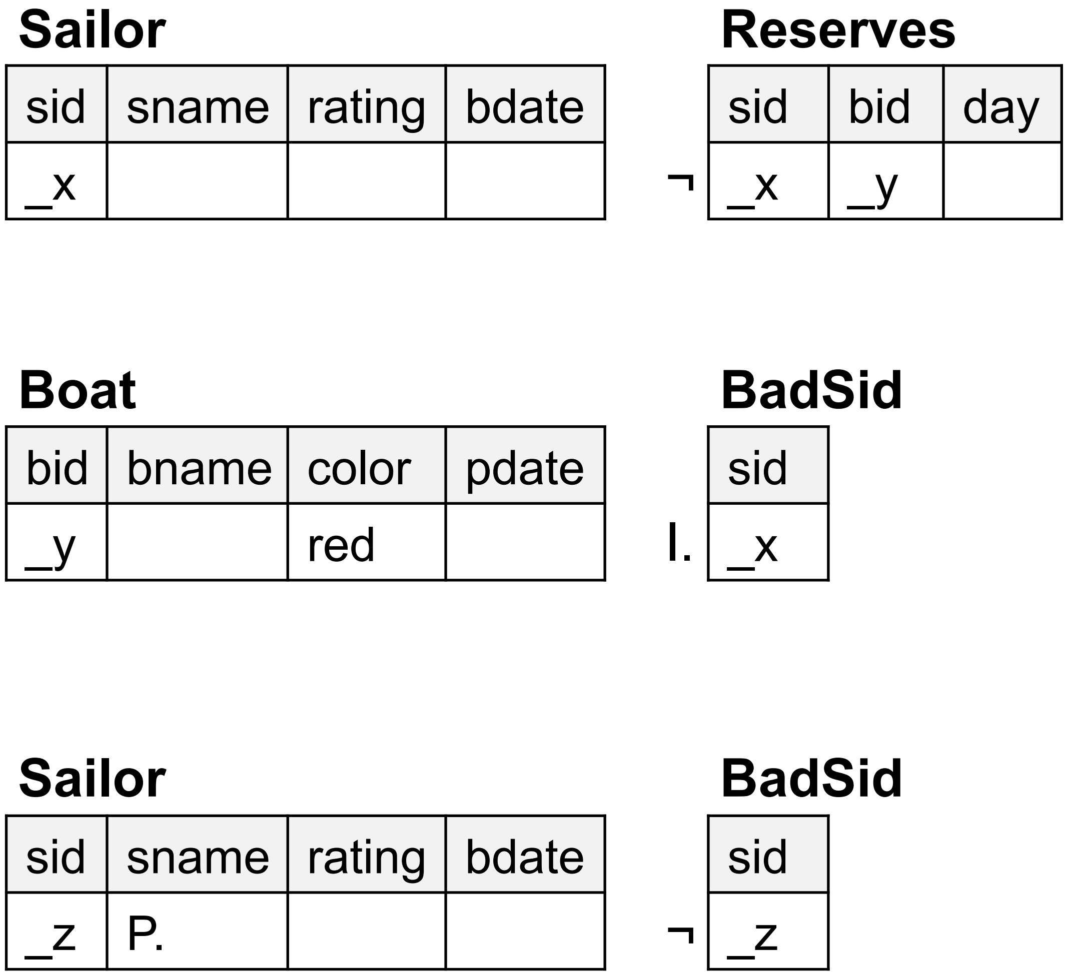
Datalog

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)

Q4: "Find sailors who reserved all red boats."

```
select distinct 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))
```



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

Datalog

```
       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)

?

Q4: "Find sailors who reserved all red boats."

```
select distinct 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))
```

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?)

Datalog

```
        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	

I. **BadSid**

sid
_x

Sailor

sid	sname	rating	bdate
_z	P.		

¬ **BadSid**

sid
_z

QBE (Query-By-Example)

Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

[Zlopf'77] suggests that we need a separate output tuple for expressing a table union

RedBoat

bid	bname	pdate
_i1	_n1	

BlueBoat

bid	bname	pdate
_i2	_n2	

Output

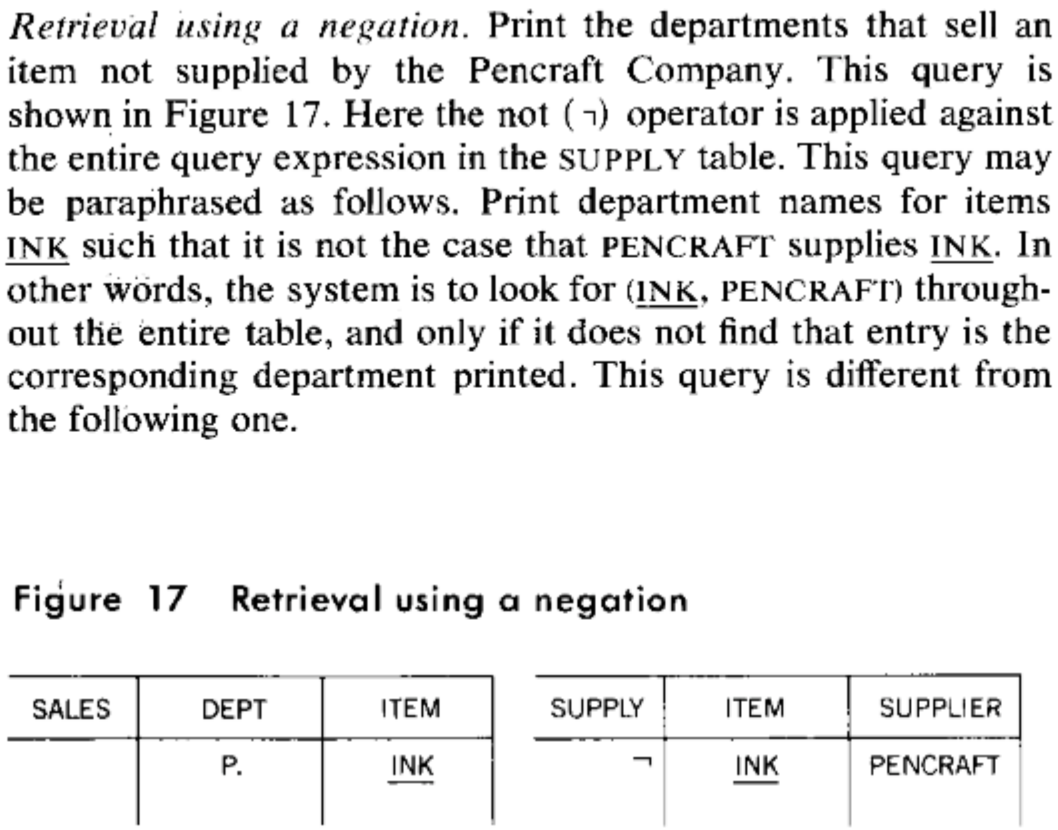
bid	bname
P._i1	P._n1
P._i2	P._n2

Datalog

Q(x,y) :- RedBoat(x,y,_)
Q(x,y) :- BlueBoat(x,y,_)

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

QBE (Query-By-Example)

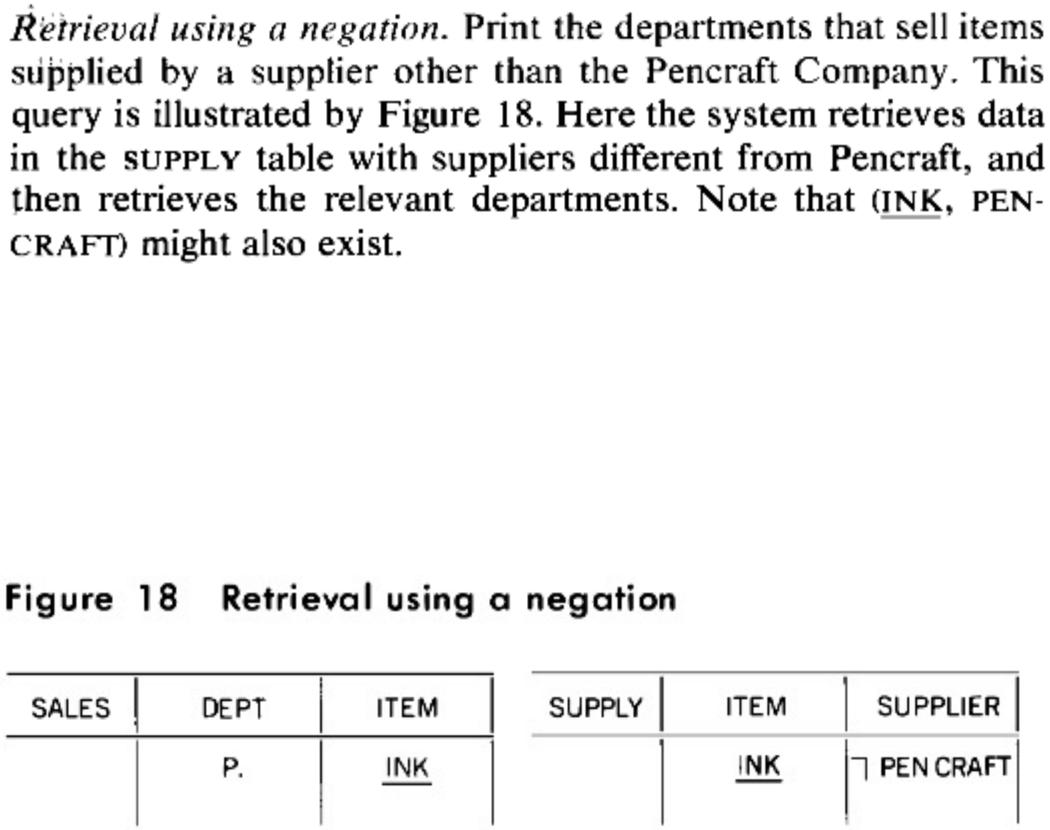


Retrieval using a negation. Print the departments that sell an item not supplied by the Pencraft Company. This query is shown in Figure 17. Here the not (¬) operator is applied against the entire query expression in the SUPPLY table. This query may be paraphrased as follows. Print department names for items INK such that it is not the case that PENCRAFT supplies INK. In other words, the system is to look for (INK, 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.	INK

SUPPLY	ITEM	SUPPLIER
¬	INK	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 (INK, PENCRAFT) might also exist.

Figure 18 Retrieval using a negation

SALES	DEPT	ITEM
	P.	INK

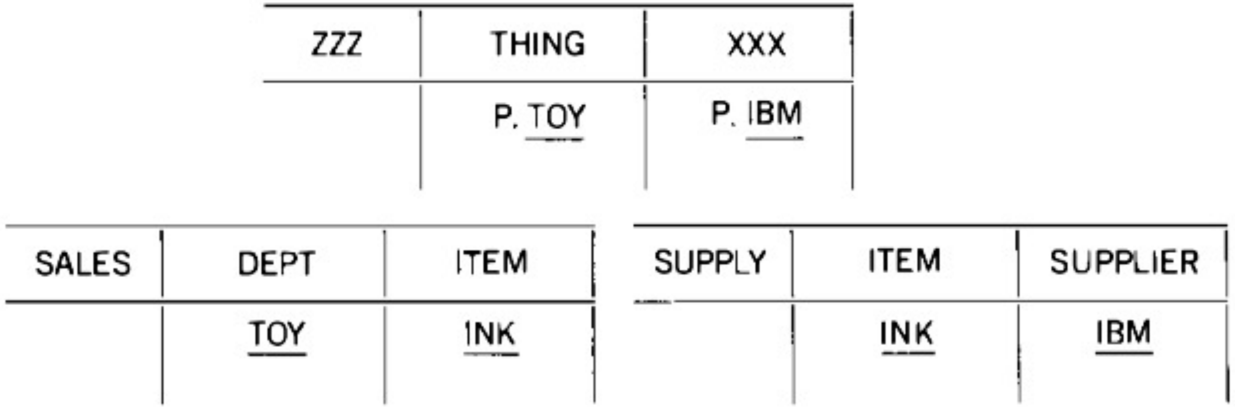
SUPPLY	ITEM	SUPPLIER
	INK	¬ PEN CRAFT

Source: 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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



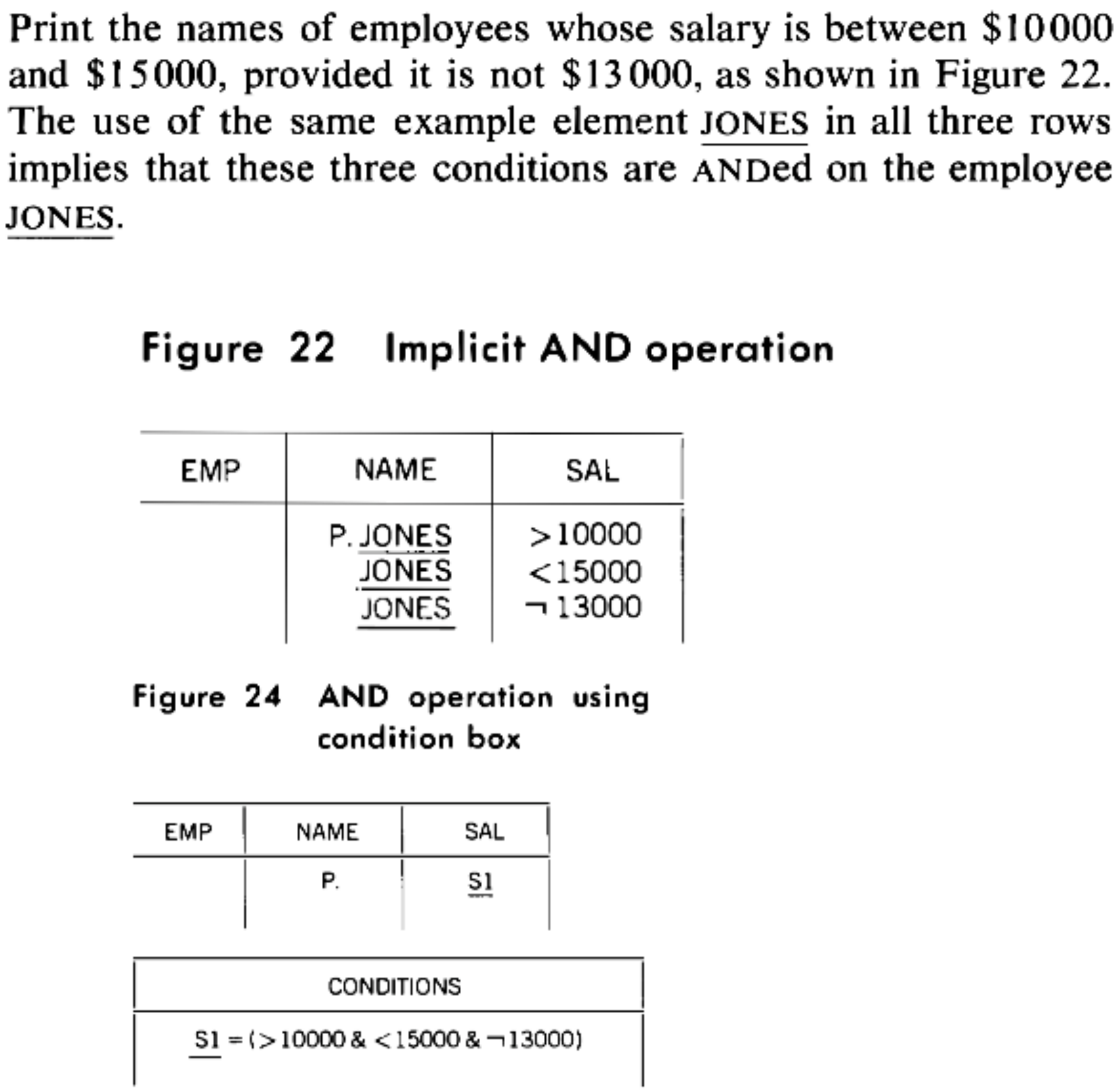
ZZZ	THING	XXX
	P. TOY	P. IBM

SALES	DEPT	ITEM
	TOY	INK

SUPPLY	ITEM	SUPPLIER
	INK	IBM

Source: Zloof. Query-by-Example: A Data Base Language. IBM Systems Journal 16(4). 1977. https://doi.org/10.1147/sj.164.0324

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.

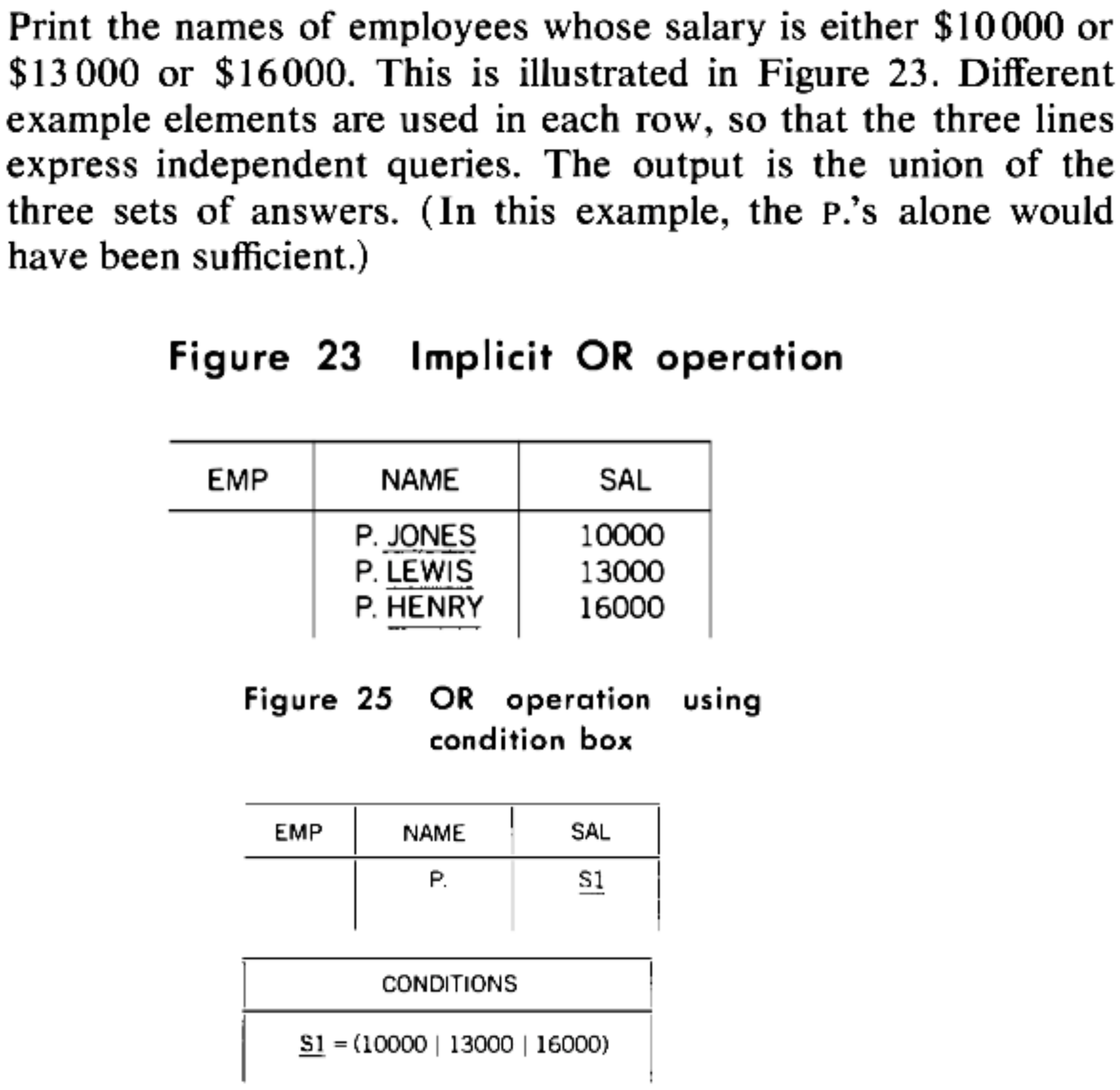
Figure 22 Implicit AND operation

EMP	NAME	SAL
	P. JONES	>10000
	JONES	<15000
	JONES	¬ 13000

Figure 24 AND operation using condition box

EMP	NAME	SAL
	P.	S1

CONDITIONS
S1 = (>10000 & <15000 & ¬13000)



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 23 Implicit OR operation

EMP	NAME	SAL
	P. JONES	10000
	P. LEWIS	13000
	P. HENRY	16000

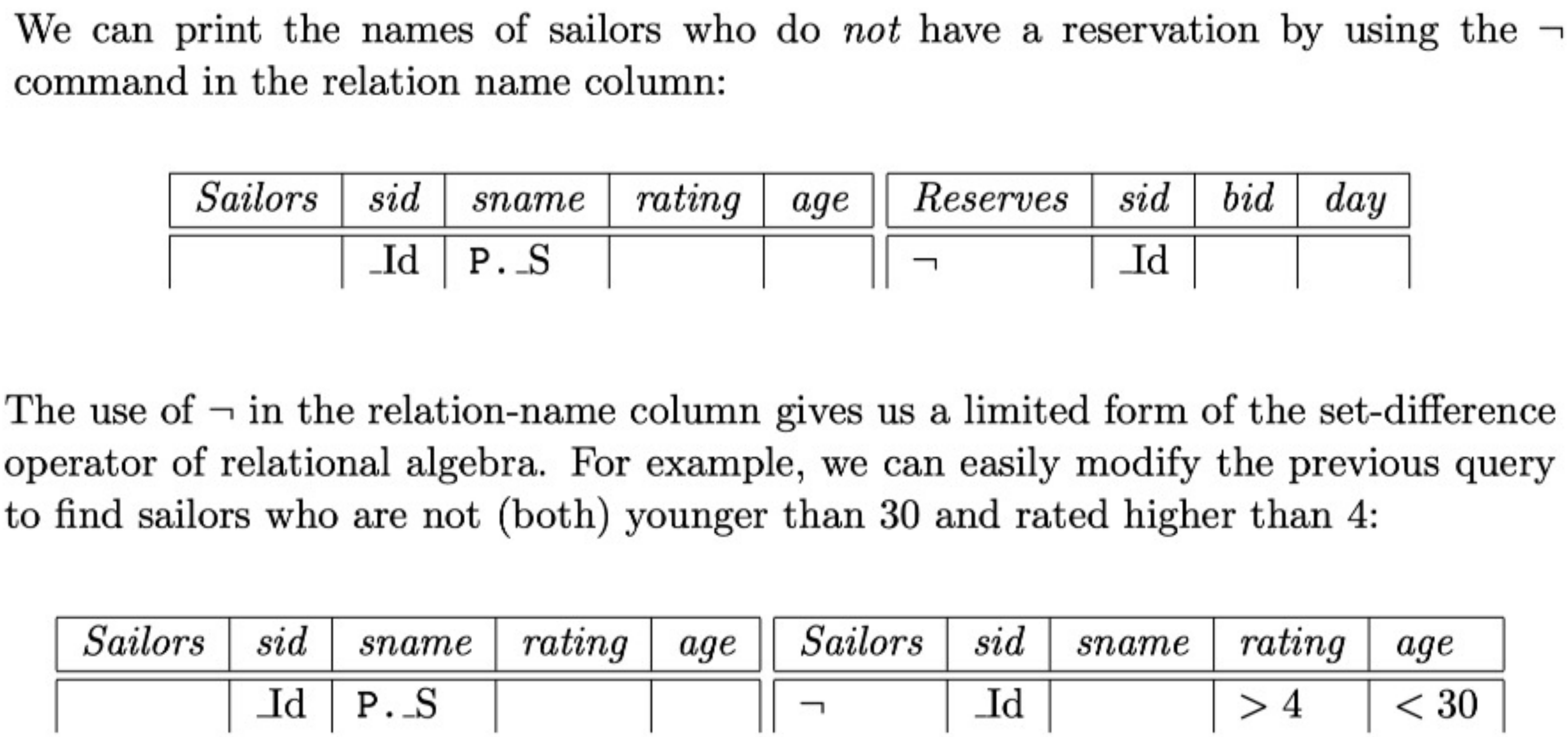
Figure 25 OR operation using condition box

EMP	NAME	SAL
	P.	S1

CONDITIONS
S1 = (10000 | 13000 | 16000)

Source: 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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:

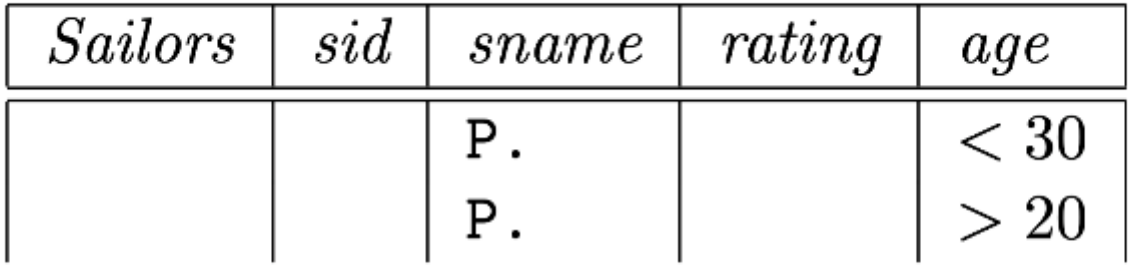
Sailors	*sid*	*sname*	*rating*	*age*
	_Id	P._S		

Sailors	*sid*	*sname*	*rating*	*age*
¬	_Id		> 4	< 30

Source: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000). Section 6. https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

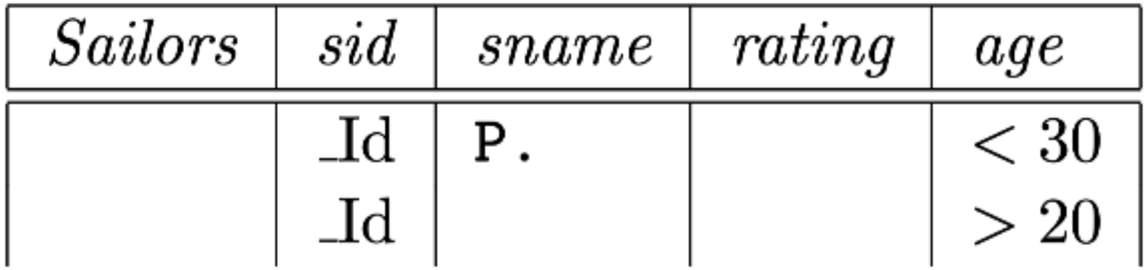
QBE (Query-By-Example)

Q: "Print the names of sailors who are younger than 30 or older than 20."



Sailors	*sid*	*sname*	*rating*	*age*
		P.		< 30
		P.		> 20

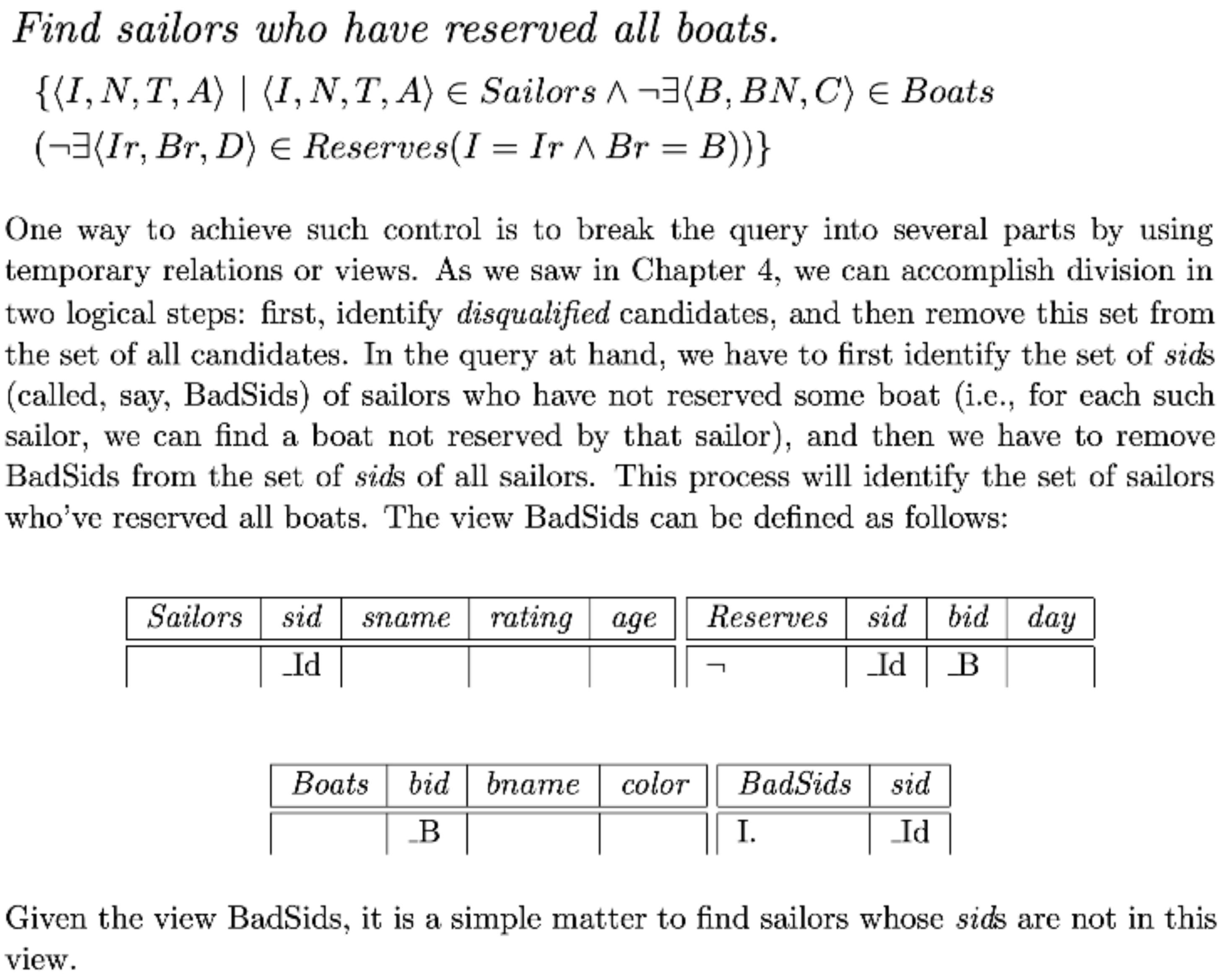
Q: "Print the names of sailors who are both younger than 30 and older than 20."



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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QBE (Query-By-Example)



Find sailors who have reserved all boats.

$$\{\langle I, N, T, A\rangle \mid \langle I, N, T, A\rangle \in Sailors \wedge \neg\exists\langle B, BN, C\rangle \in Boats$$
$$(\neg\exists\langle Ir, Br, D\rangle \in Reserves(I = Ir \wedge Br = B))\}$$

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	

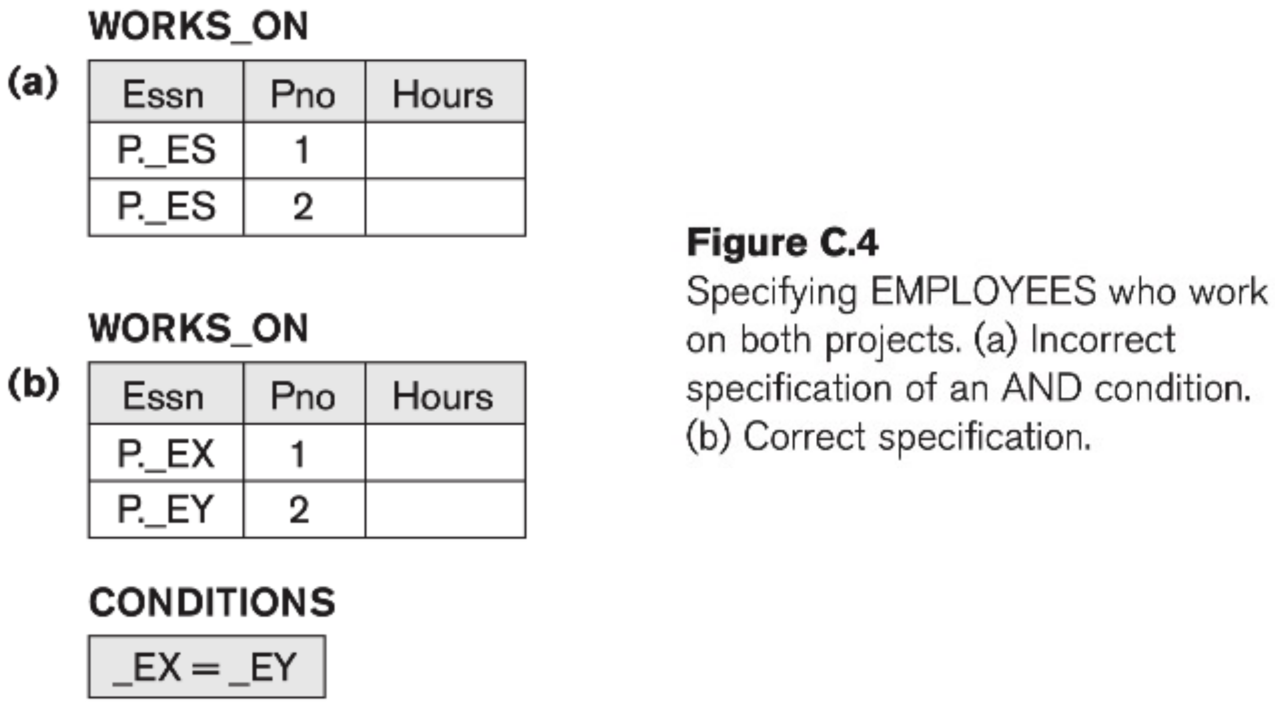
Boats	*bid*	*bname*	*color*
	_B		

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QBE (Query-By-Example)



WORKS_ON

(a)

Essn	Pno	Hours
P._ES	1	
P._ES	2	

WORKS_ON

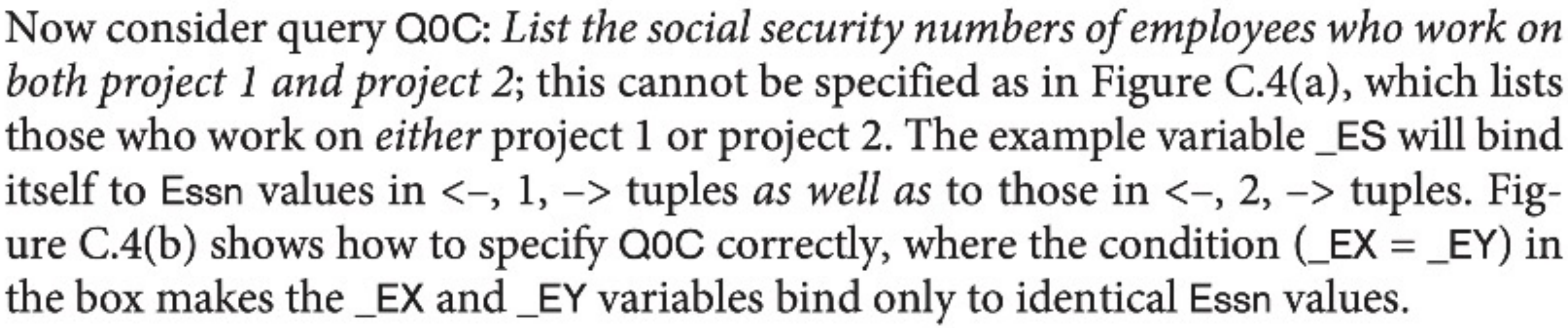
(b)

Essn	Pno	Hours
P._EX	1	
P._EY	2	

CONDITIONS

_EX = _EY

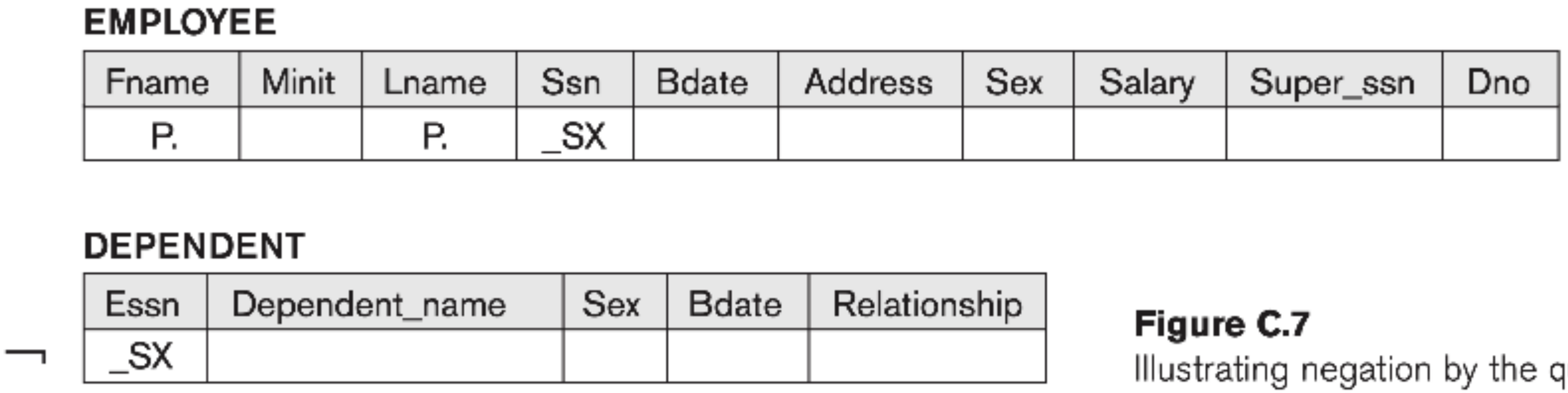
Figure C.4
Specifying EMPLOYEES who work on both projects. (a) Incorrect specification of an AND condition. (b) Correct specification.



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

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 );
```

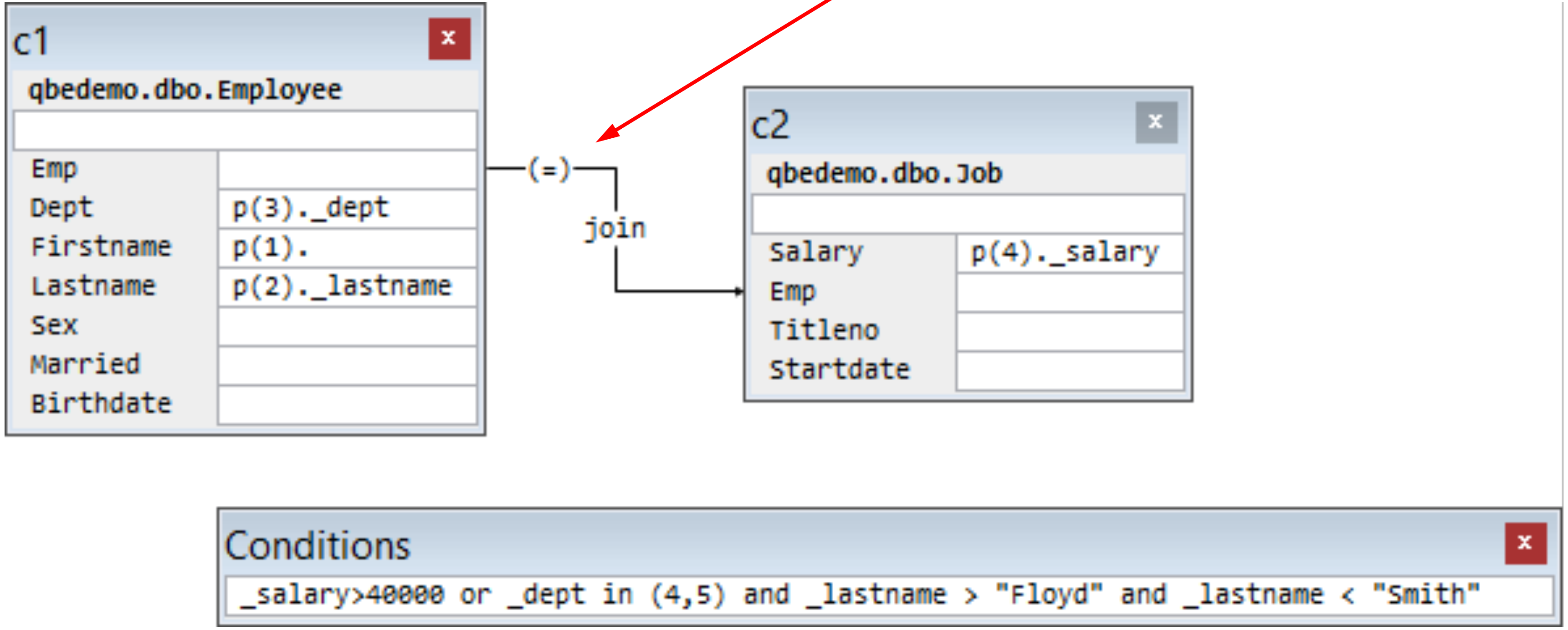
Q6: $\{q, s \mid (\exists t)(\text{EMPLOYEE}(qrstuvwxyz) \textbf{ AND } (\textbf{NOT}(\exists l)(\text{DEPENDENT}(lmnop) \textbf{ AND } t{=}l)))\}$

Source: Elmasri, Navathe. Fundamentals of Database Systems, 7th ed, 2015. Appendix C. https://dl.acm.org/doi/10.5555/2842853
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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 <u>visual join syntax</u>.



c1 (qbedemo.dbo.Employee):

Emp	
Dept	p(3)._dept
Firstname	p(1).
Lastname	p(2)._lastname
Sex	
Married	
Birthdate	

c2 (qbedemo.dbo.Job):

Salary	p(4)._salary
Emp	
Titleno	
Startdate	

Conditions:

```
_salary>40000 or _dept in (4,5) and _lastname > "Floyd" and _lastname < "Smith"
```

Source: http://www.sysdeco.no/documentation/17/querybuilder/index.html#!WordDocuments/orvisualqueryconditionboxoperator.htm

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

QBD (1990)
(Query By Diagram)

Sources used:

- 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
- Filters and join conditions are specified in separate windows (like in visual query builders)
- Our focus is here just the visual metaphors as possibly applied to relations directly

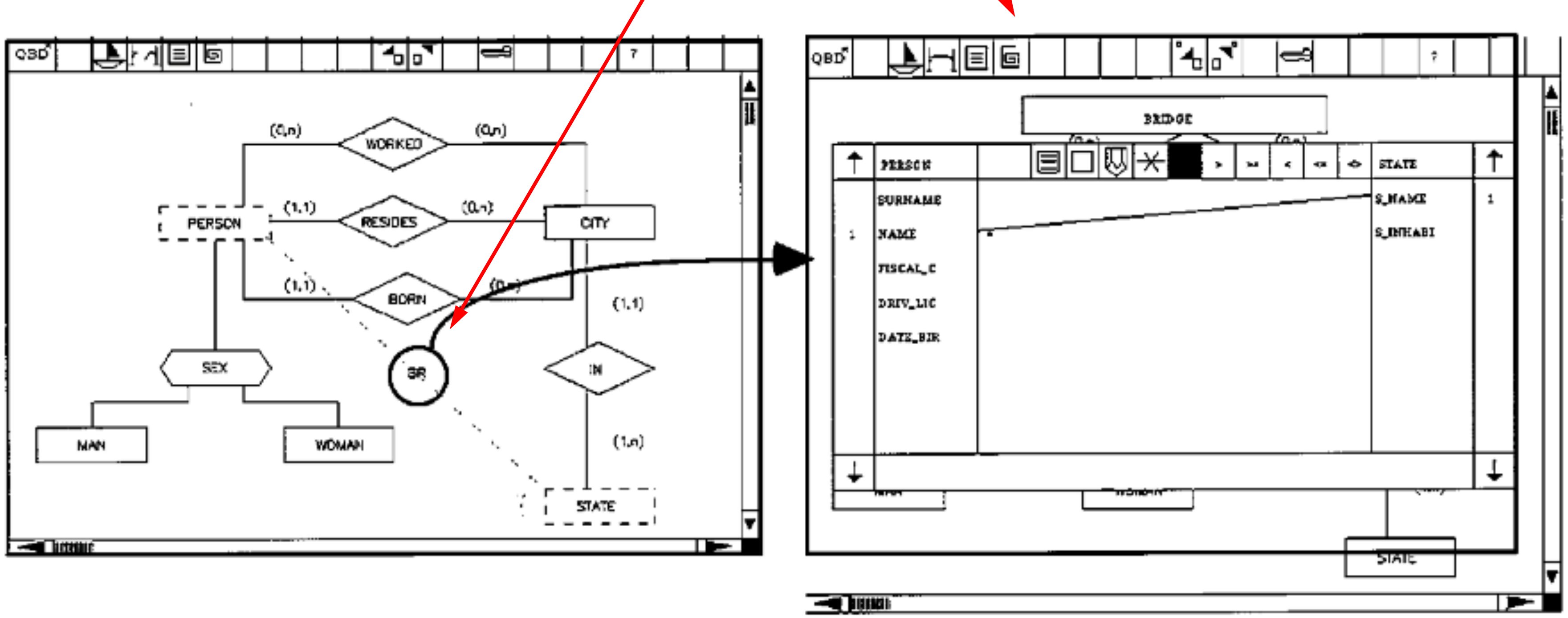


Figure source: "Catarci, Costabile, Levialdi, Batini. Visual query systems for databases: a survey. JVLC 1997. https://doi.org/10.1006/jvlc.1997.0037"
citing "Angelaccio, Catarci, Santucci. Query by Diagram*: a fully visual query system. JVLC 1990. https://doi.org/10.1016/S1045-926X(05)80009-6"
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Query 1

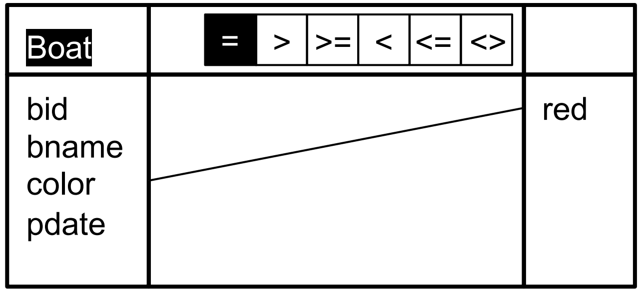
Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

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

Selection conditions are specified in separate context windows (like in visual active query builders)



Relational Algebra

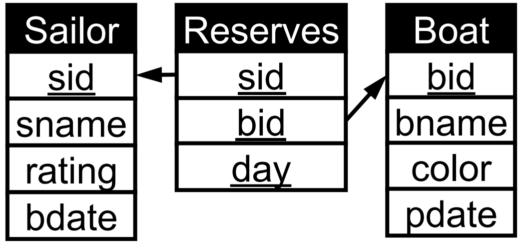
$\sigma_{\text{color='red'} \vee \text{color='blue}}\mathrm{B}$

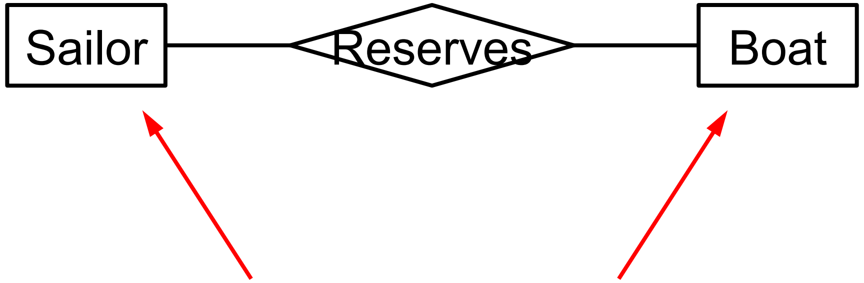
QBD (Query-By-Diagram)

Q2: "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





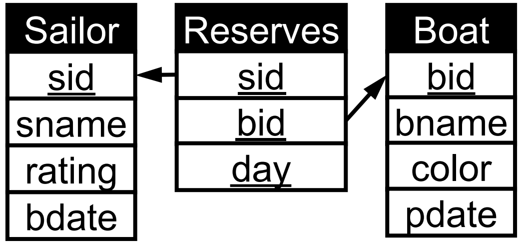
Returned attributes and selections are displayed in separate context menus (not the main panel)

Relational Algebra

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

QBD (Query-By-Diagram)

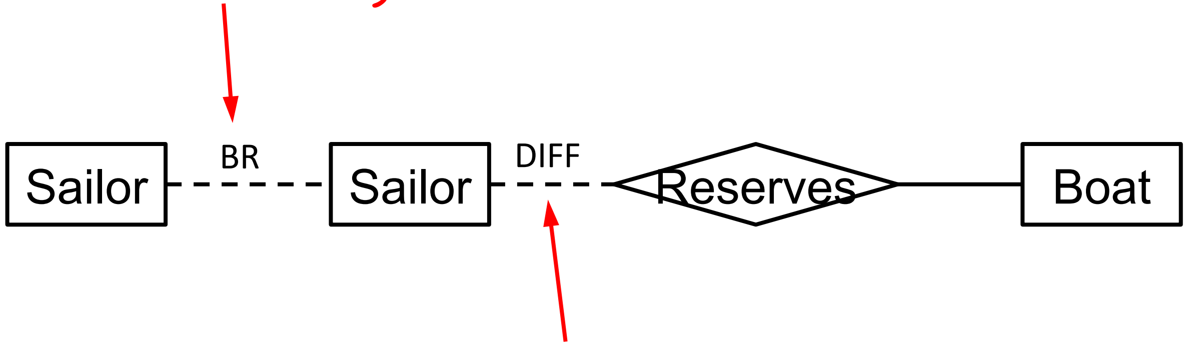
Schema



Q3: "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'))
```

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



Set difference as algebraic operator is shown by a labeled edge

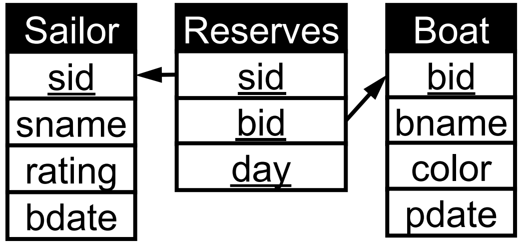
Relational Algebra

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

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 "

QBD (Query-By-Diagram)

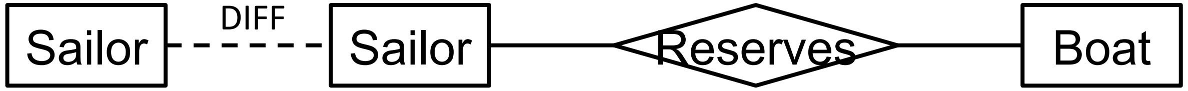
Schema



Q3: "Find sailors who reserved only red boats."

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

"except" assumes set semantics



Appropriate selections of attributes and filters for boats that are not red are done in context menus

This is a logically equivalent way of writing the same query

Relational Algebra

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

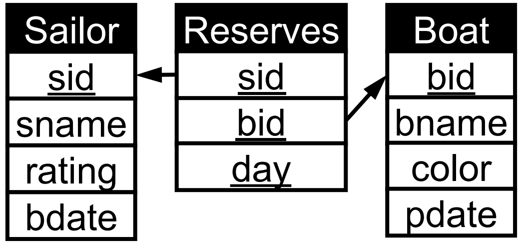
Figure drawn based on personal communication with the authors

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QBD (Query-By-Diagram)

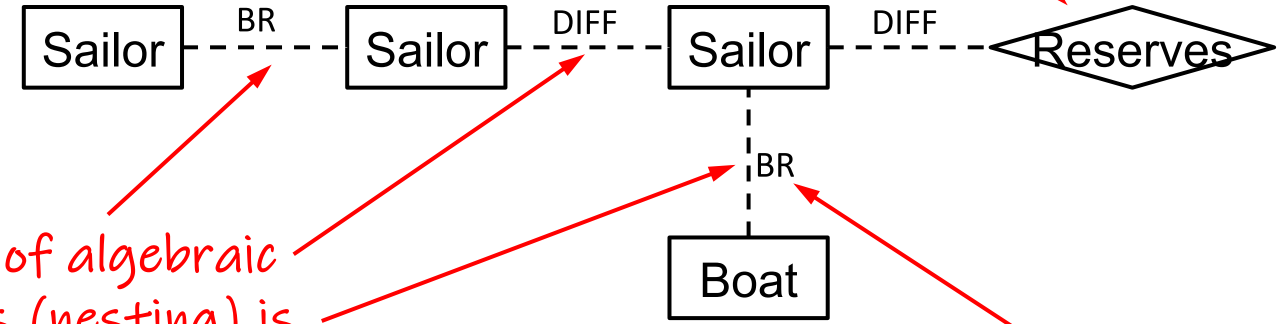
Schema



Q4: "Find sailors who reserved all red boats."

```
select distinct 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, thus "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

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

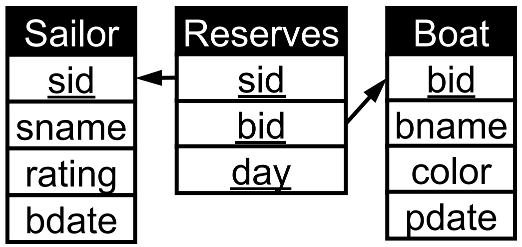
Figure drawn based on best understanding of "Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307" and "Angelaccio, Catarci, Santucci. QBD*: a graphical query language with recursion. IEEE TSE 1990. https://doi.org/10.1109/32.60295 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QBD (Query-By-Diagram)

Q4: "Find sailors who reserved all red boats."

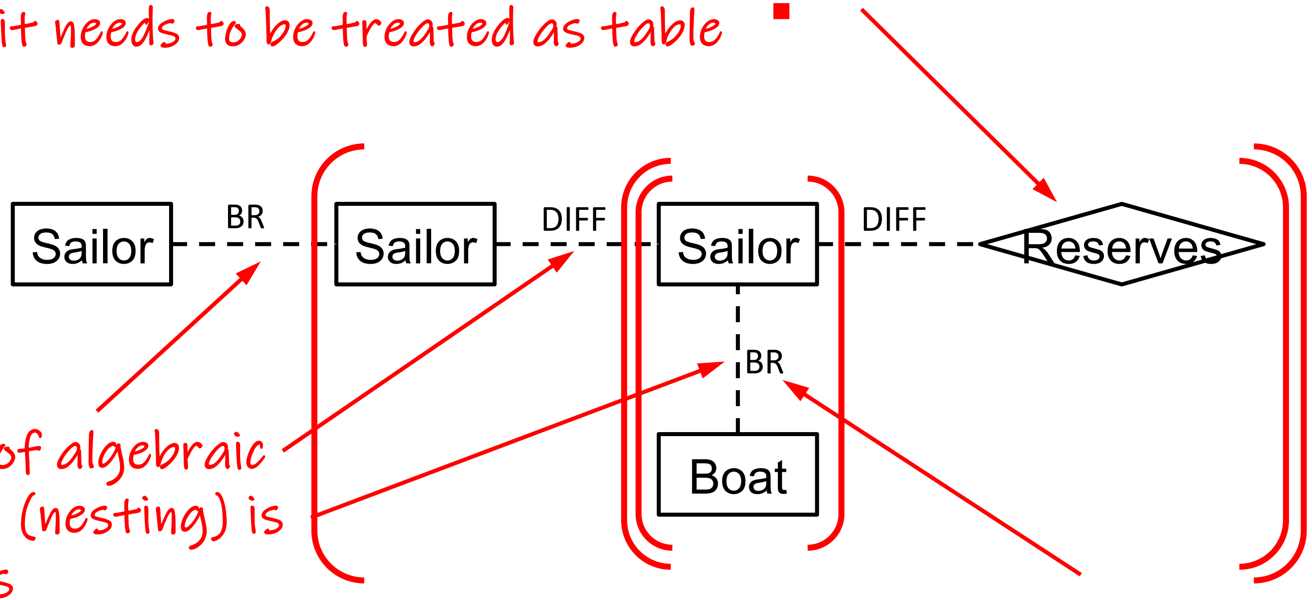
```
select distinct S.sname
from Sailor S
where not exists
    (select *
    from Boat B
    where color = 'red'
    and not exists
        (select *
        from Reserves R
        where S.sid=R.sid
        and B.bid=R.bid))
```

Schema



QBD is originally an ER model, thus "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"

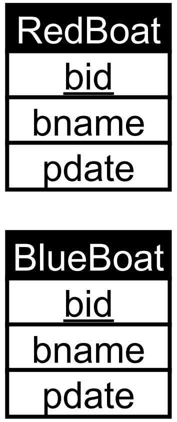
Relational Algebra

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

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

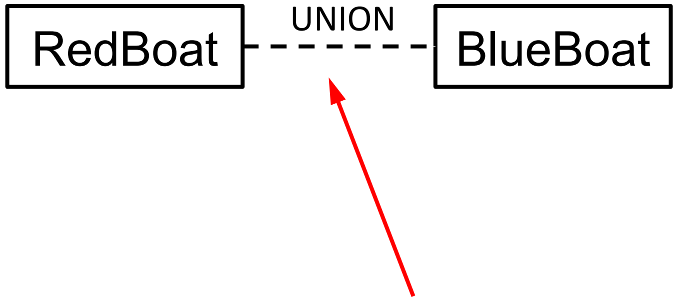
QBD (Query-By-Diagram)

Schema



Q5: "Find boats
that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```



Union as algebraic operator is shown by a label on a "bridge" that connects entities that are not connected in the original ERD

Relational Algebra

$\pi_{\text{bid, bname}}(\text{RedBoat} \cup \text{BlueBoat})$

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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.

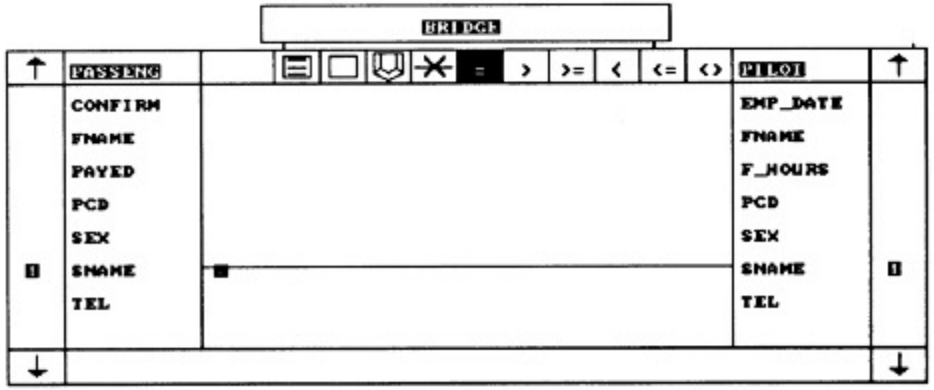


Figure 8. The windows mechanism for the 'bridge'

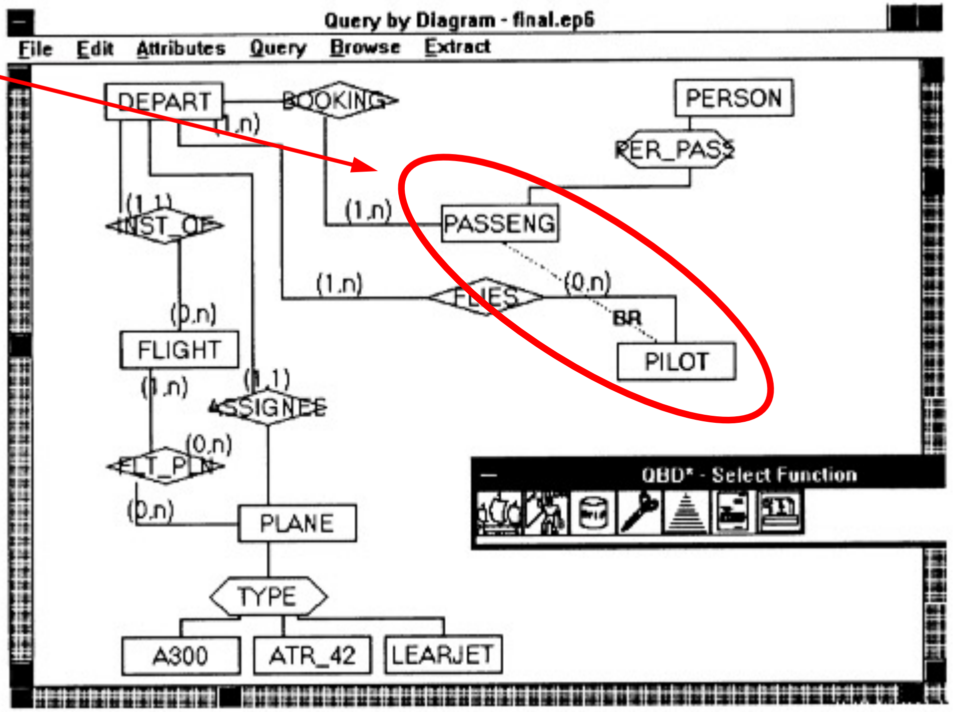


Figure 9. The new relationship

Source: Santucci, Sottile. Query by Diagram: a Visual Environment for Querying Databases. SPE 1993. https://doi.org/10.1002/spe.4380230307

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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

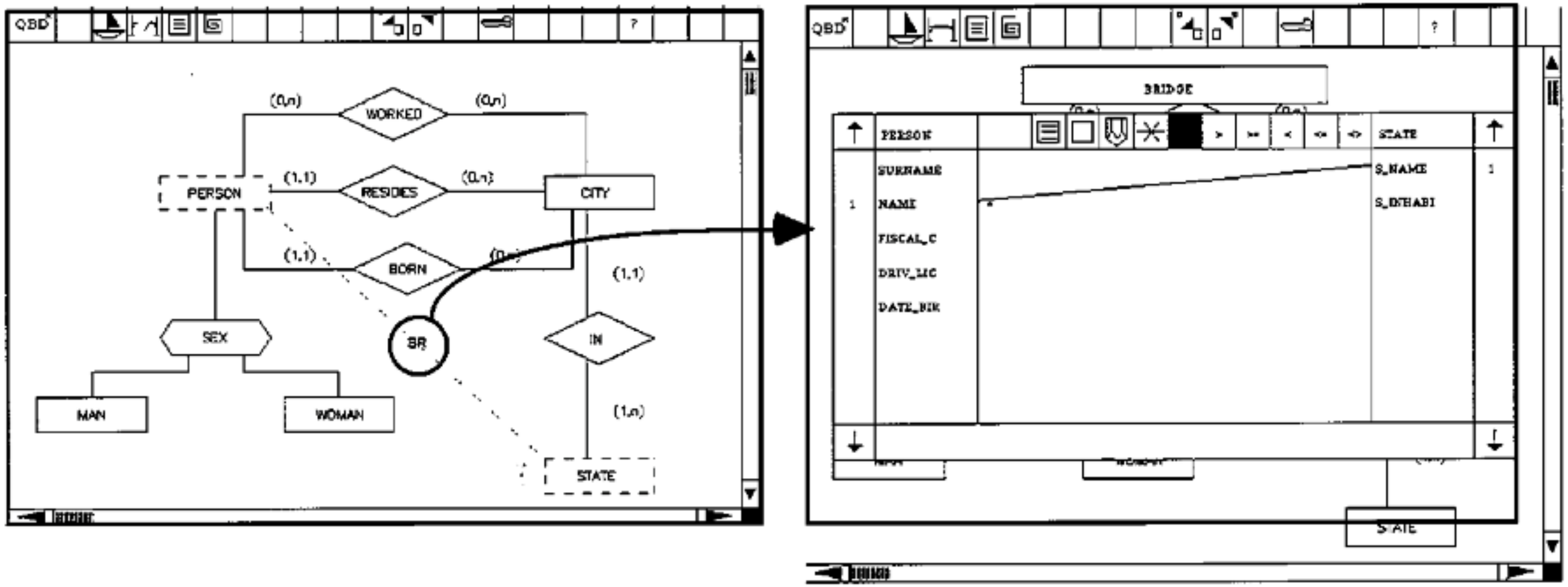


Figure 11. Unconnected path in QBD* (from Angelaccio *et al.* [4])

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"

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

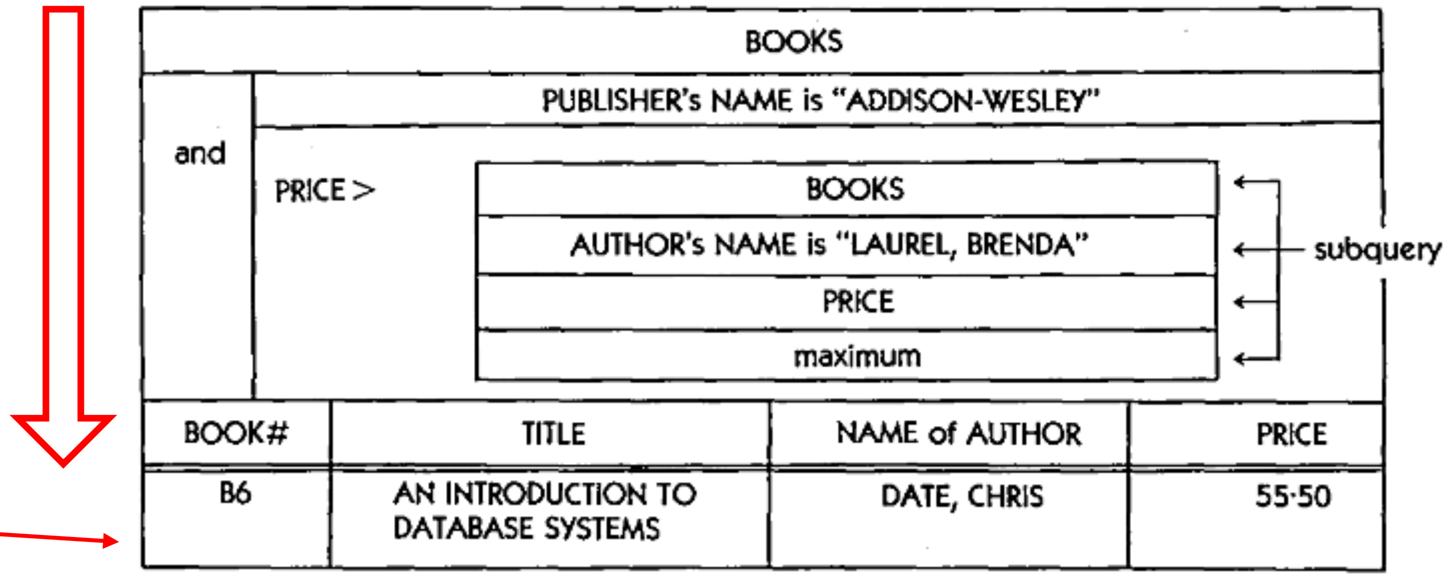
TableTalk (1991)

Sources used:

- 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



BOOK#	TITLE	NAME of AUTHOR	PRICE
B6	AN INTRODUCTION TO DATABASE SYSTEMS	DATE, CHRIS	55·50

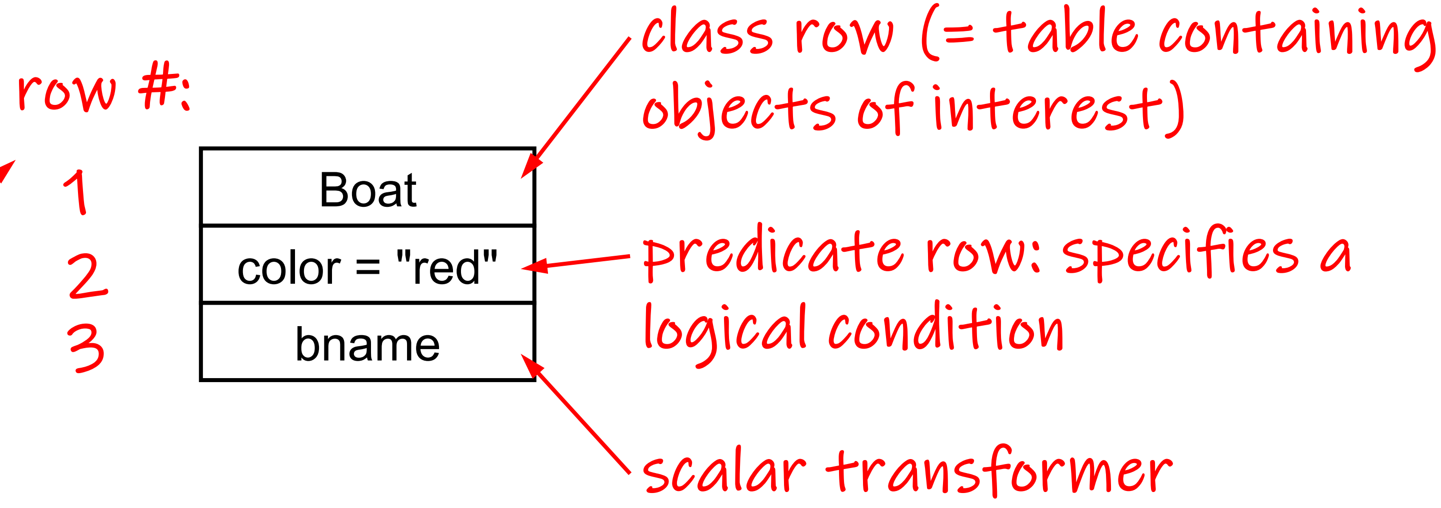
The program is "run" top-down, and displays results of the query at the bottom

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

Tabletalk

Q1b: "Find boats that are not red."

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



class row (= table containing objects of interest)

predicate row: specifies a logical condition

scalar transformer

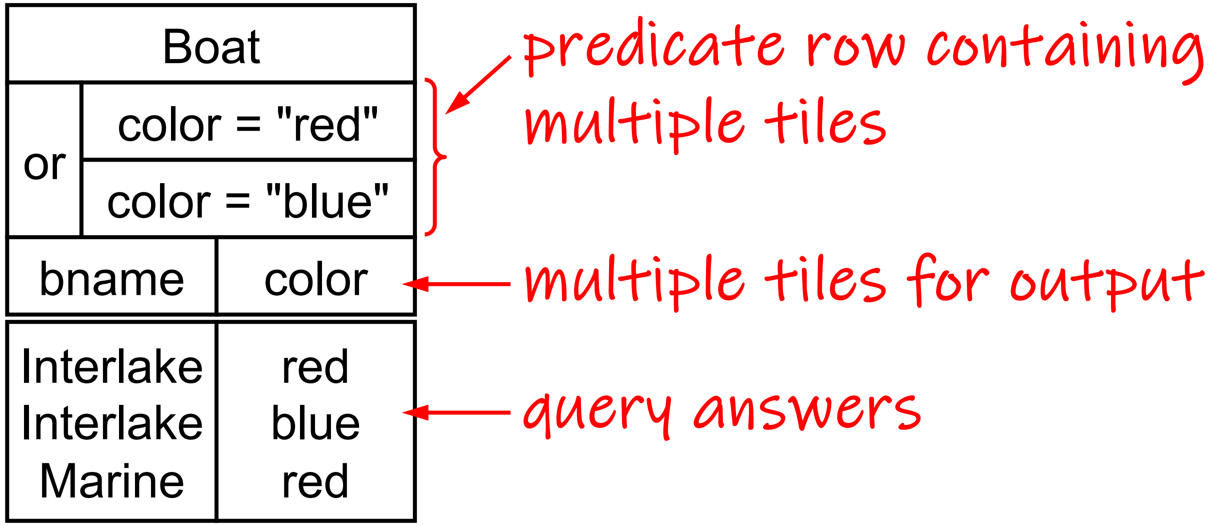
sequence processing language with rows and tiles (is read top down)

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

Tabletalk

Q1: "Find boats that are red or blue."

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



Boat

or

color = "red"

color = "blue"

bname	color
Interlake	red
Interlake	blue
Marine	red

predicate row containing multiple tiles

multiple tiles for output

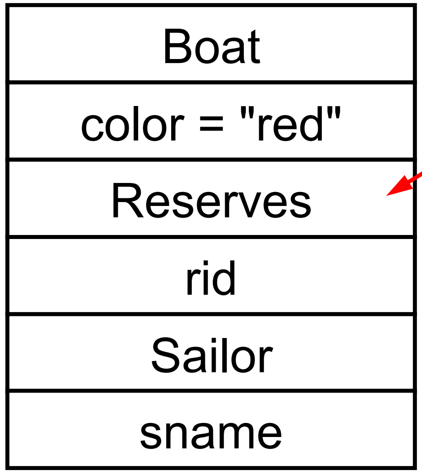
query answers

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Tabletalk

Q2: "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'
```



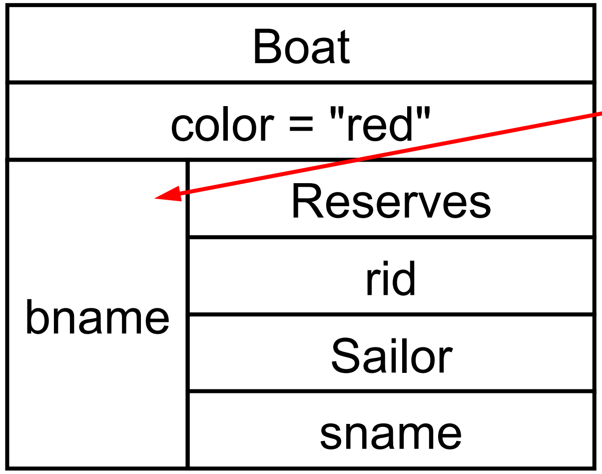
"object transformer" (assumes FK-PK constraints) "transforms the primordial objects of the main processing sequence into new primordial objects whose class is the codomain class of the object attribute."

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

Tabletalk

Q2a: "Find sailors and red boats they reserved."

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



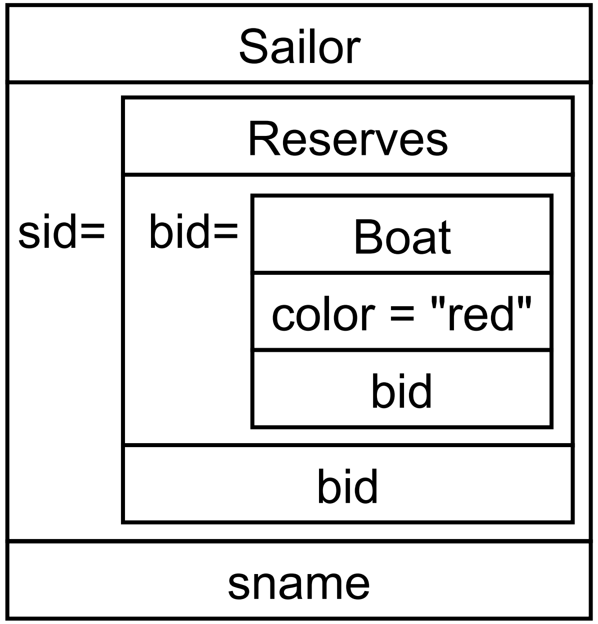
apply "product" to not "forget"

Figure drawn based on "Epstein. The TableTalk query language. JVLC, 1991. https://doi.org/10.1016/S1045-926X(05)80026-6"
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Tabletalk

Q2: "Find sailors who reserved a red boat."

```
select distinct 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

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Tabletalk

Q4: "Find sailors who reserved all red boats."

```
select distinct 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...

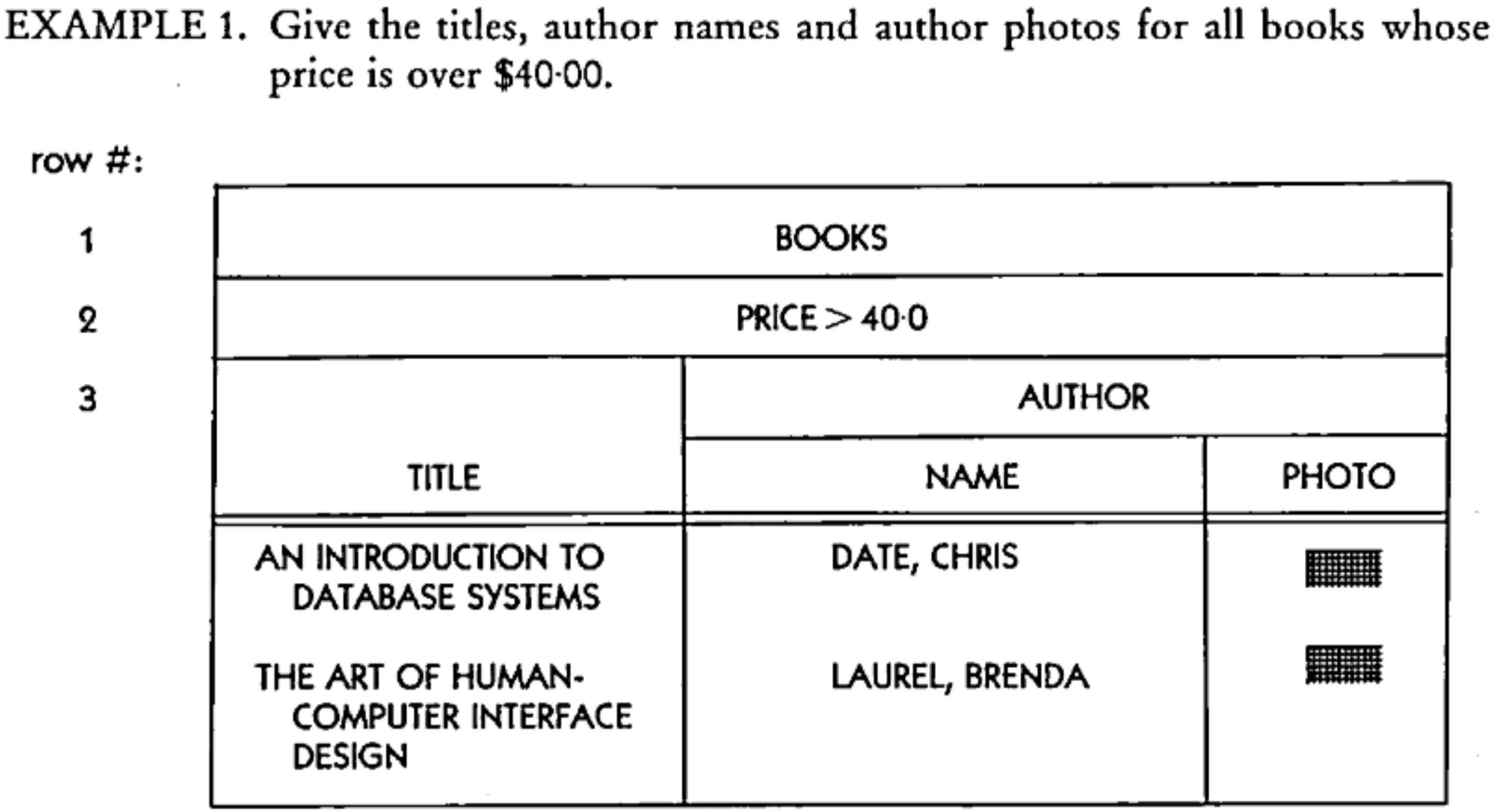
Nor does it discuss union

?

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

TableTalk (1991)
Backup

Tabletalk (1991)

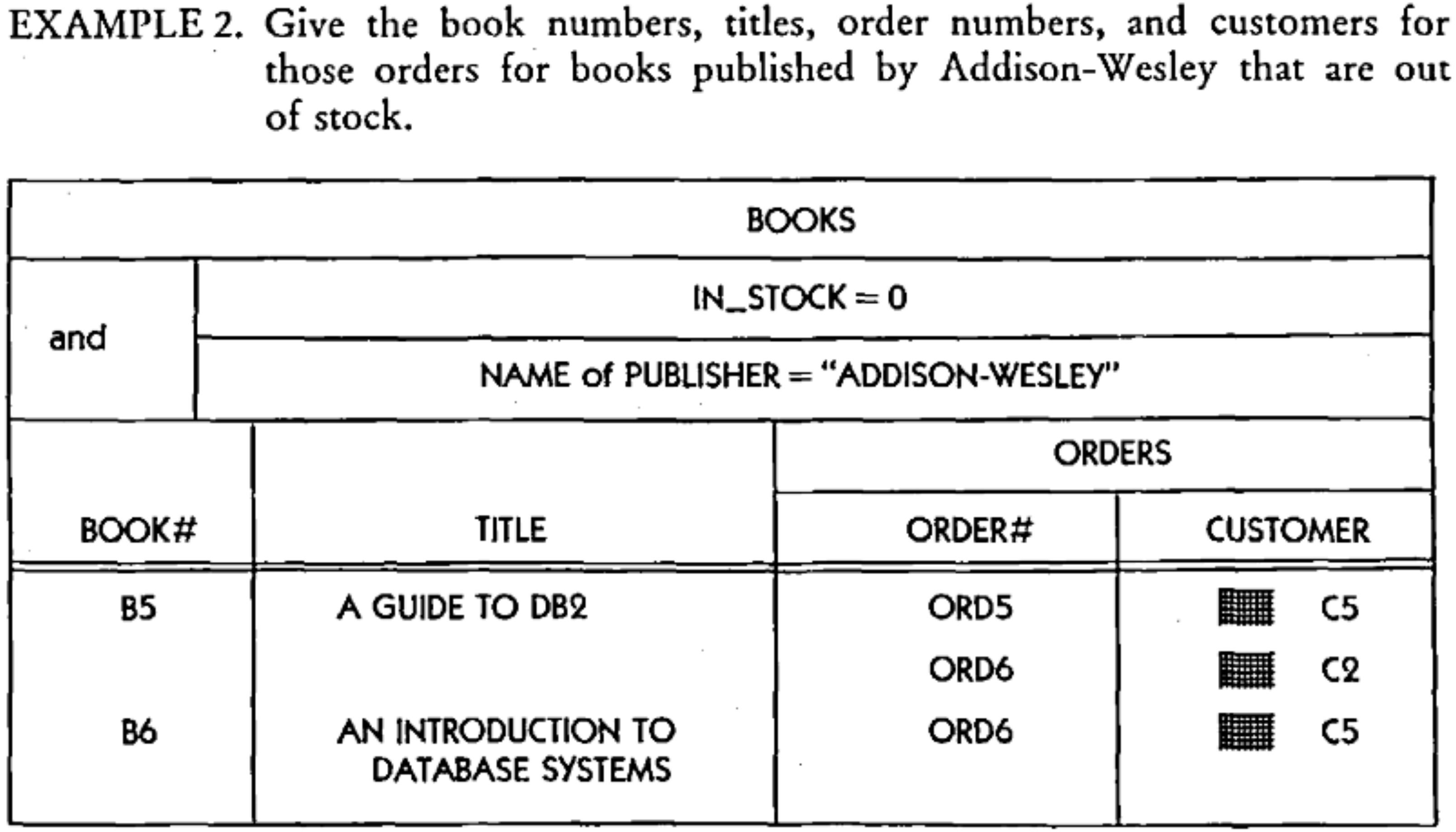


EXAMPLE 1. Give the titles, author names and author photos for all books whose price is over $40·00.

row #:			
1	BOOKS		
2	PRICE > 40·0		
3		AUTHOR	
	TITLE	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.

<table>
<tr><th colspan="4">BOOKS</th></tr>
<tr><td rowspan="2">and</td><td colspan="3">IN_STOCK = 0</td></tr>
<tr><td colspan="3">NAME of PUBLISHER = “ADDISON-WESLEY”</td></tr>
<tr><th rowspan="2">BOOK#</th><th rowspan="2">TITLE</th><th colspan="2">ORDERS</th></tr>
<tr><th>ORDER#</th><th>CUSTOMER</th></tr>
<tr><td>B5</td><td>A GUIDE TO DB2</td><td>ORD5</td><td>▦ C5</td></tr>
<tr><td></td><td></td><td>ORD6</td><td>▦ C2</td></tr>
<tr><td>B6</td><td>AN INTRODUCTION TO DATABASE SYSTEMS</td><td>ORD6</td><td>▦ C5</td></tr>
</table>

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

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

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

Sources used:

- 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"

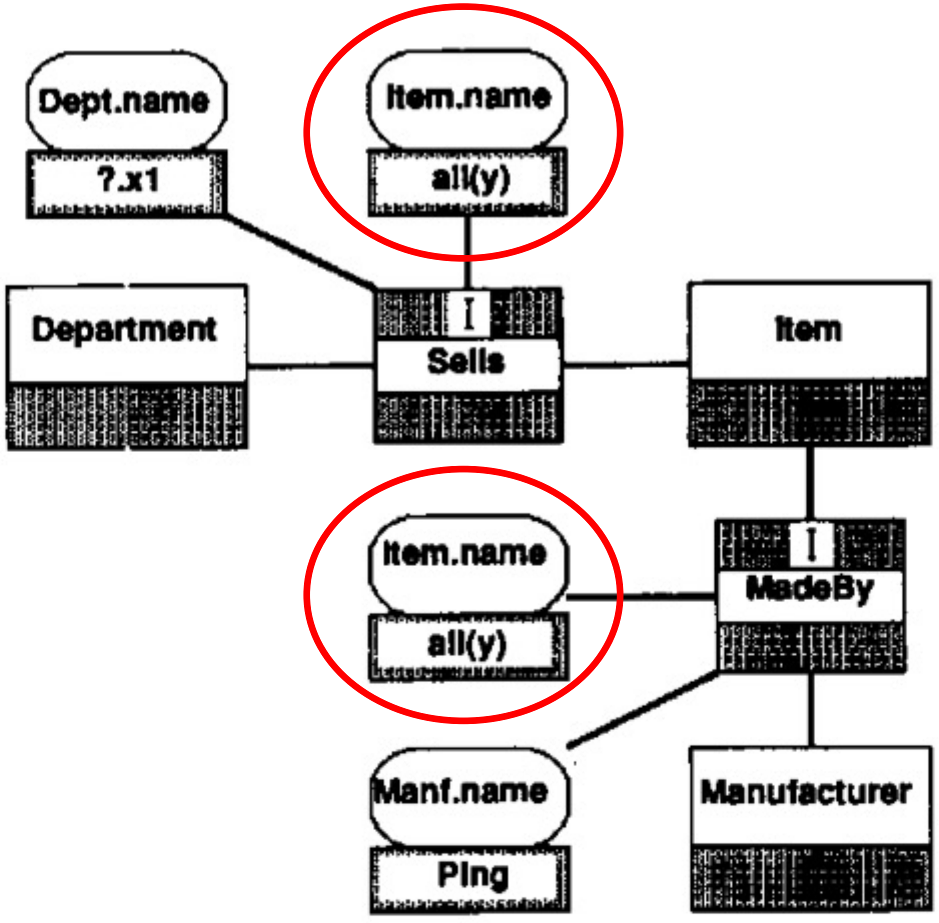


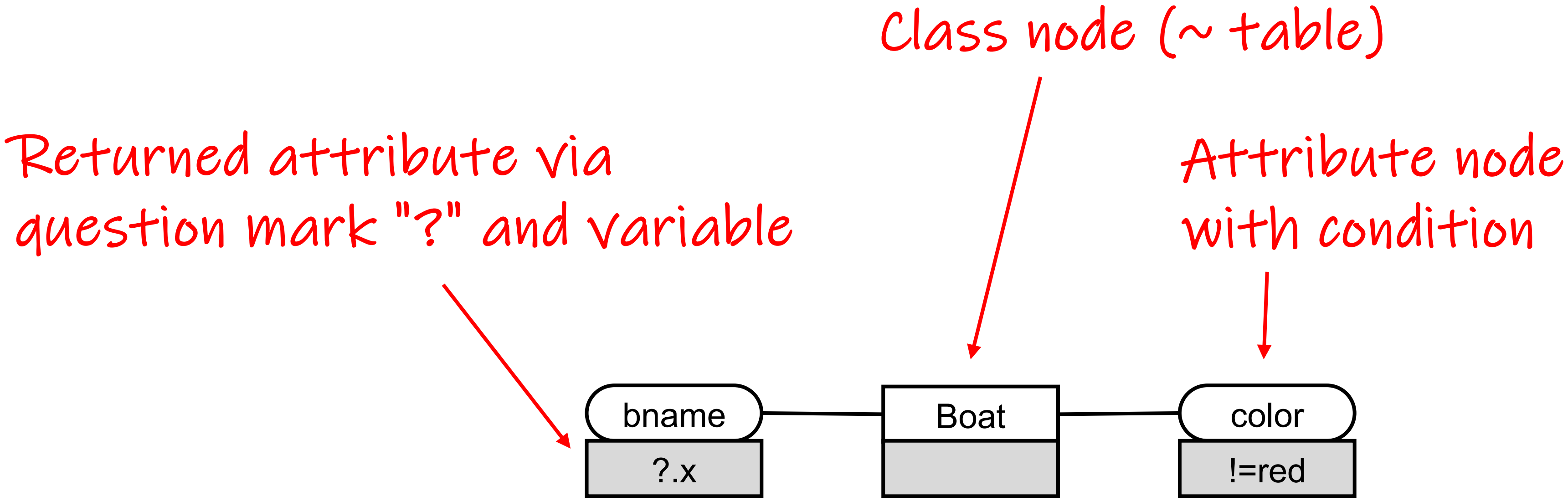
Figure source: Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q1b: "Find boats that are not red."

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



Class node (~ table)

Returned attribute via question mark "?" and variable

Attribute node with condition

DRC (Domain Relational Calculus)

$\{(x) \mid \text{Sailor}(\_,x,y,\_) \wedge (y!=\text{'red'})\}$

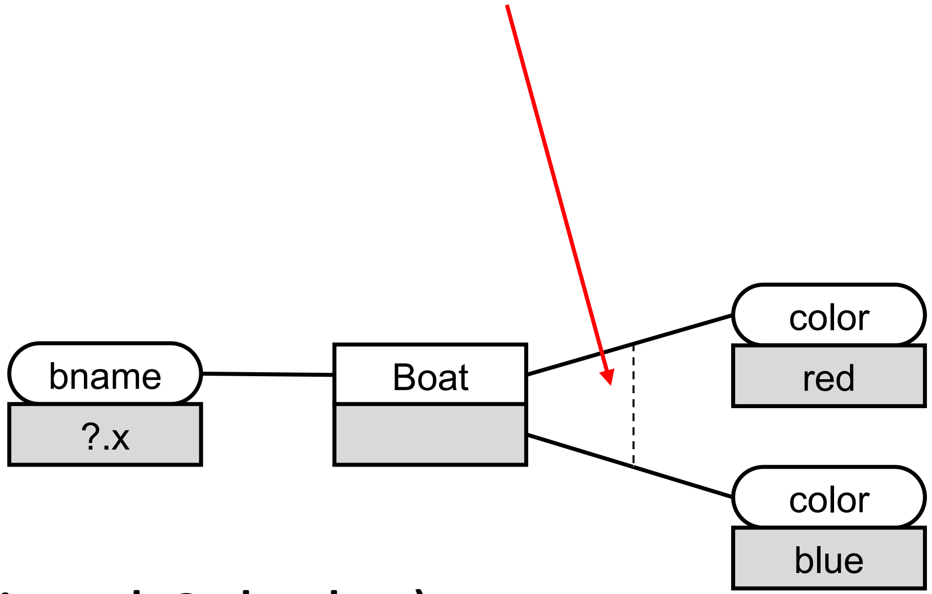
Figure drawn based on "Mohan, Kashyap. A visual query language for graphical interaction with schema-intensive databases. TKDE 1993. https://doi.org/10.1109/69.243513 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q1: "Find boats
that are red or blue."

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

Disjunction



DRC (Domain Relational Calculus)

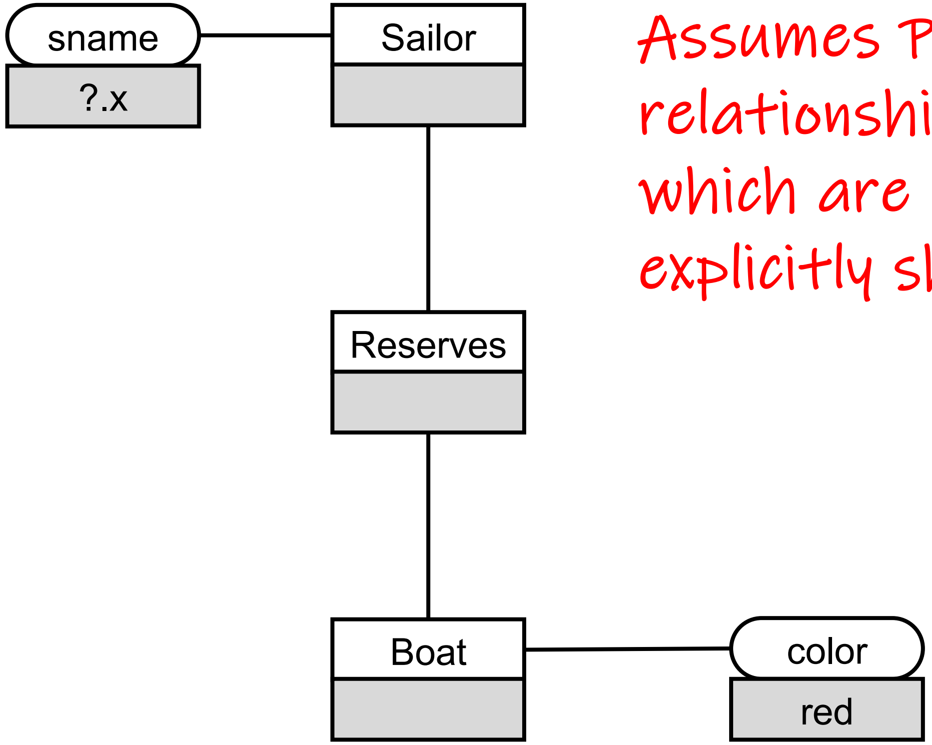
$$\{(x) \mid \text{Sailor}(\_,x,y,\_) \wedge (y=\text{'red'} \vee y=\text{'red'})\}$$

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q2: "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'
```



Assumes PK-FK relationships, which are not explicitly shown

DRC (Domain Relational Calculus)

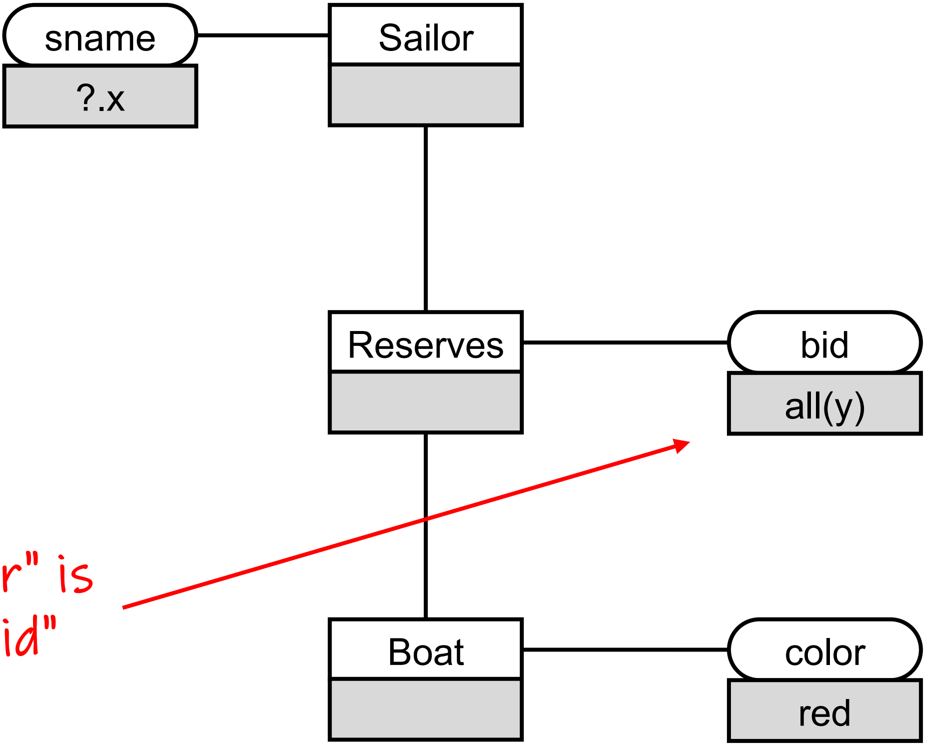
{(x) | ∃v[Sailor(v,x,_,_) ∧
∃y[Reserves(v,y,_) ∧
Boat(y,_,'red',_)]]}

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

Q3: "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'))
```



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

DRC (Domain Relational Calculus)

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

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 "

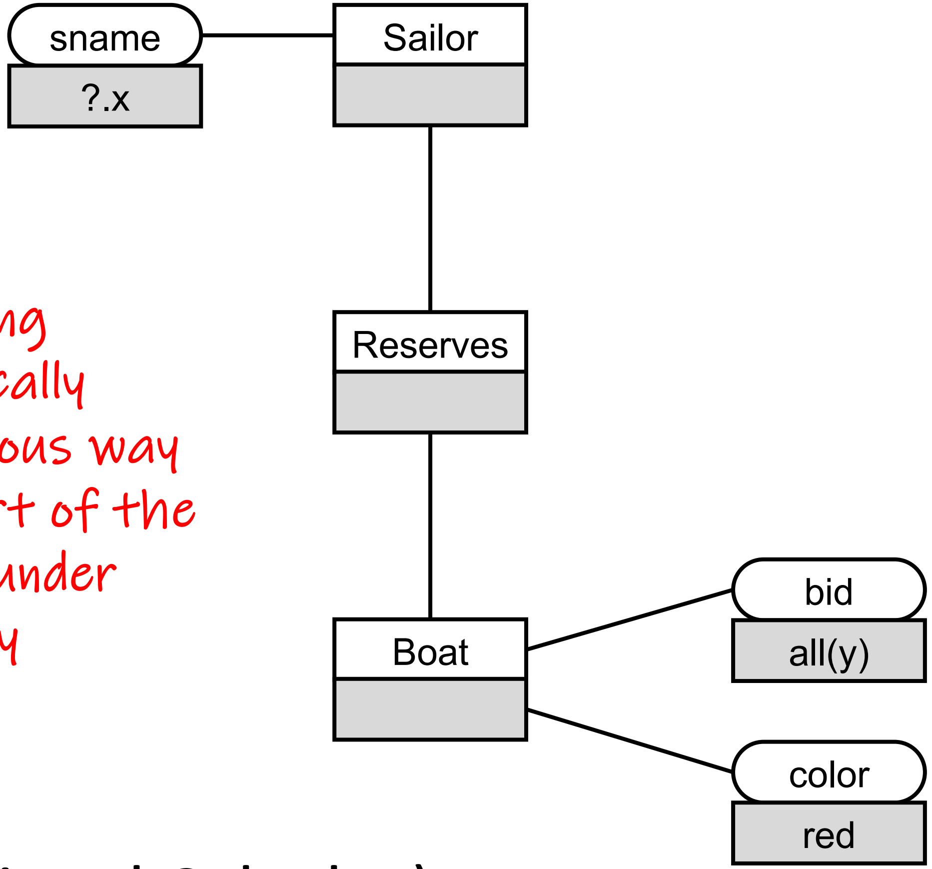
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

?

Q4: "Find sailors who reserved all red boats."

```
select distinct 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)

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

No way to express union!

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

Schema

RedBoat
bid
bname
pdate

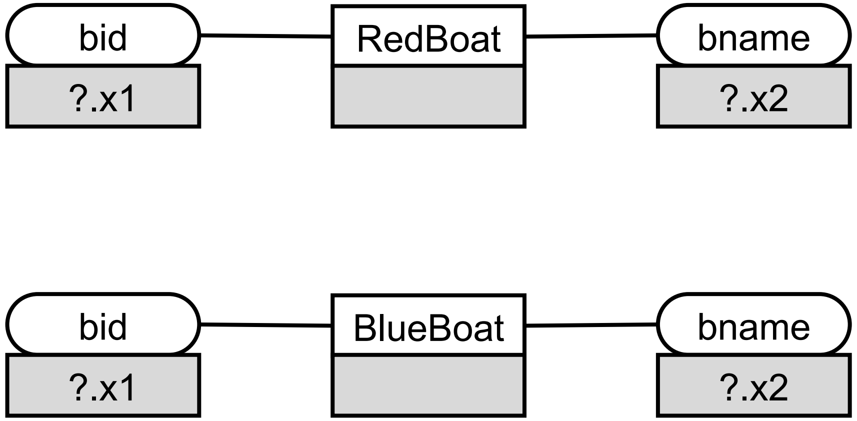
BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

Paper does not discuss any way to handle union.
But it is easy to imagine a Datalog-style "repeated derivation" presentation (?)

?



Datalog

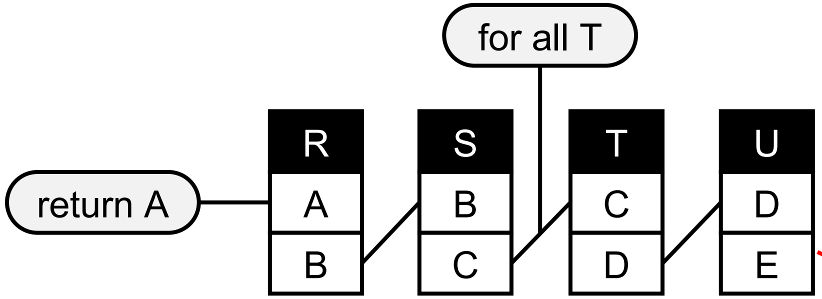
Q(x,y) :- RedBoat(x,y,_)
Q(x,y) :- BlueBoat(x,y,_)

DRC (Domain Relational Calculus)

$\{(x,y) \mid \exists z\, [\text{RedBoat}(x,y,z) \vee \text{BlueBoat}(x,y,z)]\}$

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

When the quantifier scope becomes ambiguous



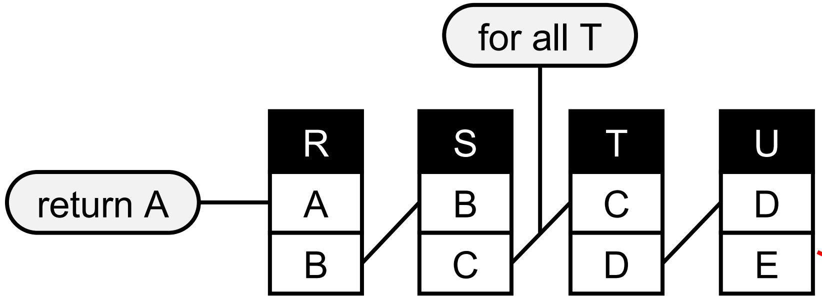
This figure allows
2 interpretations

interpretation 1

interpretation 2

?

When the quantifier scope becomes ambiguous



This figure allows
2 interpretations

interpretation 1

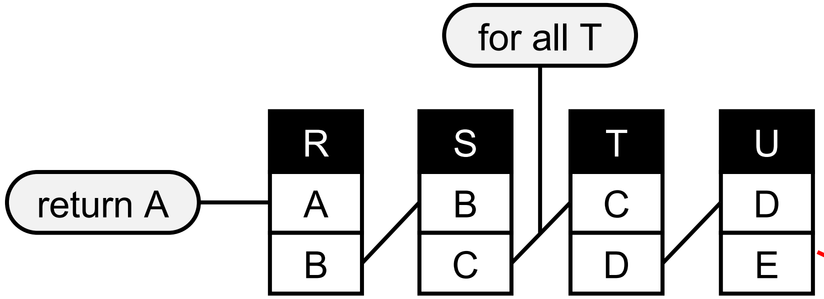
$\{(a) \mid \exists b[R(a,b)$
$(\forall c[S(b,c) \rightarrow$
$(\exists d,e[T(c,d) \wedge U(d,e)])])]\}$

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))
```

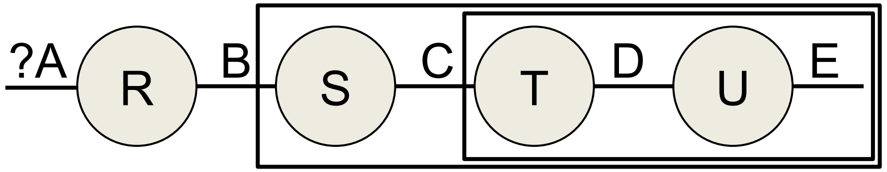
$$\{(a) \mid \exists b[R(a,b)\ (\forall c[S(b,c) \rightarrow (\exists d,e[T(c,d) \wedge U(d,e)])])]\}$$

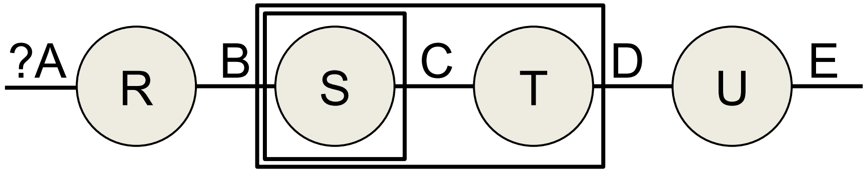
$$\{q(A) \mid \exists r \in R[q.A=r.A \wedge \neg(\exists s \in S[r.B=s.B \wedge \neg(\exists t \in T, \exists u \in U[s.C=t.C \wedge t.D=u.D])])]\}$$

```
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))
```

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

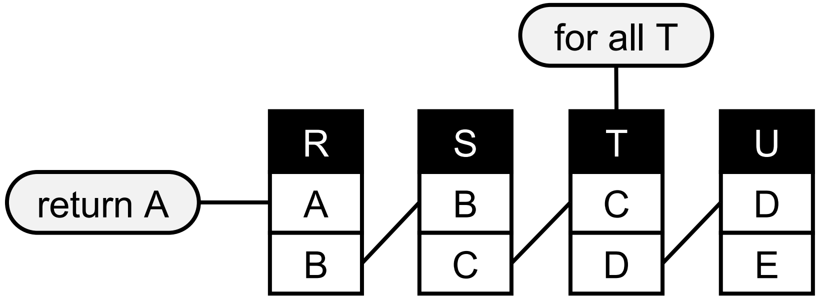
$$\{q(A) \mid \exists r \in R, u \in U[q.A=r.A\ \neg(\exists t \in T[t.D=u.D \wedge \neg(\exists s \in S[r.B=s.B \wedge t.D=u.D])])]\}$$





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

When the quantifier scope becomes ambiguous



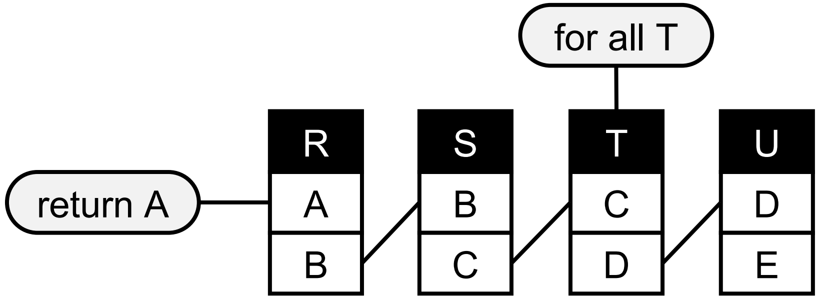
This figure also allows
2 interpretations

interpretation 3

interpretation 2

?

When the quantifier scope becomes ambiguous



This figure also allows
2 interpretations

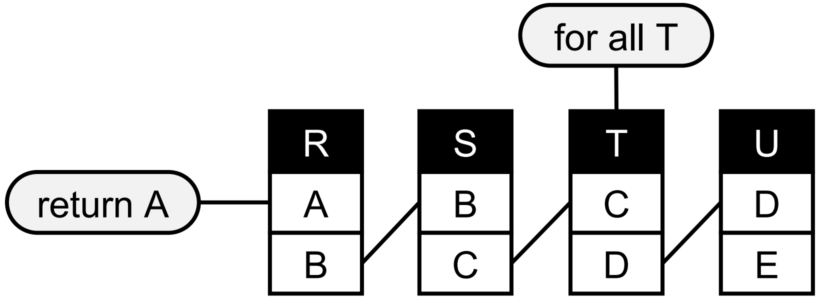
interpretation 3

interpretation 2

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

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

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))
```

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

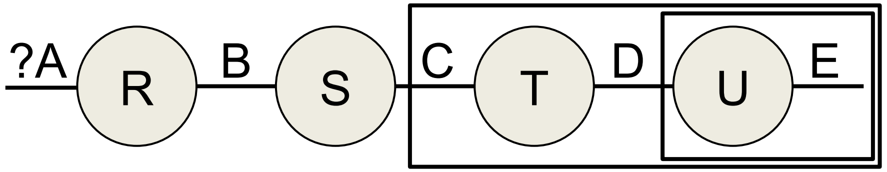
$$\{q(A) \mid \exists r \in R, s \in S[q.A=r.A \wedge r.B=s.B \wedge \neg(\exists t \in T[s.C=t.C \wedge \neg(\exists u \in U[t.D=u.D])])]\}$$

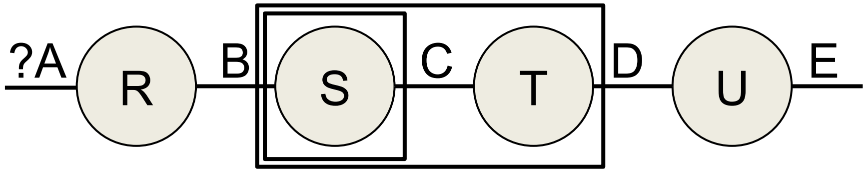
interpretation 2

```
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))
```

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

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

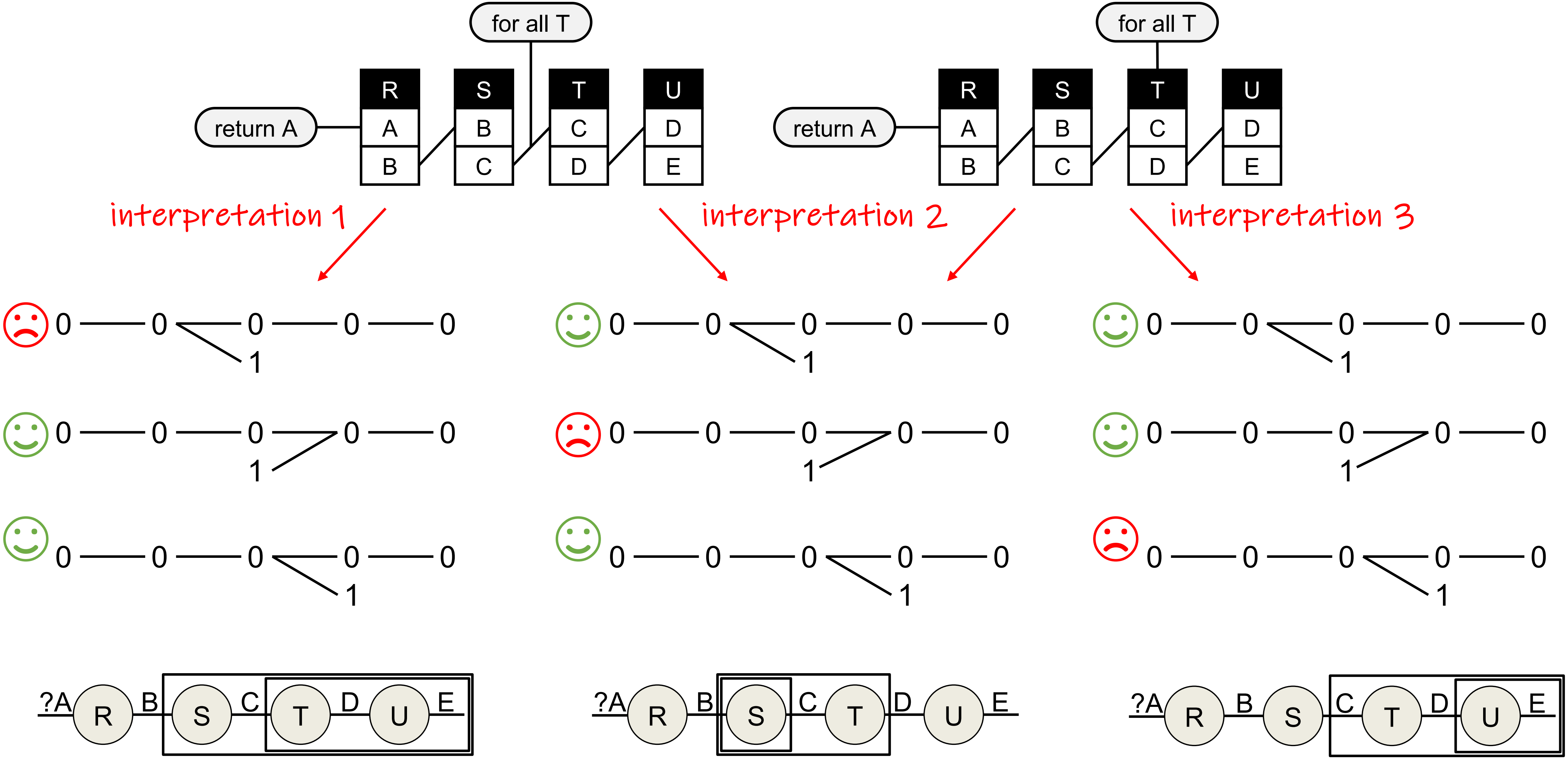




See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

When the quantifier scope becomes ambiguous



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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

"Object-oriented VQL" (1993) Backup

OO-VQL (Object-oriented VQL)

InletNeedle (idNumber, diameter, length, weight)

Q2: Find the idNumbers of all the InletNeedles that have diameter greater than 0.25 and have length greater than 12.5.

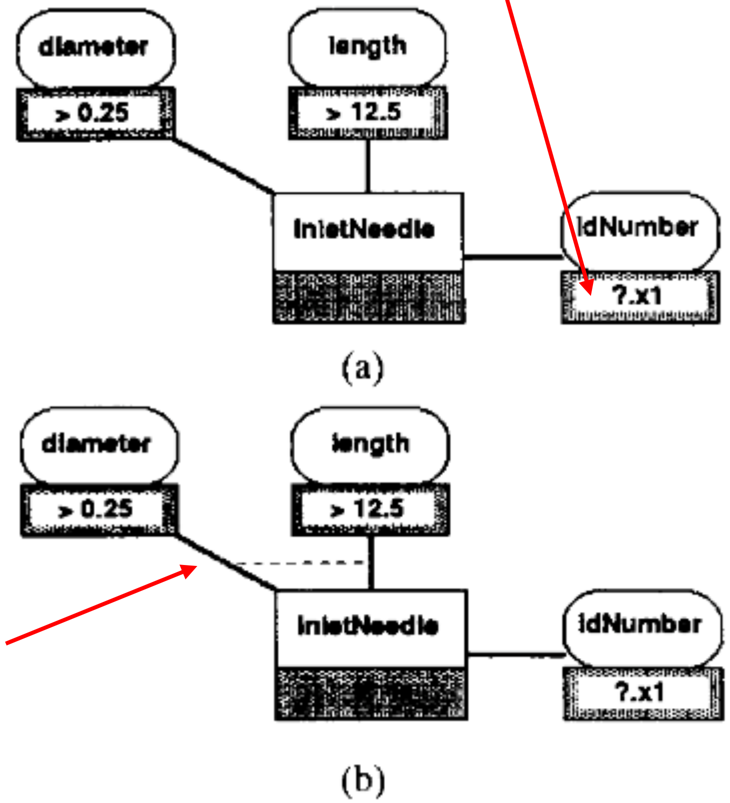
$x1$: (InletNeedle, ($x1$, > 0.25, > 12.5, _))

Q3: Find the idNumbers of all the InletNeedles that have either diameter greater than 0.25 or have length greater than 12.5.

$x1$: (InletNeedle, ($x1$, > 0.25, _, _))

$\vee$(InletNeedle, ($x1$, _, 12.5, _))

Non-graphical variables for returned predicates



Disjunction

Fig. 6. Conjunctive and disjunctive queries.

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)

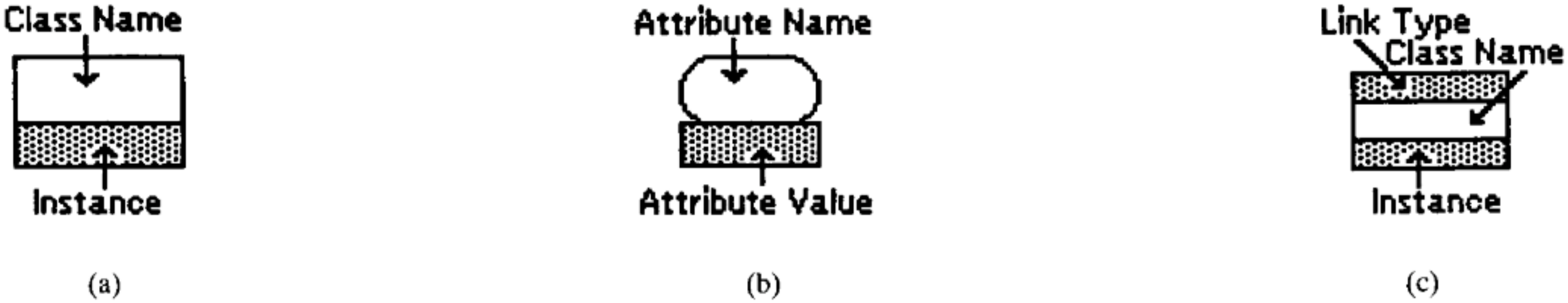


Fig. 3. Visual query primitives.

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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.

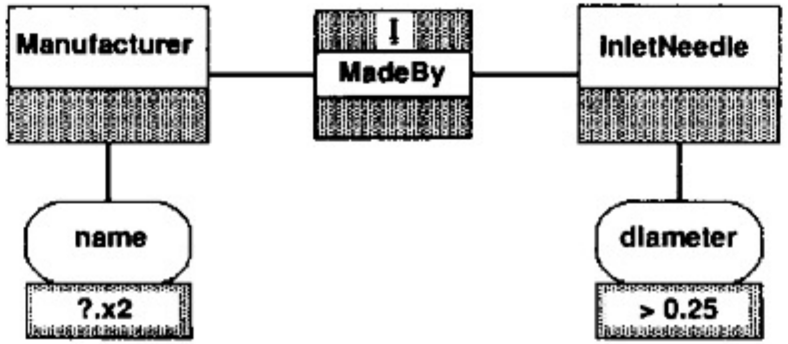


Fig. 7. An associative "Join" query using the "Interaction" link.

$x2$: (I, MadeBy, ($x1$, $x2$))
[(InletNeedle, ($x1$, $>$ 0.25, _, _)),
(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]

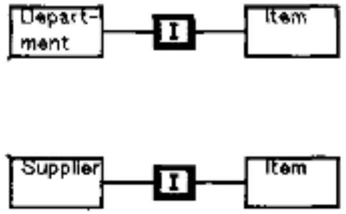
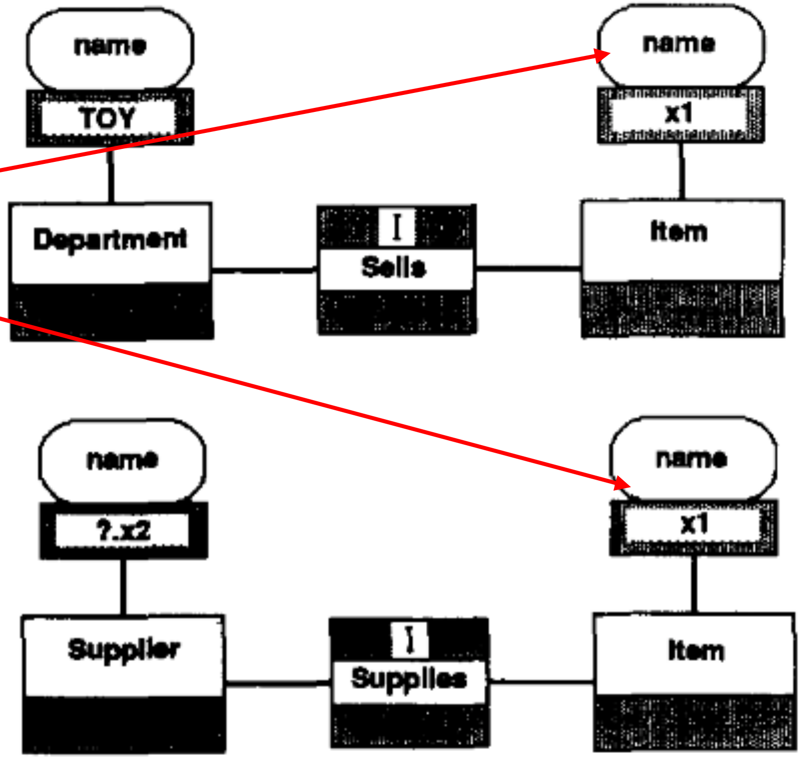


Fig. 8. Portions of two separate schemas.

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.

$$
\begin{aligned}
x2\text{: } & (\text{I},\ \text{Sells},\ (\_,\ x1))[(\text{Department},\\
& (\text{Toy},\ \_,\ \_)),\ (\text{Item},\ \_)]\\
& \vee (\text{I},\ \text{Supplies},\ (\_,\ x1))[(\text{Supplier},\\
& (x2,\ \_,\ \_,\ \_)),\ (Item,\ \_)]
\end{aligned}
$$

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
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

OO-VQL (Object-oriented VQL)

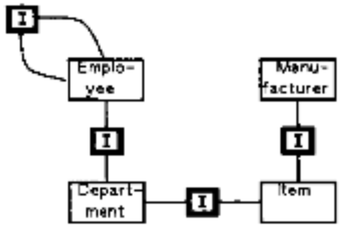
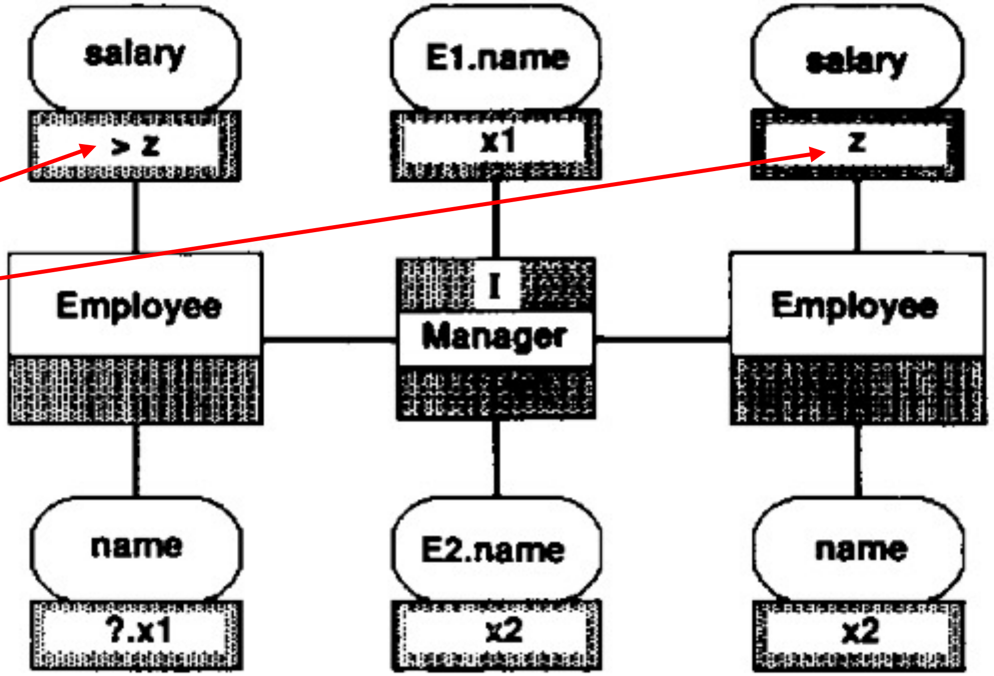


Fig. 10. A department store schema as represented in SSONET.

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.

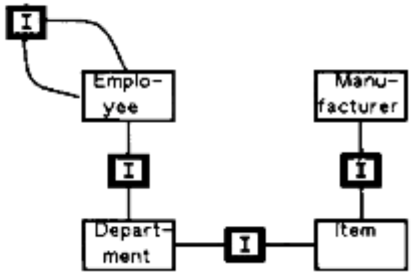
$$x1\text{: (I, ManagedBy, }(x1,\ x2))$$
$$[(\text{Employee},\ (x1,\ >z,\ \_,\ \_)),\ (\text{Employee},\ (x2,\ z,\ \_,\ \_))]$$

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]
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:

$$\{(x) \mid \text{Department}(x,\_,\_) \wedge (\forall v[\text{MadeBy}(v, \text{'Ping'}) \rightarrow (\text{Sells}(x,v)))\}$$

$$x1\text{: } \forall y(\text{I}, \textbf{MadeBy}, (y, \text{Ping}))[(\text{Item}, \_), (\text{Manufacturer}, \_)] \rightarrow (\text{I}, \textbf{Sells}, (x1, y))[(\text{Department}, \_), (\text{Item}, \_)]$$

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"

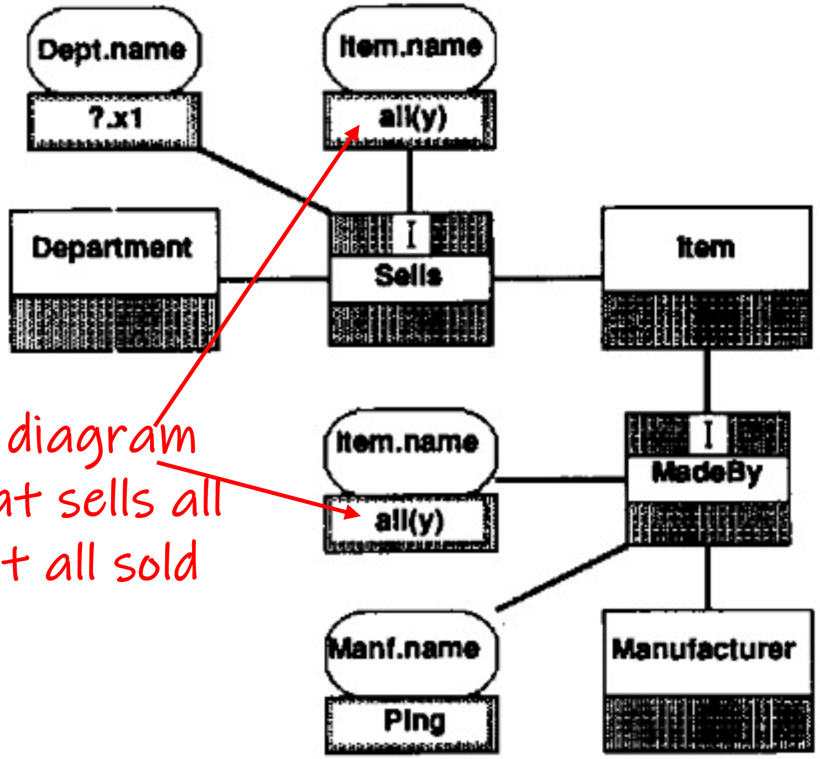


Fig. 12. A query that uses the "all" operator.

Figures from: 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

DFQL (1994)
DataFlow QL

Sources used:

- 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 mapping its visual symbols to the operators of RA
- Actually many Visual Query Languages (VQLs) become "visual" by introducing visual representations of RA operators in a dataflow

- Here, nodes represent the operations, 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

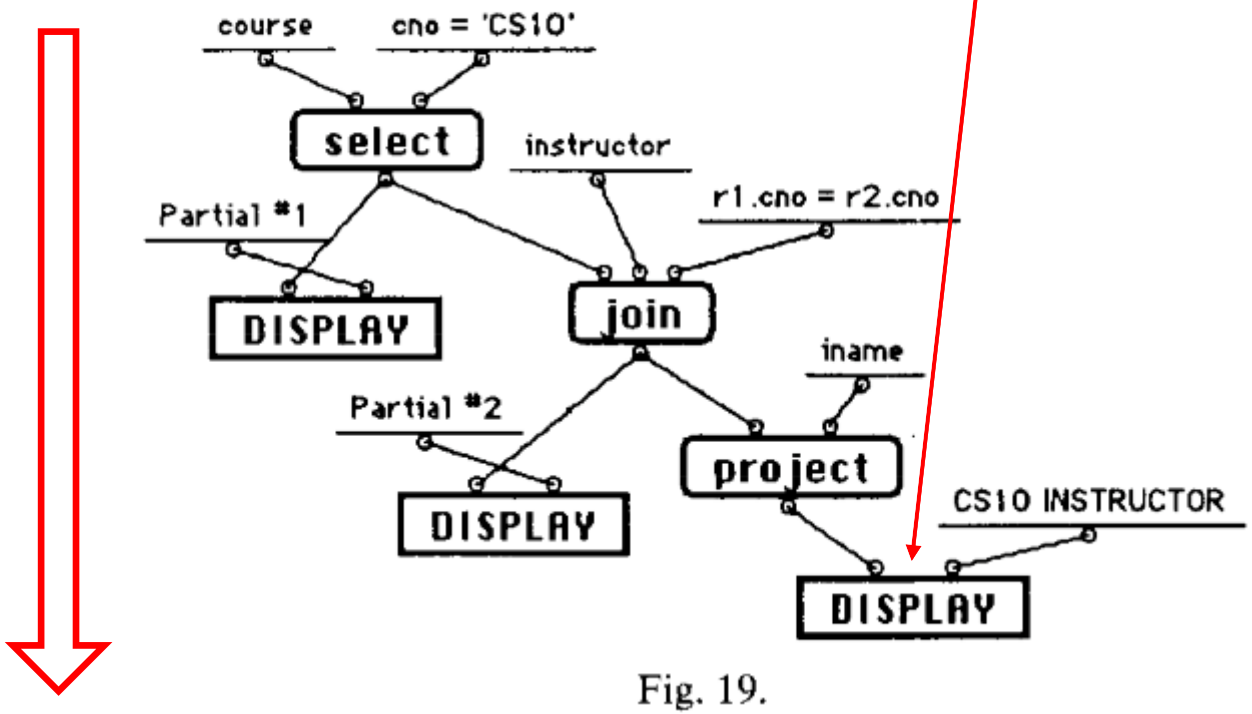


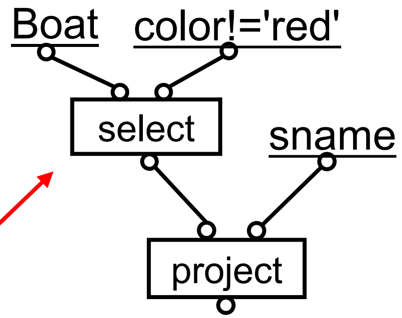
Fig. 19.

Figure source: "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

Q1b: "Find boats that are not red."

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



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

Relational Algebra

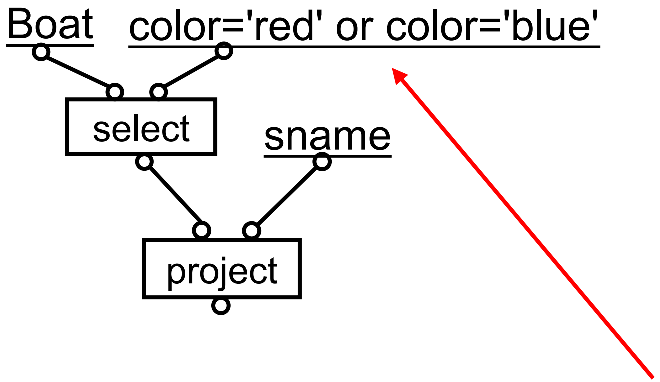
$$\pi_{\text{sname}}(\sigma_{\text{color=!'red'}}\text{B})$$

Figure drawn based on "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

Q1: "Find boats that are red or blue."

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



conditions are in text

Relational Algebra

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

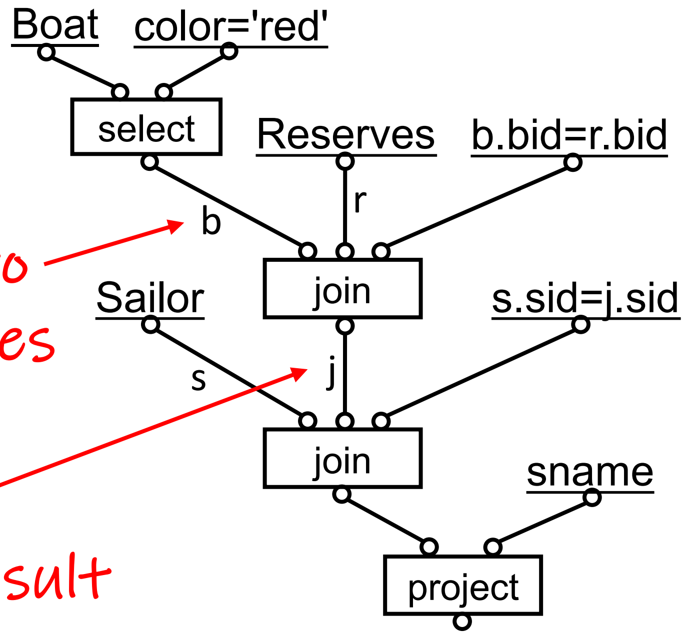
Figure drawn based on "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

Q2: "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'
```



alias on edges to avoid ambiguities

new alias for intermediate result

Relational Algebra

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

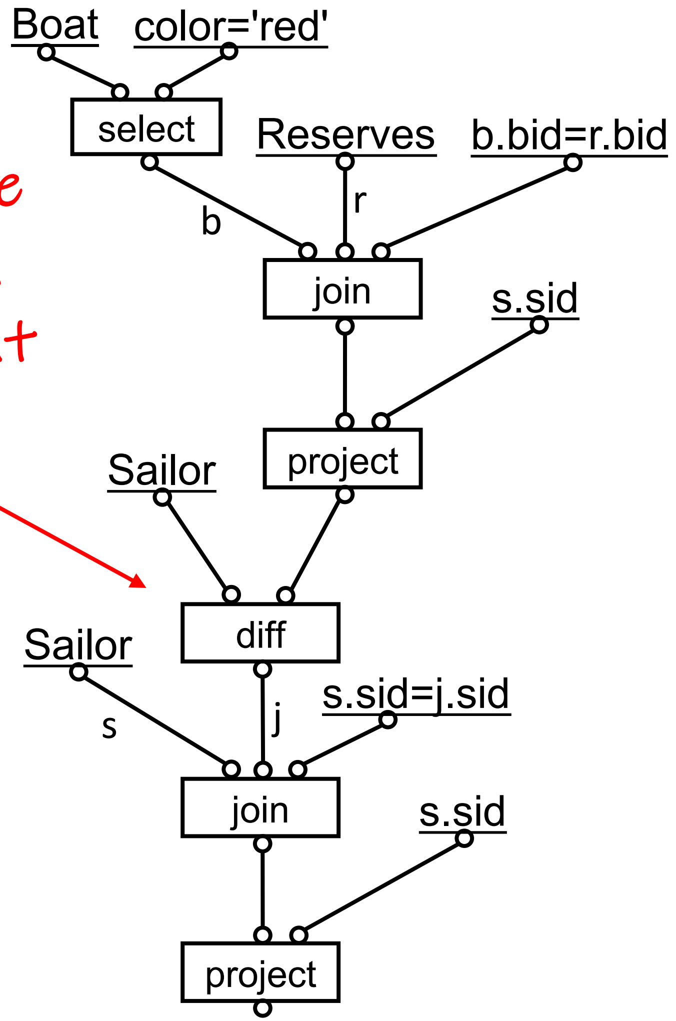
Figure drawn based on "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

Q3: "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'))
```

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



Relational Algebra

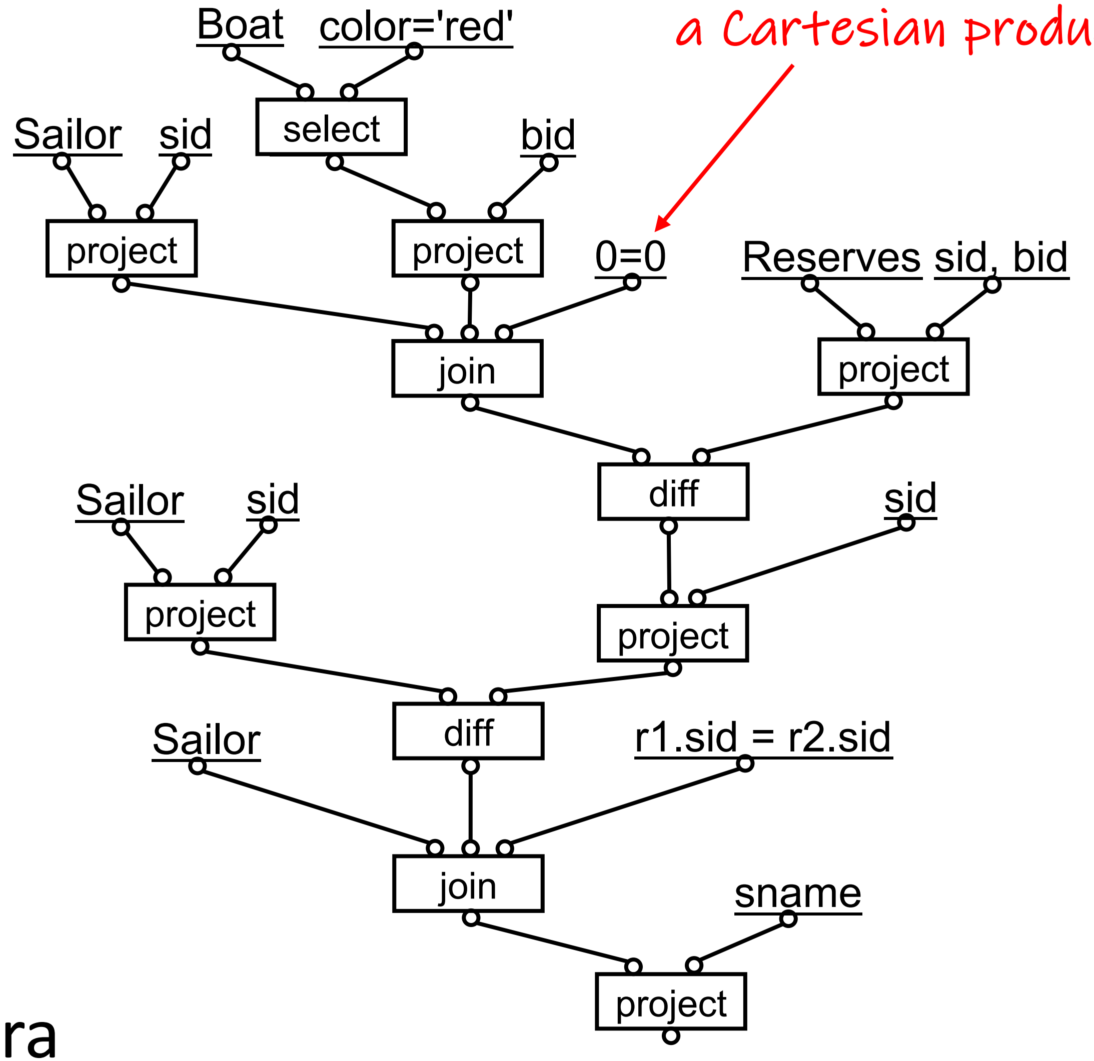
$$\pi_{\text{sname}}(S\bowtie(\pi_{\text{sid}}S-(\pi_{\text{sid}}(R\bowtie\sigma_{\text{color='red'}}B))))$$

Figure drawn based on "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

Q4: "Find sailors who reserved all red boats."

```
select distinct 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 "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

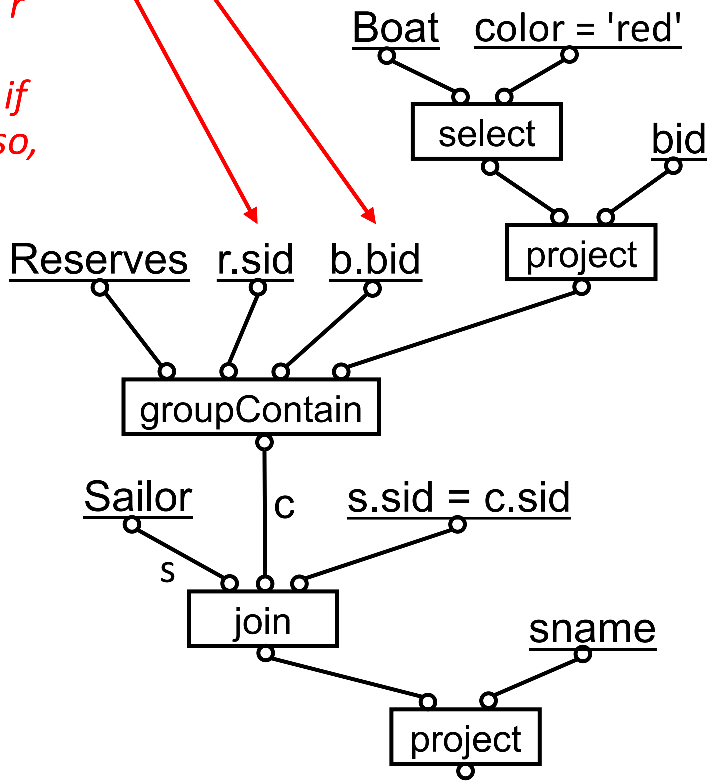
Q4: "Find sailors who reserved all red boats."

```
select distinct 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

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

Figure drawn based on "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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

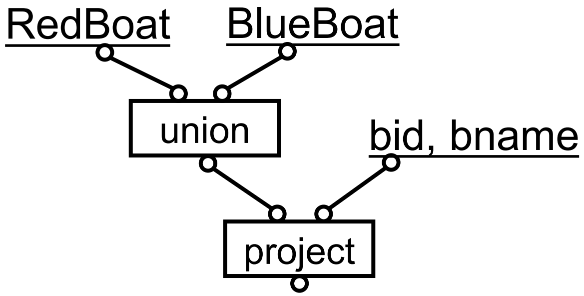
Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```



Relational Algebra

$$\pi_{\text{bid, bname}}(\text{RedBoat} \cup \text{BlueBoat})$$

Figure drawn based on "Clark, Wu. DFQL: Dataflow query language for relational databases. Information& Management, 1994. https://doi.org/10.1016/0378-7206(94)90098-1 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (1994)
DataFlow QL
BACKUP

DFQL (DataFlow Query Language)

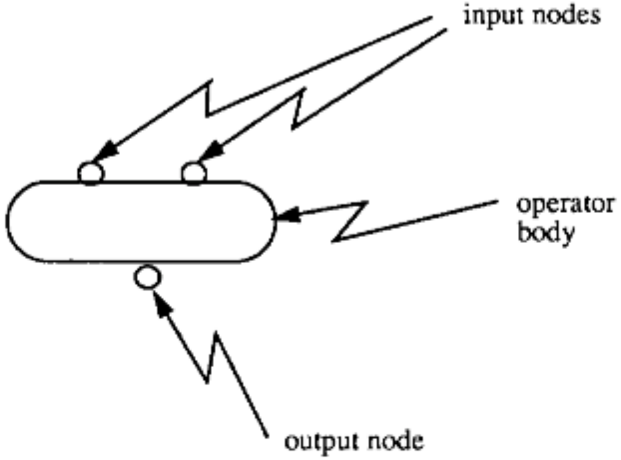


Fig. 3.

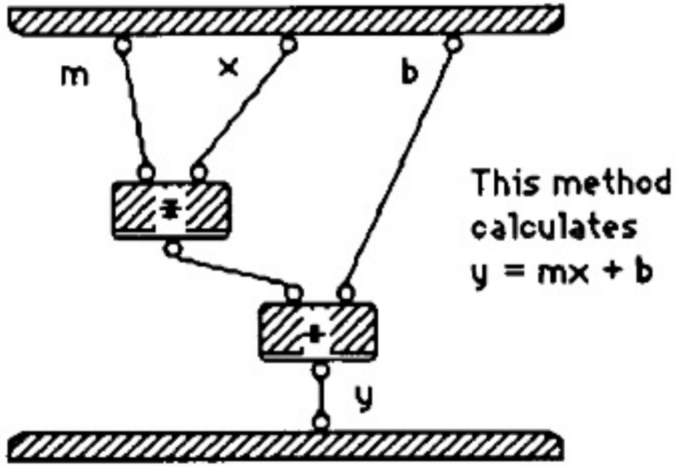
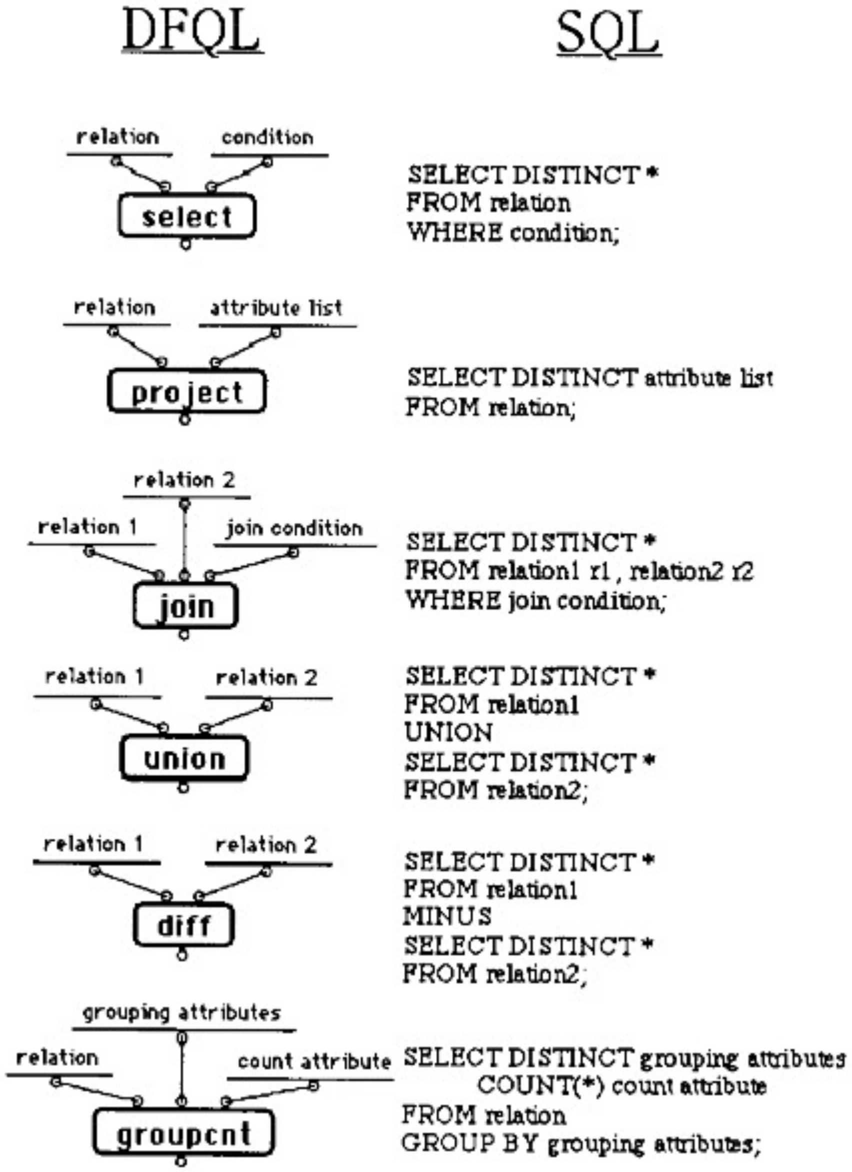


Fig. 2.



```
SELECT DISTINCT *
FROM relation
WHERE condition;
```

```
SELECT DISTINCT attribute list
FROM relation;
```

```
SELECT DISTINCT *
FROM relation1 r1 , relation2 r2
WHERE join condition;
```

```
SELECT DISTINCT *
FROM relation1
UNION
SELECT DISTINCT *
FROM relation2;
```

```
SELECT DISTINCT *
FROM relation1
MINUS
SELECT DISTINCT *
FROM relation2;
```

```
SELECT DISTINCT grouping attributes
    COUNT(*) count attribute
FROM relation
GROUP BY grouping attributes;
```

Fig. 4.

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

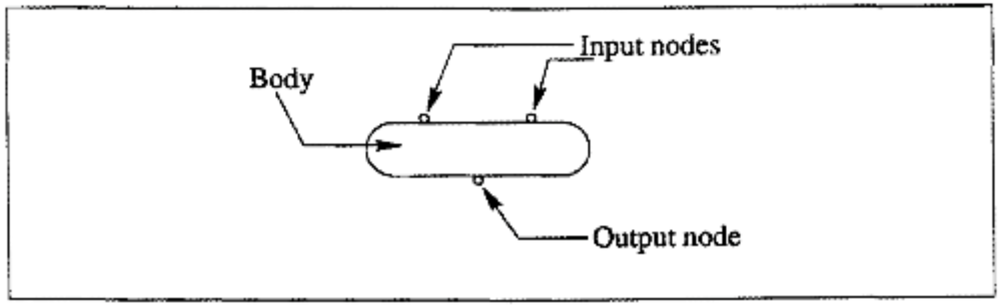


Figure 2.2: Operator Construction

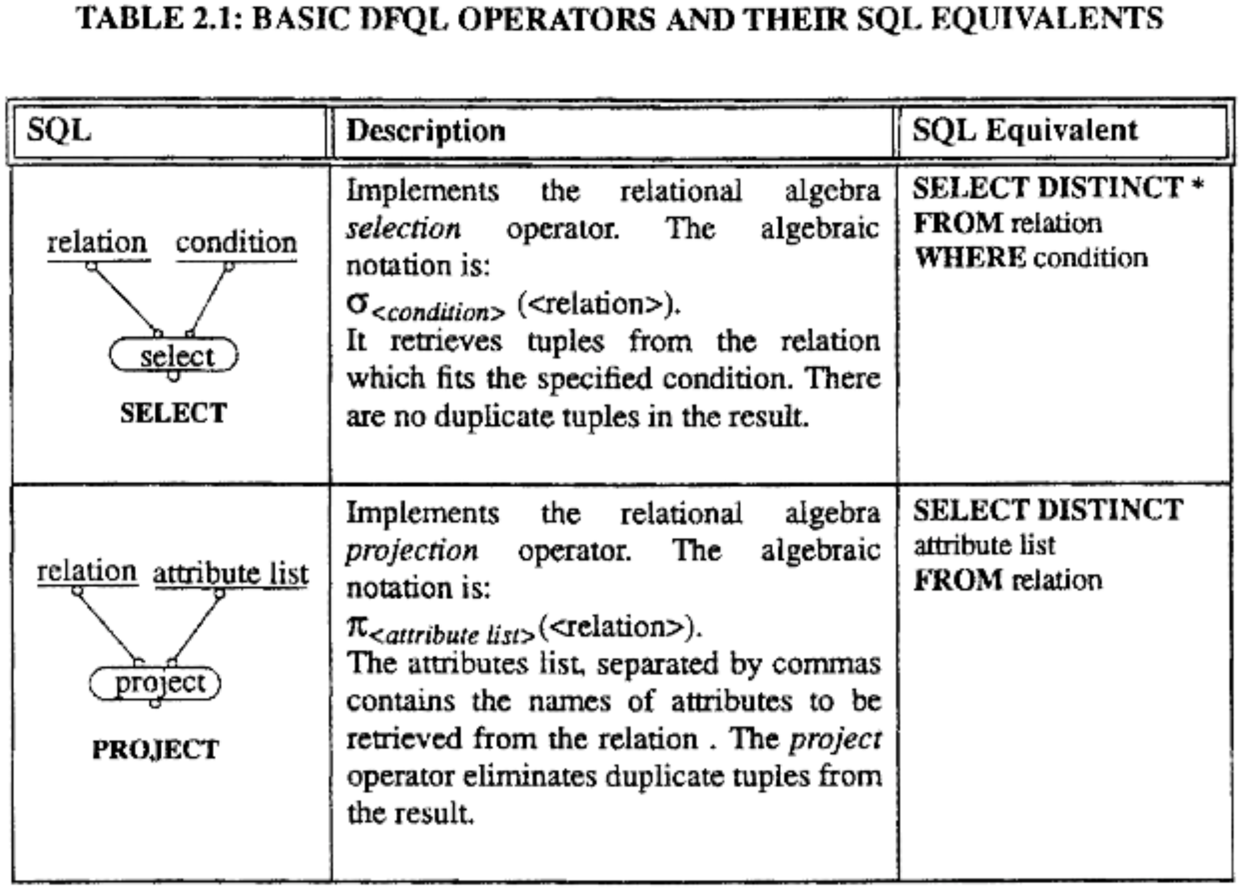


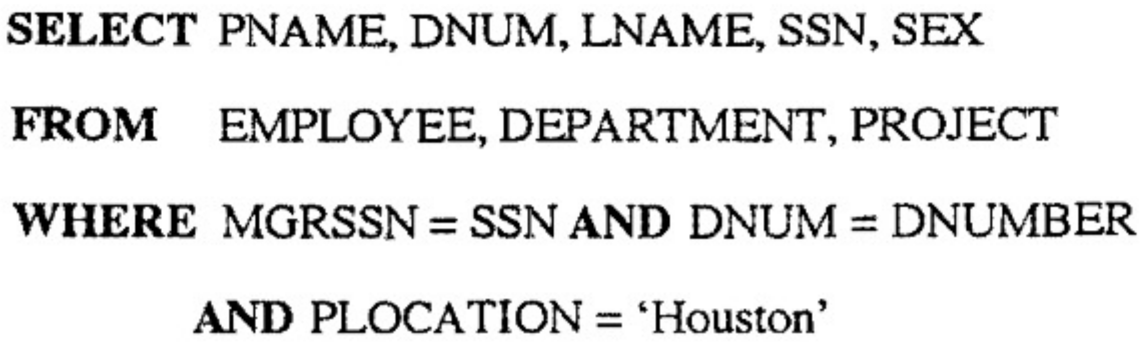
TABLE 2.1: BASIC DFQL OPERATORS AND THEIR SQL EQUIVALENTS

SQL	Description	SQL Equivalent
relation condition select **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
relation attribute list project **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

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

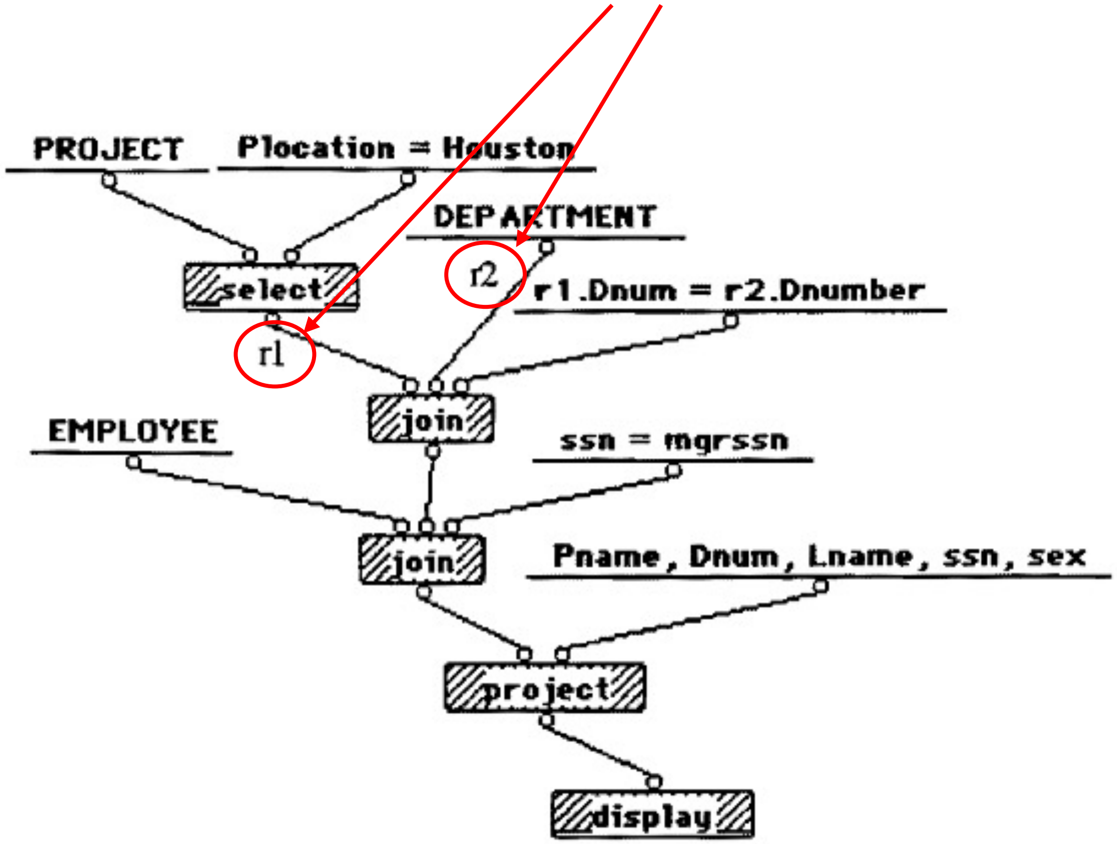
DFQL (DataFlow Query Language)

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."



```
SELECT  PNAME, DNUM, LNAME, SSN, SEX
FROM    EMPLOYEE, DEPARTMENT, PROJECT
WHERE   MGRSSN = SSN AND DNUM = DNUMBER
        AND PLOCATION = 'Houston'
```

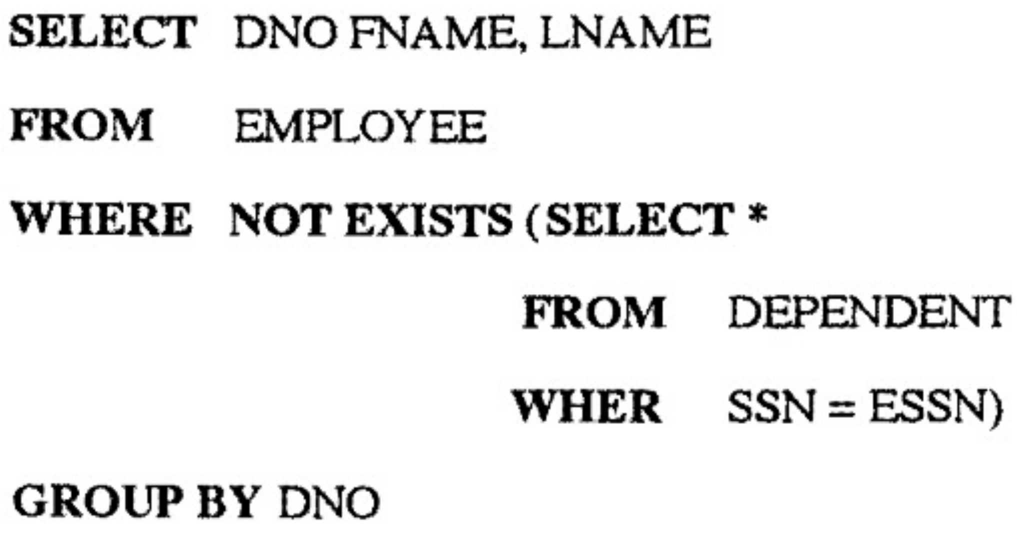
alias on edges to avoid ambiguities



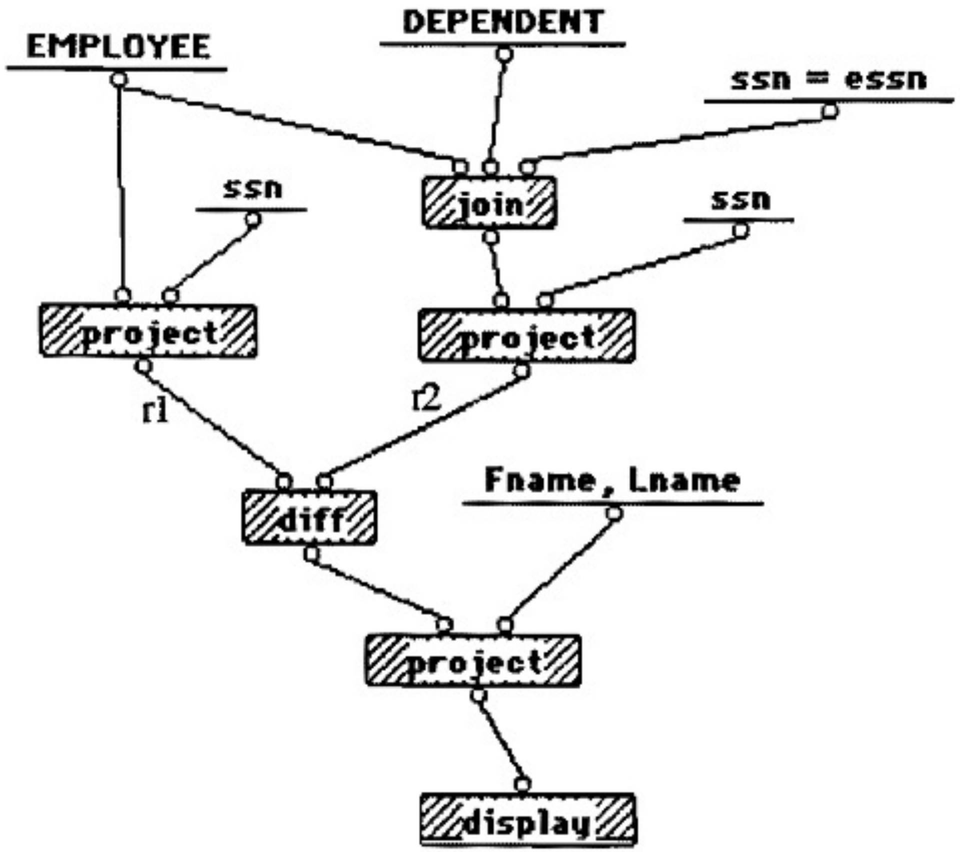
Girsang. The comparison of SQL, QBE, and DFQL as query languages for relational databases, Master thesis, Naval Postgraduate School, 1994. https://core.ac.uk/download/pdf/36723678.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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
```



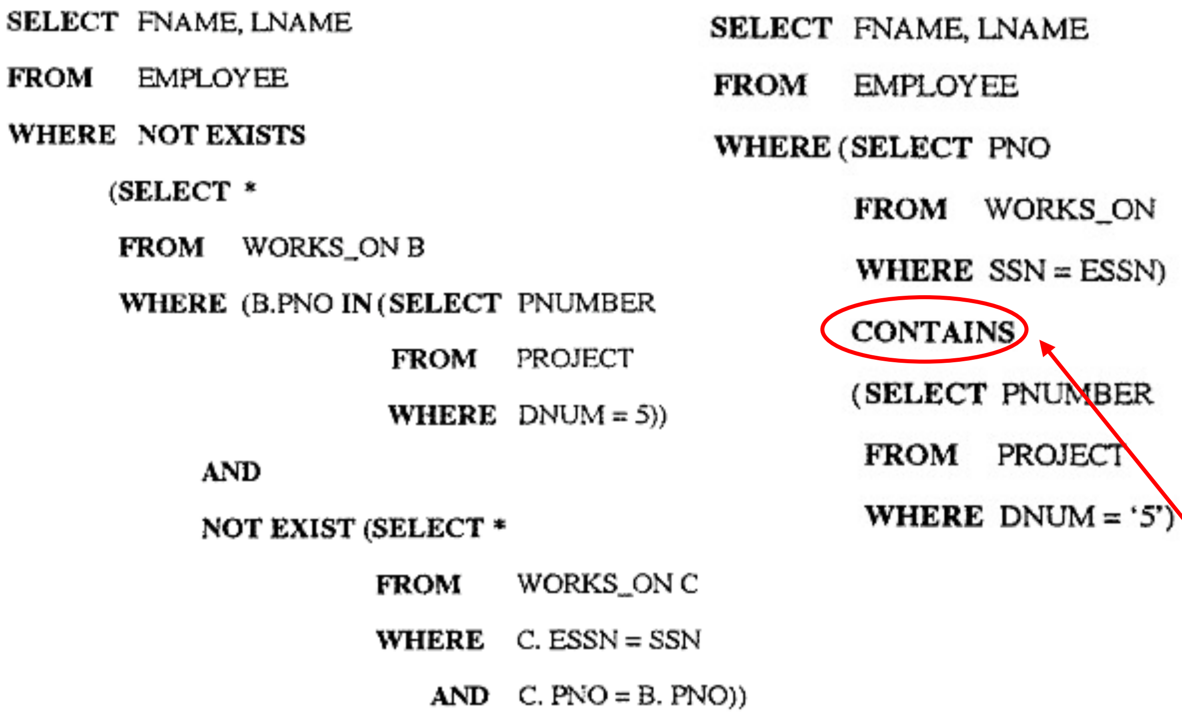
Girsang. The comparison of SQL, QBE, and DFQL as query languages for relational databases, Master thesis, Naval Postgraduate School, 1994. https://core.ac.uk/download/pdf/36723678.pdf

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

DFQL (DataFlow Query Language)

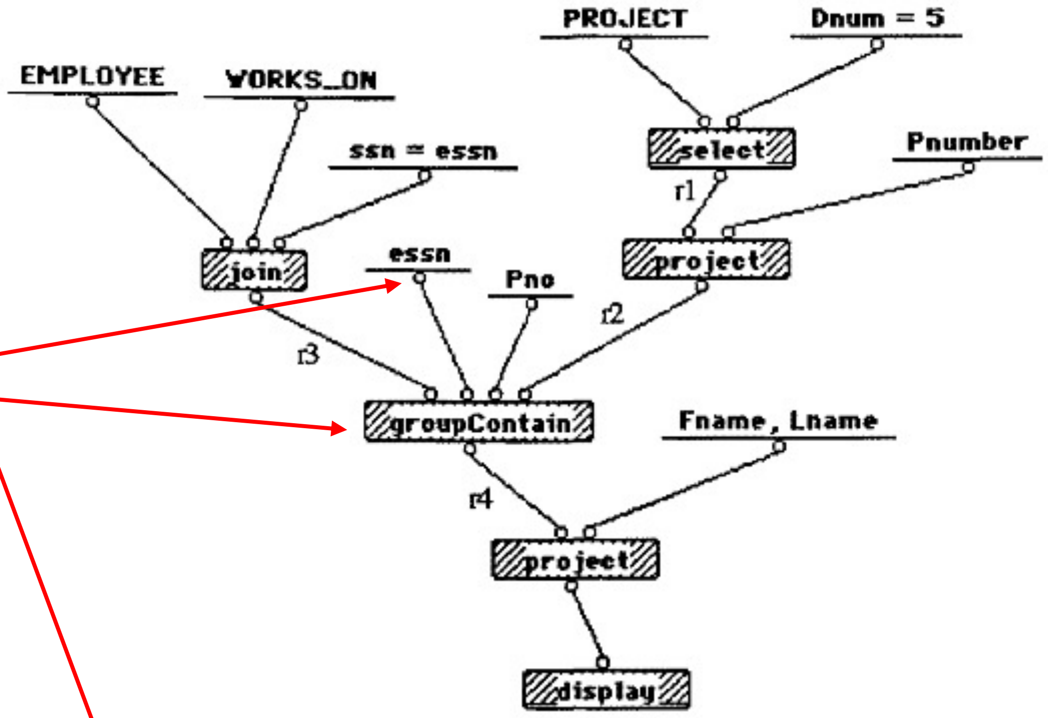
Query 9: "Retrieve the first name and last name of each employee who works on all the projects managed by department number 5."

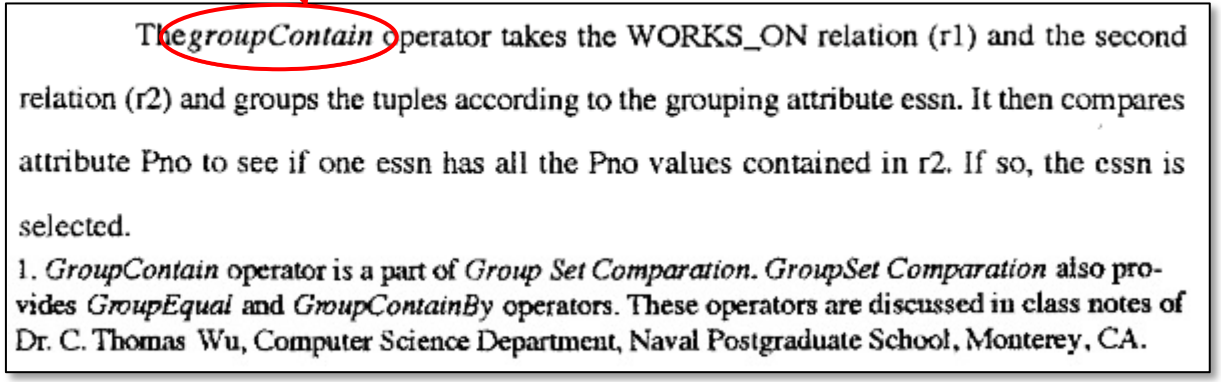
Based on the description of the operator, there seems to be an error in the visualization: the "groupContain" only returns the essn's, and a subsequent join with Employee is necessary



```
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))
```

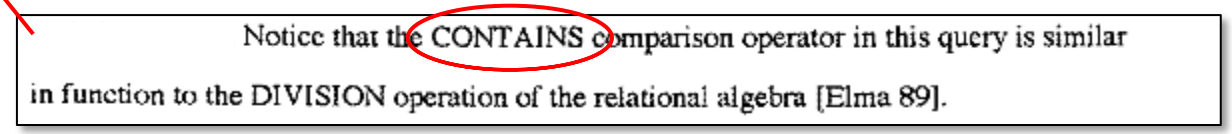
```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE (SELECT  PNO
       FROM    WORKS_ON
       WHERE   SSN = ESSN)
       CONTAINS
       (SELECT  PNUMBER
        FROM    PROJECT
        WHERE   DNUM = '5')
```





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].

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

Visual SQL (2003)

Sources used:

- 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 composition, but a Java implementation also supports the reverse functionality of query visualization to a large extent
- 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"

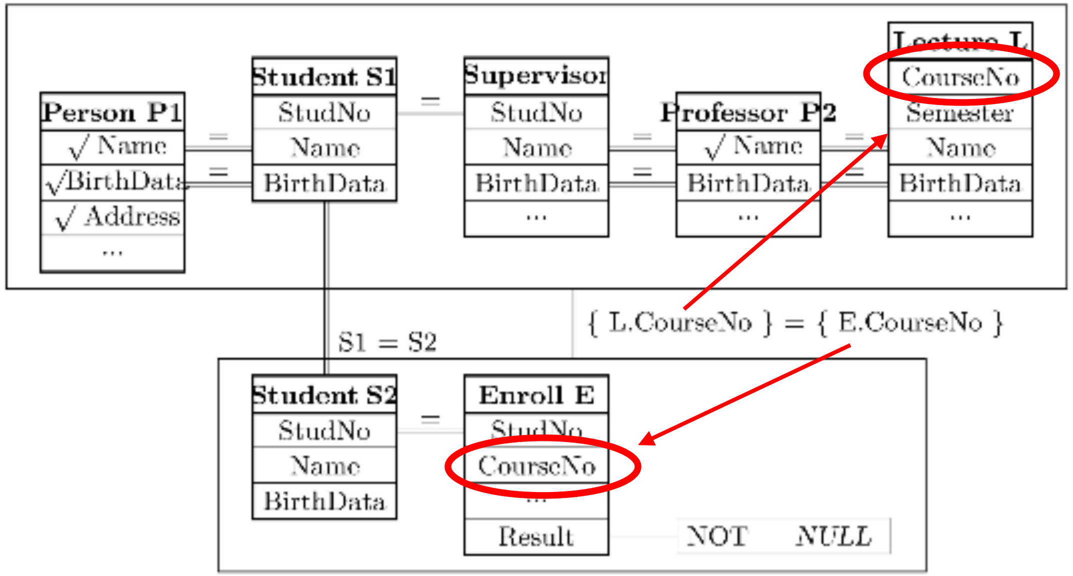


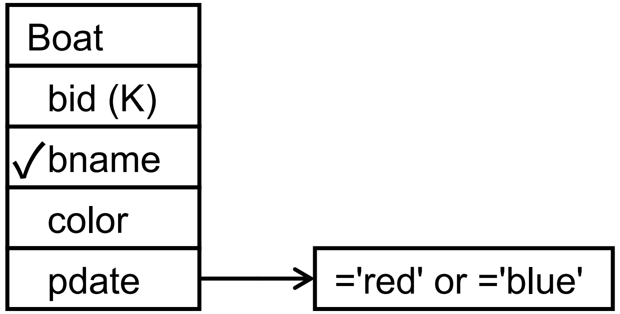
Fig. 1. Visual SQL Involving Equality On Two Visual SQL Subqueries

Figure 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

Visual SQL

Q1: "Find boats that are red or blue."

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



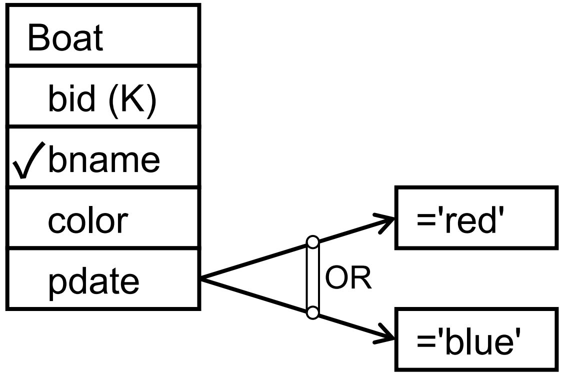
Disjunctions are explicitly expressed inside a box

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

Q1: "Find boats that are red or blue."

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



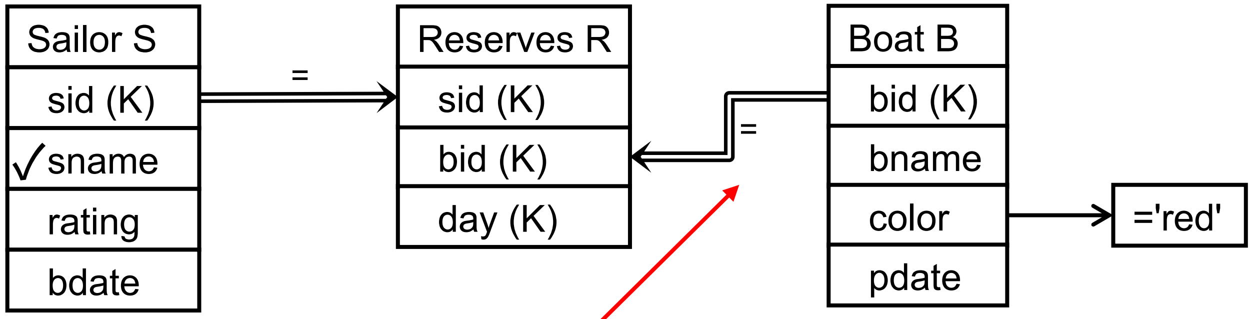
An alternative suggestion was to connect two alternative lines with an "OR" labeled connector

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

Q2: "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'
```



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

... $\underrightarrow{\text{B.bid=R.bid}}$ ...

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

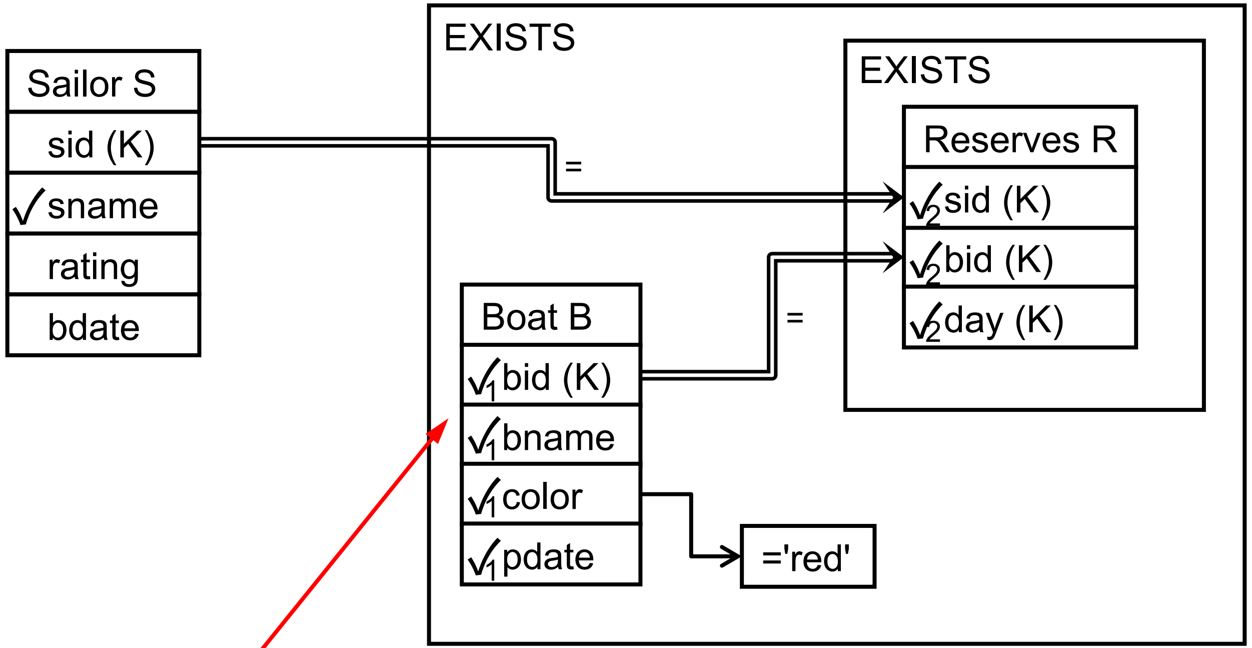
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

Q2: "Find sailors who reserved a red boat."

```
select distinct 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))
```

Nested queries preserve the scope of nestings



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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

Q4: "Find sailors who reserved all red boats."

```
select distinct 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))
```

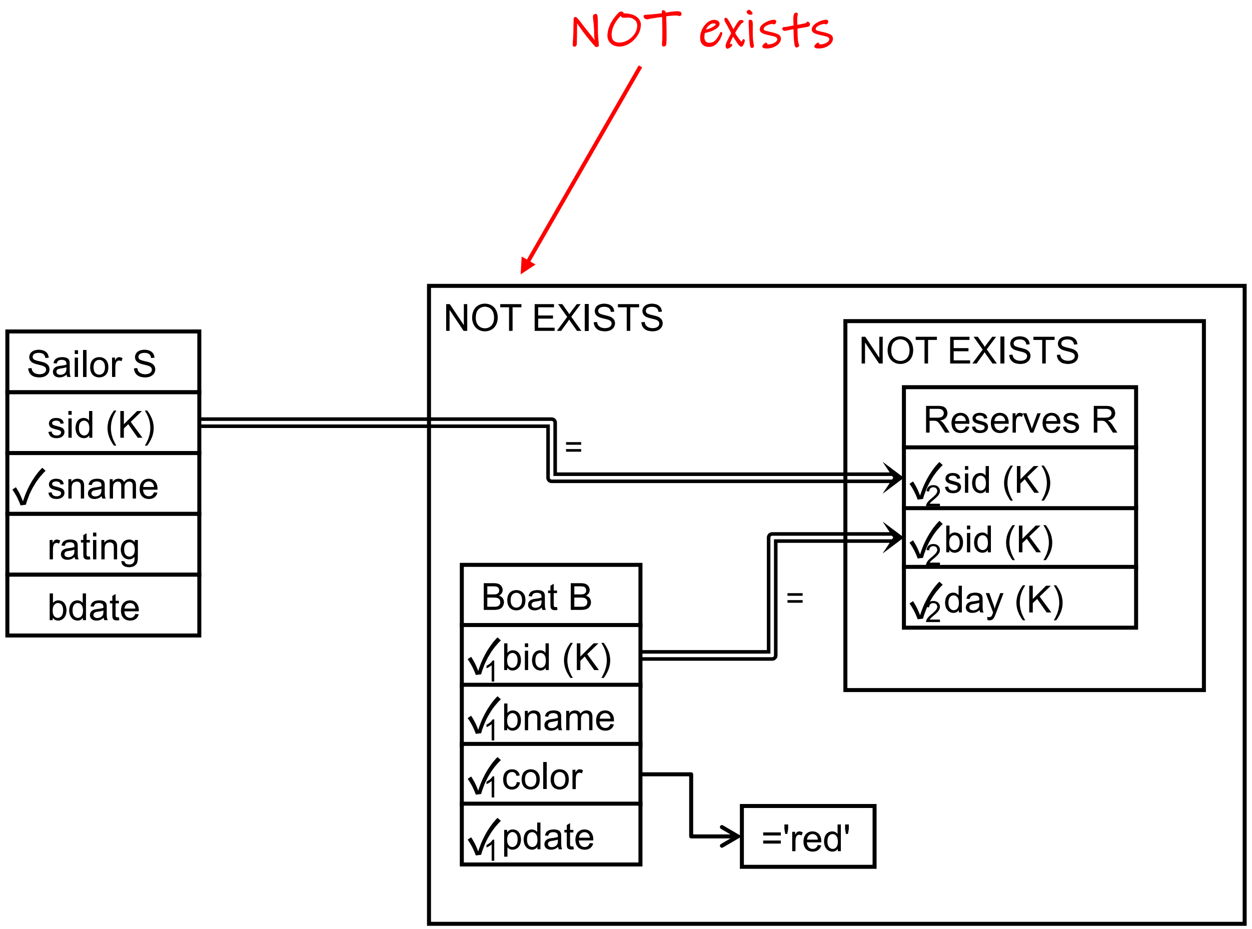


Figure drawn based on a result by Jiahui Zhang visualized with a code available on the website of University of Kiel around 4/2020

Visual SQL

Q4: "Find sailors who reserved all red boats."

```
select distinct 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

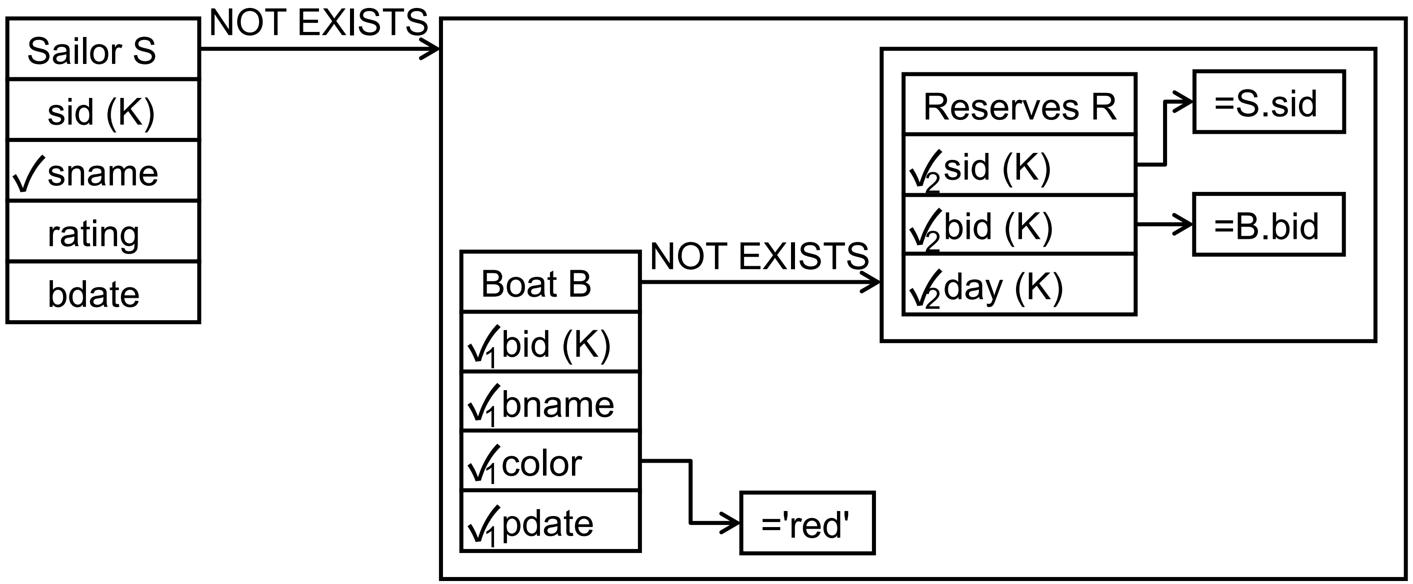


Figure drawn based on Figure 49 in "Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003."

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

Q3: "Find sailors who reserved only red boats."

```
select distinct 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.

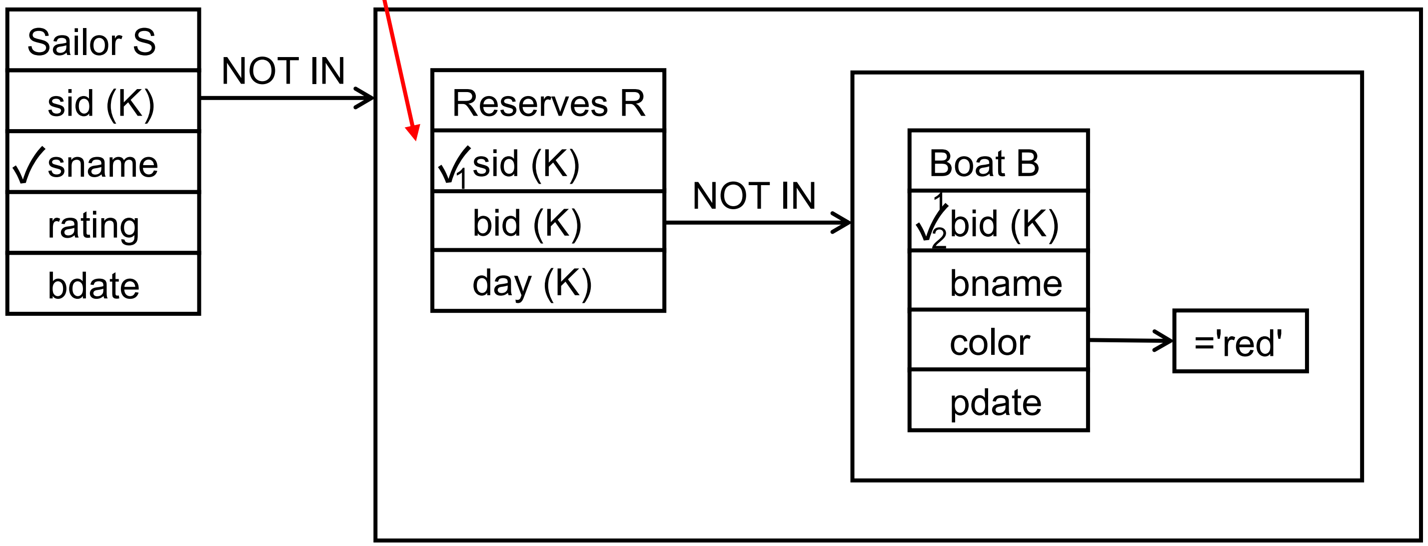


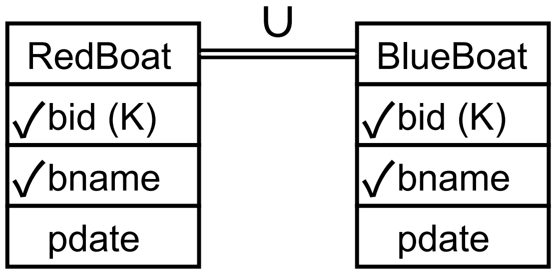
Figure drawn based on Figures 38/49 in "Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003."

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```



Schema

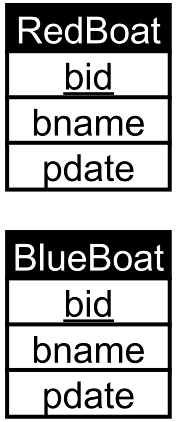


Figure drawn based on Figures 47 in "Thalheim. Visual SQL -- Eine ER-basierte Einfuehrung in die Datenbankprogrammierung. Teil I. Report I–08/03 of the Computer Science Institute at BTU Cottbus, 2003."

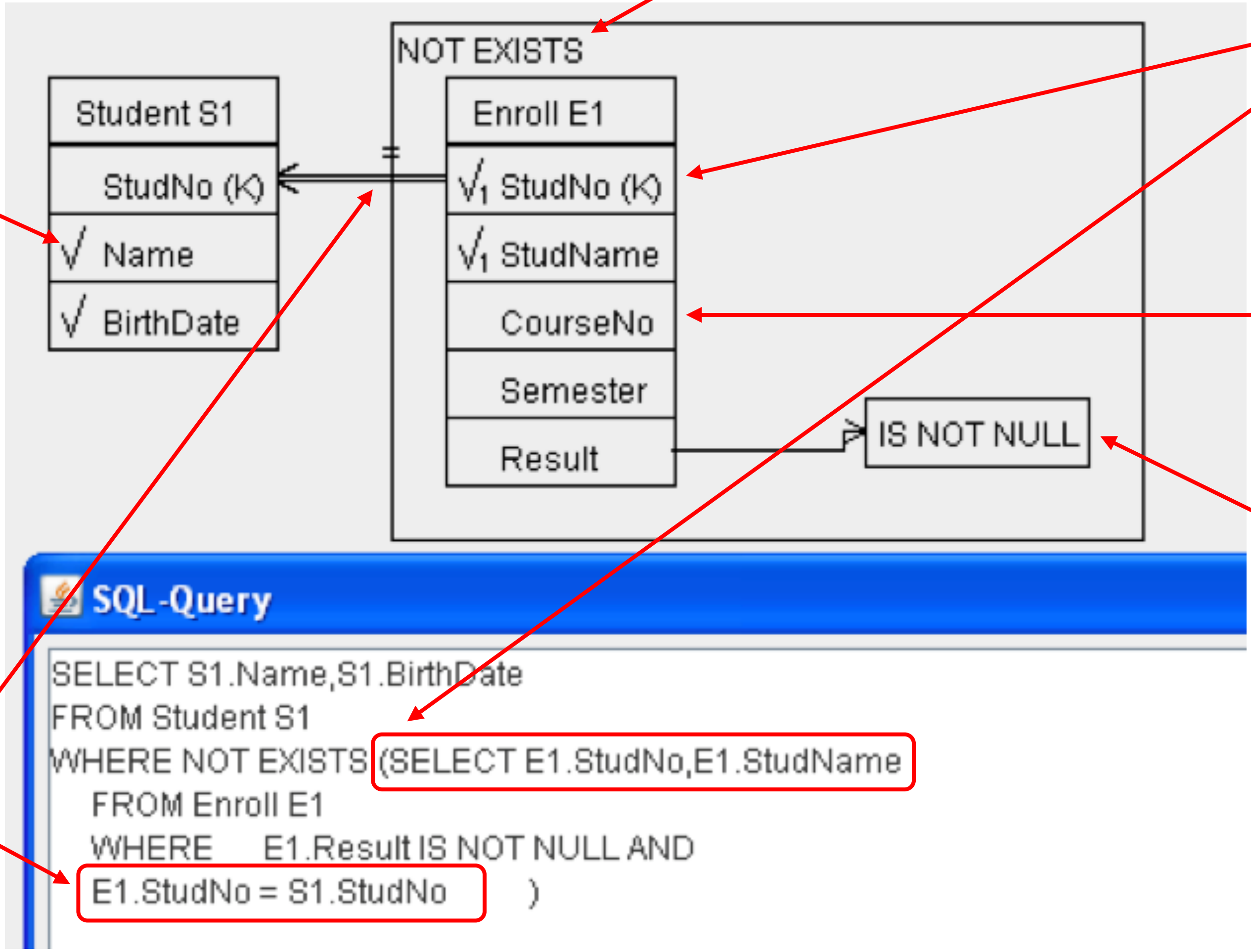
Visual SQL (2003)
BACKUP

Visual SQL

Q: "Return ("successless") students who no enrollment with null as result."

Check-marks show selected attributes

Arrows preserve the order of the attributes in the predicate



```
SELECT S1.Name,S1.BirthDate
FROM Student S1
WHERE NOT EXISTS (SELECT E1.StudNo,E1.StudName
   FROM Enroll E1
   WHERE     E1.Result IS NOT NULL AND
   E1.StudNo = S1.StudNo          )
```

EXISTS / NOT EXISTS written out with words and different from IN / NOT IN

Attributes "selected" in a subquery are marked with a labeled check mark

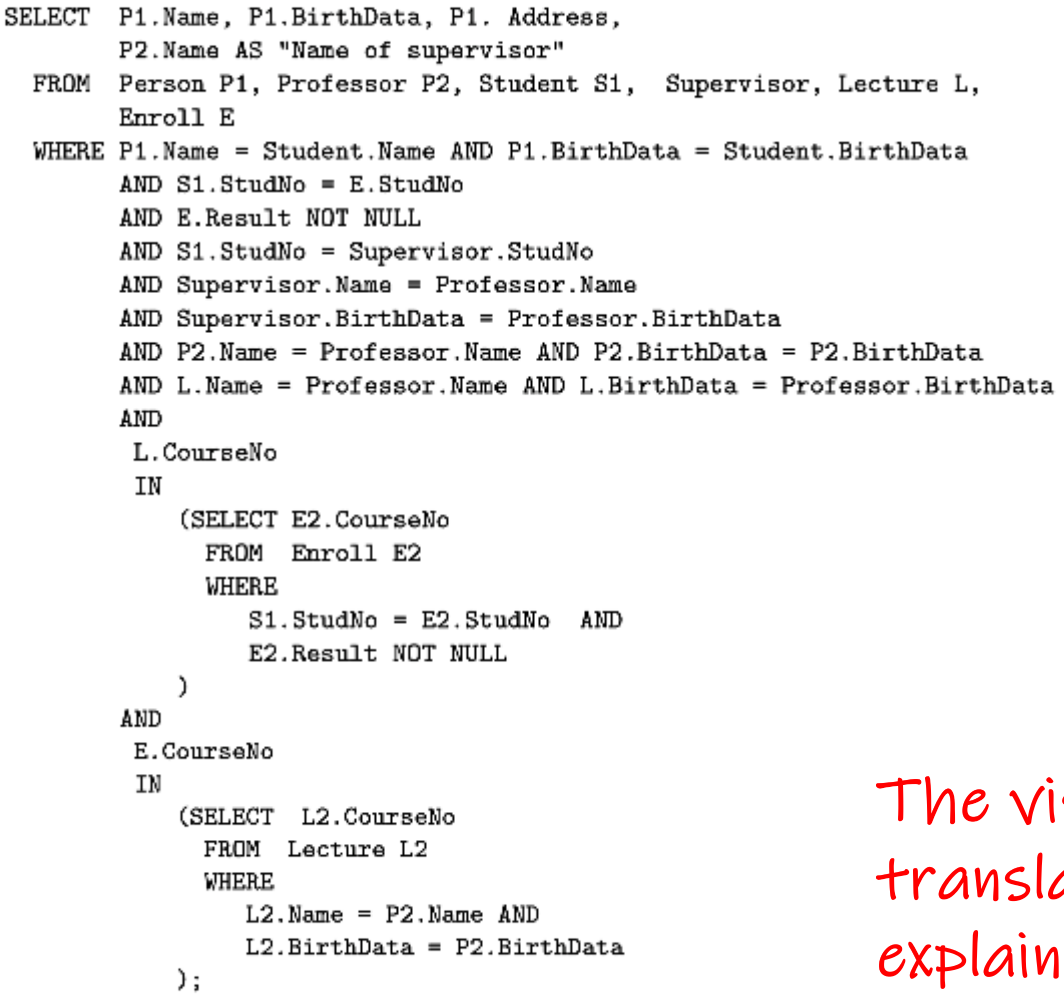
All attributes from a table are shown even if not used.

Constants for selection predicates are in separate boxes

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
            );
```

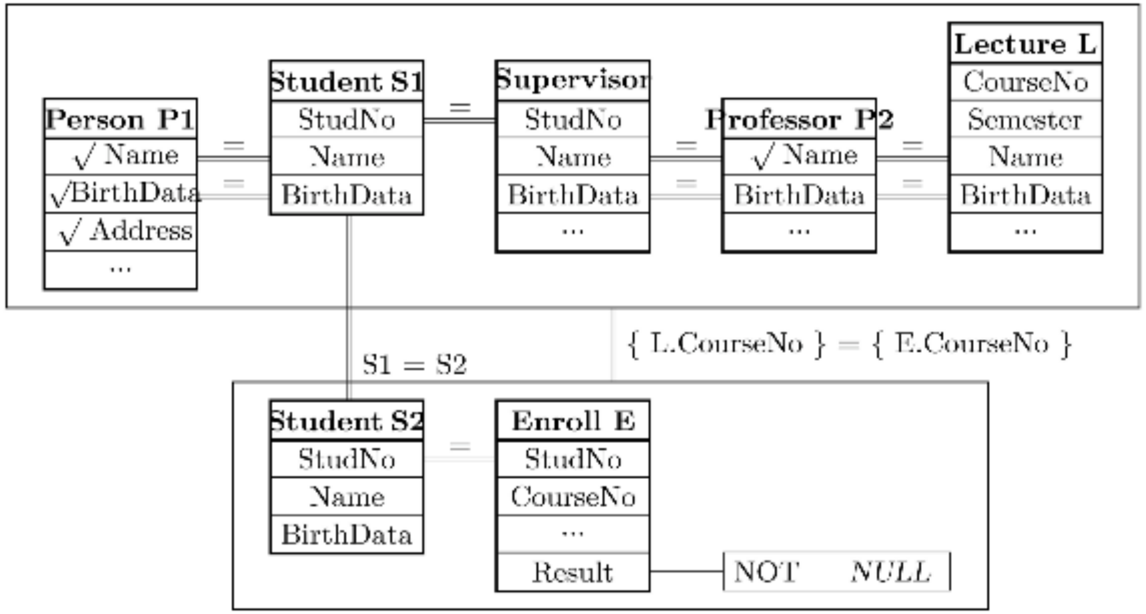


Fig. 1. Visual SQL Involving Equality On Two Visual SQL Subqueries

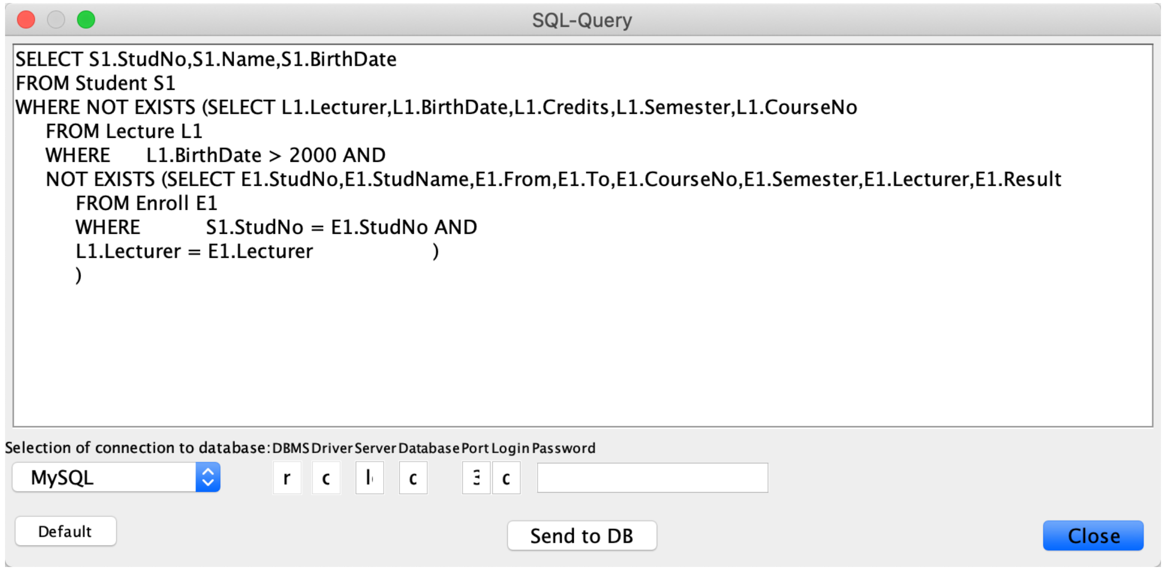
The visualization contains fewer tables and the translation between the visualization and SQL is not explained in the paper.

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

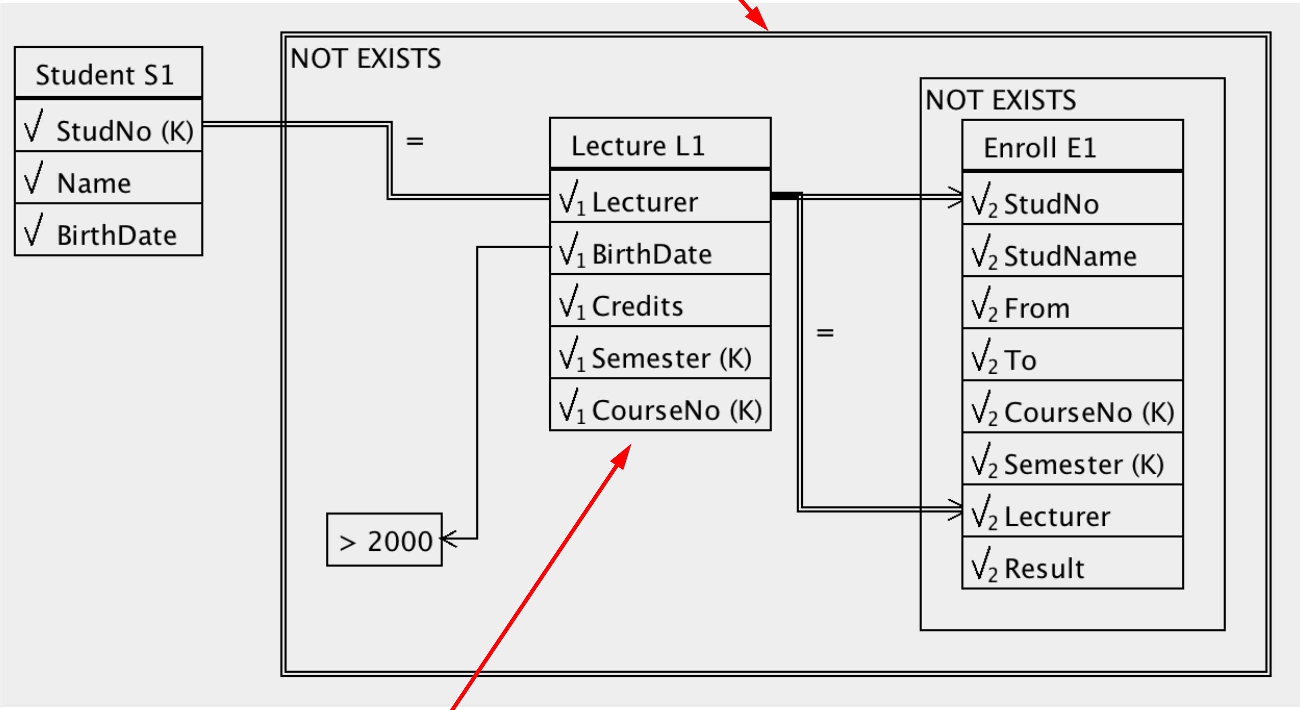
Visual SQL

Q: "Find students who have taken classes from all lecturer who were born after 2000"



```
SELECT S1.StudNo,S1.Name,S1.BirthDate
FROM Student S1
WHERE NOT EXISTS (SELECT L1.Lecturer,L1.BirthDate,L1.Credits,L1.Semester,L1.CourseNo
    FROM Lecture L1
    WHERE      L1.BirthDate > 2000 AND
    NOT EXISTS (SELECT E1.StudNo,E1.StudName,E1.From,E1.To,E1.CourseNo,E1.Semester,E1.Lecturer,E1.Result
        FROM Enroll E1
        WHERE          S1.StudNo = E1.StudNo AND
        L1.Lecturer = E1.Lecturer                    )
        )
```

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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...

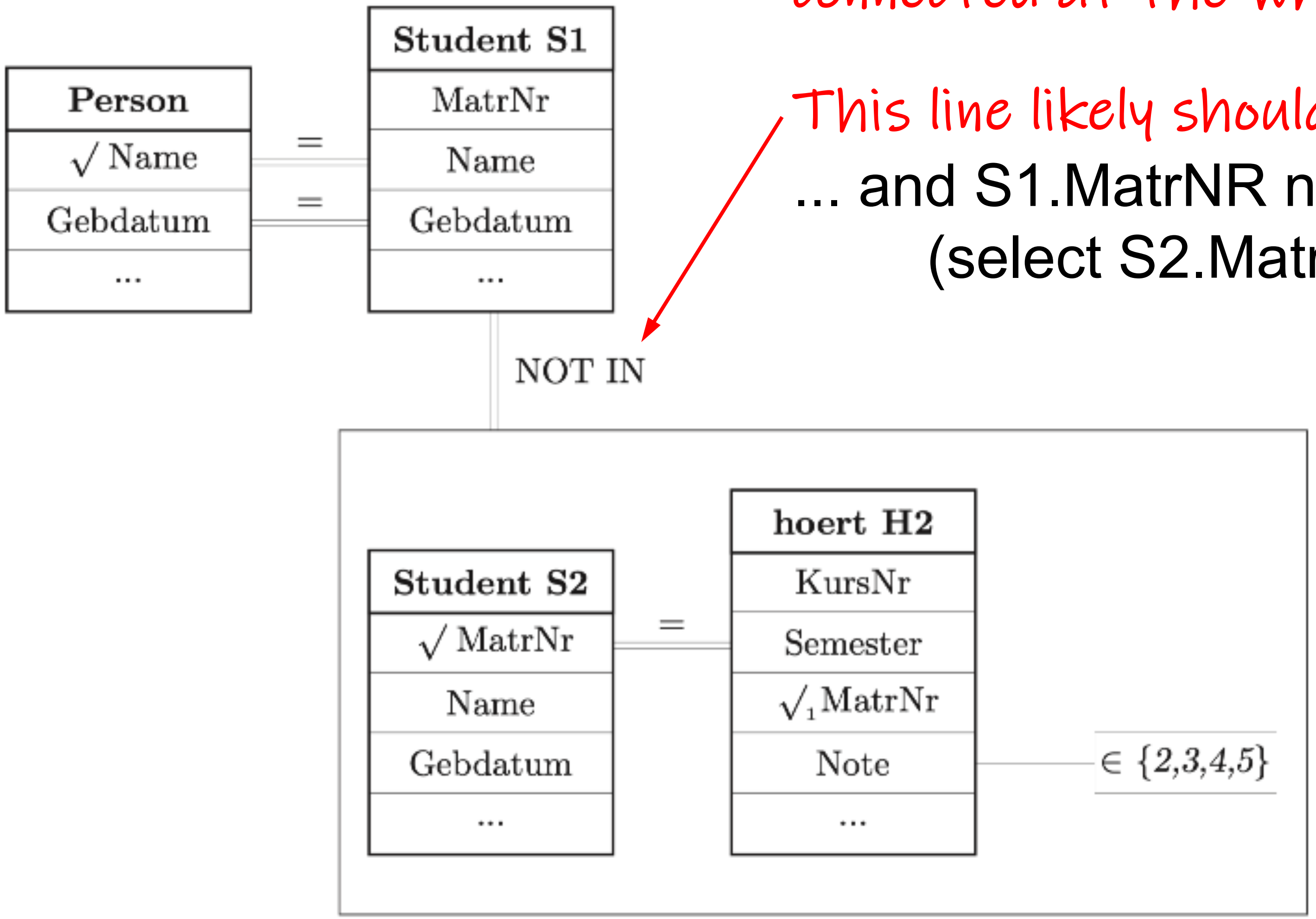


Bild 40: Teilanfrage mit Differenzbildung

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.

Visual SQL

Complicated Boolean expressions are hard to visualize (by nature)

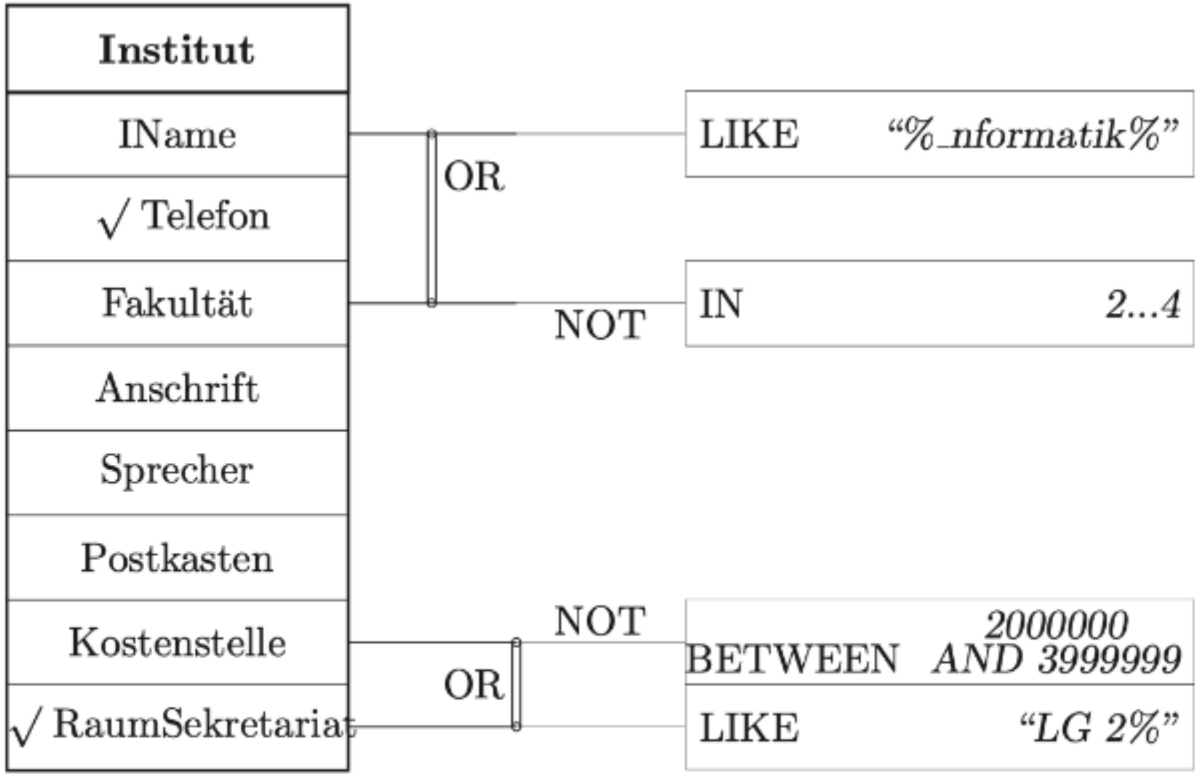


Bild 20: Auswahl in Klassen mit AND, OR und NOT

```
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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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

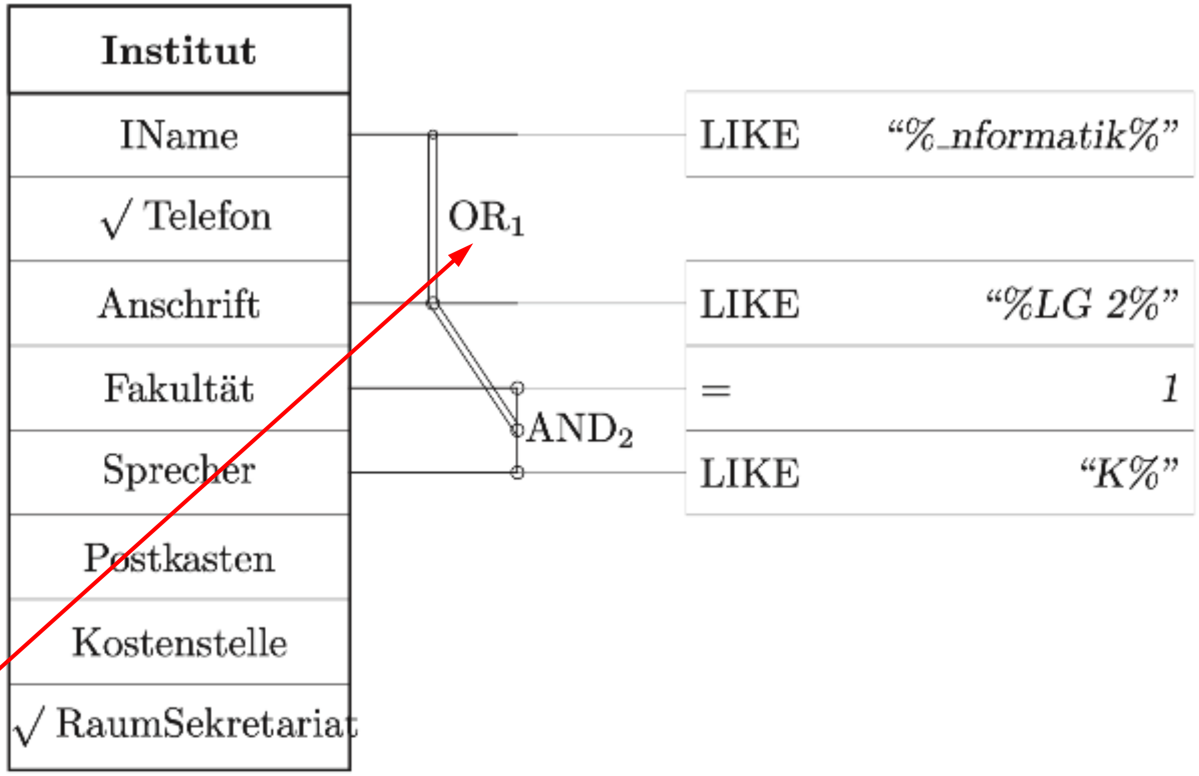
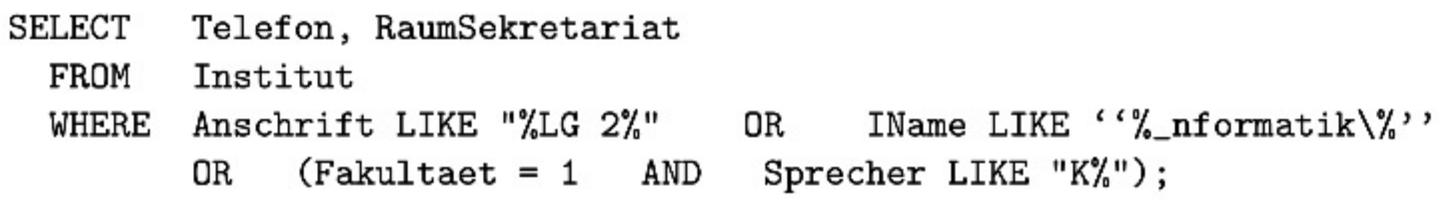


Bild 19: Auswahl in Klassen mit AND und OR



```
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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

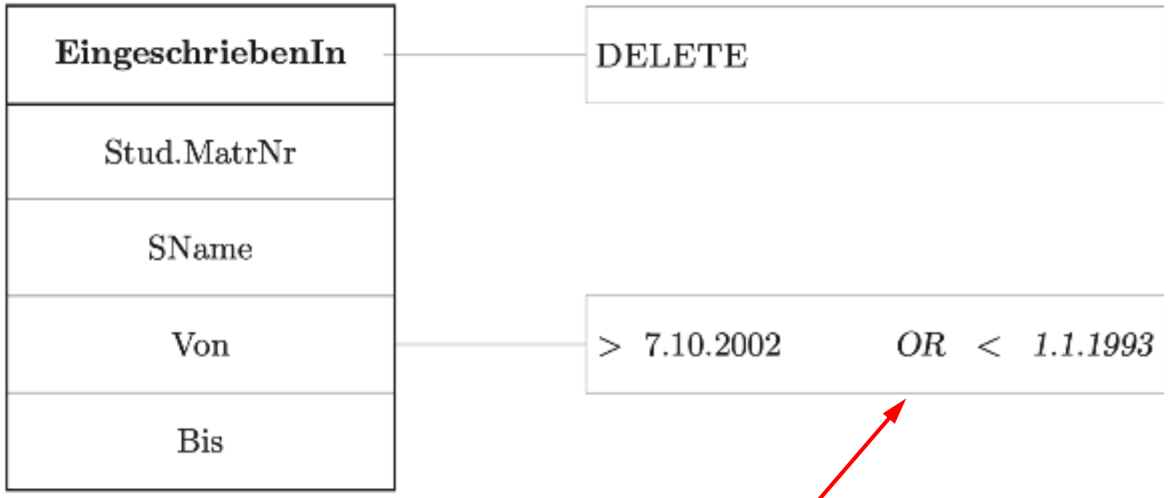
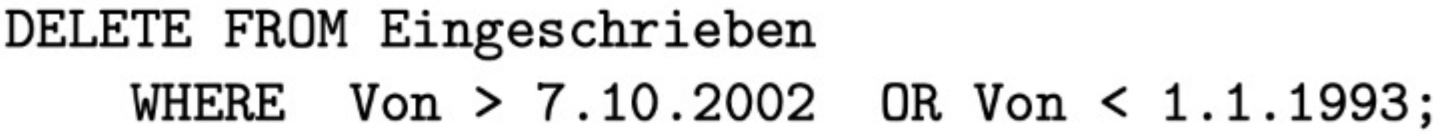


Bild 14: Streichen von Objekten mit Bedingung

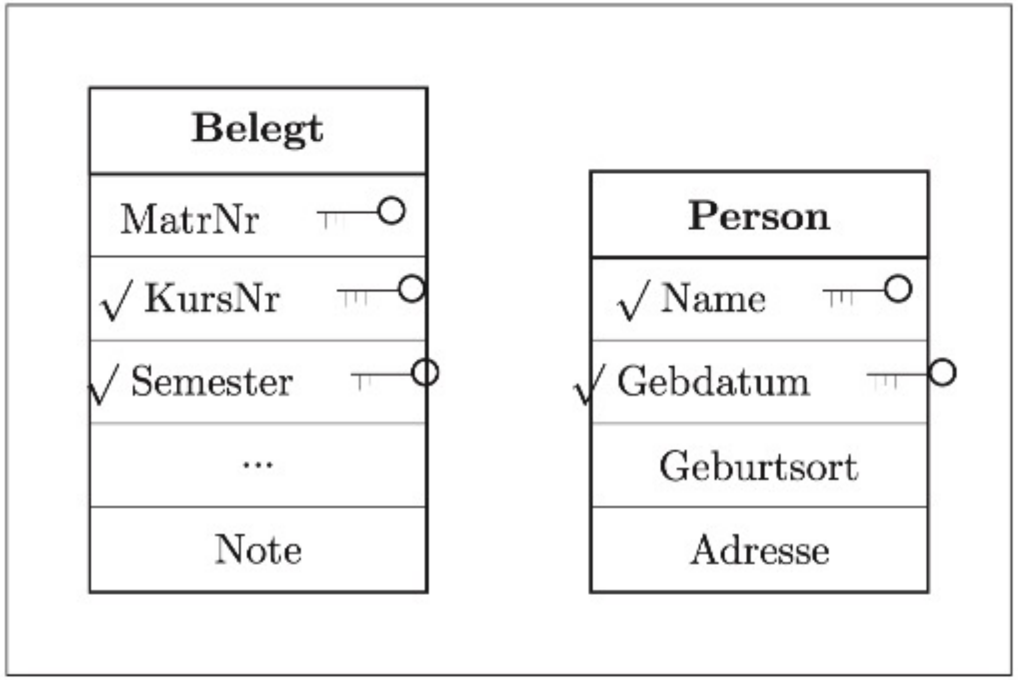
Disjunction of values
on one predicate



```
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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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.
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

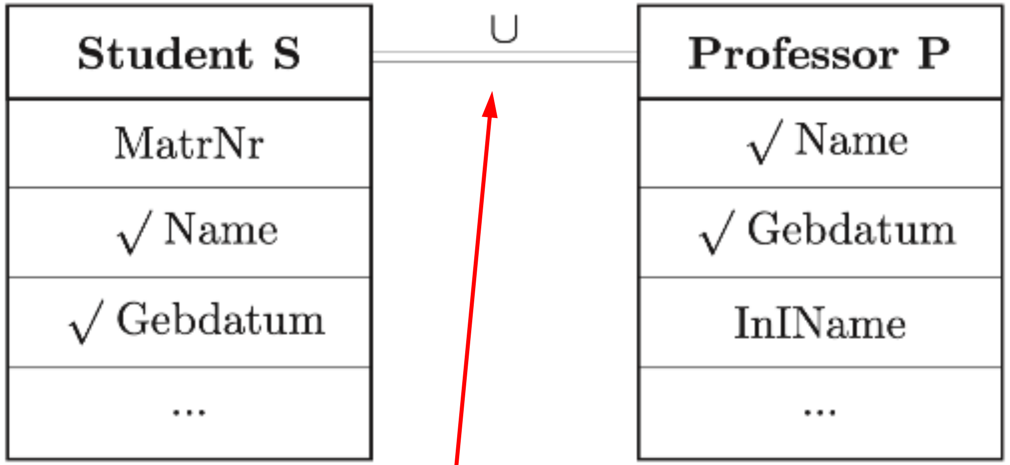


Bild 47: Vereinigung von zwei vollständig typengleichen Relationen

Union

```
SELECT      S.Name, S.Gebdatum
  FROM      Student S
UNION
SELECT      P.Name, P.Gebdatum
  FROM      Professor P;
```

Source: Figure 47 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Visual SQL

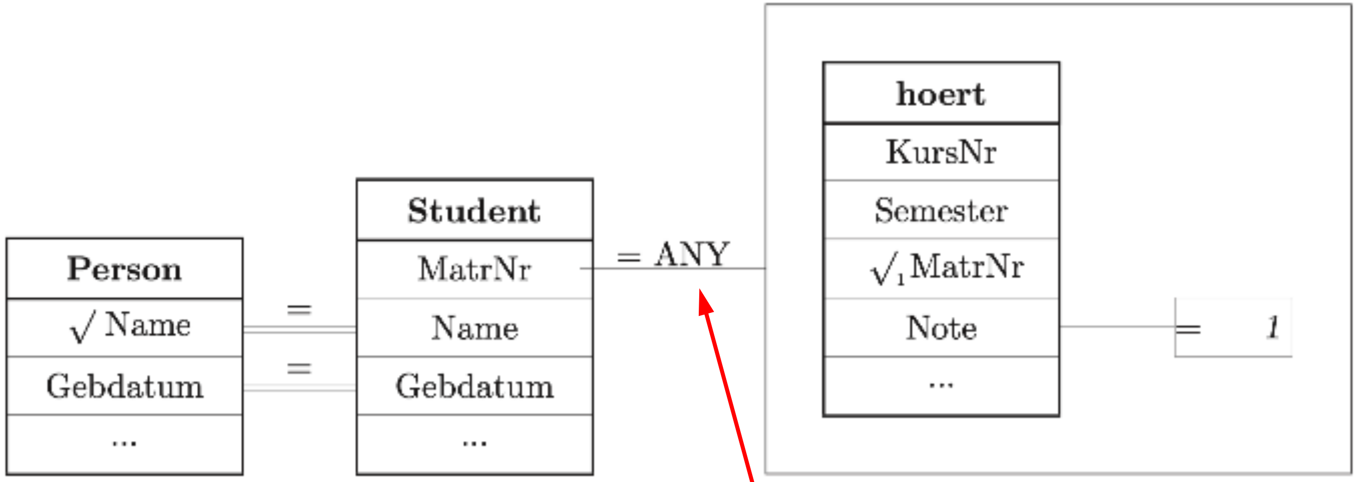


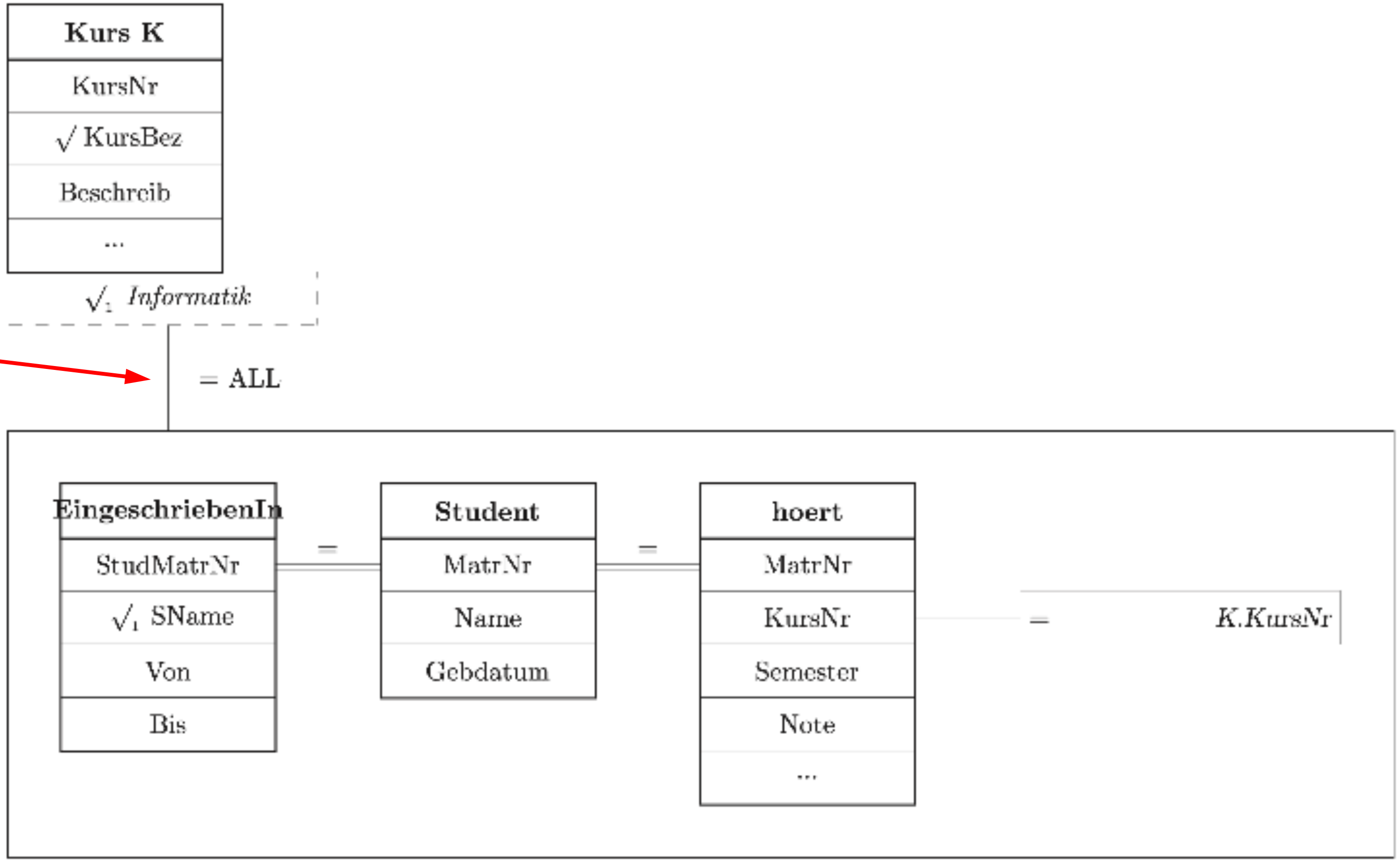
Bild 38: Teilanfrage mit Vergleich

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

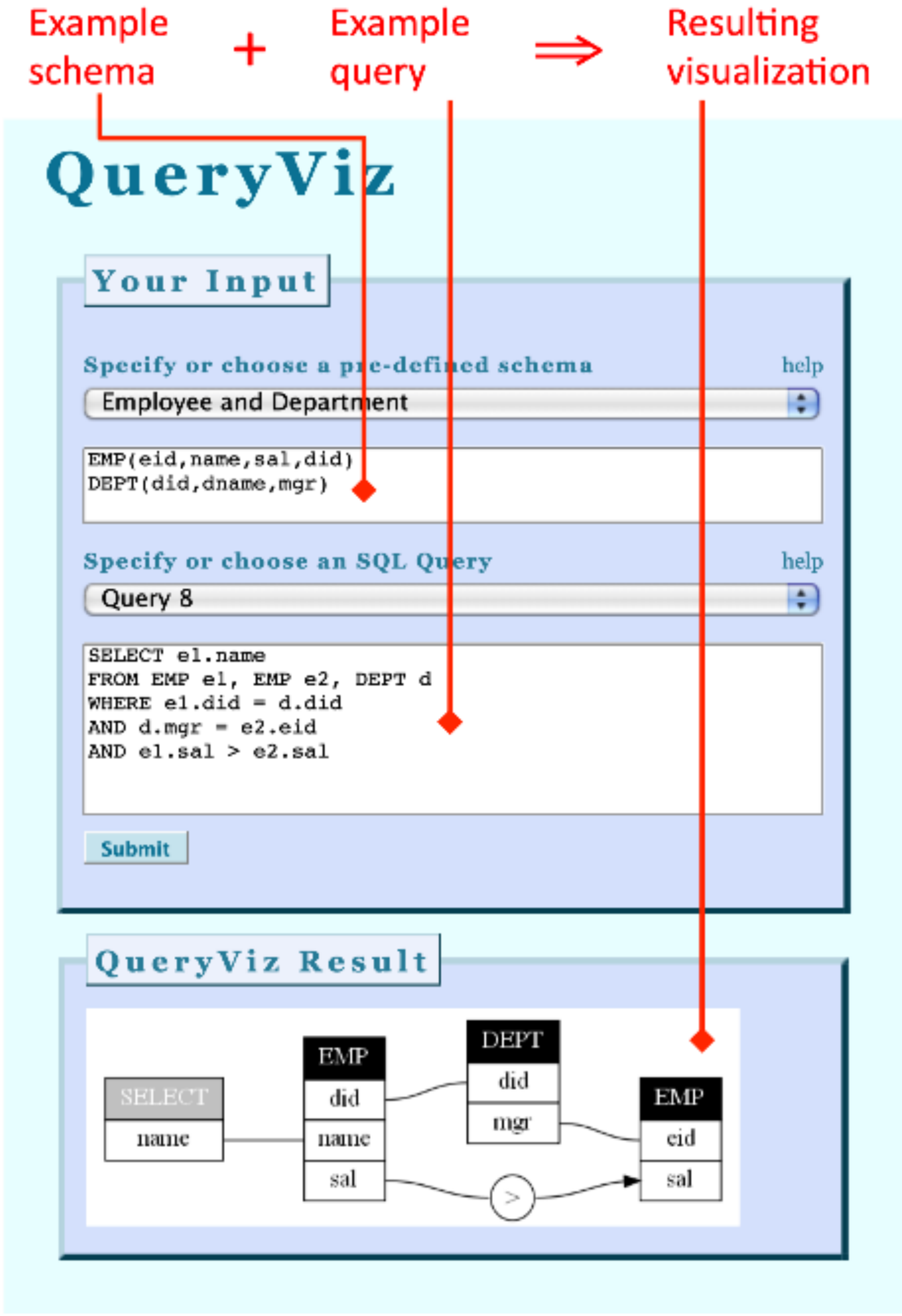
QueryVis (2011)
(also QueryViz)

Sources used:

- 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



Example schema + Example query ⇒ Resulting visualization

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

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

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

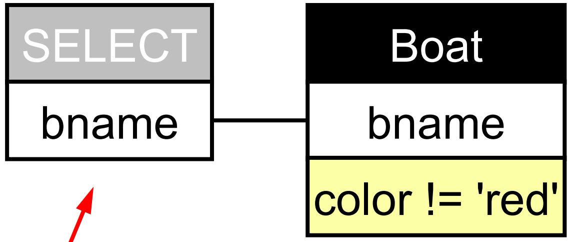
QueryVis (formerly QueryViz)

Schema

Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

```
select distinct 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

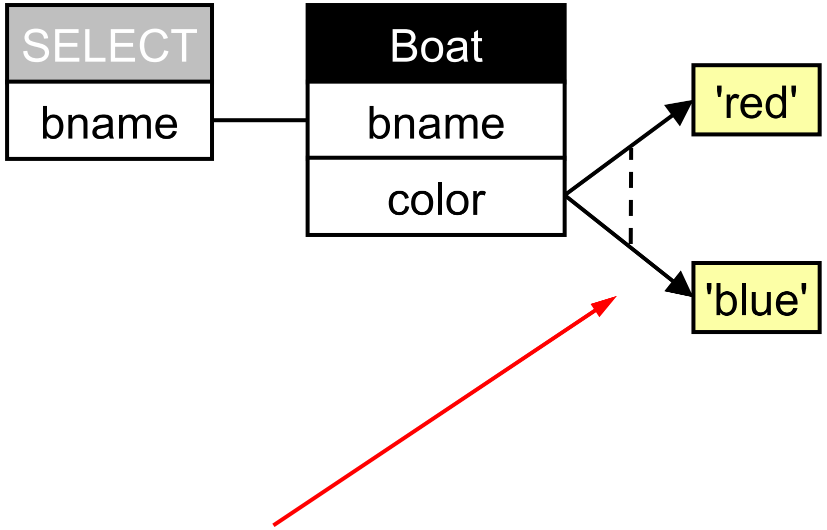
QueryVis (formerly QueryViz)

Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

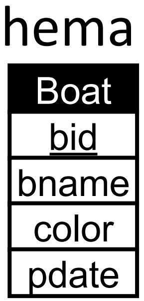
```
select distinct bname
from Boat
where color = 'red'
or color = 'blue'
```



Disjunction was never implemented in online QueryVis

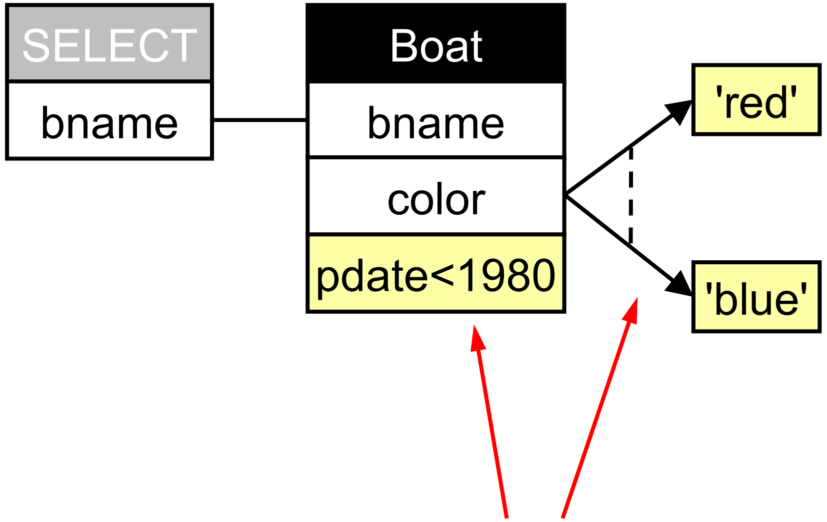
Figure drawn based on the presentations for "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805": https://www.youtube.com/watch?v=kVFnQRGAQls&list=PL_72ERGKF6DR4R0Cowx-LnnnqLXRf4ZjB

QueryVis (formerly QueryViz)



Q1c: "Find boats that are red or blue and purchased before 1980."

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



Conditions on separate attributes can be applied on each attribute separately. However, it is not clear how to express complicated Boolean expressions involving multiple attributes, such as:
"(color = 'red' and pdate < 1980) or (color = 'blue' and pdate > 1980)"
More on this later

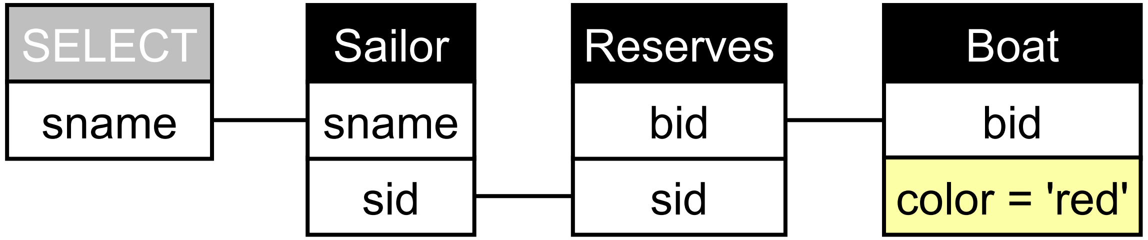
Figure drawn based on the presentations for "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805": https://www.youtube.com/watch?v=kVFnQRGAQls&list=PL_72ERGKF6DR4R0Cowx-LnnnqLXRf4ZjB

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q2: "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)

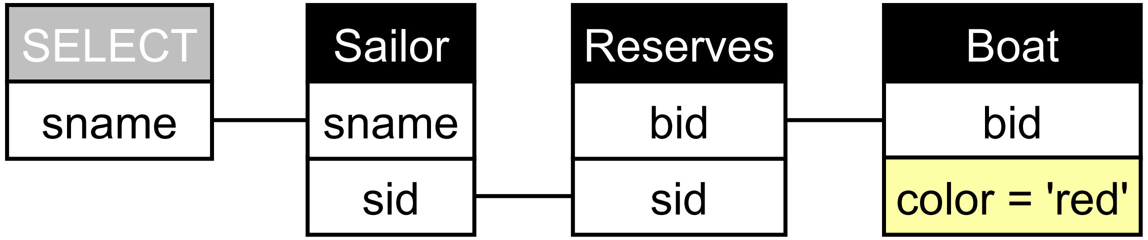
$\{q.sname \mid \exists s \in Sailor, \exists r \in Reserves, \exists b \in Boat[q.sname=s.sname \wedge r.sid=s.sid \wedge b.bid=r.bid \wedge b.color='red']\}$

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q2: "Find sailors who reserved a red boat."

```
select distinct 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)

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

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

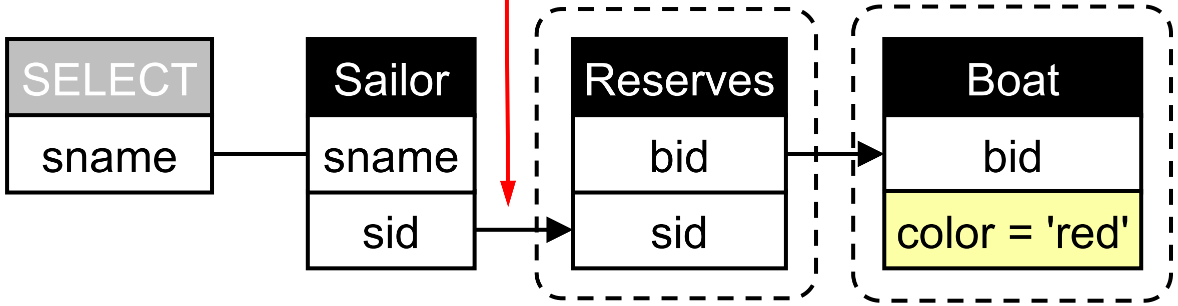
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q3: "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'))
```

Arrows encode the reading order



Dashed lines represent "not exists" (∄)

TRC (Tuple Relational Calculus)

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

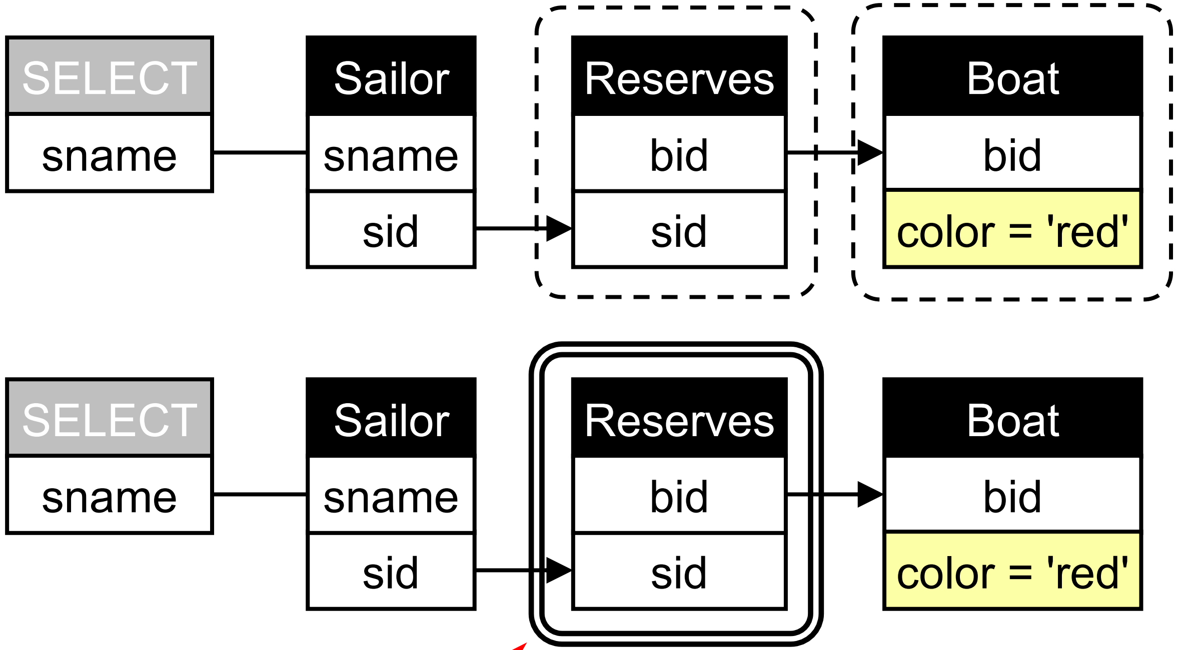
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q3: "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 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'))
```



Double lines represent for all ∀

TRC (Tuple Relational Calculus)

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

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

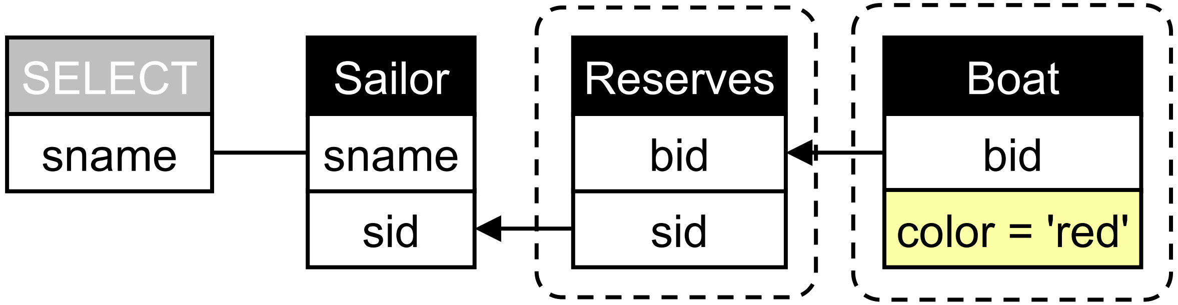
Figure drawn based on "Danaparamita, Gatterbauer. QueryViz: Helping Users Understand SQL queries and their patterns. EDBT demo. 2011. https://doi.org/10.1145/1951365.1951440 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q4: "Find sailors who reserved all red boats."

```
select distinct 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)

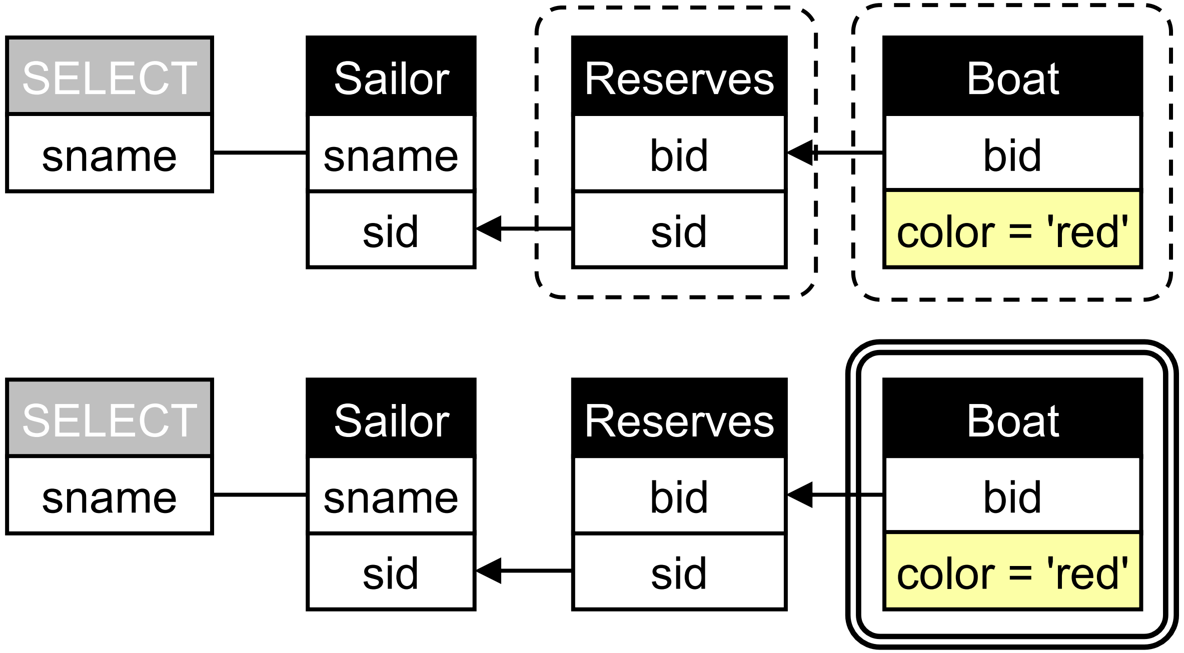
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ¬(∃b∈Boat[b.color='red' ∧ ¬(∃r∈Reserves[b.bid=r.bid ∧ r.sid=s.sid])])]}

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Q4: "Find sailors who reserved all red boats."

```
select distinct 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)

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

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

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly QueryViz)

Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

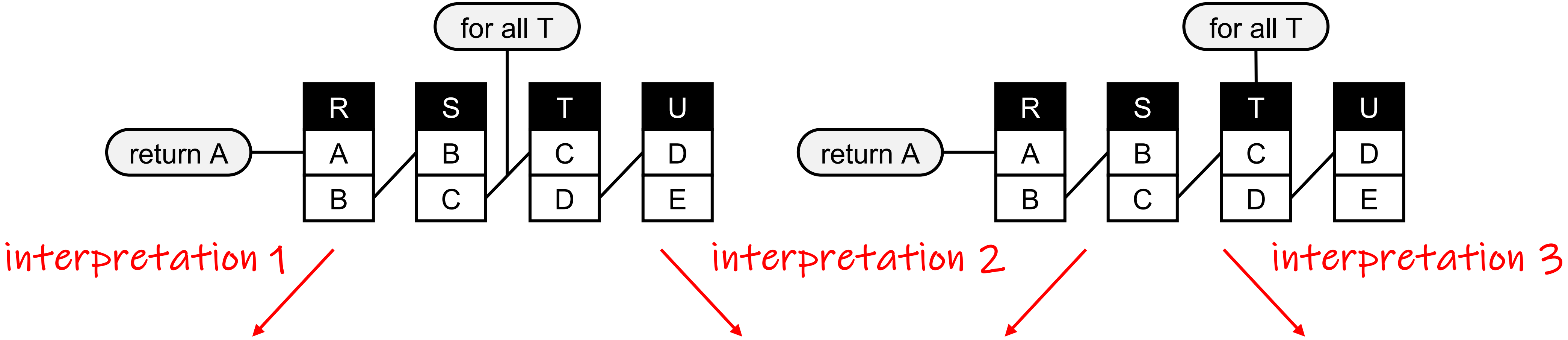
```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

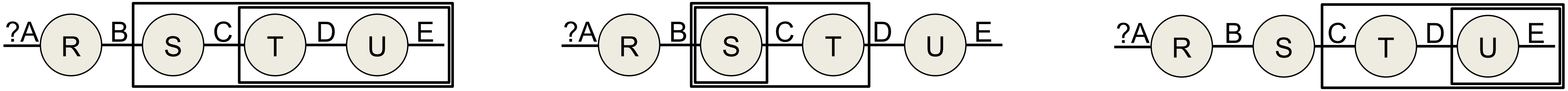
QueryVis does not support Union

TRC (Tuple Relational Calculus)

$$\{q(\text{bid},\text{bname}) \mid \exists b \in \text{RedBoat}\,[q.\text{bid}=b.\text{bid} \wedge q.\text{bname}=b.\text{bname}]$$
$$\vee\ \exists b \in \text{BlueBoat}\,[q.\text{bid}=b.\text{bid} \wedge q.\text{bname}=b.\text{bname}]\}$$

The quantifier example from OO-VQS is unambiguous

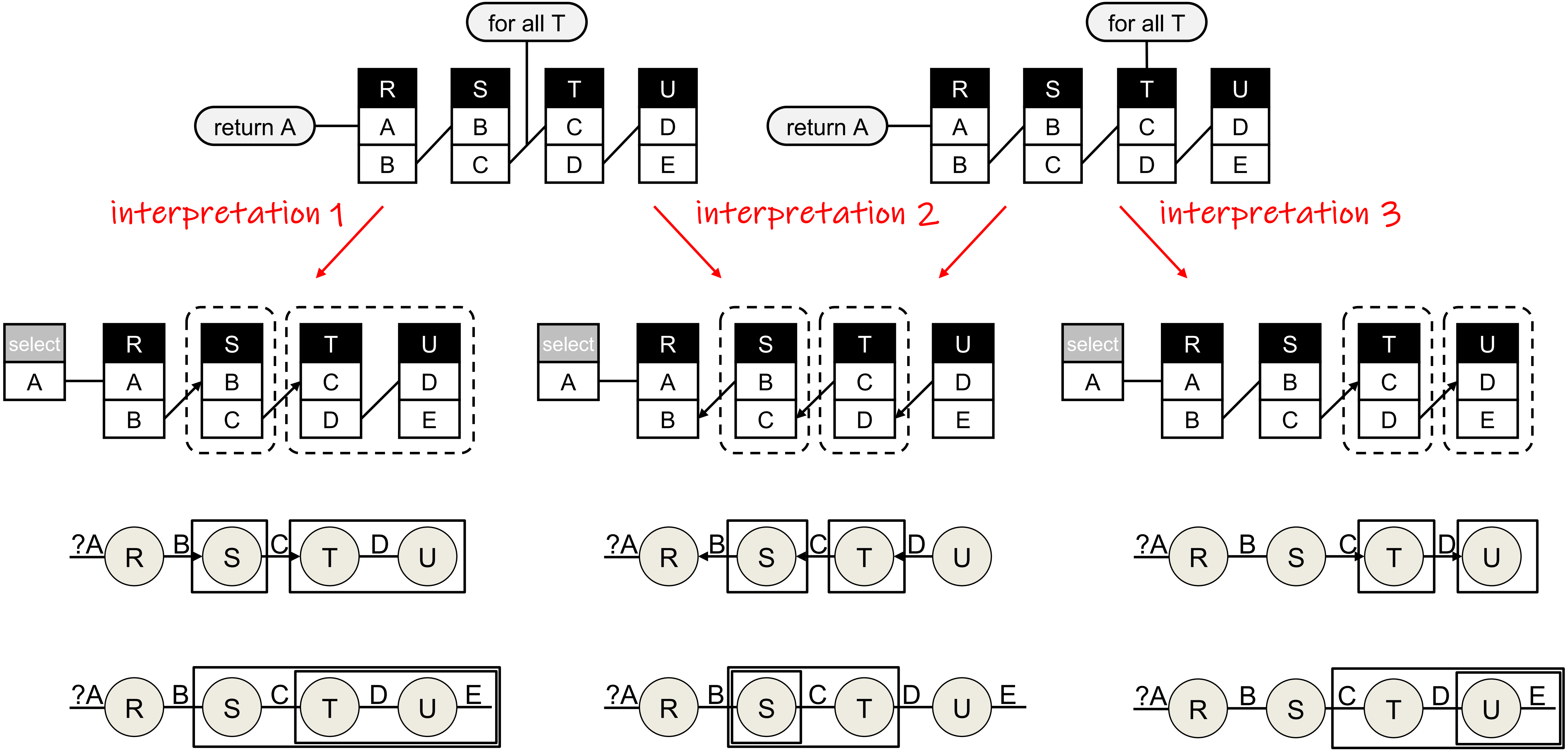




See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

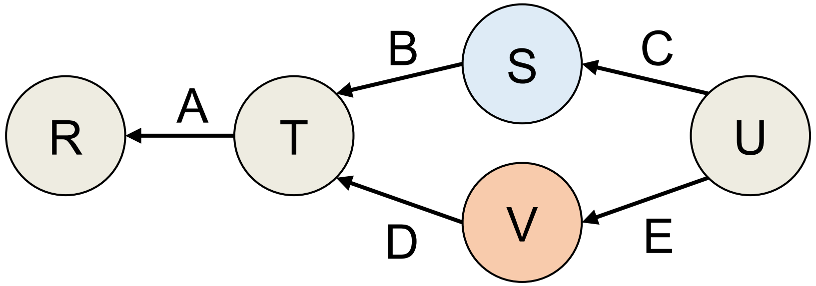
The quantifier example from OO-VQS is unambiguous



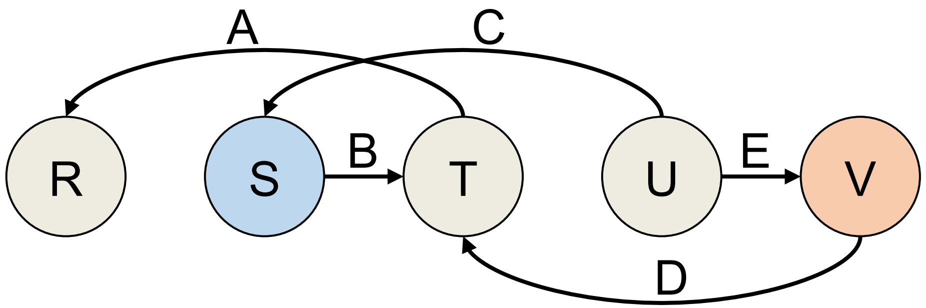
See example database "708" at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

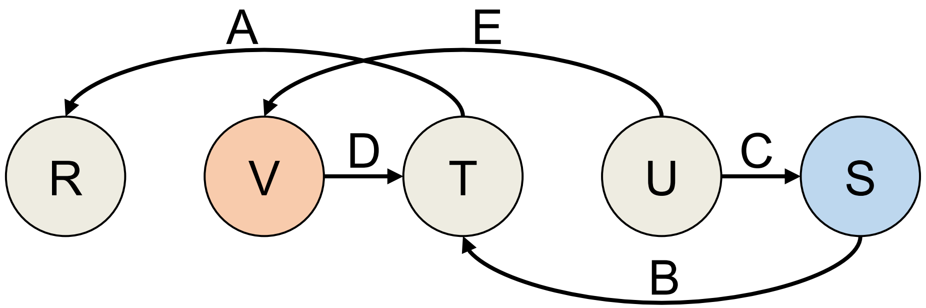
With nesting depth 4, arrows can become ambiguous

One visualization:



Two interpretations:

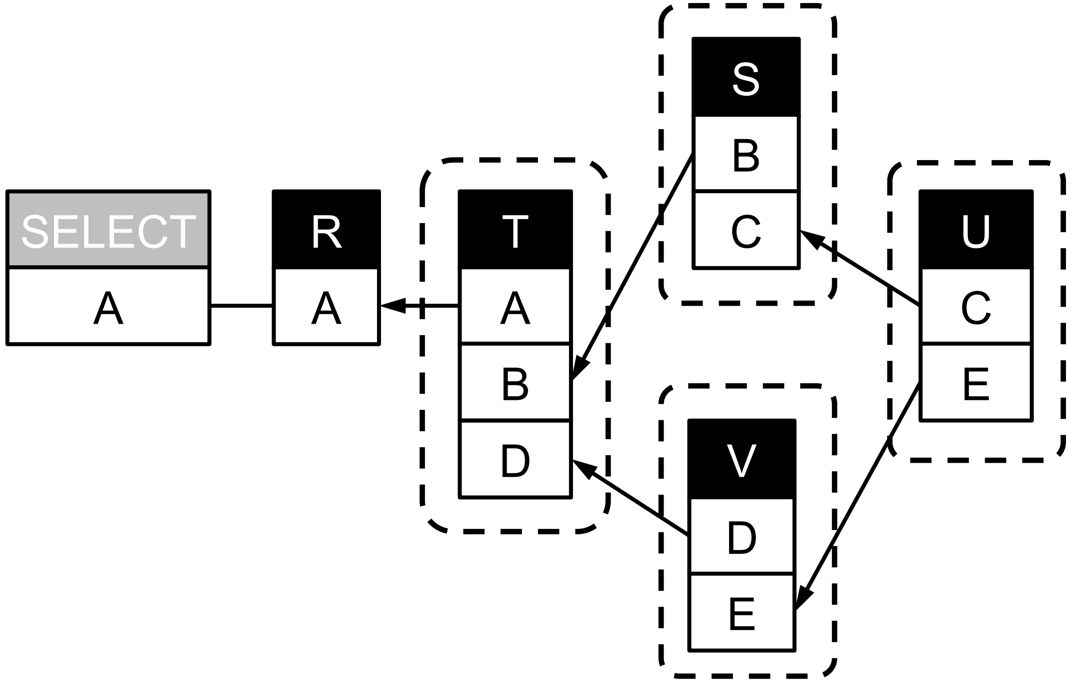




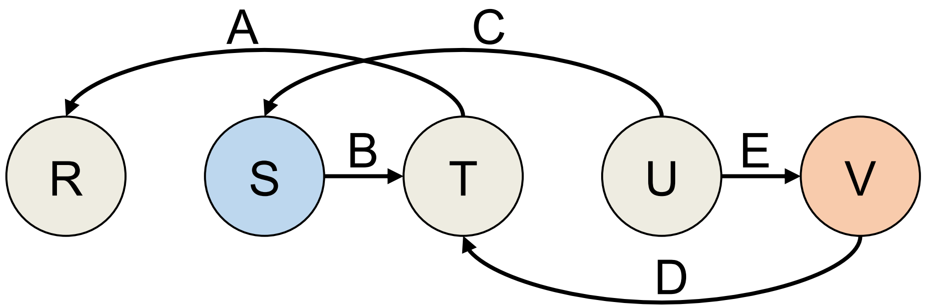
Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "

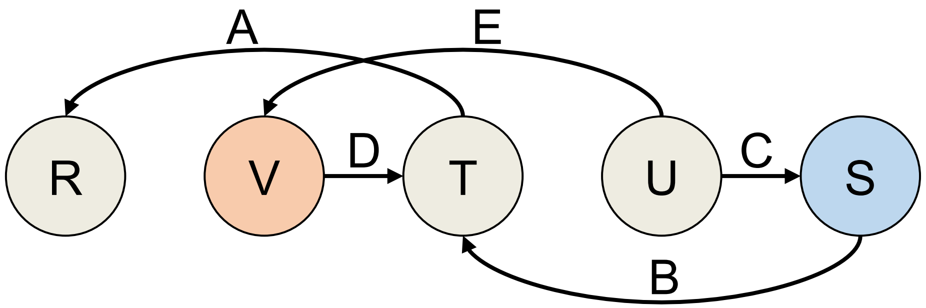
With nesting depth 4, arrows can become ambiguous

```
select distinct A
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 A
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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (2011)
(also QueryViz)
Backup

QueryVis (formerly known as QueryViz)

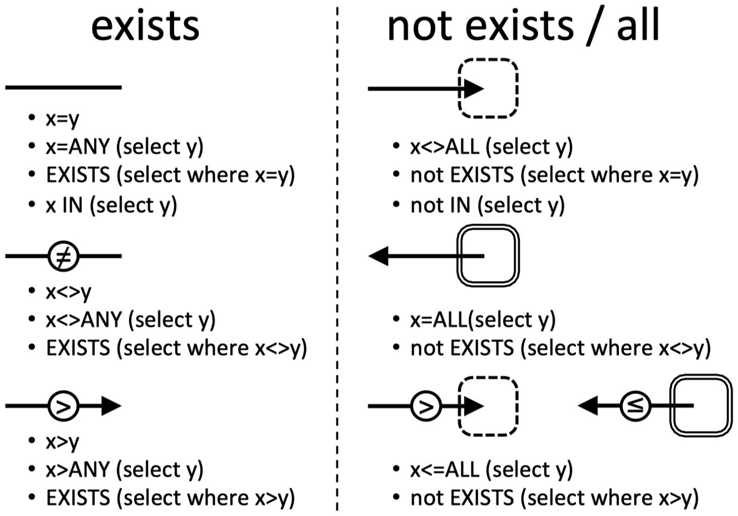


Figure 5: Lines with optional direction and comparison operators, together with bounding boxes in two line styles that group together relations suffice to express the most important syntactic constructs of nested SQL queries.

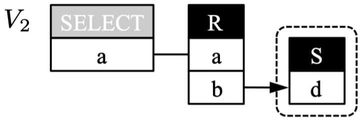
R(*a*,*b*,*c*), S(*d*,*e*)

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



$q_2: \; a : \exists b.\exists c.(R(a,b,c) \wedge \forall d.\forall e.(S(d,e) \Rightarrow d \neq b))$

$q_3: \; a : \exists b.\exists c.(R(a,b,c) \wedge b \notin \{d \mid \exists e.S(d,e)\})$

$q_4: \; a : \exists b.\exists c.(R(a,b,c) \wedge \neg\exists d.\exists e.(S(d,e) \wedge d=b))$

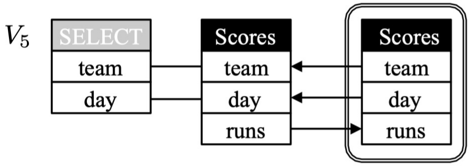
$V_2: \; a : \exists b.(R(a,b,\_) \wedge \neg(\exists d.(S(d,\_) \wedge d=b)))$

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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
```

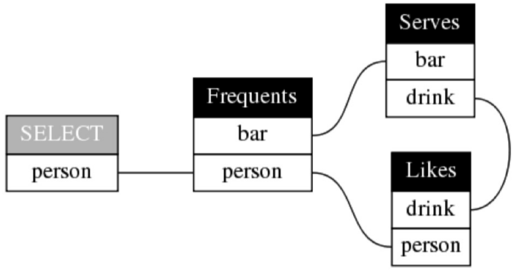


Figure 5: Visualizing a conjunctive query closely follows an all-familiar UML notation. ***Q: Find persons who frequent some bar that serves some drink they like.*** **There is nothing really new here.**

```
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))
```

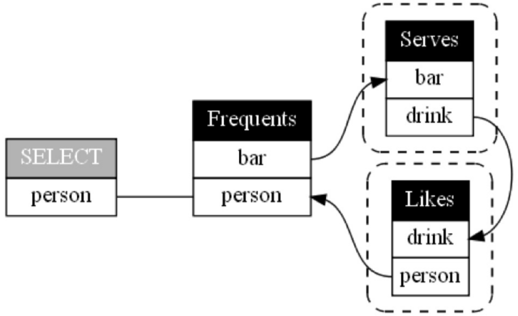


Figure 6: Visualizing a nested query still follows familiar UML notations, but now adds visual metaphors for $\nexists$ (dashed box) and reading order (arrows). ***Q: Find persons who frequent some bar that serves only drinks they like*** **$\equiv$** ***… some bar that serves no drink that is not liked by them.***

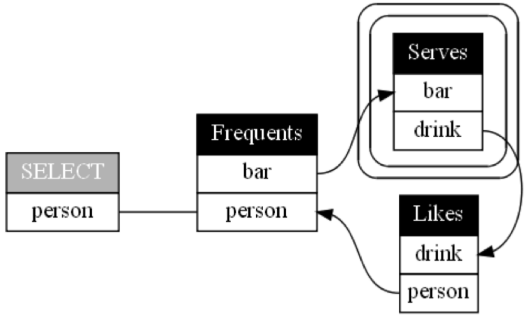
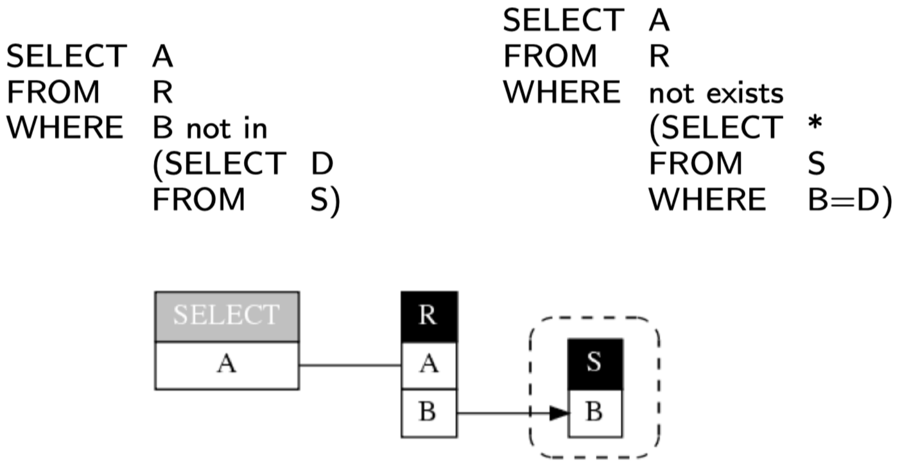


Figure 7: The visualization from Fig. 6 can be further simplified by using another visual metaphor for $\forall$ (double-lined box), a logical and intuitive operator that does not exist in SQL. ***Q: Find persons who frequent some bar that serves only drinks they like*** **$\equiv$** ***… some bar so that all drinks served are liked by them.***

Figure source: "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805 "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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 "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

QueryVis (formerly known as QueryViz)

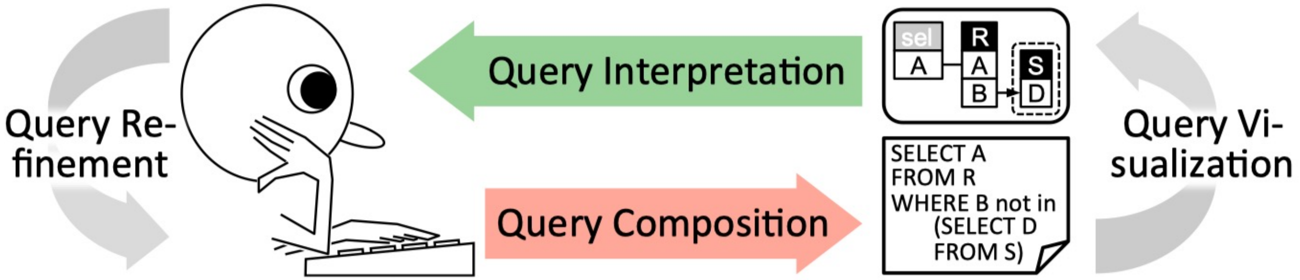
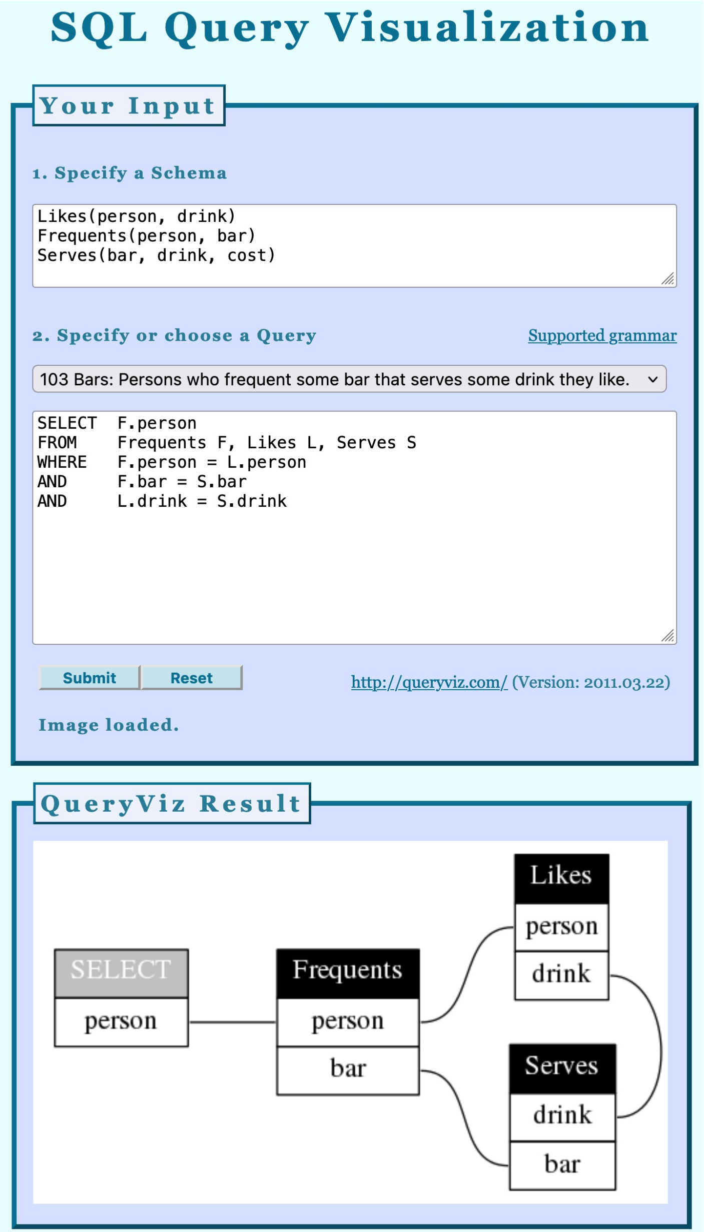


Figure 8: The vision: In the near future, *DBMSs will visualize queries too*, and not just data (as in information and scientific data visualization). This feature will allow iterative query refinement and will enhance the usability of databases.

Figure source: "Gatterbauer. Databases will Visualize Queries too. PVLDB vision 2011. https://doi.org/10.14778/3402755.3402805 "

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query Supported grammar

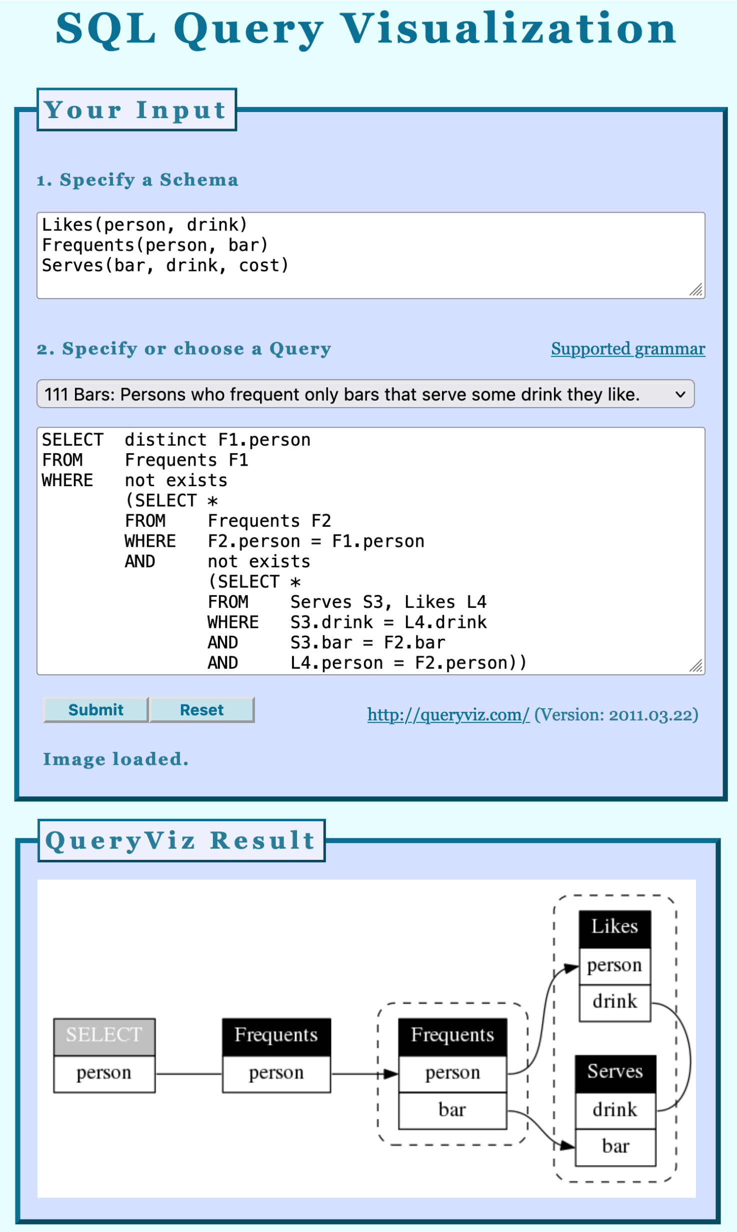
103 Bars: Persons who frequent some bar that serves some drink they like.

```
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
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query Supported grammar

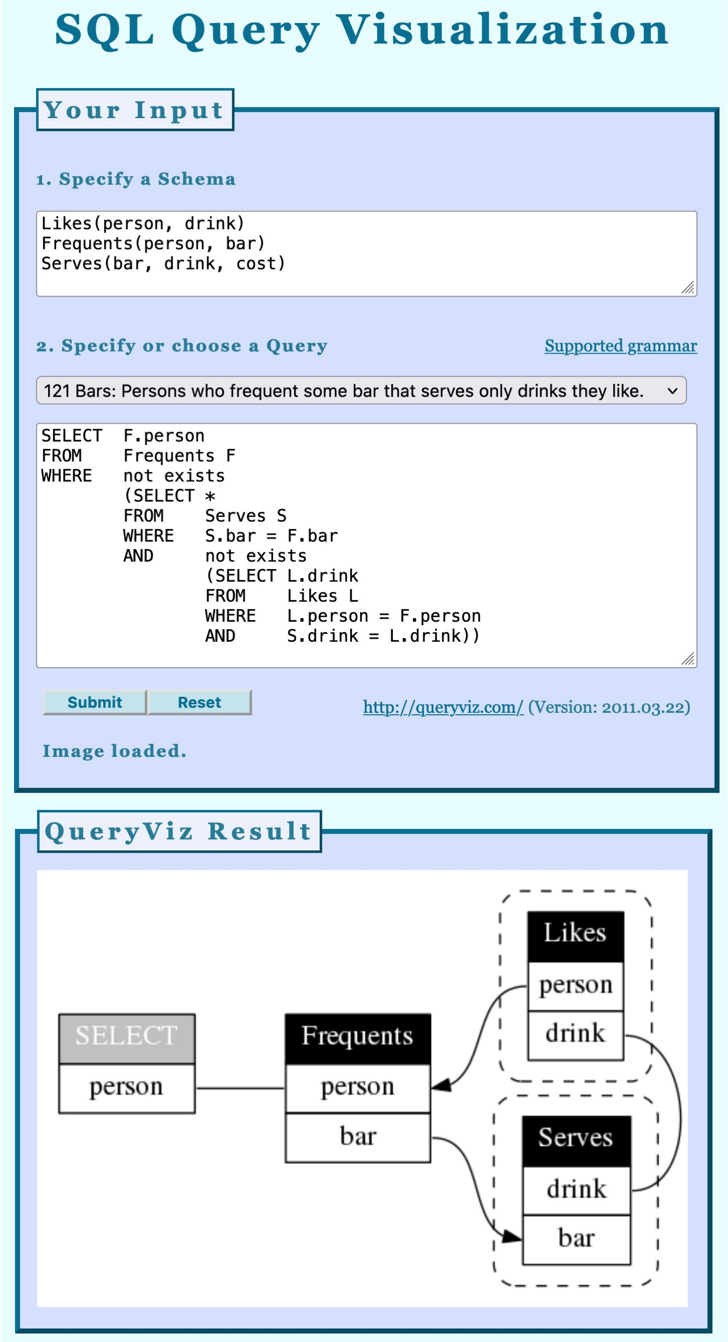
111 Bars: Persons who frequent only bars that serve some drink they like.

```
SELECT  distinct F1.person
FROM    Frequents F1
WHERE   not exists
        (SELECT *
        FROM    Frequents F2
        WHERE   F2.person = F1.person
        AND     not exists
                (SELECT *
                FROM    Serves S3, Likes L4
                WHERE   S3.drink = L4.drink
                AND     S3.bar = F2.bar
                AND     L4.person = F2.person))
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query Supported grammar

121 Bars: Persons who frequent some bar that serves only drinks they like.

```
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))
```

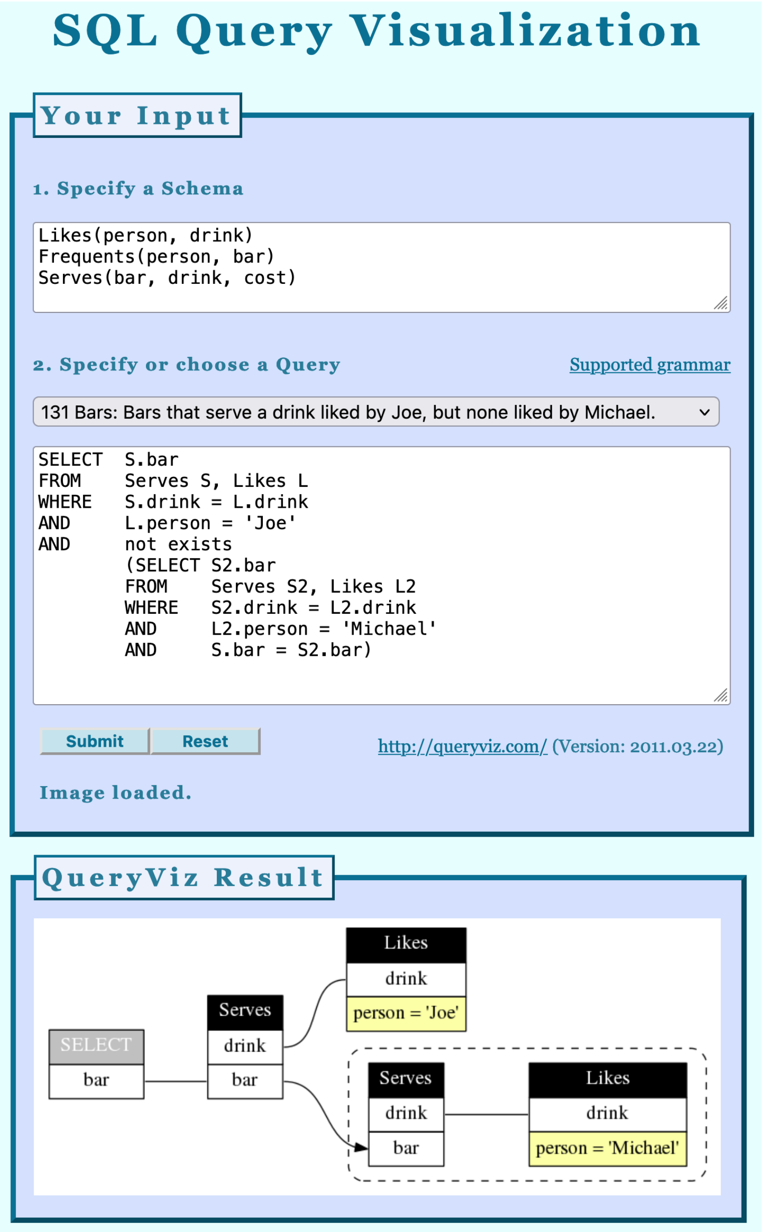
http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

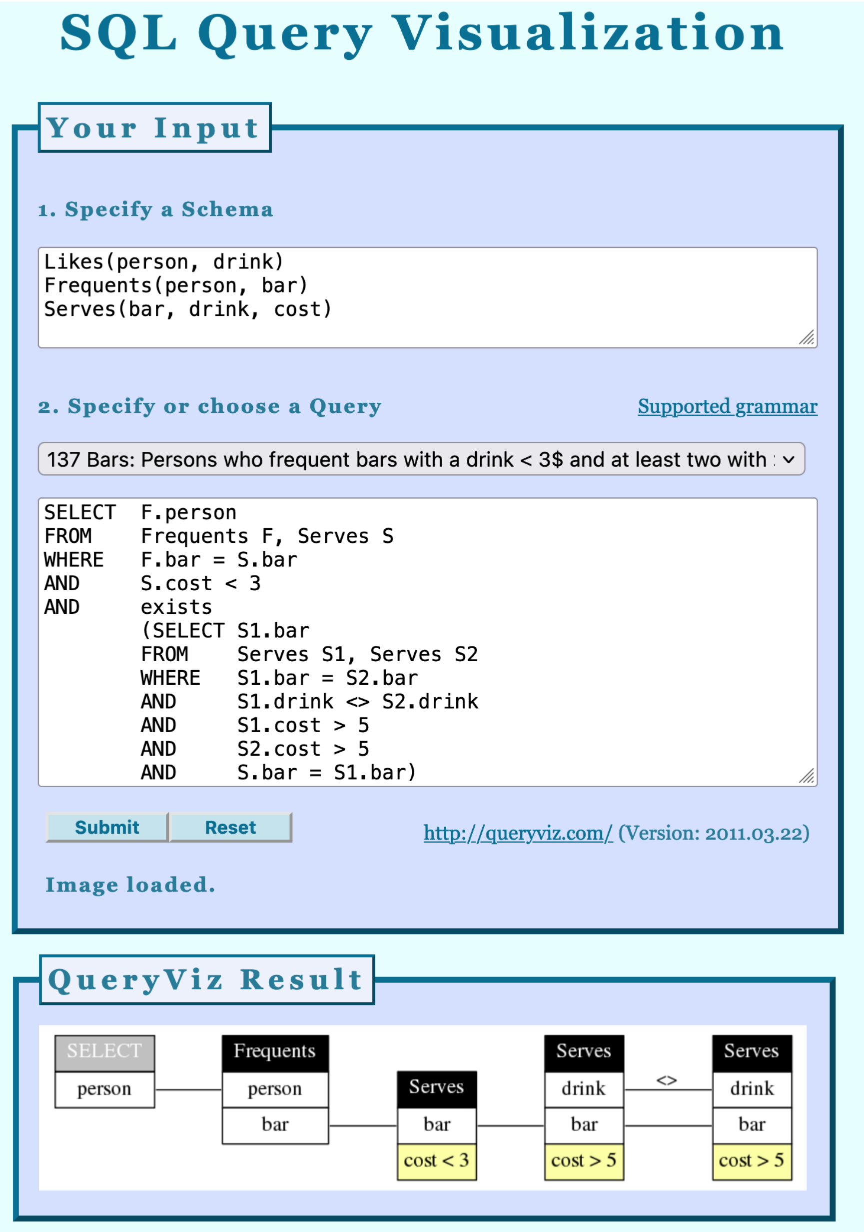
Online QueryVis



```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

131 Bars: Bars that serve a drink liked by Joe, but none liked by Michael.

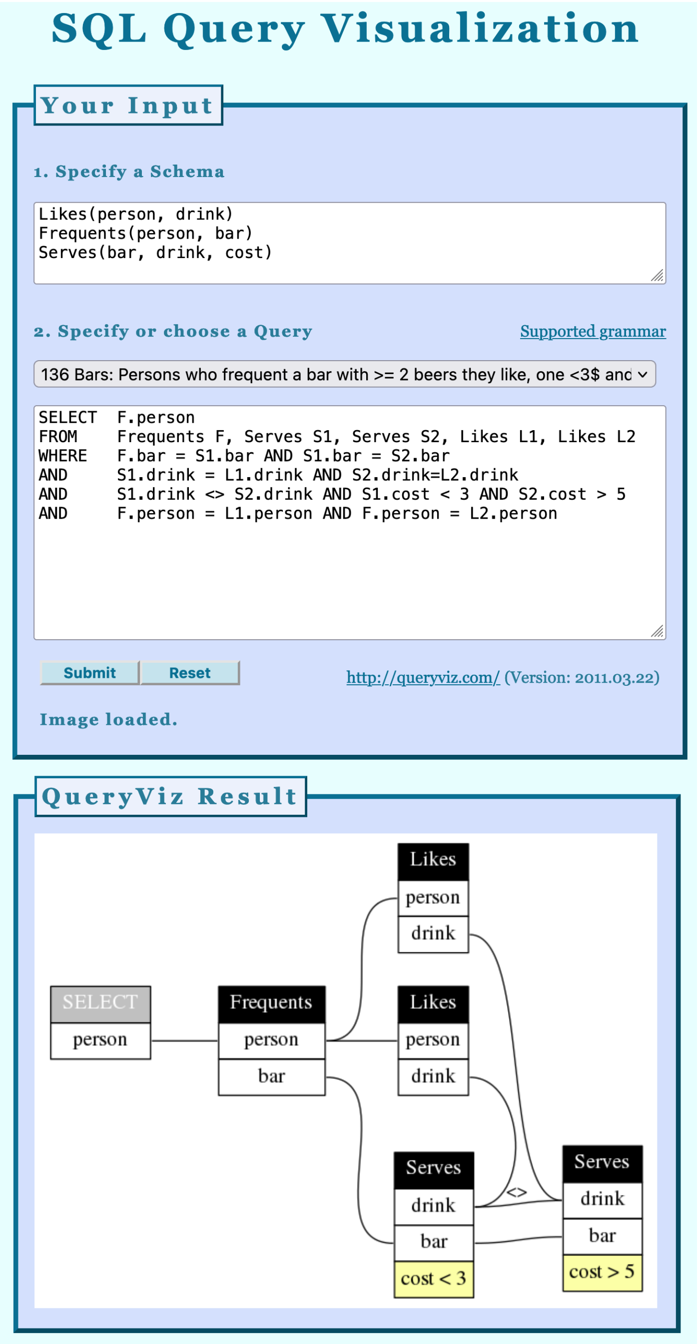
```
SELECT  S.bar
FROM    Serves S, Likes L
WHERE   S.drink = L.drink
AND     L.person = 'Joe'
AND     not exists
        (SELECT S2.bar
        FROM    Serves S2, Likes L2
        WHERE   S2.drink = L2.drink
        AND     L2.person = 'Michael'
        AND     S.bar = S2.bar)
```



```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

137 Bars: Persons who frequent bars with a drink < 3$ and at least two with :

```
SELECT  F.person
FROM    Frequents F, Serves S
WHERE   F.bar = S.bar
AND     S.cost < 3
AND     exists
        (SELECT S1.bar
        FROM    Serves S1, Serves S2
        WHERE   S1.bar = S2.bar
        AND     S1.drink <> S2.drink
        AND     S1.cost > 5
        AND     S2.cost > 5
        AND     S.bar = S1.bar)
```



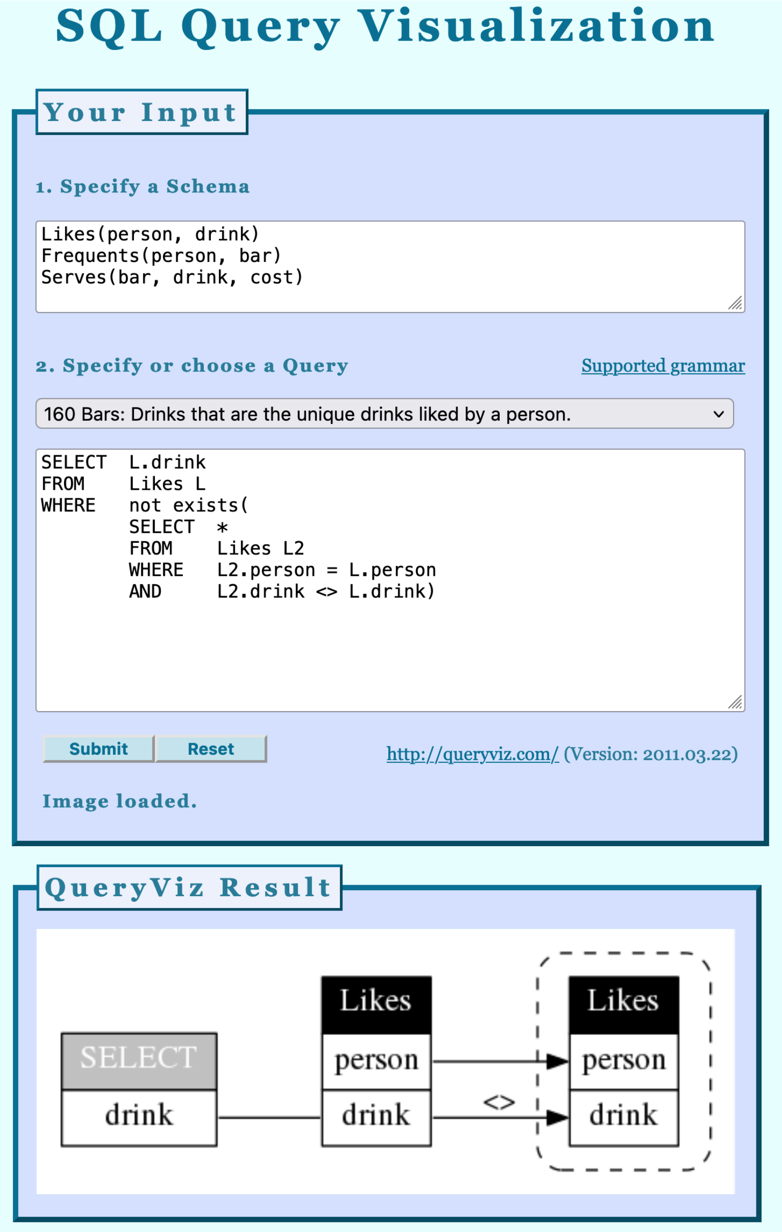
```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

136 Bars: Persons who frequent a bar with >= 2 beers they like, one <3$ anc

```
SELECT  F.person
FROM    Frequents F, Serves S1, Serves S2, Likes L1, Likes L2
WHERE   F.bar = S1.bar AND S1.bar = S2.bar
AND     S1.drink = L1.drink AND S2.drink=L2.drink
AND     S1.drink <> S2.drink AND S1.cost < 3 AND S2.cost > 5
AND     F.person = L1.person AND F.person = L2.person
```

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

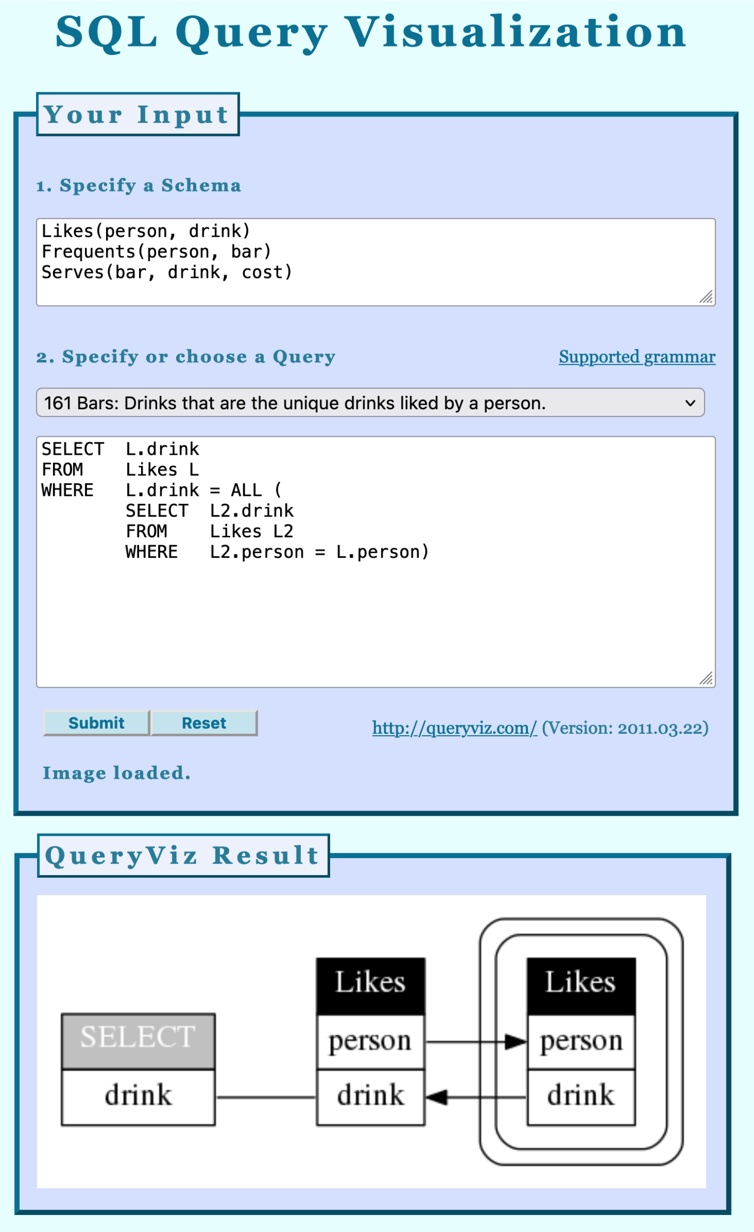
2. Specify or choose a Query

160 Bars: Drinks that are the unique drinks liked by a person.

```
SELECT  L.drink
FROM    Likes L
WHERE   not exists(
        SELECT  *
        FROM    Likes L2
        WHERE   L2.person = L.person
        AND     L2.drink <> L.drink)
```

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query

161 Bars: Drinks that are the unique drinks liked by a person.

```
SELECT  L.drink
FROM    Likes L
WHERE   L.drink = ALL (
        SELECT  L2.drink
        FROM    Likes L2
        WHERE   L2.person = L.person)
```

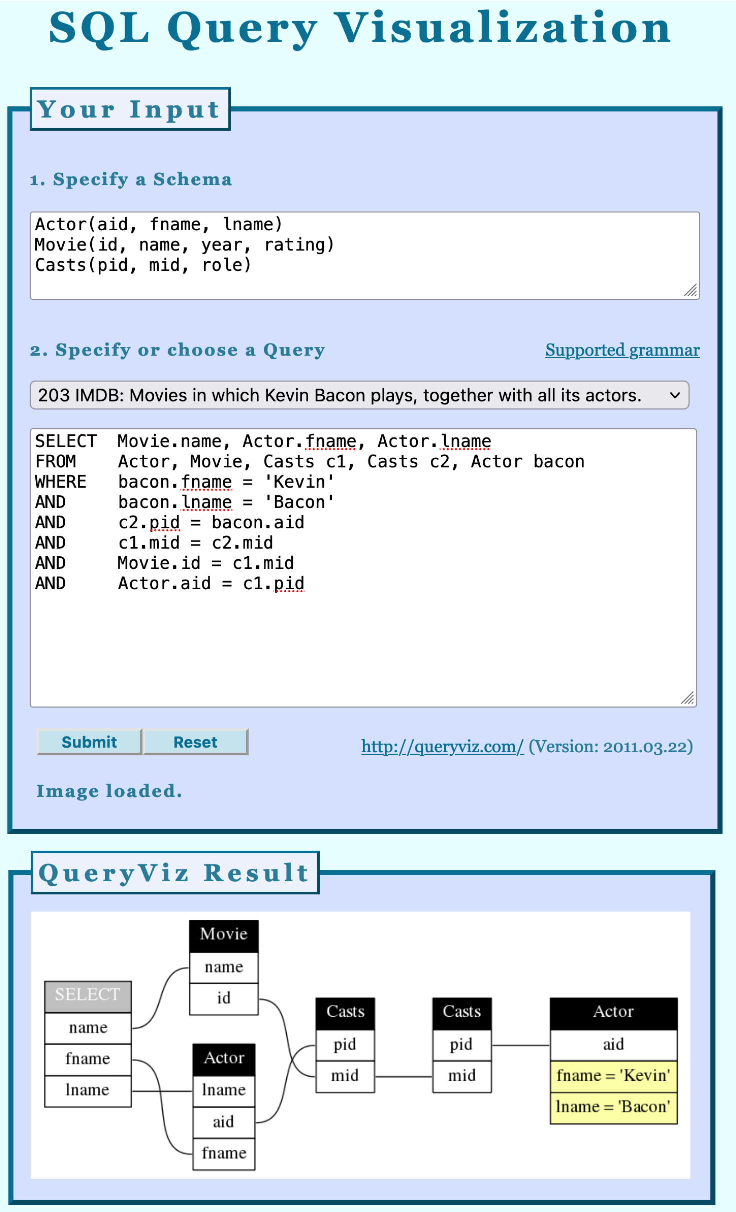
Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Online QueryVis



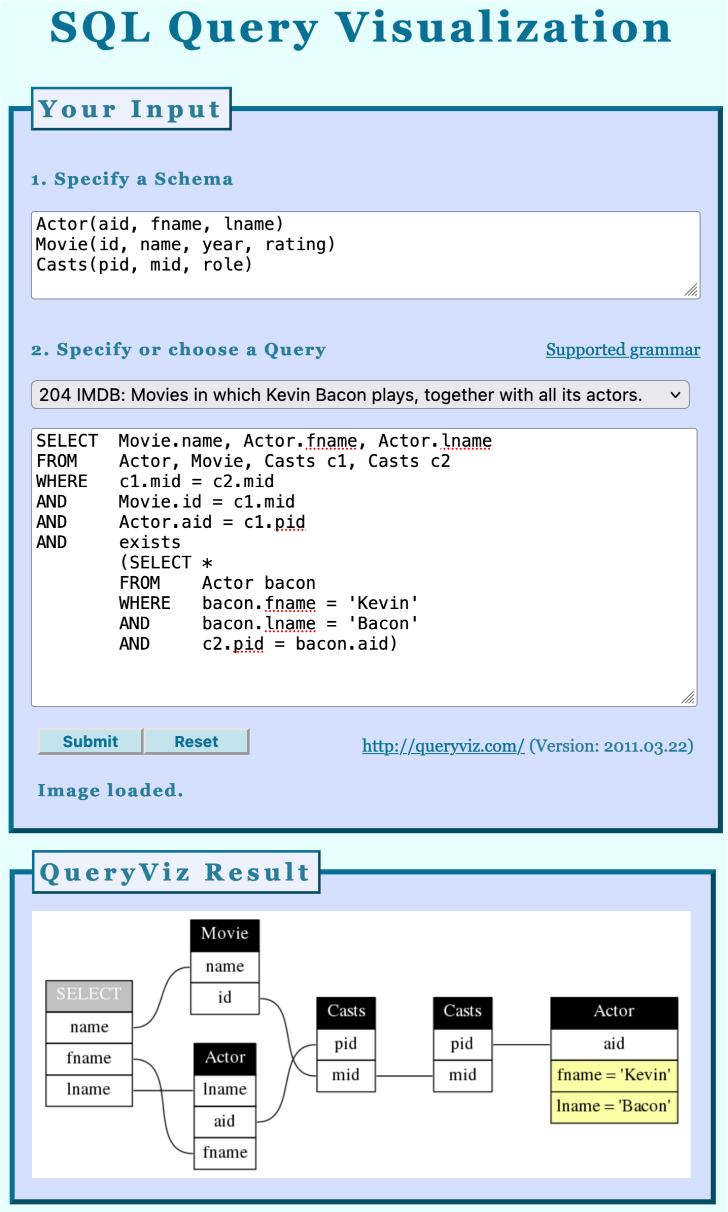
```
Actor(aid, fname, lname)
Movie(id, name, year, rating)
Casts(pid, mid, role)
```

203 IMDB: Movies in which Kevin Bacon plays, together with all its actors.

```
SELECT  Movie.name, Actor.fname, Actor.lname
FROM    Actor, Movie, Casts c1, Casts c2, Actor bacon
WHERE   bacon.fname = 'Kevin'
AND     bacon.lname = 'Bacon'
AND     c2.pid = bacon.aid
AND     c1.mid = c2.mid
AND     Movie.id = c1.mid
AND     Actor.aid = c1.pid
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.



```
Actor(aid, fname, lname)
Movie(id, name, year, rating)
Casts(pid, mid, role)
```

204 IMDB: Movies in which Kevin Bacon plays, together with all its actors.

```
SELECT  Movie.name, Actor.fname, Actor.lname
FROM    Actor, Movie, Casts c1, Casts c2
WHERE   c1.mid = c2.mid
AND     Movie.id = c1.mid
AND     Actor.aid = c1.pid
AND     exists
        (SELECT *
        FROM    Actor bacon
        WHERE   bacon.fname = 'Kevin'
        AND     bacon.lname = 'Bacon'
        AND     c2.pid = bacon.aid)
```

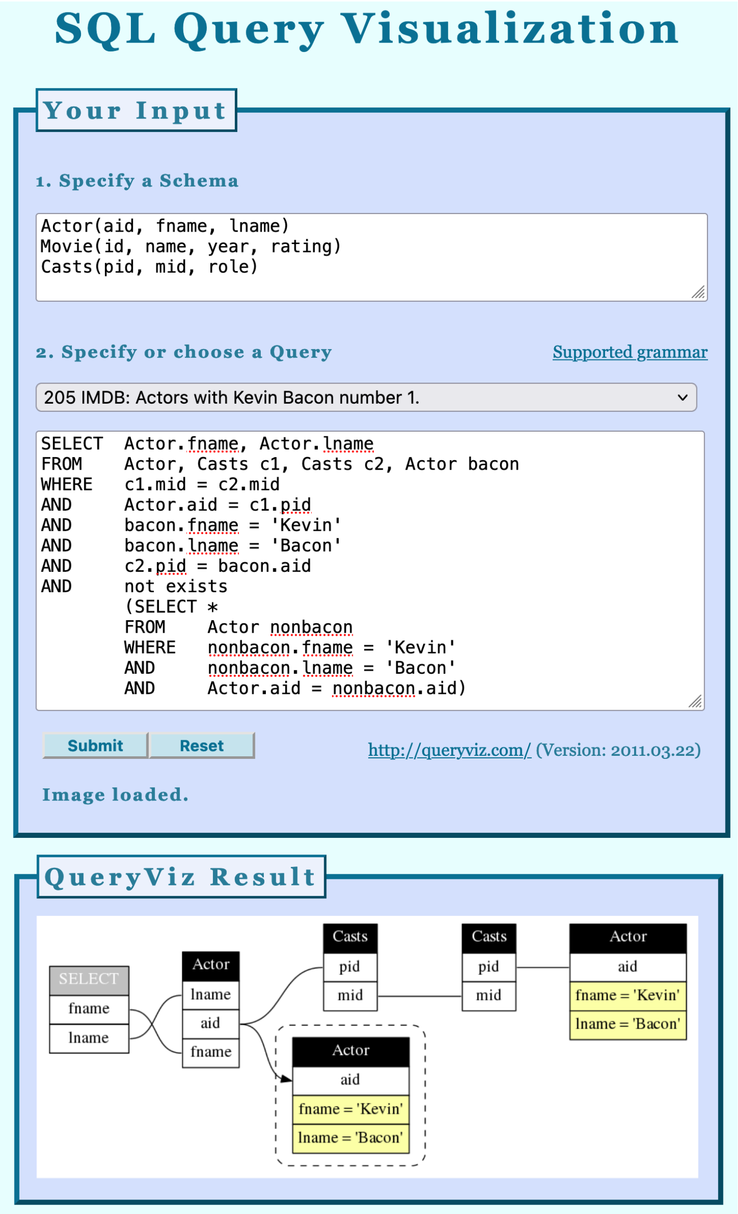
http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
Actor(aid, fname, lname)
Movie(id, name, year, rating)
Casts(pid, mid, role)
```

2. Specify or choose a Query

Supported grammar

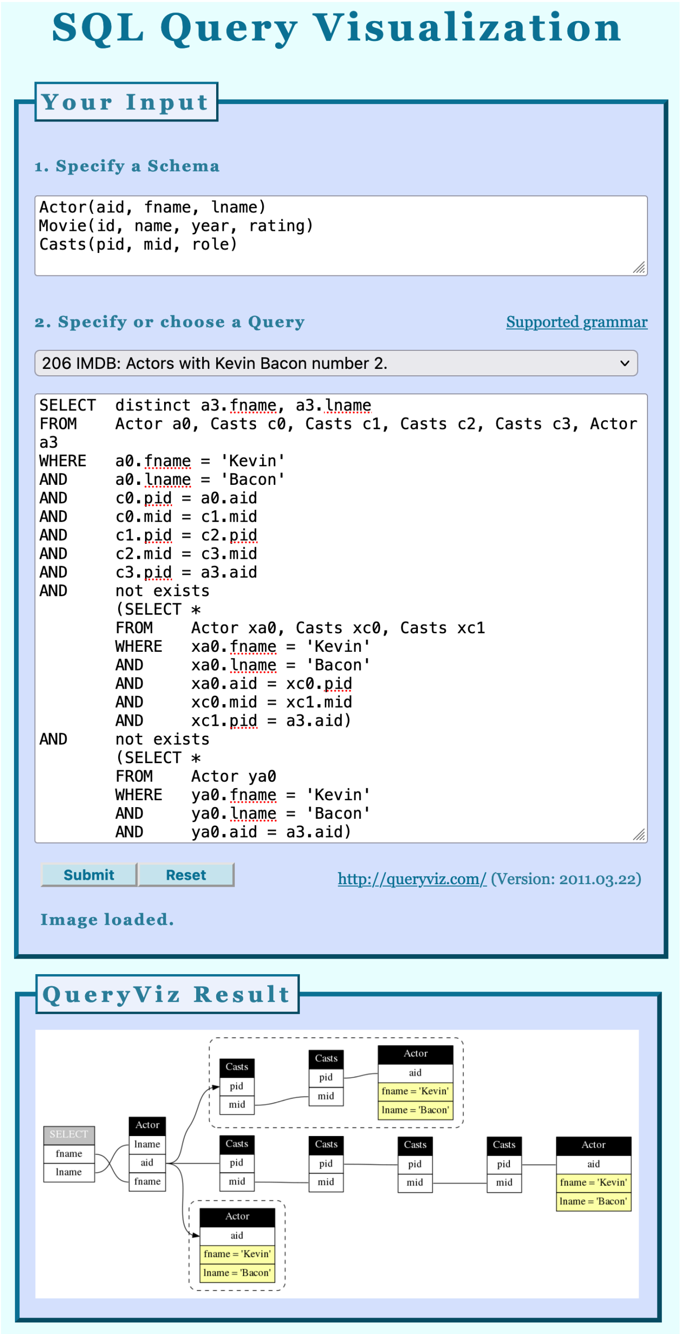
205 IMDB: Actors with Kevin Bacon number 1.

```
SELECT  Actor.fname, Actor.lname
FROM    Actor, Casts c1, Casts c2, Actor bacon
WHERE   c1.mid = c2.mid
AND     Actor.aid = c1.pid
AND     bacon.fname = 'Kevin'
AND     bacon.lname = 'Bacon'
AND     c2.pid = bacon.aid
AND     not exists
        (SELECT *
        FROM    Actor nonbacon
        WHERE   nonbacon.fname = 'Kevin'
        AND     nonbacon.lname = 'Bacon'
        AND     Actor.aid = nonbacon.aid)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Actor(aid, fname, lname)
Movie(id, name, year, rating)
Casts(pid, mid, role)
```

2. Specify or choose a Query

Supported grammar

206 IMDB: Actors with Kevin Bacon number 2.

```
SELECT  distinct a3.fname, a3.lname
FROM    Actor a0, Casts c0, Casts c1, Casts c2, Casts c3, Actor
a3
WHERE   a0.fname = 'Kevin'
AND     a0.lname = 'Bacon'
AND     c0.pid = a0.aid
AND     c0.mid = c1.mid
AND     c1.pid = c2.pid
AND     c2.mid = c3.mid
AND     c3.pid = a3.aid
AND     not exists
        (SELECT *
        FROM    Actor xa0, Casts xc0, Casts xc1
        WHERE   xa0.fname = 'Kevin'
        AND     xa0.lname = 'Bacon'
        AND     xa0.aid = xc0.pid
        AND     xc0.mid = xc1.mid
        AND     xc1.pid = a3.aid)
AND     not exists
        (SELECT *
        FROM    Actor ya0
        WHERE   ya0.fname = 'Kevin'
        AND     ya0.lname = 'Bacon'
        AND     ya0.aid = a3.aid)
```

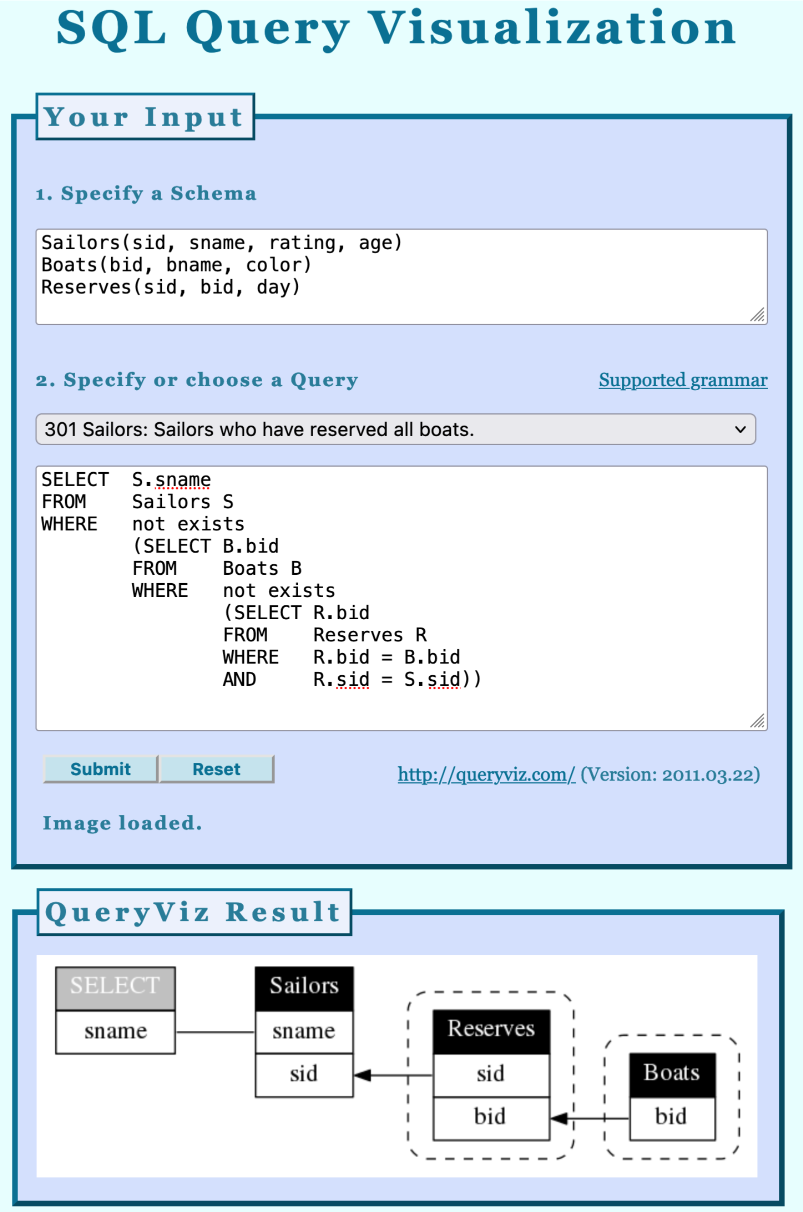
http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)
```

2. Specify or choose a Query Supported grammar

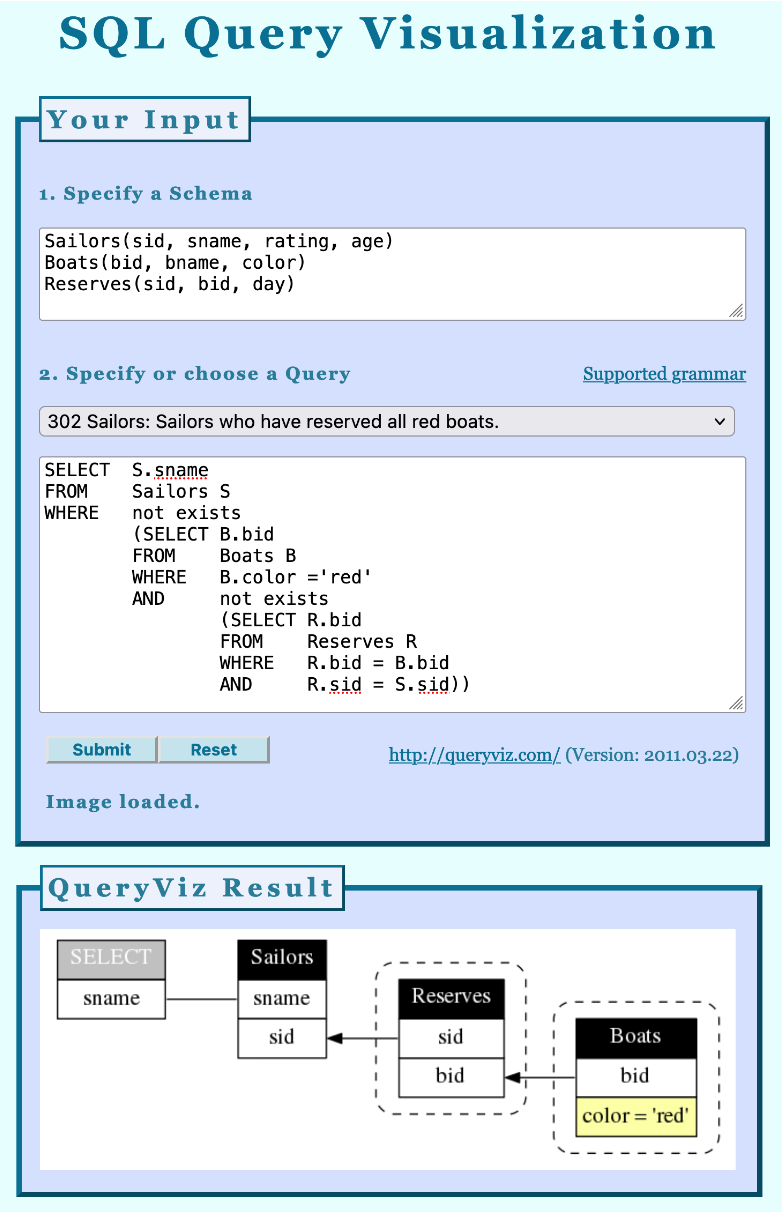
301 Sailors: Sailors who have reserved all boats.

```
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))
```

Submit Reset http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)
```

2. Specify or choose a Query Supported grammar

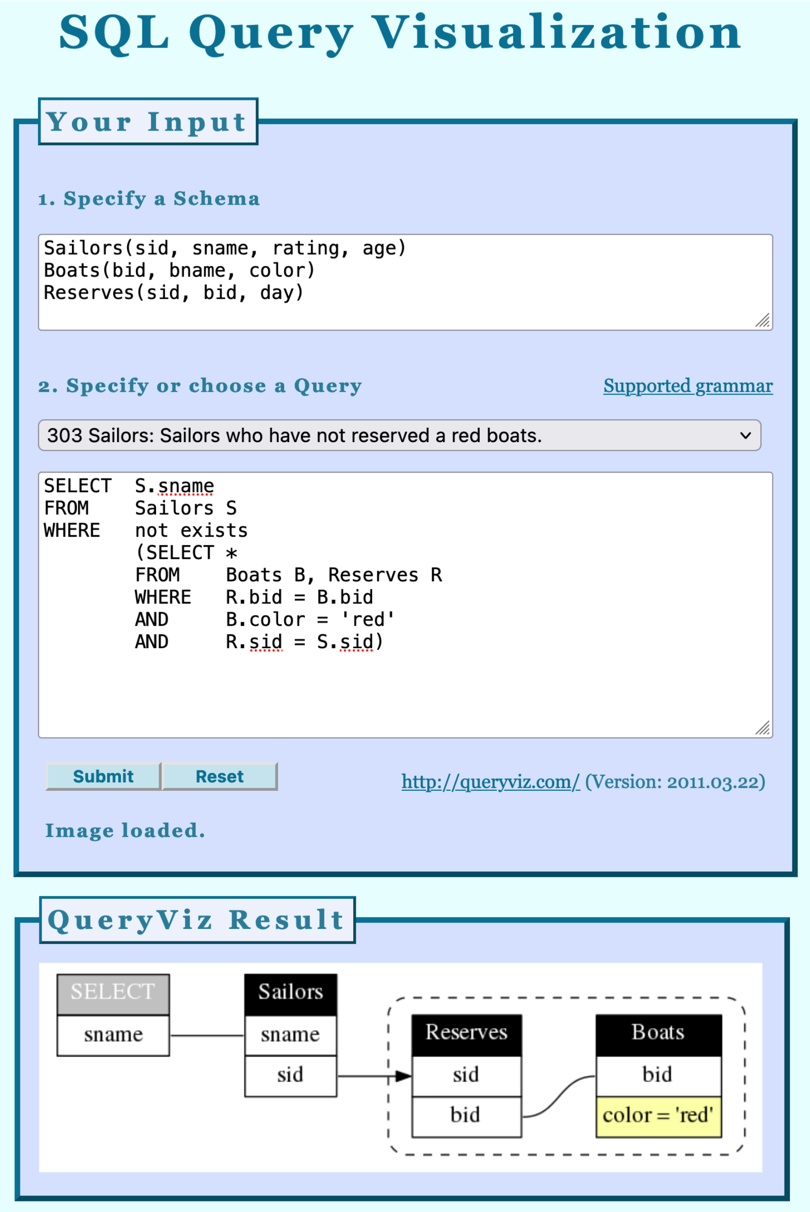
302 Sailors: Sailors who have reserved all red boats.

```
SELECT  S.sname
FROM    Sailors S
WHERE   not exists
        (SELECT B.bid
        FROM    Boats B
        WHERE   B.color ='red'
        AND     not exists
                (SELECT R.bid
                FROM    Reserves R
                WHERE   R.bid = B.bid
                AND     R.sid = S.sid))
```

Submit Reset http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)
```

2. Specify or choose a Query Supported grammar

303 Sailors: Sailors who have not reserved a red boats.

```
SELECT  S.sname
FROM    Sailors S
WHERE   not exists
        (SELECT *
        FROM    Boats B, Reserves R
        WHERE   R.bid = B.bid
        AND     B.color = 'red'
        AND     R.sid = S.sid)
```

Submit Reset http://queryviz.com/ (Version: 2011.03.22)

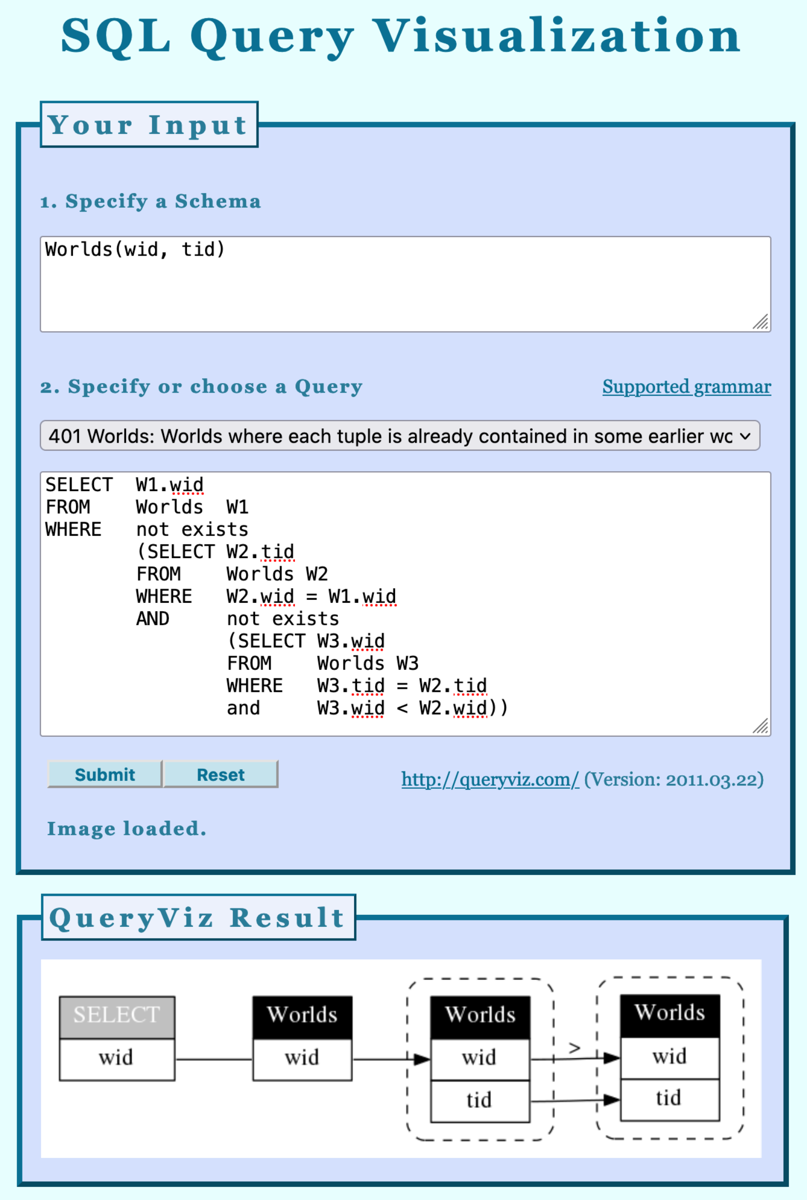
Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

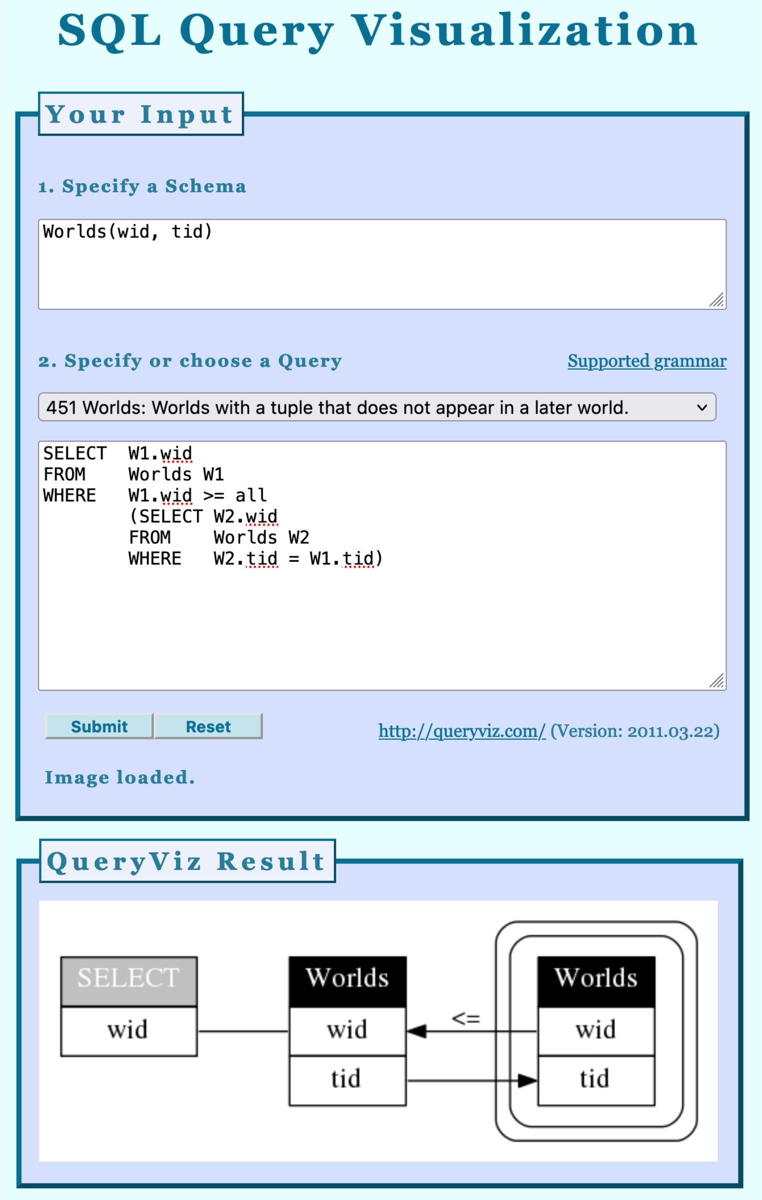
Online QueryVis



```
Worlds(wid, tid)
```

401 Worlds: Worlds where each tuple is already contained in some earlier wc

```
SELECT  W1.wid
FROM    Worlds  W1
WHERE   not exists
        (SELECT W2.tid
        FROM    Worlds W2
        WHERE   W2.wid = W1.wid
        AND     not exists
                (SELECT W3.wid
                FROM    Worlds W3
                WHERE   W3.tid = W2.tid
                and     W3.wid < W2.wid))
```



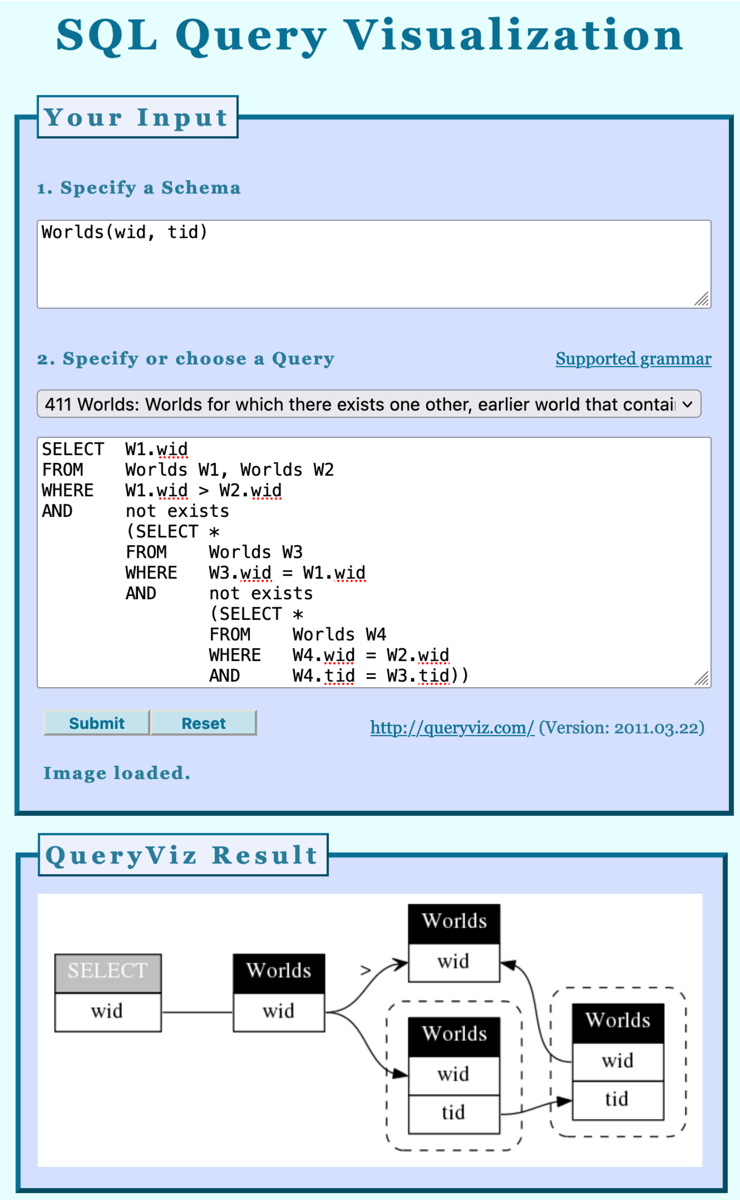
```
Worlds(wid, tid)
```

451 Worlds: Worlds with a tuple that does not appear in a later world.

```
SELECT  W1.wid
FROM    Worlds W1
WHERE   W1.wid >= all
        (SELECT W2.wid
        FROM    Worlds W2
        WHERE   W2.tid = W1.tid)
```

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
Worlds(wid, tid)
```

2. Specify or choose a Query Supported grammar

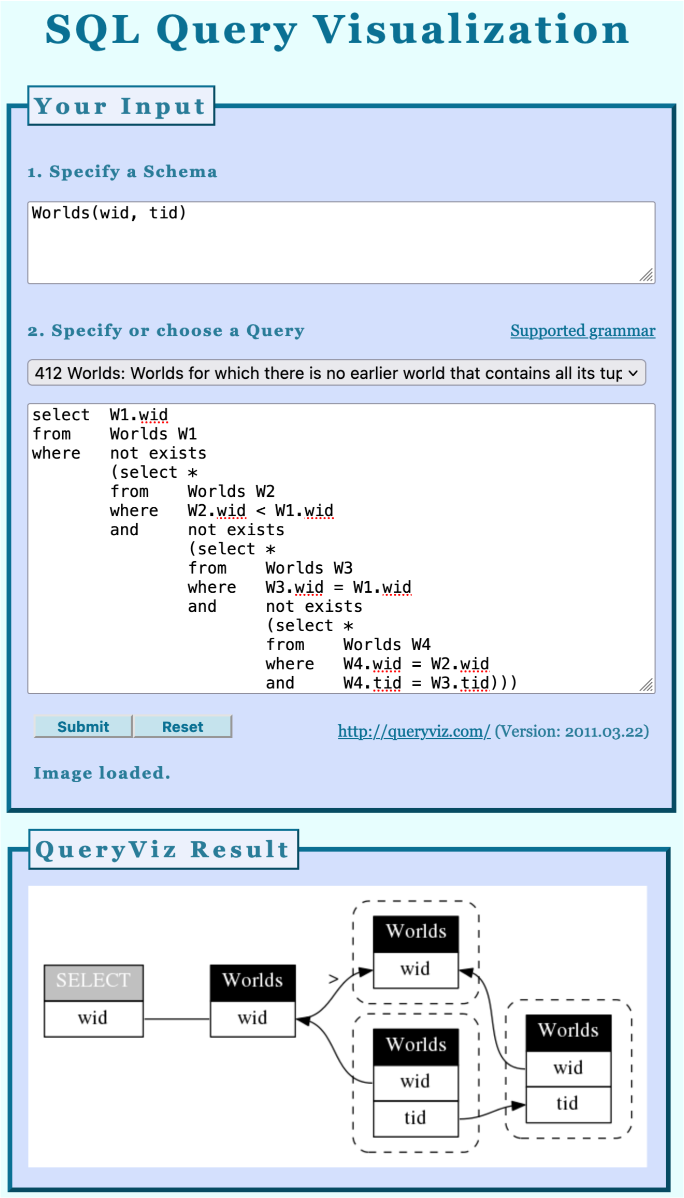
411 Worlds: Worlds for which there exists one other, earlier world that contai

```
SELECT  W1.wid
FROM    Worlds W1, Worlds W2
WHERE   W1.wid > W2.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))
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Worlds(wid, tid)
```

2. Specify or choose a Query Supported grammar

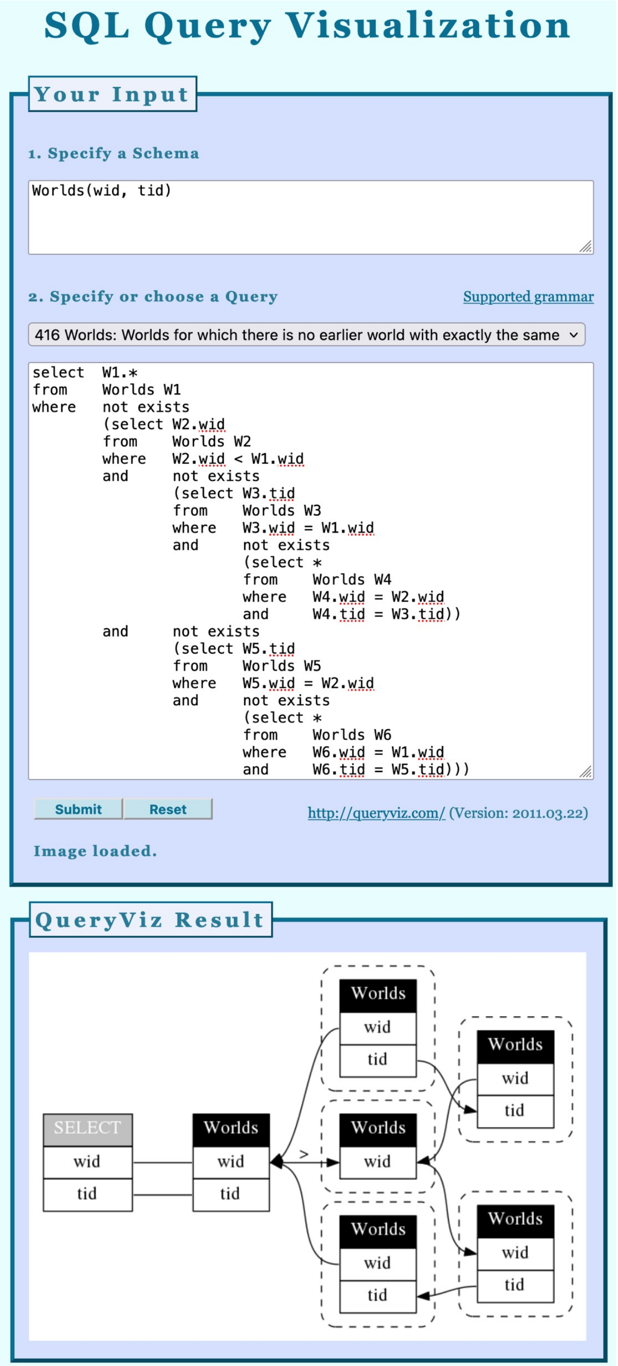
412 Worlds: Worlds for which there is no earlier world that contains all its tup

```
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)))
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
Worlds(wid, tid)
```

2. Specify or choose a Query Supported grammar

416 Worlds: Worlds for which there is no earlier world with exactly the same

```
select  W1.*
from    Worlds W1
where   not exists
        (select W2.wid
        from    Worlds W2
        where   W2.wid < W1.wid
        and     not exists
                (select W3.tid
                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))
        and     not exists
                (select W5.tid
                from    Worlds W5
                where   W5.wid = W2.wid
                and     not exists
                        (select *
                        from    Worlds W6
                        where   W6.wid = W1.wid
                        and     W6.tid = W5.tid)))
```

http://queryviz.com/ (Version: 2011.03.22)

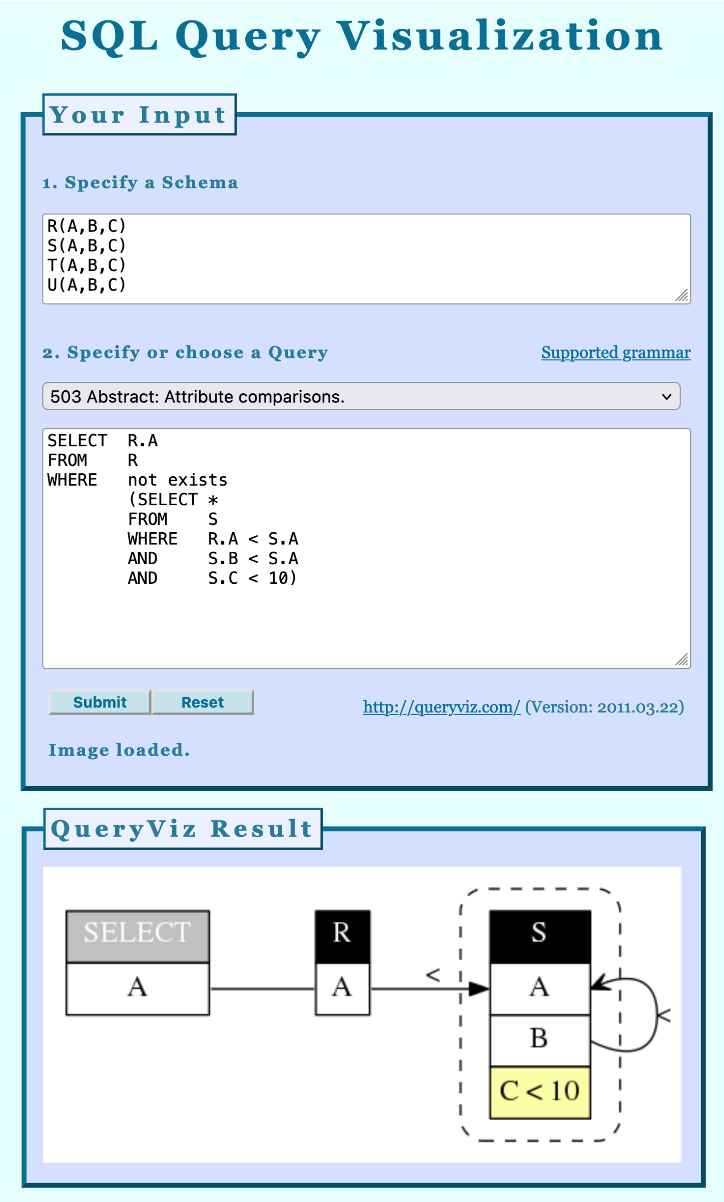
Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Online QueryVis



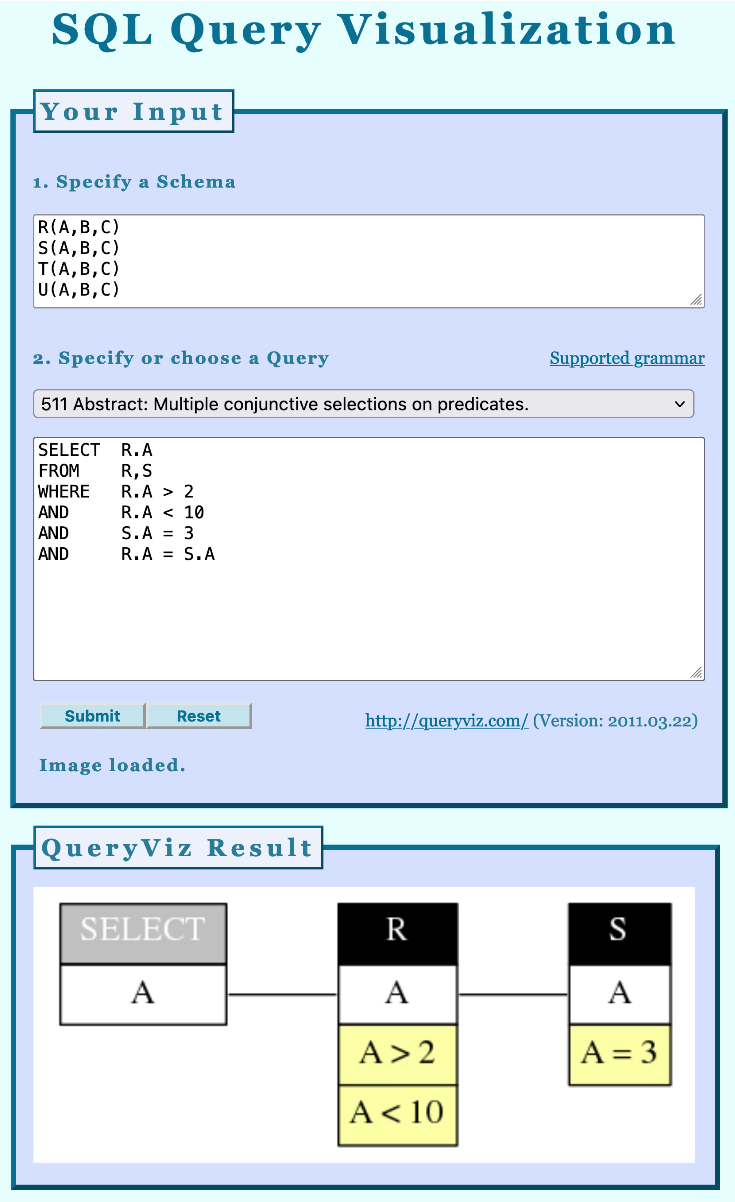
```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

503 Abstract: Attribute comparisons.

```
SELECT  R.A
FROM    R
WHERE   not exists
        (SELECT *
        FROM    S
        WHERE   R.A < S.A
        AND     S.B < S.A
        AND     S.C < 10)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.



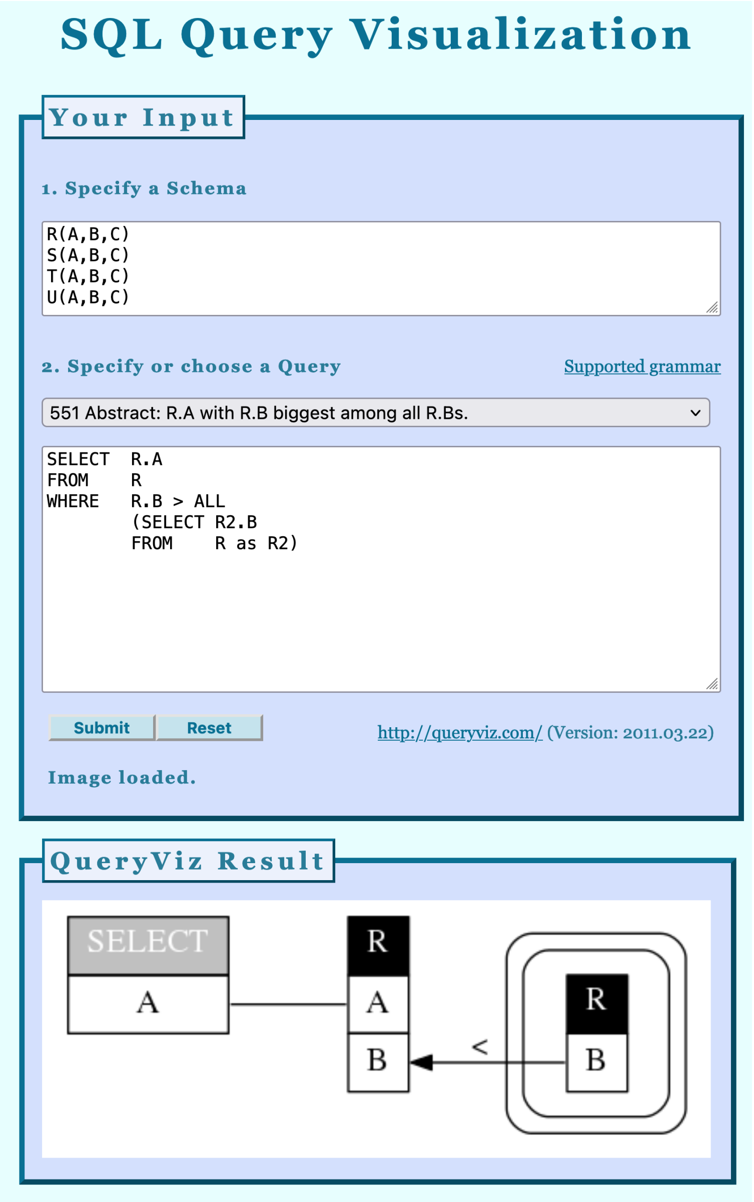
```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

511 Abstract: Multiple conjunctive selections on predicates.

```
SELECT  R.A
FROM    R,S
WHERE   R.A > 2
AND     R.A < 10
AND     S.A = 3
AND     R.A = S.A
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.



```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

551 Abstract: R.A with R.B biggest among all R.Bs.

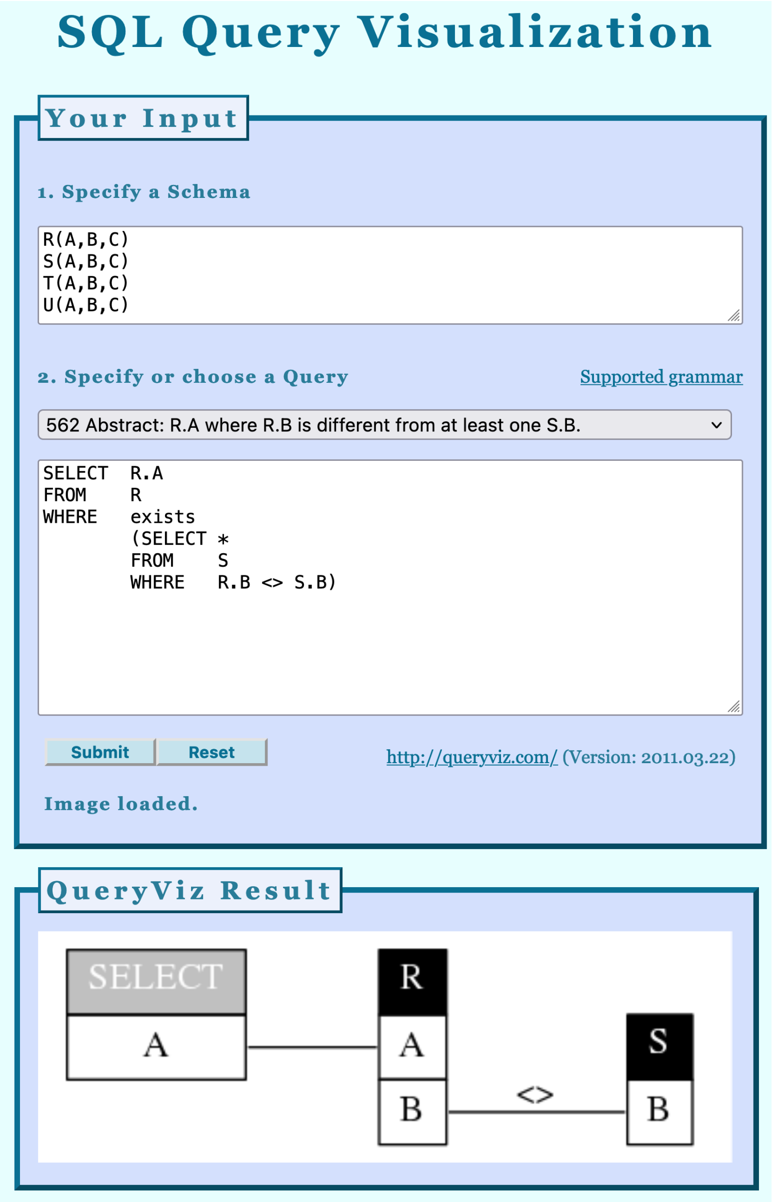
```
SELECT  R.A
FROM    R
WHERE   R.B > ALL
        (SELECT R2.B
        FROM    R as R2)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

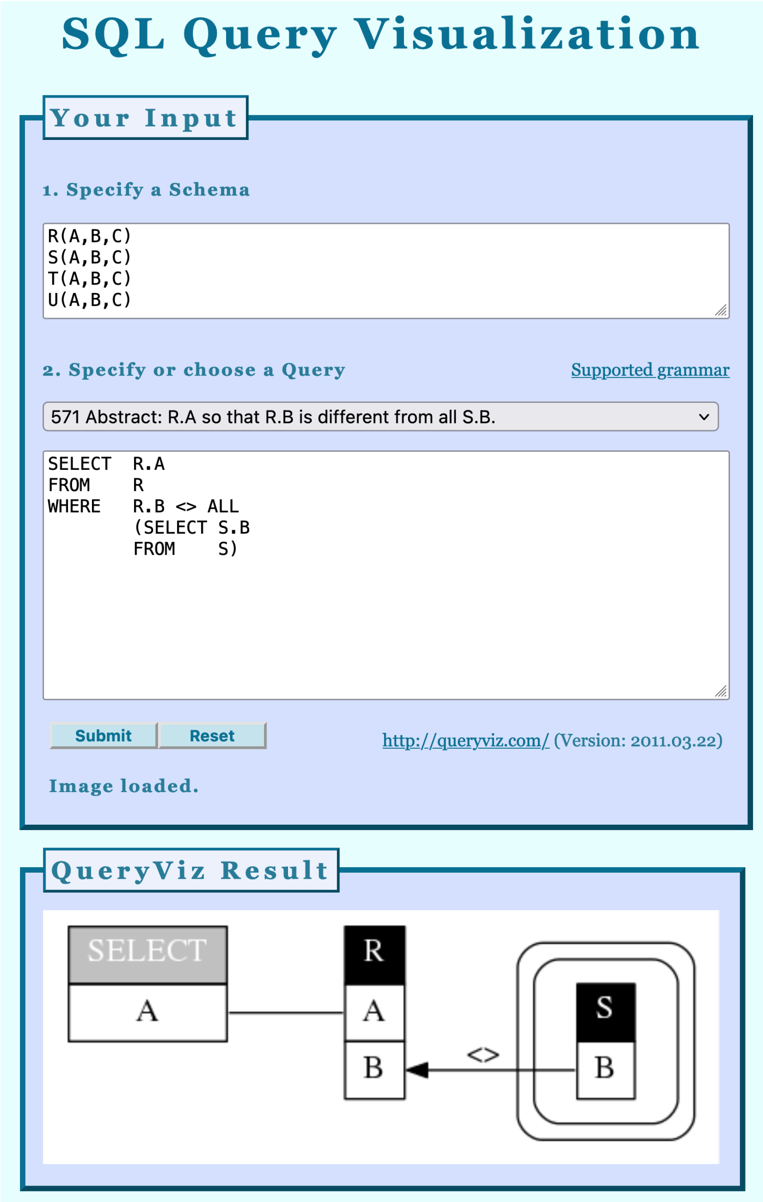
562 Abstract: R.A where R.B is different from at least one S.B.

```
SELECT  R.A
FROM    R
WHERE   exists
        (SELECT *
        FROM    S
        WHERE   R.B <> S.B)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

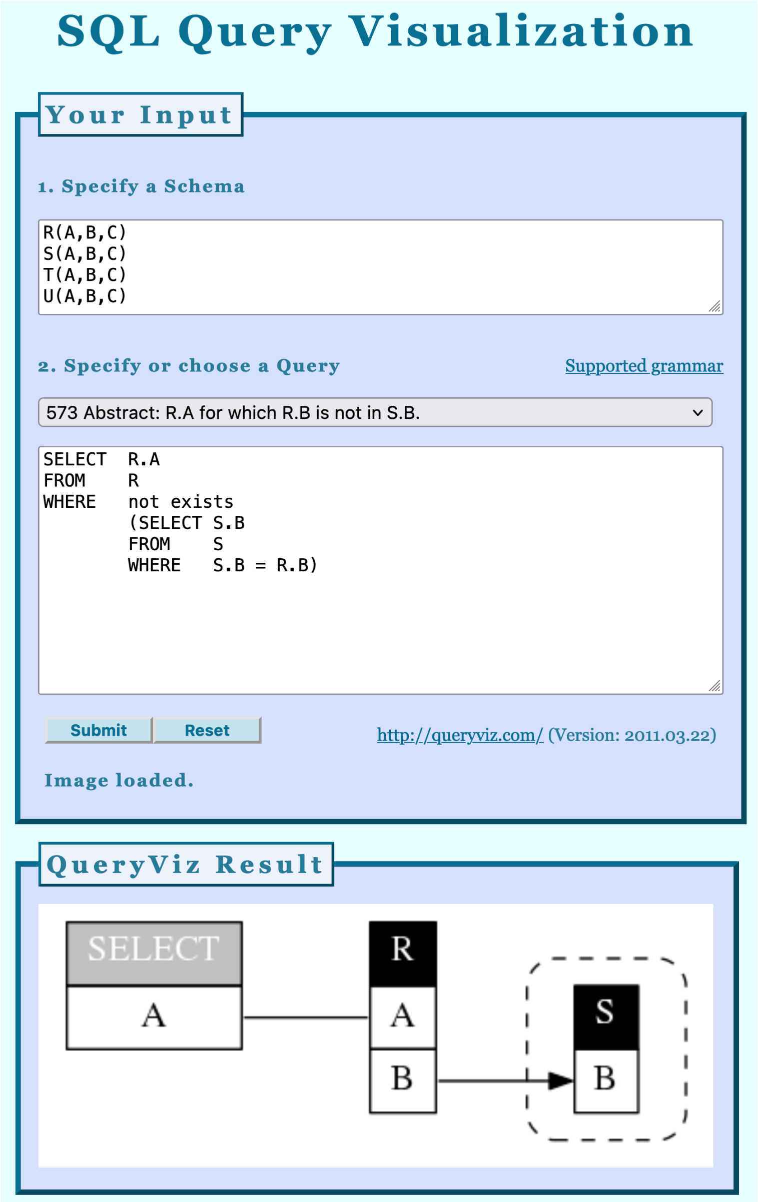
571 Abstract: R.A so that R.B is different from all S.B.

```
SELECT  R.A
FROM    R
WHERE   R.B <> ALL
        (SELECT S.B
        FROM    S)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

573 Abstract: R.A for which R.B is not in S.B.

```
SELECT  R.A
FROM    R
WHERE   not exists
        (SELECT S.B
        FROM    S
        WHERE   S.B = R.B)
```

http://queryviz.com/ (Version: 2011.03.22)

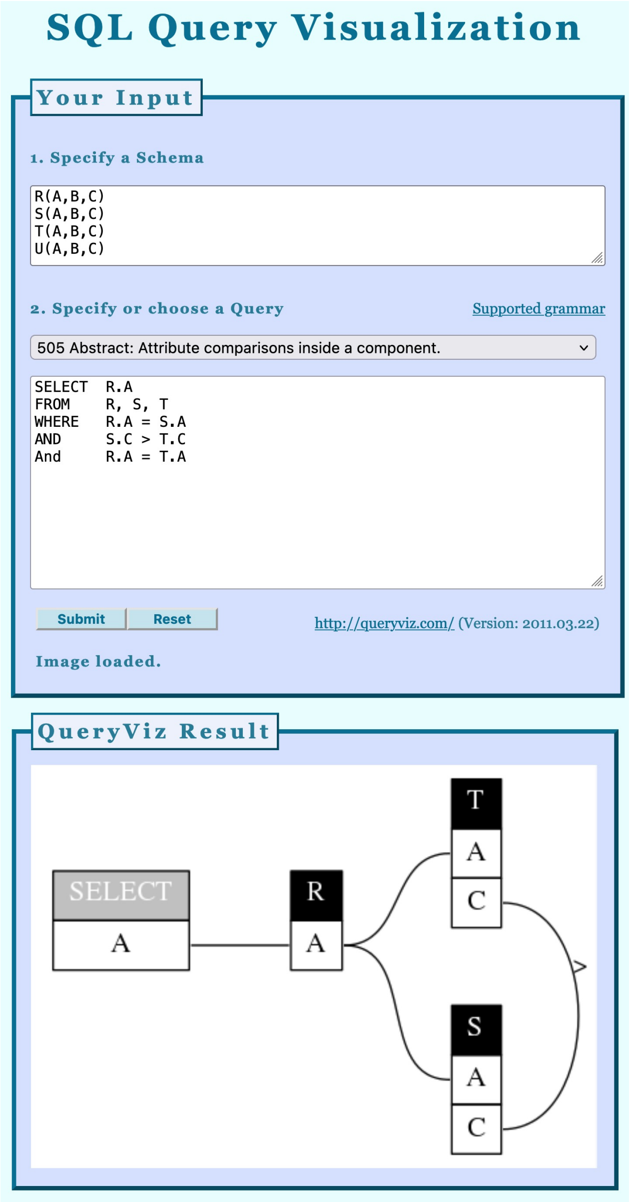
Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

Supported grammar

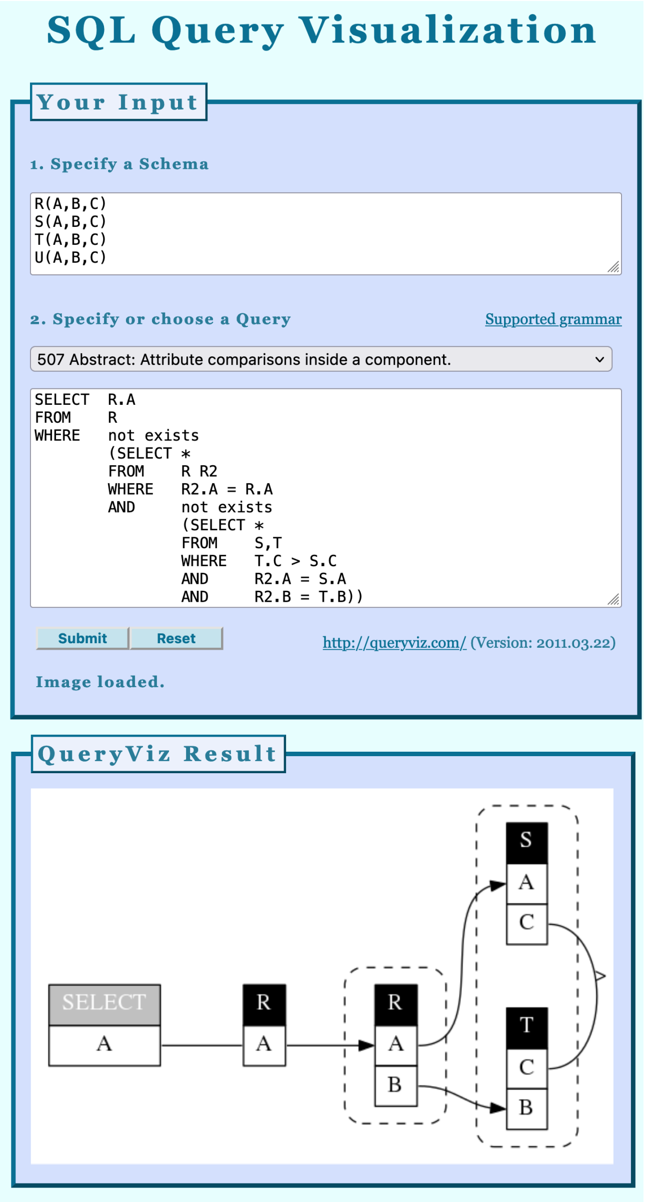
505 Abstract: Attribute comparisons inside a component.

```
SELECT  R.A
FROM    R, S, T
WHERE   R.A = S.A
AND     S.C > T.C
And     R.A = T.A
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

Supported grammar

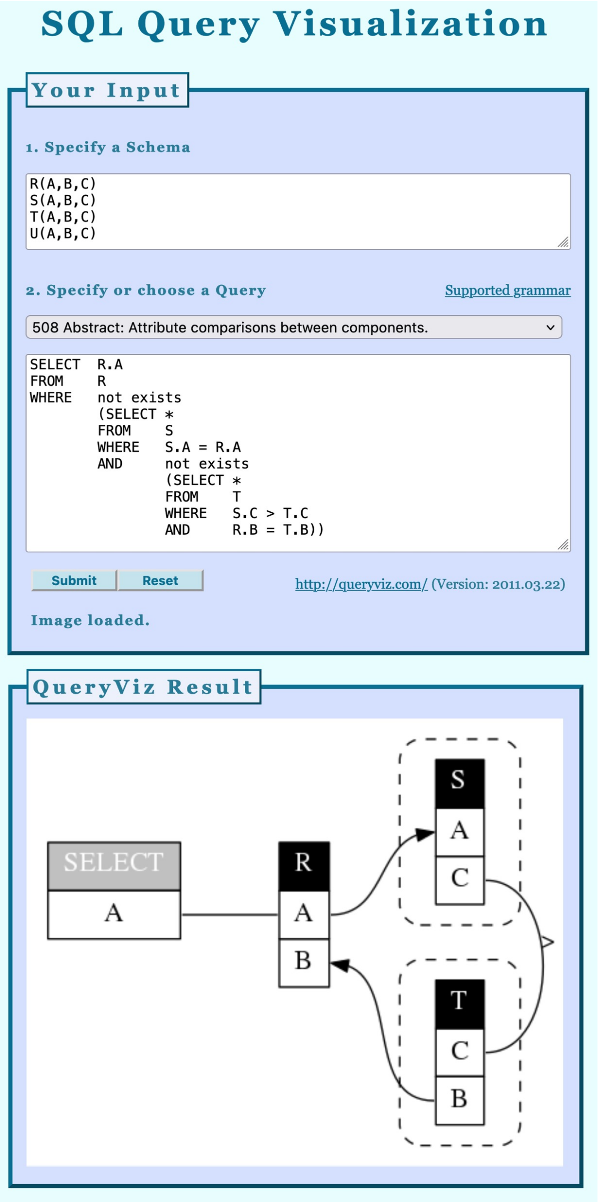
507 Abstract: Attribute comparisons inside a component.

```
SELECT  R.A
FROM    R
WHERE   not exists
        (SELECT *
        FROM    R R2
        WHERE   R2.A = R.A
        AND     not exists
                (SELECT *
                FROM    S,T
                WHERE   T.C > S.C
                AND     R2.A = S.A
                AND     R2.B = T.B))
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

Supported grammar

508 Abstract: Attribute comparisons between components.

```
SELECT  R.A
FROM    R
WHERE   not exists
        (SELECT *
        FROM    S
        WHERE   S.A = R.A
        AND     not exists
                (SELECT *
                FROM    T
                WHERE   S.C > T.C
                AND     R.B = T.B))
```

http://queryviz.com/ (Version: 2011.03.22)

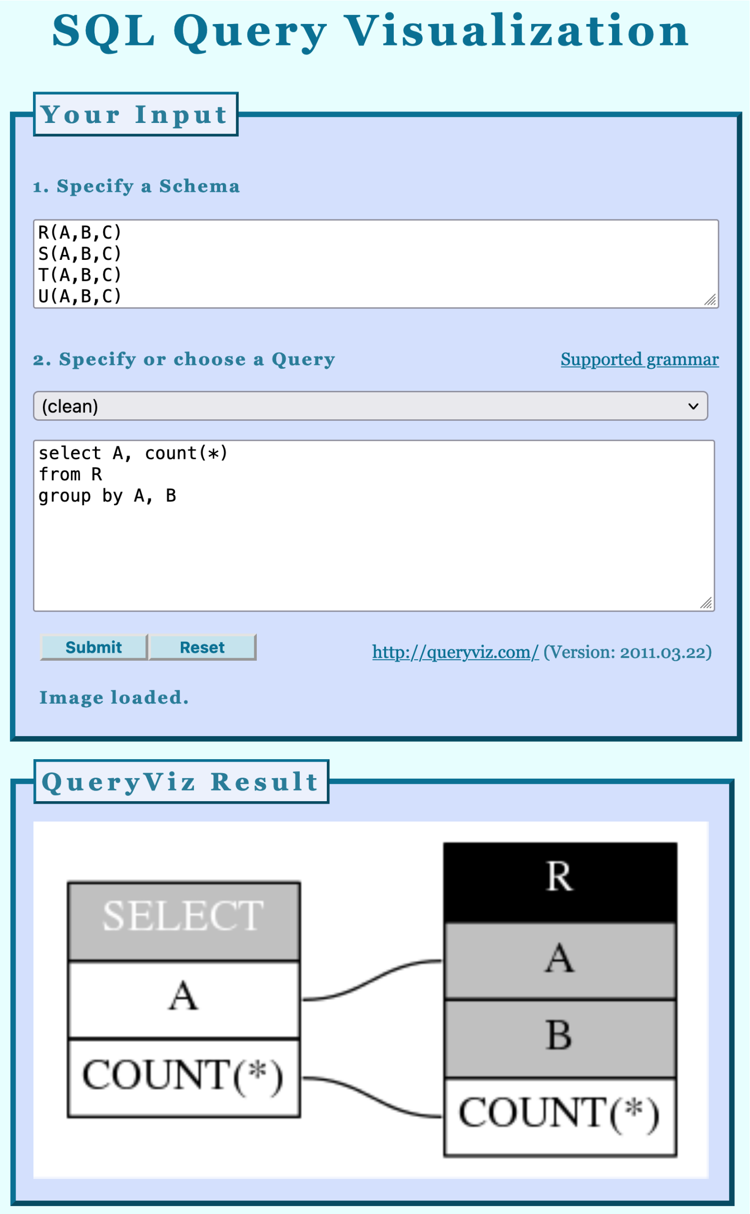
Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

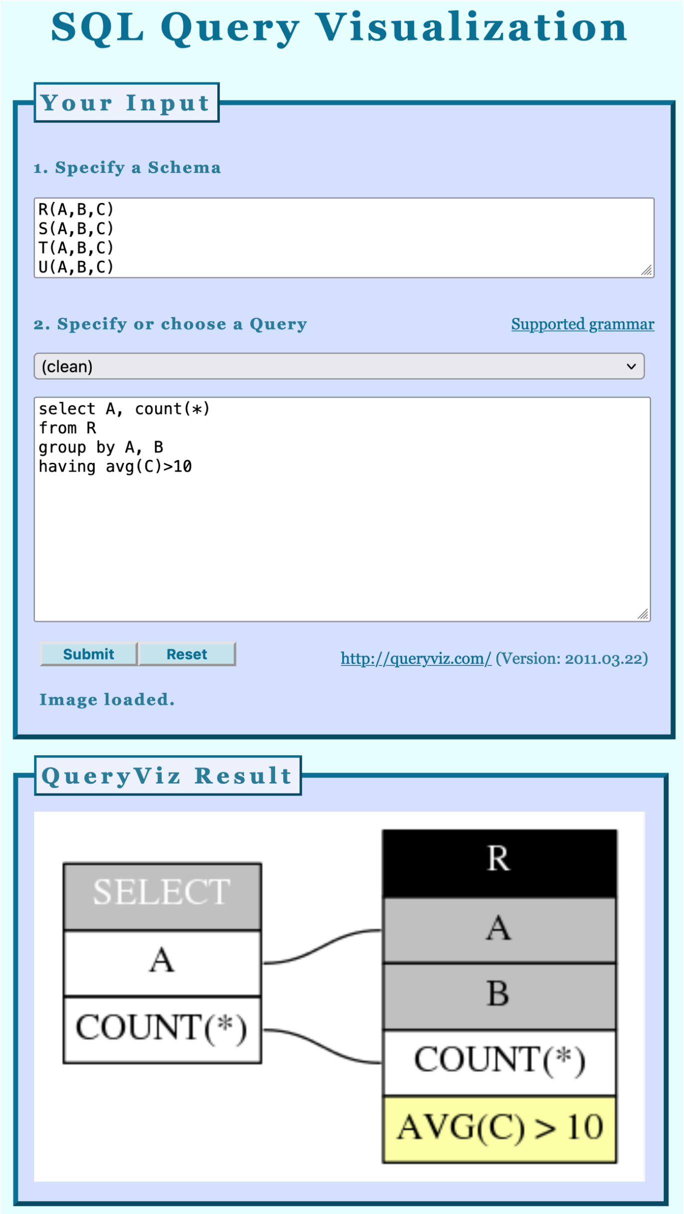
Online QueryVis



```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

(clean)

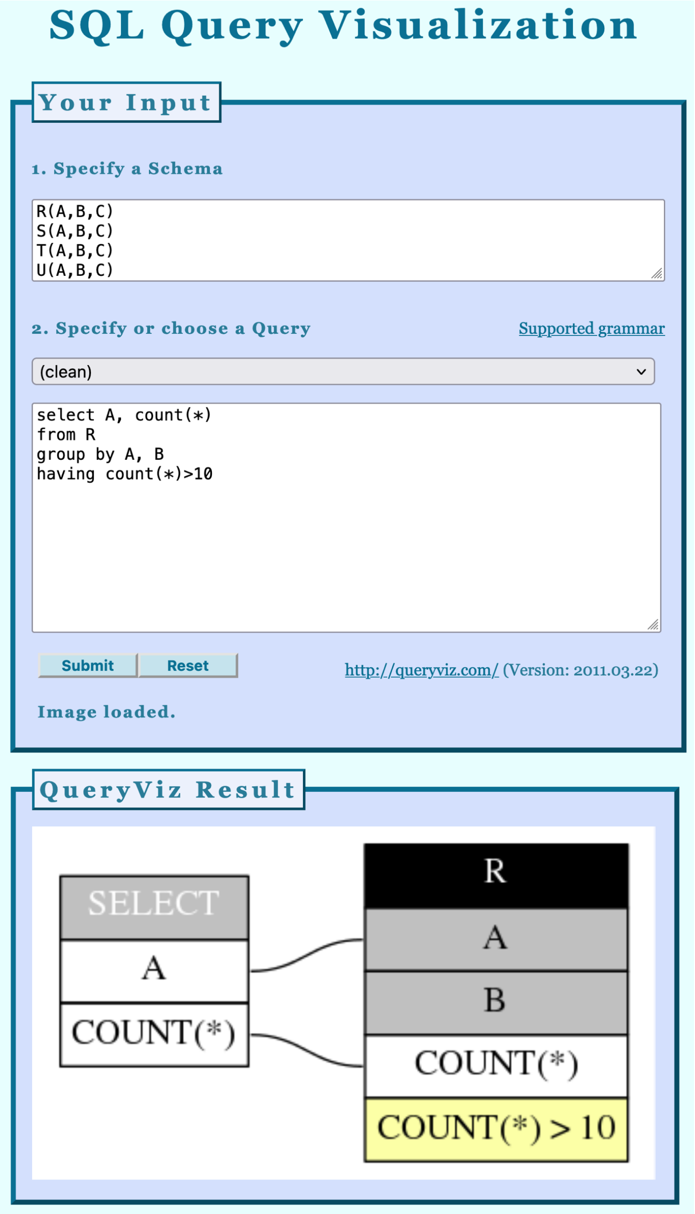
```
select A, count(*)
from R
group by A, B
```



```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

(clean)

```
select A, count(*)
from R
group by A, B
having avg(C)>10
```



```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

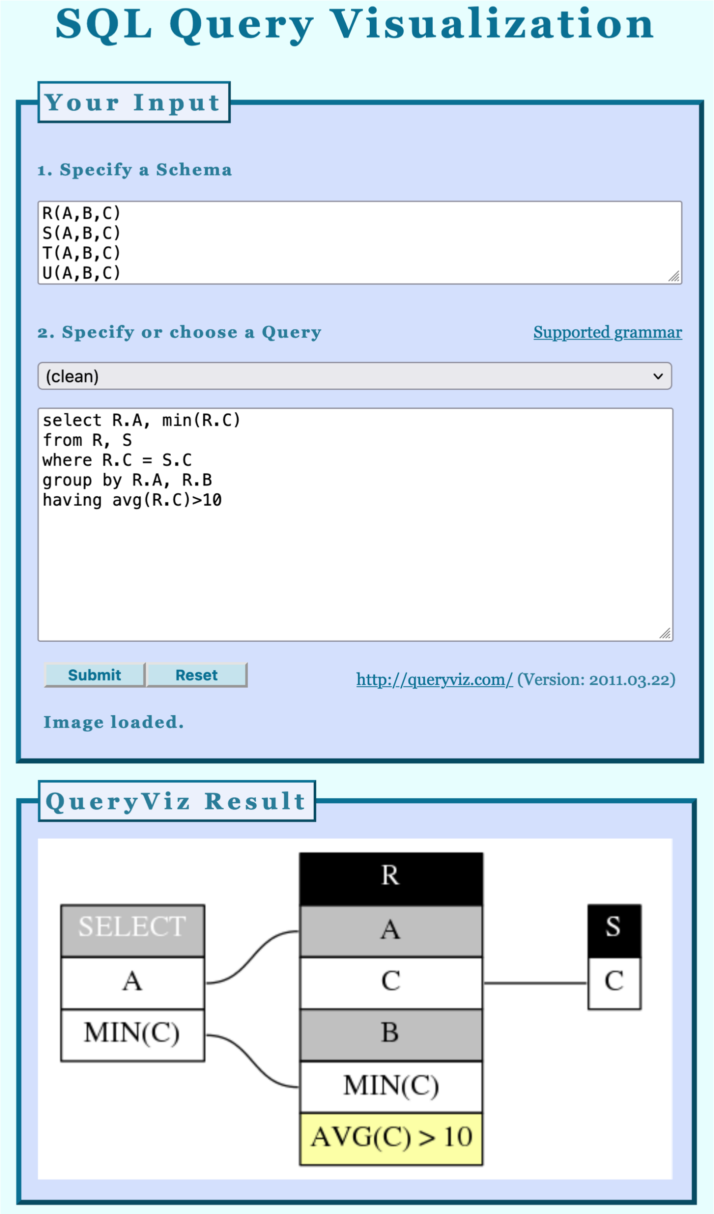
(clean)

```
select A, count(*)
from R
group by A, B
having count(*)>10
```

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

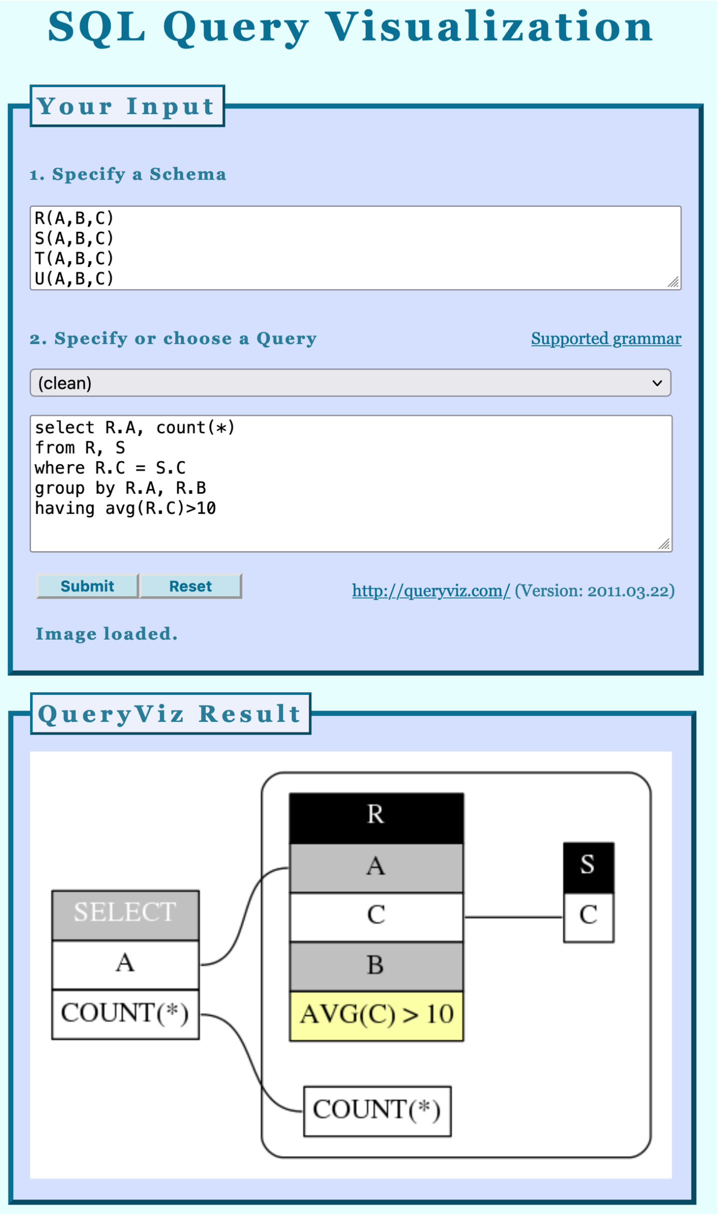
(clean)

```
select R.A, min(R.C)
from R, S
where R.C = S.C
group by R.A, R.B
having avg(R.C)>10
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

(clean)

```
select R.A, count(*)
from R, S
where R.C = S.C
group by R.A, R.B
having avg(R.C)>10
```

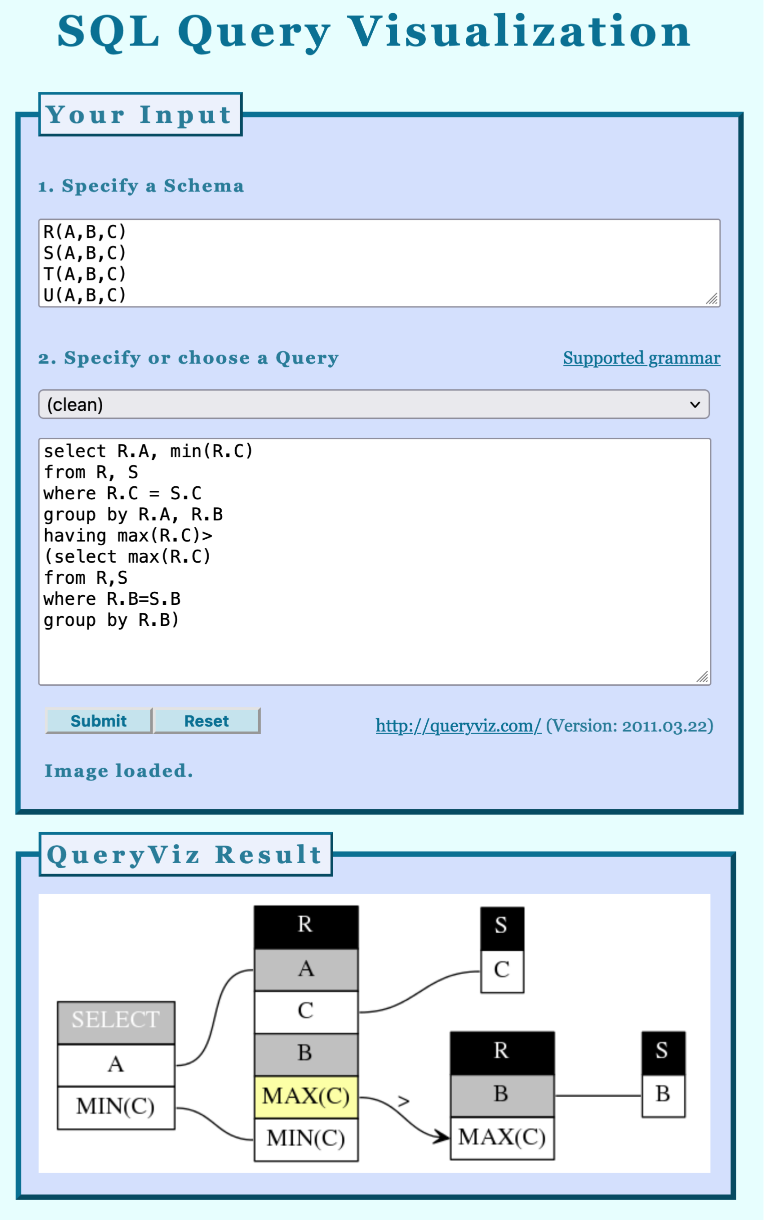
http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Online QueryVis



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query Supported grammar

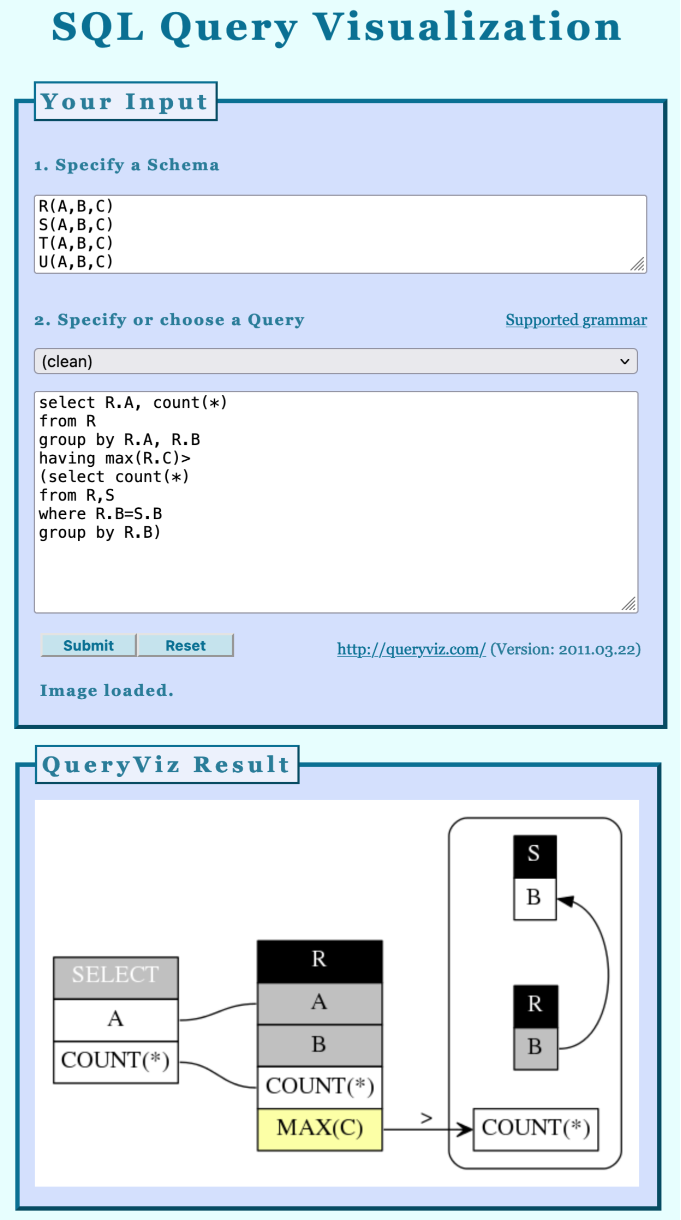
(clean)

```
select R.A, min(R.C)
from R, S
where R.C = S.C
group by R.A, R.B
having max(R.C)>
(select max(R.C)
from R,S
where R.B=S.B
group by R.B)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query Supported grammar

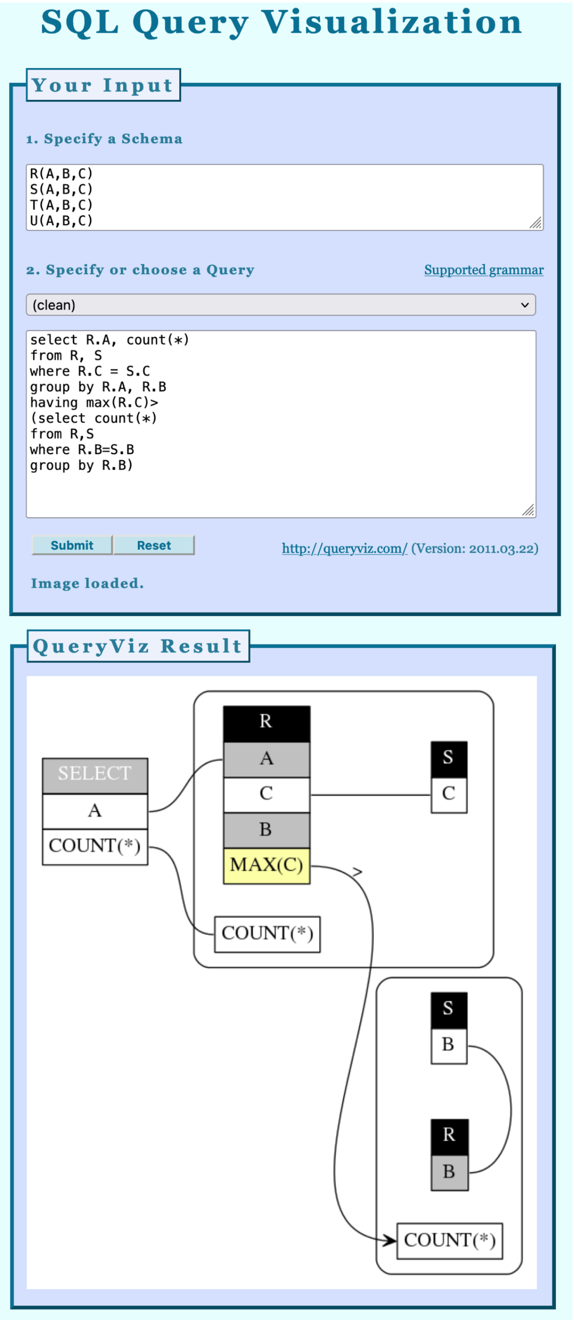
(clean)

```
select R.A, count(*)
from R
group by R.A, R.B
having max(R.C)>
(select count(*)
from R,S
where R.B=S.B
group by R.B)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query Supported grammar

(clean)

```
select R.A, count(*)
from R, S
where R.C = S.C
group by R.A, R.B
having max(R.C)>
(select count(*)
from R,S
where R.B=S.B
group by R.B)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

Dataplay (2012)

Sources used:

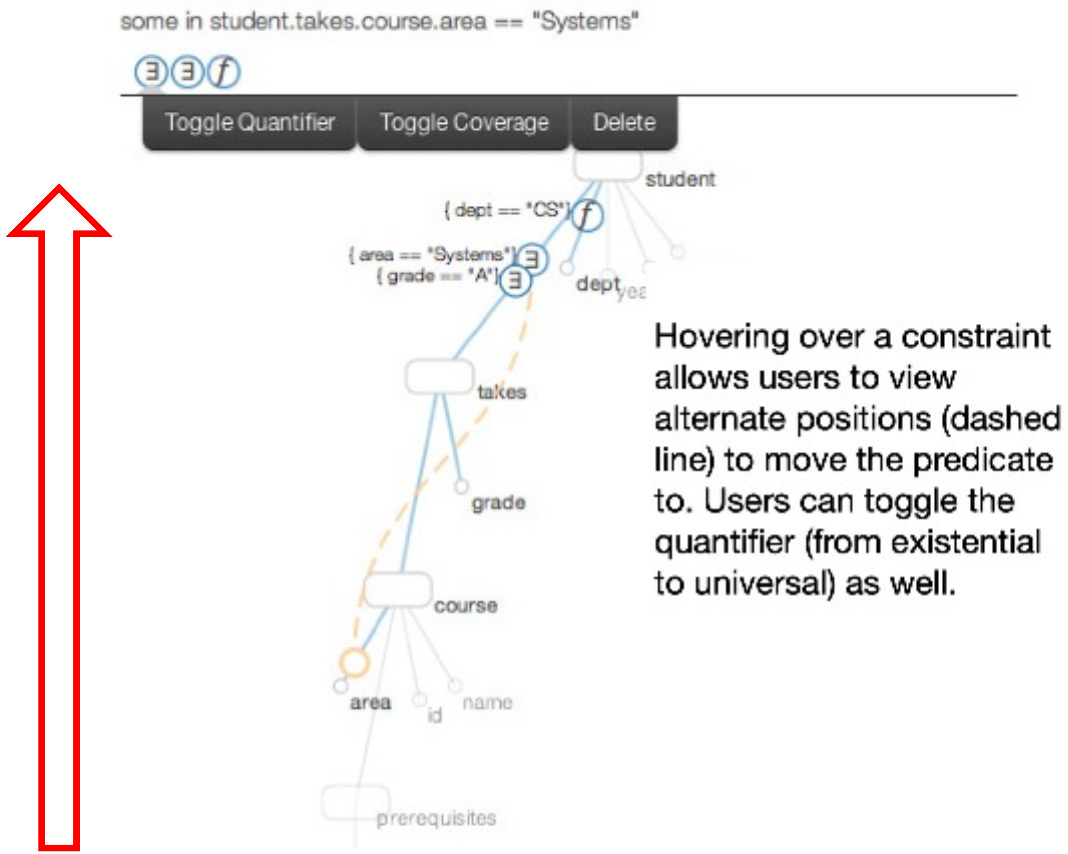
- Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144
- Abouzied, Hellerstein, Silberschatz. Playful Query Specification with DataPlay. VLDB demo 2012. https://doi.org/10.14778/2367502.2367542
- Abouzied, DataPlay Tutorial, 2012. 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 hovering, brushing) are not reflected in the visualization



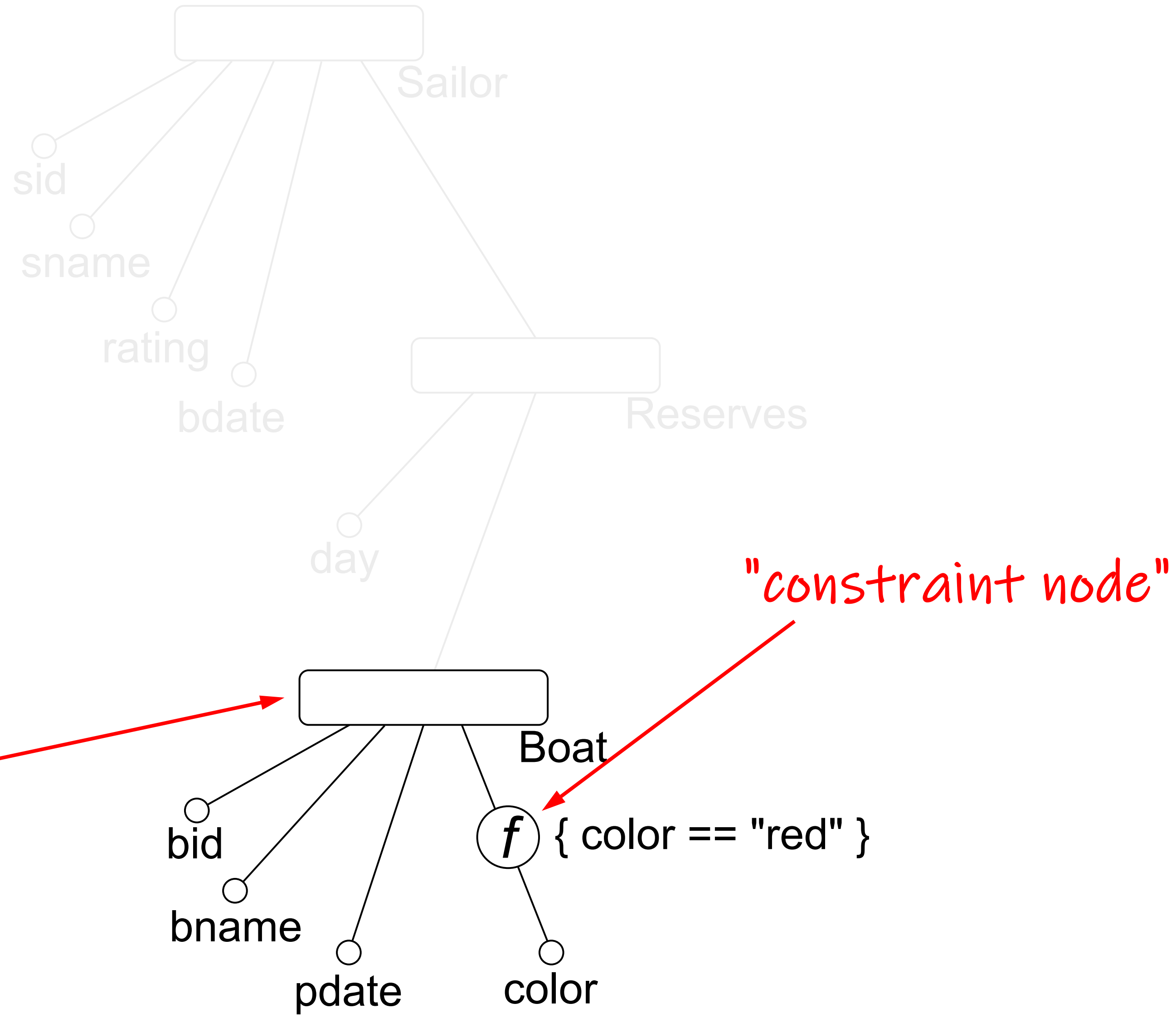
Hovering over a constraint allows users to view alternate positions (dashed line) to move the predicate to. Users can toggle the quantifier (from existential to universal) as well.

Figure source: "Abouzied, Hellerstein, Silberschatz. Playful Query Specification with DataPlay. VLDB demo 2012. https://doi.org/10.14778/2367502.2367542 "

Dataplay

Q1b: "Find boats that are not red."

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



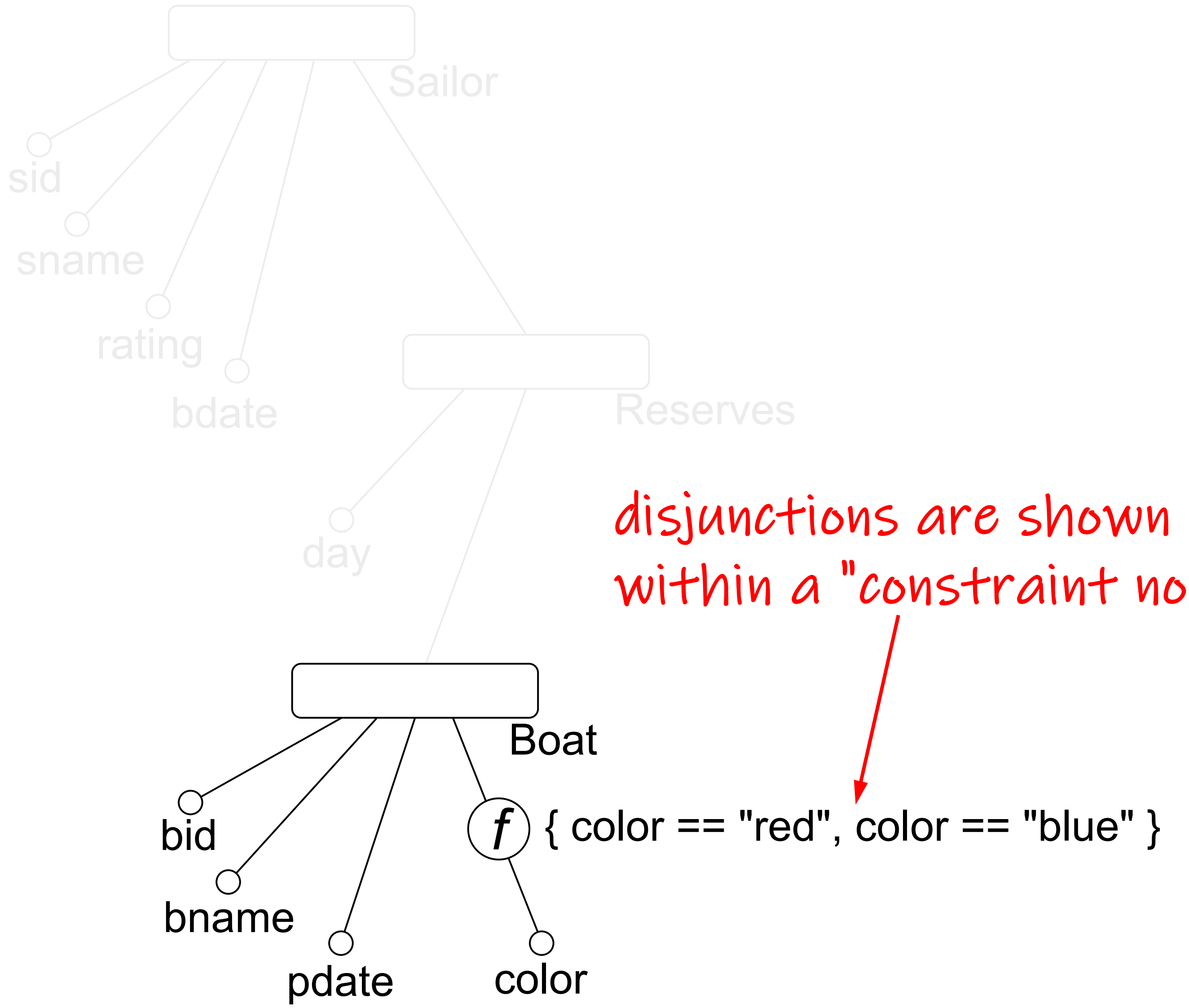
It has no visual symbol for selection

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

Q1: "Find boats that are red or blue."

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



disjunctions are shown within a "constraint node"

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

Q1c: "Find boats that are red or blue and purchased before 1980."

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

Conditions on separate attributes can be applied on each attribute separately.
Not clear how to express disjunctions *across* attributes, such as:
(color = 'red' or pdate < 1980)

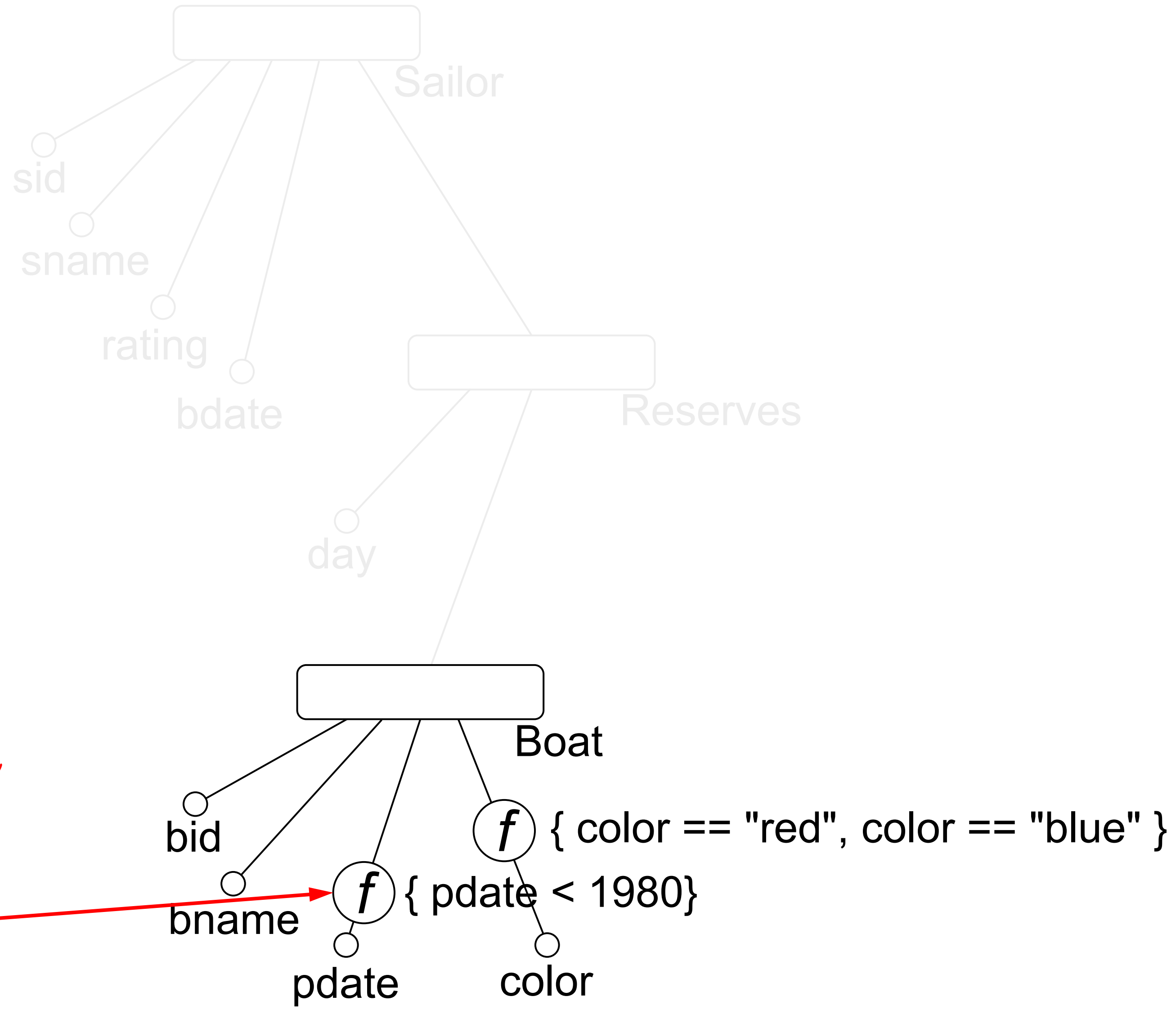


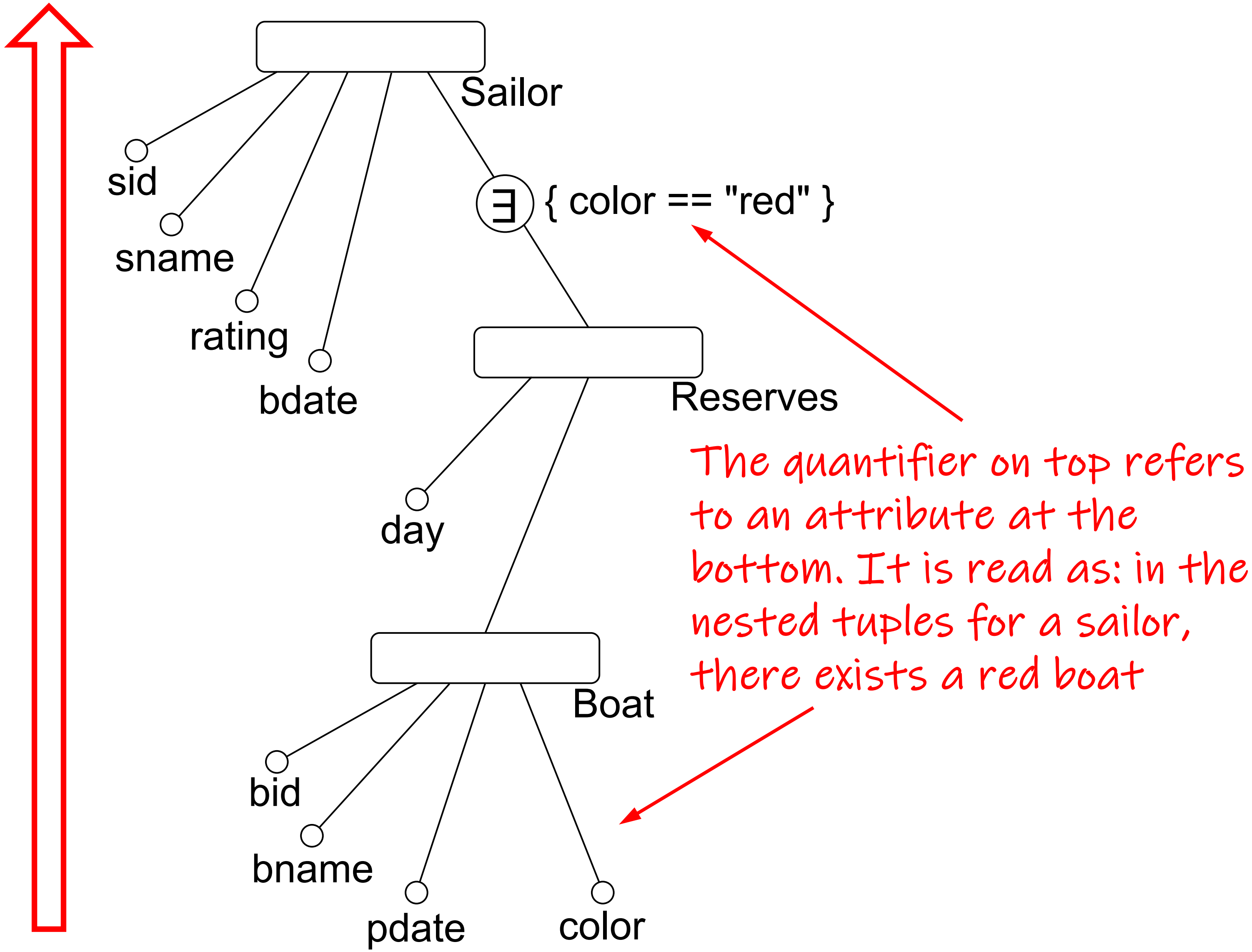
Figure drawn based on "Abouzied, Hellerstein, Silberschatz. DataPlay: interactive tweaking and example-driven correction of graphical database queries. UIST 2012. https://doi.org/10.1145/2380116.2380144

Dataplay

Q2: "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'
```

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)

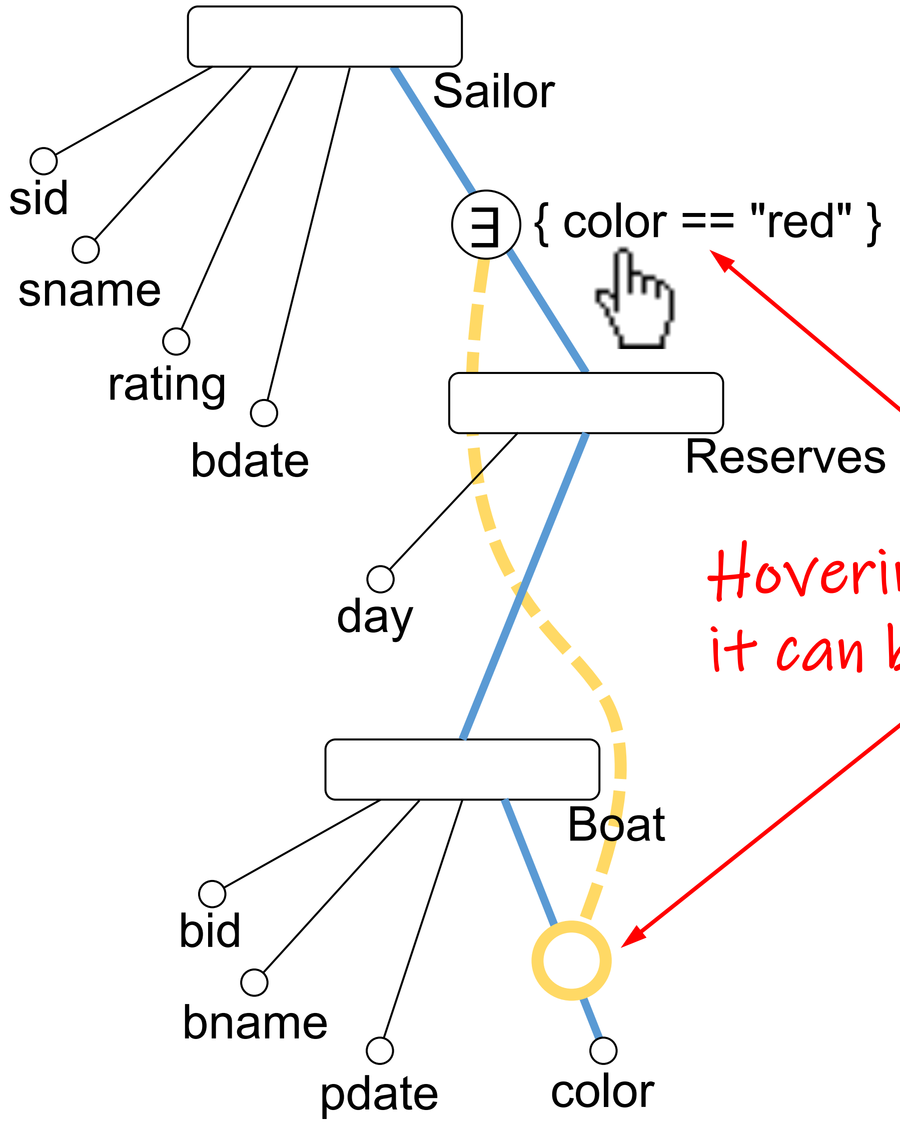
$$\{q.sname \mid \exists s \in Sailor[q.sname=s.sname \wedge \exists r \in Reserves[r.sid=s.sid \wedge \exists b \in Boat[b.bid=r.bid \wedge b.color='red']]]\}$$

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

Q2: "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'
```



Hovering shows where it can be placed

TRC (Tuple Relational Calculus)

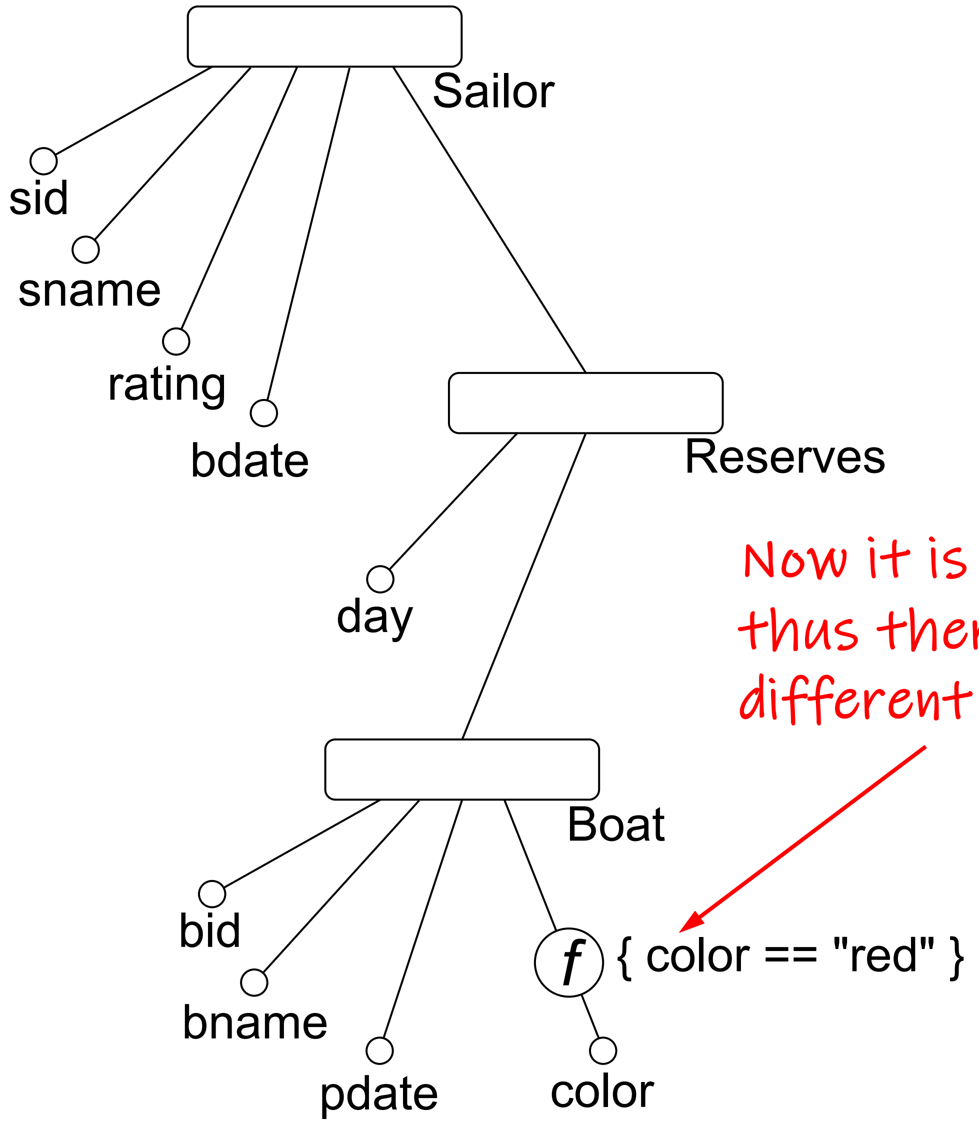
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

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

Q2: "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'
```



Now it is a constraint, thus there are two different visualizations

TRC (Tuple Relational Calculus)

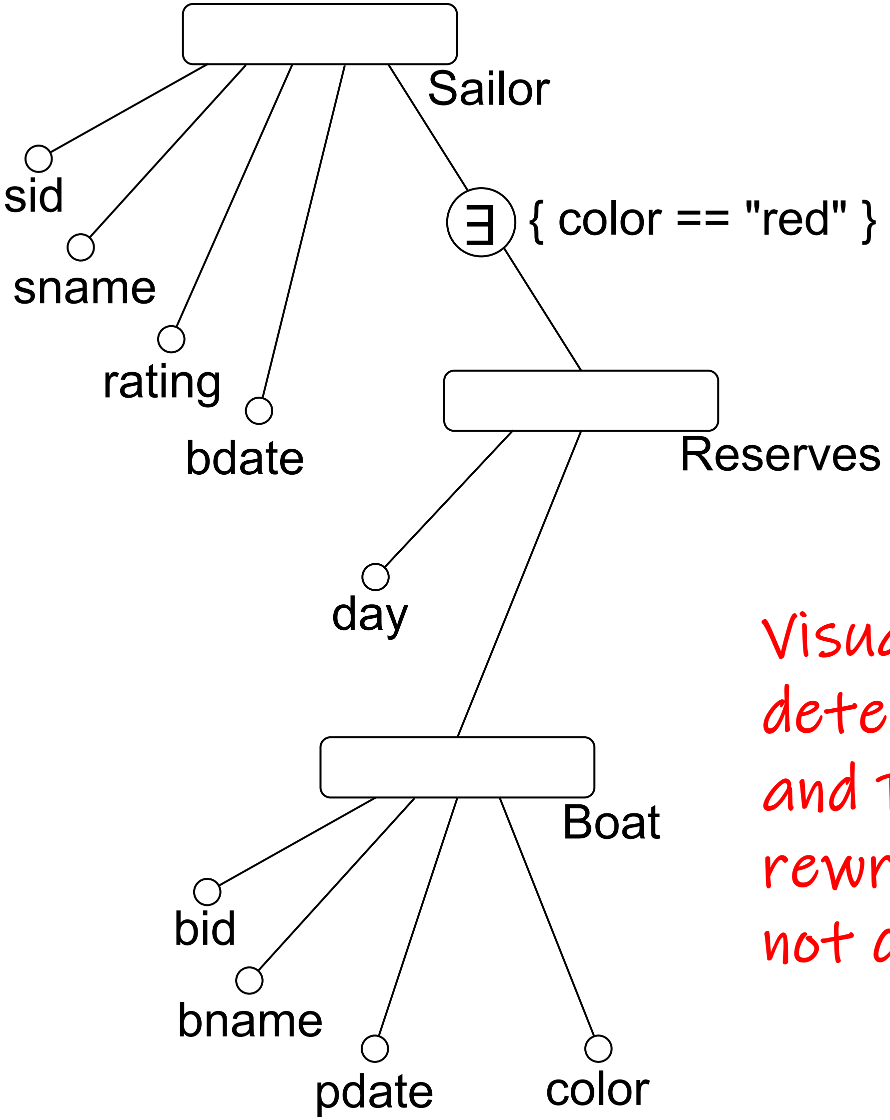
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

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

Q2: "Find sailors who reserved a red boat."

```
select distinct 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)

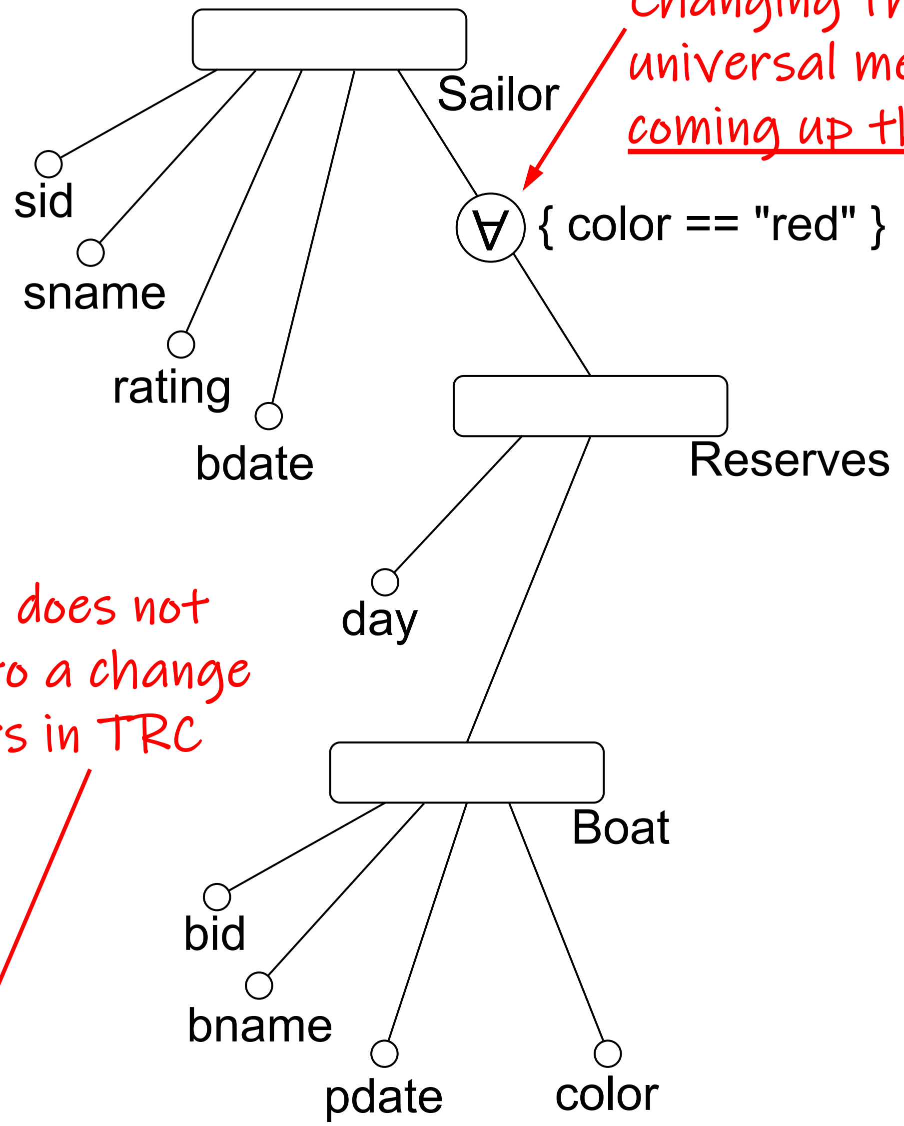
{q.sname | ∃s∈Sailor[q.sname=s.sname ∧ ∃r∈Reserves[r.sid=s.sid ∧ ∃b∈Boat[b.bid=r.bid ∧ b.color='red']]]}

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

Q3: "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'))
```



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)

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

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
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Dataplay

Q4: "Find sailors who reserved all red boats."

```
select distinct 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)

- Unions (Query Q5) are not supported

TRC (Tuple Relational Calculus)

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

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 (2012)
Backup

Dataplay

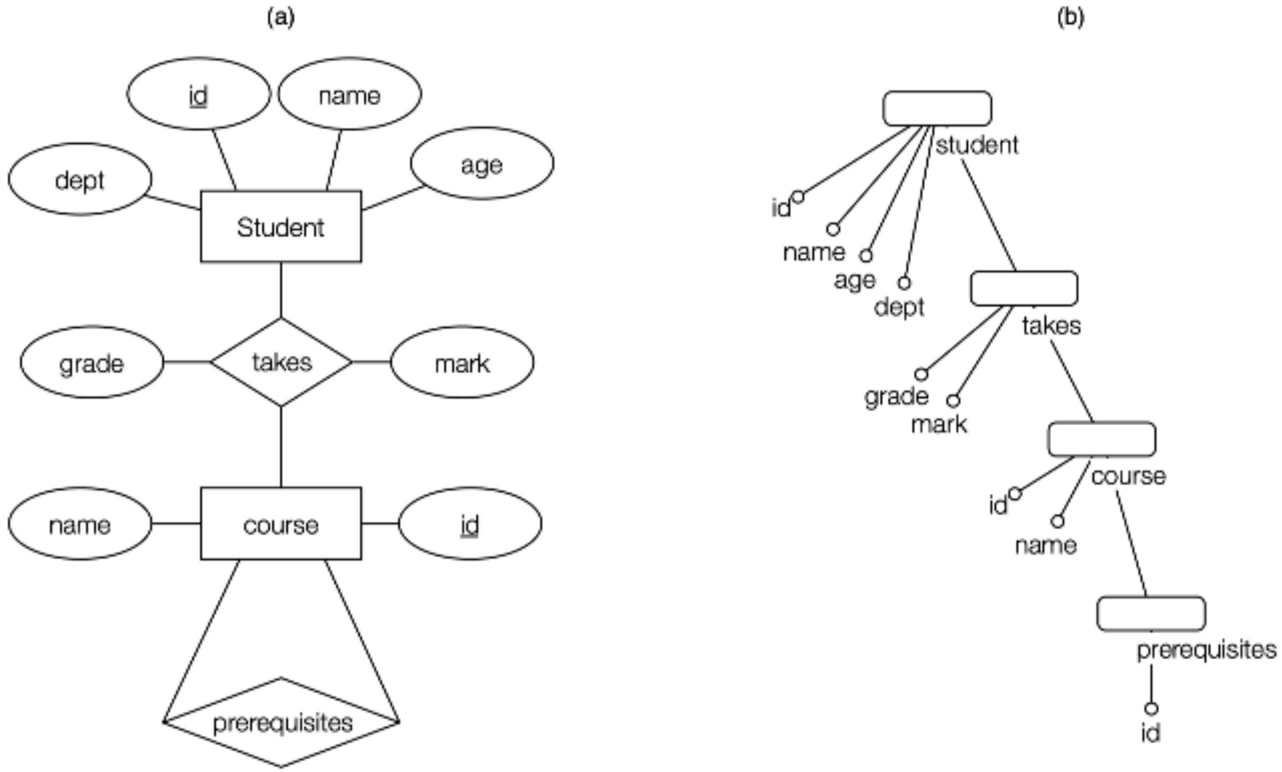


Figure 1: (a) ER-diagram of a school database; (b) The nested-UR schema with student as pivot.

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)

Source: Abouzied, Hellerstein, Silberschatz. Playful query specification with DataPlay. VLDB demo 2012. https://doi.org/10.14778/2367502.2367542
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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.

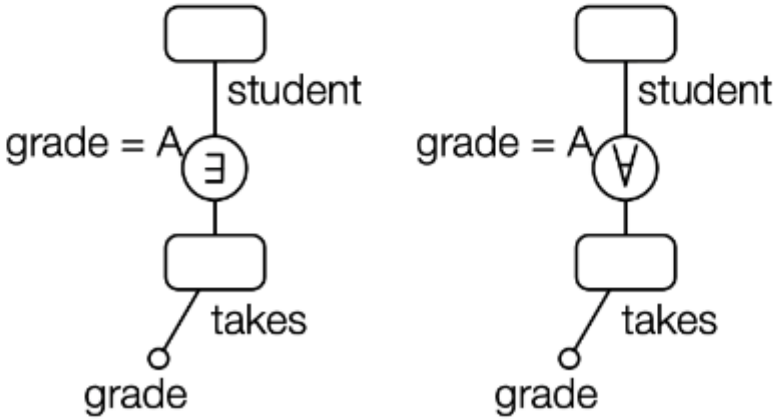


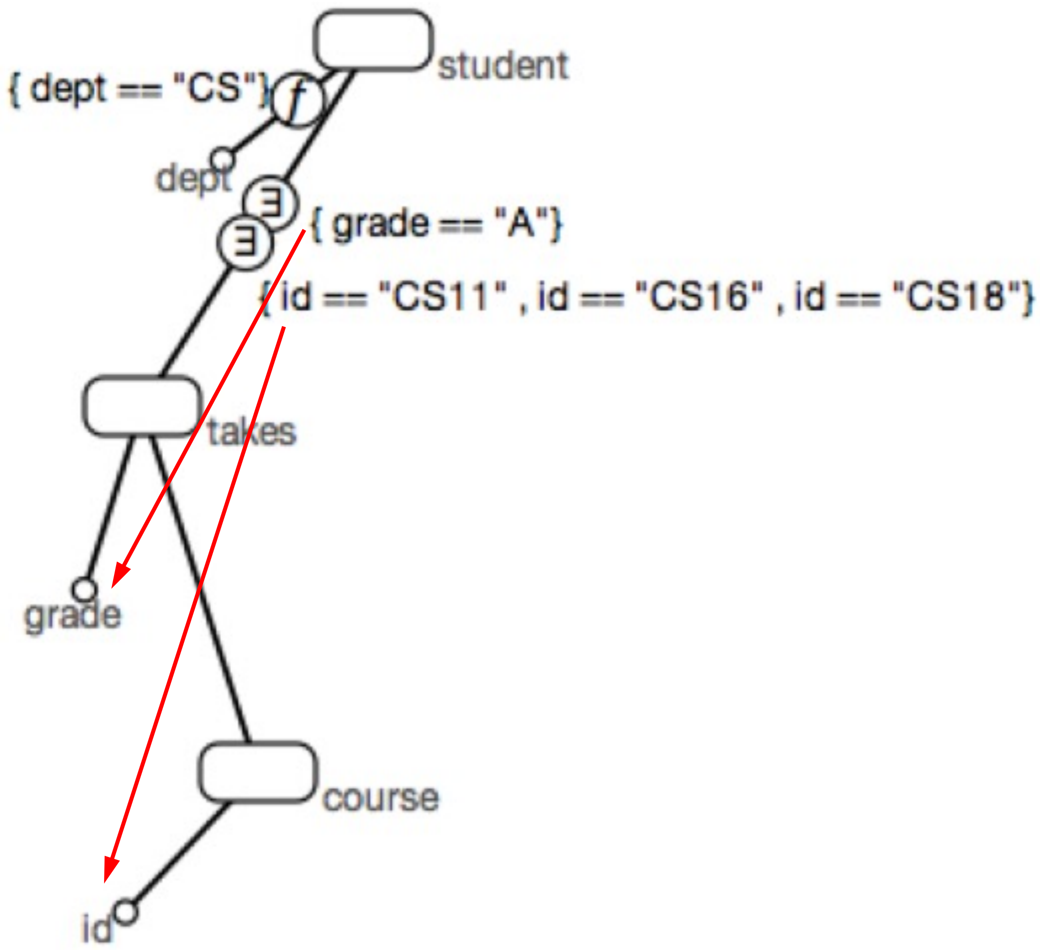
Figure 3: (a) Students with some A's (b) Straight-A's

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

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)

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

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

SIEUFERD (2016)

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

Sources used:

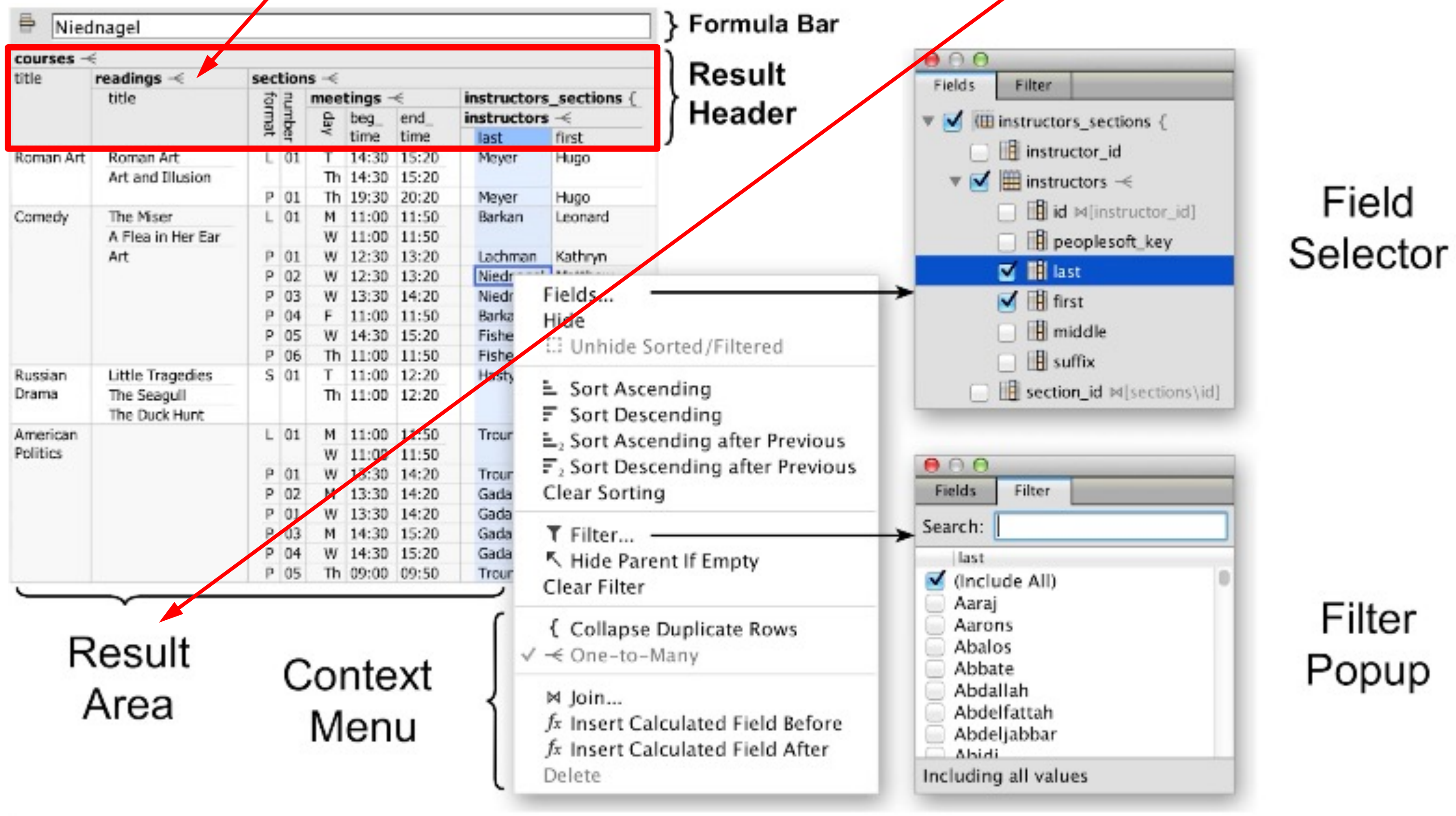
- Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
- Bakke. SIEUFERD: A Visual Query System. 2016. https://www.youtube.com/watch?v=W6xmqcb8hFQ
- Personal communication with Eirik Bakke. System reincarnated as Ultorg at https://www.ultorg.com/

SIEUFERD

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



Niednagel

courses ⋞								
title	readings ⋞	sections ⋞						
	title	format	number	meetings ⋞			instructors_sections { instructors ⋞	
				day	beg_time	end_time	last	first
Roman Art	Roman Art	L	01	T	14:30	15:20	Meyer	Hugo
	Art and Illusion			Th	14:30	15:20		
		P	01	Th	19:30	20:20	Meyer	Hugo
Comedy	The Miser	L	01	M	11:00	11:50	Barkan	Leonard
	A Flea in Her Ear			W	11:00	11:50		
	Art	P	01	W	12:30	13:20	Lachman	Kathryn
		P	02	W	12:30	13:20	Niedr	
		P	03	W	13:30	14:20	Niedr	
		P	04	F	11:00	11:50	Barka	
		P	05	W	14:30	15:20	Fishe	
		P	06	Th	11:00	11:50	Fishe	
Russian Drama	Little Tragedies	S	01	T	11:00	12:20	Hasty	
	The Seagull			Th	11:00	12:20		
	The Duck Hunt							
American Politics		L	01	M	11:00	11:50	Trour	
				W	11:00	11:50		
		P	01	W	13:30	14:20	Trour	
		P	02	M	13:30	14:20	Gada	
		P	01	W	13:30	14:20	Gada	
		P	03	M	14:30	15:20	Gada	
		P	04	W	14:30	15:20	Gada	
		P	05	Th	09:00	09:50	Trour	

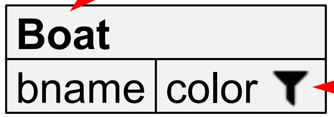
Figure source: https://www.youtube.com/watch?v=W6xmqcb8hFQ

SIEUFERD

Q1b: "Find boats that are not red."

```
select distinct 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

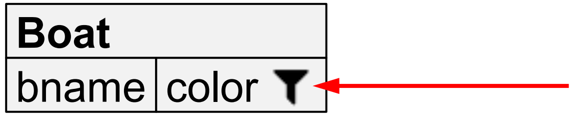
Recall that we are only focusing on the visual representation of the query. The actual system also displays the results (the goal is an interactive exploration). The backup shows screenshots from the actual system

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q1: "Find boats that are red or blue."

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



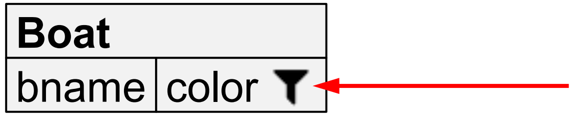
Header does not change: actual filter condition is only shown in a separate "filter pop" up window where colors "red" and "blue" are chosen

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q1c: "Find boats that are red or blue and purchased before 1980."

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



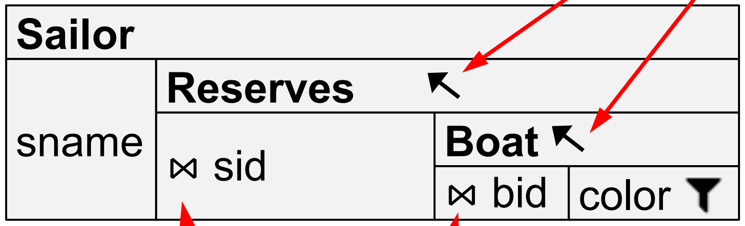
Header does not change

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q2: "Find sailors who reserved a red boat."

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

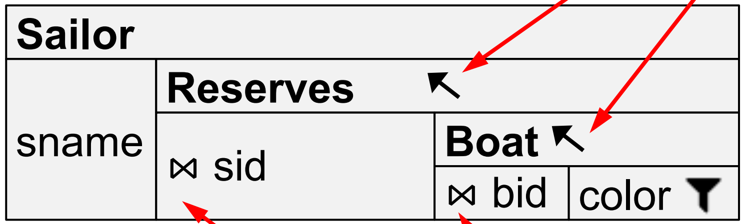
Join conditions with attributes in outer scope are not visible and only shown in a separate "field selector" popup window. Both attributes Reserves.sid and Boat. bid could also be hidden.

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q2: "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'
```



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

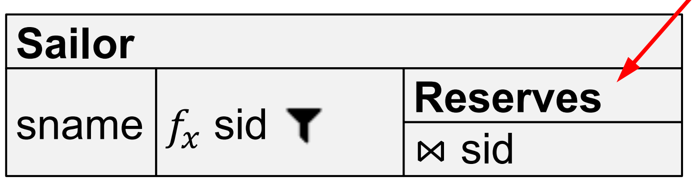
Join conditions with attributes in outer scope are not visible and only shown in a separate "field selector" popup window. Both attributes Reserves.sid and Boat.bid could also be hidden.

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

SIEUFERD

Q3a: "Find sailors who reserved no boat."

```
select distinct 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 (↖)

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"

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 distinct S.sname
from Sailor S
left join Reserves R
on S.sid = R.sid
where R.sid 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

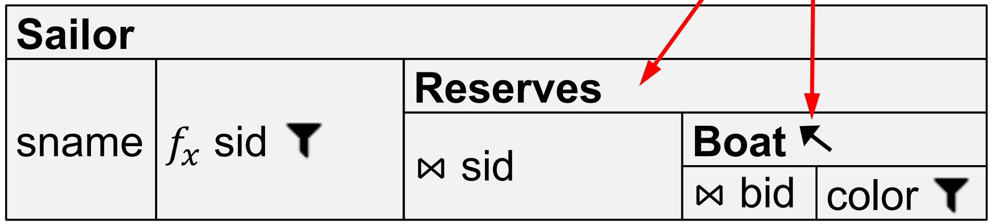
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q3b: "Find sailors who reserved no red boat."

```
select distinct 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')
```

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



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 distinct 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/

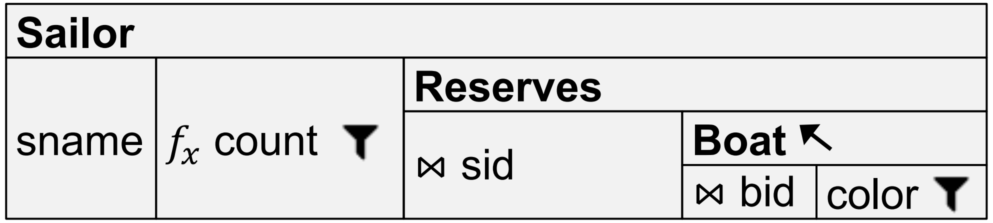
Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q3: "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'))
```



An alternative syntax using aggregation and COUNT=0

```
select distinct 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/

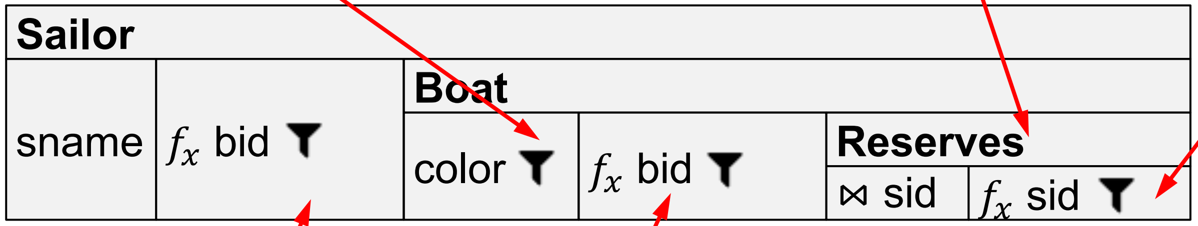
Figure drawn based on: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q4: "Find sailors who reserved all red boats."

```
select distinct 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Q5: "Find boats that are red or blue."

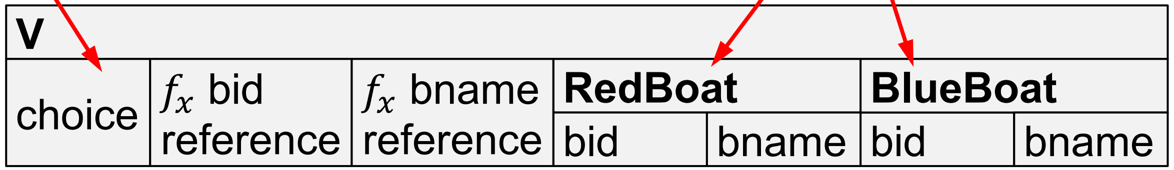
```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

Temporary valuations table V contains values true and false

RedBoat and BlueBoat tables cross joined (could also be nested)

V

choice
true
false



V						
choice	f_x bid reference	f_x bname reference	**RedBoat**		**BlueBoat**	
			bid	bname	bid	bname

Conditional statement for bid (analog for bname)
if V.choice then RedBoat.bid else BlueBoat.bid

Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

```
select distinct case when choice then R.bid
    else B.bid end,
    case when choice then R.bname
    else B.bname end
from RedBoat R, BlueBoat B,
    (select * from (values (true), (false)) V(choice) ) V
```

This SQL query shows the logic that SIEUFERD would use if it were implemented in SQL.

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

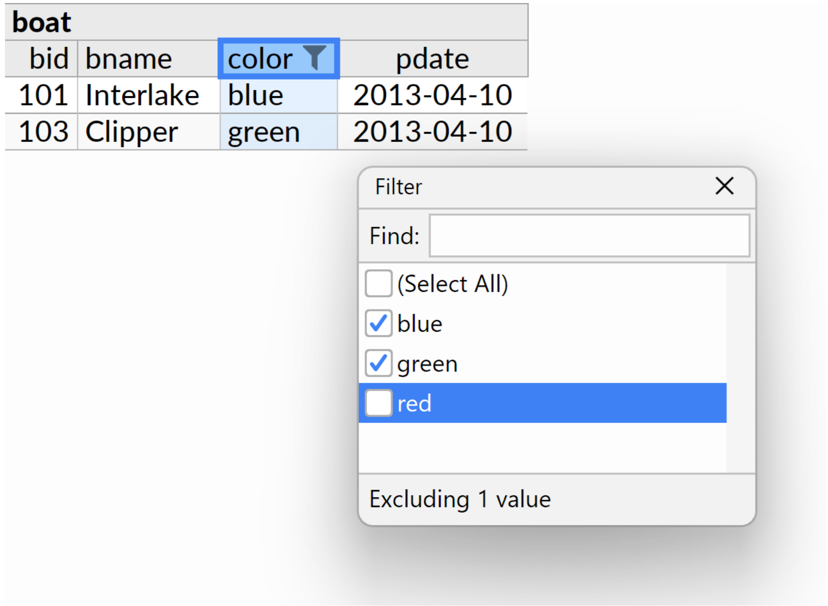
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD (2016)
Backup

SIEUFERD / Ultorg (actual screenshots)

Q1b: "Find boats that are not red."

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



boat

bid	bname	color	pdate
101	Interlake	blue	2013-04-10
103	Clipper	green	2013-04-10

Filter

Find:

☐ (Select All)
☑ blue
☑ green
☐ red

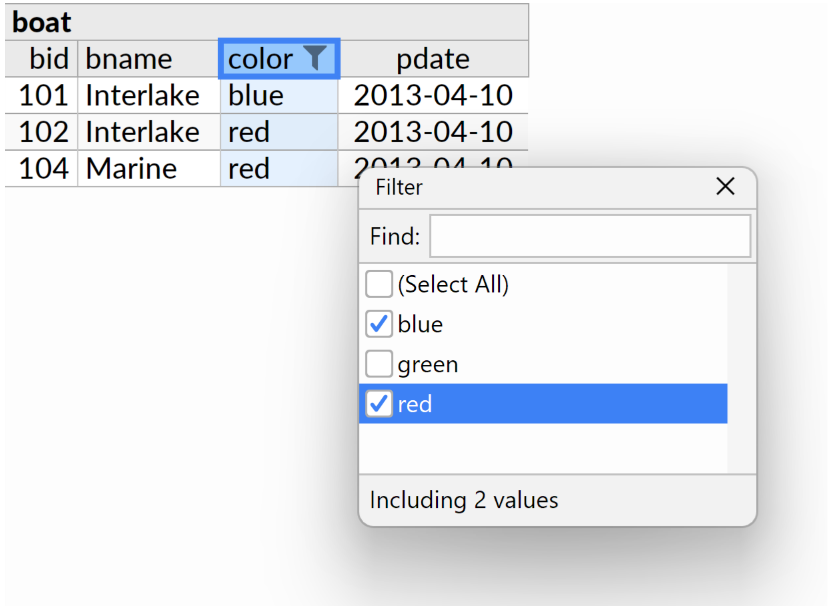
Excluding 1 value

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q1: "Find boats
that are red or blue."

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



boat

bid	bname	color	pdate
101	Interlake	blue	2013-04-10
102	Interlake	red	2013-04-10
104	Marine	red	2013-04-10

Filter

Find:

☐ (Select All)
☑ blue
☐ green
☑ red

Including 2 values

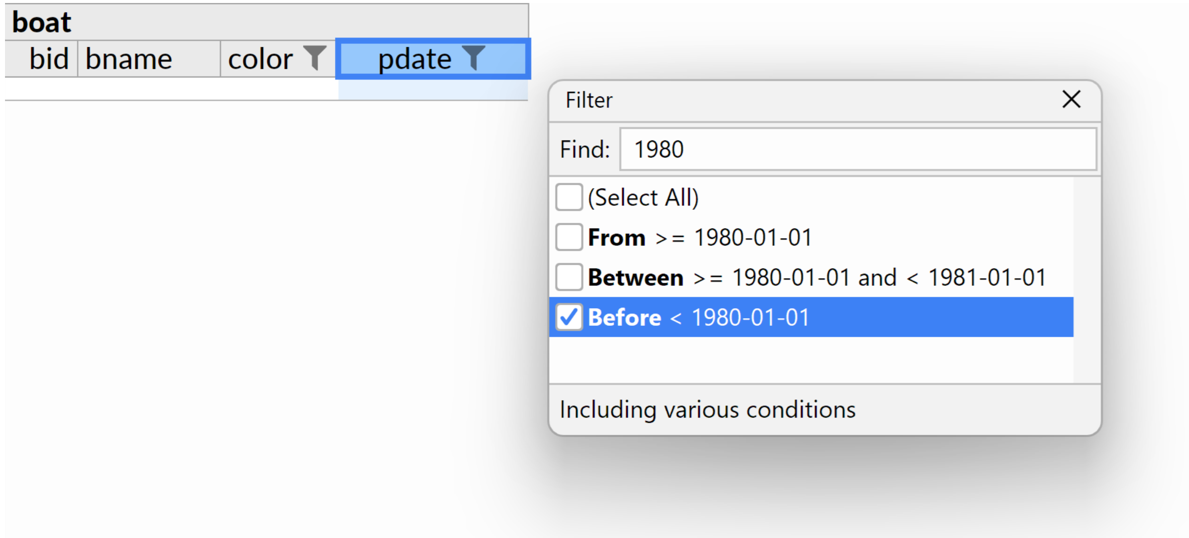
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD / Ultorg (actual screenshots)

Q1c: "Find boats that are red or blue and purchased before 1980."

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



Filter

Find: 1980

☐ (Select All)
☐ **From** >= 1980-01-01
☐ **Between** >= 1980-01-01 and < 1981-01-01
☑ **Before** < 1980-01-01

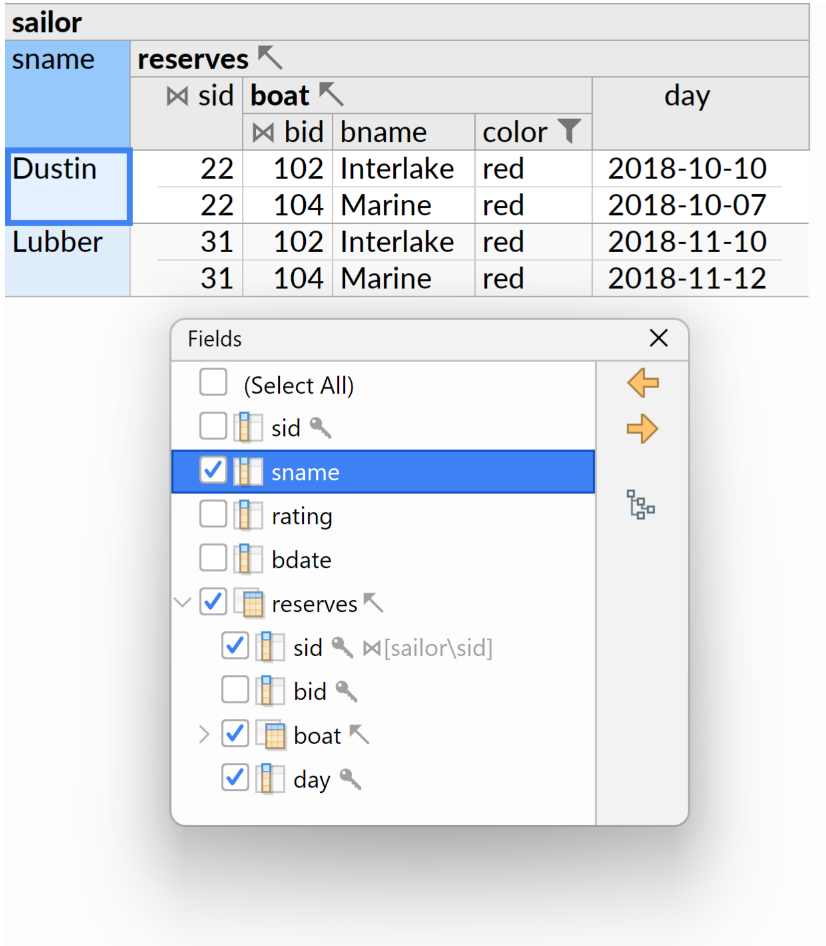
Including various conditions

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q2: "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'
```



sailor					
sname	**reserves** ↖				
	⋈ sid	**boat** ↖			day
		⋈ bid	bname	color	
Dustin	22	102	Interlake	red	2018-10-10
	22	104	Marine	red	2018-10-07
Lubber	31	102	Interlake	red	2018-11-10
	31	104	Marine	red	2018-11-12

Fields

- ☐ (Select All)
- ☐ sid
- ☑ sname
- ☐ rating
- ☐ bdate
- ☑ reserves ↖
  - ☑ sid ⋈[sailor\sid]
  - ☐ bid
  - ☑ boat ↖
  - ☑ day

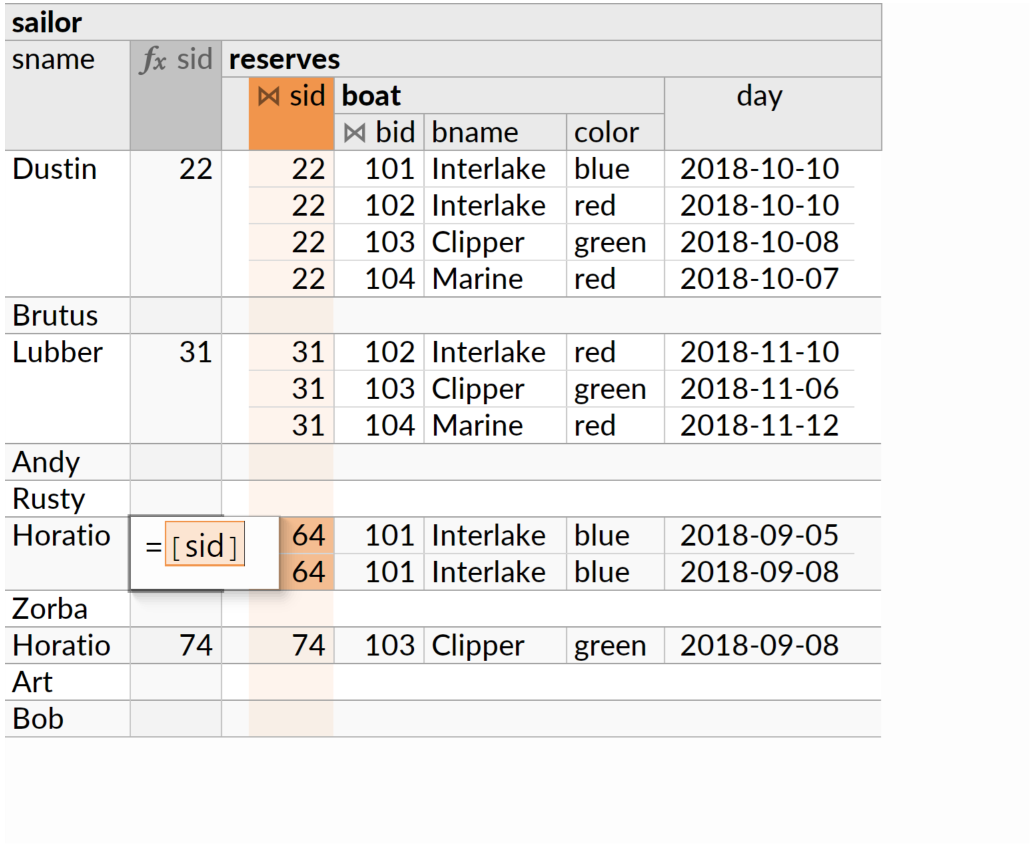
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3a: "Find sailors who reserved no boat."

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

Reference formula version (1/2)



sailor						
sname	𝑓𝑥 sid	**reserves**				
		⋈ sid	**boat**			day
			⋈ bid	bname	color	
Dustin	22	22	101	Interlake	blue	2018-10-10
		22	102	Interlake	red	2018-10-10
		22	103	Clipper	green	2018-10-08
		22	104	Marine	red	2018-10-07
Brutus						
Lubber	31	31	102	Interlake	red	2018-11-10
		31	103	Clipper	green	2018-11-06
		31	104	Marine	red	2018-11-12
Andy						
Rusty						
Horatio	= [sid]	64	101	Interlake	blue	2018-09-05
		64	101	Interlake	blue	2018-09-08
Zorba						
Horatio	74	74	103	Clipper	green	2018-09-08
Art						
Bob						

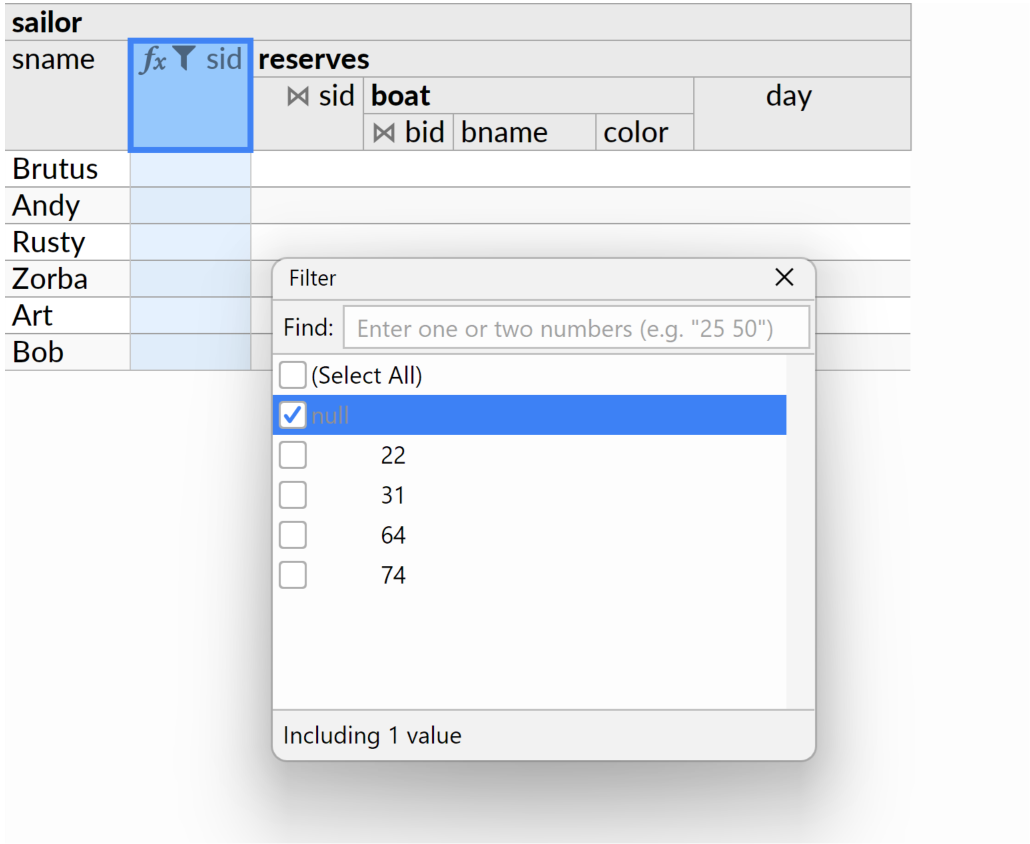
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3a: "Find sailors who reserved no boat."

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

Reference formula version (2/2)



sailor						
sname	𝑓𝑥 ▼ sid	reserves				
		⋈ sid	boat			day
			⋈ bid	bname	color	
Brutus						
Andy						
Rusty						
Zorba						
Art						
Bob						

Filter

Find: Enter one or two numbers (e.g. "25 50")

☐ (Select All)
☑ null
☐ 22
☐ 31
☐ 64
☐ 74

Including 1 value

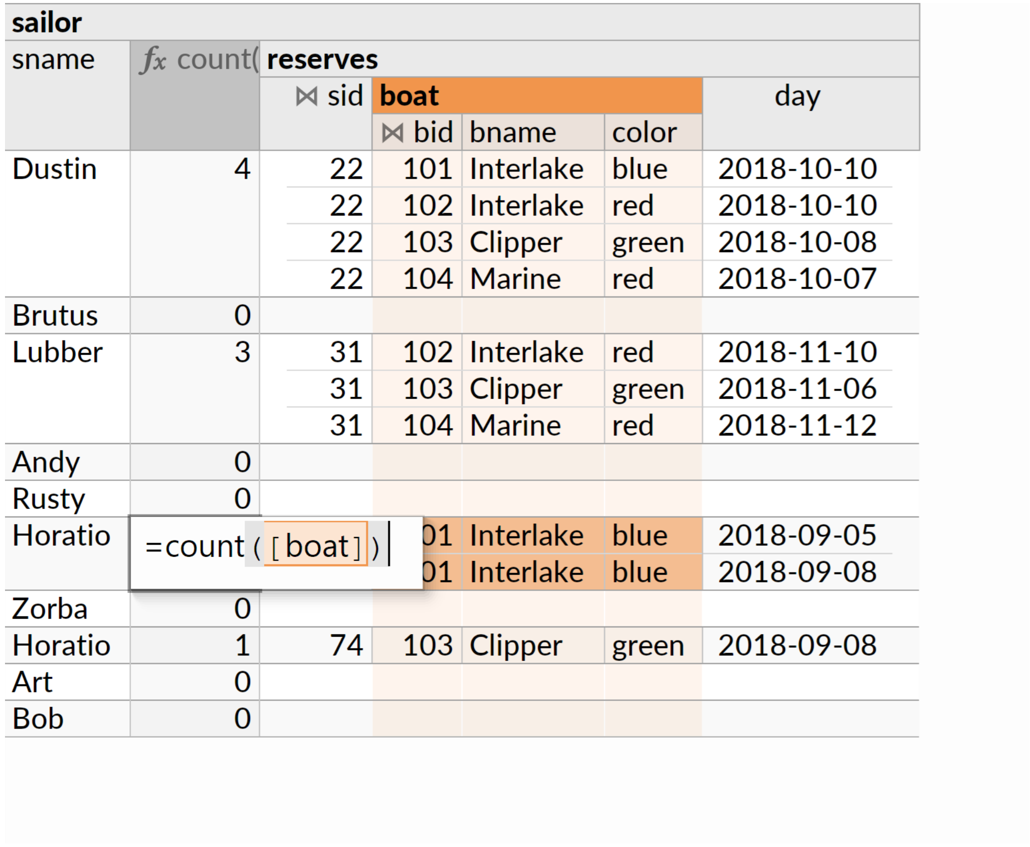
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3a: "Find sailors who reserved no boat."

```
select S.sname
from Sailor S
left join Reserves R
on S.sid = R.sid
group by S.sid, S.sname
having count(R.sid) = 0
```

Count formula version (1/2)



sailor						
sname	𝑓𝑥 count(	reserves				
		⋈ sid	boat			day
			⋈ bid	bname	color	
Dustin	4	22	101	Interlake	blue	2018-10-10
		22	102	Interlake	red	2018-10-10
		22	103	Clipper	green	2018-10-08
		22	104	Marine	red	2018-10-07
Brutus	0					
Lubber	3	31	102	Interlake	red	2018-11-10
		31	103	Clipper	green	2018-11-06
		31	104	Marine	red	2018-11-12
Andy	0					
Rusty	0					
Horatio	=count([boat])		01	Interlake	blue	2018-09-05
			01	Interlake	blue	2018-09-08
Zorba	0					
Horatio	1	74	103	Clipper	green	2018-09-08
Art	0					
Bob	0					

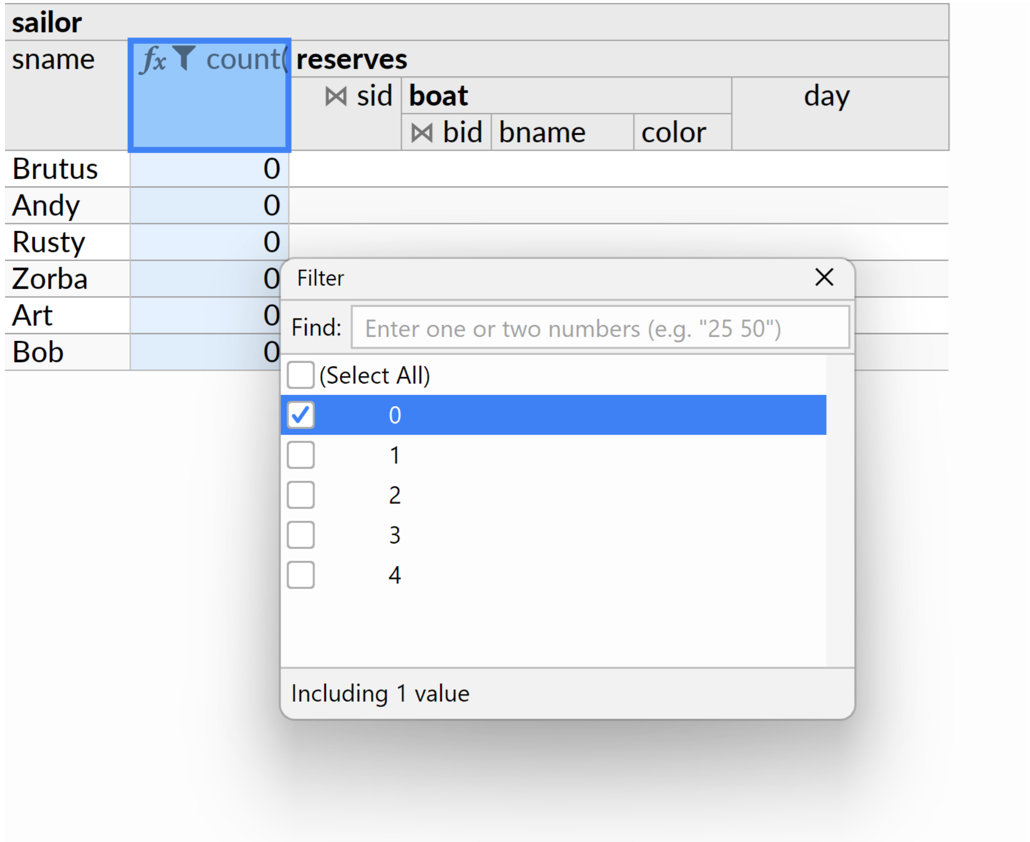
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD / Ultorg (actual screenshots)

Q3a: "Find sailors who reserved no boat."

```
select S.sname
from Sailor S
left join Reserves R
on S.sid = R.sid
group by S.sid, S.sname
having count(R.sid) = 0
```

Count formula version (2/2)



sailor						
sname	count(	reserves				
		sid	boat			day
			bid	bname	color	
Brutus	0					
Andy	0					
Rusty	0					
Zorba	0					
Art	0					
Bob	0					

Filter

Find: Enter one or two numbers (e.g. "25 50")

- ☐ (Select All)
- ☑ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4

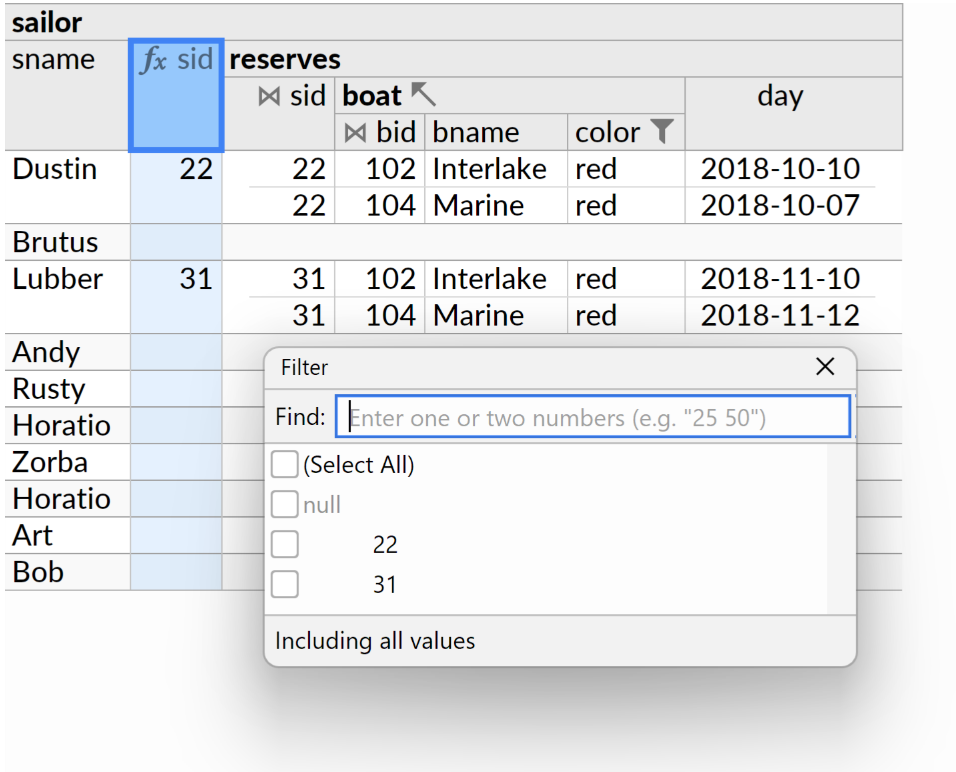
Including 1 value

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3b: "Find sailors who reserved no red boat."

```
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
```



sailor						
sname	𝑓𝑥 sid	reserves				
		⋈ sid	boat			day
			⋈ bid	bname	color	
Dustin	22	22	102	Interlake	red	2018-10-10
		22	104	Marine	red	2018-10-07
Brutus						
Lubber	31	31	102	Interlake	red	2018-11-10
		31	104	Marine	red	2018-11-12
Andy						
Rusty						
Horatio						
Zorba						
Horatio						
Art						
Bob						

Filter ✕

Find: Enter one or two numbers (e.g. "25 50")

☐ (Select All)
☐ null
☐ 22
☐ 31

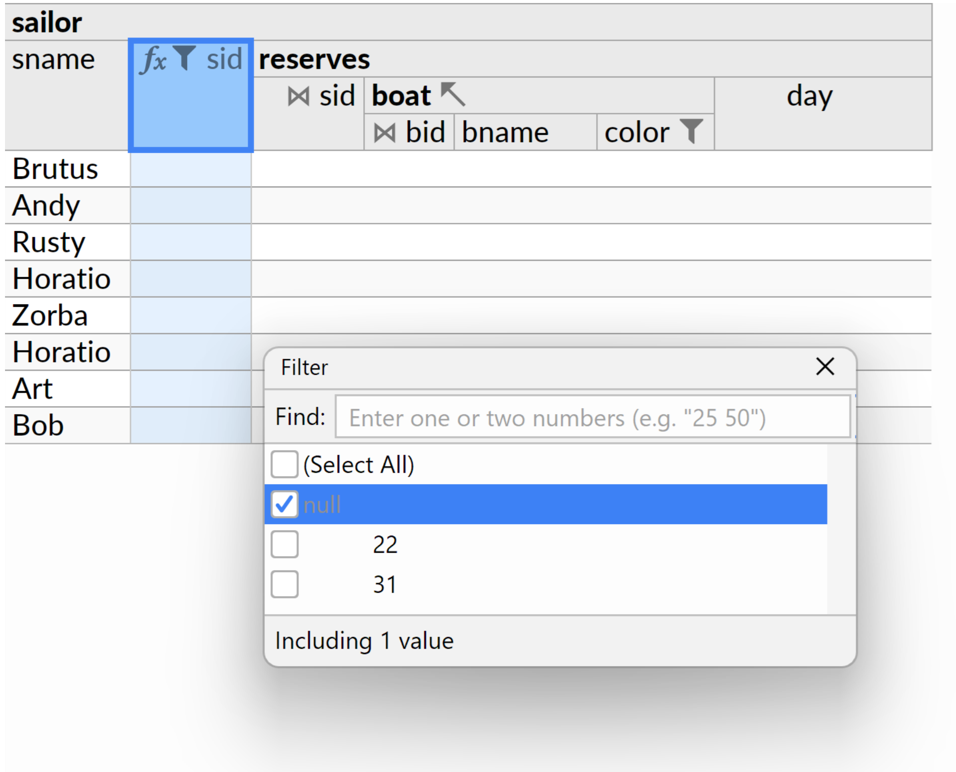
Including all values

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3b: "Find sailors who reserved no red boat."

```
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
```



sailor						
sname	*fx* ▼ sid	**reserves**				
		⋈ sid	**boat** ↖			day
			⋈ bid	bname	color ▼	
Brutus						
Andy						
Rusty						
Horatio						
Zorba						
Horatio						
Art						
Bob						

Filter ✕

Find: Enter one or two numbers (e.g. "25 50")

☐ (Select All)
☑ null
☐ 22
☐ 31

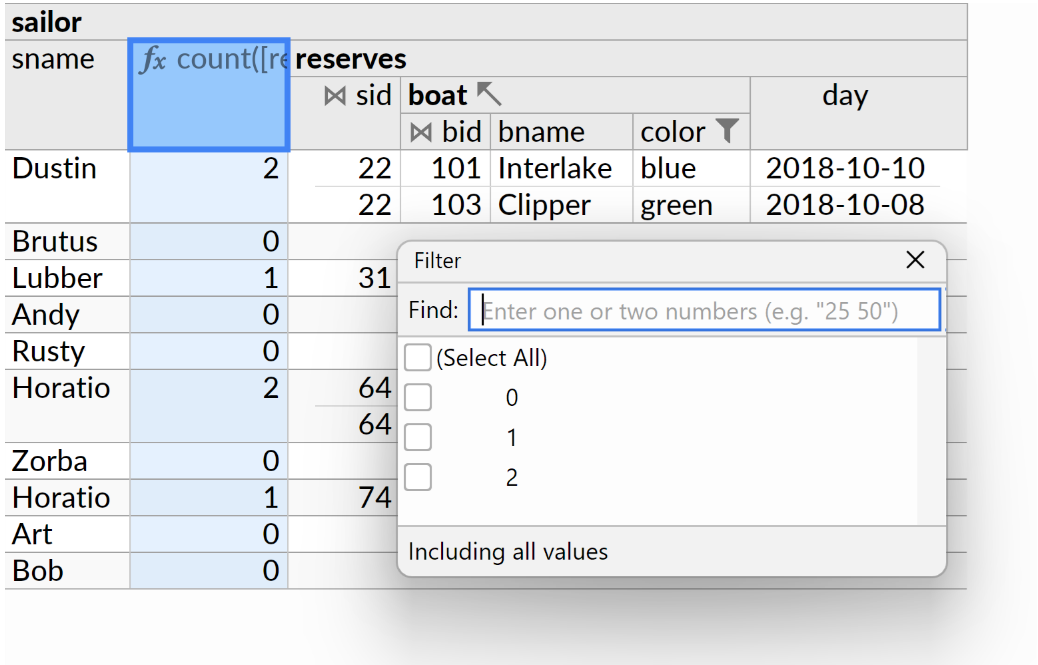
Including 1 value

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q3: "Find sailors who reserved only red boats."

```
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
```



sailor						
sname	𝑓𝑥 count([re	**reserves**				
		⋈ sid	**boat** ↖			day
			⋈ bid	bname	color	
Dustin	2	22	101	Interlake	blue	2018-10-10
		22	103	Clipper	green	2018-10-08
Brutus	0					
Lubber	1	31				
Andy	0					
Rusty	0					
Horatio	2	64				
		64				
Zorba	0					
Horatio	1	74				
Art	0					
Bob	0					

Filter ×

Find: Enter one or two numbers (e.g. "25 50")

☐ (Select All)
☐ 0
☐ 1
☐ 2

Including all values

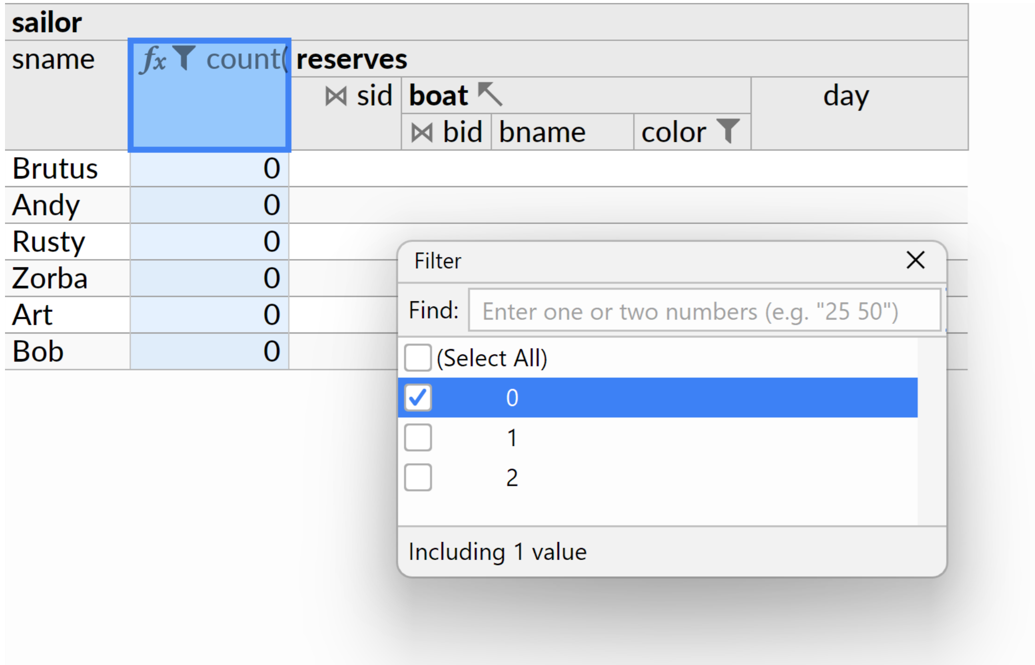
Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD / Ultorg (actual screenshots)

Q3: "Find sailors who reserved only red boats."

```
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
```



sname	*fx* count(
Brutus	0
Andy	0
Rusty	0
Zorba	0
Art	0
Bob	0

Filter

Find: Enter one or two numbers (e.g. "25 50")

☐ (Select All)
☑ 0
☐ 1
☐ 2

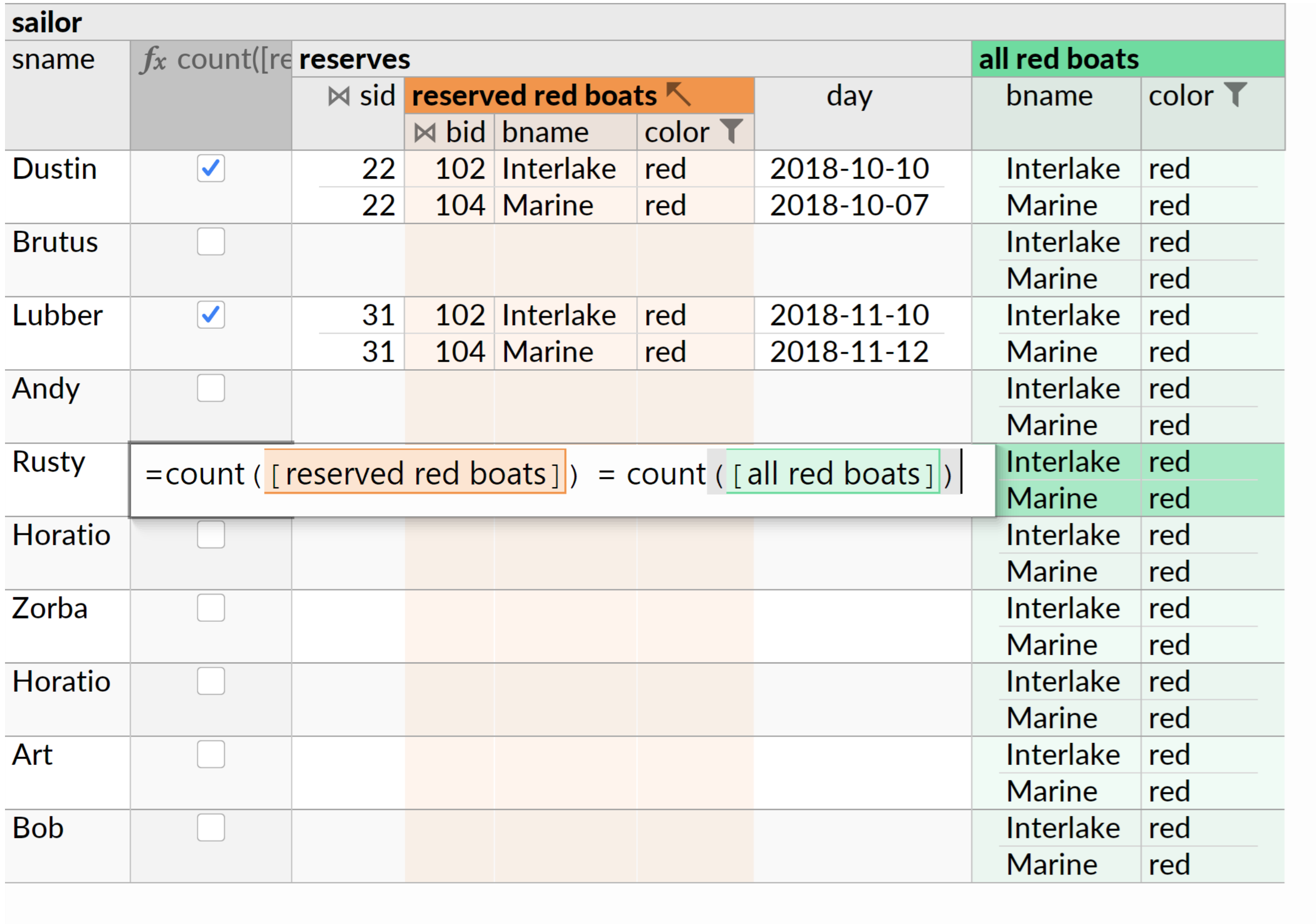
Including 1 value

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

SIEUFERD / Ultorg (actual screenshots)

Q4: "Find sailors who reserved all red boats."

```
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) =
    (select count(*)
    from Boat B
    where color = 'red')
```



sailor								
sname	*fx* count([re	reserves					all red boats	
		⋈ sid	reserved red boats			day	bname	color
			⋈ bid	bname	color			
Dustin	☑	22	102	Interlake	red	2018-10-10	Interlake	red
		22	104	Marine	red	2018-10-07	Marine	red
Brutus	☐						Interlake	red
							Marine	red
Lubber	☑	31	102	Interlake	red	2018-11-10	Interlake	red
		31	104	Marine	red	2018-11-12	Marine	red
Andy	☐						Interlake	red
							Marine	red
Rusty	=count ([reserved red boats]) = count ([all red boats])						Interlake	red
							Marine	red
Horatio	☐						Interlake	red
							Marine	red
Zorba	☐						Interlake	red
							Marine	red
Horatio	☐						Interlake	red
							Marine	red
Art	☐						Interlake	red
							Marine	red
Bob	☐						Interlake	red
							Marine	red

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD / Ultorg (actual screenshots)

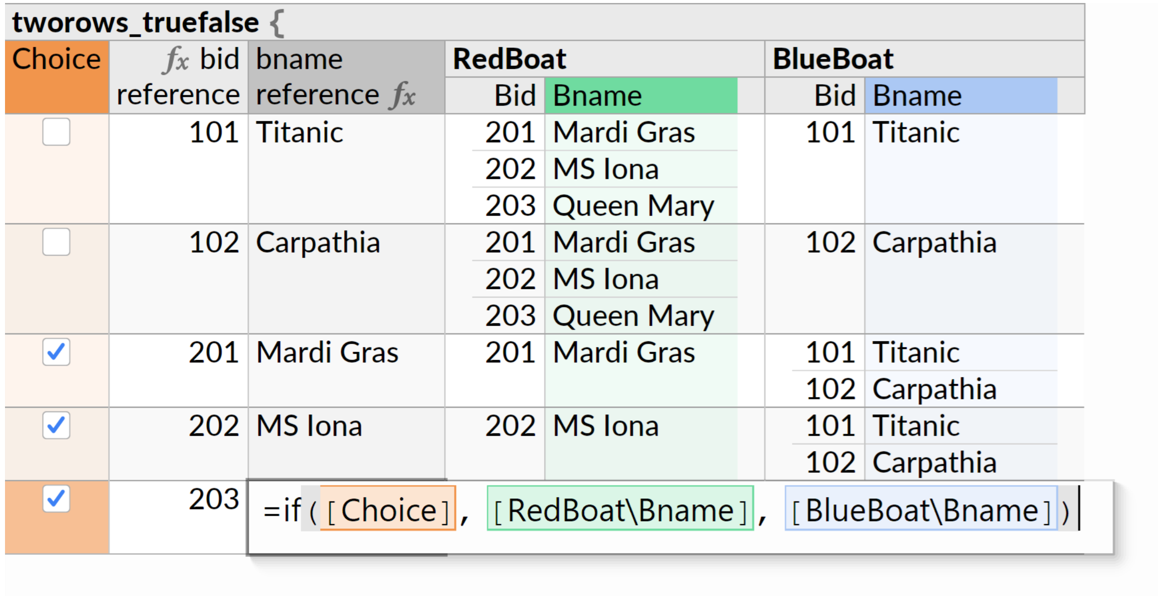
Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```



tworows_truefalse {

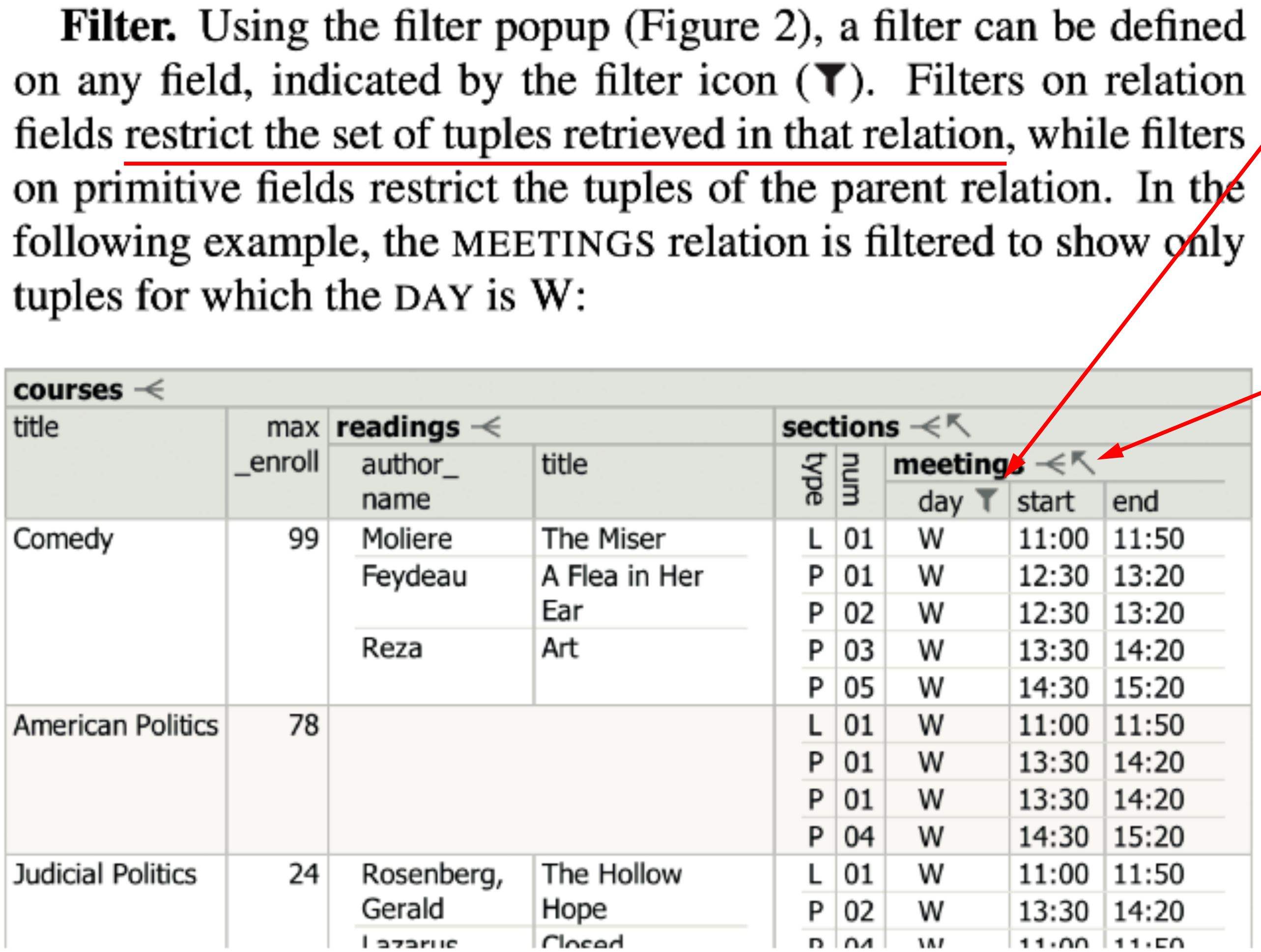
Choice	*fx* bid reference	bname reference *fx*	RedBoat		BlueBoat	
			Bid	Bname	Bid	Bname
☐	101	Titanic	201	Mardi Gras	101	Titanic
			202	MS Iona		
			203	Queen Mary		
☐	102	Carpathia	201	Mardi Gras	102	Carpathia
			202	MS Iona		
			203	Queen Mary		
☑	201	Mardi Gras	201	Mardi Gras	101	Titanic
					102	Carpathia
☑	202	MS Iona	202	MS Iona	101	Titanic
					102	Carpathia
☑	203	=if([Choice], [RedBoat\Bname], [BlueBoat\Bname])				

```
select distinct case when choice then R.bid
    else B.bid end,
     case when choice then R.bname
    else B.bname end
from RedBoat R, BlueBoat B,
    (select * from (values (true), (false)) V(choice) ) V
```

Figures courtesy of Eirik Bakke by using https://www.ultorg.com/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD



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** ≺		**sections** ≺↖				
		author_ name	title	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

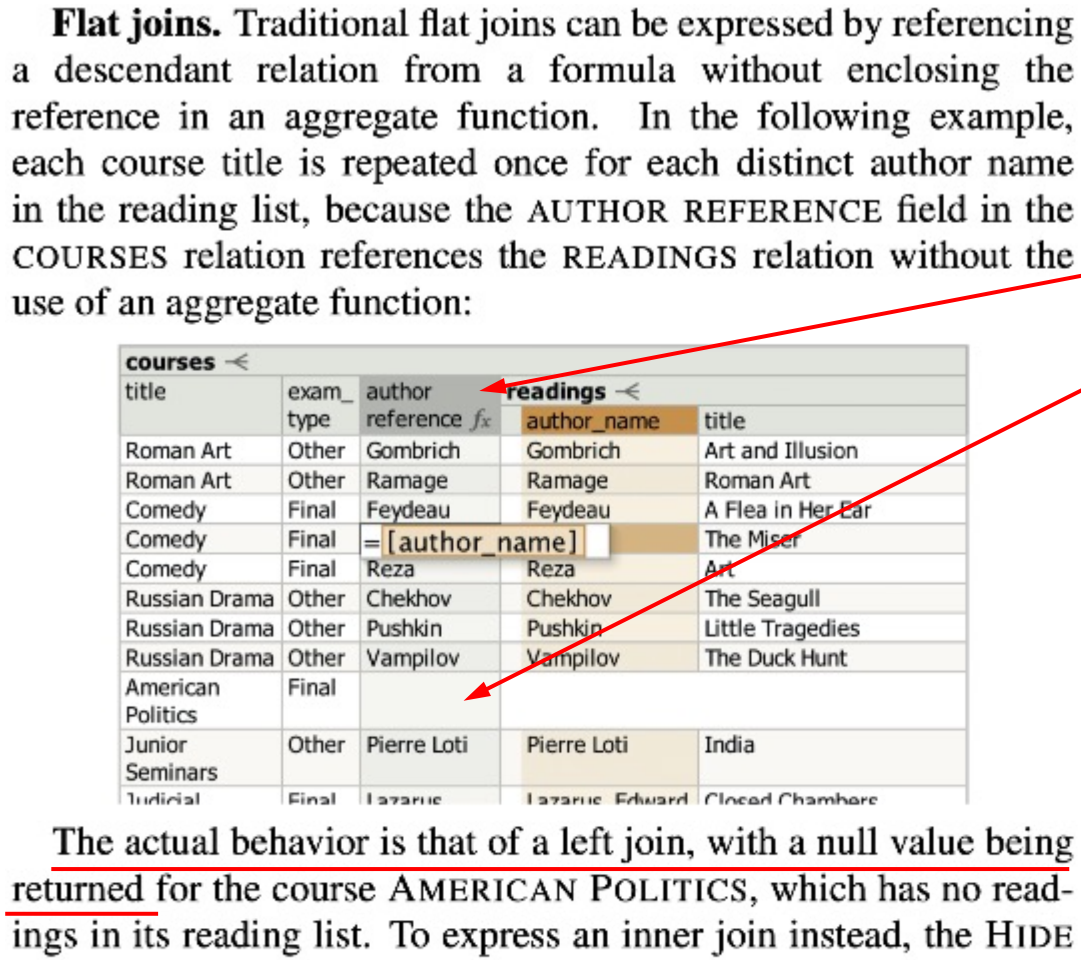
Filter icon (▼) indicates filter. Here the meetings relation is filtered to show only tuples with day = W (actual filter condition only shown in separate "filter popup" window)

"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

Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD



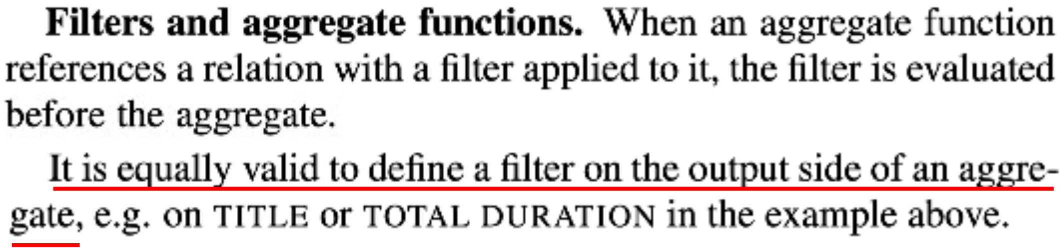
Flat joins. Traditional flat joins can be expressed by referencing a descendant relation from a formula without enclosing the reference in an aggregate function. In the following example, each course title is repeated once for each distinct author name in the reading list, because the AUTHOR REFERENCE field in the COURSES relation references the READINGS relation without the use of an aggregate function:

courses ≺

title	exam_ type	author reference *fx*	readings ≺	
			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

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

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

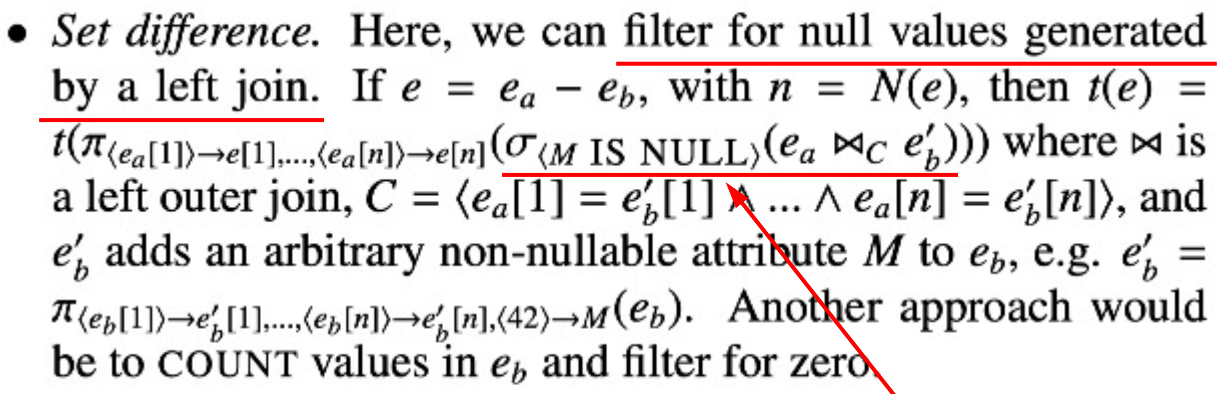


Filters and aggregate functions. When an aggregate function references a relation with a filter applied to it, the filter is evaluated before the aggregate.

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.

Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD



- *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 $t(e) = t(\pi_{\langle e_a[1]\rangle \to e[1], \dots, \langle e_a[n]\rangle \to e[n]}(\sigma_{\langle M \text{ IS NULL}\rangle}(e_a \bowtie_C e'_b)))$ where $\bowtie$ 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.

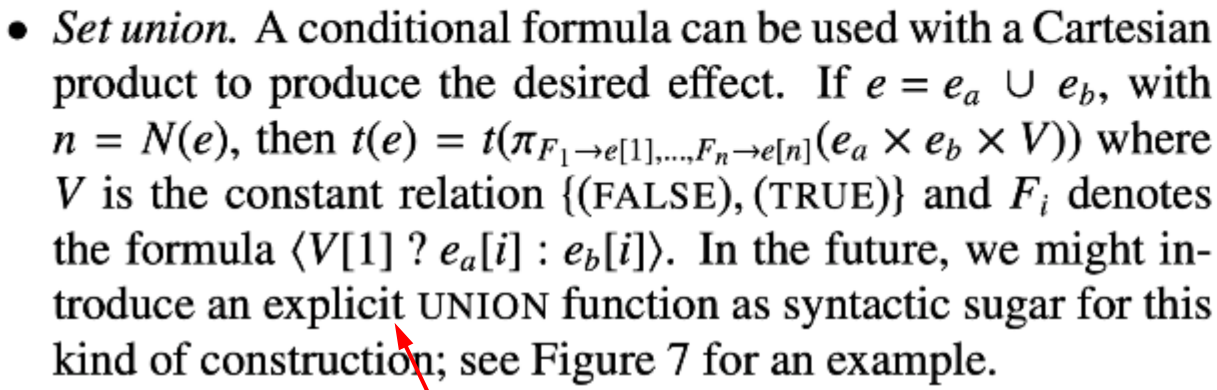
Negation (set difference) is modeled via a left join. Since filters are by applied to the relation, not the result of a join, one needs to add a "reference formula" to the relation on the left side of the join, followed by a filter

Here the filter "is null" needs to be applied to the relation resulting from the join, not the relation on the right side

Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

Union is modeled via a Cartesian Product between the two input tables, and then condition statements that leverage a table "V(choice)" that contains tuples true and false.

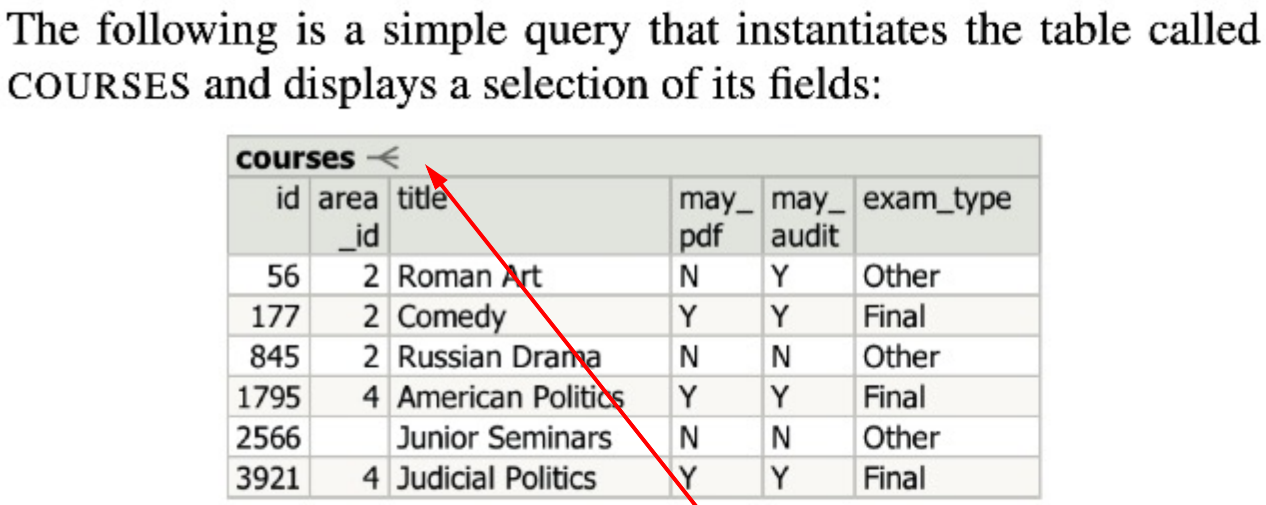


- *Set union.* A conditional formula can be used with a Cartesian product to produce the desired effect. If $e = e_a \cup e_b$, with $n = N(e)$, then $t(e) = t(\pi_{F_1 \to e[1],\dots,F_n \to e[n]}(e_a \times e_b \times V))$ where V is the constant relation {(FALSE), (TRUE)} and F_i denotes the formula $\langle V[1] \mathrel{?} e_a[i] : e_b[i] \rangle$. In the future, we might introduce an explicit UNION function as syntactic sugar for this kind of construction; see Figure 7 for an example.

Conditional statement:
if V.choice then A.i else B.i

Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

SIEUFERD



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.

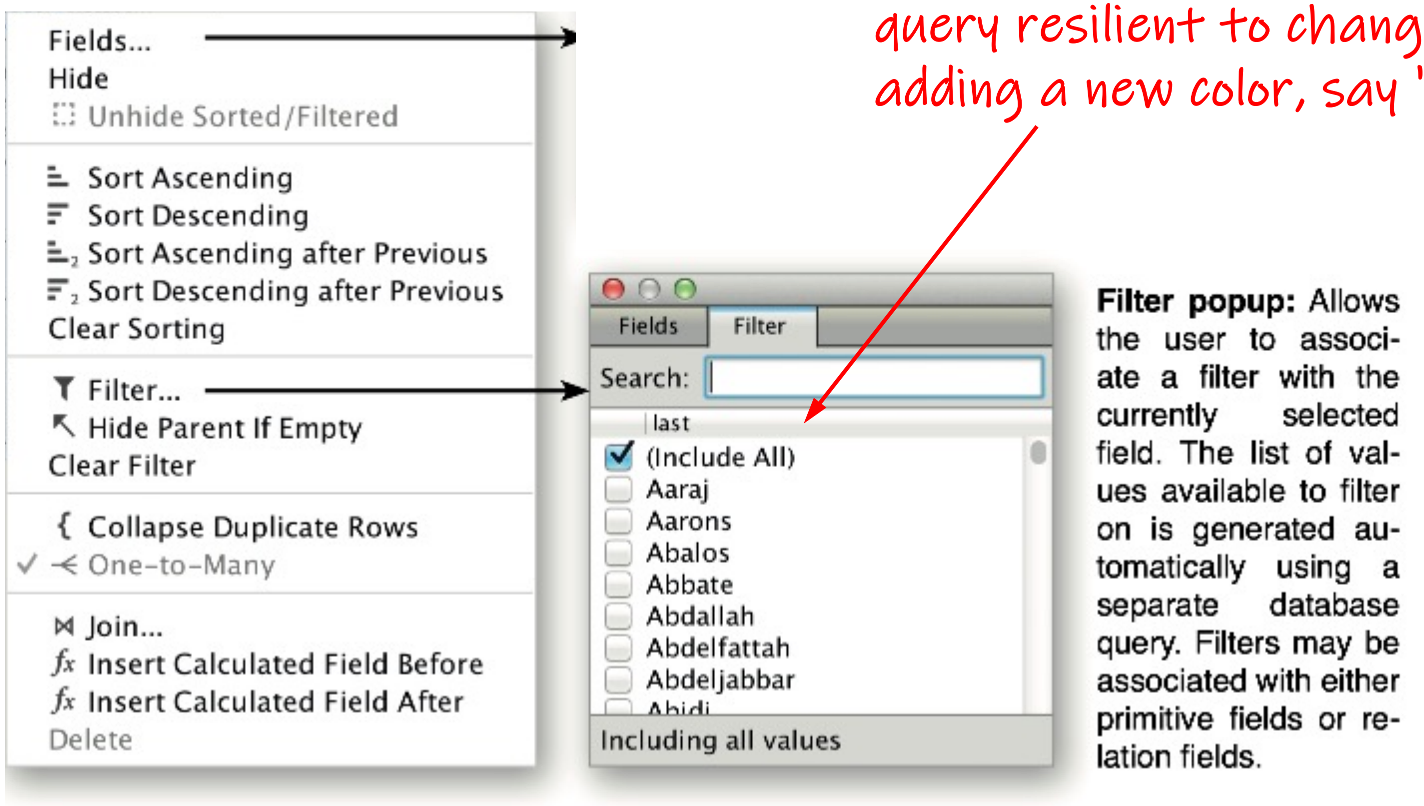
Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SIEUFERD

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"



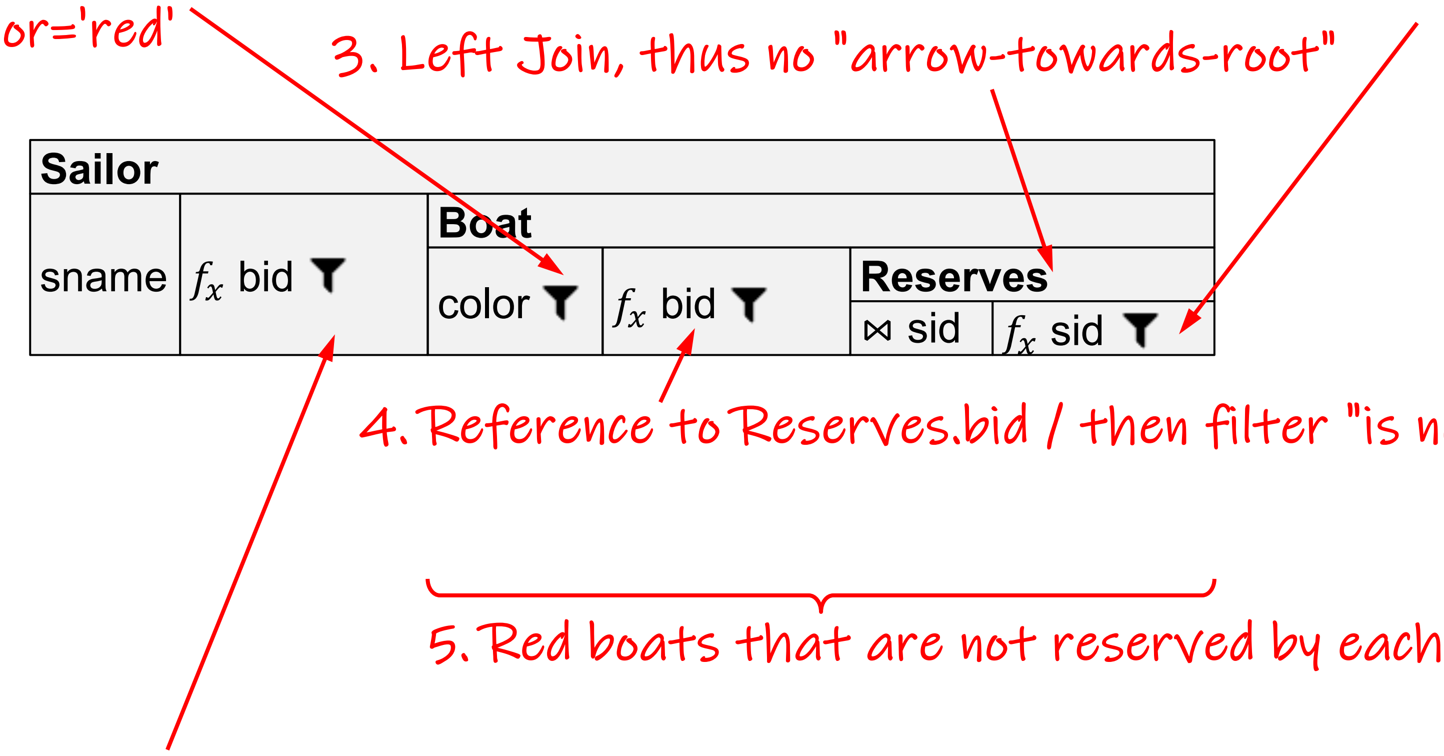
Filter popup: Allows the user to associate a filter with the currently selected field. The list of values available to filter on is generated automatically using a separate database query. Filters may be associated with either primitive fields or relation fields.

Screenshot from: Bakke, Karger. Expressive query construction through direct manipulation of nested relational results. SIGMOD 2016. https://doi.org/10.1145/2882903.2915210

SIEUFERD

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.

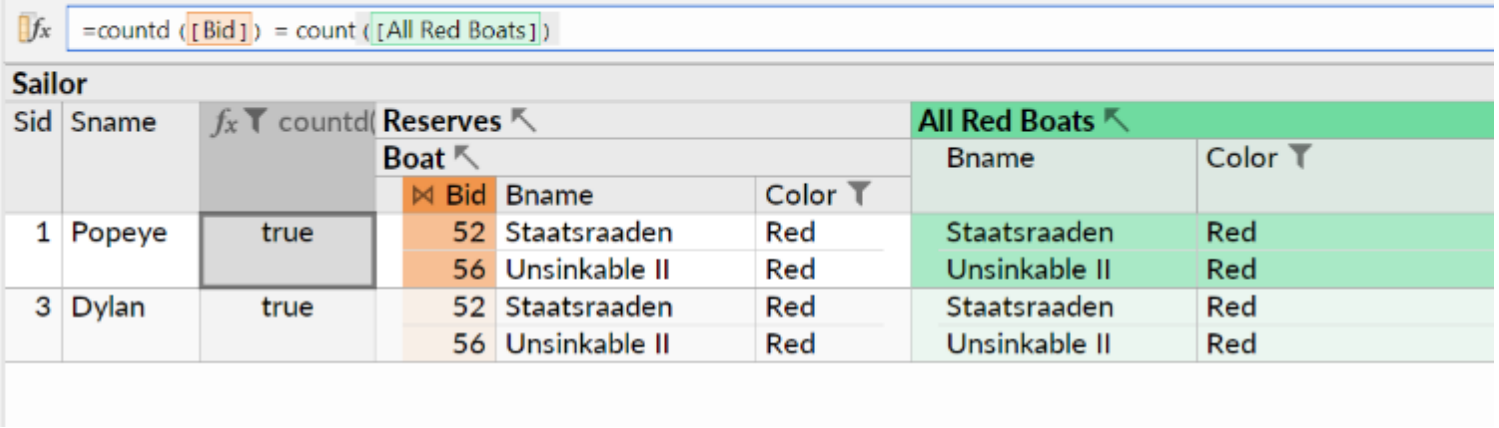
Database to run SQL queries is available as schema 341 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

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

SIEUFERD

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))
```



=countd ([Bid]) = count ([All Red Boats])

Sailor

Sid	Sname	*fx* countd(	**Reserves**			**All Red Boats**	
			Boat			Bname	Color
			⋈ Bid	Bname	Color		
1	Popeye	true	52	Staatsraaden	Red	Staatsraaden	Red
			56	Unsinkable II	Red	Unsinkable II	Red
3	Dylan	true	52	Staatsraaden	Red	Staatsraaden	Red
			56	Unsinkable II	Red	Unsinkable II	Red

```
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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

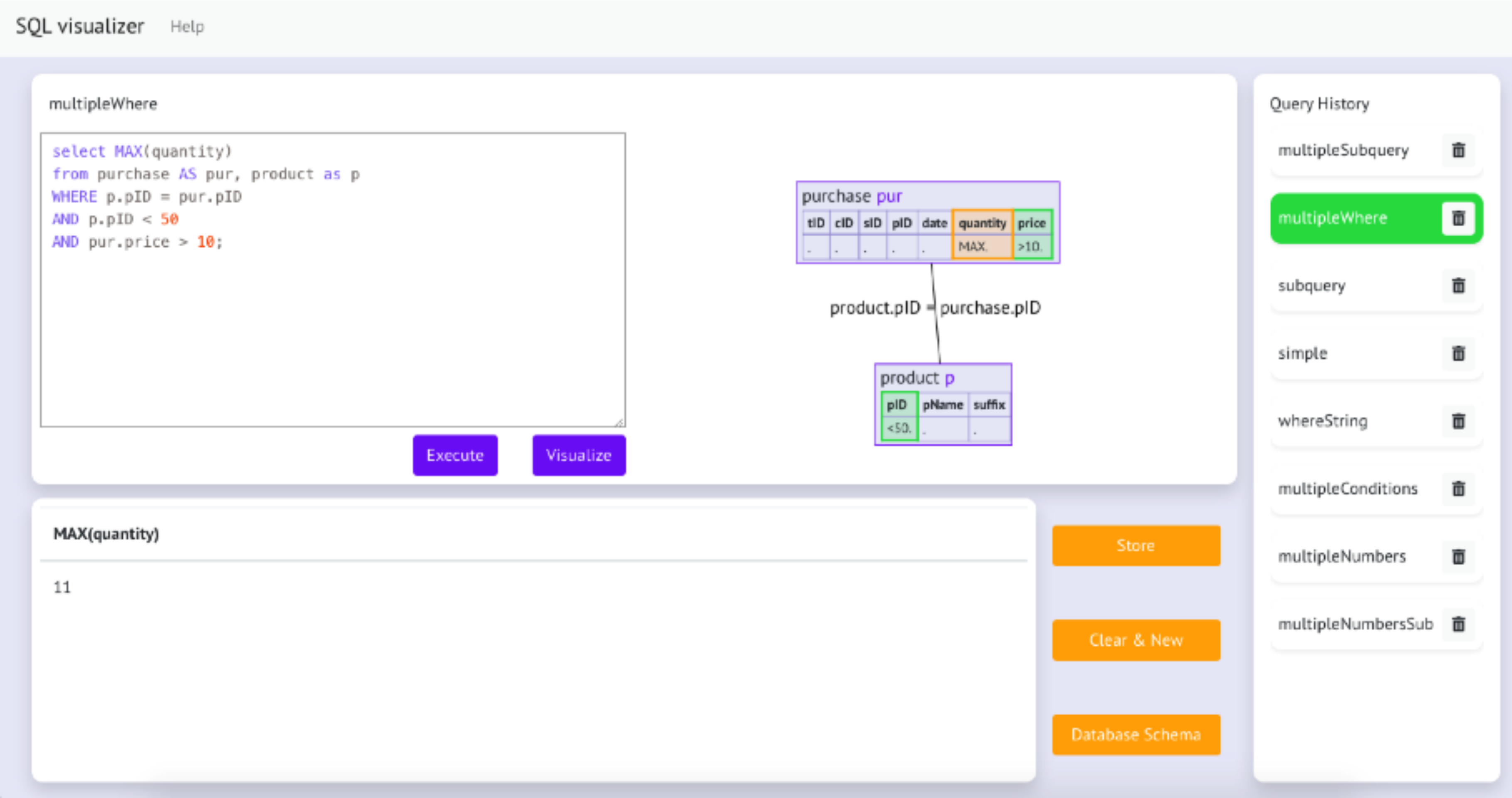
SQLVis (2021)

Sources used:

- 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

- SQLVis represents SQL queries visually while a user is composing a query.
- The goal is thus to closely capture the actual syntax of SQL (possibly also for debugging)



```
select MAX(quantity)
from purchase AS pur, product as p
WHERE p.pID = pur.pID
AND p.pID < 50
AND pur.price > 10;
```

```
MAX(quantity)
11
```

Fig source: "Miedema. Towards successful interaction between humans and databases. M. thesis. Eindhoven UoT. 2019. https://research.tue.nl/en/studentTheses/towards-successful-interaction-between-humans-and-databases "

SQLVis

Schema

Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

```
select distinct 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

SQLVis

Schema

Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

```
select distinct 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.

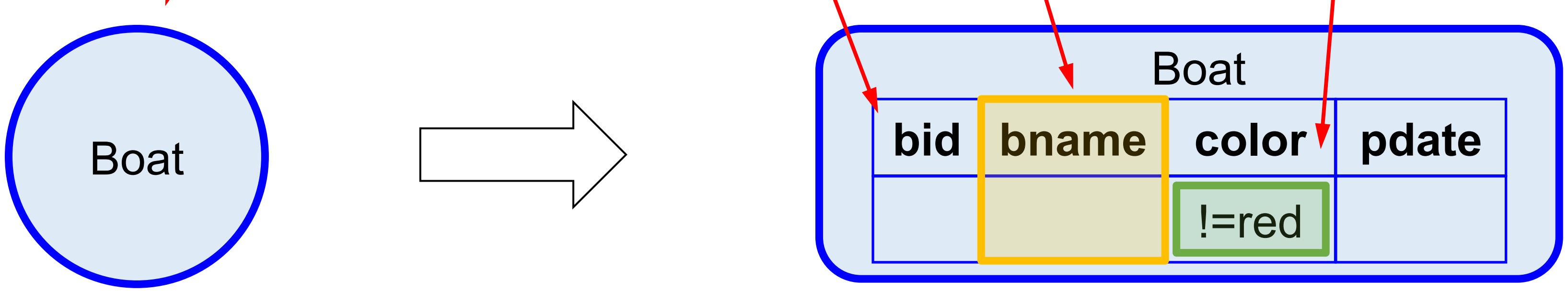


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 3. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

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

Like QBE, disjunctions are written in different rows

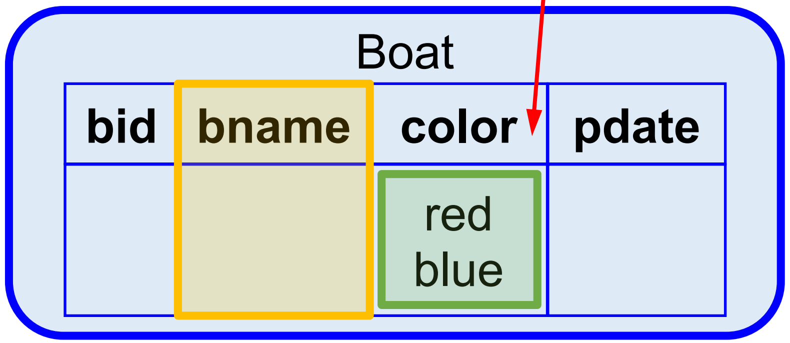


Figure drawn based on https://github.com/Giraphne/sqlvis (8/2023)

SQLVis

Schema

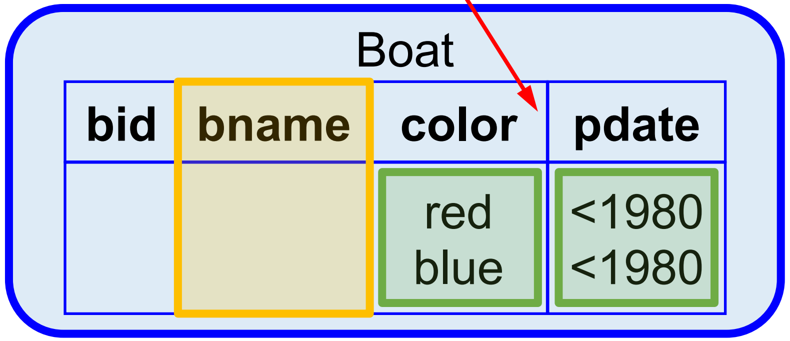
Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

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

Like QBE, more complicated Boolean expressions are written in DNF: Conditions in the same row are connected by "AND", Conditions in distinct rows are connected by "OR"

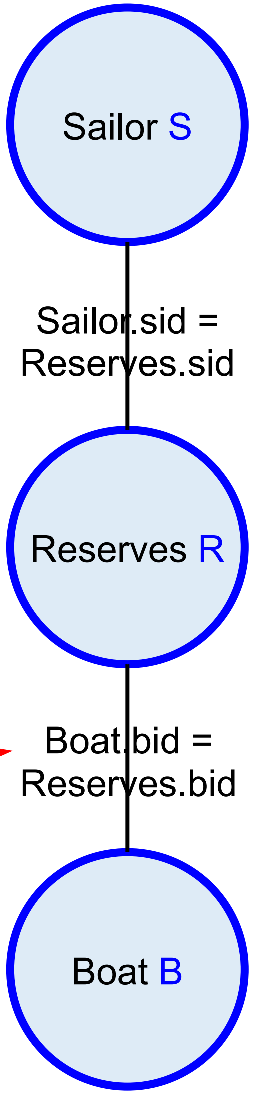
```
... where (color = 'red' and pdate < 1980)
or (color = 'blue' and pdate < 1980)
```



SQLVis

Q2: "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'
```



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

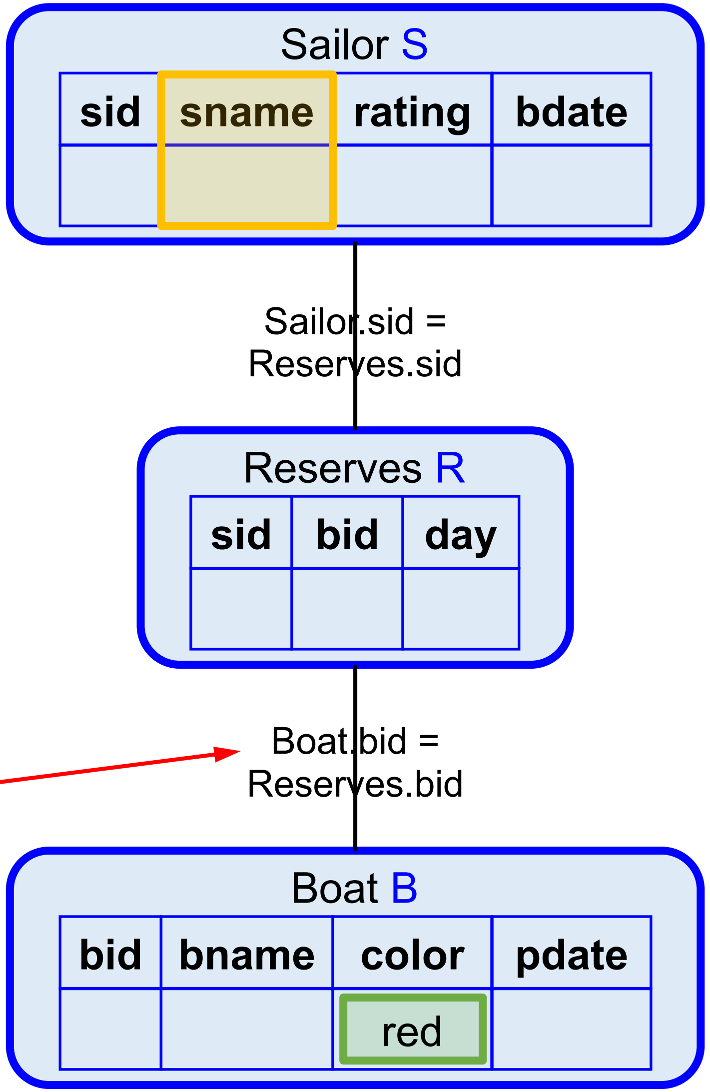
Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Q2: "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'
```



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

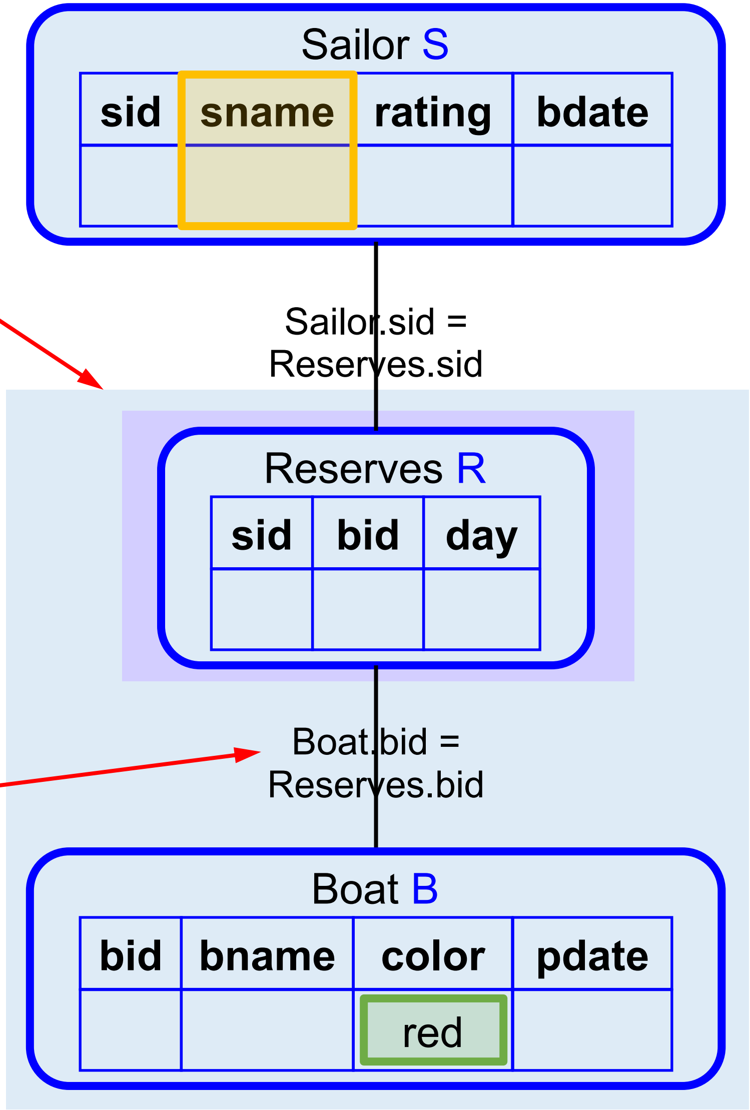
Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Q2: "Find sailors who reserved a red boat."

```
select distinct 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Q4: "Find sailors who reserved all red boats."

```
select distinct 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

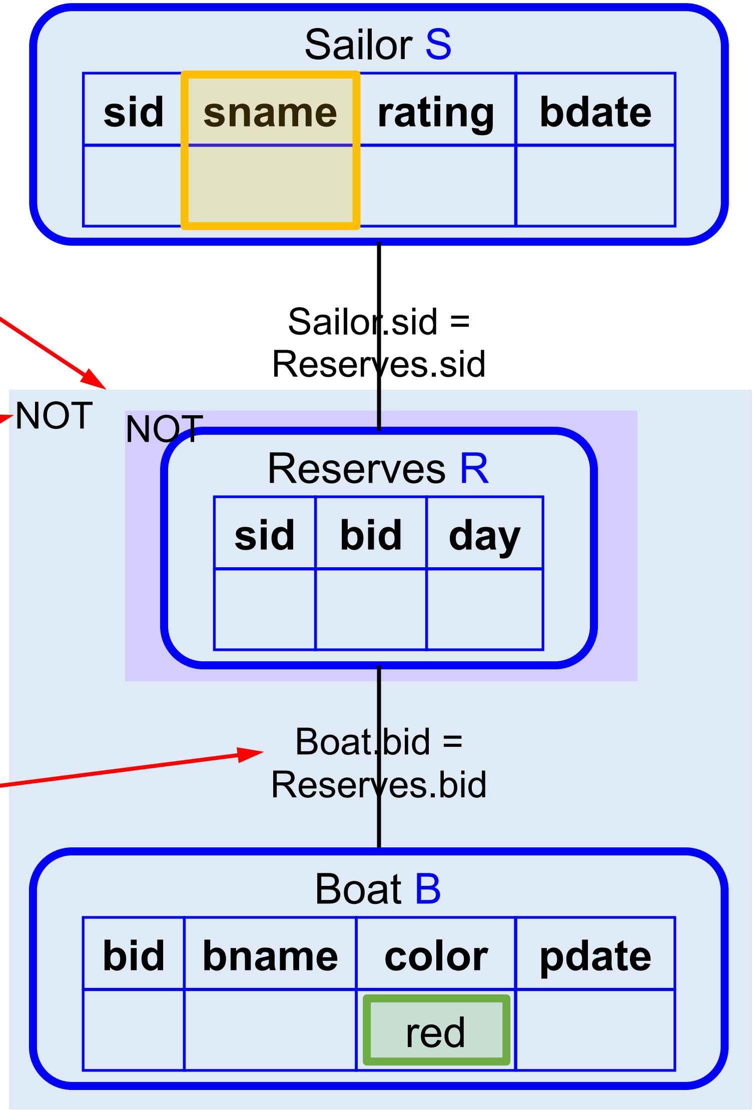


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Q3: "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'))
```

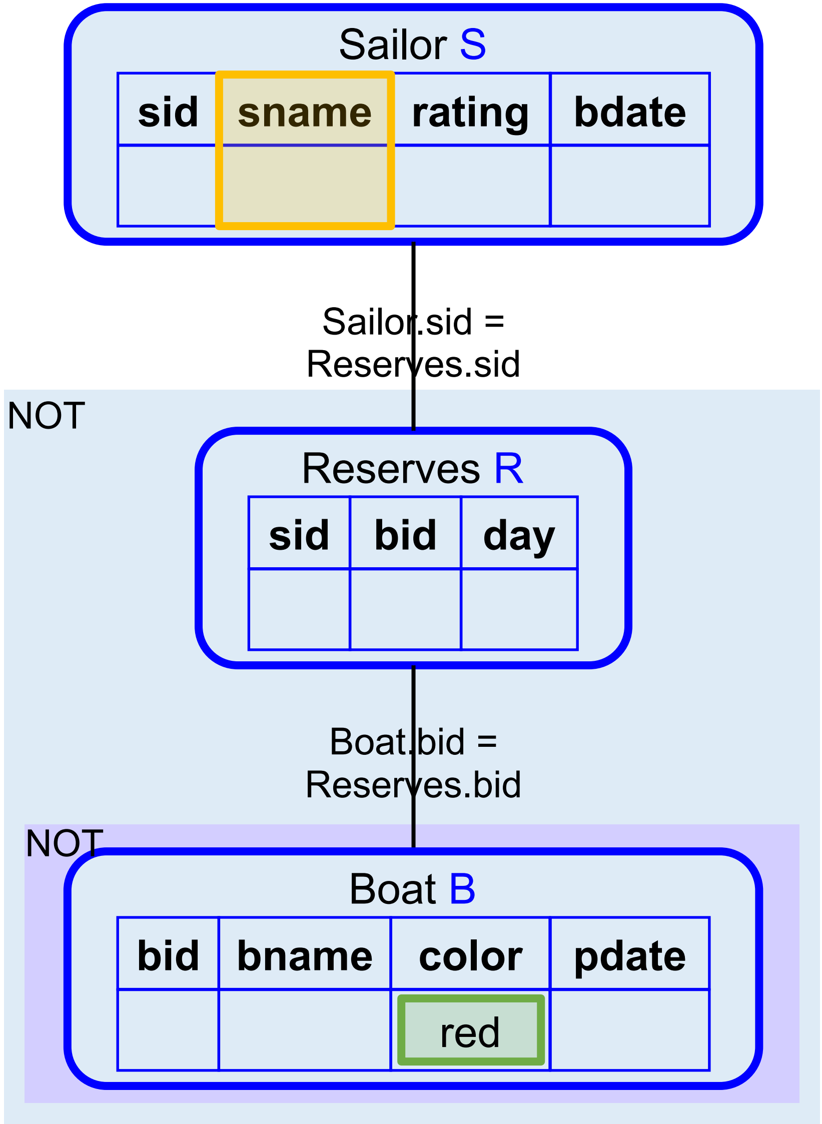


Figure drawn based on Miedema, Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. VL/HCC 2021. Fig 4b. https://doi.org/10.1109/VL/HCC51201.2021.9576431
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

Schema

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

A union of queries is not supported

SQLVis (2021)
Backup

SQLVis

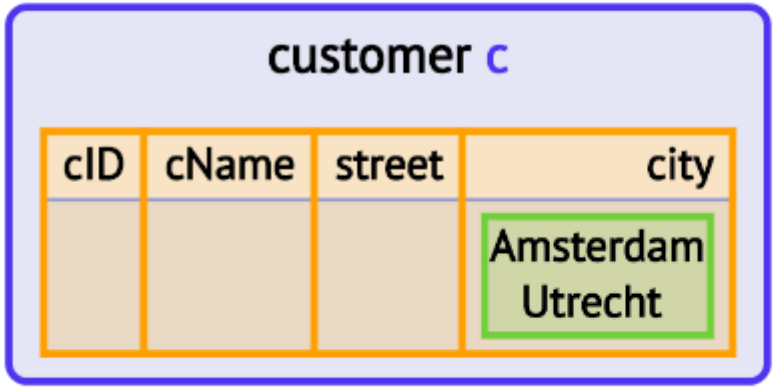
```
SELECT *
FROM store
WHERE city != London;
```

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

SQLVis

```
SELECT *
FROM customer AS c
WHERE city = "Amsterdam" OR city = "Utrecht";
```



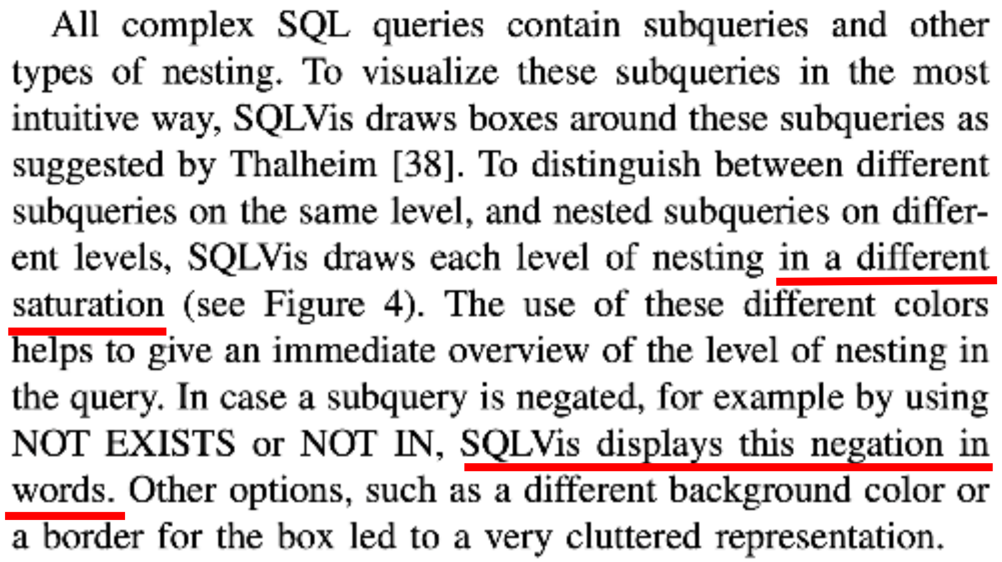
Source: https://github.com/Giraphne/sqlvis (8/2023)

SQLVis

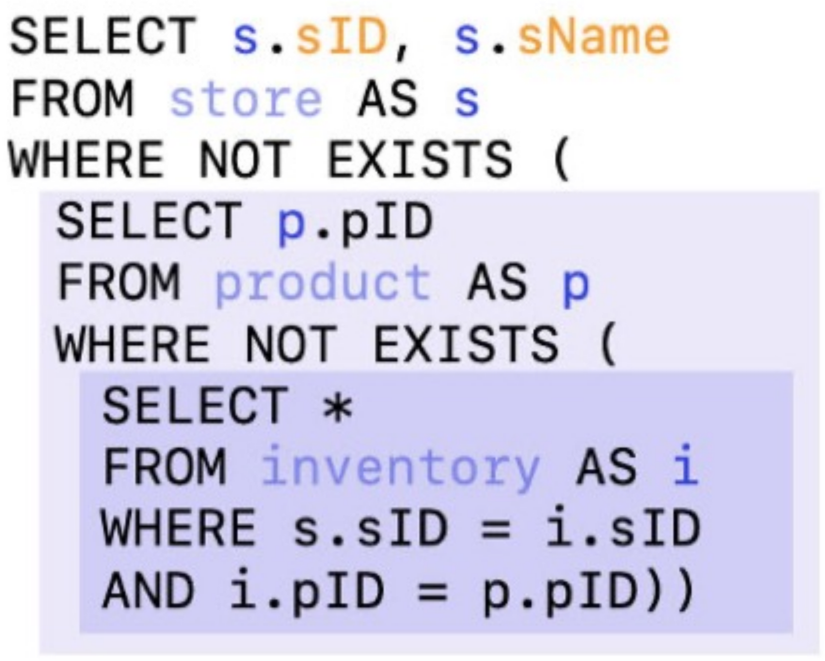
```sql
SELECT c.cName
FROM customer AS c, purchase AS p
WHERE p.cID = c.cID;
```

Source: https://github.com/Giraphne/sqlvis (8/2023)
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

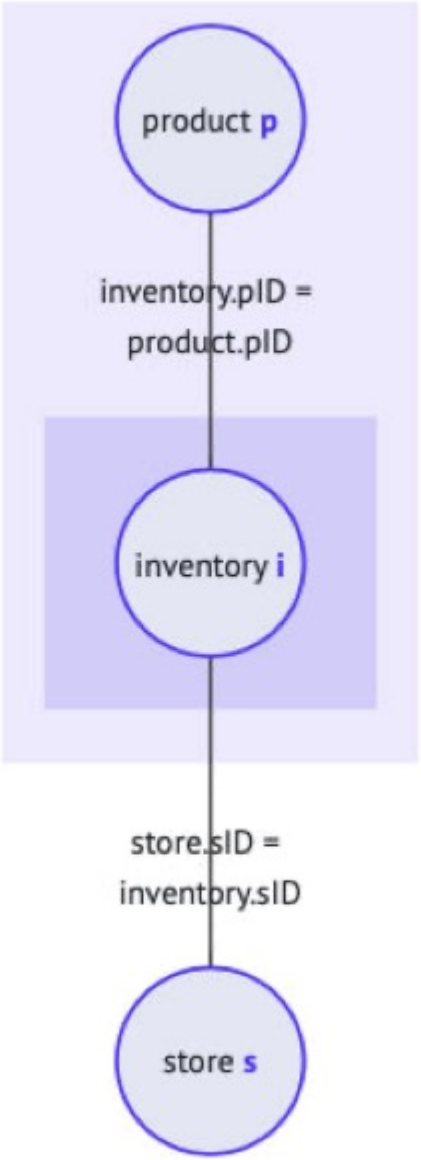
SQLVis



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.

Source: 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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Part 5: Modern Visual Query Representations (after 1970)

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)
11. Relational Diagrams (2024)

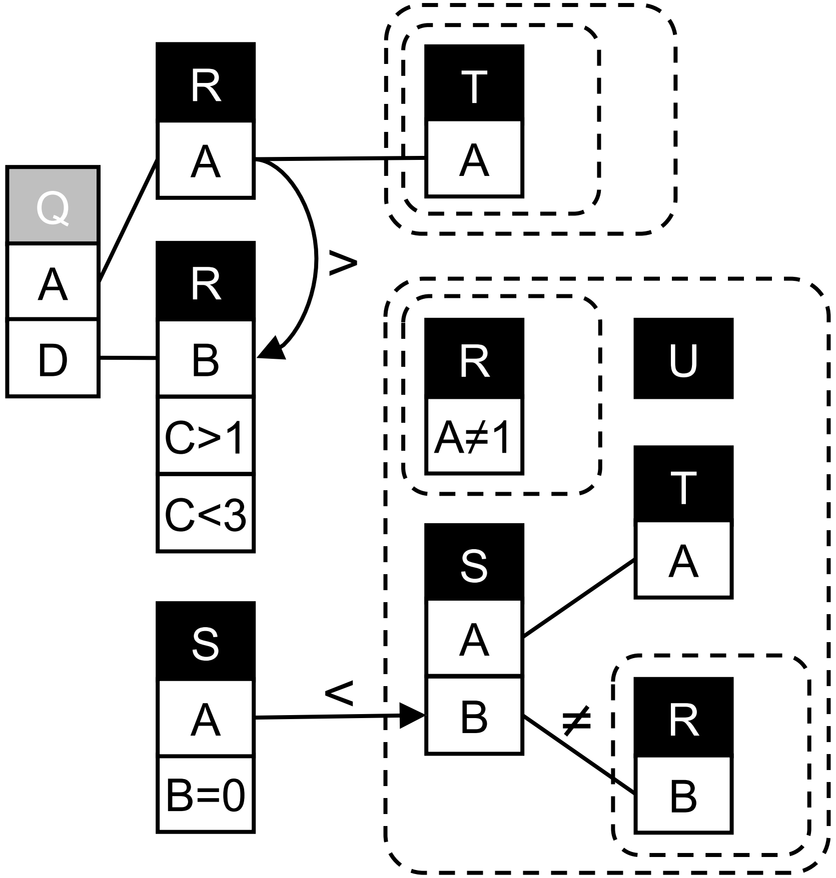
Relational Diagrams (2024)

Sources used:

- Gatterbauer, Dunne. On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages. SIGMOD 2024. https://dl.acm.org/doi/pdf/10.1145/3639316

Relational Diagrams

- "Relational Diagrams" are a variant of QueryVis, that resolves prior ambiguities and that adds expressivity.
- Two key design decisions are: 1) to use Peirce's nested negation boxes (instead of using arrow directions), and 2) to add union boxes.
- Both together makes Relational Diagrams relationally complete and allows an unambiguous interpretation for any valid Relational diagram. It can also represent Boolean queries (= logical sentences)



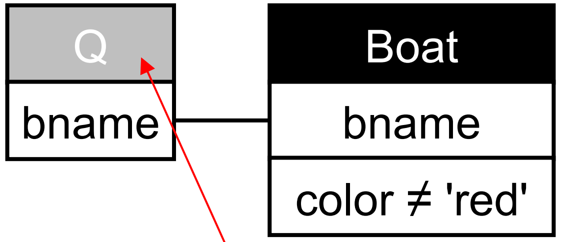
Relational Diagrams

Schema

Boat
bid
bname
color
pdate

Q1b: "Find boats that are not red."

```
select distinct bname
from Boat
where color <> 'red'
```



The visual formalism for CQs is quasi identical to QueryVis (the Q instead of SELECT on the output table is inspired by TRC)

TRC (Tuple Relational Calculus)

$\{q(\text{bname}) \mid \exists b \in \text{Boat}\ [q.\text{bname}=b.\text{bname} \land b.\text{color} \neq \text{'red'}]\}$

Relational Diagrams

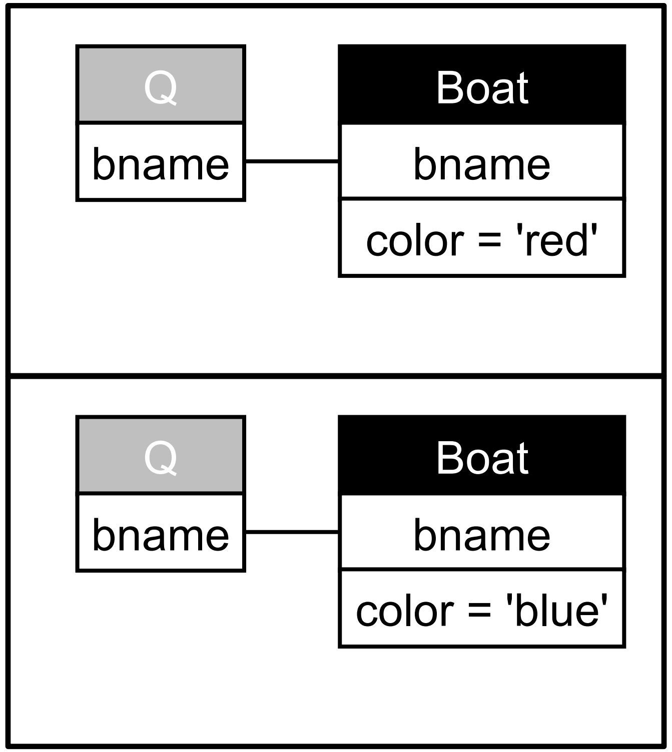
Schema

Boat
bid
bname
color
pdate

Q1: "Find boats that are red or blue."

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

Disjunctions are represented via "<u>union cells</u>": Each cell of the canvas then represents only conjunctive information, yet the relation among the different cells is disjunctive (a union of the outputs).



inspired by Venn-Peirce graphs

Strongly inspired by handling of disjunctions in Datalog

TRC (Tuple Relational Calculus)

$\{q(bname) \mid \exists b \in Boat\,[q.bname{=}b.bname \wedge (b.color{=}\text{'red'} \vee b.color{=}\text{'blue'})]\}$

$\{q(bname) \mid \exists b \in Boat\,[q.bname{=}b.bname \wedge b.color{=}\text{'red'}]$
$\quad \vee \exists b \in Boat\,[q.bname{=}b.bname \wedge b.color{=}\text{'blue'})]\}$

Datalog

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

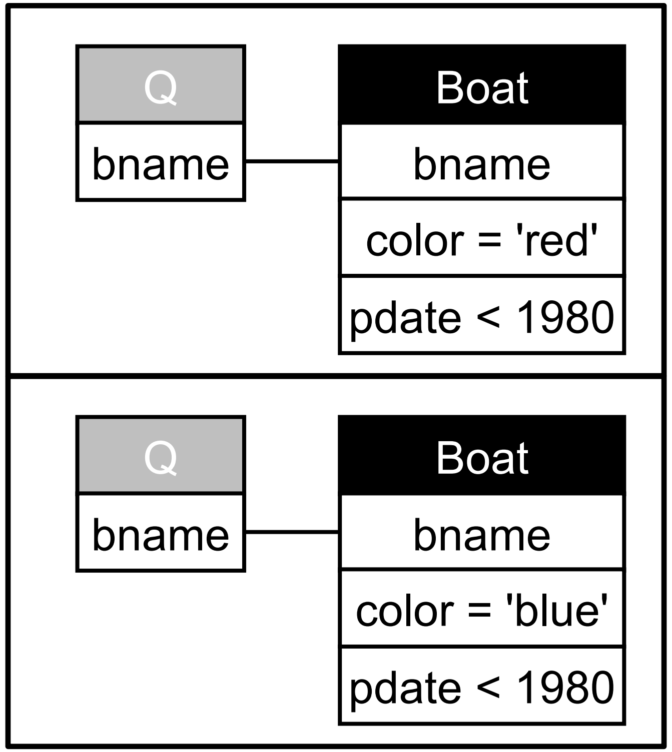
Relational Diagrams

Schema

Boat
bid
bname
color
pdate

Q1c: "Find boats that are red or blue and purchased before 1980."

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



TRC (Tuple Relational Calculus)

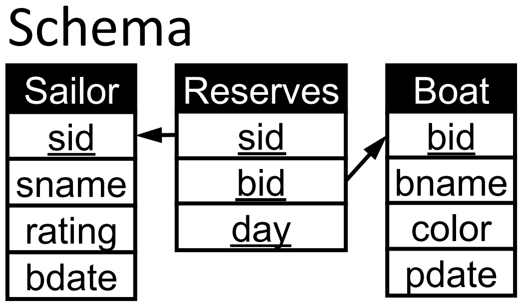
$\{q(\text{bname}) \mid \exists b{\in}\text{Boat}\,[q.\text{bname}{=}b.\text{bname} \land (b.\text{color}{=}\text{'red'} \lor b.\text{color}{=}\text{'blue'}) \land b.\text{bdate}{<}1980]\}$

$\{q(\text{bname}) \mid \exists b{\in}\text{Boat}\,[q.\text{bname}{=}b.\text{bname} \land b.\text{color}{=}\text{'red'} \land b.\text{bdate}{<}1980]$
$\lor \exists b{\in}\text{Boat}\,[q.\text{bname}{=}b.\text{bname} \land b.\text{color}{=}\text{'blue'} \land b.\text{bdate}{<}1980]\}$

Datalog

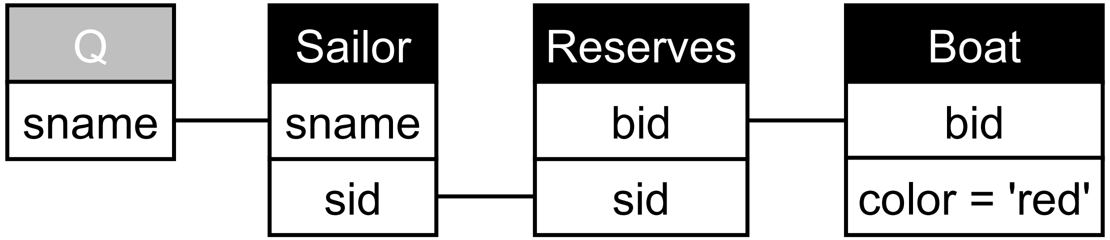
Q(x) :- Boat(_,x,'red',y), y<1980
Q(x) :- Boat(_,x,'blue',y), y<1980

Relational Diagrams



Q2: "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'
```



Relational Diagrams (just like QueryVis) show conjunctive queries just like relational schemas

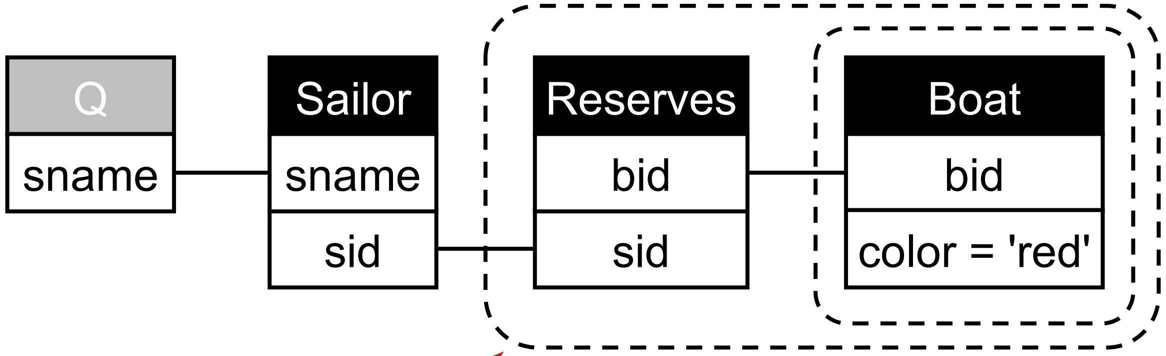
TRC (Tuple Relational Calculus)

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

Relational Diagrams

Q3: "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'))
```



Nested dashed boxes represent "not exists" (∄). Similar to Peirce (and in contrast to QueryVis), these boxes are nested, and the nesting represents the negation hierarchy

TRC (Tuple Relational Calculus)

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

Relational Diagrams

Relational Diagrams with "Peirce shading":

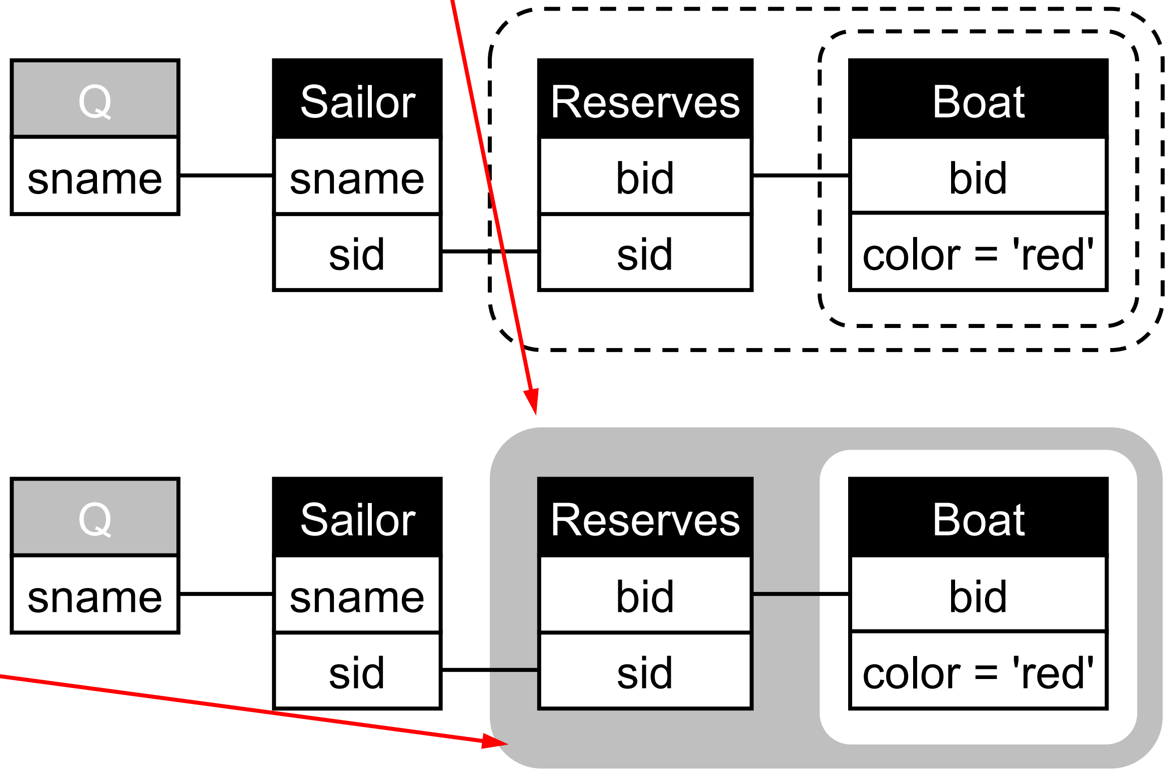
- Call "negative" (shaded) a partition that is nested in an odd number of negation boxes: 1, 3, 5, ...
- Call "positive" (not shaded) a partition that is nested in an even number of negation boxes, including zero: 0, 2, 4, ...

Q3: "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'))
```

A shaded partition that contains a nested white area (called a "scroll" by Peirce) can now be read faster as an "if (shaded) then (nested unshaded)" statement.

Here: "sailors s.t. if they reserved a boat, then it is red"



TRC (Tuple Relational Calculus)

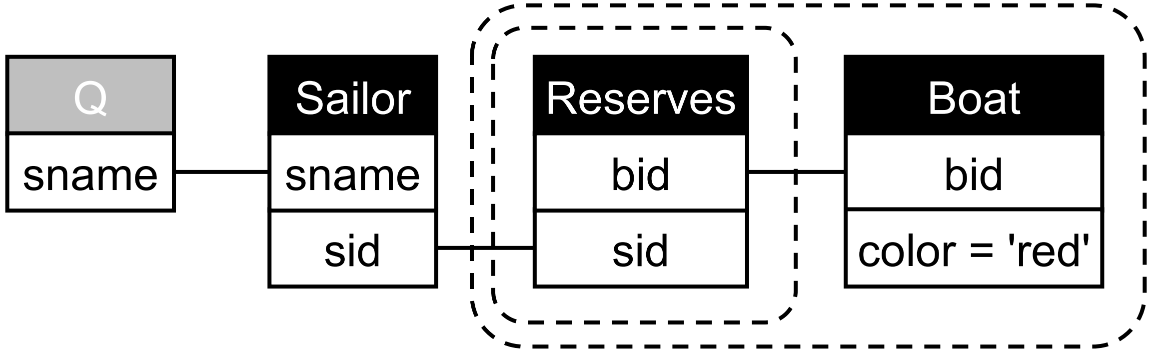
$$\{q(sname) \mid \exists s{\in}Sailor[q.sname{=}s.sname \wedge \neg(\exists r{\in}Reserves[r.sid{=}s.sid \wedge \neg(\exists b{\in}Boat[b.bid{=}r.bid \wedge b.color{=}\text{'red'}])])]\}$$
$$\{q(sname) \mid \exists s{\in}Sailor[q.sname{=}s.sname \wedge (\forall r{\in}Reserves[r.sid{=}s.sid \rightarrow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

Relational Diagrams

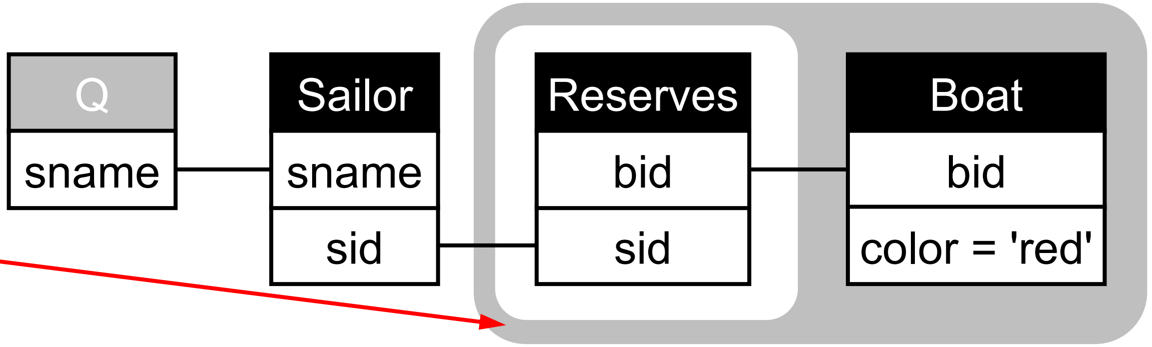
Q4: "Find sailors who reserved all red boats."

```
select distinct 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))
```

A shaded partition that contains a nested white area (called a "scroll" by Peirce) can now be read faster as an "if (shaded) then (nested unshaded)" statement.



Here: "sailors s.t. if there is a red boat, then they have reserved it"



TRC (Tuple Relational Calculus)

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

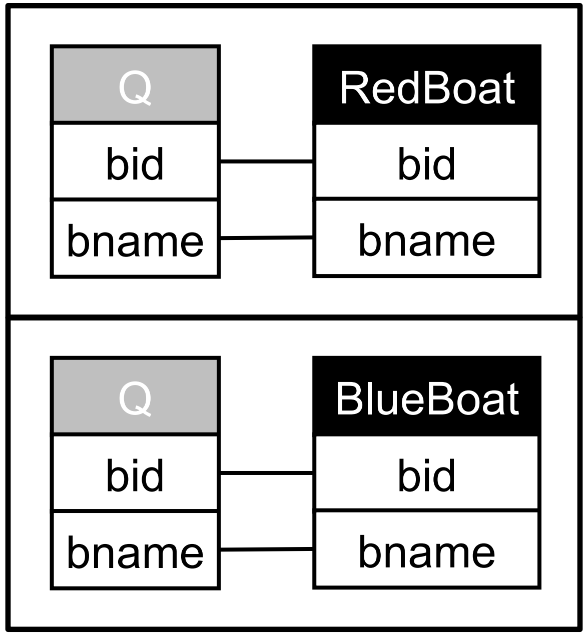
$\{q(sname) \mid \exists s \in Sailor[q.sname=s.sname \wedge (\forall b \in Boat[b.color='red' \rightarrow (\exists r \in Reserves[b.bid=r.bid \wedge r.sid=s.sid])])]\}$

Relational Diagrams

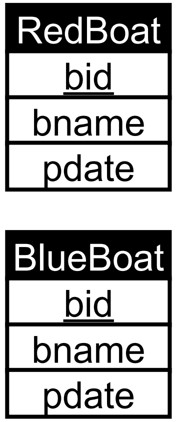
Q5: "Find boats that are red or blue."

```
select bid, bname
from RedBoat R
union
select bid, bname
from BlueBoat B
```

Unions are represented via "union cells": Each cell of the canvas represents only conjunctive information, yet the relation among the different cells is disjunctive (a union of the outputs).



Schema



Strongly inspired by handling of disjunctions in Datalog

TRC (Tuple Relational Calculus)

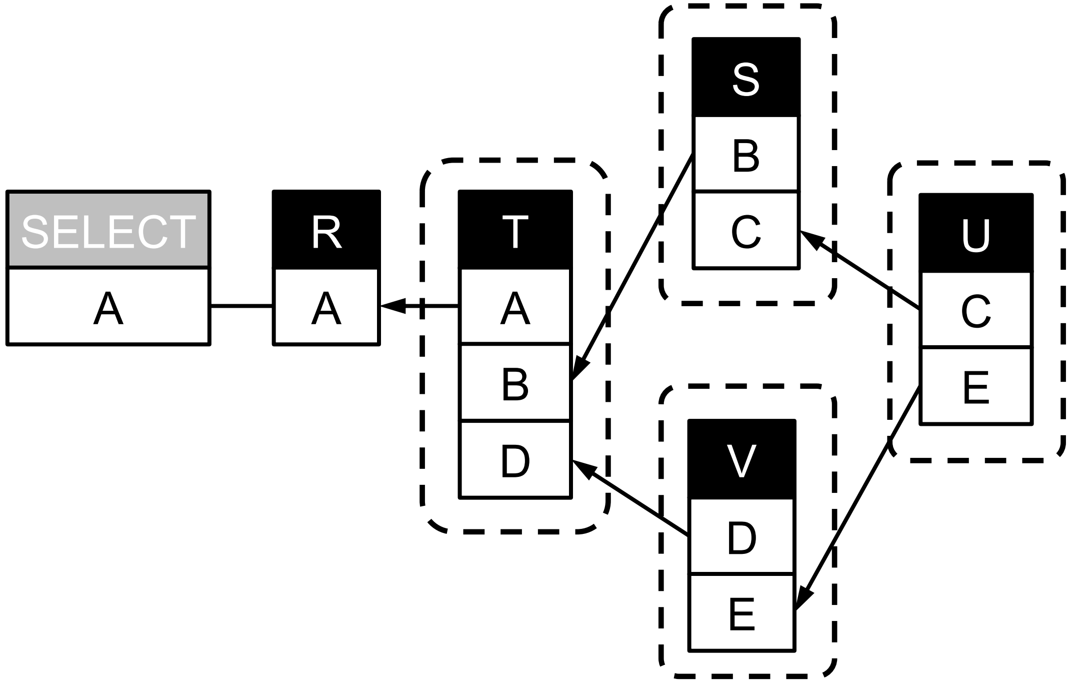
{q(bid,bname) | ∃b∈RedBoat [q.bid=b.bid ∧ q.bname=b.bname]
∨ ∃b∈BlueBoat [q.bid=b.bid ∧ q.bname=b.bname]}

Datalog

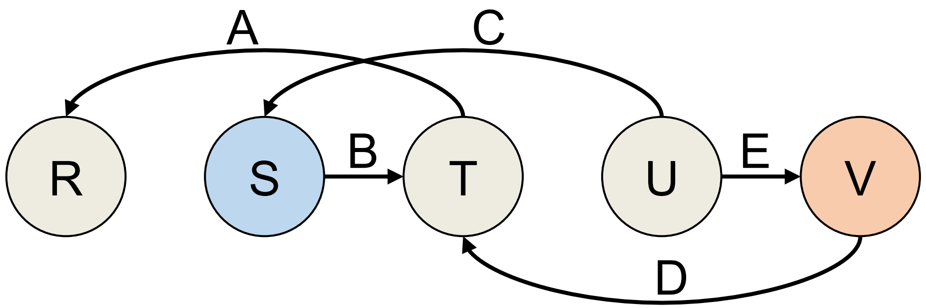
Q(x,y) :- RedBoat(x,y,_)
Q(x,y) :- BlueBoat(x,y,_)

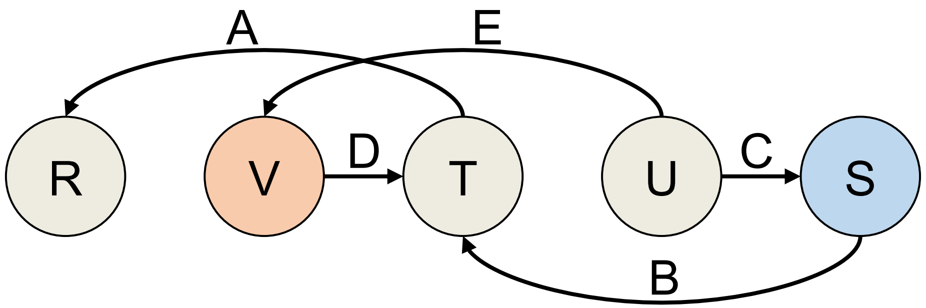
QueryVis: With nesting depth 4, arrows can become ambiguous

```
select distinct A
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 A
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))))
```

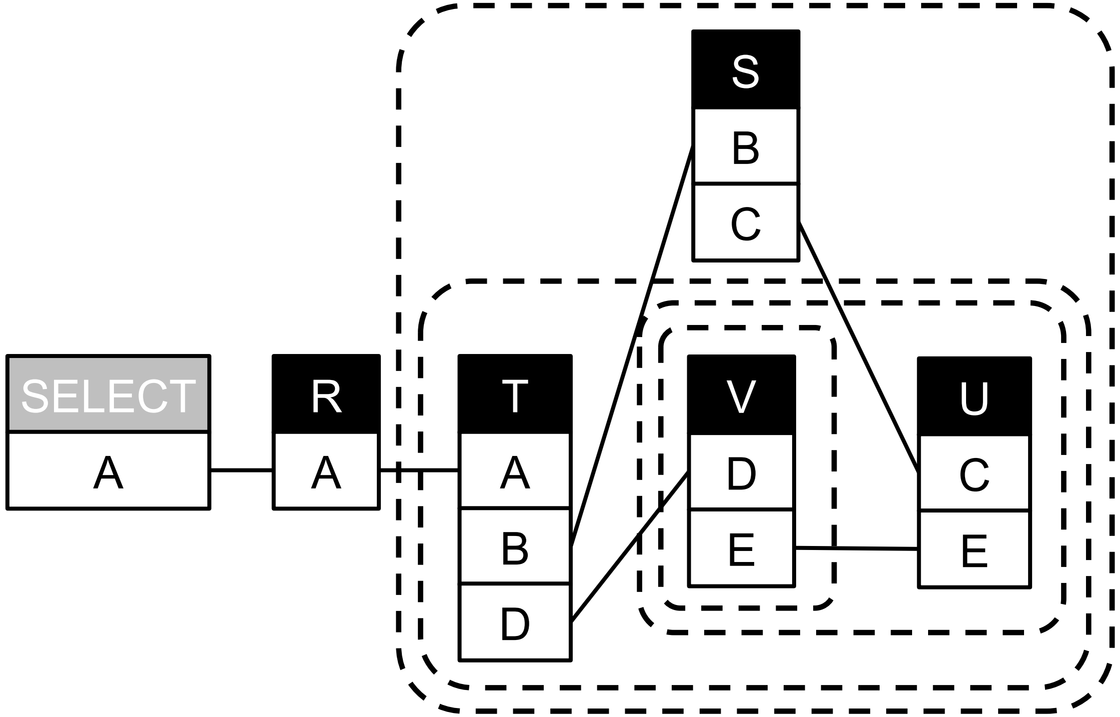




Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Relational Diagrams are unambiguous

```
select distinct A
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))))
```

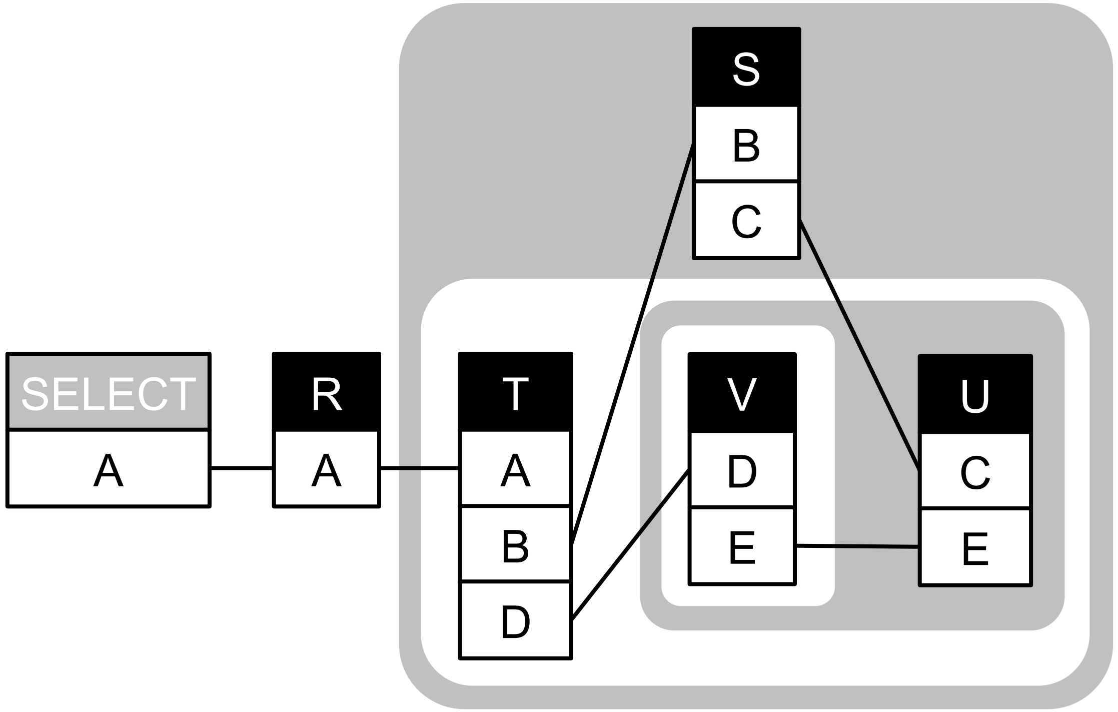


Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Relational Diagrams are unambiguous

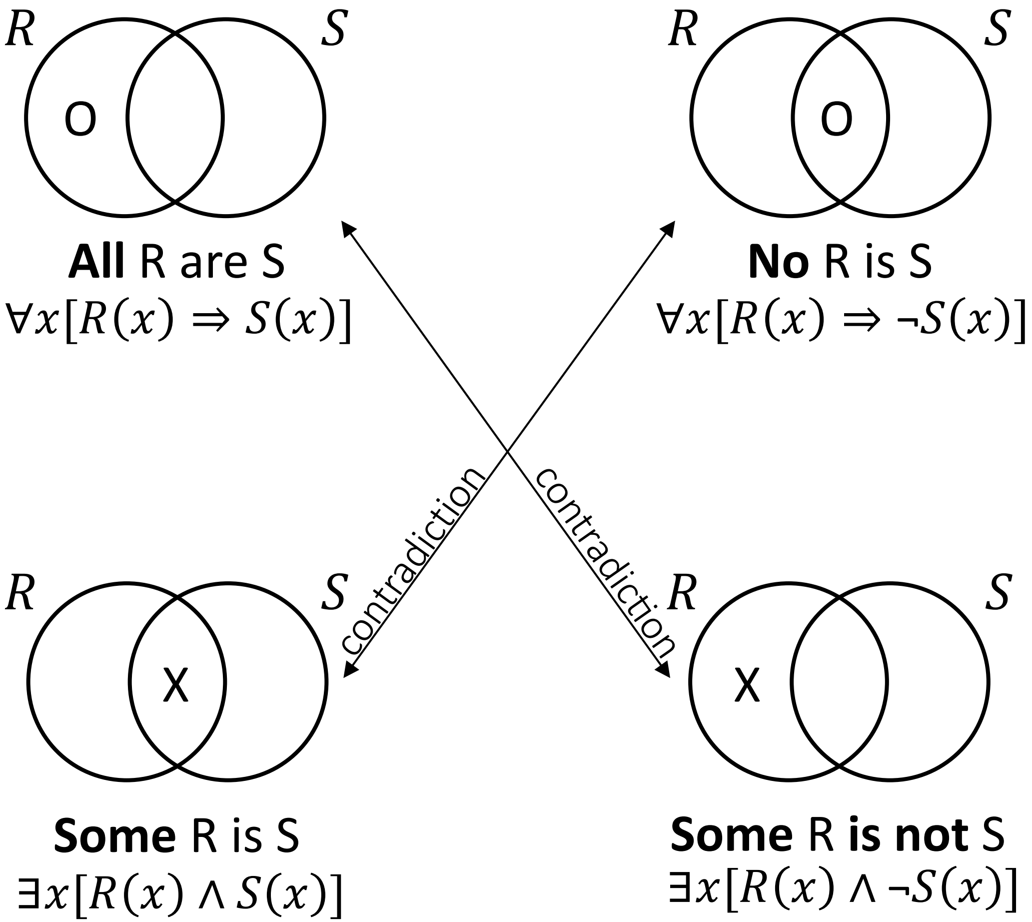
```
select distinct A
from R
where not exists
    (select *
    from S
    where not exists
        (select *
        from T
        where T.A = R.A
        and T.B = S.B
        and not exists
            (select *
            from U
            where U.C = S.C
            and not exists
          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



Example used from: "Gatterbauer, Dunne, Riedewald. Relational Diagrams. arXiv:2203.07284. 2022. https://arxiv.org/pdf/2203.07284 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Relational Diagrams vs. Venn-Peirce

Venn-Peirce vs. Relational Diagrams

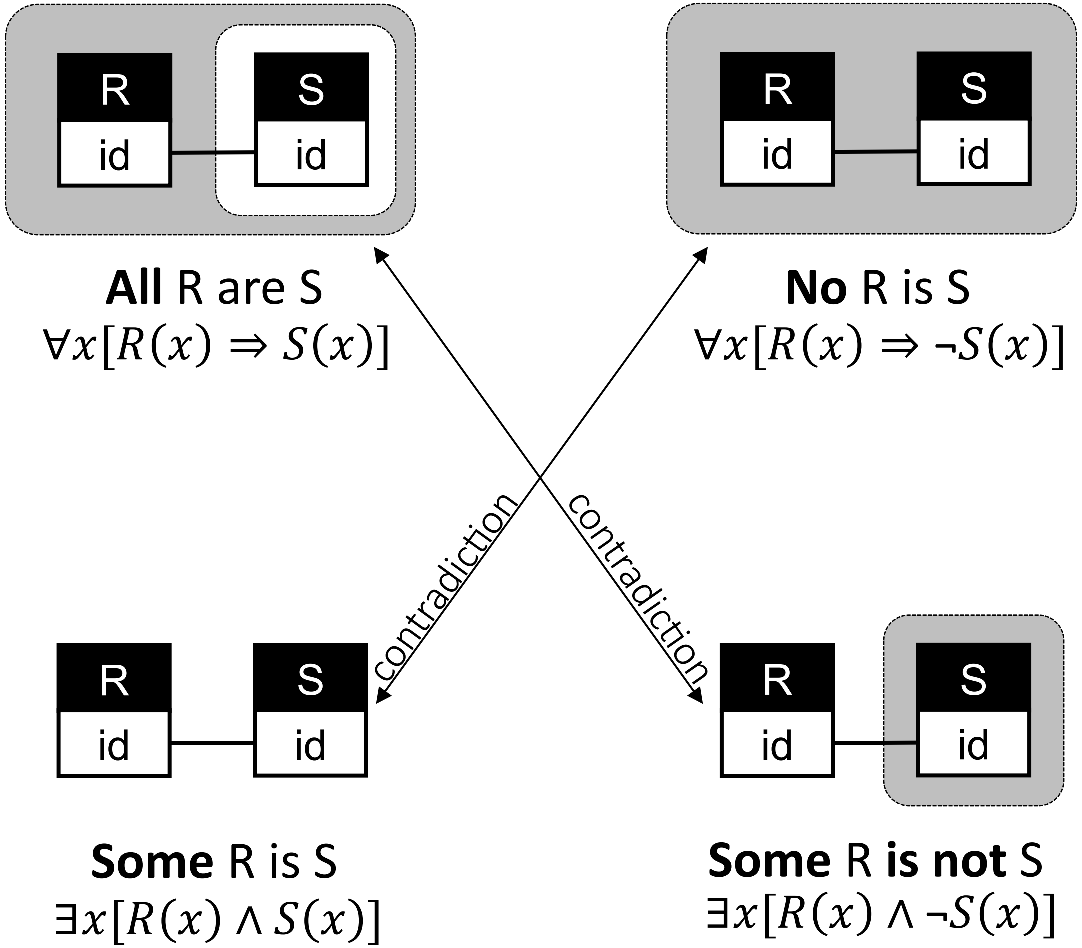


All R are S
$\forall x[R(x) \Rightarrow S(x)]$

No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

Some R is S
$\exists x[R(x) \land S(x)]$

Some R **is not** S
$\exists x[R(x) \land \neg S(x)]$



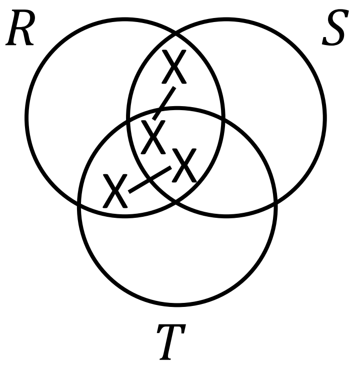
All R are S
$\forall x[R(x) \Rightarrow S(x)]$

No R is S
$\forall x[R(x) \Rightarrow \neg S(x)]$

Some R is S
$\exists x[R(x) \land S(x)]$

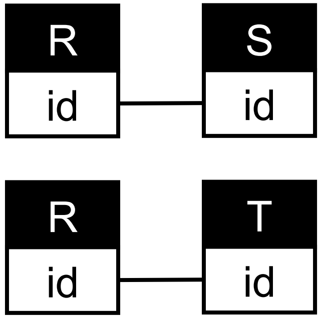
Some R **is not** S
$\exists x[R(x) \land \neg S(x)]$

Venn-Peirce vs. Relational Diagrams



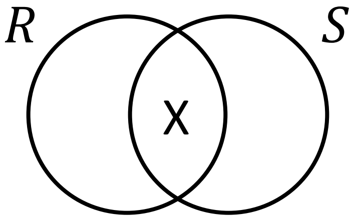
Some R is S, and **some** R is T

$\exists x[R(x) \land S(x)] \land \exists z[R(z) \land T(z)]$



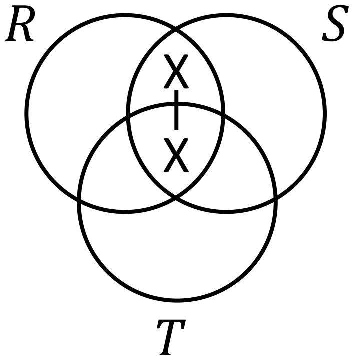
Some R is S, and **some** R is T

$\exists x[R(x) \land S(x)] \land \exists z[R(z) \land T(z)]$



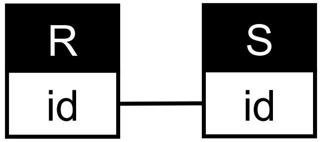
Some R is S

$\exists x[R(x) \land S(x)]$



Some R is S

$\exists x[R(x) \land S(x)]$



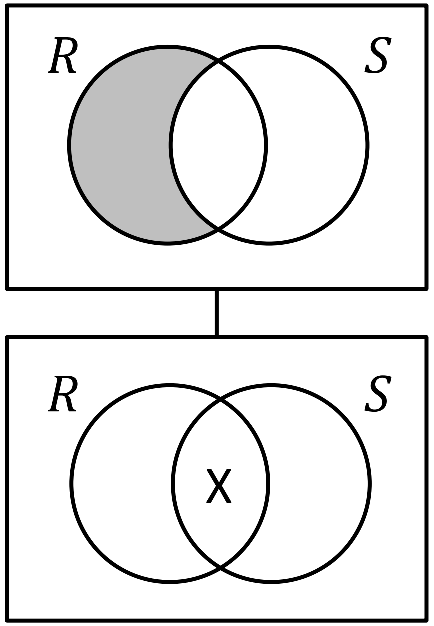
Some R is S

$\exists x[R(x) \land S(x)]$

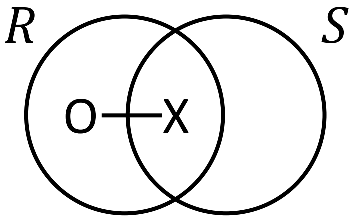
Relational Diagrams don't show predicates that are not used

Venn-Peirce vs. Relational Diagrams

Venn-Peirce-Shin ("Venn-2")



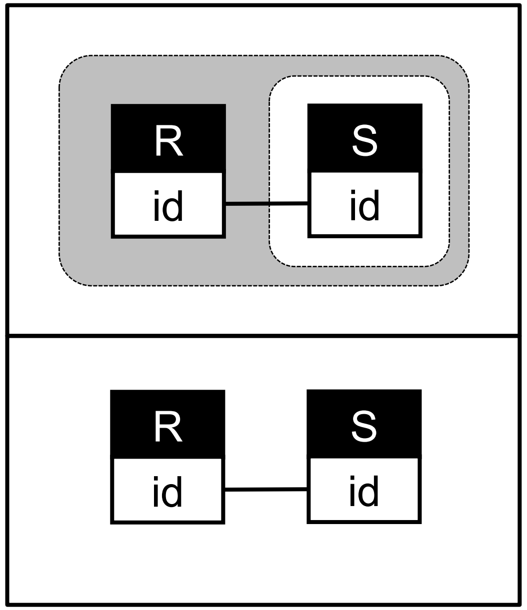
Venn-Peirce



All R are S, or **some** R is S

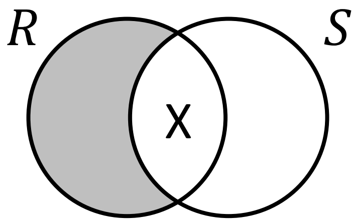
$\forall x[R(x) \Rightarrow S(x)] \lor \exists x[R(x) \land S(x)]$

Relational Diagrams

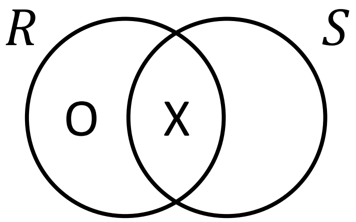


Venn-Peirce vs. Relational Diagrams

Venn-Peirce-Shin ("Venn-2")



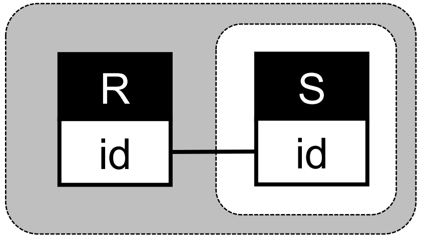
Venn-Peirce

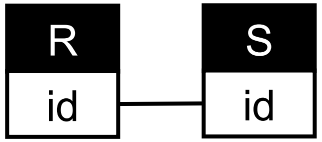


All R are S, and **some** R is S

$\forall x[R(x) \Rightarrow S(x)] \wedge \exists x[R(x) \wedge S(x)]$

Relational Diagrams

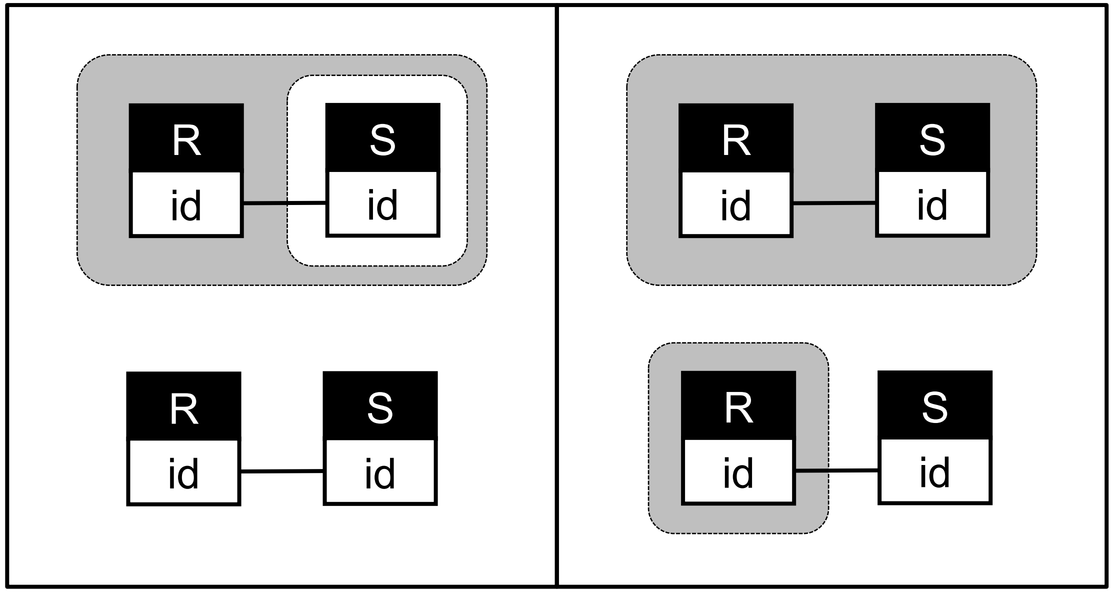


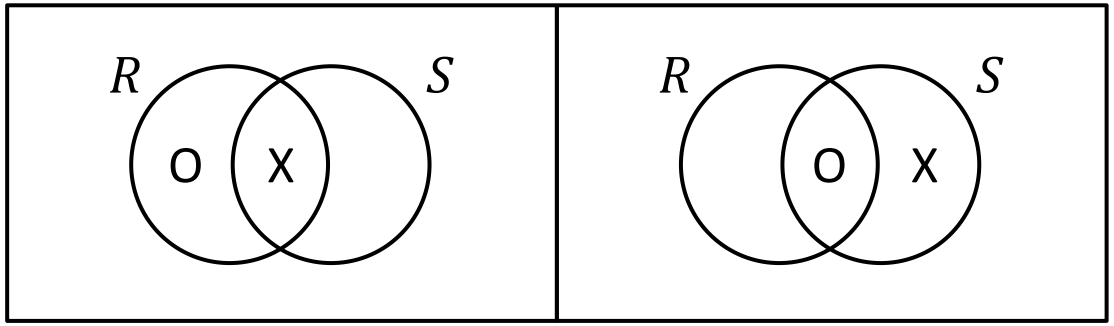


Relational Diagrams are "isomorphic" to non-disjunctive Relational Tuple Calculus. For a formal definition of "pattern isomorphisms" and "non-disjunctive Calculus" see [Gatterbauer,Dune'24]

Gatterbauer, Dunne. On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages. SIGMOD 2024. https://dl.acm.org/doi/pdf/10.1145/3639316

Venn-Peirce vs. Relational Diagrams





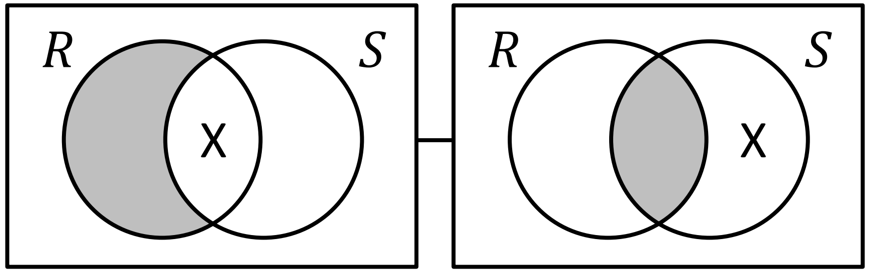
DNF

$$(\forall x[R(x) \Rightarrow \ S(x)] \wedge \exists x[R(x) \wedge \ S(x)]) \ \vee$$
$$(\forall x[R(x) \Rightarrow \neg S(x)] \wedge \exists x[S(x) \wedge \neg R(x)])$$

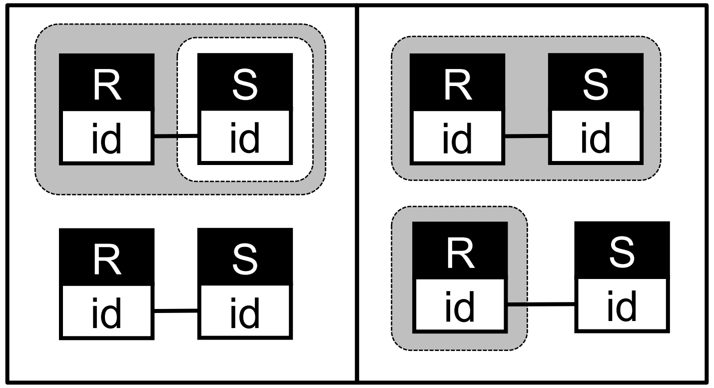
Venn-Peirce-Shin (Venn-II) vs. Relational Diagrams

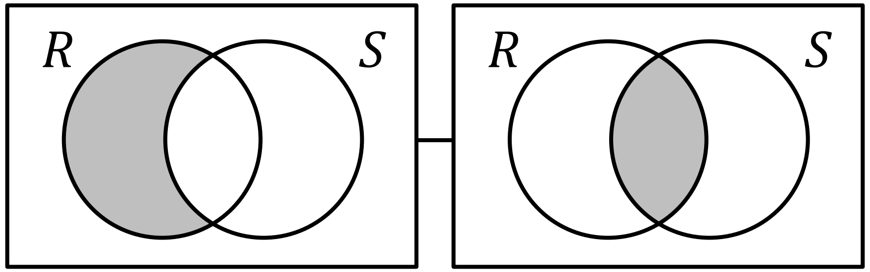
Venn-Peirce-Shin ("Venn-II")

Relational Diagrams

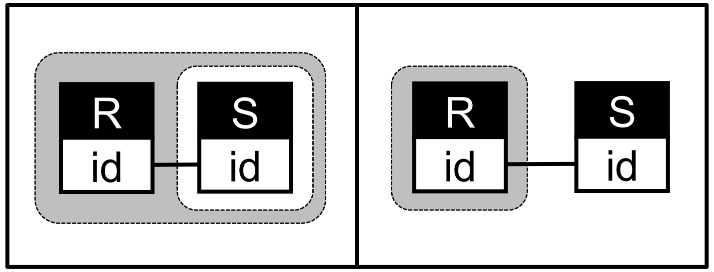


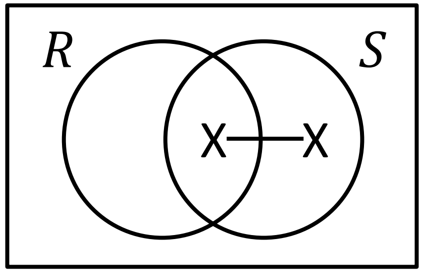
(**All** R are S, and **some** R is S) or (**No** R is S, and **some** S **is not** R)



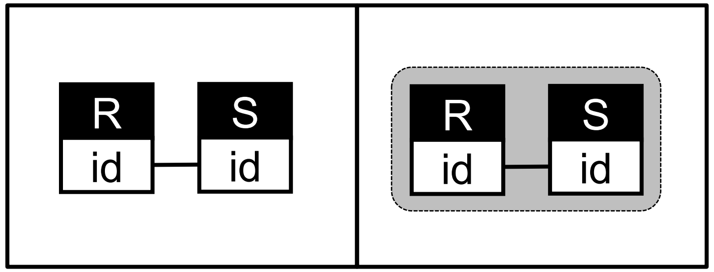


(**All** R are S) or (**no** R is S)

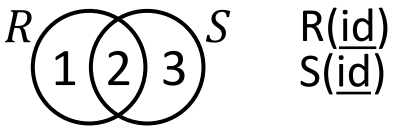


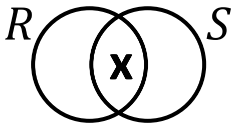
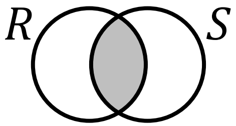
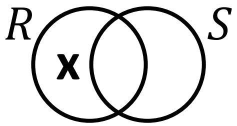
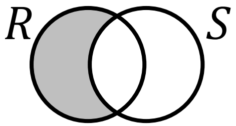


(**Some** R is S) or (**some** S **is not** R)



4 categorical propositions w/ Relational Diagrams



	true	false	true	false
Venn-Peirce				
Relational Calculus	$\exists x[R(x) \land S(x)]$	$\neg\exists x[R(x) \land S(x)]$	$\exists x[R(x) \land \neg S(x)]$	$\neg\exists x[R(x) \land \neg S(x)]$
NL	Some R is S.	No R is S. (All R are not S)	Some R is not S.	All R are S.

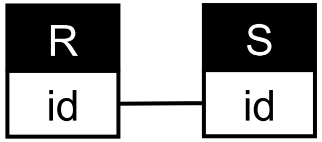
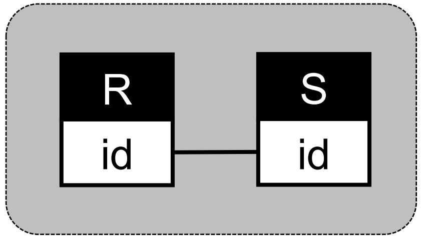
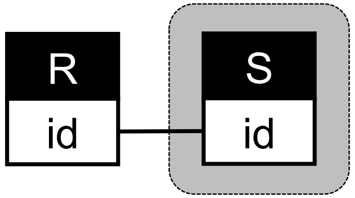
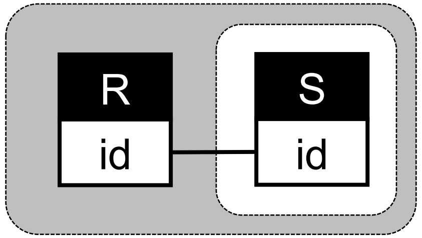
SQL

```
SELECT EXISTS(
  SELECT *
  FROM R, S
  WHERE R.id = S.id)
```

```
SELECT NOT EXISTS(
  SELECT *
  FROM R, S
  WHERE R.id = S.id)
```

```
SELECT EXISTS(
  SELECT *
  FROM R
  WHERE NOT EXISTS(
    SELECT *
    FROM S
    WHERE R.id = S.id))
```

```
SELECT NOT EXISTS(
  SELECT *
  FROM R
  WHERE NOT EXISTS(
    SELECT *
    FROM S
    WHERE R.id = S.id))
```

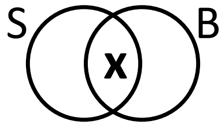
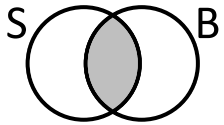
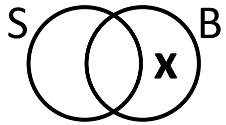
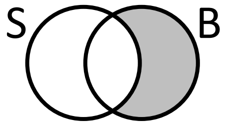
Relational Diagram				

Database to run SQL queries is available as schema 370 at https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql/

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

4 categorical propositions with Sailors

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

Cat.	 Some S is B. $\exists x[S(x) \land B(x)]$	 No S is B. (All S are not B) $\neg\exists x[S(x) \land B(x)]$	 Some B is not S. $\exists x[B(x) \land \neg S(x)]$	 All B are S. $\neg\exists x[B(x) \land \neg S(x)]$
NL	Sailors who reserved **some** red boat	Sailors who did **not** reserve **any** red boat	Sailors who did **not** reserve **all** red boats	Sailors who reserved **all** red boats

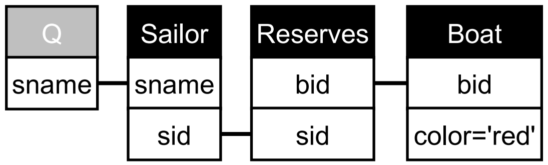
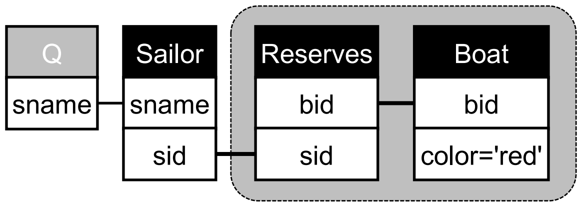
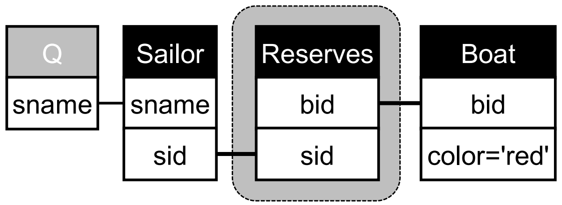
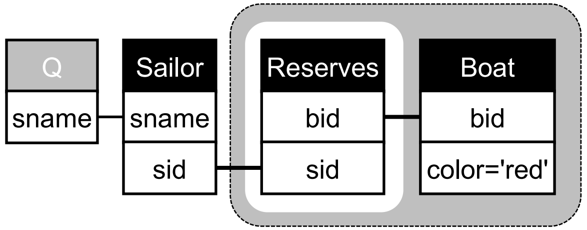
SQL

```
SELECT S.sname
FROM Sailor S
WHERE EXISTS(
  SELECT *
  FROM Reserves R
  WHERE R.sid = S.sid
  AND 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 Reserves R
  WHERE R.sid = S.sid
  AND EXISTS(
    SELECT *
    FROM Boat B
    WHERE B.color = 'red'
    AND B.bid = R.bid))
```

```
SELECT S.sname
FROM Sailor S
WHERE 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))
```

```
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))
```

RD	Q sname Sailor sname sid Reserves bid sid Boat bid color='red'	Q sname Sailor sname sid Reserves bid sid Boat bid color='red'	Q sname Sailor sname sid Reserves bid sid Boat bid color='red'	Q sname Sailor sname sid Reserves bid sid Boat bid color='red'

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Relational Diagrams vs. Beta EGs

Beta EGs vs. Relational Diagrams

Schema

	Boat
1	bid
2	bname
3	color
4	pdate

S: "There is a red boat."

∃x,y,u [Boat(x,y,'red',u)]

∃b∈Boat [b.color='red']

?

?

Beta EGs vs. Relational Diagrams

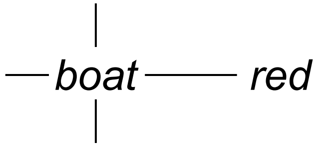
Schema

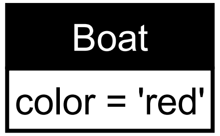
	Boat
1	<u>bid</u>
2	bname
3	color
4	pdate

S: "There is a red boat."

∃x,y,u [Boat(x,y,'red',u)]

∃b∈Boat [b.color='red']





S: "There is a boat."

∃x,y,z,u [Boat(x,y,z,u)]

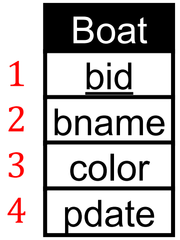
∃b∈Boat

?

?

Beta EGs vs. Relational Diagrams

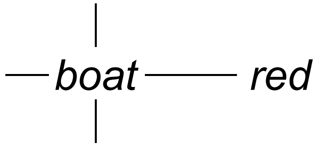
Schema

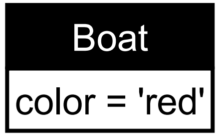


S: "There is a red boat."

$\exists x,y,u\ [\mathrm{Boat}(x,y,\text{'red'},u)]$

$\exists b \in \mathrm{Boat}\ [b.\mathrm{color}=\text{'red'}]$

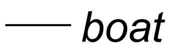




S: "There is a boat."

$\exists x,y,z,u\ [\mathrm{Boat}(x,y,z,u)]$

$\exists b \in \mathrm{Boat}$



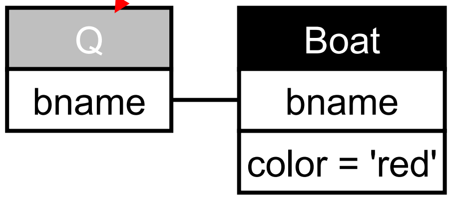
In Relational Diagrams, lines (of identity) are *not* necessary for existential quantification

Beta EGs vs. Relational Diagrams

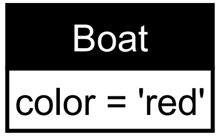
Schema

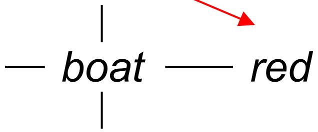
Boat
bid
bname
color
pdate

1. Beta EGs cannot represent queries (lines represent quantified variables)



2. Beta EGs cannot represent constants (need dedicated predicates)



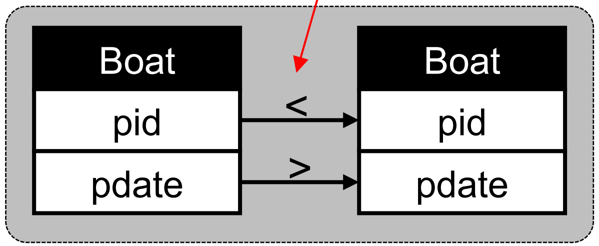


$\exists B \in Boat\ [B.color='red']$

$\exists x,y,z\ [Boat(x,y,'red',u)]$
$\exists x,y,z,u\ [Boat(x,y,z,u) \wedge Red(z)]$

3. Beta EGs follow the convention of standard FOL, which corresponds to DRC (unnamed perspective). Thus all attributes need to be quantified and shown as lines.

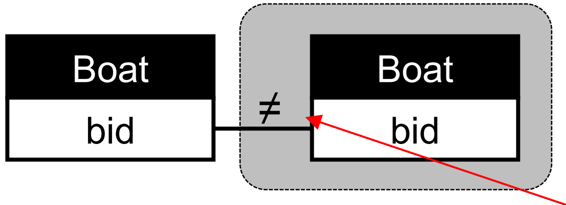
4. Beta EGs cannot represent comparison predicates (only equality = identity)

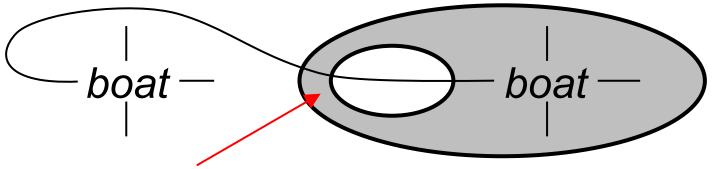


S: "The boat ids are in ascending order the purchase dates."

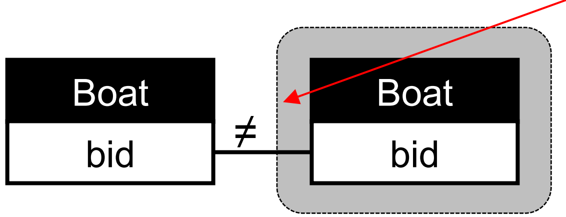
5. Lines of Identify (LIs) in Beta EGs have multiple meanings: existential quantification and identity = equality. This makes interpretation error-prone

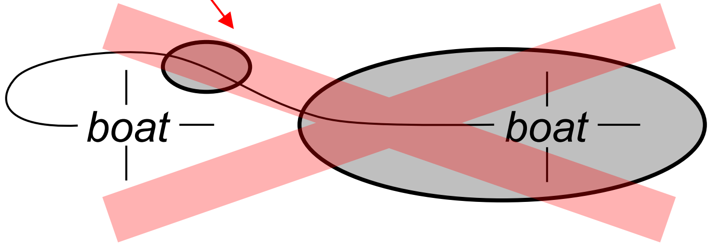
S: "There is only one boat."





Position of inequality does not matter for Relational Diagrams. But the two shown Beta EGs have *different* meanings





Beta EGs vs. Relational Diagrams

Schema

Boat
1 bid
2 bname
3 color
4 pdate

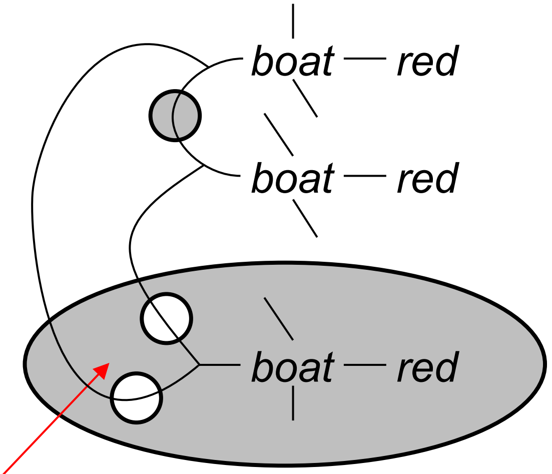
S: "There are exactly two red boats."

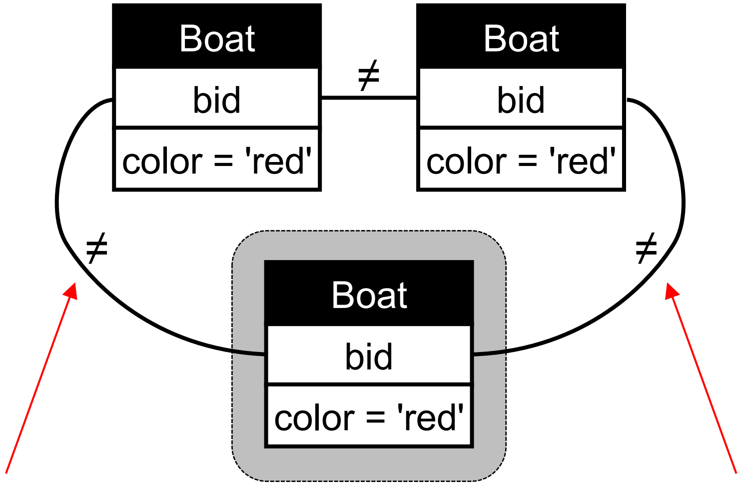
$\exists x,y,z,s,t,p\ [\text{Boat}(x,y,\text{'red'},z) \land \text{Boat}(s,t,\text{'red'},p) \land\ x\neq s \land$
$\neg(\exists u,v,w, [\text{Boat}(u,v,\text{'red'},w) \land u\neq x \land u\neq s])]$

$\exists b_1\in\text{Boat}, \exists b_2\in\text{Boat}\ [b_1.\text{color}=\text{'red'} \land b_2.\text{color}=\text{'red'} \land b_1.\text{bid}\neq b_2.\text{bid}$
$\land \neg(\exists b_3\in\text{Boat}\ [b_3.\text{color}=\text{'red'} \land b_1.\text{bid} \neq b_3.\text{bid} \land b_2.\text{bid}\neq b_3.\text{bid}])]$

$\exists x,y,z,s,t,p\ [\text{Boat}(x,y,\text{'red'},z) \land \text{Boat}(s,t,\text{'red'},p) \land\ x\neq s \land$
$\forall u,v,w, [\text{Boat}(u,v,\text{'red'},w) \rightarrow (u=x \lor u=s])]$

$\exists b_1\in\text{Boat}, \exists b_2\in\text{Boat}\ [b_1.\text{color}=\text{'red'} \land b_2.\text{color}=\text{'red'} \land b_1.\text{bid}\neq b_2.\text{bid}$
$\land \forall b_3\in\text{Boat}\ [b_3.\text{color}=\text{'red'} \rightarrow (b_1.\text{bid} = b_3.\text{bid} \lor b_2.\text{bid}=b_3.\text{bid})]]$





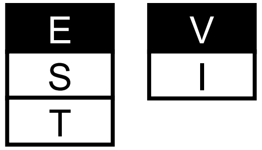
The placement of the two cuts *inside* the larger one is important; placing them outside instead would give an incorrect semantics.

The inequality signs are labels of their respective lines; thus their placement inside or outside negation boxes does *not* affect the interpretation of the diagram

Beta EG example adapted from "Sowa. Peirce's tutorial on existential graphs, Semiotica, 2011. https://doi.org/10.1515/semi.2011.060 "

Beta EGs vs. Relational Diagrams

Directed (E)dges with (S)ource and (T)arget



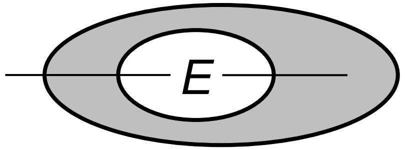
Schema

(V)ertices contain the domain = union of values in E.S and E.T

Beta EGs and Domain Relational Calculus

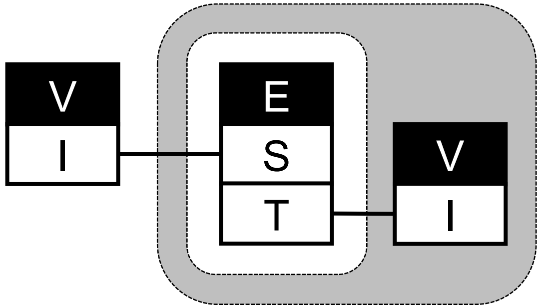
Relational Diagrams and Tuple Relational Calculus

S: "There is a node that points to all nodes."



$\exists x \quad \forall y\, [E(x,y)]$

$\exists x[\neg(\exists y\, [E(x,y)])]$



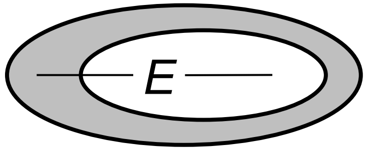
Notice that both TRC and RD make the domains always explicit

$\exists v_1 \in V\, [\quad \forall v_2 \in V\, [\quad \exists e \in E\, [e.S=v_1.I \land e.T=v_2.I]\,]$

$\exists v_1 \in V\, [\neg(\exists v_2 \in V\, [\neg(\exists e \in E\, [e.S=v_1.I \land e.T=v_2.I])]$

$\exists e_1 \in E\, [\neg(\exists e_2 \in V\, [\neg(\exists e \in E\, [e.S=e_1.S \land e.T=e_2.S])]$

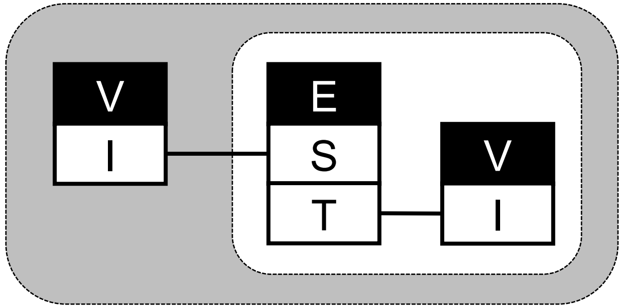
S: "All nodes are connected to some node."



$\forall x \quad \exists y\, [E(x,y)]$

$\neg(\exists x[\neg(\exists y\, [E(x,y)])])$

Variant without the vertex relation if all nodes have an outgoing edge



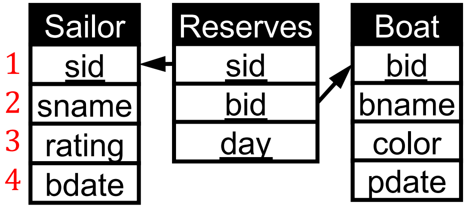
$\neg(\exists v_1 \in V\, [\quad \forall v_2 \in V,\, \forall e \in E\, [e.S=v_1.I \land e.T=v_2.I])]$

$\neg(\exists v_1 \in V\, [\neg(\exists v_2 \in V,\, \exists e \in E\, [e.S=v_1.I \land e.T=v_2.I])]$

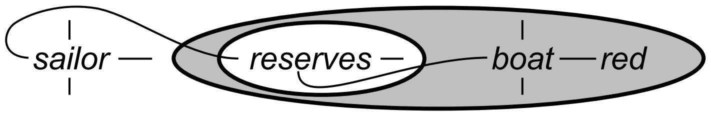
$\neg(\exists e_1 \in V\, [\neg(\exists e_2 \in V,\, \exists e \in E\, [e.S=e_1.S \land e.T=e_2.S])]$

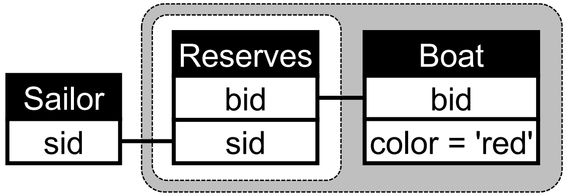
Beta EGs vs. Relational Diagrams

Schema



~Q4: "There is a sailor who reserved all red boats."

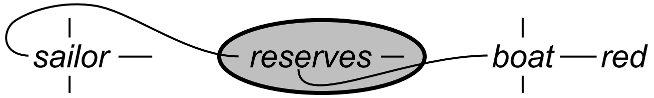


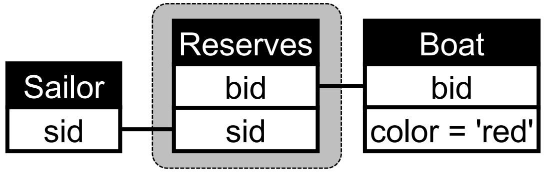


∃x,v,z,w [Sailor(v,x,z,w) ∧ ¬(∃y,u,s [Boat(y,u,'red',s) ∧ ¬(∃t [Reserves(v,y,t)])])]
∃x,v,z,w [Sailor(v,x,z,w) ∧ ∀y,u,s [Boat(y,u,'red',s) → (∃t [Reserves(v,y,t)])])]

∃s∈Sailor [¬(∃b∈Boat [b.color='red' ∧ ¬(∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid])])]
∃s∈Sailor [∀b∈Boat [b.color='red' → ∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid]]]

"There is a sailor who did not reserve all red boats."

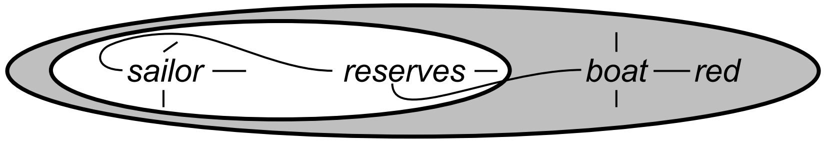


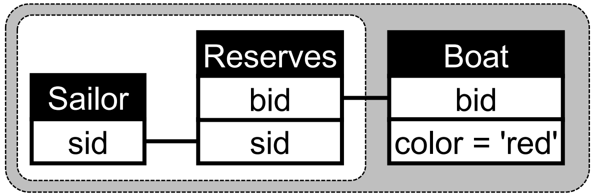


∃x,v,z,w,y,u,s [Sailor(v,x,z,w) ∧ Boat(y,u,'red',s) ∧ ¬(∃t [Reserves(v,y,t)])]

∃s∈Sailor, ∃b∈Boat [b.color='red' ∧ ¬(∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid])]

"All red boats were reserved by some sailors."

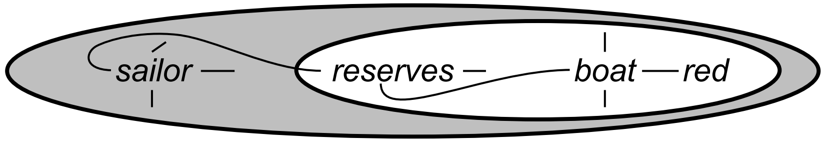


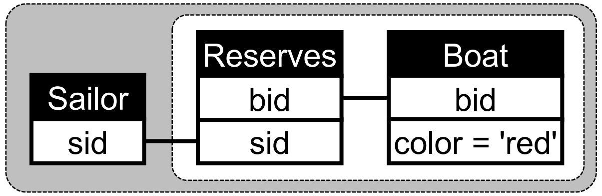


¬(∃y,u,s [Boat(y,u,'red',s) ∧ ¬(∃v,t,x,z,w [Reserves(v,y,t) ∧ Sailor(v,x,z,w)])])
∀y,u,s [Boat(y,u,'red',s) → (∃v,t,x,z,w [Reserves(v,y,t) ∧ Sailor(v,x,z,w)])])

¬(∃b∈Boat [b.color='red' ∧ ¬(∃s∈Sailor, ∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid])])
∀b∈Boat [b.color='red' → ∃s∈Sailor, ∃r∈Reserves [b.bid=r.bid ∧ r.sid=s.sid]])

"All sailors have reserved some red boat."



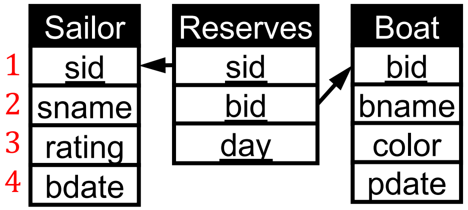


¬(∃x,v,z,w [Sailor(v,x,z,w) ∧ ¬(∃y,u,s,t [Reserves(v,y,t) ∧ Boat(y,u,'red',s)])])
∀x,v,z,w [Sailor(v,x,z,w) → (∃y,u,s,t [Reserves(v,y,t) ∧ Boat(y,u,'red',s)])])

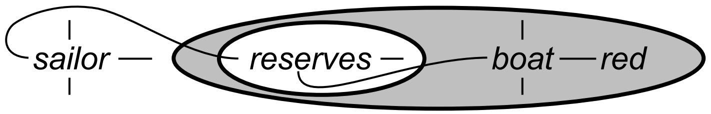
¬(∃s∈Sailor [¬(∃b∈Boat, ∃r∈Reserves [b.color='red' ∧ b.bid=r.bid ∧ r.sid=s.sid])])
∀s∈Sailor [∃b∈Boat, ∃r∈Reserves [b.color='red' ∧ b.bid=r.bid ∧ r.sid=s.sid]]

Beta EGs vs. Relational Diagrams

Schema

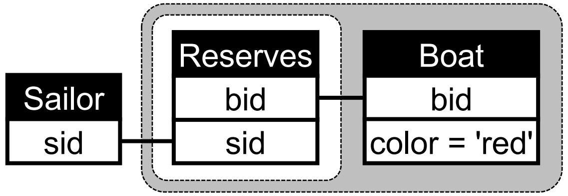


~Q4: "There is a sailor who reserved all red boats."



$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \neg(\exists y,u,s\ [\text{Boat}(y,u,\text{'red'},s) \wedge \neg(\exists t\ [\text{Reserves}(v,y,t)])])]$

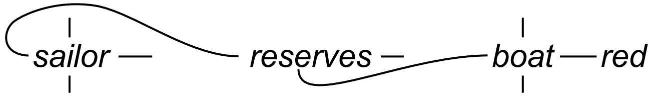
$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \forall y,u,s\ [\text{Boat}(y,u,\text{'red'},s) \rightarrow (\exists t\ [\text{Reserves}(v,y,t)])])]$



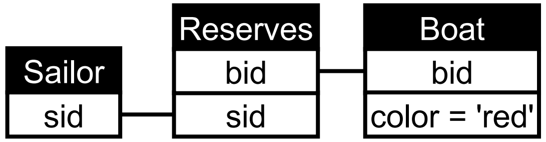
$\exists s \in \text{Sailor}\ [\neg(\exists b \in \text{Boat}\ [b.\text{color}=\text{'red'} \wedge \neg(\exists r \in \text{Reserves}\ [b.\text{bid}=r.\text{bid} \wedge r.\text{sid}=s.\text{sid}])])]$

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

~Q2: "There is a sailor who reserved a red boat."

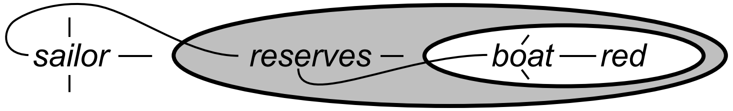


$\exists x,v,z,w,y,t,u,s\ [\text{Sailor}(v,x,z,w) \wedge \exists y,t\ [\text{Reserves}(v,y,t) \wedge \exists u,s\ [\text{Boat}(y,u,\text{'red'},s]]]$



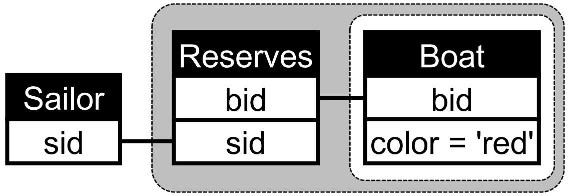
$\exists s \in \text{Sailor},\ \exists r \in \text{Reserves},\ \exists b \in \text{Boat}\ [s.\text{sid}=r.\text{sid} \wedge r.\text{bid}=b.\text{bid} \wedge b.\text{color}=\text{'red'}])]$

~Q3: "There is a sailor who reserved only red boats."



$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \neg(\exists y,t\ [\text{Reserves}(v,y,t) \wedge \neg(\exists u,s\ [\text{Boat}(y,u,\text{'red'},s)])])]$

$\exists x,v,z,w\ [\text{Sailor}(v,x,z,w) \wedge \forall y,t\ [\text{Reserves}(v,y,t) \rightarrow (\exists u,s\ [\text{Boat}(y,u,\text{'red'},s)])])]$



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

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

Beta EGs vs. Relational Diagrams

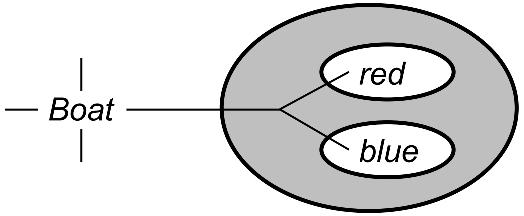
Schemas

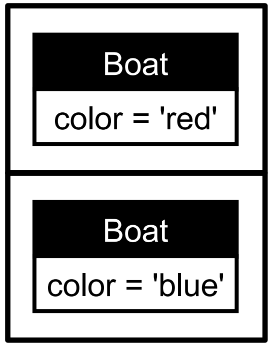
Boat
bid
bname
color
pdate

RedBoat
bid
bname
pdate

BlueBoat
bid
bname
pdate

~Q1: "There is a boat that is red or blue."

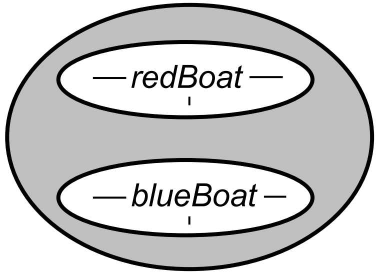


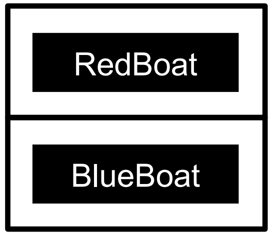


$\exists y,z,u\ [\text{Boat}(z,x,y,u) \land (y=\text{'red'} \lor y=\text{'blue'})$
$\exists y,z,u\ [\text{Boat}(z,x,y,u) \land \neg(\neg(y=\text{'red'}) \land \neg(y=\text{'blue'}))]$

$\exists b\in\text{Boat}\ [q.\text{bname}=b.\text{bname} \land (b.\text{color}=\text{'red'} \lor b.\text{color}=\text{'blue'})]$
$\exists b\in\text{Boat}\ [q.\text{bname}=b.\text{bname} \land b.\text{color}=\text{'red'}] \lor \exists b\in\text{Boat}\ [q.\text{bname}=b.\text{bname} \land b.\text{color}=\text{'blue'})]$

~Q5: "There is a red boat or a blue boat."



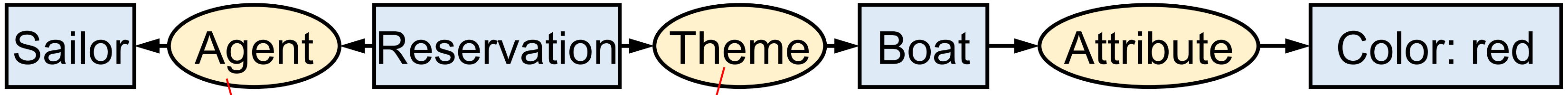


$\exists x\exists y\exists z\ [\text{RedBoat}(x,y,z) \lor \text{BlueBoat}(x,y,z)]$
$\neg(\neg(\exists x\exists y\exists z\ [\text{RedBoat}(x,y,z)]) \land \neg(\exists z\ [\text{BlueBoat}(x,y,z)]))$

$\exists b\in\text{RedBoat}\ [q.\text{bid}=b.\text{bid} \land q.\text{bname}=b.\text{bname}] \lor$
$\exists b\in\text{BlueBoat}\ [q.\text{bid}=b.\text{bid} \land q.\text{bname}=b.\text{bname}]$

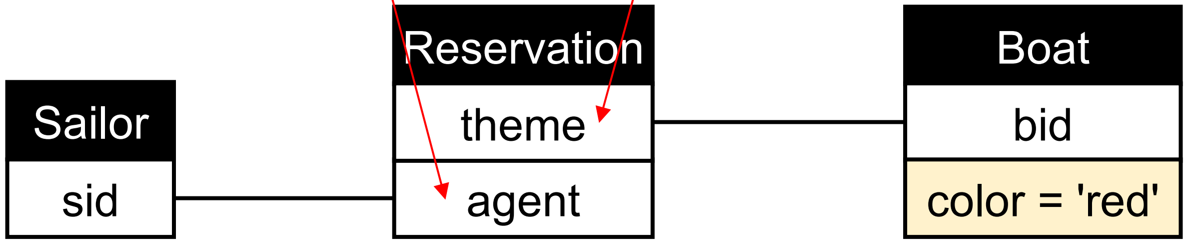
Relational Diagrams vs. Conceptual Graphs

Conceptual Graphs vs. Relational Diagrams



$$\exists x\text{: Sailor}, \exists y\text{: Boat}, \exists z\text{: Reservation}[\text{Agent}(z, x) \wedge \text{Theme}(z, y) \wedge \text{Color(red)} \wedge \text{Attribute}(x, \text{red})]$$

agent and theme really "belong to" Reservation



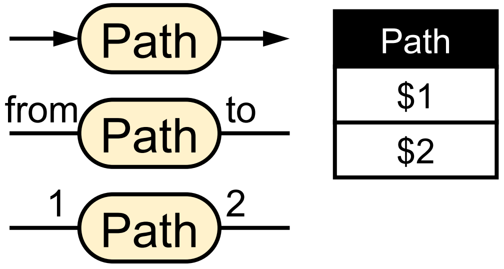
"Relational Diagrams" uses a UML-inspired representation that is consistent with familiar relational schema notation

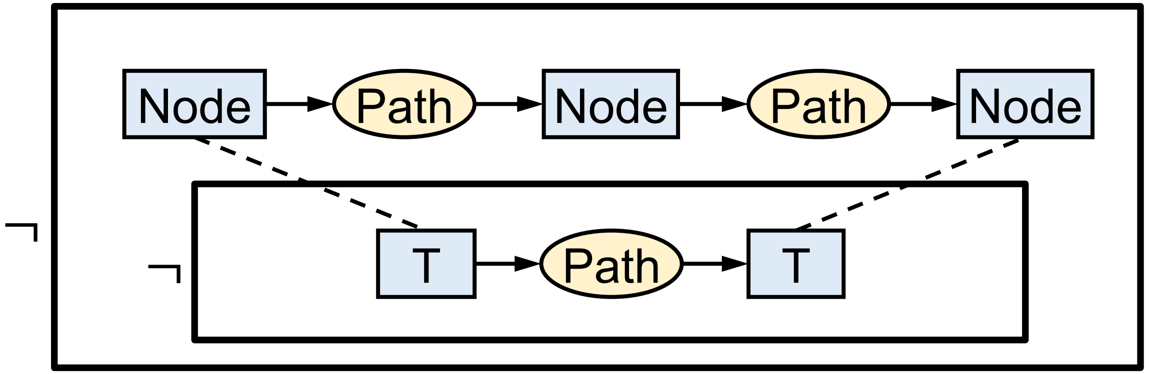
$$\exists x \in \text{Sailor}, \exists y \in \text{Boat}, \exists z \in \text{Reserves}[x.\text{sid} = y.\text{agent} \wedge y.\text{theme} = z.\text{bid} \wedge z.\text{color} = \text{'red'}]$$

After: Figure 1.9, Section 1.4 from Sowa. “Knowledge Representation: Logical, Philosophical, and Computational Foundations,” 2000. https://dl.acm.org/doi/book/10.5555/318183 and Sect 5.1 from Sowa. “Conceptual graphs,” in Handbook of Knowledge Representation, 2008. https://doi.org/10.1016/S1574-6526(07)03005-2

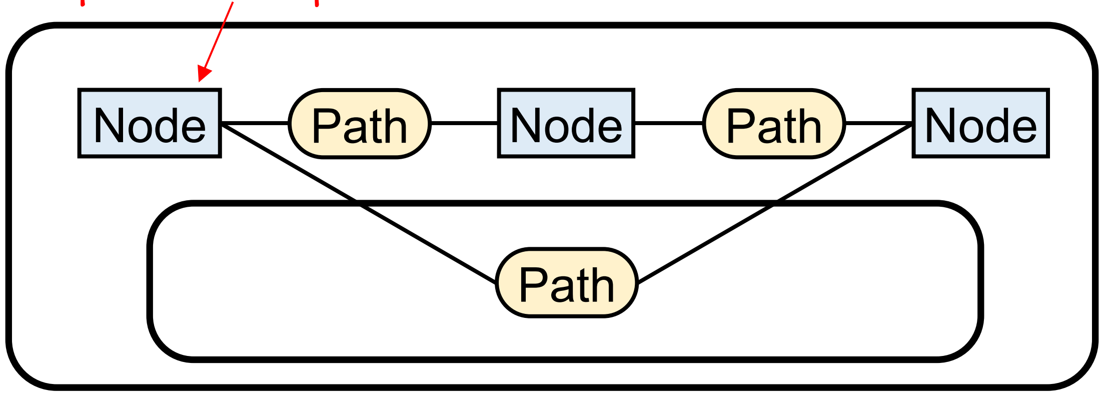
Conceptual Graphs vs. Relational Diagrams

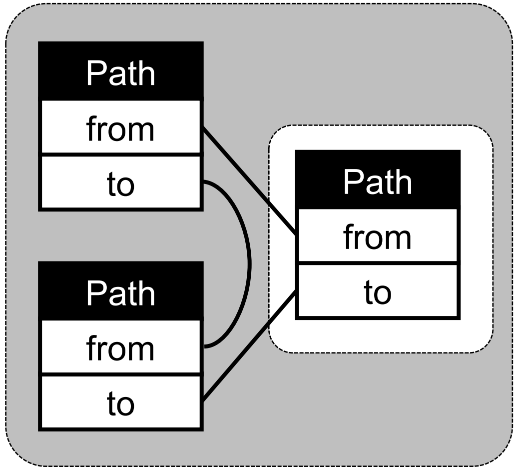
"Paths are transitive."





Even with Dau's simplification, conceptual graphs still require concepts *in addition* to relations





Relational Diagrams follow tuple relational calculus and thus don't need concept nodes

Translation to (domain relational) calculus:

$$\neg(\exists x{:}\,\text{Node}, \exists y{:}\,\text{Node}, \exists z{:}\,\text{Node}[\text{Path}(x,y) \wedge \text{Path}(y,z) \wedge \neg(\text{Path}(x,z))])$$

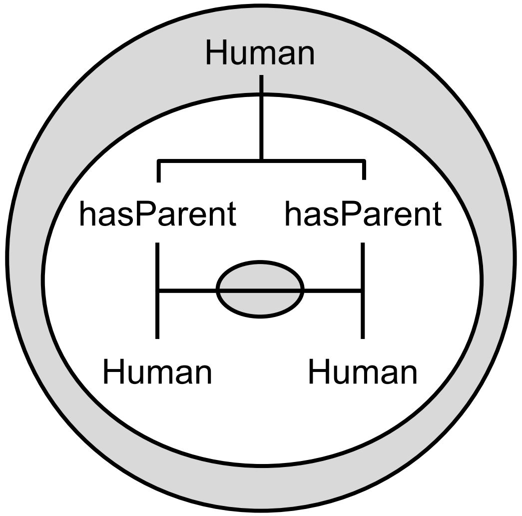
Translation to (tuple relational) calculus:

$$\neg(\exists r_1 \in R, \exists r_2 \in R[r_1.\text{to} = r_2.\text{from} \wedge \neg(\exists r_3 \in R[r_3.\text{from} = r_1.\text{from} \wedge r_3.\text{to} = r_2.\text{to}])])$$

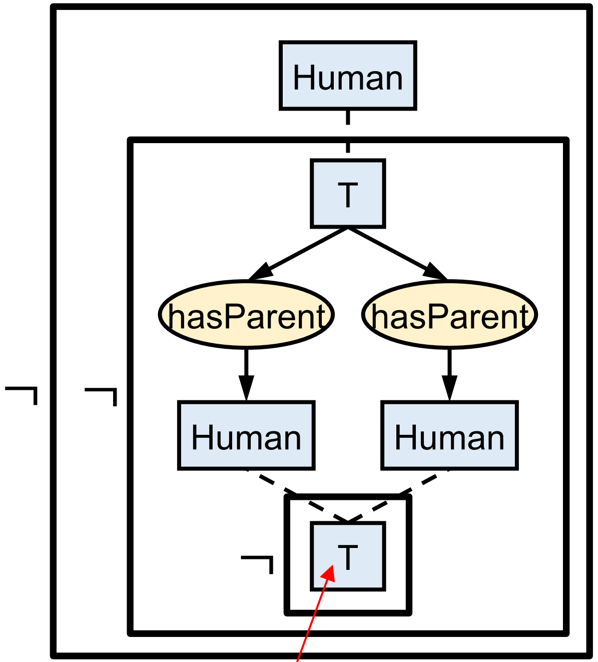
Based on "Chein, Mugnier. “Graph-based knowledge representation,” 2009. https://doi.org/10.1007/978-1-84800-286-9"

Beta Existential Graph

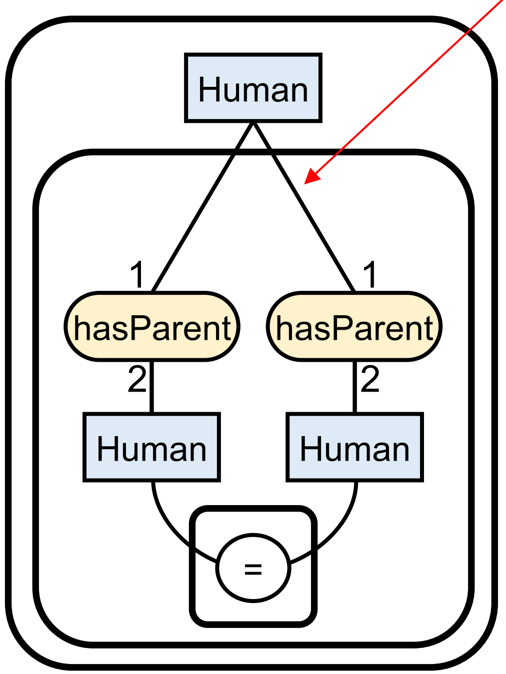
"Every Human has at least two parents who are Human."



Conceptual Graph

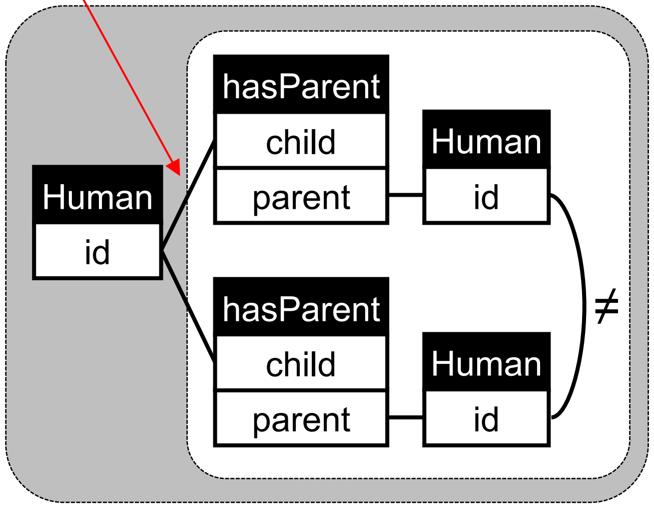


Conceptual Graphs (Dau variant)



Relational Diagram

Lines are used to connect arguments of predicates, not for existential quantification. The position of the "fork" does not matter



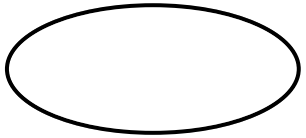
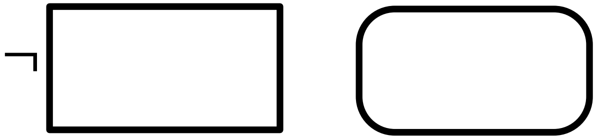
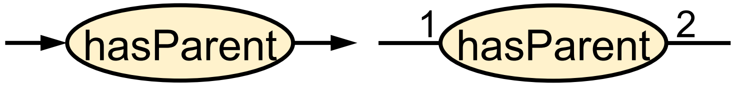
To express that y ≠ z, we have to say "No entity exists that is equal to both y and z."

Translation:

DRC: $\neg(\exists x[\text{Human}(x) \wedge \neg(\exists y, \exists z[\text{hasParent}(x,y) \wedge \text{Human}(y) \wedge \text{hasParent}(x,z) \wedge \text{Human}(z) \wedge \neg(y = z)])])$

TRC: $\neg(\exists h_1 \in \text{Human}[\neg(\exists p_1 \in \text{hasParent}, \exists p_2 \in \text{hasParent}, \exists h_2 \in \text{Human}, \exists h_3 \in \text{Human}[$
$p_1.\text{child} = h_1.\text{id} \wedge\ p_2.\text{child} = h_1.\text{id} \wedge\ p_1.\text{parent} = h_1.\text{id} \wedge p_2.\text{parent} = h_2.\text{id} \wedge h_2.\text{id} \neq h_3.\text{id}])])$

Based on "Chein, Mugnier. "Graph-based knowledge representation," 2009. https://doi.org/10.1007/978-1-84800-286-9"

	Beta Existential Graph	Conceptual Graph	Relational Diagram
Logic	domain relational calculus (DRC)	DRC	tuple relational calculus (TRC)
Visual element for existential quantification	Line (of Identity) —— Human	Concept box Human	Relation box Human
Visual negation	Oval / Cut 	Annotated Box / Cut 	Dashed rounded box 
Order or names of relational attributes	not specified — hasParent —	arc (arrowhead) / numbers 	attribute names (named perspective) hasParent child parent
Queries	No	Human: ?	Q id

Key difference of relational diagrams

Intended Agenda today

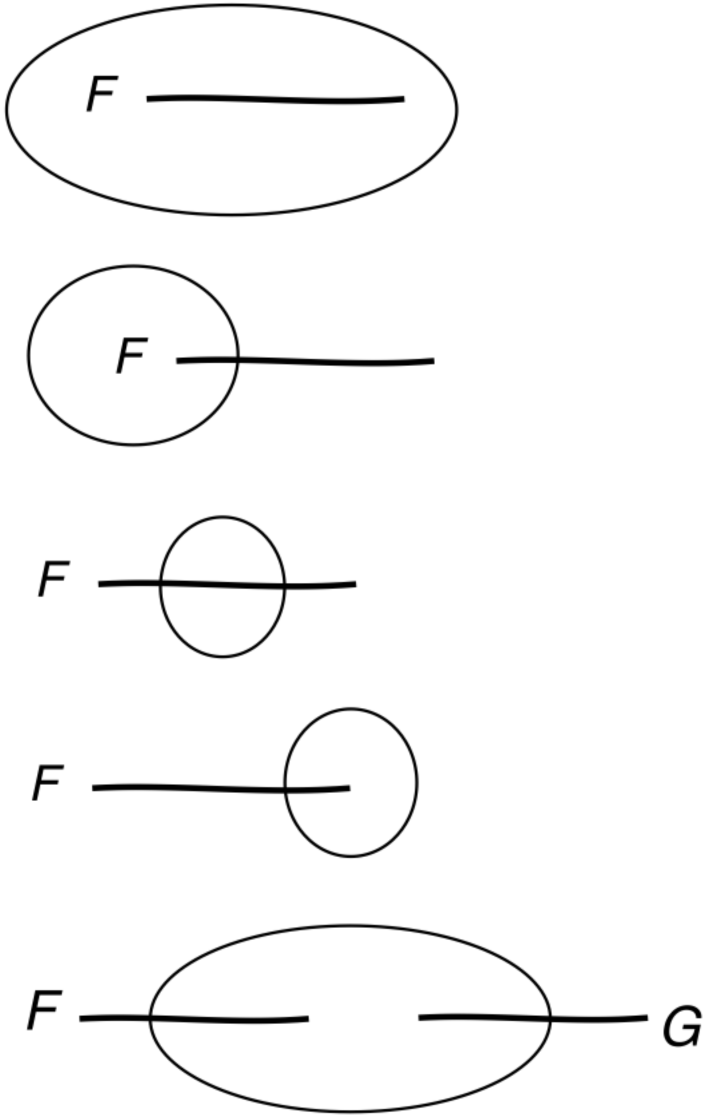
Please leave feedback ☺

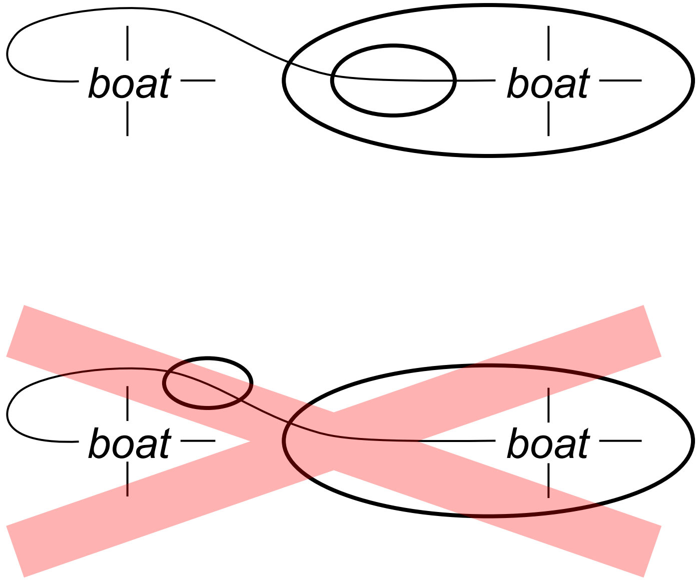
1. Why visualizing queries and why now?
2. Principles of Query Visualization
3. Logical foundations of relational query languages
4. (Early) Diagrammatic representations
5. Visual Query Representations (from DB community)
6. Lessons Learned and Open Challenges

3 Abuses of the Line

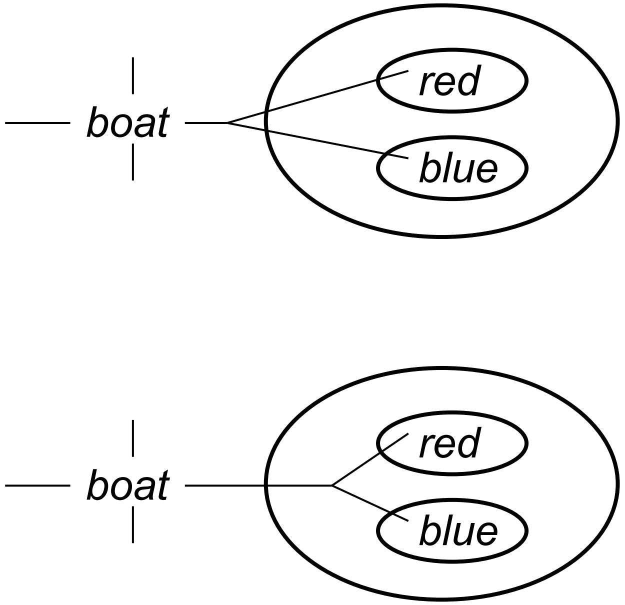
Beta EGs: Lines of Identity

The interpretation "line of identity" in Beta EGs creates a lot of confusion





different meanings

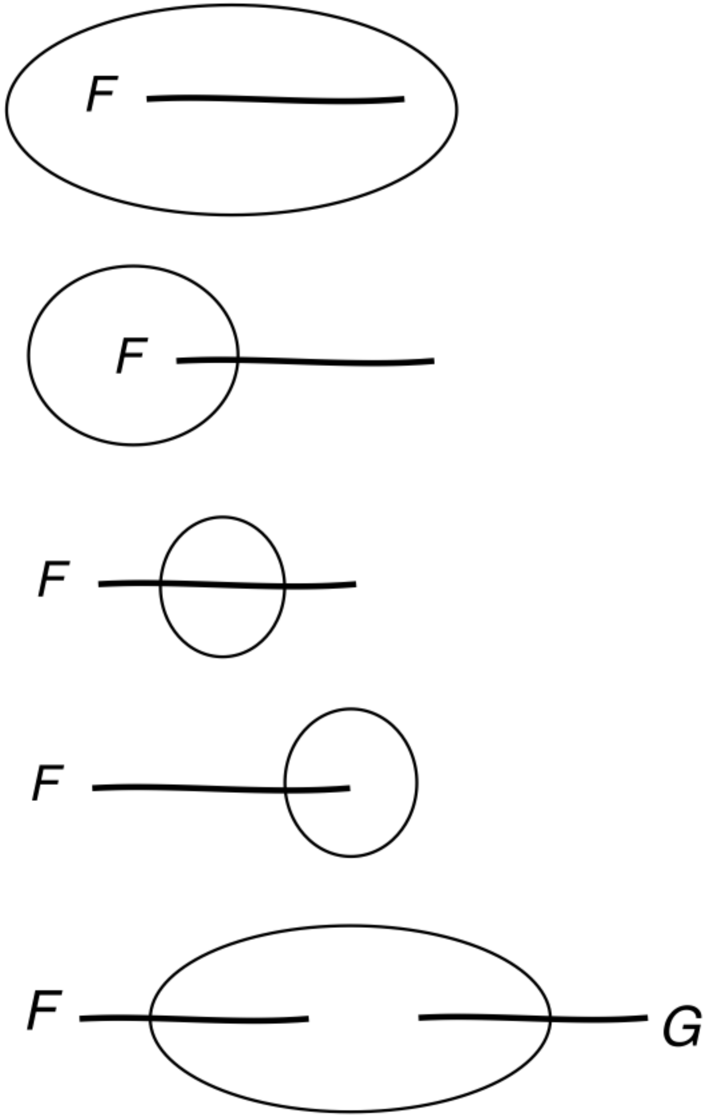


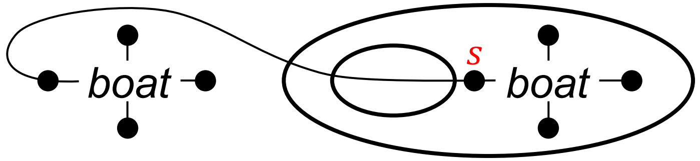
same meaning, but different interpretations

Figures in the left column taken from: "Sun-Joo Shin. The Iconic Logic of Peirce's Graphs. The MIT Press. 2002. chapter 3, p.42. https://doi.org/10.7551/mitpress/3633.001.0001 "

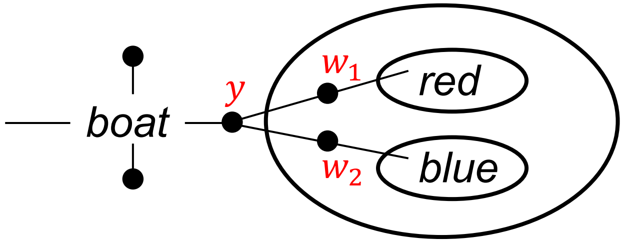
Beta EGs: Lines of Identity

Quantification via a visual symbol *different* from the line could have avoided those difficulties

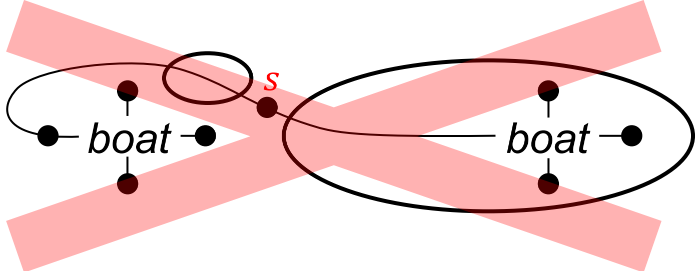




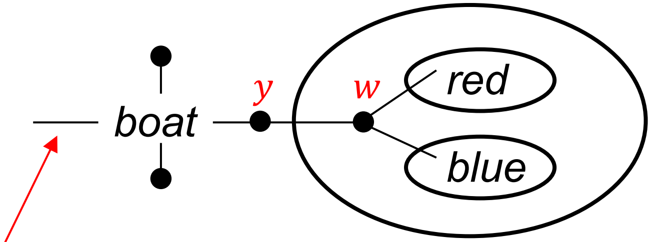
$\exists x,y,z,w\ [\text{Boat}(x,y,z,w)\ \land \neg(s,t,u,v\ [\text{Boat}(s,t,u,v) \land x{\neq}s])]$



$\{(x) \mid \exists y,z,u\ [\text{Boat}(z,x,y,u) \land \neg(\exists w_1,w_2\ [w_1=y \land w_2=y \land \neg(w_1=\text{'red'}) \land \neg(w_2=\text{'blue'})])]\}$



$\exists x,y,z,w,s\ [\text{Boat}(x,y,z,w) \land x{\neq}s \land \neg(\exists t,u,v\ [\text{Boat}(s,t,u,v)])]$



$\{(x) \mid \exists y,z,u\ [\text{Boat}(z,x,y,u) \land \neg(\exists w\ [w=y \land \neg(w=\text{'red'}) \land \neg(w=\text{'blue'})])]\}$

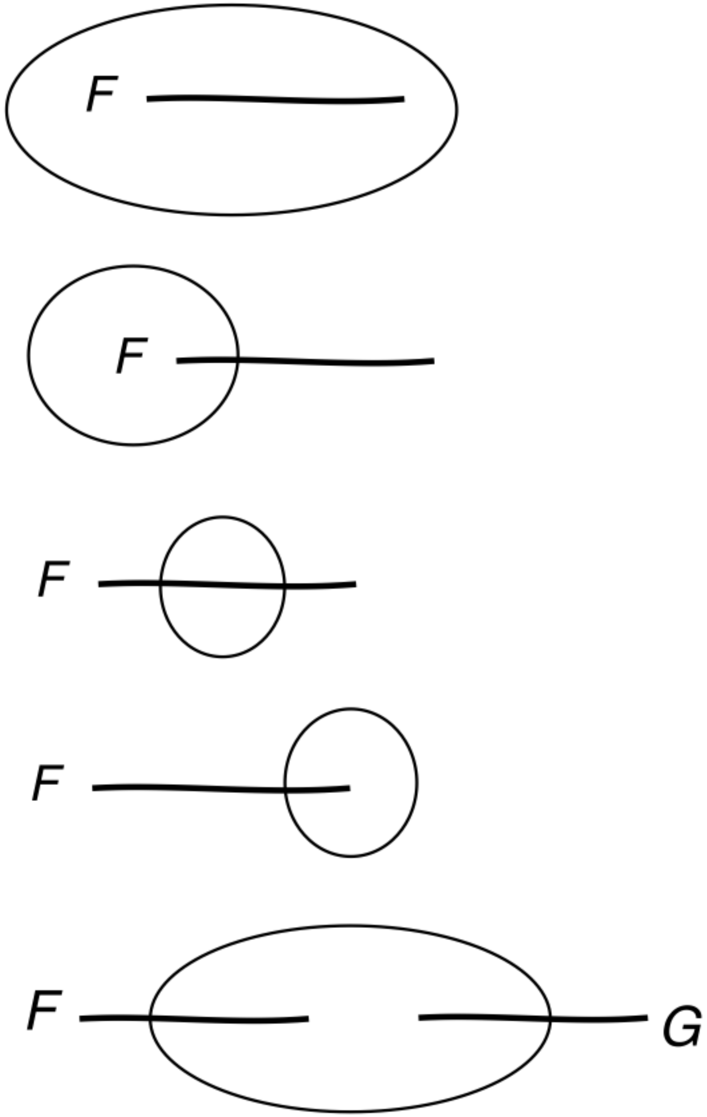
Here, the bullet (as in string diagrams, or a concept node as in conceptual graphs) is quantified.

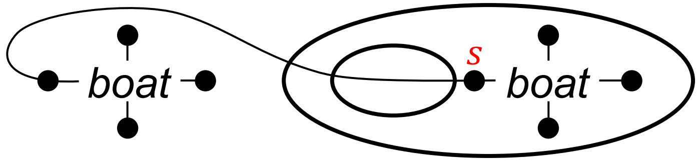
- Free variables are possible
- the interpretation is easy

Figures in the left column taken from: "Sun-Joo Shin. The Iconic Logic of Peirce's Graphs. The MIT Press. 2002. chapter 3, p.42. https://doi.org/10.7551/mitpress/3633.001.0001 "

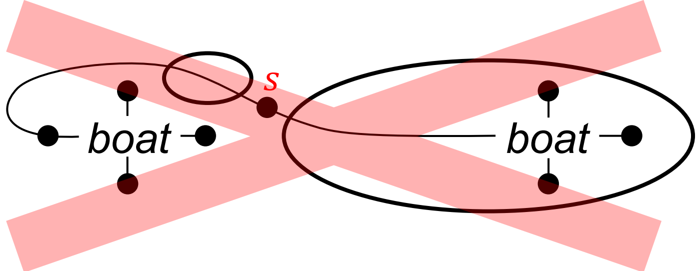
Beta EGs: Lines of Identity

Quantification via a visual symbol *different* from the line could have avoided those difficulties

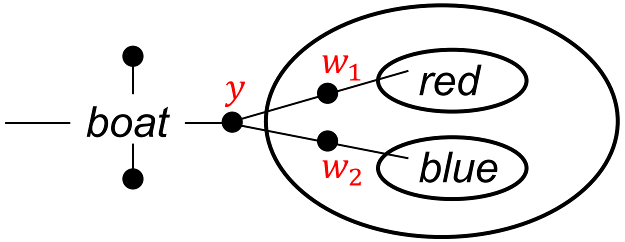




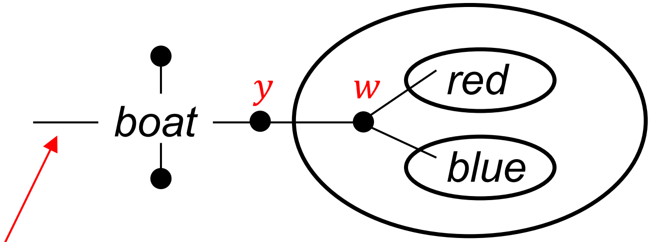
$\exists x,y,z,w\ [Boat(x,y,z,w)\ \wedge \neg(s,t,u,v\ [Boat(s,t,u,v) \wedge x \neq s])]$



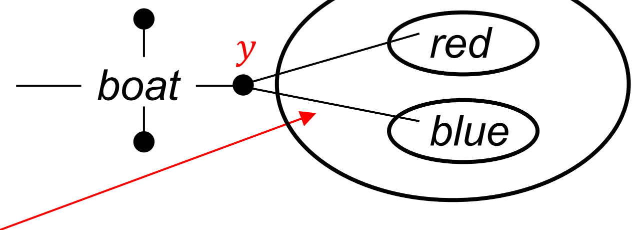
$\exists x,y,z,w,s\ [Boat(x,y,z,w) \wedge x \neq s \wedge \neg(\exists t,u,v\ [Boat(s,t,u,v)])]$



$\{(x) \mid \exists y,z,u\ [Boat(z,x,y,u) \wedge \neg(\exists w_1,w_2\ [w_1=y \wedge w_2=y \wedge \neg(w_1='red') \wedge \neg(w_2='blue')])]\}$



$\{(x) \mid \exists y,z,u\ [Boat(z,x,y,u) \wedge \neg(\exists w\ [w=y \wedge \neg(w='red') \wedge \neg(w='blue')])]\}$



$\{(x) \mid \exists y,z,u\ [Boat(z,x,y,u) \wedge \neg(\neg(y='red') \wedge \neg(y='blue'))]\}$

Here, the bullet (as in string diagrams, or a concept node as in conceptual graphs) is quantified.

- Free variables are possible
- the interpretation is easy
- a line can traverse a cut w/o quantification in that cut

Figures in the left column taken from: "Sun-Joo Shin. The Iconic Logic of Peirce's Graphs. The MIT Press. 2002. chapter 3, p.42. https://doi.org/10.7551/mitpress/3633.001.0001 "
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

1st abuse of lines and how to avoid it

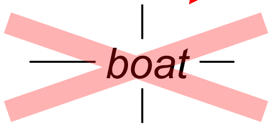
Lines of Identify (LIs) in Beta EGs have multiple meanings: existential quantification and identity = equality which leads to possible ambiguities.

RULE 1:

Lines should not be used for existential quantification.

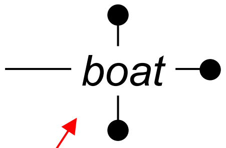
Existential quantification of variables as in DRC is better done with a dedicated symbol (e.g. the bullet in string diagrams, or concept nodes in conceptual graphs)

Domain RC



∃x,y,z,u[Boat(x,y,z,u)]

Beta EGs



{(y) | ∃x,z,u[Boat(x,y,z,u)]}

Beta EGs, adapted with bullets from string diagrams

Using bullets (or concept nodes) for quantification, and lines as symbol for equality

1st abuse of lines and how to avoid it

RULE 1:

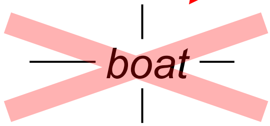
Lines should not be used for existential quantification.

Existential quantification of variables as in DRC is better done with a dedicated symbol (e.g. the bullet in string diagrams, or concept nodes in conceptual graphs)

Or, alternatively, existential quantification could be directly done with the actual relation symbols as in TRC (e.g. Relational Diagrams)

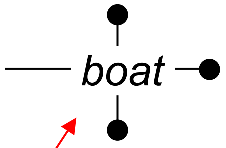
Lines of Identify (LIs) in Beta EGs have multiple meanings: existential quantification and identity = equality which leads to possible ambiguities.

Domain RC



$\exists x,y,z,u[\text{Boat}(x,y,z,u)]$

Beta EGs



$\{(y) \mid \exists x,z,u[\text{Boat}(x,y,z,u)]\}$

Beta EGs, adapted with bullets from string diagrams

Using bullets (or concept nodes) for quantification, and lines as symbol for equality

Tuple RC

The relational predicate is existentially quantified, as in TRC and SQL

boat

$\exists B \in \text{Boat}$

1st abuse of lines and how to avoid it

RULE 1:

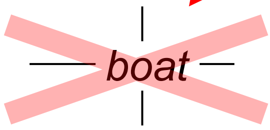
Lines should not be used for existential quantification.

Existential quantification of variables as in DRC is better done with a dedicated symbol (e.g. the bullet in string diagrams, or concept nodes in conceptual graphs)

Or, alternatively, existential quantification could be directly done with the actual relation symbols as in TRC (e.g. Relational Diagrams)

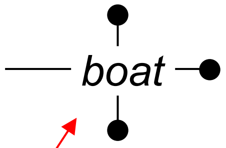
Lines of Identify (LIs) in Beta EGs have multiple meanings: existential quantification and identity = equality which leads to possible ambiguities.

Domain RC



∃x,y,z,u[Boat(x,y,z,u)]

Beta EGs



{(y) | ∃x,z,u[Boat(x,y,z,u)]}

Beta EGs, adapted with bullets from string diagrams

Using bullets (or concept nodes) for quantification, and lines as symbol for equality

Tuple RC

The relational predicate is existentially quantified, as in TRC and SQL

boat

∃B∈Boat

and then using a familiar UML notation

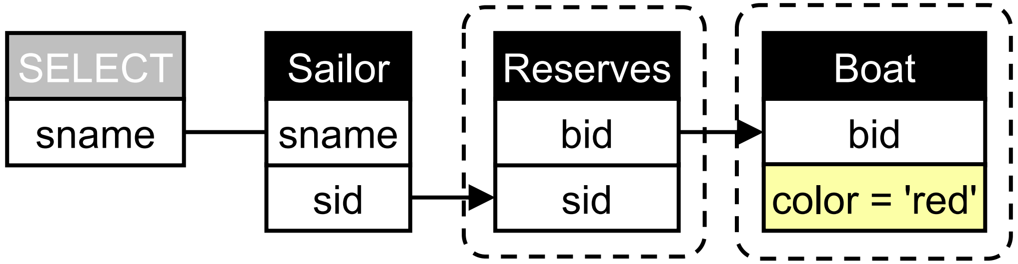
∃B∈Boat

Relational Diagrams

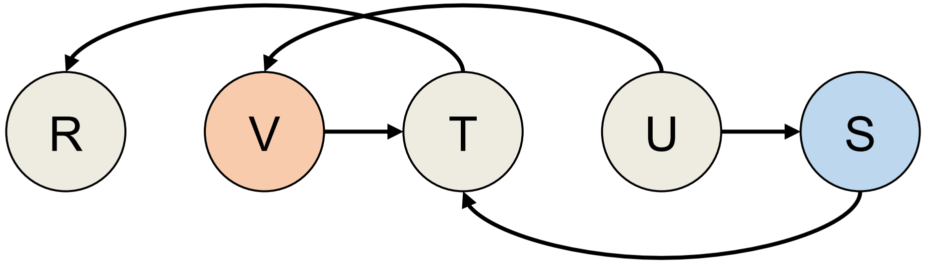
How to visualize reading direction for negations

QueryVis

Lines with arrowhead indicate
the reading direction

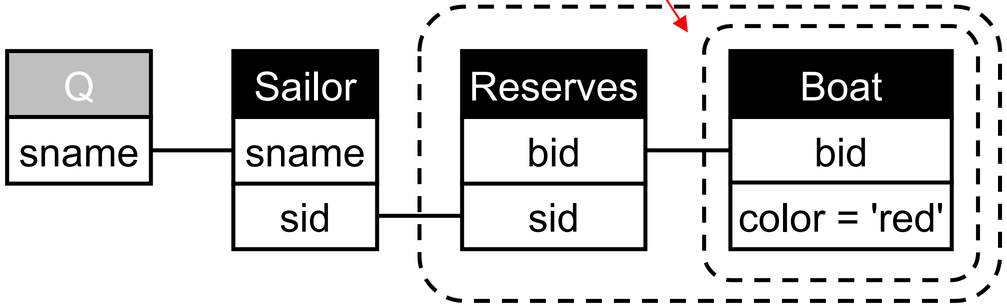


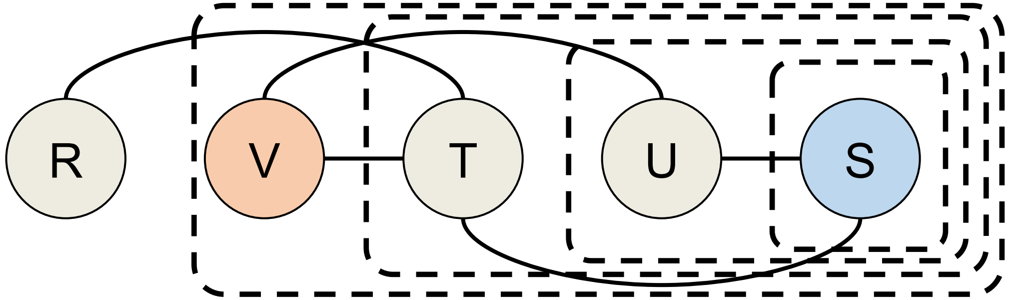
Arrows for making the reading order explicit
may reduce the visual clutter



Reading order follows implicitly from
the nesting hierarchy (outside in)

Relational Diagrams

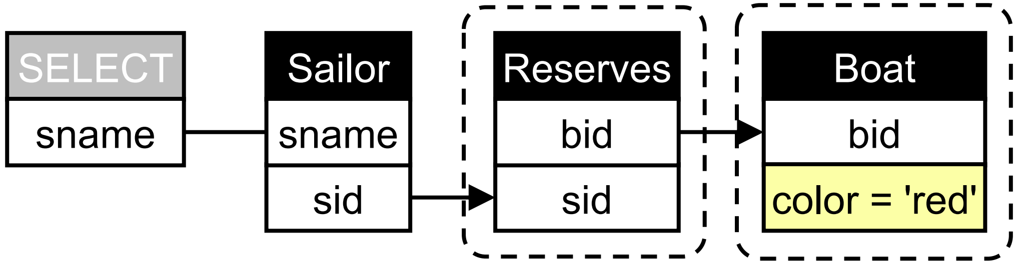




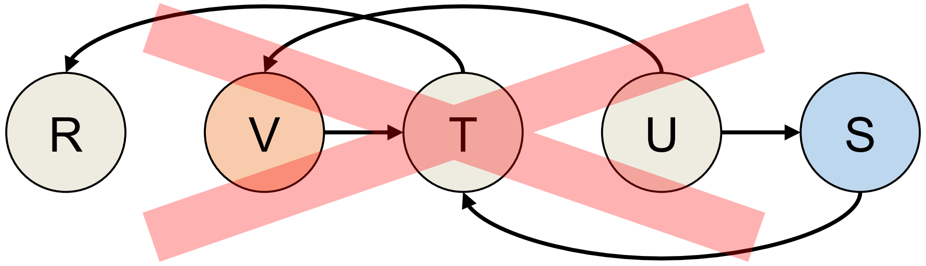
How to visualize reading direction for negations

QueryVis

Lines with arrowhead indicate the reading direction



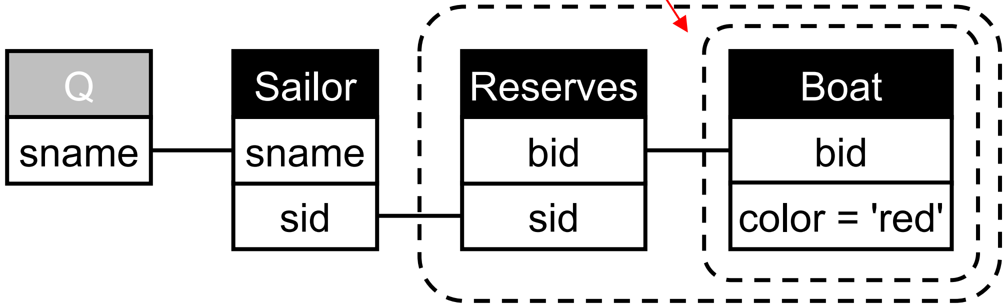
Arrows for making the reading order explicit may reduce the visual clutter

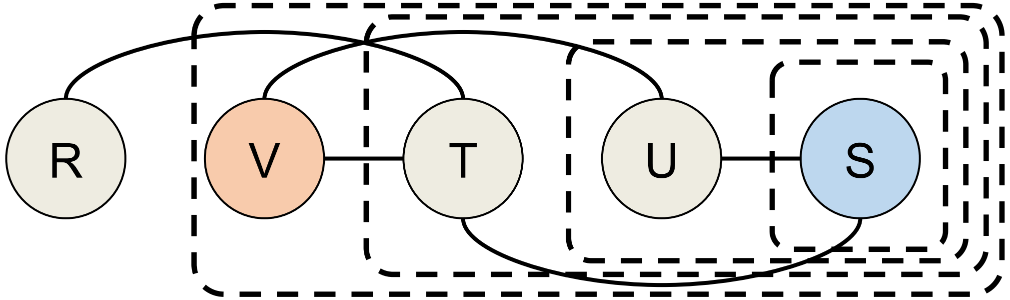


But, lines with arrowhead cannot represent all logical expressions, and can become ambiguous

Reading order follows implicitly from the nesting hierarchy (outside in)

Relational Diagrams





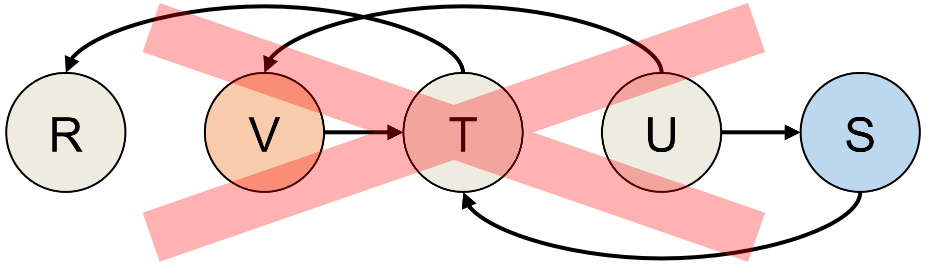
2nd abuse of lines and how to avoid it

RULE 2:

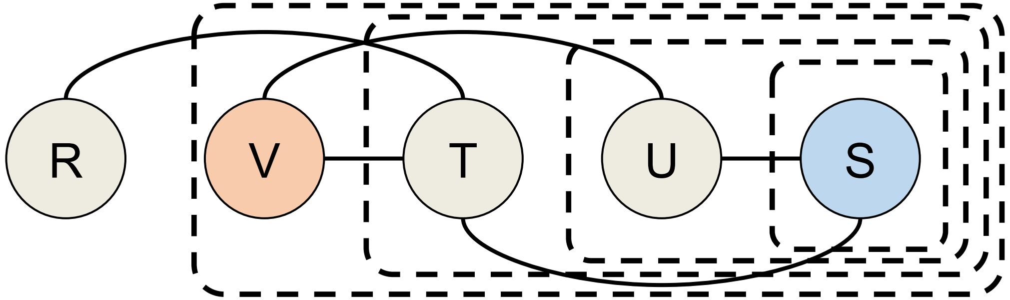
Lines should not be used as *only* symbol for indicating quantifier scoping (i.e. the nested negation scopes, via a reading order).

Nesting hierarchy can be unambiguously represented by explicit nesting of the scopes

Arrows for making the reading order explicit may reduce the visual clutter



But, lines with arrowhead cannot represent all logical expressions, and can become ambiguous



How to visualize a natural join between two tables?

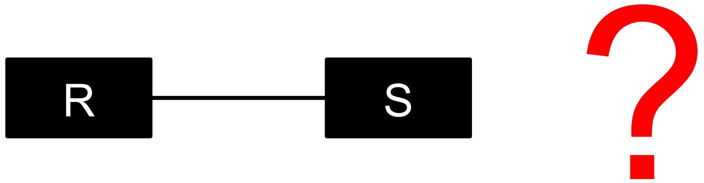
$R \bowtie S$

?

How to visualize a natural join between two tables?

$R \bowtie S$

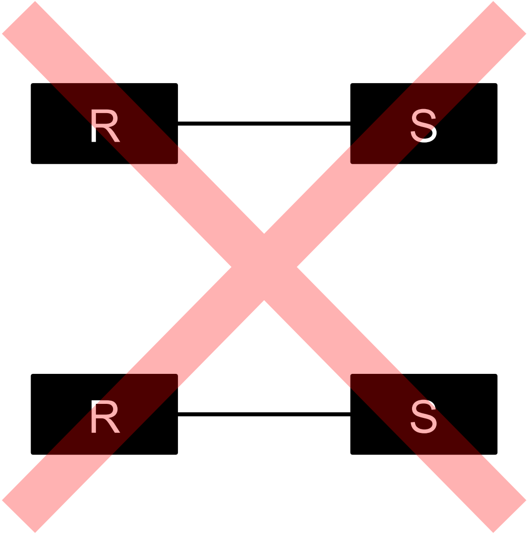
$R(A) \bowtie S(B)$



How to visualize a natural join between two tables?

$R \bowtie S$

$R(A) \bowtie S(B)$

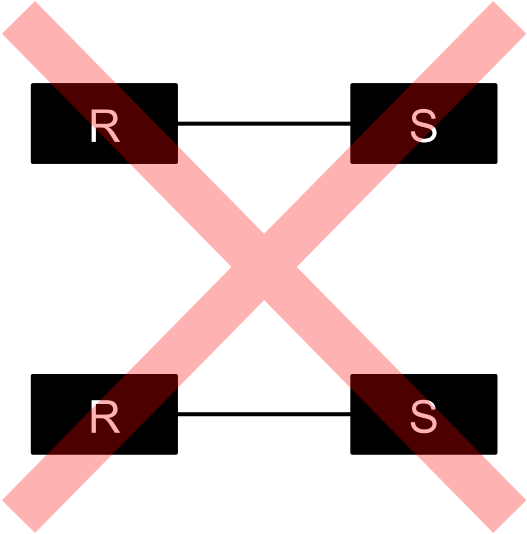


Lines representing joins (incl. Cartesian products) can lead to ambiguities (as in QBD) ☹

How to visualize a natural join between two tables?

$R \bowtie S$

$R(A) \bowtie S(B)$



Lines representing joins (incl. Cartesian products) can lead to ambiguities (as in QBD) ☹

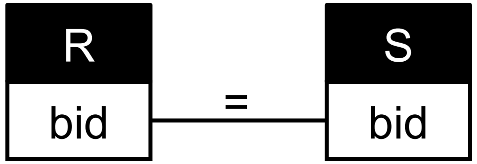
$R(A) \bowtie S(B)$

$\pi_{\emptyset}(R(A) \bowtie S(B))$

∃r∈R ∧ ∃s∈S

$R(A) \bowtie S(A)$

$= R \bowtie_{R.A=S.A} S$



Lines here represent the join condition b/w *attributes* not tables!

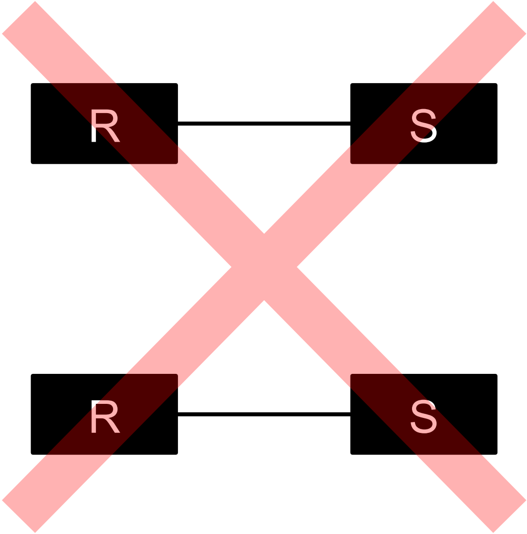
3rd abuse of lines and how to avoid it

RULE 3:

Lines should not be used for (Cartesian) joins.

Cartesian joins can be better represented by just putting tables on the same Canvas partition.

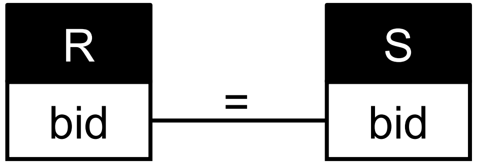
Lines are then representing the actual join condition between the respective attributes.



Lines representing joins (incl. Cartesian products) can lead to ambiguities (as in QBD) ☹

$$\pi_{\emptyset}(\mathrm{R}(\mathrm{A}) \bowtie \mathrm{S}(\mathrm{B}))$$

$$\exists r \in R \wedge \exists s \in S$$



Lines here represent the join condition b/w *attributes* not tables!

3 abuses of the line

Lines as visual symbol should only be used for comparisons (including equality, and with labels for disequality and inequality).

Lines should not be used for:

1. existential quantification (as in Peirce's Beta EGs)
2. joins = cross products (as in Query-By-Diagram)
3. for nesting directions with arrowheads (as in QueryVis)

3 abuses of the line

Lines as visual symbol should only be used for comparisons (including equality, and with labels for disequality and inequality).

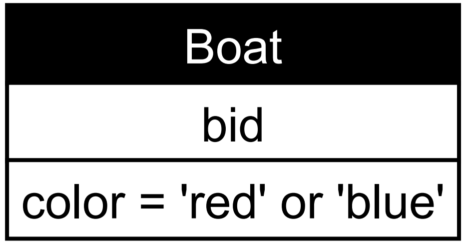
Lines should not be used for:

1. existential quantification (as in Peirce's Beta EGs)
   - instead, use dedicated symbols (e.g. bullets as in string diagrams for DRC)
   - instead, use placement of relational symbols (Relational Diagrams for TRC)
2. joins = cross products (as in Query-By-Diagram)
   - instead, use simple placement, and use lines as "join conditions"
3. for nesting directions with arrowheads (as in QueryVis)
   - instead, use the topological nesting of scoping boxes (Relational Diagrams)

Open issues

1: Showing Disjunctions diagrammatically is hard

Text:



Boat

bid

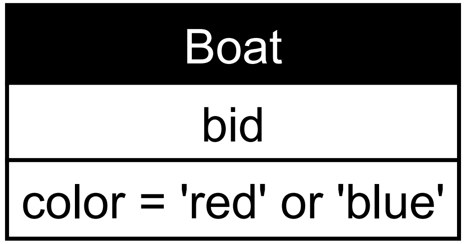
color = 'red' or 'blue'

Diagram:

?

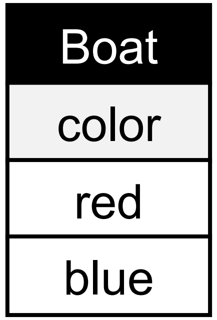
1: Showing Disjunctions diagrammatically is hard

Text:

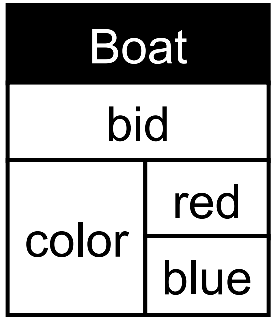


But it can get far more complicated ☹

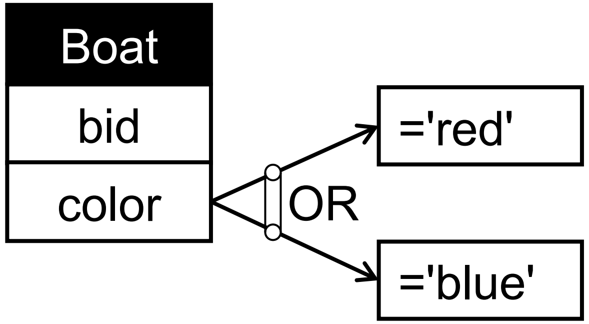
Diagram (or claimed "visual"):



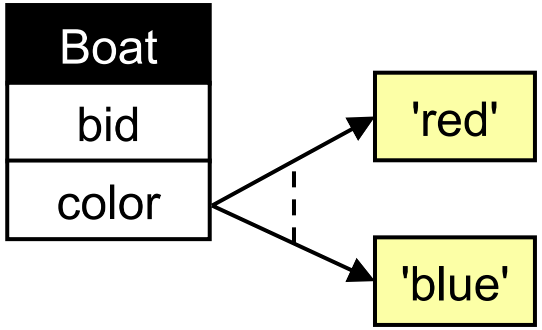
QBE (1977)



TableTalk (1991)



Visual SQL (2003)



QueryVis (2011)

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))
```

?

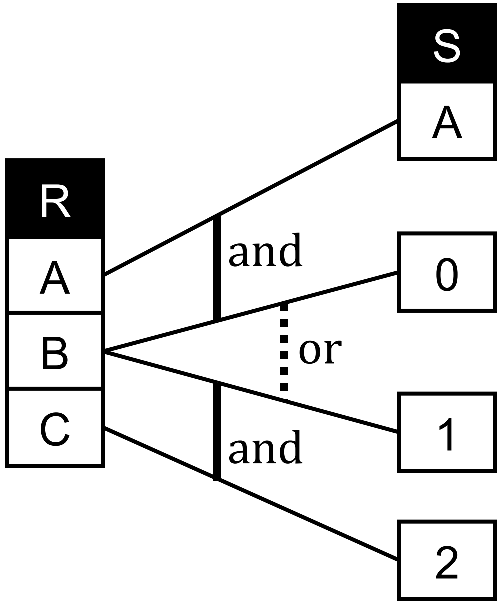
1: Showing Disjunctions diagrammatically is hard

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

How to visually represent arbitrary Boolean formulas?

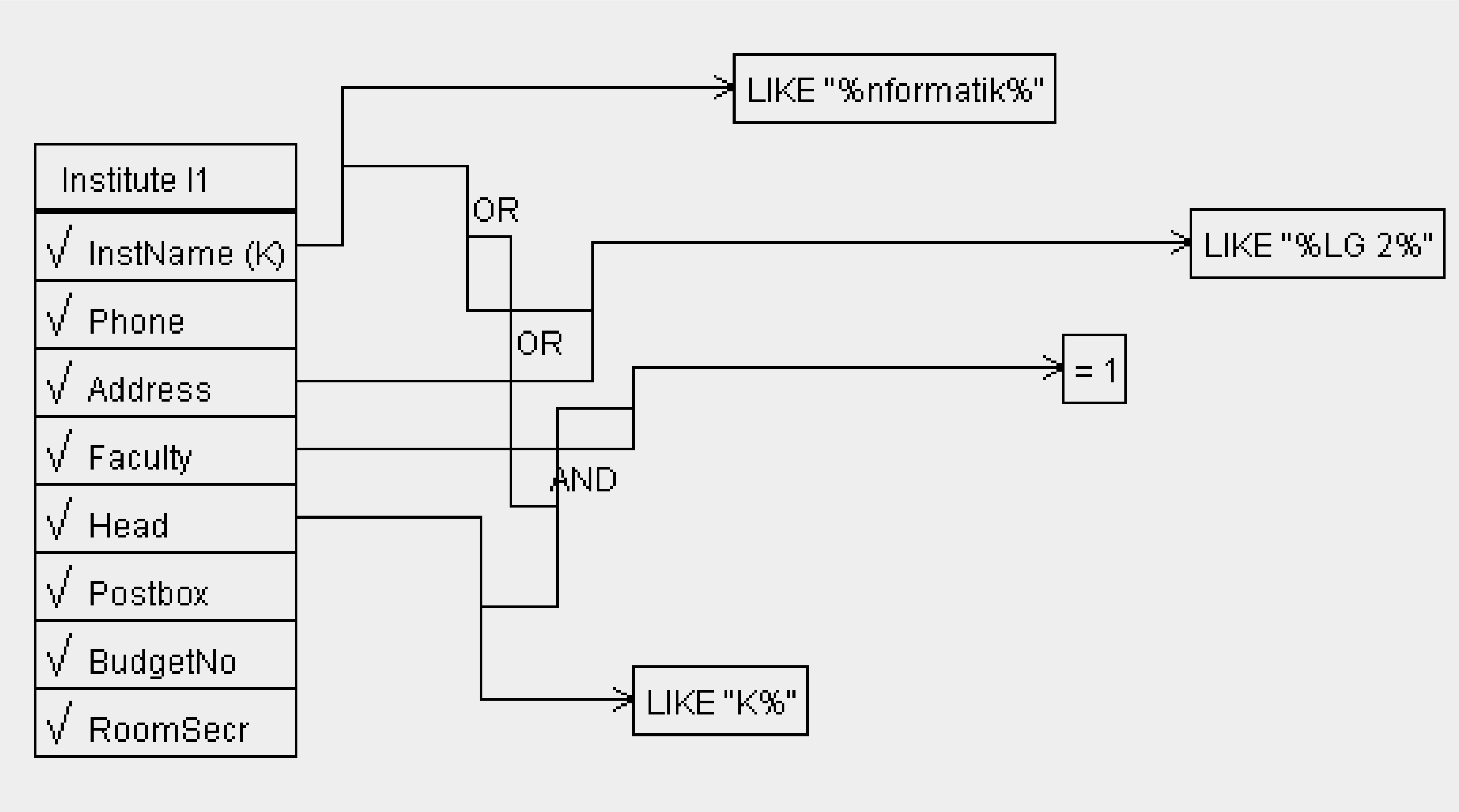
```
select exists
    (select *
    from R
    where (R.B = 0
        and exists
            (select *
            from S
            where R.A = S.A))
        and exists
    or (R.B = 1
        and R.C = 2))
```

?



quickly becomes ambiguous

1: Showing Disjunctions diagrammatically is hard

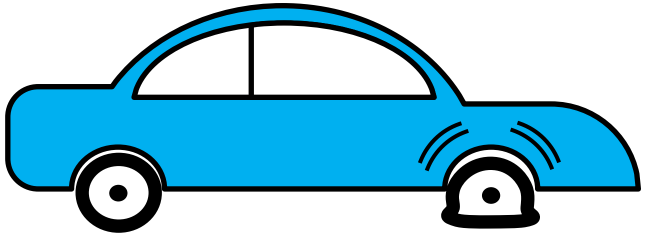


There is no simple way to represent Boolean formulas.

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

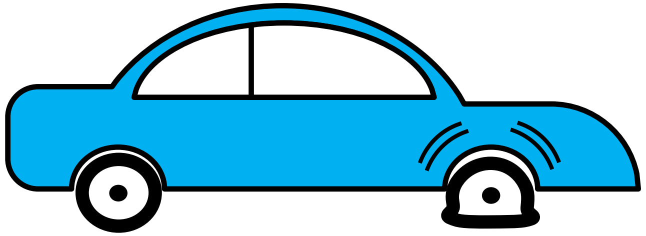


I see a car that is blue
OR that has a flat tire

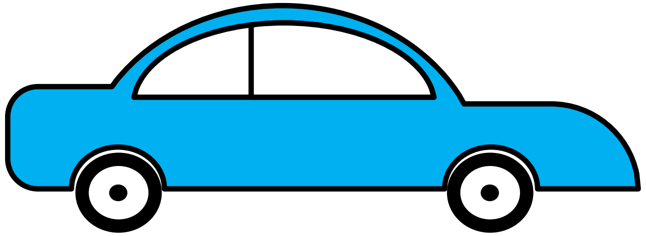
?

1: Why is visualizing disjunctive information harder?

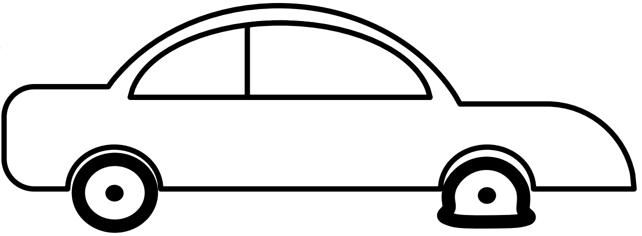
I see a car that is blue
AND that has a flat tire



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

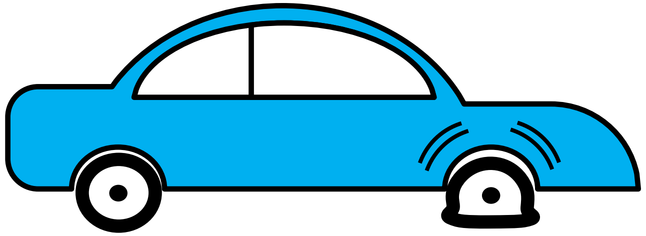


OR



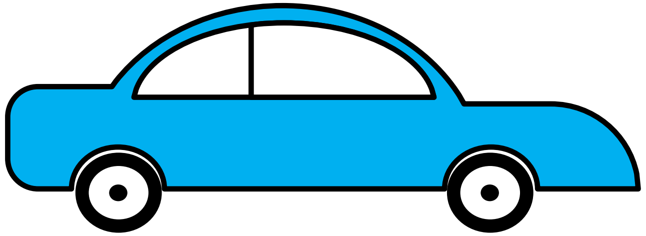
1: Why is visualizing disjunctive information harder?

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

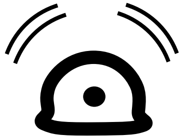


color="blue"
AND tire="flat"

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



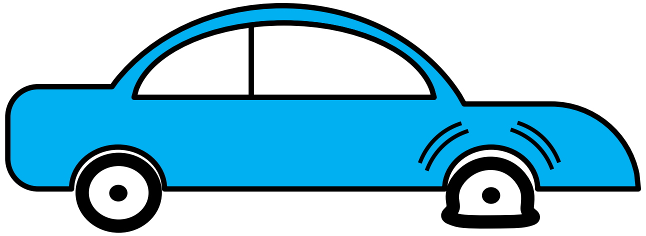
OR



color="blue" OR tire="flat"

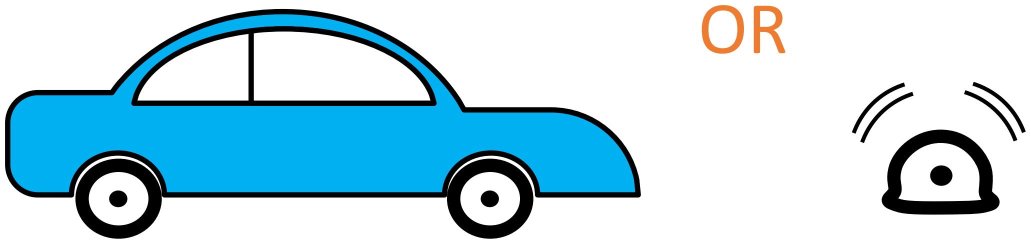
1: Why is visualizing disjunctive information harder?

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

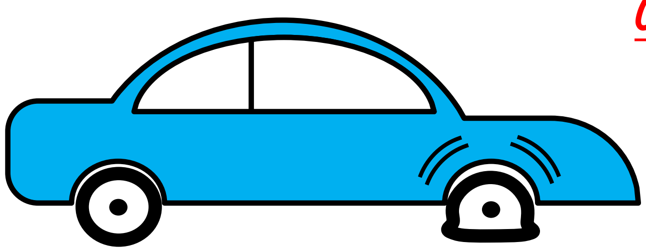


$\sigma_{\text{color="blue" OR tire="flat}}(\dots)$

1: Why is visualizing disjunctive information harder?

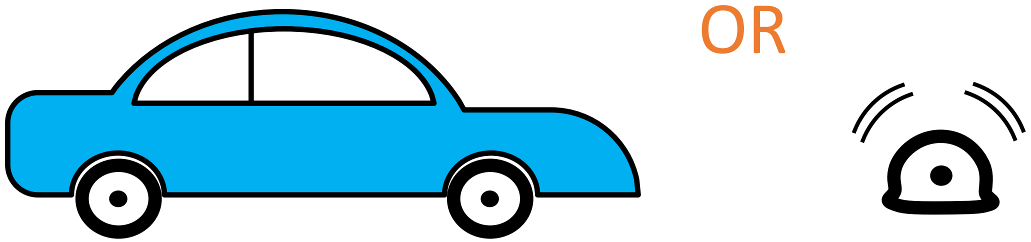
"A situation only displays <u>conjunctive</u> information."*

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

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

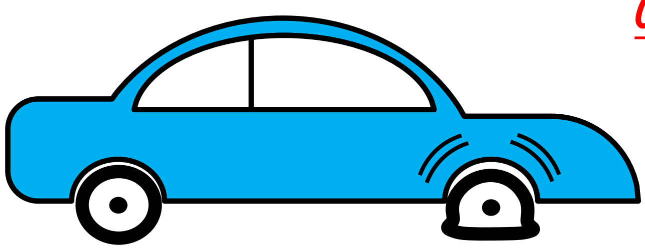
---

* Sun-Joo Shin, "The logical status of diagrams", 1995. https://doi.org/10.1017/CBO9780511574696

1: Why is visualizing disjunctive information harder?

"A situation only displays conjunctive information."*

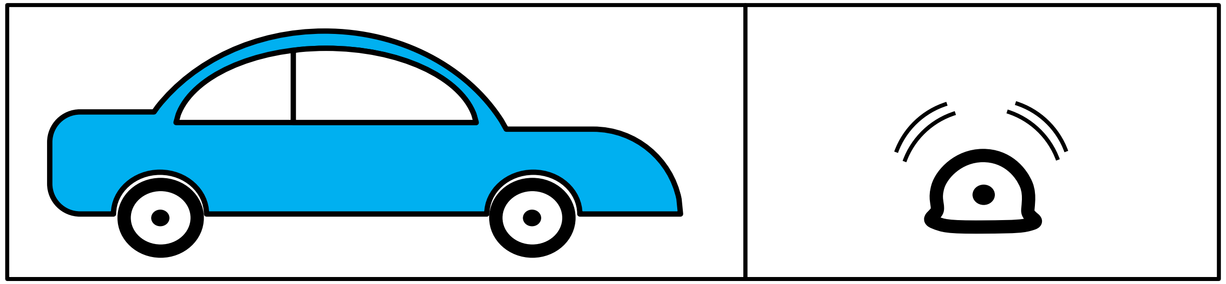
I see a car that is blue
AND that has a flat tire



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

Peirce's "compartments" strategy (a syntactic device) for disjunctions:

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

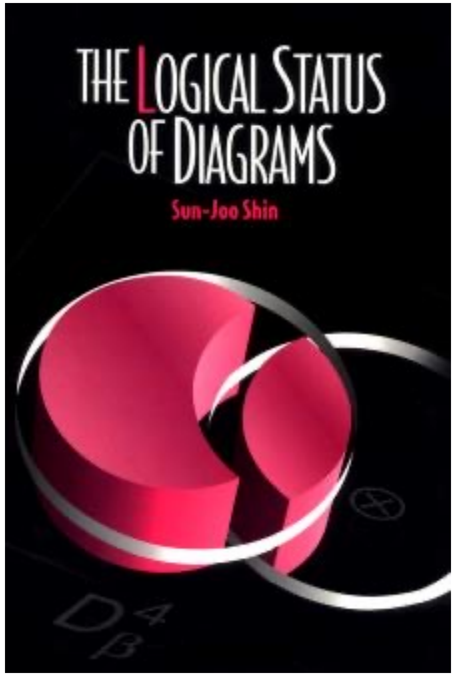


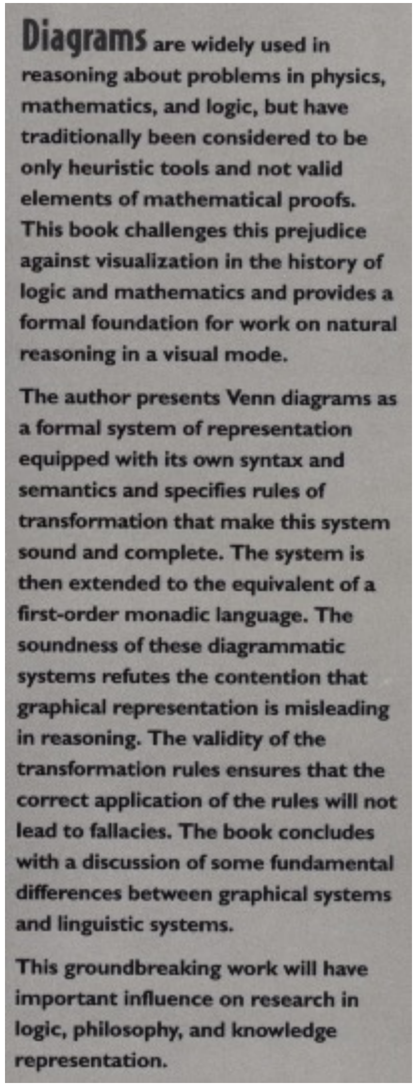
$\sigma_{\text{color="blue" OR tire="flat}}(\ldots)$

---

* Sun-Joo Shin, "The logical status of diagrams", 1995. https://doi.org/10.1017/CBO9780511574696

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

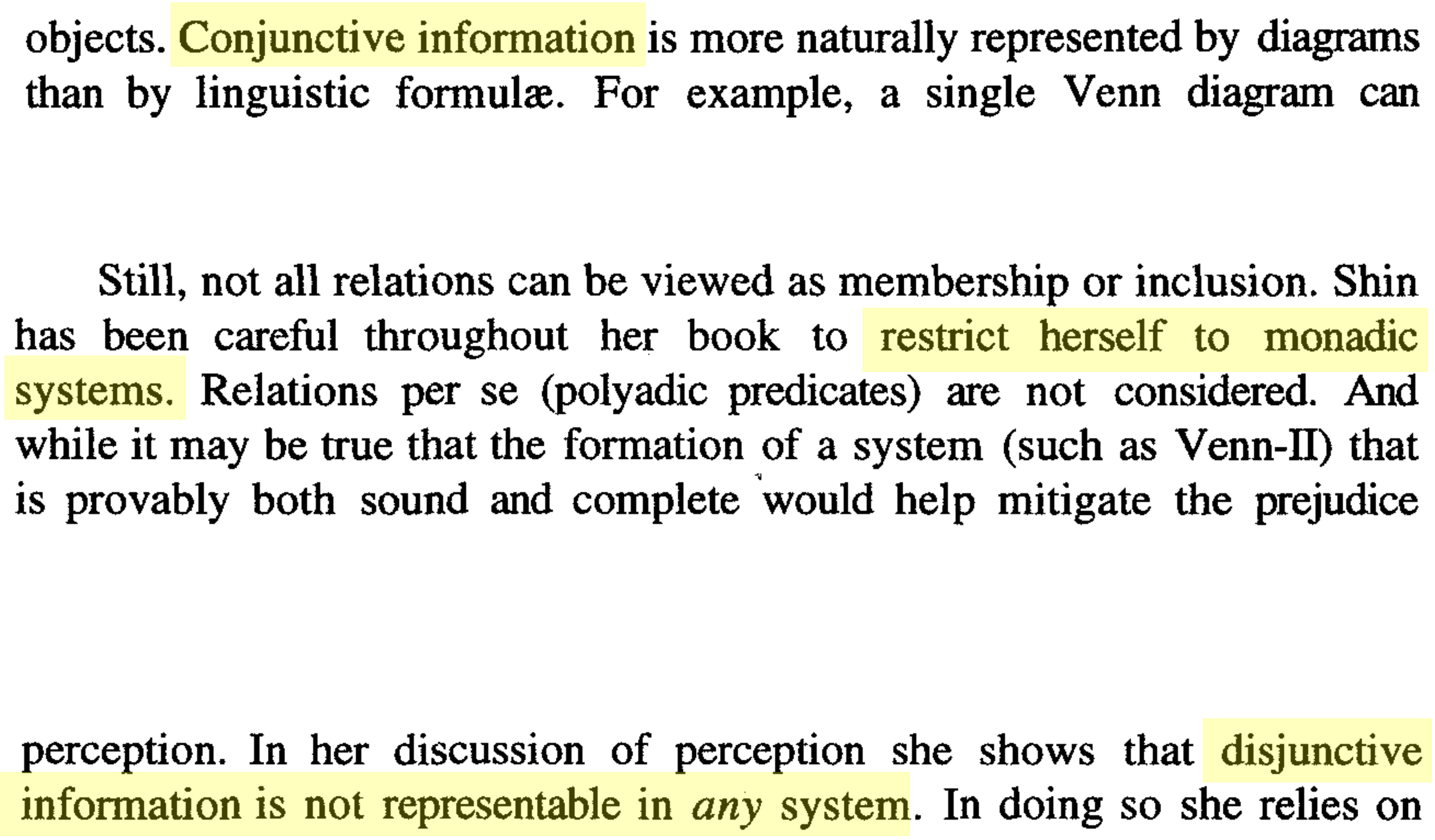




Diagrams are widely used in reasoning about problems in physics, mathematics, and logic, but have traditionally been considered to be only heuristic tools and not valid elements of mathematical proofs. This book challenges this prejudice against visualization in the history of logic and mathematics and 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", 1995. https://doi.org/10.1017/CBO9780511574696

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

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

aggregates

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

groupings

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

?

3. Other extensions beyond FOL

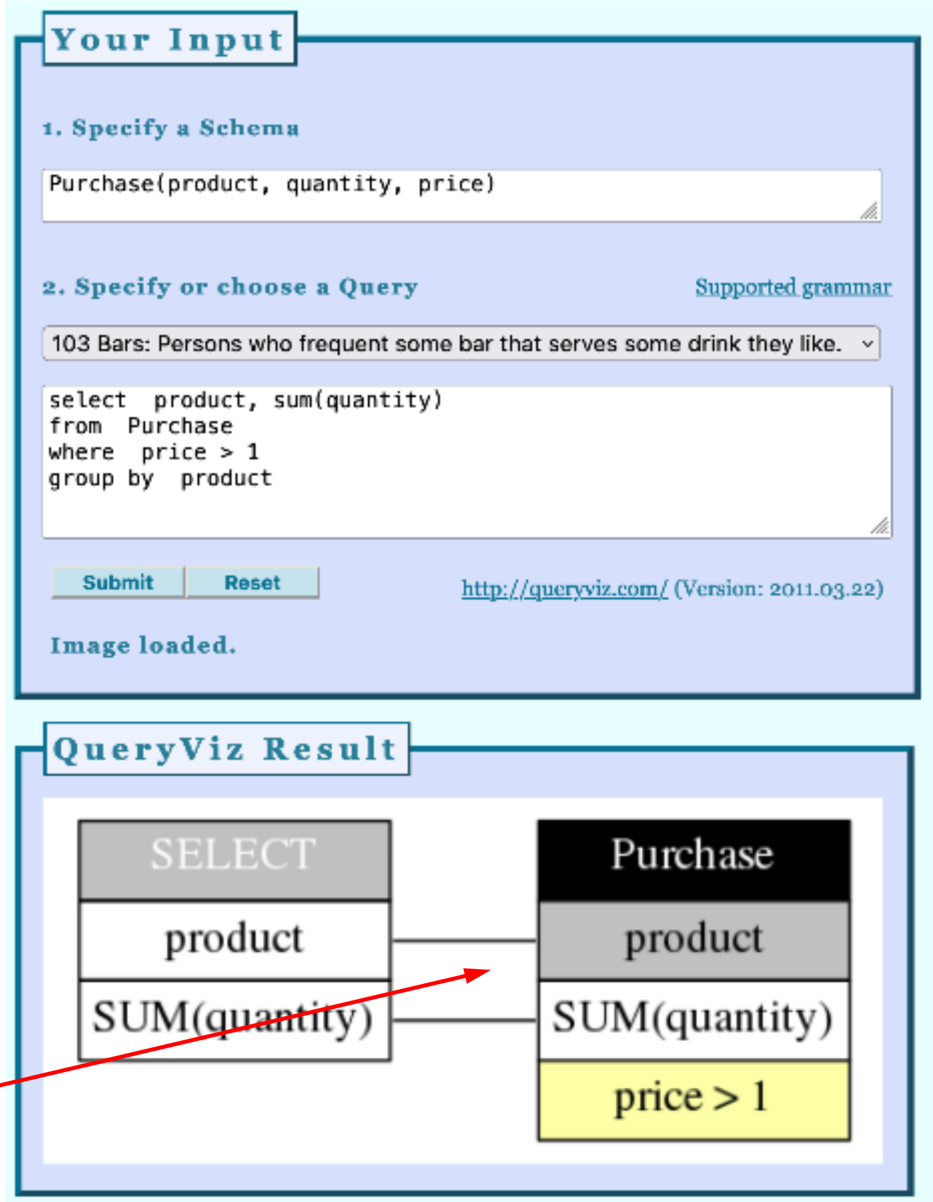
aggregates

```
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
```



1. Specify a Schema

```
Purchase(product, quantity, price)
```

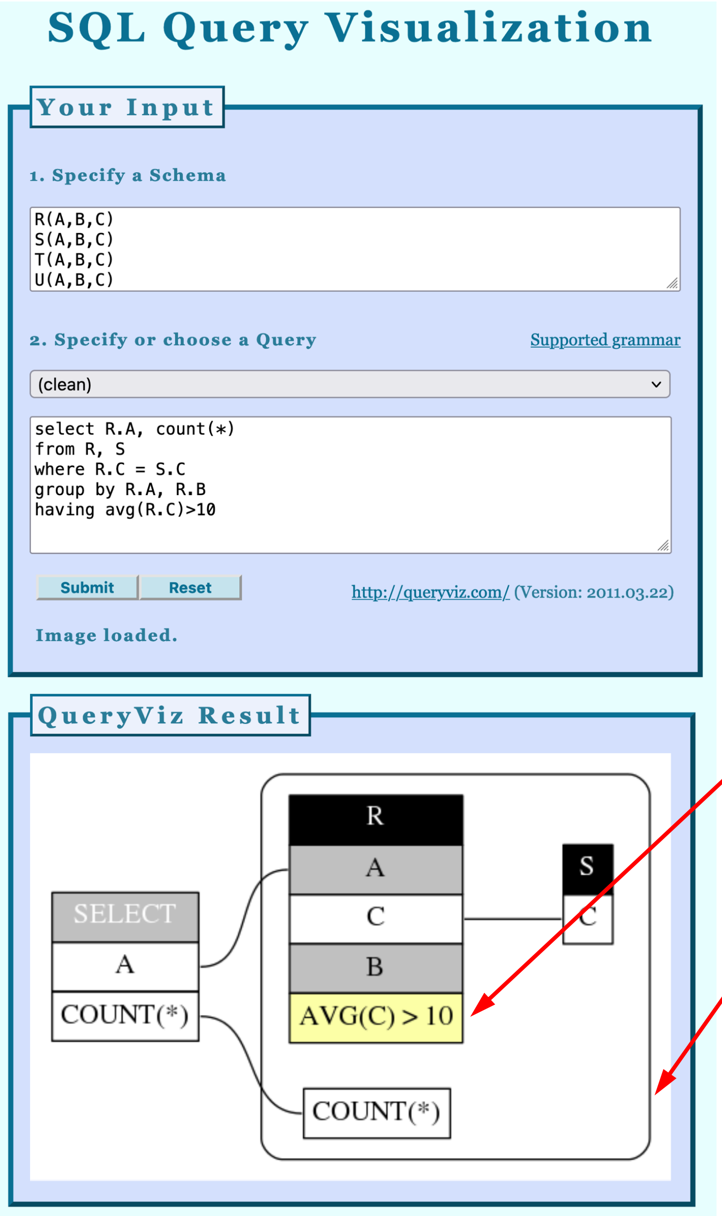
2. Specify or choose a Query

103 Bars: Persons who frequent some bar that serves some drink they like.

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

1. group by attributes are shaded

3. Other extensions beyond FOL



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

Supported grammar

(clean)

```
select R.A, count(*)
from R, S
where R.C = S.C
group by R.A, R.B
having avg(R.C)>10
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

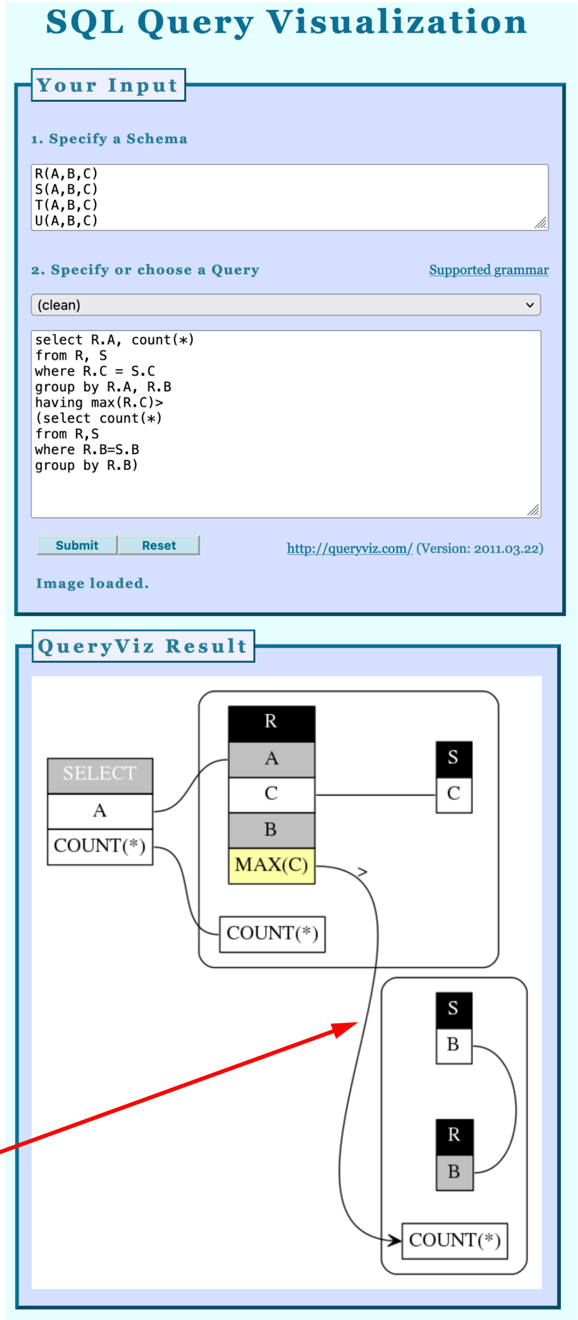
QueryViz Result

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,...)

2. having clause is treated like selections

3. count(*) aggregates interpreted on the whole join via grouping boxes

4. Nested queries in having clause



SQL Query Visualization

Your Input

1. Specify a Schema

```
R(A,B,C)
S(A,B,C)
T(A,B,C)
U(A,B,C)
```

2. Specify or choose a Query

Supported grammar

(clean)

```
select R.A, count(*)
from R, S
where R.C = S.C
group by R.A, R.B
having max(R.C)>
(select count(*)
from R,S
where R.B=S.B
group by R.B)
```

http://queryviz.com/ (Version: 2011.03.22)

Image loaded.

QueryViz Result

Figure source: Online QueryVis (formerly known as Online QueryViz): http://demo.queryvis.com

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

3. Other extensions beyond FOL

arithmetic predicates

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

- grouping & aggregates
- arithmetic predicates
- outer joins
- null values
- bag semantics
- recursion

?

4. We need a principled notion of relational patterns

What are "relational patterns"?

For first steps towards "relational patterns" see: "Gatterbauer, Dunne. On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages. SIGMOD 2024. https://dl.acm.org/doi/pdf/10.1145/3639316

Relational patterns

R(A,B), S(B)

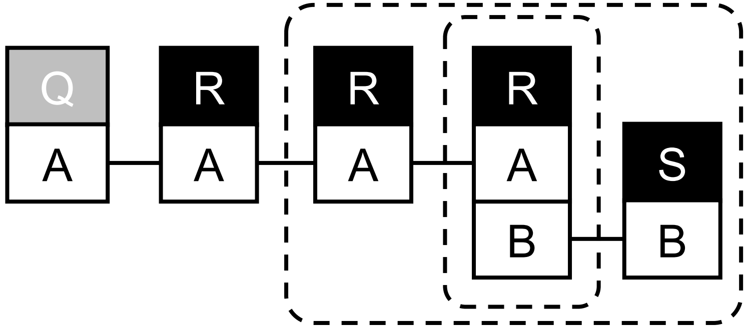
Datalog$^{\neg}$

$$I(x) :- R(x, \_), S(y), \neg R(x, y)$$
$$Q(x) :- R(x, \_), \neg I(x)$$

Relational Algebra:

$$\pi_A R - \pi_A((\pi_A R \times S) - R)$$

Relational Diagram:



Given 3 very different syntactic formulations of the same query (declarative, procedural, visual), what is a meaningful way to compare those different "query expressions"?

?

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

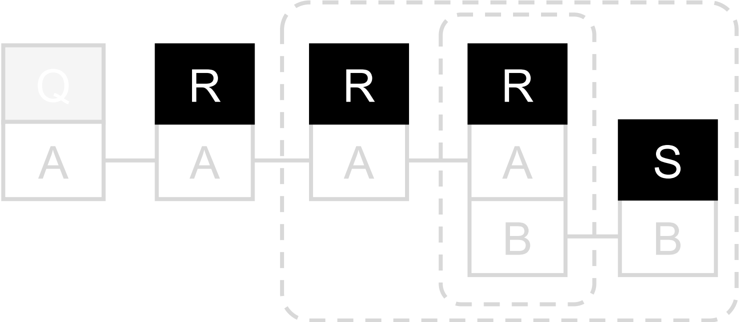
Relational patterns

R(A,B), S(B)

1. Focus on relational atoms only

$I(x) :- R(x, \_), S(y), \neg R(x, y)$
$Q(x) :- R(x, \_), \neg I(x)$

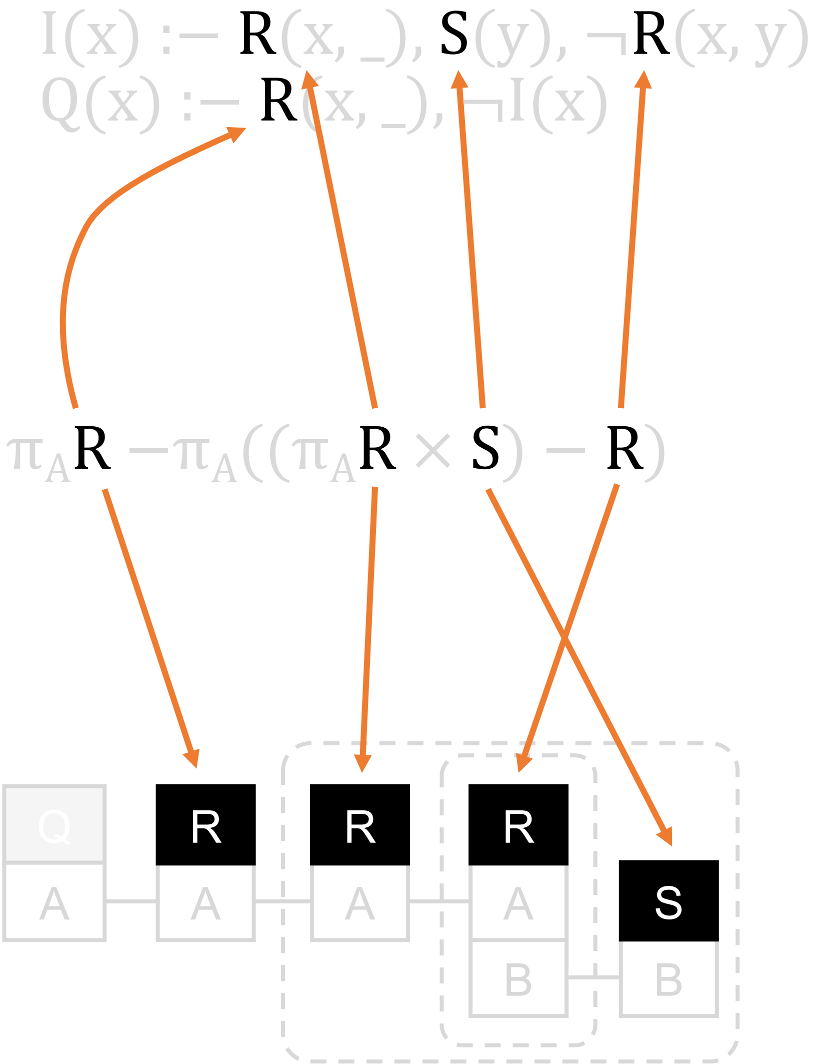
$\pi_A R - \pi_A((\pi_A R \times S) - R)$



Relational patterns

R(A,B), S(B)

1. Focus on relational atoms only
2. Find a table-preserving mapping between relational atoms

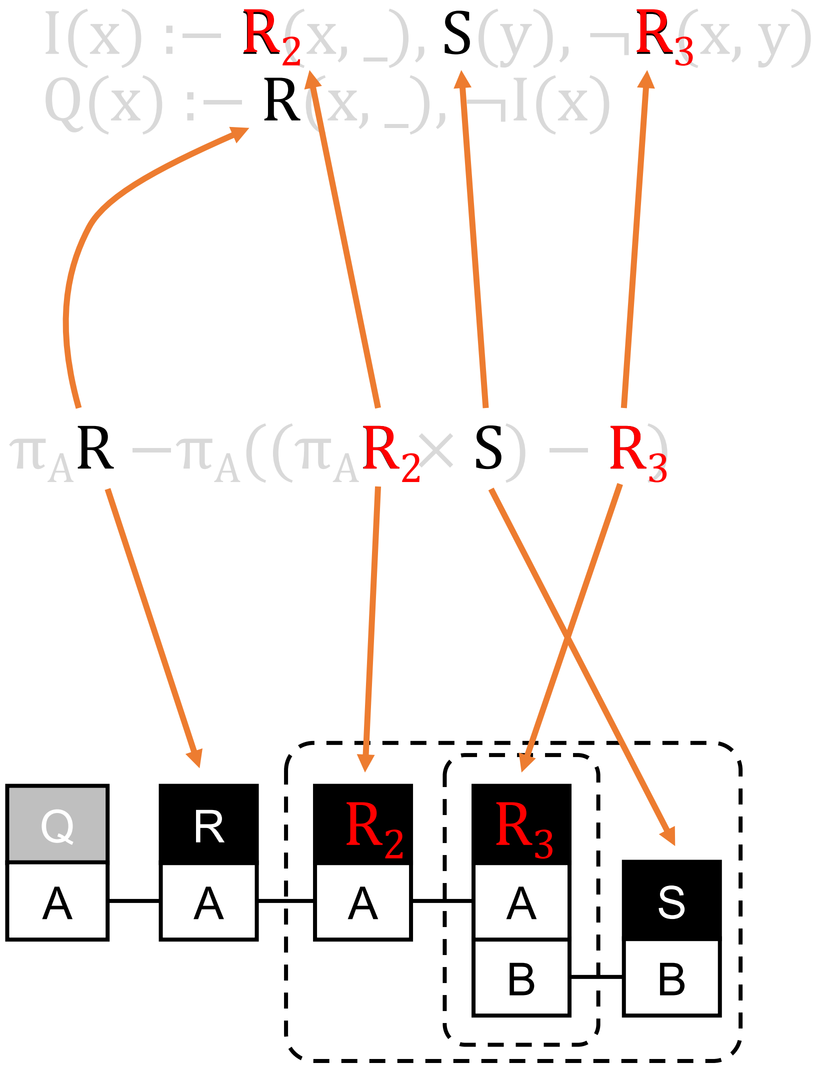


I(x) :– R(x, _), S(y), ¬R(x, y)
Q(x) :– R(x, _), ¬I(x)

$\pi_A R - \pi_A((\pi_A R \times S) - R)$

Relational patterns

$R(A,B), S(B)$



$I(x) :- R_2(x, \_), S(y), \neg R_3(x, y)$

$Q(x) :- R(x, \_), \neg I(x)$

$\pi_A R - \pi_A((\pi_A R_2 \times S) - R_3)$

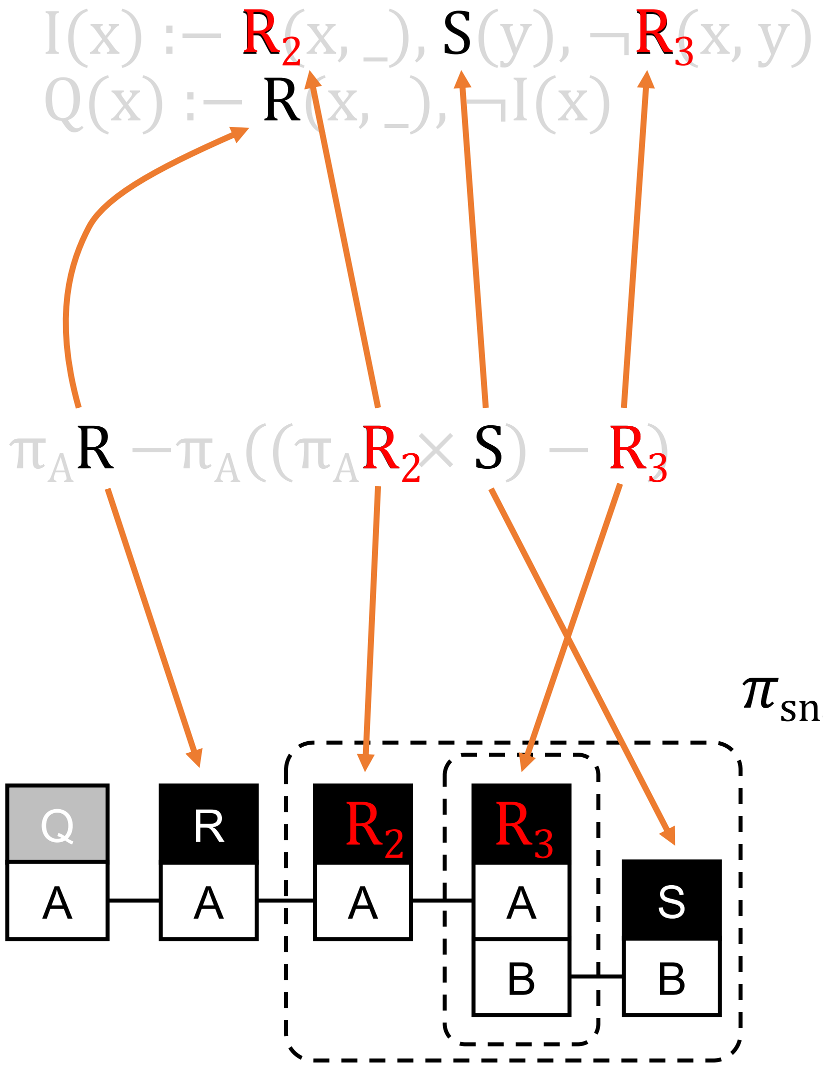
1. Focus on relational atoms only
2. Find a table-preserving mapping between relational atoms
3. But for "dissociated queries" (treat each EDB as different)

Are they still logically equivalent?

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

Why patterns?

R(A,B), S(B)



$I(x) :- R_2(x, \_), S(y), \neg R_3(x, y)$

$Q(x) :- R(x, \_), \neg I(x)$

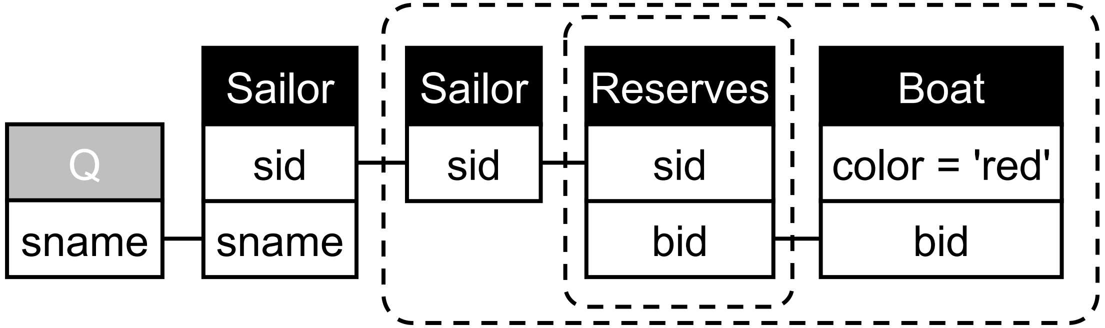
$\pi_A R - \pi_A((\pi_A R_2 \times S) - R_3)$

1. Focus on relational atoms only
2. Find a table-preserving mapping between relational atoms
3. But for "dissociated queries" (treat each EDB as different)

Are they still logically equivalent?

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

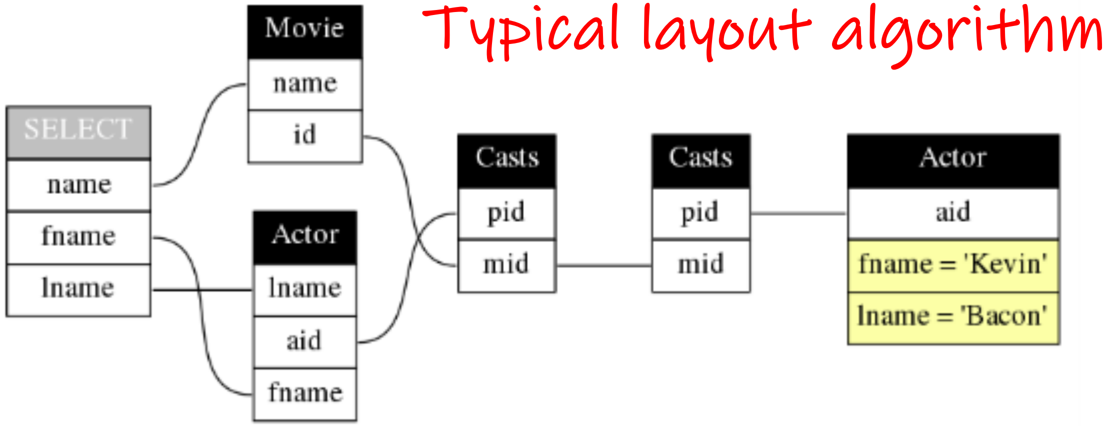
$$\pi_{\text{sname}}(\text{Sailor} \rhd_{\text{sid}}((\text{Sailor} \times \rho_{\text{color='red'}}\text{Boat}) \rhd_{\text{sid, bid}}\text{Reserves}))$$



Notice the similarities, up to renaming of tables ☺

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

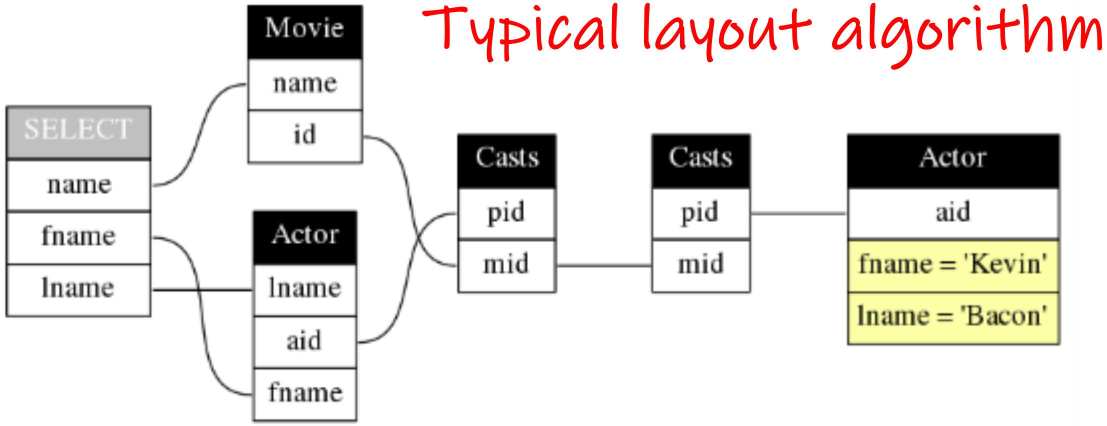
5. "Nice" layouts: Automatic layout algorithms



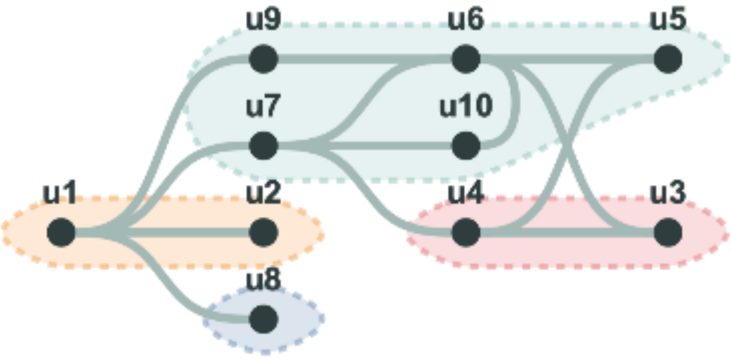
Typical layout algorithms cannot handle hierarchies well

?

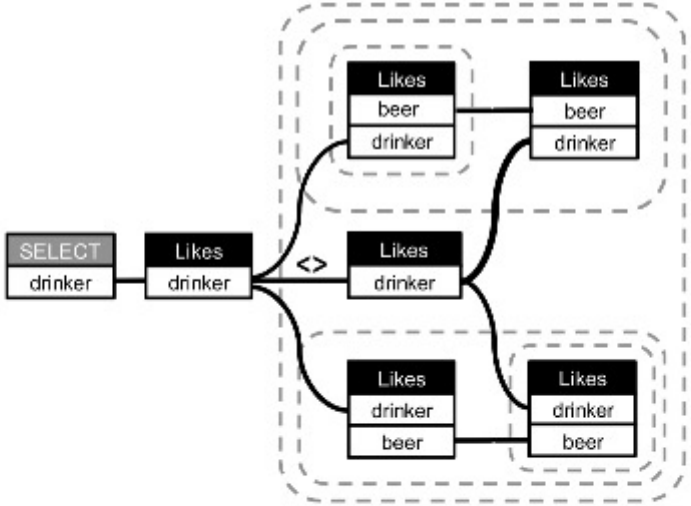
5. "Nice" layouts: Automatic layout algorithms

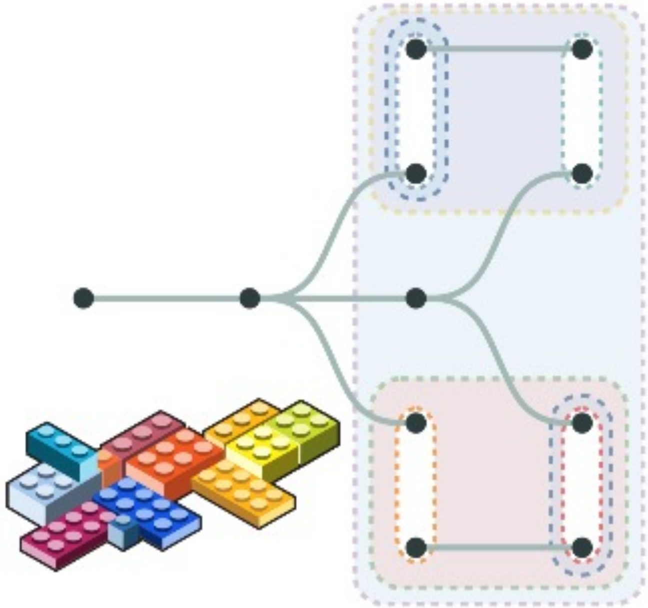


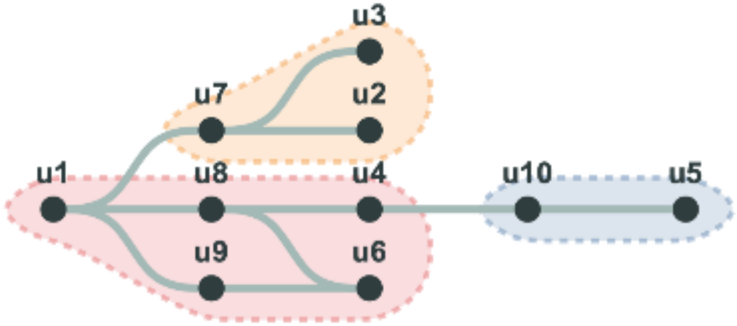
Typical layout algorithms cannot handle hierarchies well



How do we design principled layered node-link visualizations?



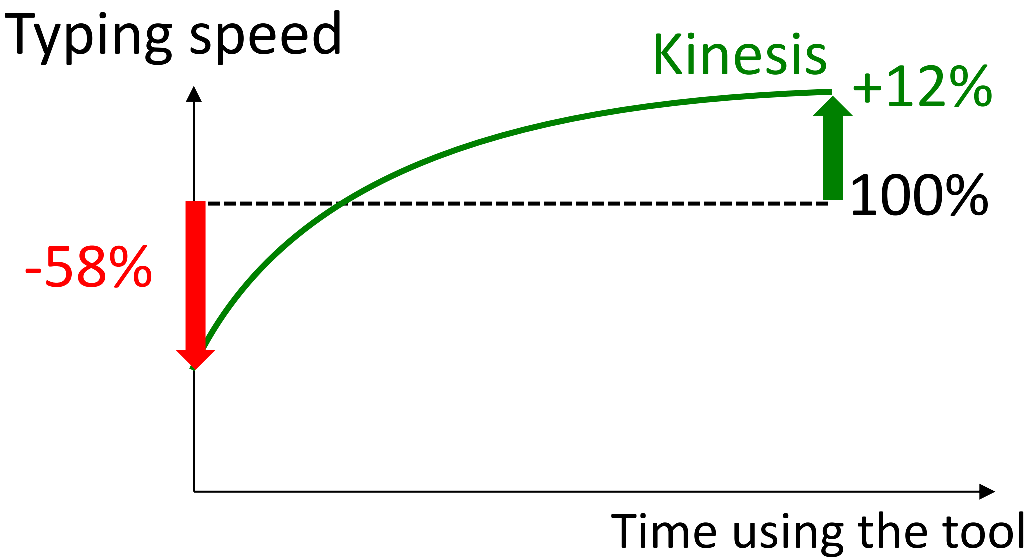




Source: STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations, IEEE TVC 2021, https://doi.org/10.1109/TVCG.2021.3114756
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

6. Principled User Studies (preregistered, beyond students)

How to design principled, reproducible, longitudinal user studies?



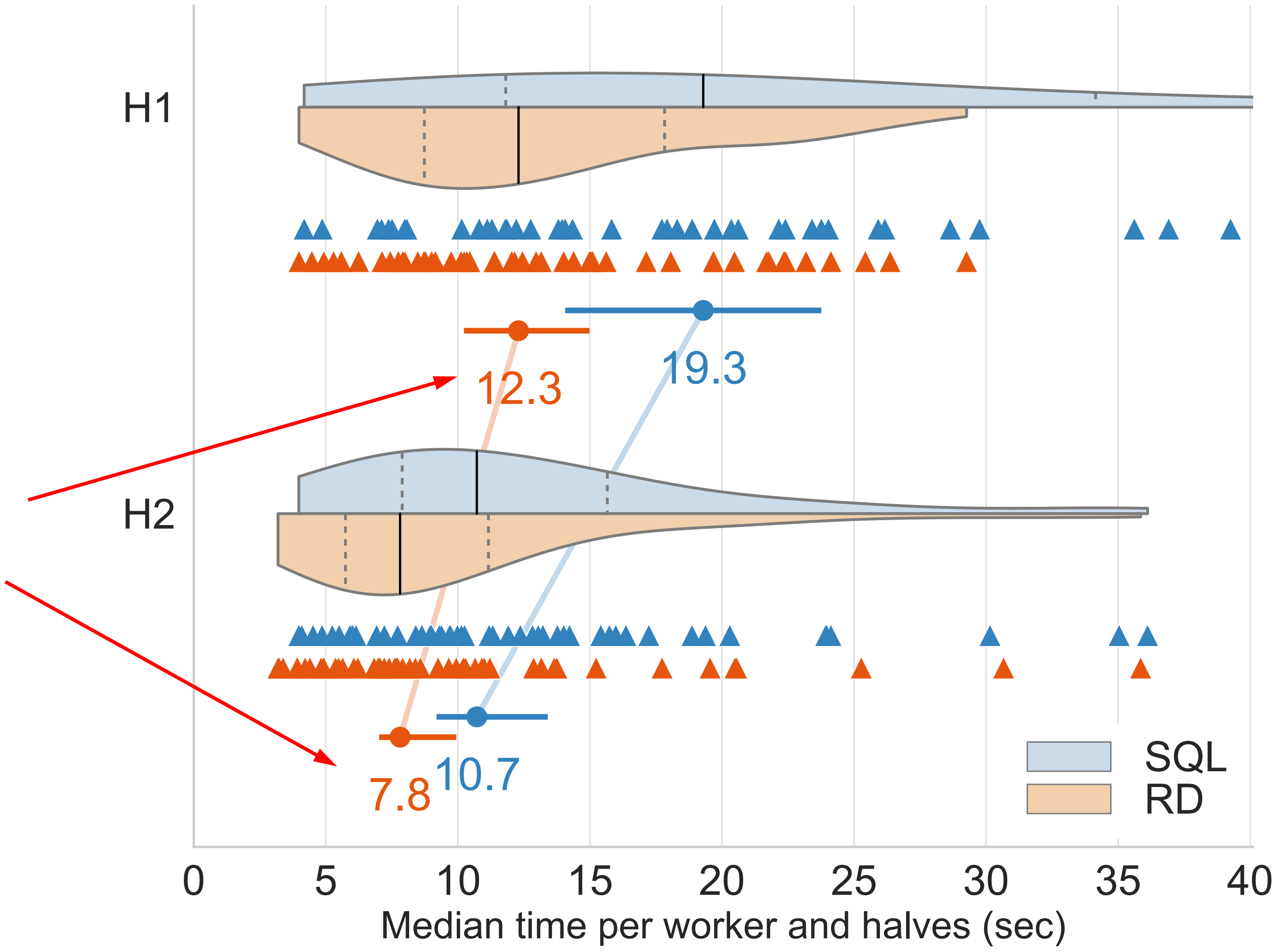
---

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

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

6. Principled User Studies (preregistered, beyond students)

A recent user study that measured the learning by participants to interpret Relational Diagrams (counterbalanced within-subjects study with randomization)

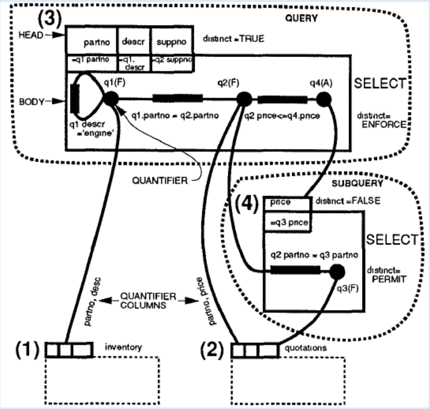


Comparing the time to understand queries for the first half vs the second half of questions

Gatterbauer, Dunne. On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages. SIGMOD 2024. https://dl.acm.org/doi/pdf/10.1145/3639316

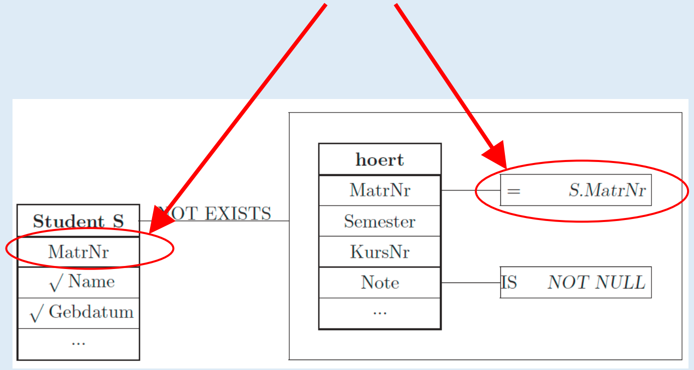
7. Revisiting P7: syntactic invariance is maybe too strong

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



Query Graph Model

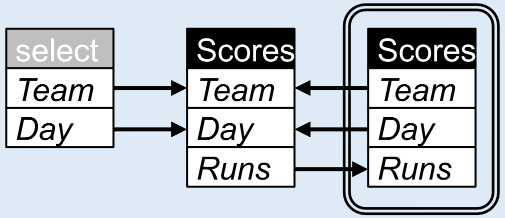
Pirahesh et al. [Sigmod'92]



Correlated nesting is preserved

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

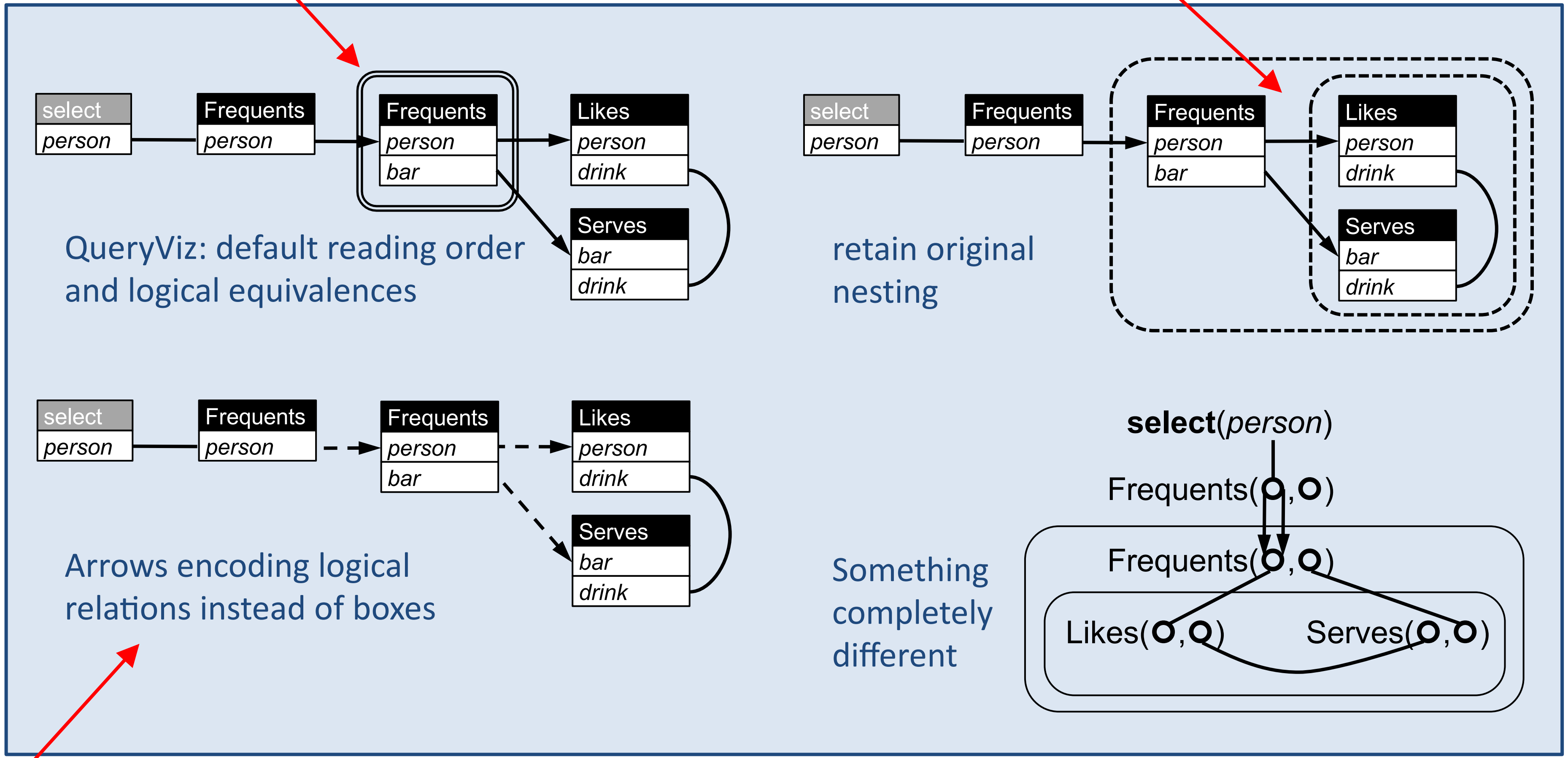
Source: Gatterbauer. Databases will visualize queries too. VLDB 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf

Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

8. What are other "better" visual metaphors?

Separate boxes was the choice of Queryvis

Nested boxes were considered but deemed visually too complex. Relational Diagrams made this choice.



QueryViz: default reading order and logical equivalences

retain original nesting

Arrows encoding logical relations instead of boxes

Something completely different

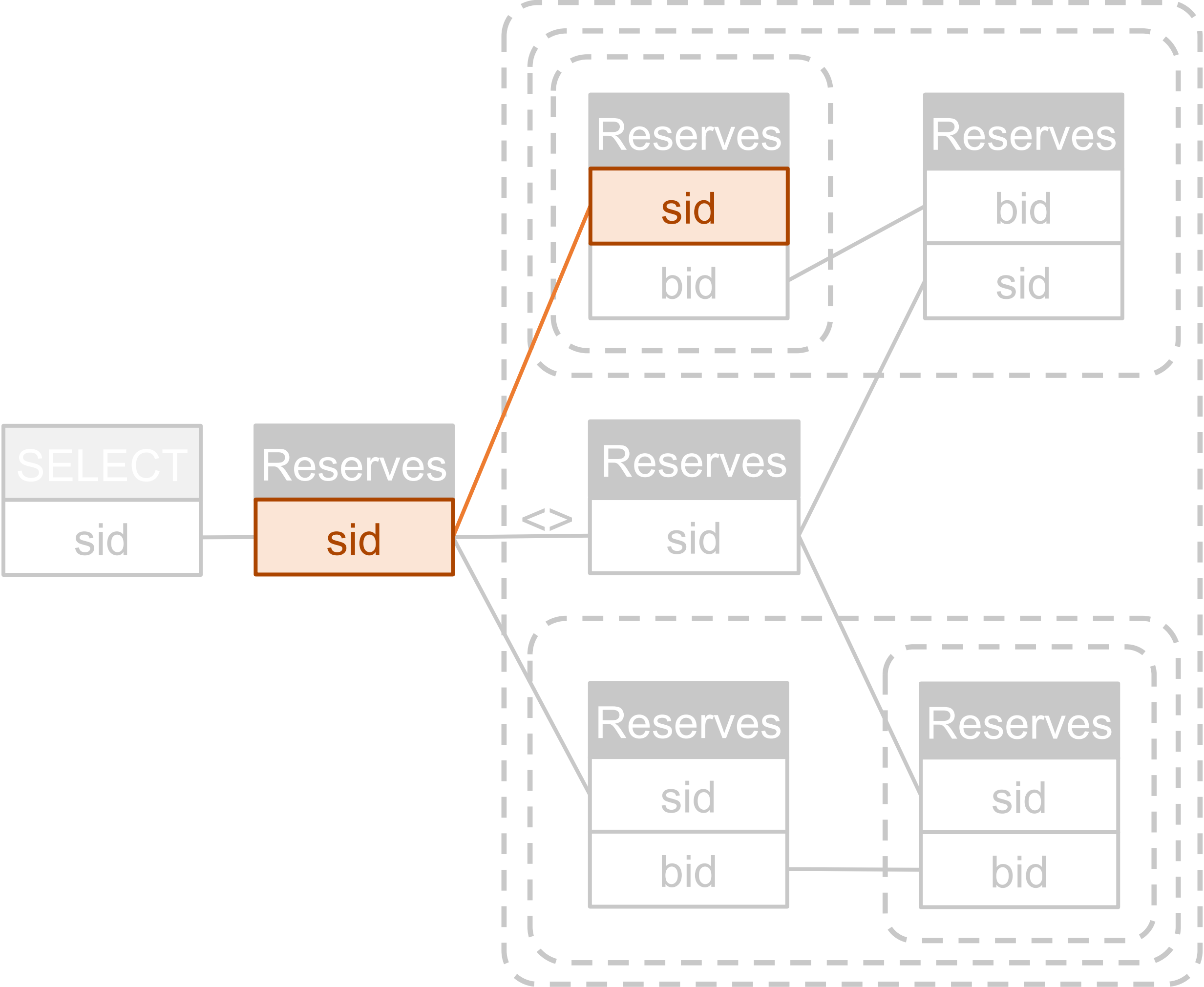
?

Likely not possible

Source: Gatterbauer. Databases will visualize queries too. VLDB 2011. https://gatterbauer.name/download/vldb2011_Database_Query_Visualization_presentation.pdf
Wolfgang Gatterbauer. A Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/

9. What about interaction with diagrams?

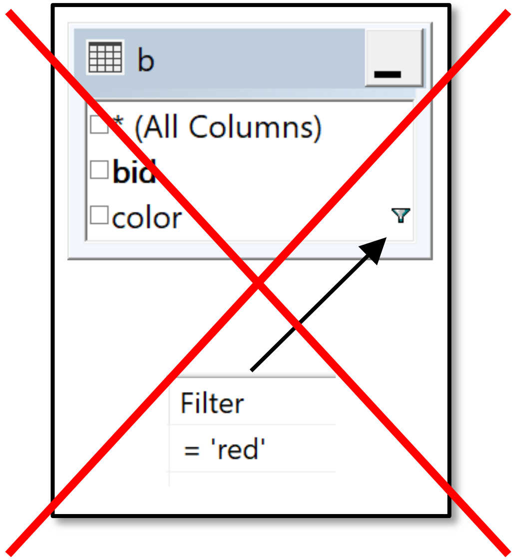
```
select distinct R1.sid
from Reserves R1
where not exists
  (select *
  from Reserves R2
  where R1.sid <> R2.sid
  and not exists
    (select *
    from Reserves R3
    where R3.sid = R2.sid
    and not exists
      (select *
      from Reserves R4
      where R4.sid = R1.sid
      and R4.bid = R3.bid))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



b

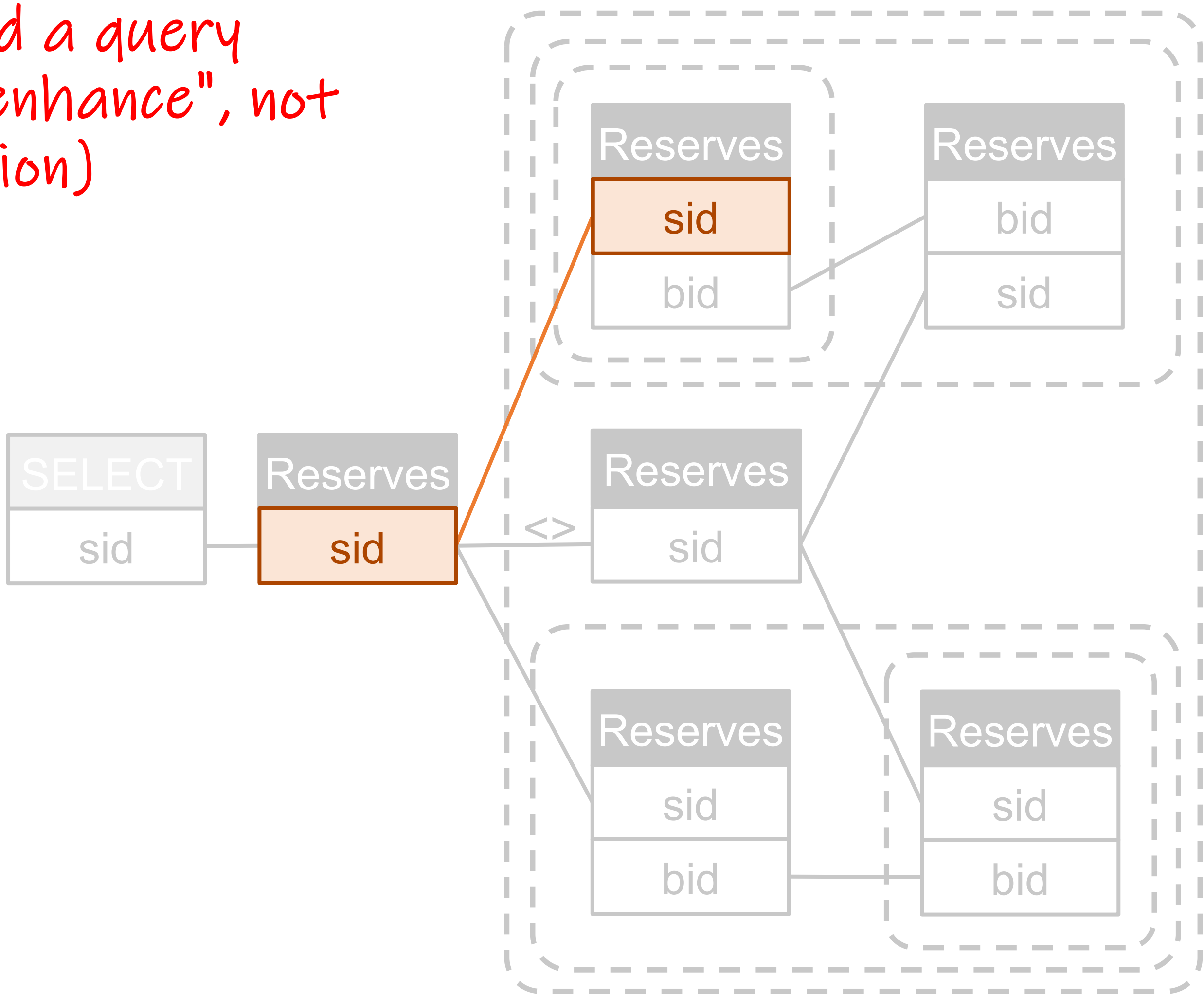
☐ * (All Columns)

☐ **bid**

☐ color

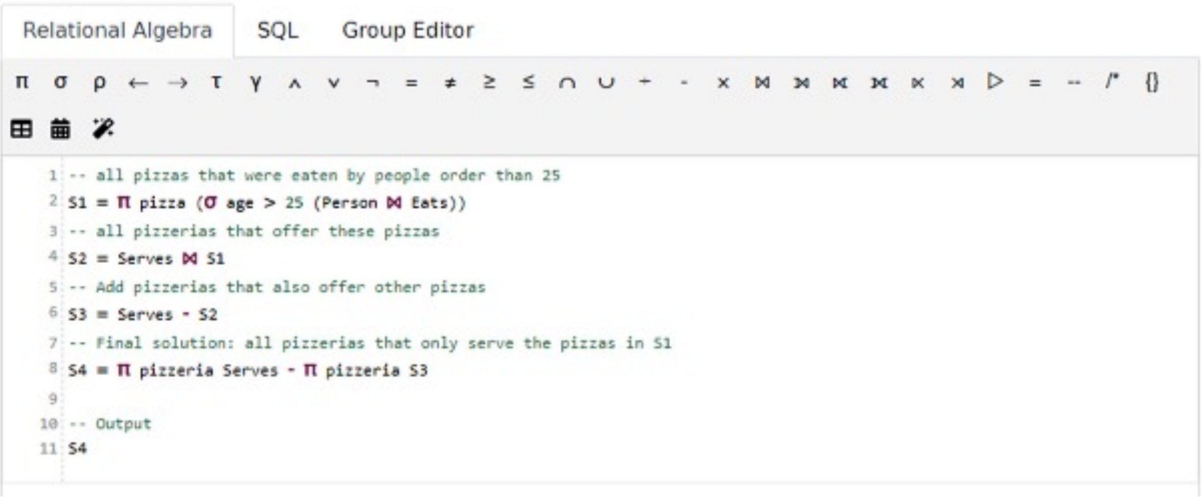
Filter

= 'red'



10. An ecosystem of tools that can translate between languages

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

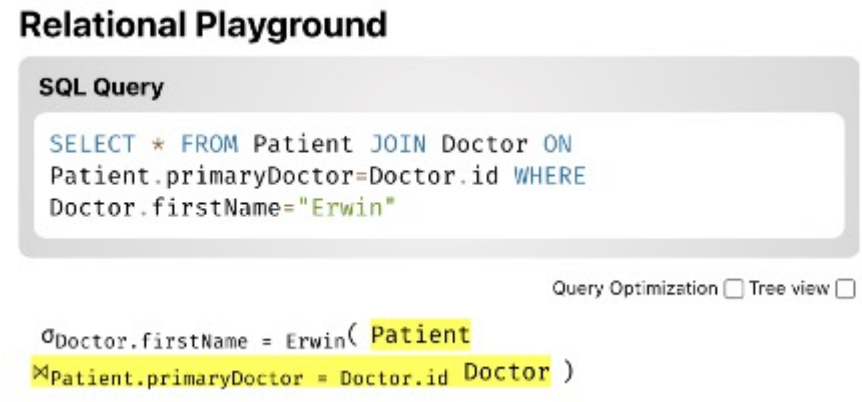


```
-- all pizzas that were eaten by people order than 25
S1 = π pizza (σ age > 25 (Person ⋈ Eats))
-- all pizzerias that offer these pizzas
S2 = Serves ⋈ S1
-- Add pizzerias that also offer other pizzas
S3 = Serves - S2
-- Final solution: all pizzerias that only serve the pizzas in S1
S4 = π pizzeria Serves - π pizzeria S3

-- Output
S4
```

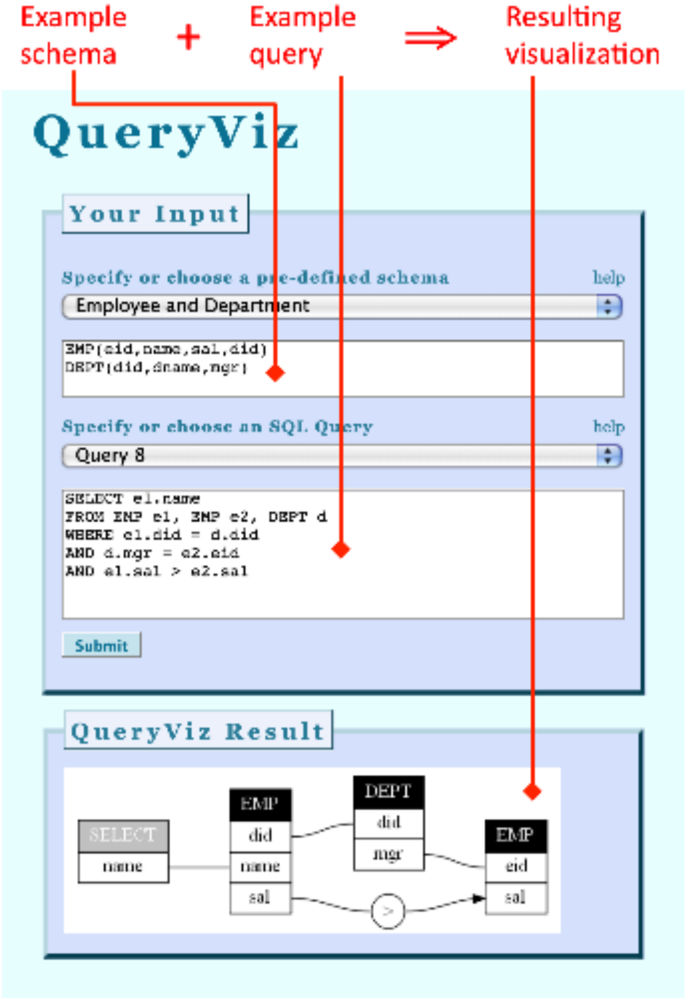
Online tools for translating b/w various query languages

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



```
SELECT * FROM Patient JOIN Doctor ON
Patient.primaryDoctor=Doctor.id WHERE
Doctor.firstName="Erwin"
```

$\sigma_{\text{Doctor.firstName = Erwin}}($ Patient $\bowtie_{\text{Patient.primaryDoctor = Doctor.id}}$ Doctor $)$



Employee and Department

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

Query 8

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

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 Comprehensive Tutorial on (...) Diagrammatic Representations (...), ICDE 2024. https://northeastern-datalab.github.io/diagrammatic-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
- Several (unresolved) issues lie in the actual details
- Solving those need rigorous and principled approaches

Thanks for your attention and please let me know about any mistakes you find. I must have missed some ☺