CompSci 516
# Database Systems

Lecture 4
Relational Algebra
and
Relational Calculus

Instructor: Sudeepa Roy



1

## Announcements

- In-person classes starting Thursday
  - Also live streaming and recording



2

## Today's topics

- Relational Algebra (RA) and Relational Calculus (RC)
- Reading material
  - [RG] Chapter 4 (RA, RC)
  - [GUW] Chapters 2.4, 5.1, 5.2

Acknowledgement:
The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.



3

## Relational Query Languages



4

## Relational Query Languages

- Query languages: Allow manipulation and retrieval of data from a database
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization
- Query Languages != programming languages
  - QLs not intended to be used for complex calculations
  - QLs support easy, efficient access to large data sets



5

## Formal Relational Query Languages

- Two "mathematical" Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
  - Relational Algebra: More operational, very useful for representing execution plans
  - Relational Calculus: Lets users describe what they want, rather than how to compute it (Non-operational, declarative, or procedural)
- Note: Declarative (RC, SQL) vs. Operational (RA)



6

# Preliminaries (recap)

- A query is applied to relation instances, and the result of a query is also a relation instance.
  - Schemas of input relations for a query are fixed
    - query will run regardless of instance
  - The schema for the result of a given query is also fixed
    - Determined by definition of query language constructs
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions, named-field notation more readable



7

# Example Schema and Instances

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

*S1*

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

*S2*

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

*R1*

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

8

# Logic Notations

- ∃ There exists
- ∀ For all
- ∧ Logical AND
- ∨ Logical OR
- ¬ NOT
- ⇒ Implies

9

# Relational Algebra (RA)

10

# Relational Algebra

- Takes one or more relations as input, and produces a relation as output
  - operator
  - operand
  - semantic
  - so an algebra!
- Since each operation returns a relation, operations can be composed
  - Algebra is "closed"



11

# Relational Algebra

- Basic operations:
  - Selection (σ) Selects a subset of rows from relation
  - Projection (π) Deletes unwanted columns from relation.
  - Cross-product (x) Allows us to combine two relations.
  - Set-difference (-) Tuples in reln. 1, but not in reln. 2.
  - Union (∪) Tuples in reln. 1 or in reln. 2.
- Additional operations:
  - Intersection (∩)
  - join ⋈
  - division(/)
  - renaming (ρ)
  - Not essential, but (very) useful.



12

## Projection

S2

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

- Deletes attributes that are not in projection list.
- Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate duplicates (Why)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it (performance)

| sname | rating |
|---|---|
| yuppy | 9 |
| lubber | 8 |
| guppy | 5 |
| rusty | 10 |

$\pi_{sname,rating}(S2)$

| age |
|---|
| 35.0 |
| 55.5 |

$\pi_{age}(S2)$

Duke CS, Fall 2016 CompSci 516: Data Intensive Computing Systems 10

13

## Selection

S2

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

- Selects rows that satisfy selection condition
- No duplicates in result. Why?
- Schema of result identical to schema of (only) input relation

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$\sigma_{rating>8}(S2)$

Duke CS, Fall 2016 CompSci 516: Data Intensive Computing Systems 11

14

## Composition of Operators

- Result relation can be the input for another relational algebra operation
  - Operator composition

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$\sigma_{rating>8}(S2)$

| sname | rating |
|---|---|
| yuppy | 9 |
| rusty | 10 |

$\pi_{sname,rating}(\sigma_{rating>8}(S2))$

15

## Union, Intersection, Set-Difference

S1

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

S2

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

- All of these operations take two input relations, which must be union-compatible:
  - Same number of fields.
  - 'Corresponding' fields have the same type
  - same schema as the inputs

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |
| 44 | guppy | 5 | 35.0 |
| 28 | yuppy | 9 | 35.0 |

$S1 \cup S2$

Duke CS, Spring 2016 CompSci 516: Data Intensive Computing Systems 12

16

## Union, Intersection, Set-Difference

S1

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

S2

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

- Note: no duplicate
  - "Set semantic"
  - SQL: UNION
  - SQL allows "bag semantic" as well: UNION ALL

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |
| 44 | guppy | 5 | 35.0 |
| 28 | yuppy | 9 | 35.0 |

$S1 \cup S2$

Duke CS, Spring 2016 CompSci 516: Data Intensive Computing Systems 12

17

## Union, Intersection, Set-Difference

S1

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

S2

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$S1 - S2$

$S1 \cap S2$

Duke CS, Spring 2016 CompSci 516: Data Intensive Computing Systems 13

18

## Cross-Product

- Each row of S1 is paired with each row of R.
- Result schema has one field per field of S1 and R, with field names `inherited' if possible.
  - Conflict: Both S1 and R have a field called sid.

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 22 | 101 | 10/10/96 |
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |



19

## Renaming Operator ρ

$(\rho_{sid \to sid1} S1) \times (\rho_{sid \to sid1} R1)$

or

$\rho(C(1 \to sid1, 5 \to sid2), S1 \times R1)$

C is the new relation name

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 22 | 101 | 10/10/96 |
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

- In general, can use ρ(<Temp>, <RA-expression>)



20

## Joins

$$R \bowtie_c S = \sigma_c (R \times S)$$

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

- Result schema same as that of cross-product.
- Fewer tuples than cross-product, might be able to compute more efficiently



21

## Find names of sailors who've reserved boat #103

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)



22

## Find names of sailors who've reserved boat #103

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$
- Solution 2: $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

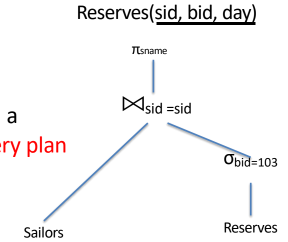


23

## Expressing an RA expression as a Tree

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

Also called a logical query plan




$\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$



24

## Find sailors who've reserved a red or a green boat

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

Use of rename operation

- Can identify all red or green boats, then find sailors who've reserved one of these boats:

Can also define Tempboats using union
Try the "AND" version yourself



25

## What about aggregates?

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Extended relational algebra
- $\gamma_{age,\ avg(rating) \rightarrow avgr}$ Sailors
- Also extended to "bag semantic": allow duplicates
  - Take into account cardinality
  - R and S have tuple t resp. m and n times
  - R ∪ S has t m+n times
  - R ∩ S has t min(m, n) times
  - R – S has t max(0, m-n) times
  - sorting(τ), duplicate removal (δ) operators



26

## Relational Calculus (RC)



27

## Relational Calculus

- RA is procedural
  - $\pi_A(\sigma_{A=a}\ R)$ and $\sigma_{A=a}\ (\pi_A\ R)$ are equivalent but different expressions
- RC
  - non-procedural and declarative
  - describes a set of answers without being explicit about how they should be computed
- TRC (tuple relational calculus)
  - variables take tuples as values
  - we will primarily do TRC
- DRC (domain relational calculus)
  - variables range over field values



28

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the name and age of all sailors with a rating above 7

∃ There exists

{P | ∃ S ϵ Sailors (S.rating > 7 ⋀ P.sname = S.sname ⋀ P.age = S.age)}

- P is a tuple variable
  - with exactly two fields sname and age (schema of the output relation)
  - P.sname = S.sname ⋀ P.age = S.age gives values to the fields of an answer tuple
- Use parentheses, ∀ ∃ ⋁ ⋀ > < = ≠ ¬ etc as necessary
- A ⇒ B is very useful too
  - next slide



29

## A ⇒ B

- A "implies" B
- Equivalently, if A is true, B must be true
- Equivalently, ¬ A ⋁ B, i.e.
  - either A is false (then B can be anything)
  - otherwise (i.e. A is true) B must be true



30

## Useful Logical Equivalences

| | |
|---|---|
| ∃ | There exists |
| ∀ | For all |
| ∧ | Logical AND |
| ∨ | Logical OR |
| ¬ | NOT |

- $\forall x\ P(x) \;=\; \neg\exists x\ [\neg P(x)]$
- $\neg(P \vee Q) \;=\; \neg P \wedge \neg Q$
- $\neg(P \wedge Q) \;=\; \neg P \vee \neg Q$

  } de Morgan's laws
  - Similarly, $\neg(\neg P \vee Q) = P \wedge \neg Q$ etc.
- $A \Rightarrow B \;=\; \neg A \vee B$

Duke CS, Spring 2022 CompSci 516: Database Systems 31

31

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the names of sailors who have reserved at least two boats

Duke CS, Spring 2022 CompSci 516: Database Systems 32

32

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the names of sailors who have reserved at least two boats

{P | ∃ S ϵ Sailors (∃ R1 ϵ Reserves ∃ R2 ϵ Reserves (S.sid = R1.sid ∧ S.sid = R2.sid ∧ R1.bid ≠ R2.bid) ∧ P.sname = S.sname)}

Duke CS, Spring 2022 CompSci 516: Database Systems 33

33

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the names of sailors who have reserved all boats
- Called the "Division" operation in RA

Duke CS, Spring 2022 CompSci 516: Database Systems 34

34

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the names of sailors who have reserved all boats

{P | ∃ S ϵ Sailors [∀B ϵ Boats (∃ R ϵ Reserves (S.sid = R.sid ∧ R.bid = B.bid))] ∧ (P.sname = S.sname)}

Duke CS, Spring 2022 CompSci 516: Database Systems 35

35

## TRC: example

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

- Find the names of sailors who have reserved all red boats

How will you change the previous TRC expression?

Duke CS, Spring 2022 CompSci 516: Database Systems 36

36

# TRC: example

Sailors(<u>sid</u>, sname, rating, age)
Boats(<u>bid</u>, bname, color)
Reserves(<u>sid, bid, day</u>)

- Find the names of sailors who have reserved all red boats

Recall that A ⇒ B is logically equivalent to ¬ A ⋁ B
so ⇒ can be avoided, but it is cleaner and more intuitive

Duke CS, Spring 2022 CompSci 516: Database Systems 37

37

# DRC: example

Sailors(<u>sid</u>, sname, rating, age)
Boats(<u>bid</u>, bname, color)
Reserves(<u>sid, bid, day</u>)

- Find the name and age of all sailors with a rating above 7

TRC:
{P | ∃ S ϵ Sailors (S.rating > 7 ⋀ P.name = S.name ⋀ P.age = S.age)}

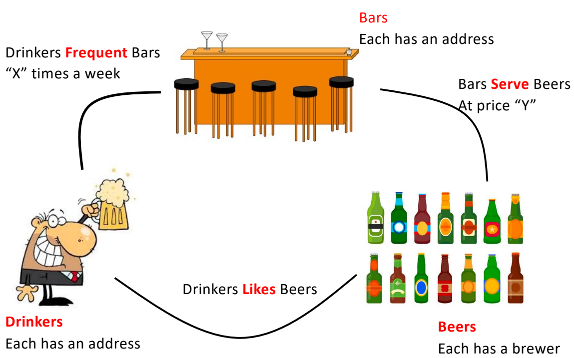
DRC:
{<N, A> | ∃ <I, N, T, A> ϵ Sailors ⋀ T > 7}

- Variables are now domain variables
- We will use use TRC
  - both are equivalent
- Another option to write coming soon!

Duke CS, Spring 2022 CompSci 516: Database Systems 38

38

# The famous "Beers" database




39

39

# "Beers" as a Relational Database

See online database for more tuples

Bar

| name | address |
|---|---|
| The Edge | 108 Morris Street |
| Satisfaction | 905 W. Main Street |

Serves

| bar | beer | price |
|---|---|---|
| The Edge | Budweiser | 2.50 |
| The Edge | Corona | 3.00 |
| Satisfaction | Budweiser | 2.25 |

Beer

| Name | brewer |
|---|---|
| Budweiser | Anheuser-Busch Inc. |
| Corona | Grupo Modelo |
| Dixie | Dixie Brewing |

Frequents

| drinker | bar | times_a_week |
|---|---|---|
| Ben | Satisfaction | 2 |
| Dan | The Edge | 1 |
| Dan | Satisfaction | 2 |

Drinker

| name | address |
|---|---|
| Amy | 100 W. Main Street |
| Ben | 101 W. Main Street |
| Dan | 300 N. Duke Street |

Likes

| drinker | beer |
|---|---|
| Amy | Corona |
| Dan | Budweiser |
| Dan | Corona |
| Ben | Budweiser |

40

40

# More Examples: RC

UNDERSTAND THE DIFFERENCE IN ANSWERS
FOR ALL FOUR DRINKERS

Acknowledgement: examples and slides by Profs. Balazinska and Suciu, and the [GUW] book

Duke CS, Spring 2022 CompSci 516: Database Systems 41

41

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

# Drinker Category 1

Find drinkers that frequent some bar that serves some beer they like.

…

42

42

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 1

Find drinkers that frequent some bar that serves some beer they like.

43

43

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 2

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serves some beer they like.

…

Free HW question hint!

44

44

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 2

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serve some beer they like.

45

45

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 3

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serve some beer they like.

Find drinkers that frequent some bar that serves only beers they like.

…

46

46

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 3

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serve some beer they like.

Find drinkers that frequent some bar that serves only beers they like.

47

47

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 4

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serve some beer they like.

Find drinkers that frequent some bar that serves only beers they like.

Find drinkers that frequent only bars that serve only beer they like.

…

48

48

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## Drinker Category 4

Find drinkers that frequent some bar that serves some beer they like.

Find drinkers that frequent only bars that serve some beer they like.

Find drinkers that frequent some bar that serves only beers they like.

Find drinkers that frequent only bars that serve only beer they like.

49

49

## Why should we care about RC

- RC is declarative, like SQL, and unlike RA (which is operational)
- Gives foundation of database queries in first-order logic
  - you cannot express all aggregates in RC, e.g. cardinality of a relation or sum (possible in extended RA and SQL)
  - still can express conditions like "at least two tuples" (or any constant)
- RC expression may be much simpler than SQL queries
  - and easier to check for correctness than SQL
  - power to use ∀ and ⇒
  - then you can systematically go to a "correct" SQL or RA query

Duke CS, Fall 2019 CompSci 516: Database Systems 50

50

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

Drinker category 5!

## From RC to SQL

Query: Find drinkers that like some beer (so much) that they frequent all bars that serve it

{x | ∃ L ϵ Likes (L.drinker = x.drinker) ∧ [ ∀ S ϵ Serves (L.beer = S.beer) ⇒ ∃ F ϵ Frequents [(F.drinker = L.drinker) ∧ (F.bar = S.bar)] ]}

Duke CS, Fall 2019 CompSci 516: Database Systems 51

51

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## From RC to SQL (or RA)

Query: Find drinkers that like some beer so much that they frequent all bars that serve it

{x | ∃ L ϵ Likes (L.drinker = x.drinker) ∧ [ ∀ S ϵ Serves [ (L.beer = S.beer) ⇒ ∃ F ϵ Frequents [(F.drinker = L.drinker) ∧ (F.bar = S.bar)] ] ]}

≡ {x | ∃ L ϵ Likes (L.drinker = x.drinker) ∧ [ ∀ S ϵ Serves [¬ (L.beer = S.beer) ∨ [∃ F ϵ Frequents [(F.drinker = L.drinker) ∧ (F.bar = S.bar)] ] ]}

Step 1: Replace ∀ with ∃ using de Morgan's Laws

∀x P(x) same as ¬∃x ¬P(x)

Q(x) = ∃y. Likes(x, y)∧ [ ¬∃ S ϵ Serves [(L.beer = S.beer) ∧ ¬ [∃ F ϵ Frequents [(F.drinker = L.drinker) ∧ (F.bar = S.bar)] ])

SQL or RA does not have ∀!
Now you got all ∃ and ¬ expressible in RA/SQL

¬(¬P∨Q) same as P∧ ¬ Q

Duke CS, Fall 2019 CompSci 516: Database Systems 52

52

Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

## From RC to SQL

Query: Find drinkers that like some beer so much that they frequent all bars that serve it

∃ L ϵ Likes ∧ ¬∃ S ϵ Serves [(L.beer = S.beer) ∧ ¬ [∃ F ϵ Frequents [(F.drinker = L.drinker) ∧ (F.bar = S.bar)])

Step 2: Translate into SQL

```
SELECT DISTINCT L.drinker
FROM Likes L
WHERE not exists
  (SELECT S.bar
   FROM Serves S
   WHERE L.beer=S.beer
        AND not exists (SELECT *
                        FROM Frequents F
                        WHERE F.drinker=L.drinker
                          AND F.bar=S.bar))
```

We will see a "methodical and correct" translation trough "safe queries" in Datalog

Duke CS, Fall 2019 CompSci 516: Database Systems 53

53

## Summary

- You learnt three query languages for the Relational DB model
  - SQL
  - RA
  - RC
- All have their own purposes
- You should be able to write a query in all three languages and convert from one to another
  - However, you have to be careful, not all "valid" expressions in one may be expressed in another
  - {S | ¬ (S ϵ Sailors)} – infinitely many tuples – an "unsafe" query
  - More when we do "Datalog", also see Ch. 4.4 in [RG]

Duke CS, Spring 2022 CompSci 516: Database Systems 54

54