CS150A Database

Lu Sun

School of Information Science and Technology

ShanghaiTech University

Mar. 27, 2024

Today:

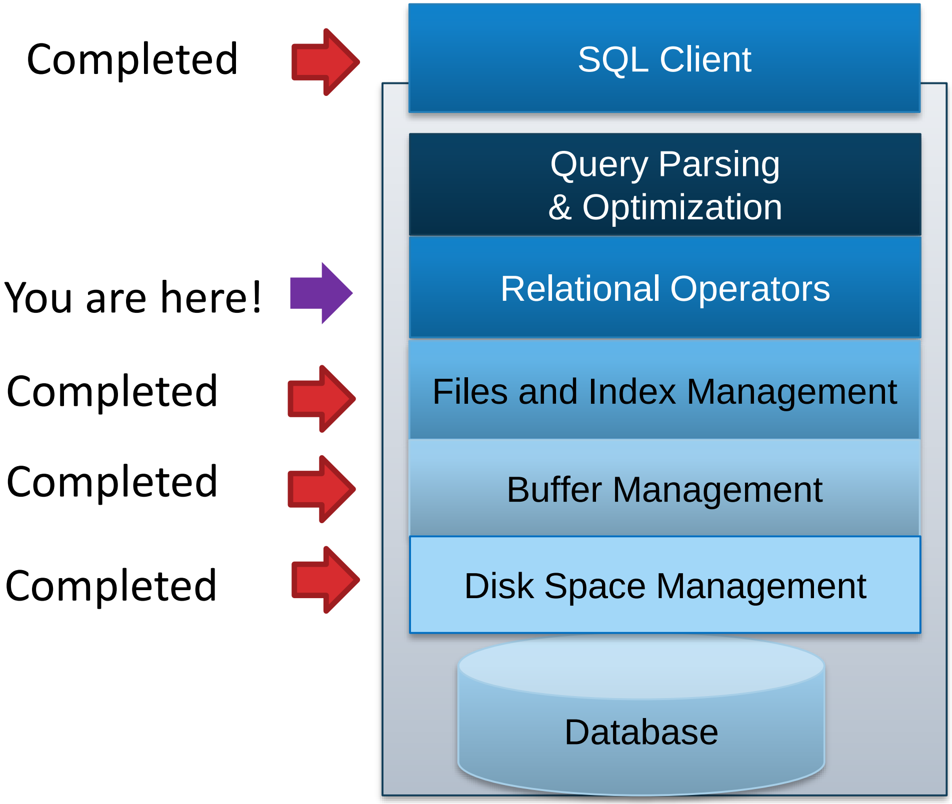
- Relational Algebra:
  - Basic Operators
  - Compound Operators

Readings:

- Database Management Systems (DBMS), Chapters 4.1-4.2

Acknowledgement: Joe Hellerstein@UCBerkeley's scourse notes

Architecture of a DBMS: What we've learned



Today: *definitions* of the relational operators.

Coming soon: *implementations*

An Overview of the Layer Above

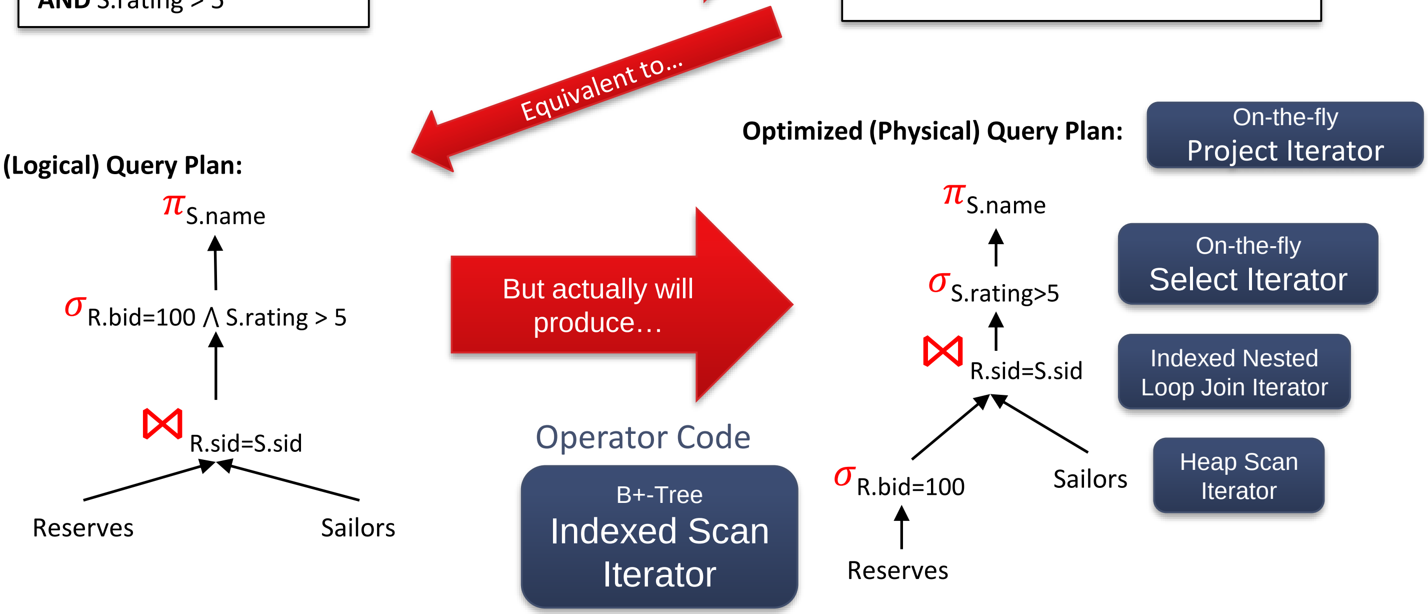
SQL Query

```
SELECT S.name
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100
AND S.rating > 5
```

Query Parser & Optimizer

Relational Algebra

$$\pi_{S.name}(\sigma_{bid=100 \wedge rating>5}(\text{Reserves} \bowtie_{R.sid=S.sid} \text{Sailors}))$$



SQL vs Relational Algebra

SQL Query

SELECT S.name
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100
AND S.rating > 5

Query Parser & Optimizer

Relational Algebra

$\pi_{S.name}(\sigma_{bid=100 \wedge rating>5}($

$Reserves \bowtie_{R.sid=S.sid} Sailors))$

Equivalent to...

SQL

A **declarative** expression of the query result

Relational Algebra

Operational description of a computation.

$\bowtie_{R.sid=S.sid}$

Reserves Sailors

Operator Code

Systems execute relational algebra query plan.

SQL (Structured Query Language)

SELECT S.name
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100
AND S.rating > 5

- Key **System** Features: *Why do we like SQL*
  - Declarative:
    - Say ***what*** you want, not ***how*** to get it
    - Enables system to optimize the ***how***
- Foundation in formal Query Languages
  - **Relational Calculus**

History: Formal Relational QL's

- **Relational Calculus:** (Basis for SQL)
  - Describe the result of computation
  - Based on first order logic
  - Tuple Relational Calculus **(TRC)**
    - $\{S \mid S \in Sailors\ \exists R \in Reserves\ (R.sid = S.sid \wedge R.bid = 103)\}$

Are these equivalent?

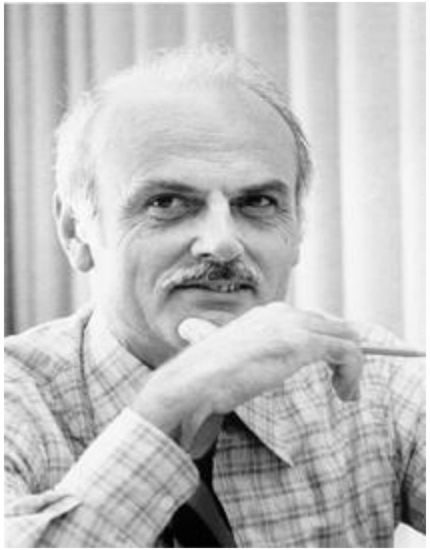
Can we go from one to the other?

- **Relational Algebra:**
  - Algebra on sets
  - Operational description of transformations

Codd’s Theorem

- Established equivalence in expressivity between :
  - Relational Calculus
  - Relational Algebra

- Why an important result?
  - Connects declarative representation of queries with operational description
  - Constructive: we can compile SQL into relational algebra



Edgar F. “Ted” Codd
(1923 - 2003)
Turing Award 1981

Relational Algebra Preliminaries

- Algebra of operators on relation instances
- $\pi_{S.name}(\sigma_{R.bid=100 \wedge S.rating>5}(R \bowtie_{R.sid=S.sid} S))$
  - Closed: result is also a relation instance
    - Enables rich composition!
  - Typed: input schema determines output
    - Can statically check whether queries are legal.

Relational Algebra and Sets

- Pure relational algebra has set semantics
  - No duplicate tuples in a relation instance
  - vs. SQL, which has multiset (bag) semantics
  - We will switch to multiset in the system discussion

Relational Algebra Operators: Unary

- Unary Operators: on **single relation**
- **Projection** (π): Retains only desired columns (vertical)
- **Selection** (σ): Selects a subset of rows (horizontal)
- **Renaming** (ρ): Rename attributes and relations.

Relational Algebra Operators: Binary

- Binary Operators: on **pairs of relations**
- **Union** ($\cup$): Tuples in r1 or in r2.
- **Set-difference** (—): Tuples in r1, but not in r2.
- **Cross-product** ($\times$): Allows us to combine two relations.

Relational Algebra Operators: Compound

- <u>Compound Operators</u>: common "*macros*" for the above
- **Intersection** ($\cap$): Tuples in r1 and in r2.
- **Joins** ($\bowtie_\theta$, $\bowtie$): Combine relations that satisfy predicates

Projection (π)

- Corresponds to the SELECT list in SQL
- Schema determined by schema of attribute list
  - Names and types correspond to input attributes
- *Selects a subset of columns (vertical)*

$$\pi_{\underbrace{\text{sname,age}}_{\text{List of Attributes}}}(S2)$$

Relational *Instance* **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

sname	age
yuppy	35.0
lubber	55.5
guppy	35.0
rusty	35.0

Projection (π), cont.

- Set semantics → results in fewer rows
  - Real systems don't automatically remove duplicates
    - Why? (Semantics and Performance reasons)
  - *Selects a subset of columns (vertical)*

$$\pi_{age}(S2)$$

Relational *Instance* **S2**

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Multiset

age
35.0
55.5
35.0
35.0

Set

age
35.0
55.5

Selection(σ)

- Corresponds to the WHERE clause in SQL
- Output schema same as input
- Duplicate Elimination? Not needed.
- *Selects a subset of rows (horizontal)*

$$\sigma_{\underbrace{\text{rating}>8}_{\text{Selection Condition (Boolean Expression)}}}(S2)$$

Relational *Instance* **S2**

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Composing Select and Project

- Names of sailors with rating > 8: $\pi_{sname}(\sigma_{rating>8}(S2))$

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

$\sigma_{rating>8}$

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

π_{sname}

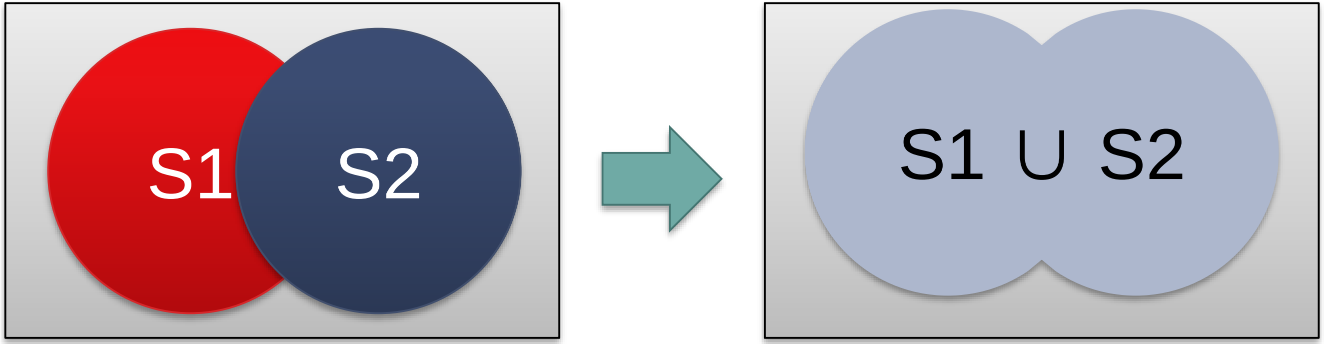
sname
yuppy
rusty

- What about: $\sigma_{rating>8}(\pi_{sname}(S2))$
  - *Invalid types. Input to $\sigma_{rating>8}$ does not contain rating.*

Union (∪)

- Two input relations, must be *compatible*:
  - Same number of fields
  - Fields in corresponding positions have same **type**
- SQL Expression: UNION

S1 ∪ S2



Union (∪) VS Union ALL

- Duplicate elimination in practice?
  - SQL's UNION vs UNION ALL

Relational *Instance* **S1**

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

Relational *Instance* **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

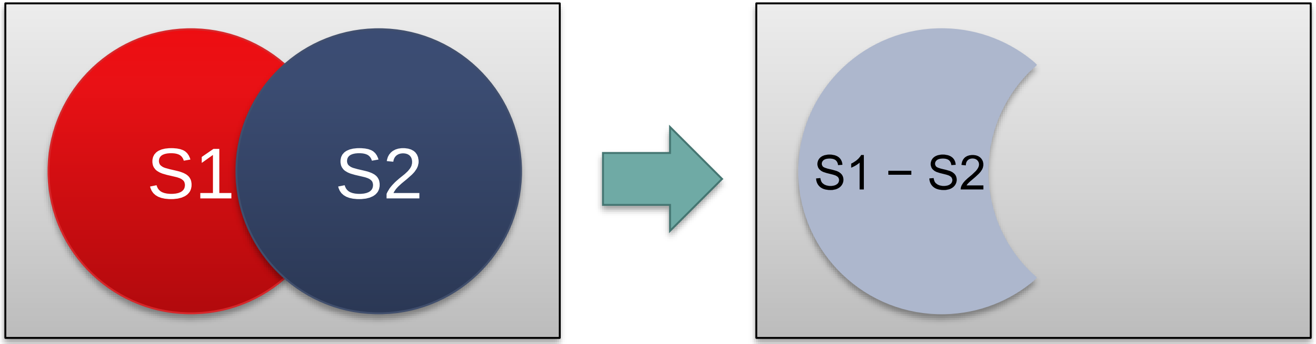
S1 ∪ S2

sid	sname	rating	age
22	dustin	7	45
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Set Difference (−)

- Same as with union, both input relations must be *compatible*.
- SQL Expression: EXCEPT

S1 – S2



Set Difference (−), cont.

- Duplicate elimination?
  - Not required
- EXCEPT vs EXCEPT ALL

Relational *Instance* **S1**

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

Relational *Instance* **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

sid	sname	rating	age
22	dustin	7	45

S2 – S1

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

Cross-Product (×)

- **R1 × S1:** *Each row of **R1** paired with each row of **S1***

R1:

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

×

S1:

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

=

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

- ***How many rows in result? |R1|*|R2|***
- ***Schema compatability? Not needed.***
- ***Duplicates? None generated.***

Renaming (ρ = "rho")

- *Renames relations and their attributes:*
- Note that relational algebra doesn't require names.
  - We could just use positional arguments.

$$\rho(\underbrace{\text{Temp1}}_{\text{Output Relation Name}}(\underbrace{1 \rightarrow \text{sid1}, 4 \rightarrow \text{sid2}}_{\text{Renaming List: position} \rightarrow \text{New Name}}), \underbrace{R1 \times S1}_{\text{Input Relation}})$$

R1 × S1

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

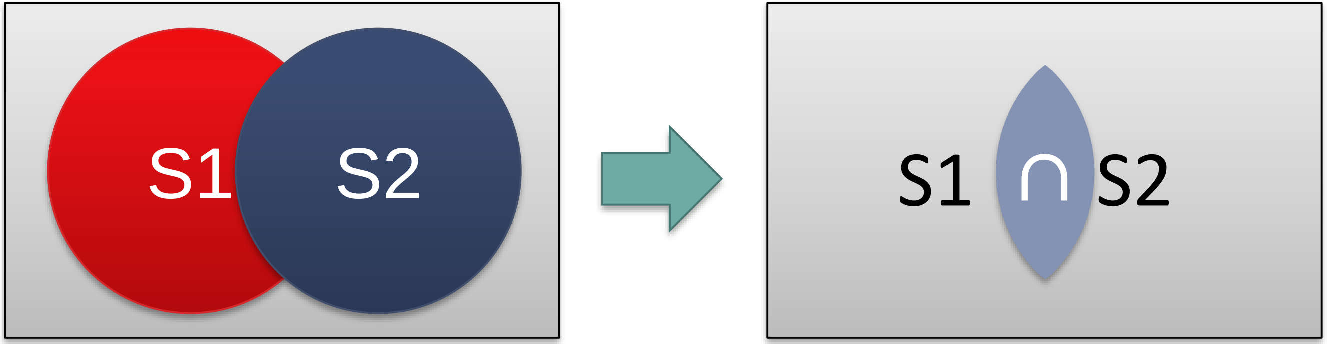
Temp1

sid1	bid	day	sid2	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

Compound Operator: Intersection

- Same as with union, both input relations must be *compatible*.
- SQL Expression: INTERSECT

S1 ∩ S2



Intersection (∩)

- Equivalent to:
  S1 — (S1 — S2)

S1 ∩ S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **S1**

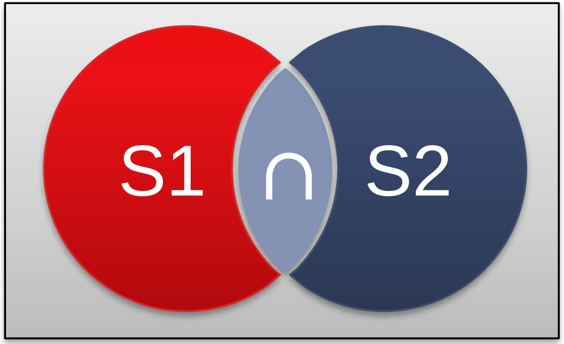
sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **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

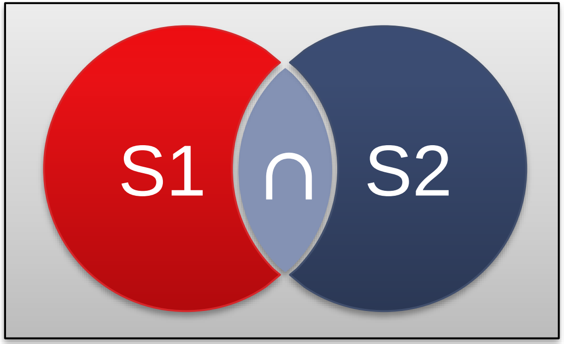
Intersection (∩), Pt 2

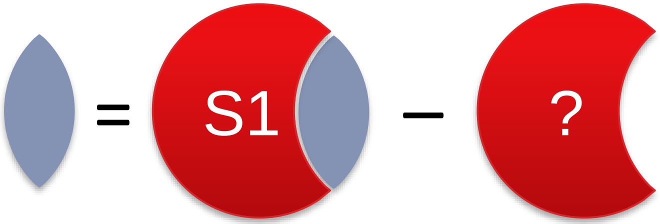
- S1 ∩ S2 = ?



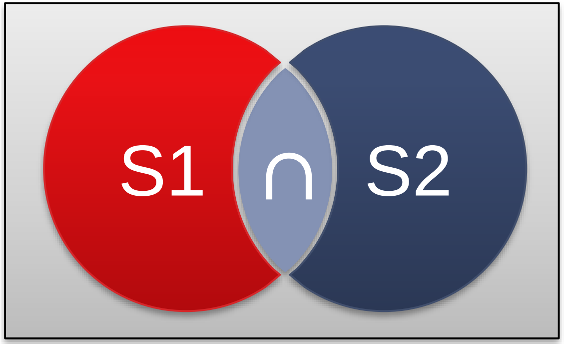
Intersection (∩), Pt 3

- $S1 \cap S2 = S1 - ?$

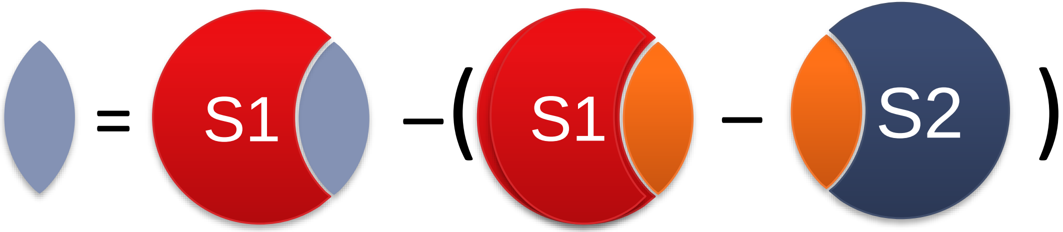




Intersection (∩), Pt 4



- $S1 \cap S2 = S1 - (S1 - S2)$



- Is intersection monotonic?

$R_1 \subseteq R_2 \quad \Rightarrow \quad S \cap R_1 \subseteq S \cap R_2$

Compound Operator: Join

- Joins are compound operators (like intersection):
  - Generally, σ_{θ}(R × S)
- Hierarchy of common kinds:
  - **Theta Join** ($\bowtie_{\theta}$): join on logical expression θ
    - **Equi-Join:** theta join with theta being a conjunction of equalities
      - **Natural Join** ($\bowtie$): equi-join on all matching column names

Note: we will need to learn a good join algorithm.

Avoid cross-product if we can!!

Theta Join ($\bowtie_\theta$) Example

- **R1** $\bowtie_{\text{sid=sid}}$ **S1**

R1:

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

$\bowtie_{\text{sid=sid}}$

S1:

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

=

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
58	103	11/12/96	58	rusty	10	35.0

- Note that output needs a rename operator!

Another Theta Join ($\bowtie_\theta$) Example

- **R $\bowtie_\theta$ S = σ_θ(R × S)**
- **Example:** *More senior sailors for each sailor.*
- S1 $\bowtie_{f4 < f8}$ S1

S1:

f1	f2	f3	f4
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S1			
f1	f2	f3	f4	f5	f6	f7	f8
22	dustin	7	45.0	22	dustin	7	45.0
22	dustin	7	45.0	31	lubber	8	55.5
22	dustin	7	45.0	58	rusty	10	35.0
31	lubber	8	55.5	22	dustin	7	45.0
31	lubber	8	55.5	31	lubber	8	55.5
31	lubber	8	55.5	58	rusty	10	35.0
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5
58	rusty	10	35.0	58	rusty	10	35.0

Another Theta Join ($\bowtie\theta$), Pt 2

- **R $\bowtie_\theta$ S = σ_θ(R × S)**
- **Example:** *More senior sailors for each sailor.*
- S1 $\bowtie_{\text{age} < \text{age2}}$ S1

S1:

f1	f2	f3	f4
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S1			
f1	f2	f3	f4	f5	f6	f7	f8
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
22	dustin	7	45.0	31	lubber	8	55.5
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~

Another Theta Join ($\bowtie_\theta$), Pt 3

- **R $\bowtie_\theta$ S = σ_θ(R × S)**
- **Example:** *More senior sailors for each sailor.*
- S1 $\bowtie_{f4 < f8}$ S1

S1:

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

S1				S1			
sid	sname	rating	age	sid	sname	rating	age2
22	dustin	7	45.0	31	lubber	8	55.5
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5

- Result *schema* same as that of cross-product.
- Special Case:
  - **Equi-Join:** *theta join with AND of = predicates*
    - *Special special case* ***Natural Join*** …

Natural Join (⋈)

- Special case of equi-join in which equalities are specified for all matching fields and duplicate fields are projected away

$$\mathbf{R} \bowtie \mathbf{S} = \pi_{\text{unique fld.}}\, \sigma_{\text{eq. matching fld.}} (\mathbf{R} \times \mathbf{S})$$

- Compute R × S
- Select rows where fields appearing in both relations have equal values
- Project onto the set of all unique fields.

Natural Join (⋈) Pt 2

- $R \bowtie S = \pi_{\text{unique fld.}}\ \sigma_{\text{eq. matching fld.}}(R \times S)$

R1 ⋈ S1

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

R1:

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

S1:

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

Natural Join (⋈), Pt 3

- $\mathbf{R \bowtie S} = \pi_{\text{unique fld.}}\ \sigma_{\text{eq. matching fld.}}(\mathbf{R \times S})$

R1 ⋈ S1

sid	bid	day	sname	rating	age
22	101	10/10/96	dustin	7	45.0
58	103	11/12/96	rusty	10	35.0

- Commonly used for foreign key joins (as above).

R1:

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

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

Extended Relational Algebra

- Group By / Aggregation Operator (γ):
  - $\gamma_{\text{age, AVG(rating)}}$(Sailors)
  - With selection (HAVING clause):
    - $\gamma_{\text{age, AVG(rating), COUNT(*)>2}}$(Sailors)

- Textbook uses two operators:
  - GROUP BY age, AVG(rating) (Sailors)
  - HAVING COUNT(*)>2
    (GROUP BY age, AVG(rating)(Sailors))

Summary

- Relational Algebra: a small set of operators mapping relations to relations
  - Operational, in the sense that you specify the explicit order of operations
  - A closed set of operators! Mix and match.
- Basic ops include: σ, π, $\times$, $\cup$, —
- Important compound ops: $\cap$, $\bowtie$