CSE 541: Database Systems I

Relational Algebra

Slides Credit: Jiannan Wang, Tianzheng Wang

So far…

- How are data laid out on disks (block devices)?
- How are data being accessed?
- How to make data access efficient?

- Record/Tuple level operations

- How to process more complicated queries (like in SQL)?

Relational Model

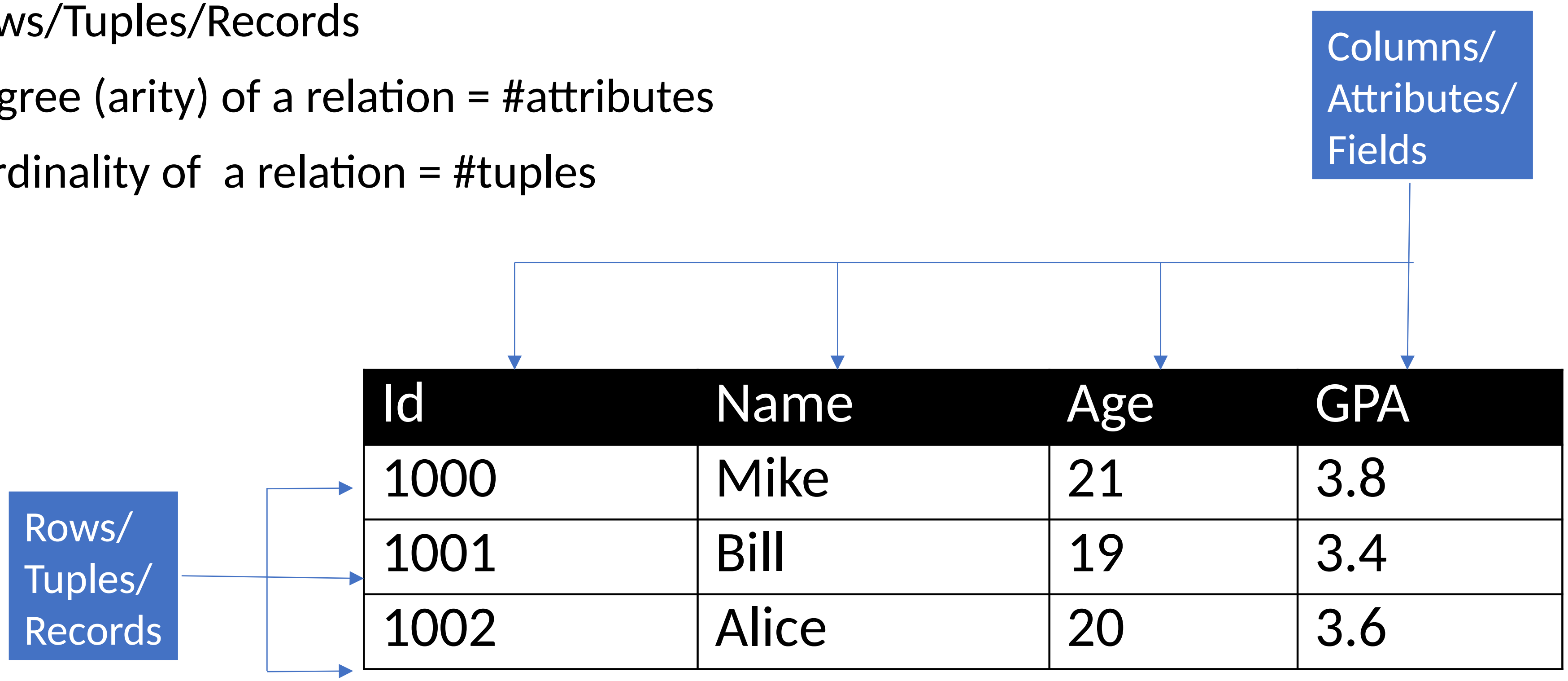
- What do we use in Math to represent a collection of data?

- Sets: {a, b, c, d, e}
    - Deterministic, Unique, Order-less

- Relation:
- Do not confuse it with Mapping (Function)

Terminology

- Relations/Tables
- Columns/Attributes/Fields
- Rows/Tuples/Records
- Degree (arity) of a relation = #attributes
- Cardinality of a relation = #tuples



Columns/
Attributes/
Fields

Id	Name	Age	GPA
1000	Mike	21	3.8
1001	Bill	19	3.4
1002	Alice	20	3.6

Rows/
Tuples/
Records

Formal Query Languages

- Relational Calculus
  - **Declarative** query language
  - Describe **what** you want, rather than *how* to compute it
  - Rooted from logic.
  - **TRC (Tuple Relational Calculus)** influences SQL.

- Relational Algebra
  - **Procedural** query language
  - Describe **how** to get the data your want step by step.
  - Used to represent execution plans.

Formal Query Languages

- Relational Calculus
  - **Declarative** query language
  - Describe **what** you want, rather than *how* to compute it
  - Rooted from logic.
  - **TRC (Tuple Relational Calculus)** influences SQL.

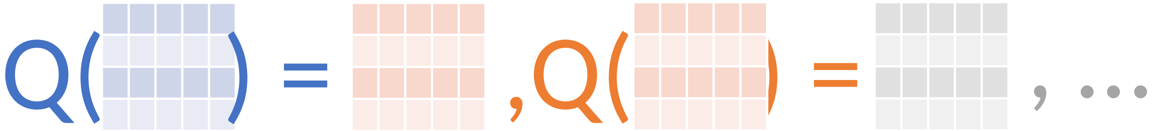
- Relational Algebra
  - **Procedural** query language
  - Describe **how** to get the data your want step by step.
  - Used to represent execution plans.

Query

- Query is a function over relations

$$Q\left(R_1,\ldots,R_n\right)=R\ result$$

- The schema of the result relation is determined by the input relation and the query
- Because the result of a query is a relation, it can be used as input to another query



Set vs. Bags

- Sets: {a, b, c}, {a, d, e, f}, {}, …
- Bags: {a, a, b, c}, {b, b, b, b, b}, …

- Relational Algebra has two flavors:
  - Set semantics = standard Relational Algebra
  - Bag semantics = extended Relational Algebra

- DB systems implement bag semantics (Why?)

Set vs. Bags

- Sets: {a, b, c}, {a, d, e, f}, {}, ...
- Bags: {a, a, b, c}, {b, b, b, b, b}, ...

- Relational Algebra has two flavors:
  - **Set semantics = standard Relational Algebra**
  - Bag semantics = extended Relational Algebra

- DB systems implement bag semantics (Why?)
- Even bag semantics: **implicit keys**.

Relation Algebra

- Recipe of query processing

- Core operators
  - Selection ()
  - Projection ()
  - Union ()
  - Set Difference ()
  - Cross product ()
- Additional operators
  - Rename ()
  - join (⋈)
  - Intersect ()

Selection

- The selection operator, σ (sigma), specifies the *rows* to be retained from the input relation
- A selection has the form: $\sigma_{condition}$(*relation*), where *condition* is a Boolean expression
  - Terms in the condition are comparisons between two fields (or a field and a constant)
  - Using one of the comparison operators: **<, ≤, =, ≠, , >**
  - Terms may be connected by **∧** (and), or **∨** (or),
  - Terms may be negated using ¬ (not)

Selection Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\sigma_{\mathit{birth} < 1981}$(Customer)

sin	firstName	lastName	birth
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

$\sigma_{\mathit{lastName} = \mathit{"Summers"}}$(Customer)

sin	firstName	lastName	birth
111	Buffy	Summers	1981
555	Dawn	Summers	1984

Projection

- The projection operator, π (pi), specifies the columns to be retained from the input relation
- A selection has the form: $\pi_{\textit{columns}}$(*relation*)
  - Where *columns* are a comma separated list of column names
  - The list contains the names of the columns to be retained in the result relation

Projection Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\pi_{firstName,lastName}$(Customer)

firstName	lastName
Buffy	Summers
Xander	Harris
Cordelia	Chase
Rupert	Giles
Dawn	Summers

π_{birth}(Customer)

birth
1981
1980
1955

Selection and Projection Notes

- Selection and projection **eliminate duplicates**
  - Since relations are **sets!!**
- Both operations require one input relation (unary)
- The schema of the result of a selection is *the same as* the schema of the input relation
- The schema of the result of a projection contains just those attributes in the projection list

Composing Selection and Projection

$\pi_{\mathit{sin},\ \mathit{firstName}}(\sigma_{\mathit{birth} < 1982\ \wedge\ \mathit{lastName} = \text{"Summers"}}(\text{Customer}))$

Customer

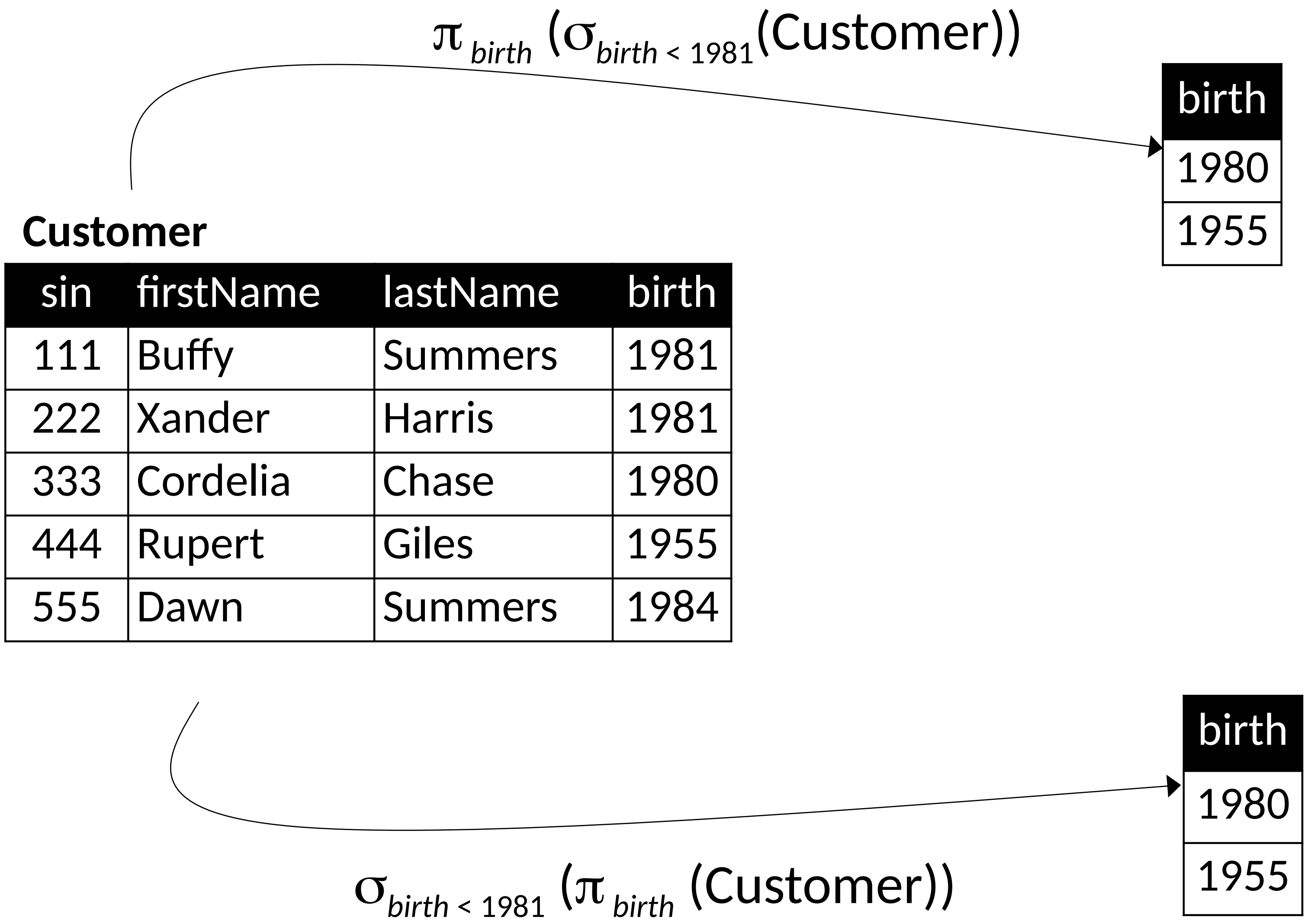
sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

intermediate relation

sin	firstName	lastName	birth
111	Buffy	Summers	1981

sin	firstName
111	Buffy

Composing Selection and Projection



$\pi_{birth}\ (\sigma_{birth\ <\ 1981}(\text{Customer}))$

birth
1980
1955

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\sigma_{birth\ <\ 1981}\ (\pi_{birth}\ (\text{Customer}))$

birth
1980
1955

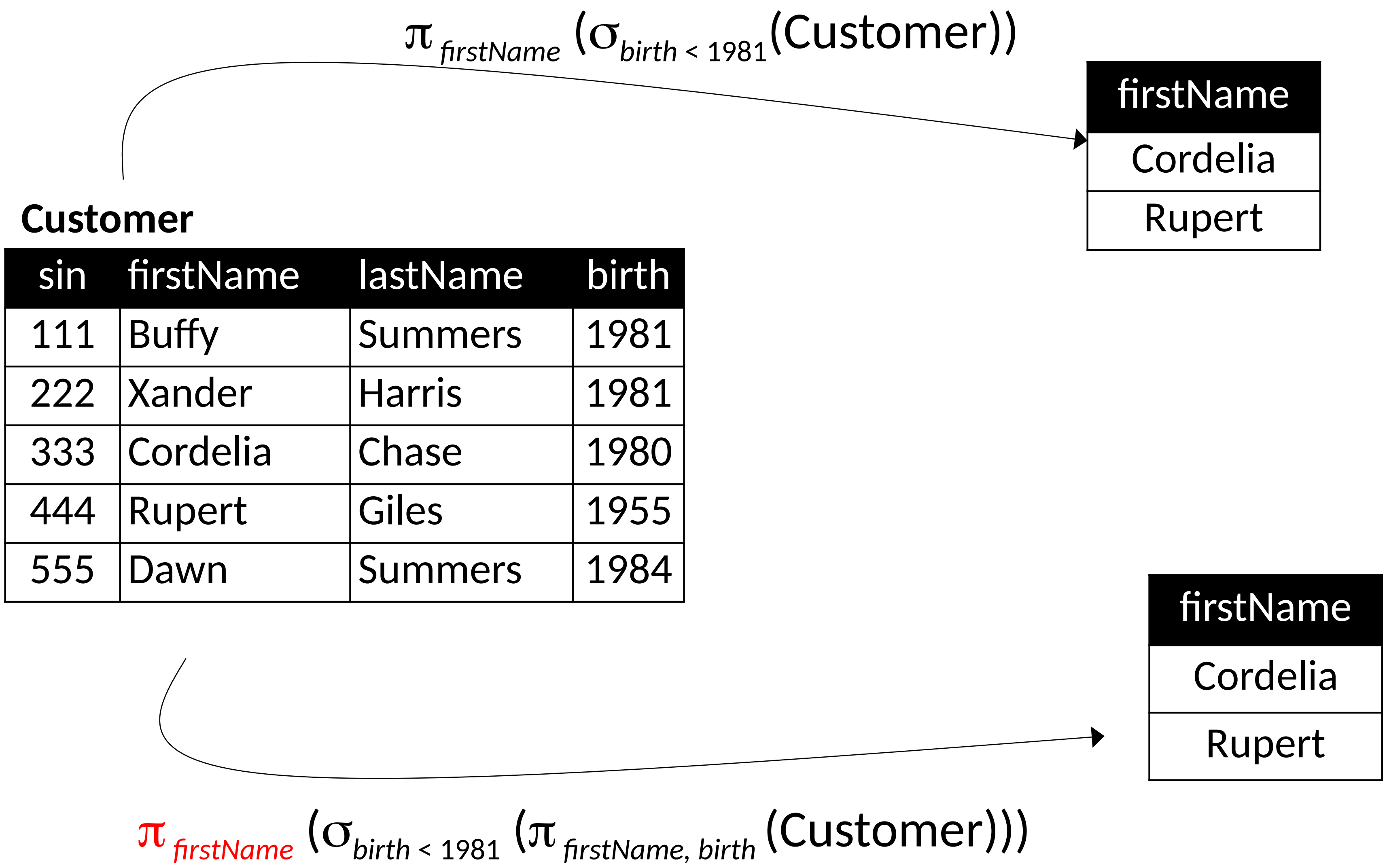
Commutative Property

- For example:
  - x + y = y + x
  - x * y = y * x

- It holds between two projections or two selections.
- Does it hold for projection and selection?

$$\pi_{\mathit{columns}}(\sigma_{\mathit{condition}}(R)) = \pi_{\mathit{condition}}(\sigma_{\mathit{columns}}(R))\ ?$$

Commutative property



$\pi_{\mathit{firstName}}\,(\sigma_{\mathit{birth} < 1981}(\text{Customer}))$

firstName
Cordelia
Rupert

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

firstName
Cordelia
Rupert

$\pi_{\mathit{firstName}}\,(\sigma_{\mathit{birth} < 1981}\,(\pi_{\mathit{firstName},\ \mathit{birth}}\,(\text{Customer})))$

Set Operations

A = {1, 3, 6}　　　　**B = {1, 2, 5, 6}**

Union ()	A B ≡ B A	A B = {1, 2, 3, 5, 6}	
Intersection()	A B ≡ B A	A B = {1, 6}	
Set Difference(–)	A – B ≠ B – A	A – B = {3}	B – A = {2, 5}

Union Compatible Relations

$$A \text{ op } B = R_{\text{result}}$$

- where op = , , or
- *A* and *B* must be union compatible
  - Same number of fields
  - Field i in each schema have the same type

Union Compatible Relations

Intersection of the Employee and Customer relations

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
412	Carol	Danvers	64000.00

The two relations are not union compatible as birth is a DATE and salary is a REAL

We can carry out preliminary operations to make the relations union compatible

$\pi_{sin,\ firstName,\ lastName}(\text{Customer}) \cap \pi_{sin,\ firstName,\ lastName}(\text{Employee})$

Union Compatible Relations

$$A \text{ op } B = R_{result}$$

- where op = , , or
- *A* and *B* must be union compatible
  - Same number of fields
  - Field i in each schema have the same type

- Result schema borrowed from A

$$\mathbf{A}(\text{age int}) \cup \mathbf{B}(\text{num int}) = \mathbf{R_{result}}\ (\text{age int})$$

Union

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

A B

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers
208	Clark	Kent
412	Carol	Danvers

Set Difference

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

A – B

sin	firstName	lastName
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

B – A

sin	firstName	lastName
208	Clark	Kent
412	Carol	Danvers

Note on Set Difference

- Notice that most operators are monotonic
  - Increasing size of inputs → outputs grow
- Set Difference is **non-monotonic**
  - Example: A – B
  - Increasing the size of B could decrease output size
- Set difference is **blocking**:
  - For A – B, must wait for all B tuples before any results

Intersection

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

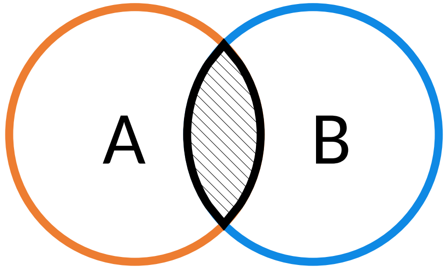
sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

A B

sin	firstName	lastName
111	Buffy	Summers

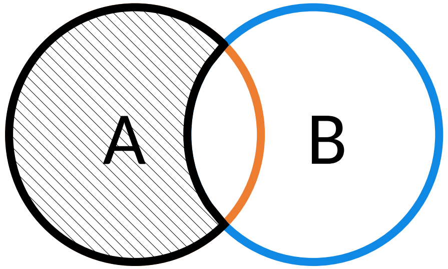
Note on Intersection

- $A \cap B = R_{result}$



- Can we express using other operators?
  - A ∩ B = A – (A B)

A – B =



Cartesian Product

$$\mathbf{A(a_1, \ldots, a_n) \text{ x } B(a_{n+1}, \ldots, a_m) = R_{result}(a_1, \ldots, a_m)}$$

- Each row of A paired with each row of B
  - Result schema concatenates A and B’s fields
  - Names are inherited if possible (i.e., if not duplicated)
    - If two field names are the same (i.e., a *naming conflict* occurs) and the affected columns are referred to by position
  - If *R* contains *m* records, and *S* contains *n* records, the result relation will contain *m* * *n* records

Renaming

- It is sometimes useful to assign names to the results of a relational algebra query
- The rename operator, ρ (rho)
  - $\rho_S(R)$ renames a relation
  - $\rho_{S(a1,a2,\ldots,an)}(R)$ renames a relation and its attributes
  - $\rho_{new/old}(R)$ renames specified attributes

R

sid1	firstName	lastName	birth	acc	type	balance	sid2
111	Buffy	Summers	1981	01	CHQ	2101.76	111
111	Buffy	Summers	1981	02	SAV	11300.03	333
111	Buffy	Summers	1981	03	CHQ	20621.00	444
555	Dawn	Summers	1984	01	CHQ	2101.76	111
555	Dawn	Summers	1984	02	SAV	11300.03	333
555	Dawn	Summers	1984	03	CHQ	20621.00	444

$\rho_{sid1/1,\ sid2/8}(R)$

Relational Algebra Exercise

- **Student** (sID, lastName, firstName, cgpa)
  - 101, Jordan, Michael, 3.8
- **Offering** (oID, prog, cNum, term, instructor)
  - abc, CSE, 541, Spring 2022, Dong
- **Took** (sID, oID, grade)
  - 101, abc, 95

1. sID of all students who have earned some grade over 80 and some grade below 50.

$$\pi_{sID}(\sigma_{grade > 80}(\text{Took})) \cap \pi_{sID}(\sigma_{grade < 50}(\text{Took}))$$

Relational Algebra Exercise

- **Student** (sID, lastName, firstName, cgpa)
    - 101, Jordan, Michael, 3.8
- **Offering** (oID, prog, cNum, term, instructor)
    - abc, CSE, 541, Spring 2022, Dong
- **Took** (sID, oID, grade)
    - 101, abc, 95

2. SID of all students who have taken CSE 541

$$\pi_{sID}(\sigma_{Offering.oID = Took.oID \wedge prog = \text{'CSE'} \wedge cNum = 541}(\text{Offering x Took}))$$

Largest Balance

Account = {accNumber, type, balance, branchName}

- Find the account with the largest balance; return *accNumber*
  1. Find accounts which are less than some other account

$$\sigma_{account.balance\ <\ d.balance}\ (\text{Account} \times \rho_d\ (\text{Account}))$$

  2. Use set difference to find the account with the largest balance

$$\pi_{accNumber}\ (\text{Account}) -$$

$$\pi_{account.accNumber}(\sigma_{account.balance\ <\ d.balance}\ (\text{Account} \times \rho_d\ (\text{Account})))$$

(Inner) Joins

- **Motivation**
  - **Simplify some queries that require a Cartesian product**

- **Natural Join:** $R \bowtie S = \pi_A (\sigma_\theta(R \times S))$

- **Theta Join:** $R \bowtie_\theta S = \sigma_\theta (R \times S)$

- **Equijoin:** $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join condition θ consists only of equalities

Natural Join

- There is often a natural way to join two relations
  - Join based on common attributes
  - Eliminate duplicate common attributes from the result

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

Customer ⋈ Employee

sin	firstName	lastName	birth	salary
111	Buffy	Summers	1981	22000.78

Natural Join

$$\mathbf{R \bowtie S}$$

- Definition: $R \bowtie S =$
- Where:
  - Selection σ_θ checks equality of all common attributes (i.e., attributes with same names)
  - Projection π_A eliminates duplicate common attributes

- The natural join of two tables with *no fields in common* is the **Cartesian product**
  - Not the empty set

Natural Join Example

R

A	B	C	D
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

S

A	B	C	E
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

$$\mathbf{R \bowtie S} \quad \mathbf{=} \pi_{A,B,C,D,E}(_{R.A=S.A \wedge R.B=S.B \wedge R.C=S.C}(R \times S))$$

A	**B**	C	D	E
111	Buffy	Summers	1981	22000.78

Theta Join

$$\mathbf{R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)}$$

- Most general form
    - θ can be any condition
- No projection in this case!
    - Result schema same as cross product

Theta Join Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
412	Carol	Danvers	64000.00

Customer $\bowtie_{\mathbf{Customer.sin} < \mathbf{Employee.sin}}$ **Employee**

1	2	3	birth	5	6	7	salary
111	Buffy	Summers	1981	208	Clark	Kent	80000.55
111	Buffy	Summers	1981	412	Carol	Danvers	64000.00
222	Xander	Harris	1981	412	Carol	Danvers	64000.00
333	Cordelia	Chase	1980	412	Carol	Danvers	64000.00

Equi-Join

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

A theta join where θ is an equality predicate

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

Customer $\bowtie_{\text{Customer.sin = Employee.sin}}$ **Employee**

1	2	3	birth	5	6	7	salary
111	Buffy	Summers	1981	111	Buffy	Summers	22000.78

(Inner) Joins Summary

- **Natural Join:** $R \bowtie S = \pi_A (\sigma_\theta(R \times S))$
  - Equality on all fields with same name in R and in S
  - Projection π_A drops all redundant attributes

- **Theta Join:** $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
  - No projection

- **Equijoin:** $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join condition θ consists only of equalities
  - No projection

Relational Algebra Exercise

- **Student** (sID, lastName, firstName, cgpa)
  - 101, Jordan, Michael, 3.8
- **Course** (prog, cNum, name, core)
  - CMPSC, 431W, DB, True
- **Offering** (oID, prog, cNum, term, instructor)
  - abc, CSE, 541, Spring 2022, Dong
- **Took** (sID, oID, grade)
  - 101, abc, 95

3. The names of all students who have passed a breadth course (grade >= 60 and core = True) with Dong

$$\pi_{\mathit{lastName,\ firstName}}(\sigma_{\mathit{core\ =\ True\ \wedge\ grade\ >\ 60\ \wedge\ instructor\ =\ 'Dong'}}(\text{Student} \bowtie \text{Took} \bowtie \text{Offering} \bowtie \text{Course}))$$

Different Plans, Same Results

- Semantic equivalence: results are ***always*** the same

$$\pi_{name}(\sigma_{cNum=354}(R \bowtie S))$$

$$\pi_{name}(\sigma_{cNum=354}(R) \bowtie S)$$

- Are they equivalent?
- Which one is more efficient?
- Can you make it even more efficient?

Other Operations

- There are additional relational algebra operators
  - Usually used in the context of query optimization
- Duplicate elimination – δ
  - Used to turn a bag into a set
- Aggregation operators
  - e.g. sum, average
- Grouping – γ
  - Used to partition tuples into groups
    - Typically used with aggregation

How DBMSs leverage RA

Table schemas:

```
Sailors(sid: integer, sname: string, rating: integer, age: real)
Reserves(sid: integer, bid: integer, day: dates, rname: string)
```

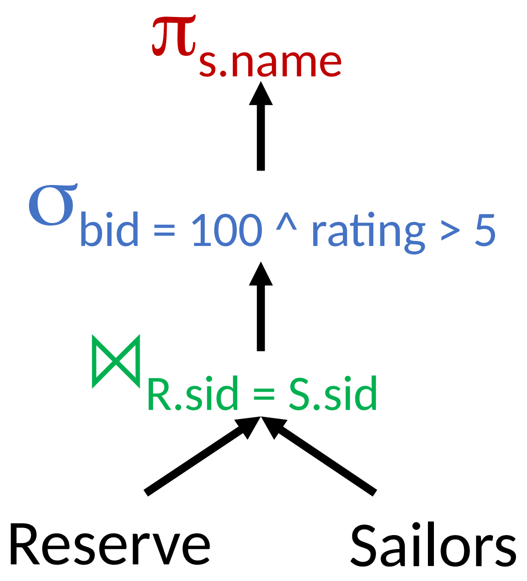
SQL query (declarative):

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

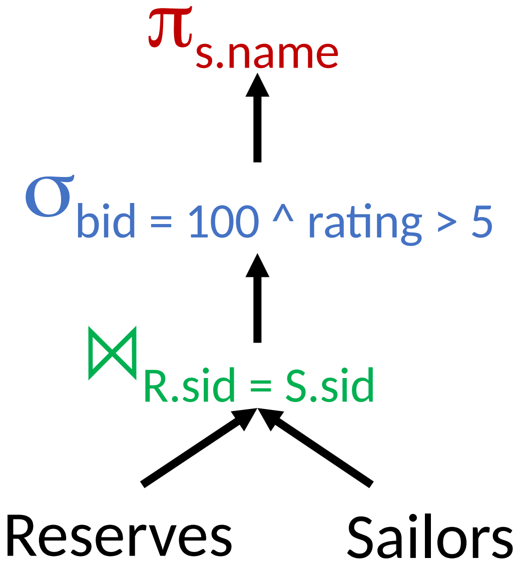
SQL parser

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

Equivalent logical query plan
(relational algebra tree):


$\pi_{s.name}$
$\sigma_{bid\ =\ 100\ \wedge\ rating\ >\ 5}$
$\bowtie_{R.sid\ =\ S.sid}$
Reserve
Sailors

Relational Operators and Query Plans



$\pi_{s.name}$

$\sigma_{bid = 100 \wedge rating > 5}$

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

Reserves

Sailors

Edges: "flow" of tuples

Vertices: relational algebra operators

- Input/output: relation

- Aka "data-flow" graph, also used in other systems

- Query optimizer determines the implementation to use for each operator
- Query executor runs the relational operators