

# Data Management

## 04 Relational Algebra

**Matthias Boehm**

Graz University of Technology, Austria  
Computer Science and Biomedical Engineering  
Institute of Interactive Systems and Data Science  
BMK endowed chair for Data Management

# Announcements/Org

## ■ #1 Video Recording

- Link in **TeachCenter** & **TUbe** (lectures will be public)
- <https://tugraz.webex.com/meet/m.boehm>



## ■ #2 Reminder Communication

- **Newsgroup:** <news://news.tugraz.at/tu-graz.lv.dbase>
- **Office hours:** Mo 12.30-1.30pm (<https://tugraz.webex.com/meet/m.boehm>)

## ■ #3 Exercise Submissions

- **Exercise 1:** Mar 30 11.59 + 7 late days, in TeachCenter

## ■ #4 KDD 2021 Cup: Time Series Anomaly Detection

- <https://compete.hexagon-ml.com/practice/competition/39/#>
- 200 train/test datasets, each test has **one anomaly**
- Phase 1: **Mar 15 – Apr 7**, Phase 2: Apr 8 – June 1
- Prices: \$2000 first, \$1000 second, \$500 third



# Recap: Relations and Terminology

- Domain  $D$  (value domain): e.g., Set  $S$ , INT, Char[20]

- Relation  $R$

- Relation schema  $RS$ :  
Set of  $k$  attributes  $\{A_1, \dots, A_k\}$
- Attribute  $A_j$ : value domain  $D_j = \text{dom}(A_j)$
- Relation: subset of the Cartesian product over all value domains  $D_j$   
 $R \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$

Attribute

	A1 INT	A2 INT	A3 BOOL
	3	7	T
	1	2	T
	3	4	F
Tuple	1	7	T

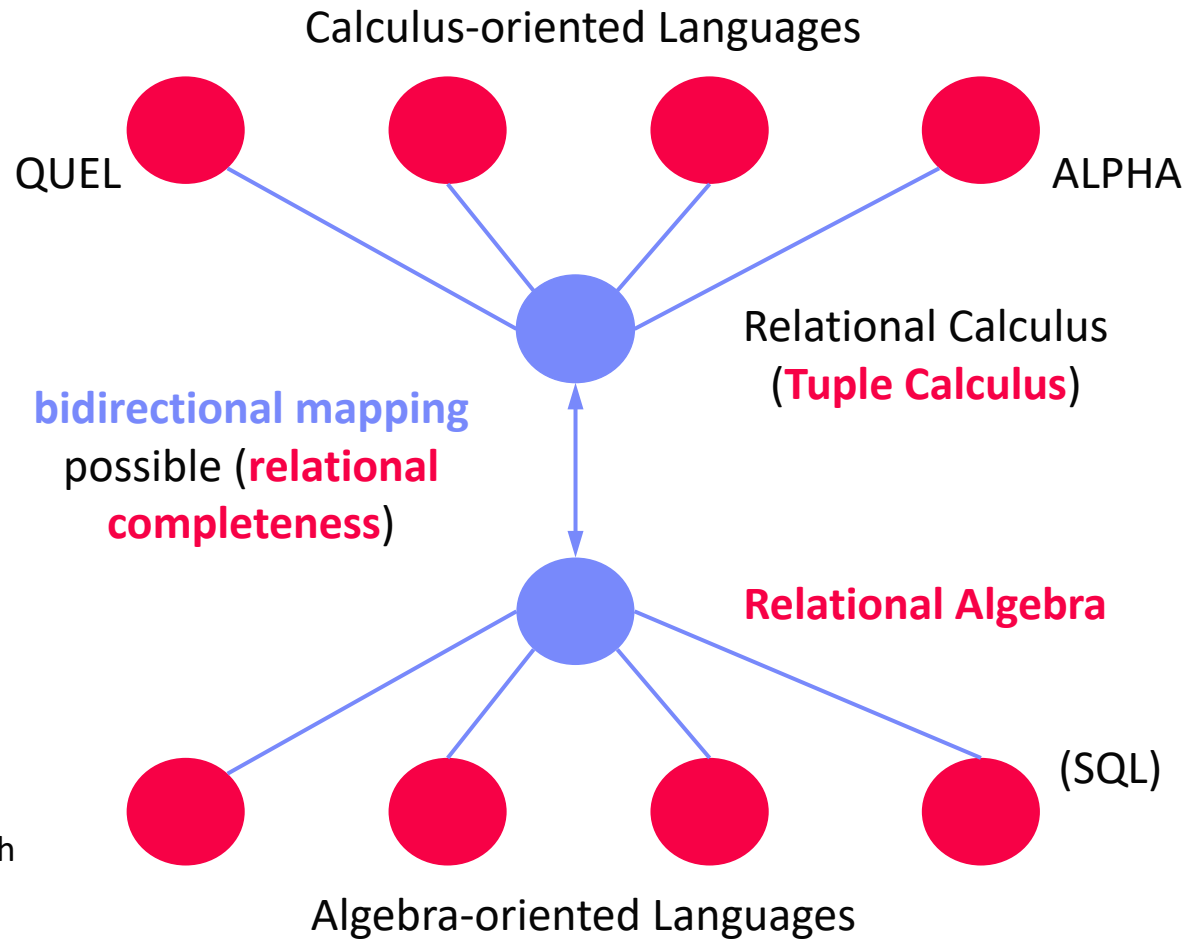
cardinality: 4  
rank: 3

- Additional Terminology

- Tuple: row of  $k$  elements of a relation
- Cardinality of a relation: number of tuples in the relation
- Rank of a relation: number of attributes
- Semantics: **Set** := no duplicate tuples (in practice: **Bag** := duplicates allowed)
- Order of tuples and attributes is irrelevant

# Relational Algebra vs Tuple Calculus

## Comparison Scheme for Data Sub Languages



[E. F. Codd: Relational Completeness of Data Base Sublanguages. IBM Research Report RJ987, **1972**]

# Agenda

- **Relational Algebra**
- **Tuple Calculus**
- **Physical Operators (Preview Lecture 08)**

# Database Research Self-Assessment 2018

<u>PID</u>	Firstname	Lastname	Affiliation	LID
102	Anastasia	Ailamaki	EPFL	1
104	Peter	Bailis	Stanford	
105	Magdalena	Balazinska	U Washington	3
107	Peter	Boncz	CWI	2
108	Surajit	Chaudhuri	MS Research	3
111	Luna	Dong	Amazon	3
113	Juliana	Freire	NYU	5
115	Joe	Hellerstein	UC Berkley	6
116	Stratos	Idreos	Harvard	7
117	Donald	Kossman	MS Research	
118	Tim	Kraska	MIT	7
120	Volker	Markl	TU Berlin	8
122	Tova	Milo	Tel Aviv University	9
123	C.	Mohan	IBM Research	10
124	Thomas	Neumann	TU Munich	11
126	Fatma	Ozcan	IBM Research	10
130	Christopher	Re	Stanford	4



# Database Research Self-Assessment 2018, cont.

<u>PID</u>	...	Affiliation	LID		<u>LID</u>	Location
102		EPFL	1	→	1	Lausanne, SUI
104		Stanford			2	Amsterdam, NLD
105		U Washington	3		3	Seattle, USA
107		CWI	2		4	Stanford, USA
108		MS Research	3		5	New York, USA
111		Amazon	3		6	Berkley, USA
113		NYU	5		7	Cambridge, USA
115		UC Berkley	6		8	Berlin, GER
116		Harvard	7		9	Tel Aviv, ISR
117		MS Research			10	San Jose, USA
118		MIT	7		11	Munich, GER
120		TU Berlin	8			
122		Tel Aviv University	9			
123		IBM Research	10			
124		TU Munich	11			
126		IBM Research	10			
130		Stanford	4			

**Recommended  
Reading**

[Daniel Abadi et al: The Seattle Report  
on Database Research, **SIGMOD Record**  
Vol 48, No 4, Dec 2019]



# Relational Algebra



# Core Relational Algebra

## Relational Algebra

- **Operands:** relations (normalized, variables for computing new values)
- **Operators:** traditional set operations and specific relational operations

## Basic Operations (minimal)

- Cartesian product  $R \times S$
- Union  $R \cup S$
- Difference  $R - S$
- Projection  $\pi_{i_1, \dots, i_m}(R)$
- Selection  $\sigma_F(R)$

## Derived Operations

- Intersection  $R \cap S$
- Join  $R \bowtie S$
- Division  $R \div S$

## Rename $\rho_S(R)$

	Traditional Set Operations	Specific Relational Operations
Basic Operations	$R \times S$ $R \cup S$ $R - S$	$\pi_{i_1, \dots, i_m}(R)$ $\sigma_F(R)$
Derived Operations	$R \cap S$	$R \bowtie S$ $R \div S$

# Extended Relational Algebra

## Extended Relational Algebra

- Relational algebra introduced with **set semantics** (no duplicates)      Set<T>
- SQL with **bag semantics** (more flexibility and performance)      Bag<T>
- **Codd'72**: *In a practical environment it would need to be augmented by a counting and summing capability, together with [...] library functions [...].*

## Additional Operations (Ext)

- Duplicate elimination  $\delta(R)$
- Grouping  $\gamma_{A,f(B)}R$
- Sorting  $\tau_A(R)$



## Bag (aka Multiset) Terminology

- Multiplicity: # occurrences of an instance
- Cardinality: # tuples (i.e., # instances weighted by multiplicity)

A	B
a	b
b	c
a	b

# Cartesian Product



- Definition:**  $R \times S := \{(r,s) \mid r \in R, s \in S\}$ 
  - Set of all pairs of inputs (equivalent in **set/bag**)

- Example**

<u>PID</u>	Firstname	Lastname	Affiliation	LID
104	Peter	Bailis	Stanford	
130	Christopher	Re	Stanford	4

SF Bay Area

<u>LID</u>	Location
4	Stanford, USA
6	Berkley, USA
10	San Jose, USA



<u>PID</u>	Firstname	Lastname	Affiliation	LID	LID	Location
104	Peter	Bailis	Stanford		4	Stanford, USA
130	Christopher	Re	Stanford	4	4	Stanford, USA
104	Peter	Bailis	Stanford		6	Berkley, USA
130	Christopher	Re	Stanford	4	6	Berkley, USA
104	Peter	Bailis	Stanford		10	San Jose, USA
130	Christopher	Re	Stanford	4	10	San Jose, USA

# Union

Basic	Ext
Derived	

- Definition:**  $R \cup S := \{x \mid x \in R \vee x \in S\}$ 
  - Set:** set union with duplicate elimination (idempotent:  $S \cup S = S$ )
  - Bag:** bag union (commutative but not idempotent)
- Example w/ set semantics**

**Note:** Intersection ( $\cap$ ) has OR semantics ( $\vee$ )

Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL
Peter	Boncz	CWI
Volker	Markl	TU Berlin
Thomas	Neumann	TU Munich

Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL
Magdalena	Balazinska	U Washington
Juliana	Freire	NYU
Tova	Milo	Tel Aviv University



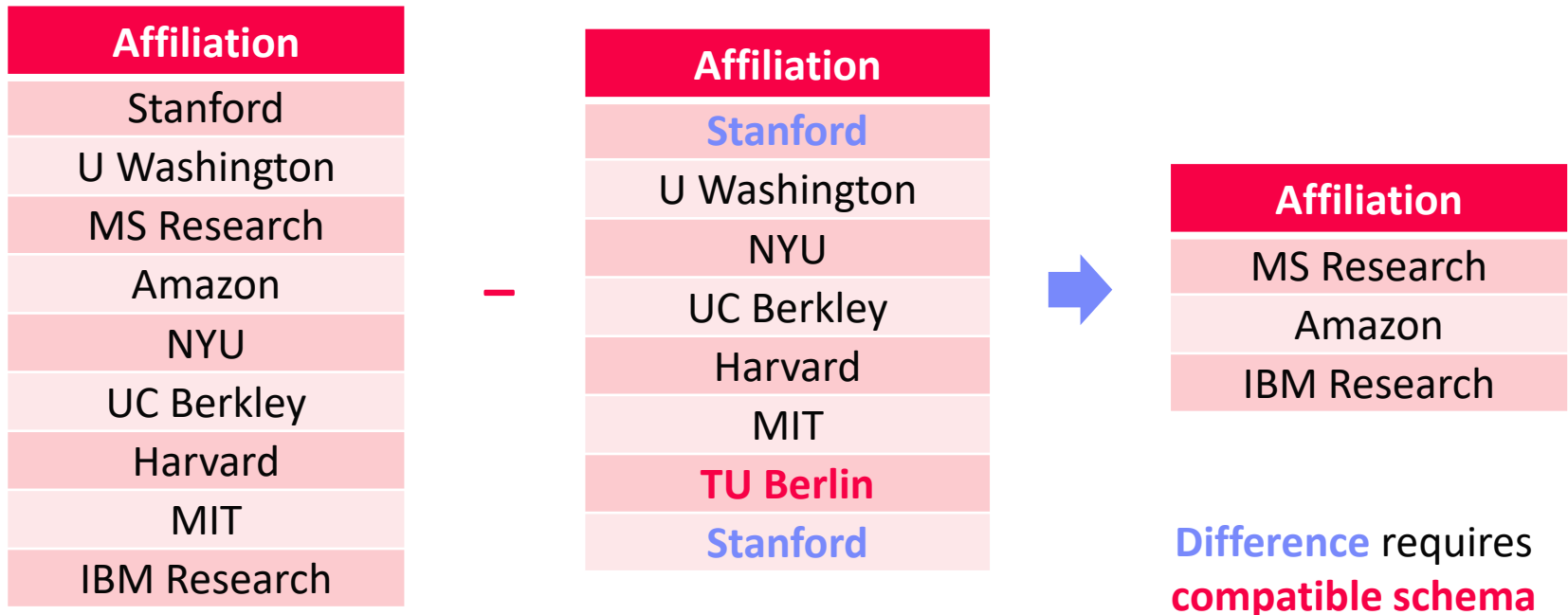
Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL
Peter	Boncz	CWI
Volker	Markl	TU Berlin
Thomas	Neumann	TU Munich
Magdalena	Balazinska	U Washington
Juliana	Freire	NYU
Tova	Milo	Tel Aviv University

**Union** requires compatible schema

# Difference

Basic	Ext
Derived	

- Definition:**  $R - S := \{x \mid x \in R \wedge x \notin S\}$  (sometimes  $\setminus$ )
  - Set:** set difference
  - Bag:** element multiplicity of R minus multiplicity  $\min(R, S)$
- Example w/ bag semantics**



# Projection and Selection

Basic	Ext
Derived	

## Projection $\pi_{i_1, \dots, i_m}(R)$

- **Set:** selection of attributes with duplicate elimination
- **Bag:** selection of attributes

## Extended Projection

- Arithmetic expressions:  $\pi_{A, A*B}(R)$
- Duplicate occurrences:  $\pi_{A, A, B}(R)$

## Selection (restriction) $\sigma_F(R)$

- Selection of tuples satisfying the predicate F (equivalent in **set**/**bag**)
- Example:  $\sigma_{\text{Affiliation}='IBM Research'}(R)$

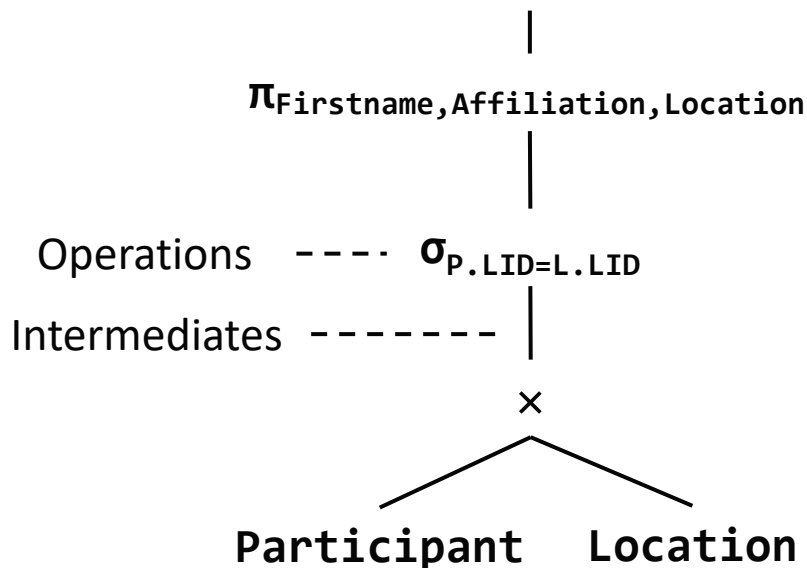
**Example:**  
 $\pi_{\text{Affiliation}}(R)$  w/  
 set semantics

Affiliation
EPFL
Stanford
U Washington
CWI
MS Research
Amazon
NYU
UC Berkley
Harvard
MIT
TU Berlin
Tel Aviv University
IBM Research
TU Munich

PID	Firstname	Lastname	Affiliation	LID
123	C.	Mohan	IBM Research	10
126	Fatma	Ozcan	IBM Research	10

# Query Trees

- Composition of Complex Queries**
  - Relational algebra expressions  $\rightarrow$  **data flow graph (tree)**
  - Leaf nodes represent base relations; root node represents result
- Example**  $\pi_{\text{Firstname,Affiliation,Location}} (\sigma_{\text{P.LID=L.LID}}(\text{Participant} \times \text{Location}))$



Firstname	Affiliation	Location
Anastasia	EPFL	Lausanne, SUI
Magdalena	U Washington	Seattle, USA
Peter	CWI	Amsterdam, NLD
Surajit	MS Research	Seattle, USA
...	...	...

# Intersection

Basic	Ext
Derived	

- Definition**  $R \cap S := R - (R - S)$  (derived from basic operations)
  - Set:** set intersection derived from difference
  - Bag:** bag intersection, with element multiplicity  $\min(R, S)$
- Example**

Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL
Peter	Boncz	CWI
Volker	Markl	TU Berlin
Thomas	Neumann	TU Munich

Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL
Magdalena	Balazinska	U Washington
Juliana	Freire	NYU
Tova	Milo	Tel Aviv University



Firstname	Lastname	Affiliation
Anastasia	Ailamaki	EPFL

**Note:** Intersection ( $\cap$ ) has AND semantics ( $\wedge$ )

**Intersection** requires **compatible schema**



# Division

Basic	Ext
Derived	

- Definition**  $R \div S := \pi_{R-S}(R) - \pi_{R-S}((\pi_{R-S}(R) \times S) - R)$ 
  - Find instances in R that satisfy S (e.g., **which students took ALL DB courses**)
  - $R \div S := \{(a_1, \dots, a_{r-s}) \mid \forall (b_1, \dots, b_s) \in S : (a_1, \dots, a_{r-s}, b_1, \dots, b_s) \in R\}$

- Example**

A	B	C	D
a	b	c	d
a	b	e	f
b	c	e	f
e	d	e	f
e	d	c	d
a	b	d	e

C	D
c	d
e	f

➔

A	B
a	b
e	d

(many-to-one set containment test)

- Example Derivation**

$\pi_{R-S}(R)$	
A	B
a	b
b	c
e	d

$\pi_{R-S}(R) \times S$			
A	B	C	D
a	b	c	d
a	b	e	f
b	c	c	d
b	c	e	f
e	d	c	d
e	d	e	f

$(\pi_{R-S}(R) \times S) - R$			
A	B	C	D
b	c	c	d

$\pi_{R-S}((\pi_{R-S}(R) \times S) - R)$	
A	B
b	c

$\pi_{R-S}(R) - \pi_{R-S}((\pi_{R-S}(R) \times S) - R)$	
A	B
a	b
e	d

# (Inner) Join

Basic	Ext
Derived	

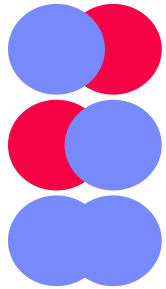
- Definition  $R \bowtie S := \pi_{\dots} (\sigma_F (R \times S))$** 
  - Selection of tuples (and attributes) from the cartesian product  $R \times S$  (equivalent in **set/bag**); **beware of NULLs**: do never match
  - Theta Join**:  $R \bowtie_{\Theta} S := \sigma_{\Theta} (R \times S)$ ; arbitrary condition e.g.,  $\Theta \in \{=, \neq, <, \leq, >, \geq\}$
  - Natural Join**:  $R \bowtie S$ ; **equi join** ( $\Theta$  is =) w/ shared attributes appearing once
- Example Natural Join Participant  $\bowtie$  Location**

<u>PID</u>	Firstname	Lastname	Affiliation	LID
102	Anastasia	Ailamaki	EPFL	1
104	Peter	Bailis	Stanford	
105	Magdalena	Balazinska	U Washington	3



PID	Firstname	Lastname	Affiliation	LID	Location
102	Anastasia	Ailamaki	EPFL	1	Lausanne, SUI
105	Magdalena	Balazinska	U Washington	3	Seattle, USA

# Other Types of Joins



## Outer Joins

- Left outer join  $\bowtie$  (tuples of lhs, NULLs for non-existing rhs)
- Right outer join  $\bowtie$  (tuples of rhs, NULLs for non-existing lhs)
- Full outer join  $\bowtie$  (tuples of lhs/rhs, NULLs for non-existing lhs/rhs)
- Example

Participant  
 $\bowtie$  Location

PID	Firstname	Lastname	Affiliation	LID	LID	Location
102	Anastasia	Ailamaki	EPFL	1	1	Lausanne, SUI
104	Peter	Bailis	Stanford	NULL	NULL	NULL
105	Magdalena	Balazinska	UW	3	3	Seattle, USA

## Semi Join

- Left semi join  $\ltimes := \pi_R(R \bowtie S)$  (filter lhs)
- Right semi join  $\rtimes$  (filter rhs)
- Example

Participant  
 $\rtimes \sigma_{GER}$  (Location)

PID	Firstname	Lastname	Affiliation	LID
120	Volker	Markl	TU Berlin	8
124	Thomas	Neumann	TU Munich	11

## Anti Join

- Left anti join  $R \triangleright S := R - R \ltimes S$  (complement of left semi join)
- Right anti join (complement of right semi join)

# Deduplication, Sorting, and Renaming



## ▪ Duplication Elimination $\delta(R)$

- Convert a **bag** into a **set** by removing all duplicate instances
- SQL: use **ALL** or **DISTINCT** to indicate w/ or w/o duplicate elimination

## ▪ Sorting $\tau_A(R)$

- Convert a **bag** into a **sorted list** of tuples; order lost if used in other ops
- SQL: sequence of attributes with ASC (ascending) or DESC (descending) order
- Example:  $\tau_{\text{Firstname ASC, Lastname ASC}}(\text{Participant})$

## ▪ Rename $\rho_S(R)$

- Define new schema (attribute names), but keep tuples unchanged
- **Example:**  $\rho_{\text{ID, Given Name, Family Name, Affiliation, LID}}(\text{Participant})$

# Grouping and Aggregation

Basic	Ext
Derived	

## Definition $\gamma_{A,f(B)}R$

- **Grouping:** group input tuples R according to unique values in A
- **Aggregation:** compute aggregate  $f(B)$  per group of tuples (aggregation w/o grouping possible)

## Example $\gamma_{\text{Affiliation},\text{COUNT}(*)}\text{Participant}$

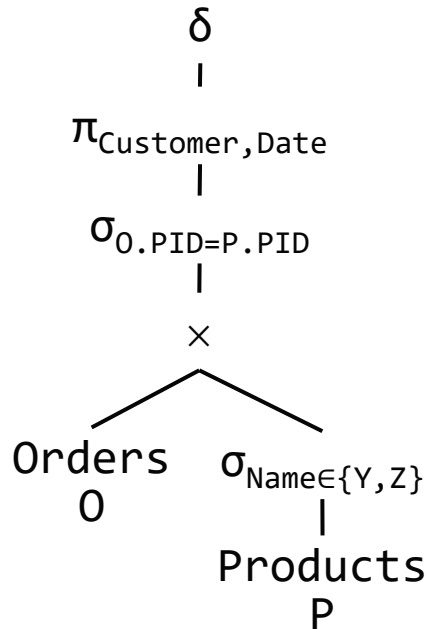
### Classification of Aggregates $f(B)$

- **Additive** aggregation functions (**SUM**, **COUNT**)
- **Semi-additive** aggregation functions (**MIN**, **MAX**)
- **Additively computable** aggregation functions (**AVG**, **STDDEV**, **VAR**)
- Aggregation functions (**MEDIAN**, **QUANTILES**)

Affiliation	COUNT
EPFL	1
Stanford	2
U Washington	1
CWI	1
MS Research	2
Amazon	1
...	...
IBM Research	2
TU Munich	1

# BREAK (and Test Yourself)

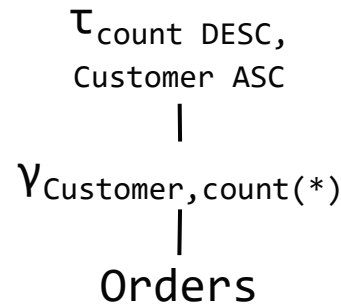
- Task: Compute the results for the following queries.



Customer	Date
A	'2019-06-22'
C	'2019-06-23'
D	'2019-06-23'

Orders

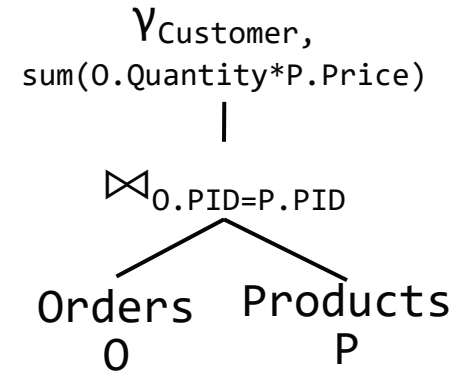
OID	Customer	Date	Quantity	PID
1	A	'2019-06-22'	3	2
2	B	'2019-06-22'	1	3
3	A	'2019-06-22'	1	4
4	C	'2019-06-23'	2	2
5	D	'2019-06-23'	1	4
6	C	'2019-06-23'	1	1



Customer	Count
A	2
C	2
B	1
D	1

Products

PID	Name	Price
1	X	100
2	Y	15
4	Z	75
3	W	120



Customer	Sum
A	120
B	120
C	130
D	75

# Tuple Calculus

# Overview Tuple Calculus

## Relational Calculus

- **Tuple Calculus** – tuple relational calculus:  $\{T \mid p(T)\}$  (tuple  $\rightarrow$  set)  
 $\rightarrow$  **examples:** see definition of relational algebra
- (**Domain Calculus** – domain relational calculus: attribute  $\rightarrow$  set)

## Characteristics Tuple Calculus

- Calculus expression **does not specify order of operations**
- Calculus expressions consist of variables, constants, comparison operators, logical concatenations, and quantifiers
- Expressions are formulas, free formal variables  $\rightarrow$  result

## Example Selection

$$\sigma_{A=7}(R) := \{t \mid t \in R \wedge t[A]=7\}$$

tuple variable

domain of tuple variable

predicate

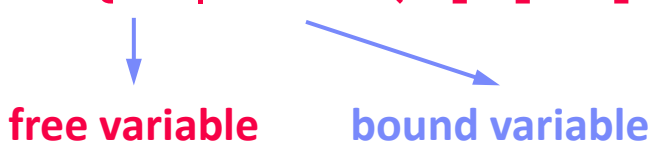


# Quantifiers

## Variables

- Free: unbound variables define the result
- Bound: existential quantifier  $\exists x$  and universal quantifier  $\forall x$  bind a variable  $x$

**Example Projection**  $\pi_{A,B}(R) := \{t \mid \exists r \in R (t[A]=r[A] \wedge t[B]=r[B])\}$



(relation T defined with two attributes A and B)

## Safe Queries

- Guarantees finite number of tuples (otherwise, unsafe)
- Example unsafe query:  $\{t \mid t \notin R\}$
- Relational completeness:** Every safe query expressible in RA and vice versa

# Relational Algebra vs Tuple Calculus **Revisited**

- **E. F. Codd argued for Tuple Calculus**

- **Criticism RA:** operator-centric
- Ease of Augmentation (w/ lib functions)
- Scope for Search Optimization
- Authorization Capabilities
- **Closeness to Natural Language**

[E. F. Codd: Relational Completeness of Data Base Sublanguages. IBM Research Report RJ987, **1972**]



→ **focus on query language**

- **System R Team used SEQUEL + RA**

- **Criticism Tuple Calculus:** too complex
- Iterating over tuples (not set-oriented)
- Quantifiers and bound variables
- Join over all variable attributes and result mapping

[Donald D. Chamberlin, Raymond F. Boyce: SEQUEL: A Structured English Query Language. SIGMOD Workshop **1974**]



- **Equivalent expressiveness + simplicity of RA + use as IR**

→ **Relational Algebra as basis for SQL und DBMS in practice**

# Excursus: The History of System R and SQL

## Gem: “The Birth of SQL – Prehistory / System R” (SQL Reunion 1995)

- [https://www.mcjones.org/System\\_R/SQL\\_Reunion\\_95/sqlr95-Prehisto.html](https://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-Prehisto.html)
- [https://www.mcjones.org/System\\_R/SQL\\_Reunion\\_95/sqlr95-System.html](https://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-System.html)
- **Don Chamberlin:** *We had this idea, that Codd had developed two languages, called the relational algebra and the relational calculus. [...] The relational calculus was a kind of a strange mathematical notation with a lot of quantifiers in it. We thought that what we needed was a language that was different from either one of those, [...].*
- **Don Chamberlin:** *Interestingly enough, Ted Codd didn't participate in that as much as you might expect. He got off into natural language processing [...]. He really didn't get involved in the nuts and bolts of System R very much. I think he may have wanted to maintain a certain distance from it in case we didn't get it right. Which I think he would probably say we didn't.*
- **Mike Blasgen:** *Oh, he has said that, many times.*

# Physical Operators

(Preview of Lecture 08 Query Processing)

# Iterator Model

Scalable (small memory)

High CPI measures

## Volcano Iterator Model

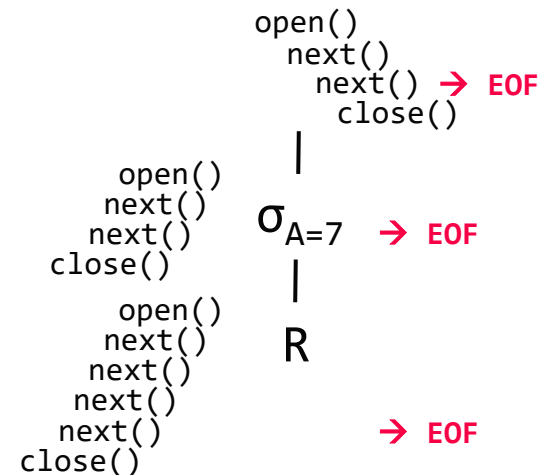
- Pipelined & no global knowledge
- Open-Next-Close (ONC) interface
- Query execution from root node (pull-based)

[Goetz Graefe: Volcano - An Extensible and Parallel Query Evaluation System. IEEE Trans. Knowl. Data Eng. 1994]



## Example $\sigma_{A=7}(R)$

```
void open() { R.open(); }
void close() { R.close(); }
Record next() {
    while( (r = R.next()) != EOF )
        if( p(r) ) //A==7
            return r;
    return EOF;
}
```



## Blocking Operators

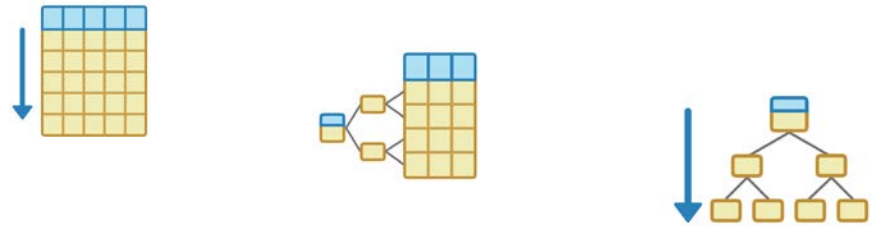
- Sorting, grouping/aggregation, build-phase of (simple) hash joins

PostgreSQL: `Init()`, `GetNext()`, `ReScan()`, `MarkPos()`, `RestorePos()`, `End()`

# (Selected) Physical Operators in PostgreSQL

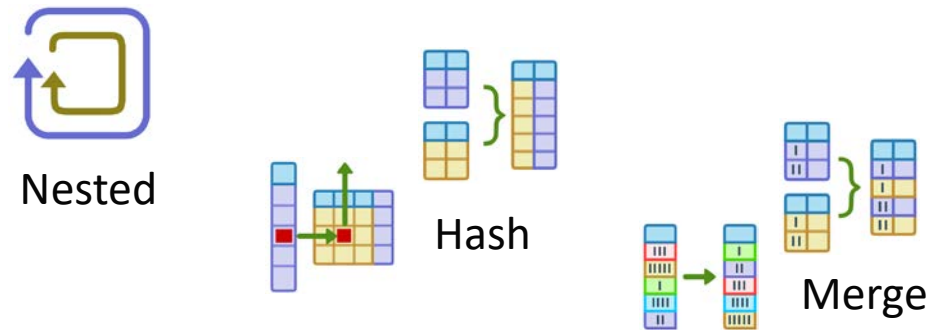
## Physical Table Access Operators

- Seq Scan (table scan)
- Index Scan
- Index Only Scan



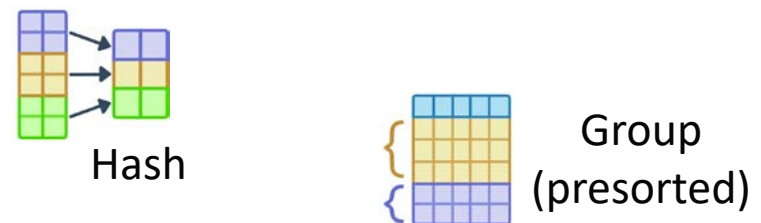
## Physical Join Operators

- Nested Loop Join
- Hash / Hash Join
- Sort / Merge Join



## Physical Grouping Operators

- Hash Aggregate
- Group Aggregate



[Images: PostgreSQL/11/pgAdmin 4/web/pgadmin/misc/static/explain/img]

# ANALYZE and EXPLAIN

**Note:** SQL for table creation and insert in the pptx notes.

- **Step 1: EXPLAIN SELECT \* FROM Participant AS R, Locale AS S WHERE R.LID=S.LID;**

Hash Join (.. rows=70 width=1592)

Hash Cond:(s.lid = r.lid)

-> Seq Scan on locale s (.. rows=140 width=520)

-> Hash (.. rows=70 width=1072)

-> Seq Scan on participant r (.. rows=70 width=1072)

} build side

- **Step 2: ANALYZE Participant, Locale;**

- **Step 3: EXPLAIN SELECT \* FROM Participant AS R, Locale AS S WHERE R.LID=S.LID;**

Hash Join (.. rows=17 width=47)

Hash Cond:(r.lid = s.lid)

-> Seq Scan on participant r (.. rows=17 width=30)

-> Hash (.. rows=11 width=17)

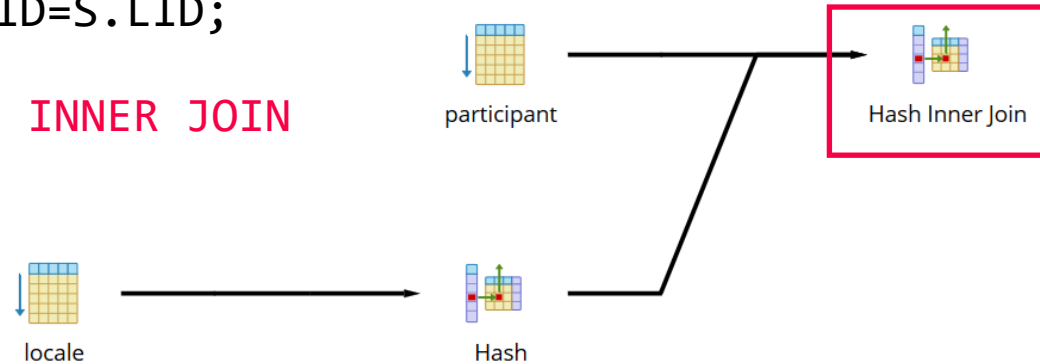
-> Seq Scan on locale s (.. rows=11 width=17)

**WHY?**

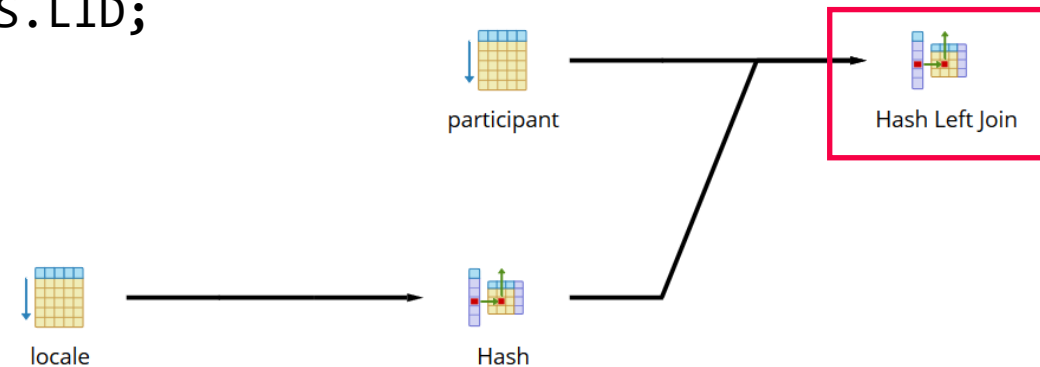
# Visual EXPLAIN

- SELECT \* FROM Participant AS R, Locale AS S  
 WHERE R.LID=S.LID;**

$\sigma_F(R \times S) \rightarrow$  INNER JOIN



- SELECT \* FROM Participant AS R LEFT JOIN Locale AS S  
 ON R.LID=S.LID;**





# Conclusions and Q&A

## ■ Summary

- **Fundamentals of relational algebra** and tuple calculus
- Preview of query trees and physical operators

## ■ Reminder Exercises

- **Exercise 1:** Mar 30 11.59 + 7 late days, in TeachCenter

## ■ Next Lectures

- **Easter break:** Mar 27 – Apr 10
- **05 Query Languages (SQL, XML, JSON), incl. Exercise 2** [Apr 12]
- **06 APIs (ODBC, JDBC, OR frameworks)** [Apr 19]
- **07 Physical Design and Tuning** [Apr 26]
- **08 Query Processing** [May 03]
- **09 Transaction Processing and Concurrency** [May 10]