TU Graz

SCIENCE
PASSION
TECHNOLOGY

Database Systems
04 Relational Algebra

Matthias Boehm

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

Last update: Mar 25, 2019

ISDS

Announcements/Org

- **#1 Video Recording**
  - Since last week, video/audio recording
  - Link in **TeachCenter** & **TUbe** (but not public yet)
- **#2 Exercise Submission**
  - Submission through **TeachCenter** (max 5MB, draft possible)
  - **Starting today** (deadline **Apr 02, 12.59pm**)

→ **Nobody left behind** (if problems, contact us via newsgroup or email)

Recap: Relations and Terminology

- **Domain D (value domain): e.g., Set S, INT, Char[20]**
- **Relation R**
  - **Relation schema** RS:
    Set of k attributes $\{A_1,\ldots,A_k\}$
  - **Attribute** A_j: value domain $D_j = dom(A_j)$
  - **Relation:** subset of the Cartesian product over all value domains D_j
    $\mathbf{R \subseteq D_1 \times D_2 \times \ldots \times D_k,\ k \geq 1}$

Attribute

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

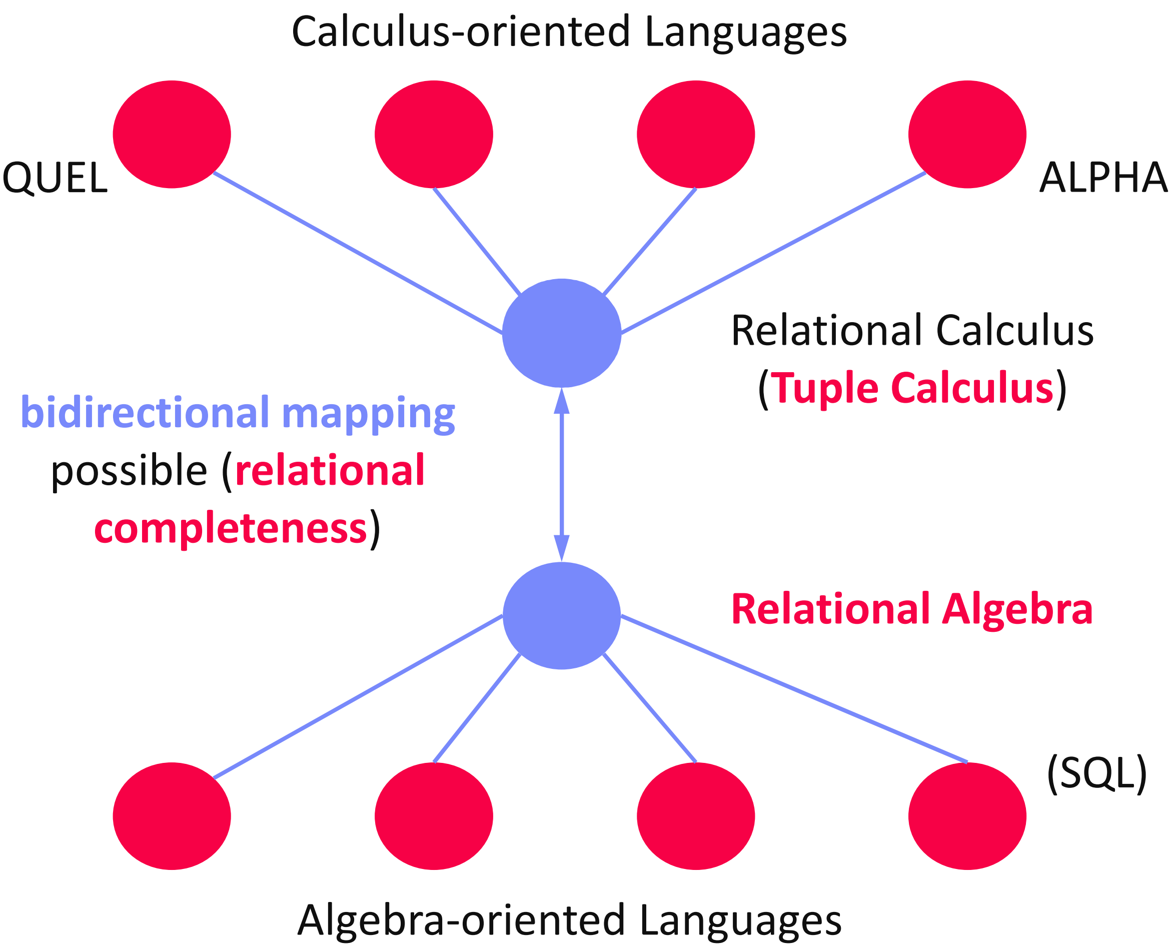
Tuple

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**



Calculus-oriented Languages

QUEL

ALPHA

Relational Calculus (**Tuple Calculus**)

bidirectional mapping possible (**relational completeness**)

Relational Algebra

(SQL)

Algebra-oriented Languages

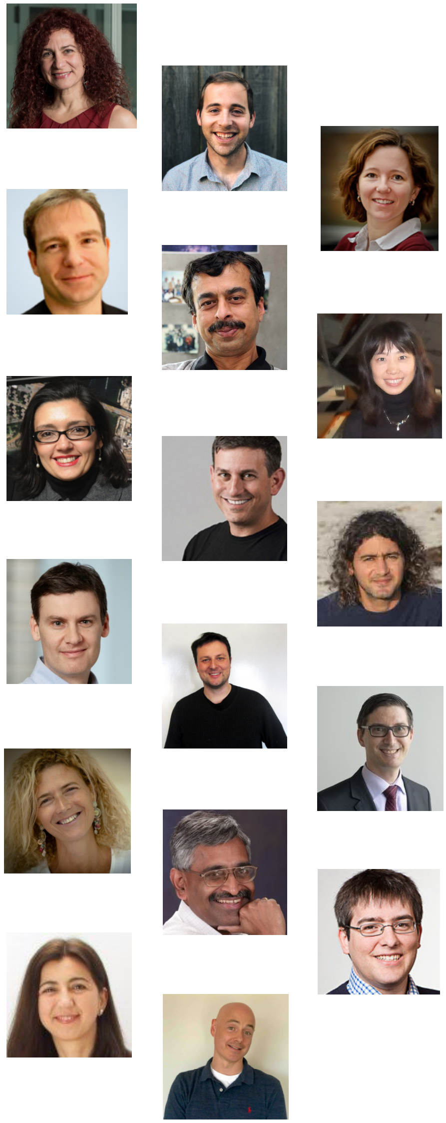
[E. F. Codd: Relational Completeness of Data Base Sublanguages. IBM Research Report RJ987, **1972**]

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.

PID	…	Affiliation	LID
102		EPFL	1
104		Stanford	
105		U Washington	3
107		CWI	2
108		MS Research	3
111		Amazon	3
113		NYU	5
115		UC Berkley	6
116		Harvard	7
117		MS Research	
118		MIT	7
120		TU Berlin	8
122		Tel Aviv University	9
123		IBM Research	10
124		TU Munich	11
126		IBM Research	10
130		Stanford	4

LID	Location
1	Lausanne, SUI
2	Amsterdam, NLD
3	Seattle, USA
4	Stanford, USA
5	New York, USA
6	Berkley, USA
7	Cambridge, USA
8	Berlin, GER
9	Tel Aviv, ISR
10	San Jose, USA
11	Munich, GER

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 duplicate tuples)
  - SQL with **bag semantics** (more flexibility and performance)
  - **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

Basic
Derived
Ext

- **Definition:** **R×S := {(r,s) | r ∈ R, s ∈ S}**
  - Set of all pairs of inputs (equivalent in **set**/**bag**)

- **Example**

PID	Firstname	Lastname	Affiliation	LID
104	Peter	Bailis	Stanford	
130	Christopher	Re	Stanford	4

×

SF Bay Area

LID	Location
4	Stanford, USA
6	Berkley, USA
10	San Jose, USA

PID	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 Derived Ext

- **Definition:** **R∪S := {x | x ∈ R ∨ x ∈ S}**
  - **Set:** set union with duplicate elimination (idempotent: **S∪S = S**)
  - **Bag:** bag union (commutative but not idempotent)
- **Example w/ set semantics**

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
Derived
Ext

- **Definition:** **R–S := {x | x ∈ R ∧ x ∉ S}** (sometimes \)
  - **Set:** set difference
  - **Bag:** element multiplicity of R minus multiplicity min(R,S)

- **Example w/ bag semantics**

Affiliation
Stanford
U Washington
MS Research
Amazon
NYU
UC Berkley
Harvard
MIT
IBM Research

–

Affiliation
Stanford
U Washington
NYU
UC Berkley
Harvard
MIT
TU Berlin
Stanford

➡

Affiliation
MS Research
Amazon
IBM Research

Difference requires **compatible schema**

Projection and Selection

Basic | Derived | Ext

- **Projection $\pi_{i_1, \ldots, i_m}(R)$**
  - **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: $\sigma_{Affiliation=‘IBM Research‘}(R)$

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

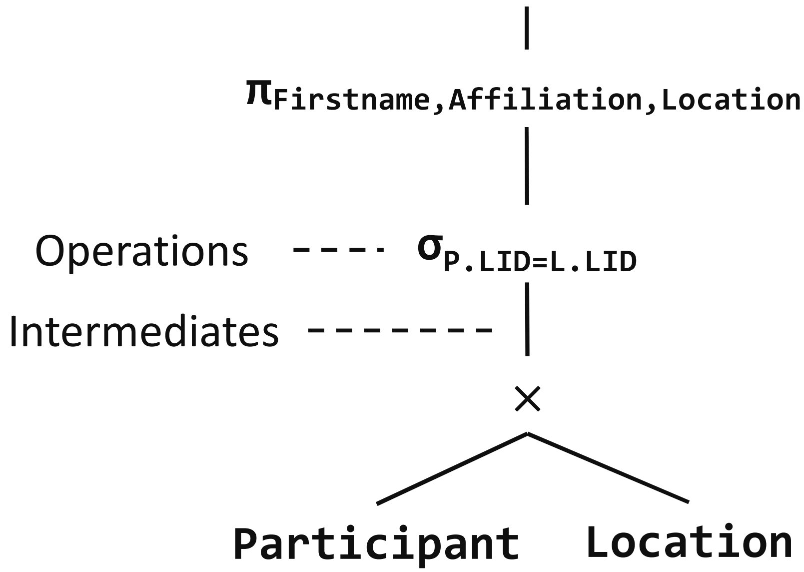
Example:

$\pi_{Affiliation}(R)$ 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

Query Trees

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



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

Intersection

Basic | Derived | Ext

- **Definition R∩S := $\boxed{R - (R - S)}$** (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

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 course**)
  - $R \div S := \{(a_1, ..., a_{r-s}) \mid \forall (b_1, ..., b_s) \in S : (a_1, ..., a_{r-s}, b_1, ..., b_s) \in R\}$

- **Example**

R

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

S

C	D
c	d
e	f

(many-to-one
set containment test)

➡

R÷S

A	B
a	b
e	d

- **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

Join

Basic
Derived
Ext

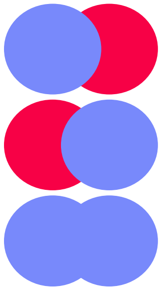
- **Definition $R \bowtie S := \pi... (\sigma_F (R \times S))$**
    - Selection of tuples (and attributes) from the catesian product $R \times S$ (equivalent in set/bag); **beware of NULLs:** do never match
    - **Theta Join:** $R \bowtie_\Theta S := \sigma_\Theta (R \times S)$; arbitrary condition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

- **Example Natural Join Participant $\bowtie$ Location**

PID	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

Types of Joins



- **Outer Joins**
  - Left outer join ⟕ (tuples of lhs, NULLs for non-existing rhs)
  - Right outer join ⟖ (tuples of rhs, NULLs for non-existing lhs)
  - Full outer join ⟗ (tuples of lhs/rhs, NULLs for non-existing lhs/rhs)
  - Example

Participant ⟕ 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 $\ltimes\ \sigma_{GER}$ (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

Basic Derived Ext

- **Duplication Elimination δ(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
Derived
Ext

- **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 $\gamma_{Affiliation,COUNT(*)}Participant$**

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

- **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)

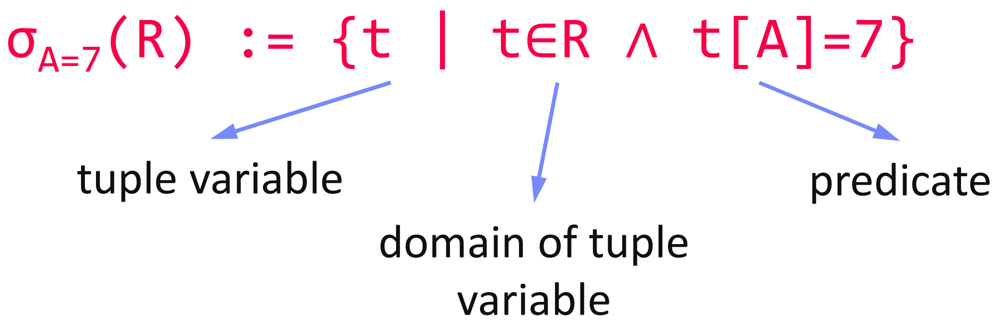
Tuple Calculus

Overview Tuple Calculus

- **Relational Calculus**
  - Tuple Calculus – tuple relational calculus: {T | p(T)} (tuple → set)
    ➔ **examples:** see definition of relational algebra
  - (Domain Calculus – domain relational calculus: attribute → 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 → result

- **Example Selection**



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

Quantifiers

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

- **Example Projection**

$$\pi_{A,B}(R) := \{\underset{\text{free variable}}{\underset{\downarrow}{t}} \mid \underset{\text{bound variable}}{\underset{\searrow}{\exists r}} \in R(t[A]=r[A] \wedge t[B]=r[B])\}$$

(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

➔ focus on query language

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

- **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-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

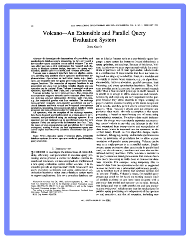
(Preview of Lecture 08 Query Processing)

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**]

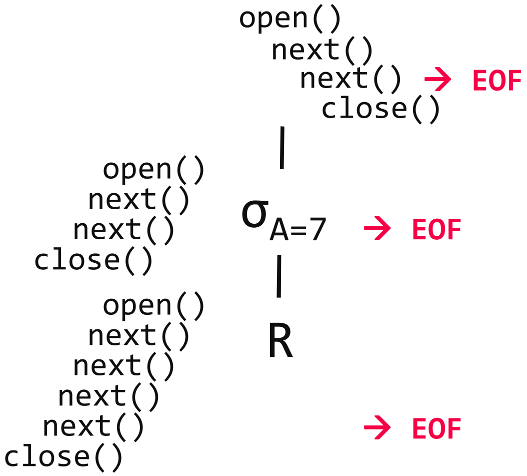


- **Example $\sigma_{A=7}(R)$**

```
void open() { R.open(); }

void close() { R.close(); }

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

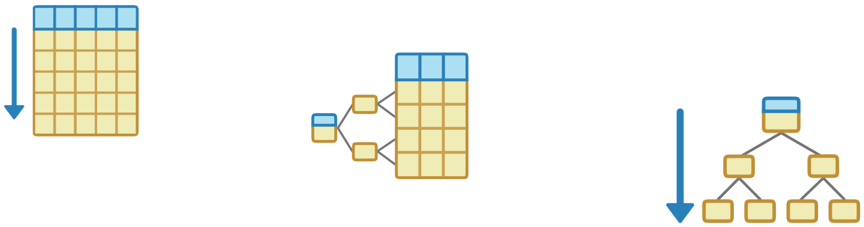


- **Blocking Operators**
  - Sorting, grouping/aggregation, build-phase of (simple) hash joins

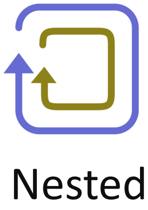
PostgreSQL: `Init()`, `GetNext()`, `ReScan()`, `MarkPos()`, `RestorePos()`, `End()`

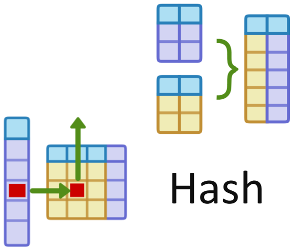
(Selected) Physical Operators in PostgreSQL

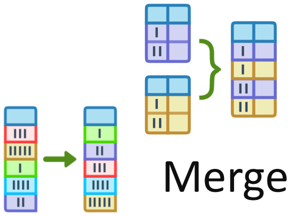
- **Physical Table Access Operators**
    - Seq Scan (table scan)
    - Index Scan
    - Index Only Scan



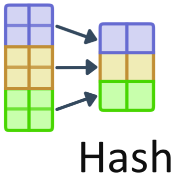
- **Physical Join Operators**
    - Nested Loop Join
    - Hash / Hash Join
    - Sort / Merge Join

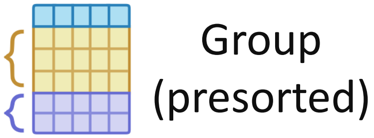






- **Physical Grouping Operators**
    - Hash Aggregate
    - Group Aggregate





[**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;`

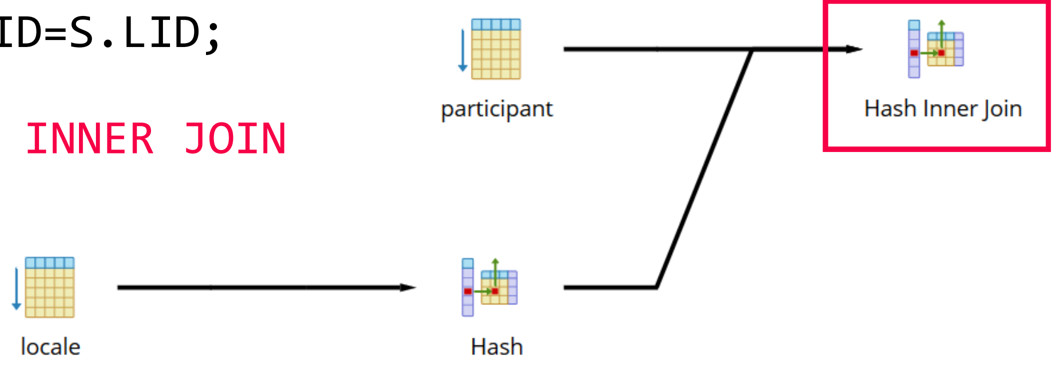
```
Hash Join (.. rows=17 width=47)
  Hash Cond:(r.lid = s.lid)
  ->  Seq Scan on participant r (.. rows=17 width=30)
  ->  Hash (.. rows=11 width=17)
        ->  Seq Scan on locale s (.. rows=11 width=17)
```

WHY?

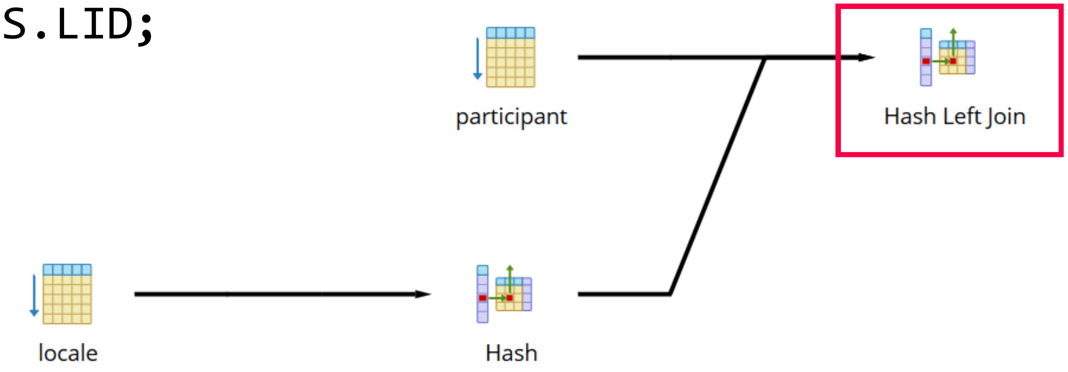
Visual EXPLAIN

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

  $\sigma_F(R \times S)$ → 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

- **Exercise 1 Reminder**
  - All background to solve tasks 1.1-1.3 since last lecture
  - Submission: **starting today**, Deadline: **Apr 02 11.59pm**

- **Next Lectures**
  - Apr 01: **05 Query Languages (SQL)**, **incl. Exercise 2**
  - Apr 08: **06 APIs (ODBC, JDBC, OR frameworks)**