ifis

Institut für Informationssysteme
Technische Universität Braunschweig

Relational Database Systems I

Wolf-Tilo Balke
Niklas Kiehne, Enrique Pinto Dominguez
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de

6 Relational Algebra

- **Motivation**
- Relational Algebra
  - basic relational algebra operations
  - Query Optimization
  - additional derived operations
- Advanced relational algebra
  - Outer Joins
  - Aggregation

6.1 Motivation

- A **data model** has three parts:
  - Structure
    - **Data structures** used to create databases representing modeled objects
  - Integrity
    - Rules expressing **constraints** placed on these data structures to ensure structural integrity
  - Manipulation
    - Operators that can be applied to the data structures, to **update** and **query** the data contained in the database

6.1 Motivation

Summary

- Last week we introduced the **relational model**
  - based on **set theory**
  - relations can be written as **tables:**

relation name

attributes

PERSON	first_name	last_name	sex
	Ash	Ketchup	m
	Ethan	Hunt	m
	Karl	Xavier	m
	March	Simpson	f
	Hermione	Stranger	f
	Steven	King	m
	Alberta	Onestone	f

tuples

…

domain values

Relational Model

Summary

- *Tables* correspond to relations, *table rows* to tuples, *table columns* to attributes or domains
  - Relations are a subset of the **Cartesian product** of its attribute domains, i.e., $R \subseteq D_1 \times \cdots \times D_n$.
- There are additional constraints on relations
  - Especially, each relation has a primary key with the ***unique key constraint***
  - There can be ***foreign key constraints***, i.e., *links* between relations

6.1 Motivation

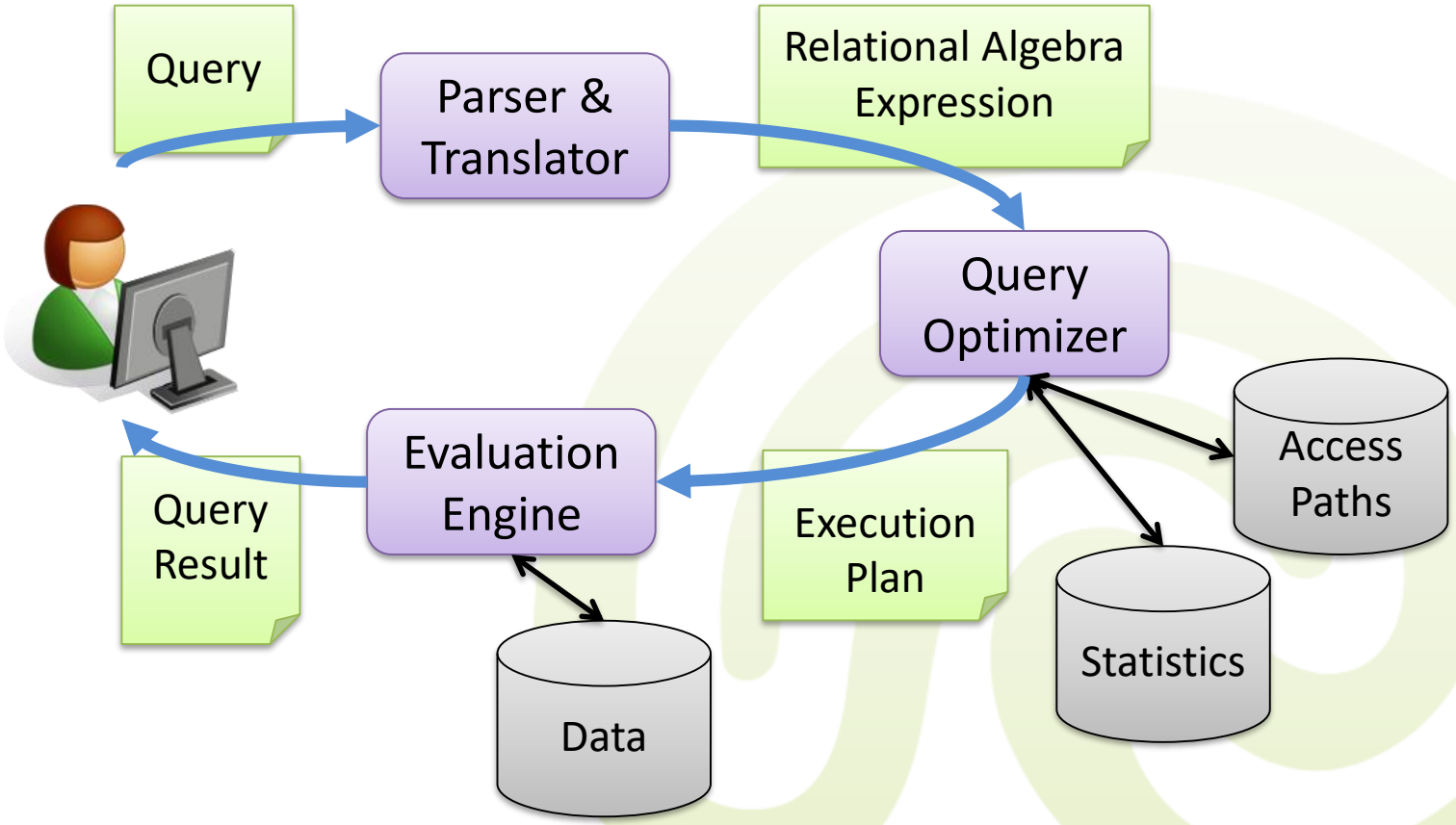
- How do you work with and query relations?
- **Relational algebra!**
  - proposed by Edgar F. Codd: *A Relational Model for Large Shared Data Banks*, Communications of the ACM, 1970
- The theoretical foundation of all relational databases
  - describes how to manipulate relations and retrieve interesting parts of available relations
  - relational algebra is mandatory for advanced tasks like **query optimization**

6.1 Motivation

- Why do we need special query languages at all?
  - *Won't conventional languages like C or Java suffice to ask and answer any computational question about relations?*
- Relational algebra is useful, **because** it is **less powerful** than C or Java.
- **2 rewards**
  - ease of programming
  - ability of the compiler to **often** produce highly optimized code

6.1 Motivation

- The first thing that happens to an SQL query is that it gets translated into relational algebra.

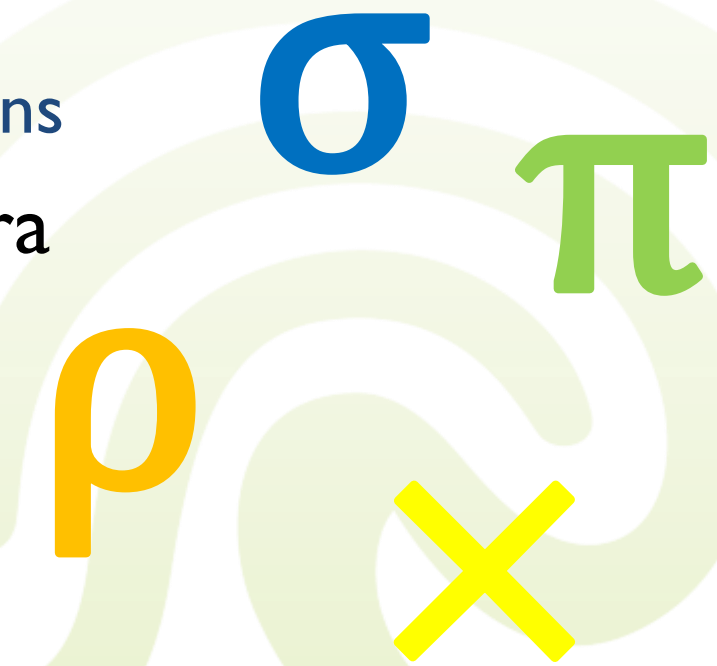


6.1 Motivation

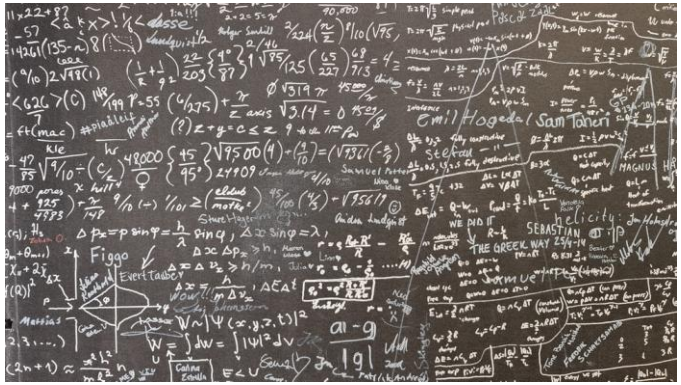
- Queries are translated into an **internal form**
  - queries posed in a **declarative** DB language
    - *what should be returned,* **not** *how the query should be processed*
  - queries can be evaluated in different ways
- Several relational algebra expressions might lead to the **same results**
  - each statement can be used for query evaluation
  - but… **different statements might also result in vastly different performance!**
- This is the area of **query optimization**, the heart of every database kernel

6 Relational Algebra

- Motivation
- **Relational Algebra**
  - basic relational algebra operations
  - Query Optimization
  - additional derived operations
- Advanced relational algebra
  - Outer Joins
  - Aggregation



6.2 Relational Algebra

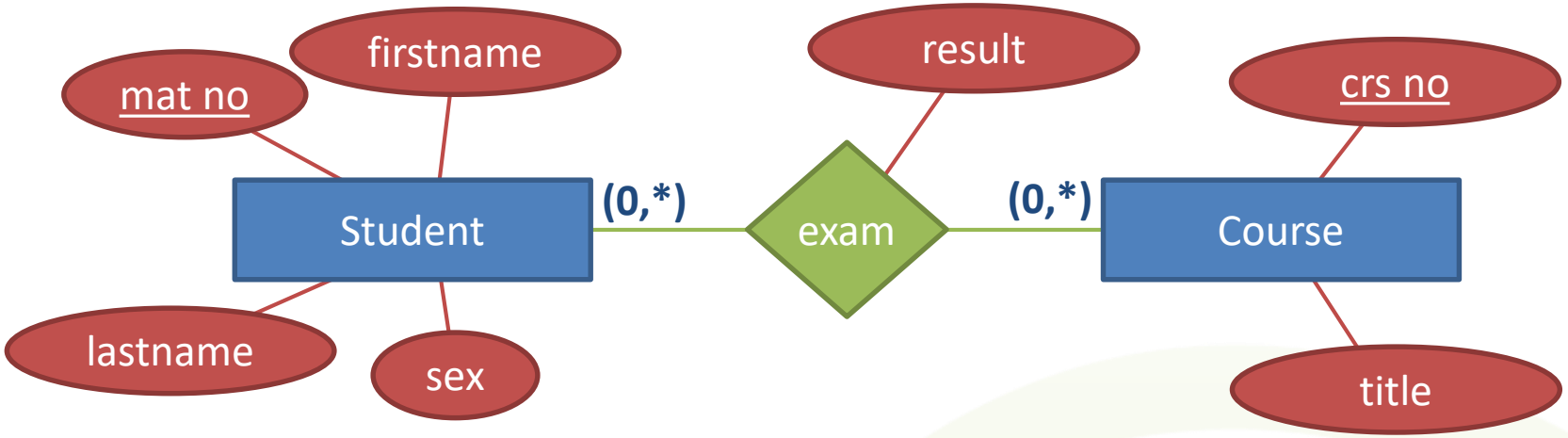


- What is an algebra after all?
  - *Study of symbols and their manipulation*
  - it consists of
    - operators and atomic operands.
  - it allows to
    - build *expressions* by applying operators to operands and / or other expressions of the algebra.
  - e.g., algebra of arithmetics
    - atomic operands: variables like *x* and constants like 15
    - operators: addition, subtraction, multiplication and division
- Relational algebra: an example of an algebra made for the relational model of databases
  - atomic operands
    - variables, that stand for relations
    - constants, which are finite relations

6.2 Relational Algebra

- Elementary operations
  - **set algebra operations**
    - Set Union ∪
    - Set Intersection ∩
    - Set Difference \
    - Cartesian Product ×
  - new **relational algebra operations**
    - Selection σ
    - Projection π
    - Renaming ρ
- Additional derived operations (for convenience)
  - all sorts of joins ⋈,⋉,⋊, ...
  - division ÷
  - …

6.2 Example Relations



Student(<u>mat_no</u>, firstname,
lastname, sex)
Course(<u>crs_no</u>, title)
exam(<u>student</u> → Student,
<u>course</u> → Course, result)

6.2 Example Relations

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
5119	Aristotle	NULL	m
6676	Mark	Salthill	m
8024	Edgar	Codd	m
9876	Adele	NULL	f

Course

crs_no	title
100	Human Emotions
101	How to sing good Songs
102	Magnificent Beards

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
9876	101	1.7
6676	100	5.0
5119	102	1.0

Student(mat_no, firstname, lastname, sex)

Course(crs_no, title)

exam(student → Student, course → Course, result)

6.2 Relational Algebra

- **Selection σ**
  - **selects** all tuples (rows) from a relation that satisfy the given **Boolean predicate** (condition)
    - *Selection* = create a new relation that contains exactly the satisfying tuples
      - **Intensional definition of** a **set of tuples!**
  - $\sigma_\varphi R := \{\ t \in R \mid t \vDash \varphi\ \}$
  - Condition (φ) is a Boolean statement given by a connection of clauses
    - Connected by $\wedge$,$\vee$ and $\neg$ (logical AND, OR, and NOT)
  - Condition clauses
    - $\langle \text{attribute} \rangle\ \theta\ \langle \text{value} \rangle$
    - $\langle \text{attribute} \rangle\ \theta\ \langle \text{attribute} \rangle$
    - $\theta \in \{=, <, \leq, \geq, >, \neq\}$

6.2 Relational Algebra

- Example

Student

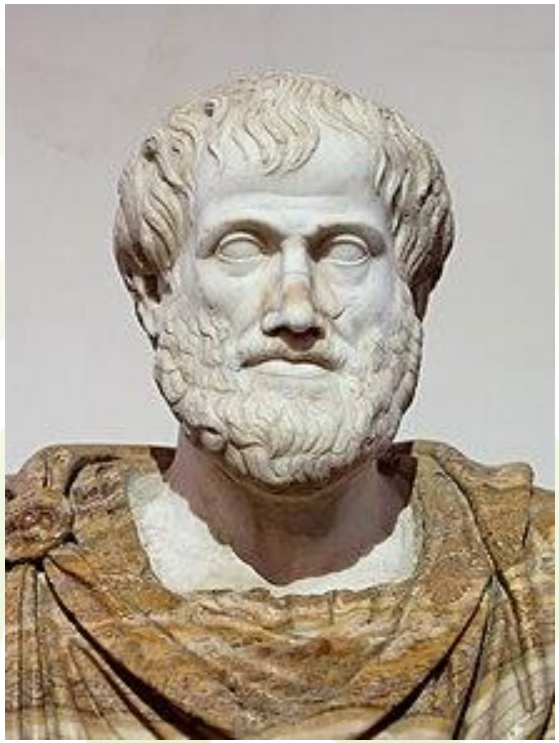
mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
5119	Aristotle	NULL	m
6676	Mark	Salthill	m
8024	Edgar	Codd	m
9876	Adele	NULL	f

$\sigma_{sex='f'}$ Student

mat_no	firstname	lastname	sex
2832	Rosa	Parks	f
9876	Adele	NULL	f

Select all female students.

$\sigma_{sex='f'}$ Student



6.2 Relational Algebra

- Example

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
9876	101	1.7
6676	100	5.0
5119	102	1.0

Select exams within course 100 that did not pass the exam.

$\sigma_{course=100 \wedge result>4.0}\ exam$

$\sigma_{course=100 \wedge result>4.0}\ exam$

student	course	result
6676	100	5.0

6.2 Relational Algebra

- **Projection π**
  - retains only specified attributes (columns)
    - again: creates a new relation with only those attribute sets which are specified within an attribute list
    - in practice (tables), if tuples are identical after projection, **duplicate tuples need to be removed**
      - This is due to the set-based nature of relational algebra
  - Let R be a relation over $R(A_1, \ldots, A_k)$ and let $A_{i_j} \in \{A_1, \ldots, A_k\}$ for $1 \leq j \leq n$.
  - $\pi_{A_{i_1},\ldots,A_{i_n}} R := \{\ t[A_{i_1}, \ldots, A_{i_n}] \mid t \in R\ \}$

All course titles.

Course

crs_no	title
100	Human Emotions
101	How to sing good Songs
102	Magnificent Beards

π_{title} Course

title
Human Emotions
How to sing good Songs
Magnificent Beards

6.2 Relational Algebra

- Of course, operations can be combined

Retrieve only the first name and last name of all female students.

$\pi_{\text{firstname, lastname}}\ \sigma_{\text{sex='f'}}\ \text{Student}$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
5119	Aristotle	NULL	m
6676	Mark	Salthill	m
8024	Edgar	Codd	m
9876	Adele	NULL	f

$\sigma_{\text{sex='f'}}\ \text{Student}$

mat_no	firstname	lastname	sex
2832	Rosa	Parks	f
9876	Adele	NULL	f

$\pi_{\text{firstname, lastname}}\ \sigma_{\text{sex='f'}}\ \text{Student}$

firstname	lastname
Rosa	Parks
Adele	NULL

6.2 Relational Algebra

- **Renaming operator ρ**
  - renames a relation and/or its attributes
  - $\rho_{S(B1, B2, ..., Bn)}\, R$ or $\rho_S R$ or $\rho_{(B1, B2, ..., Bn)}\, R$

Retrieve all course numbers contained in Course and name the resulting relation Lecture.

$\rho_{Lecture(course)}\, \pi_{crs\_no}\, Course$

Course

crs_no	title
100	Human Emotions
101	How to sing good Songs
102	Magnificent Beards

Lecture

course
100
101
102

6.2 Relational Algebra

- **Renaming operator ρ**

Select all result of course 102 from exam;
the resulting relation should be called "results"
and contain the attributes mat_no, course, and grade.

$$\rho_{\text{results(mat\_no, course, grade)}}\ \sigma_{\text{course}=102}\ \text{exam}$$

$\sigma_{\text{course}=102}$exam

student	course	result
8024	102	2.3
5119	102	1.0

results

mat_no	course	grade
8024	102	2.3
5119	102	1.0

6.2 Relational Algebra

- **Union ∪, intersection ∩,** and **set difference **
  - operators work as in set theory
    - but: operands have to be **union-compatible** (i.e. they must consist of the ***same*** attributes)
    - We define ***same*** as having the same name & same domain
  - written as $R \cup S$, $R \cap S$, and $R \setminus S$, respectively

$\sigma_{crs\_no=100}$ Course

crs_no	title
100	Human Emotions

$\sigma_{crs\_no=102}$ Course

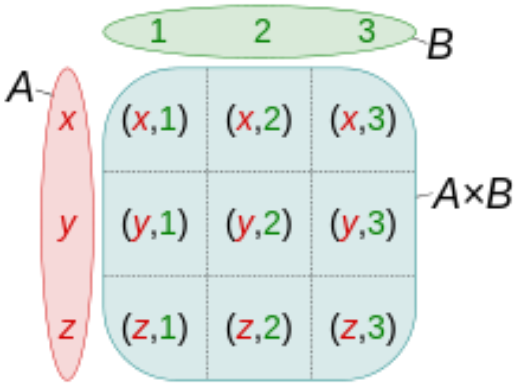
crs_no	title
102	Magnificent Beards

$\sigma_{crs\_no=100}$ Course ∪ $\sigma_{crs\_no=102}$ Course

crs_no	title
100	Human Emotions
102	Magnificent Beards

6.2 Relational Algebra

- **Cartesian product ×**
  - written as $R \times S$
    - also called cross product
  - creates a new relation by combining each tuple of the first relation with every tuple of the second relation
    - attribute set = all attributes of R plus all attributes of S
    - the resulting relation contains exactly $|R| \cdot |S|$ tuples



6.2 Relational Algebra

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
…			

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
…		

Student × exam

mat_no	firstname	lastname	sex	student	course	result
1005	Elon	Musk	m	2832	100	1.3
1005	Elon	Musk	m	1005	100	3.7
1005	Elon	Musk	m	8024	102	2.3
…						
2832	Rosa	Parks	f	2832	100	1.3
2832	Rosa	Parks	f	1005	100	3.7
…						

6.2 Relational Algebra

$\pi_{\text{lastname, title, result}}\, \sigma_{\text{mat\_no=student} \,\wedge\, \text{course=crs\_no}}\, (\text{Student} \times \text{exam} \times \text{Course})$

lastname	title	result
Musk	Human Emotions	3.7
Parks	Human Emotions	1.3
NULL	Magnificent Beards	1.0
…		

← Aristotle

Useful!

- This type of expression is very important!
  - Cartesian product, followed by selections/projections
  - selections/projections involve different source relations
  - this kind of query is called a ***join***

6.2 Relational Algebra

- **Theta join $\bowtie_\theta$** (θ is a boolean condition)
  - sometimes also called **inner join** or just **join**
  - creates a new relation by combining related tuples from different source relations
  - written as $R \bowtie_{\langle condition \rangle} S$ or $\sigma_{\langle condition \rangle}(R \times S)$
  - Joins can be used to join related relations
    - i.e., related in an ER model, and now linked via foreign keys

$$\pi_{\text{lastname, title, result}}(\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam} \bowtie_{\text{course=crs\_no}} \text{Course})$$

$$=$$

$$\pi_{\text{lastname, title, result}}\, \sigma_{\text{mat\_no=student} \wedge \text{course=crs\_no}}(\text{Student} \times \text{exam} \times \text{Course})$$

6.2 Relational Algebra

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
…			

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
…		

Course

crs_no	title
100	Human Emotions
101	How to sing good Songs
102	Magnificent Beards

$$\text{Student} \bowtie_{\text{mat\_no}=\text{student}} \text{exam} \bowtie_{\text{course}=\text{crs\_no}} \text{Course}$$

6.2 Relational Algebra

- **Equi-join $\bowtie_{\langle condition \rangle}$**
  - joins two relations only using **equality conditions**
  - $R \bowtie_{\langle condition \rangle} S$
  - $\langle condition \rangle$ may only contain equality statements between attributes ($A_1 = A_2$)
    - a special case of the theta join
    - Most used join operator

$$\pi_{\text{lastname, title, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam} \bowtie_{\text{course=crs\_no}} \text{Course})$$

$$=$$

$$\pi_{\text{lastname, title, result}} \sigma_{\text{mat\_no=student} \wedge \text{course=crs\_no}} (\text{Student} \times \text{exam} \times \text{Course})$$

6.2 Relational Algebra

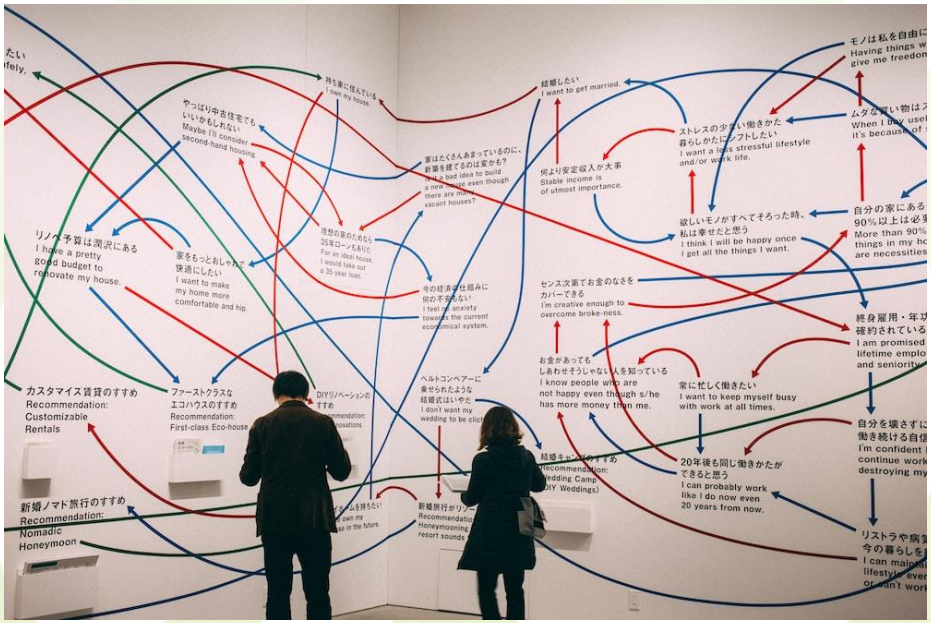
- **Natural join ⋈**
  - a special case of the equi-join
  - $R \bowtie_{\langle attributeList \rangle} S$
  - implicit join condition
    - each attribute in ⟨attributeList⟩ must be contained in **both** source relations
    - for each of these attributes, an equality condition is created
    - all these conditions are connected by logical AND
    - If no attributes are explicitly stated, all attributes with equal names are implicitly used

6.2 Query Optimization

Detour

- Relational algebra usually allows for several equivalent evaluation plans
  - respective execution costs may strongly differ
    - e.g., in memory space, response time, etc.

- Idea: Find the best plan *before* evaluating the query



6.2 Query Optimization

Detour

- Example

Select all results better than 1.7, their student's name and course title.

Student

mat_no	firstname	lastname	size
3519	Bilbo	Baggins	103
1473	Samwise	Gamgee	114
2308	Meriadoc	Brandybuck	135
1337	Erna	Broosh	86
2158	Frodo	Baggins	111
1104	Peregrin	Took	142
2480	Sméagol	NULL	98

4 Byte 30 Byte 30 Byte 4 Byte

Course

4 Byte 30 Byte

crs_no	title
41	Cooking rabbits
40	Destroying rings
42	Flying eagles

exam

4 Byte 4 Byte 8 Byte

student	course	result
1473	41	1.0
3519	40	3.3
2480	40	1.7
1337	42	1.3
2480	41	4.0
2158	40	2.3

6.2 Query Optimization

Detour

- Example

Select all results better than 1.7, their student's name and course title.

$$\pi_{\text{lastname, result, title}}\ \sigma_{\text{result}\leq 1.3\ \wedge\ \text{course}=\text{crs\_no}\ \wedge\ \text{mat\_no}=\text{student}}\ (\text{Course} \times \text{Student} \times \text{exam})$$

Student

mat_no	firstname	lastname	size
3519	Bilbo	Baggins	103
1473	Samwise	Gamgee	114
2308	Meriadoc	Brandybuck	135
1337	Erna	Broosh	86
2158	Frodo	Baggins	111
1104	Peregrin	Took	142
2480	Sméagol	NULL	98

Course (4 Byte, 30 Byte)

crs_no	title
41	Cooking rabbits
40	Destroying rings
42	Flying eagles

exam (4 Byte, 4 Byte, 8 Byte)

student	course	result
1473	41	1.0
3519	40	3.3
2480	40	1.7
1337	42	1.3
2480	41	4.0
2158	40	2.3

6.2 Query Optimization

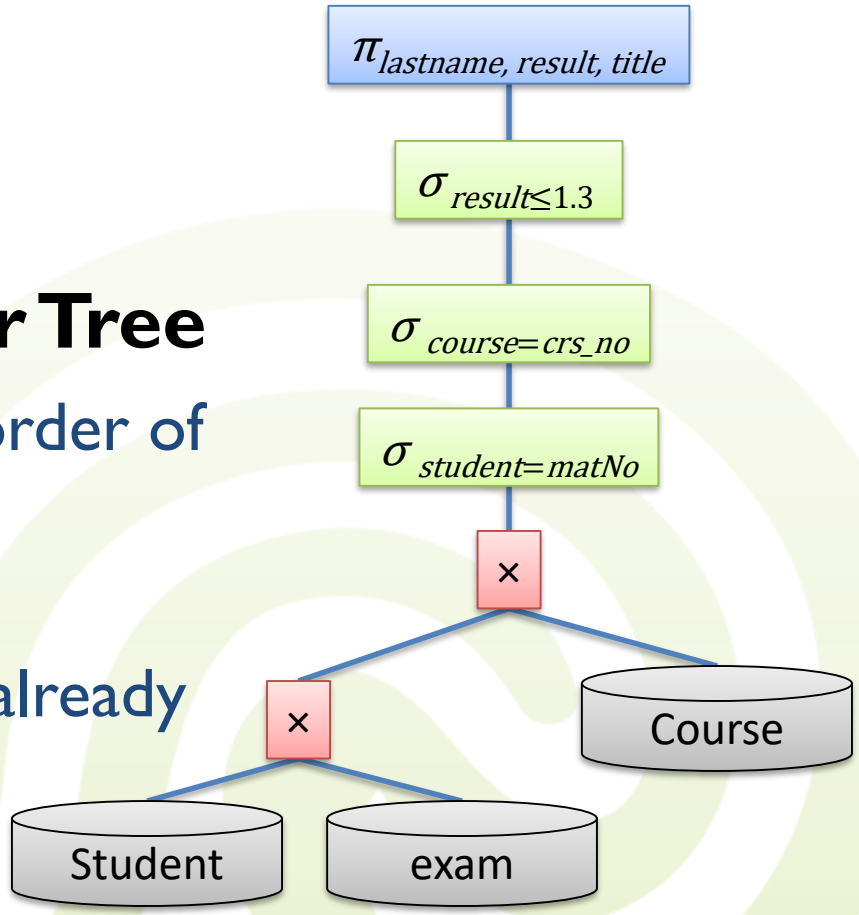
Detour

$\pi_{\text{lastname, result, title}}$

$\sigma_{\text{result} \leq 1.3 \wedge \text{course} = \text{crs\_no} \wedge \text{mat\_no} = \text{student}}$

$(\text{Course} \times \text{Student} \times \text{exam})$

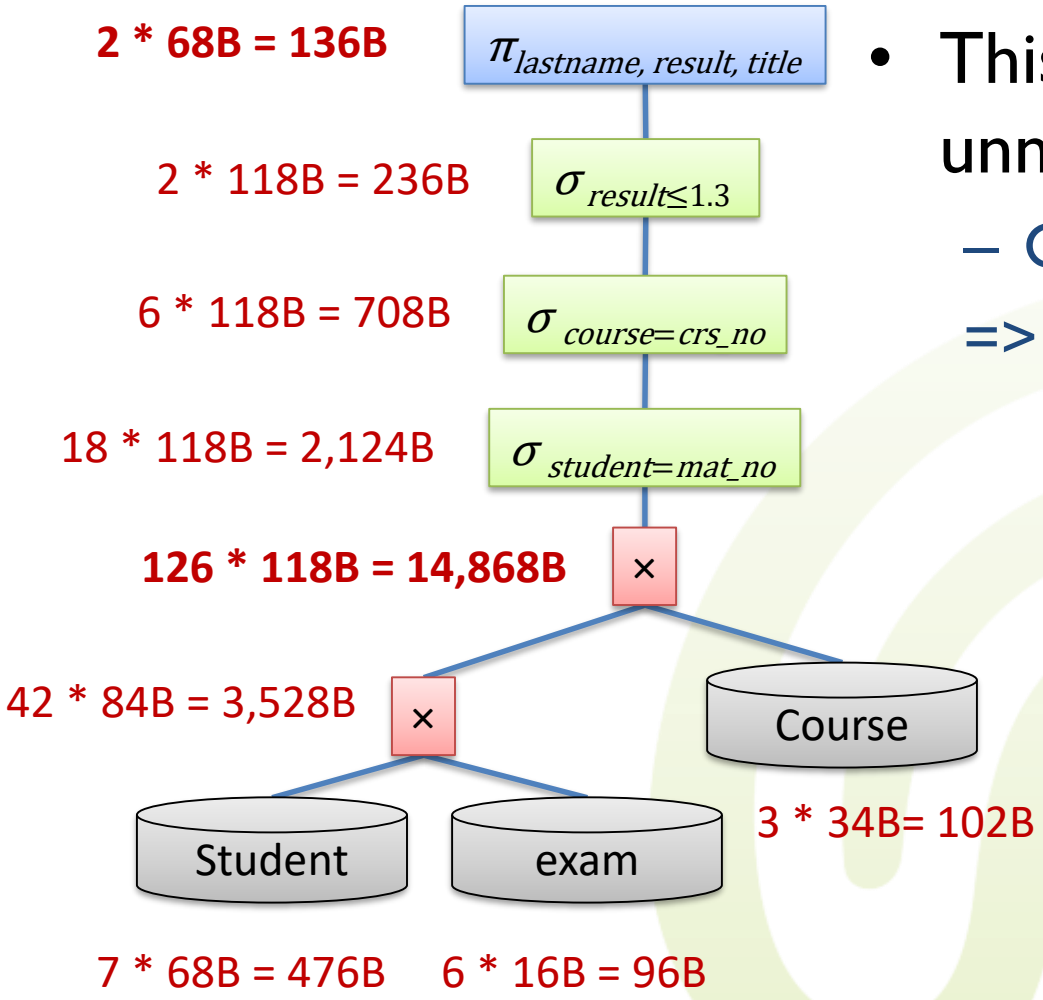
- Create Canonical **Operator Tree**
  - operator tree visualized the order of primitive functions
  - note: Illustration is not really canonical tree as selection is already split in three parts



6.2 Query Optimization

Detour

How much **space** is needed for the intermediate results?



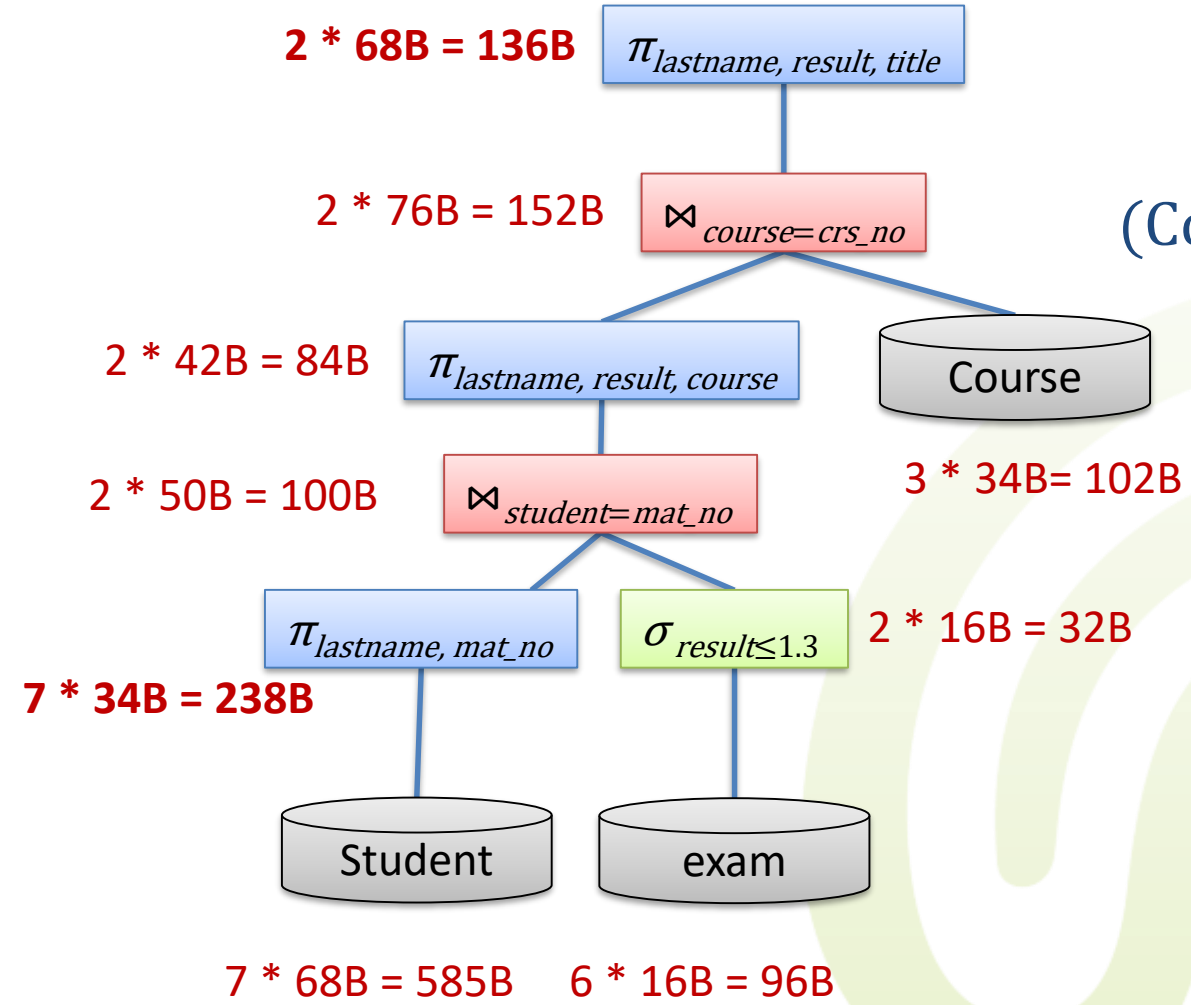
- This seems like a lot of unnecessary work!
  - Can't we optimize this?

  => yes, we can!

6.2 Query Optimization

Detour

- Optimized Operator Tree



$\pi_{\text{lastname, result, title}}$

$\sigma_{\text{result}\leq 1.3\ \wedge\ \text{course} = \text{crs\_no}\ \wedge}$

$\text{mat\_no} = \text{student}$

$(\text{Course} \times \text{Student} \times \text{exam})$

$\equiv$

$\pi_{\text{lastname, result, title}}$

$(\text{Course}$

$\bowtie_{\text{course=crs\_no}}$

$\pi_{\text{lastname, result, course}}$

$(\pi_{\text{lastname, mat\_no}}\ \text{Student}$

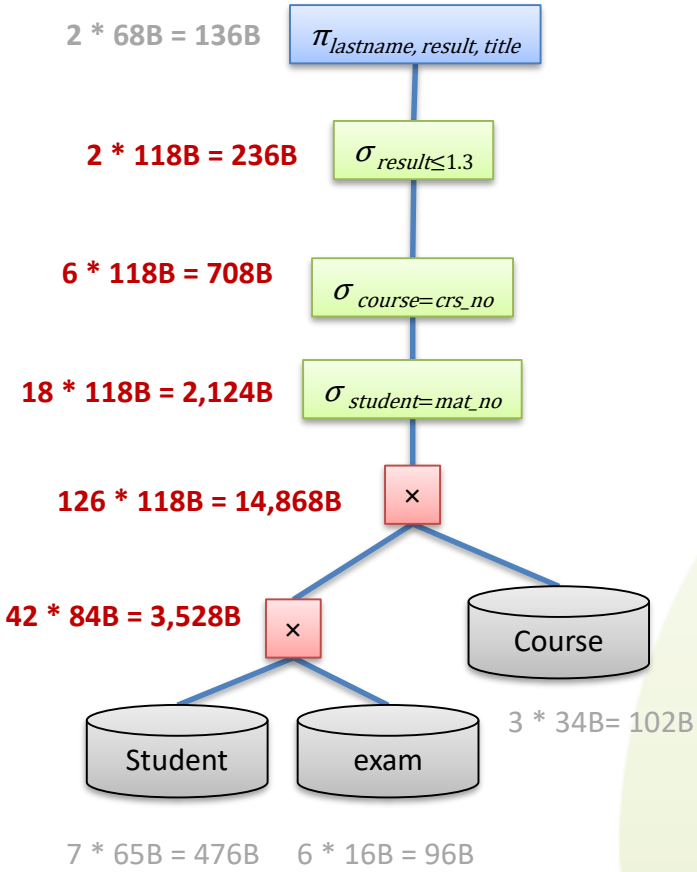
$\bowtie_{\text{mat\_no=student}}$

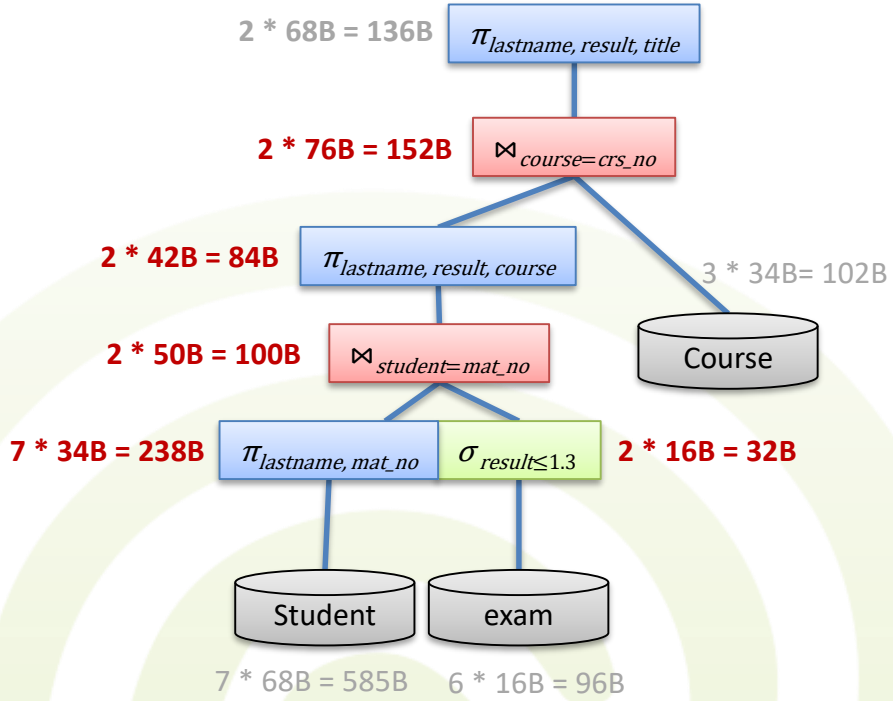
$\sigma_{\text{result}\leq 1.3}\ \text{exam}))$

6.2 Query Optimization

Detour

- Comparison of intermediate results





Intermediate result sets – summed up sizes: **21464 B** versus **606 B**

6.2 Relational Algebra

- **Left semi-join ⋉** and **right semi-join ⋊**
  - combination of a theta-join and projection
    - $R \ltimes_{\langle \text{condition} \rangle} S = \pi_{(\text{list of all attributes in } R)} (R \bowtie_{\langle \text{condition} \rangle} S)$
    - $R \rtimes_{\langle \text{condition} \rangle} S = \pi_{(\text{list of all attributes in } S)} (R \bowtie_{\langle \text{condition} \rangle} S)$
  - creates a copy of R (or S) while removing all those tuples that do not have a join partner
    - This is a filtering operation with a dictionary lookup

6.2 Relational Algebra

Left semi-join:

$$\text{Student} \ltimes_{\text{mat\_no=student}} \text{exam}$$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
9876	101	1.7

All hard-trying students (those who did exams):

$$\text{Student} \ltimes_{\text{matNr=student}} \text{exam}$$

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f

6.2 Relational Algebra

Division ÷

- written as $R \div S$
- S contains only attributes that are also present in R
- division restricts R in terms of attributes and tuples
  - result's attributes = those in R but not in S
  - result's tuples = those in R such that **every** tuple in S is a join partner
  - useful for queries like *Find the sailors who have reserved all boats.*
- more formally:
  - let A_R={attributes of R}; A_S ={attributes of S}; $A_S \subseteq A_R$
  - $A = A_R \setminus A_S$
  - $R \div S \equiv \pi_A R \setminus \pi_A \left((\pi_A (R) \times S) \setminus R \right)$
- why the name division? because $(R \times S) \div S = R$.
  - **but:** $(R \div S) \times S = R$ is not true in general

6.2 Relational Algebra

Division:

$SC = \rho_{SC}(\pi_{mat\_no,\ lastname,\ course}(Student \bowtie_{mat\_no=student} exam))$

mat_no	lastname	course
1005	Musk	100
1005	Musk	102
2832	Parks	100
4512	Pavarotti	102
4512	Pavarotti	100
4512	Pavarotti	101
6676	Salthill	103
6676	Salthill	100

$CCK = \rho_{CCK}(\pi_{course}\sigma_{lastname='Musk'} SC)$

course
100
102

$SC \div CCK$

mat_no	lastname
1005	Musk
4512	Pavarotti

Result contains all those students who took the same courses as Elon Musk (and possibly some more).

6.2 Relational Algebra

- Alternative definition
  - Read division as a *for all* statement
  - Given relations R and S based on schemas
    - $R(A_1, \dots, A_n, B_1, \dots, B_m)$ and $S(B_1, \dots, B_m)$
  - $\boldsymbol{R \div S} := \{ \boldsymbol{t} \in \boldsymbol{dom(A_1) \times \cdots \times dom(A_n)} \mid \forall \boldsymbol{u} \in \boldsymbol{S} : \boldsymbol{tu} \in \boldsymbol{R} \}$
    - tu means plain concatenation of t and u
      - e.g., $t =< 1,2 >, u =< 3,4 >, \ tu =< 1,2,3,4 >$

6.2 Relational Algebra

- Conflicting attribute names
  - In the previous examples, all relations had different attribute names
    - What to do, if not?
  - Simple case:
    - Use qualified attribute names:
      - **Student**(<u>matno</u>, name)
      - **Thesis**(<u>id</u>, student → Student, name)
      - $\pi_{\text{Student.name, Thesis.name}}(\text{Student} \bowtie_{\text{matno=student}} \text{Thesis})$
  - General case:
    - Use renaming:
      - **Person**(<u>id</u>, name, supervisor → Person)
      - $\text{Person} \bowtie_{\text{supervisor=sid}} (\rho_{\text{Supervisor(sid, sname, ssup)}} \text{Person})$

6.2 Relational Algebra

- Operator Precedence
  - **unary** operators are applied first **(σ, π, ρ, ℱ)**
  - **cross product** and **joins** are applied afterwards **(×, ⋈, …)**
  - followed by **union** und **set minus (∪, \)**
  - last step is **intersection (∩)**

$$(\text{exam} \bowtie_{\text{course}=\text{crs\_no}} (\sigma_{\text{crs\_no}=101}\ \text{Course})) \cup (\text{exam} \bowtie_{\text{course}=\text{crs\_no}} (\sigma_{\text{crs\_no}=100}\ \text{Course})) = \text{exam} \bowtie_{\text{course}=\text{crs\_no}} \sigma_{\text{crs\_no}=101}\ \text{Course} \cup \text{exam} \bowtie_{\text{course}=\text{crs\_no}} \sigma_{\text{crs\_no}=100}\ \text{Course}$$

$$\underbrace{\pi_{\text{lastname, result}}\ \text{Student}} \bowtie_{\text{mat\_no}=\text{student}} \text{exam}$$

projection is applied first!

6.2 Summary

- **Basic relational algebra**
  - Selection σ
  - Projection π
  - Renaming ρ
  - Union $\cup$, set difference $\setminus$
  - Cartesian product $\times$
  - Intersection $\cap$
- **Extended relational algebra**
  - Theta-join $\bowtie_{\theta}$
  - Equi-join $\bowtie_{<=\text{-cond}>}$
  - Natural join $\bowtie_{<\text{attributeList}>}$
  - Left semi-join $\ltimes$ and right semi-join $\rtimes$
  - Division $\div$

6.3 What does this do?

Exercise

Movie(<u>id</u>, title, year, type, remark)

produced_in(<u>movie</u> → Movie, <u>country</u> → Country)

Country(<u>name</u>, residents)

$$\pi_{country,\ title}\Big(\text{Country} \bowtie_{name=country} \text{produced\_in} \bowtie_{movie=id} \sigma_{year=1893\ \wedge\ type="cinema"}\ \text{Movie}\Big)$$

From which countries are the cinema movies of the year 1893 and what are their names?

6.3 What does this do?

Exercise

Person(id, name, sex)

plays(person → Person, movie → Movie, role)

Movie(id, title, year, type)

$$\pi_{name}\Big(\sigma_{title = \text{"Star Wars"} \wedge type=\text{"cinema"}} Movie \bowtie_{Movie.id=movie} \sigma_{role=\text{„Diplomat"}} plays \bowtie_{person=Person.id} \sigma_{sex=\text{"female"}} Person\Big)$$

Which female actors from 'Star Wars' cinema movies played a diplomat?

6.3 What does this do?

Exercise

Person(id, name, sex)

plays(person → Person, movie → Movie, role)

Movie(id, title, year, type)

has_genre(movie → Movie, genre → Genre)

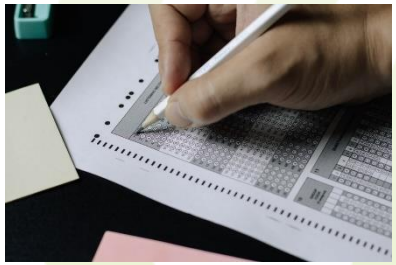
Genre(name, description)

$$\pi_{name}(\pi_{id,\ name}(Person \bowtie_{id=person} \sigma_{role="postman"} plays) \setminus \pi_{id,\ name}(Person \bowtie_{id=person} plays \bowtie_{plays.movie=has\_genre.movie} \sigma_{genre="Western"} has\_genre))$$

Which actors have played a 'postman', but never participated in a 'Western'?

6 Relational Algebra

- Motivation
- Relational Algebra
  - basic relational algebra operations
  - Query Optimization
  - additional derived operations
- **Advanced relational algebra**
  - Outer Joins
  - Aggregation



6.3 Advanced Relational Algebra

- **Advanced** relational algebra
  - **Left outer join ⟕,**
  - **Right outer join ⟖,**
  - **Full outer join ⟗,**
  - **Aggregation ℱ**

- These operations **cannot be expressed** with basic relational algebra

6.3 Advanced Relational Algebra

- **Left outer join ⟕** and **right outer join ⟖**
  - Theta-joins and its specializations join exactly those tuples that **satisfy the join condition**
    - tuples **without a matching join partner** are **eliminated** and will not appear in the result
    - all information about non-matching tuples gets lost
  - outer joins allow to keep **all tuples of a relation**
    - **padding with NULL values** if there is no join partner
    - Left outer join ⟕: Keeps all tuples of the left relation
    - Right outer join ⟖: Keeps all tuples of the right relation
    - Full outer join ⟗: Keeps all tuples of both relations

6.3 Advanced Relational Algebra

Example: List students and their exam results

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
8024	Edgar	Codd	m

exam

student	course	result
2832	100	1.3
1005	102	3.7
2832	101	2.7
9876	101	1.7

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

lastname	course	result
Parks	100	1.3
Parks	101	2.7
Musk	102	3.7

Luciano Pavarotti and Edgar Codd are lost because they didn't take any exams!
Also, exam information on student 9876 disappears…

6.3 Advanced Relational Algebra

Left outer join: List students and their exam results

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
8024	Edgar	Codd	m

exam

student	course	result
2832	100	1.3
1005	102	3.7
2832	101	2.7
9876	101	1.7

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

lastname	course	result
Parks	100	1.3
Parks	101	2.7
Musk	102	2.0
Pavarotti	NULL	NULL
Codd	NULL	NULL

All student names with course and result if present.

6.3 Advanced Relational Algebra

Right outer join: List exams and their participants.

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
8024	Edgar	Codd	m

exam

student	course	result
2832	100	1.3
1005	102	3.7
2832	101	2.7
9876	101	1.7

$$\pi_{\text{lastname, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam})$$

lastname	course	result
Parks	100	1.3
Musk	102	3.7
Parks	101	2.7
NULL	101	1.7

All exam results with student names if known.

6.3 Advanced Relational Algebra

Full outer join: Combination of left and right outer joins.

$$\pi_{\text{lastname, course, result}} (\text{Student} ⟗_{\text{mat\_no=student}} \text{exam})$$

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
8024	Edgar	Codd	m

exam

student	course	result
2832	100	1.3
1005	102	3.7
2832	101	2.7
9876	101	1.7

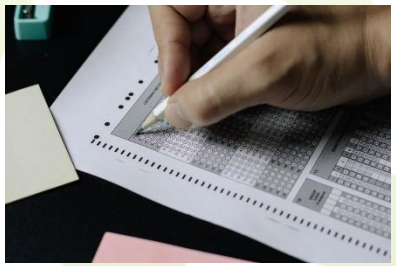
$$\pi_{\text{lastname, course, result}} (\text{Student} ⟗_{\text{mat\_no=student}} \text{exam})$$

lastname	course	result
Parks	100	1.3
Parks	101	2.7
Musk	102	3.7
Pavarotti	NULL	NULL
NULL	101	1.7
Codd	NULL	NULL

All student names with course and result if present
AND
all exam result with student names if known.

6.3 Aggregation

- **Aggregation operator:**
  Typically used in simple statistical computations
  - merges tuples into **groups**
  - **computes** (simple) statistics **for each group**
- **Examples**
  - *Compute the average exam score.*
  - *For each student, count the total number of exams he/she has taken so far.*
  - *Find the highest exam score ever.*

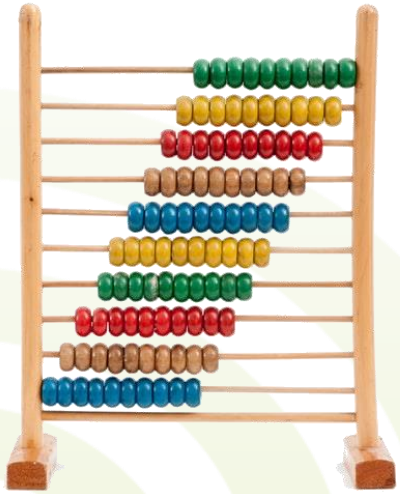


6.3 Aggregation

- Written as $_{\langle \text{grouping attributes} \rangle}\mathfrak{F}_{\langle \text{function list} \rangle}\ R$
  - $\mathfrak{F}$ is called *script F*
- Creates a new relation
  - all tuples in *R* that have identical values with respect to all grouping attributes, are grouped together
  - all functions in the function list are applied to each group
  - for each group, a new tuple is created, which contains the result values of all the functions
  - the schema of the resulting relation looks as follows:
    - the grouping attributes and, for each function, one new attribute
    - all other attributes of the old relation are lost
- Example: $_{course}\mathfrak{F}_{average(result)}\ exam$

6.3 Aggregation

- **Attention**
  - **Within groups, no duplicates are eliminated**
- Some available functions
  - **Sum**(attribute)
    - sum of all non-NULL values of attribute
  - **Average**(attribute)
    - mean of all non-NULL values of attribute
  - **Maximum**(attribute)
    - maximum value of all non-NULL values of attribute
  - **Minimum**(attribute)
    - minimum value of all non-NULL values of attribute
  - **Count**(attribute)
    - number of tuples having a non-NULL values of attribute
  - **Count(*)**
    - number of tuples



6.3 Aggregation

- Example (without grouping)
  - if there are no grouping attributes, the whole input relation is treated as **one group**

$$\rho_{\text{Without\_Groups}(\text{sum, avg\_result, min\_result, max\_result})}\left(\Im_{\text{sum(student), average(result), min(result), max(result)}}\ \text{exam}\right)$$

exam

student	course	result
2832	100	1.3
1005	100	3.7
8024	102	2.3
9876	101	1.7
6676	100	5.0

Without_Groups

sum	avg_result	min_result	max_result
28413	2.8	1.3	5.0

6.3 Aggregation

- Example (with grouping)

For each course, count results and compute the average score.

$$\rho_{\text{With\_Grouping (course, avg\_result, \#result)}} (\; {}_{\text{course}}\mathfrak{F}_{\text{average(result), count(result)}} \; \text{exam})$$

exam

student	course	result
2832	100	1.3
8024	102	2.3
9876	101	1.7
1005	100	3.7
5119	102	1.0

With_Grouping

course	avg_result	#result
100	2.5	2
101	1.7	1
102	1.65	2

6.3 Aggregation

- Example

  For each student, count the exams and compute average result.

$$\rho_{\text{Overview(mat\_no, \#exams, avg\_result)}} \Big(\text{mat\_no}\,\mathfrak{F}_{\text{count(course), avg(result)}} \big(\pi_{\text{mat\_no, course, result}} (\text{Student} \bowtie_{\text{mat\_no=student}} \text{exam}) \big) \Big)$$

exam

student	course	result
2832	100	1.3
1005	102	3.7
2832	101	2.7
9876	101	1.7

Overview

mat_no	#exams	avg_result
2832	2	2.0
1005	1	3.7
4512	0	NULL
8024	0	NULL

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
8024	Edgar	Codd	m

6.3 Advanced Relational Algebra

- **Subqueries**: Conditions used in select or join operators compare a tuple's attribute to another value or attribute
  - Condition clauses
    - ⟨attribute⟩ θ ⟨value⟩
    - ⟨attribute⟩ θ ⟨attribute⟩
    - $\theta \in \{=, <, \leq, \geq, >, \neq\}$
  - Sometimes, we need a dynamic value
  - **Convention** for advanced relational algebra:
    - We can treat a relation with a single attribute and a single row as a value (we call that a subquery)
  - Example: Select those tuples from exam which have the overall average result as a result
    - $\sigma_{result=\Im_{avg(result)}exam}exam$

6.4 Guidelines

- Writing complex Queries
  - Use intermediate results:

$$stud\_ex = \rho_{stud\_ex(stud,\ num\_ex)}\ (_{student}\Im_{count(course)}\ exam)$$

$$\sigma_{stud\_ex.num\_ex>5}\ stud\_ex$$

$$=$$

$$\sigma_{stud\_ex.num\_ex>5}\ (\rho_{stud\_ex(stud,\ num\_ex)}\ (_{student}\Im_{count(course)}\ exam))$$

  - Use proper indentation... not something like:

$$\pi_{name}(Student \ltimes_{mat\_no}\ (\pi_{mat\_no}\ \sigma_{\#exams>5\ \wedge\ avg\_result>2.7}\ \rho_{Overview(mat\_no,\ \#exams,\ avg\_result)}(_{mat\_no}\Im_{count(course),\ avg(result)}\ \pi_{mat\_no,\ course,\ result}\ (Student \rtimes_{mat\_no=student}\ exam))))$$

6.4 Exercise

Exercise

Student

mat_no	firstname	lastname	sex
1005	Elon	Musk	m
2832	Rosa	Parks	f
4512	Luciano	Pavarotti	m
5119	Aristotle	NULL	m
6676	Mark	Salthill	m
8024	Edgar	Codd	m
9876	Adele	NULL	f

Course

crs_no	title
100	Human Emotions
101	How to sing good Songs
102	Magnificent Beards
103	Secret Identities

Exam

mat_no	course	result
9876	100	1.7
2832	101	2.3
2832	102	4.0
1005	101	3.3
1005	100	3.7
1005	103	1.3
6676	102	3.7
5119	101	1.7
5119	102	1.0
5119	103	3.0

Student(<u>mat_no</u>, firstname, lastname, sex)
Course(<u>crs_no</u>, title)
Exam(<u>mat_no</u> → Student, <u>course</u> → Course, result)

6.4 Exercise

Exercise

- **List the student(s) who took the most exams.** (by matno and firstname)
  - Decompose the query in multiple parts:
    - How many exams did each student take?
    - What is the maximum number of exams a student took?
    - Who took as many exams as the maximum?

Student(<u>mat_no</u>, firstname, lastname, sex)
Course(<u>crs_no</u>, title)
Exam(<u>mat_no</u> → Student, <u>course</u> → Course, result)

6.4 Exercise

Exercise

– How many exams did each student take?

- $StudentExams =$

$$\rho_{StudentExams(mat\_no,firstname,exam\_count)}$$

$$mat\_no,firstname\ \mathfrak{F}\ count(course)$$

$$Student \bowtie_{Student.mat\_no=Exam.mat\_no} Exam$$

Student(<u>mat_no</u>, firstname, lastname, sex)
Course(<u>crs_no</u>, title)
Exam(<u>mat_no</u> → Student, <u>course</u> → Course, result)

StudentExams

mat_no	firstname	exam_count
1005	Elon	3
2832	Rosa	2
4512	Luciano	0
5119	Aristotle	3
6676	Mark	1
8024	Edgar	0
9876	Adele	1

6.4 Exercise

Exercise

– **What is the maximum number of exams a student took?**

- $Max\_Exams = \mathfrak{F}_{max(exam\_count)}\ StudentExams$

max_exam_count
3

– **List the student(s) who took the most exams.**

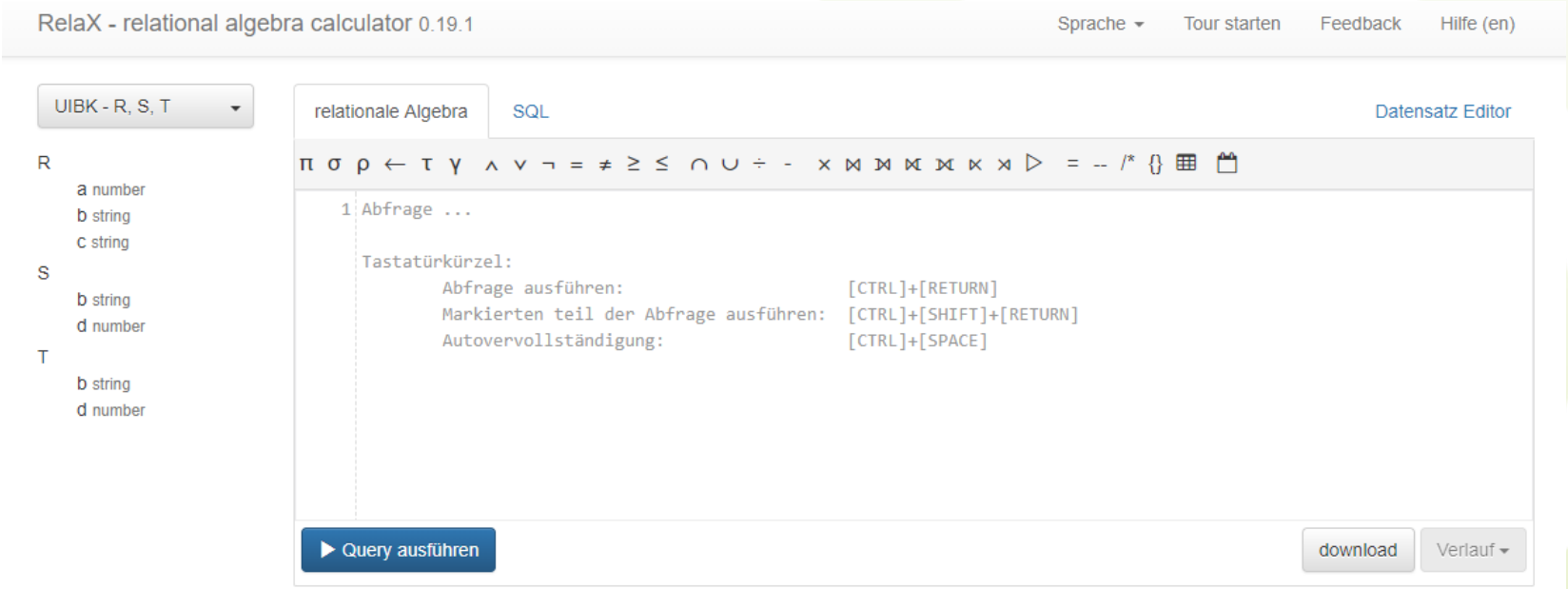
- $StudentExams\ =$
  $\rho_{StudentExams(mat\_no,firstname,exam\_count)}$
  $_{mat\_no,firstname}\mathfrak{F}\ count(course)$
  $Student \bowtie_{Student.mat\_no=Exam.mat\_no} Exam$
- $Max\_Exams = \mathfrak{F}_{max(exam\_count)}\ StudentExams$
- $\pi_{matno,firstname}\sigma_{exam\_count=Max\_Exams}$ StudentExams

mat_no	firstname
1005	Elon
5119	Aristotle

6.4 Recommendation

Exercise

- The University of Innsbruck built a **relational algebra calculator**
  - https://dbis-uibk.github.io/relax/calc
  - It can be helpful to improve your relational algebra skills



6 Next Lecture

- Tuple relational calculus
  - foundation of SQL
- Domain relational calculus
  - foundation of Query-by-example (QBE)

$$F_2(t_1) \equiv \exists t_3 \, (SC(t_3) \wedge t_3.matNr = t_1.matNr \wedge t_3.crsNr = t_2.crsNr)$$