ifis
Institut für Informationssysteme
Technische Universität Braunschweig

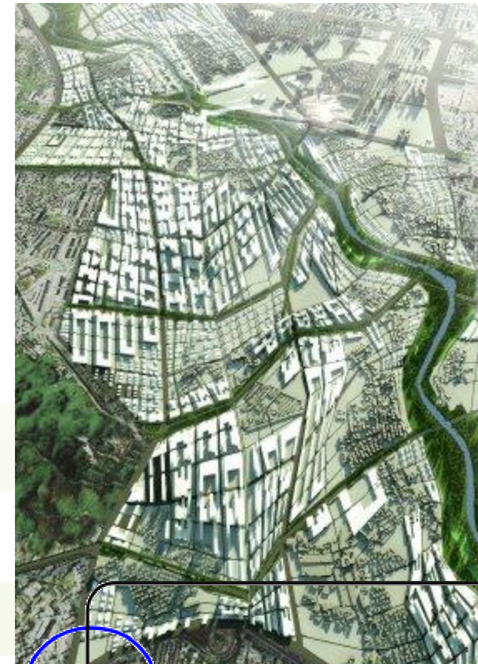
Relational Database Systems 1

Wolf-Tilo Balke
Simon Barthel, Philipp Wille

Institut für Informationssysteme
Technische Universität Braunschweig
www.ifis.cs.tu-bs.de

Overview

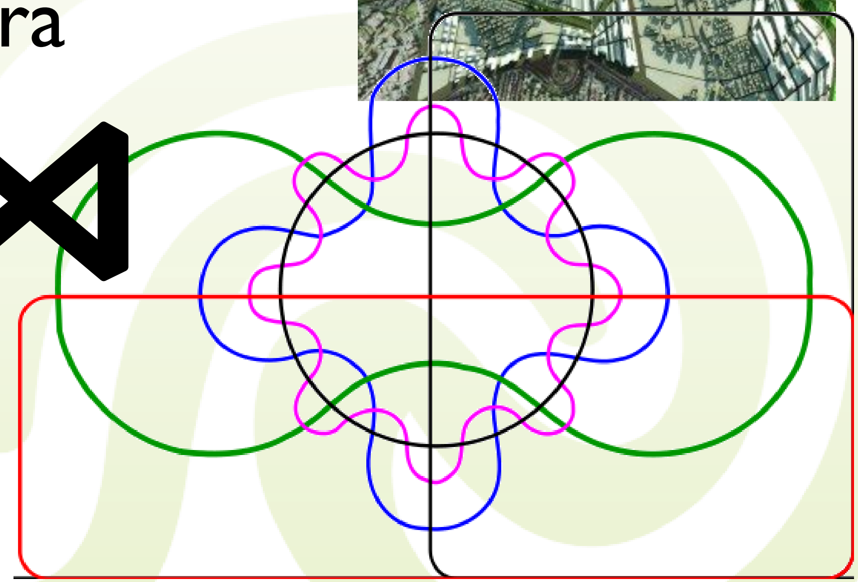
σ



- Relational Algebra
  - Basic relational algebra operations
  - Additional derived operations
- Query Optimization
- Advanced relational algebra
  - Outer Joins
  - Aggregation

$\bowtie$

π



6.1 Motivation

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

6.1 Motivation

- Last week we introduced the **relational model**
  – Based on **set theory**
  – A relation is a **subset** of the **Cartesian product** over a list of **domains**
- Relations can be written as **tables:**

relation name

attributes

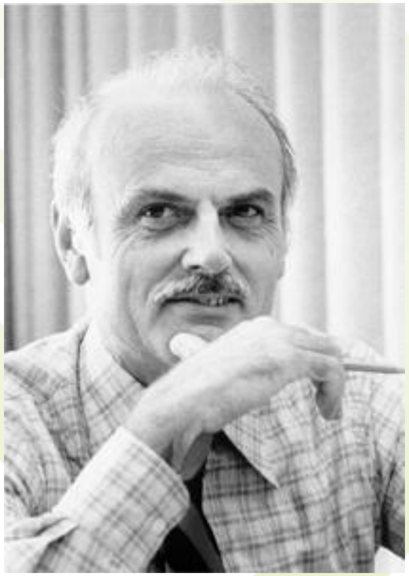
PERSON	firstName	lastName	sex
	Clark Joseph	Kent	m
	Louise	Lane	f
	Lex	Luthor	m
…	Charles	Xavier	m
	Erik	Magnus	m
	Jeanne	Gray	f
	Ororo	Munroe	f

tuples

domain values

6.1 Motivation

- How do you work with 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**



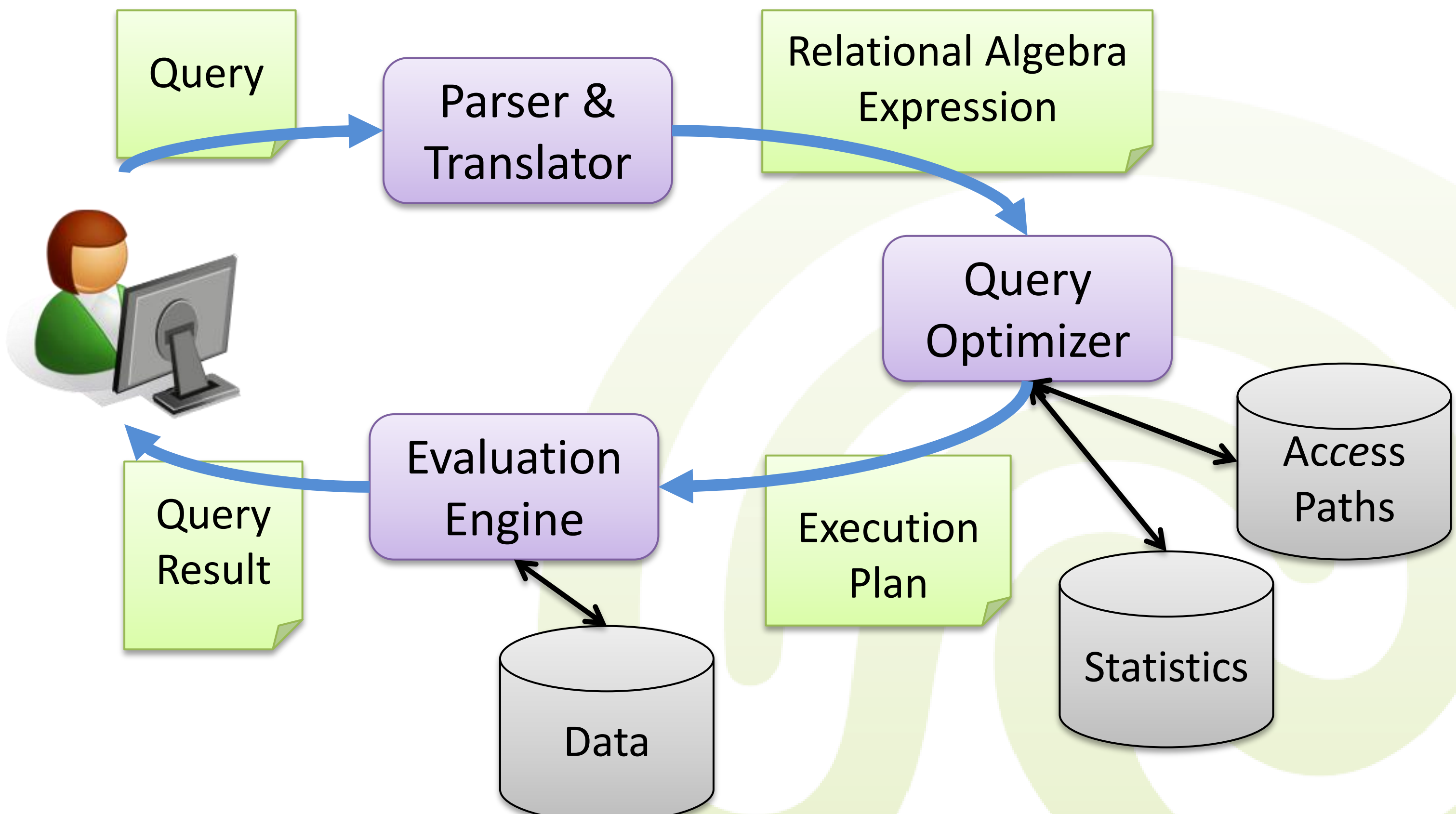
6.1 Motivation

- 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?
- The relational algebra is usefull, because it is **less** powerful than C or Java.
- **2 huge rewards**:
  - Ease of programming
  - Ability of the compiler to produce highly optimized code

[GUW09, 2]

6.1 Motivation

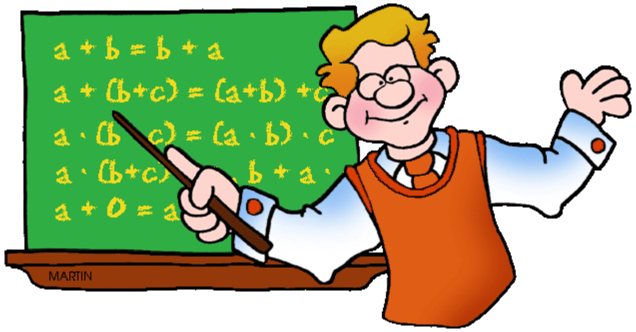
- The first thing that happens to a SQL query is that it gets translated into relational algebra.



6.1 Motivation

- Queries need to be translated to an **internal form**
  - Queries posed in a **declarative** DB language
    - "what should be returned", not "how should it be returned"
  - Queries can be evaluated in different way
- 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

6.2 Relational Algebra

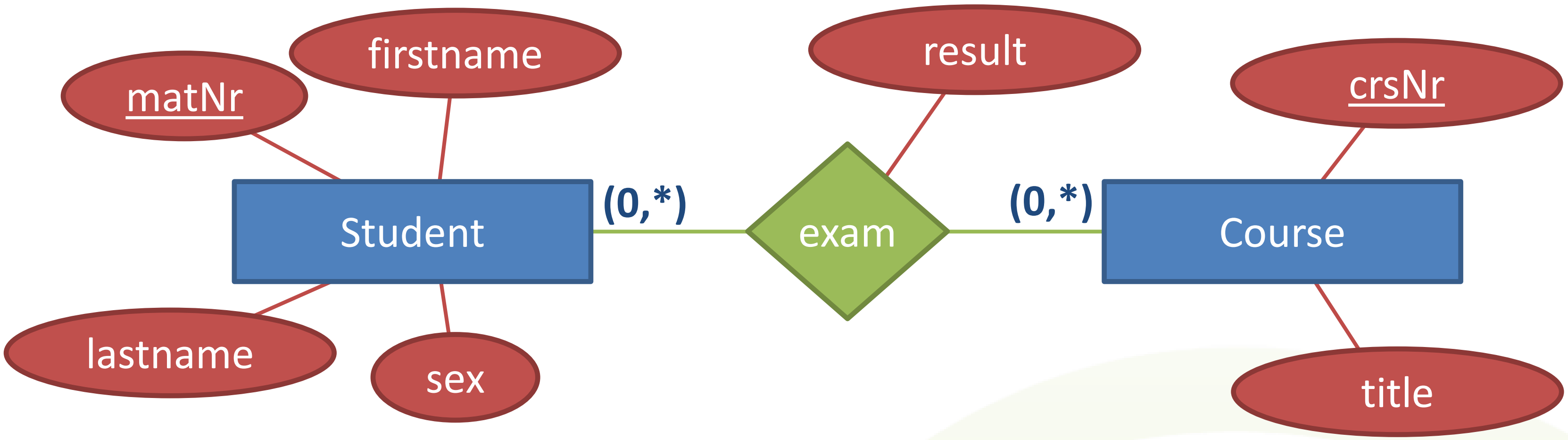


- What is an algebra after all?
  - It consists of
    - operators and atomic operands.
  - It allows to
    - build *expressions* by applying operators to operands and / or other expressions of the algebra.
  - Example: algebra of arithmetics
    - Atomic operands: variables like *x* and constants like 15
    - Operators: addition, subtraction, multiplication and division
- Relational algebra: another example of an algebra
  - 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>matNr</u>, firstname, lastname, sex)

Course(<u>courseNr</u>, title)

exam(<u>student</u> → Student, <u>course</u> → Course, result)

6.2 Example Relations

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m
6676	Erik	Magnus	m
8024	Jeanne	Gray	f
9876	Logan		m

Course

crsNr	title
100	Intro to being a Superhero
101	Secret Identities 2
102	How to take over the world

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3
6676	102	4.3
5119	101	1.7

6.2 Relational Algebra

- **Selection σ**
  - **Selects** all tuples (rows) from a relation that satisfy some given **Boolean predicate** (condition)
    - "Selection" = Create a new relation that contains exactly the satisfying tuples
  - $\sigma_{\text{<condition>}} R$
  - Condition clauses:
    - <attribute> θ <value>
    - <attribute> θ <attribute>
    - $\theta \in \{=, <, \leq, \geq, >, \neq\}$
  - Clauses may be connected by ∧,∨ and ¬ (logical AND, OR, and NOT)

6.2 Relational Algebra

- Example:

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m
6676	Erik	Magnus	m
8024	Jeanne	Gray	f
9876	Logan		m

"Select all female students"

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

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

matNr	firstname	lastname	sex
2832	Louise	Lane	f
8024	Jeanne	Gray	f

6.2 Relational Algebra

- Example:

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3
6676	102	4.3
5119	101	1.7

"Select exams within course 100 with a result of worse than 3.0"

$\sigma_{course=100\ \wedge\ result>3.0}\ exam$

$\sigma_{course=100\ \wedge\ result>3.0}\ exam$

student	course	result
9876	100	3.7

6.2 Relational Algebra

- **Projection π**
  - Retains only attributes (columns) with given names
    - Again: Creates a new relation with only those attribute sets which are specified within some attribute list
    - If tuples are identical after projection, **duplicate tuples are removed**
  - $\pi_{\text{<attributeList>}} R$

"All courses titles":

Course

crsNr	title
100	Intro. to being a Superhero
101	Secret Identities 2
102	How to take over the world

π_{title} Course

title
Intro. to being a Superhero
Secret Identities 2
How to take over the world

6.2 Relational Algebra

- Of course, these operations can be combined

"Retrieve first name and last name of all female students"

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

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m
6676	Erik	Magnus	m
8024	Jeanne	Gray	f
9876	Logan		m

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

matNr	firstname	lastname	sex
2832	Louise	Lane	f
8024	Jeanne	Gray	f

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

firstname	lastname
Loise	Lane
Jeanne	Gray

6.2 Relational Algebra

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

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

$\rho_{\text{Lecture}}\ \pi_{\text{crsNr}}\ \text{Course}$

Course

crsNr	title
100	Intro. to being a Superhero
101	Secret Identities 2
102	How to take over the world

→

Lecture

crsNr
100
101
102

6.2 Relational Algebra

- **Renaming operator ρ**

"Select all result of course 100 from exam; the resulting relation should be called 'result' and contain the attributes 'matNo', 'crsNo', and 'grade'"

$$\rho_{\text{results(matNo, crsNo, grade)}}\ \sigma_{\text{course}=100}\ \text{exam}$$

$\sigma_{\text{course}=100}\text{exam}$

student	course	result
9876	100	3.7
1005	100	1.3

results

matNo	crsNo	grade
9876	100	3.7
1005	100	1.3

6.2 Relational Algebra

- **Union** ∪, **intersection** ∩, and **set difference** \
  - Operators work as in set theory
    - Operands have to be **union-compatible** (i.e., they must consist of the same attributes)
  - Written as $R \cup S$, $R \cap S$, and $R \setminus S$, respectively

$\sigma_{crsNr=100}$ Course

crsNr	title
100	Intro. to being a Superhero

$\sigma_{crsNr=102}$ Course

crsNr	title
102	How to take over the world

$\sigma_{crsNr=100}$ Course $\cup$ $\sigma_{crsNr=102}$ Course

crsNr	title
100	Intro. to being a Superhero
102	How to take over the world

6.2 Relational Algebra

- **Cartesian product ×**
  - Written as R × 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| · |S| tuples

6.2 Relational Algebra

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
…			

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
…		

Student × exam

matNr	firstname	lastname	sex	student	course	result
1005	Clark	Kent	m	9876	100	3.7
1005	Clark	Kent	m	2832	102	2.0
1005	Clark	Kent	m	1005	101	4.0
…						
2832	Louise	Lane	f	9876	100	3.7
2832	Louise	Lane	f	2832	102	2.0
…						

Useless!

6.2 Relational Algebra

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

lastname	course	result
Kent	Secret Identities 2	4.0
Kent	Intro to being a Superhero	1.3
Xavier	Secret Identities 2	1.7
…		

Useful!

- This type of statement 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 $R \bowtie_{\text{<condition>}} S$ or $\sigma_{\text{<condition>}} (R \times S)$
  - Theta joins are very similar to selections

$$\pi_{\text{lastname, title, result}} (\text{Student} \bowtie_{\text{matNo=student}} \text{exam} \bowtie_{\text{course=crsNo}} \text{Course})$$

$$=$$

$$\pi_{\text{lastname, title, result}}\, \sigma_{\text{matNo=student} \land \text{course=crsNo}} (\text{Student} \times \text{exam} \times \text{Course})$$

6.2 Relational Algebra

- **Equi-join** $\bowtie_{\text{<condition>}}$
  - Joins two relations only using **equality conditions**
  - $R \bowtie_{\text{<condition>}} S$
  - <condition> may only contain equality statements between attributes ($A_1 = A_2$)
    - A special case of the theta join

$$\pi_{\text{lastname, title, result}}\ (\text{Student} \bowtie_{\text{matNo=student}} \text{exam} \bowtie_{\text{course=crsNo}} \text{Course})$$
$$=$$
$$\pi_{\text{lastname, title, result}}\ \sigma_{\text{matNo=student} \wedge \text{course=crsNo}}\ (\text{Student} \times \text{exam} \times \text{Course})$$

6.2 Relational Algebra

- **Natural join** $\bowtie_{\text{<attributeList>}}$
  - A special case of the equi-join
  - $R \bowtie_{\text{<attributeList>}} S$
  - 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 <attributeList> is empty:
      - Join attributes = All attributes that are shared between the two relations (that is, that have the same name)

6.2 Relational Algebra

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
…			

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
…		

Course

crsNr	title
100	Intro. to being a Superhero
101	Secret Identities 2
102	How to take over the world

$$\text{Student} \bowtie_{\text{matNo}=\text{student}} \text{exam} \bowtie_{\text{course}=\text{crsNo}} \text{Course}$$

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 actually executing the query

Query Optimization

Detour

- Example:

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

Student

matNr	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

crsNr	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

Query Optimization

Detour

- Example:

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

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

Course (4 Byte, 30 Byte)

crsNo	title
41	Cooking rabbits
40	Destroying rings
42	Flying eagles

Student

matNo	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

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

Query Optimization

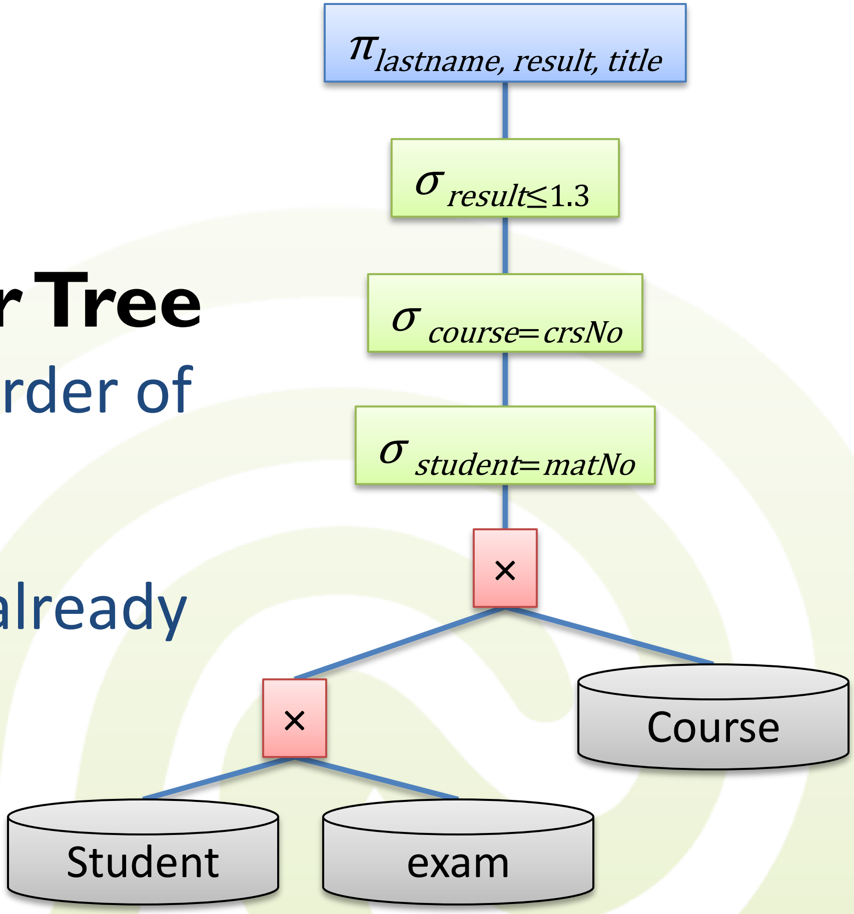
Detour

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

$\sigma_{\text{result} \leq 1.3\ \wedge\ \text{course} = \text{crsNo}\ \wedge\ \text{matNo} = \text{student}}$

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

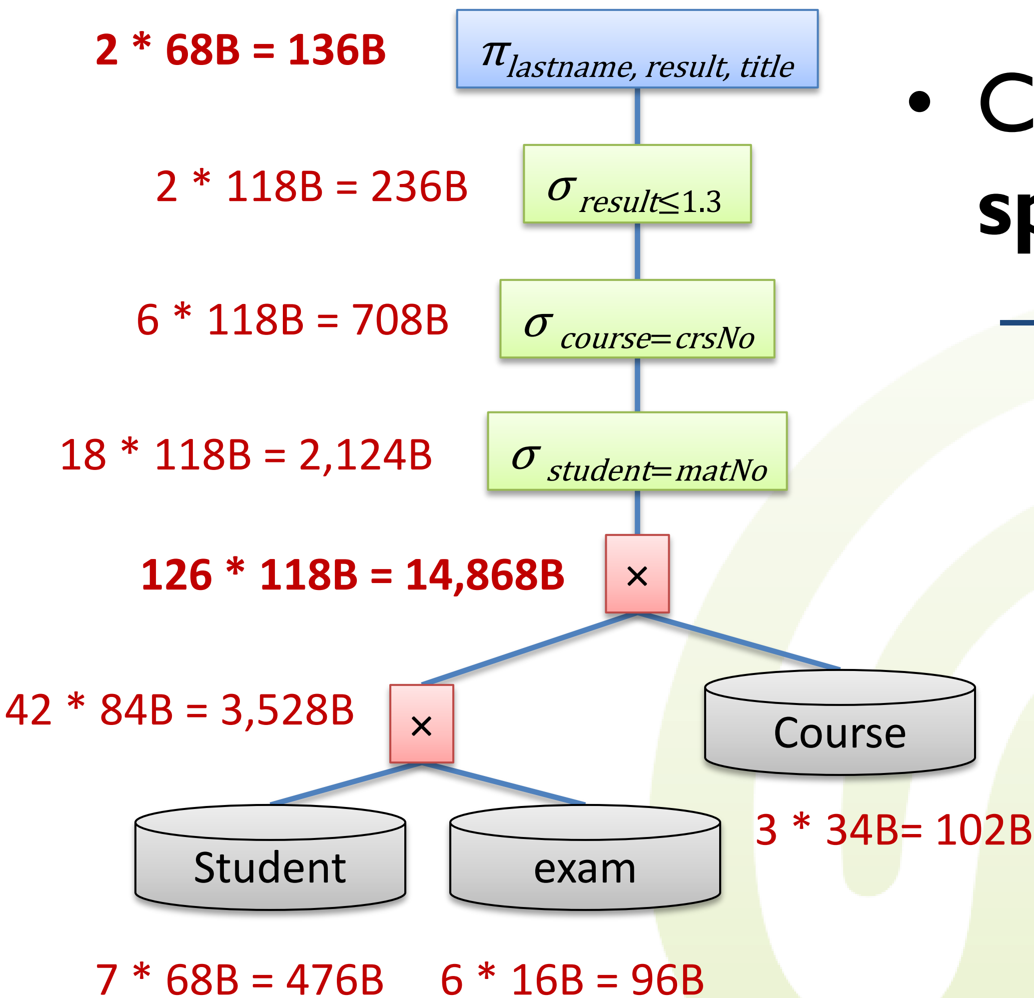
- Create Canonical **Operator Tree**
  - Operator tree visualized the order of primitive functions
  - (Note: Illustration is not really canonical tree as selection is already split in three parts)



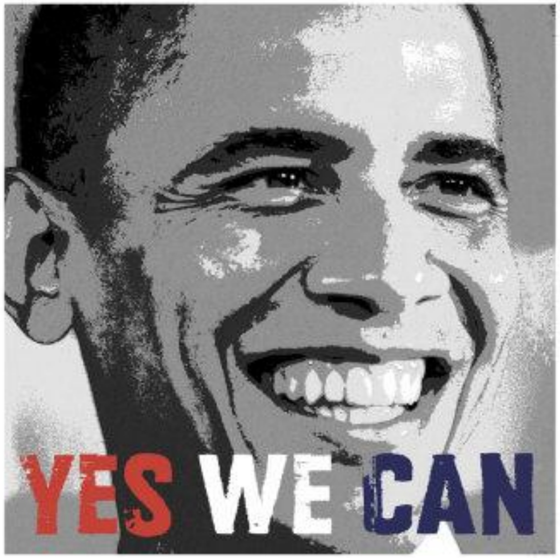
Query Optimization

Detour

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



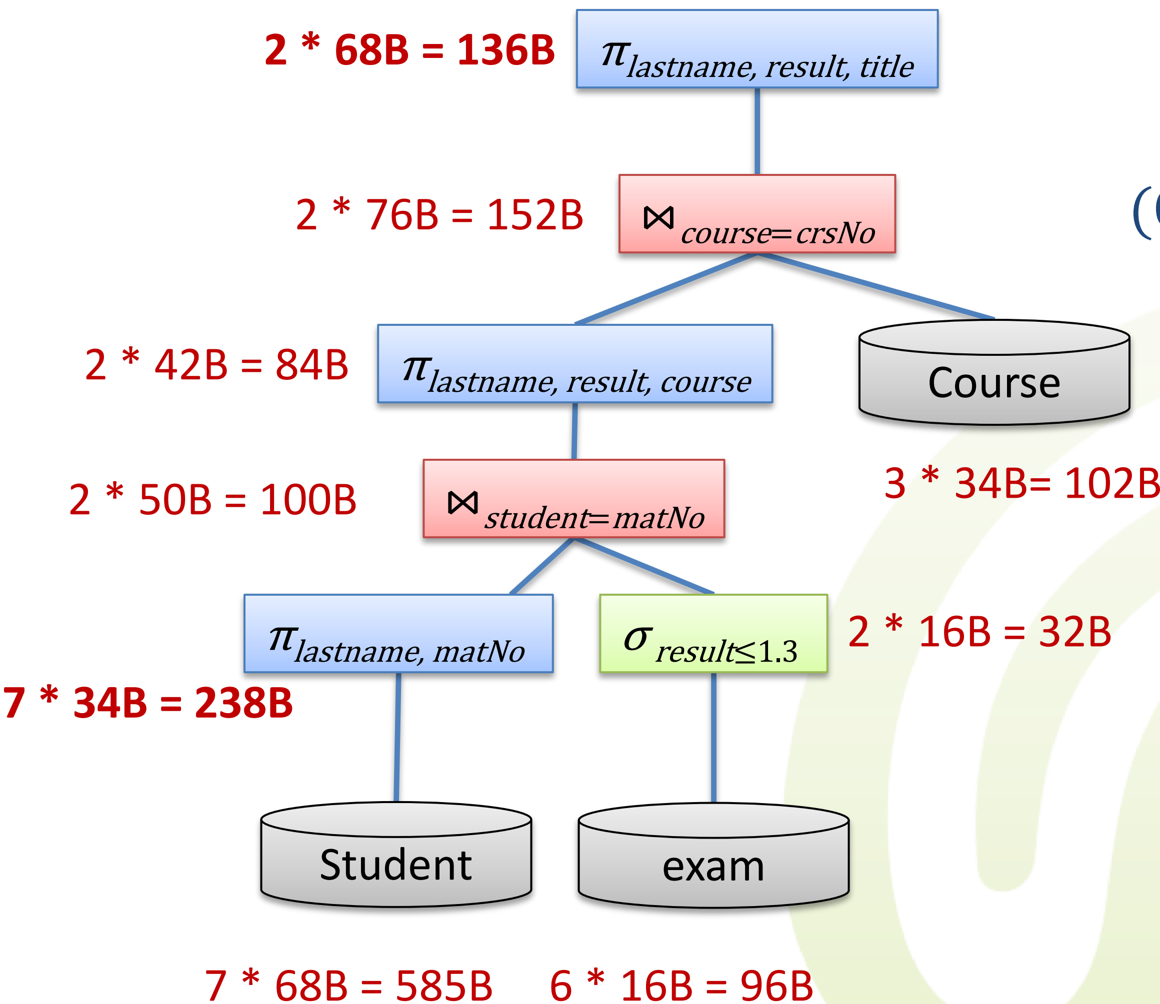
- Can we optimize **space usage**?
  - e.g. by **reorganizing** operators?



Query Optimization

Detour

- Optimized Operator Tree

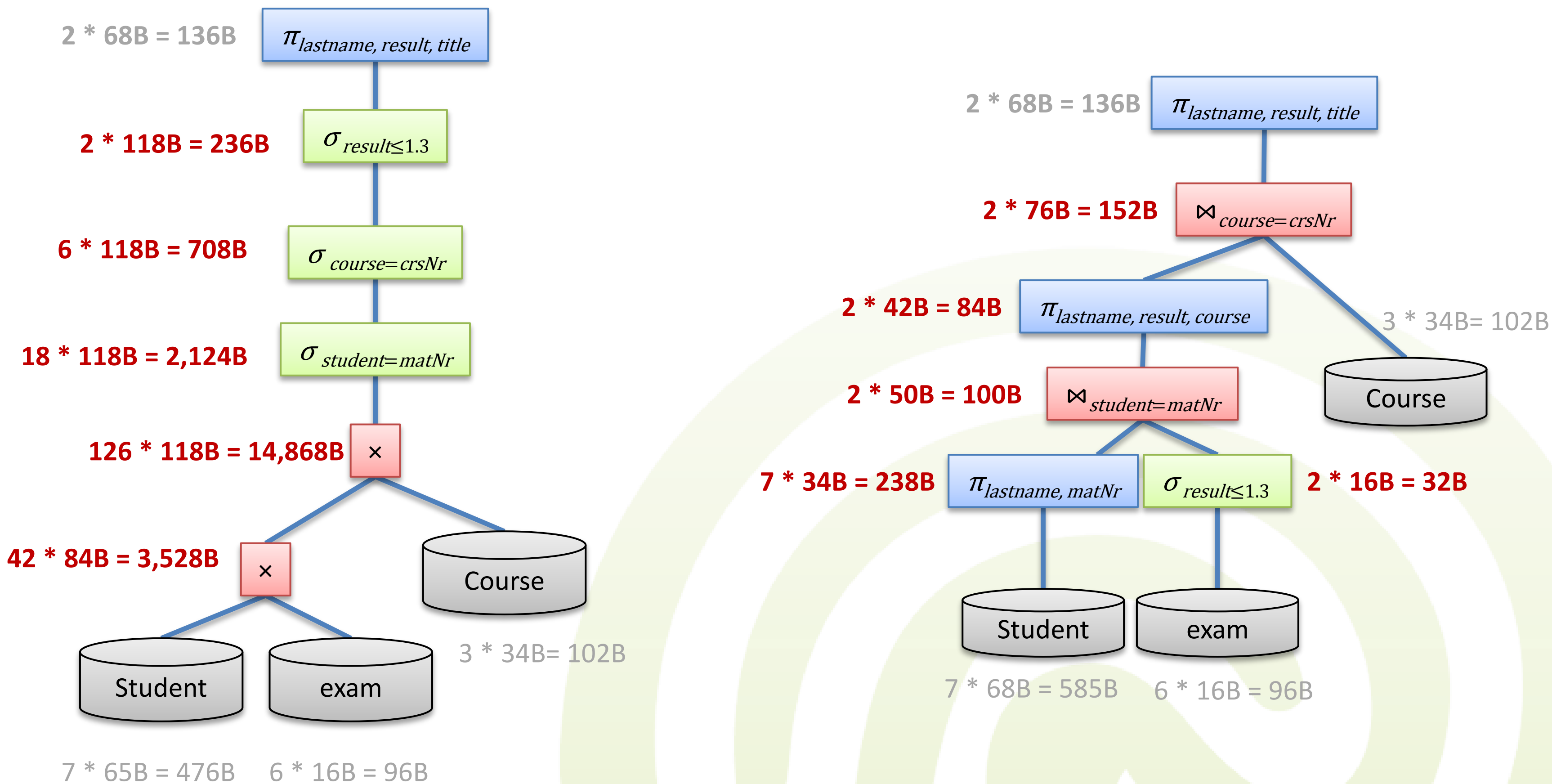


$$\pi_{\text{lastname, result, title}}\ \sigma_{\text{result} \leq 1.3\ \wedge\ \text{course} = \text{crsNo}\ \wedge\ \text{matNo} = \text{student}} (\text{Course} \times \text{Student} \times \text{exam})$$

$$=$$

$$\pi_{\text{lastname, result, title}} (\text{Course} \bowtie_{\text{crsNo}=\text{course}} \pi_{\text{lastname, result, course}} (\pi_{\text{lastname, matNo}}\ \text{Student} \bowtie_{\text{matNo}=\text{student}} \sigma_{\text{result} \leq 1.3}\ \text{exam}))$$

- Comparison of intermediate results



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

6.2 Relational Algebra

- **Left semi-join** $\ltimes$ and **right semi-join** $\rtimes$
  - Combination of a theta-join and projection
    - $R \ltimes_{\text{<condition>}} S = \pi_{\text{(list of all attributes in R)}} (R \bowtie_{\text{<condition>}} S)$
    - $R \rtimes_{\text{<condition>}} S = \pi_{\text{(list of all attributes in S)}} (R \bowtie_{\text{<condition>}} S)$
  - Creates a copy of R (or S) while removing all those tuples that do not have a join partner
    - A filtering operation
  - Works like a natural join if <condition> is empty

6.2 Relational Algebra

Left semi-join:

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

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

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

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f

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

6.2 Relational Algebra

Division ÷

- Written $R \div S$
- 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 **any** tuple in S is a join partner
  - Useful for queries like "Find the sailors who have reserved all boats."
- More formal:
  - Let A_R={attributes of R}; A_S ={attributes of S}; $A_S \subseteq A_R$
  - $A = A_R \setminus A_S$
  - $R \div S \equiv \pi_A R \setminus \pi_A((\pi_A(R) \times S) \setminus R)$
- Why the name division? Because $(R \times S) \div S = R$.
  - **But:** $(R \div S) \times S = R$ is not true in general

Division:

$SC = \rho_{SC}\,(\pi_{matNr,\ lastname,\ crsNr}\ (Student \bowtie_{matNr=student}\ exam))$

matNr	lastname	crsNr
1000	Kent	100
1000	Kent	102
1001	Lane	100
1002	Luther	102
1002	Luther	100
1002	Luther	101
1003	Xavier	103
1003	Xavier	100

$CCK = \rho_{CCK}(\pi_{crsNr}\sigma_{name=\text{'Clark Kent'}}\ SC)$

crsNr
100
102

$SC \div CCK$

matNr	lastname
1000	Kent
1002	Luther

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

6.2 Summary

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

Translation of Relational Algebra Expressions

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

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

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

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

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

Translation of Relational Algebra Expressions

Person(<u>id</u>, name, sex)

plays(<u>person</u> → Person, <u>movie</u> → Movie, role)

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

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

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

Translation of Relational Algebra Expressions

Person(<u>id</u>, name, sex)

plays(<u>person</u> → Person, <u>movie</u> → Movie, role)

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

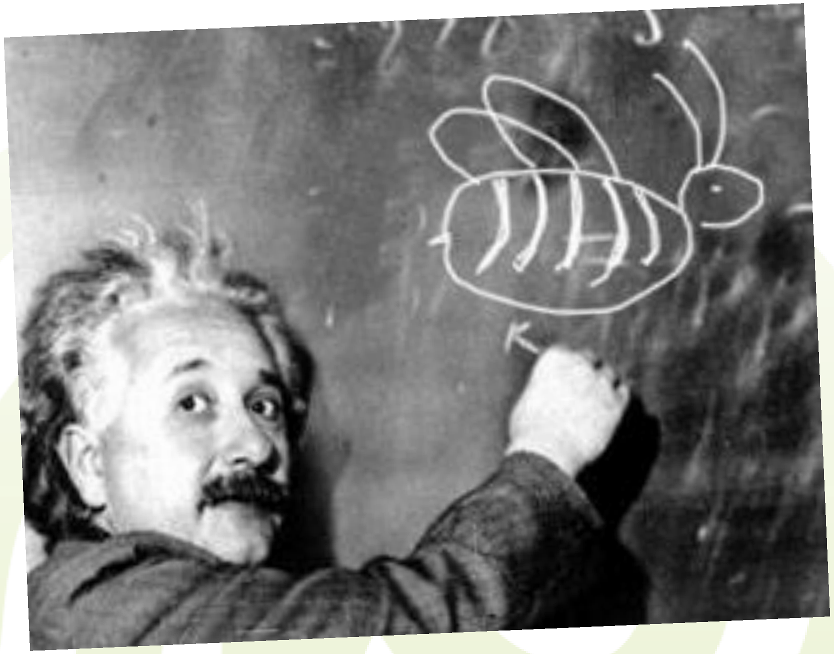
has_genre(<u>movie</u> → Movie, <u>genre</u> → Genre)

Genre(<u>name</u>, description)

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

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

- **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

Example: List students and their exam results

$\pi_{\text{lastname, crsNr, result}}\ (\text{Student} \bowtie_{\text{matNr=student}} \text{exam})$

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

$\pi_{\text{lastname, crsNr, result}}\ (\text{Student} \bowtie_{\text{matNr=student}} \text{exam})$

lastname	crsNr	result
Kent	100	1.3
Kent	101	4.0
Lane	102	2.0

Lex Luther and Charles Xavier are lost because they didn't take any exams!
Also, information on student 9876 disappears…

Left outer join: List students and their exam results

$$\pi_{\text{lastName, crsNr, result}} (\text{Student} ⟕_{\text{matNr=student}} \text{exam})$$

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

$$\pi_{\text{lastname, course, result}} (\text{Student} ⟕_{\text{matNr=student}} \text{exam})$$

lastname	course	result
Kent	100	1.3
Kent	101	4.0
Lane	102	2.0
Luther	NULL	NULL
Xavier	NULL	NULL

All student names with courseNo and result if present.

Right outer join: List exams and their participants.

$$\pi_{\text{lastname, crsNr, result}}\ (\text{Student} ⟖_{\text{matNr=student}}\ \text{exam})$$

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

$$\pi_{\text{lastname, course, result}}\ (\text{Student} ⟖_{\text{matNr=student}}\ \text{exam})$$

lastname	course	result
Kent	100	1.3
Kent	101	4.0
Lane	102	2.0
NULL	100	3.7

All exam result with student names if known.

6.3 Advanced Relational Algebra

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

$$\pi_{\text{lastname, crsNr, result}} (\text{Student} ⟗_{\text{matNr=student}} \text{exam})$$

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

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

lastname	course	result
Kent	100	1.3
Kent	101	4.0
Lane	102	2.0
Luther	NULL	NULL
NULL	100	3.7
Xavier	NULL	NULL

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

6.3 Aggregation

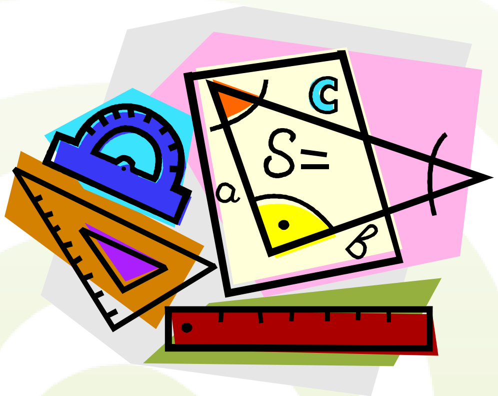
- **Aggregation operator:**
  Typically used in simple statistical computations
  - Merges tuples into **groups**
  - **Computes** (simple) statistics **for each group**
- **Examples:**
  - "Compute the average exam score"
  - "For each student, count the total number of exams he/she has taken so far"
  - "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
    - The name of each new attribute is "function name + argument name"
- Example: ${}_{course}\mathfrak{F}_{average(result)}\ exam$

6.3 Aggregation

- **Attention:**
  - **Duplicates are usually NOT eliminated when grouping**
- Some available functions:
  - **Sum**
    - Sum of all non-NULL values
  - **Average**
    - Mean of all non-NULL values
  - **Maximum**
    - Maximum value of all non-NULL values
  - **Minimum**
    - Minimum value of all non-NULL values
  - **Count**
    - Number of tuples having a non-NULL value



6.3 Aggregation

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

$$\rho_{\text{WithoutGroups(sum, avgResult, minResult, avgResult)}} (\mathfrak{F}_{\text{sum(student), average(result), min(result), max(result)}} \text{ exam})$$

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
6676	102	5.0
5119	101	1.7

WithoutGroups

sum	avgResult	minResult	maxResult
25508	3.28	1.7	5.0

6.3 Aggregation

- Example (with grouping):

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

$$\rho_{\text{WithGrouping (crsNo, avgResult, \#result)}} \left({}_{\text{course}}\mathfrak{F}_{\text{average(result), count(result)}} \text{ exam} \right)$$

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3
6676	102	4.3

WithGrouping

crsNo	avgResult	#result
100	2.5	2
101	4.0	1
102	3.15	2

6.3 Aggregation

- Example:
  "For each student, count the exams and compute average result."

$$\rho_{\text{Overview(matNo, \#exams, avgResult)}} (\\ {}_{\text{matNo}}\mathfrak{F}_{\text{count(course), avg(result)}} \\ \pi_{\text{matNo, course, result}} \\ (\text{Student} \bowtie_{\text{matNo=student}} \text{exam}) \\ )$$

exam

student	course	result
9876	100	3.7
2832	102	2.0
1005	101	4.0
1005	100	1.3

Overview

matNo	#exams	avgResult
1005	2	2.65
2832	1	2.0
4512	0	NULL
5119	0	NULL

Student

matNr	firstname	lastname	sex
1005	Clark	Kent	m
2832	Louise	Lane	f
4512	Lex	Luther	m
5119	Charles	Xavier	m

6.3 Operator Precedence

- If you want to save some brackets…
  - **Unary** operators are applied first **(σ, π, ρ, $\mathfrak{F}$)**
  - **Cross product** and **joins** are applied afterwards **($\times$, $\bowtie$, …)**
  - Followed by **Union** und **set minus ($\cup$, \\)**
  - Last step is **Intersection ($\cap$)**

$$\begin{aligned}(\text{exam} \bowtie_{\text{course=crsNo}} (\sigma_{\text{crsNo}=101}\ \text{Course})) \\ \cup\ (\text{exam} \bowtie_{\text{course=crsNo}} (\sigma_{\text{crsNo}=100}\ \text{Course}))\end{aligned} = \begin{aligned}\text{exam} \bowtie_{\text{course=crsNo}} \sigma_{\text{crsNo}=101}\ \text{Course} \\ \cup\ \text{exam} \bowtie_{\text{course=crsNo}} \sigma_{\text{crsNo}=100}\ \text{Course}\end{aligned}$$

$$\underbrace{\pi_{\text{lastname, result}}\ \text{Student}}_{\text{Projection is applied first!}} \bowtie_{\text{matNo=student}} \text{exam}$$

6.3 Conflicting Attribute Names

- In most previous examples, all relations had different attribute names
  - The real world is different
- In case of **conflicting names**, the **full name** of relation *R*'s attribute *A* is *R.A*
  - Still, attribute names need to be **unique within each relation**
- **Example:**
  - $\sigma_{\text{Students.matNo} = \text{Grade.matNo}}\ (\text{Student} \times \text{Grade})$

Next Lecture

- Relational tuple calculus
  - SQL
- Domain tuple calculus
  - Query-by-example (QBE)

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