Technische Hochschule Rosenheim

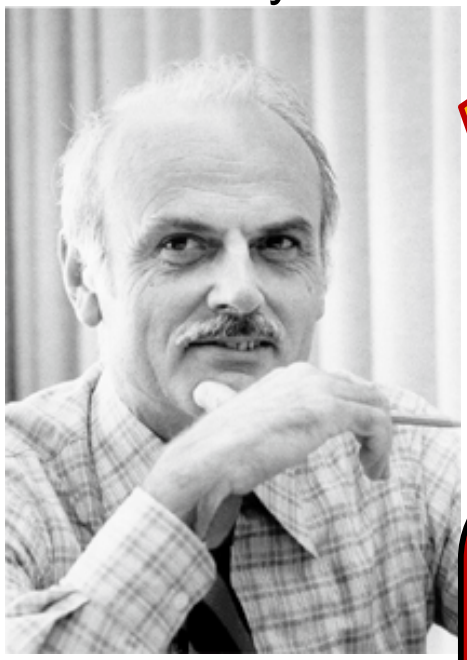
Chapter 2 – Relational models and relational algebra

Databases lectures

Dr Kai Höfig

Relational algebra, calculus and languages

Theory



In the 1960s and 1970s, Edgar Codd created the relational model that forms the basis for relational databases, which are still a standard of database technology today.

Practical requirements

Sorting
Grouping
Recursion
etc.

Relational algebra
(Tupel) Relational calculus

Domain relational calculus

Logical foundation

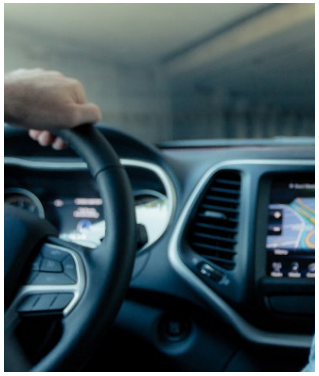
Without understanding the logical foundation, we interact with an unknown communication partner using the database languages

Practicable database languages

SQL

QBE

Practice



Relational algebra and SQL

- **Relational algebra** is the foundation for the **tupel relational calculus**, which is implemented by the **Sequence Query Language** (SQL)
  - Relational calculus is **declarative**, i.e. the calculation sequence is not visible
    - Easier to use than procedural instructions – user does not need to know how the DBMS calculates the result.
    - DBMS has freedom with processing, so it can choose one that is as efficient as possible.
  - Relational algebra is **procedural**, i.e. it specifies the processing sequence ("from inside to outside").
    - There are laws for the transformation of algebra expressions.
    - DBMS translates the SQL query (=tuple relational calculus) into an algebra expression, optimises it (query optimiser) and executes it.
    - This "execution plan" is used by the DB administrator to optimise the DB (e.g. by means of index structures).

The relational data model

- Structure of the data:
  Data is stored in relations (tables)

- Operations on the data: 2 alternatives
  1. Relational calculus (implemented in SQL)
  2. Relational algebra (base for SQL)

- Integrity constraints: The most important:
  1. Key constraints
  2. Referential integrity = foreign key constraints
  3. Domain constraints (restrictions on values allowed for an attribute)
  - Integrity constraints can be formulated as conditions in relational algebra, relational calculus or SQL

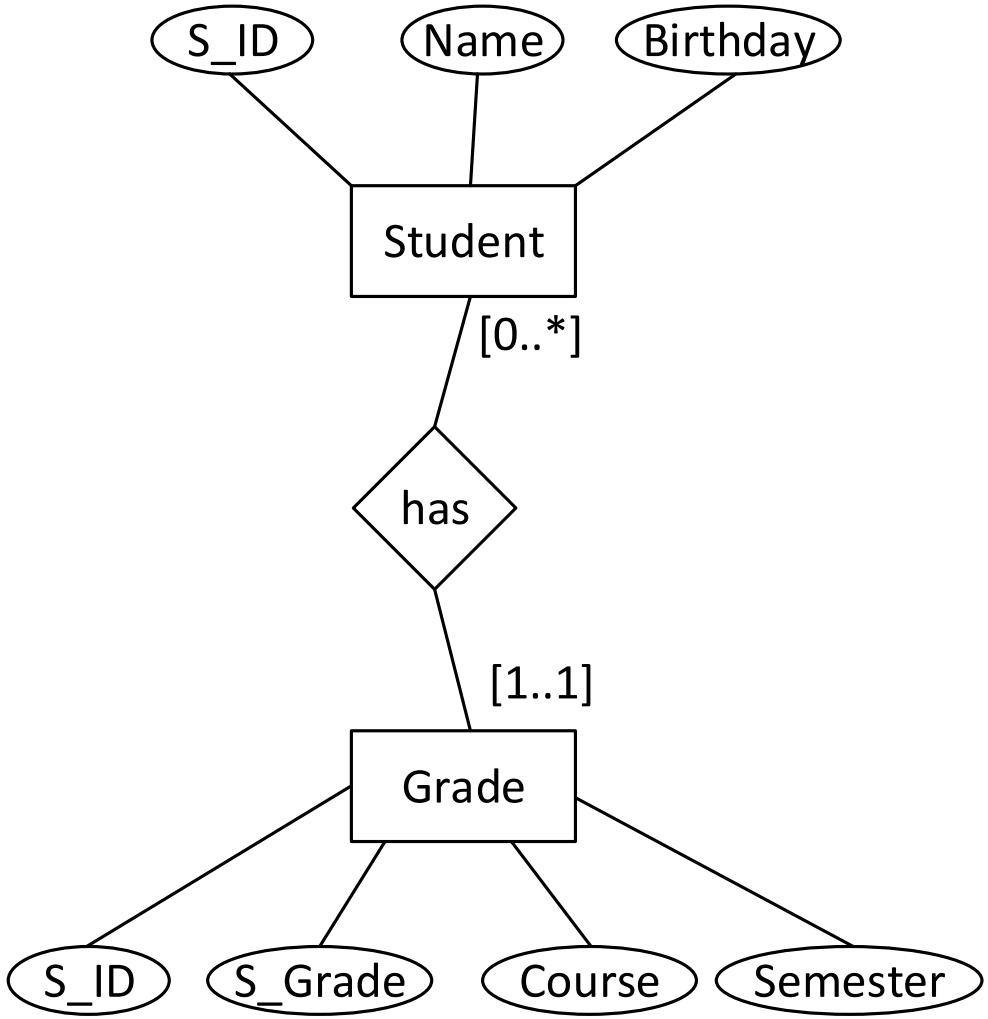
Running example for this lecture

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

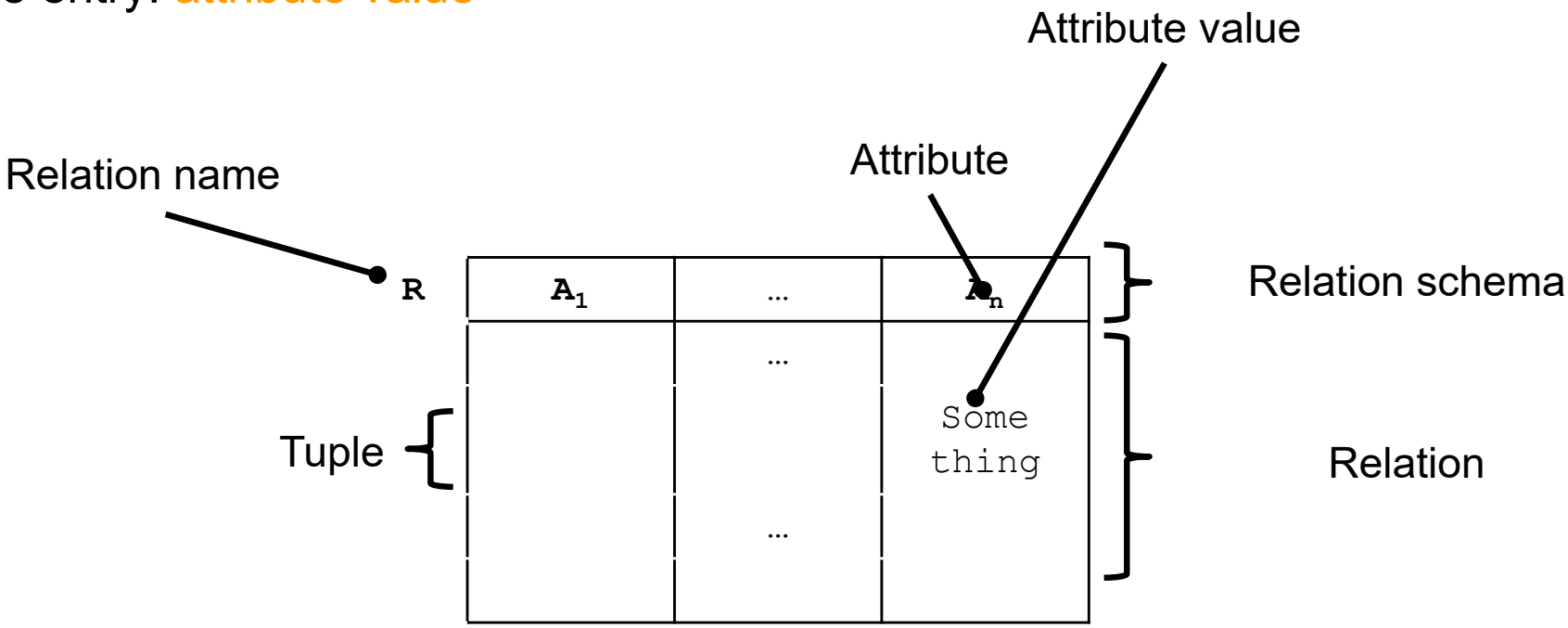
Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23



Representation of relations and terms

- Representation
  - First row: relation schema
  - Other entries in the table: relation
  - One row of the table: tuple
  - A column heading: attribute
  - One entry: attribute value



Definition of the relational data model (1)

- In the relational data model, we only have the relational schema for structural modelling.

- A relational schema with the identifier *R* and Attributes A

$$R = (A_1, \ldots, A_n)$$

- A domain function assigning value ranges to the attributes

$$dom : \{A_1, \ldots, A_k\} \rightarrow \{D_1, \ldots, D_s\}, s \geq 1$$

- A relational database schema containing the relational schema

$$S = \{R_1(A_1, \ldots, A_i), \ldots, R_m(A_j, \ldots, A_k)\}$$

$$\begin{aligned} GradingSystem &= \{Students(S\_ID, Name, Birthday), \\ &= Grade(S\_ID, S\_Grade, Course, Semester) \\ dom : \{S\_ID, Name, Birthday, S\_Grade, Course, Semester\} &\rightarrow int, varchar \end{aligned}$$

Definition of the relational data model (2)

- A relation *r* of a relational schema *R* is a finite set of tuples and is also referred to as base relation.

$$r(R) = t_0, \ldots, t_n$$

- A database via a database schema *S* is a set of relations.

$$d = r_0, \ldots, r_n$$

$$\begin{aligned} r_{Students} &= t_0, t_1, t_2, t_3 \\ d_{GradingSystem} &= r_{Students}, r_{Grades} \\ t_0(Name) &= Max \end{aligned}$$

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Important integrity constraints in RM: Key constraint

- One or more attributes fulfil (at a given point in time) the **key property**, if there are no two tuples in that relation with the same value of those attributes. We also say, that one or more attributes **identify** uniquely stored tuples.

Key constraint: There are no 2 tuples in `Students` with the same value of `S_ID`. From the domain, we know that this is a candidate for a key.

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

- Note
  - A key is domain knowledge
  - Attribute combinations can also be keys
  - A table can have multiple keys
  - When the DB is designed, one key is selected as the **primary key**
  - Keys are generally marked by underlining

Key constraint: There are no 2 tuples in `Students` with the same value of `Name`. Despite the key constraint is currently fulfilled, we know that is cannot be a key.

Exercise

- Which single attributes fulfil the **key attribute**?
- Which attribute **combinations** fulfil the key attribute?
- What attributes are **key candidates**?
- What are **primary keys**?

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23

Important integrity constraints in RM: Referential Integrity

- Two sets of attributes are of referential integrity, if one set of attribute values is a subset of the other set of attribute values.

Students:

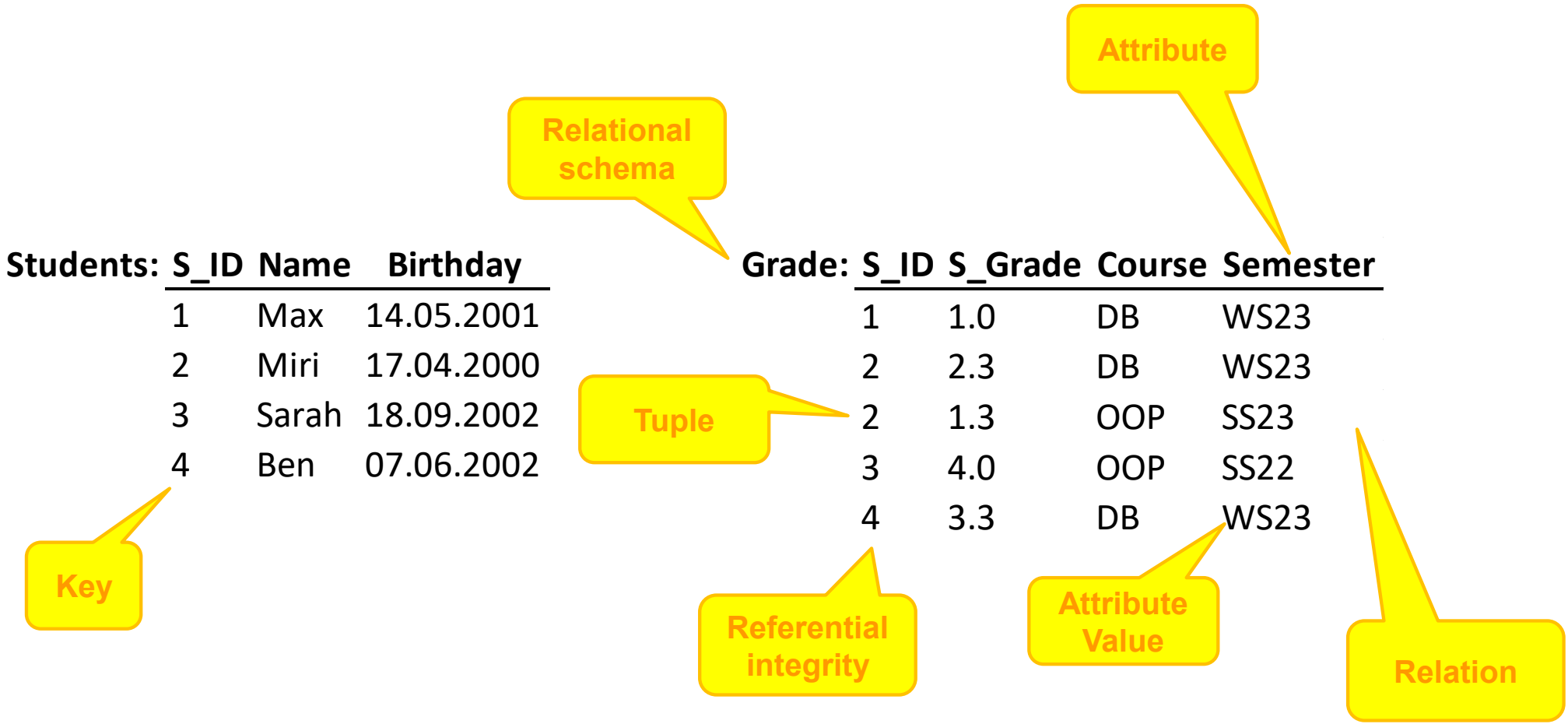
S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23

- Here, the values of `S_ID` in `Grade` is a subset of `S_ID` in `Students`.
- Typically, a foreign key in one relation is a subset of a key in another relation

Structure is set!



But how do I generate relations from base-relations?

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23

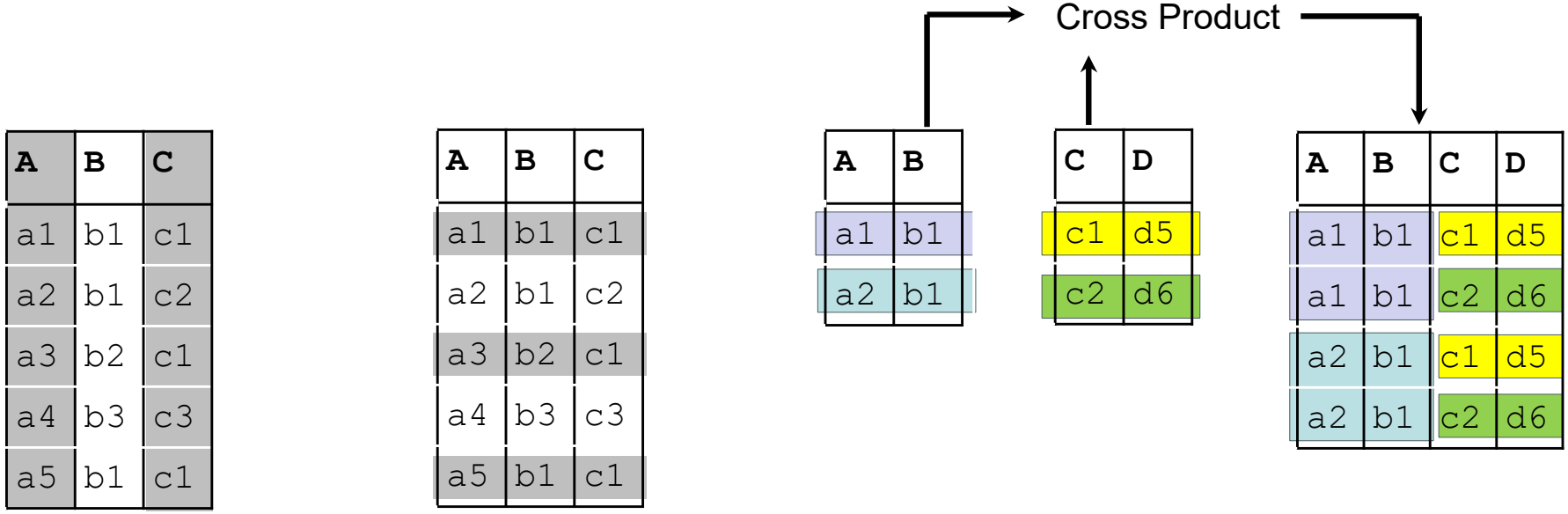
„I can recommend all students as tutors, that have a grade better than 2.0“

Query operations on tables

- Relational algebra: set of basic operations on relations to compute new (resulting) relations from base relations
  - can be combined in any way
  - thereby create an algebra for “calculating with tables”
  - Provides the background fro the calculation of declarative languages like SQL
- Revision from mathematics:
  algebra = value range + operations defined on these
- Here
  - value range = contents of the database = tables
  - operations = functions for calculating new tables

Relational algebra: overview

- Three main operations: Selection, Projection, Cross Product



Projection π

Selection σ

Cross Product $\times$

Projection π Definition

- Projection π (Pi): selection of columns by specifying an attribute list
- Syntax $\pi_{<AttributeList>}(Relation)$
- Example

R

A	B
1	2
1	3
3	4

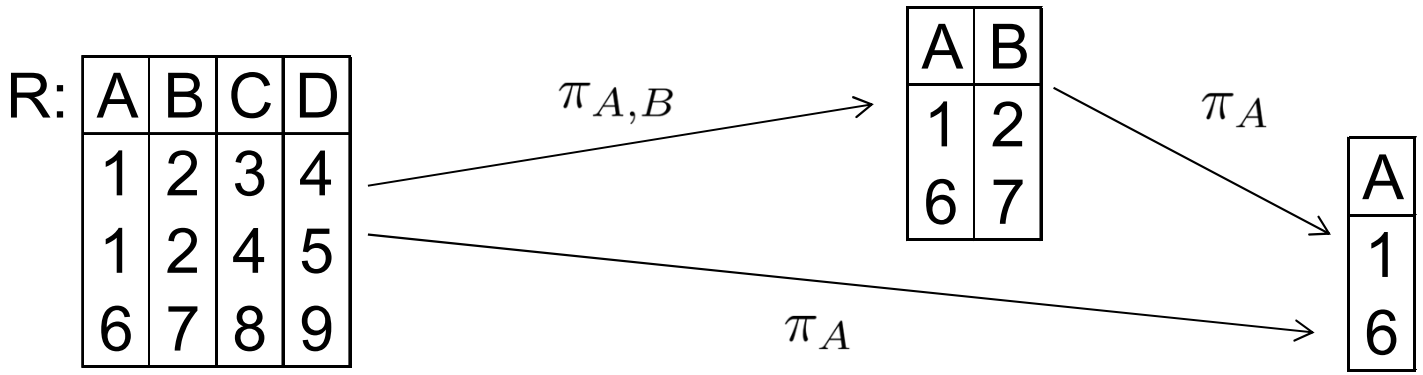
$\pi_A(R)$

A
1
3

- The projection removes duplicate tuples
- **Set semantics**, → only in relational algebra!

Projection: Augmentation

- For $A \subseteq B \subseteq R : \pi_A(\pi_B(R)) = \pi_A(R)$



- Optimization: when projecting on attributes and then projecting on a subset, we can directly project on that subset

Selection σ Definition

- Selection σ (Sigma): selection of rows of a table based on a selection predicate
- Syntax $\sigma_{<Constraint>}(Relation)$
- Example

R

A	B
1	2
1	3
3	4

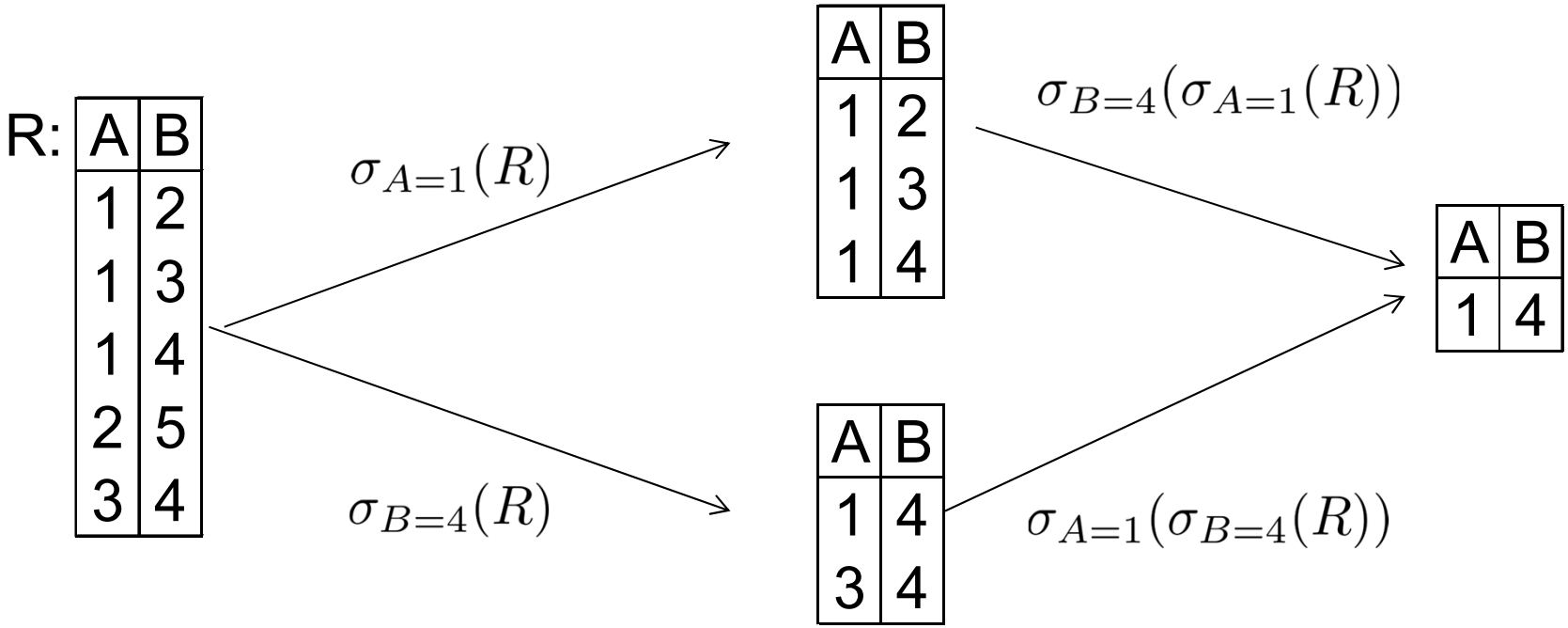
$\sigma_{A=1}(R)$

A	B
1	2
1	3

- Conditions: Attributes and constants using =, ≠, ≤, <, ≥ or >, linking and combining of multiple conditions using ∧, ∨ or ¬ and (,).

Selection: Commutativity

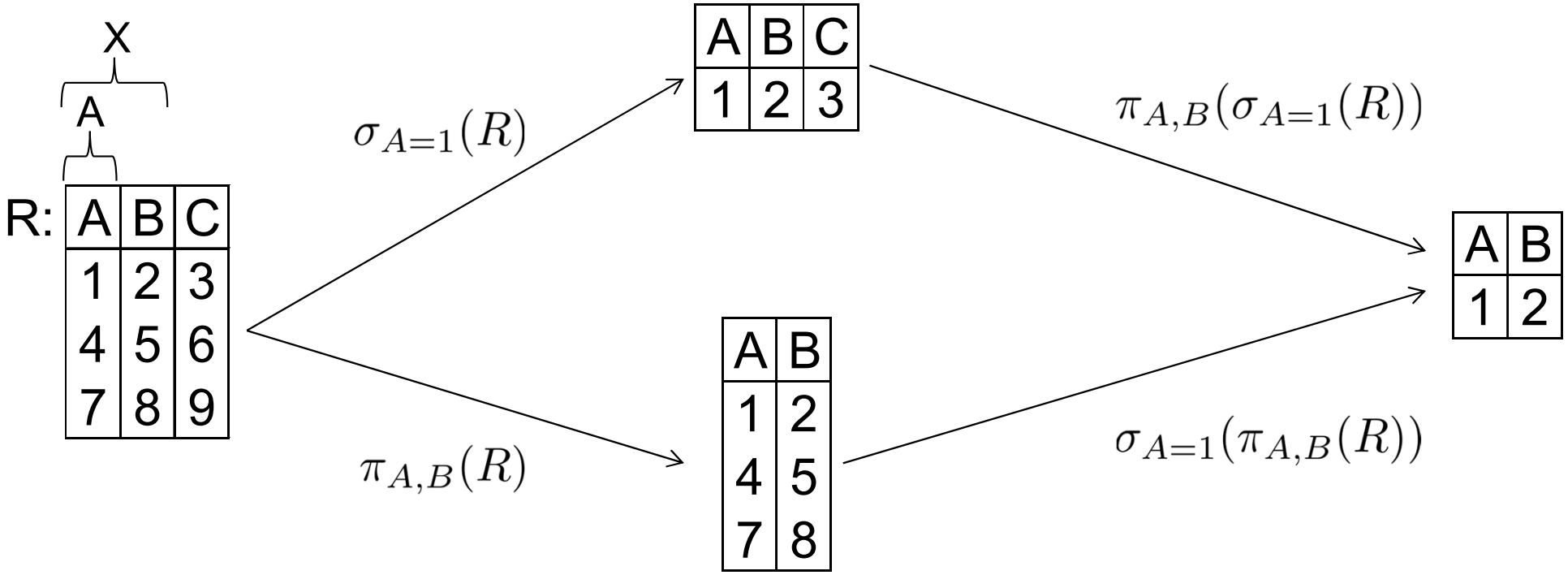
$$\sigma_{A=a}(\sigma_{B=b}(R)) = \sigma_{B=b}(\sigma_{A=a}(R))$$



- Optimization: selection is commutative and sometimes changing the order of execution can be more efficient

Selection: Transposition

- If $A \in X, X \subseteq R : \pi_X(\sigma_{A=a}(R)) = \sigma_{A=a}(\pi_X(R))$



- Optimization: Sometimes first select and then project can be more efficient

Cross product ×

- Cross product × (Cartesian product, cross join): links two tables by combining each tuple of the first with each tuple of the second.
  - Be careful: result for tables with *n* or *m* tuples has *n*m* tuples!
- Syntax <Relation1> × <Relation2>
- Semantics $R \times S := \{x_1, \ldots, x_n, \ldots, x_{n+m} | x_1, \ldots, x_n \in r, x_n, \ldots, x_{n+m} \in s\}$
- Example:

R:

A	B
1	2
3	4

S:

C	D
5	6
7	8

R × S:

A	B	C	D
1	2	5	6
1	2	7	8
3	4	5	6
3	4	7	8

Attributes with the same name are uniquely identified by prefixing the relations name

But how do I generate relations from base-relations?

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Conceptual Schema

Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23

„I can recommend all students as tutors, that have a grade better than 2.0“

External Schema

$$\sigma, \quad \pi, \quad \times$$

Let us start with $Students \times Grade$

Another example using renaming

„I need a list of persons, their children and grandchildren“

R:

Person	Child
Charles the Great	Louis the Pious
Louis the Pious	Lothar the I
Louis the Pious	Charles the Bald
Lothar the I	Louis the II

$$\sigma, \quad \pi, \quad \times$$

Set operations: union

- Union of two relations: collects the tuple sets of two relations under a common schema
- Attribute sets of both relations must be **identical**, we care for attribute names, we do not care for attribute order.
- Example:

R:

A	B	C
1	2	3
4	5	6

S:

A	B	C
4	5	6
7	8	9

R ∪ S:

A	B	C
1	2	3
4	5	6
7	8	9

Example Union

Students:

S_ID	Name	Birthday
1	Max	14.05.2001
2	Miri	17.04.2000
3	Sarah	18.09.2002
4	Ben	07.06.2002

Grade:

S_ID	S_Grade	Course	Semester
1	1.0	DB	WS23
2	2.3	DB	WS23
2	1.3	OOP	SS23
3	4.0	OOP	SS22
4	3.3	DB	WS23

„All student IDs that did pass DB better than 2.0 or are born before 2001“

Set operations: difference

- Difference eliminates the tuples from the first relation that also occur in the second relation
- Attribute sets of both relations must be identical
- Example:

R:

A	B	C
1	2	3
4	5	6

S:

A	B	C
4	5	6
7	8	9

R - S:

A	B	C
1	2	3

Set operations: intersection

- Intersection returns the tuples that are common to both relations
- Attribute sets of both relations must be identical
-
- Example:

R:

A	B	C
1	2	3
4	5	6

S:

A	B	C
4	5	6
7	8	9

R ∩ S:

A	B	C
4	5	6

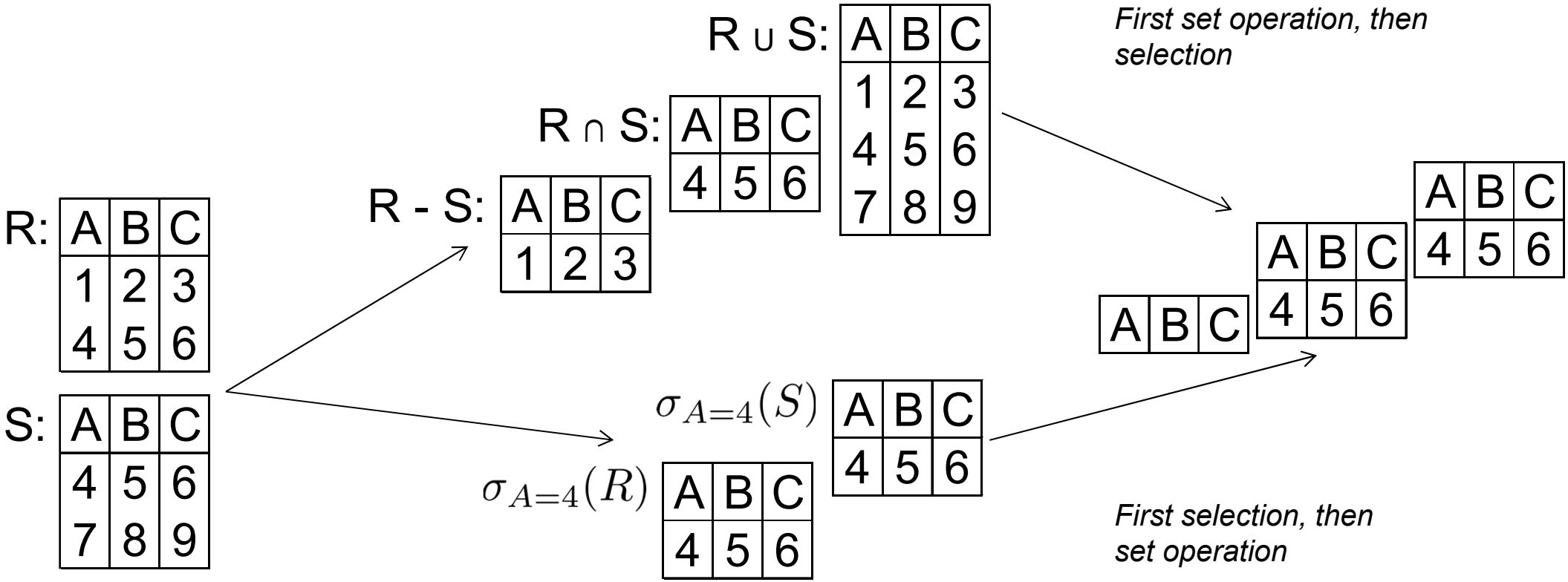
- Intersection can be replaced by difference

$$R \cap S = R - (R - S)$$

Set operations: Laws for transformation

- Distributivity regarding ∩, ∪, -

$$\sigma_{A=4}(R \cup S) = \sigma_{A=4}(R) \cup \sigma_{A=4}(S)$$



Relational algebra

- **Hide columns**: Projection π
- **Search for rows**: Selection σ
- **Link tables**: cross product $\times$
- **Unify tables**: Union $\cup$
- **Subtract tables from each other**: Difference and intersection $\cap$

Application example

„All bicycles that are produced in DE and 22" in size.“

Bike (B)

BName	BManufacturer	BSize
Stereo 150	Cube	22
Balance bike	Puky	8

Manufacturer (M)

MName	MCountry
Cube	49
Puky	49
Yeti Cycles	1

Country (C)

CCode	CName
49	DE
1	USA

Independence and completeness

- A query language is called relationally complete if every relational algebra operation in the language can be executed by (one or more) commands
- There is a minimal set of operations within the relational algebra from which all other operations can be composed: $\Omega = \pi, \sigma, \times, \cup$ and -
  - Ω is independent: no operator can be omitted without losing completeness
  - Other independent, complete sets: replace $\times$ with $\triangleright\triangleleft$ and β
- Thus: it is sufficient to show that all operations from Ω can be expressed in a query language to show that this is relationally complete
- SQL is relationally complete!

Discussion

- Which structural elements of the relational data model are we familiar with?
- What relational algebra operations are there?
- Which set of relational algebra operations is relationally complete and independent? Why is this important?
- Which integrity constraints do we know from the relational data model?