Technische Hochschule Rosenheim

Chapter 2 – Relational models and relational algebra

Databases lectures

Dr Kai Höfig

Discussion/revision

- What is a database and what is the motivation to use it?
- What are the advantages of data storage in relations?
- What is meant by the schema architecture?
- What are the architectural patterns for databases and what are they good for?
- What is meant by a data model?

Overview

- 2.1 Relational data model
- 2.2 Integrity constraints
- 2.3 Relational algebra

The relational data model

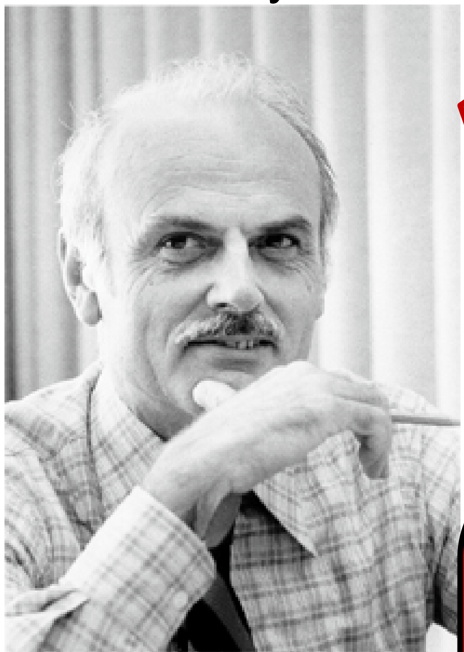
- Structure of the data:
  Data is stored in relations (tables)

- Operations on the data: 2 alternatives
  1. Relational algebra: practically implemented
     in SQL language (Structured Query Language)
  2. Relational calculus: practically implemented
     in QBE (Query by Example) / QBC (Query by Criteria) language

- Integrity constraints: lots and lots! 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

Relational algebra, calculus and languages

Theory



In the 1960s and 1970s, 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
Tuple relational calculus

Domain relational calculus

Logical foundation

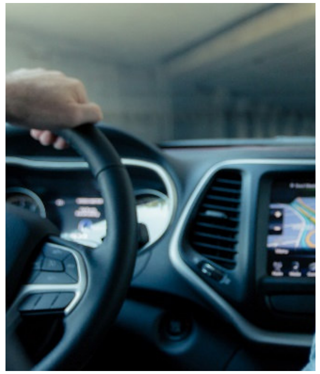
Without understanding the logical foundation, we interact with an unknown communication partner using the database languages

Practicable database languages

SQL

QBE

Practice



Relational algebra, calculus and languages - script

- Without a logical foundation, it is not possible to design good languages for practice.

- Why do we need the relational algebra and two forms of calculus?
  - Tuple relational calculus is the basis of SQL → important for understanding SQL
  - Domain relational calculus is the basis of QBE (and also QBC) → important for understanding QBE
  - Calculuses are declarative, i.e. the calculation sequence is not visible, i.e.
    - 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.
  - 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).

WINES

WineID	Name	Colour	Vintage	Vineyard
1042	La Rose GrandCru	Red	1998	Chateau La Rose
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
3478	Pinot Noir	Red	1999	Helena
4711	Riesling Reserve	White	1999	Müller
4961	Chardonnay	White	2002	Bighorn

PRODUCER

Vineyard	Growing_area	Region
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Chateau La Rose	Saint-Emilion	Bordeaux
Chateau La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hesse
Bighorn	Napa Valley	California

RECOMMENDATION

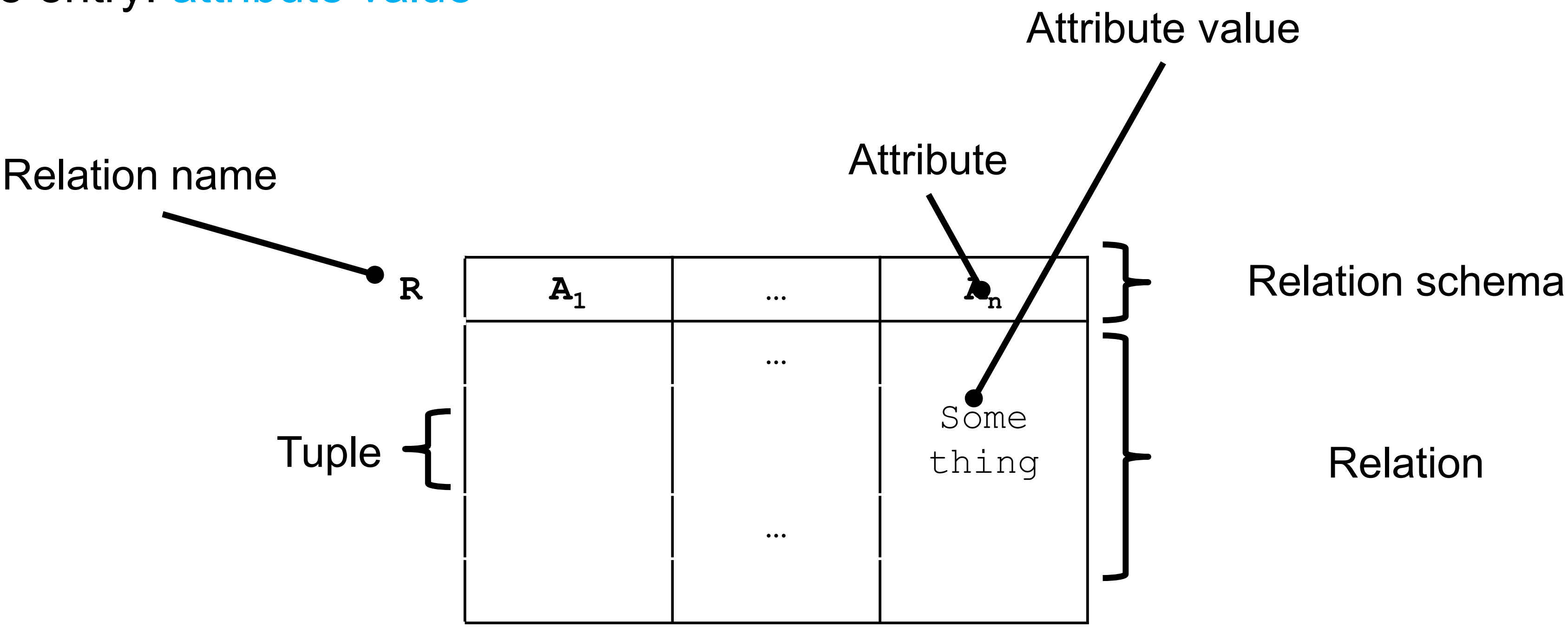
Wine
La Rose Grand Cru
Riesling Reserve
Merlot Selection
Sauvignon Blanc

WINE_LIST

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve

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.

- $R=\{A_1,..,A_k\}$ is a **relational schema** with the identifier R via the attributes $A_1,..,A_k$ to the value ranges $D_1,..,D_s$ with
  $dom: \{A_1,..,A_k\} \rightarrow \{D_1,..,D_s\}$, $s \geq 1$, the value range function.
  - Example:
    `PRODUCER` = { `Vineyard, Growing_area, Region` } with
    dom(`Vineyard`) = `string`, *dom*(`Growing_area`) = `string`, *dom*(`Region`) = `string`

- A **relational database schema** is a finite non-empty set $S = \{ R_1(\alpha_1),.., R_m(\alpha_m) \}$ of relational schemas via subsets of a common attribute set $\alpha = \alpha_1,.., \alpha_m$

- Thereby, all identifiers of relations are different in pairs and different from all attribute identifiers.

Definition of the relational data model (2)

- A **relation** r via a relational schema $R=\{A_1,..,A_k\}$ in short $r(R)$ is a finite set of tuples that map each attribute A_j to a value from $dom(A_j)$.
  - Example: a potential relation r via the relational schema `PRODUCER` is with

t_2(`Vineyard`)=`'Helena'` t_2(`Growing_area`)=`'Napa Valley'` t_2(`Region`)=`'California'`
t_3(`Vineyard`)=`'Müller'` t_3(`Growing_area`)=`'Rheingau'` t_3(`Region`)=`'Hesse'`

- A **database** via a database schema $S = R_1(\alpha_1),.., R_m(\alpha_m)$ is a set of relations $d:=\{r_1,..,r_p\}$ where each relation r_i is defined via the relational schema R_i : $R_i(R_i)$
- A relation $r \in d$ is referred to as a **base relation**
  - Example: Our database schema WineDB contains 4 base relations, which we formally refer to as r1(WINES), r2(PRODUCER), r3(RECOMMENDATION), r4(WINE_LIST)

Attendance exercise 1

WINES

<u>WineID</u>	Name	Colour	Vintage	Vineyard → PRODUCER
1042	La Rose GrandCru	Red	1998	Chateau La Rose
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
3478	Pinot Noir	Red	19[illegible]	Helena
4711	Riesling Reserve	Whit[illegible]	[illegible]999	Müller
4961	Chardonnay	[illegible]e	2002	Bi[illegible]orn

Vineyard in WINES uses → foreign keys

PRODUCER

<u>Vineyard</u>	[illegible]owing_area	Region
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Chateau La Rose	Saint-Emilion	Bordeaux
Chateau La Pointe	Pomerol	[illegible]
Müller	Rheingau	[illegible]
Bighorn	[illegible]	[illegible]

Vineyard is key in PRODUCER

Key constraint: There are no 2 tuples in PRODUCER with the same value of Vineyard

Referential integrity: Every value of Vineyard in WINES is available in PRODUCER

But how do I get to views?

WINES

WineID	**Name**	**Colour**	**Vintage**	**Vineyard → PRODUCER**
1042	La Rose GrandCru	Red	1998	Chateau La Rose
2168	Creek Shiraz	Red	2003	Creek

“I can generally recommend wines from California, no matter which region they are from

RECOMMENDATION

Wine	**Colour**	**Vintage**	**Region**
Zinfandel	Red	2004	Napa Valley
Chardonnay	White	2002	Napa Valley
Pinot Noir	Red	1999	Napa Valley

Chateau La Rose	Saint-Emilion	Bordeaux
Chateau La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hesse
Bighorn	Napa Valley	California

Query operations on tables

- Relational algebra: set of basic operations on relations to compute new (result) relations
  - can be combined in any way
  - thereby create an algebra for “calculating with tables”

- 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, Join

A	B	C
a1	b1	c1
a2	b1	c2
a3	b2	c1
a4	b3	c3
a5	b1	c1

Projection

A	B	C
a1	b1	c1
a2	b1	c2
a3	b2	c1
a4	b3	c3
a5	b1	c1

Selection

A	B
a1	b1
a2	b1
a3	b2
a4	b3
a5	b1

B	C
b1	c5
b2	c6
b4	c5

Join

A	B	C
a1	b1	c5
a2	b1	c5
a3	b2	c6
a5	b1	c5

Join

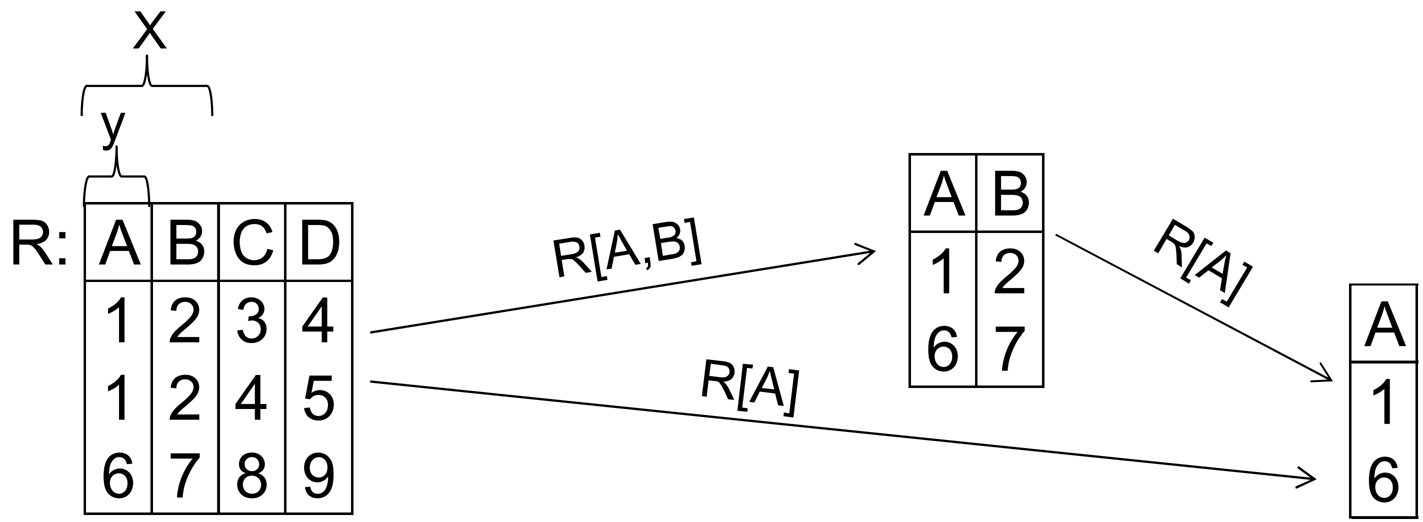
Projection

- Semantics

$$\pi_X(r) := \{ t(X) \mid t \in r \}$$

for $r(R)$ and $X \subseteq R$ attribute set in R

- Property for $Y \subseteq X \subseteq R$: $\quad \pi_Y(\pi_X(r)) = \pi_Y(r)$



- Note: π removes duplicates (set semantics)

Projection: further examples

- Relation

PRODUCER

Vineyard	**Growing_area**	**Region**
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Chateau La Rose	Saint-Emilion	Bordeaux
Chateau La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hesse
Bighorn	Napa Valley	California

- $\pi_{\mathrm{Region}}(\mathrm{PRODUCER})$

Region
South Australia
California
Bordeaux
Hesse

$\pi_{\mathrm{Growing\_area,Region}}(\mathrm{PRODUCER})$

Growing_area	**Region**
Barossa Valley	South Australia
Napa Valley	California
Saint-Emilion	Bordeaux
Pomerol	Bordeaux
Rheingau	Hesse

Selection σ definition

- Selection σ (Sigma): selection of rows of a table based on a selection predicate
- Syntax: $\sigma_{<Constraint>}$(<Relation>) or <Relation>[<Constraint>]
- Semantics (for $A \in R$)

$$\sigma_{A=a}(r) := \{ t \in r \mid t(A) = a \}$$

- Example:

$\sigma_{A=1}$(R) or R[A=1]

R:

A	B
1	2
3	4

R[A=1]:

A	B
1	2

Selection conditions

- **Constant selection**

  $$\texttt{Attribute}\ \theta\ \texttt{Constant}$$

  Boolean predicate θ is = or $\neq$, for linear value ranges also $\leq$, <, $\geq$ or >

- **Attribute selection**

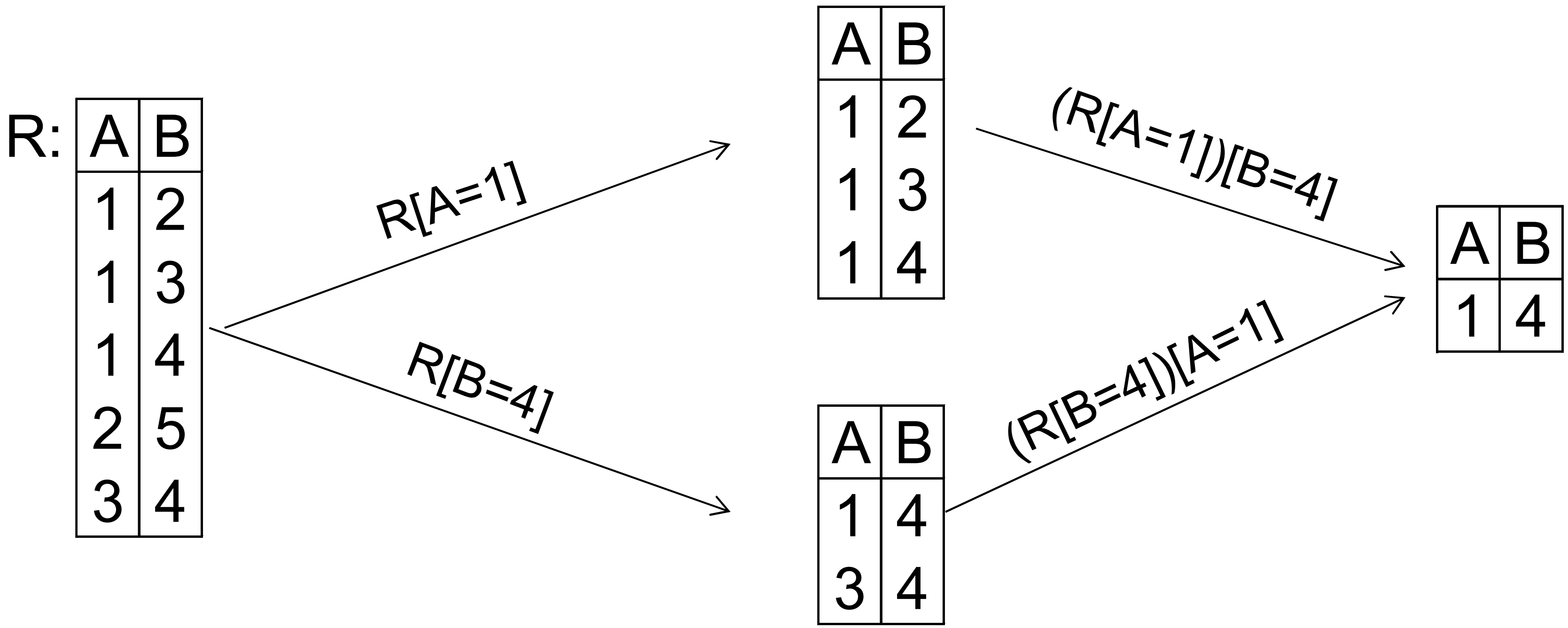
  $$\texttt{Attribute}_1\ \theta\ \texttt{Attribute}_2$$

- **Logical linking** of multiple constant selections or attribute selections with $\wedge$, $\vee$ or $\neg$.

Selection: Laws for transformation - commutativity

- Commutativity

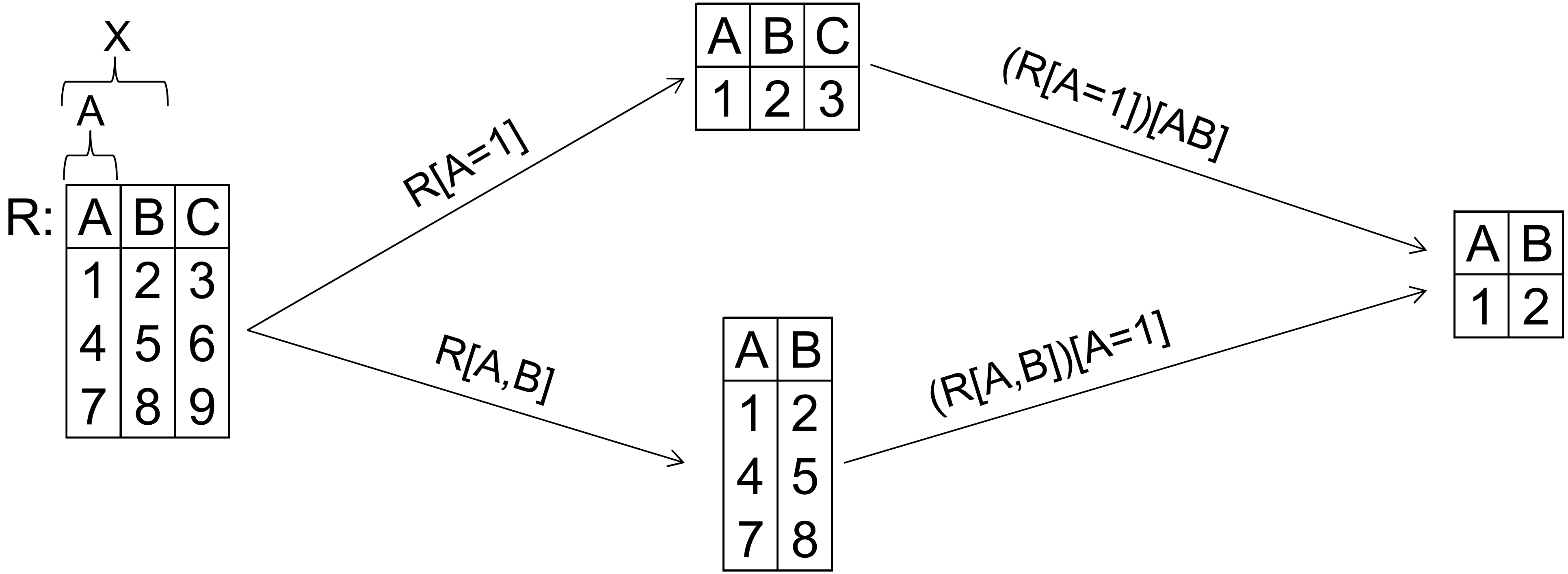
$$\sigma_{A=a}(\sigma_{B=b}(r)) = \sigma_{B=b}(\sigma_{A=a}(r))$$



Selection: Laws for transformation - transposition

- If $A \in X$, $X \subseteq R$

$$\pi_X(\sigma_{A=a}(r)) = \sigma_{A=a}(\pi_X(r))$$



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: $R \times S := \{ x_1..x_n..x_{n+m} \mid R(x_1,..,x_n) \wedge S(x_{n+1},..,x_{n+m}) \}$

- 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 renamed

Cross product × example

- Example:

$$\text{WINES} \times \text{BOTTLE}$$

BOTTLE

Type	Contents
Normal	700
Small	375

WineID	Name	Colour	Vintage	Vineyard	Type	Contents
1042	La Rose GrandCru	Red	1998	Chateau La Rose	Normal	700
1042	La Rose GrandCru	Red	1998	Chateau La Rose	Small	375
2168	Creek Shiraz	Red	2003	Creek	Normal	700
2168	Creek Shiraz	Red	2003	Creek	Small	375
3456	Zinfandel	Red	2004	Helena	Normal	700
3456	Zinfandel	Red	2004	Helena	Small	375
…	…	…	…	…	…	…

Natural join $\bowtie$

- Natural join: $\bowtie$ links tables via columns of the same name, by merging two tuples if they have the same values there
  - Tuples that do not find a partner (dangling tuples) are eliminated.

- Syntax: <Relation1> $\bowtie$ <Relation2>

- Semantics for *A* attributes of *R*, *C* attributes of *S* and *B* attributes with intersection

$$R \bowtie S := \pi_{A1,..,Am,R.B1,..,R.Bk,S1,..,Sn}(\sigma_{R.B1=S.B1 \wedge .. \wedge R.Bk=S.Bk}(R \times S))$$

- Example:

R:

A	B
1	2
3	4
9	4
~~5~~	~~6~~

S:

B	C
2	3
2	4
4	5
~~7~~	~~8~~

R $\bowtie$ S:

A	B	C
1	2	3
1	2	4
3	4	5
9	4	5

Natural join example ▷◁

- Example:

  WINES ▷◁ PRODUCER

WineID	Name	...	Vineyard	Growing_area	Region
1042	La Rose GrandCru	...	Chateau La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	South Australia
3456	Zinfandel	...	Helena	Napa Valley	California
2171	Pinot Noir	...	Creek	Barossa Valley	South Australia
3478	Pinot Noir	...	Helena	Napa Valley	California
4711	Riesling Reserve	...	Müller	Rheingau	Hesse
4961	Chardonnay	...	Bighorn	Napa Valley	California

  - The “Château La Pointe” vineyard has disappeared from the result

Join: Laws for transformation

$R \bowtie S := \pi_{A1,..,Am,R.B1,..,R.Bk,S1,..,Sn}(\sigma_{R.B1=S.B1 \wedge .. \wedge R.Bk=S.Bk}(R \times S))$

- From $R_1 \cap R_2 = \{\}$ follows $r_1 \bowtie r_2 = r_1 \times r_2$.
  If there are no attributes with intersection, the condition σ of the definition of the natural join is removed.

- Commutativity: $R_1 \bowtie r_2 = r_2 \bowtie r_1$
  The conditions and projections in the definition are commutative.

- Associativity: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$

  therefore allowed: $\bowtie^{p}_{i=1} r_i$

Renaming β

- Renaming β (Beta): adjusting and renaming attributes

- Syntax:

  $\beta_{\text{<Target attribute>} \leftarrow \text{<Source attribute>}}(\text{<Relation>})$ or

  $\text{<Relation>}[\text{Source attribute} \rightarrow \text{Target attribute}]$

- Semantics

  $$\beta_{B \leftarrow A}(r) := \{\, t' \mid \exists\, t \in r : t'(R\text{-}A) = t(R\text{-}A) \wedge t'(B) = t(A) \,\}$$

- Example:

R:

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

R[A→X]:

X	B	C	D
1	2	3	4
6	7	8	9

- Now possible by renaming
  - joins, where Cartesian products were previously carried out (different attributes are named the same),
  - Cartesian products, where previously joins were carried out (same attributes are named differently),
  - set operations

Renaming example

- How can the operations presented so far be used to create a table with the people, their children and their grandchildren?

R:

Person	Kind
Karl der Große	Ludwig der Fromme
Ludwig der Fromme	Lothar der I
Ludwig der Fromme	Karl II der Kahle
Lothar der I	Ludwig der II

R':

Person	Kind	Enkel
Karl der Große	Ludwig der Fromme	Lothar der I
Karl der Große	Ludwig der Fromme	Karl II der Kahle
Ludwig der Fromme	Lothar der I	Ludwig der II

- $R \rhd\lhd \ (\beta_{Child \leftarrow Person}(\beta_{Grandchild \leftarrow Child}(R)))$ or
  $R \rhd\lhd \ ((R[Child \rightarrow Grandchild])[Person \rightarrow Child])$

Calculation of the cross product from natural join

- Natural join degenerates to a cross product if no common attributes exist
- Forcing by renaming: $R_1(A, B, C)$ and $R_2(C, D)$

$$R_1 \times R_2 \equiv R_1 \bowtie \beta_{E \leftarrow C}(R_2)$$

- Cross product + selection simulates natural join

$$R_1 \bowtie R_2 \equiv \sigma_{R_1.C=R_2.C}(R_1 \times R_2)$$

Discussion

- Which structural elements of the relational data model are we familiar with?
- What relational algebra operations are there?

Combination of operations

- Combinations of operations are possible

- Example

$$\pi_{\text{Name,Colour,Vineyard}}(\sigma_{\text{Vintage>2000}}(\text{WINES}) \triangleright\triangleleft \; \sigma_{\text{Region=“California”}}(\text{PRODUCER}))$$

$$(\text{WINES[Vintage>2000]} \quad \triangleright\triangleleft \; \text{PRODUCER[Region=“California”])[Name,Colour,Vineyard]}$$

results in

Name	**Colour**	**Vineyard**
Zinfandel	Red	Helena
Chardonnay	White	Bighorn

Combination of operations

- Using brackets for the expression is important!
- Example

  $\pi_{\text{Name,Colour,Vineyard}}(\sigma_{\text{Vintage>2000}}(\text{WINES})) \rhd\lhd \ \sigma_{\text{Region=“California”}}(\text{PRODUCER})$

  (WINES[Vintage>2000])[Name,Colour,Vineyard] $\rhd\lhd$ PRODUCER[Region= “California”]

  results in

Name	**Colour**	**Vineyard**	**Growing_area**	**Region**
Zinfandel	Red	Helena	Napa Valley	California
Chardonnay	White	Bighorn	Napa Valley	California

Set operations: union

- Union $r_1 \cup r_2$ of two relations r_1 and r_2: collects the tuple sets of two relations under a common schema
- Attribute sets of both relations must be identical
- Semantics: for $r_1(R)$ and $r_2(R)$

$$r_1 \cup r_2 := \{ t \mid t \in r_1 \vee t \in r_2 \}$$

- 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 of union

WINE_LIST

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve

RECOMMENDATION

Wine
La Rose Grand Cru
Riesling Reserve
Merlot Selection
Sauvignon Blanc

- WINE_LIST $\cup$ $\beta_{\text{Name}\leftarrow\text{Wine}}$(RECOMMENDATION)

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve
Merlot Selection
Sauvignon Blanc

Set operations: difference

- Difference r_1 - r_2 eliminates the tuples from the first relation that also occur in the second relation

- Semantics: for $r_1(R)$ and $r_2(R)$

$$r_1 - r_2 \quad := \{ \, t \mid t \in r_1 \land t \notin r_2 \, \}$$

- 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

Example of difference

- WINE_LIST - $\beta_{Name \leftarrow Wine}$(RECOMMENDATION) results in

Name
Creek Shiraz
Zinfandel
Pinot Noir

Set operations: intersection

- Intersection $r_1 \cap r_2$: returns the tuples that are common to both relations

- Semantics: for $r_1(R)$ and $r_2(R)$

$$r_1 \cap r_2 := \{ t \mid t \in r_1 \wedge t \in r_2 \}$$

- 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 $\cap$ due to $r_1 \cap r_2 = r_1 - (r_1 - r_2)$ is superfluous

Example intersection

- WINE_LIST $\cap$ $\beta_{\text{Name} \leftarrow \text{Wine}}$(RECOMMENDATION) results in

Name
La Rose Grand Cru
Riesling Reserve

Set operations: Laws for transformation

- Distributivity regarding $\cap$, $\cup$, -

$$\sigma_{A=a}(r \cup s) = \sigma_{A=a}(r) \cup \sigma_{A=a}(s)$$

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

R ∩ S:

A	B	C
4	5	6

R ∪ S:

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

First set operation, then selection

A	B	C

A	B	C
4	5	6

A	B	C
4	5	6

S[A=4]:

A	B	C
4	5	6

R[A=4]:

A	B	C
4	5	6

First selection, then set operation

Relational algebra

- **Hide columns**: Projection π
- **Search for rows**: Selection σ
- **Link tables**: Join $\bowtie$ and cross product $\times$
- **Unify tables**: Union $\cup$
- **Subtract tables from each other**: Difference and intersection $\cap$
- **Rename columns**: Rename β
  (important for $\bowtie$ and $\cup$, -)

Application example

- All bicycles that are produced in Germany and 22" in size.

Bike (B)

BName	BManufacturer	BSize
Stereo 150	Cube	22
Balance bike	Puky	8

Manufacturer (M)

MName	MCountry
Bicicomp	49
Puky	49
Yeti Cycles	1

Country (C)

CCode	CName
49	DE
1	USA

With cross product:

$$\pi_{FName}(\sigma_{\substack{BSize=22 \\ \wedge\ CName=DE \\ \wedge\ BManufacturer=MName \\ \wedge\ MCountry=CCode}}(B \times M \times C))$$

With natural join:

$$\pi_{FName}(\sigma_{\substack{BSize=22 \\ \wedge\ CName=DE}}(\beta_{MName \leftarrow BManufacturer}(B) \bowtie \beta_{CCode \leftarrow MCounty}(M) \bowtie C))$$

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: $\Omega = \pi, \sigma, \bowtie, \beta, \cup$ and -
  - Ω is independent: no operator can be omitted without losing completeness
  - Other independent, complete sets: replace $\bowtie$ and β with $\times$

- 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?