Part VII

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Query Basics: Algebra & Calculus

Learning goals for today . . .

- Understanding of formal basics of relational query languages

Learning goals for today . . .

- Understanding of formal basics of relational query languages
- Knowledge to formalize queries with relational algebra

Learning goals for today . . .

- Understanding of formal basics of relational query languages
- Knowledge to formalize queries with relational algebra
- Knowledge to formalize calculus queries

Criteria for Query Languages

Introduction

- so far:
  - Relation schemata with basic relations that are saved in databases
- now:
  - "derived" relation schemata with virtual relations that are calculated from basis relations (basis relations keep unchanged)

Terms

- Query: Sequence of operations that calculate a result relation from basis relations
  - Present result relations interactively on a monitor or
  - further processing via program ("Embedding")
- View: Sequence of operations that is long-term saved under a view name and that can be called again with this name; results in a view relation
- **Snapshot:** Result relation of a query that is stored under a Snapshot-Name, but that is never calculated twice (with changed basis relations) (e.g., annual balance sheets)

Criteria for Query Languages

- **Ad-Hoc-Formalization**: User should be able to formalize a query without a need to write a complete program
- **Descriptiveness**: User should formalize "What do I want?" instead of "How do I get what I want?"
- **Collection-based**: Each operation should operate on a collection of data at the same time, not navigating on single elements ("one-tuple-at-a-time")
- **Closure**: Result is again a relation that can be used as input for the next query

Criteria for Query Languages /2

- **Adequacy**: All constructs of the underlying data model are supported
- **Orthogonality**: Language constructs are in similar situations also similar applicable
- **Optimizability**: Language consists of few operations for which optimization rules exist
- **Efficiency**: Any operation is efficient executable (in the relation model each operation has a complexity $\leq O(n^2), n$ number of tuples of the relation).

Criteria for Query Languages /3

- **Safety**: No query, that is syntactically correct, may result in an endless loop or give an infinite result
- **Limitation**: (comes from Safety, Optimizability, Efficiency) Query language should not be a complete programming language
- **Completeness**: The language must at least being able to express the queries of a standard language (such as the to be introduced relation algebra of this chapter or a safe relational calculus)

Query Algebras

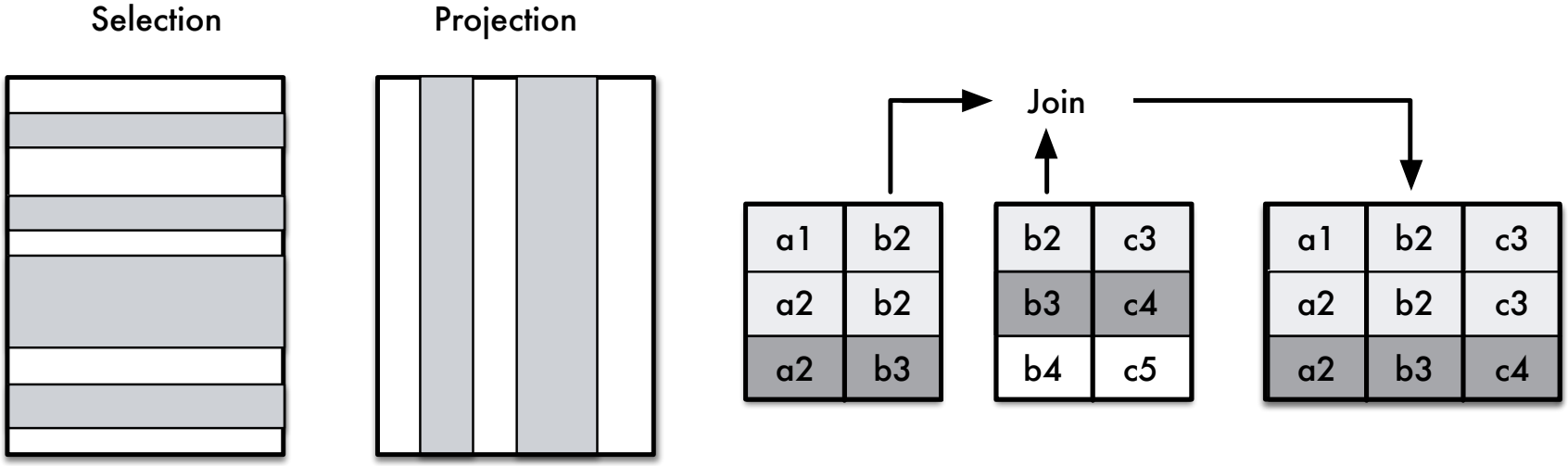
Query Algebra

- Mathematics: Algebra defined as a range of values and on this defined operators
- For database queries: Contents of the database are values, and operators define functions for the calculation of query results
  - Relational Algebra
  - Algebra Extensions

Relational Algebra

- **Hide Columns**: Projection π
- **Search Rows**: Selection σ
- **Joining Tables**: Join $\bowtie$
- **Union of Tables**: Union $\cup$
- **Subtract Tables from each other**: Difference $-$
- **Rename Columns**: Renaming β
  (important for $\bowtie$ and $\cup, -$)

Relational Algebra: Overview



Projection

- Syntax

$$\pi_{\mathit{AttributeSet}}\,(\mathit{Relation})$$

- Semantics

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

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

- Property for $Y \subseteq X \subseteq R$

$$\pi_Y(\pi_X(r)) = \pi_Y(r)$$

- Attention: π removes duplicates (Set Semantics)

Projection: Example

$$\pi_{\texttt{Region}}(\texttt{PRODUCER})$$

Region
South Australia
Kalifornien
Bordeaux
Hessen

Projection: Example 2

$$\pi_{\texttt{District},\texttt{Region}}(\texttt{PRODUCER})$$

District	**Region**
Barossa Valley	South Australia
Napa Valley	Kalifornien
Saint-Emilion	Bordeaux
Pomerol	Bordeaux
Rheingau	Hessen

Selection

- Syntax

$$\sigma_{\mathit{Condition}}(\mathit{Relation})$$

- Semantics (for $A \in R$)

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

Selection Conditions

- **Constant Selection**

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

  boolean predicate θ is $=$ or $\neq$, for linear ordered range of values also $\leq$, $<$, $\geq$ or $>$

- **Attribute Selection**

$$\texttt{Attribute1}\ \theta\ \texttt{Attribute2}$$

- logic connection of multiple Constant- or Attribute-Selections with $\wedge$, $\vee$ or $\neg$

Selection: Properties

- Commutativity

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

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

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

- Distributivity respect. $\cup$, $\cap$, $-$

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

Selection: Example

$$\sigma_{\texttt{Vintage}>2000}(\texttt{WINES})$$

WineID	Name	Color	Vintage	Vineyard
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
4961	Chardonnay	White	2002	Bighorn

Join

- Syntax of the natural join

$$\texttt{Relation1} \bowtie \texttt{Relation2}$$

- Semantics

$$\begin{aligned} r_1 \bowtie r_2 \quad := \quad & \{t \mid t(R_1 \cup R_2) \wedge \\ & [\forall i \in \{1,2\} \exists t_i \in r_i : t_i = t(R_i)]\} \end{aligned}$$

- Join links tables over equally named columns at equal attribute values

Join: Properties

- Schema for $r(R) \bowtie r(S)$ is union of the attribute sets $RS = R \cup S$
- from $R_1 \cap R_2 = \{\}$ follows $r_1 \bowtie r_2 = r_1 \times r_2$
- Commutativity: $r_1 \bowtie r_2 = r_2 \bowtie r_1$
- Associativity: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$
- thus allowed:

$$\bowtie_{i=1}^{p} r_i$$

Join: Example

$$\texttt{WINES} \bowtie \texttt{PRODUCER}$$

WineID	Name	...	Vineyard	District	Region
1042	La Rose Grand Cru	...	Ch. La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	South Australia
3456	Zinfandel	...	Helena	Napa Valley	Kalifornien
2171	Pinot Noir	...	Creek	Barossa Valley	South Australia
3478	Pinot Noir	...	Helena	Napa Valley	Kalifornien
4711	Riesling Reserve	...	MÃ¼ller	Rheingau	Hessen
4961	Chardonnay	...	Bighorn	Napa Valley	Kalifornien

Renaming

- Syntax

$$\beta_{new \leftarrow old}(Relation)$$

- Semantic

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

- changes attribute names from *old* to *new*

$$\beta_{\text{Name} \leftarrow \text{LastName}}\ (\texttt{CRITIC})$$

- with renaming now possible
  - Join, where Cartesian products were applied (different attributes get equal naming),
  - Cartesian products, where Joins were applied (equal attributes get different naming),
  - Set operations

Calculation of the Cross Product

- Natural Join degenerates to a cross product, when no shared attributes exist
- Enforce by renaming
  - Example: $R1(A, B, C)$ and $R2(C, D)$

$$R1 \times R2 \equiv R1 \bowtie \beta_{E \leftarrow C}(R2)$$

- Cross product + selection simulates natural join

$$R1 \bowtie R2 \equiv \sigma_{R1.C=R2.C}(R1 \times R2)$$

Set Operations: Semantics

- formal for $r_1(R)$ and $r_2(R)$
  - Union $r_1 \cup r_2 := \{t \mid t \in r_1 \vee t \in r_2\}$
  - Intersection $r_1 \cap r_2 := \{t \mid t \in r_1 \wedge t \in r_2\}$
  - Difference $r_1 - r_2 := \{t \mid t \in r_1 \wedge t \notin r_2\}$
- Intersection $\cap$ is superfluous as $r_1 \cap r_2 = r_1 - (r_1 - r_2)$

Independence and Completeness

- Minimal relational algebra:

$$\Omega = \pi, \sigma, \bowtie, \beta, \cup \text{ and } -$$

- independence: no operator can be left off without loosing completeness
- other independent set: $\bowtie$ and β replaced by $\times$
- relational completeness: every other set of operators with same expressive power as Ω
- strict relational completeness: for any expression with operators out of Ω there is an equivalent expression also with the other set of operations

Relational Algebra: Extensions

Relational Algebra: Extensions

- further Join operations
- Division
- Grouping and nested grouping relations
- . . .

Join Variants

- for $L(AB)$, $R(BC)$, $S(DE)$
- Equi-Join: Equality condition over explicit specified and possibly different attributes

$$r(R) \bowtie_{C=D} r(S)$$

- Theta-Join (θ-*join*): arbitrary join condition

$$r(R) \bowtie_{C>D} r(S)$$

- Semi-Join: only attributes of one operand appear in the result

$$r(L) \ltimes r(R) = \pi_L(r(L) \bowtie r(R))$$

- Outer Join

Outer Join

- Adoption of "dangling tuples" into the result and fill up with null values
- **Full Outer Join** takes all tuples of both operands

$$r \mathbin{⟗} s$$

- **Left Outer Join** takes all tuples of the left operand

$$r \mathbin{⟕} s$$

- **Right Outer Join** takes all tuples of the right operand

$$r \mathbin{⟖} s$$

Outer Join /2

LEFT

A	B
1	2
2	3

RIGHT

B	C
3	4
4	5

$\bowtie$

A	B	C
2	3	4

$\supset\bowtie\subset$

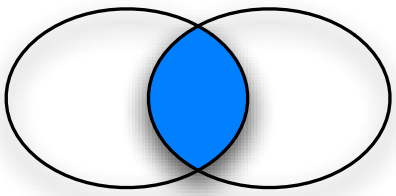
A	B	C
1	2	$\perp$
2	3	4
$\perp$	4	5

$\supset\bowtie$

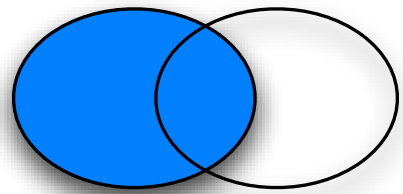
A	B	C
1	2	$\perp$
2	3	4

$\bowtie\subset$

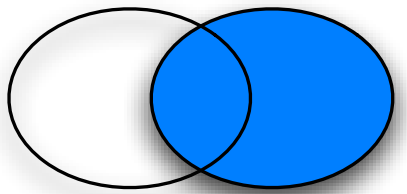
A	B	C
2	3	4
$\perp$	4	5



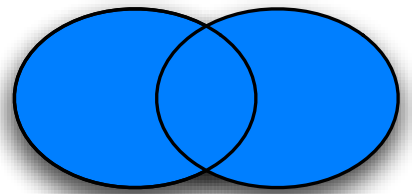
natural join



left outer join



right outer join



full outer join

Problem: Quantifiers

- Express universal quantification (allquantor) in relational algebra, even though quantification in the selection condition is not allowed
- Division (can be derived from Ω)
- $r_1(R_1)$ and $r_2(R_2)$ given with $R_2 \subseteq R_1$, $R' = R_1 - R_2$. Than is

$$\begin{aligned} r'(R') &= \{t \mid \forall t_2 \in r_2 \exists t_1 \in r_1 : t_1(R') = t \wedge t_1(R_2) = t_2\} \\ &= r_1 \div r_2 \end{aligned}$$

- Division of r_1 by r_2

$$r_1 \div r_2 = \pi_{R'}(r_1) - \pi_{R'}((\pi_{R'}(r_1) \bowtie r_2) - r_1)$$

Division Example I

WINE_RECOMMENDATION

Wine	Critic
La Rose Grand Cru	Parker
Pinot Noir	Parker
Riesling Reserve	Parker
La Rose Grand Cru	Clarke
Pinot Noir	Clarke
Riesling Reserve	Gault-Millau

GUIDES1

Critic
Parker
Clarke

GUIDES2

Critic
Parker
Gault-Millau

Division Example II

- Division with first table

$$\texttt{WINE\_RECOMMENDATION} \div \texttt{GUIDES1}$$

gives

Wine
La Rose Grand Cru
Pinot Noir

- Division with second critics table

$$\texttt{WINE\_RECOMMENDATION} \div \texttt{GUIDES2}$$

gives

Wine
Riesling Reserve

Term Division

- Analogy to the arithmetic operation of integer division

The integer division is in this sense the inverse to the multiplication by giving the result of the biggest number for which the multiplication with the divisor is smaller than the dividend.
Analogously holds: $r = r_1 \div r_2$ is the biggest relation for which $r \bowtie r_2 \subseteq r_1$ is valid.

Division in SQL

- Simulation of the Allquantor (Division)
- with double Negation:

```
select distinct Wine
from WINE_RECOMMENDATION w1
where not exists (
   select * from GUIDES2 g
   where not exists (
      select * from WINE_RECOMMENDATION w2
      where g.Critic = w2.Critic and w1.Wine = w2.Wine))
```

  - "Gives all wines, thus no wine exists that is not recommended by all critics in the relation `GUIDES2`".
  - (We use the relation `GUIDES2`, as the in the textbook stated result relation refers to this comparison relation.)

Grouping

- Grouping Operator γ:

$$\gamma_{f_1(x_1),f_2(x_2),\ldots,f_n(x_n);A}(r(R))$$

  - Extends the attribute scheme of $r(R)$ by new attributes that corresponds with the function application $f_1(x_1), f_2(x_2), \ldots, f_n(x_n)$
  - Application of the function $f_i(x_i)$ on the subset of the tuples of $r(R)$ which have the same attribute values for the attributes A

```
select f1(x1), f2(x2), ..., fn(xn); A
from R
group by A
```

Semantics of the Grouping Operator

- empty attribute set $A = \emptyset$:

$$\gamma_{F(X);\emptyset}(r(R)) = r(R) \times r(R)^{F(X)}$$

  with $r(R)^{F(X)}$ is relation with attribute $F(X)$ and a tuple as value of $F(X)$ on $r(R)$

- without function:

$$\gamma_{\emptyset;\emptyset}(r(R)) = r(R)$$

- general case:

$$\gamma_{F(X);A}(r(R)) = \bigcup_{t \in R} \gamma_{F(X);\emptyset}(\sigma_{A=t.A}(r(R)))$$

Query Calculi

Query Calculus

- Calculus: A formal logic language to formalize statements
- Goal: The use of such a calculus to formalize database queries
- Logic-based approach:
  - Database contents match a logical expression of a predicate logic
  - Query: derived predicates

A General Calculus

- Motivation: Mathematical Notation

$$\{x^2 \mid x \in \mathbb{N} \wedge x^3 > 0 \wedge x^3 < 1000\}$$

- Query has the form

$$\{f(\bar{x}) \mid p(\bar{x})\}$$

  - $\bar{x}$ denotes set of free variables

$$\bar{x} = \{x_1 : D_1, \ldots, x_n : D_n\}$$

A General Calculus /2

- Function f denotes result function over $\overline{x}$
  - Important special cases: Declaration of a variable itself (f is here the identity function) and tuple construction (result of type **tuple of**)
- p selection predicate over free variables $\overline{x}$
  - Terms of variables, constants, and function applications
  - Predicates of data types, such as $\leq$, $<$, $>$, $\geq$, ...
    $\rightarrow$ atomic formulas over terms
  - Relation to current database $\rightarrow$ database predicates, e.g., relation name in the RM
  - predicate logical operators $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$
    $\rightarrow$ formulas

Result Declaration of a Query

$$\bar{x} = \{x_1 : D_1, \ldots, x_n : D_n\}$$

1. Declare all assignments of free variables in $\bar{x}$, for which the predicate p gets true.
2. Apply function f on the values given by this assignment.

Under which circumstances do calculus queries give infinite results?

$\rightarrow$ Safety of queries

Relational Calculi

- **Domain Calculus**: Variables take values of elementary data type (*domains*)
- **Tuple calculus**: Variables vary over tuple values (according to the lines of a relation)

Tuple Calculus

Tuple Calculus

- Basics of SFW-Queries in SQL
- Variables are tuple valued
- Example:

$$\{w \mid w \in \texttt{WINES} \wedge w.\texttt{Color} = \texttt{'Red'}\}$$

- in SQL:

```
select *
from WINES w
where w.Name = ‘Red‘
```

Tuple Calculus: Examples

- constructed tuple

$$\{\langle w.\texttt{Name}, w.\texttt{Vineyard}\rangle \mid w \in \texttt{WINES} \wedge w.\texttt{Color} = \texttt{'Red'}\}$$

- Join

$$\{\langle e.\texttt{Vineyard}\rangle \mid e \in \texttt{PRODUCER} \wedge w \in \texttt{WINES} \wedge e.\texttt{Vineyard} = w.\texttt{Vineyard}\}$$

in SQL:

```
select e.Vineyard
from Producer e, WINES w
where e.Vineyard = w.Vineyard
```

- Nesting

$$\{\langle w.\texttt{Name}, w.\texttt{Vineyard}\rangle \mid w \in \texttt{WINES} \wedge$$
$$\exists e \in \texttt{PRODUCER}(w.\texttt{Vineyard} = e.\texttt{Vineyard} \wedge e.\texttt{Region} = \texttt{'Bordeaux'})\}$$

QBE and MS Access

Motivation: The Language QBE

- "Query by Example"
- Queries in QBE: Entries in table frameworks
- Intuition: Example entries in tables
- Precursor of several table-based query interfaces of commercial systems
- Based on logical calculus with domain variables

Queries in QBE: Selection and Projection

- Query: "All rock-albums of the years before 2006"

Album	ANr	Title	Year	Genre	Price	MNr
		P.	<2006	Rock		

$$\{t \mid \texttt{Album}(\_, t, j, \texttt{'Rock'}, \_, \_) \wedge j < 2006\}$$

Queries in QBE: Join

- Query: "All Rock albums of German musicians"

Album	ANr	Title	Year	Genre	Price	MNr
		P.		Rock		_musician

Musician	MNr	Name	Country
	_musician		Germany

$$\{t \mid \texttt{Album}(\_, t, \_, \texttt{'Rock'}, \_, m) \wedge \texttt{Musician}(m, \_, \texttt{'Germany'})\}$$

Queries in QBE: Self-Join

- Query: "Countries with two or more musicians"

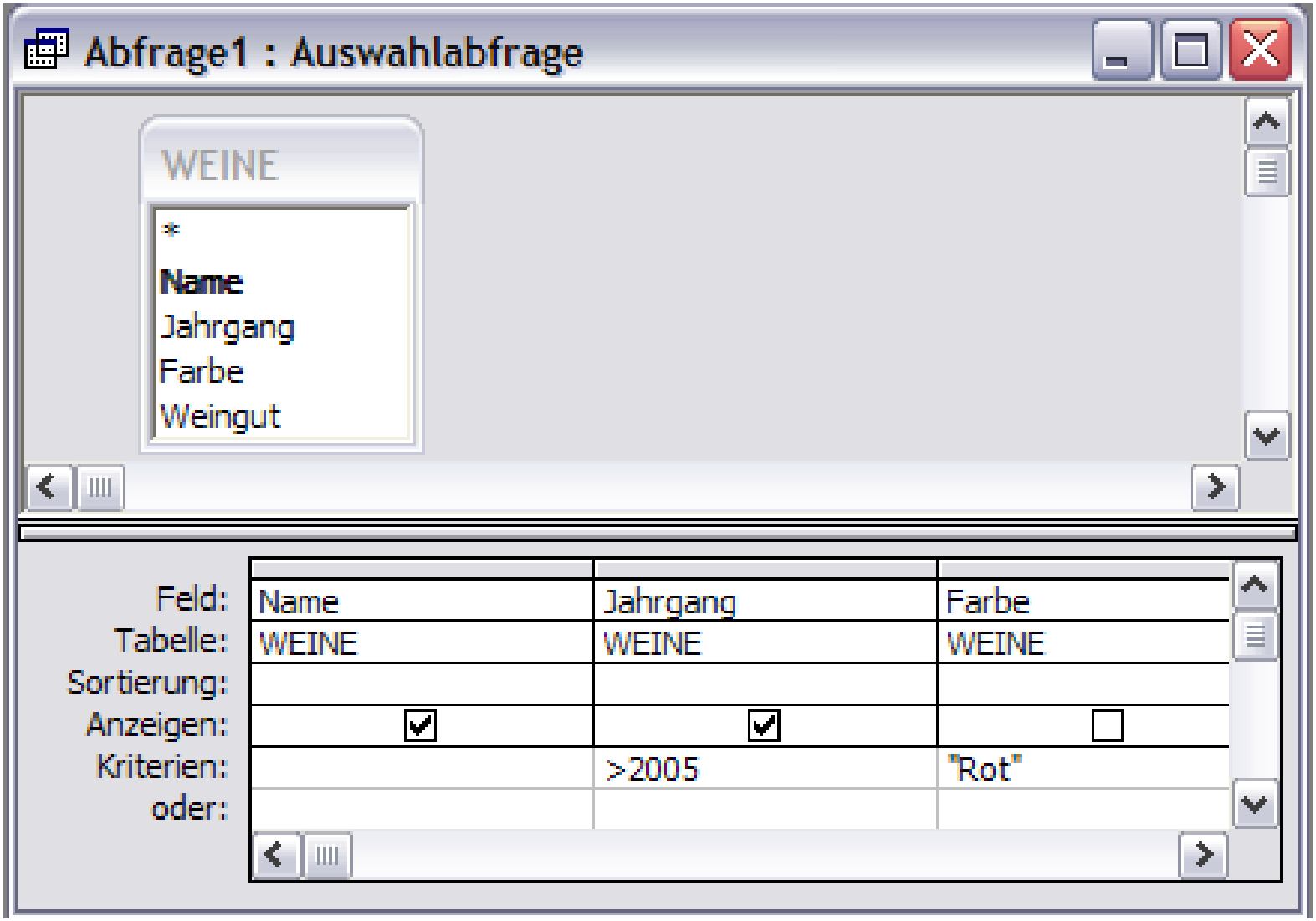
Musician	MNr	Name	Country
	_one ¬ _one		**P.** _country _country

$$\{l \mid \texttt{Musician}(x, \_, l) \wedge \texttt{Musician}(y, \_, l) \wedge x \neq y\}$$

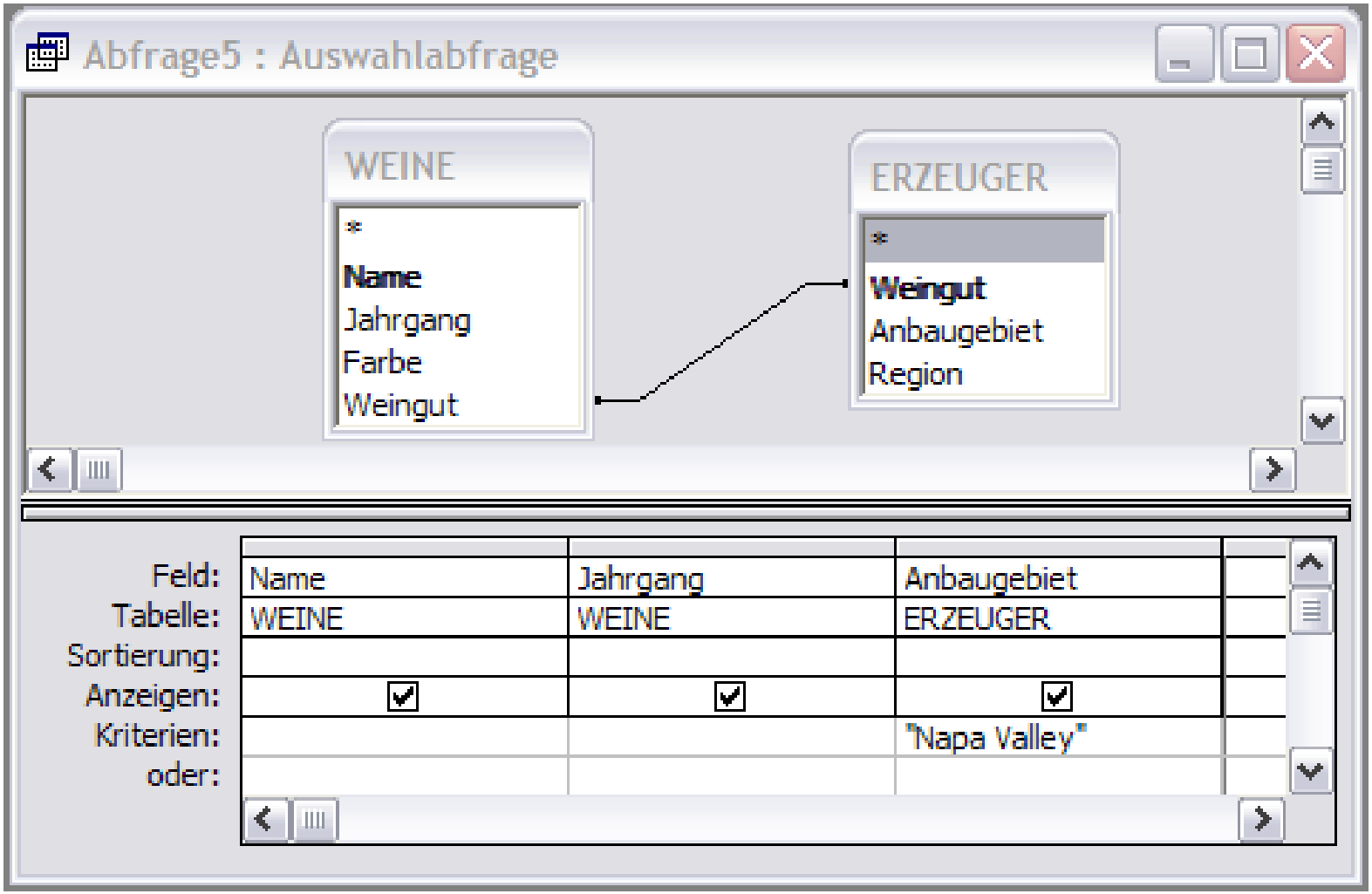
QBE in MS-Access

- MS-Access: Database program for Windows
  - Basis relations with keys
  - Foreign keys with graphical statement of relations
  - Graphical definition of queries (SQL-similar)
  - Interactive definition of forms and reports
- Support of QBE

Access: Projection and Selection



Access: Join


Abfrage5 : Auswahlabfrage
WEINE
*
Name
Jahrgang
Farbe
Weingut
ERZEUGER
*
Weingut
Anbaugebiet
Region
Feld: Name Jahrgang Anbaugebiet
Tabelle: WEINE WEINE ERZEUGER
Sortierung:
Anzeigen:
Kriterien: "Napa Valley"
oder:

Domain Calculus

Domain Calculus

- Terms:
  - Constants, such as 42 or `'MZ-4'`
  - Variables for data types, such as x
    Data type declaration usually takes place implicitly and is not declared explicitly!
  - Function application $f(t_1, \ldots, t_n)$: Function f, terms t_i, such as $plus(12, x)$ resp. in infix notation $12 + x$
- Atomic formulas:
  - Predicate application $\Theta(t_1, \ldots, t_n)$, $\Theta \in \{<, >, \leq, \geq, \neq, =, \ldots\}$ data-type predicate, terms t_i
    Binary predicates are usually in infix-notation.
    Example: $x = y$, $42 > x$ or $3 + 7 = 11$.

Domain Calculus /2

- Atomic formulas (advanced):
  - Predicate application for database predicates noted as $R(t_1, \ldots, t_n)$ for a relation name R
    Providing: n must be the arity of the relation R and all t_i must be of the same type
    Example: $\texttt{PRODUCER}(x, \text{'Hessen'}, z)$
- Formula as usual with $\wedge$, $\vee$, $\neg$, $\forall$ and $\exists$

Domain Calculus /3

- *Query*: $\{x_1, \ldots, x_n \mid \phi(x_1, \ldots, x_n)\}$
  - ϕ is formula over the variables contained in the result list x_1 to x_n
  - Result is set of tuples
  - Tuple construction comes implicitly from the values of the variables in the result list
- Example

$$\{x \mid \texttt{PRODUCER}(x, y, z) \wedge z = \text{'Hessen'}\}$$

Basic Domain Calculus

- Restriction of the domain calculus:
  - **Range of values**: Integer
  - **Data type predicates** are restricted to equivalence and elementary comparison operations as in relational algebra
  - **Function applications** are not allowed; besides domain variables only constants can be used as terms
- used for theoretical investigations, e.g., proves of properties

Safety

- Safe Queries (also **semantic safe queries**):

Queries that give a finite result for each database state $\sigma(\mathcal{R})$

- Example for unsafe queries:

$$\{x, y \mid \neg R(x, y)\}$$

- Example for safe queries:

$$\{x, y \mid R(x, y)\}$$

Safe Queries /2

- Further example for safe queries:

$$\{x, y \mid y = 10 \wedge x > 0 \wedge x < 10\}$$

Safety comes directly from the rules of arithmetics.

Semantic safety is in general not decidable!

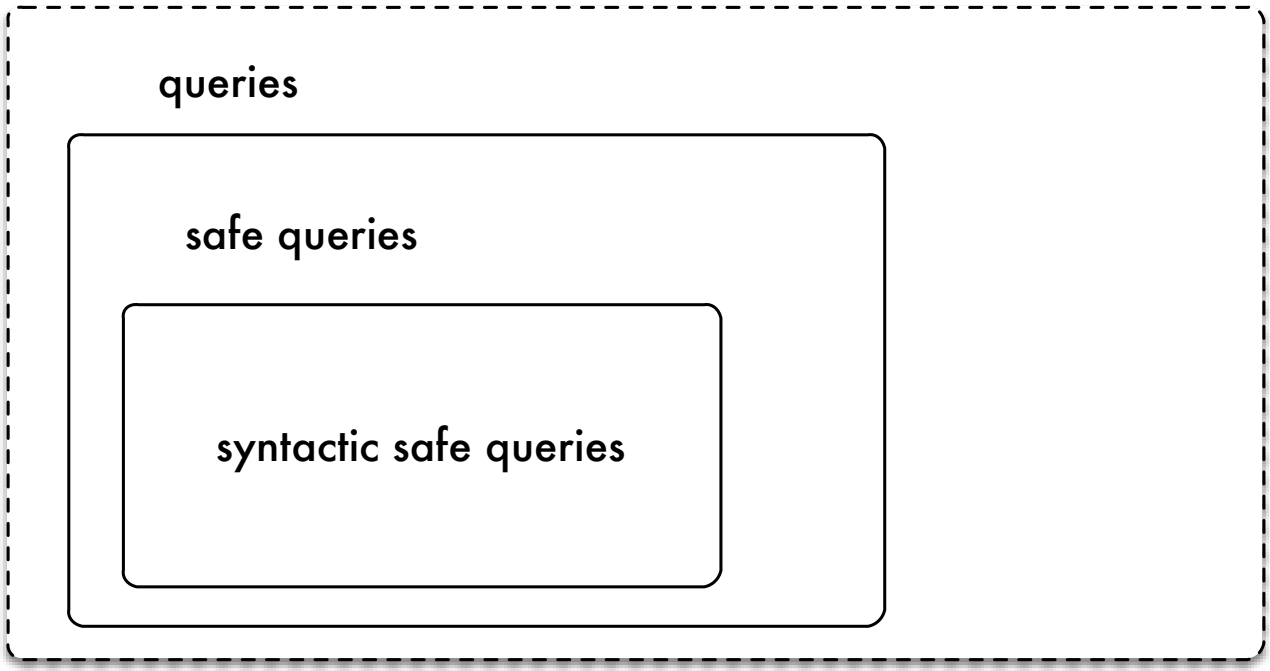
Syntactically Safe Queries

- Syntactically Safe Queries: Queries that are subject to syntactic restrictions to enforce semantic safety.
- Basic idea:

Any free variable x_i must be bound to a finite domain anywhere in $\phi(x_1, \dots)$ by a positive occurrence $x_i = t$ or $R(\dots, x_i, \dots)$.

- Binding to finite domains must hold for the whole condition, in particular for all branches of an disjunction

Safe Queries Overview



Examples for the Domain Calculus

Examples Domain Calculus

- Query: "All vineyards of producers in Hessen."

$$\{x \mid \texttt{PRODUCER}(x, y, z) \wedge z = \text{'Hessen'}\}$$

- Simplified notation: Each unbound variable (here y and z) in the condition part existentially bound with $\exists$
- Complete version:

$$\{x \mid \exists y \exists z \texttt{PRODUCER}(x, y, z) \wedge z = \text{'Hessen'}\}$$

- Saving of domain variables, by using constants as parameters of the predicate:

$$\{x \mid \texttt{PRODUCER}(x, y, \text{'Hessen'})\}$$

Examples Domain Calculus /2

- Abbreviation for arbitrary, different existentially bound variables is the _ symbol:

$$\{x \mid \texttt{PRODUCER}(x, \_, z) \wedge z = \text{'Hessen'}\}$$

- Different appearances of the symbol _ stand for pairwise different variables

Examples Domain Calculus /3

- Query: "Regions with more than two vineyards."

$$\{z \mid \texttt{PRODUCER}(x, y, z) \wedge \texttt{PRODUCER}(x', y', z) \wedge x \neq x'\}$$

- Query shows a join binding on the third attribute of the PRODUCER-relation
- Join binding can easily come from the use of the same domain variables as parameter in different relation predicates

Examples Domain Calculus /4

- Query: "From which region are which wines with the vintage before 1970 in the supply?"'

$$\{y, r \mid \texttt{WINES}(x, y, z, j, w) \wedge \texttt{PRODUCER}(w, a, r) \wedge j < 1970\}$$

- Join over two relations

Examples Domain Calculus /5

- Query: "From which regions are red wines?"'

$$\{z \mid \texttt{PRODUCER}(x, y, z) \wedge \exists a \exists b \exists c \exists d(\texttt{WINE}(a, b, c, d, x) \wedge c = \text{'Red'})\}$$

- Use of a existentially bound subquery
- Such subqueries could be dissolved due to the rules of predicate logic as follows:

$$\{z \mid \texttt{PRODUCER}(x, y, z) \wedge (\texttt{WINE}(a, b, c, d, x) \wedge c = \text{'Red'})\}$$

Examples Domain Calculus /6

- Query: "Which vineyard has only wines of the vintage after 1995 in its supply?"'

$$\{x \mid \texttt{PRODUCER}(x, y, z) \wedge \forall a \forall b \forall c \forall d (\texttt{WINE}(a, b, c, d, x) \implies d > 1995)\}$$

- Universally bound subformulas cannot be dissolved

Expressiveness of the Domain Calculus

Domain calculus is strict relational complete, i.e. to any term τ of the relational algebra there is an equivalent (safe) term η of the domain calculus.

Implementation of Relational Operations

Given: Relation schema $R(A_1, \ldots, A_n)$ and $S(B_1, \ldots, B_m)$

- Union (for $n = m$)

$$R \cup S \hat{=} \{x_1 \ldots x_n \mid R(x_1, \ldots, x_n) \vee S(x_1, \ldots, x_n)\}$$

- Difference (for $n = m$)

$$R - S \hat{=} \{x_1 \ldots x_n \mid R(x_1, \ldots, x_n) \wedge \neg S(x_1, \ldots, x_n)\}$$

- Natural Join

$$R \bowtie S \hat{=} \quad \{x_1 \ldots x_n x_{n+1} \ldots x_{n+m-i} \mid R(x_1, \ldots, x_n) \wedge S(x_1, \ldots, x_i, x_{n+1}, \ldots, x_{n+m-i})\}$$

Assumption: the first i attributes of R and S are the join attributes, thus $A_j = B_j$ for $j = 1 \ldots i$

Implementation of Relational Operations /2

- Projection

$$\pi_{\overline{A}}(R) \hat{=} \{y_1 \ldots y_k \mid \exists x_1 \ldots \exists x_n (R(x_1, \ldots, x_n) \wedge y_1 = x_{i_1} \wedge \cdots \wedge y_k = x_{i_k})\}$$

  Attribute list of the projection: $\overline{A} = (A_{i_1}, \ldots, A_{i_k})$

- Selection

$$\sigma_{\phi}(R) \hat{=} \{x_1 \ldots x_n \mid R(x_1, \ldots, x_n) \wedge \phi'\}$$

  ϕ' is derived from ϕ, by insertion of the variable x_i at the place of the attribute names A_i

Summary

Summary

- Formal models for queries in database systems
- Relational algebra
  - Operational approach
  - Queries as nesting of operations on relations
- Query calculus
  - Logic-based approach
  - Queries as derived predicates
  - *see Book: sections 4.2.3, 4.2.4 and 9.3*

Control Questions

- Which meaning do equivalence, independence and completeness have in the relational algebra?

Control Questions

- Which meaning do equivalence, independence and completeness have in the relational algebra?
- How can the semantics of the extended SQL operations be expressed in relational algebra?

Control Questions

- Which meaning do equivalence, independence and completeness have in the relational algebra?
- How can the semantics of the extended SQL operations be expressed in relational algebra?
- What is the difference between relational algebra and relational query calculus?

Control Questions

- Which meaning do equivalence, independence and completeness have in the relational algebra?
- How can the semantics of the extended SQL operations be expressed in relational algebra?
- What is the difference between relational algebra and relational query calculus?
- What is the role of safety in queries?