Grundvorlesung Informationssysteme

SS 10

Joachim Biskup
ISSI - Informationssysteme und Sicherheit
Technische Universität Dortmund

Inhaltsverzeichnis

Teil II:
Relationale Systeme mit strukturierten Daten 95

Teil I

Grundlagen

1 Modelle und Architekturen

1.1 Überblick: Thema der Vorlesung

Gliederung

- Grundlagen
- relationale Systeme mit strukturierten Daten
- erweiterte und alternative Systeme
- Effizienz
- Mehrbenutzerbetrieb, Transaktionen und Sicherheit

Literatur-Empfehlungen

Joachim Biskup: Grundlagen von Informationssystemen,
Vieweg, Braunschweig/Wiesbaden, 1995.
543 Seiten.

Joachim Biskup: Grundvorlesung Informationssysteme
(korrigierte Fassung der vollständigen Vorlesungsfolien vom 8.4.2005),
http://ls6-www.cs.uni-dortmund.de/issi/teaching/unterlagen/index.html#ISDPO2001.
519 Seiten.

Ramez Elmasri, Shamkat B. Navathe: Fundamentals of Database Systems (3rd edition),
Addison Wesley, Reading etc, 2000.
955 + 30 + 24 Seiten.

Jeffrey D. Ullman, Jennifer Widom: A First Course in Database Systems,
Prentice Hall, Upper Saddle River, 1997.
470 Seiten.

Georg Lausen: Datenbanken - Grundlagen und XML-Technologien,
Elsevier-Spektrum, 2005.
281 Seiten.

Aufgaben eines Informationssystems

- große Mengen von (large amount)
- strukturierten oder halb-strukturierten Daten ((semi-)structured data)
- dauerhaft (persistent)
- verlässlich (dependable)
- für im Allgemeinen viele
  und verschiedenartige Benutzer verfügbar (shared)
- effizient verwalten (management)
  d.h. Anfragen und Änderungen bearbeiten (queries/updates)

Grundformen

- Datenbanksystem (database system)
- Wissensbanksystem (knowledgebase system)
- “Information Retrieval-System” (information retrieval system)
- “Multimedia-System” (multi-media system)

für viele und verschiedenartige Benutzer

- Verlässlichkeit
- konzeptionelle Modellierung (ER, UML) und Formalisisierung mit *Datendefinitionssprache*
- (konzeptionelles) *Schema* und (externe) *Sichten*
- *Ontologie*

verschiedenartige Blickwinkel auf ein Informationssystem

- Modell einer “Miniwelt” bilden
- eigenständige Miniwelt verwalten
- Mitteilungen vermitteln

weitreichendster Blickwinkel

Ein Informationssystem vermittelt Mitteilungen
zwischen “in einem Unternehmen” handelnden Menschen.

Diese Menschen handeln in einer Miniwelt,

deren Gegebenheiten und Vorgänge
durch Dokumente abgebildet werden,

über die sich die Menschen
mit Hilfe des Informationssystems
Mitteilungen machen.

1.2 Information und Kommunikation: Vermittlung durch Informationssysteme

Information

- Es wird ein *Rahmen von Möglichkeiten* abgesteckt,
  wobei Unsicherheit über die tatsächlich vorliegenden Verhältnisse besteht.

- Es werden aufeinanderfolgende mögliche Ereignisse
  zueinander in Beziehung gesetzt unter dem Gesichtspunkt,

  wie weit die *Unsicherheit*
  über die tatsächlich vorliegenden Verhältnisse
  verringert wurde.

eine wahrscheinlichkeitstheoretische Sicht von Information

- "Rahmen der Möglichkeiten" durch *Zufallsvariable* beschrieben:

  a_i mögliche Werte der Zufallsvariablen

  $p(a_i)$ Wahrscheinlichkeiten ihres Eintreffens

- *Informationsgehalt* eines Wertes a_i :

  die durch das tatsächliche Eintreffen von a_i beseitigte *Unsicherheit*

- "Nichtereignis" (*vorher*) in Beziehung gesetzt zum Eintreffen (*nachher*)

eine wahrscheinlichkeitstheoretische Sicht von Information: Formalisierung

- *Informationsgehalt* eines Wertes a_i :

  $I(a_i) = -\operatorname{ld} p(a_i) = \operatorname{ld} 1/p(a_i)$ für $p(a_i) \neq 0$

- Randfälle:

  $p(a_i) = 1$: $I(a_i) = \operatorname{ld} 1 = 0$

  $p(a_1) = p(a_2) = 1/2$: $I(a_j) = \operatorname{ld} 2 = 1$

- Entropie, mittlerer *Informationsgehalt* der möglichen Werte:

$$H(a) = \sum_{i=1}^{k} p(a_i) \cdot I(a_i)$$

eine modelltheoretische Sicht von Information

- "Rahmen der Möglichkeiten" durch Klasse κ von Strukturen beschrieben:

  $M = (d, r_1, ..., r_n)$ wobei jeweils d nichtleeres *Universum* (*domain*)

  r_i *Relationen* über d, d.h. $r_i \subset d \times ... \times d$

- "Ereignisse" sind in einer geeigneten Logiksprache formalisierte Aussagen

- *Unsicherheit* bei Aussagenmenge Φ:

  welches der Modelle für Φ aus Klasse κ liegt vor?

- Randfälle und Beispiele:

  genau ein Modell: *keine* Unsicherheit
  alle Modelle: *höchste* Unsicherheit

Kommunikation

- "Information" und "Kommunikation" aufeinander bezogen:

  bei "Information": auch "Informations*übertragung*" betrachten
  bei "Kommunikation": auch übermittelten "*Information*sgehalt" betrachten

- Beteiligte müssen über einen gemeinsamen *Bezugsrahmen* verfügen

- Unterstützung von Kommunikation durch "Informationssysteme":

  ausdrücklich vereinbarter Bezugsrahmen
  durch "***Schema***" und zusätzlicher "***Ontologie***"

Kommunikation als eine Handlung

- ***Kommunikation***:

  Handlung zwischen vernunft- und sprachbegabten Menschen

- ***Sprache***:

  nicht alleiniges, aber herausragendes Mittel der menschlichen Kommunikation

- ***natürliche Sprache***:
  - unermesslicher gemeinsamer Erfahrungsschatz als vorgefundener Bezugsrahmen
  - zur Verabredung von Bezugsrahmen,
  - zur Entfaltung neuer Bezüge
  - zur Definition formaler Sprachen

- ***formale Sprachen*** zur vermittelten Kommunikation:
  - ***Datendefinitionssprache***
  - ***Datenmanipulationssprache*** mit ***Anfragesprache***

- Verdichtung, Verengung oder auch Verarmung der Ausdrucksmöglichkeiten

Grundzüge menschlichen Handelns

Handlungszug	Funktion	Prägung durch	Weltauffassung	Geltungs-anspruch
teleologisch (strategisch)	Zweck verwirklichen	Sachverhalte	objektiv	Wahrheit und Wirksamkeit
normenreguliert	Beziehungen pflegen	Erwartungen anderer	sozial	Berechtigung
dramaturgisch	sich selbst darstellen	eigenes Erleben	subjektiv	Wahrhaftigkeit

kommunikatives Handeln (nach Habermas)

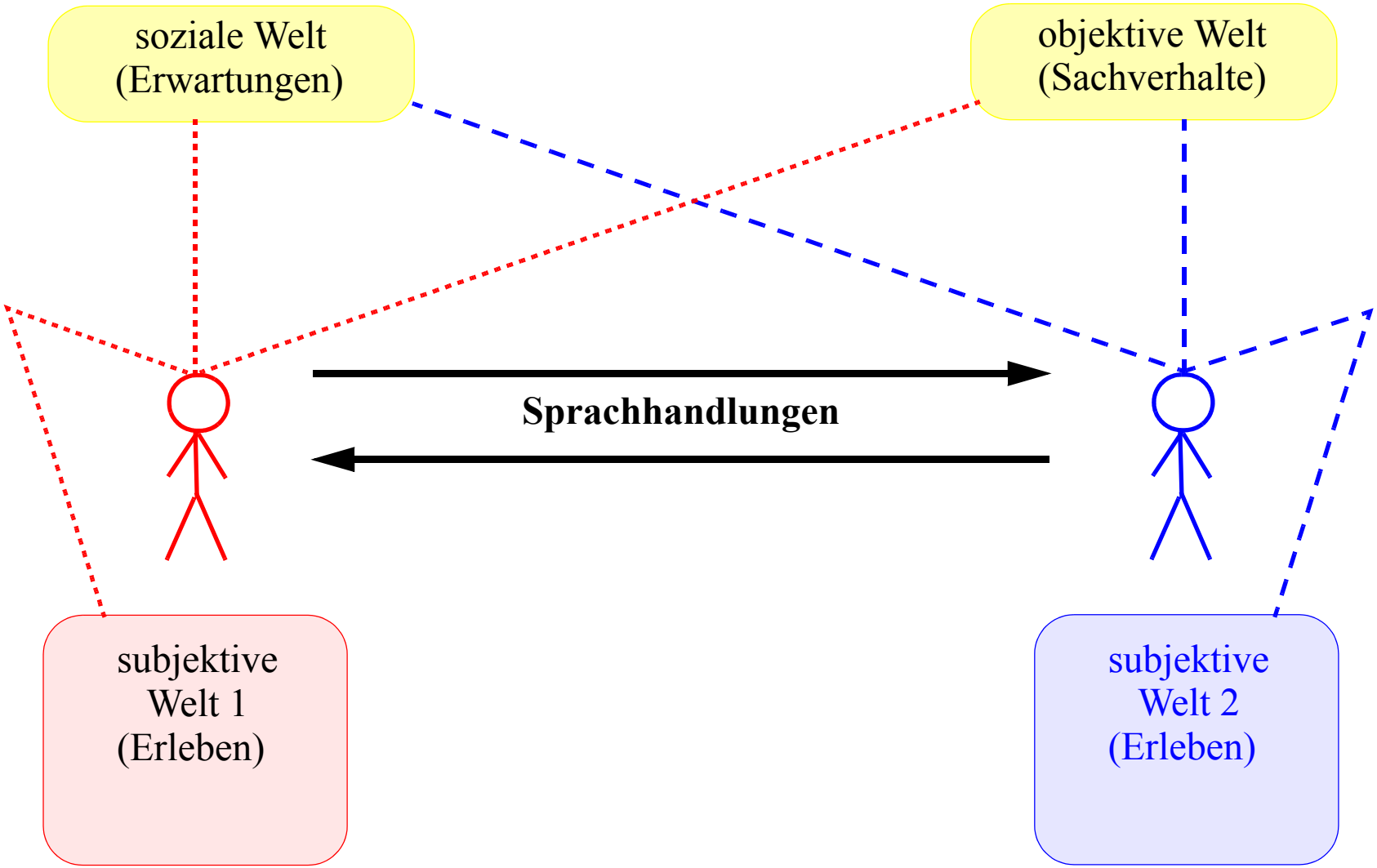
- im *kommunikativen Handeln* verbinden sich die genannten Züge
- zwei oder mehrere Handelnde

  “suchen eine Verständigung über ihre Handlungssituation,
  um ihre Handlungspläne und damit ihre Handlungen einvernehmlich zu koordinieren”
  (J. Habermas)
- wichtigstes Hilfsmittel dabei ist die Sprache:

  Handelnden versuchen, ihre Einstellungen zur “Welt” gegenseitig darzustellen,

  - zu den (objektiven) Sachverhalten,
  - zu den (sozialen) Erwartungen anderer,
  - zu dem (subjektiven) eigenen Erleben
- auf der Grundlage eines gemeinsamen erfahrenen
  Ausschnittes der “Lebenswelt”

Veranschaulichung kommunikativ Handelnder mit Weltbezügen



Gestaltung von Informationssystemen

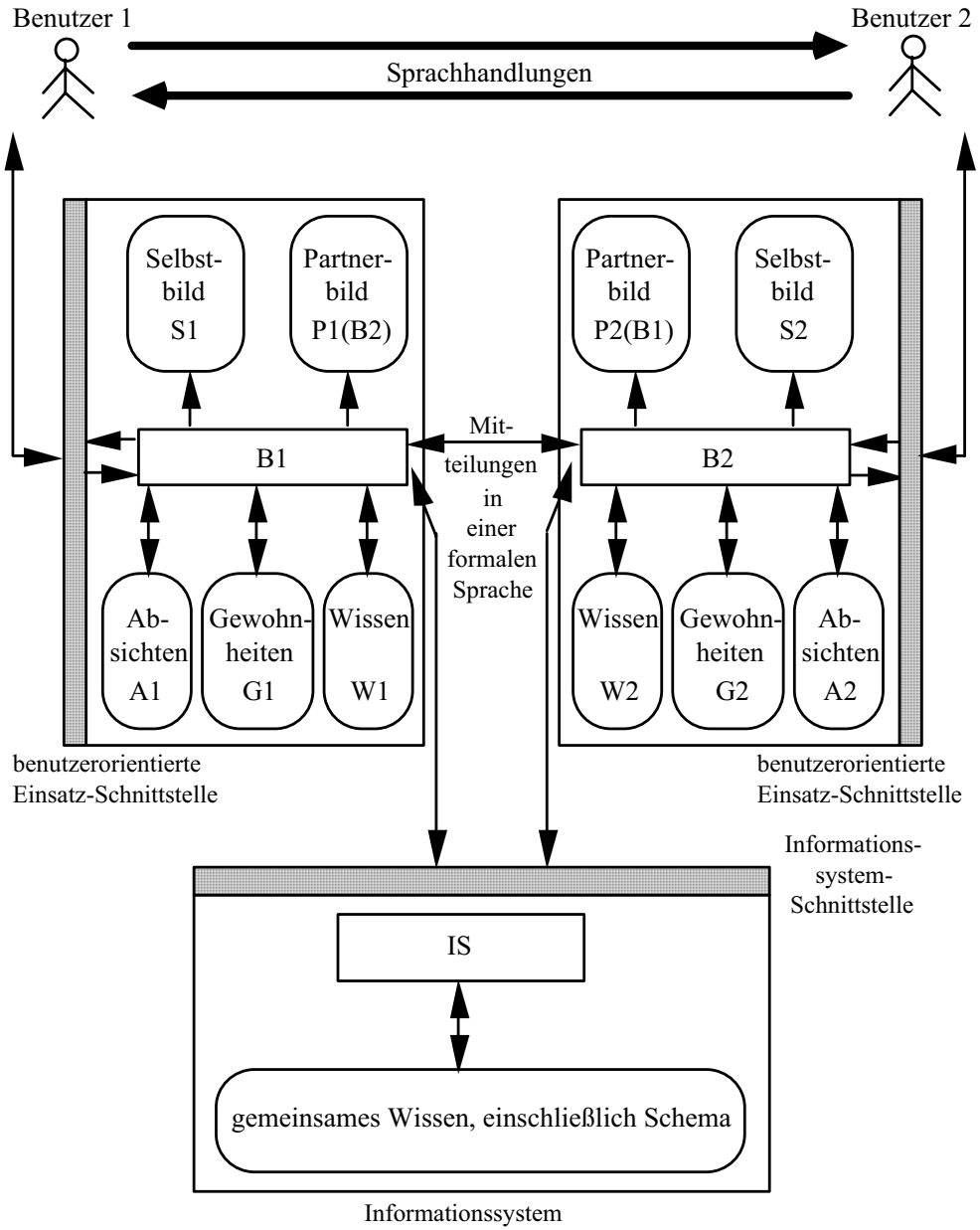
- zwei Grundeinsichten:
  - menschliche Kommunikation auffassen als hochkomplexe Handlung, die nur näherungs- und modellweise verstanden (und auch nur verstehbar) ist
  - jede Vermittlung durch "Informationssysteme" kann dazu führen, dass möglicherweise wichtige Gesichtspunkte unterdrückt werden

- Leitfaden:

  wenigstens *ansatzweise* wichtige Bestandteile von Kommunikation unter *Benutzung formaler Sprachen* nachbilden

- Unterschiede überbrücken

Vermittlung von Kommunikation durch ein Informationssystem



Kommunikation und Mensch-Rechner-Interaktion

- ***Kommunikation***:

  Handlung zwischen vernunft- und sprachbegabten Menschen

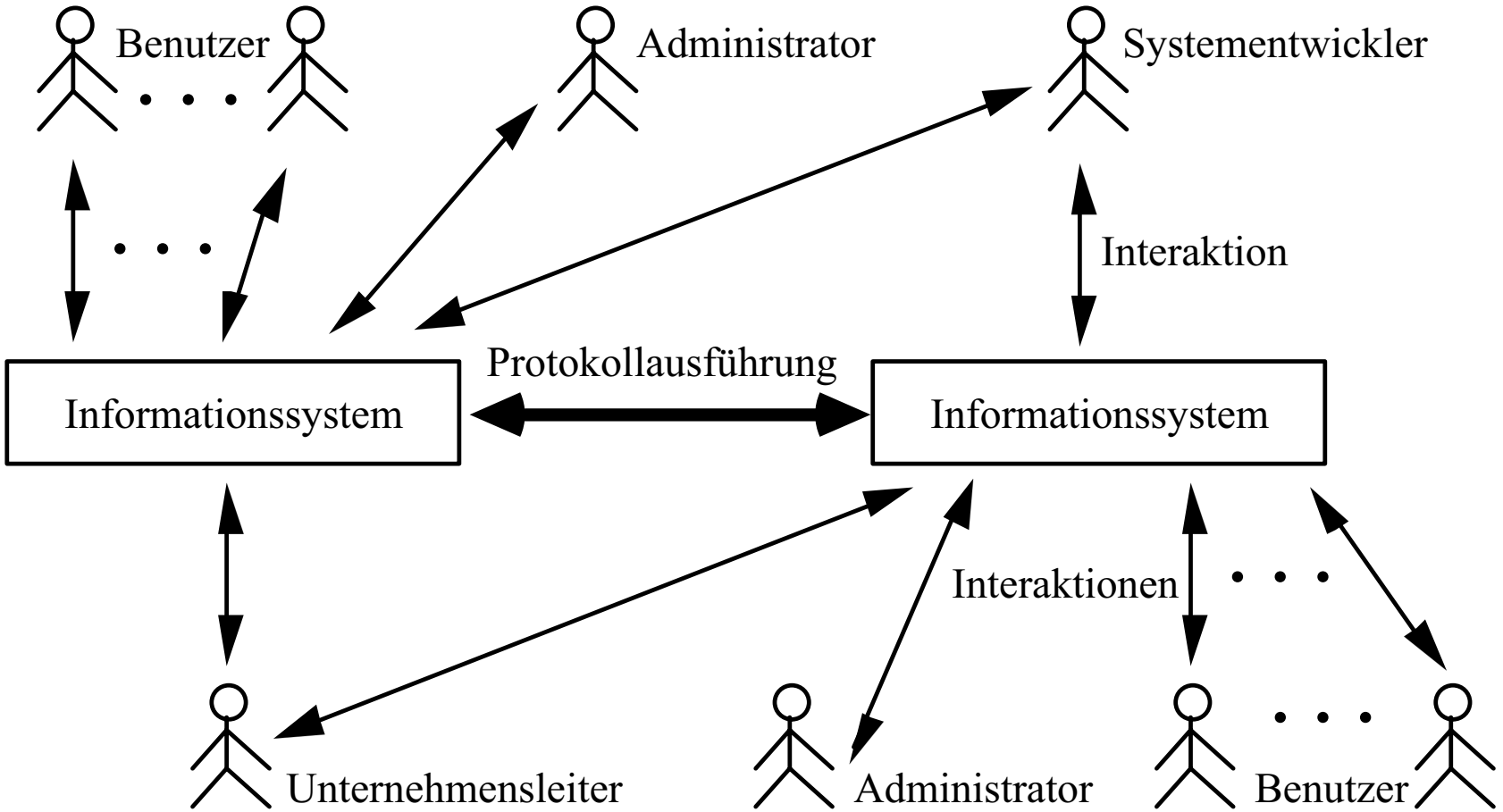
- ***Interaktion***:

  Wechselbeziehung zwischen
  - einem handelnden Menschen
  - einem Rechensystem

- ***Protokollausführung***:

  Wechselbeziehung zwischen verschiedenen Rechensystemen
  (oder hinreichend selbständigen Teilen eines Rechensystems)

Interaktion und Protokollausführung

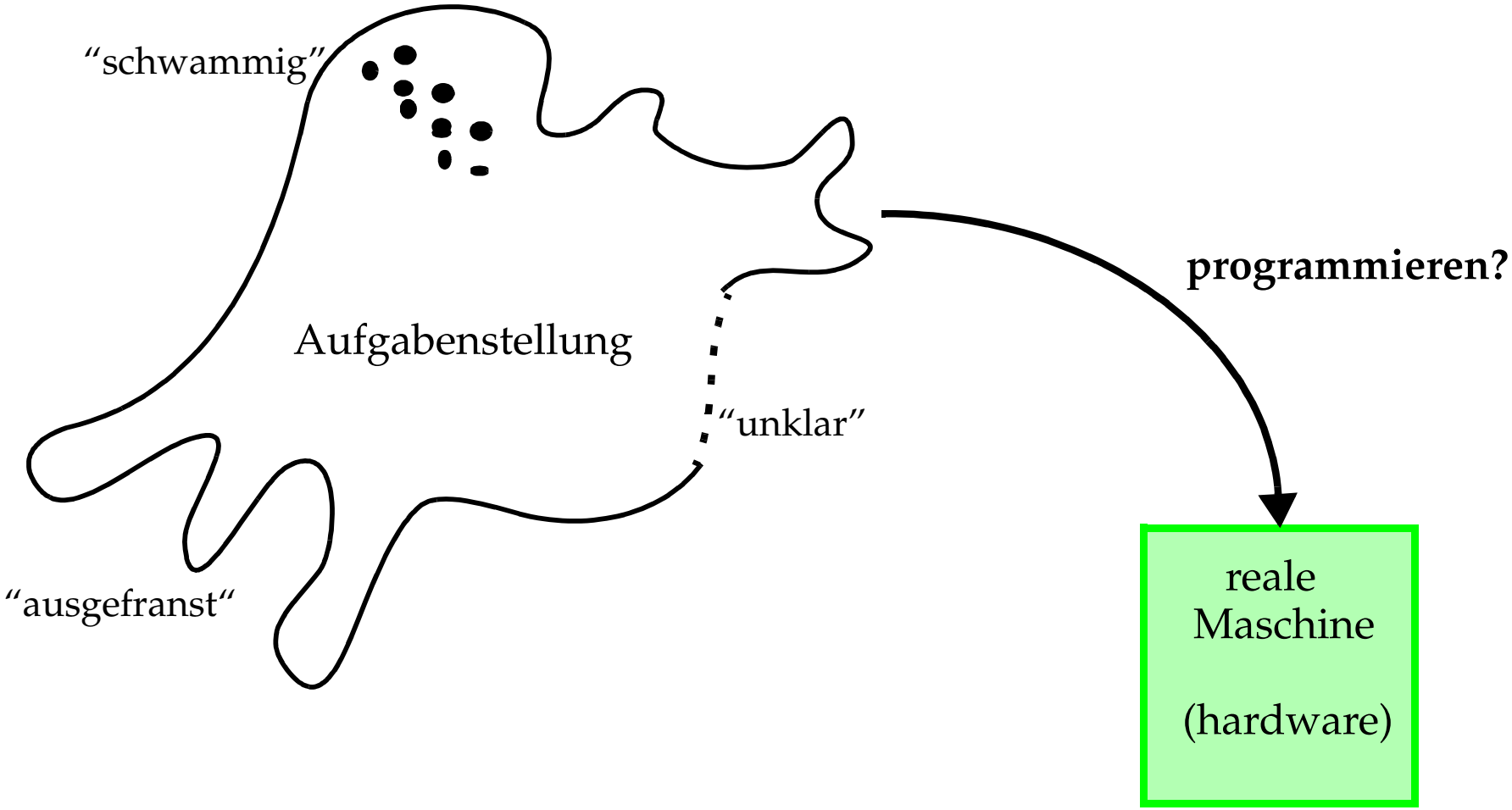


Vermittlung durch ein Informationssystem

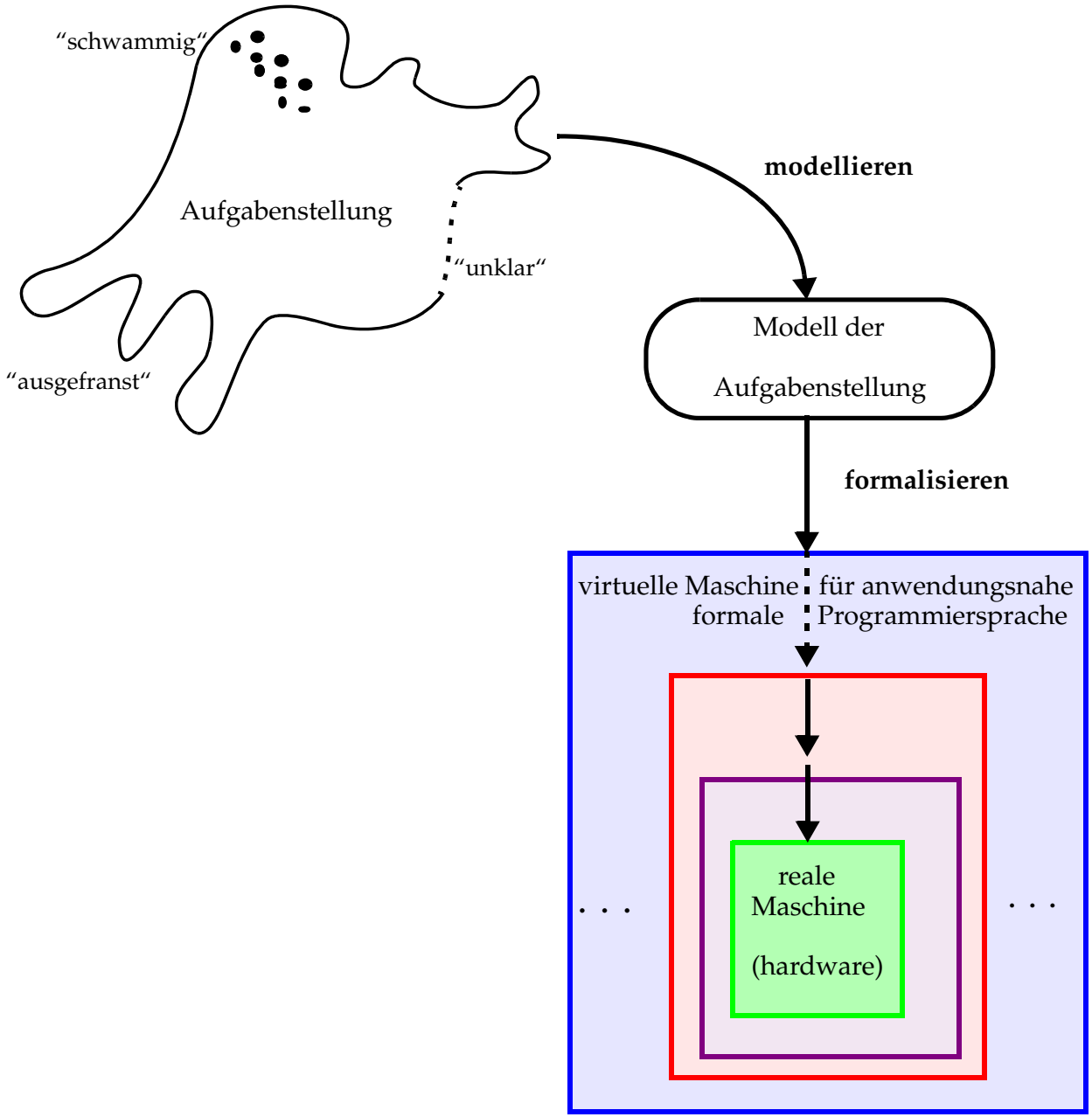
- Es stehen jeweils *große Mengen* von Daten vermittlungsbereit zur Verfügung.
- Die Vermittlung erfolgt im allgemeinen *zeitlich verzögert*.
- Die *Qualität* der Vermittlung wird durch besondere Protokolle abgesichert, die der unmittelbaren Kontrolle der (End-) Benutzer entzogen werden.
- Die Vermittlung wird im Allgemeinen
  vielen und verschiedenartigen Handelnden angeboten.
- Wird die Vermittlung tatsächlich in Anspruch genommen,
  so geschieht sie unter *einengenden zeitlichen und räumlichen* Anforderungen.

1.3 Modellierung: Wirklichkeit und Modell

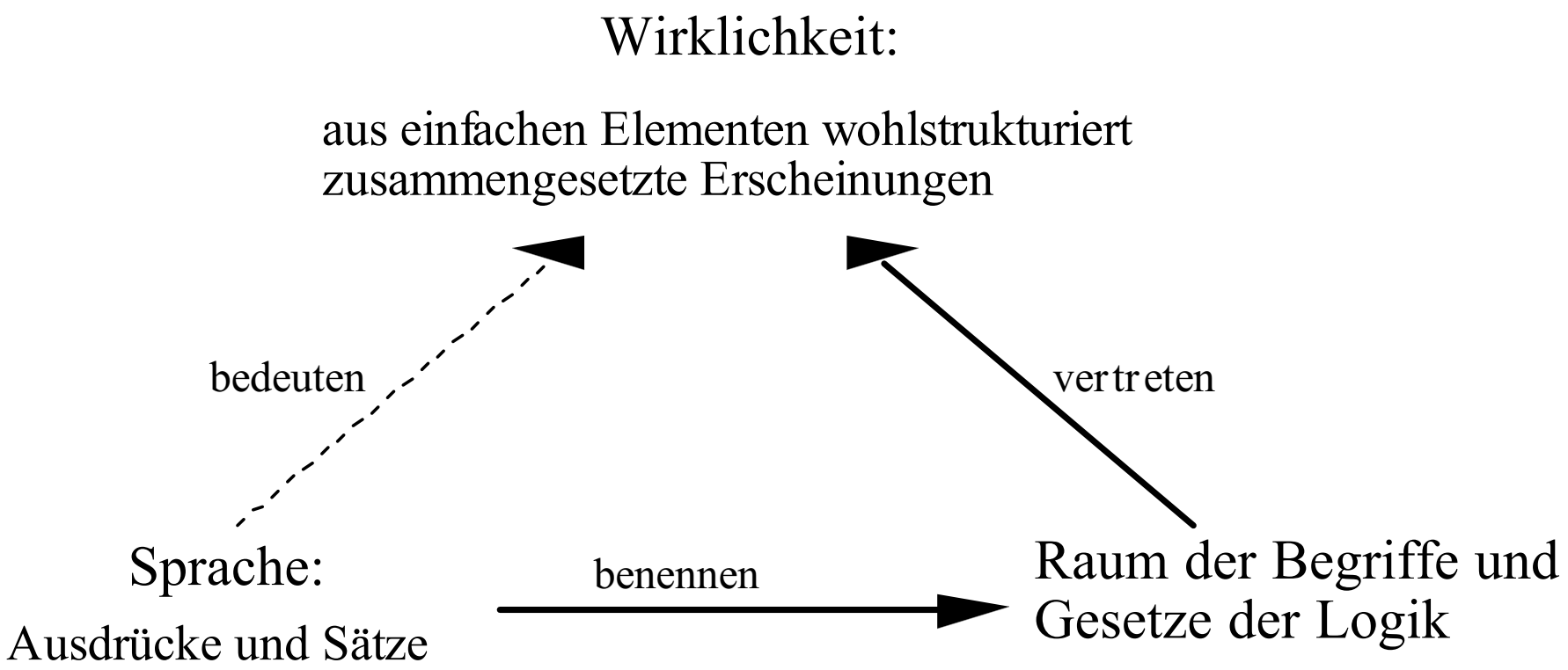
Programmieren?



Programmieren: Modellieren und Formalisieren



Wirklichkeit - Begriffsraum - Sprache



Begriffsgerüst der Entity-Relationship Modellierung

- **Seiendes**, **Entität** (entity): Einheit der "Mini-Welt"
- **Beziehung** (relationship): ein Zusammenhang zwischen Seienden
- **Eigenschaft** / **Attribut** (predicate / attribute):
  beschreibt Seiendes oder Beziehung
- **Rolle** (role): Bedeutung eines Seienden in einer Beziehung
- **Klassenbildung** (abstraction):Zusammenfassung
  gleichartiger Seienden/Beziehungen
- **Aussonderung** (separation/ specialization):
  für eine Aussageform und eine Klasse
  diejenigen Seienden/Beziehungen,
  für die die entsprechende Aussage gilt
- **Verallgemeinerung** (generalization): Vereinigung wohlbestimmter Klassen
- **Aggregation** (aggregation): Beziehungen als Seiende betrachtet

statische Gesichtspunkte eines Unternehmens

- ***Aufzählungen***
  - der vorhandenen Seienden,
  - ihrer zutreffenden Eigenschaften und
  - ihrer Beziehungen

- ***Bedingungen*** (constraints):
  schränken die Aufzählungen ein

- ***Regeln*** (rules):
  erschließen Seiendes, Beziehungen etc.

typische Bedingungen

- ***Schlüsselbedingung*** (key constraint)
- ***Aussonderungsbedingung*** (specialization constraint, isa-constraint)
- ***Verallgemeinerungsbedingung*** oder
  Partitionsbedingung (partition constraint)
- ***viele-eins-Bedingung*** (many-one-constraint, *n*:1-constraint)
- ***Seinsbedingung*** (existence constraint)
- ***Verweisbedingung*** (referential constraint)

typische Regeln

- ***Gesamtheitsregel*** (universe of discourse rule):

  Die Gesamtheit der möglichen Seienden einer Art wird
  abschließend durch die Angaben ... < die hier genauer einzufügen wären >... beschrieben.

- ***Verneinungsregel*** (negation rule):

  Ist eine Beziehung in der Gesamtheit der möglichen Beziehungen,
  aber weder in der Aufzählung noch erschließbar,
  so soll die entsprechende Aussage *nicht* gelten.

- ***Sichtregel*** (view rule):

  Sind die Beziehungen R_1,...,R_k in der Aufzählung oder schon erschlossen,
  so soll auch die Beziehung erschlossen werden können,
  die man aus R_1,...,R_k
  gemäß den Angaben ...< die hier genauer einzufügen wären >... erstellen kann.

dynamische Gesichtspunkte eines Unternehmens

- ***Bedingungen*** beschreiben auch ***dynamische Gesichtspunkte***:
  - eine Mitteilung fordert den Empfänger auf, sein Wissen zu verändern
  - dabei sind die vereinbarten (statischen) Bedingungen aufrecht zu erhalten
- direkte ***dynamische Gesichtspunkte***:
  - eine Änderung im Wissen eines Senders, d.h. eine *Information,*
  - löst eine *Mitteilung* an einen (oder mehrere) Empfänger aus,
  - die dann ihrerseits ihr Wissen geeignet verändern oder
    eine anschließende Handlung auslösen,
    d.h. zu *verstehen* versuchen.
- offen für Annahme oder Ablehnung und für Anschlusskommunikation:
  auch (formale) ***Handlungsfolgen*** beschreiben
- Wissen *und* Handlung, statische *und* dynamische Gesichtspunkte:
  stets unmittelbar aufeinander bezogen,
  nur für Formalisierung begrifflich getrennt

zeitabhängige und zeitunabhängige Teile

- im Allgemeinen als ***zeitabhängig*** angesehen,
  häufige Veränderungen:

  - aufzählend dargestelltes Wissen

- im Allgemeinen als ***zeitunabhängig*** angesehen,
  keine oder seltene Änderungen:

  - Formate für die Aufzählungen
  - Bedingungen
  - Regeln für die Gesamtheit der möglichen Seienden und für negative Information
  - formale Handlungen

Schema und Instanz, “Ontologie”

- ***Schema*** (Deklaration): Zusammenfassung der zeitunabhängigen Teile
  - Formate (Typen) für Aufzählungen
  - Bedingungen (Invarianten) für Aufzählungen
  - Regeln (vordefinierte Prozeduren) zum Erschließen von Nicht-Aufgezähltem

- ***Instanz*** (Zustand): Zusammenfassung der zeitabhängigen Teile

  Aufzählung entsprechend Formaten, erfüllt Invarianten

- ***Gesamtbeschreibung***:

  Schema als *Selbstbeschreibung* ansehen

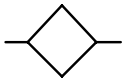
- “***Ontologie***”:

  meistens nur umgangssprachliche oder halb-formale inhaltliche Deutungen von Schema und Instanzen

ER-Diagramme

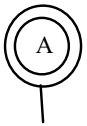
Zeichen für Klassen

Seiendenklasse

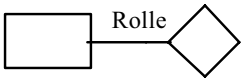


Beziehungenklasse

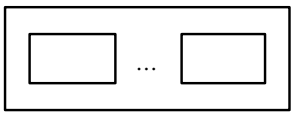
Attribut



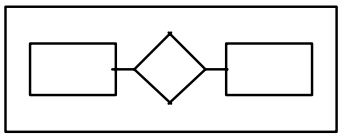
mengenwertiges Attribut



Rolle

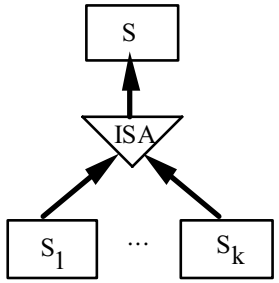


Aggregation



Aggregation durch Beziehungenklasse, aufgefaßt als Seiendenklasse

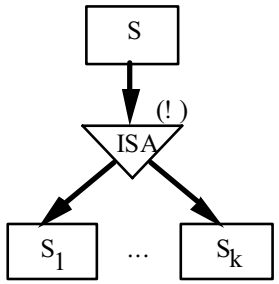
Zeichen für Bedingungen



(Verallgemeinerung)

Aussonderungsbedingung: jedes Seiende aus S_i muß auch in S sein

(Aussonderung)



(Verallgemeinerung)

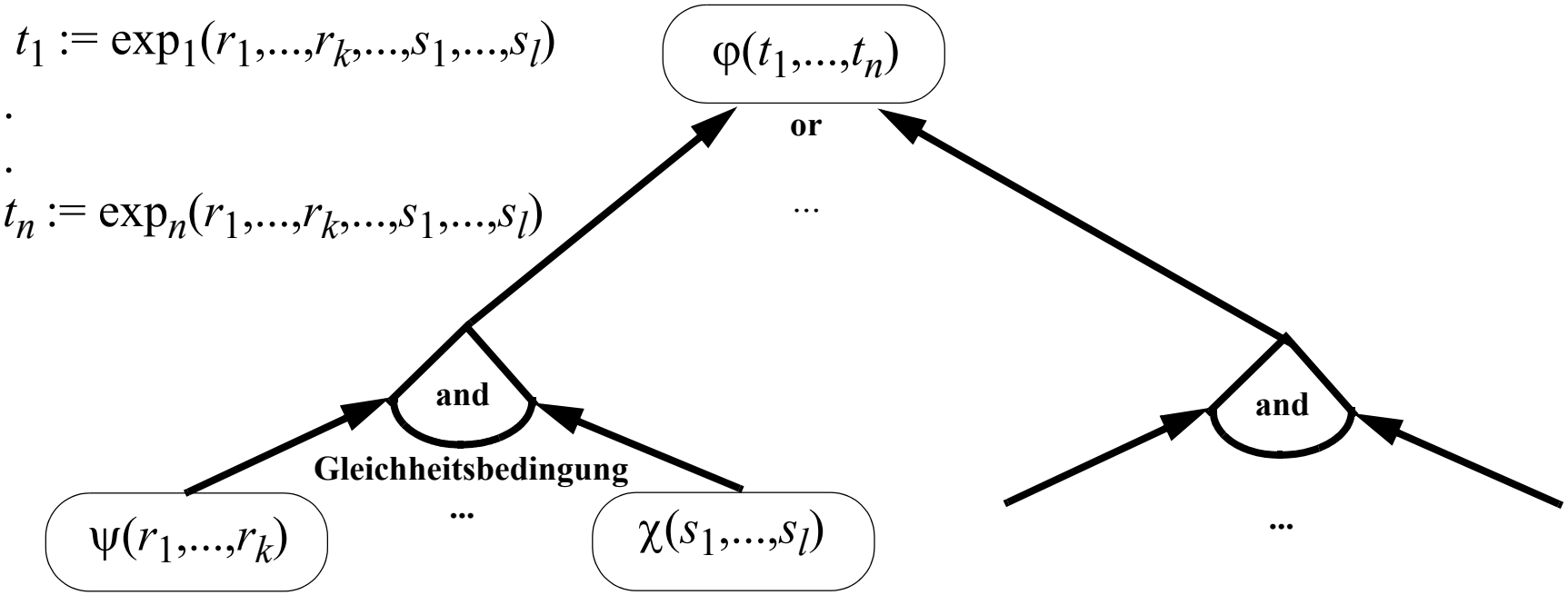
Verallgemeinerungsbedingung (Partitionsbedingung): jedes Seiende aus S muß auch in (genau) einem S_i sein

(Aussonderung)

Regeln und Regelgraphen

“**Wenn** die Aussagen $\psi(r_1,...,r_k)$,..., $\chi(s_1,...,s_l)$ und die Gleichheitsbedingung
alle zutreffen
oder
eine alternative Menge
von Aussagen mit entsprechender Gleichheitsbedingung zutrifft,

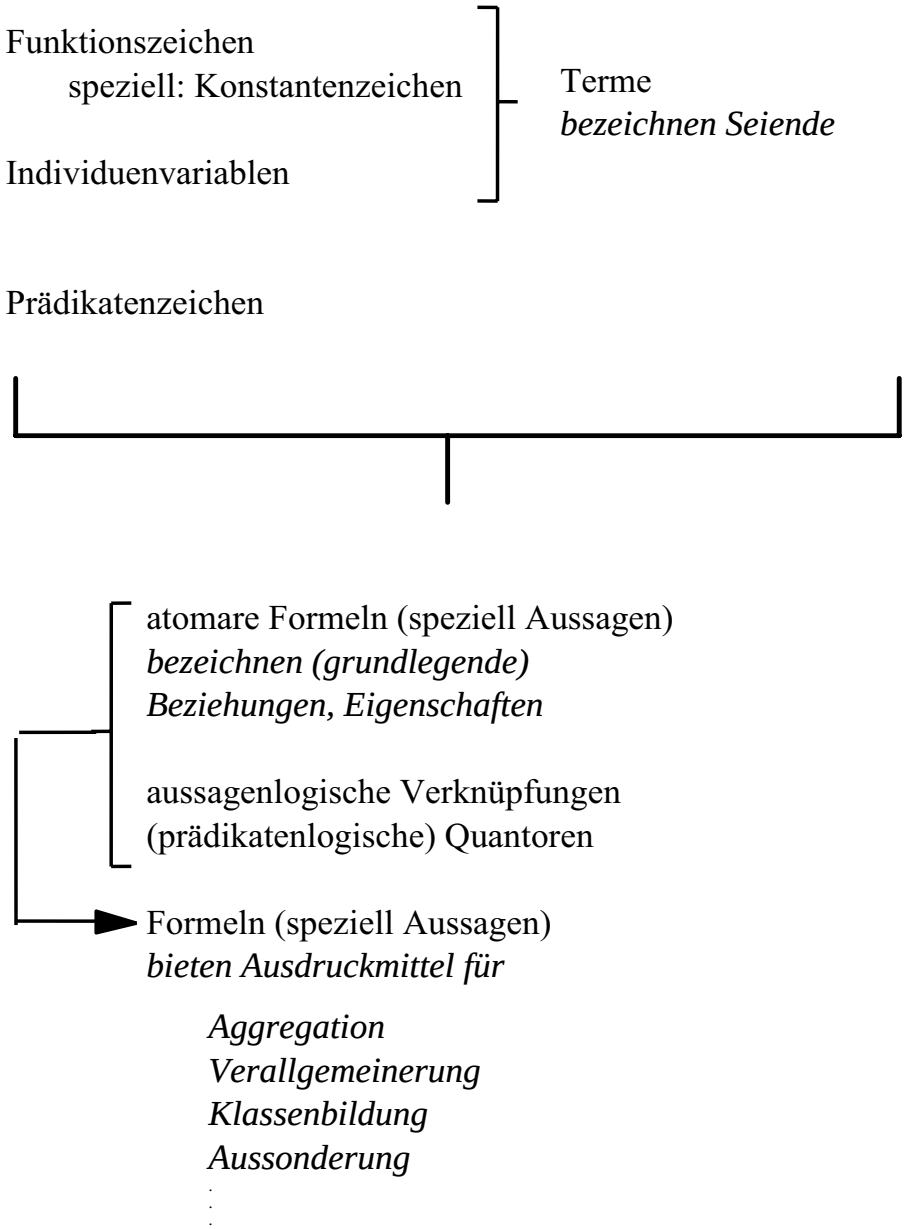
dann trifft auch die Aussage $\varphi(t_1,...,t_n)$ zu,
wobei die Terme t_i wie angegeben gebildet werden.”



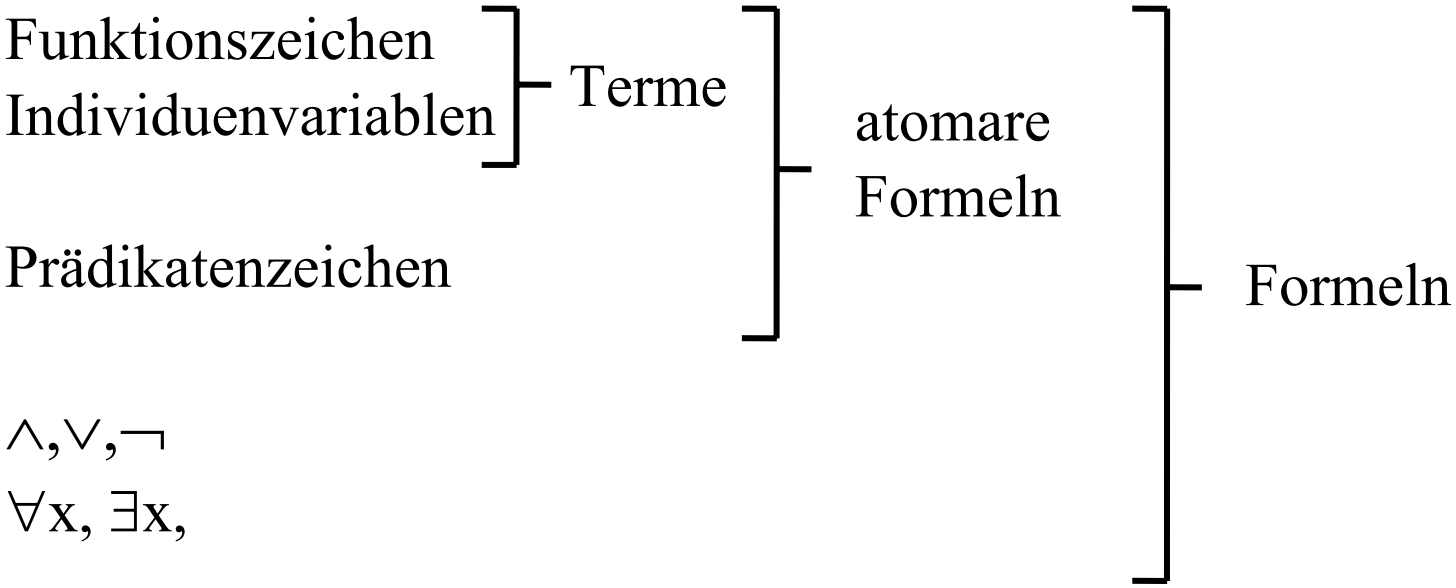
Prädikatenlogik

- **Funktionszeichen**: **F** *einfache Seiende*
- **Individuenvariablen**: **V** *unbestimmte (“beliebige”) Seiende*
- **Terme**: induktiv aus **F** und **V** gebildet,
  zusammengesetzte Seiende, Gegebenheiten
- **Grundterme**: Terme ohne Individuenvariablen
- **Prädikatenzeichen**: **P** mögliche *Beziehungen* und
  Eigenschaften (*Attribute*)
- **Gleichheitszeichen:** = *Identität* zwischen Seienden
- **Formeln (1. Stufe):** induktiv aus Prädikatenzeichen und Termen
  mit aussagenlogischen Junktoren und
  Quantoren für Individuen gebildet,
  Aussagen und *Aussageformen* über
  Seiende, Beziehungen und Eigenschaften

Prädikatenlogik und ER-Modellierung



Syntax


Funktionszeichen
Individuenvariablen
Terme
Prädikatenzeichen
atomare
Formeln
Formeln
∧,∨,¬
∀x, ∃x,

Semantik: Strukturen und Variablenbelegungen

Struktur (Interpretation): $M = (d,\delta)$ mit

- d nichtleere (Werte-) Menge (*Universum*, *domain*)
- δ Zuordnung

  von Funktionszeichen zu Funktionen auf d, $\delta(f^n) : \times_{i=1,...,n} d \to d$

  und von Prädikatenzeichen zu Relationen auf d: $\delta(P^n) \subset \times_{i=1,...,n} d$

  (Gleichheitszeichen = stets durch $\{(x,x) \mid x \in d\}$ interpretiert)

Variablenbelegung zur Struktur $M = (d,\delta)$: $ß : \mathbf{V} \to d$

Semantik: Gültigkeit

Struktur $M = (d,\delta)$ und **Variablenbelegung** ß legen *Bedeutung* **aller** Sprachmittel fest:

1. ß auf beliebige Terme fortsetzen:

$$ß(f^0) := \delta(f^0)$$
$$ß(f^n(t_1,...,t_n)) := \delta(f^n)(ß(t_1),...,ß(t_n))$$

2. Formeln auswerten:

$\models_{M,ß} P^n(t_1,...,t_n)$	:gdw	$(ß(t_1),...,ß(t_n)) \in \delta(P^n)$
$\models_{M,ß} (\Phi \wedge \Psi)$	:gdw	$\models_{M,ß} \Phi$ und $\models_{M,ß} \Psi$
$\models_{M,ß} (\Phi \vee \Psi)$	:gdw	$\models_{M,ß} \Phi$ oder $\models_{M,ß} \Psi$
$\models_{M,ß} (\neg\Phi)$	:gdw	nicht $\models_{M,ß} \Phi$
$\models_{M,ß} (\forall x)\ \Phi$	:gdw	für alle ß' mit $ß =_x ß'$ gilt $\models_{M,ß'} \Phi$
$\models_{M,ß} (\exists x)\ \Phi$	:gdw	es gibt ß' mit $ß =_x ß'$ mit $\models_{M,ß'} \Phi$

$ß =_x ß'$ bedeutet hier:
die Variablenbelegungen ß und ß' sind für alle Variablen gleich, bis auf möglicherweise für die Variable x

Semantik: Modelle und logische Implikation

- **Modell einer Formel**:

  $M = (d,\delta)$ ist *Modell* einer Formel Φ,
  $\models_M \Phi$ (Φ *gültig* in M) :gdw für alle Belegungen ß gilt : $\models_{M,ß} \Phi$

- **Modellklasse einer Formel (-menge) Φ**:

  Mod (Φ) := $\{M \mid \Phi$ ist gültig in $M\}$

- **Allgemeingültigkeit**:

  Formel Φ ist *allgemeingültig* :gdw jede Struktur M ist Modell von Φ

- **Erfüllbarkeit**:

  Formel Φ ist *erfüllbar* :gdw es gibt Struktur M, die Modell von Φ ist

- **Unerfüllbarkeit**:

  Formel Φ ist *unerfüllbar* :gdw es gibt keine Struktur M,
  die Modell von Φ ist

- **logische Implikation**:

  Formel (-menge) Φ *impliziert logisch* Formel (-menge) Ψ, $\Phi \models \Psi$,
  :gdw Mod (Φ) $\subset$ Mod (Ψ)

Mengenlehre

- Bildung neuer Mengen aus vorgegebenen Mengen, hier insbesondere:

  aus einer Menge d von (einfachen) Werten
  neue Mengen von möglicherweise *strukturierten* Werten induktiv gewinnen:

  ist d eine nichtleere Menge,
  so aus d induktiv eine Klasse κ_d von Mengen konstruieren:

  - d ist in κ_d
  - sind m, n aus κ_d, so auch

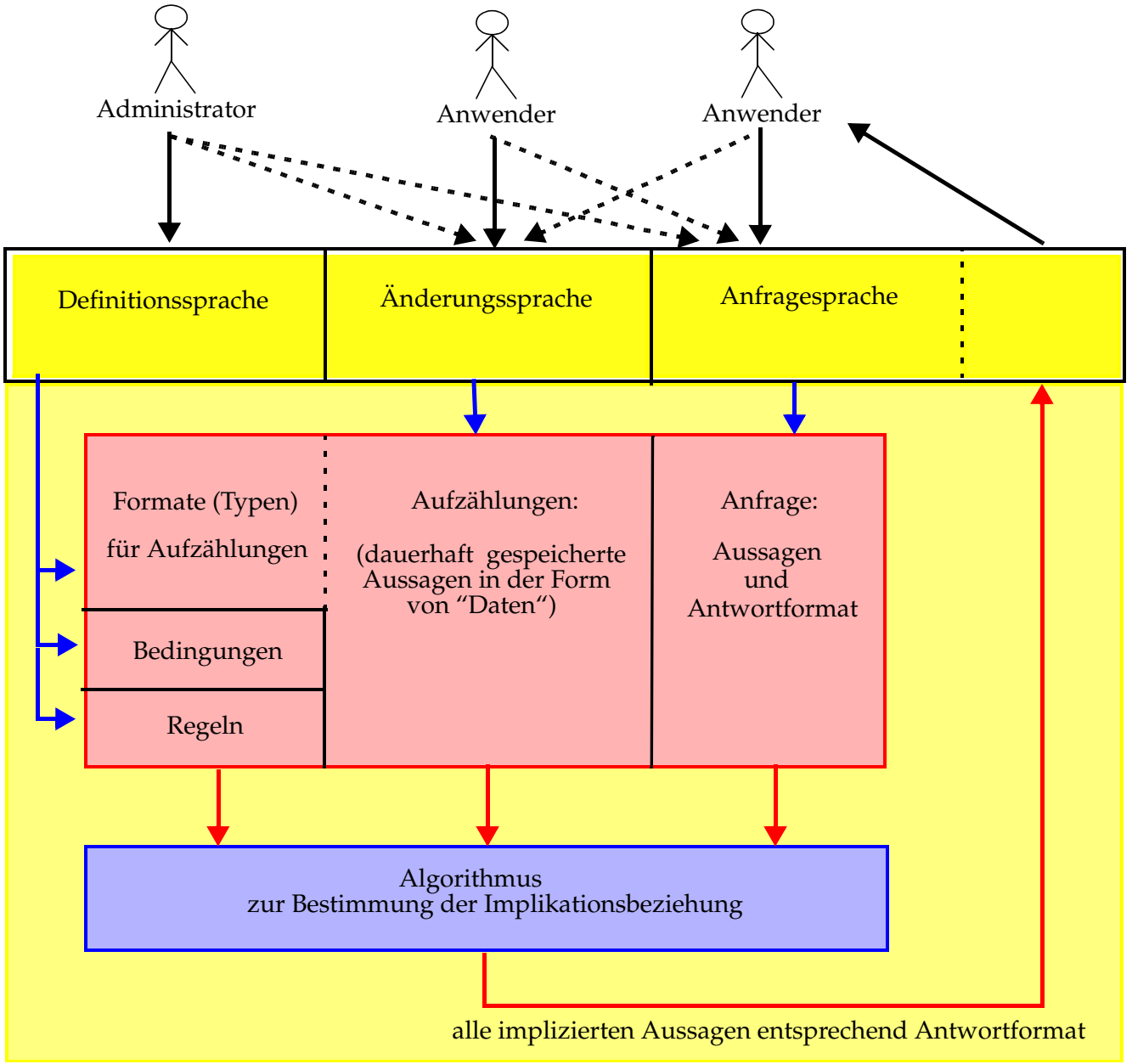
    $m \times n$ *kartesisches Produkt,*
    $m \cup n$ *Vereinigung,*
    $\wp m$ *Potenzmenge*

- Bildung von Teilmengen oder ausgesonderten Mengen:

  ist $M = (d,\delta)$ eine Struktur und
  ist Φ eine Formel, in der die Variablen $x_1,...,x_k$ frei vorkommen,
  so die durch Φ (aus $\times_{i=1,...,k}\ d$) *ausgesonderte Menge* konstruieren:

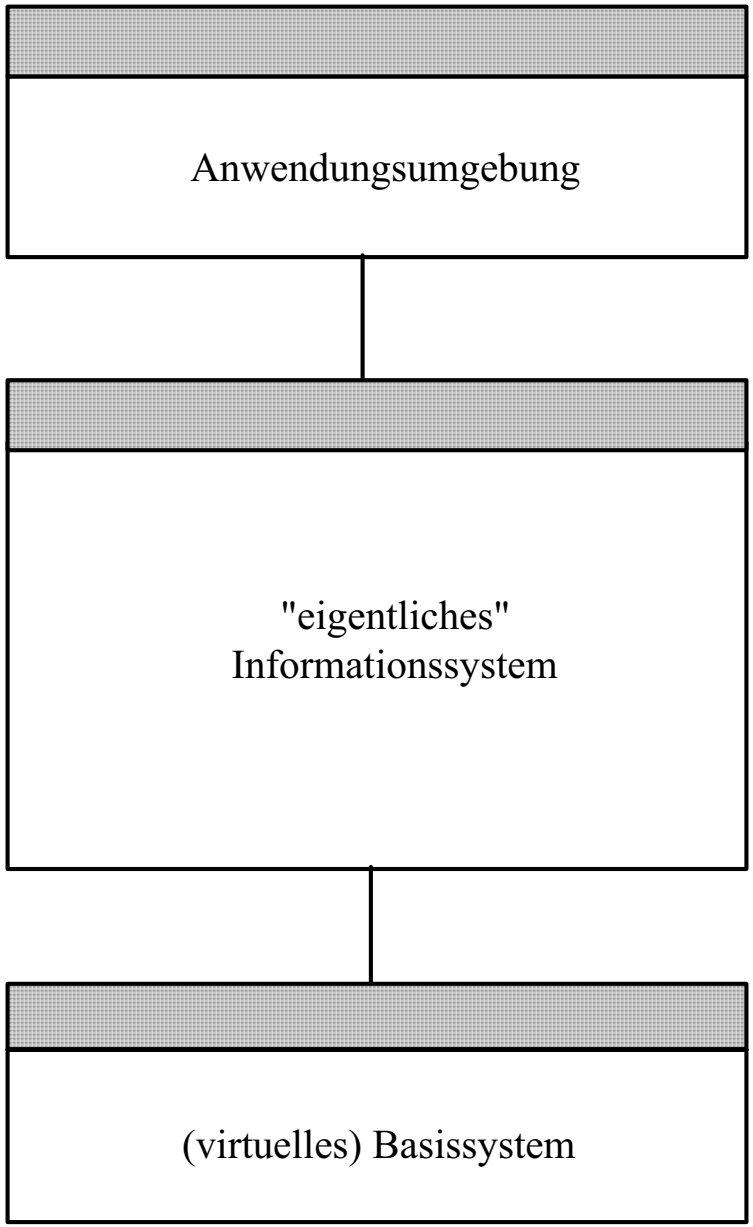
  $$d_{M,\Phi} := \{\ (ß(x_1),...,ß(x_k)) \quad | \quad ß \text{ ist Variablenbelegung mit } \models_{M,ß} \Phi \quad \}$$

Logik und Informationssysteme



1.4 Architekturen

Architektur eines Informationssystems: Grobstruktur

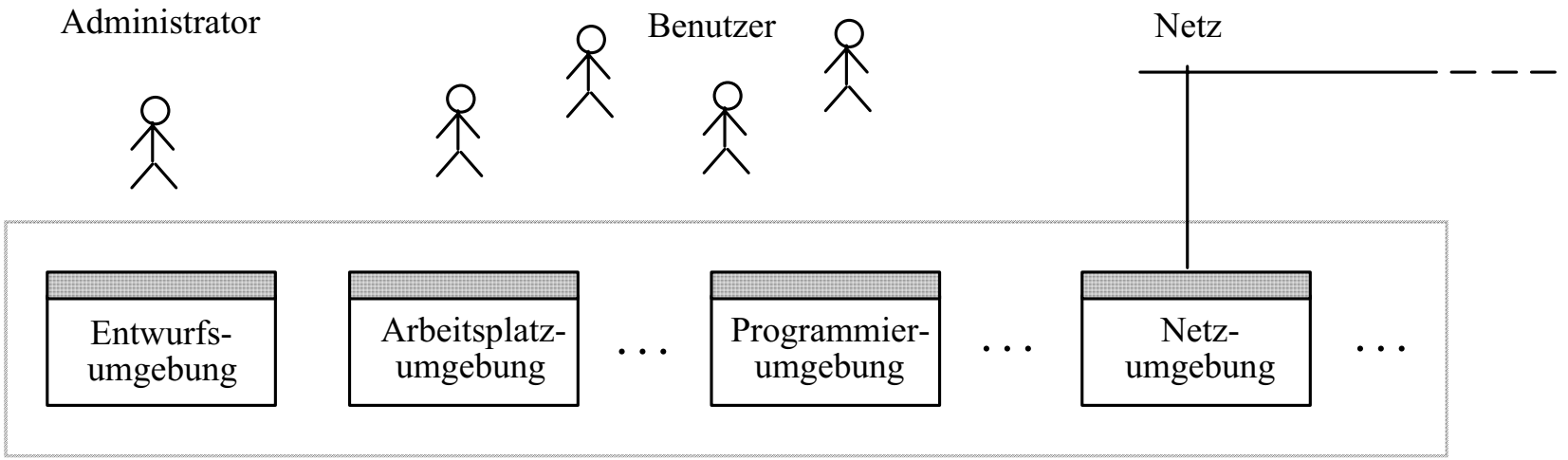


Einsatz-Schnittstelle

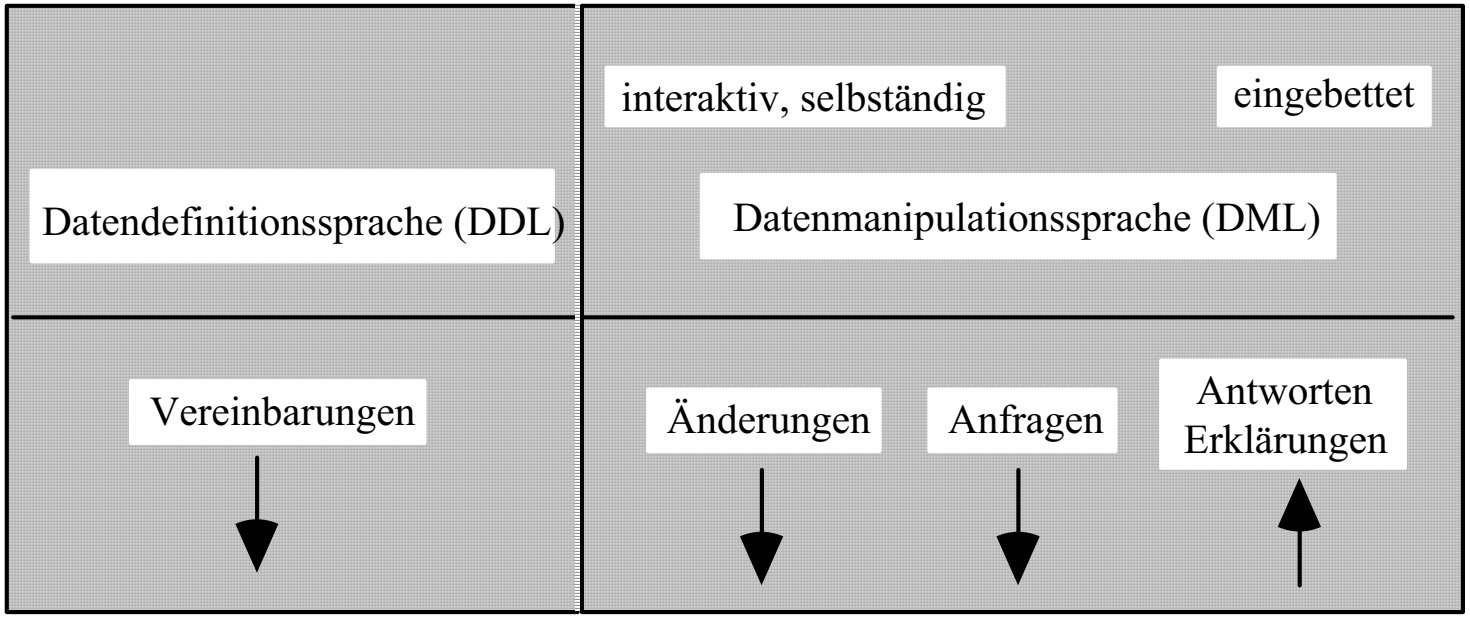
Informationssystem-
Schnittstelle

Basissystem-
Schnittstelle

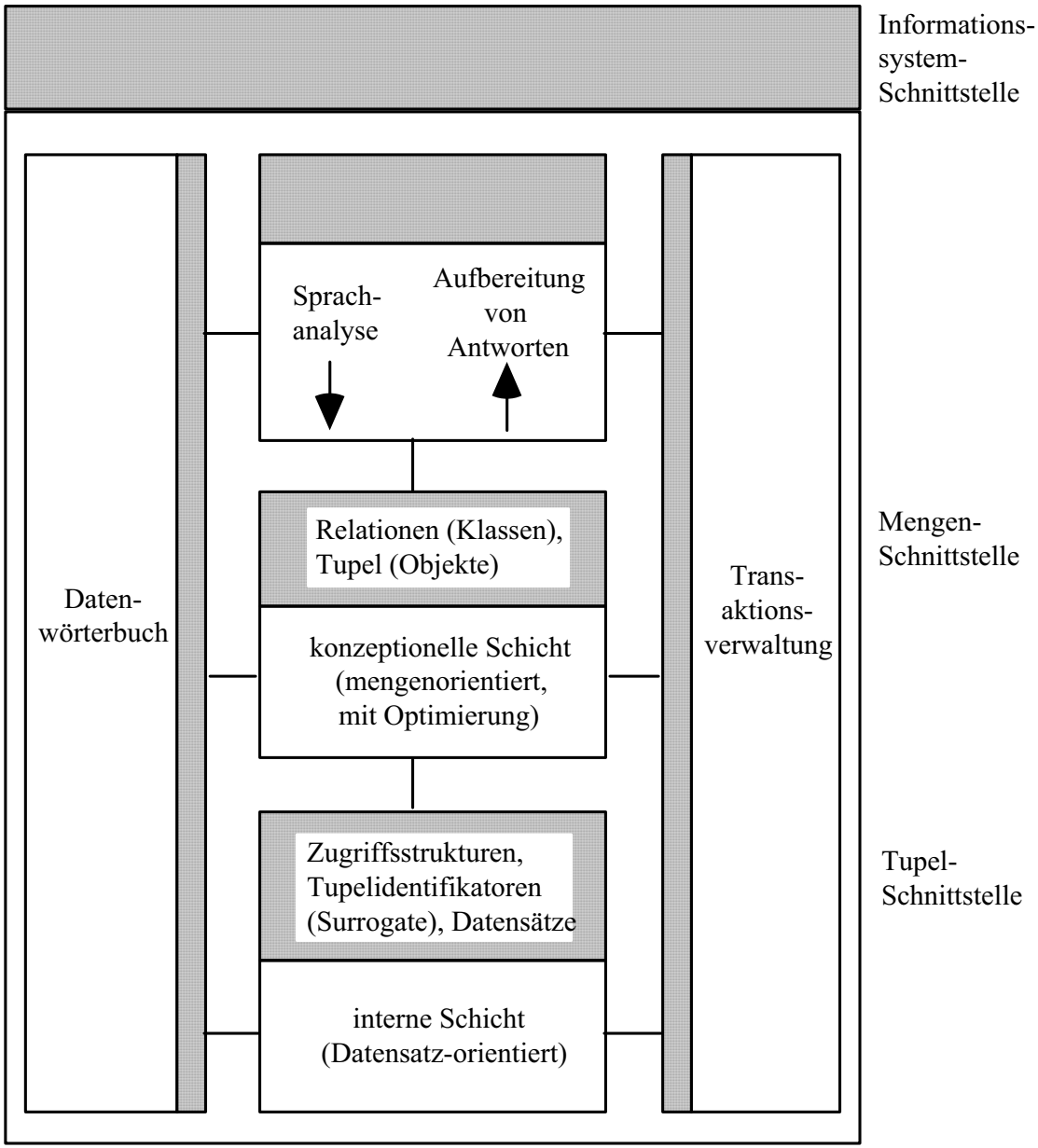
Anwendungsumgebungen für ein Informationssystem



Informationssystem-Schnittstelle



Schichtung und Komponenten eines Informationssystems



Mengen-Schnittstelle

- ***Änderungen*:**
  - Relationen (Klassen), Tupel (Objekte)
  - Einfügen, Entfernen, Abändern (und weitere Klassenfunktionen)

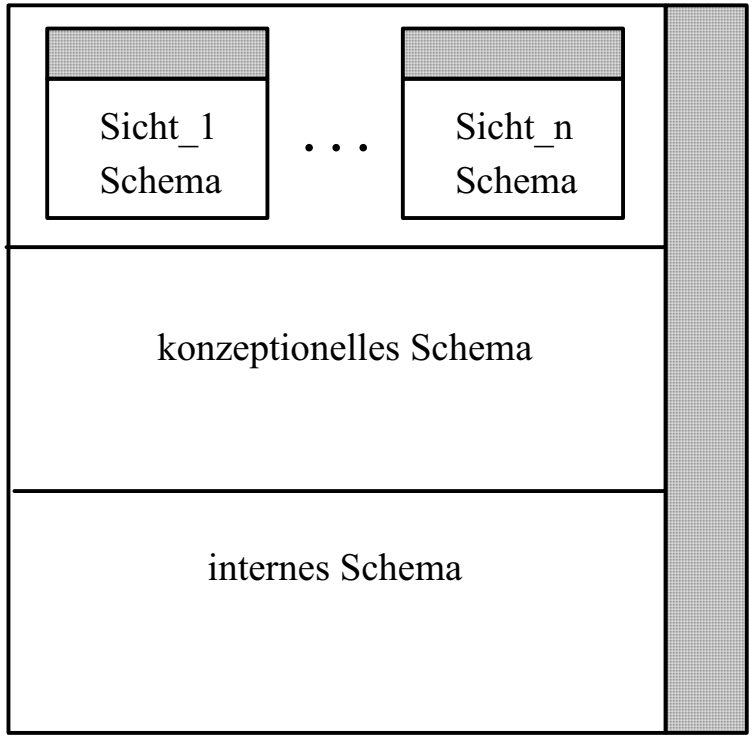
- ***Anfragen*:**
  - Relationen (Klassen)
  - Relationen- (Klassen-) Algebra (Klassenfunktionen), möglicherweise Erweiterung durch Rekursion usw.

Tupel-Schnittstelle

- ***Strukturen***:
  - Relationen (Klassen) als Dateien
  - Zugriffsstrukturen: sequentielle Listen, Indexe, Links
  - Relationendurchläufe (Klasseniteratoren)
  - Tupel (Objekte) als Datensätze
  - Tupelidentifikatoren (Surrogate)

- ***Operationen***:
  - Anlegen und Entfernen von Relationen (Klassen)
  - Anlegen und Entfernen von Zugriffsstrukturen, insbesondere Sortieren
  - Öffnen, Schließen von Durchläufen
  - Bestimmen eines Tupelidentifikators (Surrogates), insbesondere bei Durchläufen
  - Transport eines Tupels (Objektes) als Datensatz, insbesondere
  - Herstellen einer Hauptspeicherkopie bzw.
  - (Zurück-) Schreiben in den dauerhaften Speicher
  - Erzeugen, Vergleichen, Abändern, Entfernen
    von Hauptspeicherkopien von Tupeln (Objekten)

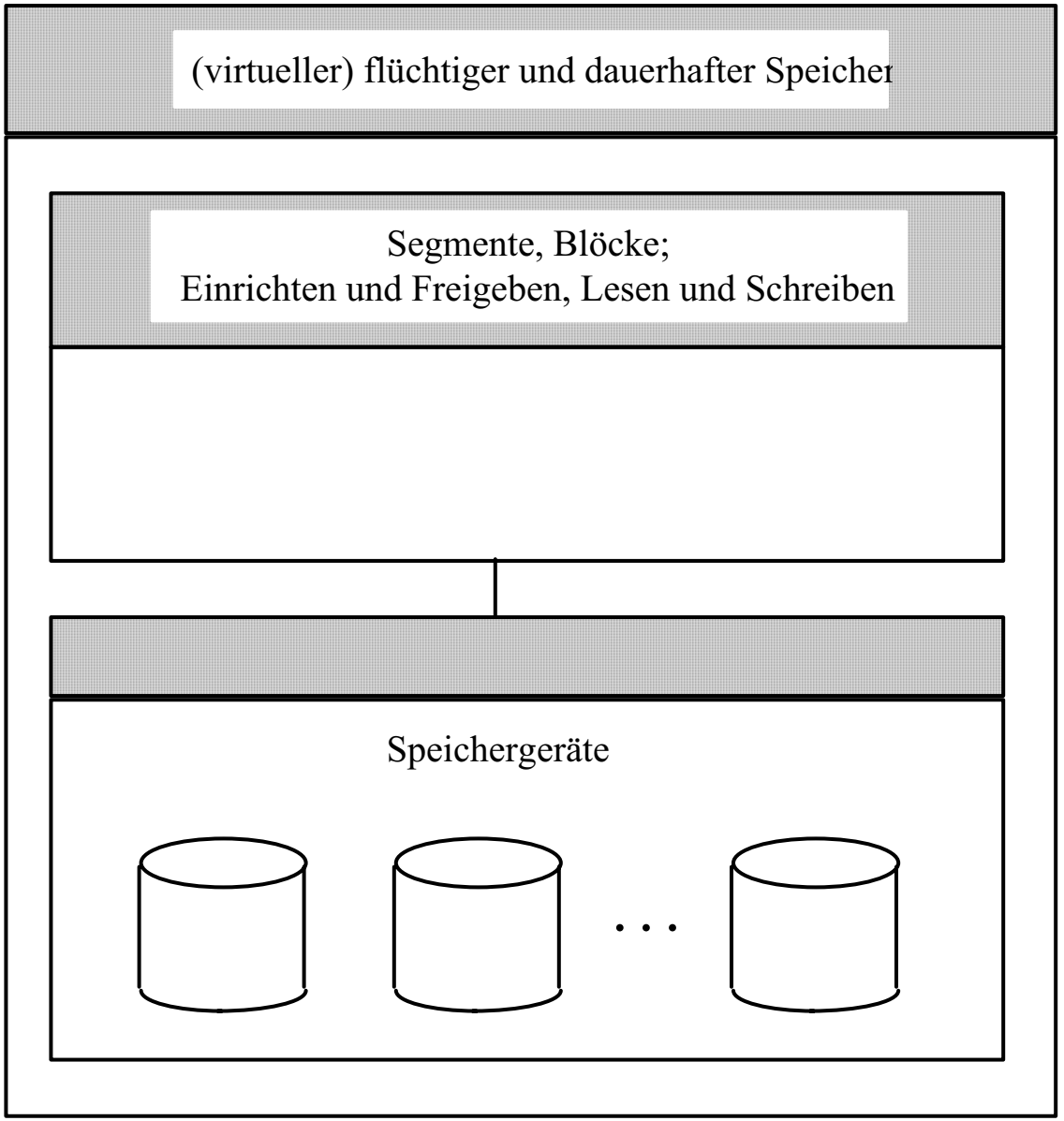
Datenwörterbuch (data dictionary)

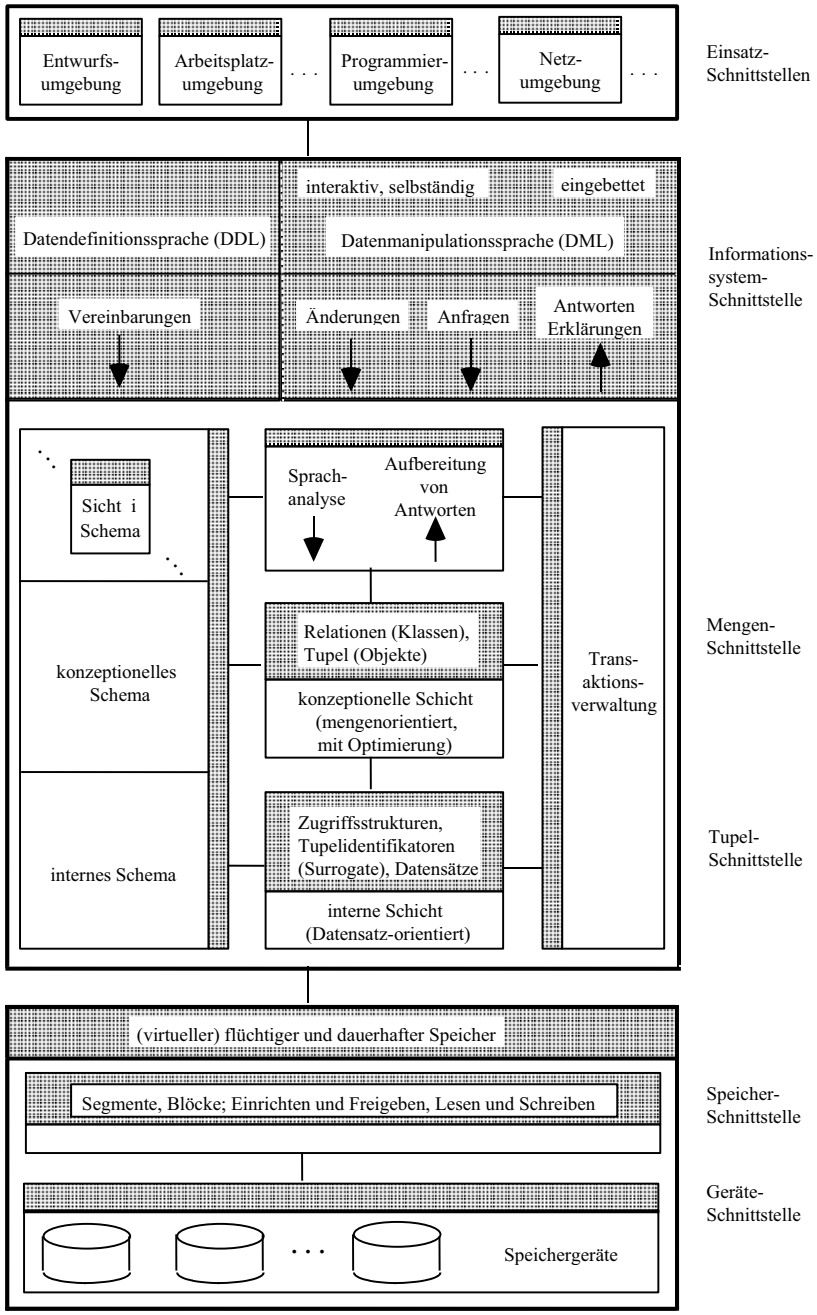


Transaktionsverwaltung

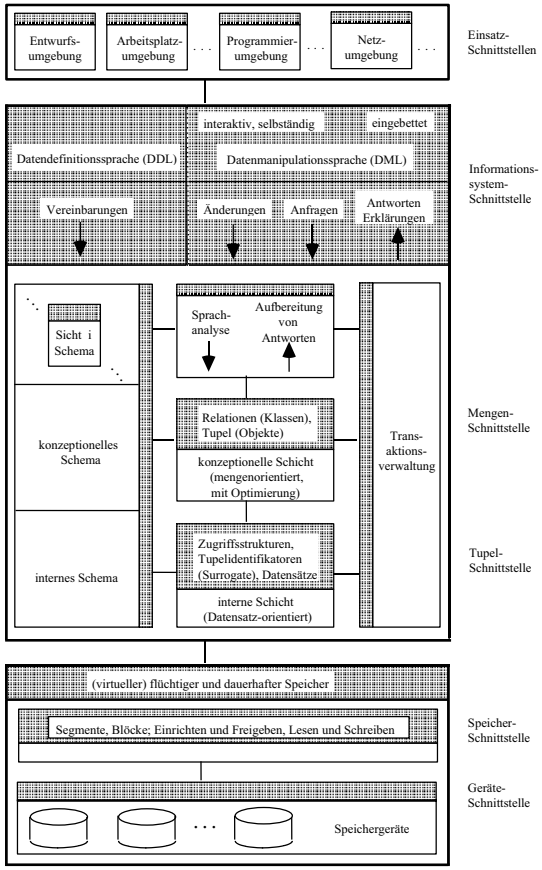
- für Mehrbenutzerbetrieb, Konsistenzwahrung, Verlässlichkeit, Fehlertoleranz, ...
- im Allgemeinen schichtenübergreifend angelegt

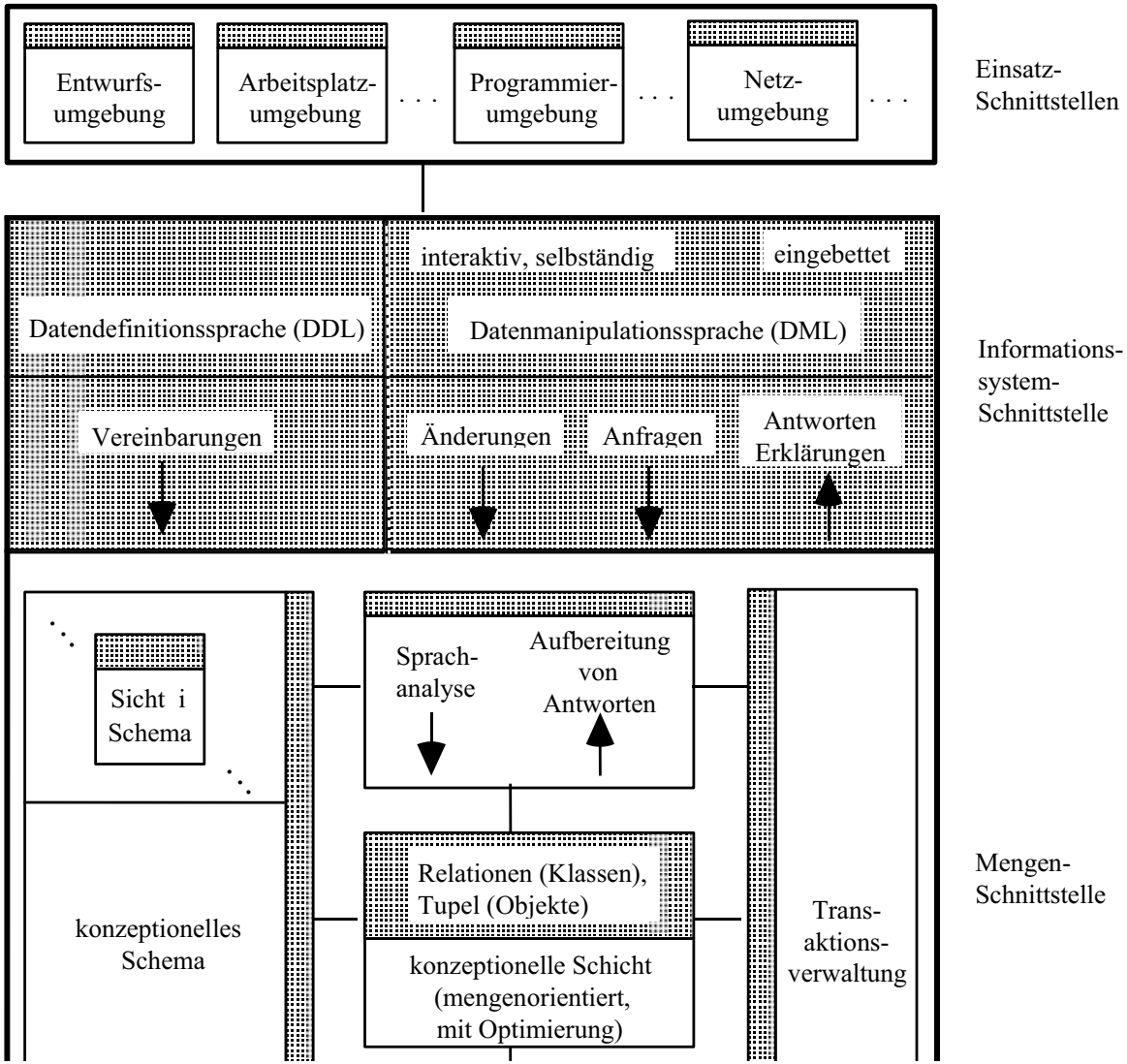
Basissystem



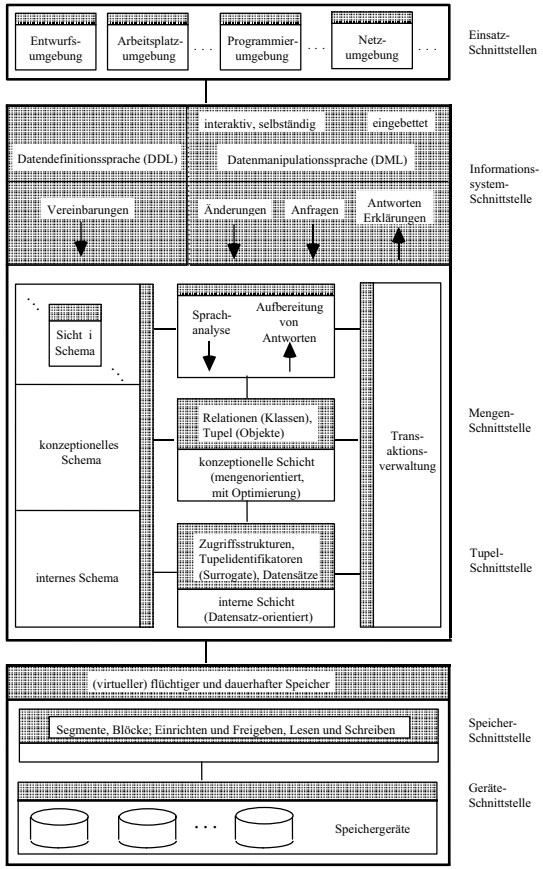

Entwurfs-umgebung
Arbeitsplatz-umgebung
Programmier-umgebung
Netz-umgebung
Einsatz-Schnittstellen
interaktiv, selbständig
eingebettet
Datendefinitionssprache (DDL)
Datenmanipulationssprache (DML)
Informations-system-Schnittstelle
Vereinbarungen
Änderungen
Anfragen
Antworten Erklärungen
Sicht i Schema
konzeptionelles Schema
internes Schema
Sprach-analyse
Aufbereitung von Antworten
Relationen (Klassen), Tupel (Objekte)
konzeptionelle Schicht (mengenorientiert, mit Optimierung)
Zugriffsstrukturen, Tupelidentifikatoren (Surrogate), Datensätze
interne Schicht (Datensatz-orientiert)
Trans-aktions-verwaltung
Mengen-Schnittstelle
Tupel-Schnittstelle
(virtueller) flüchtiger und dauerhafter Speicher
Segmente, Blöcke; Einrichten und Freigeben, Lesen und Schreiben
Speicher-Schnittstelle
Speichergeräte
Geräte-Schnittstelle

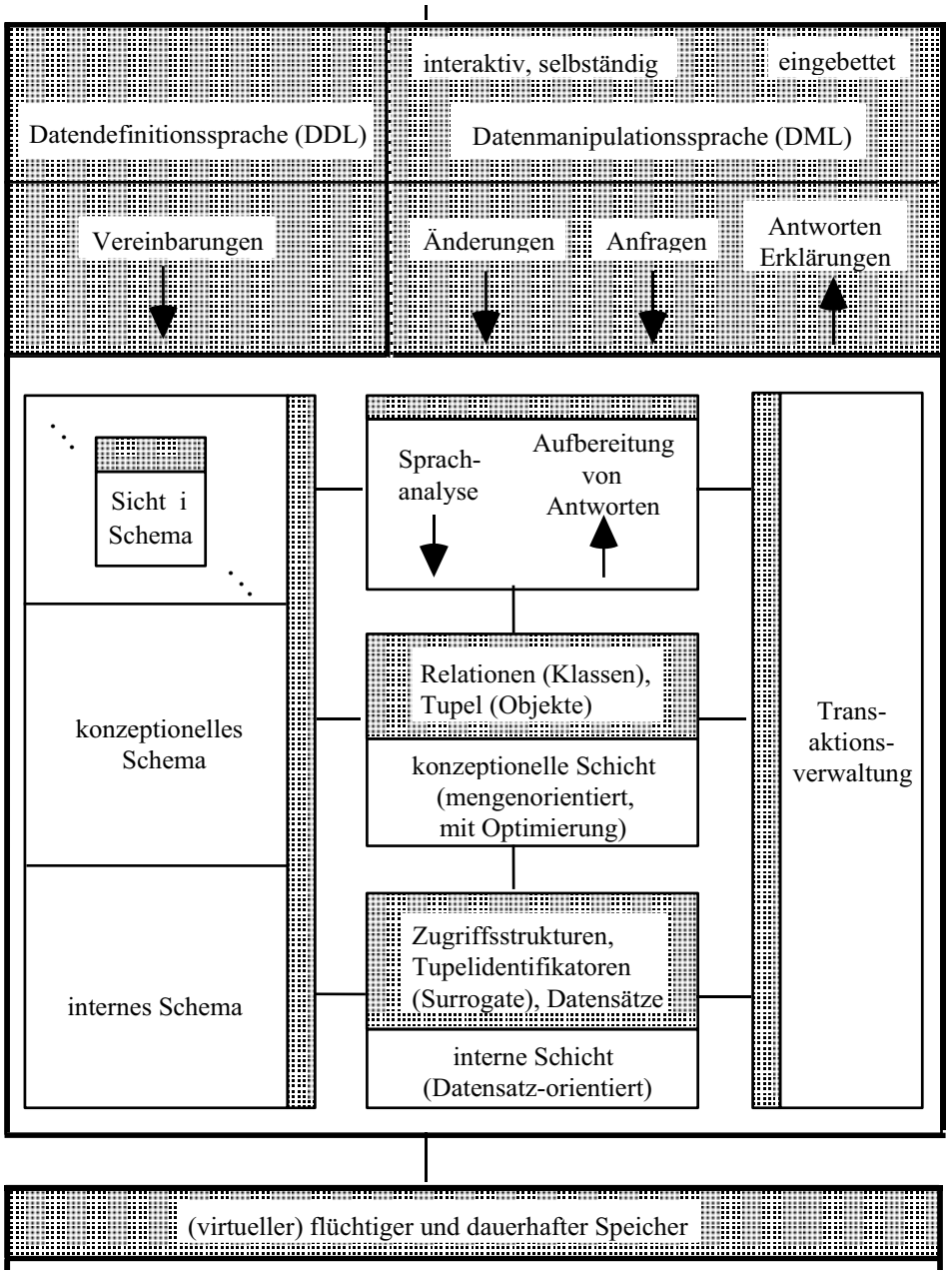
Architektur: benutzersichtbarer Teil



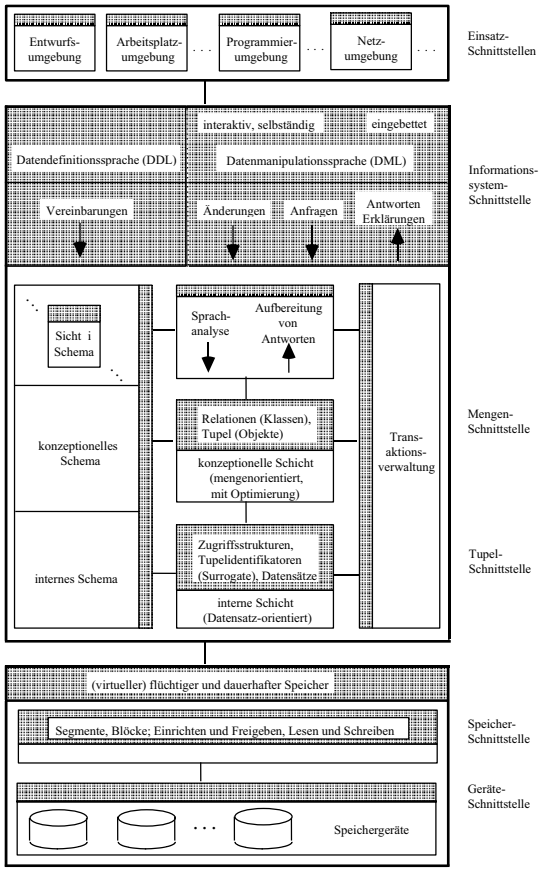


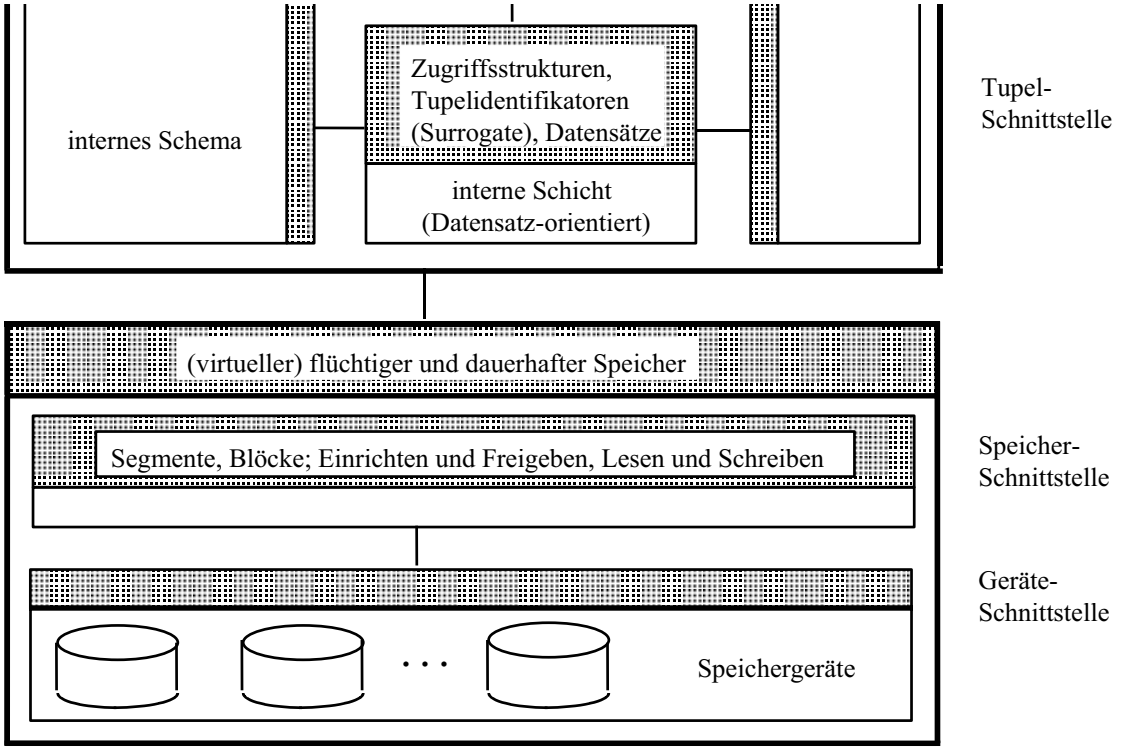
Architektur: “eigentliches Informationssystem”



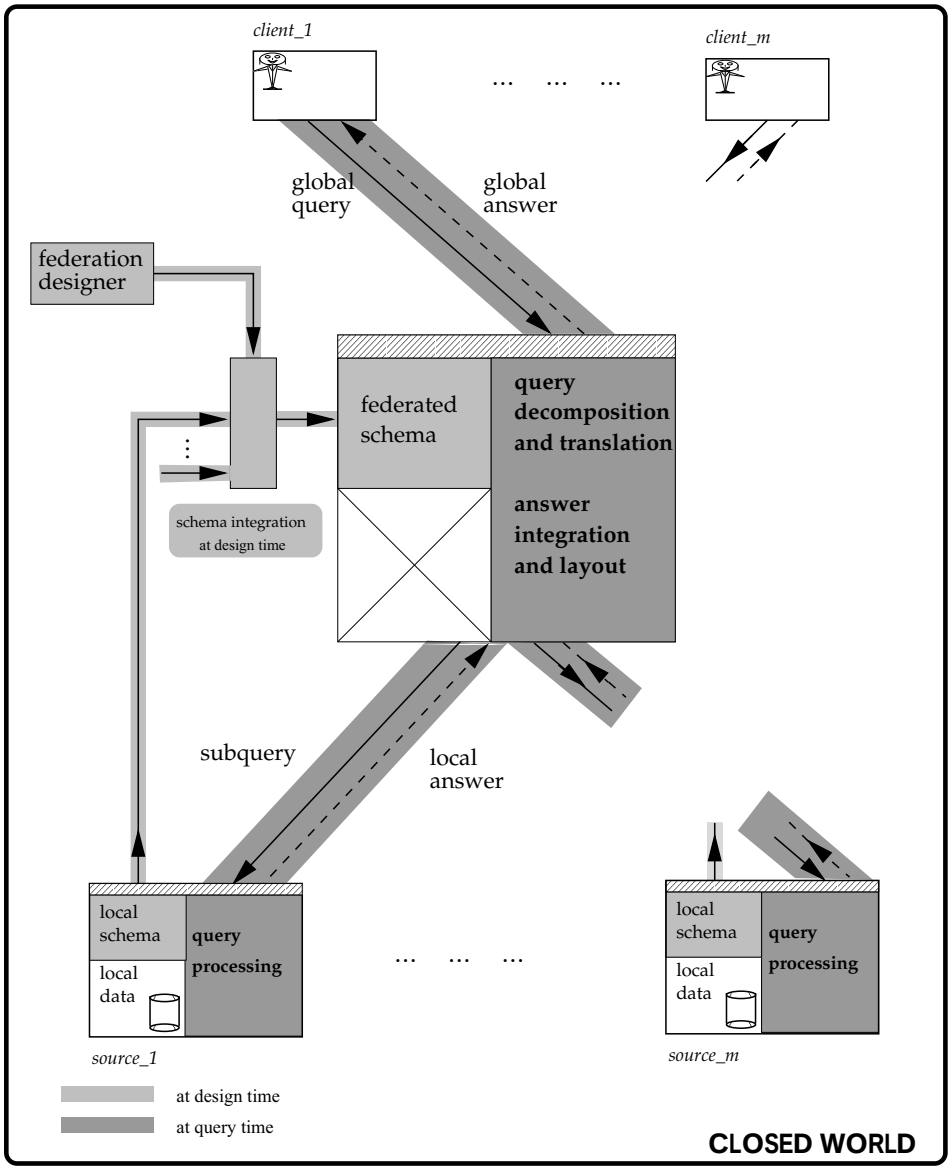


Architektur: Basissystem



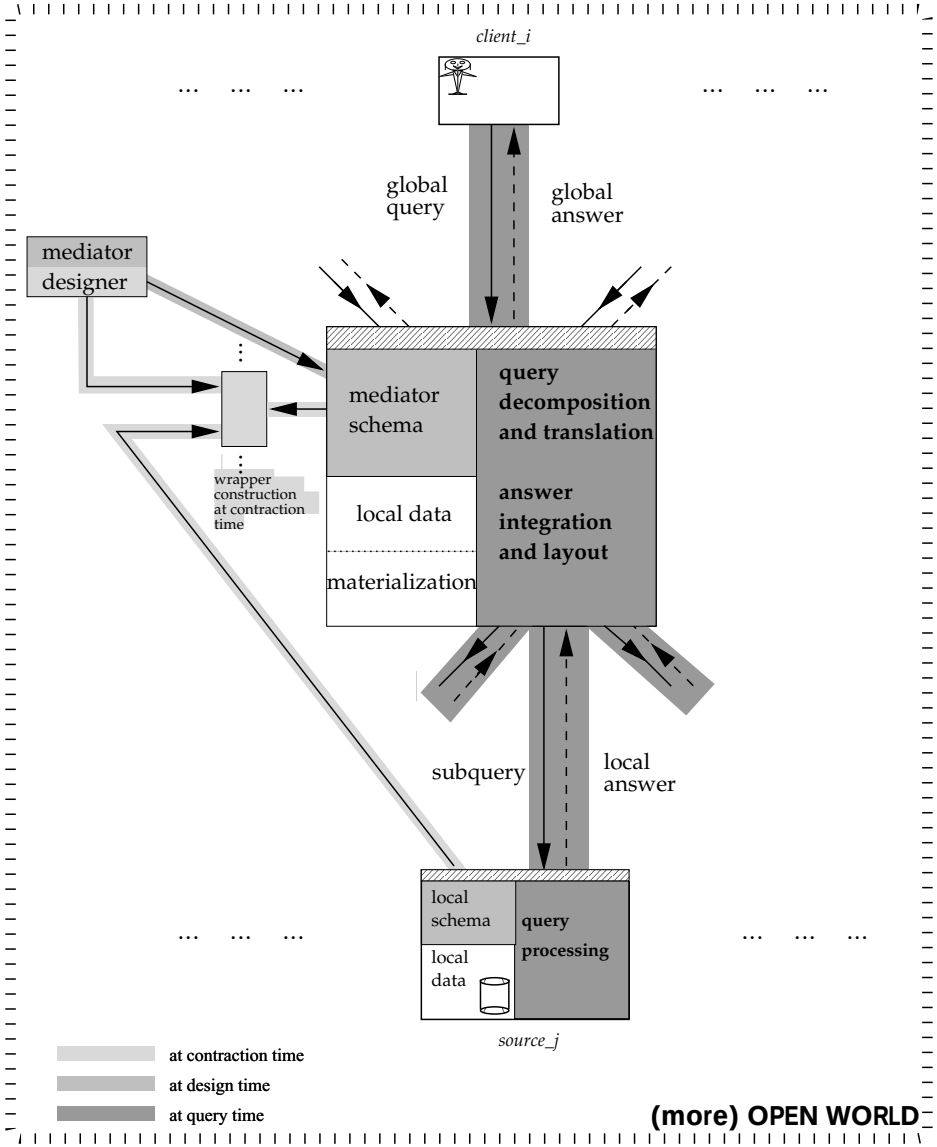


föderierte Systeme



(more) static environment
(more) organizational structures

mediierte Systeme



(more) dynamic environment
(more) contractual relationships

1.5 drei Haltungen zu Informationssystemen

- **“eher positivistisch”**
- **“eher skeptizistisch”**
- **“algorithmisch”**

eine “eher positivistische” Haltung

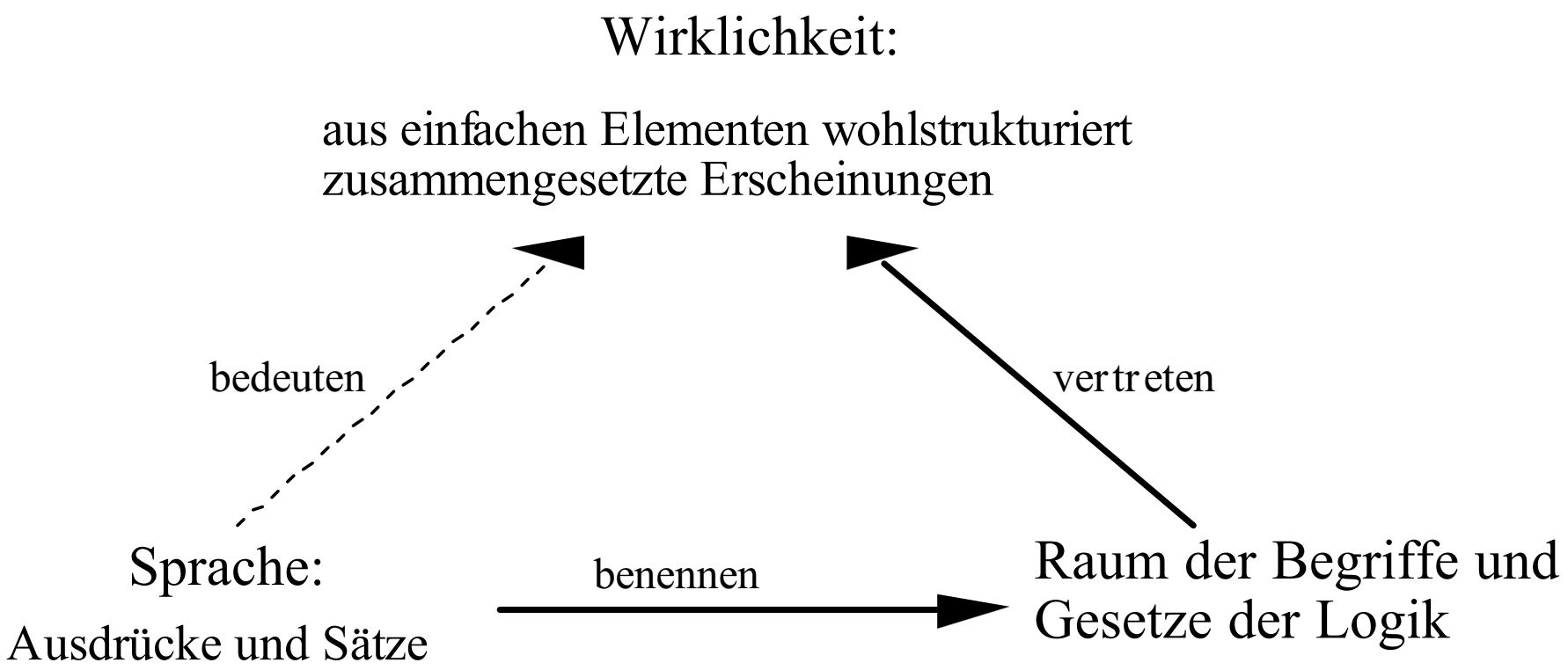
1.1.3 Die Tatsachen im logischen Raum sind die Welt.

6.13 Die Logik ist keine Lehre,
sondern ein Spiegelbild der Welt.

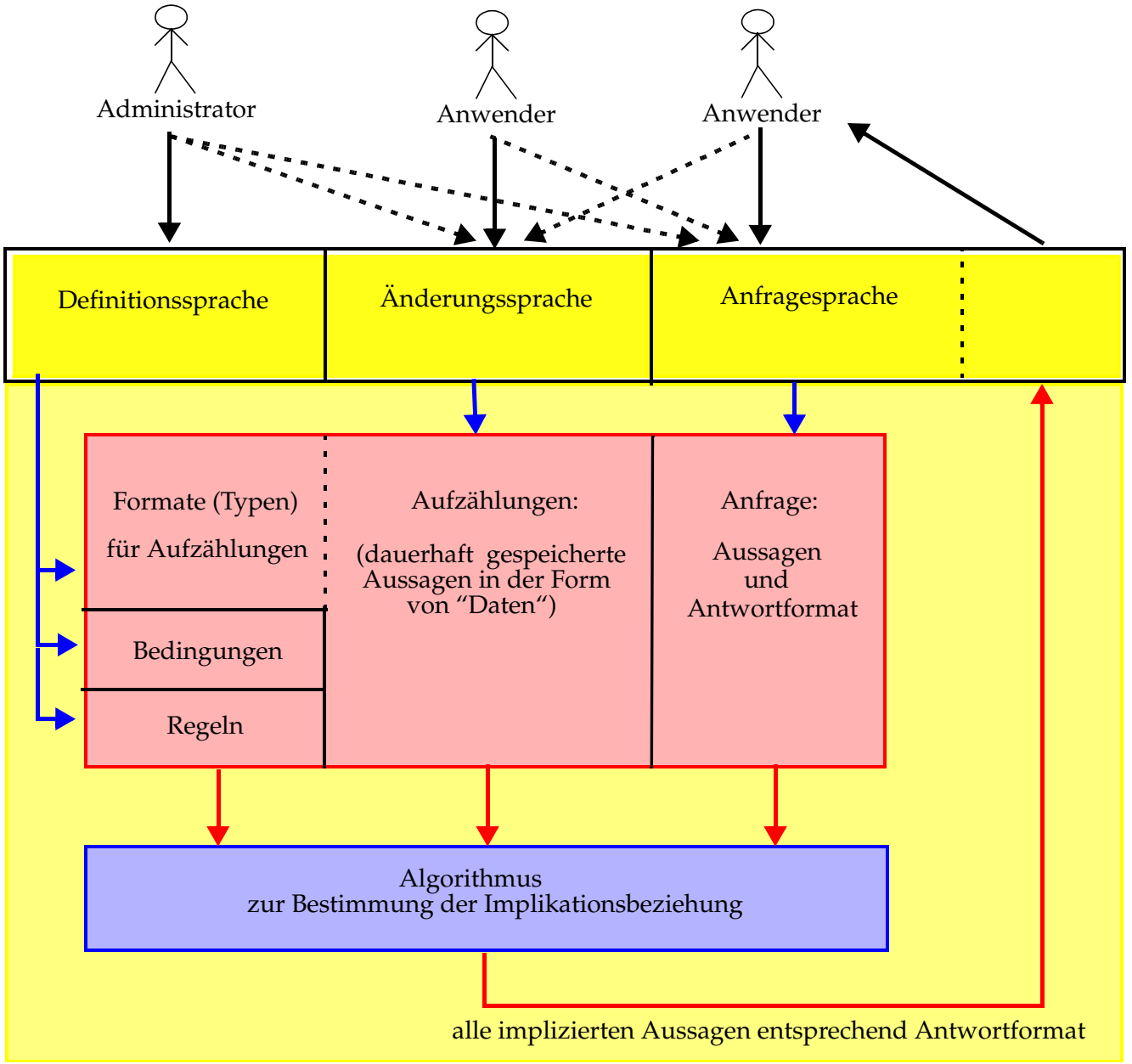
7 Wovon man nicht sprechen kann,
darüber muss man schweigen.

L. Wittgenstein
Tractatus logico-philosophicus
1921

Wirklichkeit - Begriffsraum - Sprache



Logik und Informationssysteme



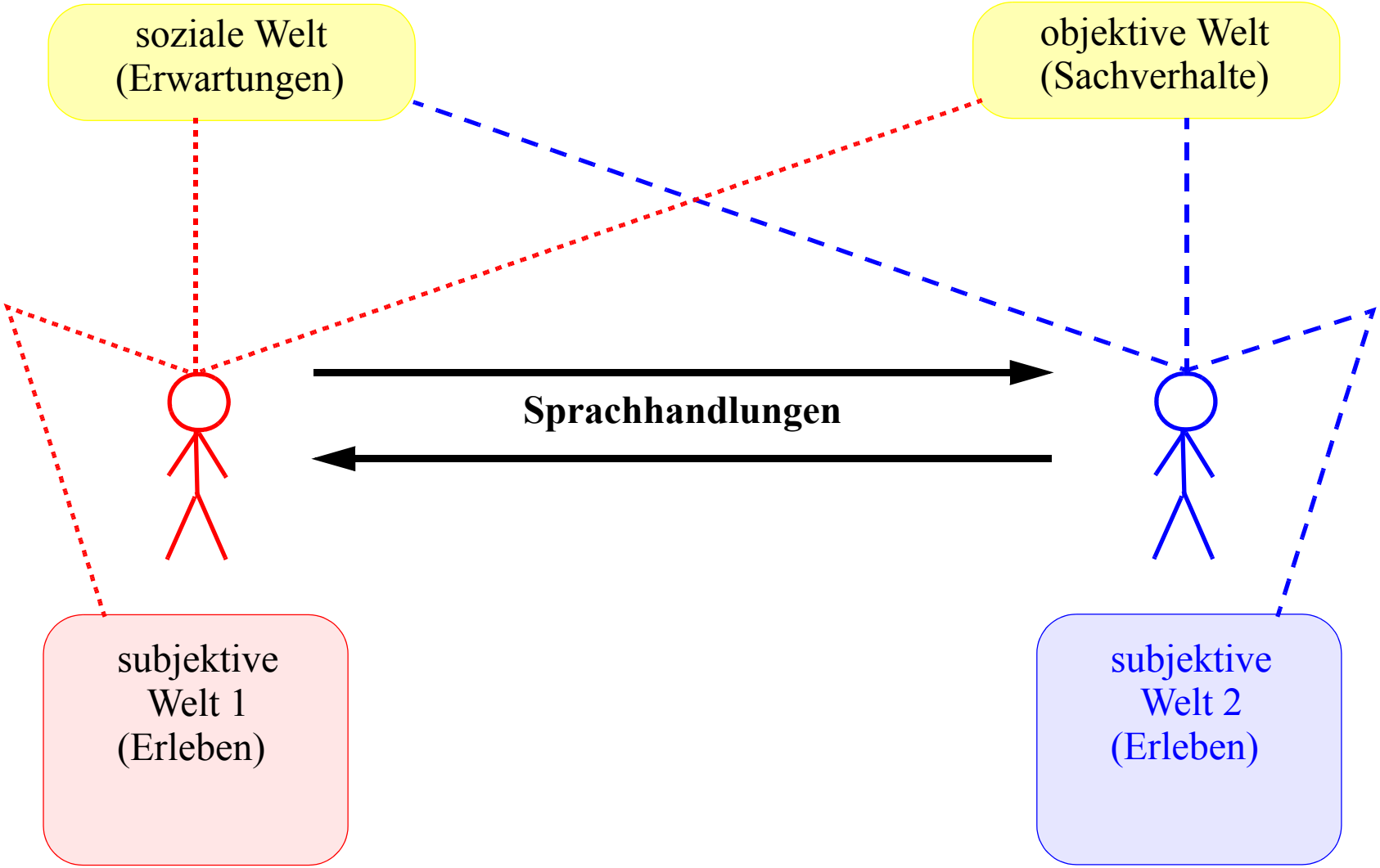
eine “eher skeptizistische” Haltung

In Wirklichkeit sieht alles ganz anders aus,
als es wirklich ist.

Alle unsere unterschiedlichen Fiktionen
ergeben zusammen die gemeinsame Wirklichkeit.

S.J. Lec
Unfrisierte Gedanken
1971

Veranschaulichung kommunikativ Handelnder mit Weltbezügen

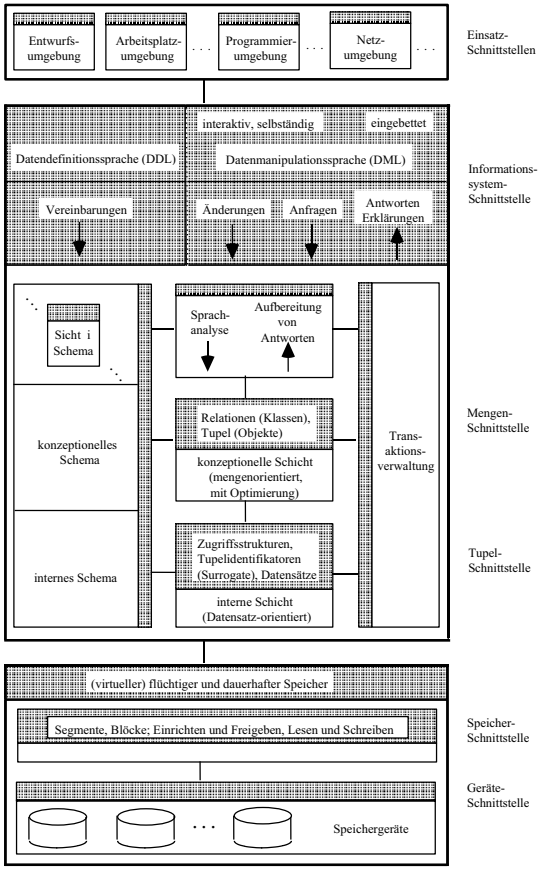


eine “algorithmische” Haltung

Der Computer ist kein Wunder.
Er arbeitet nur deshalb so schnell,
weil er nicht denkt.

G. Laub
Erlaubte Freiheiten - Aphorismen
1975

Architektur: “eigentliches Informationssystem”



interaktiv, selbständig
eingebettet
Datendefinitionssprache (DDL)
Datenmanipulationssprache (DML)
Vereinbarungen
Änderungen
Anfragen
Antworten Erklärungen
Informations-system-Schnittstelle
Sicht i Schema
konzeptionelles Schema
internes Schema
Sprach-analyse
Aufbereitung von Antworten
Relationen (Klassen), Tupel (Objekte)
konzeptionelle Schicht (mengenorientiert, mit Optimierung)
Zugriffsstrukturen, Tupelidentifikatoren (Surrogate), Datensätze
interne Schicht (Datensatz-orientiert)
Trans-aktions-verwaltung
Mengen-Schnittstelle
Tupel-Schnittstelle
(virtueller) flüchtiger und dauerhafter Speicher

Teil II

Relationale Systeme mit strukturierten Daten

2 Relationales Datenmodell: Strukturen

Datenmodell

(abstrakter) Datentyp für Informationssysteme:

- ***Strukturen*:** **Schema und Instanzen**
- ***Operationen*:** **Änderungen und Anfragen**

bekannte Datenmodelle:

- hierarchisch
- Netzwerk-
- **relational**
- logikorientiert
- objektorientiert
- ...

relationales Datenmodell

- *Relationen*: Abstraktion von Dateien gleichartiger Records
- *modelltheoretisch* logikorientiert: *Relationen* (Tabellen) stellen *Strukturen* dar
- *werteorientiert*: aus Benutzersicht erfolgen Identifikationen nur über Werte
- ...

2.1 relationale Strukturen: Theorie

Beispiel

PERSON	Name	Geschlecht
	Mensch	Geschlecht
	wilhelmine	weib
	fritz	mann
	anton	mann
	theresia	weib
	maria	weib
	hugo	mann
	alfons	mann
	josef	mann
	gerda	weib

ELT	Name	Eltern
	Mensch	Mensch
	maria	anton
	hugo	anton
	alfons	theresia
	anton	wilhelmine
	anton	fritz
	theresia	wilhelmine
	theresia	fritz
	gerda	josef

BEH	Patient	Arzt	ArtPro
	Mensch	Mensch	ArtPro
	maria	gerda	labor
	maria	gerda	hausbesuch
	maria	gerda	röntgen
	anton	josef	untersuchung
	anton	josef	labor
	anton	josef	beratung
	fritz	josef	beratung
	theresia	josef	röntgen

Schreibweisen: bespielhaft

Relationensymbol
$R \in \mathbf{R} \setminus \{=\}$

Menge von Attributen
$X \subset_{\text{endlich}} \mathbf{A}$

Vereinbarung semantischer Bereichsnamen für Attribute **a**

BEH	Patient	Arzt	ArtPro
	Mensch	Mensch	ArtPro
	.	.	.
	.	.	.
	anton	josef	labor

semantische Bereichsnamen

Tupel $\mu : X \to \mathbf{C}$

Menge der Konstanten-zeichen

$\mu(\text{Patient}) = \text{anton}$ $\mu(\text{Arzt}) = \text{josef}$ $\mu(\text{ArtPro}) = \text{labor}$

$\text{dom}\ \mu = \{ \text{Patient}, \text{Arzt}, \text{ArtPro} \}$

Schreibweisen: allgemein

$\mathbf{A}$	unendliche Menge der ***Attribute***
$R, S, \ldots, X, Y, Z$	endliche Mengen von Attributen
$\mathbf{B}$	Menge der ***semantischen Bereichsnamen***
$\mathbf{C}$	unendliche Menge der ***Konstantenzeichen***
$\mathbf{b} : \mathbf{B} \to \wp\mathbf{C}$	ordnet semantischen Bereichsnamen ***Domänen*** zu
$\mathbf{R}$	Menge der ***Relationensymbole***, mit Gleichheit(ssymbol) =
$\mu : X \to \mathbf{C}$	***Tupel*** mit
$\mathrm{dom}\,\mu := X \subset_{\text{endlich}} \mathbf{A}$	***Definitionsbereich***

für $X = \{A_1, \ldots, A_n\}$ kann man μ aufschreiben als $\mu = \left(\begin{pmatrix} A_1 \\ \mu(A_1) \end{pmatrix}, \ldots, \begin{pmatrix} A_n \\ \mu(A_n) \end{pmatrix} \right)$

oder als $\mu = (\mu(A_1), \ldots, \mu(A_n))$

$\mu\lceil Y$ mit $\mu\lceil Y : \mathrm{dom}\,\mu \cap Y \to \mathbf{C}$,
$\mu\lceil Y(A) := \mu(A)$ für $A \in \mathrm{dom}\,\mu \cap Y$ ***Einschränkung*** von μ auf Y

$r \subset_{\text{endlich}} \{ \mu \mid \mu : X \to \mathbf{C} \}$ für ein $X \subset \mathbf{A}$ ***Relation***

$\mathrm{dom}\,r := \mathrm{dom}\,\mu$ für $\mu \in r$ ***Definitionsbereich***

semantische Bedingungen

- **lokale Bedingungen:**
  beziehen sich auf *ein* Relationenschema

  - funktionale Abhängigkeiten
  - mehrwertige Abhängigkeiten
  - Verbundabhängigkeiten
  - ...

- **globale (interrelationale) Bedingungen:**
  beziehen sich auf *mehrere* Relationenschemas

  - Enthaltenseinsabhängigkeiten
  - ...

funktionale Abhängigkeiten (functional dependencies, FDs)

kurz: $X_1,...,X_k \rightarrow Y$

inhaltlich: Attributwerte für $X_1,...,X_k$ bestimmen Attributwert für Y

Formel: $Y = Y'$:- $R(\ X_1,...,X_k,\ Y\ ,\ Z_1,...,Z_l\ )_{,(\text{and})}$
$R(\ X_1,...,X_k,\ Y'\ ,\ Z'_1,...,Z'_l\ )$

Beispiel: Name → Geschlecht im Relationenschema für PERSON

der (eindeutig vergebene) Attributwert für Name
bestimmt (in der Relation für PERSON)
den Attributwert für Geschlecht

Geschlecht = Geschlecht' :- PERSON(Name, Geschlecht)$_{,(\text{and})}$
PERSON(Name, Geschlecht')

mehrwertige Abhängigkeiten (multivalued dependencies, MVDs)

kurz: $X_1,...,X_k \rightarrow\rightarrow Y_1,...,Y_l \,/\, Z_1,...,Z_m$

inhaltlich: Attributwerte für $X_1,...,X_k$ bestimmen *Menge* der Attributwerte für $Y_1,...,Y_l$ (bzw. $Z_1,...,Z_m$)

Formel:

$$R(X_1,...,X_k, Y'_1,...,Y'_l, Z_1,...,Z_m) :-$$
$$R(X_1,...,X_k, Y_1,...,Y_l, Z_1,...,Z_m),_{(\text{and})}$$
$$R(X_1,...,X_k, Y'_1,...,Y'_l, Z'_1,...,Z'_m)$$

Beispiel: wird später behandelt

Verbundabhängigkeiten (join dependencies, JDs)

Verallgemeinerung von mehrwertigen Abhängigkeiten

Enthaltenseinsabhängigkeiten (inclusion dependencies, INDs)

kurz ("algebraisch"): $\pi_{X1,...,Xk}(S) \subset \pi_{Y1,...,Yk}(R)$

inhaltlich: Attributwerte für $X_1,...,X_k$ in (der Relation für) S
kommen auch vor als
Attributwerte für $Y_1,...,Y_k$ in (der Relation für) R

Formel: $R1(X_1,...,X_k)$:- $S1(X_1,...,X_k)$
mit $R1$ und $S1$
"geeignete Projektionen" von Basisrelationen R und S

Beispiel: $\pi_{\text{Patient}}(\text{BEH}) \subset \pi_{\text{Name}}(\text{PERSON})$

jeder Attributwerte für Patient in BEH
kommt auch vor als
Attributwert für Name in PERSON

Relationenschema und Instanz

Syntax: $< R \mid X \mid SC >$ (Relationen-) *Schema* mit

- $R \in \mathbf{R} \setminus \{=\}$ Relationensymbol (aber nicht die Gleichheit)
- $X \subset_{\text{endlich}} \mathbf{A}$ Attribute
- SC lokale semantische Bedingungen

Semantik: (d, r) *Instanz* (oder *gültige Ausprägung*) zu $< R \mid X \mid SC >$:gdw

i) $d \subset_{\text{endlich}} \mathbf{C}$

ii) $\text{dom}\, r = X$ und $r \subset_{\text{endlich}} \{ \mu \mid \mu : X \to d \}$

iii) (d, r) ist Modell von SC

Instanzenmenge:

$\text{sat}_{< R \mid X \mid SC >} := \{ (d, r) \mid (d, r)$ ist Instanz zu $< R \mid X \mid SC > \}$

Relationales Datenbankschema und Instanz

Syntax: $RS = < < R_1 \mid X_1 \mid SC_1 > , \ldots , < R_n \mid X_n \mid SC_n > \mid SC \mid \mathbf{a} >$

(relationales Datenbank-) Schema mit

- $< R_i \mid X_i \mid SC_i >$ Relationenschema mit $R_i \neq R_j$ für $i \neq j$
- SC (globale) semantische Bedingungen
- $\mathbf{a} : \bigcup_{i=1,\ldots,n} X_i \rightarrow \mathbf{B}$ Vereinbarung der semantischen Bereichsnamen

Semantik: $M = (d , r_1 , \ldots , r_n)$ *Instanz* (*gültige Ausprägung*) zu RS :gdw

i) $d \subset_{\text{endlich}} \mathbf{C}$

ii) (d , r_i) ist Instanz zu $< R_i \mid X_i \mid SC_i >$

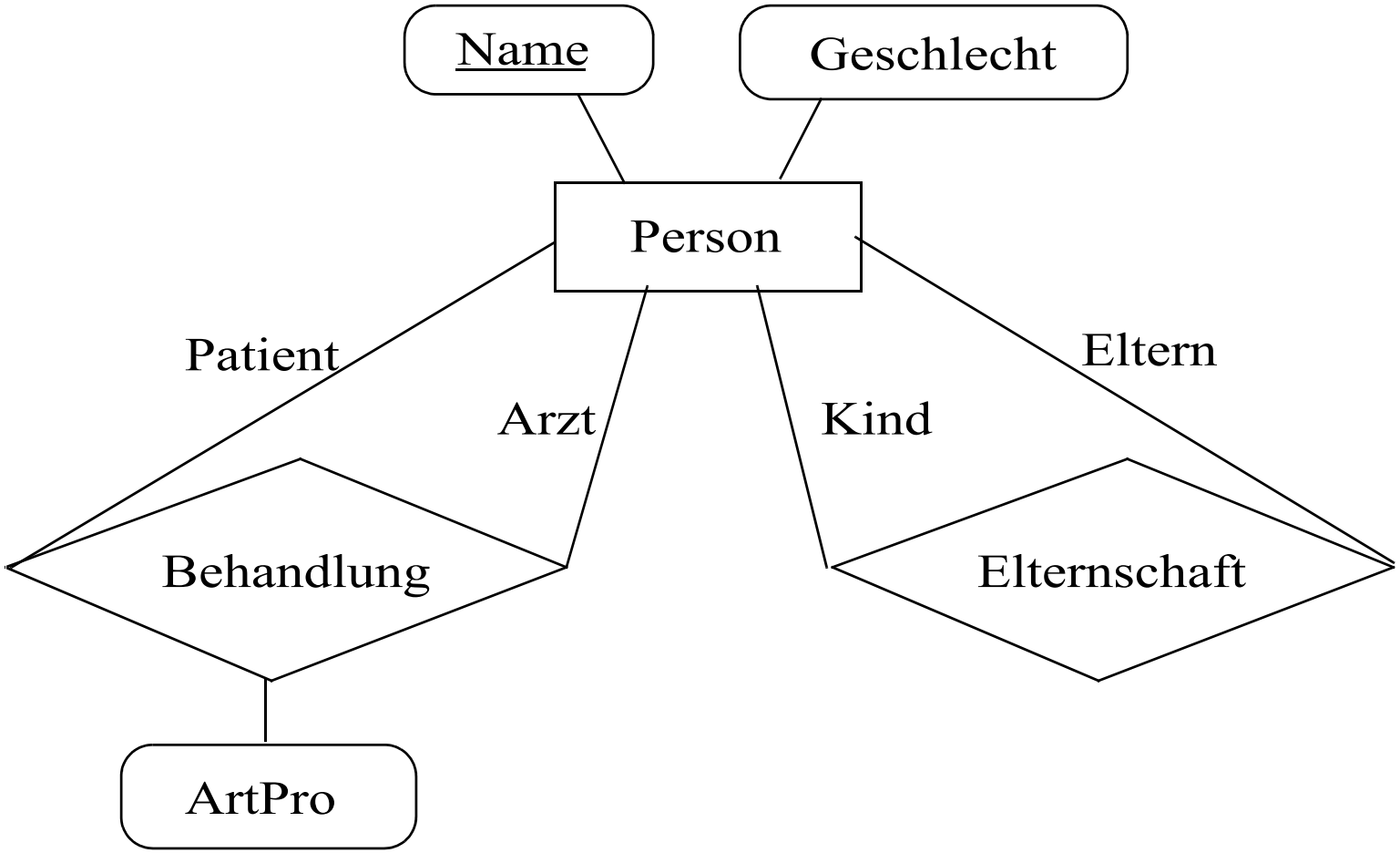
iii) $(d , r_1 , \ldots , r_n)$ ist Modell von SC

iv) falls $\mu \in r_i$ und $A \in X_i$, dann gilt $\mu(A) \in \mathbf{b} \circ \mathbf{a} (A)$

Instanzenmenge:

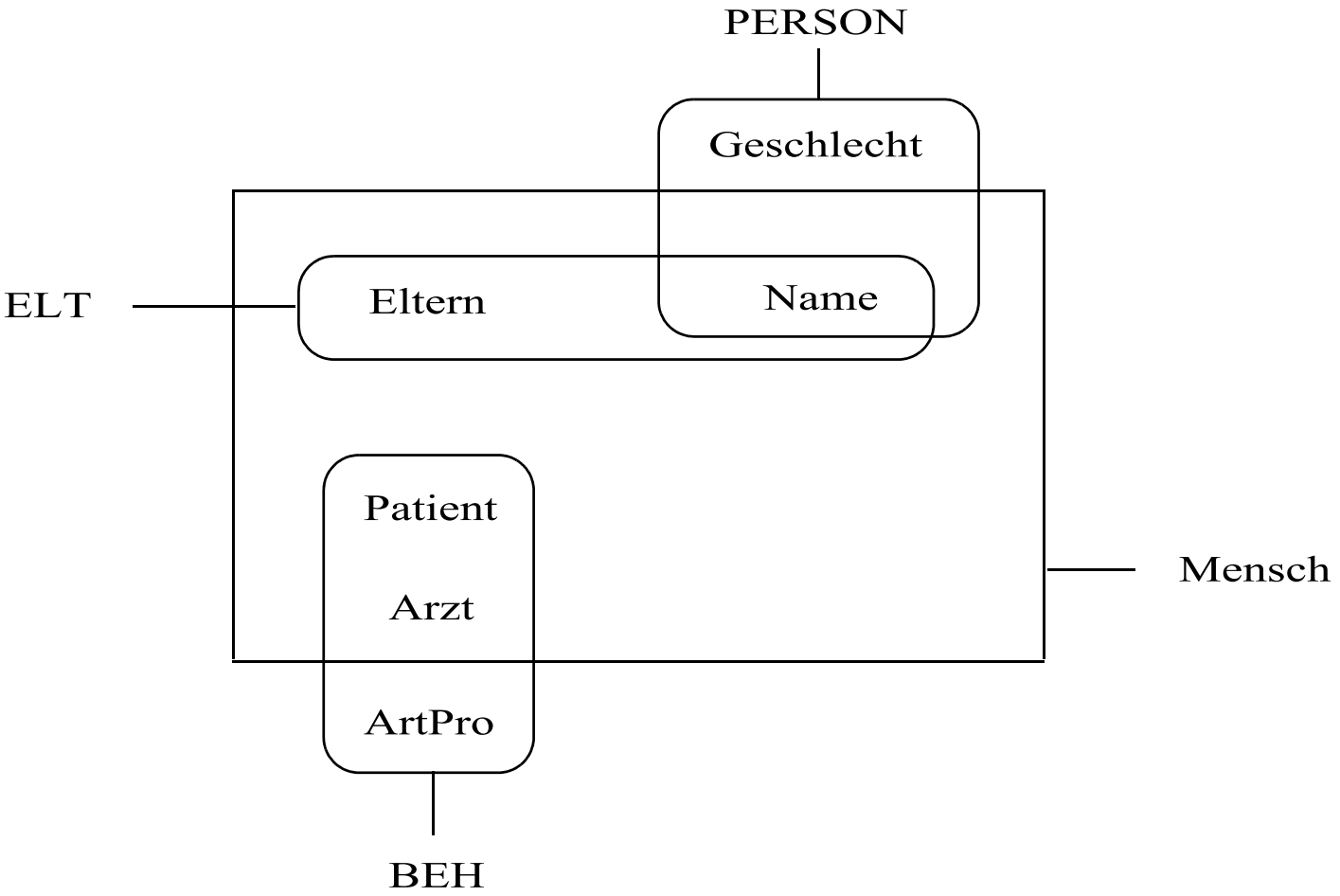
$\text{sat}_{RS} := \{ (d , r_1 , \ldots , r_n) \mid (d , r_1 , \ldots , r_n)$ ist Instanz zu $RS \}$

Beispiel: (vereinfachte) Arztpraxis als ER-Diagramm



Beispiel: (vereinfachte) Arztpraxis als Hypergraph des Schemas

Knoten: Attribute
runde Hyperkanten: Relationenschemas
rechteckige Hyperkanten: (nichttrivial benutzte) semantische Bereichsnamen



Beispiel: (vereinfachte) Arztpraxis als Datenbankschema

RS = <

< PERSON | {Name, Geschlecht} | {Name → Geschlecht} >,
< BEH | {Patient, Arzt, ArtPro} | ∅ >,
< ELT | {Name, Eltern} | ∅ >

|

{ $\pi_{Patient}$ (BEH) ⊂ π_{Name} (PERSON),
π_{Arzt} (BEH) ⊂ π_{Name} (PERSON),
π_{Name} (ELT) ⊂ π_{Name} (PERSON),
π_{Eltern} (ELT) ⊂ π_{Name} (PERSON) }

|

a (Name) := **a** (Patient) := **a** (Arzt) := **a** (Eltern) := Mensch,
a (Geschlecht) := Geschlecht,
a (ArtPro) := ArtPro

>

b (Mensch) := Σ^* ⊂ **C**

b (Geschlecht) := { mann, weib } ⊂ **C**

b (ArtPro) := { untersuchung, beratung, hausbesuch, labor, röntgen } ⊂ **C**

Beispiel: Tabellengerüste zum relationalen Datenbankschema

PERSON	Name	Geschlecht
	Mensch	Geschlecht

ELT	Name	Eltern
	Mensch	Mensch

BEH	Patient	Arzt	ArtPro
	Mensch	Mensch	ArtPro

Beispiel: eine Instanz zum relationalen Datenbankschema

PERSON	Name	Geschlecht
	Mensch	Geschlecht
	wilhelmine	weib
	fritz	mann
	anton	mann
	theresia	weib
	maria	weib
	hugo	mann
	alfons	mann
	josef	mann
	gerda	weib

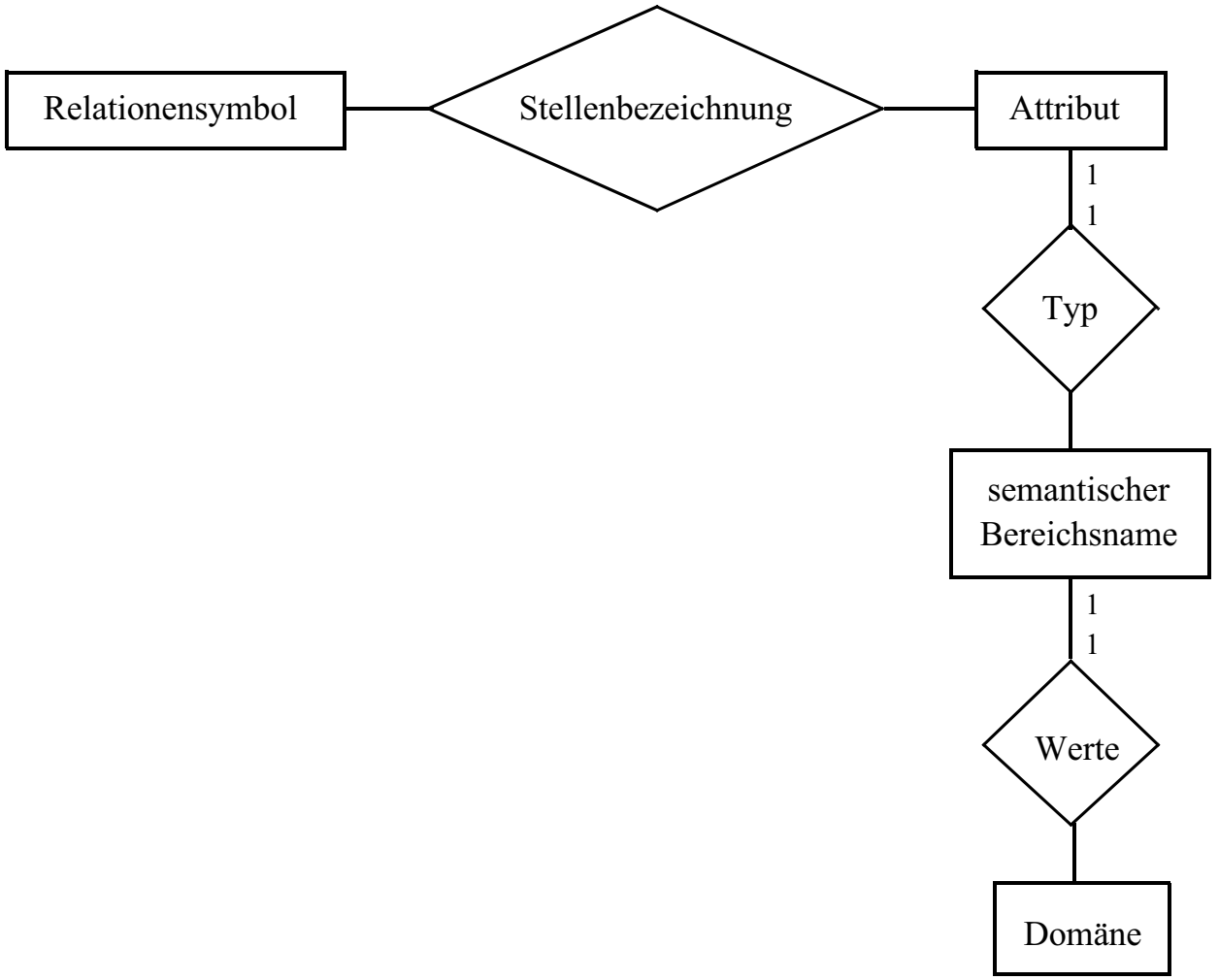
ELT	Name	Eltern
	Mensch	Mensch
	maria	anton
	hugo	anton
	alfons	theresia
	anton	wilhelmine
	anton	fritz
	theresia	wilhelmine
	theresia	fritz
	gerda	josef

BEH	Patient	Arzt	ArtPro
	Mensch	Mensch	ArtPro
	maria	gerda	labor
	maria	gerda	hausbesuch
	maria	gerda	röntgen
	anton	josef	untersuchung
	anton	josef	labor
	anton	josef	beratung
	fritz	josef	beratung
	theresia	josef	röntgen

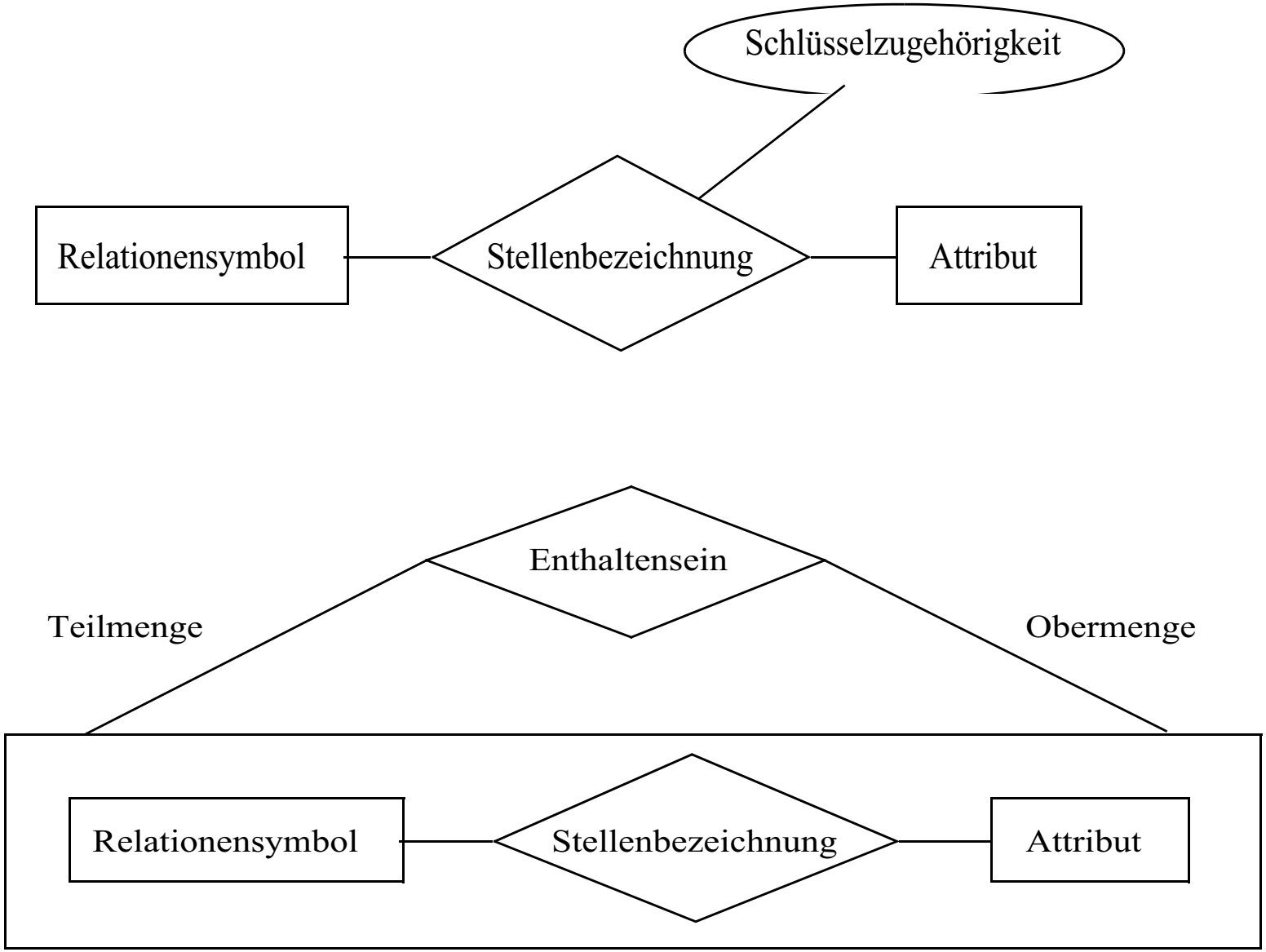
2.2 relationale Meta-Strukturen: Theorie

Schema als Selbstbeschreibung

- mit den gleichen Techniken wie die Instanzen behandeln
- als ***zeitunabhängig*** angesehene Gegebenheiten vereinfacht modellieren:



Modellierung der semantischen Bedingungen



(vereinfachtes) “Metaschema” für relationale Datenbanken

RS_{Schema} = <

< RELATION | {Symbol, Attribut, Schlüssel} | {Symbol, Attribut → Schlüssel} >,
< TYP | {Attribut, SemBereich} | {Attribut → SemBereich} >,
< WERTE | {SemBereich, Domäne} | {SemBereich → Domäne} >,
< ENTHALTEN | {Teil_Symbol, Teil_Attribut, Ober_Symbol, Ober_Attribut} | Ø>

|

{ $\pi_{Teil\_Symbol,\ Teil\_Attribut}$ (ENTHALTEN) ⊂ $\pi_{Symbol,\ Attribut}$ (RELATION),
$\pi_{Ober\_Symbol,\ Ober\_Attribut}$ (ENTHALTEN)⊂ $\pi_{Symbol,\ Attribut}$ (RELATION) }

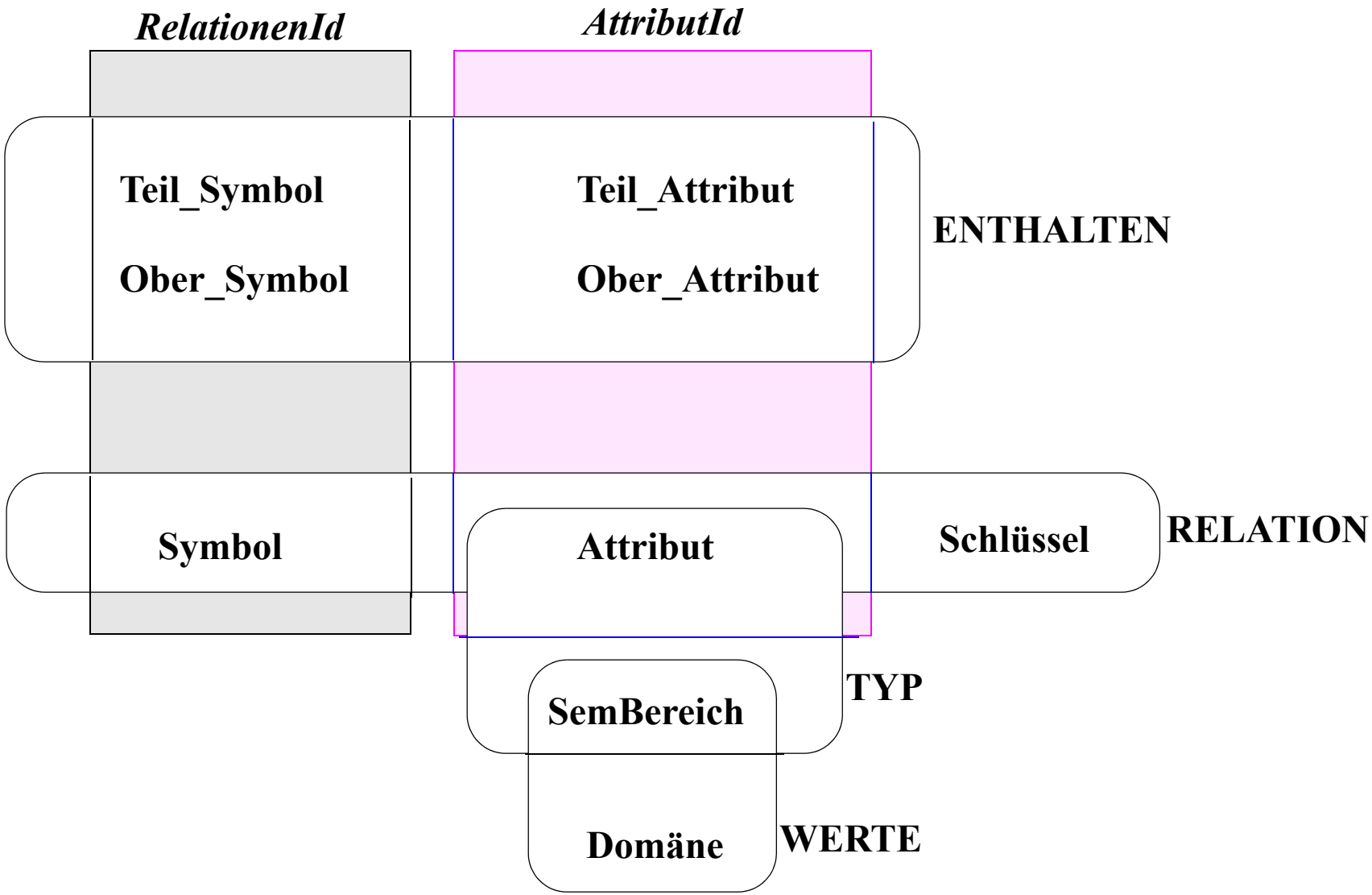
|

a (Symbol) := **a** (Teil_Symbol) := **a** (Ober_Symbol) := RelationenId,
a (Attribut) := **a** (Teil_Attribut) := **a** (Ober_Attribut) := AttributId,
a (SemBereich) := SemBereich,
a (Domäne) := Domäne,
a (Schlüssel) := SchlüsselInd

>

b (RelationId) := **b** (AttributId) := **b** (SemBereich) := Σ^* ⊂ **C**
b (Domäne) := {boolean, integer, cardinal, real, string, ...} ⊂ **C**
b (SchlüsselInd) := {false, true} ⊂ **C**

Hypergraph zum relationalen Datenbankschema für Schemas ("Metaschema")



Beispiel: Instanz zum relationalen Datenbankschema für Schemas

RELATION	Symbol	Attribut	Schlüssel
	RelationenId	AttributId	SchlüsselInd
	PERSON	Name	true
	PERSON	Geschlecht	false
	BEH	Patient	true
	BEH	Arzt	true
	BEH	ArtPro	true
	ELT	Name	true
	ELT	Eltern	true

TYP	Attribut	SemBereich
	AttributId	SemBereich
	Name	Mensch
	Patient	Mensch
	Arzt	Mensch
	Eltern	Mensch
	Gechlecht	Geschlecht
	ArtPro	ArtPro

WERTE	SemBereich	Domäne
	SemBereich	Domäne
	Mensch	string
	Geschlecht	weibmann
	ArtPro	ubhlr

ENTHALTEN	Teil_Symbol	Teil_Attribut	Ober_Symbol	Ober_Attribut
	RelationenId	AttributId	RelationenId	AttributId
	BEH	Patient	PERSON	Name
	BEH	Arzt	PERSON	Name
	ELT	Name	PERSON	Name
	ELT	Eltern	PERSON	Name

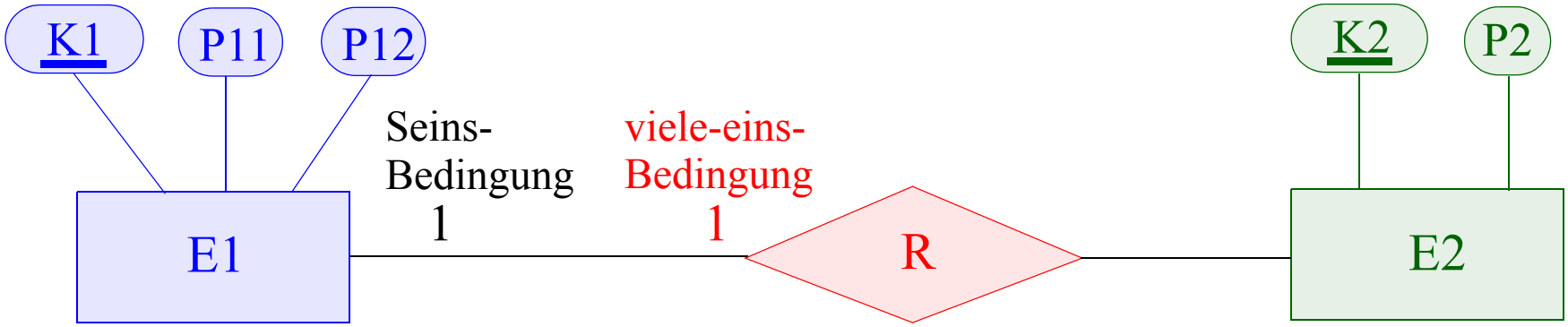
2.3 relationale Strukturen: Pragmatik

Die Pragmatik wird in den Übungsaufgaben anhand von Beispielen erarbeitet.

Anhand der dabei gewonnenen Erfahrungen
kann man grobe Faustregeln aufstellen,
wie man die Bestandteile einer Modellierung (Konstrukte eines ER-Diagramms)
in die Komponenten eines relationalen Datenbankschemas überträgt.

eine Faustregel für “von Modellierung zu Formalisierung”

- Modellierung als ER-Diagramm



- Formalisierung durch relationales Datenbankschema

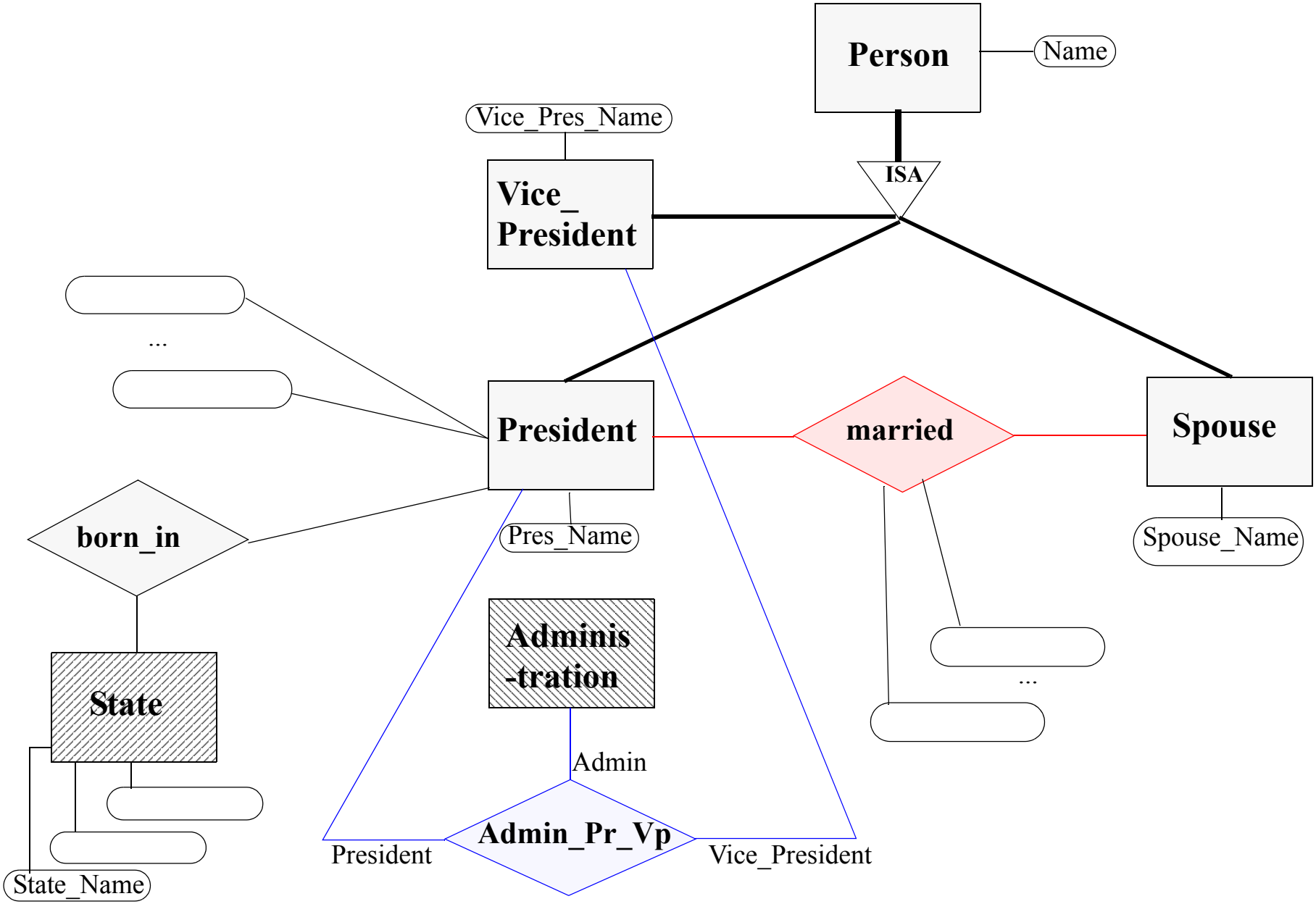
$$< \text{E1} \mid \{K1, P11, P12\} \mid \{K1 \rightarrow P11,P12\} >$$
$$< \text{E2} \mid \{K2, P2\} \mid \{K2 \rightarrow P2\} >$$
$$< \text{R} \mid \{K1, K2\} \mid \{K1 \rightarrow K2\} >$$

$$\pi_{K1}(R) \subset \pi_{K1}(E1), \quad \pi_{K2}(R) \subset \pi_{K2}(E2)$$

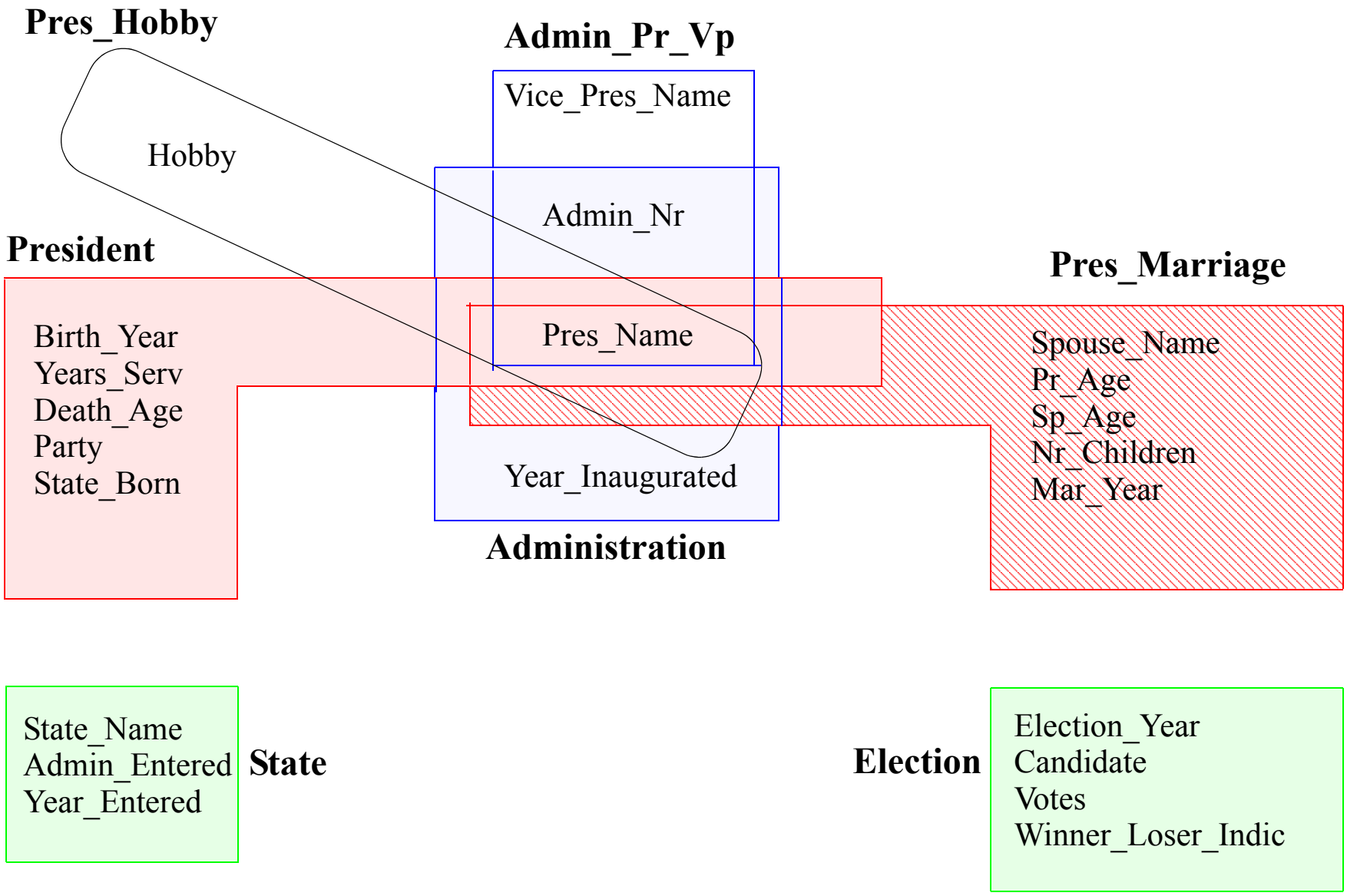
$$\pi_{K1}(R) \supset \pi_{K1}(E1)$$

2.4 relationale Strukturen: Praxis (Oracle-SQL)

ER-Diagramm für Präsidenten-Datenbank (Ausschnitt)



Hypergraph zum Oracle-Schema für Präsidenten-Datenbank



Oracle-Schema für Präsidenten-Datenbank

```
create table State (
   State_Name          varchar2(17)      primary key,
   Admin_Entered       number(3),
   Year_Entered        number(4)         not null
);

create table President (
   Pres_Name           varchar2(17)      primary key
                                         check (Pres_Name LIKE '% _'),
   Birth_Year          number(4)         not null,
   Years_Serv          number(2)         not null,
   Death_Age           number(3),
   Party               varchar2(12)      not null,
   State_Born          varchar2(17)      not null
);
```

Oracle-Schema für Präsidenten-Datenbank (Fortsetzung)

```
create table Pres_Hobby  (
   Pres_Name          varchar2(17)        references President
                                          on delete cascade,
   Hobby              varchar2(18),
                                          primary key ( Pres_Name, Hobby )
);


create table Administration (
   Admin_Nr           number(3),
   Pres_Name          varchar2(17)        references President,
   Year_Inaugurated   number(4)           not null,
                                          primary key ( Admin_Nr, Pres_Name ),
                                          check (Year_Inaugurated
                                                BETWEEN 1785+4*Admin_Nr
                                                AND 1789+4*Admin_Nr)
 );
```

Oracle-Schema für Präsidenten-Datenbank (Fortsetzung)

```
create table Admin_Pr_Vp (
   Admin_Nr          number(3),
   Pres_Name         varchar2(17),
   Vice_Pres_Name    varchar2(17),
                     foreign key ( Admin_Nr, Pres_Name ) references Administration,
                     primary key ( Admin_Nr, Pres_Name, Vice_Pres_Name ),
                     check (Pres_Name != Vice_Pres_Name)
 );
```

Oracle-Schema für Präsidenten-Datenbank (Fortsetzung)

```
create table Pres_Marriage (
   Pres_Name            varchar2(17)     references President,
   Spouse_Name          varchar2(17),
   Pr_Age               number(3)        not null,
   Sp_Age               number(3)        not null,
   Nr_Children          number(2)        not null,
   Mar_Year             number(4)        not null,
                                         primary key ( Pres_Name, Spouse_Name )
);
```

```
create table Election  (
   Election_Year        number(4),
   Candidate            varchar2(17),
   Votes                number(3),
   Winner_Loser_Indic char(1)            not null,
                                         primary key ( Election_Year, Candidate )
);
```

einige syntaktische Regeln für Oracle-SQL-Schemavereinbarungen

Relationensymbole:

create table <table_identifier> (...) vereinbart *table_identifier* als Relationensymbol

Attribute:

<attribute_identifier> <domain> ... vereinbart *attribute_identifier* als Attribut (Stellenbezeichner) mit Domäne (Type) *domain*

semantische Bedingungen:

not null	verlangt einen definierten Wert, in Schlüsselvereinbarungen eingeschlossen
... primary key	vereinbart entsprechendes Attribut als **(Primär-) Schlüssel**, d.h. insbesondere die **funktionale Abhängigkeit** “entsprechendes Attribut” → “alle Attribute”
primary key (<attribute_list>)	vereinbart die aufgelisteten Attribute als (Primär-) Schlüssel, d.h. insbesondere die **funktionale Abhängigkeit** “aufgelistete Attribute” → “alle Attribute”
check (<tuple_condition>)	verlangt bei Einfügungen und Änderungen die genannte Bedingung

... references <table_identifier>

bezieht sich auf den
Schlüssel des genannten Relationensymbols,

vereinbart eine **Enthaltenseinsabhängigkeit**
zwischen dem entsprechenden Attribut und
dem Schlüssel des genannten Relationensymbols

... on delete cascade

vereinbart eine “kaskadierende Folgeaktion”
(z.B. Entfernen weiterer Tupel)
bei Verletzung der Enthaltenseinsabhängigkeit
durch Entfernen eines Tupels
bezüglich des genannten Relationensymbols

foreign key (<attribute_list>) references <table_identifier>

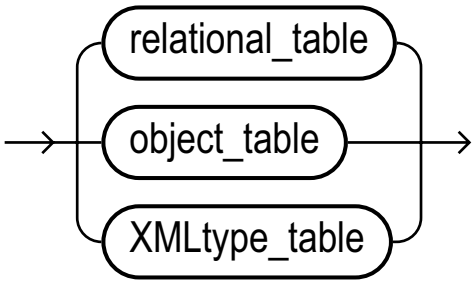
bezieht sich auf den
Schlüssel des genannten Relationensymbols,

vereinbart eine **Enthaltenseinsabhängigkeit**
zwischen den aufgelisteten Attributen und
dem Schlüssel des genannten Relationensymbols

Oracle-SQL Syntax für “CREATE TABLE”

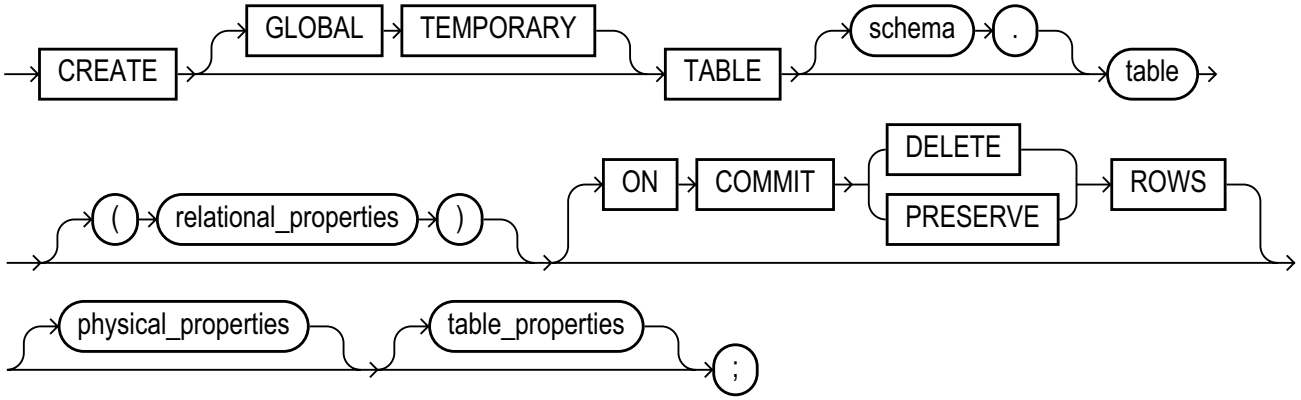
Auszug aus:
Oracle 9*i* - SQL Reference, Release 2 (9.2), March 2002, Part No.A96540-01,
pp. 15-7 bis 15-79

create_table::=

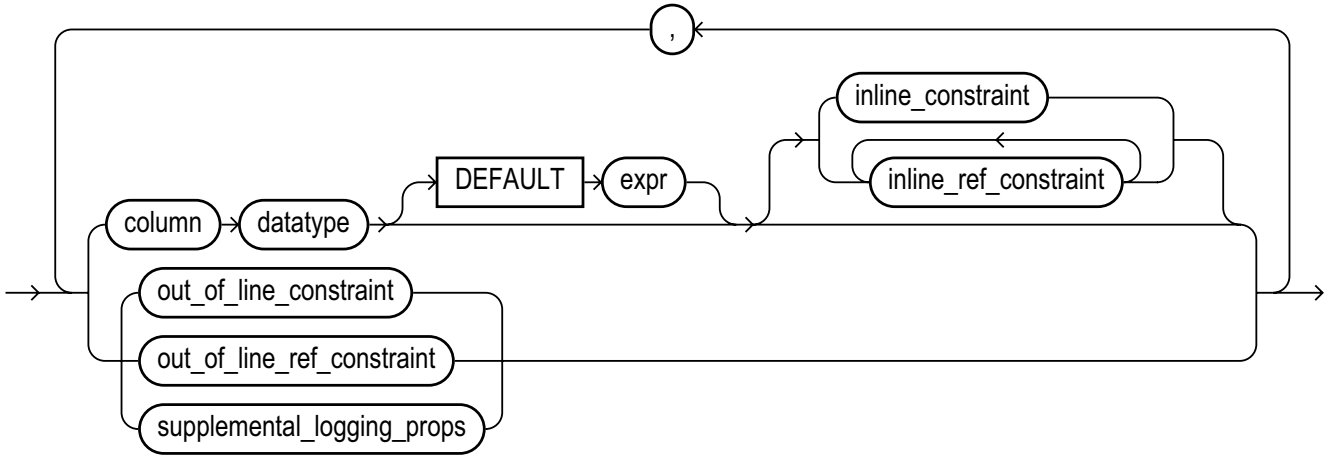


Im Folgenden wird nur der erste Fall, relationale Tabellen, sehr kurz behandelt.

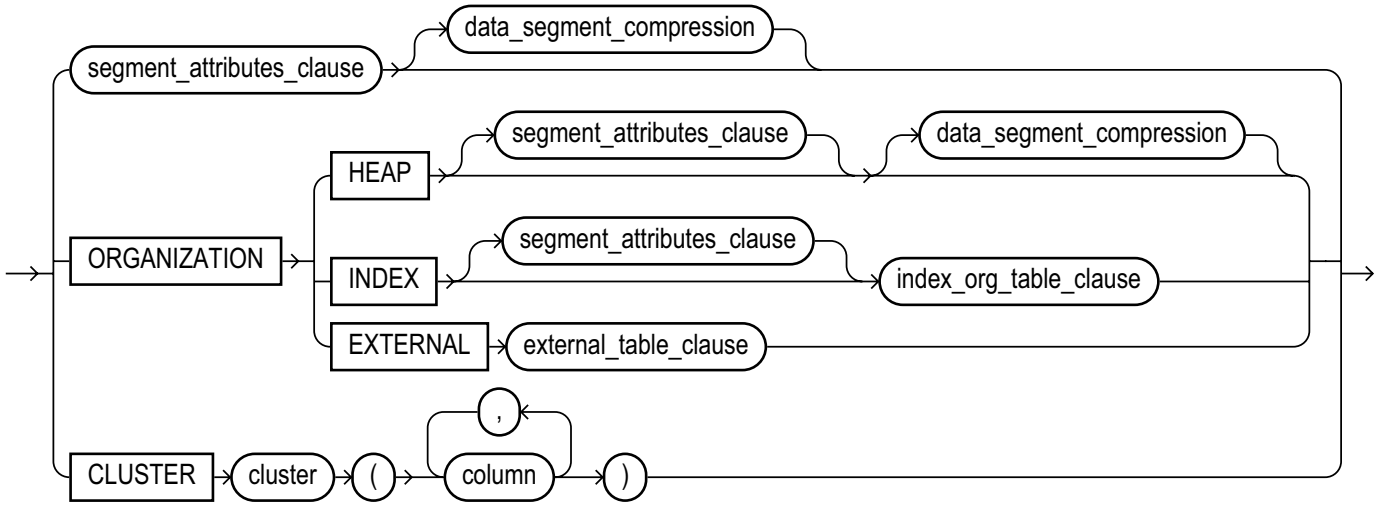
relational_table::=



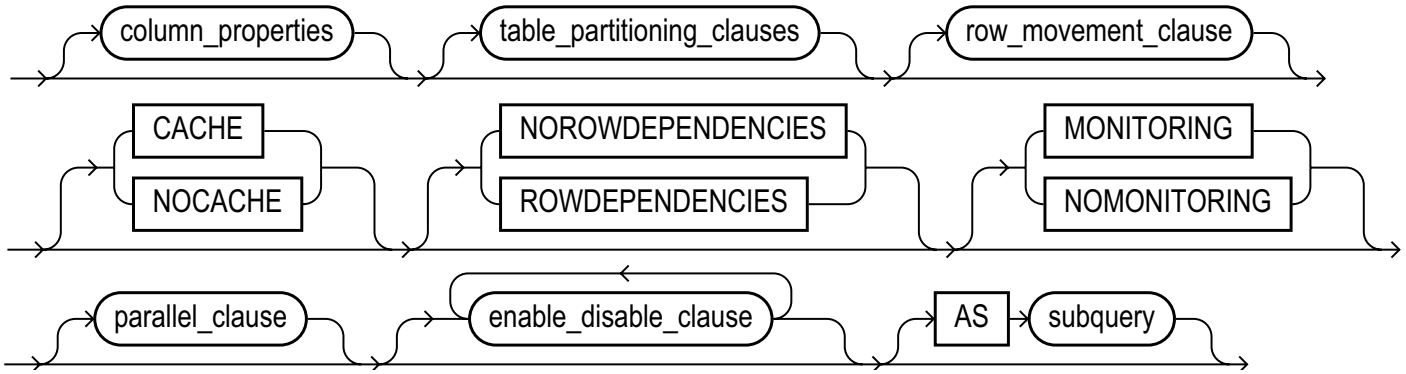
relational_properties::=



physical_properties::=



table_properties::=



3 Relationales Datenmodell: Operationen

relationales Datenmodell

- **Strukturen**

  Schema:
  - Relationenschemas
  - (globale) semantische Bedingungen
  - semantische Bereichsnamen

  Instanzen:
  - Universum
  - Relationen

- **Operationen**

  Änderungen:

  - elementar: `insert` $R_i(c_1, \ldots, c_k)$
    `delete` $R_i(c_1, \ldots, c_k)$

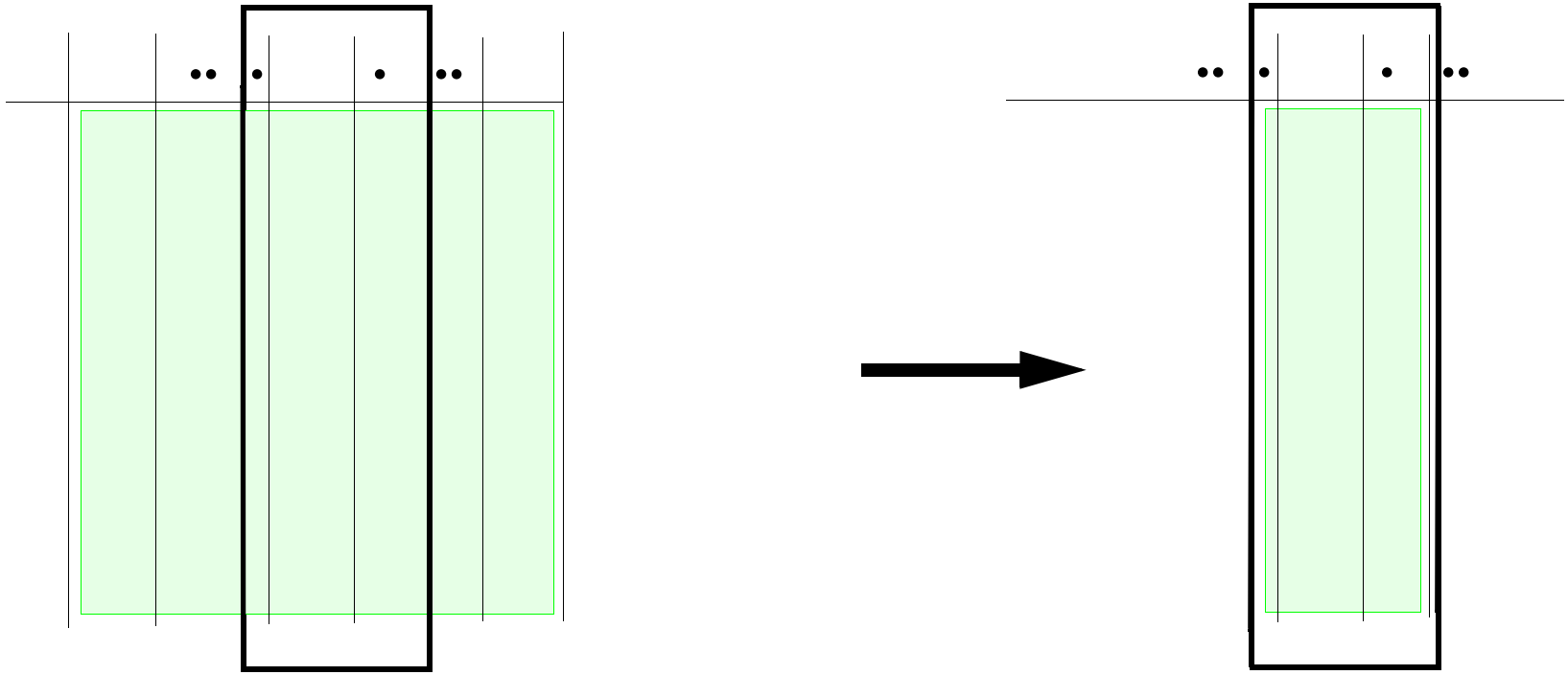
  - als Transaktion: `begin` op_1 `;` `...` `;` op_m `end`

  - deklarativ: Parameter mengenorientiert,
    durch Anfrage bestimmt

  Anfragen

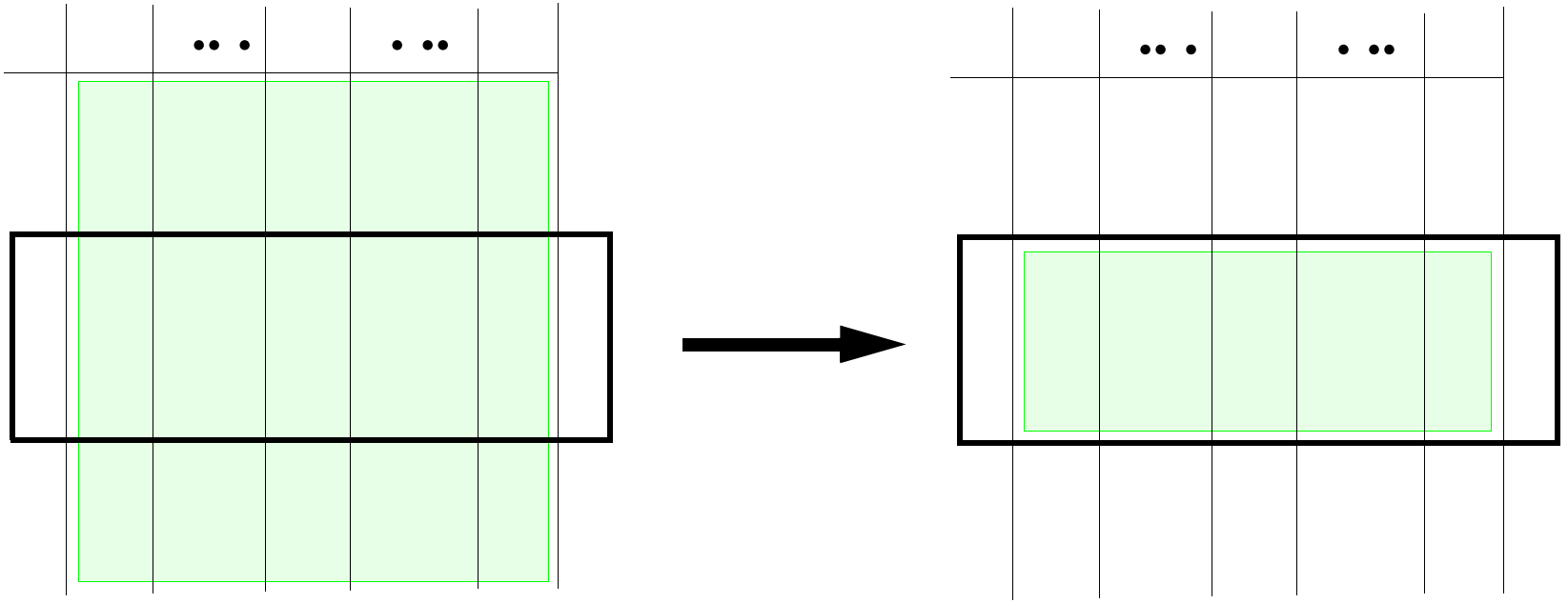
3.1 relationale Anfrageoperationen: Theorie

Standard-Operationen auf einzelner Relation (Tabelle): anschaulich



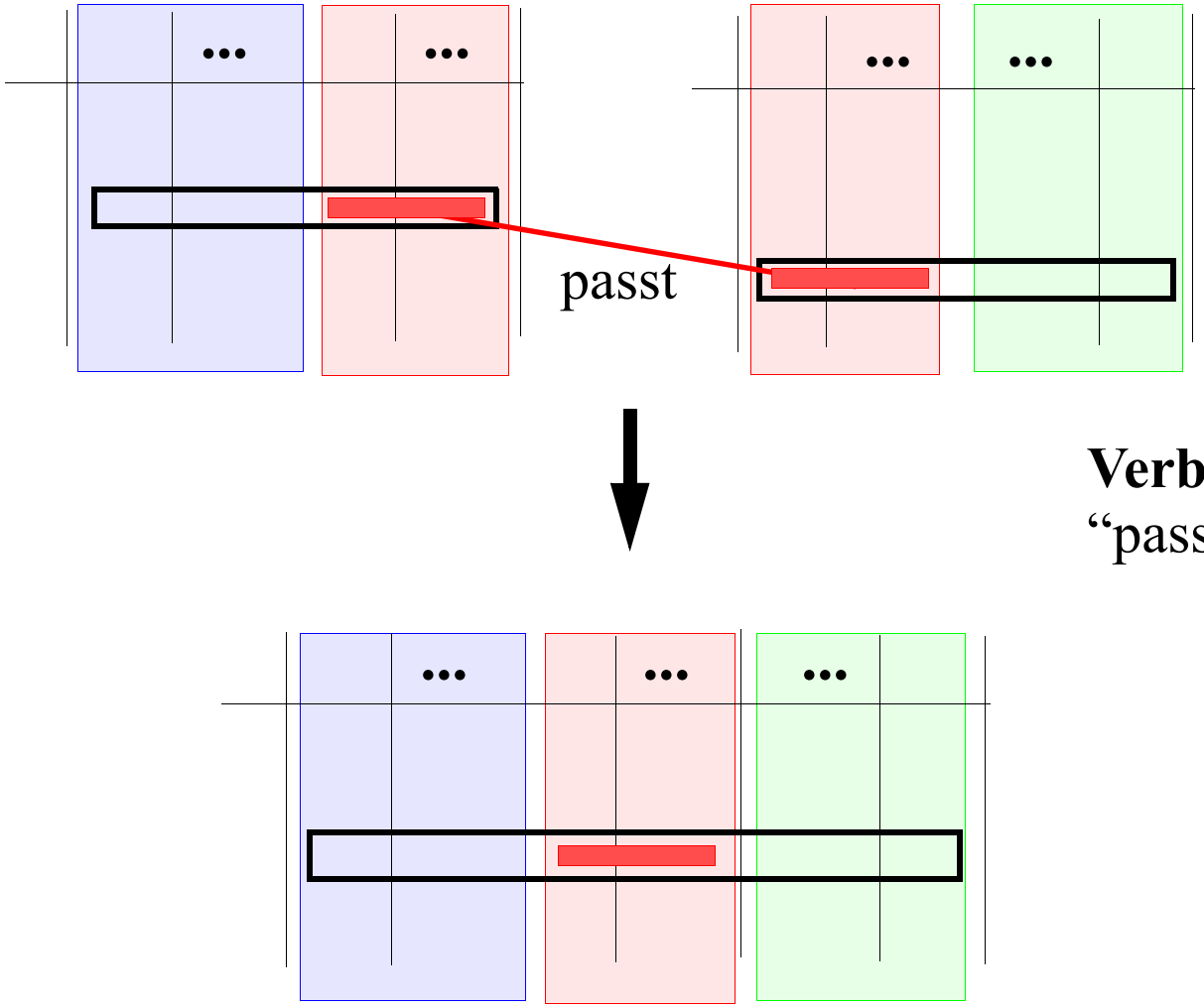
Projektion:
Spalten auswählen,
mit Duplikat-Entfernung
(umbenennen, verdoppeln)

Standard-Operationen auf einzelner Relation (Tabelle): anschaulich



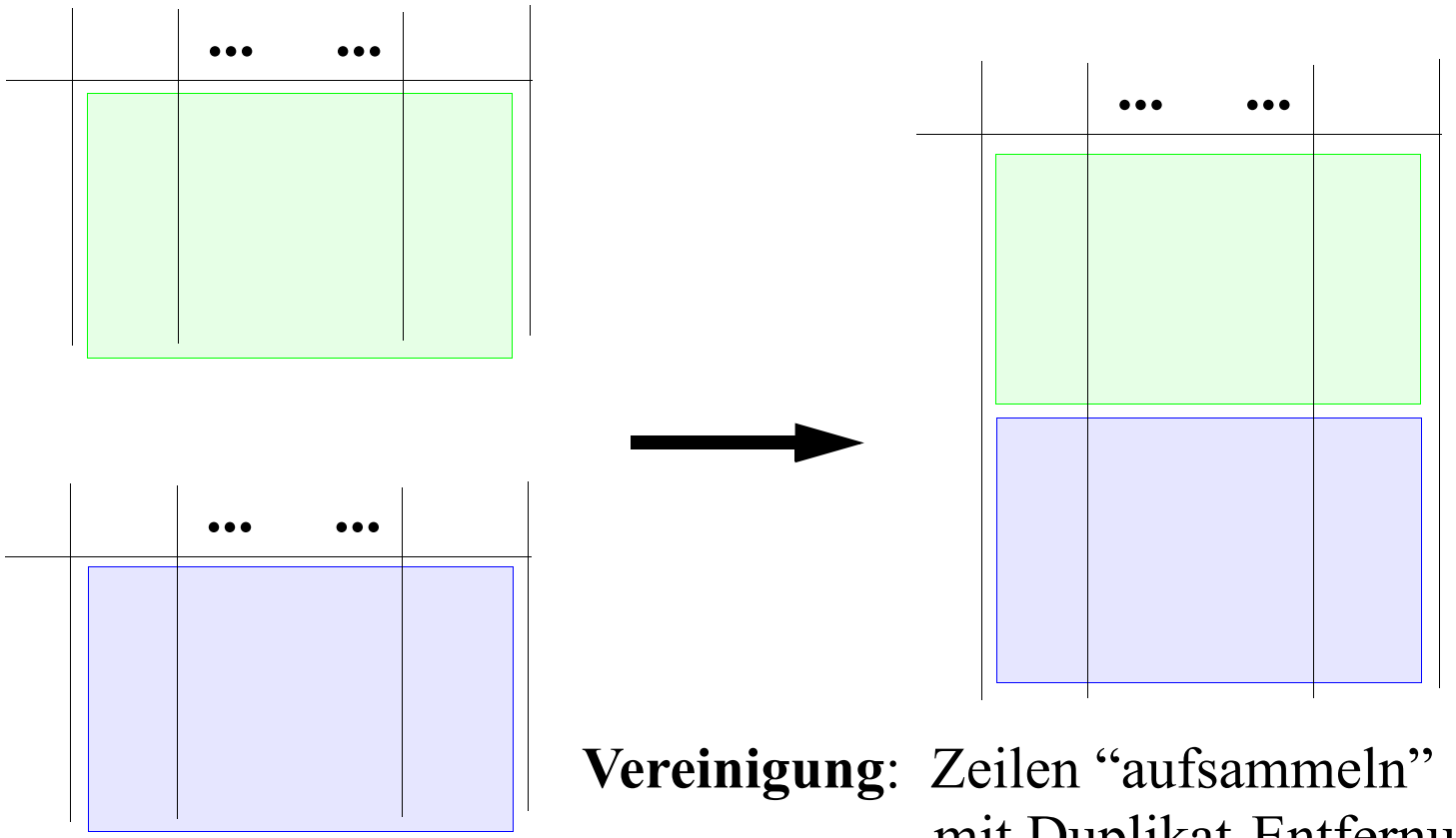
Selektion / Vergleich:
Zeilen auswählen
(nach Auswahl-Werten /
innerer Struktur)

Standard-Operationen auf zwei Relationen (Tabellen): anschaulich



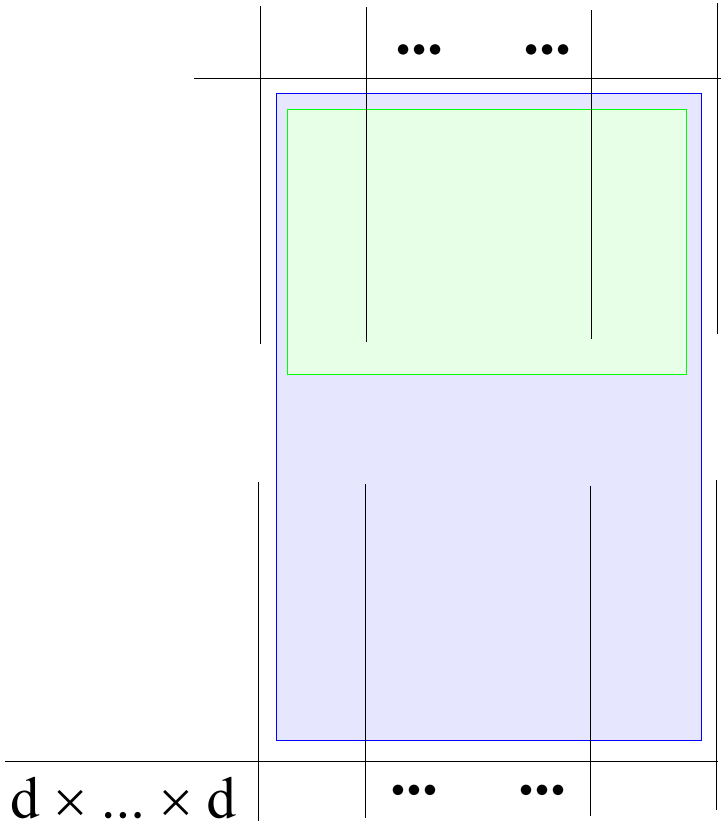
Verbund:
“passende” Zeilen verbinden

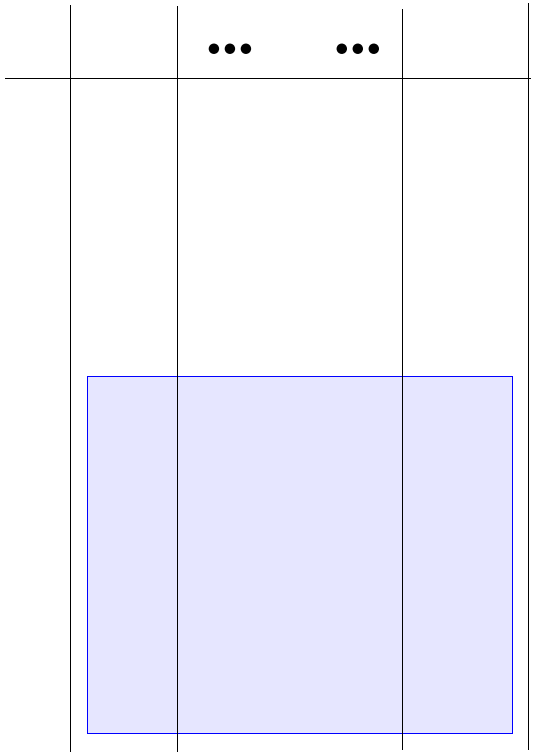
Standard-Operationen auf zwei Relationen (Tabellen): anschaulich



Vereinigung: Zeilen “aufsammeln”
mit Duplikat-Entfernung
(ggf. “auffüllen”)

Standard-Operationen auf zwei Relationen (Tabellen): anschaulich





Komplement:
bezüglich des
kartesischen Produkts
eines Universums *d*

Operationen der relationalen Algebra: formal

natürlicher Verbund

Selektion

Projektion

Vergleich

Vereinigung

Komplement

Differenz

Division

natürlicher Verbund (natural join): Definition

- **zweistellig**

$$r \bowtie s := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \rightarrow \mathbf{C},$$
$$\mu\lceil \mathrm{dom}\, r \in r \quad \textbf{und} \quad \mu\lceil \mathrm{dom}\, s \in s \}$$

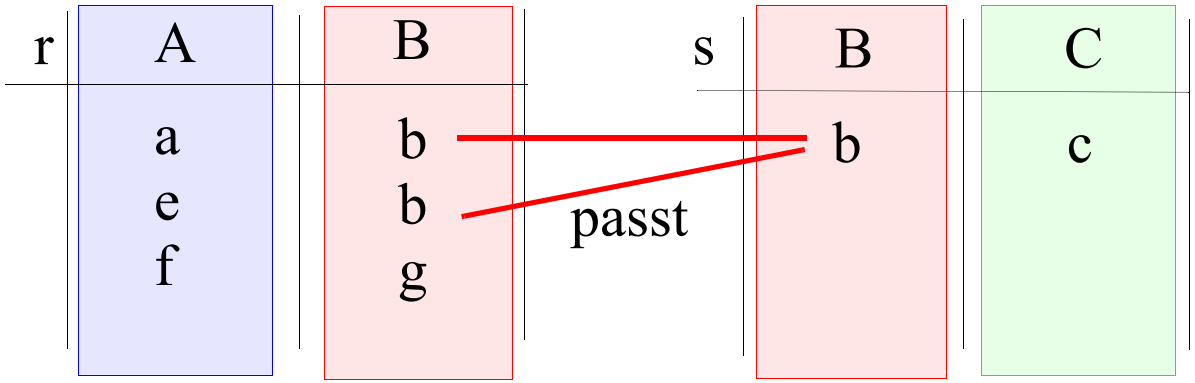
$$= \{ \mu \mid \text{es gibt } \alpha \in r, \text{ es gibt } \beta \in s :$$
$$\alpha(A) = \beta(A) \text{ für alle } A \in \mathrm{dom}\, r \cap \mathrm{dom}\, s;$$
$$\mu = \alpha \cup \beta \}$$

- **allgemein**

$$\bowtie_{i=1,\ldots,k} r_i := \{ \mu \mid \mu : \bigcup_{i=1,\ldots,k} \mathrm{dom}\, r_i \rightarrow \mathbf{C},$$
$$\mu\lceil \mathrm{dom}\, r_i \in r_i \quad \textbf{für } \boldsymbol{i = 1,\ldots,k} \}$$

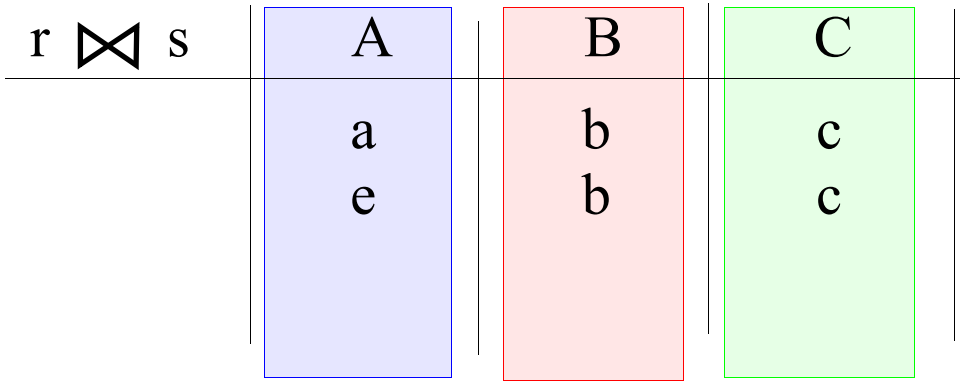
natürlicher Verbund: Beispiel

$$r \bowtie s := \{ \quad \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \rightarrow \mathbf{C},\ \mu\lceil \mathrm{dom}\, r \in r \ \textbf{und}\ \mu\lceil \mathrm{dom}\, s \in s \}$$



$\mathrm{dom}\, r \cup \mathrm{dom}\, s = A\ B\ C$

$\mathrm{dom}\, r \cap \mathrm{dom}\, s = B$



einfacher Algorithmus für natürlichen Verbund: nested loop

$r \bowtie s \;=$

$\{ \; \mu \;\; | \qquad$ Ausgabe

$\quad$ es gibt $\alpha \in r$, $\qquad$ äußerer loop: durchsuche r

$\quad\quad$ es gibt $\beta \in s$: $\qquad$ innerer loop: durchsuche s

$\quad\quad \alpha(A) = \beta(A)$ für alle $A \in \text{dom } r \cap \text{dom } s$; $\qquad$ α und β passend ?

$\quad\quad \mu = \alpha \cup \beta \qquad$ falls zutreffend: Ausgabe-Tupel konstruieren

$\}$

Laufzeit: $O (\| r \| \cdot \| s \|)$

Spezialfälle des natürlichen Verbunds

- ***kartesisches Produkt***

  falls dom $r \cap$ dom $s = \emptyset$: $\quad r \bowtie s = r \times s$

- ***Durchschnitt***

  falls dom $r =$ dom s: $\quad r \bowtie s = r \cap s$

natürlicher Verbund ist **doppelgesichtig**:

- kann ***aggregierend*** wirken,
  liefert dann (quadratisch) *vergrößerte* Ergebnisse
- kann ***aussondernd*** wirken,
  liefert dann *verkleinerte* Ergebnisse
- beide Wirkungen können sich ***überlagern***

A=c-Selektion (A=c-selection) als abgeleitete Operation: Definition

für $A \in \mathrm{dom}\, r$ und $c \in \mathbf{C}$:

$$\sigma_{A=c}(r) := r \bowtie \{(A,c)\}$$

$$= \{ \mu \mid \mu : \mathrm{dom}\, r \cup \{A\} \rightarrow \mathbf{C}, \; \mu\lceil \mathrm{dom}\, r \in r \text{ und } \mu\lceil \{A\} \in \{(A,c)\} \}$$

$$= \{ \mu \mid \mu \in r \text{ und } \mu(A) = c \}$$

A=c-Selektion: Beispiel

$$\sigma_{A=c}(r) := \{ \mu \mid \mu \in r \text{ und } \mu(A) = c \}$$

r	A	B
	a	b
	e	b
	f	g

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

A	B
a	b
e	b

algebraische Eigenschaften des natürlichen Verbunds

		$r \bowtie s$	$=$	$s \bowtie r$	kommutativ
		$(r \bowtie s) \bowtie t$	$=$	$r \bowtie (s \bowtie t)$	assoziativ
$r \subset s$	$\Rightarrow$	$r \bowtie t$	$\subset$	$s \bowtie t$	monoton
$r \subset s$	$\Rightarrow$	$r \bowtie s$	$=$	r	absorbtiv
		$r \bowtie r$	$=$	r	idempotent
		$r \bowtie \varnothing$	$=$	$\varnothing$	Nullelement
$A \in \mathrm{dom}\, r$	$\Rightarrow$	$\sigma_{A=c}(r \bowtie s)$	$=$	$\sigma_{A=c}(r) \bowtie s$	σ–distributiv
$A \in \mathrm{dom}\, r \cap \mathrm{dom}\, s$	$\Rightarrow$	$\sigma_{A=c}(r \bowtie s)$	$=$	$\sigma_{A=c}(r) \bowtie \sigma_{A=c}(s)$	σ–distributiv

Projektion und q-Projektion: Definition

Projektion (projection) der Relation r auf Attributmenge X
für $X \cap \text{dom}\, r \neq \emptyset$:

$$\pi_X(r) := \{ \nu\lceil X \mid \nu \in r \} \quad \text{mit}$$

$$\text{dom}\, \pi_X(r) = X \cap \text{dom}\, r$$

q-Projektion der Relation r vermöge Attribut-Transformation q
für $q : X \to Y, \quad Y \subset \text{dom}\, r, \quad X \neq \emptyset$:

$$\pi_q(r) := \{ \mu \mid \mu : X \to \mathbf{C}, \text{ es gibt } \nu \in r \text{ mit } \mu = \nu \circ q \}$$

wobei: Funktionengleichheit $\mu = \nu \circ q$ gdw für alle $A \in X$: $\mu(A) = \nu \circ q\,(A)$
$= \nu\,(q\,(A))$

Projektion: Beispiel

$$\pi_X(r) := \{ \nu\lceil X \mid \nu \in r \} \quad \text{für} \quad X := \{ \text{B} \}$$

r	A	B
	a	b
	e	b
	f	g

$\pi_B(r)$	B
	b
	g

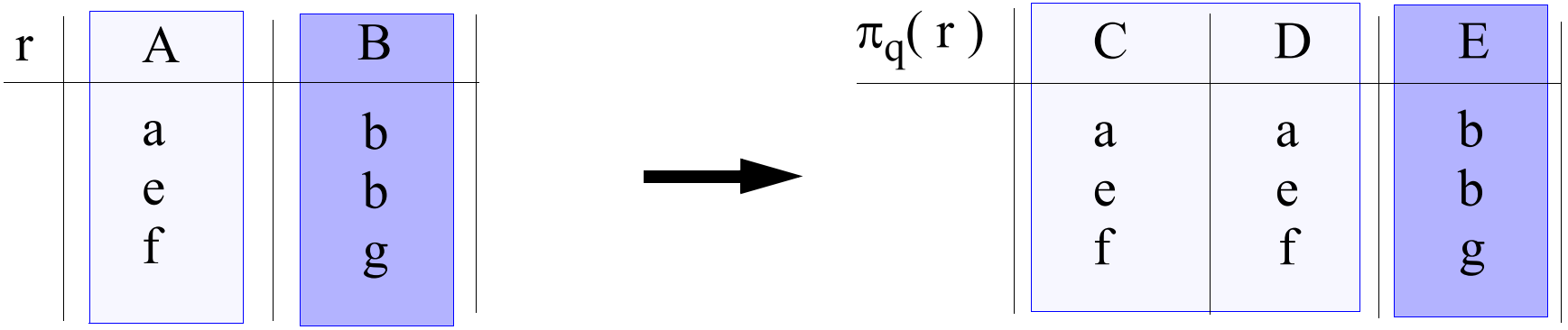
mit Duplikat-Entfernung

q-Projektion: Beispiel

$$\pi_q(r) := \{ \mu \mid \mu : X \to \mathbf{C}, \text{ es gibt } \nu \in r \text{ mit } \mu = \nu \circ q \}$$

für $q(\mathrm{C}) := \mathrm{A}$
$q(\mathrm{D}) := \mathrm{A}$
$q(\mathrm{E}) := \mathrm{B}$

Umbenennung: C ist "neuer Name" für A
Verdoppelung: D ebenfalls
Umbenennung: E ist "neuer Name" für B



Projektion und q-Projektion

sei $q : X \to X$
$q(A) := A$ für $A \in X \subset \text{dom } r$

dann gilt:

$$\pi_q(r) = \{ \mu \mid \mu : X \to \mathbf{C}, \text{ es gibt } \nu \in r \text{ mit } \mu = \nu \circ q \} \quad \text{Def } \pi_q$$

$$= \{ \mu \mid \mu : X \to \mathbf{C}, \text{ es gibt } \nu \in r \text{ mit } \mu = \nu \lceil X \} \quad \text{Def } q$$

$$= \{ \nu \lceil X \mid \nu \in r \} \quad \text{Def } \lceil$$

$$= \pi_X(r) \quad \text{Def } \pi_X$$

einfacher Algorithmus für Projektion: sequential scan

$\pi_X(r) =$

$\{ \quad \nu\restriction X \quad |$ Ausgabe-Tupel konstruieren und Duplikate entfernen, z.B.:

falls das Tupel $\nu\restriction X$ noch nicht im Zwischenergebnis ist,
so füge es ein;
Elementtest-Und-Einfüge-Operation
mit geeigneter Datenstruktur (**sortierte Liste**, **B*-Baum**, ...)

$\nu \in r \quad \}$ durchsuche r

Laufzeit: $O(\| r \| \cdot \log \| r \|)$

Projektion und natürlicher Verbund (als eine Art “Quasi-Inverse”)

eine durch überdeckende Projektionen zerlegte Relation r
ist im natürlichen Verbund der Projektionen enthalten:

$$\text{für } \bigcup_{i=1,\ldots,k} X_i = \text{dom } r: \quad r \subset \bowtie_{i=1,\ldots,k} \pi_{X_i}(r)$$

Gleichheit kann durch eine *Verbundabhängigkeit* erzwungen werden.

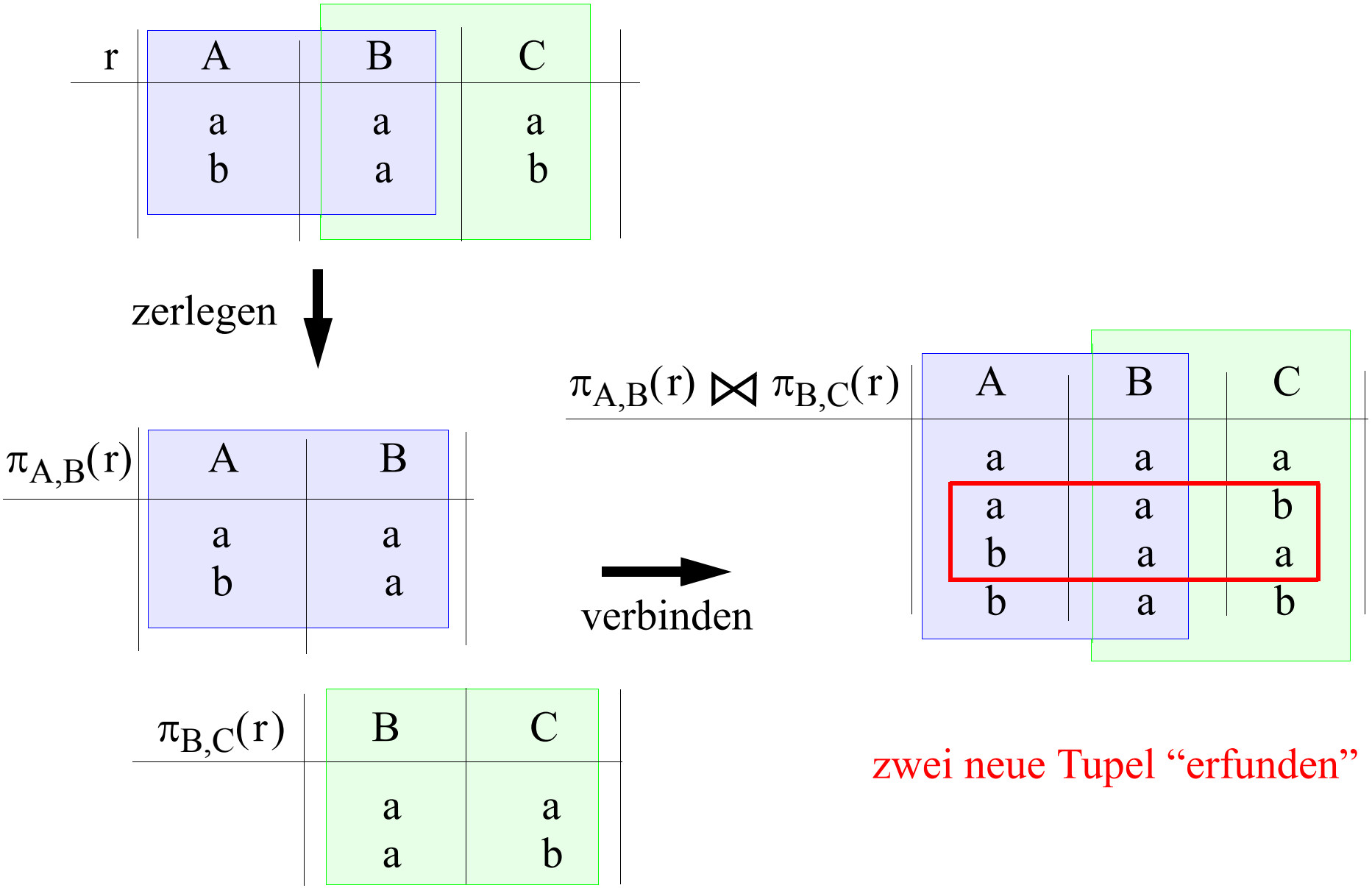
natürlicher Verbund und Projektion (als eine Art “Quasi-Inverse”)

eine an einem natürlichen Verbund teilnehmende Relation r_j
enthält die entsprechende Projektion des Verbundes:

$$\pi_{\text{dom } r_j} \left(\bowtie_{i=1,\ldots,k} r_i \right) \subset r_j$$

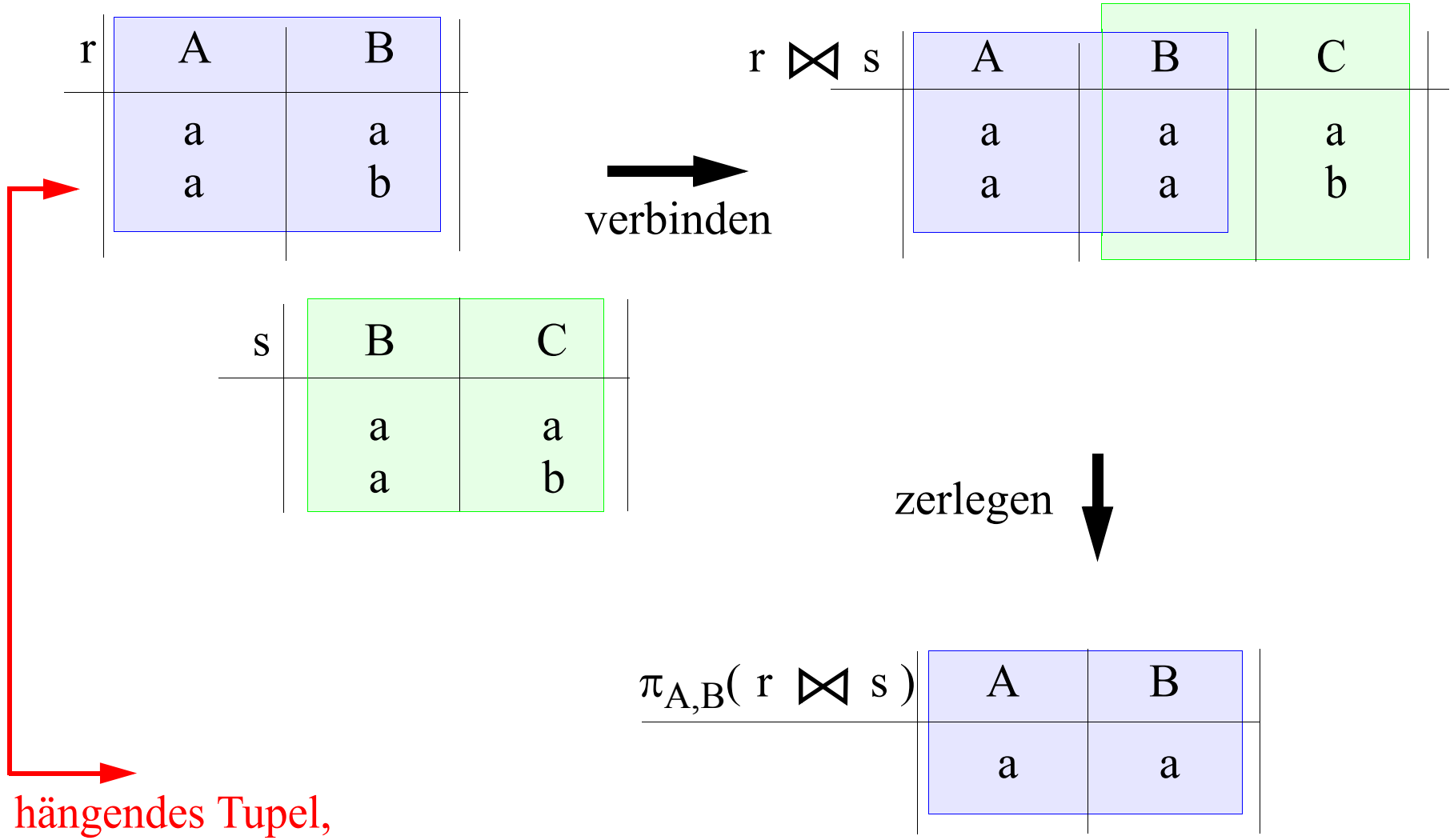
Gleichheit kann durch *Enthaltenseinsabhängigkeiten* erzwungen werden.

eine “verlustbehaftete” Zerlegung (lossy join)



zwei neue Tupel “erfunden”

ein hängendes Tupel (dangling tuple)



Teilverbund (semijoin): Definition

$$r \ltimes s := \pi_{\text{dom } r}(r \bowtie s)$$

$$= r \bowtie \pi_{\text{dom } r \cap \text{dom } s}(s)$$

Projektion und andere Operationen (optimierende Umformungen)

falls $\text{dom}\, r \cap \text{dom}\, s \subset X$: $\quad \pi_X(r \bowtie s) = \pi_X(r) \bowtie \pi_X(s)$

falls $A \in X \cap \text{dom}\, r$: $\quad \pi_X(\sigma_{A=c}(r)) = \sigma_{A=c}(\pi_X(r))$

für $X \cap Y \cap \text{dom}\, r \neq \emptyset$: $\quad \pi_X(\pi_Y(r)) = \pi_{X \cap Y}(r)$

speziell für $X \subset Y \subset \text{dom}\, r$: $\quad \pi_X(\pi_Y(r)) = \pi_X(r)$

Vereinigung: Definition

$$+(d, r, s) := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \to d, \; \mu\lceil \mathrm{dom}\, r \in r \quad \textbf{oder} \quad \mu\lceil \mathrm{dom}\, s \in s \}$$

Vereinigung: Beispiel

$$+(d,r,s) := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \rightarrow d, \; \mu \lceil \mathrm{dom}\, r \in r \quad \textbf{oder} \quad \mu \lceil \mathrm{dom}\, s \in s \}$$

d = { e , f , g }

r	A	B
	e	f
	e	g
	f	f

s	A	C
	e	f

aufsammeln und auffüllen

+(d,r,s)	A	B	C
	e	f	e
	e	f	f
	e	f	g
	e	g	e
	e	g	f
	e	g	g
	f	f	e
	f	f	f
	f	f	g
	e	e	f
	e	f	f
	e	g	f

Duplikate

grundlegende Eigenschaften der Vereinigung

1. $+(d, r, s) = r \cup s$ für dom r = dom s (*vereinigungsverträglich*)

2. $+(d, r, s) = +(d, s, r)$

3. $(r \cup s) \bowtie t = (r \bowtie t) \cup (s \bowtie t)$

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

4. $\pi_X(+(d, r, s)) = +(d, \pi_X(r), \pi_X(s))$

   $\pi_X(r \cup s) = \pi_X(r) \cup \pi_X(s)$ speziell für dom r = dom s

A=B-Vergleich und A≠B-Vergleich: Definition

$$\sigma_{A=B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) = \mu(B) \}$$

$$\sigma_{A\neq B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) \neq \mu(B) \}$$

A=B-Vergleich und A≠B-Vergleich: grundlegende Eigenschaften

1. $\sigma_{A=B}(r) \quad \cup \quad \sigma_{A\neq B}(r) \quad = \quad r$

2. $\sigma_{A=B}(r) \quad \cap \quad \sigma_{A\neq B}(r) \quad = \quad \emptyset$

3. für $X = \{A_1,\ldots,A_k\}$,
   $Y = \{B_1,\ldots,B_k\}$ mit bekannter Reihenfolge der Aufzählung,
   $X \cap Y = \emptyset$,
   $X \cup Y \subset \text{dom } r$:

   $\sigma_{X=Y}(r) := \sigma_{A_1 = B_1}(\sigma_{A_2 = B_2}(\ldots \sigma_{A_k = B_k}(r) \ldots))$

Komplement: Definition

$$\gamma(d, r) := \{ \mu \mid \mu : \mathrm{dom}\, r \to d \} \setminus r$$

Komplement: Beispiel

$d = \{ e , f , g \}$

r	A	B
	e	f
	e	g
	f	f

d × d	A	B
	e	e
	e	f
	e	g
	f	e
	f	f
	f	g
	g	e
	g	f
	g	g

γ (d,r)	A	B
	e	e
	e	f
	e	g
	f	e
	f	f
	f	g
	g	e
	g	f
	g	g

Differenz: Definition

für $\text{dom}\, r \cap \text{dom}\, s \neq \emptyset$:

$$-(d, r, s) := \{ \mu \mid \mu \in r \quad \textbf{und} \quad \mu\lceil \text{dom}\, r \cap \text{dom}\, s \notin \pi_{\text{dom}\, r \cap \text{dom}\, s}(s) \}$$

$$= r \bowtie \gamma(d, \pi_{\text{dom}\, r \cap \text{dom}\, s}(s))$$

Differenz: Beispiel direkt

$$-(d,r,s) := \{ \mu \mid \mu \in r \quad \textbf{und} \quad \mu\lceil \mathrm{dom}\, r \cap \mathrm{dom}\, s \notin \pi_{\mathrm{dom}\, r \cap \mathrm{dom}\, s}(s) \}$$

r	A	B
	e	f
	e	g
	f	f

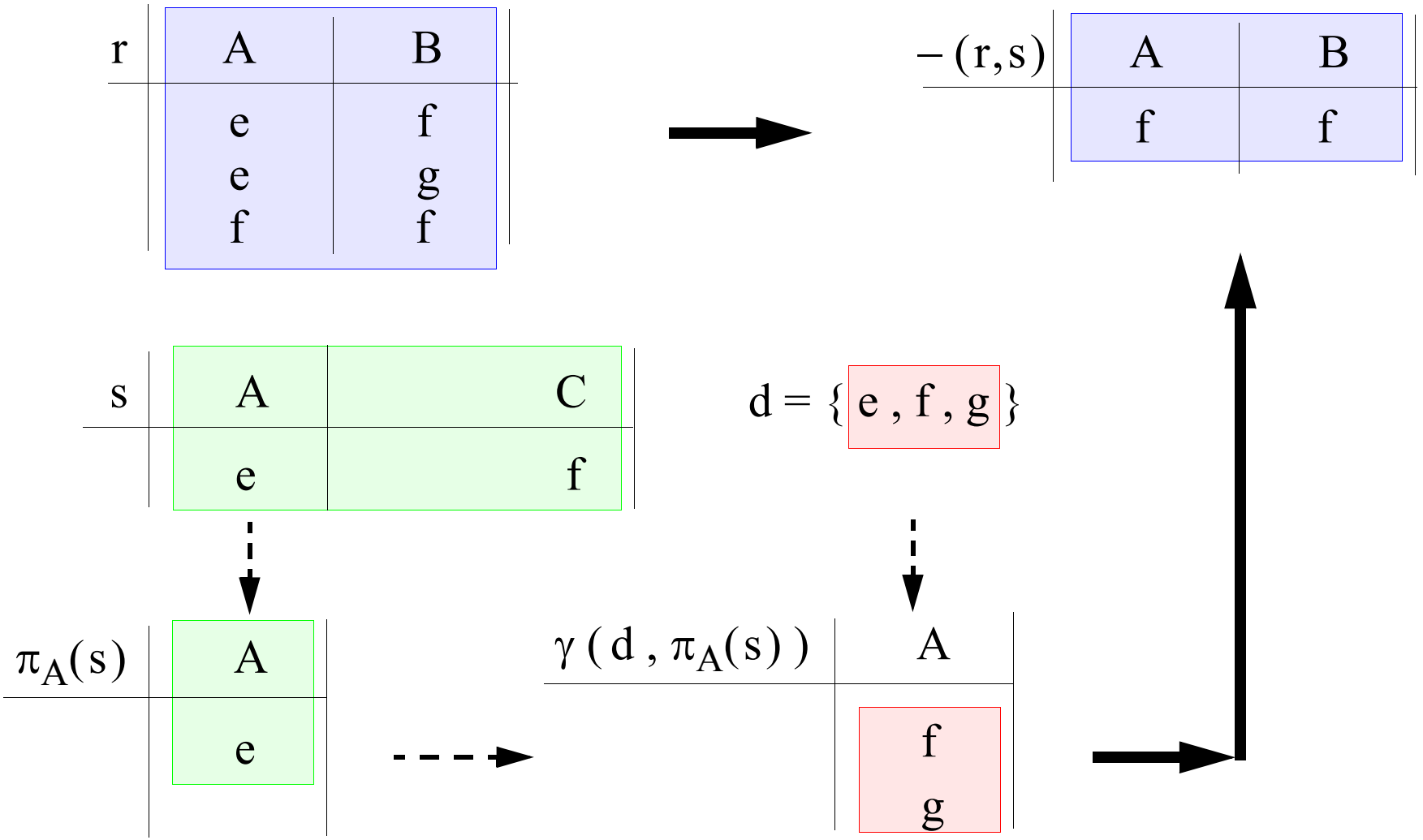
−(r,s)	A	B
	f	f

s	A	C
	e	f

$\pi_A(s)$	A
	e

Differenz: Beispiel mit Umschreibung

$$-(d, r, s) := \quad r \bowtie \gamma(d, \pi_{\text{dom } r \cap \text{dom } s}(s))$$



grundlegende Eigenschaften der Differenz

1. falls dom r = dom s: $-(d, r, s) = r \setminus s$

2. falls dom $r \cap$ dom $s \subset X$: $\pi_X(-(d, r, s)) = -(d, \pi_X(r), \pi_X(s))$

3. falls $A \in$ dom r: $\sigma_{A=c}(-(d, r, s)) = -(d, \sigma_{A=c}(r), s)$

4. falls $A \in$ dom $r \cap$ dom s: $\sigma_{A=c}(-(d, r, s)) = -(d, \sigma_{A=c}(r), \sigma_{A=c}(s))$

Division: Definition

für $\emptyset \neq \text{dom}\, r \cap \text{dom}\, s \neq \text{dom}\, r$ und $s \neq \emptyset$:

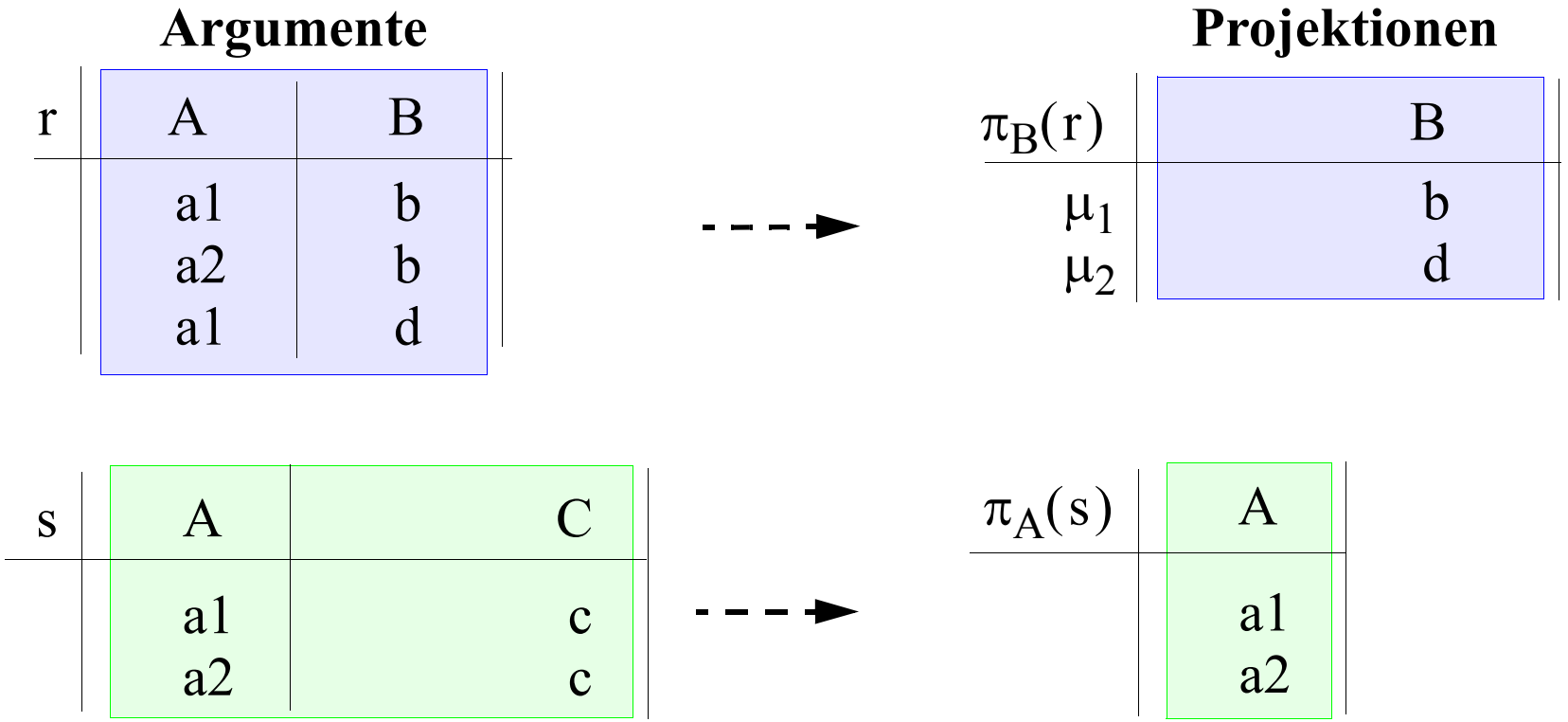
$$r / s := \{ \mu \mid \mu : \text{dom}\, r \setminus \text{dom}\, s \rightarrow \mathbf{C},$$

für alle ν :
wenn $\nu \in \pi_{\text{dom}\, r \cap \text{dom}\, s}(s)$,
dann $\mu \cup \nu \in r$ }

$$= \{ \mu \mid \mu \in \pi_{\text{dom}\, r \setminus \text{dom}\, s}(r),$$

$\{\mu\} \bowtie \pi_{\text{dom}\, r \cap \text{dom}\, s}(s) \subset r$ }

Division: Beispiel



für Tupel aus $\pi_B(r)$:

- für $\mu_1 := (B, b)$: $\{\mu_1\} \bowtie \pi_A(s) \subset r$
- für $\mu_2 := (B, d)$: $\{\mu_2\} \bowtie \pi_A(s) \not\subset r$

also: $r / s = \{(B, b)\}$

grundlegende Eigenschaften der Division

1. $(r \bowtie s) \,/\, s = r$ für $\text{dom}\, r \cap \text{dom}\, s = \varnothing$ und $s \neq \varnothing$

2. $$r \,/\, s = \pi_{\text{dom}\, r \setminus \text{dom}\, s}(r) \setminus \pi_{\text{dom}\, r \setminus \text{dom}\, s}\Big(\pi_{\text{dom}\, r \setminus \text{dom}\, s}(r) \bowtie \pi_{\text{dom}\, r \cap \text{dom}\, s}(s) \setminus r\Big)$$

3.2 relationale Anfrageoperationen: Pragmatik

Formalisierung von Anfragen (grobes “Kochrezept”)

- ***formale Entsprechungen bestimmen*** für

  - umgangssprachliche **Begriffe**:
    Attribute, *Relationensymbole*, *semantische Bereichsnamen*, *Konstantenzeichen*

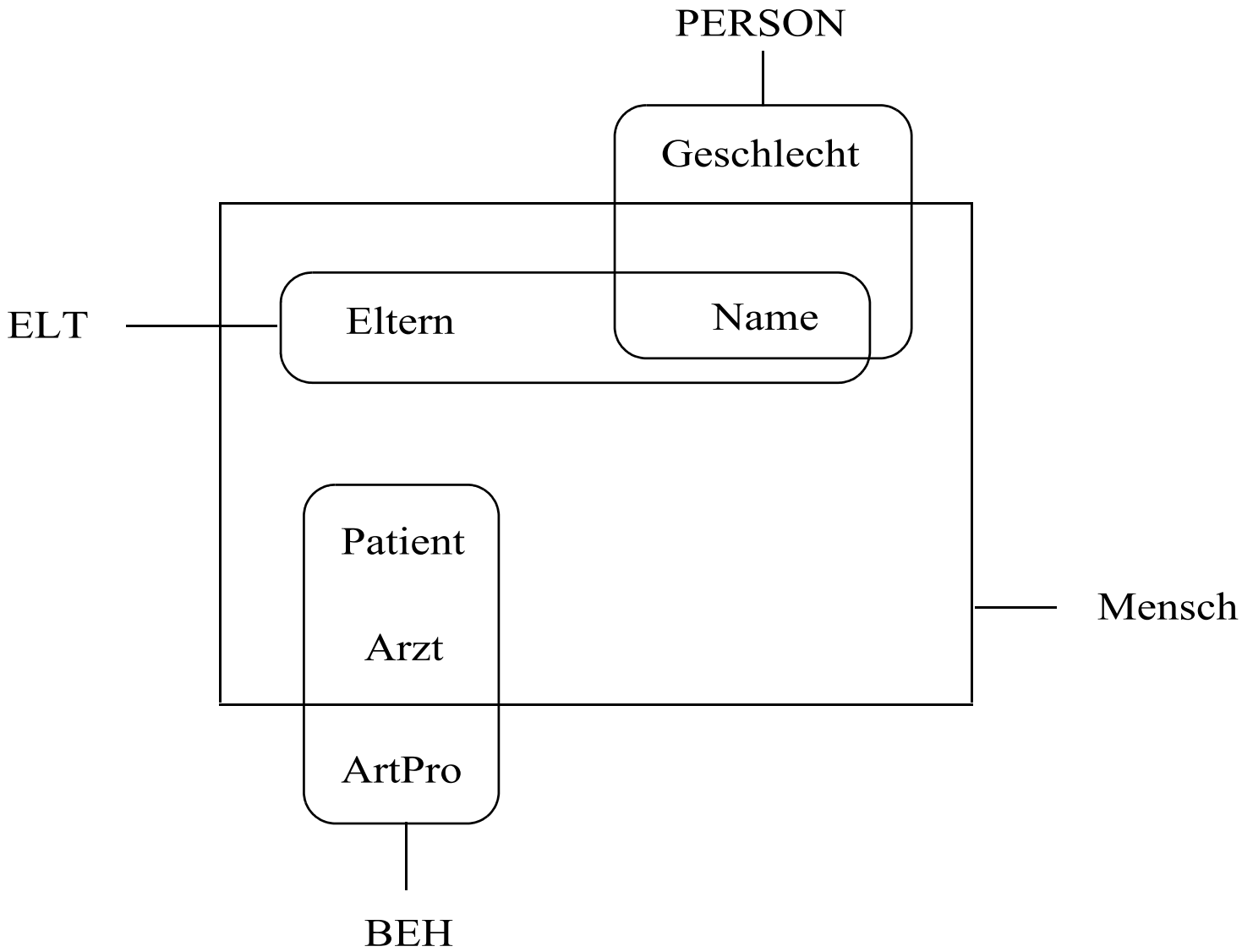
  - umgangssprachliche **Zusammenhänge** zwischen Begriffen:
    Pfade im Schema-Hypergraphen / *Verbunde*

  - umgangssprachliche **Zwecke** des Anfragewunsches:
    Selektionen, *Projektionen*

- ggf. abschließend ***optimieren***

  mit Hilfe der algebraischen Eigenschaften
  (durch Benutzer oder
  automatisch vom Datenbankmanagementsystem)

Hypergraph für Beispiel “Arztpraxis”



Instanz für Beispiel “Arztpraxis”

PERSON	Name	Geschlecht
	Mensch	Geschlecht
	wilhelmine	weib
	fritz	mann
	anton	mann
	theresia	weib
	maria	weib
	hugo	mann
	alfons	mann
	josef	mann
	gerda	weib

ELT	Name	Eltern
	Mensch	Mensch
	maria	anton
	hugo	anton
	alfons	theresia
	anton	wilhelmine
	anton	fritz
	theresia	wilhelmine
	theresia	fritz
	gerda	josef

BEH	Patient	Arzt	ArtPro
	Mensch	Mensch	ArtPro
	maria	gerda	labor
	maria	gerda	hausbesuch
	maria	gerda	röntgen
	anton	josef	untersuchung
	anton	josef	labor
	anton	josef	beratung
	fritz	josef	beratung
	theresia	josef	röntgen

Anfrage: “Bestimme für Mädchen `theresia` ihr Geschlecht und ihre Eltern!”

- Schritt 1:

  nominale Begriffe: *Mädchen*, *Geschlecht*, *Eltern*
  formal: `ELT.Name`, `PERSON.Geschlecht`, `ELT.Eltern`

  possessive Begriffe: *ihr* (Geschlecht), *ihre* (Eltern)
  formal: `PERSON`, `ELT`

- Schritt 2:

  Zusammenhang: Mädchen ihr Geschlecht und ihre Eltern
  formal: Hypergraph-Pfad “`PERSON, Name, ELT`”
  $\text{PERSON} \bowtie \text{ELT}$

- Schritt 3:

  Zweck: für `theresia`
  formal: $\sigma_{\text{Name}=\text{theresia}}(\text{PERSON} \bowtie \text{ELT})$

- Optimierung:

  Selektion nach `theresia` für `PERSON.Name` / `ELT.Name` vorziehen:

  $$\sigma_{\text{Name}=\text{theresia}}(\text{PERSON}) \bowtie \sigma_{\text{Name}=\text{theresia}}(\text{ELT})$$

Anfrage: “Bestimme alle Ärzte, die weibliche Patienten behandeln!”

- Schritt 1:

  nominale Begriffe: *Arzt*, *Patient*
  formal: `BEH.Arzt`, `BEH.Patient`

  adjektiver Begriff: *weiblich*
  formal: `weib` für `PERSON.Geschlecht`

  verbaler Begriff: *behandeln*
  formal: `BEH`

- Schritt 2:

  Zusammenhang: Ärzte, die weibliche Patienten behandeln
  formal: Hypergraph-Pfad
  “`BEH, Patient, Mensch, Name, PERSON`”

  π_q (BEH) $\bowtie$ PERSON mit
  q := { (Name, Patient), (Arzt, Arzt), (ArtPro, ArtPro) } oder

  $\sigma_{\text{Patient}=\text{Name}}$ (BEH $\bowtie$ PERSON)

- Schritt 3:

  Zweck: Ärzte gemäß Zusammenhang

  formal: $\pi_{Arzt}\,(\ \sigma_{Geschlecht=weib}\,(\ \pi_{q}(\,BEH\,)\ \bowtie\ PERSON\,)\ )$ oder

  $\pi_{Arzt}\,(\ \sigma_{Geschlecht=weib}\,(\ \sigma_{Patient=Name}\,(\ BEH\ \bowtie PERSON\,)\,)\,)$

- Optimierung:

  Selektion nach `weib` für `PERSON.Geschlecht` sowie
  Entfernen des Attributs `BEH.ArtPro` vorziehen:

  $$\pi_{Arzt}\,(\pi_{q1}\,(BEH)\ \bowtie\ \sigma_{Geschlecht=weib}\,(PERSON))$$

  mit $q1 = \{\ (\,Patient\ ,\ Name\,)\,,$
  $(\,Arzt\ ,\ Arzt\ )\ \}$

  oder

  $$\pi_{Arzt}\,(\ \sigma_{Patient=Name}\,(\ \pi_{Patient,Arzt}\,(BEH)\ \bowtie\ \sigma_{Geschlecht=weib}\,(\,PERSON\,)\ )\ )$$

Anfrage: “Für welche Patienten wurden alle medizintechnischen Möglichkeiten ausgenutzt?”

- Schritt 1:

  nominale Begriffe: *Patient*, *Möglichkeit*
  formal: `BEH.Patient`, `BEH.ArtPro`

  adjektiver Begriff: *medizintechnisch*
  formal: `{labor, röntgen}` für `BEH.ArtPro`

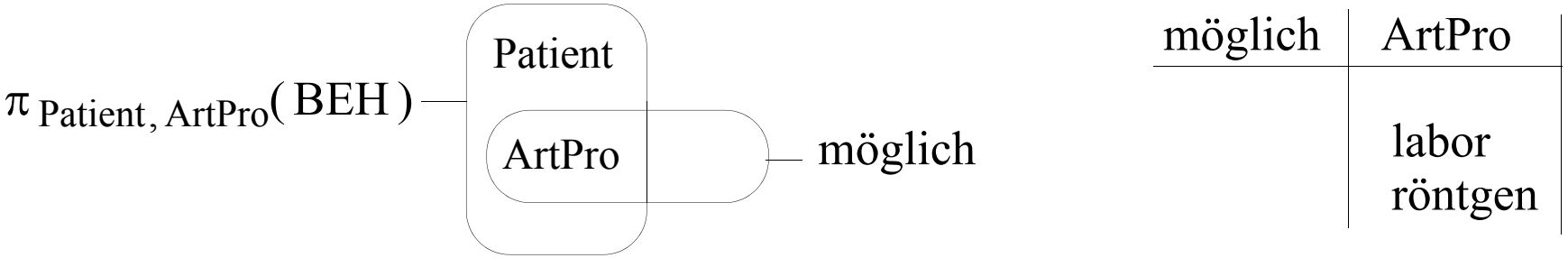
  verbaler Begriff: *ausgenutzt*
  formal: BEH

- Schritt 2:

Zusammenhang: zwischen Patienten
und tatsächlich ausgenutzten Behandlungsarten einerseits
und möglichen Behandlungsarten andererseits

formal: nicht unmittelbar aus dem Schema-Hypergraphen ablesbar;

- erste Komponente stellt eine Teil-Hyperkante dar;
- zweite Komponente als durch Anfragewunsch ergänzt denken:



Zusammenhang enthält hier All-Aussage,
durch Division ausdrückbar:

$\pi_{Patient, ArtPro}(BEH) \ / \ möglich$

- Schritt 3: entfällt hier

- Optimierung:

  Divisor `möglich` als konstante Relation bekannt,

  also *All-Aussage* schon vom Benutzer in *Und-Aussage* umwandelbar:

$$\pi_{\text{Patient}}\left(\ \sigma_{\text{ArtPro}=\text{labor}}\left(\text{BEH}\right)\right) \quad \bowtie \quad \pi_{\text{Patient}}\left(\sigma_{\text{ArtPro}=\text{röntgen}}\left(\text{BEH}\right)\right)$$

3.3 relationale Anfrageoperationen: Vorgriff auf Praxis (Oracle-SQL)

- **natürlicher Verbund:**

  für Relationenschemas $< R \mid \{A_1,\ldots,A_k, \quad B_1,\ldots,B_l\} \mid >$ und
  $< S \mid \{ \quad B_1,\ldots,B_l, \quad C_1,\ldots,C_m\} \mid >$:

```
SELECT DISTINCT  R.A1,...,R.Ak,
                 R.B1,...,R.Bl,
                 S.C1,...,S.Cm
  FROM R, S

    WHERE R.B1 = S.B1   AND   R.B2 = S.B2   AND ... AND   R.Bl = S.Bl
```

- **A=c-Selektion:**

  für Relationenschema $< R \mid X \mid >$ mit $A \in X$:

```
SELECT DISTINCT *
  FROM R

    WHERE A = 'c'
```

- **Projektion**

  für Relationenschema $< R \mid X \mid >$ mit $\{A_1, \ldots, A_k\} \subset X$:

```
SELECT DISTINCT A1,...,Ak
  FROM R
```

- **Vereinigung**

  für vereinigungsverträgliche Relationenschemas $< R \mid X \mid >$ und $< S \mid X \mid >$:

```
SELECT DISTINCT *
  FROM R
UNION
SELECT DISTINCT *
  FROM S
```

- **A=B-Vergleich**

  für Relationenschema $< R \mid X \mid >$ mit $\{A, B\} \subset X$:

```
SELECT DISTINCT *
  FROM R
    WHERE A = B
```

- **Differenz**

  für vereinigungsverträgliche Relationenschemas $< R \mid X \mid >$ und $< S \mid X \mid >$:

```
  SELECT DISTINCT *  FROM R
MINUS
  SELECT DISTINCT *  FROM S
```

4 Relationale Anfragesprachen

relationale Anfragen: anschaulich

- sind Operationen auf ***Instanzen*** von Schemas:

  Argumente: Relationen und gegebenenfalls das Universum
  Werte: Relationen

- behandeln Relationen als ***Mengen*** von Tupeln:

  Mengen ungeordnet,
  keine Eigenschaften der Codierungen berücksichtigt

- behandeln Konstantenzeichen als ***atomare, uninterpretierte*** Dinge:

  das Informationssystem “weiß” nur
  - deren *Gleichheit* bzw. *Ungleichheit* und
  - die in der *aktuellen Instanz* $(d, r_1, \ldots, r_n)$ ausgedrückten Beziehungen

relationale Anfragen: formale Definition

Q ist ***relationale Anfrage*** (relational query) der Signatur $X_1, \ldots, X_n \to X$:gdw

- [***getypt*** *mit Signatur* $X_1, \ldots, X_n \to X$]

  $Q: \text{sat}_{RS} \to \text{sat}_R$,

  wobei $RS = < < \mid X_1 \mid > , \ldots , < \mid X_n \mid > \mid \mid >$ und $R = < \mid X \mid >$

- [***universumstreu***]

  falls $\mu \in Q(d, r_1, \ldots, r_n)$ und $A \in X$, dann $\mu(A) \in d$

- [***berechenbar***]

  Q partiell rekursiv

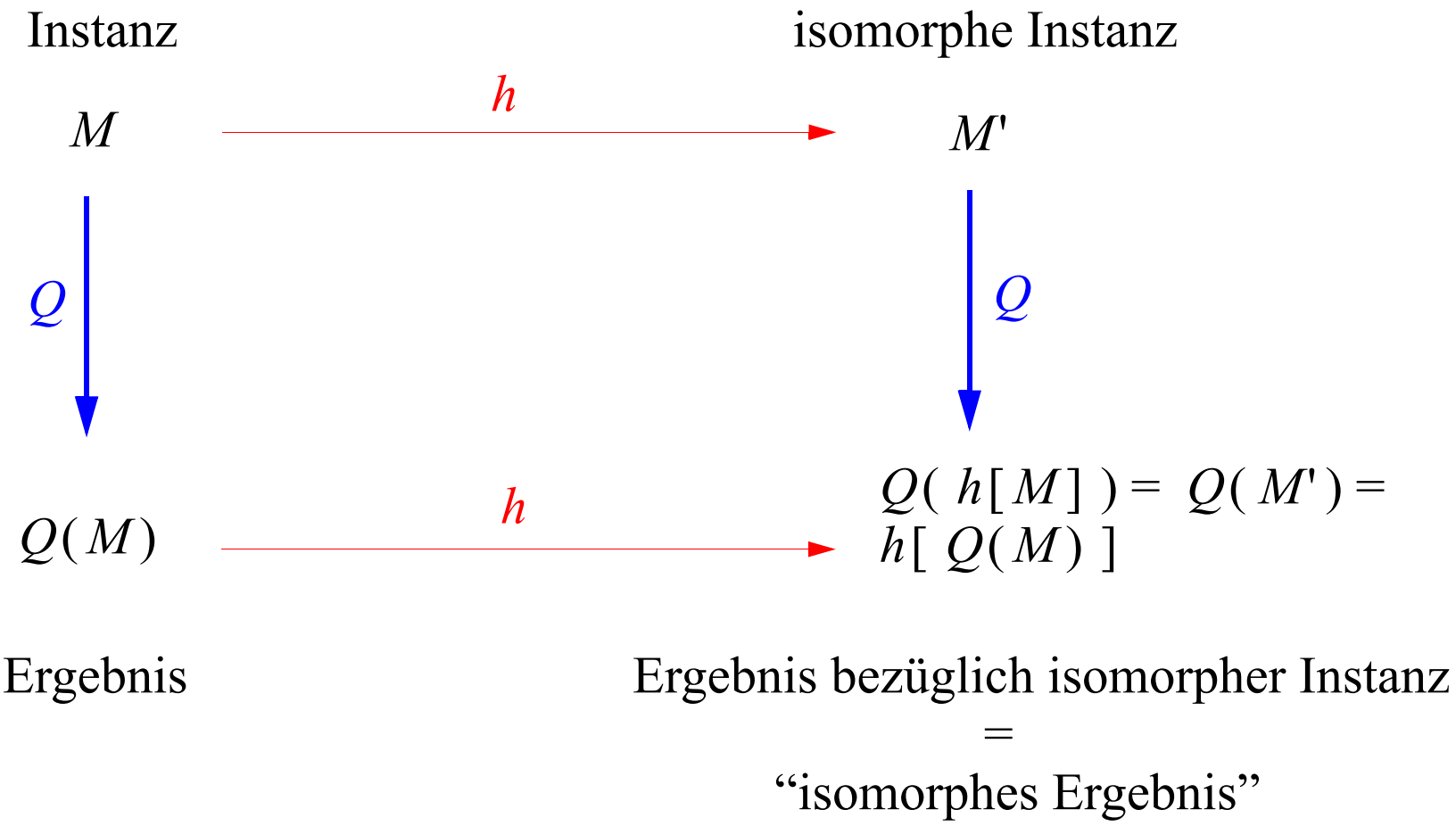
- [***isomorphietreu***]

  sind Instanzen $M = (d, r_1, \ldots, r_n)$ und $M' = (d', r_1', \ldots, r_n')$

  isomorph unter einer Bijektion $h : d \xrightarrow[\text{auf}]{\text{1-1}} d'$, also $h[M] = M'$,

  so gilt: $h[Q(M)] = Q(h[M])$

Isomorphietreue:



relationale Anfragesprache: Übersicht

- **Definition**

  eine formale Sprache ***L*** (Syntax) mit:
  für jedes Wort $\Phi \in \boldsymbol{L}$ liefert die Semantik $\mathrm{eval}(\Phi)$ eine relationale Anfrage

- **Beispiele**

  ***Relationenalgebra*:**
  benutzt Operationszeichen für grundlegende relationale Operationen

  ***Relationenkalkül*:**
  benutzt Prädikatenlogik und Mengenlehre

  - *werteorientiert*: Individuenvariablen durch Konstantenzeichen belegen
  - *tupelorientiert*: Individuenvariablen durch ganze Tupel belegen

  Kern der ***Structured Query Language*** (**SQL**):
  tupelorientierter Relationenkalkül mit algebraischen Ergänzungen

Beispiel für relationale Anfragesprache: Relationenalgebra

- **Voraussetzungen**
  - $RS = < < R_1 \mid X_1 \mid > , \ldots , < R_n \mid X_n \mid > \mid \mid >$ Datenbankschema
  - $\mathrm{D} \notin \mathbf{A} \cup \mathbf{R}$ Name für das Universum

- ***Syntax* von L_{Al}** (***relationale Ausdrücke*** mit Definitionsbereichen)

1. $R_1, \ldots, R_n \in L_{Al}$ mit $\mathrm{dom}\, R_i := X_i$
   $\mathrm{D} \in L_{Al}$ mit $\mathrm{dom}\, \mathrm{D} := \{ \mathrm{D} \}$

2. sind $\Phi, \Psi \in L_{Al}$, $A, B \in \mathbf{A} \cup \{ \mathrm{D} \}$, $X \subset_{\text{endlich}} \mathbf{A} \cup \{\mathrm{D}\}$, dann induktiv auch:

- $(\Phi \bowtie \Psi) \in L_{Al}$ mit $\mathrm{dom}\, (\Phi \bowtie \Psi) := \mathrm{dom}\, \Phi \cup \mathrm{dom}\, \Psi$
- $(\Phi + \Psi) \in L_{Al}$ mit $\mathrm{dom}\, (\Phi + \Psi) := \mathrm{dom}\, \Phi \cup \mathrm{dom}\, \Psi$
- $\pi_q(\Phi) \in L_{Al}$, wobei $q : X \to \mathrm{dom}\, \Phi$ mit $\mathrm{dom}\, \pi_q(\Phi) := X$
- $\sigma_{A=B}(\Phi) \in L_{Al}$, wobei $\{A,B\} \subset \mathrm{dom}\, \Phi$ mit $\mathrm{dom}\, \sigma_{A=B}(\Phi) := \mathrm{dom}\, \Phi$
- $\sigma_{A\neq B}(\Phi) \in L_{Al}$, wobei $\{A,B\} \subset \mathrm{dom}\, \Phi$ mit $\mathrm{dom}\, \sigma_{A\neq B}(\Phi) := \mathrm{dom}\, \Phi$
- $\gamma(\Phi) \in L_{Al}$ mit $\mathrm{dom}\, \gamma(\Phi) := \mathrm{dom}\, \Phi$

• *Semantik* von L_{Al}

$\text{eval}(\Phi) : \text{sat}_{RS} \rightarrow \text{sat}_{<\ |\ \text{dom}\,\Phi\ |\ >}$ für alle $\Phi \in \boldsymbol{L}_{Al}$

1. $\text{eval}(R_i)(d,r_1,\ldots,r_n) := r_i$

   $\text{eval}(\mathrm{D})(d,r_1,\ldots,r_n) := \{\ (\mathrm{D},a) \mid a \in d\ \} \cong d$

2. $\text{eval}(\Phi \bowtie \Psi)(d,r_1,\ldots,r_n) := \bowtie(\text{eval}(\Phi)(d,r_1,\ldots,r_n), \text{eval}(\Psi)(d,r_1,\ldots,r_n))$

   $\text{eval}(\Phi + \Psi)(d,r_1,\ldots,r_n) := +(d, \text{eval}(\Phi)(d,r_1,\ldots,r_n), \text{eval}(\Psi)(d,r_1,\ldots,r_n))$

   $\text{eval}(\pi_q(\Phi))(d,r_1,\ldots,r_n) := \pi_q(\text{eval}(\Phi)(d,r_1,\ldots,r_n))$

   $\text{eval}(\sigma_{A=B}(\Phi))(d,r_1,\ldots,r_n) := \sigma_{A=B}(\text{eval}(\Phi)(d,r_1,\ldots,r_n))$

   $\text{eval}(\sigma_{A\neq B}(\Phi))(d,r_1,\ldots,r_n) := \sigma_{A\neq B}(\text{eval}(\Phi)(d,r_1,\ldots,r_n))$

   $\text{eval}(\gamma(\Phi))(d,r_1,\ldots,r_n) := \gamma(d, \text{eval}(\Phi)(d,r_1,\ldots,r_n))$

grundlegende Eigenschaft

- **Behauptung**

  Für alle Ausdrücke $\Phi \in \boldsymbol{L}_{Al}$
  ist die Semantik $\mathrm{eval}(\Phi)$ eine relationale Anfrage.

- **Begründung**

  ergibt sich unmittelbar
  aus den Definitionen der relationalen Operationen:

  diese benutzen nämlich nur

  - die *Mengeneigenschaften* der Argumentrelationen
  - *Gleichheits*- bzw. *Ungleichheitsbeziehungen* zwischen Konstantenzeichen oder aus Konstantenzeichen aufgebauten Tupeln

- **formaler Beweis**

  Übungsaufgabe !

Bemerkungen zur grundlegenden Eigenschaft

- *Selektion(en)* $\sigma_{A=c}$ sind im streng formalen Sinne *keine* relationalen Anfragen:
  das Konstantenzeichen c wird "nicht isomorphietreu verarbeitet "

- *Information Retrieval-Systeme* und *Multimedia-Systeme* betrachten
  anstelle von Gleichheitsbeziehungen häufig
  schwächere "*Ähnlichkeitsbeziehungen*",
  die im Allgemeinen auch "nicht isomorphietreu verarbeitet" werden (können)

- Umkehrung des Satzes gilt *nicht*:
  es gibt relationale Anfragen,
  die *nicht* in der Relationenalgebra ausgedrückt werden können

Mächtigkeit der Relationenalgebra: als Definierbarkeit

die Relation s
ist in der Anfragesprache $\boldsymbol{L}_{Al}$
aus der Instanz $(d, r_1, \ldots, r_n)$ ***definierbar***

genau dann, wenn

die Relation s
ist ***invariant*** unter allen Automorphismen
von der Instanz $(d, r_1, \ldots, r_n)$

Mächtigkeit der Relationenalgebra: formaler Satz

Sei

$RS = < < R_1 \mid X_1 \mid > , \ldots , < R_n \mid X_n \mid > \mid \mid >$ Datenbankschema,

$(d, r_1, \ldots, r_n) \in \text{sat}_{RS}$ Instanz,

s mit $\text{dom}\, s = X$ Relation.

Dann

gibt es einen Ausdruck $\Phi \in \boldsymbol{L}_{Al}$ mit $\text{eval}\,(\Phi)\,(d, r_1, \ldots, r_n) = s$

[s ist in $\boldsymbol{L}_{Al}$ aus $(d, r_1, \ldots, r_n)$ definierbar]

genau dann, wenn

für alle Automorphismen $h : d \xrightarrow[\text{auf}]{\text{1-1}} d$ mit $h[\, d, r_1, \ldots, r_n] = (d, r_1, \ldots, r_n)$

gilt: $h[\, s\,] = s$

[s ist invariant unter allen Automorphismen von $(d, r_1, \ldots, r_n)$].

Beweis von "$\Rightarrow$"

sei $\Phi \in \boldsymbol{L}_{Al}$ mit $\text{eval}(\Phi)(d, r_1, \ldots, r_n) = s$

eval (Φ) ist relationale Anfrage und
damit insbesondere isomorphietreu

also gilt für alle Automorphismen $h : d \xrightarrow[\text{auf}]{\text{1-1}} d$ mit
$h[d, r_1, \ldots, r_n] = (d, r_1, \ldots, r_n)$:

$$
\begin{aligned}
h[s] &= h[\ \text{eval}(\Phi)(d, r_1, \ldots, r_n)\ ] && \text{Voraussetzung} \\
&= \text{eval}(\Phi)(\ h[d, r_1, \ldots, r_n\ ]\ ) && \text{eval}(\Phi) \text{ isomorphietreu} \\
&= \text{eval}(\Phi)(d, r_1, \ldots, r_n) && h \text{ Automorphismus} \\
&= s && \text{Voraussetzung}
\end{aligned}
$$

Beweisidee für "$\Leftarrow$"

- für Instanz $M := (d, r_1, \ldots, r_n)$ ist die

  Automorphismengruppe $\mathrm{Aut}_M := \{\ h \mid\ h : d \xrightarrow[\text{auf}]{\text{1-1}} d \quad \text{mit} \quad h[M] = M\ \}$

  durch eine Relation aut_M derart darstellbar, dass:
  es gibt Ausdruck $\Phi_M \in \boldsymbol{L}_{Al}$ mit $\mathrm{eval}(\Phi_M)(M) = \mathrm{aut}_M$

- die Relation s ist mit Hilfe von aut_M derart darstellbar, dass:
  es gibt Ausdruck $\Psi_s \in \boldsymbol{L}_{Al}$ mit $\mathrm{eval}(\Psi_s)(\mathrm{aut}_M) = s$

- also:
  es gibt Ausdrücke $\Phi_M, \Psi_s \in \boldsymbol{L}_{Al}$ mit $\mathrm{eval}(\Psi_s)(\ \mathrm{eval}(\Phi_M)(M)\ ) = s$

- Ausdruck Φ_M geeignet in Ψ_s einsetzen:
  es gibt Ausdruck $\Phi \in \boldsymbol{L}_{Al}$ mit $\mathrm{eval}(\Phi)(M) = s$

Beispiel für relationale Anfragesprache: Relationenkalkül

- **Voraussetzungen**
  - $RS = < < R_1 \mid X_1 \mid > , \ldots , < R_n \mid X_n \mid > \mid \mid >$ Datenbankschema
  - $D \notin \mathbf{A} \cup \mathbf{R}$ Name für das Universum

- ***Syntax* von L_K** (relationale ***Formeln*** mit Definitionsbereichen)

1. für Attributmenge $X_i = \{ Ai_1, \ldots, Ai_k \}$,
   paarweise verschiedene Individuenvariablen $v_{Aj_1}, \ldots, v_{Aj_k}$:

   $R_i (Ai_1 : v_{Aj_1} , \ldots , Ai_k : v_{Aj_k}) \in \boldsymbol{L}_K$

   mit dom $R_i (Ai_1 : v_{Aj_1} , \ldots , Ai_k : v_{Aj_k}) := \{ Aj_1, \ldots, Aj_k \}$

   für Individuenvariablen v_{Ai}, v_{Aj} :

   $(v_{Ai} = v_{Aj}) \in \boldsymbol{L}_K$ mit dom $(v_{Ai} = v_{Aj}) := \{ Ai , Aj \}$

Syntax von L_K (relationale *Formeln* mit Definitionsbereichen): Fortsetzung

2. sind $\Phi, \Psi \in L_K$ Formeln des Relationenkalküls, dann induktiv auch:

$(\Phi \wedge \Psi) \in L_K$ mit $\mathrm{dom}\ (\Phi \wedge \Psi) := \mathrm{dom}\ \Phi \cup \mathrm{dom}\ \Psi$

$(\Phi \vee \Psi) \in L_K$ mit $\mathrm{dom}\ (\Phi \vee \Psi) := \mathrm{dom}\ \Phi \cup \mathrm{dom}\ \Psi$

$(\neg \Phi) \in L_K$ mit $\mathrm{dom}\ (\neg \Phi) := \mathrm{dom}\ \Phi$

3. ist $\Phi \in L_K$ mit $Ai \in \mathrm{dom}\ \Phi$, dann induktiv auch:

$(\exists v_{Ai})\ \Phi \in L_K$ mit $\mathrm{dom}\ (\exists v_{Ai})\ \Phi := \mathrm{dom}\ \Phi \setminus \{Ai\}$

$(\forall v_{Ai})\ \Phi \in L_K$ mit $\mathrm{dom}\ (\forall v_{Ai})\ \Phi := \mathrm{dom}\ \Phi \setminus \{Ai\}$

- **Semantik von L_K**

$$\text{eval}(\Phi): \text{sat}_{RS} \rightarrow \text{sat}_{<|\text{dom}\,\Phi|>} \quad \text{für alle} \quad \Phi \in \boldsymbol{L}_K$$

$$\text{eval}(\Phi)(d, r_1, \ldots, r_n) := \{ \mu \mid \mu : \text{dom}\,\Phi \rightarrow d, \models_{(d, r_1, \ldots, r_n), \mu} \Phi \}$$

- **anschaulich**

zurückgeliefert werden diejenigen Tupel μ, für die gilt:

- der *Definitionsbereich* des Tupels μ ist gleich dem Definitionsbereich $\text{dom}\,\Phi$ der relationalen Formel Φ

- das Tupel μ ist aufgebaut aus Konstantenzeichen des Universums d der (gespeicherten) Instanz

- das Tupel μ macht die "relationale Formel Φ wahr in der Instanz $(d, r_1, \ldots, r_n)$"

- **Bemerkung**

Definitionsbereiche entsprechen genau
den ***frei vorkommenden*** Variablen:

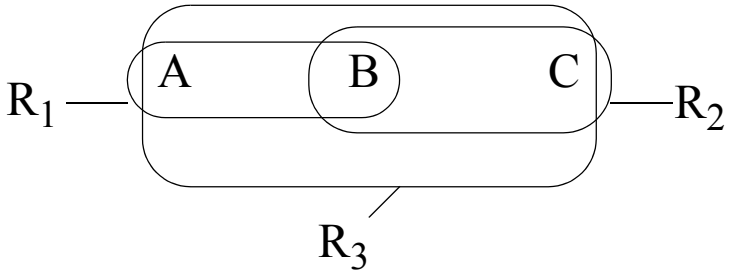
$Aj \in \text{dom}\ \Phi$ genau dann, wenn v_{Aj} kommt in Φ frei vor

hier vorausgesetzt:
eine eineindeutige Korrespondenz
zwischen Individuenvariablen v_{Aj} und Attributen Aj

manchmal:
Individuenvariable v_{Aj} einfach mit Attribut Aj identifiziert

Beispiel für relationale Formel

- **Voraussetzung**
  - Datenbankschema:
    $RS = < < R_1 \mid \{A,B\} \mid > , < R_2 \mid \{B,C\} \mid > , < R_3 \mid \{A,B,C\} \mid > \mid \mid >$
  - Hypergraph:



- **umgangssprachliche Umschreibung**

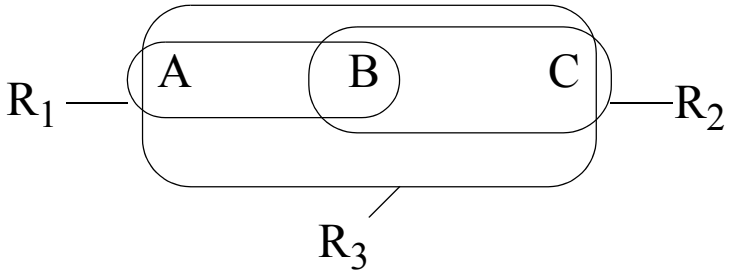
  liefere die Tupel über Attribute A und B,
  die in der Instanz von R_1 sind

- **relationale Formel**

  $R_1 (A : v_A , B : v_B)$

Beispiel für relationale Formel

- **Voraussetzung**
  - Datenbankschema:
    $RS = < < R_1 \mid \{A,B\} \mid > , < R_2 \mid \{B,C\} \mid > , < R_3 \mid \{A,B,C\} \mid > \mid \mid >$
  - Hypergraph:



- **umgangssprachliche Umschreibung**

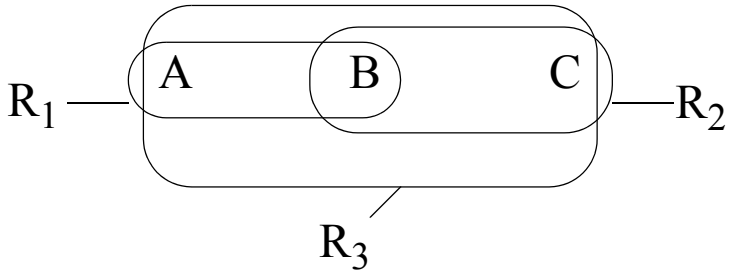
  liefere die Tupel über Attribute B und C,
  die in der Instanz von R_2 sind

- **relationale Formel**

  $R_2 (B : v_B , C : v_C)$

Beispiele für relationale Formel

- **Voraussetzung**



- **umgangssprachliche Umschreibungen mit relationalen Formeln**
  - liefere die "Zusammensetzungen von *passenden* Tupeln aus der Instanz von R_1 und der Instanz von R_2":

    $(\; R_1(\; A : v_A \; , \; B : v_B \;) \quad \wedge \quad R_2(\; B : v_B \; , \; C : v_C \;) \;)$

  - liefere die Tupel, die in vorgenannter Menge sind *oder* die in der Instanz von R_3 sind:

    $(\quad R_1(\; A : v_A \; , \; B : v_B \;) \quad \wedge \quad R_2(\; B : v_B \; , \; C : v_C \;) \quad) \quad \vee$
    $R_3(\; A : v_A \; , \; B : v_B \; , \; C : v_C \;) \quad)$

  - liefere die *umbenannten* Tupel über Attribute D und E, die in der Instanz von R_1 sind:

    $R_1(\; A : v_D \; , \; B : v_E \;)$

Gleichmächtigkeit von Algebra und Kalkül

Der Relationenkalkül kann durch die Relationenalgebra *verwirklicht* werden:

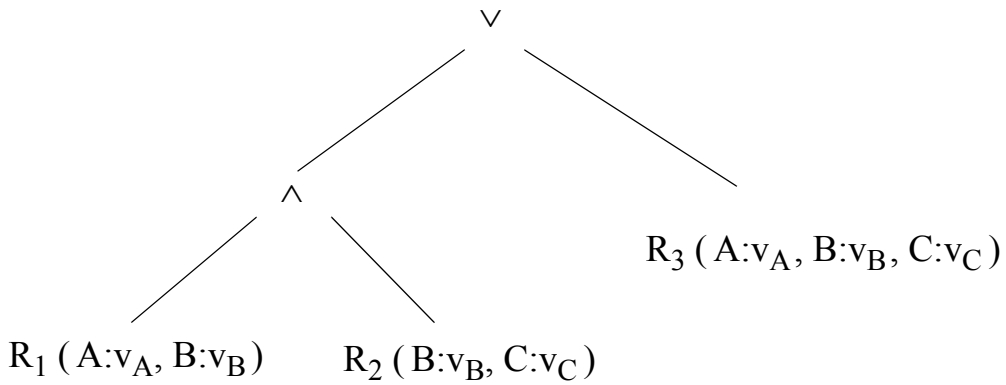
relationale Formeln
werden uniform *übersetzt*
in ***relationale Ausdrücke***

und damit wird die
algorithmische Auswertung von relationalen Formeln ermöglicht.

Gleichmächtigkeit von Algebra und Kalkül: einfaches Beispiel

relationale Formel: $((R_1(A{:}v_A, B{:}v_B) \wedge R_2(B{:}v_B, C{:}v_C)) \vee R_3(A{:}v_A, B{:}v_B, C{:}v_C))$

Syntaxbaum der Formel:



Übersetzungen:

$R_1 (A{:}v_A , B{:}v_B)$	in	R_1
$R_2 (B{:}v_B , C{:}v_C)$	in	R_2
$(R_1 (A{:}v_A , B{:}v_B) \wedge R_2 (B{:}v_B , C{:}v_C))$	in	$R_1 \bowtie R_2$
$((R_1 (A{:}v_A, B{:}v_B) \wedge R_2 (B{:}v_B, C{:}v_C)) \vee R_3 (A{:}v_A, B{:}v_B, C{:}v_C))$	in	$(R_1 \bowtie R_2) + R_3$

Gleichmächtigkeit von Algebra und Kalkül: Satz

- **Die Relationenalgebra und der Relationenkalkül sind gleichmächtig.**

- Grundlegendes Ergebnis der Theorie des relationalen Datenmodells:

- die *deklarative* Semantik des Relationenkalküls
  kann durch die Relationenalgebra
  korrekt und ***vollständig*** operationalisiert werden

- die Ausdrucksfähigkeit der *operational* gegebenen Relationenalgebra
  lässt sich durch den Relationenkalkül ***deklarativ*** genau beschreiben

- (nicht immer erreichtes) Vorbild für andere Datenmodelle!

- konstruktiver Beweis liefert verifiziertes ***Übersetzungsverfahren***

Behauptung 1: Kalkül in Algebra äquivalent übersetzbar

zu zeigen:

für alle Formeln $\chi \in \boldsymbol{L}_K$
gibt es einen Ausdruck $\chi^* \in \boldsymbol{L}_{Al}$
mit $\text{eval}_K(\chi) = \text{eval}_{Al}(\chi^*)$

Beweis:

wir konstruieren durch Induktion über den Aufbau von $\boldsymbol{L}_K$

für jede Formel $\chi \in \boldsymbol{L}_K$

einen Ausdruck $\chi^* \in \boldsymbol{L}_{Al}$

mit $\text{eval}_K(\chi) = \text{eval}_{Al}(\chi^*)$

1a.

$$\text{eval}_K(\ R_i(\ Ai_1 : v_{Aj_1}\ ,\ldots,\ Ai_k : v_{Aj_k}\ )\ )(d, r_1, \ldots, r_n)$$

$=$Definition eval_K

$$\{\ \mu\ \mid\ \mu : \{Aj_1, \ldots, Aj_k\} \rightarrow d,\ |=_{(d, r_1, \ldots, r_n),\mu}\ R_i(\ Ai_1 : v_{Aj_1}\ ,\ldots,\ Ai_k : v_{Aj_k}\ )\ \}$$

$=$Definition $|=$

$$\{\ \mu\ \mid\ \mu : \{Aj_1, \ldots, Aj_k\} \rightarrow d,\ \ \mu \circ q^{-1} \in r_i\ \ \ \}\ \text{mit}$$

q: $\{\ Aj_1, \ldots, Aj_k\} \rightarrow \{\ Ai_1, \ldots, Ai_k\ \}$

$q\ (\ Aj_l\ )\ :=\ Ai_l\quad$ für $\ l = 1, \ldots, k$

$=$Definition π_q

$$\pi_q(\ r_i\ )$$

$=$Definition eval_{Al}

$$\text{eval}_{Al}(\ \pi_q(\ R_i\ )\ )(d, r_1, \ldots, r_n)$$

1b.

$\text{eval}_K(\ v_{Ai} = v_{Aj}\ )(d, r_1, \ldots, r_n)$

=Definition eval_K

$$\{\ \mu\ \mid\ \mu : \{Ai, Aj\} \rightarrow d,\qquad \models_{(d, r_1, \ldots, r_n), \mu}\ (v_{Ai} = v_{Aj})\qquad\}$$

=Definition $\models$

$$\{\ \mu\ \mid\ \mu : \{Ai, Aj\} \rightarrow d,\qquad \mu(v_{Ai}) = \mu(v_{Aj})\qquad\}$$

=Definition π_{qij}

$\pi_{qij}(d)$ mit $qij : \{Ai, Aj\} \rightarrow \{\text{D}\}$

=Definition eval_{Al}

$\text{eval}_{Al}(\ \pi_{qij}(\text{D})\ )(d, r_1, \ldots, r_n)$

2a. (Induktionsschritt für Konjunktion)

$\text{eval}_K(\ \Phi \wedge \Psi\ )(d, r_1, \ldots, r_n)$

=Definition eval_K

$$\{\ \mu\ \mid\ \mu : \text{dom}\,\Phi \cup \text{dom}\,\Psi \rightarrow d,\ \models_{(d, r_1, \ldots, r_n), \mu}\ (\Phi \wedge \Psi)\ \}$$

=Definition $\models$

$$\{\ \mu\ \mid\ \mu : \text{dom}\,\Phi \cup \text{dom}\,\Psi \rightarrow d,\ \models_{(d, r_1, \ldots, r_n), \mu}\ \Phi \quad \text{und}$$
$$\models_{(d, r_1, \ldots, r_n), \mu}\ \Psi\ \}$$

=

$$\{\ \mu\ \mid\ \mu : \text{dom}\,\Phi \cup \text{dom}\,\Psi \rightarrow d,\ \models_{(d, r_1, \ldots, r_n), \mu\lceil \text{dom}\,\Phi}\ \Phi \quad \text{und}$$
$$\models_{(d, r_1, \ldots, r_n), \mu\lceil \text{dom}\,\Psi}\ \Psi\ \}$$

=Definition eval_K

$$\{\ \mu\ \mid\ \mu : \text{dom}\,\Phi \cup \text{dom}\,\Psi \rightarrow d,\ \mu\lceil \text{dom}\,\Phi \in \text{eval}_K(\Phi)(d, r_1, \ldots, r_n)\ \text{und}$$
$$\mu\lceil \text{dom}\,\Psi \in \text{eval}_K(\Psi)(d, r_1, \ldots, r_n)\ \}$$

... (siehe nächste Seite)

$$\{ \quad \mu \quad | \quad \mu : \mathrm{dom}\,\Phi \cup \mathrm{dom}\,\Psi \rightarrow d, \; \mu\lceil \mathrm{dom}\,\Phi \in \mathrm{eval}_K(\Phi)(d, r_1, \ldots, r_n) \;\; \text{und} \;\; \mu\lceil \mathrm{dom}\,\Psi \in \mathrm{eval}_K(\Psi)(d, r_1, \ldots, r_n) \quad \}$$

=Induktionsannahme

$$\{ \quad \mu \quad | \quad \mu : \mathrm{dom}\,\Phi \cup \mathrm{dom}\,\Psi \rightarrow d, \; \mu\lceil \mathrm{dom}\,\Phi \in \mathrm{eval}_{Al}(\Phi^*)(d, r_1, \ldots, r_n) \;\; \text{und} \;\; \mu\lceil \mathrm{dom}\,\Psi \in \mathrm{eval}_{Al}(\Psi^*)(d, r_1, \ldots, r_n) \quad \}$$

=Definition $\bowtie$

$$\mathrm{eval}_{Al}(\Phi^*)(d, r_1, \ldots, r_n) \quad \bowtie \quad \mathrm{eval}_{Al}(\Psi^*)(d, r_1, \ldots, r_n)$$

=Definition eval_{Al}

$$\mathrm{eval}_{Al}(\; \Phi^* \bowtie \Psi^* \;)(d, r_1, \ldots, r_n)$$

entsprechend beweist man:

- $\text{eval}_K(\ \Phi \vee \Psi\ )(d,r_1,\ldots,r_n) = \text{eval}_{Al}(\ \Phi^* + \Psi^*\ )(d,r_1,\ldots,r_n)$

- $\text{eval}_K(\ \neg\, \Phi\ )(d,r_1,\ldots,r_n) = \text{eval}_{Al}(\ \gamma\,(\Phi^*)\ )(d,r_1,\ldots,r_n)$

3. (Induktionsschritt für Existenzquantor)

$\text{eval}_K\,(\ (\exists\, v_{Ai})\ \Phi\ )\,(d, r_1, \ldots, r_n)$

$=$Definition eval_K

$$\{\ \mu\ \mid\ \mu : \text{dom}\ \Phi \setminus \{Ai\} \to d,\quad \models_{(d, r_1, \ldots, r_n),\,\mu}\ (\exists\, v_{Ai})\ \Phi\ \}$$

$=$Definition $\models$

$$\{\ \mu\ \mid\ \mu : \text{dom}\ \Phi \setminus \{Ai\} \to d,\quad \text{es gibt}\ \mu'\ \text{mit}\ \mu' : \text{dom}\ \Phi \to d,\ \mu =_{Ai} \mu'\ \text{und}\ \models_{(d, r_1, \ldots, r_n),\,\mu'}\ \Phi\ \}$$

$=$

$$\{\ \mu\ \mid\ \mu : \text{dom}\ \Phi \setminus \{Ai\} \to d,\quad \text{es gibt}\ \mu'\ \text{mit}\ \mu' : \text{dom}\ \Phi \to d,\ \mu = \mu' \lceil \text{dom}\ \Phi \setminus \{Ai\}\ \text{und}\ \models_{(d, r_1, \ldots, r_n),\,\mu'}\ \Phi\ \}$$

... (siehe nächste Seite)

$$\{ \quad \mu \quad | \quad \mu : \mathrm{dom}\,\Phi \setminus \{Ai\} \rightarrow d, \quad \text{es gibt } \mu' \text{ mit } \mu' : \mathrm{dom}\,\Phi \rightarrow d,$$
$$\mu = \mu' \lceil \mathrm{dom}\,\Phi \setminus \{Ai\} \quad \text{und}$$
$$\models_{(d, r_1, \ldots, r_n), \mu'} \Phi \qquad \}$$

=Definition eval_K

$$\{ \quad \mu \quad | \quad \mu : \mathrm{dom}\,\Phi \setminus \{Ai\} \rightarrow d, \quad \text{es gibt } \mu' \text{ mit } \mu' : \mathrm{dom}\,\Phi \rightarrow d,$$
$$\mu = \mu' \lceil \mathrm{dom}\,\Phi \setminus \{Ai\} \quad \text{und}$$
$$\mu' \in \mathrm{eval}_K(\Phi)(d, r_1, \ldots, r_n) \qquad \}$$

=Induktionsannahme

$$\{ \quad \mu \quad | \quad \mu : \mathrm{dom}\,\Phi \setminus \{Ai\} \rightarrow d, \quad \text{es gibt } \mu' \text{ mit } \mu' : \mathrm{dom}\,\Phi \rightarrow d,$$
$$\mu = \mu' \lceil \mathrm{dom}\,\Phi \setminus \{Ai\} \quad \text{und}$$
$$\mu' \in \mathrm{eval}_{Al}(\Phi^*)(d, r_1, \ldots, r_n) \qquad \}$$

=Definition π

$$\pi_{\mathrm{dom}\,\Phi \setminus \{Ai\}} \left(\mathrm{eval}_{Al}(\Phi^*)(d, r_1, \ldots, r_n) \right)$$

=Definition eval_{Al}

$$\mathrm{eval}_{Al} \left(\pi_{\mathrm{dom}\,\Phi \setminus \{Ai\}}(\Phi^*) \right)(d, r_1, \ldots, r_n)$$

entsprechend beweist man:

$$\mathrm{eval}_K\left(\left(\forall v_{Ai}\right) \Phi \right)(d, r_1, \ldots, r_n)$$

$=$

$$\mathrm{eval}_{Al}(\Phi^*)(d, r_1, \ldots, r_n) \;/\; \mathrm{eval}_{Al}\left(\pi_{qi}(\mathrm{D}) \right)(d, r_1, \ldots, r_n)$$

mit $qi : \{Ai\} \rightarrow \{\mathrm{D}\}$

$=$

$$\mathrm{eval}_{Al}\left(\chi^* \right)(d, r_1, \ldots, r_n)$$

wobei man χ^* aus Φ^* und $\pi_{qi}(\mathrm{D})$ konstruiert

entsprechend der Simulation der Division
vermöge Projektion, Komplement und Verbund

Behauptung 2: Algebra in Kalkül äquivalent übersetzbar

zu zeigen: für alle Ausdrücke $\chi \in \boldsymbol{L}_{Al}$

gibt es eine Formel $\chi^* \in \boldsymbol{L}_K$ mit $\text{eval}_{Al}(\chi) = \text{eval}_K(\chi^*)$

Beweis: wir konstruieren χ^* durch Induktion über den Aufbau von χ

1. $R_i^* \equiv R_i(\ Ai_1 : v_{Ai_1}, \ldots, Ai_k : v_{Ai_k}\ )$

$D^* \equiv (\ v_D = v_D\ )$

2. $(\Phi \bowtie \Psi)^* \equiv (\Phi^* \wedge \Psi^*)$

$(\Phi + \Psi)^* \equiv (\Phi^* \vee \Psi^*)$

$(\sigma_{Ai=Aj}(\Phi))^* \equiv (\Phi^* \wedge (v_{Ai} = v_{Aj}))$

$(\sigma_{Ai\neq Aj}(\Phi))^* \equiv (\Phi^* \wedge \neg(v_{Ai} = v_{Aj}))$

$(\gamma(\Phi))^* \equiv (\neg \Phi^*)$

Beispiel für Konstruktion von $(\pi_q (\Phi))^*$

Voraussetzungen:

$\Phi \equiv R$ mit $\text{dom}\, R = \{A1, A2, A3, A4\}$

$q : \{A1, A2, A5\} \rightarrow \{A1, A2, A3, A4\}$ mit $q\,(A1) := A1$, $q\,(A2) := A3$, $q\,(A5) := A1$

Formel für das Argument von π_q, d.h. für R:

$R\,(\ A1 : v_{A1}\ ,\ A2 : v_{A2}\ ,\ A3 : v_{A3}\ ,\ A4 : v_{A4}\ )$

$\{ A2 , A4 \}$ = dom R \ range q:

entferne die durch $A2$, $A4$ bezeichneten Spalten:

$(\exists\, v_{A2})\,(\exists\, v_{A4})\, R\,(\, A1 : v_{A1}\,, A2 : v_{A2}\,, A3 : v_{A3}\,, A4 : v_{A4}\,)$

$q(A2) = A3$:

benenne die durch $A3$ bezeichnete Spalte in $A2$ um

(führe Variable v_{A2} durch ein Gleichheitsatom ein und "enferne $A3$"):

$(\exists\, v_{A3})\,(\exists\, v_{A9})\,(\exists\, v_{A4})\,(\, R\,(\, A1 : v_{A1}\,, A2 : v_{A9}\,, A3 : v_{A3}\,, A4 : v_{A4}\,) \ \wedge\ (v_{A3} = v_{A2})\,)$

... (siehe nächste Seite)

$$(\exists\, v_{A3})\,(\exists\, v_{A9})\,(\exists\, v_{A4})\,(\,R\,(\,A1 : v_{A1}\,,\,A2 : v_{A9}\,,\,A3 : v_{A3}\,,\,A4 : v_{A4}\,)\ \wedge\ (v_{A3} = v_{A2})\,)$$

$q(A1) = q(A5) = A1$:

verdopple die durch $A1$ bezeichnete Spalte;
benenne Verdopplungen durch $A1$ bzw. $A5$:

$$(\exists\, v_{A8})\,(\exists\, v_{A3})\,(\exists\, v_{A9})\,(\exists\, v_{A4})$$

$$[\quad R\,(A1 : v_{A8}, A2 : v_{A9}, A3 : v_{A3}, A4 : v_{A4}\,)\ \wedge\ (v_{A3} = v_{A2})$$

$$\wedge\,(v_{A8} = v_{A1})\ \wedge\ (v_{A8} = v_{A5})$$

$$]$$

Beispiel für Übersetzung “Algebra in Kalkül”

- **Voraussetzungen**

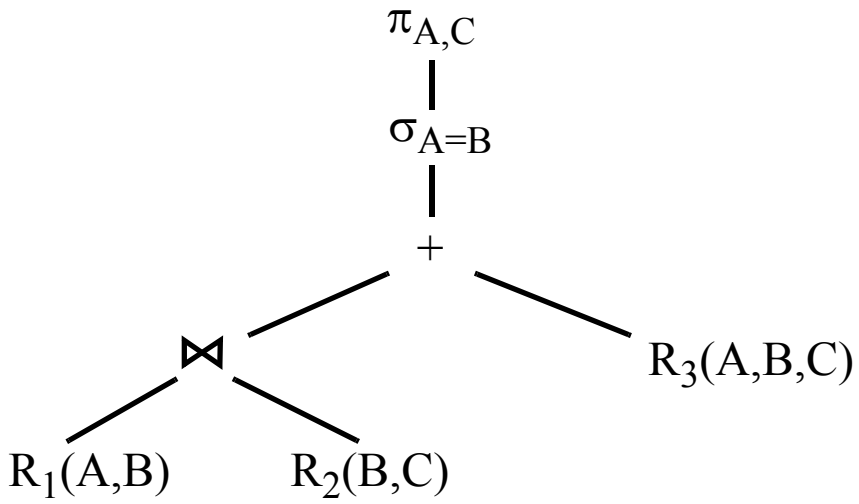
  Datenbankschema:

  $$RS = < < R_1 \mid \{A,B\} \mid >, < R_2 \mid \{B,C\} \mid >, < R_3 \mid \{A,B,C\} \mid > \mid \mid >$$

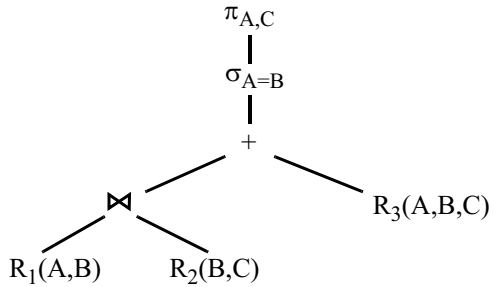
  relationaler Ausdruck:

  $$\chi = \pi_{A,C}(\sigma_{A=B}((R_1 \bowtie R_2) + R_3))$$

- **Syntaxbaum des Ausdrucks**



- **Übersetzungen**, entsprechend Baumdurchlauf:



$R_1^* = R_1(A{:}v_A, B{:}v_B)$

$R_2^* = R_2(B{:}v_B, C{:}v_C)$

$(R_1 \bowtie R_2)^* = R_1(A{:}v_A, B{:}v_B) \wedge R_2(B{:}v_B, C{:}v_C)$

$R_3^* = R_3(A{:}v_A, B{:}v_B, C{:}v_C)$

$((R_1 \bowtie R_2) + R_3)^* = (R_1(A{:}v_A, B{:}v_B) \wedge R_2(B{:}v_B, C{:}v_C)) \vee R_3(A{:}v_A, B{:}v_B, C{:}v_C)$

$(\sigma_{A=B}((R_1 \bowtie R_2) + R_3)^*$
$= ((R_1(A{:}v_A, B{:}v_B) \wedge R_2(B{:}v_B, C{:}v_C)) \vee R_3(A{:}v_A, B{:}v_B, C{:}v_C)) \wedge (v_A = v_B)$

$(\pi_{A,C}(\sigma_{A=B}((R_1 \bowtie R_2) + R_3))^*$
$= (\exists v_B)((R_1(A{:}v_A, B{:}v_B) \wedge R_2(B{:}v_B, C{:}v_C)) \vee R_3(A{:}v_A, B{:}v_B, C{:}v_C)) \wedge (v_A = v_B))$

Zusammenfassung: Relationenalgebra und logische Verknüpfungen

natürlicher Verbund

$$r \bowtie s := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \to \mathbf{C}, \quad \mu\lceil \mathrm{dom}\, r \in r \quad \textbf{und} \quad \mu\lceil \mathrm{dom}\, s \in s \}$$

Projektion

$$\pi_q(r) := \{ \mu \mid \mu : X \to \mathbf{C}, \quad \textbf{es gibt } \nu \in r \text{ mit } \mu = \nu \circ q \}$$

Vergleich

$$\sigma_{A=B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) = \mu(B) \}$$

$$\sigma_{A\neq B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) \neq \mu(B) \}$$

Vereinigung

$$+(d,r,s) := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \to d, \quad \mu\lceil \mathrm{dom}\, r \in r \quad \textbf{oder} \quad \mu\lceil \mathrm{dom}\, s \in s \}$$

Komplement

$$\gamma(d,r) := \{ \mu \mid \mu : \mathrm{dom}\, r \to d, \quad \mu \notin r \quad \text{(d.h. } \mu \text{ ist } \textbf{nicht} \text{ Element von } r) \}$$

Differenz

$$-(d,r,s) := \{ \mu \mid \mu \in r, \quad \mu\lceil \mathrm{dom}\, r \cap \mathrm{dom}\, s \notin \pi_{\mathrm{dom}\, r \cap \mathrm{dom}\, s}(s) \}$$

Division

$$r / s := \{ \mu \mid \mu : \mathrm{dom}\, r \setminus \mathrm{dom}\, s \to \mathbf{C}, \quad \textbf{für alle } \nu : \text{wenn } \nu \in \pi_{\mathrm{dom}\, r \cap \mathrm{dom}\, s}(s), \text{ dann } \mu \cup \nu \in r \}$$

5 Structured Query Language - SQL

relationale Anfragen und relationale Anfragesprachen (Zusammenfassung)

- relationale Anfragen
  - Operationen auf ***Instanzen*** von Schemas
  - behandeln Relationen als ***Mengen*** von Tupeln
  - behandeln Konstantenzeichen als ***atomare, uninterpretierte*** Dinge
  - formal: *getypt*, *universumstreu*, *berechenbar*, *isomorphietreu*

- relationale Anfragesprache

  formale Sprache $\boldsymbol{L}$:
  für alle $\Phi \in \boldsymbol{L}$ liefert die Semantik $\mathrm{eval}(\Phi)$ eine relationale Anfrage

- Beispiele
  - Relationenalgebra
  - Relationenkalkül
  - Kern der **Structured Query Language (SQL)**:
    ein tupelorientierter Relationenkalkül mit algebraischen Ergänzungen

5.1 Kern von Structured Query Language (SQL): Theorie

Structured Query Language, SQL

- genormte **Datendefinitions- und Datenmanipulationssprache**

- enthält eine *relationale Anfragesprache* als Teilsprache

- aus theoretischer Sicht:
  - *tupel*orientierter Relationen*kalkül*
  - mit einigen *algebraischen* Elementen
  - angereichert durch *arithmetische* und
  - *textverarbeitende* Elemente
  - sowie mit vielen weiteren *Diensten*

- in praktischen Ausprägungen:

  sehr umfassende Programmiersysteme für
  Informationssysteme aller Art in vielfältigen Anwendungen und Umgebungen

SQL-Produkte

- weitverbreitete Produkte verfügbar
- kommerziell: **Oracle** (für den privaten Gebrauch frei verfügbar), DB2, ...
- Open-Source: MySQL (auch kommerzielle Lizenz verfügbar), PostgreSQL, ...
- Web-Links (Stand: 11.07.2005):
  - *Oracle:* http://www.oracle.com
    http://www.oracle.com/technology/software/products/database/oracle10g/index.html
  - *DB2:* http://www-306.ibm.com/software/data/db2/
  - *MySQL:* http://www.mysql.com
    http://dev.mysql.com/downloads/
  - *PostgreSQL:* http://www.postgresql.org

Ansatz für Grundform des Kalkülteils

1. Zugriff auf gespeicherte Daten

Semantik: erzeuge alle Kombinationen von Tupeln μ_1 aus $R_1,\ldots,\mu_k$ aus R_k
(wobei die R_i nicht notwendig verschieden sein müssen)

Syntax: binde Tupelvariable μ_1 an Relationensymbol $R_1,\ldots,$
Tupelvariable μ_k an Relationensymbol R_k

2. Auswahl zugegriffener Daten

Semantik: wähle diejenigen Kombinationen aus, für die eine Formel Φ mit Prädikatenzeichen für Komponentenvergleiche wie $=$ oder $\neq$ gültig ist

Syntax: setze aus Tupelvariablen und passenden Attributen gebildete Attributterme der Form $\mu.A$ in aussagenlogische Kombination von Prädikatenzeichen für Vergleiche von Werten ein

3. Konstruktion der Ausgabedaten

Semantik: liefere dann die Komponenten $\mu_{i1}(A_1),\ldots,\mu_{il}(A_l)$ dieser Kombinationen

Syntax: bilde Liste der bezeichnenden Attributterme $\mu_{i1}.A_1,\ldots,\mu_{il}.A_l$

Ansatz beschreibt Dreischrittverfahren

- **algebraisch:** $\pi_{\{R_{i1}.A_1, \ldots, R_{il}.A_l\}} (\sigma_\Phi (R_1 \times \ldots \times R_k))$

- **in der SQL-Syntax:**
  - Teil 1 [Zugriff]: durch `FROM` eingeleitet
  - Teil 2 [Auswahl]: durch `WHERE` eingeleitet
  - Teil 3 [Konstruktion]: durch `SELECT` eingeleitet, aber vorangestellt

entsprechender SQL-Ausdruck:

`SELECT DISTINCT` $\mu_{i1}.A_1$,..., $\mu_{il}.A_l$

`FROM` R_1 μ_1 ,..., R_k μ_k

`WHERE` Φ

einige Vereinfachungen

- anstatt in der FROM-Klausel
  durch "R_i μ_i" eine Tupelvariable μ_i an das Relationensymbol R_i zu binden:

  das Relationensymbol selbst wie eine Tupelvariable benutzen und
  die entsprechende Bindung "R_i μ_i" zu "R_i" abkürzen

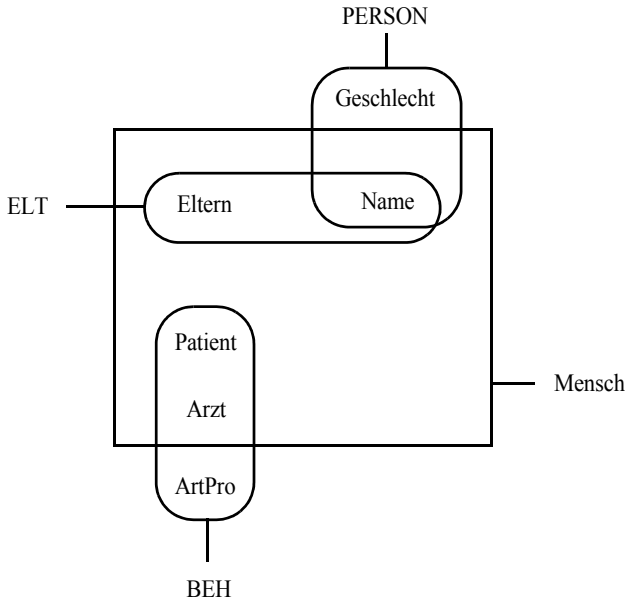
- anstatt eine Komponente $\mu(A)$
  durch den entsprechenden vollständigen Attributterm $\mu.A$ zu bezeichnen:

  nur das Attribut A verwenden,
  wenn die fehlende Tupelvariable eindeutig ergänzt werden kann

- anstatt einer Attributtermliste in der SELECT-Klausel,
  die *alle* entsprechend der FROM-Klausel bildbaren Attributterme enthält:

  eine solche Attributtermliste durch "*" abkürzen

Beispiel



< PERSON | { Name , Geschlecht } | >
< BEH | { Patient , Arzt , ArtPro } | >
< ELT | { Name , Eltern } | >

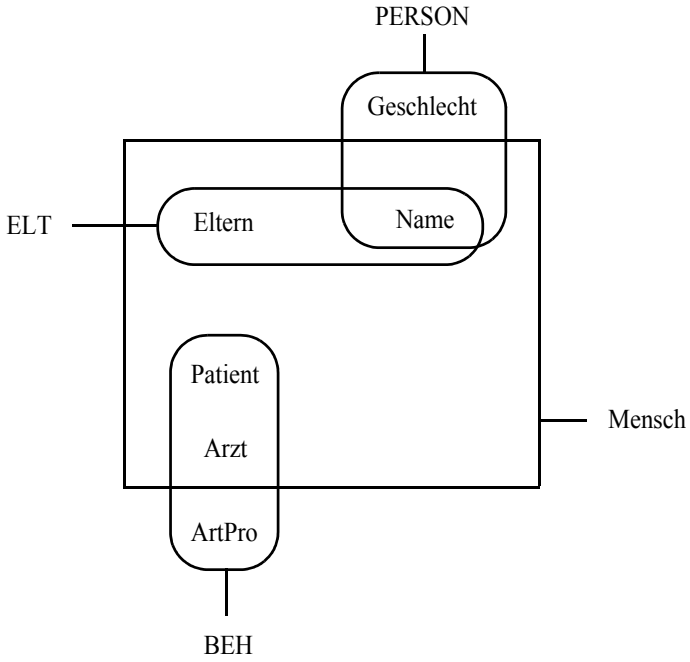
Anfragewunsch: “Bestimme für Kind `theresia` Geschlecht und Eltern!”

Ausdruck der Relationenalgebra: $\sigma_{Name=theresia}$ (PERSON $\bowtie$ ELT)

SQL-Anfrage:

```
SELECT  DISTINCT  *
   FROM  PERSON , ELT
      WHERE   PERSON.Name = ELT.Name
              AND
              PERSON.Name = 'theresia'
```

Beispiel



< PERSON | { Name , Geschlecht } | >
< BEH | { Patient , Arzt , ArtPro } | >
< ELT | { Name , Eltern } | >

Anfragewunsch: “Bestimme alle Ärzte, die weibliche Patienten behandeln!”

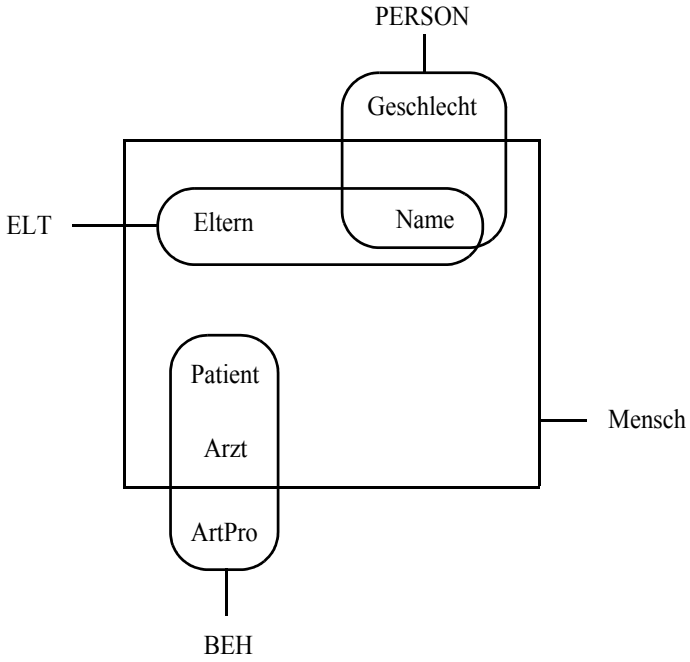
Ausdruck der Relationenalgebra:

$$\pi_{Arzt}(\sigma_{Geschlecht = weib}(\sigma_{Patient = Name} (BEH \bowtie PERSON)))$$

SQL-Anfrage:

```
SELECT DISTINCT  Arzt
    FROM   BEH , PERSON
       WHERE  BEH.Patient = PERSON.Name
              AND  PERSON.Geschlecht = 'weib'
```

Beispiel



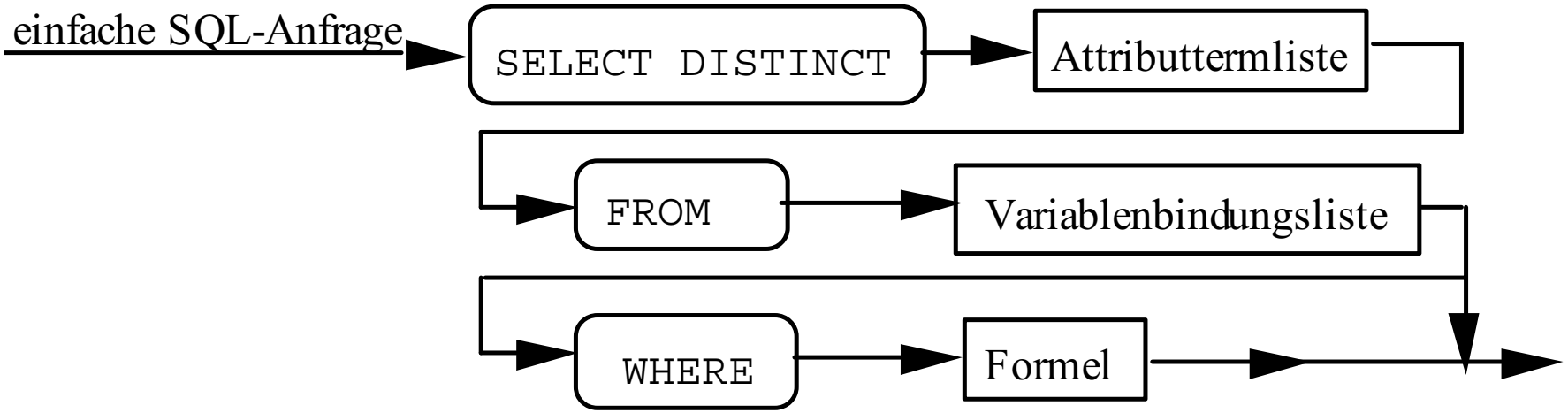
< PERSON | { Name , Geschlecht } | >
< BEH | { Patient , Arzt , ArtPro } | >
< ELT | { Name , Eltern } | >

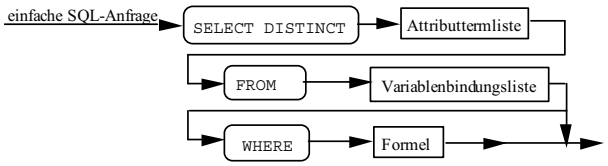
Anfragewunsch: “Bestimme die Großeltern des Kindes `theresia`!”

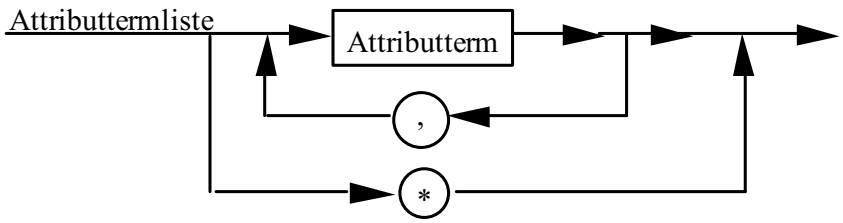
SQL-Anfrage:

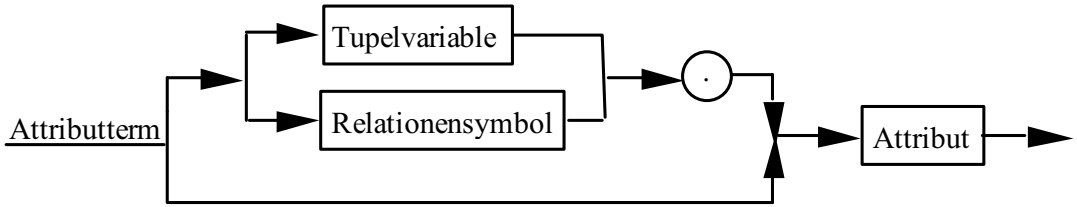
```
SELECT  GRAD2.Eltern
   FROM   ELT GRAD1  , ELT GRAD2
      WHERE  GRAD1.Eltern = GRAD2.Name
             AND
             GRAD1.Name = 'theresia'
```

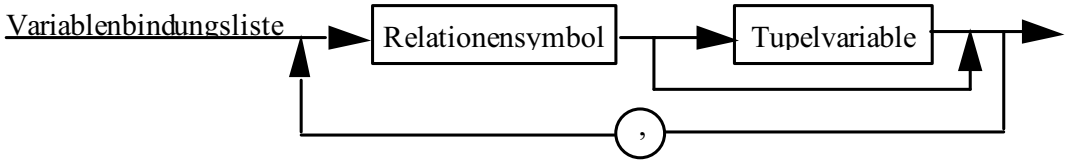
(stark vereinfachte, abstrakte) Syntax der Kalkülteile von SQL

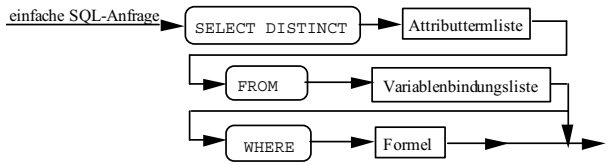


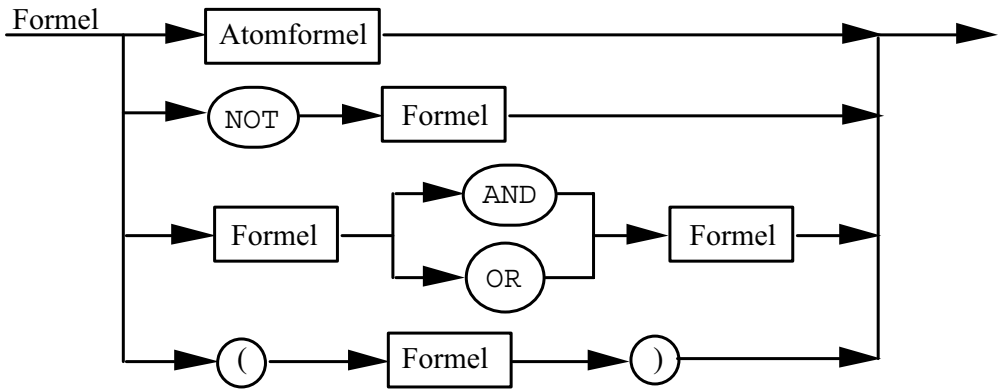


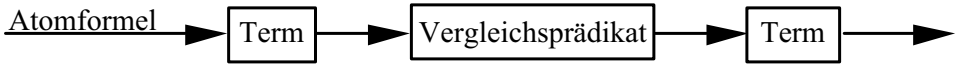

Attributtermliste
Attributterm
,
*

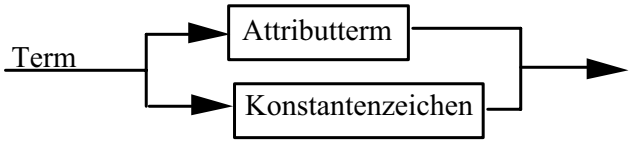

Tupelvariable
Attributterm
Relationensymbol
.
Attribut

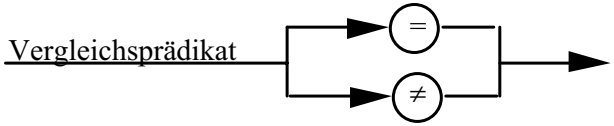

Variablenbindungsliste
Relationensymbol
Tupelvariable
,


einfache SQL-Anfrage
SELECT DISTINCT
Attributtermliste
FROM
Variablenbindungsliste
WHERE
Formel

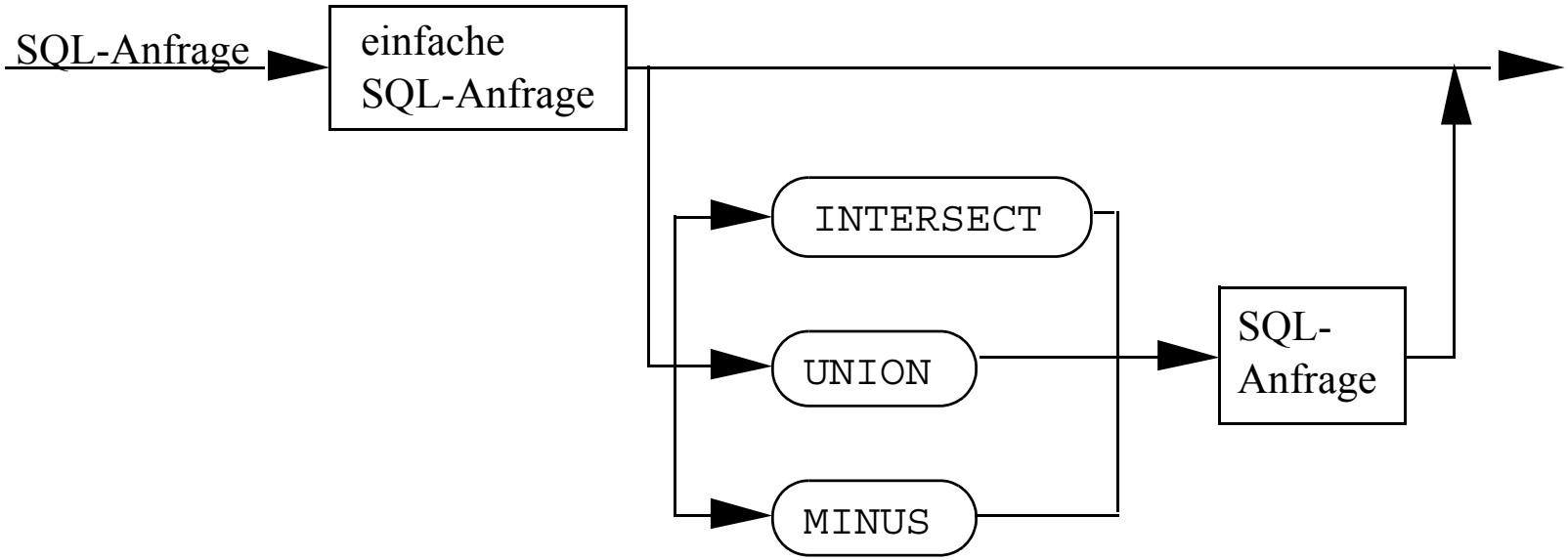

Formel
Atomformel
NOT
Formel
Formel
AND
OR
Formel
(
Formel
)


Atomformel
Term
Vergleichsprädikat
Term


Term
Attributterm
Konstantenzeichen


Vergleichsprädikat
=
≠

(vereinfachte, abstrakte) Syntax des algebraischen Teils von SQL



Übersicht über Sprachmittel von SQL

- Kalkülteil

  beschreibt *Dreischrittverfahren* für "Zugriff-Auswahl-Konstruktion"

- *algebraischer* Teil

  verknüpft durch den Kalkülteil gebildete Ergebnisse *mengentheoretisch*

- *arithmetischer* Teil
  - *Grundrechenarten*
  - *arithmetischen Vergleichsoperationen*
  - *Aggregatfunktionen*

- *textverarbeitender* Teil

  behandelt Konstantenzeichen als *nichtatomare Zeichenfolgen*

- weitere Sprachmittel
  - SQL-Anfragen ineinander *schachteln*
  - in Ergebnissen *Entfernung von Duplikaten unterdrücken*
  - Ergebnisse weiter aufbereiten
  - ...

- Relationenalgebra für “universums-unabhängige” Anfragen ausdrückbar:

natürlicher Verbund

allgemein:

$$r \bowtie s := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \to \mathbf{C},\ \mu\lceil \mathrm{dom}\, r \in r \text{ und } \mu\lceil \mathrm{dom}\, s \in s \}$$

$$\{ \mu \mid \text{es gibt } \alpha \in r,\ \beta \in s : \alpha(A) = \beta(A) \text{ für alle } A \in \mathrm{dom}\, r \cap \mathrm{dom}\, s\ ,\ \mu = \alpha \cup \beta \}$$

äquivalent umgeschrieben und speziell für Relationenschemas

$< R \mid \{A_1, \ldots, A_k, B_1, \ldots, B_l\} \mid >$ und $< S \mid \{B_1, \ldots, B_l, C_1, \ldots, C_m\} \mid >$:

$$r \bowtie s := \{ \mu \mid \text{es gibt } \alpha \in r,\ \beta \in s : \alpha(B_j) = \beta(B_j) \text{ für } j = 1, \ldots, l\ ,\ \mu = \alpha \cup \beta \}$$

in SQL:

```
SELECT  DISTINCT  R.A1 , ... , R.Ak ,
                  R.B1 , ... , R.Bl ,
                  S.C1 , ... , S.Cm
FROM    R , S
WHERE   R.B1=S.B1  AND  R.B2=S.B2  AND  ...  AND  R.Bl=S.Bl
```

A=c-Selektion

allgemein für Relationenschema $< R \mid X \mid >$ mit $A \in X$:

$$\sigma_{A=c}(r) := \{ \mu \mid \mu : \mathrm{dom}\, r \rightarrow \mathbf{C}, \quad \mu \in r \text{ und } \mu(A) = c \}$$

speziell in SQL:

```
SELECT  DISTINCT   *

   FROM   R

      WHERE   A = 'c'
```

Projektion

allgemein:

$$\pi_X(r) := \{ \mu \mid \mu : X \cap \mathrm{dom}\, r \to \mathbf{C}, \quad \text{es gibt } \nu \in r \text{ mit } \mu = \nu\lceil X \}$$

speziell für Relationenschema $< R \mid Y \mid >$ mit $X = \{ A_1, \ldots, A_k \} \subset Y$:

$$\pi_X(r) := \{ \mu \mid \mu : \{A_1, \ldots, A_k\} \to \mathbf{C}, \quad \text{es gibt } \nu \in r \text{ mit } \mu = \nu\lceil \{A_1, \ldots, A_k\} \}$$

speziell in SQL:

```
SELECT  DISTINCT    A1 , ... , Ak
   FROM    R
```

Vereinigung

allgemein:

$+(d,r,s) := \{ \ \mu \mid \ \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \to d, \ \mu\lceil \mathrm{dom}\, r \in r \ \text{ oder } \ \mu\lceil \mathrm{dom}\, s \in s \ \}$

speziell für vereinigungsverträgliche Relationenschemas
$< R \mid X \mid >$ und $< S \mid X \mid >$,
d.h. insbesondere $X = \mathrm{dom}\, r = \mathrm{dom}\, s = \mathrm{dom}\, r \cup \mathrm{dom}\, s$:

$+(d,r,s) = r \cup s$

speziell in SQL:

```
   SELECT  DISTINCT   *
   FROM    R

UNION

   SELECT  DISTINCT   *
   FROM    S
```

A=B-Vergleich und A≠B-Vergleich

allgemein für Relationenschema $< R \mid X \mid >$ mit $\{ A , B \} \subset$ X:

$$\sigma_{A=B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) = \mu(B) \}$$

$$\sigma_{A\neq B}(r) := \{ \mu \mid \mu \in r, \quad \mu(A) \neq \mu(B) \}$$

in SQL:

```
SELECT  DISTINCT   *
   FROM   R
      WHERE   A = B
```

```
SELECT  DISTINCT   *
   FROM   R
      WHERE   A ≠ B
```

Differenz

allgemein:

$$-(d,r,s) := \{ \mu \mid \mu \in r, \quad \mu\lceil \mathrm{dom}\, r \cap \mathrm{dom}\, s \notin \pi_{\mathrm{dom}\, r \cap \mathrm{dom}\, s}(s) \quad \}$$

speziell für vereinigungsverträgliche Relationenschemas
$< R \mid X \mid >$ und $< S \mid X \mid >$,
d.h. insbesondere $X = \mathrm{dom}\, r = \mathrm{dom}\, s = \mathrm{dom}\, r \cup \mathrm{dom}\, s$:

$$-(d,r,s) = \{ \mu \mid \mu \in r, \quad \mu \notin s \quad \}$$

speziell in SQL:

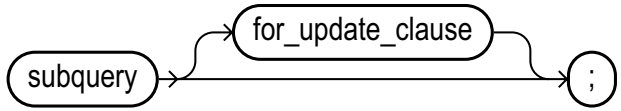
```
  SELECT   DISTINCT     *     FROM    R

MINUS

  SELECT   DISTINCT     *     FROM    S
```

5.2 Structured Query Language (SQL): Praxis mit Oracle

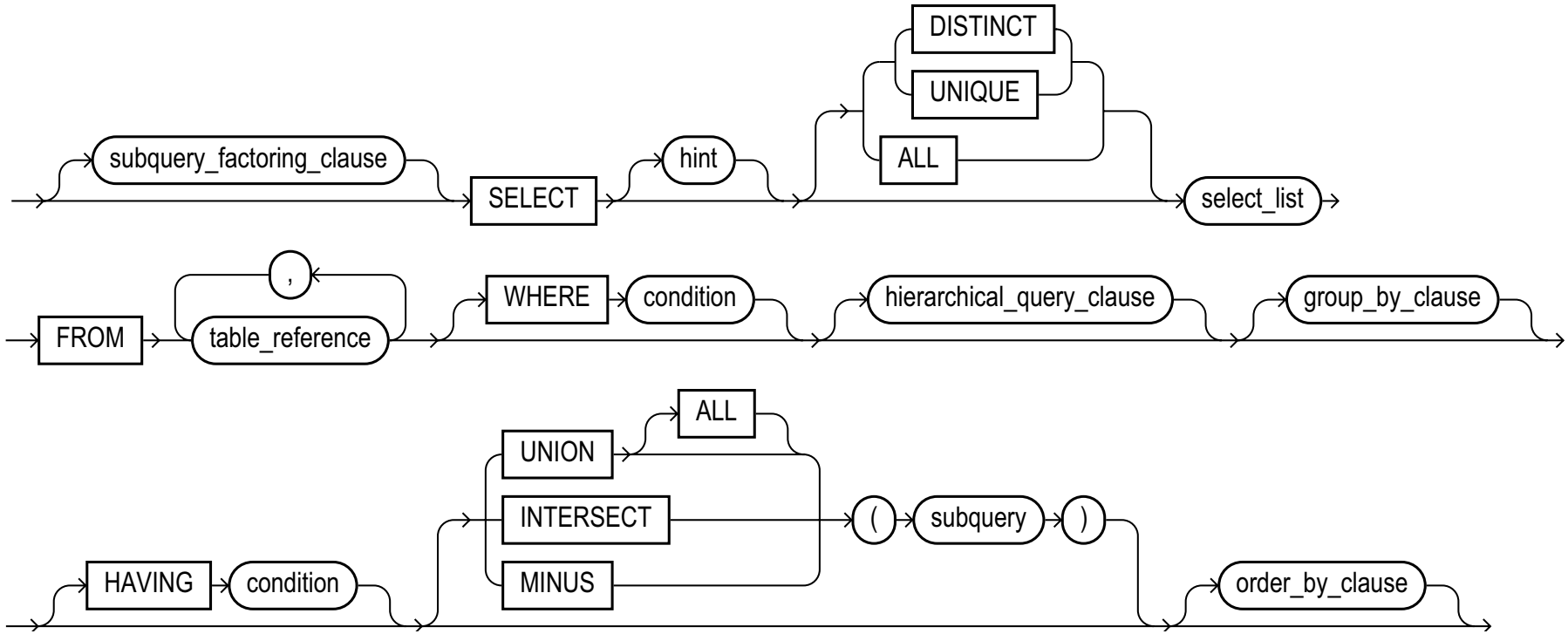
Oracle-SQL Syntax für “SELECT”

Oracle 9*i* - SQL Reference, Release 2 (9.2), March 2002, Part No.A96540-01, pp. 8-1 bis 8-16

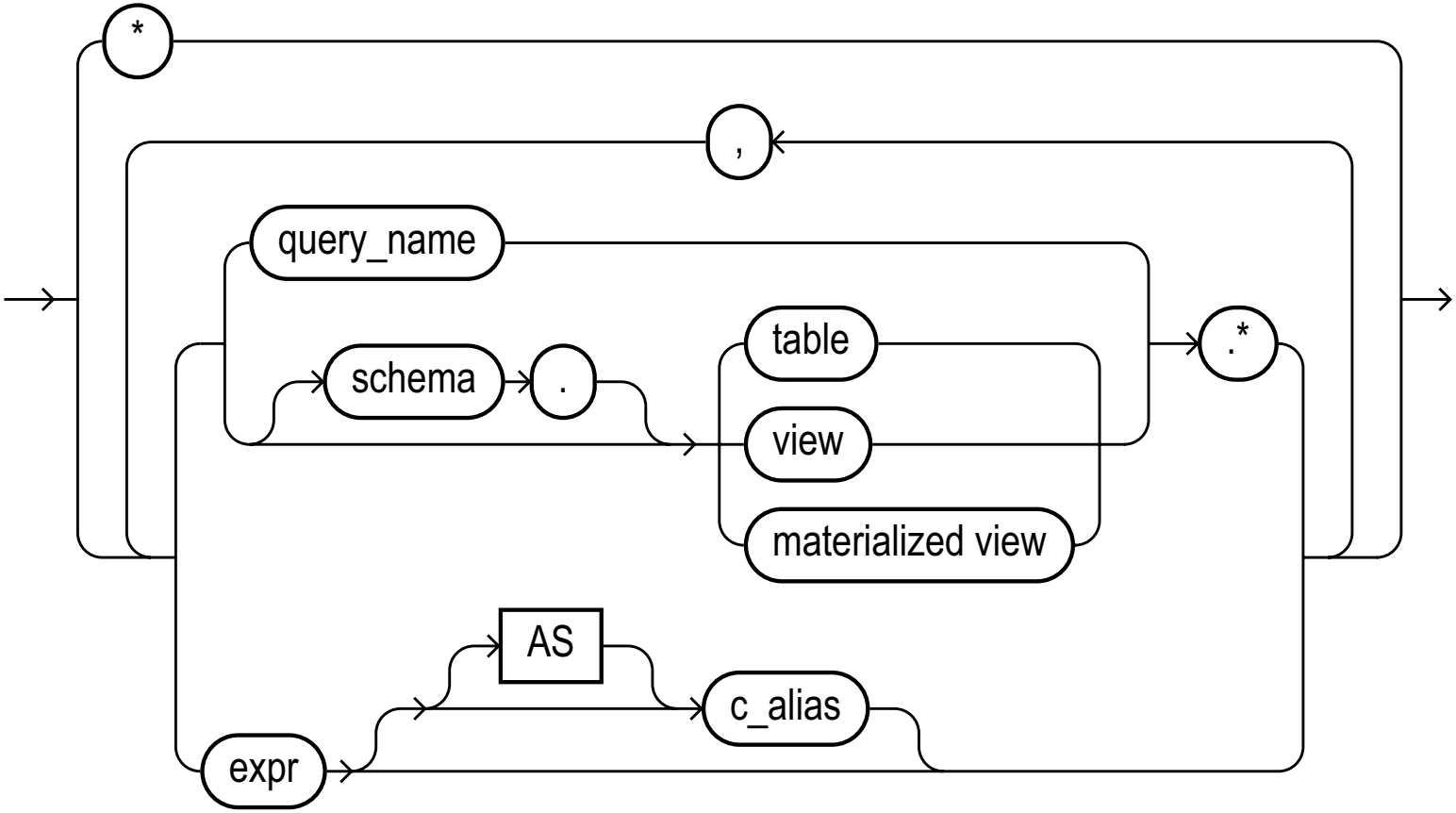
select::=



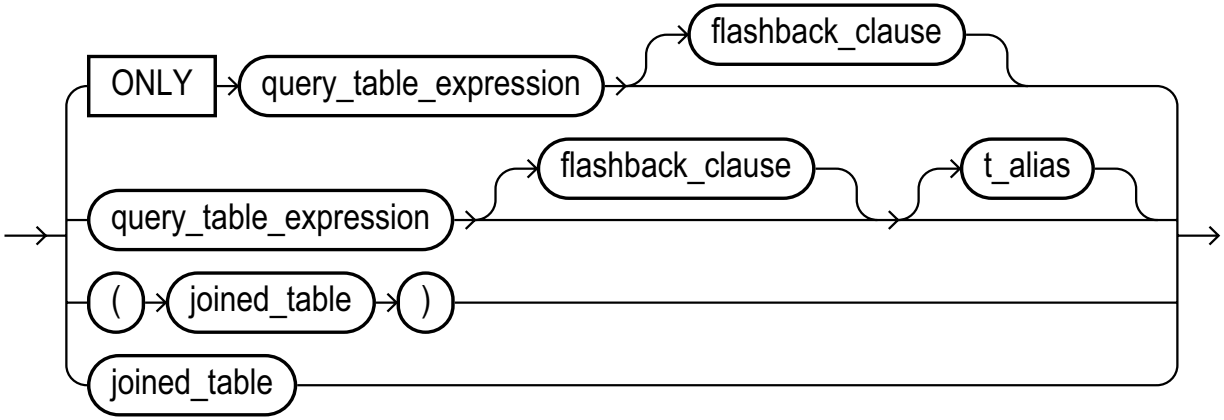
subquery::=



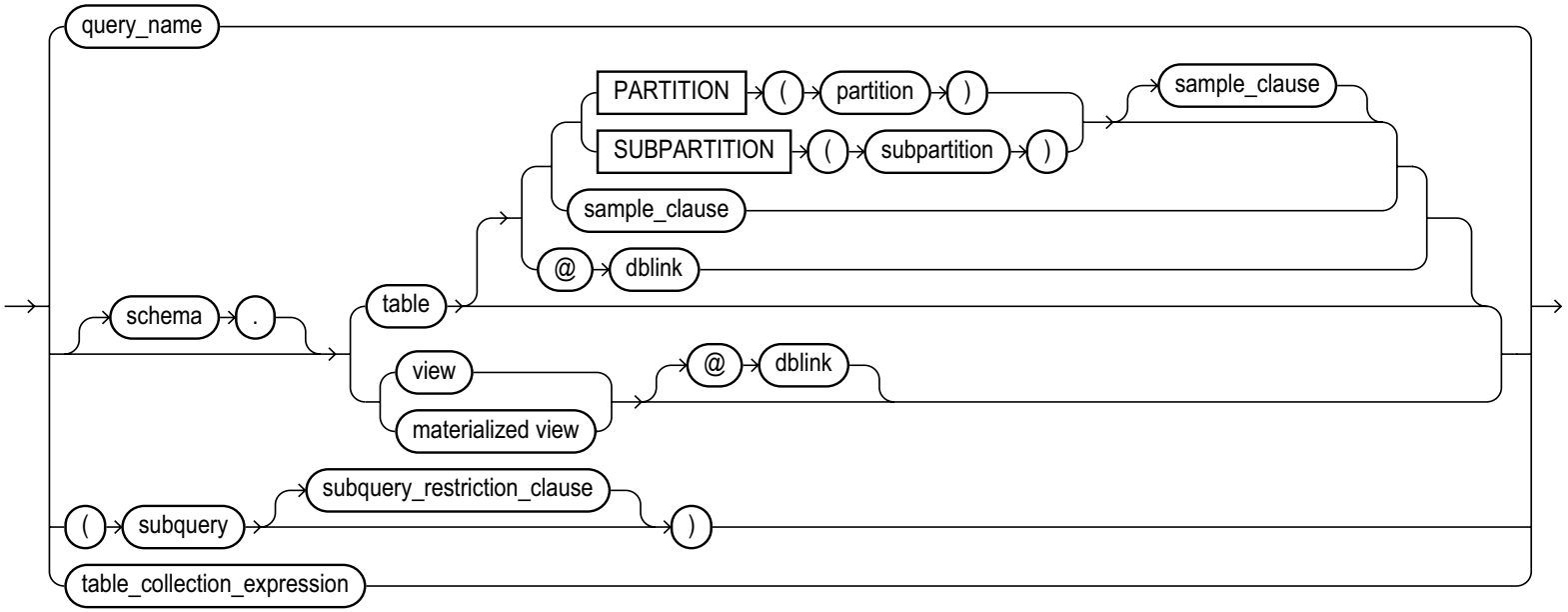
select_list::=


*
,
query_name
schema
.
table
view
materialized view
.*
AS
c_alias
expr

table_reference::=

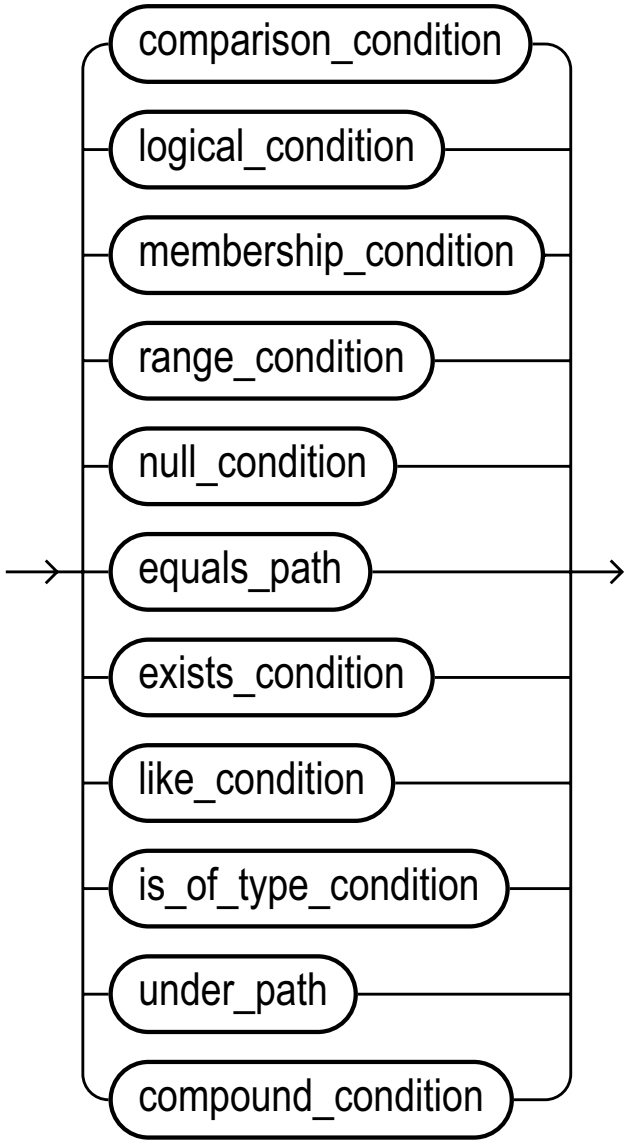


query_table_expression::=



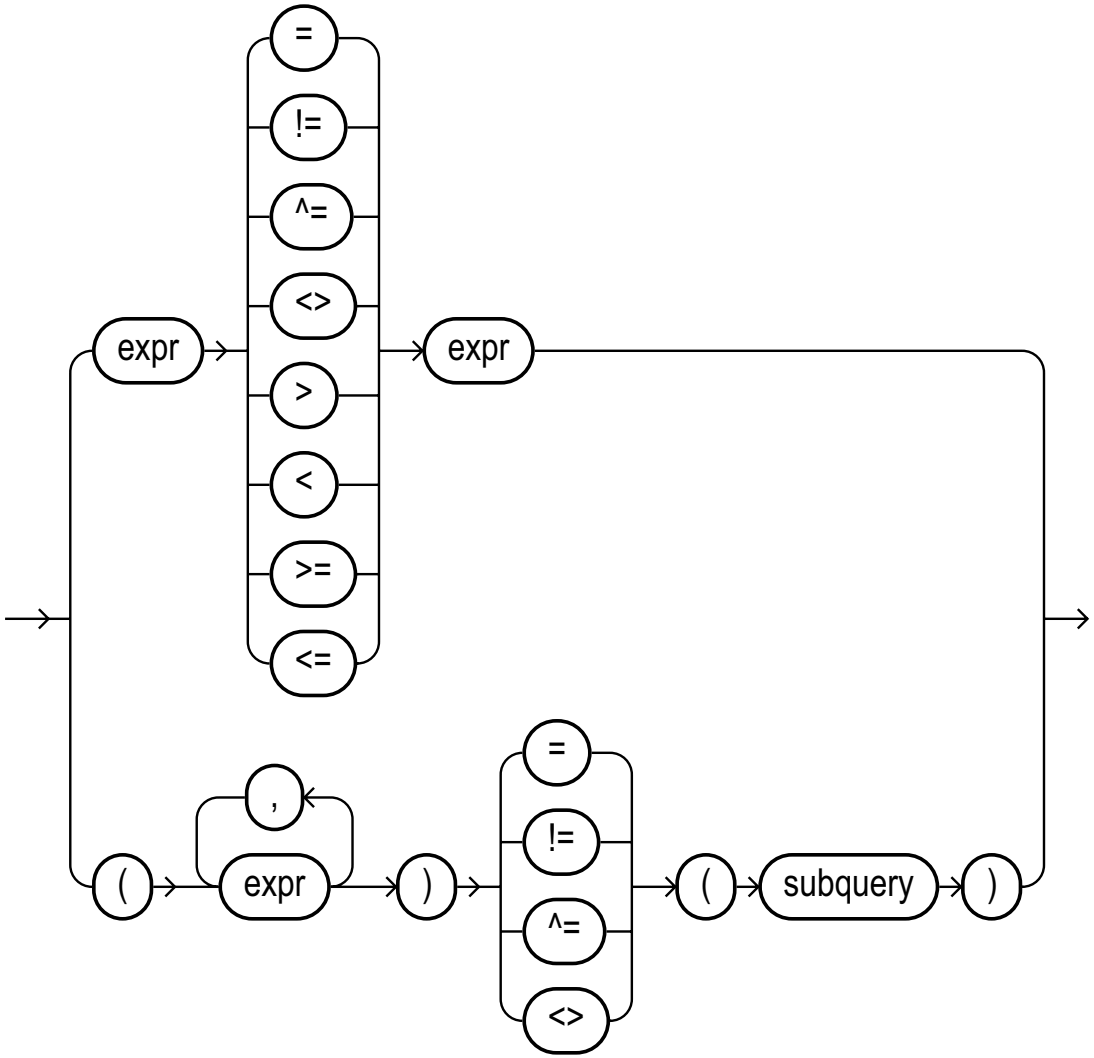
Oracle 9*i* - SQL Reference, Release 2 (9.2), March 2002, Part No.A96540-01, pp. 5-1 bis 5-22

condition::=

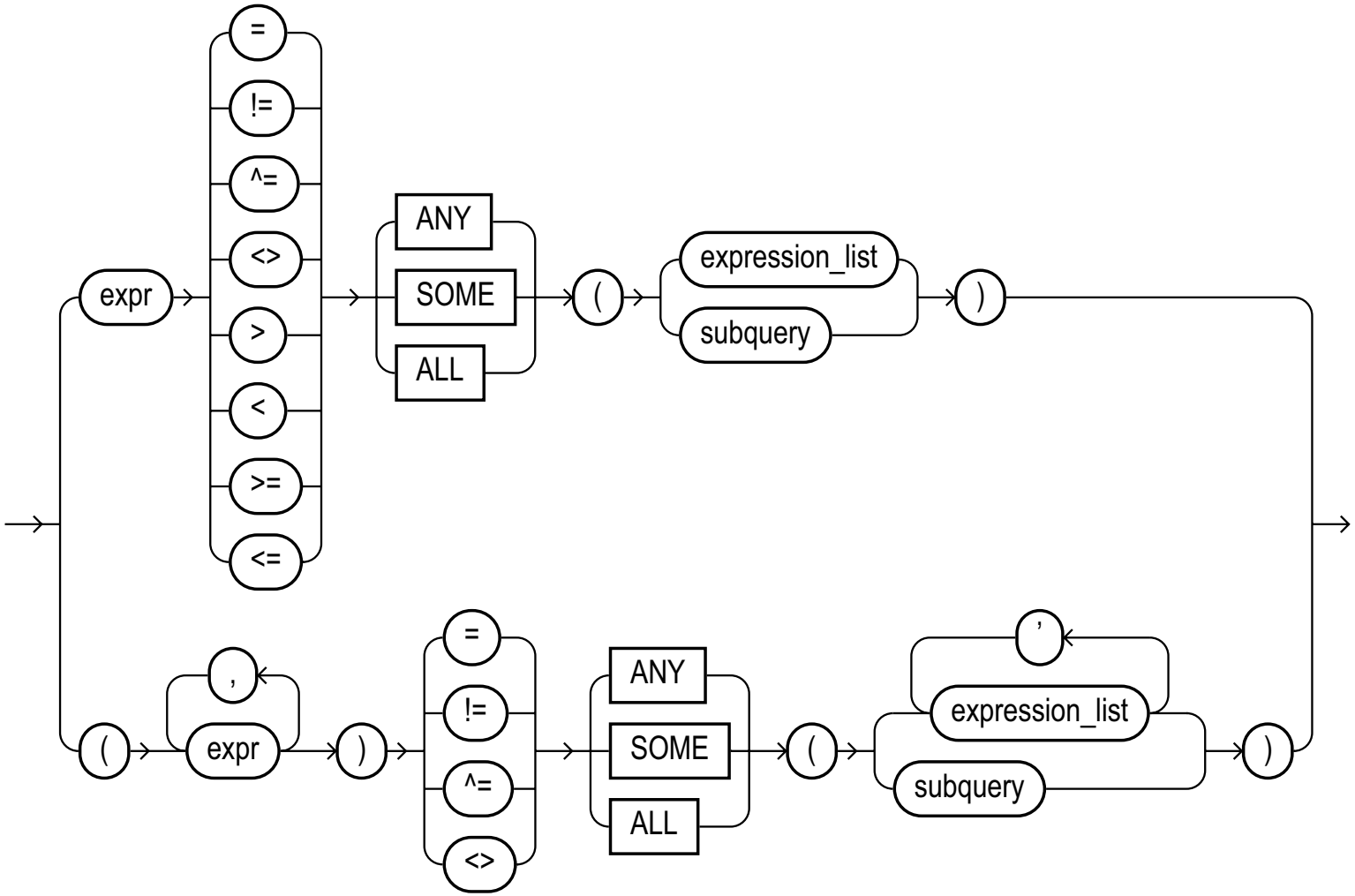


Type of Condition	Purpose	Example
=	Equality test.	SELECT * FROM employees WHERE salary = 2500;
!= ^= < > ¬=	Inequality test. Some forms of the inequality condition may be unavailable on some platforms.	SELECT * FROM employees WHERE salary != 2500;
>	"Greater than" and "less than" tests.	SELECT * FROM employees WHERE salary > 2500;
<		SELECT * FROM employees WHERE salary < 2500;
>=	"Greater than or equal to" and "less than or equal to" tests.	SELECT * FROM employees WHERE salary >= 2500;
<=		SELECT * FROM employees WHERE salary <= 2500;
ANY SOME	Compares a value to each value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, >=. Evaluates to FALSE if the query returns no rows.	SELECT * FROM employees WHERE salary = ANY (SELECT salary FROM employees WHERE department_id = 30);
ALL	Compares a value to every value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, >=. Evaluates to TRUE if the query returns no rows.	SELECT * FROM employees WHERE salary >= ALL (1400, 3000);

simple_comparison_condition::=



group_comparison_condition::=

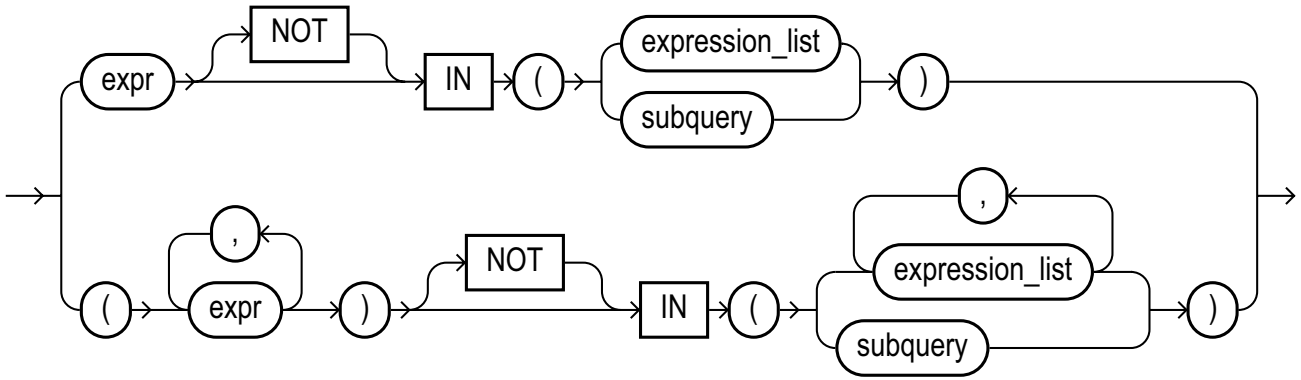


Type of Condition	Operation	Example

IN	"Equal to any member of" test. Equivalent to "= ANY".	`SELECT * FROM employees`
		`  WHERE job_id IN`
		`  ('PU_CLERK','SH_CLERK');`
		`SELECT * FROM employees`
		`  WHERE salary IN`
		`  (SELECT salary`
		`  FROM employees`
		`  WHERE department_id =30);`

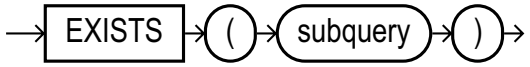
NOT IN	Equivalent to "!=ALL". Evaluates to FALSE if any member of the set is NULL.	`SELECT * FROM employees`
		`  WHERE salary NOT IN`
		`  (SELECT salary`
		`  FROM employees`
		`  WHERE department_id = 30);`
		`SELECT * FROM employees`
		`  WHERE job_id NOT IN`
		`  ('PU_CLERK', 'SH_CLERK');`

membership_condition::=

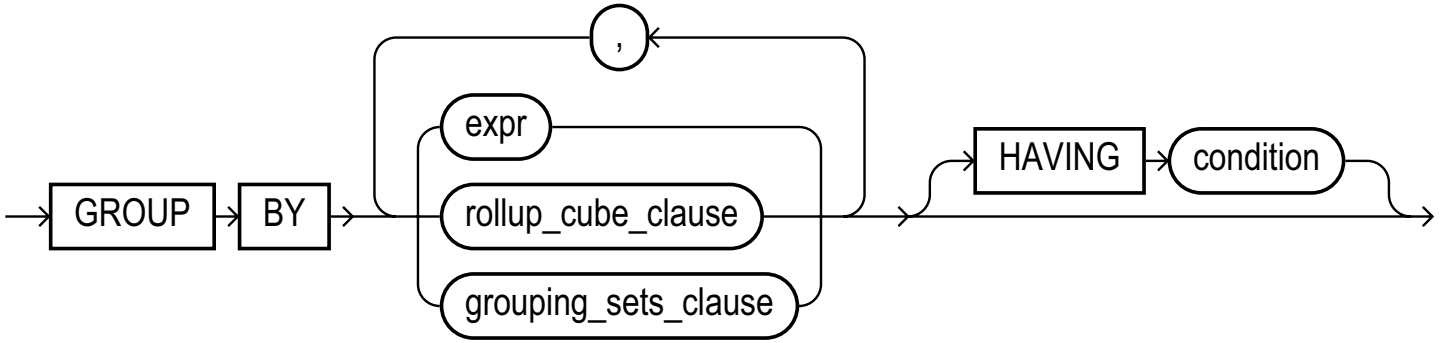


Type of Condition	Operation	Example
EXISTS	TRUE if a subquery returns at least one row.	SELECT department_id FROM departments d WHERE EXISTS (SELECT * FROM employees e WHERE d.department_id = e.department_id);

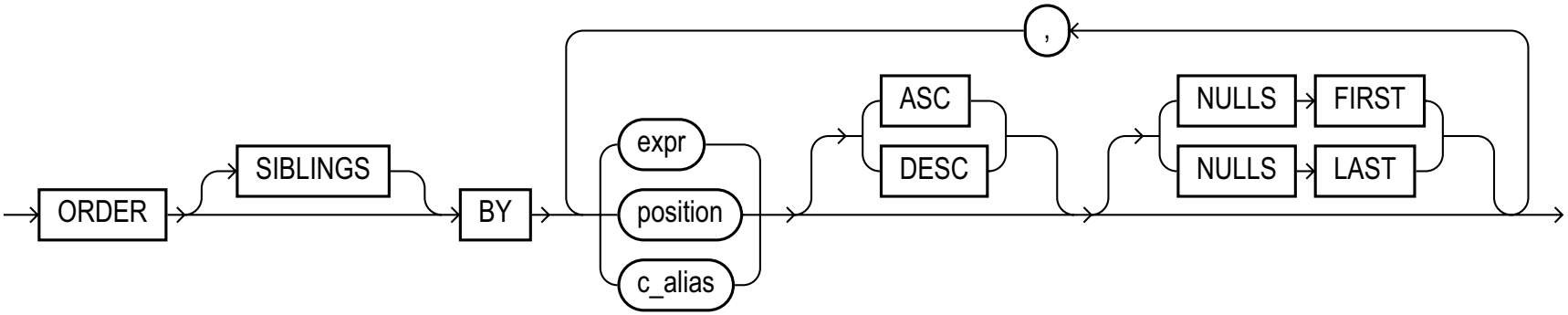
exists_condition::=



group_by_clause::=



order_by_clause::=



Teil III

Erweiterte und alternative Systeme

6 Relationales Datenmodell und Erweiterungen

einige Eigenschaften des relationalen Datenmodells

- **strukturelle Gesichtspunkte**
- **operationale Gesichtspunkte**
- **Architektur- und Anwendungsgesichtspunkte**

strukturelle Gesichtspunkte

- werteorientiert
  - Identifikation von Tupeln nur vermöge ihrer Werte
  - Gleichheit von Tupeln als Gleichheit aller Attributwerte
  - keine benutzersichtbaren Tupelidentifikatoren

- flache Strukturen (“in 1. Normalform”) mit eingeschränkten Konstruktoren
  - Tupel aus Konstantenzeichen aufgebaut
  - Relationen sind homogene Mengen von Tupeln

- modelltheoretisch (und damit vollständig)
  - Instanz gedeutet als (endliche) Struktur im Sinne der Prädikatenlogik
  - in Instanz nur Wahrheitswerte *atomarer* Aussagen explizit darstellbar

- klassische Prädikatenlogik
  - aussagenlogische Junktoren, Quantoren für Individuenvariablen
  - zweiwertige Wahrheitsfunktion
  - Negation als “Nicht-Gültigkeit”
  - logische Implikation betrachtet *alle* Strukturen
- keine Funktionszeichen
- keine (ausführbaren Programme für) Anfragen als Attributwerte
- strukturiert durch Schema
- “vollständiges Wissen” mit Annahme der “geschlossenen Welt”
- “genaues Wissen”
- “bestimmtes Wissen”

operationale Gesichtspunkte

- wenige Standardoperationen
  - Anfragen, Tupel-Einfügen, -Entfernen, -Ändern
  - als “Mehrwertdienst” nur logische Gültigkeit (oder Implikation)
- nur Gleichheits- und Ungleichheits-Vergleiche bezüglich Konstantenzeichen
- Konstantenzeichen uninterpretiert
- nur “sprungfreie Anfrage-Programme”
- unmittelbar vom Benutzer gesteuert
- ideelle Trennung von “Programmiersystem” und “Informationssystem”

Architektur- und Anwendungsgesichtspunkte

- “zentral” und “top-down” mit im Vorhinein deklariertem Schema

- “homogen”
  - alle Daten vom strukturell gleichen Datentyp
  - globale Interpretation semantischer Bereichsnamen

- “statisch”
  - Administrator deklariert zeitlich unverändertes Schema

- “normativ”
  - semantische Bedingungen als zu erzwingende Invarianten angesehen

- client-server Architektur

Varianten, Erweiterungen, Verallgemeinerungen

- **relationales Datenmodell:**
  - wohl fundiert, hinreichend effizient, automatisch optimierbar, zusätzliche Funktionalität näherungsweise "simulierbar", ...
  - aber für manche Anwendungen (zu) "ausdrucksschwach"

- **Ziele:**
  - Erweiterungen unter Beibehaltung der "guten Eigenschaften"
  - Verträglichkeit der Erweiterungen untereinander sicherstellen

- **Wirklichkeit:**
  - unzählige Einzelvorschläge
  - wenige mächtige integrierte Systeme

Eigenschaften als Ausgangspunkt für weitergehende Modelle

• werteorientiert	→	objektorientiert
• flache Strukturen	→	genestete Strukturen mit beliebigen Konstruktoren
• modelltheoretisch	→	beweistheoretisch
• Gültigkeit auswerten	→	Implikationen bestimmen
• klassische Prädikatenlogik	→	“nichtklassische Logiken” für Wissen, Glauben, ...

• keine Funktionszeichen	→	Funktionszeichen
• keine Anfragen als Attributwerte	→	speicherbare Programme
• strukturiert	→	halbstrukturierte Daten (XML)
	→	beliebige Objekte (Multimedia)
• “vollständiges Wissen”	→	Nullwerte
	→	“offene Welt”
• “genaues Wissen”	→	“ungenaues Wissen”
• “bestimmtes Wissen”	→	“nichtklassische Logiken”
	→	fuzzy logic

- wenige Standardoperationen → anwendungsbestimmte Methoden

- nur Gleichheits- und Ungleichheits-Vergleiche
  bezüglich Konstantenzeichen → Ähnlichkeitsvergleiche
  bezüglich Multimedia-Objekten

- Konstantenzeichen uninterpretiert → vordefinierte Datentypen

- "sprungfreie Anfrage-Programme" → Rekursionen
  → (geschichtete) Fixpunkte

- unmittelbar vom Benutzer gesteuert → "aktive" Systeme mit Regeln
  → E(vent)C(ondition)A(ction)

- Trennung von “Programmiersystem” und “Informationssystem” → “semantische Lücken” schließen
- “zentral” und “top-down” → verteilt
- “homogen” → föderiert, 5-Schema-Architektur
- “statisch” → mediiert, “wrapper”
- “normativ” → “faktisch und explorativ”
- client-server Architektur → Performative mit “Ontologien”

7 Erweiterte Navigationsmöglichkeiten

Navigation in Schema und Instanz

- **relationales Datenmodell**

  Schemaebene: "fest-beschränkte" Pfade im Schema-Hypergraphen
  Instanzenebene: Pfad-Auswertung bezüglich Instanz durch Verbunde

  "Universalrelationsicht": (halb-) automatische Generierung solcher Pfade

- ***beweistheoretische* Modelle** mit ***Horn-Formeln*** als Anfragen

  Schemaebene: "rekursive Pfadmuster" im Schema-Hypergraphen

  Instanzenebene: unbeschränkte, datenabhängige Pfad-Auswertung bezüglich Instanz durch Fixpunkt

- **objektorientierte Modelle** mit Dereferenzierung in Anfragen

  Schemaebene: "fest-beschränkte" Pfade im "Schema-Verweis-Graphen"

  Instanzenebene: Pfad-Auswertung bezüglich Instanz durch "Verbunde über Dereferenzierungen"

Navigation im relationalen Datenmodell: “Kochrezept”

finde formale Entsprechungen für:

- **Begriffe**

  als *Attribute*, *Relationensymbole*, *semantische Bereichsnamen* oder *Konstantenzeichen*

- **Zusammenhänge**

  als ***Pfad im Hypergraphen*** des **Schemas** (entsprechend einer Folge von *Verbunden*)

- **Zwecke**

  als *Selektionen* und *Projektionen*

also:

- **Schemaebene**:

  “***fest-beschränkter***” Pfad im Schema-Hypergraphen
  (***feste Anzahl*** von Verbunden)

- **Instanzenebene**:

  Pfad-Auswertung mit ***datenunabhängiger*** Anzahl von Verbunden

7.1 erweiterte Navigation durch Horn-Formeln als Anfragen

beweistheoretische Deutung der relationalen Strukturen

Schema $< < R_1 \mid X_1 \mid SC_1 > , \ldots , < R_n \mid X_n \mid SC_n > \mid SC \mid \mathbf{a} >$

Basisrelationen **EDB** $:= \{ R_1 , \ldots , R_n \}$
(extensional database relations)

Instanz: $(d, r_1, \ldots, r_n)$

Ansatz

Tupel $\mu \in r_i$ mit $X_i = \{ A_1, \ldots, A_{si} \}$

als **Grundfakt** $R_i(A_1 : c_1 , \ldots , A_{si} : c_{si}).$ mit $c_j = \mu(A_j)$

kurz $R_i(c_1, \ldots, c_{si}).$

entsprechend **Instanz** $(\_ , r_1, \ldots, r_n)$

als **Menge von Grundfakten** $\bigcup_{i = 1, \ldots, n} r_i$

Schreib- und Redeweisen für Horn-Formeln

- **syntaktisches Material**

C	unendliche Menge der *Konstantenzeichen*
R	Menge der *Relationensymbole* (einschließlich =)
V	Menge der *Individuenvariablen*
T := **C** $\cup$ **V**	Menge der *Terme*

- **Formeln**

atomare Formel: $R(t_1,...,t_s)$ mit $R \in \mathbf{R}$ und $t_1,...,t_s \in \mathbf{T}$

atomare Grundformel: $R(c_1,...,c_s)$ mit $R \in \mathbf{R}$ und $c_1,...,c_s \in \mathbf{C}$

(Horn-) Klausel: A_0 :- A_1 , ... , A_m. Abkürzung für

$A_0 \vee \neg A_1 \vee ... \vee \neg A_m$

mit $A_0, A_1, ..., A_m$ atomare Formeln
und $m \in \mathrm{IN}_0$

Konklusion: A_0

Prämissen $A_1, ..., A_m$

- **weitere Bezeichnungen**

$\text{rel}(K)$:= Menge der Relationensymbole von Klausel $K \equiv A_0 \text{ :- } A_1, \ldots, A_m.$

$\text{concl}(K)$:= Relationensymbol aus Konklusion A_0 von Klausel $K \equiv A_0 \text{ :- } A_1, \ldots, A_m.$

Fakt: $A_0.$ Abkürzung für $A_0 \text{ :- }.$

Grundfakt: $A_0.$ mit A_0 atomare Grundformel

Regel: $A_0 \text{ :- } A_1, \ldots, A_m.$ mit $m \geq 1$

Menge der (***Horn-***)***Klauseln***: **HK**

Menge der ***Grundfakten***: **GF**

erweiterte Operationen mit beweistheoretischer Deutung

Syntax:

$< R \mid \boldsymbol{Q} >$ LOGODAT-**Anfrage** mit

$R \in \mathbf{R} \setminus \{=\}$ Ergebnis-Relationensymbol (für *Format* der Ergebnisse)

$\boldsymbol{Q} \subset_{\text{endlich}} \mathbf{HK}$ mit $\text{concl}(\boldsymbol{Q}) \subset \mathbf{R} \setminus (\mathbf{EDB} \cup \{=\})$

(Hornklausel-)**Anfrageprogramm**
(zum Erschließen der Ergebnisse)

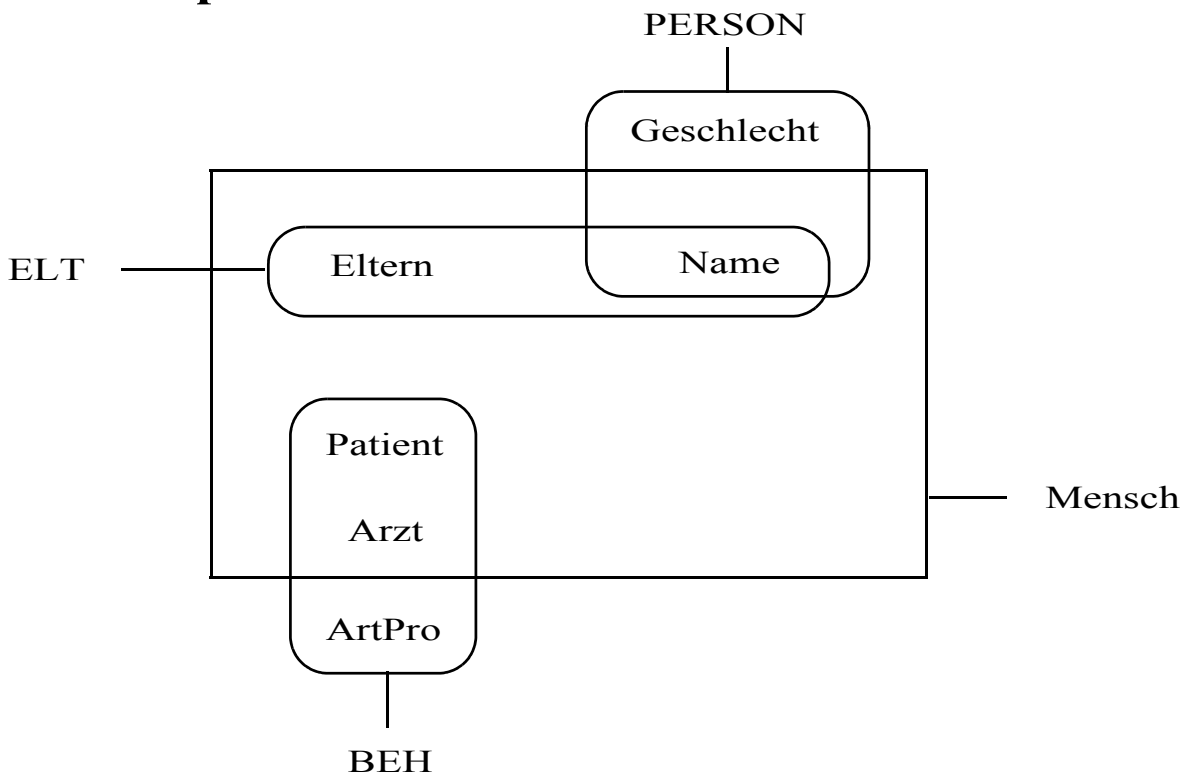
beabsichtigte deklarative Semantik:

alle logischen Implikationen

von der gespeicherten **Instanz** und dem **Anfrageprogramm**

gemäß **Format**

Hypergraph für Beispiel:



Beziehung	Pfad
Tochter-Elternteil-Beziehung:	Geschlecht , Name , Eltern
Vorfahr-Beziehung:	Name , Eltern_1=Name_2 , Eltern_2=Name_3 , ... , Eltern_*t*
gleiche-Generation-Beziehung:	simultane Pfade gleicher Länge, die Nachfahren bestimmen

Beispiel “Tochter-Elternteil-Beziehung”

< TOCHTER |

{ TOCHTER (Name , Eltern) :- ELT (Name , Eltern) ,(AND)

PERSON (Name , Geschlecht) ,(AND)

= (Geschlecht , weib).

}

>

Beispiel “Vorfahr-Beziehung” (transitive Hülle von ELT)

```
<      VORFAHR     |

  {    VORFAHR ( Name , Ahn )  :-   ELT ( Name,  Ahn ).          ,(OR)

       VORFAHR ( Name , Ahn )  :-   ELT ( Name ,  Eltern ) ,(AND)
                                    VORFAHR( Eltern , Ahn ).
  }

>
```

Beispiel “gleiche-Generation-Beziehung”

```
<     GLEICHGEN    |

  {   GLEICHGEN ( Name , Name )  :-   ELT ( Name , Eltern ).        ,(OR)

      GLEICHGEN ( Eltern , Eltern ) :-   ELT ( Name , Eltern ).    ,(OR)

      GLEICHGEN ( Name_1 , Name_2)
                                  :-   ELT ( Name_1 , Eltern_1 ) ,(AND)
                                       ELT ( Name_2 , Eltern_2 ) ,(AND)
                                       GLEICHGEN ( Eltern_1 , Eltern_2).
  }

>
```

deklarative Semantik von LOGODAT-Anfragen: Definition

Schema	RS mit **EDB** $= \{ R_1,...,R_n \}$
LOGODAT-Anfrage	$< R \mid \boldsymbol{Q} >$
deklarative Semantik	$\text{eval}_{RS}(< R \mid \boldsymbol{Q} >)$
ordnet Instanz (Grundfakten-Menge)	f

eine Grundfakten-Menge
zum Relationensymbol R zu:

$\text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$

$:= \{ R(c_1,...,c_s). \mid f \cup \boldsymbol{Q} \models R(c_1,...,c_s). \text{ mit } c_1,...,c_s \in \mathbf{C} \}$

|= logische Implikation *für LOGODAT-Strukturen*:
- Universum **C**
- Konstantenzeichen aus **C** durch sich selbst interpretiert
- insbesondere unique name assumption

Grundfakten-Transformation für operationale Fixpunktsemantik

zur Klausel $K \equiv A_0 \text{ :- } A_1, \ldots, A_m.$ mit $\text{rel}(A_0) \in \mathbf{R} \setminus \{=\}$

Grundfakten-Transformation

$$T_K : \wp\ \mathbf{GF} \rightarrow \wp\ \mathbf{GF}$$

$$T_K(g) := \{\ R(c_1, \ldots, c_s).\ \mid$$

es gibt Variablenbelegung γ mit

(1) $\gamma(A_0) \equiv R(c_1, \ldots, c_s)$

(2) für alle $i = 1, \ldots, m$:
$\gamma(A_i.) \in g \cup \{ =(c,c). \mid c \in \mathbf{C} \}$

$\}$

zur Klauselmenge $\boldsymbol{K}$ mit $\text{concl}(K) \in \mathbf{R} \setminus \{=\}$ für alle $K \in \boldsymbol{K}$

Grundfakten-Transformation

$$T_{\boldsymbol{K}} : \wp\ \mathbf{GF} \rightarrow \wp\ \mathbf{GF}$$

$$T_{\boldsymbol{K}}(g) := \cup_{K \in \boldsymbol{K}}\ T_K(g)$$

Grundfakten-Transformation und Resolution

Grundfakten-Transformation zu
$K \equiv A_0 \text{ :- } A_1, \ldots, A_m.$ mit $\text{rel}(A_0) \in \mathbf{R} \setminus \{=\}$

$T_K : \wp\,\mathbf{GF} \rightarrow \wp\,\mathbf{GF}$

$T_K(g) := \{ \; R(c_1, \ldots, c_s). \mid$ es gibt Variablenbelegung γ mit

(1) $\gamma(A_0) \equiv R(c_1, \ldots, c_s)$

(2) für alle $i = 1, \ldots, m$:
$\gamma(A_i.) \in g \cup \{ =(c,c). \mid c \in \mathbf{C} \}$ $\}$

beinhaltet im Wesentlichen:

- alle (derzeit ausführbaren) (Hyper-)**Resolutionen**
- von der Klausel K einerseits
- mit Atomen aus g und Gleichheitsatomen andererseits
- mit jeweiligem **allgemeinsten Unifikator** γ

iterierte Anwendung der Grundfakten-Transformation mit Fixpunkt

$\boldsymbol{K}$:= { ELT (anton, fritz). , (1)
ELT (maria, anton). , (2)
ELT (hugo, anton). , (3)
GLEICHGEN (Name, Name) :- ELT (Name, Eltern). , (4)
GLEICHGEN (Eltern, Eltern) :- ELT (Name , Eltern). , (5)

GLEICHGEN (Name_1, Name_2) :- ELT (Name_1, Eltern_1), (6)
ELT (Name_2, Eltern_2),
GLEICHGEN (Eltern_1, Eltern_2). }

$g_1 := T_{\boldsymbol{K}}(\emptyset) =$ { ELT (anton, fritz). , ELT (maria, anton). , ELT (hugo, anton). }

$g_2 := T_{\boldsymbol{K}}(g_1) =$ { ELT (anton, fritz). , ELT (maria, anton). , ELT (hugo, anton). ,
GLEICHGEN (anton,anton). , GLEICHGEN (maria, maria). ,
GLEICHGEN (hugo,hugo). , GLEICHGEN (fritz, fritz). }

$g_3 := T_{\boldsymbol{K}}(g_2) =$ { ELT (anton, fritz). , ELT (maria, anton). , ELT (hugo, anton). ,
GLEICHGEN (anton, anton)., GLEICHGEN (maria, maria). ,
GLEICHGEN (hugo, hugo). , GLEICHGEN (fritz, fritz). ,
GLEICHGEN (maria, hugo). , GLEICHGEN (hugo, maria). }

Grundlagen zum Fixpunktsatz für Grundfakten-Transformation

bekannt $(\wp\, \mathbf{GF}, \subset)$ vollständiger Verband

Hilfssatz 1 T_K ist *monoton*: für alle $g, g' \subset \mathbf{GF}$ gilt:

$$g \subset g' \Rightarrow T_K(g) \subset T_K(g')$$

Hilfssatz 2 T_K ist *stetig*: für alle ω-Ketten $g_0 \subset g_1 \subset g_2 \subset ... \subset \mathbf{GF}$ gilt:

$$T_K(\bigcup_{i \in \omega} g_i) = \bigcup_{i \in \omega} T_K(g_i).$$

Fixpunktsatz von Tarski-Kleene

Voraussetzung: - $(\boldsymbol{H}, \subset)$ *ω-vollständige* Halbordnung mit kleinstem Element $\varnothing$ und Supremumsoperation $\cup$

- $T : \boldsymbol{H} \to \boldsymbol{H}$ *monotone* und *stetige* Transformation

1. T besitzt eindeutig bestimmten kleinsten Fixpunkt $fix \in \boldsymbol{H}$:
   es gibt $fix \in \mathbf{H}$ mit $T(fix) = fix$
   und für alle $x \in \boldsymbol{H}$: $T(x) = x \Rightarrow fix \subset x$

2. $$fix = \bigcup_{i \in \omega} T^i(\varnothing)$$

operationale Fixpunktsemantik von LOGODAT-Anfragen: Definition

Schema RS mit $\mathbf{EDB} = \{ R_1, \ldots, R_n \}$

LOGODAT-Anfrage $< R \mid \boldsymbol{Q} >$

operationale Semantik $\text{eval}_{RS}^{\text{fix}}(< R \mid \boldsymbol{Q} >)$

ordnet Instanz (Grundfakten-Menge) f

eine Grundfakten-Menge

zum Relationensymbol R zu:

$$\text{eval}_{RS}^{\text{fix}}(< R \mid \boldsymbol{Q} >)(f)$$
$$:= \{ \; R(c_1,\ldots,c_s). \mid R(c_1,\ldots,c_s). \in \mathit{fix}(f \cup \boldsymbol{Q}) \; \}$$

$\mathit{fix}(f \cup \boldsymbol{Q})$ kleinster Fixpunkt der Grundfakten-Transformation $T_{f \cup \boldsymbol{Q}}$ zu $f \cup \boldsymbol{Q}$

Äquivalenz von deklarativer und operationaler Semantik

Korrektheit und Vollständigkeit der Fixpunktsemantik

für alle Instanzen f zu RS:

$$\text{eval}_{RS}^{\text{fix}}(< R \mid \boldsymbol{Q} >)(f) \quad = \quad \text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$$

Monotonie der Grundfakten-Transformation

- **Hilfssatz 1** T_K **ist** ***monoton***:

  für alle $g, g' \subset$ **GF** gilt: $g \subset g' \Rightarrow T_K(g) \subset T_K(g')$

- **Beweis**

  folgt direkt aus der Definition der Grundfakten-Transformation

Stetigkeit der Grundfakten-Transformation

- **Hilfssatz 2** T_K **ist** ***stetig***:

  für alle ω-Ketten $g_0 \subset g_1 \subset g_2 \subset \ldots \subset \mathbf{GF}$ gilt:

  $T_K(\bigcup_{i \in \omega} g_i) = \bigcup_{i \in \omega} T_K(g_i)$

- **Beweis** "⊃": gemäß Monotonie

"⊂":

betrachte: $R(c_1,\ldots,c_s). \in T_K(\bigcup_{i \in \omega} g_i)$

Definition T_K: es gibt Klausel $K \equiv A_0 :- A_1,\ldots, A_m. \in K$

und Variablenbelegung γ mit

(1) $\gamma(A_0) \equiv R(c_1,\ldots,c_s)$

(2) $\gamma(A_j.) \in \bigcup_{i \in \omega} g_i \cup \{ =(c,c). \mid c \in \mathbf{C} \}$ für $j = 1,\ldots,m$

K endlich: es gibt ein Kettenelement g_e mit

(2*) $\gamma(A_j.) \in g_e \cup \{ =(c,c). \mid c \in \mathbf{C} \}$ für $j = 1,\ldots,m$

(1) und (2*): $R(c_1,\ldots,c_s). \in T_K(g_e) \subset \bigcup_{i \in \omega} T_K(g_i)$

Fixpunktsatz von Tarski-Kleene

$T : \boldsymbol{H} \rightarrow \boldsymbol{H}$ *monotone* und *stetige* Transformation für
ω-vollständige Halbordnung $(\boldsymbol{H}, \subset)$
mit kleinstem Element $\varnothing$ und Supremumsoperation $\cup$:

T besitzt einen ***eindeutig bestimmten kleinsten Fixpunkt*** $fix \in \boldsymbol{H}$ mit

$fix = \bigcup_{i \in \omega} T^i(\varnothing)$

Beweis

(1) $\boldsymbol{T^i(\varnothing) \subset T^{i+1}(\varnothing)}$ **für alle** $\boldsymbol{i} \in \omega$**:**

$i = 0$: $T^0(\varnothing) = \varnothing \subset T^1(\varnothing)$

$i > 0$:

$T^{i+1}(\varnothing) = T(T^i(\varnothing))$		Definition von T^{i+1}
$\supset T(T^{i-1}(\varnothing))$		Induktionsannahme und Monotonie
$= T^i(\varnothing)$		Definition von T^i

(2) $\bigcup_{\boldsymbol{i} \in \omega} \boldsymbol{T^i(\varnothing)} \in \boldsymbol{H}$ ω-Vollständigkeit

(3) Fixpunkt-Eigenschaft:

$$T\left(\bigcup_{i\in\omega} T^i(\varnothing)\right) = \bigcup_{i\in\omega} T\left(T^i(\varnothing)\right) \qquad \text{(1) und Stetigkeit von } T$$

$$= \bigcup_{i\in\omega} T^{i+1}(\varnothing) \qquad \text{Definition von } T^{i+1}$$

$$= \bigcup_{i\in\omega} T^i(\varnothing) \qquad T^0(\varnothing) = \varnothing$$

(4) kleinster Fixpunkt, d.h.

für beliebigen Fixpunkt x mit $T(x) = x$ gilt: $T^i(\varnothing) \subset x$ für alle $i \in \omega$:

$i = 0$: $T^0(\varnothing) = \varnothing \subset x$

$i + 1$: $T^{i+1}(\varnothing) = T\left(T^i(\varnothing)\right)$ Definition von T^{i+1}

$\subset T(x)$ Induktionsannahme und Monotonie

$= x$ $T(x) = x$

$T_K^i(\emptyset)$ enthält nur Implikationen von K

$i = 0$: trivial wegen $T_K^0(\emptyset) = \emptyset$

$i > 0$:

Induktionsannahme: $K \models T_K^{i-1}(\emptyset)$ (1)

Voraussetzung: $R(c_1,\ldots,c_s). \in T_K^i(\emptyset) \setminus T_K^{i-1}(\emptyset)$ (2)

Voraussetzung: M Modell von K (3)

zu zeigen: M Modell von $R(c_1,\ldots,c_s)$

Beweis:

(2), Definition von T_K: es gibt Klausel $K \equiv A_0 :- A_1,\ldots,A_m. \in K$

und Variablenbelegung γ mit

$\gamma(A_0) \equiv R(c_1,\ldots,c_s)$ und (4)

$\gamma(A_j.) \in T_K^{i-1}(\emptyset) \cup \{ =(c,c). \mid c \in \mathbf{C} \}$ für $j = 1,\ldots,m$ (5)

(3), (1), Gleichheitsfakten trivial gültig:

M Modell von $\gamma(A_j.)$ für $j = 1,\ldots,m$ (6)

(3): M Modell von $\gamma(K)$ (7)

(6), (7), mit (4): M Modell von $\gamma(A_0) \equiv R(c_1,\ldots,c_s)$

kleinster Fixpunkt von T_K enthält nur Implikationen von K

$$K \models fix(K)$$

- $T_K^i(\emptyset)$ enthält nur Implikationen von K: $K \models T_K^i(\emptyset)$
- $fix(K) = \bigcup_{i \in \omega} T_K^i(\emptyset)$

kleinster Fixpunkt von T_K als Modell von K

$$Model(\ fix(K)\ ) \in \mathbf{Mod}(K)$$

- **kleinster Fixpunkt von T_K:** Grundfakten-Menge $fix(K)$
- **kanonisch als LOGODAT-Struktur auffassen:** $Model(\ fix(K)\ )$

angenommen: $M := Model(\ fix(\mathbf{K})\ )$ kein Modell von $\mathbf{K}$

Def. Modell: es gibt $K \equiv A_0 \text{ :- } A_1,\ldots,A_m. \in \mathbf{K}$ mit: M kein Modell von K

d.h.: es gibt Variablenbelegung β mit

$|\neq_{M,\beta}\ A_0.$ (1a)

$|=_{M,\beta}\ A_j.$ für $j = 1,\ldots,m$ (1b)

Def. M, (1): $\beta(A_0.) \notin fix(\mathbf{K})$ (2a)

$\beta(A_j.) \in fix(\mathbf{K}) \cup \{\ =(c,c)\ \mid\ c \in \mathbf{C}\ \}$ für $j = 1,\ldots,m$ (2b)

(2b), Def. Grundfakten-Transformation, Fixpunkteigenschaft:

$\beta(A_0.) \in T_{\mathbf{K}}(\ fix(\mathbf{K})\ ) = fix(\mathbf{K})$ (3)

Widerspruch zu (2a)!

Äquivalenz von deklarativer Semantik und Fixpunktsemantik: Satz

für alle Instanzen f zu RS:

$$\text{eval}_{RS}^{\text{fix}}(< R \mid \boldsymbol{Q} >)(f) \quad = \quad \text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$$

- jedes von Fixpunktsemantik erzeugte Grundfakt ist ***korrekt***:
  es ist eine logische Implikation der zu verarbeitenden Klauseln
- *alle* logisch implizierten Grundfakten werden ***vollständig***
  von der Fixpunktsemantik erzeugt

für kleinsten Fixpunkt $fix := fix(f \cup \boldsymbol{Q})$ von $T_{f \cup \boldsymbol{Q}}$,
für $c_1,\ldots,c_s \in \mathbf{C}$:

$R(c_1,\ldots,c_s). \in fix$ **gdw** $f \cup \boldsymbol{Q} \models R(c_1,\ldots,c_s).$

Korrektheit

kleinster Fixpunkt enthält nur logische Implikationen

Vollständigkeit

betrachte: $R(c_1,\ldots,c_s). \notin fix$

Definition $Model(fix)$: $Model(fix) \notin \mathrm{Mod}(\ R(c_1,\ldots,c_s).\ )$ (1)

andererseits: $Model(fix) \in \mathrm{Mod}(\ f \cup \boldsymbol{Q}\ )$ (2)

(1), (2), Definition $\models$: $f \cup \boldsymbol{Q} \not\models R(c_1,\ldots,c_s).$

naive Auswertung von LOGODAT-Anfragen

- **übersetze Klauseln in relationale Algebra:**
  - einzelne Prämisse in Projektion-Selektion-Vergleich-Ausdruck
  - Folge der Prämissen in Vergleich-Verbund-Ausdruck
  - Konklusion in Projektion und “Relationenvariable”
  - Klauseln mit gleicher Konklusion in Wertzuweisung der Form “Relationenvariable” := Projektion(Vereinigung-Ausdruck)
- **bilde Abhängigkeitsgraphen**

  als “Petrinetz-ähnliches Datenfluss-Modell” für alle Wertzuweisungen
- **führe Wiederholungsanweisung durch**:

  berechne simultan alle Wertzuweisungen,
  bis keine Änderungen mehr erfolgen

einfaches Beispiel für naive Auswertung: transitive Hülle

- **Klauselmenge:** $K_1 \equiv T(T1:x, T2:y) :- P(P1:x, P2:y).$

  $K_2 \equiv T(T1:x, T2:y) :- P(P1:x, P2:z), T(T1:z, T2:y).$

- **Wertzuweisung mit algebraischem Ausdruck:**

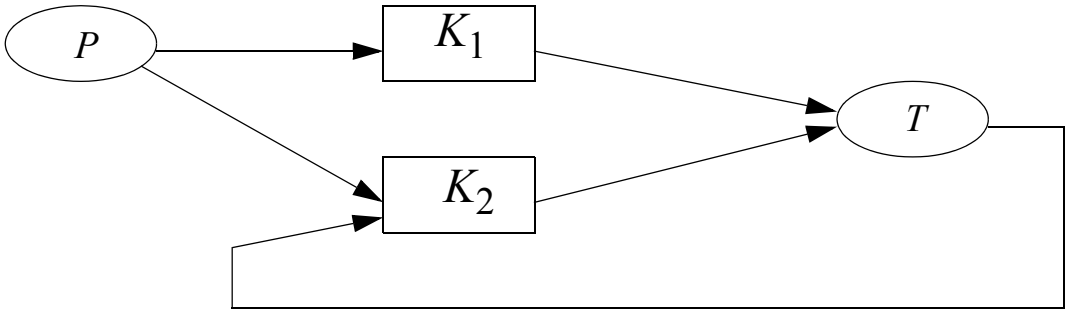
  $T(T1, T2) :=$

  $\pi_{[(T1,X),(T2,Y)]}(\pi_{[(X,P1),(Y,P2)]}(P))$

  $\cup$

  $\pi_{[(T1,X),(T2,Y)]}(\pi_{[(X,P1),(Z,P2)]}(P) \bowtie \pi_{[(Z,T1),(Y,T2)]}(T))$

- **Abhängigkeitsgraph:**



- **Wiederholung:** $T\_neu := \emptyset$;

  REPEAT $T := T\_neu$;

  $T\_neu :=$ Expression(P, T) wie oben definiert

  UNTIL $T = T\_neu$;

differentielle und optimierte Auswertung von LOGODAT-Anfragen

- **vermeide Duplikate**

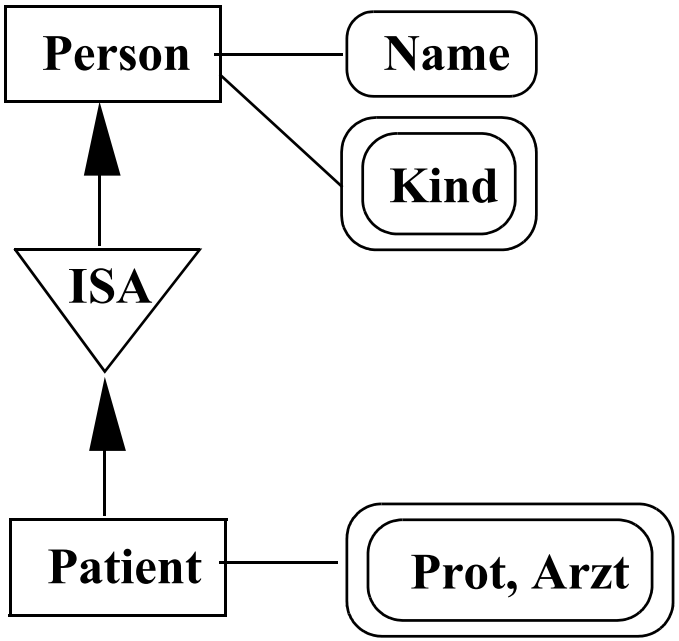
  jeweils für ein Argument einer Wertzuweisung
  nur “gerade neu erzeugte Fakten” benutzen

- **(“magic set-”) optimiere durch “Vorziehen von Selektionen”**

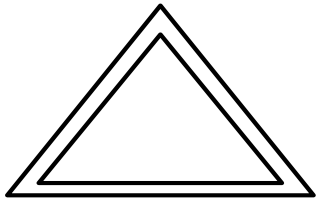
  jeweils frühestmöglich
  Variablen mit Konstanten binden

7.2 erweiterte Navigation durch Dereferenzierung

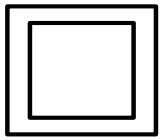
Beispiel: eine Modellierung mit objektorientierter Formalisierung



Zeichenerklärung

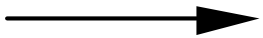


Objekt vom Typ `Dictionary`(als B*-Baum)



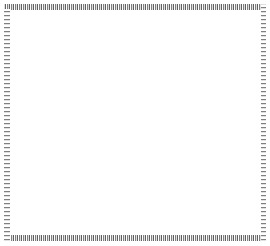
Objekt vom Typ `Set`

Objekt vom Typ `Person_Typ, Patient_Typ` oder `Behandlung_Typ`

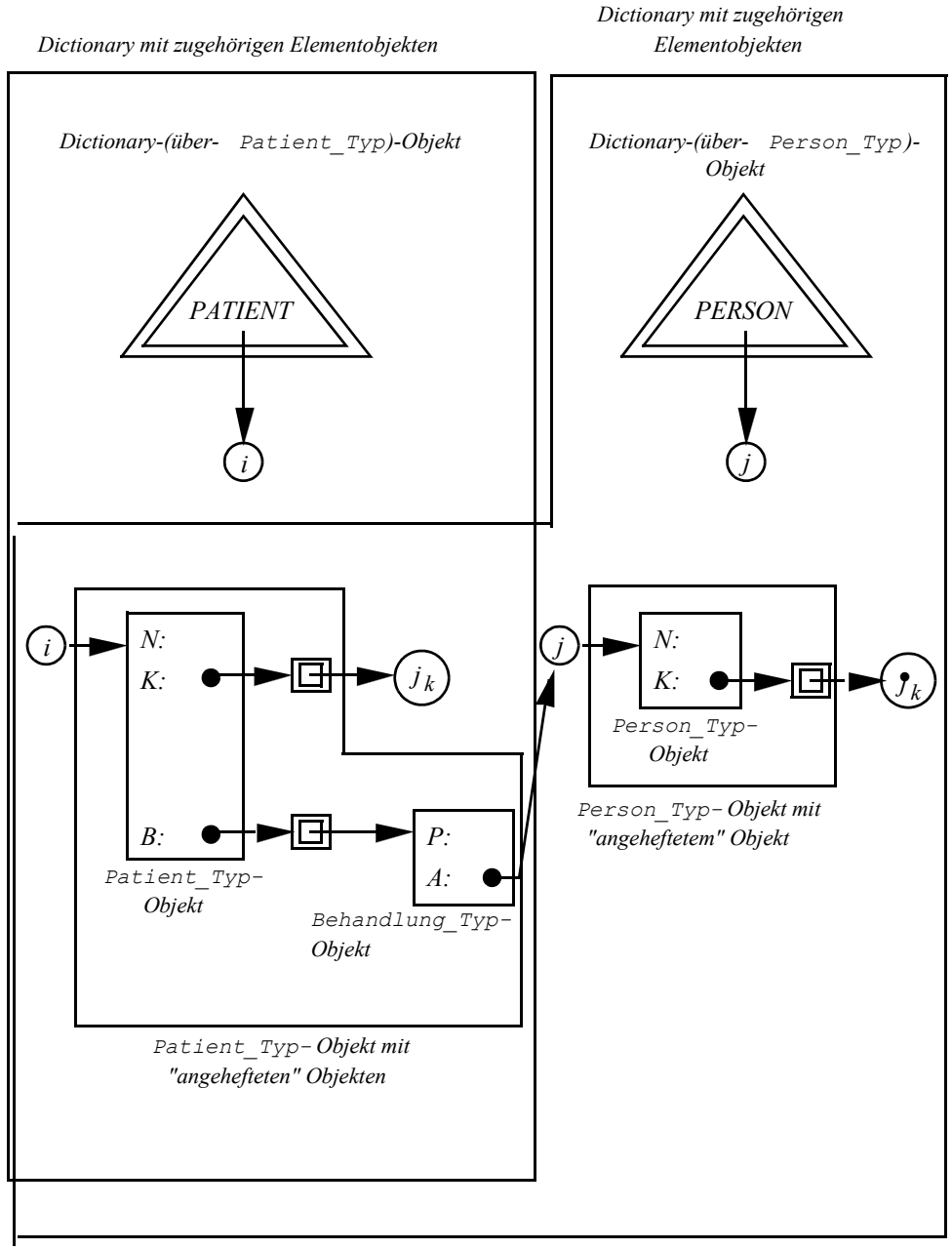


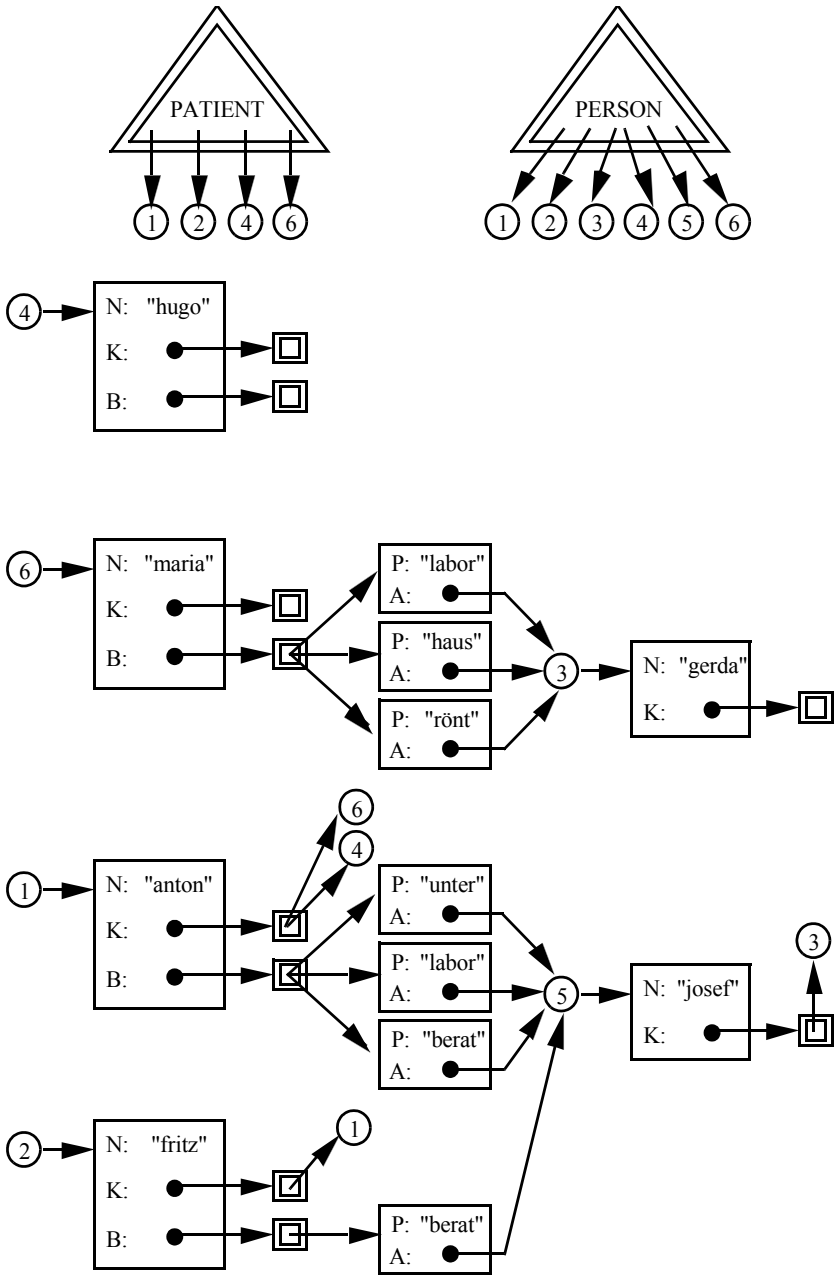
Verweise über Adressen / UIDs (wobei die Schreibweise

eine durchgezogene Linie ersetzt)



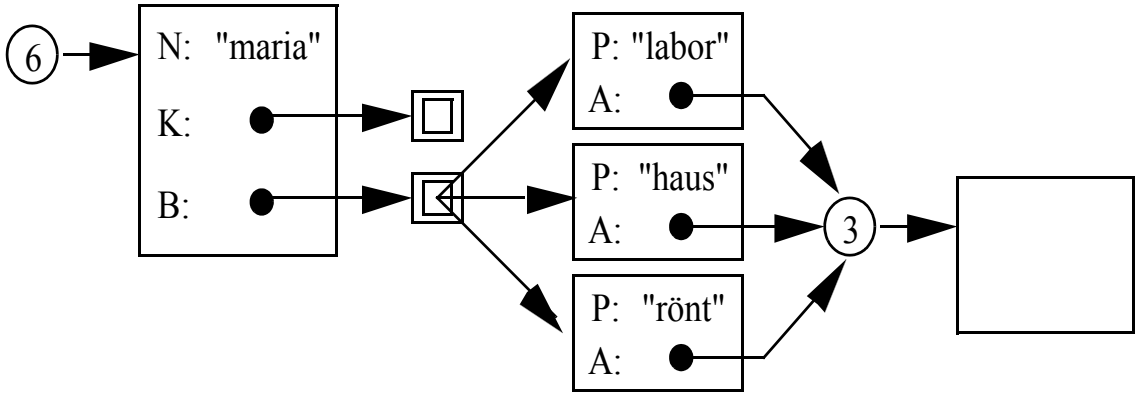
durch Verweisstrukturen gebildete
"zusammengesetzte Objekte"


Dictionary mit zugehörigen Elementobjekten
Dictionary mit zugehörigen Elementobjekten
Dictionary-(über- Patient_Typ)-Objekt
Dictionary-(über- Person_Typ)-Objekt
PATIENT
PERSON
i
j
i
N:
K:
B:
j_k
Patient_Typ-Objekt
P:
A:
Behandlung_Typ-Objekt
Patient_Typ- Objekt mit "angehefteten" Objekten
j
N:
K:
j_k
Person_Typ-Objekt
Person_Typ- Objekt mit "angeheftetem" Objekt


PATIENT
1
2
4
6
PERSON
1
2
3
4
5
6
4
N: "hugo"
K:
B:
6
N: "maria"
K:
B:
P: "labor"
A:
P: "haus"
A:
P: "rönt"
A:
3
N: "gerda"
K:
1
N: "anton"
K:
B:
6
4
P: "unter"
A:
P: "labor"
A:
P: "berat"
A:
5
N: "josef"
K:
3
2
N: "fritz"
K:
B:
1
P: "berat"
A:

objektorientierte Formalisierung eines Patienten namens Maria

graphisch:



in F-Logik: geeignet “logisch simulierbar”

Regeln mit Dereferenzierungen in F-Logik: Übersicht

Sichten (Anfragen) in F-Logik:	aufgebaut aus ***Objekten***, durch ***Terme*** bezeichnet
Terme:	mit Hilfe von geeigneten ***Funktionszeichen*** gebildet
Signaturmoleküle:	vereinbaren das Schema der Sicht
Horn-Klauseln:	bestimmen die Objekte der Sicht

umgangssprachliches Beispiel

Anfrage für Surrogatpaare der Eltern-Kind-Beziehungen:

wenn *X* ein Objekt der Klasse *person* bezeichnet,
das unter dem mengenwertigen Attribut *kinder*
auf ein Objekt *Y* (vom Typ *person_typ*) verweist,

dann bezeichne *el_ki(X, Y)* ein Objekt der Sicht *sur_relationELT*,
das unter dem skalaren Attribut *eltern* auf das Objekt *X*
und unter dem skalaren Attribut *kind* auf das Objekt *Y* verweist

8 Datenmodelle für halb-strukturierte Daten

8.1 Anforderungen und wichtige Konzepte

strukturierte versus halb-strukturierte Daten

strukturiert	**halb-strukturiert**
Schemavereinbarung	
- im *Vorhinein* vereinbart	- im *Einzelfall* gebildet
- *statisch* (zeitunabhängig) bei Einrichtung	- *dynamisch* bei Datenerzeugung
- für alle (Instanz-)Daten *verbindlich*	- *individuell* für einzelnes Datum gültig
- von Instanz(-Daten) "*getrennt speicherbar*"	- zusammen *mit Datum gespeichert*
- "*globale*" Selbstbeschreibung der Daten*gesamtheit*	- "*lokale*" Selbstbeschreibung eines *einzelnen* Datums
Anfragen	
- Schema *datenunabhängig* anfragbar	- Schemas nur *datenabhängig* anfragbar
- Schema-*bezogene* Ausdrucksmittel, ergänzt durch Konstantenzeichen	- "Schema-*erkundende*" Ausdrucksmittel, ergänzt durch Konstantenzeichen
- Navigation in *bekannter* statischer (Hypergraph-)*Schema*struktur	- "*erkundende* Navigation" in *erratener* oder *erfragter* dynamischer *Instanz*struktur
- schon *vor* Ausführung optimierbar	- nur *während* Ausführung optimierbar

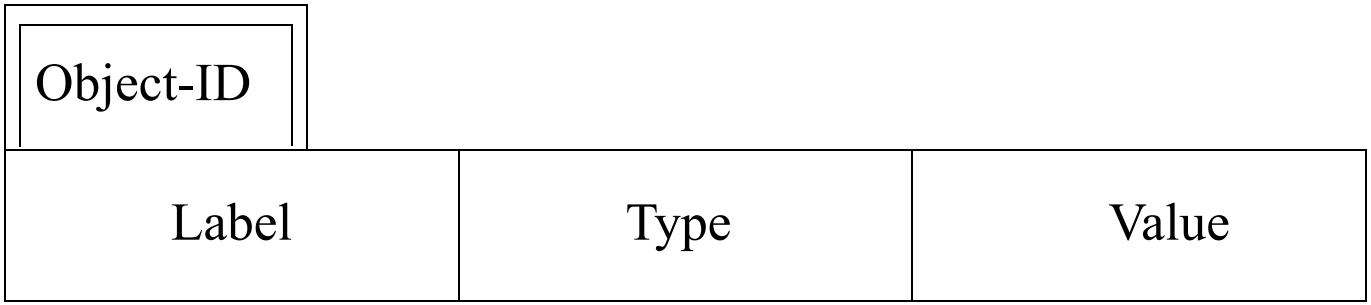
einige Anforderungen an Datenmodelle für halb-strukturierte Daten

- dynamisch angeforderter Austausch und/oder “semantische” Integration
  von heterogenen komplexen Objekten (Daten)
  zwischen autonomen Agenten
  sollen ermöglicht werden
- Austausch und “semantische” Integration
  sollen miteinander verwoben werden
  und halb-algorithmisch durchführbar sein (gegebenenfalls mit Benutzer-Interaktion)
- austauschbare und “semantisch” integrierbare Objekte
  sollen selbstbeschreibend sein
  bezüglich “Benutzer-Ontologie” für erkundende Navigation durch Benutzer und
  bezüglich syntaktischer Typen (Domänen) für syntaktische Analyse durch System
- Objekte sollen mit wenigen, elementaren Konstrukten gebildet werden,
  die (möglichst) “universell” sind
  (bezüglich aller bei beteiligten Agenten eingesetzten Konstrukte)
- im Übrigen: möglichst viele “schöne Seiten” von Datenmodellen für strukturierte Daten (insbesondere des relationalen Datenmodells) sollen erhalten bleiben

Beispiel: OEM (Object Exchange Model)

- entworfen im Rahmen des Projekts TSIMMIS,
  "The Stanford-IBM Manager of Multiple Information Sources"

- **Strukturen:**
  aus elementaren Objekten gebildete,
  dreischichtig überlappende,
  (im Allgemeinen vorzugsweise) baumartige Geflechte für
  - "Benutzer-Ontologie" ("menschliche Semantik", ...) zum Navigieren: Label
  - syntaktische Analyse für automatische Auswertung: Type
  - atomare Werte oder Referenzen: Value



- **Operationen:**
  - Erzeugen, Ändern, Entfernen
  - Anfragen, durch SQL-ähnliche Syntax bezeichnet

Aufbau elementarer OEM-Objekte

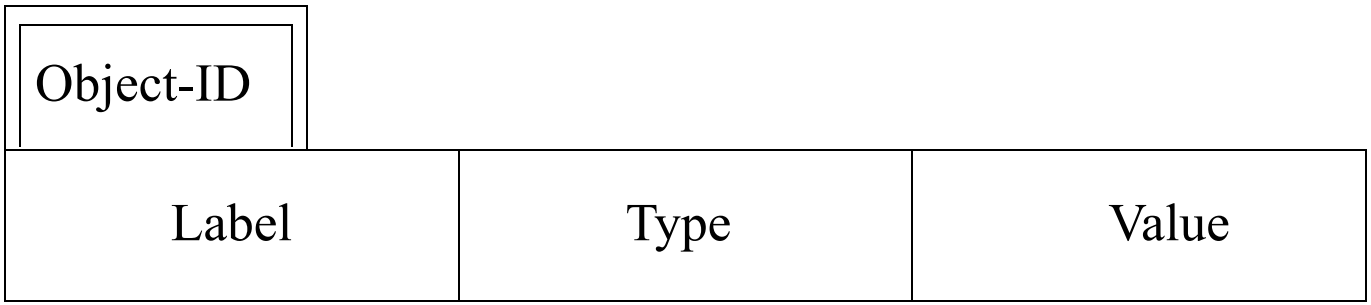
Object-ID:

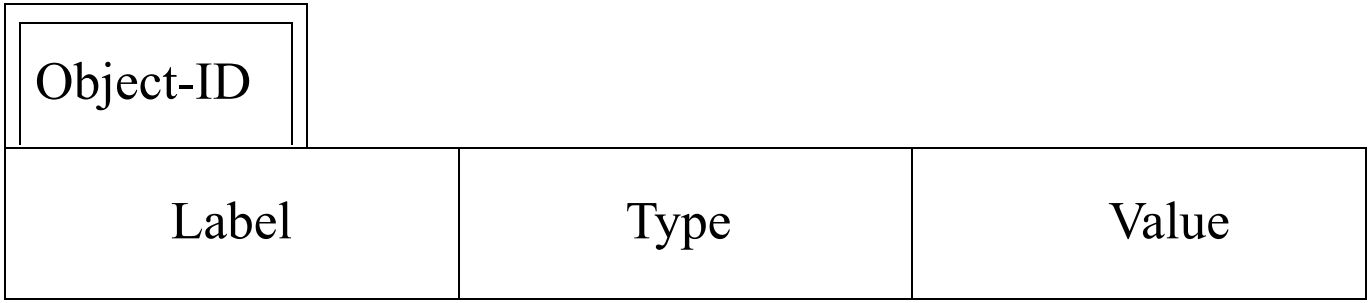
- Local Object Reference: identifiziert ein Objekt lokal,
  z.B. durch Speicheradresse oder lokale OID

- Remote ID: identifiziert ein Objekt in (entfernter) Informationsquelle

- lexical: “druckbar”,
  von Benutzer in Anfragen “als Einstiegspunkt” verwendbar

- non-lexical: “nicht druckbar”,
  vom System bei Anfrageauswertung
  als Belegung von Objektvariablen verwendbar

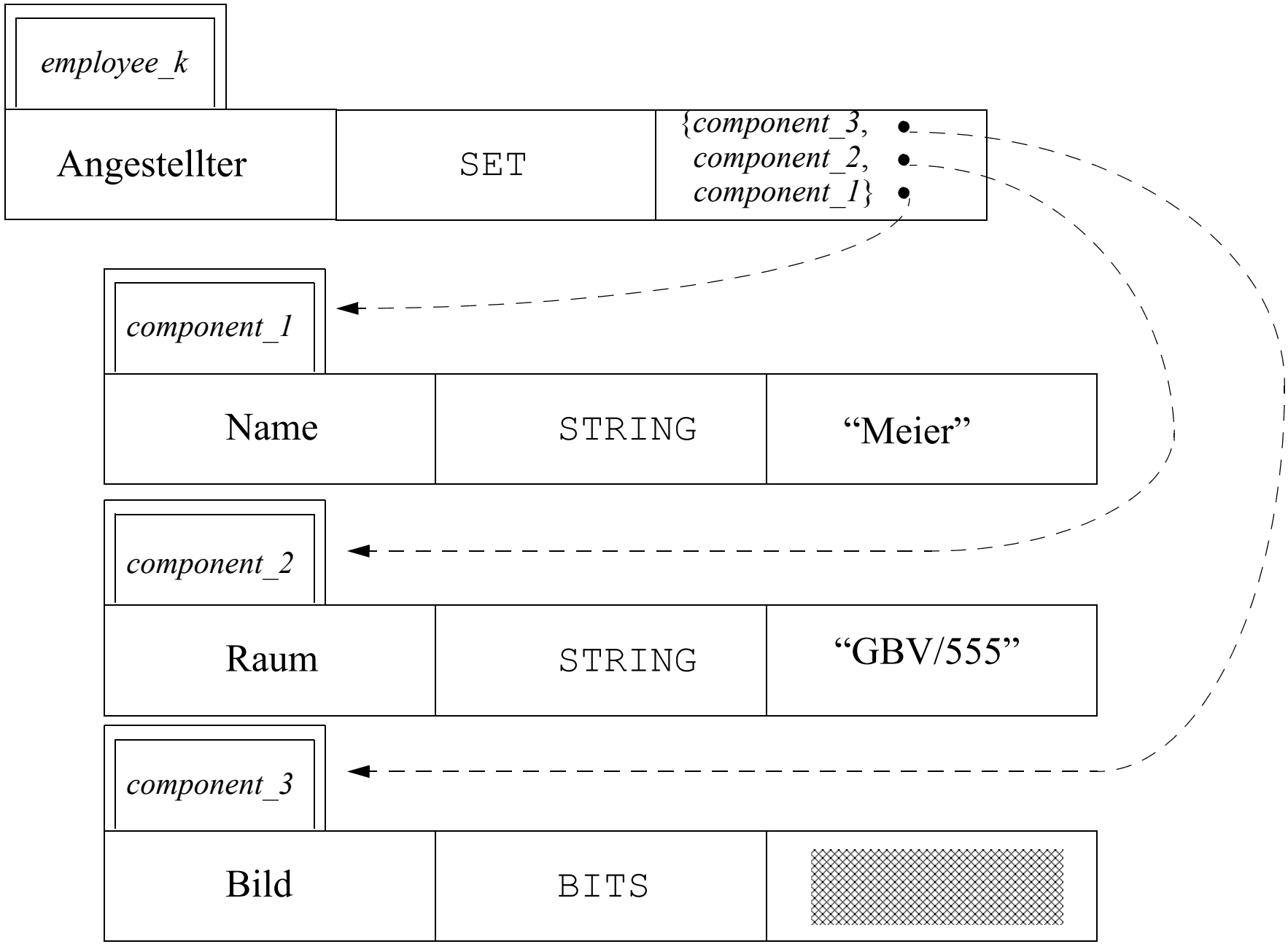




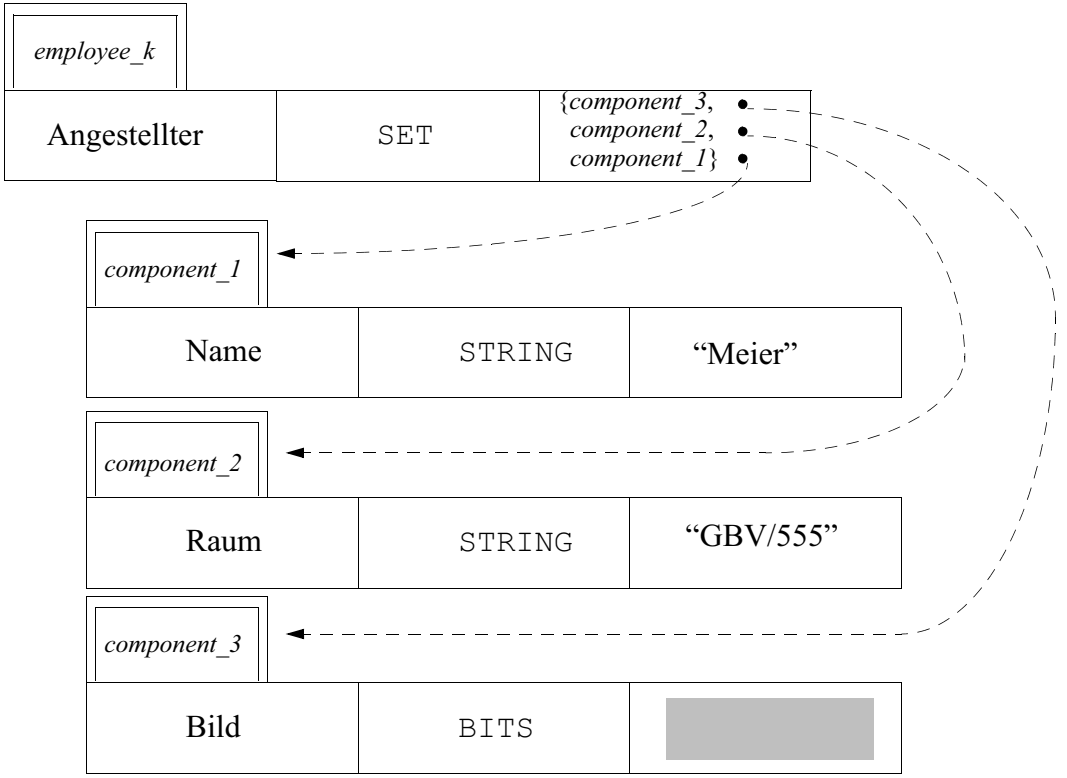
Label:
- Zeichenkette variabler Länge,
- “ontologisch gedeutet” als inhaltliche (Selbst-)Beschreibung
  (aber nicht im Vorhinein durch einen Administrator global verbindlich festgelegt),
  benutzt zum Navigieren bei Anfragen

Type:
- atomic (z.B. `STRING`, `INTEGER`, ...) oder
- `SET` bzw. `LIST`

Value
- zum atomaren Typ passender Wert variabler Länge oder
- passende Menge bzw. Liste von Object-IDs


employee_k
Angestellter
SET
{component_3,
component_2,
component_1}
component_1
Name
STRING
“Meier”
component_2
Raum
STRING
“GBV/555”
component_3
Bild
BITS

graphische Veranschaulichung und textuelle Schreibweise



employee_k is [Angestellter, SET, {*component_1*, *component_2*, *component_3* }]

component_1 is [Name, STRING, “Meier”]
component_2 is [Raum, STRING, “GBV/555”]
component_3 is [Bild, BITS, ▨]

ein Geflecht in textueller Schreibweise

root is [bibliography, SET, {*doc_1*, *doc_2*, ... , *doc_n* }]

doc_1 is [document, SET, {*authors_1*, *topic_1*, *call_1* }]
authors_1 is [author-set, SET, {*author_1_1*}]
author_1_1 is [author-last-name, STRING, "Ullman"]
topic_1 is [topic, STRING, "Databases"]
call_1 is [internal-call-no, INTEGER, 25]

doc_2 is [document, SET, {*authors_2*, *topic_2*, *call-number_2* }]
authors_2 is [author-set, SET, {*author_2_1*, *author_2_2*, *author_2_3*}]
author_2_1 is [author-last-name, STRING, "Aho"]
author_2_2 is [author-last-name, STRING, "Hopcroft"]
author_2_3 is [author-last-name, STRING, "Ullman"]
topic_2 is [topic, STRING, "Algorithms"]
call-number_2 is [dewey-decimal, STRING, "BR273"]

...

doc_n is [document, SET, {*authors_n*, *topic_n*, *call_n* }]
authors_n is [single-author, STRING, "Michael Crichton"]
topic_n is [topic, STRING, "Dinosaurs"]
call_n is [fiction-call-no, INTEGER, 95]

baumartige Geflechte für

Pfade:	**Typen:**	**Adressen/Referenzen mit Werten:**	
bibliography	SET	*root*	
document	SET	*doc_1*	
author-set	SET	*authors_1*	
author-last-name,	STRING	*author_1_1*	:“Ullman”
topic	STRING	*topic_1*	:“Databases”
internal-call-no	INTEGER	*call_1*	:25
document	SET	*doc_2*	
author-set	SET	*authors_2*	
author-last-name	STRING	*author_2_1*	:“Aho”
author-last-name	STRING	*author_2_2*	:“Hopcroft”
author-last-name	STRING	*author_2_3*	:“Ullman”
topic	STRING	*topic_2*	:“Algorithms”
dewey-decimal	STRING	*call-number_2*	: “BR273”
...	...	...	
document	SET	*doc_n*	
single-author	STRING	*authors_n*	:“Michael Crichton”
topic	STRING	*topic_n*	: “Dinosaurs”
fiction-call-no	INTEGER	*call_n*	: 95

Grundform von OEM-Anfragen: Syntax und beabsichtigte Semantik

`SELECT` *Fetch-Expression*

3) wählt für qualifizierte Objekte jeweils Teilobjekt und bildet Objekte für *Value* von Rückgabe-Objekt

(mögliche Erweiterung: ***Konstruktoren*** für komplexe, zusammengesetzte OEM-Objekte)

`FROM` (*SET-)Object*

1) alle im VALUE-Teil des *(Set-)Objects* aufgeführten Objekte werden betrachtet

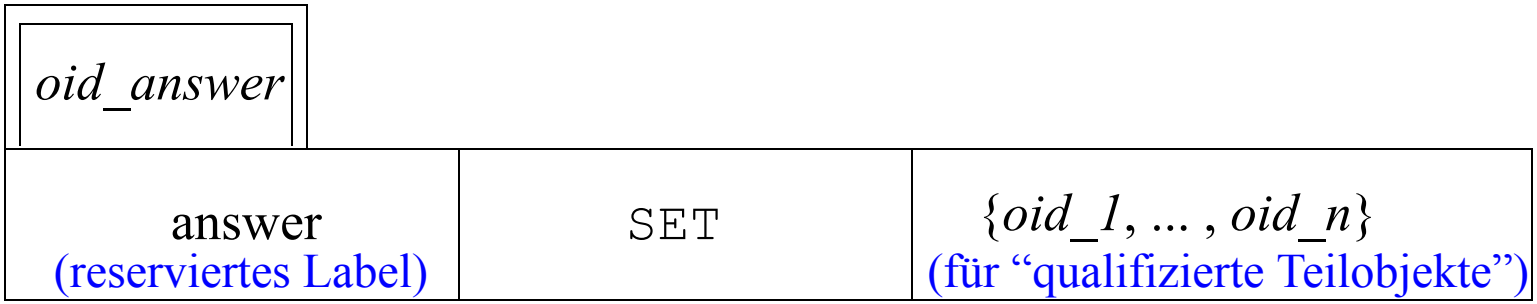
(mögliche Erweiterung: Objekt-Objektvariablen-***Bindungslisten*** für kartesische Produkte)

`WHERE` *Condition*

2) und jeweils untersucht, ob sie sich bezüglich *Condition* qualifizieren;

(Im Wesentlichen: Vergleichs-Prädikate zwischen ***Pfaden*** und ***Konstanten***)

mengenorientiertes, zusammengesetztes Rückgabe-Objekt



Definition von OEM-Pfaden

durch “`.`” getrennte Folgen von

- Label-Ausprägungen (Zeichenketten, ohne “`.`”, “`?`”, “`*`” und “`OID`”)
- “`?`” Platzhalter für Label
- “`*`” Platzhalter für Pfad
- “`OID`” für Rückgabe von *Object-Id(entifier)* anstelle von *Value*, nur als Pfadabschluss,

wobei Folgenglieder durch ***Variablen*** unterschieden werden können

Syntax in BNF

Query ::= `SELECT` *Fetch-Exp* `FROM` *Object* `WHERE` *Condition*

Fetch-Exp ::= *Path* | *Path*`.OID`

Path ::= *Label* | *Label*`.`*Path*

Label ::= string_as_label [`(`variable`)`] | `?` [`(`variable`)`] | `*` [`(`variable`)`]

Object ::= string_as_lexical_object_identifier

Condition ::=
- `true` — immer wahr
- | *Path* — Pfad muss existieren
- | predicate `(` Value`,` ... `,` Value `)` — Werte müssen Prädikat erfüllen
- | *Condition* `and` *Condition* — beide Bedingungen müssen wahr sein

Value ::= *Path* | constant

Beispiel

root	is	[bibliography,	SET,	{*doc_1*, *doc_2*, ... , *doc_n* }]
doc_1	is	[document,	SET,	{*authors_1*, *topic_1*, *call_1* }]
authors_1	is	[author-set,	SET,	{*author_1_1*}]
author_1_1	is	[author-last-name,	STRING,	"Ullman"]
topic_1	is	[topic,	STRING,	"Databases"]
call_1	is	[internal-call-no,	INTEGER,	25]
doc_2	is	[document,	SET,	{*authors_2*, *topic_2*, *call-number_2* }]
authors_2	is	[author-set,	SET,	{*author_2_1*, *author_2_2*, *author_2_3*}]
author_2_1	is	[author-last-name,	STRING,	"Aho"]
author_2_2	is	[author-last-name,	STRING,	"Hopcroft"]
author_2_3	is	[author-last-name,	STRING,	"Ullman"]
topic_2	is	[topic,	STRING,	"Algorithms"]
call-number_2	is	[dewey-decimal,	STRING,	"BR273"]
...				
doc_n	is	[document,	SET,	{*authors_n*, *topic_n*, *call_n* }]
authors_n	is	[single-author,	STRING,	"Michael Crichton"]
topic_n	is	[topic,	STRING,	"Dinosaurs"]
call_n	is	[fiction-call-no,	INTEGER,	95]

Anfrage

```
SELECT    bibliography.document.topic
FROM      root
WHERE     bibliography.document.author-set.author-last-name = "Ullman"
```

(für Qualifikation: es muss Pfad vorhanden und Gleichheitsbedingung erfüllt sein)

liefert zusammengesetztes Rückgabe-Objekt

object_return	is	[answer,	SET,	{*obj_1*, *obj_2*}]
obj_1	is	[topic,	STRING,	"Databases"]
obj_2	is	[topic,	STRING,	"Algorithms"]

Joker (wildcards) in Pfadausdrücken

"?" kann mit *jedem* **Label** belegt werden
(innerhalb einer Anfrage müssen zwei Vorkommen von "?" gleich belegt werden, sofern nicht durch Variablen ausdrücklich unterschieden), z.B. Anfrage

```
SELECT    bibliography.?.topic
FROM      root
WHERE     bibliography.?.internal-call-no
          (für Qualifikation: solch ein Pfad muss vorhanden sein)
```

liefert:

```
object_return   is  [ answer,  SET,     {obj_1}      ]
  obj_1         is  [ topic,   STRING,  "Databases"  ]
```

"*" kann mit *jedem* nichttrivialen **Pfad** belegt werden
(innerhalb einer Anfrage müssen zwei Vorkommen von "*" gleich belegt werden, sofern nicht durch Variablen ausdrücklich unterschieden), z.B. Anfrage

```
SELECT    *.topic
FROM      root
WHERE     *.internal-call-no
          (für Qualifikation: solch ein Pfad muss vorhanden sein)
```

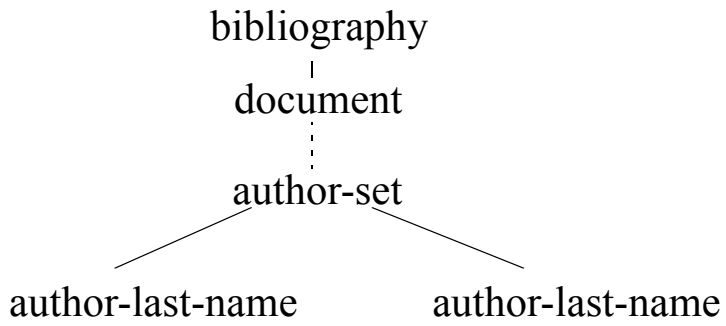
liefert: gleiches Rückgabe-Objekt

Variablen in Pfadausdrücken

unterscheiden mehrfach gewünschte Kanten für Beschreibung von Bäumen, z. B. **Anfrage**

```
SELECT    bibliography . document
FROM      root
WHERE     bibliography . document . * . author-last-name (author_1) = "Ullman"
          and
          bibliography . document . * . author-last-name (author_2) = "Aho"
```

(für Qualifikation: solch ein Baum



muss vorhanden sein)

liefert **Rückgabe-Objekt**

object_return	is	[answer,	SET,	{*obj_1*}]
obj_1	is	[document,	SET,	{*authors_2*, *topic_2*, *call-number_2* }]
authors_2	is	[author-set,	SET,	{*author_2_1*, *author_2_2*, *author_2_3*}]
author_2_1	is	[author-last-name,	STRING,	"Aho"]
author_2_2	is	[author-last-name,	STRING,	"Hopcroft"]
author_2_3	is	[author-last-name,	STRING,	"Ullman"]
topic_2	is	[topic,	STRING,	"Algorithms"]
call-number_2	is	[dewey-decimal,	STRING,	"BR273"]

Rückgabe von Objekt-Identifikatoren

wird durch Abschluss einer Fetch-Expression durch `.OID` verlangt, z.B. **Anfrage**

```
SELECT      bibliography.document.OID
FROM        root
WHERE       *.dewey-decimal
```

(für Qualifikation: solch ein Pfad muss vorhanden sein)

liefert **Rückgabe-Objekt**

object_return is [answer, `SET`, {*doc_2*}]

(*doc_2* ist der Objekt-Identifikator des gewählten Teilobjekts des qualifizierten Objekts)

8.2 XML als Standard

XML (Extensible Markup Language)

- Erweiterung einer Teilsprache von SGML (Standard Generalized Markup Language)
- unter Entwicklung und Standardisierung durch www consortium, W3C:
  `www.w3.org/XML/`
- derzeit das kommerziell meist benutzte Datenmodell für halb-strukturierte Daten
- (mindestens) zwei verschiedene Entwicklungslinien für ***Produkte***:
  - eigenständige XML-Systeme
  - Integration von XML in vorhandene (objekt-relationale) Systeme (z.B. Oracle)

- Entwicklungslinien für Sprachdefinitionen und Einsatzumgebungen, u.a.:
  - DOM, Document Object Model: API für objekt-orientierte Sichten
  - SAX, Simple API for XML: ereignisbasierte API mit Rückmeldungen
  - namespaces: erlaubt URL-ähnliche Referenzen
  - XSL, Extensible Stylesheet Language: für Transformationen von XML Dokumenten
  - XPath, XML PathLanguage: einfache Anfragesprache
  - XPointer, XML Pointer Language: erlaubt erweiterte Navigation mit Referenzen
  - XLink, XML Linking Language: erlaubt "symbolische Referenzen"
  - XHTML, Extensible HTML: Redefinition von HTML mit Hilfe von XML
  - XML Schema: versucht "schöne Seiten" von Schemas (wieder) einzuführen
  - RDF, Resource Description Framework:
    unterstützt Austauch und "semantische" Integration zwischen Agenten
  - XML Anfragesprachen:
    XML-QL,
    LOREL (Leightweight Object Repository Language),
    XML Query Algebra,
    XQuery,
    ...
- ...

Beispiel für XML Dokument (ohne Kopf, mit “Einheitstyp”)

```
< bibliography >

    <document>
        <author-last-name>    Ullman          </author-last-name>
        <topic>               Databases       </topic>
        <internal-call-no>    25              </internal-call-no>
    </document>

    <document>
        <author-last-name>    Aho             </author-last-name>
        <author-last-name>    Hopcroft        </author-last-name>
        <author-last-name>    Ullman          </author-last-name>
        <topic>               Algorithms      </topic>
        <dewey-decimal>       BR273           </dewey-decimal>
    </document>
    ...
    <document>
        <single-author>       Michael Crichton  </single-author>
        <topic>               Dinosaurs       </topic>
        <fiction-call-no>     95              </fiction-call-no>
    </document>

</bibliography >
```

Beispiel für Document Type Definition (DTD)

```
<!ELEMENT bibliography      (document)*                    >

<!ELEMENT document          (  (author-last-name)*
                               (single-author)?
                               topic
                               (internal-call-no)?
                               (dewey-decimal)?
                               (fiction-call-no)?      )  >

<!ELEMENT author-last-name  (#PCDATA)                      >
<!ELEMENT single-author     (#PCDATA)                      >
<!ELEMENT topic             (#PCDATA)                      >
<!ELEMENT internal-call-no  (#PCDATA)                      >
<!ELEMENT dewey-decimal     (#PCDATA)                      >
<!ELEMENT fiction-call-no   (#PCDATA)                      >
```

einige Sprachelemente für DTDs

* :	null oder mehr Vorkommen
+ :	ein oder mehr Vorkommen
? :	null oder ein Vorkommen
(...) :	Reihenfolge-unabhängige “Liste”
(, ... ,) :	(Reihenfolge-abhängige) Liste
#PCDATA :	“parseable character data” (verschiedene OEM-Typangaben im Beispiel ignoriert)

8.3 Oracle und XML

Oracle: bietet sehr umfangreiche XML-Funktionalität

Oracle9*i*: XML Database Developer's Guide - Oracle XML DB
Release 2 (9.2), October 2002, Part No. A96620-02,
1044 pages (!):

- Introducing Oracle XML DB
- Storing and Retrieving XML Data in Oracle XML DB
- Using XML Type APIs to Manipulate XML Data
- Viewing Existing Data as XML
- Oracle XML DB Repository: Foldering, Security, and Protocols
- Oracle Tools that Support Oracle XML DB
- XML Data Exchange Using Advanced Queueing
- Oracle XML DB Case Study: Web Services Retrieve and Display XML Documents
- Appendix A-G

Oracle: Präsidenten-Datenbank als XML Data

- hier als Beispiel und nur auszugsweise

- Oracle Schema: XMLPres

  beispielsweise vereinbart durch:

```
create table XMLPres(
   document    XMLTYPE
)
```

- Oracle Instanz: ein Tupel (mit genau dem Attribute document)

  enthält als Wert jeweils ein XML Dokument für einen Präsidenten

- Aufbau der Tupelwerte: beschreibbar durch Document Type Definition,

  hier aus ursprünglichem Schema gewonnen:

- Attribut ohne “not null” (Death_Age)	durch	“?”
- flach dargestellte “Mengen” (Hobby, Spouse_Name)	durch	“*”
- Enthaltenseinsabhängigkeiten (“references President”)	durch	Nestung (Pfade)

eine DTD für Ausschnitt der Präsidenten-Datenbank

```
<!ELEMENT   president       ( pres_name
                              birth_year
                              years_serv
                              (death_age)?
                              party
                              state_born
                              (hobby)*
                              (pres_marriage)*     )      >

<!ELEMENT   pres_marriage   ( spouse_name
                              pr_age
                              sp_age
                              nr_children
                              mar_year             )      >

<!ELEMENT   pres_name       (#PCDATA)                     >
<!ELEMENT   birth_year      (#PCDATA)                     >
...
<!ELEMENT   mar_year        (#PCDATA)                     >
```

einige Sprachelemente

- XPath-Anfragen werden ausgewertet mit der Funktion

  EXTRACT(<XML-Doc> , <XPath-Expr>)

  wobei:
  <XML-Doc> bezeichnet ein XML-Dokument, d.h. einen Term vom Typ XMLTYPE;
  <XPath-Expr> ist eine auszuwertende XPath-Anfrage;
  alle durch die XPath-Anfrage selektierten “Knoten” werden zurückgeliefert

- zurückgelieferte “Knoten” werden als Zeichenkette dargestellt mit der Funktion

  getStringVal()

- das Vorhandensein eines Pfades mit entsprechendem (Abschluss-)Knoten wird geprüft mit der Funktion

  EXISTSNODE(<XML-Doc> , <XPath-Expr>)

  wobei geliefert wird:
  1 gdw für <XPath-Expr> wird mindestens ein Knoten in <XML-Doc> gefunden
  0 gdw sonst

ein Beispiel für Sprachelemente

- Anfrage umgangssprachlich:

  Welche Präsidenten waren nicht verheiratet?

- Anfrage in Oracle:

```
SELECT   EXTRACT(   document ,   '/president/pres_name'   ).getStringVal()
FROM     XMLPres
WHERE    EXISTSNODE( document ,   '/president/pres_marriage') = 0;
```

werte für die ausgewählten Tupel jeweils
den Pfad von “president” zu “pres_name” aus und
stelle den Abschluss-Knoten (d.h. “pres_name”) als Zeichenkette dar

betrachte alle Tupel in der Instanz zum Relationenschema XMLPres

wähle diejenigen Tupel aus, deren Wert (vom Typ XMLTYPE)
keinen Pfad beinhaltet
vom Knoten “president”
zum Knoten “pres_marriage”

9 Information Retrieval

9.1 Aufgaben, Anforderungen und Ansätze

Information Retrieval Aufgabe

Benutzer: hat jeweils “Informationsbedürfnis”

System: verwaltet Dokumente, die “Information enthalten”

Aufgabe: Benutzer und System
versuchen *zusammen* und gegebenenfalls *iterativ*,
jeweils genau solche Dokumente zu bestimmen und aufzufinden,
die für das “Informationsbedürfnis” ***relevant*** sind

Relevanz: sehr schwieriger Begriff

Relevanz: Schwierigkeiten

- erst *genauer* bestimmbar, *nachdem* Dokument bereits aufgefunden
- nur *näherungsweise* bestimmbar
- nur (Benutzer-)*subjektiv* bewertbar
- recht verschiedenartig beschreibbar

Relevanz: Beschreibungsansätze

im Dokument “enthaltene Information”

- ***erfüllt*** das “Informationsbedürfnis”
- ist ***ähnlich*** dem “Informationsbedürfnis”
- ***enthält Teil*** des “Informationsbedürfnisses”
- ist ***Instanz*** (***Beispiel***) für das “Informationsbedürfnis”
- ist ***spezieller*** als das “Informationsbedürfnis”
- hat ***geringen Abstand*** von dem “Informationsbedürfnis”
- ist ***nicht unabhängig*** von dem “Informationsbedürfnis”
- - ...

Such-Paradox

wie kann man etwas finden,
von dem man gar nichts weiß?

- was man weiß, braucht man nicht (wieder)zufinden!
- was man *nicht* weiß, kann man auch *nicht* (wieder)finden!

Auflösung des Such-Paradoxes

- für Datenmodelle mit strukturierten Daten:
  Fundierung mit Hilfe von Metaschema und Schemas:

  - "Verstecken" (Speichern) ***und*** Suche
    auf gemeinsamer Grundlage einer ***vorausgehenden*** Verständigung:
    einem formalen ***Schema*** und
    der zugrunde liegenden Modellierung mit Hilfe einer Ontologie

  - Speicherer: "versteckt" Konstantenzeichen unter bekanntem Schema
    Anfrager: sucht Konstantenzeichen unter bekanntem Schema

  - wenn man das ***Metaschema*** weiß,
    kann man das (bislang unbekannte) ***Schema*** finden

  - wenn man das ***Schema*** weiß,
    kann man die (bislang unbekannten) "versteckten" ***Konstantenzeichen*** finden

- welche Fundierung *ohne Schema*
  kann man für Information Retrieval finden?

Ostereiersuchen im Wald (Grundlagen)

- (auch unbekannte) Ostereier sind *erkennbar verschieden*
  von allen anderen Dingen im Wald
  (so viel weiß schon jedes Kind)

- Ostereier “verraten sich” durch *äußere Merkmale* wie bunte Farben

- Ostereier sind selten (fast alles im Wald ist *kein* Osterei)

- Ostereier sind *tatsächlich vorhanden*
  (hoffentlich jedenfalls, so dass die Suche lohnt)

Ostereiersuchen im Wald (Verfahren)

- irgendwo muss man *anfangen zu suchen* (notfalls zufällig anfangen)
- wo man ein Osterei findet, könnten *nahe bei* auch noch mehr Ostereier sein
- auch bei *schneller grober Suche* kann man schon einmal ein Osterei finden
- um *alle* Ostereier zu finden, muss man den gesamten Wald durchsuchen
- wenn man genug Ostereier hat, kann man auch *vorzeitig aufhören*
- die Eltern können auch *Hinweise* geben, etwa “heiß” und “kalt”
- Kinder können die “Versteckmethoden” der Eltern *lernen*
- ...

Suchen von Stecknadeln im Heuhaufen (Grundlagen)

- Stecknadeln und Heu(fasern) sind verschieden
- fast alles ist Heu
- zufälliges Suchen ist zwecklos,
  Erfolgswahrscheinlichkeit zu gering: #(Stecknadeln) / #(Heu)
- vollständiges Suchen ist zu aufwändig,
  Laufzeitkomplexität linear in #(Heu),
  statt logarithmisch in #(Heu) oder linear in #(Stecknadel)

Suchen von Stecknadeln im Heuhaufen (Verfahren)

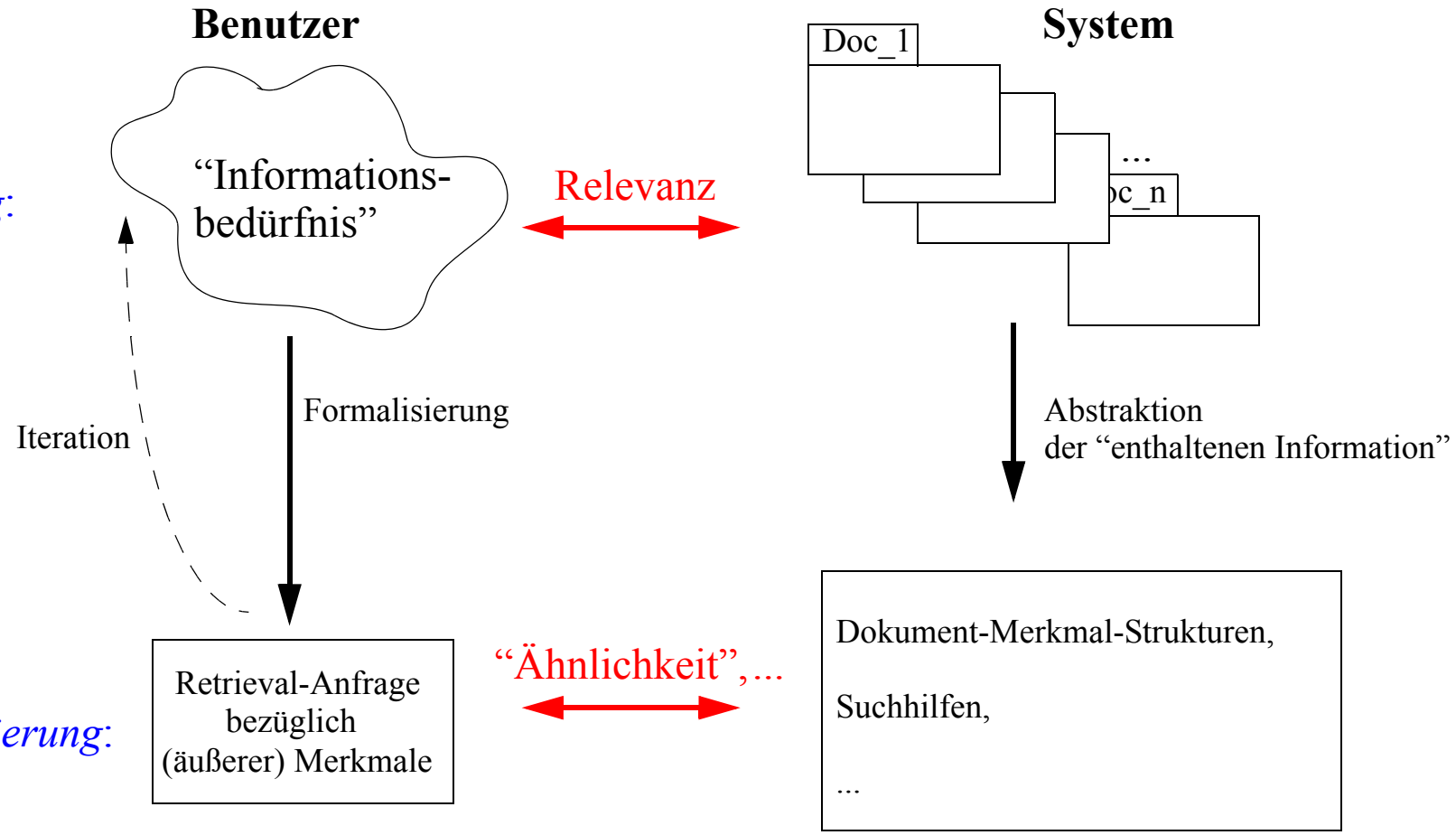
- ***Obermengen*** von Stecknadeln und Heu
  sind ***trennbar*** aufgrund von "***äußeren Merkmalen***", z.B.:

  - Magnetismus: zieht alles (magnetisch) Metallene aus Nichtmetallischem heraus, also auch Stecknadeln aus Heu

  - Feuer: vernichtet zumindest das Heu, lässt auch Stecknadeln übrig

- einfachere ***Restaufgabe***:
  iterierte Suche in nunmehr getrennter Obermenge

eine einfache Modellierung von Information Retrieval



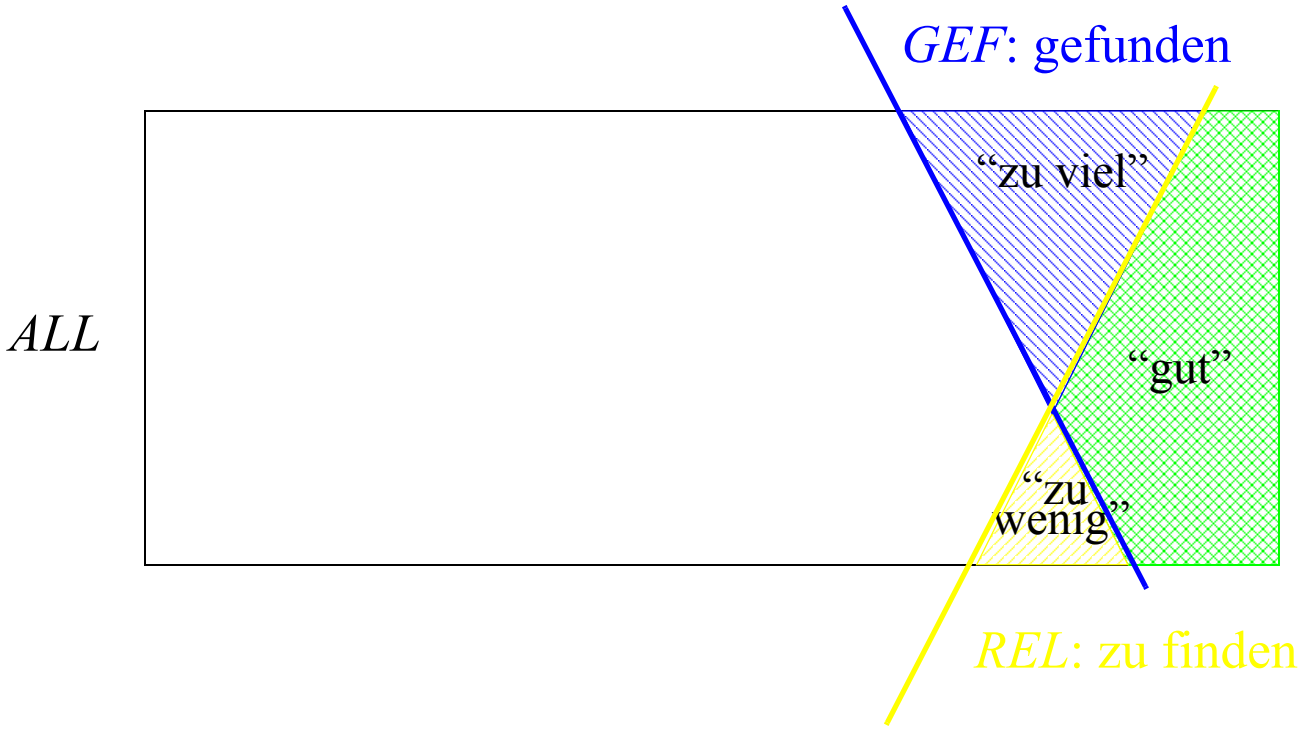
Qualitätsbeurteilung:

Übereinstimmung zwischen

einerseits dem Gesuchten: Relevanz enthaltener Information für Informationsbedürfnis

andererseits dem Gefundenen: "Ähnlichkeit" der Merkmale mit Anfrage

Qualitätsbeurteilung bezüglich (im Allgemeinen fiktiver) Relevanz



Qualitätsbeurteilung bezüglich (im Allgemeinen fiktiver) Relevanz

falls

ALL = Menge *aller* Dokumente

REL = Menge der für Informationsbedürfnis *relevanten* Dokumente:
"zu liefern"
(im Allgemeinen nicht tatsächlich bestimmbar)

GEF = Menge der für Retrieval-Anfrage als ähnlich *gefundenen* Dokumente:
"geliefert"
(von Benutzer *nachträglich* "beurteilbar" bezüglich Relevanz)

dann

$GEF \cap REL$ = "gut geliefert" : gefunden und relevant
(Benutzer-beurteilbar)

$GEF \setminus REL$ = "zu viel geliefert" : gefunden, aber nicht relevant
(Benutzer-beurteilbar)

$REL \setminus GEF$ = "zu wenig geliefert": relevant, aber nicht gefunden
(nicht bestimmbar)

wichtige Verhältnisse

falls

ALL	=	Menge *aller* Dokumente
REL	=	Menge der für Informationsbedürfnis *relevanten* Dokumente, "zu liefern"
GEF	=	Menge der für Retrieval-Anfrage als ähnlich *gefundenen* Dokumente, "geliefert"

dann

$GEF \cap REL$	= "gut geliefert"	: gefunden und relevant	(Benutzer-beurteilbar)
$GEF \setminus REL$	= "zu viel geliefert"	: gefunden, aber nicht relevant	(Benutzer-beurteilbar)
$REL \setminus GEF$	= "zu wenig geliefert"	: relevant, aber nicht gefunden	(nicht bestimmbar)

Precision $p = \frac{|GEF \cap REL|}{|GEF|}$: "gut" versus "geliefert" (Benutzer-beurteilbar)

Recall $r = \frac{|GEF \cap REL|}{|REL|}$: "gut" versus "zu liefern" (nicht bestimmbar)

Fallout $f = \frac{|GEF \setminus REL|}{|ALL \setminus REL|}$: "zu viel" versus "nicht zu liefern"

Benutzer-Beurteilbarkeit von Precision

falls

ALL	=	Menge *aller* Dokumente
REL	=	Menge der für Informationsbedürfnis *relevanten* Dokumente, "zu liefern"
GEF	=	Menge der für Retrieval-Anfrage als ähnlich *gefundenen* Dokumente, "geliefert"

dann

$GEF \cap REL$	= "gut geliefert"	: gefunden und relevant	(Benutzer-beurteilbar)
$GEF \setminus REL$	= "zu viel geliefert"	: gefunden, aber nicht relevant	(Benutzer-beurteilbar)
$REL \setminus GEF$	= "zu wenig geliefert"	: relevant, aber nicht gefunden	(nicht bestimmbar)

Precision $p = \frac{|GEF \cap REL|}{|GEF|}$: "gut" versus "geliefert" (Benutzer-beurteilbar)

beurteilbar allenfalls

- empirisch,
- Benutzer-abhängig
- näherungsweise

"Relevanz" nur ***fiktiv definiert*** als

- gewünschte,
- aber tatsächlich *nicht vollständig* bekannte Beziehung

Recall-Precision-Paare und einfaches Muster für Suchverfahren

Precision $p = \frac{|REL \cap GEF|}{|GEF|}$ Verhältnis von “gut” zu “geliefert”

Recall $r = \frac{|REL \cap GEF|}{|REL|}$ Verhältnis von “gut” zu “zu liefern”

einfaches Muster für Suchverfahren

```
continue := true;
i := select_a_first_object;
gefunden := {};
while continue do
   if query matches doc[i] then insert(doc[i], gefunden);
   i := select_another_object;
   continue := evaluate_achievements(gefunden, i, ... )
               od.
```

bester Fall: jedes gefundene Dokument ist relevant

```
continue := true;
  i := select_a_first_object;
  gefunden := {};
  while continue   do
     if query matches doc[i]  then insert(doc[i], gefunden);
     i := select_another_object;
     continue := evaluate_achievements(gefunden, i, ... )
                    od.
```

- `gefunden` enthält stets nur relevante Dokumente,
  also **stets**: *Precision* = 1 (100%)

- `gefunden` enthält anfänglich kein Dokument,
  also **anfänglich**: *Recall* = 0 (0%)

- `gefunden` enthält schließlich genau die relevanten Dokumente,
  also **schließlich**: *Recall* = 1 (100%)

häufig tatsächlich vorliegender Fall

gefundene Dokumente werden vom System mit einem *Rang* bewertet und gemäß der Bewertung absteigend sortiert ausgegeben;

gefundene Dokumente mit hohem Rang: “eher relevant”,
gefundene Dokumente mit niedrigem Rang: “eher nicht relevant”;

es wird nur ein *Präfix* (Anfangsstück der gesamten sortierten Ausgabe) tatsächlich gezeigt

- **kurzer Präfix** enthält “eher vorwiegend relevante” Dokumente,
  aber noch nicht alle relevanten:

  Precision “eher hoch”, aber mit Länge des Präfixes abnehmend,
  Recall “eher niedrig”, aber mit Länge des Präfixes zunehmend

- **langer Präfix** enthält “eher auch nicht relevante” Dokumente,
  aber insgesamt mehr relevante:

  Precision “eher niedrig”
  Recall “eher hoch”

Wechselwirkungen Benutzer-Interessen versus Benutzer-Aufwand

- Benutzer-Interesse: möglichst *viele* (alle) relevante(n) Dokumente

  widerstreitender Benutzer-Aufwand:

  viele nicht relevante Dokumente “per Hand herausfinden”

- Benutzer-Interesse: möglichst *ausschließlich* relevante Dokumente

  widerstreitender Benutzer-Aufwand:

  Erschwernis bei der Aufgabe,
  die dem “Informationsbedürfnis” zugrunde liegt

Beispiel für Recall-Precision Paare bei Rang-sortierter Ausgabe

Dokument-nummer	$\|GEF\|$	**x = relevant** (für $\|REL\| = 5$)	**"gut"** $\|GEF \cap REL\|$	**Recall** $\frac{\|GEF \cap REL\|}{\|REL\|}$	**Precision** $\frac{\|GEF \cap REL\|}{\|GEF\|}$
588	1	x	1	0.2	1.00
589	2	x	2	0.4	1.00
576	3		2	0.4	0.67
590	4	x	3	0.6	0.75
986	5		3	0.6	0.60
592	6	x	4	0.8	0.67
984	7		4	0.8	0.57
988	8		4	0.8	0.50
578	9		4	0.8	0.44
985	10		4	0.8	0.40
103	11		4	0.8	0.36
591	12		4	0.8	0.33
772	13	x	5	1.0	0.38
990	14		5	1.0	0.36

9.2 einige grundlegende Techniken

Merkmale: Abstraktion

- **Abstraktion** der in einem Dokument “enthaltenen” Information als **Menge von Merkmalen**

Beispiele:
- Schlagworte (key words)
- geometrische Muster (für Bild-Dokumente)
- “Melodie-Themen” (für Musik-Dokumente)
- “Icons” (für Multimedia-Dokumente)
- ...

Merkmale: Extraktion und Normalisierung

- **Extraktion** von Merkmalen:
  - "per Hand" oder (ansatzweise) automatisch
  - "auf Vorrat" (bei Speicherung oder nachträglich) oder auch erst "bei Bedarf" (bei Anfragen)

- **Normalisierung** von Merkmalen:

  Beispiele:
  - grammatische Grundform
  - Skalierung bezüglich eines "Einheitsmaßes"
  - Transposition in "Einheitstonleiter"
  - ...

Merkmale: Retrieval-Anfragen

Syntax:
Boolesche Kombination von (normalisierten) Merkmalen

Semantik:
als "**Ähnlichkeit**" zwischen Anfrage und abstrahierten Dokumenten:

- **Erfüllung** (Wahrheit) der *Merkmalskombination* aus Retrieval-Anfrage bezüglich der *extrahierten Merkmalsmenge* eines Dokuments

- alle in diesem Sinne ähnlichen Dokumente werden geliefert

Merkmale: Beispiel für Retrieval-Anfrage

Dokument	*Merkmale (Schlagworte)*
1	Schema, halb-strukturiert, navigieren
2	Schema, relational, Algebra, Kalkül, SQL, anfragen
3	relational, anfragen

Anfrage	*Rückgabe-Dokumente*
Schema	1, 2
Schema `and` relational	2
relational `and` `(` SQL `or` anfragen `)`	2, 3
relational `and` `(not` SQL `)`	3
Schema `or` relational `or` SQL	1, 2, 3

Merkmale: Bewertung mit Rängen

- Berechnung eines “Grades der Erfüllung”

- Beispiel: **Anzahl der erfüllten Disjunkte** in Boolesch-normalisierter Anfrage

Dokument	***Merkmale (Schlagworte)***
1	Schema, halbstrukturiert, navigieren
2	Schema, relational, Algebra, Kalkül, SQL, anfragen
3	relational, anfragen

normalisierte Anfrage	***1***	***2***	***3***
Schema	1	1	0
Schema `and` relational	0	1	0
`(`relational `and` SQL`)` `or` `(`relational `and` anfragen`)`	0	2	1
relational `and` `(not` SQL`)`	0	0	1
Schema `or` relational `or` SQL	1	3	1

Merkmale: Invertierung mit Dokument-Merkmal-Zugriffsstruktur

für jedes mögliche oder vorkommende einzelne Merkmal
wird die Menge (Liste) der erfüllenden Dokumente unterhalten

Beispiel mit Mengen:

Algebra	2
anfragen	2 , 3
halb-strukturiert	1
Kalkül	2
navigieren	1
relational	2 , 3
Schema	1 , 2
SQL	2

Beispiel mit Merkmals-Bitlisten:

	Dok_1	Dok_2	Dok_3
Algebra	0	1	0
anfragen	0	1	1
halb-strukturiert	1	0	0
Kalkül	0	1	0
navigieren	1	0	0
relational	0	1	1
Schema	1	1	0
SQL	0	1	0

Merkmale: effiziente Auswertung von Retrieval-Anfragen durch Operationen auf der Dokument-Merkmal-Zugriffsstruktur

Beispiel mit Mengen:
Mengenoperationen

Anfrage: relational `and` (SQL `or` anfragen)

Auswertung: `set`(relational) $\cap_{\texttt{set}}$ (`set`(SQL) $\cup_{\texttt{set}}$ `set`(anfragen))

$= \{2, 3\}$ $\cap_{\texttt{set}}$ ($\{2\}$ $\cup_{\texttt{set}}$ $\{2, 3\}$)

$= \{2, 3\}$

Merkmale: effiziente Auswertung von Retrieval-Anfragen durch Operationen auf der Dokument-Merkmal-Zugriffsstruktur

Beispiel mit Merkmals-Bitlisten:
komponenten-weise ausgeführte Bit-Operationen

Anfrage: relational `and` (SQL `or` anfragen)

Auswertung: `bit`(relational) $\cap_{\texttt{bit}}$ (`bit`(SQL) $\cup_{\texttt{bit}}$ `bit`(anfragen))

$$= \begin{bmatrix}0\\1\\1\end{bmatrix} \cap_{\texttt{bit}} \left(\begin{bmatrix}0\\1\\0\end{bmatrix} \cup_{\texttt{bit}} \begin{bmatrix}0\\1\\1\end{bmatrix} \right)$$

$$= \begin{bmatrix}0\\1\\1\end{bmatrix}$$

aus Effizienzgründen: Komprimierung der (im Allgemeinen "dünnen") Bitlisten

hierarchische Klassifikation (1)

- **besondere Merkmale:**

  inhaltlich zusammengehörige, *hierarchische* Klassen,
  anwendungsbezogene “Ontologie”, z.B. ACM Computing Reviews Classification:

Hauptklassen:	A.	General Literature
	B.	Hardware
	C.	Computer Systems Organization
	D.	Software
	E.	Data
	F.	Theory of Computation
	G.	Mathematics of Computing
	H.	Information Systems
	I.	Computing Methodologies
	J.	Computing Applications
	K.	Computing Milieux
Unter-Unterklassen:, z.B.:	H.3	Information Storage and Retrieval
	H.3.0	General
	H.3.1	Content Analysis and Indexing
	H.3.2	Information Storage
	H.3.3	Information Search and Retrieval
	H.3.4	System and Software
	H.3.5	Online Information Systems
	H.3.6	Library Automation
	H.3.m	Miscellaneous

hierarchische Klassifikation (2)

- **Extraktion:** als Klassifikation, **meistens "per Hand"**
- **Normalisierung:** vermöge vorgegebener Klassenbezeichner
- **Retrieval-Anfragen:** wie allgemein bei Merkmalen,
  mit Berücksichtigung (Expansion) der Hierarchie
- **Invertierung:** wie allgemein bei Merkmalen,
  mit Berücksichtigung (Expansion) der Hierarchie

flaches Clustering (1)

- **besondere Merkmale:**
  ***mutmaßlich* inhaltlich zusammengehörige, *disjunkte* Klassen,**
  z.B. Clustering gemäß **Verweisstrukturen von Web-Seiten**

Dokumente: Web-Seiten

Verweisstruktur: Web-Seite *A* enthält (textuell) einen Verweis (als URL) auf Web-Seite *B*

Beobachtungen:
- manche Web-Seiten liegen im Zentrum vieler Verweise
- manche Web-Seiten verweisen wechselseitig aufeinander

Annahme: Verweise deuten daraufhin,
dass enthaltene “Informationsgehalte zusammengehörig” sind,
so dass betroffene Web-Seiten
als “ähnlich” behandelt werden können

flaches Clustering (2)

- **algorithmische Clusterbildung**, z.B. mit folgender Heuristik:
  - fasse Web-Seiten eines vollständigen Unter-Verweisgraphen zu einem Cluster zusammen
  - falls eine Web-Seite ausschließlich in genau ein Cluster verweist, so füge sie diesem Cluster hinzu
  - ...

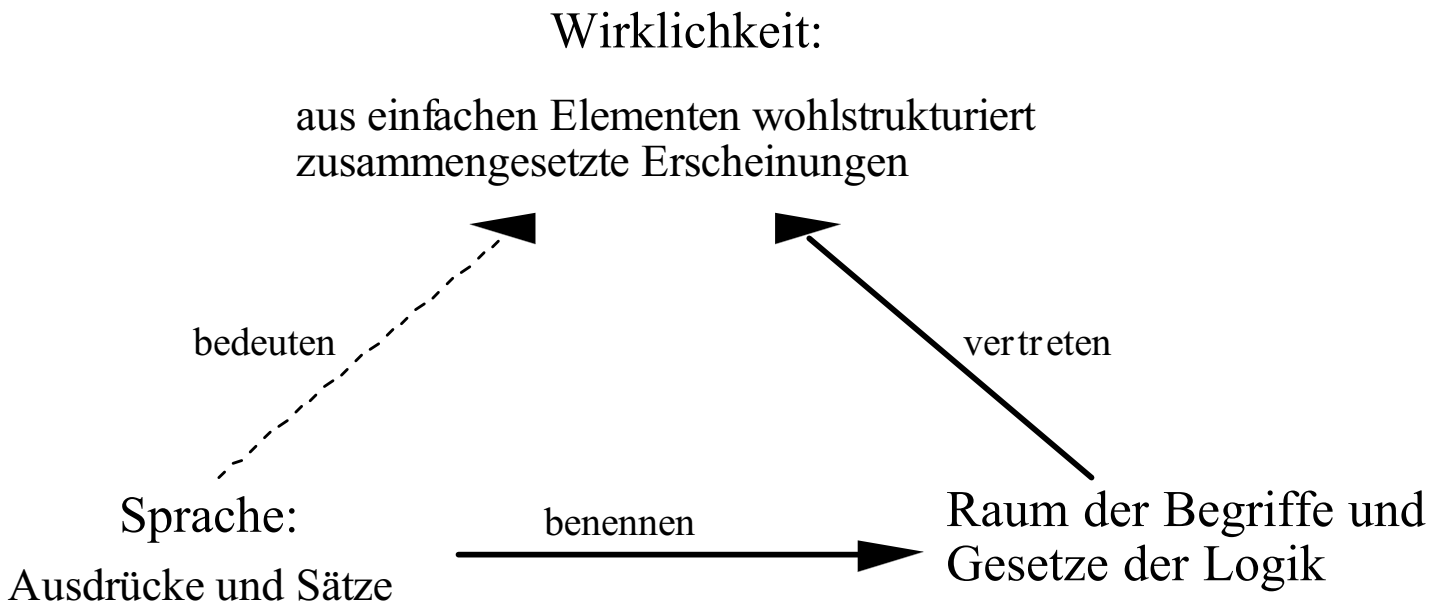
- **Retrieval-Anfragen**

  liefern mit einer (auf andere Weise) gefundenen Web-Seite (ggf. erst auf Anforderung) auch die anderen Web-Seiten des Clusters

Thesaurus

- eine geordnete Zusammenstellung von Begriffen
  mit ihren natürlichsprachlichen Benennungen

- auffassbar als operationalisierter Ausschnitt
  des “semiotischen Dreiecks” Wirklichkeit-Begriffsraum-Sprache
  als Teil einer anwendungsbezogenen Ontologie:



Thesaurus: terminologische Kontrolle der *Benennungen*

- **linguistische Normalisierung**

  (Beispiel: grammatische Grundform,
  Gleichsetzung von Verb und Substantiv, ...)

- Erfassung von **Synonymen**

  (syntaktisch verschiedene Benennungen für semantisch gleiche Konzepte,
  Schreibweisenvarianten, unterschiedliche Sprachstile, ...)

- Kennzeichnung von **Homonymen** und **Polysemen**

  (syntaktisch gleiche Benennungen für semantisch verschiedene Konzepte,
  bei Homonymen: von Betonung beim Sprechen abhängig)

- Festlegung von **Vorzugsbenennungen** (Deskriptoren)

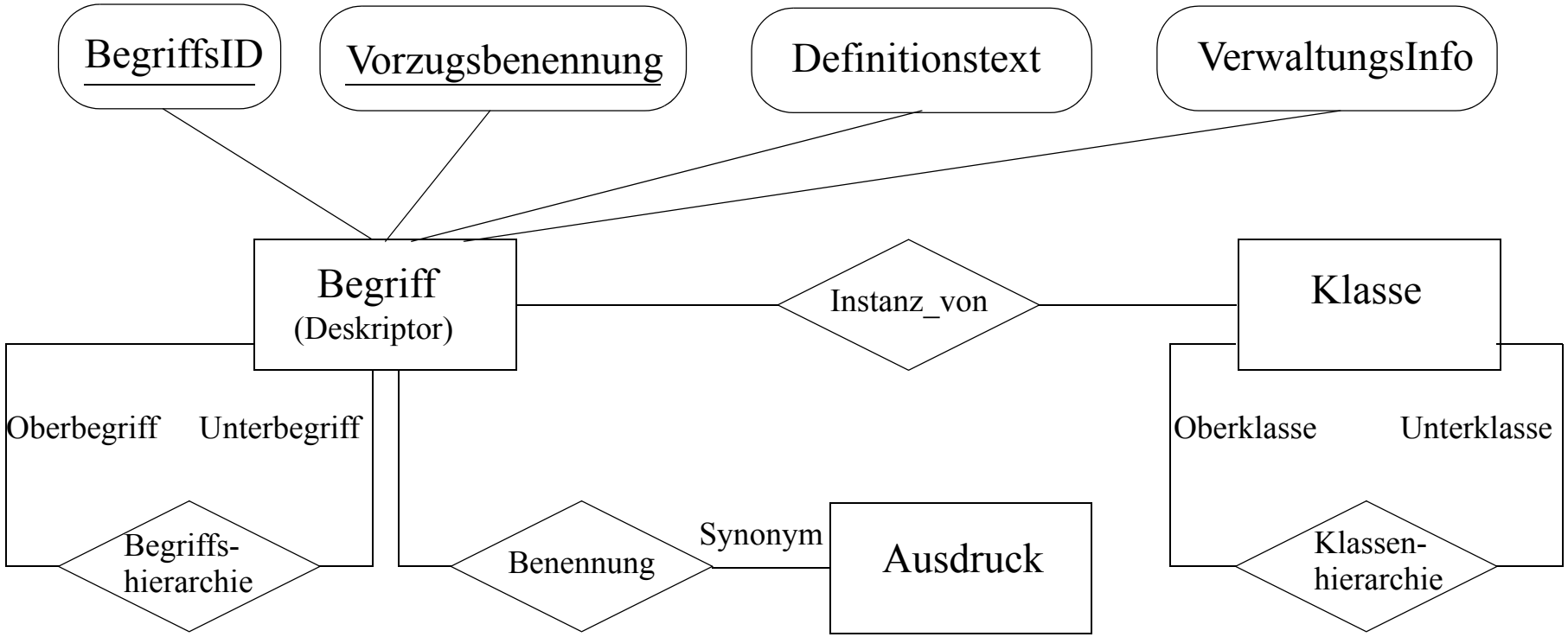
Thesaurus: Beziehungen zwischen *Begriffen*

bereits unabhängig von Benennungen,
im Hinblick auf vertretene Wirklichkeit:

- Oberbegriff-Unterbegriff
- hierarchische Klassifikation
- “Ähnlichkeit”
- - ...

Thesaurus: strukturierte Darstellung

z.B. als (relationale oder objekt-orientierte) Formalisierung folgender grober Modellierung (durch ein ER-Diagramm):



Thesaurus: Zugriffsstrukturen

zum Beispiel

- **Index** für alphabetischen Zugriff zu Ausdrücken (oder Vorzugsbenennungen)
- **Liste** für Klassen-systematischen Zugriff zu Vorzugsbenennungen
- **KWIC** (keyword in context) als Index
  für kontextbezogenen Zugriff zu Vorzugsbenennungen
- - ...

9.3 Modelle für Information Retrieval

Modelle mit Merkmals-Vektoren: Strukturen

- **zugrunde liegende Abstraktion**
  - mehrdimensionale (heterogene) Vektoren mit Merkmalen (oder Merkmalsmengen)
  - Abstandsmaß zwischen Vektoren

- **Dokument-Abstraktion:**
  - Zuordnung von Dokumenten (Objekten) zu Vektoren
  - Dokument-Vektoren bilden (im Allgemeinen dünne) Teilmenge des Gesamtraums

- **Informationsbedürfnis-Formalisierung**:

  Bildung eines Vektors

Modelle mit Merkmals-Vektoren: Operationen

- **grundlegende Anfragetypen:**
  - bestimme "**nächsten Nachbarn**"
    (des Anfrage-Vektors unter vorkommenden Dokument-Vektoren)
  - bestimme "δ-**Umgebung**"
    (des Anfrage-Vektors bezüglich vorkommender Dokument-Vektoren)

- **mehrdimensionale Such-(Zugriffs-)Strukturen für "Regionen"**
  - (approximative) Suche auf erfolgversprechende Regionen begrenzen

ein (stark vereinfachtes) wahrscheinlichkeitstheoretisches Modell

- **Relevanz als (fiktive) Wahrscheinlichkeit:**

  bei gegebener **Anfrage** q:

  jedem **Dokument** d wird zugeordnet

  die (fiktive) **Wahrscheinlichkeit** $Prob(R_{q,d})$

  seiner **Relevanz** für das durch q formalisierte Informationsbedürfnis

- **Entscheidungssituation für ein einzelnes Dokument:**

Fall 1: Dokument relevant und Dokument geliefert : "gut"

Fall 2: Dokument relevant und Dokument nicht geliefert : Verlust $l_{zuwenig}$

Fall 3: Dokument nicht relevant und Dokument geliefert : Verlust l_{zuviel}

Fall 4: Dokument nicht relevant und Dokument nicht geliefert: richtig

- **Minimierung des Verlusts bei Entscheidung über Lieferung**:

  Fall 1+3:

  Dokument *d* liefern bringt erwarteten Verlust: $(1 - Prob(R_{q,d})) \cdot l_{zuviel}$

  Fall 2+4:

  Dokument *d* nicht liefern bringt erwarteten Verlust: $Prob(R_{q,d}) \cdot l_{zuwenig}$

  also:

  Dokument *d* liefern gdw $Prob(R_{q,d}) \cdot l_{zuwenig} > (1 - Prob(R_{q,d})) \cdot l_{zuviel}$

  $$\text{gdw} \quad \frac{Prob(R_{q,d})}{1 - Prob(R_{q,d})} > \frac{l_{zuviel}}{l_{zuwenig}}$$

  $$\text{gdw} \quad Prob(R_{q,d}) > \frac{l_{zuviel}}{l_{zuviel} + l_{zuwenig}}$$

- **Retrieval-Antwort für *q*:**

  - alle Dokumente *d* mit $Prob(R_{q,d}) > \frac{l_{zuviel}}{l_{zuviel} + l_{zuwenig}}$

  - nach Wahrscheinlichkeiten absteigend sortiert

Schätzung der fiktiven Wahrscheinlichkeiten

“unbekanntes” event: Relevanz eines Dokuments d

“beobachtbare” condition: Vorkommen eines Merkmals x
(im Allgemeinen zusammengesetzt)

dann ersetzen:

statt “Relevanz eines Dokuments d”, d.h.
$Prob(R_{q,d})$

einfacher “Relevanz eines Dokuments
mit beobachtbarem Merkmal x”:
$Prob(R_{q,x})$

empirisch näherungsweise bestimmen anhand von Stichproben:

$$Prob(R_{q,x})$$

Schätzung vom Merkmal abhängiger Parameter

- ersatzweise:

  anstelle von $Prob(R_{q,d})$

  andere **Parameter schätzen**,
  die vom Merkmal x abhängig sind

- man verwendet typischerweise den **Satz von Bayes**:

$$Prob(event|condition) \cdot Prob(condition) = Prob(condition|event) \cdot Prob(event)$$

- man sichert **Invarianz der Rangordnung**

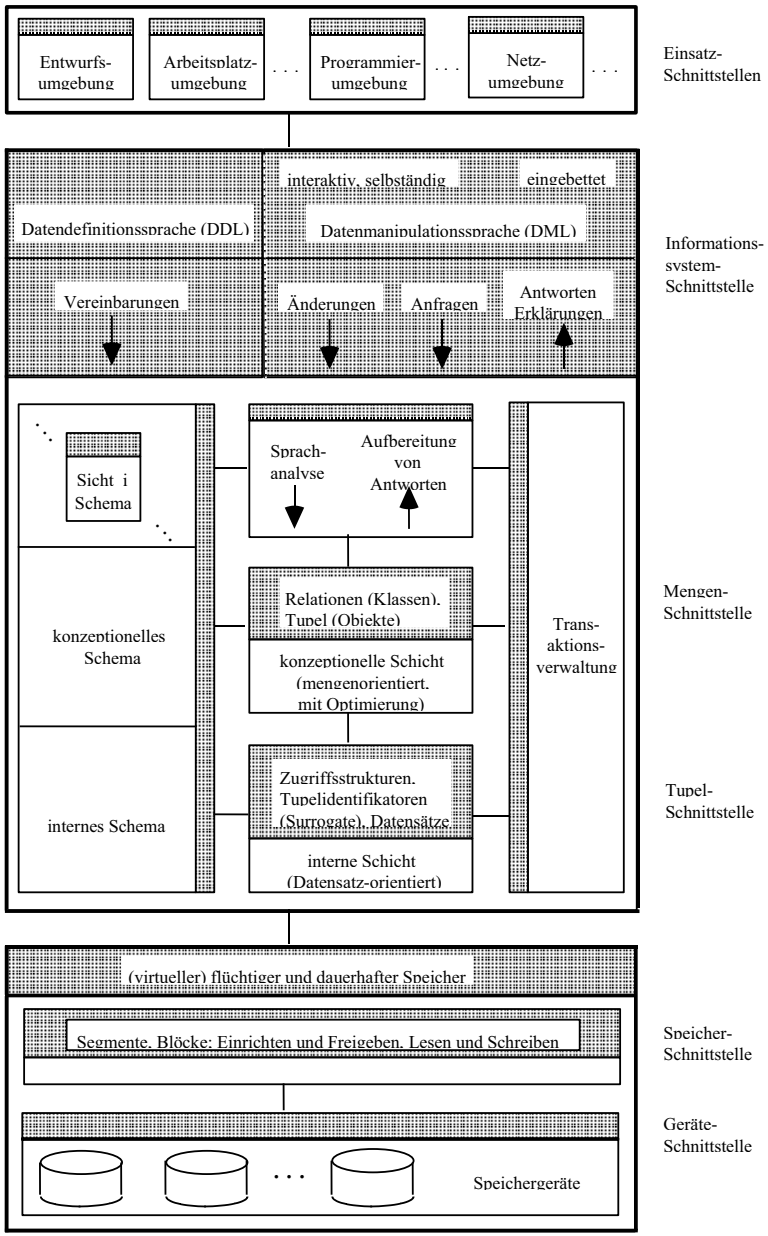
Teil IV

Effizienz (in relationalen Systemen)

10 Zugriffsstrukturen und Verbund-Algorithmen

10.1 Anforderungen an die interne Schicht

Verwirklichung der grundlegenden relationalen Operationen



verlangt geeignete
Datenstrukturen und **Algorithmen**,
um insbesondere:

große Mengen von Daten

dauerhaft und

effizient zu verwalten, d.h.
Anfragen und ***Änderungen***
zu bearbeiten

Dauerhaftigkeit: Identifikation von Tupeln

- aus der *Sicht des Benutzers* durch seine ***Schlüsselwerte***
- auf der Ebene der *Speichermedien* durch seine ***Adresse***
- auf der Ebene des *Datenbanksystems* durch einen ***Tupelidentifikator,*** TID

Dauerhaftigkeit: Tupelidentifikator

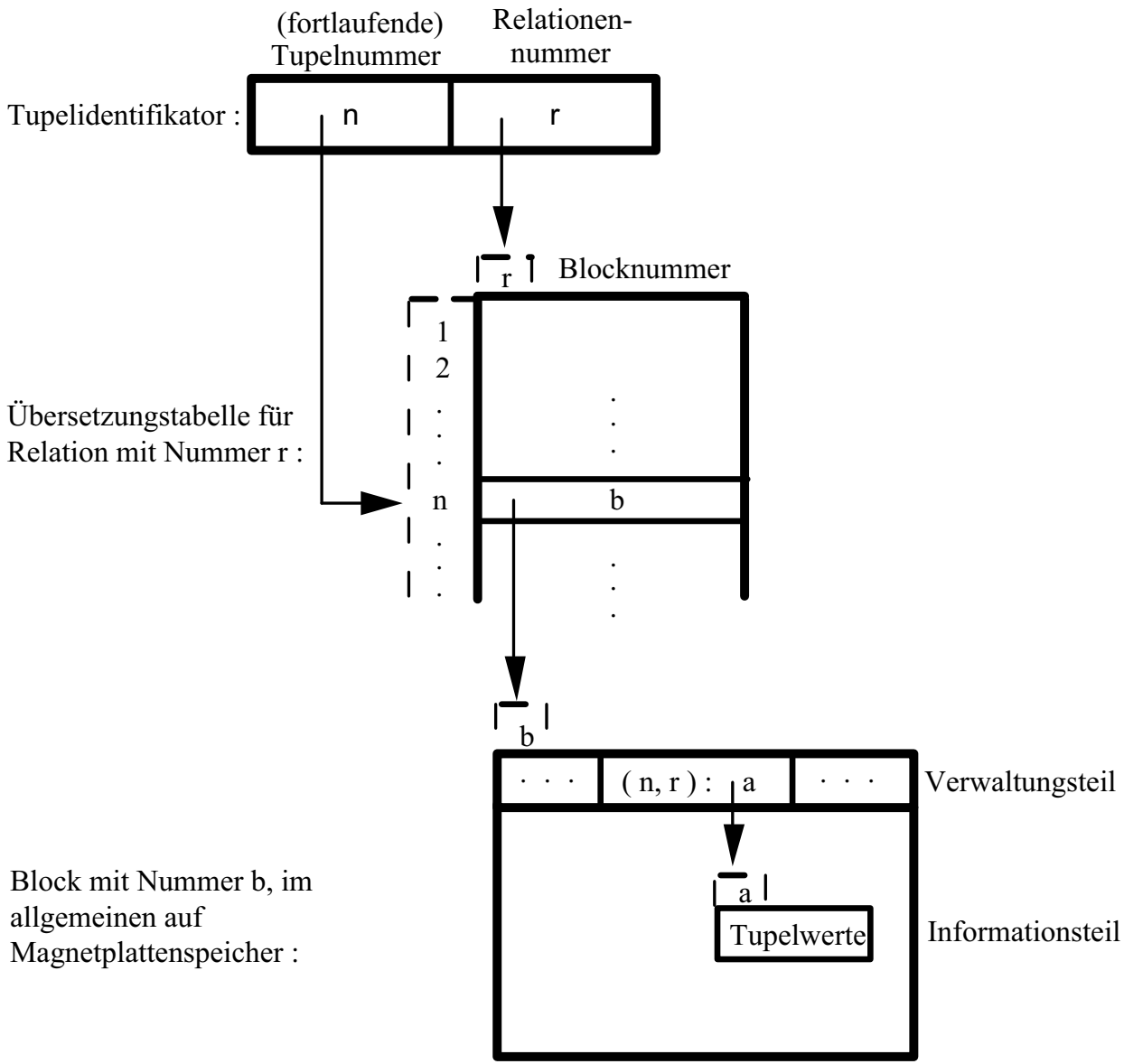
- wird bei der erstmaligen Eingabe eines Tupels erzeugt
- automatisch durch das System
- im Allgemeinen für den Benutzer unsichtbar
- systemweit eindeutig
- über die Zeit unveränderbar
- mit dem eigentlichen Tupel stets “verbunden”

Dauerhaftigkeit: Lebensdauer eines Tupels

- von der benutzergesteuerten Eingabe
  bis zum benutzergesteuerten Entfernen

- währenddessen bleibt ein ***Tupelidentifikator***
  stets ***invariant*** und ***dauerhaft***,
  insbesondere also:
  - bei benutzergesteuerten Änderungen der Schlüsselwerte
  - bei Verlagerungen innerhalb der Speichermedien
  - zwischen zwei Sitzungen eines Benutzers
  - bei Systemzusammenbrüchen

Bestimmung von Adressen aus Tupelidentifikatoren



Effizienz für Anfragen und Änderungen

stark abhängig von der
effizienten ***Umrechnung der drei Identifikationsgrößen*** eines Tupels:

- benutzersichtbare Schlüsselwerte
- Tupelidentifikator
- Speicheradresse

genauer:
abhängig vom ***Aufwand***
wichtiger ***Operationen der internen Schicht***

wichtige Operationen der internen Schicht

- **schlüsselbezogenes Suchen**

  Umwandlung
  "benutzersichtbare Schlüsselwerte $\mapsto$ Tupelidentifikator (Speicheradresse)"

- **physisches Suchen und Transport**

  Umwandlung
  "Tupelidentifikator $\mapsto$ Speicheradresse",
  mitsamt gegebenenfalls Transport des Tupels
  vom Hintergrundspeicher in den Hauptspeicher

- **inhaltsbezogenes Suchen**

  Bestimmung
  "(benutzersichtbare) Attributwerte $\mapsto$ Tupelidentifikator (Speicheradresse)",
  etwa für die A=c-Selektion bzw. für den natürlichen Verbund

- **Relationendurchlauf**

  systematischer vollständiger Durchlauf einer Relation

10.2 Zugriffsstrukturen zur Steigerung der Effizienz

Zugriffsstrukturen

- **Ziele:**

  Effizienzanforderungen erfüllen
  beim schlüsselbezogenen und inhaltsbezogenen Suchen sowie
  beim Relationendurchlauf

- **Ansatz:**

  speichern der eigentlichen Relationen mit ihren Tupeln
  (***primäre*** Information)
  und zusätzlich *Hinweise zum Auffinden* dieser Tupel
  (***sekundäre*** Information)

- **Wechselwirkungen:**

  entstehende Redundanz
  - vergrößert den ***Speicheraufwand***,
    aber verringert im Mittel erheblich den ***Zeitaufwand für Anfragen***
  - erhöht den ***Zeitaufwand für das Einfügen***,
    aber verringert den ***Zeitaufwand für das Entfernen***

gängige Zugriffsstrukturen:

- ***sequentielle Listen***,

  etwa durch Felder oder Verkettungen verwirklicht,
  zur Unterstützung von Relationendurchläufen

- ***Indexe***,

  etwa als B*-Bäume oder durch Hash-Verfahren verwirklicht,
  zur Unterstützung des Suchens in einer Relation

- ***Links***,

  zur Unterstützung des gleichzeitigen Suchens in mehreren Relationen

- ***Hashfunktionen***,

  zur verkleinernden Filterung einer Relation im Hinblick auf eine andere

B*-Baum als Index

- **Verwendung:**
  - *Index* für schlüsselbezogenes Suchen
  - *sequentielle Liste* für systematischen Durchlauf der TIDs einer Relation

- **Voraussetzung:**
  auf der Domäne der Schlüsselwerte ist eine lineare Ordnung definiert

- **Eigenschaften:**
  bekannt aus DAP

B*-Baum: Wiederholung

- *teil-ausgeglichener*, *geordneter* Baum, dessen *Blätter alle die gleiche Höhe* besitzen
- ein Blattknoten enthält eine geordnete Folge von Paaren
  < (benutzersichtbarer) Schlüsselwert, TID >
- zusätzlich werden die Blattknoten untereinander doppelt verkettet
- durchläuft man alle Blätter von links nach rechts,
  so erhält man die sortierte Folge (ohne Wiederholung)
  aller (derzeit) vorhandenen Schlüsselwerte
- ein Elternknoten enthält eine geordnete Folge der Art
  Verweis auf Kind, *Schlüsselwert*, *Verweis auf Kind*, ... ,
  Schlüsselwert, *Verweis auf Kind*
- durchläuft man alle Knoten in inorder-ähnlicher Weise,
  so erhält man eine sortierte Folge (mit Wiederholung)
  aller (derzeit) vorhandenen Schlüsselwerte
- der Baum *wächst von den Blättern zur Wurzel*,
  indem ein voller Knoten geteilt wird und
  die entsprechenden Verweise im Elternknoten eingetragen werden
- die Größe eines Knotens ist im Allgemeinen die
  für den *Hintergrundspeicher verwendete Blockgröße*

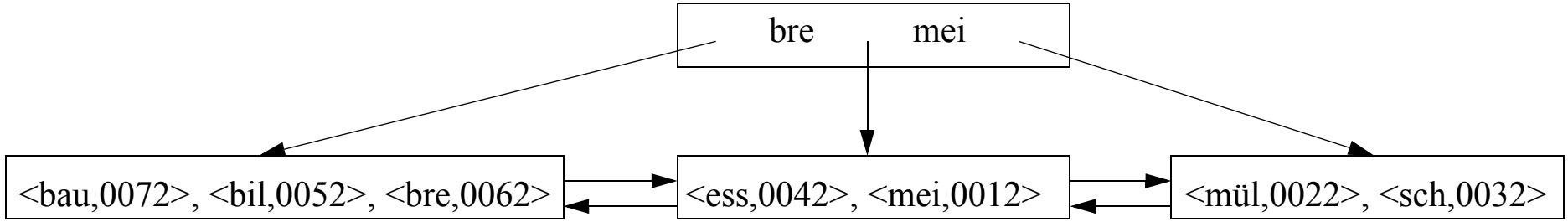
Beispiel für einen B*-Baum

Relation:	`mitglied`
Schlüsselattribut:	`Name`
Eigenschaftsattribut:	`Ort` und gegebenenfalls weitere Attribute
interne Relationennummer:	2
fortlaufende Tupelnummern:	dezimal dreistellig
Knotengröße:	höchstens 7 Einträge (Attributwert, TID oder Verweis)

Reihenfolge der Tupeleingaben:

a.	meier	dort(mund)
b.	müller	boch(um)
c.	schmidt	bonn
d.	esser	aach(en)
e.	biller	dort(mund)
f.	brenner	dort(mund)
g.	bauer	duis(burg)

sich ergebender B*-Baum:



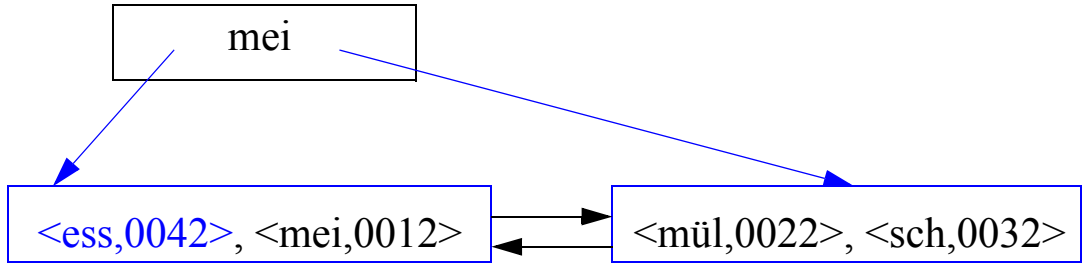
schrittweiser Aufbau des Beispiels, Einfügungen a-e

a. Einfügung [meier, dortmund]: <mei,0012>

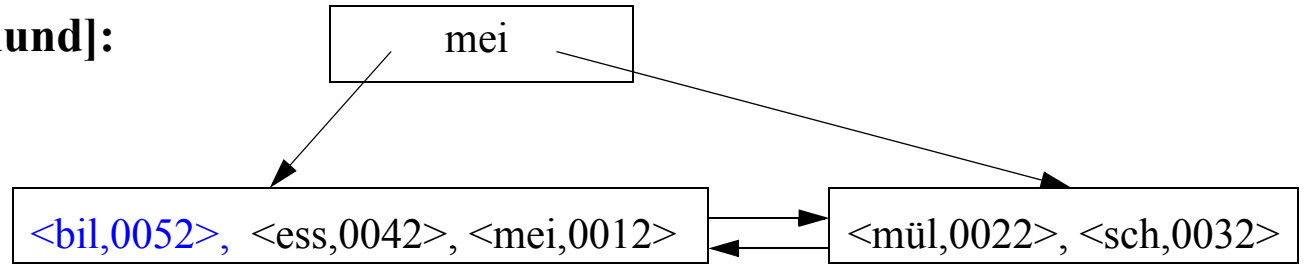
b. Einfügung [müller, bochum]: <mei,0012>, <mül,0022>

c. Einfügung [schmidt, bonn]: <mei,0012>, <mül,0022>, <sch,0032>

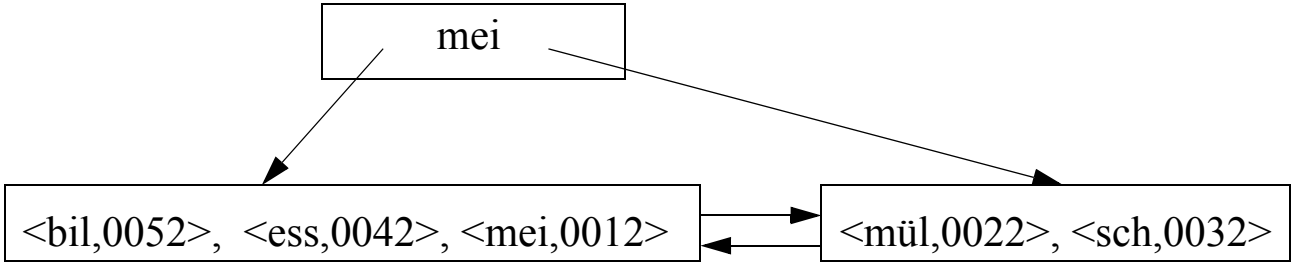
d. Einfügung [esser, aachen]:



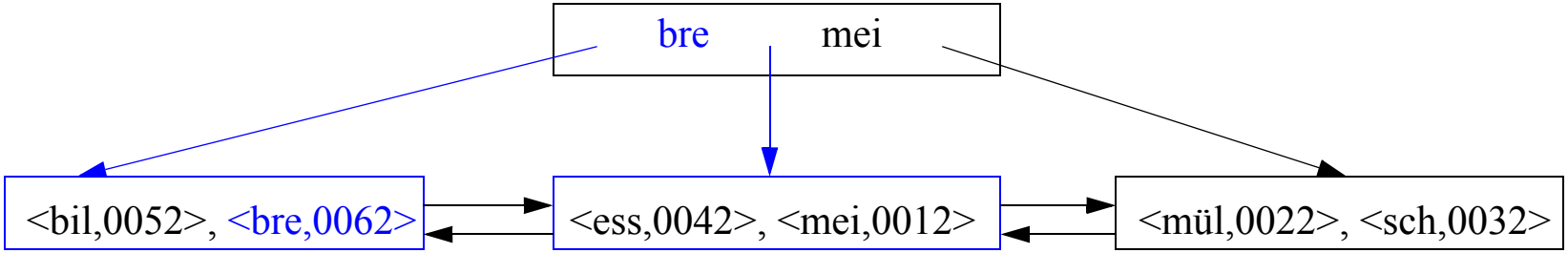
e. Einfügung [biller, dortmund]:



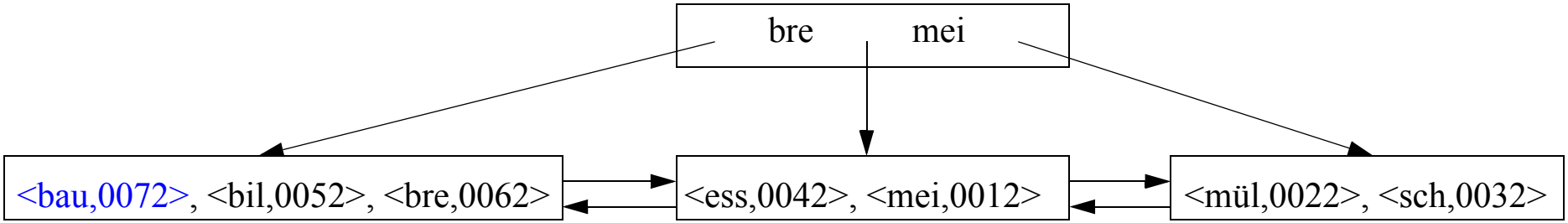
schrittweiser Aufbau des Beispiels, Einfügungen f-g



f. Einfügung [brenner, dortmund]:



g. Einfügung [bauer, duisburg]:

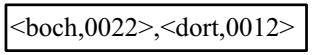


Index bezüglich eines Nichtschlüsselattributs

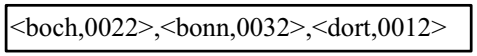
- zur Unterstützung inhaltsbezogener Suche
- als B*-Baum mit Blattknoten der Form
  < (benutzersichtbarer) Attributwert, *Folge* von TIDs >
- Beispiel für das Nichtschlüsselattribut `Ort`

a.

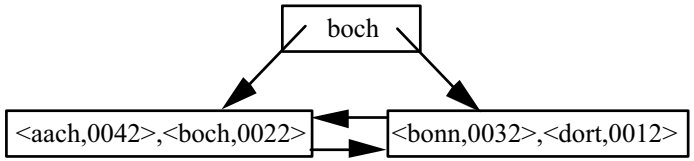
b.



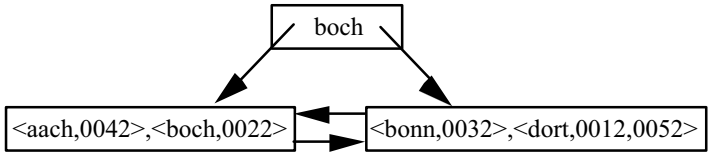
c.



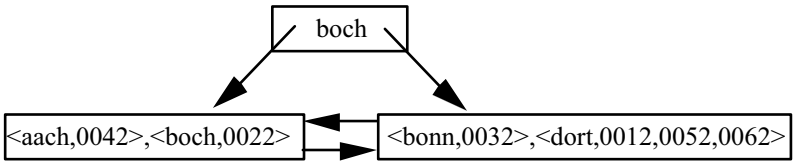
d



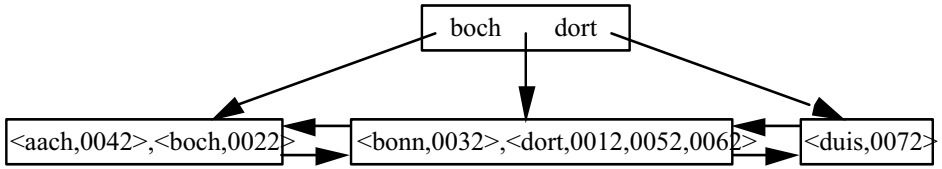
e.



f.



g.



Links

- für zwei (oder auch mehr) Relationen jeweils
  einen gemeinsamen Index bezüglich
  der gleichen Domäne (Attribut) bereitstellen

- B*-Bäume mit Einträgen in die Blattknoten folgender Art:
  < (benutzersichtbarer) Attributwert;
  Folge von TIDs für ***erste*** Relation;
  Folge von TIDs für ***zweite*** Relation >

- manchmal auch sinnvoll:
  in die Blätter die Tupel als Ganzes eintragen
  (anstelle von TIDs)

Beispiel: gemeinsamer B*-Baum für `mitglied` und `entfernung`

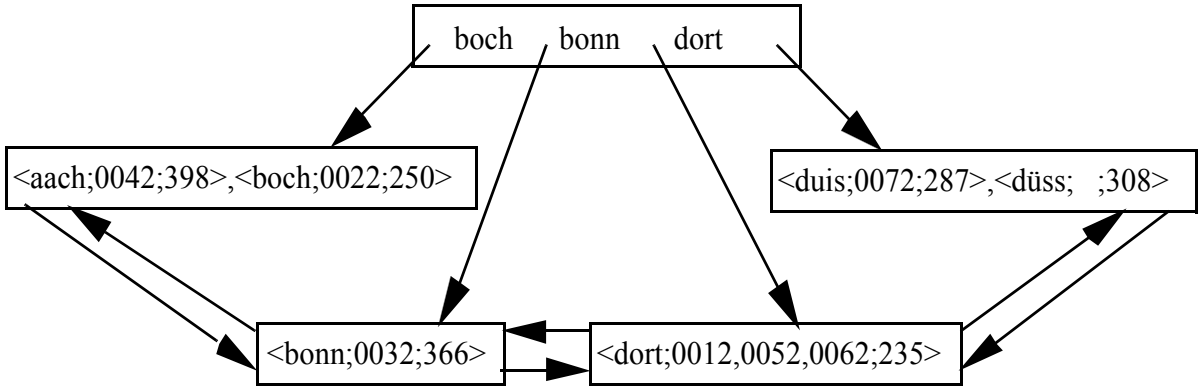
erste Relation: `mitglied`
Schlüsselattribut: `Name`
Eigenschaftsattribut: `Ort` und gegebenenfalls weitere Attribute
Instanz: wie oben

zweite Relation: `entfernung`
Schlüsselattribut: `Ort`
Eigenschaftsattribut: `Weglänge` (von Hildesheim)
Instanz:

h.	dort	235
i.	boch	250
j.	bonn	366
k.	aach	398
l.	duis	287
m.	düss	308

Einträge in **Blattknoten des gemeinsamen B*-Baumes** (Indexes):

< Stadtname;
Folge von TIDs für `mitglied`;
km-Angabe für `entfernung` >



dieser Baum verwirklicht:

- eine ***sequentielle Liste*** für die Tupel der Relation `entfernung`, sortiert nach dem Stadtnamen
- eine ***sequentielle Liste*** für die TIDs der Relation `mitglied`, sortiert nach dem Stadtnamen
- einen ***Index*** für die Relation `entfernung` bezüglich des Stadtnamens
- einen ***Index*** für die Relation `mitglied` bezüglich des Stadtnamens
- einen ***Link*** für die beiden Relationen `entfernung` und `mitglied` bezüglich des Stadtnamens

10.3 Verbund-Algorithmen

Verbund: Definition

$$r \bowtie s := \{ \mu \mid \mu : \mathrm{dom}\, r \cup \mathrm{dom}\, s \rightarrow \mathbf{C},\ \mu\lceil \mathrm{dom}\, r \in r \quad \textbf{und} \quad \mu\lceil \mathrm{dom}\, s \in s \}$$

$$= \{ \mu \mid \text{es gibt } \alpha \in r, \text{ es gibt } \beta \in s : \alpha(A) = \beta(A) \text{ für alle } A \in \mathrm{dom}\, r \cap \mathrm{dom}\, s; \mu = \alpha \cup \beta \}$$

Verbund: einfacher Algorithmus NestedLoop

$r \bowtie s \;=$

$\{ \; \mu \;\; | \;$ — Ausgabe

es gibt $\alpha \in r$, — äußerer loop: durchsuche r

es gibt $\beta \in s$: — innerer loop: durchsuche s

$\alpha(A) = \beta(A)$ für alle $A \in \text{dom } r \cap \text{dom } s$; — α und β **passend** ?

$\mu = \alpha \cup \beta$ — falls zutreffend: Ausgabe-Tupel konstruieren, d.h. passende Tupel **zusammenfügen**

$\}$

Laufzeit: $O(\| r \| \cdot \| s \|)$

NestedLoop als Funktionsprozedur

```
PROCEDURE NestedLoopJoin( r , s : Relation ):Relation;
VAR   ergebnis :  Relation;
      α, β     :  Tupel;
BEGIN
   ergebnis := Ø;
   FOR ALL  α ∈ r   DO
      FOR ALL  β ∈ s   DO
         IF  Passend(α,β)
         THEN ergebnis := ergebnis ∪ {  α ∪ β  }
         END
      END
   END;
   RETURN ergebnis
END NestedLoopJoin;
```

gdw $\alpha(A) = \beta(A)$ für alle $A \in \text{dom}\, r \cap \text{dom}\, s$

fügt passende Tupel zusammen

Hauptaufgabe jeder Verwirklichung des natürlichen Verbunds

- die *passenden Paare* (α , β) ***finden*** mit

  $\alpha \in r$ und

  $\beta \in s$ und

  $\alpha(A) = \beta(A)$ für alle $A \in \text{dom } r \cap \text{dom } s$

- dann α und β tatsächlich ***zusammenfügen*** zu

  einem Tupel $\alpha \cup \beta$

- und dieses Tupel für die Ergebnisrelation t ***ausgeben***

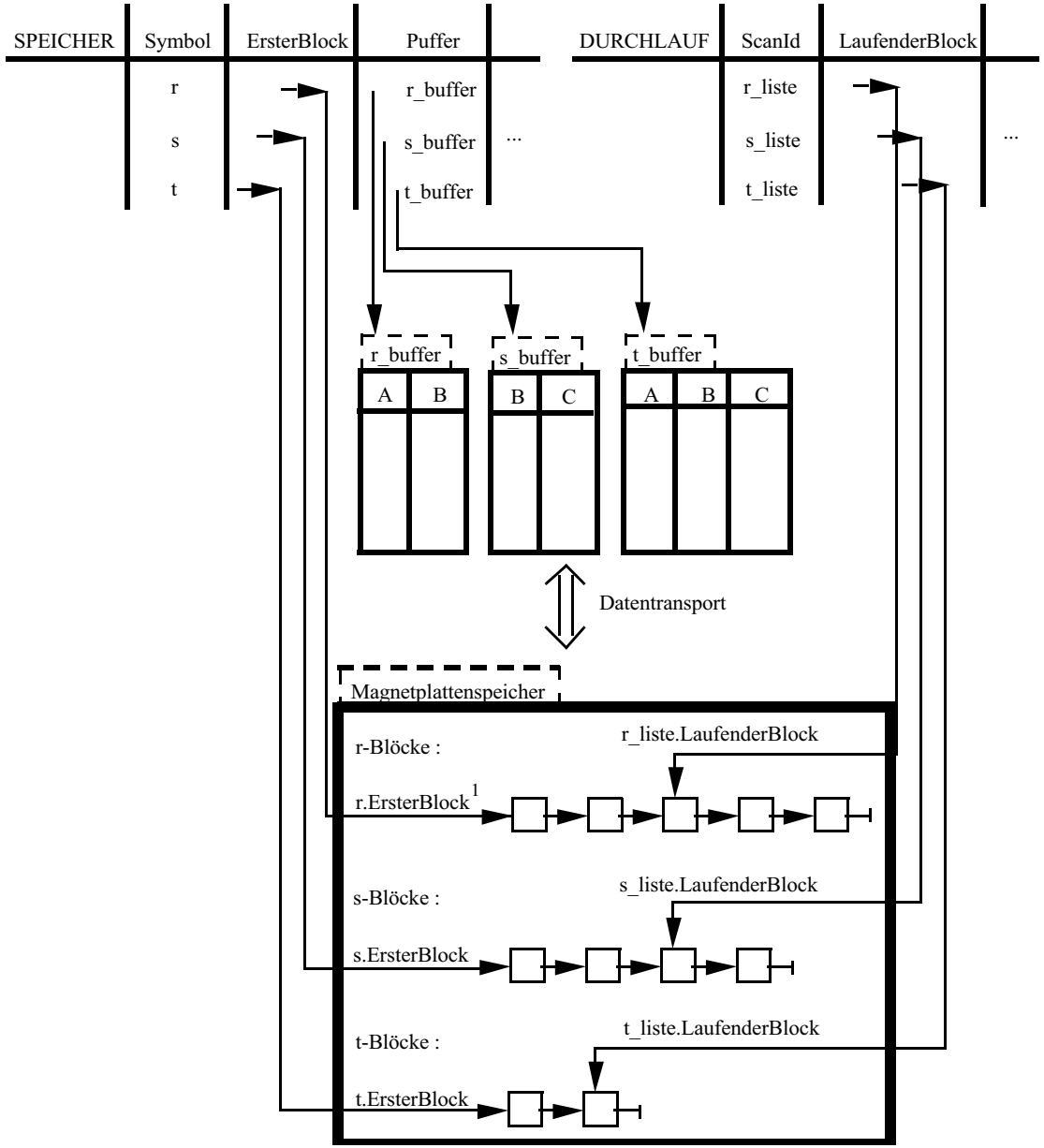
grundlegende Verwirklichungen des Verbunds

- NestedLoop
- Sortiertes Mischen
- Link-Verbund
- Hash-Filter-Verbund

NestedLoop mit Blockliste: Daten- und Zugriffsstrukturen

- Argumentrelationen *r* und *s* sind
  blockweise auf einem *Magnetplattenspeicher* abgelegt
- für jede Relation ist eine *sequentielle Liste der Blöcke* verfügbar
- Verweis auf den ersten Block ist in einem
  Datenwörterbuch (data dictionary) eingetragen
- jeder Block enthält einen Verweis auf seinen Nachfolger
- für jede Relation ist ein *Pufferbereich* im Hauptspeicher eingerichtet
- die Ergebnisrelation *t* soll wieder
  blockweise auf dem *Magnetplattenspeicher* erzeugt werden,
  wobei gleichzeitig eine *sequentielle Liste* ihrer Blöcke aufgebaut werden soll
- um die sequentiellen Listen der Blöcke zu benutzen bzw. aufzubauen,
  sind *Relationendurchläufe* (relation scans) verfügbar
- in einer *Durchlauftabelle* (scan table)
  wird für jeden Durchlauf ein *Verweis* auf den derzeit betrachteten Block abgelegt

Veranschaulichung der (vereinfachten) Strukturen



Operationen auf den Strukturen

- OpenScan
- CloseScan
- GetNextBlock
- EndOfScan
- CreateRelation
- AppendScan

OpenScan

```
OpenScan( r            : Relationensymbol;
          VAR scanid   : Scanidentifikator );
```

erzeugt einen neuen Durchlauf `scanid` für die Relation r:

in die Relation `DURCHLAUF` wird ein neues Tupel μ eingetragen mit

μ(ScanId) = (neu erzeugter Wert von) `scanid`,

μ(LaufenderBlock) = r . ErsterBlock

CloseScan

```
CloseScan( scanid : ScanIdentifikator );
```

beendet den Durchlauf `scanid`:

in der Relation `DURCHLAUF`

wird das Tupel μ mit μ(ScanId) = (Wert von) `scanid`

wieder entfernt

GetNextBlock

```
GetNextBlock( scanid      :   Scanidentifikator;
              VAR buffer  :   Relationenblock );
```

transportiert den Block, auf den `scanid.LaufenderBlock` verweist,
vom Magnetplattenspeicher in den Pufferbereich `buffer`
und setzt dann `scanid.LaufenderBlock`
auf den im transportierten Block enthaltenen Verweis zum Nachfolgerblock

EndOfScan

```
EndOfScan( scanid :  Scanidentifikator ) : Boolean;
```

prüft, ob

`scanid.LaufenderBlock` ein leerer Verweis ist

CreateRelation

```
CreateRelation( t              : Relationensymbol;
                VAR scanid     : Scanidentifikator );
```

erzeugt eine (leere) Relation *t*,
richtet einen zugehörigen Pufferbereich ein,
richtet einen leeren ersten Block auf dem Magnetplattenspeicher ein
und erzeugt einen Durchlauf `scanid` für die Relation :
in die Relation `SPEICHER` wird ein neues Tupel ν eingetragen mit

ν(Symbol)	=	(Wert von) *t*,
ν(ErsterBlock)	=	Verweis auf den eingerichteten ersten Block,
ν(Puffer)	=	Verweis auf den eingerichteten Puffer

und in die Relation `DURCHLAUF` wird ein neues Tupel μ eingetragen mit

μ(ScanId)	=	(neu erzeugter Wert von) `scanid`,
μ(LaufenderBlock)	=	ν(ErsterBlock)

AppendScan

```
AppendScan( scanid  :   Scanidentifikator;
            buffer  :   Relationenblock);
```

transportiert den im Pufferbereich `buffer` stehenden Block
in den Magnetplattenspeicher,

verkettet diesen Block mit dem
durch `scanid.LaufenderBlock` bezeichneten Block

und setzt dann `scanid.LaufenderBlock`
auf den Verweis zum transportierten Block

NestedLoop mit Blockliste: Verfahren

```
PROCEDURE  NestedLoop_BlockScan_Join (
     r, s                                  :    Relationensymbol;
     t                                     :    Relationensymbol  );
VAR  r_liste,   s_liste, t_liste           :    Scanidentifikator;
     r_buffer,  s_buffer,t_buffer          :    Relationenblock;
PROCEDURE  InternalJoin( r_buffer, s_buffer :   Relationenblock;
                         VAR  t_buffer      :   Relationenblock);
  VAR  α, β : RelTupel;
  BEGIN
    FOR ALL α ∈ r_buffer DO
      FOR ALL β ∈ s_buffer DO
          IF Passend ( α , β )
          THEN t_buffer := t_buffer ∪ { α ∪ β };
               IF full (t_buffer)
               THEN AppendScan (t_liste, t_buffer);
                    t_buffer := ∅
               END
          END
      END
    END
  END  InternalJoin;
```

```
BEGIN (*NestedLoop_BlockScan_Join*)
                                                          initialisieren
  CreateRelation (t, t_liste);
  t_buffer := ∅;
  OpenScan (r, r_liste);
                                                          bearbeiten
  WHILE NOT EndOfScan (r_liste) DO                        äußerer Durchlauf
    GetNextBlock (r_liste, r_buffer);
    OpenScan (s, s_liste);
    WHILE NOT EndOfScan (s_liste) DO                      innerer Durchlauf
      GetNextBlock ( s_liste, s_buffer);
      InternalJoin ( r_buffer, s_buffer, t_buffer)
    END;
    CloseScan (s_liste)
  END;
                                                          abschließen
  CloseScan (r_liste);
  IF  t_buffer ≠ ∅ THEN  AppendScan (t_liste, t_buffer) END;
  CloseScan (t_liste)

END  NestedLoop_BlockScan_Join;
```

NestedLoop mit Blockliste: Aufwand

- **Annahme:** ein Block kann höchstens $tpb(u)$ viele Tupel (**t**uple **p**er **b**lock) der Relation u aufnehmen

- **lesende Blockzugriffe:** ungefähr $\frac{||r||}{tpb(r)} \cdot \left(1 + \frac{||s||}{tpb(s)}\right)$

- **schreibende Blockzugriffe:** ungefähr $\frac{||join(r,s)||}{tpb(join(r,s))}$

- lesende und schreibende Blockzugriffe können ***überlappend*** sein

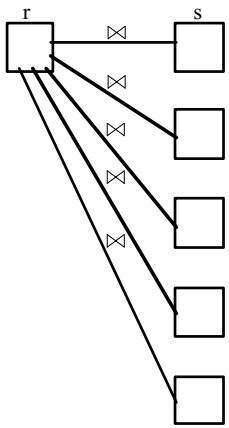
Aufwand bei vergrößerten Pufferbereichen

- gegebenenfalls weniger lesende Zugriffe
- passt zum Beispiel die kleinere Relation, etwa *r*, vollständig in den Puffer,

  so kann diese Relation dort verbleiben

  und wir benötigen nur

$$1 + \frac{||s||}{tpb(s)}$$ ***lesende Blockzugriffe***:



Sortiertes Mischen: Daten- und Zugriffsstrukturen

- wie bei NestedLoop

- zusätzlich:
  sequentielle Liste der Blöcke und
  die ***Anordnung der Tupel*** in den Blöcken derart, dass
  Tupel (aufsteigend) ***sortiert*** sind
  entsprechend Werten auf den Verbundattributen

Sortiertes Mischen: algorithmische Idee

- $T := \text{dom}\, r \cap \text{dom}\, s$

- Tupel entsprechend der Sortierung fortlaufend numeriert,
  etwa $r = \{ \alpha_1, \ldots, \alpha_{\| r \|} \}$ und $s = \{ \beta_1, \ldots, \beta_{\| s \|} \}$

- $r \bowtie s = \{ \alpha_1, \ldots, \alpha_{\| r \|} \} \bowtie s$

  $= \bigcup_{\alpha_i \in r} (\{ \alpha_i \} \bowtie s)$

  $= \bigcup_{\alpha_i \in r} (\{ \alpha_i \} \bowtie \sigma_{T = \alpha_i \lceil T}(s))$

- $\sigma_{T = \alpha_i \lceil T}(s)$: ***zusammenhängender Abschnitt*** in Liste von s

  falls nichtleer: von Form $\{ \beta_{anf}, \beta_{anf+1}, \ldots, \beta_{end} \}$

- alle zueinander passenden Paare $< \alpha_i , \sigma_{T = \alpha_i \lceil T}(s) >$ bestimmen

  in einem *einzigen* Durchlauf durch r und einen *einzigen* Durchlauf durch s

Sortiertes Mischen: Verfahren

durchlaufe Relation r entsprechend Sortierung:

für jedes $\alpha_i \in r$:

prüfe, ob $\sigma_{T = \alpha_i \lceil T}(s)$ nichtleer und

bestimme gegebenenfalls `anf` und `end` des entsprechenden Abschnittes gemäß folgender Fälle bezüglich des vorangehenden Tupels $\alpha_{i-1} \in r$:

Fall 1: $\alpha_{i-1} \lceil T < \alpha_i \lceil T$ **:**

suche in s entsprechend Sortierung nach passenden Tupeln:
beginne dabei beim zuletzt betrachteten Tupel $\beta_j \in s$;
brich Suche ab, sobald $\alpha_i \lceil T < \beta_j \lceil T$;
falls passendes Tupel gefunden, merke `anf` und `end` des Abschnittes und verbinde α_i mit jedem Tupel aus $\{ \beta_{anf}, \ldots, \beta_{end} \}$ für Ausgaberelation

Fall 2: $\alpha_{i-1} \lceil T = \alpha_i \lceil T$ **:**

[der zu α_i passende Abschnitt ist gleich dem zu α_{i-1} passenden Abschnitt]
falls dieser Abschnitt nichtleer ist, [dann `anf` und `end` bekannt]
verbinde α_i mit jedem Tupel aus $\{ \beta_{anf}, \ldots, \beta_{end} \}$ für Ausgaberelation

Sortiertes Mischen: Beispiel

r	A	B	C
α_1	b	b	
α_2	b	c	
α_3	d	c	
α_4	c	c	
α_5	b	f	
α_6	c	f	
α_7	d	g	
α_8	d	h	

s	A	B	C
β_1		a	b
β_2		b	b
β_3		c	d
β_4		c	e
β_5		d	b
β_6		d	f
β_7		h	a

r ⋈ s	A	B	C	Nummer i des betrachteten Tupels $\alpha_i \in r$	$\alpha_i \lceil T$	anf	end	Nummer j des zuletzt betrachteten Tupels $\beta_j \in s$	$\beta_j \lceil T$
	b	b	b	1	b	2	2	3	c
	b	c	d	2	c	3	4	5	d
	b	c	e						
	d	c	d	3	c	3	4	5	d
	d	c	e						
	c	c	d	4	c	3	4	5	d
	c	c	e						
				5	f	-	-	7	h
				6	f	-	-	7	h
				7	g	-	-	7	h
	d	h	a	8	h	7	7	∞	

Sortiertes Mischen: Aufwand bezüglich Vergleiche

- jedes $\alpha \in r$ wird bezüglich seiner T-Werte mit seinem Vorgänger verglichen:

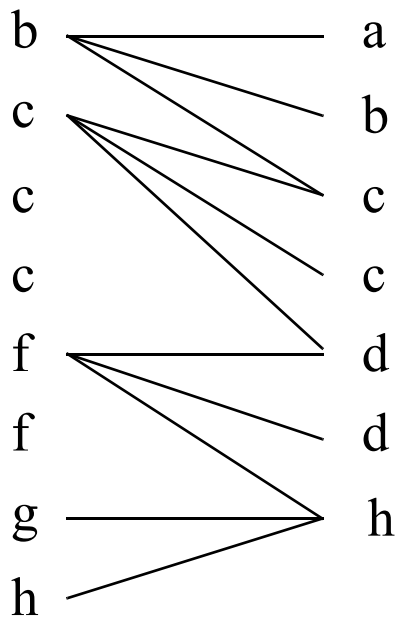
  Anzahl Vergleiche etwa $\| r \|$

- in jedem "Passend-Vergleich"
  zwischen einem Tupel $\alpha \in r$ und einem Tupel $\beta \in s$
  wird mindestens eines dieser Tupel *erstmalig* benutzt:

  Anzahl Vergleiche höchstens $\| r \| + \| s \|$

- also gesamtes Verfahren:

  Anzahl Vergleiche höchstens etwa $2 \cdot \| r \| + \| s \|$



Sortiertes Mischen: Aufwand bezüglich Blockzugriffe:

- die Blöcke der Relation r werden ***einmal lesend*** durchlaufen

- Fallunterscheidung für die Blöcke der Relation s:

  Fall 1: $\sigma_{T=\alpha_i \lceil T}(s)$-Abschnitte passen jeweils ganz in den Puffer:

  die Blöcke der Relation s nur ***einmal lesend*** durchlaufen

  Fall 2: $\sigma_{T=\alpha_i \lceil T}(s)$-Abschnitte gelegentlich größer als Puffer:

  Blöcke der Relation s müssen gelegentlich ***wiederholt gelesen*** werden, mit (geschätztem) ***Wiederholungsfaktor*** $w \geq 1$

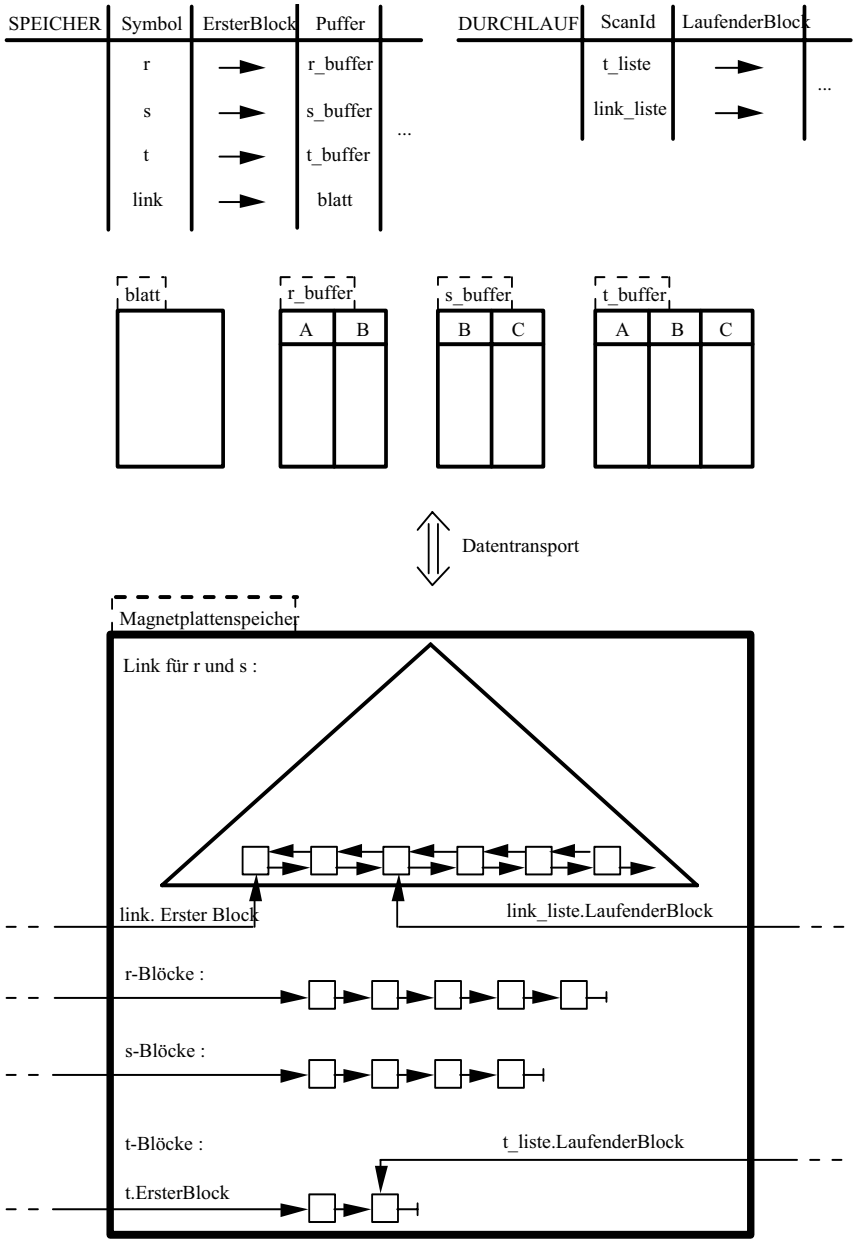
- also ingesamt Anzahl ***lesende Blockzugriffe*** $\frac{||r||}{tpb(r)} + w \cdot \frac{||s||}{tpb(s)}$

- wird Sortierung erst zum Zeitpunkt der Anfrage aufgebaut, so Aufwand für das Sortieren hinzurechnen

Link-Verbund: Daten- und Zugriffsstrukturen

- die Argumentrelationen *r* und *s* sind
  blockweise auf einem *Magnetplattenspeicher* abgelegt
- für die Relationen *r* und *s* ist ein *Link* bezüglich der Verbundattribute *T*
  als B*-Baum vorhanden
- damit sind ebenfalls je eine *sequentielle Liste für die TIDs* der Relationen verfügbar
- für jede Relation und für den Link ist ein *Pufferbereich* im Hauptspeicher eingerichtet,
  der jeweils genau einen Block aufnehmen kann
- die Ergebnisrelation *t* wird wieder *blockweise* auf dem *Magnetplattenspeicher* erzeugt,
  wobei gleichzeitig eine *sequentielle Liste ihrer Blöcke* aufgebaut wird
- das *Datenwörterbuch* enthält auch Angaben über die vorhandenen Links
- um die sequentiellen Listen zu benutzen bzw. aufzubauen,
  wird wieder eine *Durchlauftabelle* angelegt
- es gibt ein geeignetes Verfahren zur *Umwandlung* von TIDs in Speicheradressen

Veranschaulichung der (vereinfachten) Strukturen



Operationen auf den Strukturen

wie bei NestedLoop:

```
OpenScan        (   linkid       :   Relationen_oder_Linksymbol;
                    VAR scanid   :   Scanidentifikator );

CloseScan       (   scanid       :   Scanidentifikator );

GetNextBlock    (   scanid       :   Scanidentifikator;
                    VAR buffer   :   Relationen_oder_Linkblock );

EndOfScan       (   scanid       :   Scanidentifikator ) : Boolean;

CreateRelation  (   t            :   Relationensymbol;
                    VAR scanid   :   Scanidentifikator );

AppendScan      (   scanid       :   Scanidentifikator;
                    buffer       :   Relationenblock );
```

zusätzlich:

```
DetermineLink   (   r, s         :   Relationensymbol;
                    X, Y         :   Attributmenge;
                    VAR linkid   :   Linksymbol );

LocateFetch     (   tid          :   Tupelidentifikator;
                    VAR buffer   :   Relationenblock;
                    VAR tupel    :   RelTupel );
```

DetermineLink

```
DetermineLink (  r, s          :  Relationensymbol;
                 X, Y          :  Attributmenge;
                 VAR linkid    :  Linksymbol );
```

bestimmt aus Angaben im Datenwörterbuch

das Linksymbol für den Link zu r und s

bezüglich der Attribute X von r bzw. Y von s

und liefert diesen als Wert des Ausgabeparameters `linkid` zurück

LocateFetch

```
LocateFetch ( tid          :   Tupelidentifikator;
              VAR buffer   :   Relationenblock;
              VAR tupel    :   RelTupel );
```

sucht `tid` zunächst im Pufferbereich `buffer`;

falls `tid` dort nicht vorhanden ist,
wird (mittels der Umwandlung von TIDs in Speicheradressen)
der `tid` enthaltende Block
vom Magnetplattenspeicher in den Pufferbereich `buffer` transportiert;
anschließend wird das durch `tid` identifizierte Tupel
als Wert des Ausgabeparameters `tupel` zurückgeliefert

Link-Verbund: algorithmische Idee

- $T := \text{dom}\, r \cap \text{dom}\, s$

- $$r \bowtie s = \bigcup_{\alpha \in r} \bigcup_{\beta \in s} (\{ \alpha \} \bowtie \{ \beta \})$$

  $$= \bigcup_{\mu \in \pi_T(r) \cap \pi_T(s)} (\bigcup_{\alpha \in r, \alpha|T = \mu} \bigcup_{\beta \in s, \beta|T = \mu} \{ \alpha \cup \beta \})$$

  $$= \bigcup_{\mu \in \pi_T(r) \cap \pi_T(s)} (\sigma_{T=\mu}(r) \bowtie \sigma_{T=\mu}(s))$$

- Verfahren durchläuft die Blattknoten des Links:
  Einträge: $<$ T-Wert : Folge von TIDs für r ; Folge von TIDs für s $>$,
  für T-Wert μ alle TIDs aus $\sigma_{T=\mu}(r)$ bzw. aus $\sigma_{T=\mu}(s)$

- also: für jeden Eintrag
  die Tupel der Teilrelationen $\sigma_{T=\mu}(r)$ und $\sigma_{T=\mu}(s)$ auffinden,
  dann $\sigma_{T=\mu}(r) \bowtie \sigma_{T=\mu}(s)$ bilden

```
PROCEDURE  Link_Join (r, s : Relationensymbol;  t : Relationensymbol);
VAR link :                          Linksymbol;
    link_liste, t_liste :           Scanidentifikator;
    r_buffer, s_buffer, t_buffer :  Relationenblock;
    blatt :                         Linkblatt;

BEGIN
                                                          initialisieren
  CreateRelation (t, t_liste);
  t_buffer := ∅;
  DetermineLink (r, s, T, T, link);
                                                          bearbeiten
  OpenScan (link, link_liste);
  WHILE NOT EndOfScan (link_liste) DO
      GetNextBlock (link_liste, blatt);
      Blattjoin (blatt)
  END;
  CloseScan (link_liste);
                                                          abschließen
  IF  t_buffer ≠ ∅  THEN   AppendScan (t_liste, t_buffer);
  CloseScan (t_liste)

END  Link_Join;
```

```
PROCEDURE  Blattjoin (blatt);

(*  r_buffer, s_buffer, t_buffer und t_liste
    werden als globale Variablen verwendet;

    blatt sei wie folgt aufgebaut

      anzahl :   CARDINAL;
      liste  :   ARRAY [1..anzahl] OF

          RECORD
          schlüssel :   Wert;
          anz_tids_1 :  CARDINAL;
          anz_tids_2 :  CARDINAL;
          tids_1 :      ARRAY[1..anz_tids_1] OF Tupelidentifikator;
          tids_2 :      ARRAY [1..anz_tids_2] OF Tupelidentifikator
          END;
*)

VAR   eintrag, tid_1, tid_2 :  CARDINAL;
      tupel_1, tupel_2 :       RelTupel;
```

```
BEGIN (* Blattjoin *)
  FOR eintrag := 1  TO  anzahl  DO
    FOR tid_1 := 1   TO  liste [eintrag].anz_tids_1  DO
      FOR tid_2 := 1  TO  liste [eintrag].anz_tids_2  DO

        LocateFetch (liste [eintrag].tids_1 [tid_1], r_buffer, tupel_1);

        LocateFetch (liste [eintrag].tids_2 [tid_2], s_buffer, tupel_2);

        t_buffer := t_buffer ∪ {tupel_1 ∪ tupel_2};

        IF full (t_buffer)  THEN   AppendScan (t_liste, t_buffer);
                                   t_buffer := ∅
        END

      END
    END
  END
END  Blattjoin;
```

Link-Verbund: Aufwand

- bezüglich “Passend-Vergleiche”: entfällt

- bezüglich lesender Blockzugriffe: schwierig analytisch zu bestimmen

- `LocateFetch` kann sehr schnell oder sehr langsam sein:

  schnell: Tupel im Pufferbereich, dortige Adresse bekannt

  langsam: Tupel im Magnetplattenspeicher suchen und in den Pufferbereich holen

Link-Verbund: günstiger Fall

- Relationen r und s nach den Verbundattributen (aufsteigend) sortiert gespeichert

- Pufferbereiche sind groß genug, dass keine Blöcke wiederholt gelesen werden müssen

- dann Anzahl **lesende Blockzugriffe** $BL + \frac{||r||}{tpb(r)} + \frac{||s||}{tpb(s)}$,

  wobei

  - BL die Anzahl der Blattknoten des Links sei,
  - jeder Block ein zum Ergebnis beitragendes Tupel enthalte

Link-Verbund: ungünstiger Fall

- die Tupel der Relationen r und s
  sind unglücklich auf die Blöcke verteilt:

  jede Ausführung von `LocateFetch` verursacht

  eine Suche im Magnetplattenspeicher

  mit etwa jeweils k vielen Blockzugriffen

- dann Anzahl **lesende Blockzugriffe** $BL + 2 \cdot k \cdot ||join(r, s)||$

- wird der Link erst zum Zeitpunkt der Anfrage aufgebaut,
  so Aufwand für seine Erstellung hinzurechnen

Hash-Filter-Verbund: Daten- und Zugriffsstrukturen

- alle Strukturen wie bei NestedLoop mit Block-Liste

- ***Hashfunktion*** für Werte der Attribute aus T:

  $hash$: Domäne (T) $\rightarrow$ { 1, ... , k }

  mit $k \geq \max$ ($\| r \|$, $\| s \|$) lässt *wenige Kollisionen* erwarten

- zwei ***Bitlisten*** `bit_r` und `bit_s` der Länge k

  für die Relationen r und s ,

  jeweils mit `false` initialisiert

Hash-Filter-Verbund: algorithmische Idee

- $r \bowtie s = (r \bowtie \pi_{\text{dom } r \cap \text{dom } s}(s)) \bowtie (s \bowtie \pi_{\text{dom } r \cap \text{dom } s}(r))$

  $= (r \ltimes s) \bowtie (s \ltimes r)$

  mit $\ltimes$ abgeleitete relationale Operation des ***Teilverbunds*** (semi-join)

- Teilverbünde $r \ltimes s$ bzw. $s \ltimes r$ sind *erwartungsgemäß* bedeutend *kleiner* als ursprüngliche Relationen r bzw. s

- Teilverbünde möglichst gut nach oben *abschätzen* durch ***Filterrelationen***

  `r_filter` bzw. `s_filter` mit

  $r \supset r\_filter \supset r \ltimes s$ bzw. $s \supset s\_filter \supset s \ltimes r$

  dann: $r \bowtie s = r\_filter \bowtie s\_filter$

- Filterrelationen bestimmen mit Hilfe von ***Bitlisten*** und einer ***Hashfunktion***, dabei vorzugsweise Filterrelationen ***sortiert*** aufbauen und ***Indexe*** bzg. der Verbundattribute bzw. einen ***gemeinsamen Link*** erzeugen

Hash-Filter-Verbund: Verfahren

1. `bit_r` und `bit_s` mit `false` initialisieren;

2. `FOR All` $\alpha \in r$ `DO`
   `bit_r( hash(` $\alpha\lceil$`T) ) := true;` — berechne *hash*($\alpha\lceil T$); setze Bit

3. `FOR All` $\beta \in s$ `DO`
   `stelle := hash(` $\beta\lceil$`T );` — berechne *hash*($\beta\lceil T$)
   `IF bit_r( stelle )` — Bit in `bit_r` gesetzt?
   `THEN bit_s( stelle ) := true;` — setze dieses Bit auch in `bit_s`
   `insert(` β`, s_filter )` — füge β zu `s_filter` hinzu
   `END;` — (mit Index und sortiert)

4. `FOR All` $\alpha \in r$ `DO`
   `IF bit_s( hash(` $\alpha\lceil$`T ))` — Bit in `bit_s` gesetzt?
   `THEN insert(` α `, r_filter )` — füge α zu `r_filter` hinzu
   `END;` — (mit Index und sortiert)

5. `r_filter` $\bowtie$ `s_filter` berechnen. — etwa mit Link-Verbund

Hash-Filter-Verbund: Beispiel

a)

r	A	B	C
	1	1	
	2	1	
	5	5	
	7	1	
	16	3	
	1	2	
	2	2	

s	A	B	C
	5		1
	1		1
	9		9
	19		1
	3		3
	19		7

b)

	bit_r	bit_s
0		
1	t	t
2	t	
3		
4		
5	t	t
6		
7	t	
8		
9		
10		

c)

s_filter	A	B	C
	5		1
	1		1

r_filter	A	B	C
	1	1	
	5	5	
	16	3	
	1	2	

d)

s_filter ⋈ r_filter	A	B	C
	5	5	1
	1	1	1
	1	2	1

(sehr einfache) Hashfunktion:

Divisions-Rest-Methode
mit Parameter $k = 11$,

$hash(x) := x \bmod 11$

Hash-Filter-Verbund: Aufwand

- Relation r ***zweimal*** und Relation s ***einmal*** durchlaufen, so dass

  lesende Blockzugriffe: $$2 \cdot \frac{||r||}{tpb(r)} + \frac{||s||}{tpb(s)}$$

  schreibende Blockzugriffe: $$\frac{||rfilter||}{tpb(rfilter)} + \frac{||sfilter||}{tpb(sfilter)}$$

  Aufrufe Hashfunktion etc.: $$2 \cdot ||r|| + ||s||$$

- zusätzlich Aufwand für
  - **Sortierungen** von `r_filter` und `s_filter` und
  - **Links**

Hash-Filter-Verbund: Bewertung des Aufwands

- falls ***Filterwirkung gering***:
  - Aufwand für Aufrufe Hashfunktion etc. im Wesentlichen ***vergeblich***
  - aber aufgebaute Zugriffsstrukturen zu `r_filter` und `s_filter` ermöglichen noch effizienten Verbund-Algorithmus, etwa Link-Verbund

- falls ***Filterwirkung stark***:
  - (in Größe von *r* und *s*) linearer Aufwand für Filterrelationen ***lohnt***, weil Filterrelationen ***erheblich kleiner*** als *r* bzw. *s*

Zusammenfassung der Verbund-Algorithmen

- **NestedLoop**

  durchlaufe “äußere Relation”:
  für jedes Tupel der äußeren Relation durchlaufe “innere Relation”:
  füge jeweils passende Tupel zusammen

- **Sortiertes Mischen**

  (sortiere beide Relationen auf Verbundattributen;)
  durchlaufe beide Relationen simultan entsprechend der Sortierung:
  für jedes Tupel der “äußeren Relation” bestimme bzw. durchlaufe
  den Abschnitt passender Tupel der “inneren Relation”
  und füge jeweils passende Tupel zusammen

- **Link-Verbund**

  (erstelle Link, gemeinsamen Index, beider Relationen bezüglich Verbundattributen;)
  durchlaufe Blattknoten des Links und füge jeweils passende Tupel zusammen

- **Hash-Filter-Verbund**

  erstelle Filterrelationen mit Hilfe von Hashfunktion und Bitlisten
  als Abschätzungen der beiden Teilverbunde;
  verbinde die Filterrelationen mit Hilfe von Link-Verbund oder Sortiertem Mischen

11 Anfrage-Optimierung

Optimierung von Anfragen

- **Informationssystem:**

  stellt mächtige ***Anfragesprachen*** zur Verfügung

- **Benutzer:**

  drückt damit ***Anfrage*** aus,
  die im Wesentlichen beschreibt,
  was (welche Aussagen, Tupel, Objekte, Objektwerte)
  der Benutzer als Ergebnisse erwartet

- **Informationssystem:**

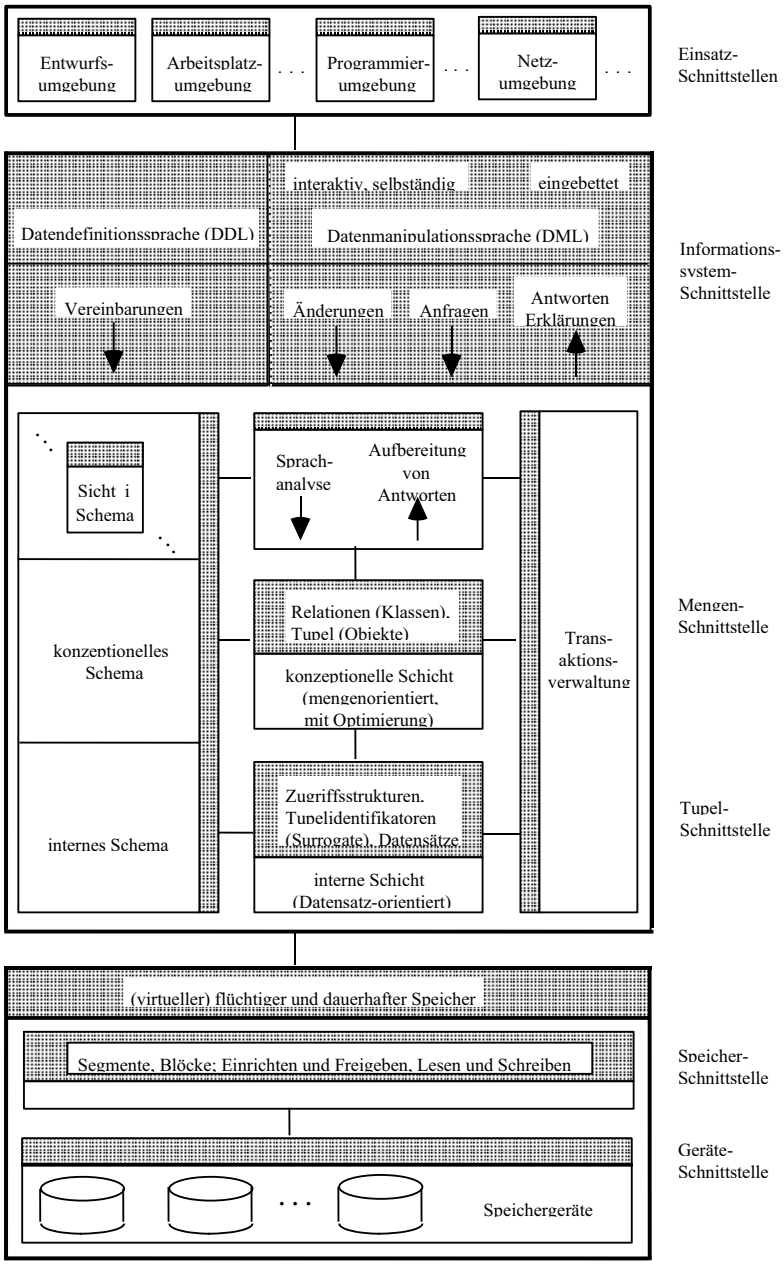
  ermittelt aus der Anfrage,
  wie die gewünschten Ergebnisse bzw. deren Bestandteile
  möglichst schnell (und platzsparend)
  erzeugt bzw. aufgefunden werden können

Optimierungsaufgabe

bestimme aus der *Beschreibung* des **Was** (eine Anfrage)

einen guten *Ausführungsplan* für das **Wie** (einen Algorithmus)!

Optimierung: besonders wichtig und besonders schwierig



- **Benutzer kann**

anwendungsnahe, mächtige Sprachmittel
auf sehr große Mengen von Daten anwenden

- **Informationssystem muss**

die volle Kluft zwischen Benutzersprache und Speicherzugriffen
über alle Zwischenschichten hinweg überbrücken

Beispiel: “naive Auswertungen”

- **Annahmen**

  Schema: $\text{dom}\, R = \{ X, Y \}$, $\text{dom}\, S = \{ Y, Z \}$

  Benutzer-Anfrage in relationaler Algebra: $\sigma_{Z=c}(R \bowtie S)$

  Anzahl Tupel in R bzw. S: 10^6

- **Auswertung durch**

  natürlichen Verbund vermöge

  - NestedLoop führt zu Laufzeit: $c_{NL} \cdot (10^6)^2$
  - sortiertes Mischen kann führen zu Laufzeit: $c_{SM} \cdot 10^6$

  gefolgt von einem **Selektion**sverfahren;

  ggf. vorherige **Sortierung** mit Laufzeit: $2 \cdot c_{Sort} \cdot 20 \cdot 10^6$
  (mit gutem O($n \cdot \log n$)-Verfahren,
  $20 \approx$ Duallogarithmus von 10^6)

Beispiel: verbesserte Auswertung

- **algebraische Umformung**

  da $Z \in \text{dom}\, S$: $\quad \sigma_{Z=c}(R \bowtie S) = R \bowtie \sigma_{Z=c}(S)$

  Selektionsergebnis $\sigma_{Z=c}(S)$ im Allgemeinen viel kleiner als S

- **semantische Bedingung**

  falls Attribut Z für Relation S Schlüsselattribut,
  dann enthält $\sigma_{Z=c}(S)$ ***höchstens ein Tupel***

- **Zugriffsstruktur**

  falls zusätzlich B*-Baum für Attribut Z von S,
  dann $\sigma_{Z=c}(S)$ durch eine ***einzige Suche*** im B*-Baum bestimmen

- **"reduzierende Umformung"**

  Suchergebnis: Tupel μ aus S
  reduzierter Anfrageausdruck:

$$\begin{aligned} & R \bowtie \{\mu\} \\ = \; & R \bowtie \{(Y, \mu(Y))\} \bowtie \{(Z, c)\} \\ = \; & \sigma_{Y=\mu(Y)}(R) \bowtie \{(Z, c)\} \end{aligned}$$

  - Selektion auf R nach zwischenzeitlich ermitteltem Wert $\mu(Y)$
  - jedes Ergebnistupel der Selektion mit dem Wert c für Attribut Z verbinden

Beispiel: günstiger Fall

Anfrage: $\sigma_{Z=c}(R \bowtie S)$

Auswertung unter allen günstigen Annahmen: $\sigma_{Y=\mu(Y)}(R) \bowtie \{(Z,c)\}$

- Attribut Z für Relation S Schlüsselattribut
- B*-Baum bezüglich Attribut Z von S
- Relation R nach Attribut Y sortiert

Größenordnung des Aufwands:

5 Zugriffe längs eines Pfades im ***B*-Baum***
(der Tiefe 5) für Relation S

und

20 Zugriffe zur ***logarithmischen Suche***
des Blocks von Relation R mit zu selektierenden Tupeln

Beispiel: Zusammenfassung

fiktive Zeiteinheit: 1 µsec

Bandbreite im Beispiel:

NestedLoop: $10^{12} \cdot 10^{-6}$ sec = 10^{6} sec ≈ 12 Tage

Sortiertes Mischen: $(20 + 1) \cdot 10^{6} \cdot 10^{-6}$ sec = 21 sec

Spezialfall mit B*-Baum und sortierter Speicherung: $25 \cdot 10^{-6}$ sec = 25 µsec

Ziel der Optimierung

Auswertung einer Anfrage:

- zunächst *bestimmte Daten* (Werte, Tupel, Objekte)
  in der gegenwärtigen Instanz des Informationssystems ***auffinden***

- anschließend aus bzw. in Abhängigkeit von diesen aufgefundenen Daten
  die *gewünschten Ergebnisse* ***erzeugen***

Wunsch-Anforderungen (i. Allg. kaum erfüllbar):

O1.[*Suchraum beschränken*]

ausschließlich auf die zu findenden Daten zugreifen

O2.[*Wiederholungen vermeiden*]

auf jedes zu findende Datum ***nur einmal*** zugreifen

Methoden der Optimierung

1. *äquivalente Umformungen der Anfrage*

2. *Erstellung von Ausführungsplänen*

3. *Aufwandsschätzung*

4. *Suche nach einem kostengünstigen Ausführungsplan*

Methode 1: äquivalente Umformungen der Anfrage

auf der Ebene der Anfragesprache
semantisch ***äquivalente Anfragen*** bestimmen,
wobei:

- ***redundante Teile entfernt*** werden

- Variablen (oder entsprechende Konzepte) an ***möglichst kleine Mengen gebunden*** werden

Methode 2: Erstellung von Ausführungsplänen

mit Methode 1 bestimmte Anfragen jeweils ***übersetzen***

in ein oder mehrere ***Ausführungspläne***,

deren Anweisungen auf tieferen Schichten des Informationssystems liegen

Methode 3: Aufwandsschätzung

für mit Methoden 1 und 2 gewonnene Ausführungspläne

den ***voraussichtlichen Aufwand*** abschätzen

Methode 4: Suche nach einem kostengünstigen Ausführungsplan

alle erfolgversprechenden äquivalenten **Anfragen** erzeugen (Methode 1)

und diese in **Ausführungspläne** übersetzen (Methode 2)

und jeweils deren **Aufwand** schätzen (Methode 3)

mit dem Ziel:

möglichst schnell einen Ausführungsplan

als ***kostengünstig***
(oder im besten Fall als ***kostengünstigsten***)

im Vergleich mit den anderen erkennen

Grundlagen der Optimierung

- Kenntnisse über die ***Semantik*** von Anfragen,
  insbesondere Regeln für äquivalenzerhaltende Umformungen
- ***Schema***vereinbarung,
  insbesondere semantische Bedingungen
- Kenntnisse über verfügbare ***Datenstrukturen*** und ***Algorithmen***
- ***Laufzeitinformation*** über gespeicherte ***Objekte***
  (z.B. Anzahl der gespeicherten Tupel einer Relation)
- ***Laufzeitinformation*** über bereits angelegte ***Zugriffsstrukturen***

einige Heuristiken zur Optimierung relationaler Ausdrücke

- Ausdrücke zusammenfassen
- Redundanz entfernen
- Selektionen und Projektionen vorziehen
- Verbund-Reihenfolge auswählen

Zusammenfassen von Ausdrücken

- direkt hintereinander auszuführende ***Selektionen***

  $\sigma_{A1=c1} \circ \sigma_{A2=c2} \circ \ldots \circ \sigma_{An=cn}(R)$

  zusammenfassen zu ***einer*** Selektion

  $\sigma_{A1=c1 \wedge A2=c2 \wedge \ldots \wedge An=cn}(R)$

- direkt hintereinander auszuführende ***Projektionen***

  $\pi_{X1} \circ \ldots \circ \pi_{Xn}(R)$

  zusammenfassen zu ***einer*** Projektion

  $\pi_{X1 \cap \ldots \cap Xn}(R)$

- einen ***Projektion-Verbund***-Ausdruck

  $\pi_X(R_1 \bowtie R_2)$

  durch ***einen*** Algorithmus gemeinsam auswerten, indem
  bei der Erzeugung eines Tupels μ von $R_1 \bowtie R_2$
  nur die Einschränkung $\mu\lceil X$ geliefert wird

- ...

Entfernen von Redundanz

- offensichtlich redundante Teile eines Ausdruckes können entfernt werden

- Beispiel:
  wegen des **Absorbtiv-Gesetzes für den natürlichen Verbund** gilt**:**

  falls bekannt ist, dass

  der Wert von R stets Teilmenge des Wertes von S ist,

  dann kann der Ausdruck $R \bowtie S$

  verkürzt werden zu R

Vorziehen von Selektionen und Projektionen

- Selektionen und Projektion ***möglichst früh*** ausführen
- durch Semantik-erhaltende Vertauschung mit vorangehender Operation: ***Zwischenergebnisse möglichst klein*** halten!
- Vertauschungen entsprechend ***Umformungsregeln***

Umformungsregeln mit Vorbedingungen

- falls $A \in \text{dom}\, R$: $\sigma_{A=c}(R \bowtie S) = \sigma_{A=c}(R) \bowtie S$
- falls $A \in \text{dom}\, R \cap \text{dom}\, S$: $\sigma_{A=c}(R \bowtie S) = \sigma_{A=c}(R) \bowtie \sigma_{A=c}(S)$
- $\sigma_{A=c}(R \cup S) = \sigma_{A=c}(R) \cup \sigma_{A=c}(S)$
- $\sigma_{A=c}(R \setminus S) = \sigma_{A=c}(R) \setminus \sigma_{A=c}(S)$
- falls $A \in X \cap \text{dom}\, R$: $\sigma_{A=c}(\pi_X(R)) = \pi_X(\sigma_{A=c}(R))$
- falls $\text{dom}\, R \cap \text{dom}\, S \subset X$: $\pi_X(R \bowtie S) = \pi_X(R) \bowtie \pi_X(S)$
- $\pi_X(R \cup S) = \pi_X(R) \cup \pi_X(S)$
- ...

Auswahl von Verbund-Reihenfolge

- Zwischenergebnisse möglichst klein halten !

- zuerst: “Durchschnitt-ähnliche” Verbünde ausführen,
  anschließend: “Produkt-ähnliche” !

- “hängende Tupel” vermeiden !

Entfernen von Redundanz: logik-orientiert

Bestimmung der **logischen Implikation** $\models$
ist wesentlicher Kern der **Semantik** (Wiederholung):

RS mit **EDB** $= \{ R_1, \ldots, R_n \}$ Schema

$< R \mid \boldsymbol{Q} >$ LOGODAT-Anfrage mit (Hornklausel-)Anfrageprogramm

$f = (r_1, \ldots, r_n)$ Ausprägung zu **EDB**

$$\text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$$

$$:= \quad \{ \quad R(c_1, \ldots, c_s). \quad \mid \quad f \cup \boldsymbol{Q} \models R(c_1, \ldots, c_s). \quad \text{mit } c_1, \ldots, c_s \in \mathbf{C} \quad \}$$

$$= \quad \{ \quad R(c_1, \ldots, c_s). \quad \mid \quad R(c_1, \ldots, c_s). \in \mathit{fix}(f \cup \boldsymbol{Q}) \quad \}$$

äquivalente Umformungen von Anfragen: Lemma

wenn $\boldsymbol{Q1} \models \boldsymbol{Q2}$,

dann für alle Ausprägungen f zu RS:
$\text{eval}_{RS}(< R \mid \boldsymbol{Q1} >)(f) \supset \text{eval}_{RS}(< R \mid \boldsymbol{Q2} >)(f)$

Beweis:

betrachte: $R(c_1, \ldots, c_s). \in \text{eval}_{RS}(< R \mid \boldsymbol{Q2} >)(f)$

Definition deklarative Semantik: $f \cup \boldsymbol{Q2} \models R(c_1, \ldots, c_s).$ (1)

zu zeigen: $f \cup \boldsymbol{Q1} \models R(c_1, \ldots, c_s).$

betrachte dazu: M Modell von $f \cup \boldsymbol{Q1}$ (2)

nach Voraussetzung: $\boldsymbol{Q1} \models \boldsymbol{Q2}$

also mit (2): M Modell von $f \cup \boldsymbol{Q2}$

gemäß (1): M Modell von $R(c_1, \ldots, c_s).$

Redundanz von Anfrageklauseln oder Prämissen: Definition

Anfrageklausel K ***redundant*** **in** $\boldsymbol{Q}$ **bezüglich** ***RS***

:gdw für alle Instanzen f zu RS :

$$\text{eval}_{RS}(\ <R \mid \boldsymbol{Q}\ >)(\ f\ )$$

$$=\ \text{eval}_{RS}(\ <R \mid \boldsymbol{Q} \setminus \{\ K\ \}>)(\ f\ )$$

Prämisse A_i von $K \equiv A_0$:- $A_1, \ldots, A_i, \ldots, A_m$. ***redundant*** **in** ***K*** **bezüglich** ***RS, Q***

:gdw für alle Instanzen f zu RS :

$$\text{eval}_{RS}(\ <R \mid \boldsymbol{Q}\ >)(\ f\ )$$

$$=\ \text{eval}_{RS}(\ <R \mid (\ \boldsymbol{Q} \setminus \{\ K\ \}\ ) \cup \{\ A_0 \text{ :- } A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m.\ \}>)(\ f\ )$$

Redundanz von Anfrageklauseln oder Prämissen: triviale Forderung

o.B.d.A.
nur Anfragen mit $R = \text{concl}(K) \in \text{concl}(\boldsymbol{Q} \setminus \{ K \})$ betrachten;
in geforderten Gleichheiten ist **jeweils eine Inklusion trivial**,
denn gemäß Lemma über äquivalente Umformungen:

wegen $\boldsymbol{Q} \models \boldsymbol{Q} \setminus \{ K \}$

folgt für alle Instanzen f zu RS:

$$\text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$$
$$\supset \text{eval}_{RS}(< R \mid \boldsymbol{Q} \setminus \{ K \} >)(f)$$

bzw.

wegen $(\boldsymbol{Q} \setminus \{ K \}) \cup \{ A_0 \text{ :- } A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m. \} \models \boldsymbol{Q}$

folgt für alle Instanzen f zu RS:

$$\text{eval}_{RS}(< R \mid \boldsymbol{Q} >)(f)$$
$$\supset \text{eval}_{RS}(< R \mid (\boldsymbol{Q} \setminus \{ K \}) \cup \{ A_0 \text{ :- } A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m. \} >)(f)$$

Aufwandsschätzung: Faktoren

- ***Kenngrößen der gespeicherten Relationen***, z.B.
  - derzeitige Anzahl der Tupel (Eigenschaft der Instanz)
  - (maximale) Länge eines Tupels (Eigenschaft des Schemas)
  - Anzahl der möglichen Werte für ein Attribut (Eigenschaft des Schemas)

- ***Statistische Annahmen***,

  um Erwartungswerte für die Kenngrößen

  der Zwischenergebnis-Relationen zu berechnen

- ***Kostenfunktionen***

  für die verfügbaren Algorithmen

  zur Verwirklichung der relationalen Operationen

Aufwandsschätzung: einfaches Beispiel

Schema: $\text{dom}\, R = \{ X, Y \}$ und $\text{dom}\, S = \{ Y, Z \}$

äquivalente Anfragen: $\sigma_{Z=c}(R \bowtie S)$ und $R \bowtie \sigma_{Z=c}(S)$

***Kenngrößen der gespeicherten Relationen*:**

kr := Anzahl der Tupel in Instanz von R

ks := Anzahl der Tupel in Instanz von S

ky := Anzahl der Konstantenzeichen in Domäne $\mathbf{b}(Y)$

kz := Anzahl der Konstantenzeichen in Domäne $\mathbf{b}(Z)$

***statistische Annahme*:**

alle Zufallsvariablen für interessierende Ereignisse
sind *gleichverteilt* und *statistisch unabhängig*

***Kostenfunktion*:**

Laufzeiten der einfachen Algorithmen:

$$kosten\,(\,\sigma_{A=c}(\,T\,)\,) \;:=\; \text{Anzahl der Tupel in } T$$

$$kosten\,(\,U \bowtie V\,) \;:=\; (\,\text{Anzahl der Tupel in } U\,) \cdot (\,\text{Anzahl der Tupel in } V\,)$$

Kostenfunktion für zusammengesetzte Ausdrücke ***additiv fortsetzen***,
Größe von Zwischenergebnissen durch ***Erwartungswert*** schätzen:

$$\begin{aligned} kosten\,(\,\sigma_{Z=c}(\,R \bowtie S\,) &= kr \cdot ks + kr \cdot ks / ky \\ &= (\,kr \cdot ks\,) \cdot (\,1 + 1 / ky\,) \\ kosten\,(\,R \bowtie \sigma_{Z=c}(\,S\,)\,) &= ks + kr \cdot ks / kz \\ &= ks \cdot (\,1 + kr / kz\,) \end{aligned}$$

im Allgemeinen:
Kosten mit vorgezogener Selektion kleiner

beispielsweise:

$$kr = ks = ky = kz := 10^6$$

$$\begin{aligned} kosten(\sigma_{Z=c}(R \bowtie S)) &= (kr \cdot ks) \cdot (1 + 1/ky) \\ &= 10^6 \cdot 10^6 \cdot (1 + 1/10^6) \\ &\approx 10^{12} \end{aligned}$$

$$\begin{aligned} kosten(R \bowtie \sigma_{Z=c}(S)) &= ks \cdot (1 + kr/kz) \\ &= 10^6 \cdot (1 + 1) \\ &\approx 2 \cdot 10^6 \end{aligned}$$

12 relationaler Schemaentwurf

Schemaentwurf: Modellierung und Formalisierung

ein Informationssystem dient insbesondere dazu:

- Daten für *verschiedenartige* Benutzer verfügbar zu halten
- Anfragen und Änderungen *effizient* zu bearbeiten

Grundlage dafür sind:

- gute ***Modellierung*** des *Anwendungsfalles* mit ***Verständigung*** über:
  - welche *Seienden* (entities) der Welt sind *grundlegend* ?
  - welche *Beziehungen* (relationships) sind *grundlegend* ?

    ***Vollständigkeit*:**
    alle anderen bedeutsamen Beziehungen lassen sich daraus erschließen;
    ***Redundanzfreiheit*:**
    grundlegende Beziehungen lassen sich *nicht* auseinander erschließen
  - welche *(formalen) Handlungen* sind *grundlegend* ?

- geeignete ***Formalisierung*** der zeitunabhängigen Teile der Modellierung zu einem (Datenbank-)***Schema*** für das gewählte Datenmodell

relationale Formalisierung: Entwurfsheuritiken

- **Trennung von Gesichtspunkten**

  eine *Basisrelation* zählt ***genau einen Gesichtspunkt*** auf
  (*genau eine* Klasse von Seienden oder von grundlegenden Beziehungen):
  zu Schlüsselwerten nur *genau* zugehörige Eigenschaften hinzufügen

- **Trennung von Spezialisierungen**

  eine *Basisrelation* zählt nur *Spezialisierungen gleichen Formats* auf:
  für unterschiedliche Formate getrennte Basisrelationen einrichten

- ***Erschließbarkeit von Gesichtspunkten***

  Sichtrelationen erschließen nicht aufgezählte Gesichtspunkte:
  durch in der relationalen Algebra *ausdrückbare Anfragen* gewinnen

- **Trennungsheuristiken:** sichern *Redundanzfreiheit* für Formalisierung
- **Erschließbarkeitsheuristik:** sichert *Vollständigkeit* für Formalisierung

relationale Formalisierung: Kostenarten

- ***Speicherungskosten***

  durch Speichern der *Basisrelationen*

- ***Anfragekosten***

  bei Auswertung von *Anfragen*
  - Schätzung der Anfragekosten nur bedingt möglich;
  - bekannte Wünsche durch *Sichtrelationen* unterstützen

- ***Änderungskosten***

  bei Ausführung von *Änderungen*, inbesondere durch
  Überprüfen und Sicherstellen der *semantischen Bedingungen*

- die drei *Kostenarten* entsprechen den drei ***Darstellungsarten von Wissen***:
  - Aufzählungen
  - (Sicht-)Regeln
  - Bedingungen

12.1 funktionale Abhängigkeiten und Schlüssel

funktionale Abhängigkeiten: Konzept

- **Voraussetzungen:**

$< R \mid U \mid F >$	Relationenschema
R	Relationensymbol
U	Menge von Attributen
F	Menge von funktionalen Abhängigkeiten (über R und U)

- **inhaltlich:**

  Werte für Attribute aus $X \subset U$ bestimmen Werte für Attribute aus $Y \subset U$

- **Beispiel:**

  Name $\rightarrow$ Geschlecht im Relationenschema für PERSON:

  der (eindeutig vergebene) Attributwert für Name
  bestimmt (in der Relation für PERSON)
  den Attributwert für Geschlecht

funktionale Abhängigkeiten: Syntax

in ***Kurzform***: $X \rightarrow Y$ mit $X \subset U, Y \subset U$

als ***Menge von Horn-Klauseln***: $\boldsymbol{K} = \{ K_1, \ldots, K_l \}$ mit für $e = 1, \ldots, l$:

$$K_e \equiv a_{je} = a'_{je} \text{ :- } R(A_1 : a_1, \ldots, A_n : a_n), \text{(AND)}$$
$$R(A_1 : a'_1, \ldots, A_n : a'_n), \text{(AND)}$$
$$a_{i1} = a'_{i1}, \ldots, a_{ik} = a'_{ik}.$$

ist *gleichheitsbestimmende* Klausel für Attribut $A_{je} \in Y$,
wobei
$U = \{ A_1, \ldots, A_n \}$, $X = \{ A_{i1}, \ldots, A_{ik} \}$, $Y = \{ A_{j1}, \ldots, A_{jl} \}$

zwei auf den X-Attributen übereinstimmende Tupel aus (der Ausprägung zu) R

stimmen auch auf dem Attribut A_{je} überein

funktionale Abhängigkeiten: Semantik

- eine funktionale Abhängigkeit $X \to Y$

  ist ***gültig*** in einer Relation r mit dom $r = U$
  (r ist ***Instanz*** von $X \to Y$)
  :gdw

  für alle Tupel $\mu, \nu \in r$ **:**

  wenn $\mu\lceil X = \nu\lceil X$,

  dann auch $\mu\lceil Y = \nu\lceil Y$

- eine Menge von funktionalen Abhängigkeiten $F = \{ X_1 \to Y_1, \ldots, X_p \to Y_p \}$

  ist ***gültig*** in einer Relation r mit dom $r = U$
  (r ist ***Instanz*** von F)
  :gdw

  alle $X_i \to Y_i$ **aus** F **sind gültig in** r

funktionale Abhängigkeiten: Pragmatik

- eine funktionale Abhängigkeit $X \rightarrow Y$
  verlangt als semantische Bedingung,
  dass in den Instanzen zu einem Relationenschema
  die X-Werte eines Tupels die Y-Werte eindeutig bestimmen

- typischerweise die ***Formalisierung*** einer Modellierung der folgenden Art:
  - ein durch die X-Werte identifiziertes Seiendes
    bestimmt eindeutig
    die durch die Y-Werte beschriebenen Eigenschaften
  - ein durch die X-Werte identifiziertes Seiendes
    steht ***mit genau einem durch die Y-Werte identifizierten Seienden***
    in Beziehung

funktionale Abhängigkeiten: logische Implikation

- **einerseits:** ausdrückliche Vereinbarung im Relationenschema $< R \mid U \mid F >$
- **andererseits:** in Instanzen von F sind im Allgemeinen weitere, ***nicht ausdrücklich*** vereinbarte funktionale Abhängigkeiten gültig
- **also zu betrachten:**
  in allen Instanzen von F gültige funktionale Abhängigkeiten
- F ***impliziert (logisch)*** $X \to Y$, $F \models X \to Y$
  :gdw

  für alle *r* mit dom *r* = *U* :

  wenn F in r gültig,

  dann auch $X \to Y$ in r gültig
- $F^+ := \{ X \to Y \mid F \models X \to Y \}$

Reflexivität, Transitivität und Erweiterung

- ***korrekte Implikationen*** für Klasse der funktionalen Abhängigkeiten:

  [***Reflexivität***] für alle $Y \subset X$: $\quad \emptyset \quad \models \quad X \rightarrow Y$

  [***Transitivität***] für alle X, Y, Z: $\quad \{ X \rightarrow Y, Y \rightarrow Z \} \quad \models \quad X \rightarrow Z$

  [***Erweiterung***] für alle X, Y und
  für alle $W \supset Z$: $\quad \{ X \rightarrow Y \} \quad \models \quad X \cup W \rightarrow Y \cup Z$

- hieraus ***Entscheidungsalgorithmus*** gewinnen

  für das Problem "$X \rightarrow Y \in F^+$":

  - erzeuge für X mit Hilfe von F schrittweise alle Attribute A mit $X \rightarrow A \in F^+$;

  - prüfe, ob Y in der erzeugten Menge enthalten ist

Grundlage für Entscheidungsalgorithmus: Definition von Abschluss

- ***Abschluss*** (closure)
  einer Attributmenge $X \subset U$ unter funktionalen Abhängigkeiten F,
  $cl(F, X)$,
  ist die (bezüglich $\subset$) kleinste Menge mit den folgenden Eigenschaften:

  1. $X \subset cl(F, X)$

  2. **wenn** $R \subset cl(F, X)$
     und $R \rightarrow S \in F$,
     dann auch $S \subset cl(F, X)$

Korrektheit und Vollständigkeit des Entscheidungsverfahrens: Satz

1. $X \rightarrow cl\,(F, X) \in F^+$

2. $X \rightarrow Y \in F^+$ *genau dann, wenn* $Y \subset cl\,(F, X)$

Korrektheit des Abschlusses: Beweis

Behauptung: $X \to cl(F, X) \in F^+$

betrachte
"Berechnung" von $cl(F, X)$, d.h.
Folge $X = X_0, X_1, \ldots, X_k = cl(F, X)$ mit

für $i = 0, \ldots, k\text{-}1$:	Schritt
$R_i \subset X_i$ und $R_i \to S_i \in F$,	Bedingung
$X_{i+1} = X_i \cup S_i$	Anwendung

durch Induktion über *i*: $X \to X_i \in F^+$

i* = 0:** $X \to X \in F^+$ *Spezialfall der* ***Reflexivität

***i*+1:** $X \to X_{i+1} \in F^+$ prüfen anhand Definition von "impliziert"

zu zeigen: $X \rightarrow X_{i+1} \in F^+$

betrachte Relation r mit: dom $r = U$ und
F in r gültig (1a)

insbesondere: $R_i \rightarrow S_i$ in r gültig (1b)

gemäß Induktionsannahme: $X \rightarrow X_i$ in r gültig (2)

betrachte Tupel $\mu, \nu \in r$ mit: $\mu\lceil X = \nu\lceil X$ (3)

mit (3) wegen (2): $\mu\lceil X_i = \nu\lceil X_i$ (4)

mit (4) wegen $R_i \subset X_i$: $\mu\lceil R_i = \nu\lceil R_i$ (5)

mit (5) wegen (1b): $\mu\lceil S_i = \nu\lceil S_i$ (6)

mit (4), (6) wegen $X_{i+1} = X_i \cup S_i$: $\mu\lceil X_{i+1} = \nu\lceil X_{i+1}$

Korrektheit und Vollständigkeit: Beweis

Behauptung: $X \rightarrow Y \in F^+$ *genau dann, wenn* $Y \subset cl(F, X)$

"$\Leftarrow$" [Korrektheit]:

betrachte: $Y \subset cl(F, X)$ (7)

gemäß Korrektheit des Abschlusses: $X \rightarrow cl(F, X) \in F^+$ (8)

mit (7), (8),
offensichtlich nach Definition: $X \rightarrow Y \in F^+$

Behauptung: $X \rightarrow Y \in F^+$ *genau dann, wenn* $Y \subset cl(F, X)$

"⇒" [Vollständigkeit in Kontraposition]:

betrachte: $Y \not\subset cl(F, X)$ (9)

Gegenbeispiel für "F impliziert $X \rightarrow Y$"

als Armstrong-Relation r :

a) F in r gültig

b) $X \rightarrow Y$ in r ***nicht*** gültig

r enthalte genau zwei Tupel μ und ν: $\mu(A) := 1$ für alle $A \in U$ (10)

$\nu(A) := 1$ falls $A \in cl(F, X)$ (11)

$\nu(A) := 0$ falls $A \notin cl(F, X)$ (12)

r	$cl(F, X)$	$U \setminus cl(F, X)$
μ	1 ... 1	1 ... 1
ν	1 ... 1	0 ... 0

nach Voraussetzung (9): ex. Attribut $A_0 \in Y \setminus cl(F, X)$

damit nach Konstruktion: μ verschieden von ν

also: Eigenschaft b) ist erfüllt

Wiederholung: Definition der Armstrong-Relation

r enthalte genau zwei Tupel μ und ν:

$$\mu(A) := 1 \quad \text{für alle } A \in U \tag{10}$$

$$\nu(A) := 1 \quad \text{falls } A \in cl(F, X) \tag{11}$$

$$\nu(A) := 0 \quad \text{falls } A \notin cl(F, X) \tag{12}$$

r	$cl(F, X)$	$U \setminus cl(F, X)$
μ	1 ... 1	1 ... 1
ν	1 ... 1	0 ... 0

zeige Eigenschaft a) F in r gültig

betrachte: $R_i \rightarrow S_i \in F$

angenommen,

für die einzigen zwei Tupel aus r: $\mu\lceil R_i = \nu\lceil R_i$

nach Konstruktion von r: $R_i \subset cl(F, X)$

nach Definition des Abschlusses: $S_i \subset cl(F, X)$

nach Konstruktion von r: $\mu\lceil S_i = \nu\lceil S_i$

Beispiel

Schema:

U = { Id , Geschlecht , DaPa , Arzt , DaAr , ArtPro }

F = { Id $\rightarrow$ Geschlecht,
Id $\rightarrow$ DaPa,
Arzt $\rightarrow$ DaAr,
Id,Arzt $\rightarrow$ ArtPro }

Abschluss *cl* (*F* , { Id , Arzt }) berechnen:

X_0 = { Id , Arzt }

X_1 = { Id , Geschlecht , Arzt }	vermöge	Id $\rightarrow$ Geschlecht
X_2 = { Id , Geschlecht , DaPa , Arzt }	vermöge	Id $\rightarrow$ DaPa
X_3 = { Id , Geschlecht , DaPa , Arzt , DaAr }	vermöge	Arzt $\rightarrow$ DaAr
X_4 = { Id , Geschlecht , DaPa , Arzt , DaAr , ArtPro }	vermöge	Id,Arzt $\rightarrow$ ArtPro

also: Id,Arzt $\rightarrow U \in F^+$

ferner: $cl(F, \{ \text{Id} \}) = \{ \text{Id} , \text{Geschlecht} , \text{DaPa} \}$

$cl(F, \{ \text{Arzt} \}) = \{ \text{Arzt} , \text{DaAr} \}$

insbesondere: Id $\rightarrow U \notin F^+$

Arzt $\rightarrow U \notin F^+$

formale Definition von “Schlüssel”

anschaulich: minimale Attributmenge, deren Werte
ein Tupel innerhalb einer Ausprägung eindeutig bestimmen

formal:
$< R \mid U \mid F >$ Relationenschema

1. eine Menge von Attributen $X \subset U$ ist ***Schlüssel*** (key) von $< R \mid U \mid F >$
   :gdw
   1. [***Eindeutigkeit***] $X \rightarrow U \in F^+$
   2. [***Minimalität***] für alle $Y \subset\neq X$: $Y \rightarrow U \notin F^+$

2. ein Attribut $A \in U$ ist ***Schlüsselattribut*** von $< R \mid U \mid F >$
   :gdw
   es gibt einen Schlüssel X von $< R \mid U \mid F >$ mit $A \in X$

3. ein Attribut $A \in U$ ist ***Nichtschlüsselattribut*** von $< R \mid U \mid F >$
   :gdw
   A kommt in *keinem* Schlüssel von $< R \mid U \mid F >$ vor

Beispiel

Schema:

$$U = \{ \text{Id}, \text{Geschlecht}, \text{DaPa}, \text{Arzt}, \text{DaAr}, \text{ArtPro} \}$$

$$F = \{ \text{Id} \rightarrow \text{Geschlecht},\ \text{Id} \rightarrow \text{DaPa},\ \text{Arzt} \rightarrow \text{DaAr},\ \text{Id,Arzt} \rightarrow \text{ArtPro} \}$$

schon gezeigt:

$$\text{Id, Arzt} \rightarrow U \in F^+$$

$$\text{Id} \rightarrow U \notin F^+$$

$$\text{Arzt} \rightarrow U \notin F^+$$

also: $\{ \text{Id, Arzt} \}$ Schlüssel von $< | U | F >$

im Allgemeinen: ein Relationenschema kann mehrere Schlüssel besitzen

Beispiel:

$$U = \{ A, B, C \}$$

$$F = \{ A,B \rightarrow C,\quad C \rightarrow B \}$$

- sowohl $\{ A, B \}$ als auch $\{ A, C \}$ sind Schlüssel
- alle Attribute sind Schlüsselattribute

Relationenschemas mit genau einem Schlüssel

$< R \mid U \mid F >$ Relationenschema

$extrem(U, F) := \{ A \mid \quad A \in U,\ U \setminus \{ A \} \rightarrow A \notin F^+ \}$ ***Extremalattribute***

folgende Aussagen sind äquivalent:

1. $< R \mid U \mid F >$ besitzt ***genau einen*** Schlüssel
2. $extrem(U, F)$ ist Schlüssel von $< R \mid U \mid F >$
3. $extrem(U, F) \rightarrow U \in F^+$

Beweis: entfällt

12.2 Normalformen und Schema-Transformationen

3.Normalform und Boyce / Codd-Normalform: Konzept

- **Formalisierung** der Entwurfsheuristik "**Trennung von Gesichtspunkten**"

- zeichnet diejenigen Relationenschemas aus, in denen

  die linken Seiten aller
  (vereinbarten oder implizierten) funktionalen Abhängigkeiten

  einen **Schlüssel** enthalten

- die einzigen durch funktionale Abhängigkeiten *ausdrückbaren Strukturen*

  gegeben durch

  - **Schlüssel** und
  - die jeweils von ihnen funktional abhängigen Attribute

3.Normalform und Boyce / Codd-Normalform: Definition

1. ein Relationenschema $< R \mid U \mid F >$ ist in ***3.Normalform***
   :gdw
   für alle $Z \subset U$,
   für alle *Nichtschlüsselattribute* $A \in U$:
   wenn $Z \rightarrow A \in F^+$ und $A \notin Z$,
   dann $Z \rightarrow U \in F^+$

2. ein Relationenschema $< R \mid U \mid F >$ ist in ***Boyce / Codd-Normalform***
   :gdw
   für alle $Z \subset U$,
   für alle $A \in U$:
   wenn $Z \rightarrow A \in F^+$ und $A \notin Z$,
   dann $Z \rightarrow U \in F^+$

Boyce / Codd-Normalform ist **Verschärfung** von *3.Normalform*,
weil Anforderung für *alle* Attribute

3.Normalform und Boyce / Codd-Normalform: Eigenschaften

- *Boyce / Codd-Normalform* ist **echte Verschärfung** von *3.Normalform*

- Beispiel: $< \quad R \quad | \quad \{ A, B, C \} \; | \quad \{ A,B \rightarrow C, \; C \rightarrow B \} \quad >$
  - in 3.Normalform, da keine Nichtschlüsselattribute
  - *nicht* in Boyce / Codd-Normalform, weil
    $C \rightarrow B \in F \subset F^+$, aber
    $C \rightarrow A \notin F^+$

- eigentlich: schärfere Boyce / Codd-Normalform immer ***wünschenswert***

- aber: mit schwächerer 3.Normalform begnügen, wenn Boyce / Codd-Normalform "unverträglich"

partielle Abhängigkeiten

in einem Relationenschema in **Boyce / Codd-Normalform** kann folgende (unerwünschte) Situation ***nicht auftreten***:

partielle Abhängigkeit der Form:

X ist Schlüssel

$Y \subset\neq X$, d.h. insbesondere $Y \rightarrow X \notin F^+$

$A \notin Y$

$Y \rightarrow A \in F^+$

Attribut A ist nur von einer ***echten Teilmenge*** des Schlüssels X funktional abhängig

transitive Abhängigkeiten

in einem Relationenschema in **Boyce / Codd-Normalform** kann folgende (unerwünschte) Situation *nicht auftreten*:

transitive Abhängigkeit der Form:

$$X \rightarrow Y \in F^+$$
$$Y \rightarrow X \notin F^+$$
$$A \notin Y$$
$$Y \rightarrow A \in F^+$$

Attribut A ist ***transitiv*** über Y von X funktional abhängig, ohne dass Y funktional äquivalent mit X ist

Relationenschemas in Boyce / Codd-Normalform

- **Definition**

  ein Relationenschema $< R \mid U \mid F >$ ist in ***Boyce / Codd-Normalform***
  :gdw für alle $Z \subset U$, für alle $A \in U$:
  wenn $Z \to A \in F^+$ und $A \notin Z$,
  dann $Z \to U \in F^+$

- **Satz**

  ein Relationenschema $< R \mid U \mid F >$ ist in ***Boyce / Codd-Normalform***
  :gdw
  für alle $X \to Y \in F$:
  wenn $Y \not\subset X$
  dann $X \to U \in F^+$

Bemerkung: Satz liefert effizientes *Entscheidungsverfahren*

Beweis: entfällt

Kosten und Normalformen

- für Basisrelation in ***Normalform***

  Struktur im Wesentlichen bereits durch ***Schlüssel***,
  am besten durch *genau einen Schlüssel* bestimmt

- dann ***Anfragekosten*** für Selektionen gering durch

  geeignet angelegte ***Zugriffsstrukturen***
  (B*-Bäume, Hash-Verfahren, Sortierungen bezüglich der Schlüsselattribute)

- dann ***Änderungskosten*** gering:

  - weitgehend nur *lokale Schlüsselbedingungen* zu überprüfen
  - “Änderungs-Anomalien” treten nicht auf
  - “redundanzfreie Speicherung”

Überlegung zur Redundanzfreiheit

(3., Boyce/Codd und weitere)
Normalformen besagen auch,

dass die gespeicherten Relationen
im gewissen Sinne ***redundanzfrei*** sind und

dass damit die ***Speicherungskosten*** gering sind

Überlegung zur Redundanzfreiheit: Boyce / Codd-Normalform

betrachte: $< R \mid U \mid F >$ in ***Boyce / Codd-Normalform***,
nichttriviale funktionale Abhängigkeit $Z \rightarrow A$

dann: $Z \rightarrow U \in F^+$

zerlege probeweise durch Projektion:

$$Y_1 := Z \cup \{ A \} \quad \text{und} \quad Z \subset Y_2 := U \setminus \{ A \}$$

vererbte Eindeutigkeitseigenschaft:

jede Projektion einer gespeicherten Relation r
enthält genauso viele Tupel wie r :

$$\| \pi_{Y_1}(r) \| = \| \pi_{Y_2}(r) \| = \| r \|$$

beim Wiederherstellen [siehe später: unter BCNF stets möglich]**:**

$$r = \pi_{Y_1}(r) \bowtie \pi_{Y_2}(r)$$

keine “wirklich neuen Tupel erzeugt”,
sondern passende Tupel “verlängern einander”

vererbte Eindeutigkeitseigenschaft (Wiederholung)**:**

jede Projektion einer gespeicherten Relation r
enthält genauso viele Tupel wie r :

$$\| \pi_{Y_1}(r) \| \quad = \quad \| \pi_{Y_2}(r) \| \quad = \quad \| r \|$$

beim Wiederherstellen (Wiederholung)**:**

$$r \quad = \quad \pi_{Y_1}(r) \bowtie \pi_{Y_2}(r)$$

keine “wirklich neuen Tupel erzeugt”,
sondern passende Tupel “verlängern einander”

also: Speicherungskosten durch Zerlegen ***niemals*** **vermindert**,
sondern ***stets erhöht*** durch *mehrfaches Abspeichern*
der Werte für Verbundattribute

umgekehrt: liegt *keine* Normalform vor,
so gibt es ein ***Beispiel*** r mit

$$\| \pi_{Y_1}(r) \| \; < \quad \| r \|$$

also Normalisierung: durch “Semantik-erhaltendes” Zerlegen

Instanzenunterstützung von Datenbankschemas: Definition

$$RS = < < R_1 | X_1 | SC_1 >, \ldots, < R_n | X_n | SC_n > | SC | >$$
$$RS' = < < R'_1 | X'_1 | SC'_1 >, \ldots, < R'_{n'} | X'_{n'} | SC'_{n'} > | SC' | >$$

1. *RS'* ***unterstützt*** *RS* (mit Anfragesprache ***L***)
   :gdw
   es gibt (in ***L*** ausdrückbare) relationale Anfragen $Q_1, \ldots, Q_n$ mit:

   i) $Q_i : \text{sat} < < | X'_1 | > , \ldots , < | X'_{n'} | > | | > \rightarrow \text{sat} < | X_i | >$

   ii) die Anfrage $Q := (Q_1 , \ldots , Q_n)$ mit $Q(f') := (Q_1(f') , \ldots , Q_n(f'))$
   ist **surjektiv bezüglich Instanzen**, d.h.
   $\mathbf{sat_{RS} \subset Q [sat_{RS'}]}$

2. *RS'* ***unterstützt*** *RS* ***treu***, wenn zusätzlich $\mathbf{Q [sat_{RS'}] \subset sat_{RS}}$

3. *RS'* ***unterstützt*** *RS* ***eindeutig***, wenn zusätzlich *Q* **auf** $\mathbf{sat_{RS'}}$ **injektiv**

Verbundabhängigkeiten

$< R \mid U \mid F >$ Universalrelation-Schema

$U = \bigcup_{i=1,\ldots,n} X_i$ Überdeckung der Attributmenge

Syntax: $\bowtie [X_1, \ldots, X_n]$ ist ***Verbundabhängigkeit***

Semantik: $\bowtie [X_1, \ldots, X_n]$ ist ***gültig*** in Relation r mit dom $r = U$

(r ist ***Instanz*** von $\bowtie [X_1, \ldots, X_n]$)

:gdw

$r = \bowtie_{i=1,\ldots,n} \pi_{X_i}(r)$

Erinnerung: "$\subset$" gilt immer!

Implikationen: entsprechend wie für funktionale Abhängigkeiten definiert; es gibt Entscheidungsverfahren!

Verbund-Unterstützung eines Universalrelation-Schemas: Satz

$< R \mid U \mid SC >$ Universalrelation-Schema

$RS = \; << R_1 \mid X_1 \mid \emptyset >, \ldots, < R_n \mid X_n \mid \emptyset > \mid \emptyset \mid >$ Datenbankschema

$U = \bigcup_{i=1,\ldots,n} X_i$ Überdeckung

$Q : \text{sat}_{RS} \rightarrow \text{sat}_{< R \mid U \mid >}$ Verbund-Anfrage

$Q(r_1, \ldots, r_n) := \bowtie_{i=1,\ldots,n} r_i$

Dann sind folgende Aussagen äquivalent:

1. $\text{sat}_{< R \mid U \mid SC >} \subset Q[\text{sat}_{RS}]$,
   d.h. RS ***unterstützt*** $< R \mid U \mid SC >$ mit ***Unterstützungsanfrage*** Q

2. $\bowtie [X_1, \ldots, X_n] \in SC^+$

Beweis: Übungsaufgabe (folgt unmittelbar aus den Definitionen)

Verbund-Unterstützung bei funktionalen Abhängigkeiten: Satz

$< R \mid U \mid F >$ Universalrelation-Schema

F funktionale Abhängigkeiten

$RS = \; << R_1 \mid X_1 \mid \emptyset >, < R_2 \mid X_2 \mid \emptyset > \mid \emptyset \mid >$ Datenbankschema mit

$X_1 \cup X_2 = U$

$X_1 \cap X_2 \neq \emptyset$

Dann sind folgende Aussagen äquivalent:

1. RS unterstützt $< R \mid U \mid F >$
   mit Unterstützungsanfrage $Q(r_1, r_2) := r_1 \bowtie r_2$

2. $\bowtie [X_1, X_2] \in F^+$

3. $X_1 \cap X_2 \rightarrow X_1 \setminus X_2 \in F^+$ oder $X_1 \cap X_2 \rightarrow X_2 \setminus X_1 \in F^+$

Beweis: entfällt

Verbund-unterstützte Zerlegung in Boyce / Codd-Normalform: Satz

$< R \mid U \mid F >$ Universalrelation-Schema

F funktionale Abhängigkeiten

Dann gibt es ein Datenbankschema

$RS = \; < \; < R_1 \mid X_1 \mid F_1 > , \ldots , < R_n \mid X_n \mid F_n > \; \mid \mid \; >$

mit folgenden Eigenschaften:

1. RS unterstützt $< R \mid U \mid F >$
   mit Unterstützungsanfrage $Q(r_1 , \ldots , r_n) := \bowtie_{i=1, \ldots, n} r_i$
2. $F_i = \{ \; X \rightarrow Y \; \mid \; X \cup Y \subset X_i , \; X \rightarrow Y \in F^+ \; \}$
3. jede Komponente $< R_i \mid X_i \mid F_i >$ ist in ***Boyce / Codd-Normalform***

(konstruktiver, induktiver) Beweis gemäß folgender Idee:

zerlege Schema schrittweise,
bis alle “verbotenen Teilstrukturen” entfernt sind

1. RS unterstützt $< R \mid U \mid F >$ mit Unterstützungsanfrage $Q(r_1, \ldots, r_n) := \bowtie_{i=1,\ldots,n} r_i$
2. $F_i = \{ \quad X \rightarrow Y \quad \mid \quad X \cup Y \subset X_i, \; X \rightarrow Y \in F^+ \quad \}$
3. jede Komponente $< R_i \mid X_i \mid F_i >$ ist in Boyce / Codd-Normalform

Induktionsbeginn:

$RS_0 := < \quad < R \mid U \mid F^+ > \quad \mid \quad \mid \quad >$

erfüllt sicherlich die Eigenschaften 1. und 2.

1. RS unterstützt $< R \mid U \mid F >$ mit Unterstützungsanfrage $Q(r_1, \ldots, r_n) := \bowtie_{i=1,\ldots,n} r_i$
2. $F_i = \{ \; X \to Y \mid X \cup Y \subset X_i, \; X \to Y \in F^+ \; \}$
3. jede Komponente $< R_i \mid X_i \mid F_i >$ ist in Boyce / Codd-Normalform

Induktionsannahme: RS_j erfülle 1. und 2., **aber noch nicht 3.**

dann: es gibt "verbotene" Komponente $< R_i \mid X_i \mid F_i >$ mit

$$Z \subset X_i, \quad A \in X_i, \quad Z \to A \in F^+, \quad A \notin Z, \; Z \to X_i \notin F^+$$

zerlege:

$$X_{i1} := Z \cup \{ \; A \mid Z \to A \in F^+ \; \}$$
$$X_{i2} := Z \cup (X_i \setminus X_{i1})$$

***RS*$_{j+1}$:** ersetze "verbotene" Komponente $< R_i \mid X_i \mid F_i >$ durch

$< R_{i1} \mid X_{i1} \mid F_{i1} >$ und $< R_{i2} \mid X_{i2} \mid F_{i2} >$;

definiere dabei F_{i1} und F_{i2} gemäß Eigenschaft 2.

nach Konstruktion: RS_{j+1} erfüllt wieder 1. und 2.

Zerlegungsvorgang bricht ab:

- U ist endlich

- Attributmengen X_{i1} und X_{i2} der neuen Komponenten
  sind jeweils *echte* Teilmengen
  der Attributmenge X_i der alten Komponente

Beispiel für Zerlegung in Boyce / Codd-Normalform

U = { Id , Geschlecht , DaPa , Arzt , DaAr , ArtPro }
F = { Id → Geschlecht,
Id → DaPa,
Arzt → DaAr,
Id,Arzt → ArtPro }

< | U | F > nicht in Boyce / Codd-Normalform:
Arzt → DaAr "verbotene Teilstruktur", entsprechend zerlegen:

< R_1 | { Arzt , DaAr } | { Arzt → DaAr } >
< R_2 | { Id , Geschlecht, DaPa , Arzt , ArtPro } | { Id →Geschlecht ,
Id → DaPa , Id,Arzt →ArtPro } >

zweites Relationenschema nicht in Boyce / Codd-Normalform:
Id → DaPa "verbotene Teilstruktur", entsprechend zerlegen:

< R_{21} | { Id , Geschlecht , DaPa } | { Id → Geschlecht , Id → DaPa } >
< R_{22} | { Id , Arzt , ArtPro } | { Id,Arzt → ArtPro } >
R_1, R_{21} und R_{22} in Boyce / Codd-Normalform

Boyce / Codd-Normalform und treue Unterstützung: Beispiel

$< \quad R \quad | \quad \{ A , B , C \} \quad | \quad \{ A,B \rightarrow C,\ C \rightarrow B \} \quad >$
nicht in Boyce / Codd-Normalform:

$C \rightarrow B$ verbotene Teilstruktur, (versuchsweise) entsprechend zerlegen:

$< \quad R_1 \quad | \quad \{ B , C \} \quad | \quad \{ C \rightarrow B \} \quad >$
$< \quad R_2 \quad | \quad \{ A , C \} \quad | \quad \emptyset \quad >$

Beobachtungen:

- ursprünglich vereinbarte funktionale Abhängigkeit $A,B \rightarrow C$
  nicht mehr "repräsentiert"
- es gibt Beispielinstanz (r_1 , r_2) für Zerlegung,
  deren Verbund keine Instanz des Universalrelation-Schemas ist
- keine andere Zerlegung liefert *gleichzeitig*
  Relationenschemas in Boyce / Codd-Normalform und treue Unterstützung

Beispielinstanz (r_1, r_2) für Zerlegung,
deren Verbund keine Instanz des Universalrelation-Schemas ist:

r_1	B	C
	b	c
	b	d

r_2	A	C
	a	c
	a	d

$r_1 \bowtie r_2$	A	B	C
	a	b	c
	a	b	d

treue Zerlegungen: Satz

$< R \mid U \mid F >$ Universalrelation-Schema

$RS = < < R_1 \mid X_1 \mid F_1 >, \ldots, < R_n \mid X_n \mid F_n > \mid \mid >$ Datenbankschema mit

$U = \bigcup_{i=1,\ldots,n} X_i$ Überdeckung

F und $F_1, \ldots, F_n$ Mengen von funktionalen Abhängigkeiten

$Q : \text{sat}_{RS} \rightarrow \text{sat}_{< R \mid U \mid >}$ Verbund-Anfrage mit

$Q(r_1, \ldots, r_n) := \bowtie_{i=1,\ldots,n} r_i$

Wenn $(\bigcup_{i=1,\ldots,n} F_i)^+ \supset F$,

dann $Q[\text{sat}_{RS}] \subset \text{sat}_{< R \mid U \mid F >}$

treue Zerlegungen: Beweis

Behauptung: **wenn** $(\bigcup_{i=1,\ldots,n} F_i)^+ \supseteq F$,
dann $Q[\text{sat}_{RS}] \subset \text{sat}_{<R \mid U \mid F>}$

betrachte: $(r_1, \ldots, r_n) \in \text{sat}_{RS}$ und
$r := \bowtie_{i=1,\ldots,n} r_i$

wir zeigen unten: alle $X \to Y \in \bigcup_{i=1,\ldots,n} F_i$ in r gültig

dann ebenfalls: alle $X \to Y \in (\bigcup_{i=1,\ldots,n} F_i)^+$ in r gültig

Voraussetzung: alle $X \to Y \in F$ in r gültig

also: $r \in \text{sat}_{<R \mid U \mid F>}$

noch zu zeigen: alle $X \rightarrow Y \in \bigcup_{i=1,\ldots,n} F_i$ in r gültig

betrachte: $X \rightarrow Y \in F_i$

insbesondere gilt: $X \cup Y \subset X_i$

betrachte: zwei Tupel $\mu \in r$ und $\nu \in r$ mit
$\mu\lceil X = \nu\lceil X$

Definition von r:
$\mu_i := \mu\lceil X_i \in r_i$
$\nu_i := \nu\lceil X_i \in r_i$

ferner: $\mu_i\lceil X = \nu_i\lceil X$

$X \rightarrow Y$ in r_i gültig: $\mu_i\lceil Y = \nu_i\lceil Y$

also auch: $\mu\lceil Y = \nu\lceil Y$

treue Verbund-Unterstützung bei funktionalen Abhängigkeiten: Satz

$< R \mid U \mid F >$ Universalrelation-Schema

$RS = < < R_1 \mid X_1 \mid F_1 >, \ldots, < R_n \mid X_n \mid F_n > \mid \mid >$ Datenbankschema mit

$U = \bigcup_{i=1,\ldots,n} X_i$ Überdeckung

F und $F_1, \ldots, F_n$ Mengen von funktionalen Abhängigkeiten

$F_i^+ \subset \{ R \to S \mid R \cup S \subset X_i, \; R \to S \in F^+ \}$

Wenn $\bowtie [X_1, \ldots, X_n] \in F^+$ **und**

$(\bigcup_{i=1,\ldots,n} F_i)^+ \supset F$,

dann *unterstützt* RS das Universalrelation-Schema $< R \mid U \mid F >$ *treu* mit Unterstützungsanfrage

$Q(r_1, \ldots, r_n) := \bowtie_{i=1,\ldots,n} r_i$.

Beweis: Korollar zu vorangehenden Aussagen

treue und Verbund-unterstützende Synthese in 3.Normalform: Satz

$< R \mid U \mid F >$ Universalrelation-Schema

F funktionale Abhängigkeiten

Dann gibt es ein Datenbankschema

$RS = \quad < \quad < R_1 \mid X_1 \mid F_1 > , \ldots , < R_n \mid X_n \mid F_n > \quad \mid \mid \quad >$

mit folgenden Eigenschaften:

1. RS unterstützt $< R \mid U \mid F >$ ***treu***
   mit Unterstützungsanfrage $Q(r_1 , \ldots , r_n) := \bowtie_{i=1, \ldots, n} r_i$

2. $F_i = \{ \quad X \rightarrow Y \quad \mid \quad X \cup Y \subset X_i , \quad X \rightarrow Y \in F^+ \quad \}$

3. jede Komponente $< R_i \mid X_i \mid F_i >$ ist in ***3.Normalform***

konstruktive Beweisidee:

- bestimme zu gegebenen funktionalen Abhängigkeiten F

  eine dazu äquivalente Menge G mit $F^+ = G^+$,
  die in gewissem Sinne ***redundanzfrei*** ist

- "synthetisiere" aus G Relationenschemas,
  die die ***funktionalen Abhängigkeiten "repräsentieren"***

- sichere ***Verbund-Unterstützung***:

  füge ein weiteres Relationenschema mit
  Schlüssel K für die gesamte Attributmenge U hinzu,

  falls keines der "synthetisierten" Relationenschemas
  schon einen solchen Schlüssel enthält

Syntheseverfahren

1.[*mache funktionale Abhängigkeiten elementar*]

setze $G := \{ X \to A \mid$ es gibt $X \to Y \in F$ mit $A \in Y \}$

(ersetze $X \to \{ A_1, \ldots, A_k \} \in F$ durch $X \to A_1, \ldots, X \to A_k$)

2.[*entferne lokale Redundanz*]

für $X \to A \in G$ bestimme minimales $X' \subset X$ mit $X' \to A \in G^+$

und, falls $X' \subset\neq X$, ersetze $X \to A$ durch $X' \to A$

3.[*entferne globale Redundanz*]

für $X \to A \in G$ prüfe, ob $X \to A \in (G \setminus \{ X \to A \})^+$;

und, falls zutreffend, entferne $X \to A$ aus G

4.[*"synthetisiere" Relationenschemas, "repräsentiere" F*]

setze $Soll_G := \{ \quad X \quad | \quad$ es gibt $A \in U$ mit $X \to A \in G \quad \}$

(in G vorkommenden linken Seiten)

und für $Y_i \in Soll_G$ bilde Relationenschema $< R_i \mid X_i \mid F_i >$ mit

$$X_i := Y_i \cup \{ \quad A \quad | \quad Y_i \to A \in G \}$$

$$F_i := \{ \quad X \to Y \quad | \quad X \cup Y \subset X_i, \quad X \to Y \in G^+ \quad \}$$

$$\supset \{ \quad Y_i \to A \quad | \quad Y_i \to A \in G \quad \}$$

5.[*Verbund-Unterstützung*]

falls für alle $Y_i \in Soll_G$ gilt: $Y_i \to U \notin G^+$

dann bestimme einen Schlüssel K von $< R \mid U \mid F >$ und

bilde ein weiteres Relationenschema mit

$X_i := K$ und $F_i := \emptyset$

Beispiel für Syntheseverfahren

Eingaben: U = { Id , Geschlecht , DaPa , Arzt , DaAr , ArtPro }

F = { Id → Geschlecht,
Id → DaPa,
Arzt → DaAr,
Id,Arzt → ArtPro }

Schritte 1-3: $G = F$

(vereinbarte funktionale Abhängigkeiten bleiben unverändert)

Schritt 4: $Soll_G$ = { Id , Arzt , { Id, Arzt } }

und "synthetisiere":

< R_1 | { Id , Geschlecht , DaPa } | { Id → Geschlecht, Id → DaPa } >

< R_2 | { Arzt , DaAr } | { Arzt → DaAr } >

< R_3 | { Id , Arzt , ArtPro } | { Id,Arzt → ArtPro } >

Schritt 5: cl (F, { Id , Arzt }) = U,

so dass kein weiteres Relationenschema hinzugefügt wird

Teil V

Mehrbenutzerbetrieb, Transaktionen

und Sicherheit

13 Transaktionen

13.1 Anforderungen und Read/Write-Modell

Transaktionen: einige Vorüberlegungen

- auch (formale) ***Handlungsfolgen***

- semantische Bedingungen auch als ***dynamische Gesichtspunkte***:
  - derart ändern, dass die semantischen Bedingungen wieder erfüllt sind
  - neben der eigentlich gewünschten Änderung gegebenenfalls auch Folgeänderungen notwendig
  - *Änderungsfolgen* als *unteilbare Operationen* ausführen, d.h. entweder *gar nicht* oder *vollständig*

- viele und verschiedenartige Benutzer

  arbeiten im Allgemeinen
  parallel und möglichst unabhängig voneinander

Parallelität und Unabhängigkeit: ein Beispiel

Datenbankobjekt: R

ein Programm: $P \equiv$

```
BEGIN
    Read R.A;
    R.A := R.A + 1;
    Write R.A
END
```

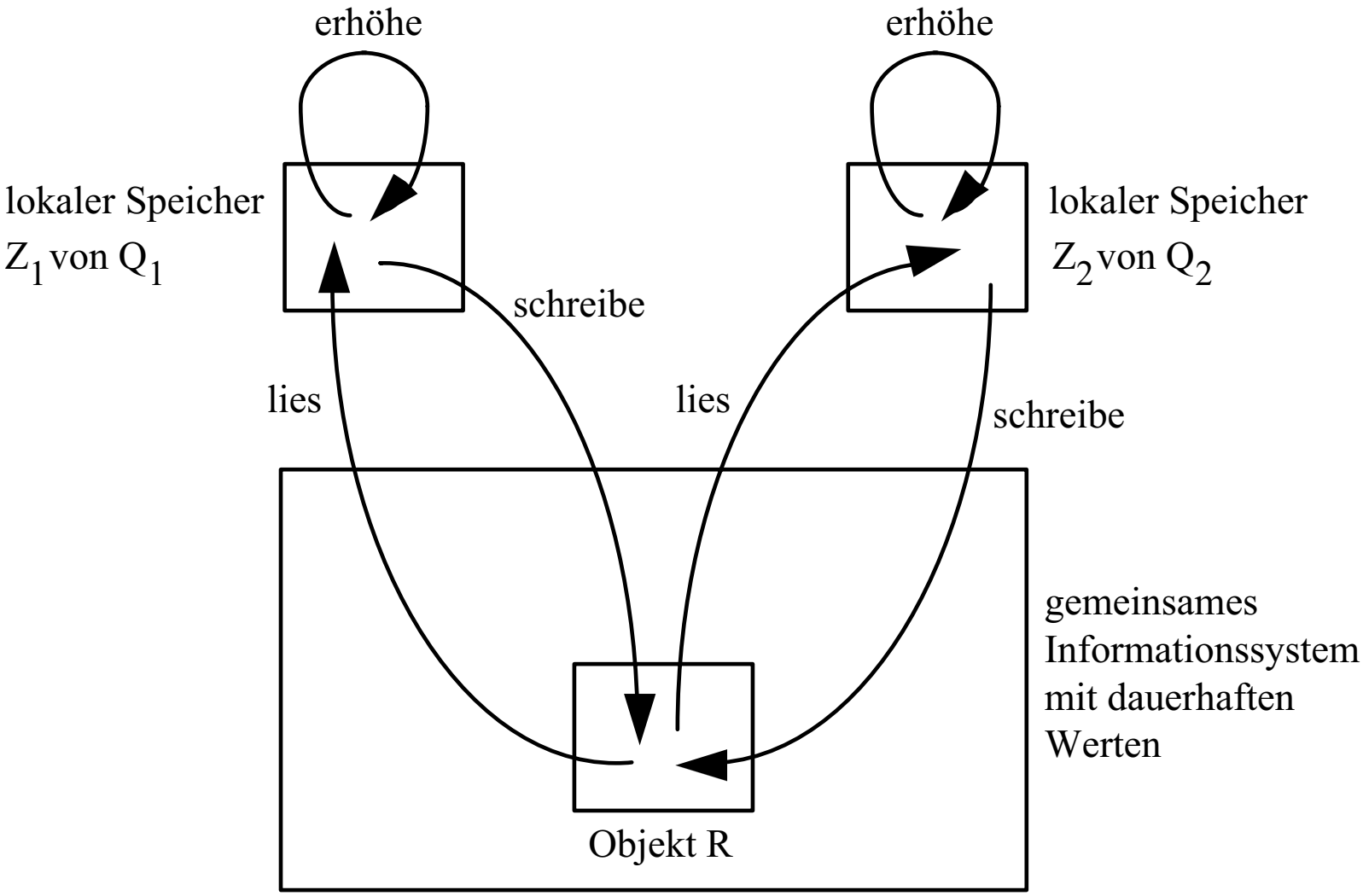
zwei parallel und unabhängig voneinander das Programm P ausführende Prozesse: Q_1 bzw. Q_2

lokaler Speicher für Prozess Q_i : Z_i

***Semantik* des Programms:**

- lies A-Komponente von R im gemeinsamen Informationssystem, merke Wert im lokalen Speicher: $Z_i := R.A$
- erhöhe diesen Wert im lokalen Speicher um 1: $Z_i := Z_i + 1$
- schreibe zurück in A-Komponente von R im gemeinsamen Informationssystem: $R.A := Z_i$

eine parallele Nutzung eines gemeinsamen Informationssystems



eine unerwünschte Ausführung

Zeit	Anweisung für Prozeß		ausgeführte Semantik	Wertverlauf im gemeinsamen Informationssystem	im lokalen Speicher	
	Q_1	Q_2		R.A	Z_1	Z_2
1				10	—	—
2	Read R.A		Z_1:= R.A	10	10	—
3		Read R.A	Z_2:= R.A	10	10	10
4a	R.A := R.A+1		Z_1:= Z_1 + 1	10	11	
4b		R.A := R.A+1	Z_2:= Z_2 + 1			11
5		Write R.A	R.A := Z_2	11	11	11
6	Write R.A		R.A := Z_1	11	11	11

eine weitere Vorüberlegung

- Daten ***dauerhaft*** und ***verlässlich*** verwalten,

  insbesondere:

  einmal als wirksam erklärte
  Änderungen der gemeinsamen Daten sollen auch

  - bei schwierigen Fehlerlagen oder

  - gar vollständigen Systemzusammenbrüchen

  ihre Gültigkeit behalten

Transaktionen: Pragmatik und Syntax

- programmiersprachliches Konstrukt zur Unterstützung der Vorüberlegungen
- fassen eine Folge von Anweisungen zu einer Einheit zusammen
- durch ein geeignetes Klammerpaar syntaktisch gekennzeichnet, etwa BEGIN_TRANS und END_TRANS

Transaktionen: semantische Eigenschaften

- Transaktionen *erhalten die semantischen Bedingungen*
- Transaktionen können *parallel* ablaufen
- parallel ablaufende Transaktionen sind im Wesentlichen voneinander *unabhängig* und beeinflussen einander allenfalls in *genau vorhersehbarer* Weise
- Transaktionen sind *unteilbar*, d.h. werden gar nicht oder vollständig wirksam
- wirksam gewordene Transaktionen verändern Daten *dauerhaft*

ACID- Eigenschaften

- *A*tomarität (atomicity)
  im Sinne von Unteilbarkeit
- *C*onsistenzbewahrung (consistency preservation),
  im Sinne von Erhaltung der semantischen Bedingungen
- *I*solation (isolation)
  im Sinne von Unabhängigkeit
- *D*auerhaftigkeit (durability)
  insbesondere im Sinne von Fehlertoleranz

Transaktionen erhalten Bedingungen

- ***Benutzer*:** sorgfältiger Entwurf der Transaktionen

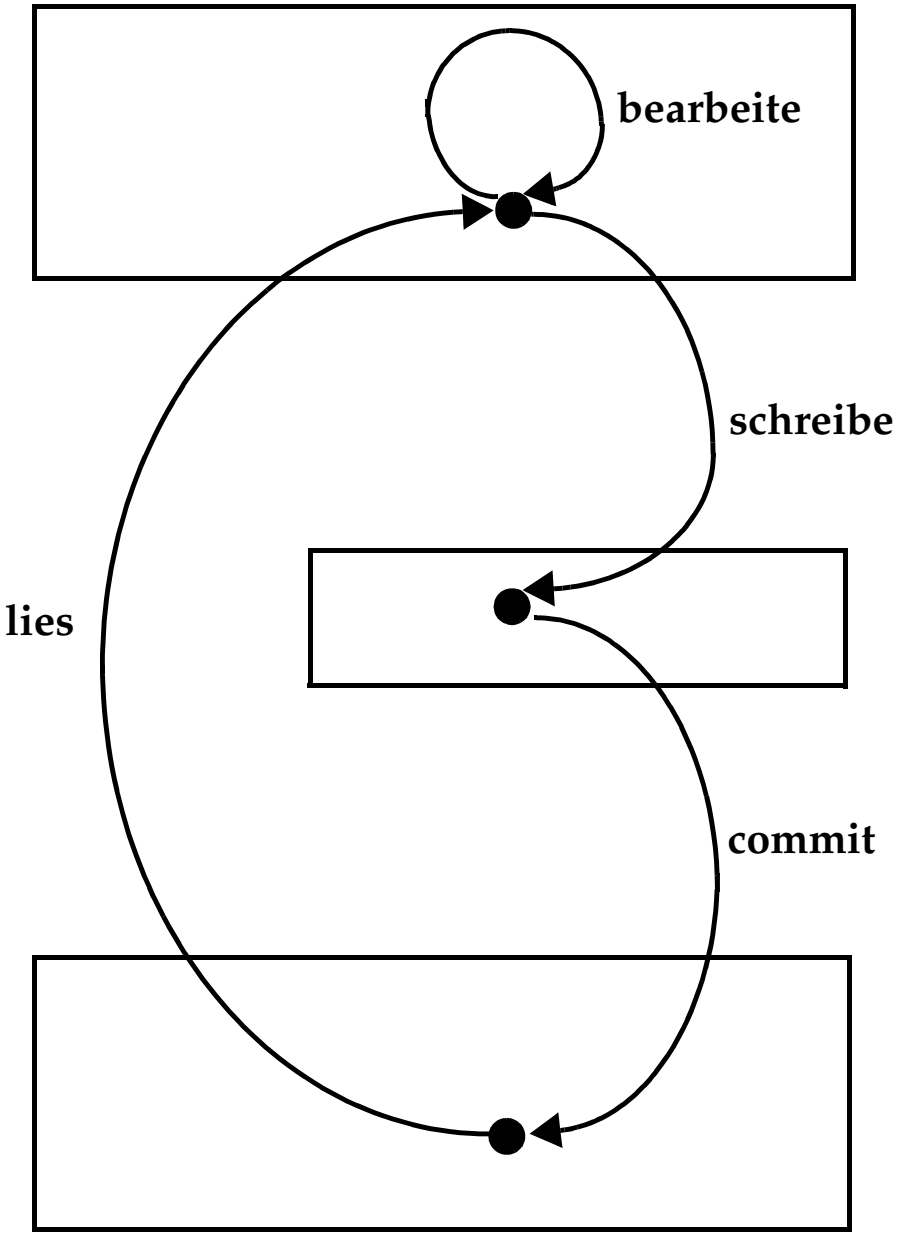
- ***Informationssystem*:** programmiersprachliche Konstrukte

  - **Abort_Trans -** Anweisung
    zum vorzeitigen Abbruch einer Transaktion:
    lässt gesamte Transaktion ***gar nicht wirksam*** werden

  - **Commit_Trans -** Anweisung:
    lässt Transaktion ***vollständig wirksam*** werden

  - Meldungen über
    den Erhalt bzw. die Verletzung von Bedingungen bei Änderungen,
    ggf. mit Liste der betroffenen Objekte

Datenebenen und Wirksamwerden: einfachstes Modell



lokale Daten

zwischenzeitliche Kopien

dauerhafte Daten

Transaktionen laufen parallel und voneinander unabhängig ab

- muss durch das Informationssystem selbst durchgesetzt werden

- von den Benutzern wird gegebenenfalls verlangt,
  dass ihre Transaktionen einen vorgeschriebenen Aufbau (*Protokolle*) einhalten

- der *Scheduler* hat dann die Aufgabe,
  die Anweisungen von parallel auszuführenden Transaktionen
  in einer geeigneten Reihenfolge anzuordnen

Randfall: beliebige Verschränkung

Scheduler mischt die Transaktionen *beliebig ineinander verschränkt*:

- einfache Verwirklichung:

  die **Anweisungen** in der Reihenfolge angeordnet belassen,
  wie sie angefordert werden

- Scheduler im Wesentlichen nur ***Warteschlange für Anweisungen***

- *hohe Parallelität*

- *große Unsicherheit* über die wechselseitige Beeinflussung der Transaktionen

Randfall: serielle Anordnung

Scheduler ordnet die Transaktionen *als Ganzes seriell* an:

- einfache Verwirklichung:

  die **Transaktionen** in der Reihenfolge angeordnet belassen,
  wie sie angefordert werden

- Scheduler im Wesentlichen nur ***Warteschlange für Transaktionen***

- im *Wesentlichen keine Parallelität*

- wechselseitige Beeinflussung der Transaktionen *genau vorhersehbar*

korrekte Reihenfolgen und Serialisierbarkeit

korrekte Reihenfolgen (geeignet) definieren,
um Vorteile der Randfälle möglichst gut gleichzeitig zu erreichen:

K1. [*sichere Vorhersehbarkeit*]
eine Reihenfolge, die durch
beliebige serielle Anordnung der Transaktionen ***als Ganzes*** entsteht,
sei korrekt

K2. [*erlaubte Parallelität*]
jede Reihenfolge, deren Auswirkung (unter den jeweiligen Annahmen)
ununterscheidbar
ist von der Auswirkung einer nach K1 korrekten Reihenfolge,
sei ebenfalls korrekt

in Praxis und Theorie:
verschiedene Annahmen und Auswirkungen führen
zu verschiedenen Begriffen von Korrektheit bzw. Serialisierbarkeit

Anweisungen, Transaktionen, Pläne

Menge von *Operatoren* (Aktionen): **A**

Menge von *Objekten*: **O**

Menge der *Anweisungen*: $\mathbf{S} := \mathbf{A} \times \mathbf{O}$

Menge der *Transaktionen*: $\mathbf{T} := \{ t \mid t : \{ 1, \ldots, n \} \rightarrow \mathbf{S}, t \text{ injektiv} \}$

für eine Menge von Transaktionen
$T = \{ t_1, \ldots, t_k \}$ mit $t_i = (t_{i.1}, \ldots, t_{i.ni})$

Plan (schedule) : $P : \{ 1, \ldots, n1+\ldots+nk \}$ ----> $\{ i.j \mid 1 \leq i \leq k, 1 \leq j \leq ni \}$
mit

- P bijektiv
- für alle i, für alle $j1, j2$:
  $j1 < j2 \Leftrightarrow P^{-1}(i.j1) < P^{-1}(i.j2)$

Schreibweisen: Transaktionen und ihre Komponenten

$t_i = (t_{i.1}, \ldots, t_{i.ni})$ $t_{i.j}$ ist j-te Anweisung von t_i

$= (Op_{i.1}\ o_{i.1}, \ldots, Op_{i.ni}\ o_{i.ni})$ $t_{i.j} = Op_{i.j}\ o_{i.j}$ mit $Op_{i.j} \in \mathbf{A}$, $o_{i.j} \in \mathbf{O}$

$= (1: Op_{i.1}\ o_{i.1}, \ldots, ni: Op_{i.ni}\ o_{i.ni})$

$= (i.1: Op_{i.1}\ o_{i.1}, \ldots, i.ni: Op_{i.ni}\ o_{i.ni})$

für $i.j$: $Op_{i.j}$ $o_{i.j}$ bezeichnet also:

i eine *Transaktion* t_i

j eine bezüglich der Transaktion t_i *lokale Anweisungsnummer*

$Op_{i.j}$ die auszuführende *Operation*

$o_{i.j}$ das betroffene *Objekt*

Schreibweisen: Pläne

$P = (p_1, \ldots, p_m)$ $\quad m = n1+\ldots+nk,$

$p_l := i_l.j_l$

$:= P(l)$

$= (t_{p\_1}, \ldots, t_{p\_m})$ $\quad t_{p\_l} = t_{i\_l.j\_l}$

$= Op_{i\_l.j\_l}\ o_{i\_l.j\_l}$

j_l-te Anweisung der i_l-ten Transaktion

	Transaktionsnr.		Anweisungsnr.		Operation	Objekt
$=$ (	i_1	.	j_1	:	$Op_{i\_1.j\_1}$	$o_{i\_1.j\_1}$
, … ,						
	i_m	.	j_m	:	$Op_{i\_m.j\_m}$	$o_{i\_m.j\_m}$)

Definitionsbereich von *P*: beschreibt die ***Zeitpunkte***, zu denen die Anweisungen ausgeführt werden

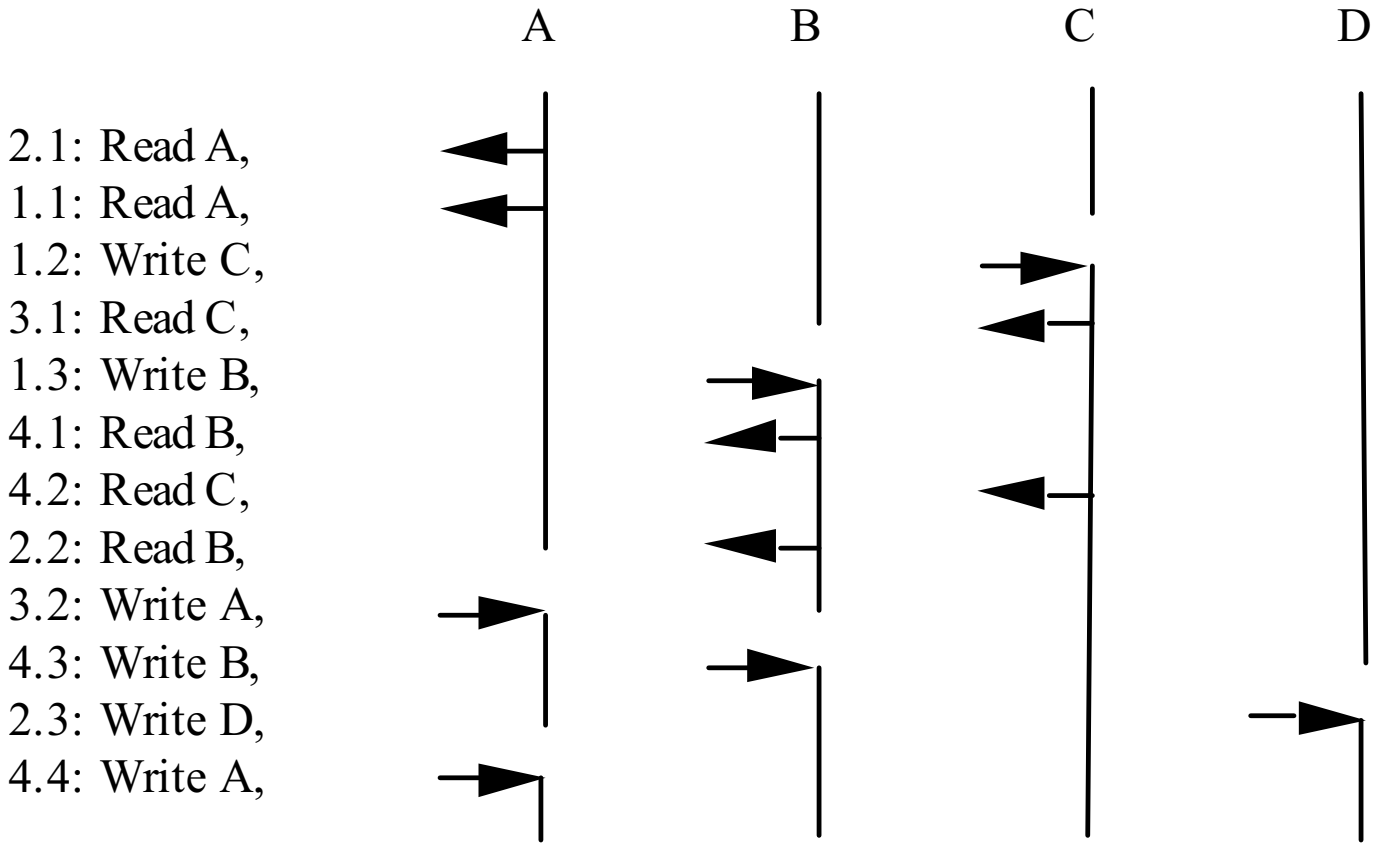
Plan *P* zu Transaktionen $T = \{ t_1, t_2, t_3, t_4 \}$ mit "Datenflüssen"

t_1 := (1.1: Read A,
1.2: Write C,
1.3: Write B)

t_2 := (2.1: Read A,
2.2: Read B,
2.3: Write D)

t_3 := (3.1: Read C,
3.2: Write A)

t_4 := (4.1: Read B,
4.2: Read C,
4.3: Write B,
4.4: Write A)



Read/Write-Modell

- betrachtet das Lesen und Schreiben von Daten als getrennte Aktionen
- nimmt an,
  dass im lokalen Speicher alle jemals gelesenen Werte gemerkt werden
- beruht auf einer Reihe von Annahmen

Read/Write-Modell: Annahme A1*

Im Operationsteil einer Anweisung seien nur die Operatoren *Read* und *Write* erlaubt.

beabsichtigte Semantik von *Read*:

die Daten aus dem im Operandenteil bezeichneten Objekt
werden in den lokalen Speicher des
die Transaktion ausführenden Prozesses gelesen

beabsichtigte Semantik von *Write*:

die Daten des im Operandenteil bezeichneten Objekts
werden überschrieben mit (möglicherweise) neuen Werten,
die sich aus der Verarbeitung aller vorher gelesenen Daten ergeben können

Read/Write-Modell: Annahme A2

Weitere Einzelheiten der Semantik, insbesondere über die Art der Verarbeitung seien nicht bekannt.

Read/Write-Modell: Annahme A3*

**Nachdem eine Read-Anweisung ausgeführt ist,
bleiben die gelesenen Daten zeitlich unbegrenzt und unverändert
im lokalen Speicher verfügbar;**

entsprechend liest eine Transaktion aus jedem Objekt höchstens einmal;

**ferner schreibe eine Transaktion in jedes Objekt höchstens einmal
(nach Abschluss aller Verarbeitungen);**

**wenn eine Transaktion ein Objekt sowohl liest als auch schreibt,
so gehe das Lesen dem Schreiben voran.**

Read/Write-Modell: Annahmen A4/A5/A6

A4.
Alle Objekte seien unstrukturiert und
stehen in keinerlei Beziehung zueinander.

A5.
Alle Transaktionen werden vollständig wirksam.

A6.
Alle Transaktionen liegen
als unverzweigte Anweisungsfolgen textuell als Ganzes vor.

Read/Write-Modell

A1*. Im Operationsteil einer Anweisung seien nur die Operatoren Read und Write erlaubt.

beabsichtigte Semantik von *Read*: die Daten aus dem im Operandenteil bezeichneten Objekt werden in den lokalen Speicher des die Transaktion ausführenden Prozesses gelesen

beabsichtigte Semantik von *Write*: die Daten des im Operandenteil bezeichneten Objekts werden überschrieben mit (möglicherweise) neuen Werten, die sich aus der Verarbeitung aller vorher gelesenen Daten ergeben können

A2. Weitere Einzelheiten der Semantik, insbesondere über die Art der Verarbeitung seien nicht bekannt.

A3*. Nachdem eine Read-Anweisung ausgeführt ist, bleiben die gelesenen Daten zeitlich unbegrenzt und unverändert im lokalen Speicher verfügbar;

entsprechend liest eine Transaktion aus jedem Objekt höchstens einmal;

ferner schreibe eine Transaktion in jedes Objekt höchstens einmal (nach Abschluss aller Verarbeitungen);

wenn eine Transaktion ein Objekt sowohl liest als auch schreibt, so gehe das Lesen dem Schreiben voran.

A4. Alle Objekte seien unstrukturiert und stehen in keinerlei Beziehung zueinander.

A5. Alle Transaktionen werden vollständig wirksam.

A6. Alle Transaktionen liegen als unverzweigte Anweisungsfolgen textuell als Ganzes vor.

13.2 Konflikte und Serialisierbarkeit

Konflikte: Definition

zwei Transaktionen liegen im *Konflikt*,

wenn sie ein Objekt *o* gemeinsam nutzen,

wobei mindestens eine der Transaktionen in *o* schreibt

- für eine Menge von Transaktionen T
  kann man alle Konflikte im obigen Sinne bestimmen

- für einen Plan P
  kann man jeweils die Reihenfolge
  der im Konflikt liegenden Transaktionen bestimmen

Aufgaben zur Konfliktbehandlung

gegeben: Plan P

gesucht: serieller Plan P',
der alle Konflikte genauso wie P behandelt

falls ein solcher Plan P' existiert,

wird der Plan P angesehen

als "*korrekt bezüglich der Behandlung von Konflikten*"

erzeuge solchermaßen "korrekte" Pläne

Konflikte: formale Definition

umgangssprachlich:

zwei Transaktionen liegen im *Konflikt*,

wenn sie ein Objekt *o* gemeinsam nutzen,

wobei mindestens eine der Transaktionen in *o* schreibt

formal:

Sei $T = \{ t_1 , ... , t_k \}$ eine Menge von Transaktionen.

Zwei das gleiche Objekt o nutzende Anweisungen aus T,

$$i1 . j1 : Op1\ o \quad \text{und} \quad i2 . j2 : Op2\ o$$

liegen (wegen *o*) *im Konflikt*

:gdw

$i1 \neq i2$ und

$Op1 = \text{Write}$ oder $Op2 = \text{Write}$.

Konflikt-Äquivalenz und Konflikt-Serialisierbarkeit: Definition

Seien P und P' Read/Write-Pläne zu einer Menge von Transaktionen T.
P* ist *Konflikt-äquivalent zu P' :gdw
je zwei im Konflikt liegende Anweisungen werden in P und P'
in der gleichen Reihenfolge ausgeführt.

Ein Read/Write-Plan P zu einer Menge von Transaktionen T ist
Konflikt-serialisierbar :gdw
es gibt einen Read/Write-Plan P', so dass gilt:

(K1) P' ist serieller Read/Write-Plan.

(K2) P und P' sind Konflikt-äquivalent.

Konfliktgraph: Definition

gegeben: Read/Write-Plan P zu einer
Menge von Transaktionen $T = \{ t_1, \ldots , t_k \}$

zu bestimmen: ist P ***Konflikt-serialisierbar*** ?

konstruiere: ***Konfliktgraphen*** $G_{konflikt}(P) = (E_{konflikt} , K_{konflikt})$ mit

Eckenmenge $E_{konflikt} := T$

Kantenmenge $K_{konflikt} := \{ (t_{i1}, t_{i2}) \mid t_{i1} \neq t_{i2}$, und

es gibt $o \in \mathbf{O}(T), j1, j2$:
$t_{i1.j1} = Op1\ o$ und $t_{i2.j2} = Op2\ o$,
$Op1 = \text{Write}$ oder $Op2 = \text{Write}$, und
$P^{-1}(i1.j1) < P^{-1}(i2.j2)$ }

Kante von Transaktion t_{i1} nach Transaktion t_{i2} :

t_{i1} enthält eine Anweisung, die im Konflikt liegt zu
einer in P folgenden, aus t_{i2} stammenden Anweisung

Test für Konflikt-Serialisierbarkeit: Satz

P	Read/Write-Plan zu
$T = \{ t_1, \ldots, t_k \}$	Menge von Transaktionen
$G_{konflikt}(P) = (T, K_{konflikt})$	zugehöriger Konfliktgraph

1. P ist ***Konflikt-serialisierbar***

   genau dann, wenn

   $G_{konflikt}(P)$ ist ***azyklisch***.

2. Wenn $G_{konflikt}(P)$ ***azyklisch*** ist
   und P' ein serieller Read/Write-Plan ist,
   der durch *topologisches Sortieren* aus $G_{konflikt}(P)$ gewonnen wird,

   dann sind P und P' ***Konflikt-äquivalent***.

Beispiel zum Test: Transaktionen und Konflikte

Transaktionen: $T = \{ t_1, t_2, t_3, t_4 \}$

t_1 := (1.1: Read A,
1.2: Write C,
1.3: Write B)

t_2 := (2.1: Read A,
2.2: Read B,
2.3: Write D)

t_3 := (3.1: Read C,
3.2: Write A)

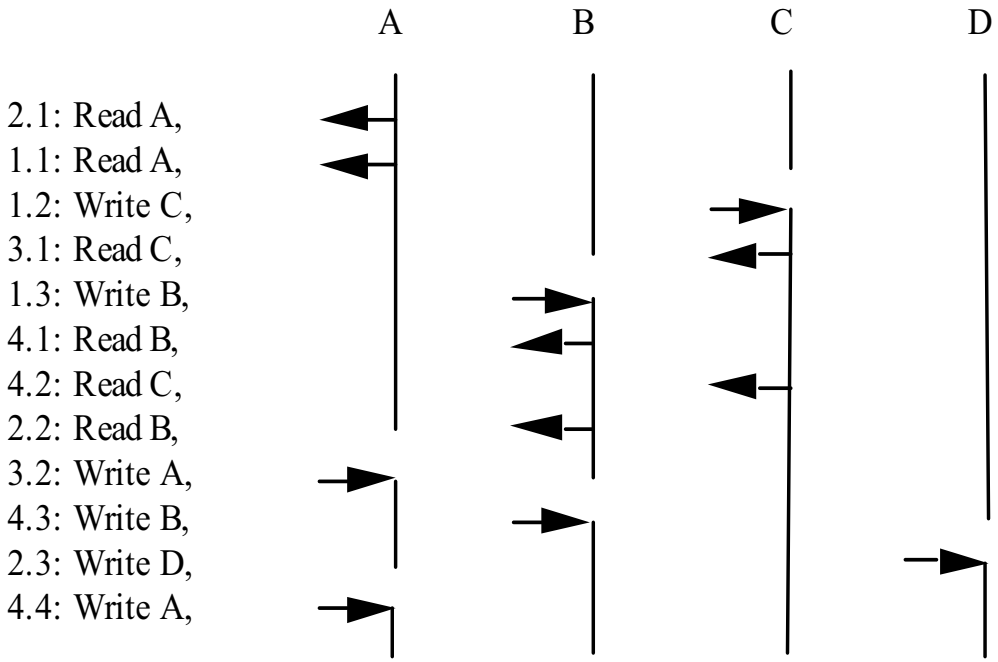
t_4 := (4.1: Read B,
4.2: Read C,
4.3: Write B,
4.4: Write A)

Konflikte in *T* :

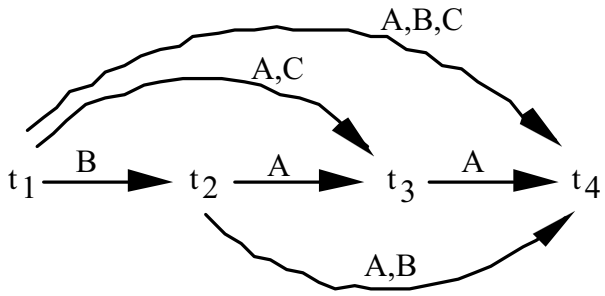
wegen Objekt A:	1.1: Read A	und	3.2: Write A
	1.1: Read A	und	4.4: Write A
	2.1: Read A	und	3.2: Write A
	2.1: Read A	und	4.4: Write A
	3.2: Write A	und	4.4: Write A
wegen Objekt B:	1.3: Write B	und	2.2: Read B
	1.3: Write B	und	4.1: Read B
	1.3: Write B	und	4.3: Write B
	2.2: Read B	und	4.3: Write B
wegen Objekt C:	1.2: Write C	und	3.1: Read C
	1.2: Write C	und	4.2: Read C
wegen Objekt D:	keine Konflikte		

Beispiel zum Test: Plan und Konfliktgraph

Plan *P*:



Konfliktgraph:



Plan *P* ist Konflikt-äquivalent zum seriellen Plan $P' = (t_1, t_2, t_3, t_4)$,
also **Konflikt-serialisierbar**

Satz über Test für Konflikt-Serialisierbarkeit: Beweis

1. P ist ***Konflikt-serialisierbar*** genau dann, wenn $G_{konflikt}(P)$ ***azyklisch*** ist.
2. Wenn $G_{konflikt}(P)$ ***azyklisch*** ist und P' ein serieller Read/Write-Plan ist, der durch *topologisches Sortieren* aus $G_{konflikt}(P)$ gewonnen wird, dann sind P und P' ***Konflikt-äquivalent***.

Beweis von "$\Leftarrow$" und 2.:

Voraussetzung: $G_{konflikt}(P)$ azyklisch ,
P' durch topologisches Sortieren aus $G_{konflikt}(P)$ gewonnen

betrachte: zwei im Konflikt liegende Anweisungen
$i1.j1 : Op_1\ o$ und $i2.j2 : Op_2\ o$
mit $P^{-1}(i1.j1) < P^{-1}(i2.j2)$

Def. Konfliktgraph: $(t_{i1}, t_{i2}) \in K_{konflikt}$

topologisches Sortieren: in P' wird t_{i1} vor t_{i2} ausgeführt, d.h.
$P'^{-1}(i1.j1) < P'^{-1}(i2.j2)$.

also: P und P' Konflikt-äquivalent

Satz über Test für Konflikt-Serialisierbarkeit: Beweis

1. P ist **Konflikt-serialisierbar** genau dann, wenn $G_{konflikt}(P)$ *azyklisch* ist.

Beweis von "⇒" (in Kontraposition):

Voraussetzung: $G_{konflikt}(P)$ zyklisch, d.h.
$G_{konflikt}(P)$ enthält einen Zykel $(t_{i1}, \ldots, t_{ie}, t_{i1})$

indirekte Annahme: P' sei ein serieller, zu P Konflikt-äquivalenter Plan.

Konflikt-äquivalent: in P' müssen die am Zykel beteiligten Transaktionen in der durch den Zykel gegebenen Reihenfolge durchgeführt werden

damit speziell: in P' wird t_{i1} vor sich selbst ausgeführt,
ein offensichtlicher Widerspruch

14 Scheduler und Versionsverwaltung

14.1 Versionsverwaltung

Abbruch, Wirksamwerden und Versionen

- eine Transaktion soll

  gar nicht
  oder vollständig
  wirksam werden

- bislang gemäß Annahme A5:

  alle Transaktionen werden *vollständig wirksam*

- nunmehr auch zulassen:

  Transaktionen werden *gar nicht wirksam*

Anlässe für Nicht-Wirksamwerden

- durch Benutzer mit **Transaktions-Anweisung *Abort_Trans***,
  um semantische Bedingungen zu erhalten

- durch Scheduler mit **Abbruch einer Transaktion**
  (und späterem erneuten Start),
  um Serialisierbarkeit (oder andere Systemeigenschaften) zu erreichen

- durch Betriebssystem mit **Abbruch des Prozesses**,
  um Fehlersituation aufzulösen

- durch Missgeschick als **Zusammenbruch des Rechensystems**,
  z.B. Stromausfall oder nicht behebbarer Plattenfehler

- ...

absichtlicher Abbruch einer Transaktion

- alle vorher durchlaufenen

  Schreibanweisungen der Transaktion müssen unwirksam werden:

  - ***nicht beobachtbar werden***
    für *zukünftige* Leseanweisungen anderer Transaktionen

  - ***nicht bedeutsam bleiben***
    für *vorangegangene* Leseanweisungen anderer Transaktionen,

absichtlicher Abbruch: zukünftige Leseanweisungen

Forderung:
alle vorher durchlaufenen Schreibanweisungen der Transaktion
werden nicht beobachtbar
für *zukünftige* Leseanweisungen anderer Transaktionen

Maßnahme:
nach Schreibanweisungen müssen zunächst
zwei ***Versionen*** eines Objekts unterhalten werden:

- die ***alte Version***,
  die (spätestens) nach einem Abbruch der Transaktion
  (wieder) als die gültige anzusehen ist

- die ***neue Version***,
  die (spätestens) nach dem Wirksamwerden der Transaktion
  gültig werden muss

absichtlicher Abbruch: vorangegangene Leseanweisungen

Forderung:
alle vorher durchlaufenen Schreibanweisungen der Transaktion
bleiben nicht bedeutsam
für *vorangegangene* Leseanweisungen anderer Transaktionen

Maßnahme:

- andere Transaktionen,
  die schon die neue Version (fälschlicherweise) als gültig angesehen haben
  (“schmutzige Daten” gelesen haben),
  müssen **ebenfalls** abgebrochen werden

- solche Folgeabbrüche können auch *kaskadenhaft* auftreten

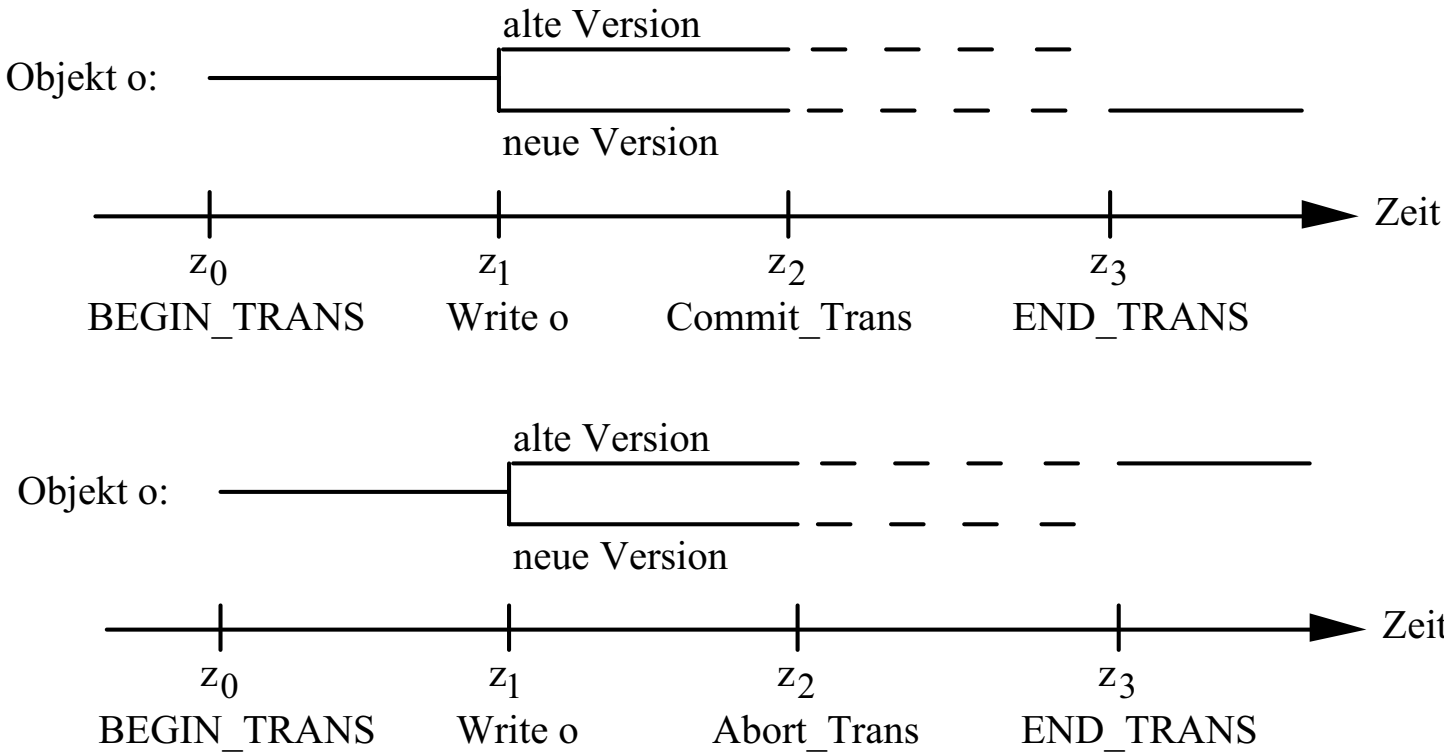
Versionsverwaltung

legt genau fest,

- welche Version
- zu welchem Zeitpunkt
- für welche Transaktionen

gültig sein soll

Versionsverwaltung: einige Anforderungen



- zwischen Zeitpunkten z_0 und z_2 (einschließlich): absichtliche Abbrüche erlauben
- zwischen Zeitpunkten z_1 und z_2: regeln,
  welche Version vom Objekt *o* unter welchen Umständen als gültig angesehen wird
- zwischen den (möglichst kurz aufeinanderfolgenden) Zeitpunkten z_2 und z_3:
  Gültigkeit abschließend und für alle Transaktionen verbindlich regeln,
  dabei Objekt für andere Transaktionen vollständig sperren

Versionsfunktion zu einem Read/Write-Plan:

- **bestimmt für eine *Leseanweisung*:**
  zu lesende Version

- **bestimmt für eine *Schreibanweisung*:**
  - ob eine *alte Version* überschrieben werden soll, und wenn ja, welche
  - oder ob eine *neue Version* angelegt werden soll
  - oder ob überhaupt nichts getan werden soll

14.2 Scheduler

Scheduler

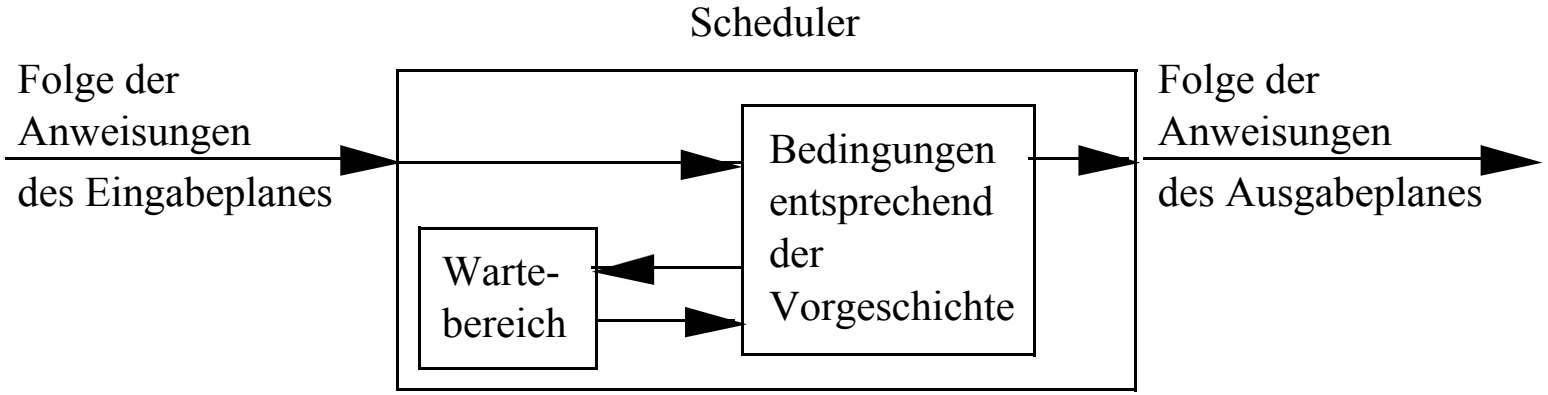
- ordnet für ***parallel auszuführende Transaktionen*** die Anweisungen in einer ***geeigneten Reihenfolge*** an

- bestimmt die zu benutzenden **Versionen**

- soll insbesondere erreichen:
  - ***Serialisierbarkeit***
  - ***Zuverlässigkeit***

im Folgenden nur unter ***vereinfachenden Annahmen***:
- alle Transaktionen werden wirksam (sofern nicht vom Scheduler abgebrochen)
- nur Konflikt-Serialisierbarkeit
- keine Versionen

vereinfachtes Modell eines Schedulers



- ***Transformation*:**
  ordnet jedem ***(Eingabe-)Plan*** zu einer Menge von Transaktionen *T*
  einen ***(Ausgabe-)Plan*** zu *T zu,*
  der ***Konflikt-serialisierbar*** ist

- ***Eingabeplan*:** gemäß dem zeitlich geordneten Eintreffen der Anweisungen,
  wie von Transaktions-Prozessen angefordert

- ***Ausgabeplan*:** eine angeforderte Anweisung gegebenenfalls **verzögern,**
  d.h. zunächst in einen ***Wartebereich*** einfügen und
  erst nach dem Eintreffen bestimmter Bedingungen ausführen

Konfliktgraphen-Scheduler

- einfacher Ansatz:
  - baut den ***Konfliktgraphen*** dynamisch auf
  - prüft jeweils auf ***Zykelfreiheit***
  - bricht gegebenenfalls eine Zykel-erzeugende Transaktion ab

- lässt genau die ***Konflikt-serialisierbaren Pläne unverändert*** durch

- leider ***praktisch kaum brauchbar***

Sperrprotokoll-Scheduler: Sperranweisungen

- Transaktionen müssen solche Objekte,
  aus denen sie lesen oder in die sie schreiben wollen,
  vorher ***ausdrücklich sperren*** (lock) und
  nachher wieder ***ausdrücklich freigeben*** (unlock)

- Transaktionen müssen ***Sperrprotokoll*** genügen,
  d.h. aus folgenden Anweisungen aufgebaut werden

Read o	:	***lies*** aus Objekt *o*
Write o	:	***schreibe*** in Objekt *o*
RLock o	:	***sperre*** Objekt *o* ***zum Lesen***
WLock o	:	***sperre*** Objekt *o* ***zum Schreiben*** (und zum Lesen)
Unlock o	:	***gib*** Objekt *o* ***frei***

Sperrprotokoll-Scheduler: ergänztes Read/Write-Modell

- jeder ***Leseanweisung*** *Read o* muss
  - vorausgehen: genau eine Anweisung der Form ***RLock o*** oder ***WLock o***
  - folgen: genau eine Anweisung der Form ***Unlock o***
- jeder ***Schreibanweisung*** *Write o* muss
  - vorausgehen: genau eine Anweisung der Form ***WLock o***
  - folgen: genau eine Anweisung der Form ***Unlock o***
- jedes Lock-Unlock-Paar
  umklammere eine entsprechende Lese- oder Schreiboperation

beabsichtigte Semantik von Lesesperren

wird eine ***Sperranweisung***
i*1.*j*1 : *RLock o
erfolgreich ausgeführt,

so hält die Transaktion t_{i1} eine ***Lesesperre*** auf dem Objekt *o*,

bis die zugehörige ***Freigabeanweisung***
i*1.*j*2 : *Unlock o
ausgeführt wird:

dazwischen

- kann t_{i1} aus *o* **lesen**, und
- *andere Transaktionen* können ebenfalls Lesesperren auf *o*, aber *keine Schreibsperren* auf *o* erhalten

beabsichtigte Semantik von Schreibsperren

wird eine ***Sperranweisung***
i*1.*j*1 : *WLock o
erfolgreich ausgeführt,

so hält die Transaktion t_{i1} eine ***Schreibsperre*** auf dem Objekt *o*,

bis die zugehörige ***Freigabeanweisung***
i*1.*j*2 : *Unlock o
ausgeführt wird:

dazwischen

- kann t_{i1} in *o* **schreiben** oder
  erst aus *o* **lesen** und dann in *o* **schreiben**, und
- *andere Transaktionen* können
  weder eine Lese- noch eine Schreibsperre auf *o* erhalten

Verträglichkeiten

- Lesesperren können ***geteilt*** werden (shared locks)
- Schreibsperren werden ***exklusiv*** gehalten (exclusive locks)
- ***Konflikte***

  zwischen Lese- und Schreibanweisungen verschiedener Transaktionen entsprechen genau
  den ***Unverträglichkeiten*** zwischen den zugehörigen Sperranweisungen

- ***Verträglichkeitsmatrix***:

		von einer anderen Transaktion bereits gehaltene Sperre auf o:	
		RLock	WLock
von t_{i1} für o angeforderte Sperre:	RLock	+	-
	WLock	-	-

Sperrprotokoll-Scheduler: Verfahren

beachtet nur die ***Sperr- und Freigabeanweisungen***:

- wenn eine ***Sperranweisung*** neu angefordert wird,
  prüfen, ob sie mit den schon gehaltenen Sperren verträglich ist:
  - falls ***verträglich***:
    die Sperranweisung ausführen (verlangte Sperre vergeben)
  - falls ***unverträglich***:
    die Sperranweisung verzögern und in den Wartebereich einfügen
- wenn eine ***Freigabeanweisung*** neu angefordert wird, so
  - die entsprechende Sperre aufheben
  - gegebenenfalls
    auf diese Freigabe wartende Sperranweisungen erneut bearbeiten

Sperrprotokoll-Scheduler: Zwei-Phasen-Sperrprotokoll

ein Sperrprotokoll-Scheduler bearbeitet solche Transaktionen erfolgreich,

die dem ***Zwei-Phasen-Sperrprotokoll*** genügen, d.h.:

alle Sperranweisungen liegen

vor allen Freigabeanweisungen

Konflikt-Serialisierbarkeit unter Zwei-Phasen-Sperrprotokoll: Satz

P Plan zu

$T = \{ t_1, \ldots, t_k \}$ Menge von Transaktionen mit:

a. [Zwei-Phasen-Sperrprotokoll]
alle Transaktionen aus T genügen dem ***Zwei-Phasen-Sperrprotokoll***

b. [Sperrprotokoll-Scheduler]
im Plan P halten je zwei Transaktionen ***keine unverträglichen Sperren***
(d.h. P ist möglicher ***Ausgabeplan des Sperrprotokoll-Schedulers***)

1. Für $i = 1, \ldots, k$ sei die ***Sperrzeit*** z_i derjenige Zeitpunkt von P,
zu dem die Transaktion t_i ihre letzte Sperranweisung ausführt,
und $\pi : \{1, \ldots, k\} \rightarrow \{1, \ldots, k\}$ sei Permutation mit
$z_{\pi(1)} < z_{\pi(2)} < \ldots < z_{\pi(k)}$:

P ist ***Konflikt-äquivalent zum seriellen Plan*** $P' = (t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(k)})$.

2. P ist ***Konflikt-serialisierbar***.

(jeweils nach dem Weglassen der Sperr- und Freigabeanweisungen)

Konflikt-Serialisierbarkeit unter Zwei-Phasen-Sperrprotokoll: Beweis

zu zeigen: je zwei im Konflikt liegende Anweisungen werden in P und P' in der gleichen Reihenfolge ausgeführt

betrachte solche: Zeitpunkte $w_1 < w_2$

Anweisungen $P(w_1) = i1.j1 : Op_1\ o$

$P(w_2) = i2.j2 : Op_2\ o$

mit $Op_1 = Write$ oder $Op_2 = Write$

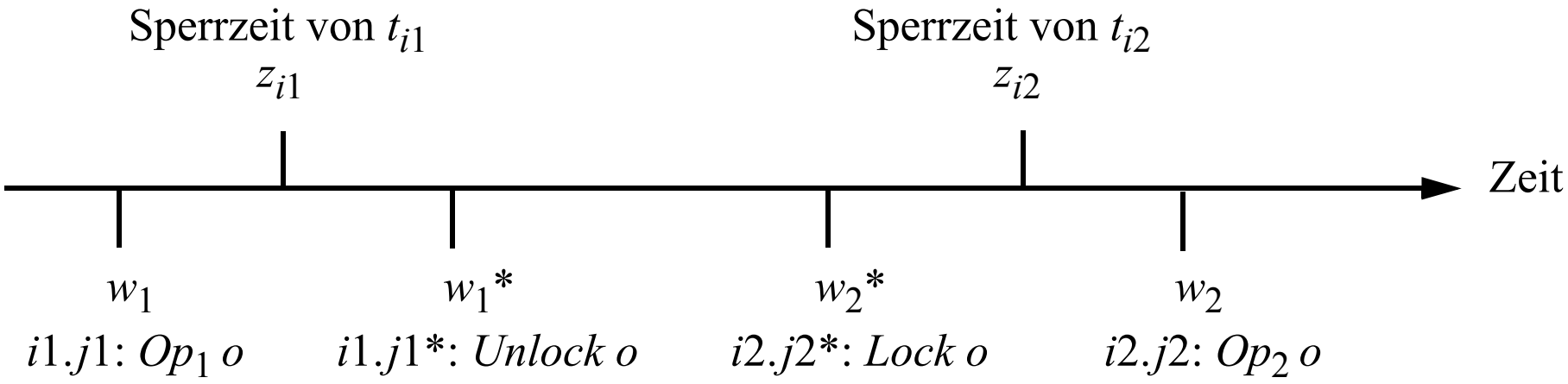
exklusive Sperre: t_{i1} muss *nach* dem Zeitpunkt w_1,
etwa zur Zeit w_1*, ein *Unlock o* ausführen,
bevor t_{i2} *vor* dem Zeitpunkt w_2,
etwa zur Zeit w_2*, ein *WLock o* bzw. *RLock o* ausführt

Zwei-Phasen-Sperrprotokoll: alle Sperranweisungen vor allen Freigabeanweisungen

also: $z_{i1} < w_1^* < w_2^* \leq z_{i2}$

also: im seriellen Plan P' werden betrachtete Anweisungen in der gleichen Reihenfolge ausgeführt

Konflikt-Serialisierbarkeit unter Zwei-Phasen-Sperrprotokoll: Veranschaulichung



Sperrprotokoll-Scheduler verlangt Zwei-Phasen-Sperrprotokoll

Die Transaktion t_1 genüge einem ***Sperrprotokoll***,
entspreche aber ***nicht dem Zwei-Phasen-Aufbau***.

Dann gibt es eine Transaktion t_2
und für $T := \{ t_1 , t_2 \}$ einen Plan P,
so dass gilt:

1. [Sperrprotokoll-Scheduler]
   t_1 und t_2 halten keine unverträglichen Sperren.

2. P ist ***nicht*** Konflikt-serialisierbar.

Beweis: entfällt

Verklemmungen

- Transaktionen geraten in eine ***Verklemmung*** (deadlock),
  wenn sie wechselseitig
  schon von jeweils einer anderen Transaktion gehaltene Sperren anfordern

- ein **typisches Muster**:

angeforderte Anweisung	**Verhalten des Schedulers**
1.1 *: WLock A*	t_1 erhält Schreibsperre auf *A*
2.1 : *WLock B*	t_2 erhält Schreibsperre auf *B*
1.2 : *WLock B*	Anforderung ist unverträglich: t_1 muss warten auf 2.*j*2 : *Unlock B*
2.2 : *WLock* A	Anforderung ist unverträglich: t_2 muss warten auf 1.*j*1 : *Unlock A*

Verklemmung: t_1 wartet auf t_2, und t_2 wartet auf t_1

Verklemmungen: Entdecken/Auflösen oder Verhindern

- Verklemmungen ***entdecken*** und
  durch Abbruch einer Transaktion ***auflösen***

- Verklemmungen durch geeignete Maßnahmen ***verhindern***

Striktes Sperrverhalten

- **Phase 1**:[*Sperren*]

  zusammen mit Phase 4: Zwei-Phasen-Sperrprotokoll

  alle benötigten Objekte sperren

  (falls als unteilbare Operation verwirklicht,
  werden sogar Verklemmungen vermieden;
  andernfalls können Phase 1 und 2 auch ineinander verschränkt werden)

- **Phase 2**:[*Schreiben in eine Logdatei*]

  zusammen mit Phase 3: für Zuverlässigkeit

  zunächst alle Schreiboperationen in gesonderter *Logdatei* durchführen

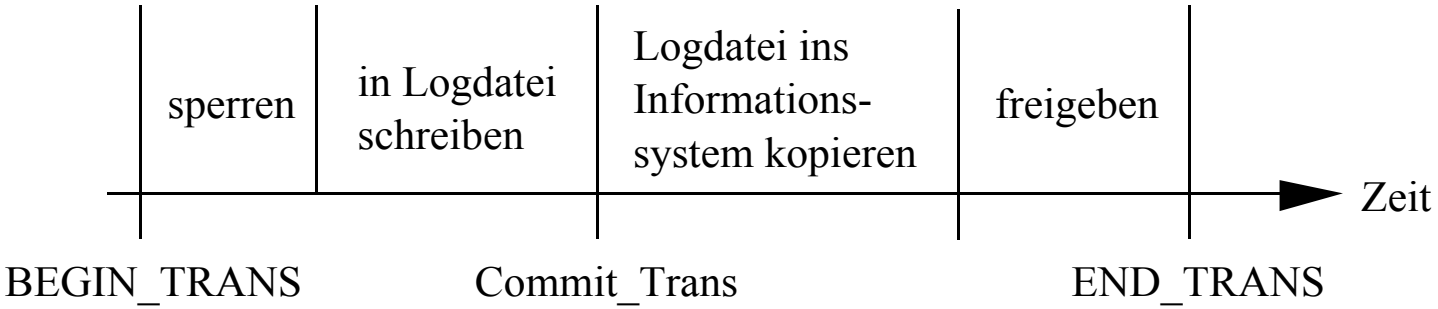
  (*nicht* im gemeinsamen Informationssystem;
  Logdatei enthält für andere Transaktionen unzugängliche Versionen)

- **Phase 3**:[*Kopieren der Logdatei in das Informationssystem*]
  erreicht die Transaktion die Commit_Trans-Anweisung, dann

  Logdatei in gemeinsames Informationssystem kopieren

- **Phase 4**:[*Freigeben*]
  alle gesperrten Objekte wieder freigegeben

Veranschaulichung:



Zeitmarken-Scheduler

- ***Transaktion*** t_i bei ihrem Start ***(statische) Zeitmarke*** zuordnen:

  s_i := **Startzeit von** t_i

- für ***Objekt*** *o* zwei ***(dynamische) Zeitmarken*** unterhalten:

  o_{read} := **max** { s_i | t_i **hat aus Objekt** *o* **gelesen** }

  o_{write} := **max** { s_i | t_i **hat in Objekt** *o* **geschrieben** }

- Zeitmarken beobachten und verändern mit ***Ziel***:
  nur solche Ausgabe-Pläne zulassen,
  die ***(Konflikt-)äquivalent*** zu demjenigen ***seriellen Plan*** sind,
  der Transaktionen in der ***Reihenfolge ihrer (statischen) Zeitmarken*** ausführt

- wenn ein sichtbar werdender Konflikt dieses Ziel gefährdet,
  dann eine am Konflikt beteiligte Transaktion abbrechen

Zeitmarken-Scheduler: Leseanforderung

wenn Transaktion t_i eine ***Leseanweisung*** ***Read o*** anfordert,

dann die Zeitmarke s_i vergleichen mit den Zeitmarken der Transaktionen,
die vorangehende,
im ***Konflikt liegende Schreibanweisungen*** enthalten:

- falls $o_{write} < s_i$: Leseanweisung ausführen;

  $o_{read} := \max \{ o_{read}, s_i \}$

- falls $o_{write} > s_i$: Transaktion t_i abbrechen

Zeitmarken-Scheduler: Schreibanforderung

wenn Transaktion t_i eine ***Schreibanweisung*** ***Write o*** anfordert,

dann die Zeitmarke s_i vergleichen mit den Zeitmarken der Transaktionen, die vorangehende,
im Konflikt liegende Lese- oder Schreibanweisungen enthalten:

- falls $o_{read} \leq s_i$
  und $o_{write} < s_i$: Schreibanweisung ausführen;

  $o_{write} := s_i$

- falls $o_{read} > s_i$
  oder $o_{write} > s_i$: Transaktion t_i abbrechen

Beispiel für Zeitmarken-Scheduler

Transaktionen:

t_1 := (1.1: Read A,
1.2: Write C,
1.3: Write B)

t_2 := (2.1: Read A,
2.2: Read B,
2.3: Write D)

t_3 := (3.1: Read C,
3.2: Write A)

t_4 := (4.1: Read B,
4.2: Read C,
4.3: Write B,
4.4: Write A)

Zeitmarken, entsprechend der Startzeiten zugeordnet:

$s_1 := 1$, $s_2 := 2$, $s_3 := 4$, $s_4 := 6$

Plan *P*:

$P(1)$	=	1.1:	Read A
$P(2)$	=	2.1:	Read A
$P(3)$	=	1.2:	Write C
$P(4)$	=	3.1:	Read C
$P(5)$	=	1.3:	Write B
$P(6)$	=	4.1:	Read B
$P(7)$	=	4.2:	Read C
$P(8)$	=	2.2:	Read B
$P(9)$	=	3.2:	Write A
$P(10)$	=	4.3:	Write B
$P(11)$	=	2.3:	Write D
$P(12)$	=	4.4:	Write A

P	z	A_r	A_w	B_r	B_w	C_r	C_w	D_r	D_w	erfüllte Bedingung
	0	0	0	0	0	0	0	0	0	
1.1: Read A	1	1	.	.	.	.	.	.	.	$A_w < s_1$
2.1: Read A	2	2	.	.	.	.	.	.	.	$A_w < s_2$
1.2: Write C	3	.	.	.	.	.	1	.	.	$C_r <= s_1 \ \wedge C_w < s_1$
3.1: Read C	4	.	.	.	.	4	.	.	.	$C_w < s_3$
1.3: Write B	5	.	.	.	1	.	.	.	.	$B_r <= s_1 \ \wedge B_w < s_1$
4.1: Read B	6	.	.	6	.	.	.	.	.	$B_w < s_4$
4.2: Read C	7	.	.	.	.	6	.	.	.	$C_w < s_4$
2.2: Read B	8	.	.	6	.	.	.	.	.	$B_w < s_2$
3.2: Write A	9	.	4	.	.	.	.	.	.	$A_r <= s_3 \ \wedge A_w < s_3$
4.3: Write B	10	.	.	.	6	.	.	.	.	$B_r <= s_4 \ \wedge B_w < s_4$
2.3: Write D	11	.	.	.	.	.	.	.	2	$D_r <= s_2 \ \wedge D_w < s_2$
4.4: Write A	12	.	6	.	.	.	.	.	.	$A_r <= s_4 \ \wedge A_w < s_4$

würde man im Plan P
die ersten beiden Anweisungen vertauschen,

so erhielte Transaktion t_1 die Startzeit $s_1 := 2$,
Transaktion t_2 die Startzeit $s_2 := 1$

dieser Plan wäre ***nicht Konflikt-äquivalent***
zum seriellen Plan
$P'' = (t_2 , t_1 , t_3 , t_4)$:

die Anforderung der Leseanweisung 2.2 : *Read B*
würde zum Zeitpunkt 8
zum Abbruch der Transaktion t_2 führen,

weil die Bedingung $B_w < s_2$
für $B_w = 2$ und $s_2 = 1$
nicht erfüllt wäre

Konflikt-Serialisierbarkeit unter Zeitmarken-Scheduler: Satz

P möglicher Ausgabeplan des Zeitmarken-Schedulers zu
$T = \{ t_1, \ldots, t_k \}$ Menge von Transaktionen

1. Für $i = 1, \ldots, k$ sei s_i die Startzeit von t_i und
   $\pi : \{ 1, \ldots, k \} \rightarrow \{ 1, \ldots, k \}$ Permutation mit
   $s_{\pi(1)} < s_{\pi(2)} < \ldots < s_{\pi(k)}$.

   P ist ***Konflikt-äquivalent zum seriellen Plan*** $P' = (t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(k)})$.

2. P ist ***Konflikt-serialisierbar***.

zu zeigen für 1.: je zwei ***im Konflikt liegende*** Anweisungen werden in P und P' ***in der gleichen Reihenfolge*** ausgeführt

Zeitmarken-Scheduler stellt genau dies sicher!

Beweis von 2.: folgt unmittelbar aus 1.

nutzlose Schreibanweisungen

durch Verfeinerung der Regeln kann man
im Konflikt liegende, aber ***erkennbar nutzlose*** Schreibanweisungen
einfach überspringen:

wenn Transaktion t_i eine ***Schreibanweisung*** ***Write o*** anfordert,

dann die Zeitmarke s_i vergleichen mit den Zeitmarken der Transaktionen,
die vorangehende,
im Konflikt liegende Lese- oder Schreibanweisungen enthalten:

- falls $o_{read} \leq s_i$
  und $o_{write} < s_i$: Schreibanweisung ausführen;
  $o_{write} := s_i$

- falls $o_{read} > s_i$: Transaktion t_i abbrechen

- falls $o_{read} \leq s_i$
  und $o_{write} > s_i$: Schreibanweisung einfach überspringen

Unvergleichbarkeit von Zwei-Phasen-Sperrprotokoll und Zeitmarken-Scheduler

- Zwei-Phasen-Sperrprotokoll:

  Konflikt-Äquivalenz zu demjenigen seriellen Plan,
  der die Transaktionen in der Reihenfolge ihrer ***Sperrzeiten*** ausführt

- Zeitmarken-Scheduler:

  Konflikt-Äquivalenz zu demjenigen seriellen Plan,
  der die Transaktionen in der Reihenfolge ihrer ***Startzeiten*** ausführt

- beide Verfahren schließen manche Konflikt-serialisierbaren Pläne aus

- die Klassen der jeweils erzeugbaren Ausgabepläne

  sind (bezüglich Mengeninklusion) ***unvergleichbar***

pessimistische und optimistische Scheduler

- ***pessimistischer* Scheduler:**

  durch ***weitreichende Vorsorge***
  nichtserialisierbare oder verklemmte Situationen
  möglichst ***frühzeitig verhindern***;

  im Allgemeinen erforderlich:
  - verhältnismäßig großer Aufwand für den Scheduler oder
  - verhältnismäßig einschneidende Protokolle
    (Einschränkungen an die erlaubten Transaktionen)

- ***optimistischer* Scheduler:**

  zunächst die Anweisungen möglichst unmittelbar wie angefordert führen;

  erst ***nachträglich*** nichtserialisierbare oder verklemmte Situationen
  durch ***Abbruch*** einer Transaktion wieder auflösen

15 Sicherheit

15.1 Zusammenarbeit unter Bedrohung

mehrseitige, Werte-geleitete Sicherheit

- ***Sicherheit*:** umfassende Eigenschaft
  eines in Umgebung eingebetteten Systems

- ***Sicherheitseigenschaften*:** berücksichtigen die *Interessen* aller Beteiligten,
  wägen *Konflikte* angemessen ab

- **zugrunde liegende *Werte*:** *Unversehrtheit* und *Selbstbestimmung*,
  Wahrung von *Geheimnissen*,
  Schutz von *Eigentum*,
  Erhaltung des *Lebensraums*,
  gesellschaftliche *Mitwirkung* und *Mitbestimmung,*
  staatliche Durchsetzung des *Rechts,*
  ...

- ***Anforderungen***: gewünschte *Dienste* und
  Abwehr von *Bedrohungen*

Interessen/Ziele und Maßnahmen

- ***Interessen/Ziele*:**
  - ***Verfügbarkeit*** von Daten und Diensten
  - ***Vertraulichkeit*** von Informationen und Handlungen
  - ***Integrität*** - inhaltliches *Zutreffen* von Daten;
    - *unveränderter Zustand* von Daten, Programmen und Prozessen
  - ***Authentizität*** von Handelnden
  - ***Anerkennung*** von Handlungen vor Dritten
  - ...

- ***Maßnahmen*:**
  - Verletzungen vorbeugend ***verhindern***
  - dennoch eingetretene Schäden ***begrenzen***
  - deren Auswirkungen ***ausgleichen***

- ***Bewertung*:** - Grad der *Erfüllung* der Interessen/Ziele durch Maßnahmen
  - zugrunde liegende *Annahmen,* zugesprochenes *Vertrauen*
  - *Aufwand* entsprechend erkannten *Risiken*

Informationsgesellschaft

- umfassende *Informationsgesellschaft*:
  - alle einzelnen Bürger, IT aktiv nutzend oder von IT passiv betroffen
  - alle gemeinschaftlichen Einrichtungen
  - alle privaten Organisationen
  - ...
- *technologische Grundlagen*:
  - öffentliche und private *Telekommunikationsdienste*
  - rechnergestützte *Netzwerke* von Rechensystemen aller Art
  - von persönlichen häuslichen oder tragbaren Rechnern
    über Arbeitsplatzsysteme mit im Hintergrund arbeitenden
    lokalen oder spezialisierten globalen Dienstsystemen
    bis zu mächtigen Zentralrechnern
- Nutzung für alle erdenklichen *Aufgaben*:
  - Information verarbeiten und verwalten
  - informationelle Dienste anbieten oder anfordern
  - informationelle Zusammenarbeit ermöglichen und durchführen

15.2 Informationelle Selbstbestimmung

Grundrechte

- **Schutz des Einzelnen:**

GG 1(1): Die *Würde des Menschen* ist unantastbar.
Sie zu achten und zu schützen ist Verpflichtung aller staatlichen Gewalt.

GG 2(1): Jeder hat das Recht auf die *freie Entfaltung seiner Persönlichkeit*, soweit er nicht die Rechte anderer verletzt und nicht ...

GG 20(3): Die Gesetzgebung ist an die verfassungsmäßige Ordnung, die vollziehende Gewalt und die Rechtsprechung sind an Gesetz und Recht gebunden.

- **Schutz von Gruppen:**

GG 9(1): Alle Deutschen haben des Recht, Vereine und Gesellschaften zu ...

- **Anspruch auf (öffentliche) Daten:**

GG 20(2): Alle Staatsgewalt geht vom Volke aus. ...

- **Informationsfreiheit:**

GG 5(1): Jeder hat das Recht, seine Meinung in Wort, Schrift und Bild *frei zu äußern und zu verbreiten* und sich aus allgemein zugänglichen Quellen *ungehindert zu unterrichten*. ...

Leitsätze zum “Volkszählungsurteil”, 1983

(1) Unter den Bedingungen der modernen Datenverarbeitung
wird der Schutz des Einzelnen gegen
    unbegrenzte Erhebung,
    Speicherung,
    Verwendung und Weitergabe
seiner persönlichen Daten von dem *allgemeinen Persönlichkeitsrecht*
des Art. 2 Abs. 1 in Verbindung mit Art. 1 Abs. 1 GG umfaßt.
Das Grundrecht gewährleistet insoweit die Befugnis des Einzelnen,
    grundsätzlich selbst über
    die Preisgabe und Verwendung seiner persönlichen Daten zu entscheiden.

(2) Einschränkungen dieses Rechts auf “*informationelle Selbstbestimmung*”
sind nur im überwiegenden Allgemeininteresse zulässig.
Sie bedürfen einer verfassungsmäßigen Grundlage,
die dem rechtsstaatlichen Gebot der Normenklarheit entsprechen muß.
Bei seinen Regelungen hat der Gesetzgeber ferner
den Grundsatz der Verhältnismäßigkeit zu beachten.
Auch hat er organisatorische und verfahrensrechtliche Vorkehrungen zu treffen,
welche der Gefahr einer Verletzung des Persönlichkeitsrechts entgegenwirken.

S.D. Warren und L.D. Brandeis: “The Right of Privacy”, 1890

“Später erkannte man
die geistige Natur des Menschen,
seine Gefühle und seinen Intellekt.

Schrittweise weitete sich der Rahmen dieser Rechte
[gegen physische Beeinträchtigungen von Leben und Eigentum],
und jetzt wird das Recht auf Leben als das Recht verstanden,

sich des Lebens zu freuen
– das Recht in Ruhe gelassen zu werden;

das Recht auf Freiheit sichert
die Ausübung ausgedehnter bürgerlicher Rechte

und der Begriff “Eigentum” erfaßt heute jede Form von Besitz
– unkörperlich oder körperlich.”

informationelle Selbstbestimmung als Schutze der Privatsphäre

- Daten dürfen *nur aufgrund eines Gesetzes* oder
  mit *ausdrücklicher Einwilligung* des Betroffenen verarbeitet werden.

- Daten dürfen *nur* zu dem *Zweck* verarbeitet werden,
  für den sie *ursprünglich erhoben* wurden.

- Daten sollen möglichst nur *direkt* beim Betroffenen erhoben werden.

- "Falsche" Daten müssen *berichtigt* werden;
  "nicht mehr benötigte" Daten müssen *gelöscht* werden.

- Den "Datenverarbeitern" wird eine "aktive Pflicht" zum Schutz der Privatsphäre auferlegt.

Bundesdatenschutzgesetz

- §1(1): Zweck dieses Gesetzes ist es,

  den Einzelnen davor zu schützen,
  dass er durch den Umgang mit seinen personenbezogenen Daten
  in seinem *Persönlichkeitsrecht* beeinträchtigt wird.

- §1(2): Dieses Gesetz gilt für die

  Erhebung,
  Verarbeitung und
  Nutzung

  personenbezogener Daten durch ...

Bundesdatenschutzgesetz

I. Allgemeine und gemeinsame Bestimmungen
- §1 Zweck und Anwendungsbereich
- §2 öffentliche und nicht-öffentliche Stellen
- §3 weitere Begriffsbestimmungen
- §3a Datenvermeidung und Datensparsamkeit
- §4 Zulässigkeit der Datenerhebung, -verarbeitung und -nutzung
- §4a Einwilligung
- §4b Übermittlung ins Ausland sowie an über- und zwischenstaatliche Stellen
- §4c Ausnahmen
- §4d/e Meldepflicht
- §4f/g Beauftragter für den Datenschutz
- §5 Datengeheimnis
- §6 Unabdingbare Rechte des Betroffenen
- §6a Automatisierte Einzelentscheidung
- §6b Beobachtung öffentlich zugänglicher Räume
- §6c mobile personenbezogene Speicher- und Verarbeitungsmedien
- §§7/8 Schadensersatz
- §9 Technische und organisatorische Maßnahmen
- §9a Datenschutzaudit
- §10 Einrichtung automatisierter Abrufverfahren
- §11 Erhebung, Verarbeitung oder Nutzung im Auftrag

II. Datenverarbeitung der öffentlichen Stellen
- §§12–18 Rechtsgrundlagen
- §§19–21 Rechte der Betroffenen: Auskunft, Benachrichtigung, Berichtigung, Löschung, Sperrung
- §§22–26 Bundesbeauftragter für den Datenschutz und die Informationsfreiheit

III. Datenverarbeitung nicht-öffentlicher Stellen und öffentlich-rechtlicher Wettbewerbsunternehmen
- §§27–32 Rechtsgrundlagen
- §§33–35 Rechte der Betroffenen: Auskunft, Benachrichtigung, Berichtigung, Löschung, Sperrung
- §§36–38 Aufsichtsbehörde

IV. Sondervorschriften
- §§39–42 Berufs- und Amtsgeheimnis, Forschung, Medien

V. Schlußvorschriften
- §§43–44 Strafen und Bußgelder

andere Rechtsvorschriften zur informationellen Selbstbestimmung

- entsprechende *Landes*datenschutzgesetze
- *bereichsspezifische Regelungen:*
  - Hippokratischer Eid und §203 Strafgesetzbuch: *ärztliche Schweigepflicht*
  - §35 Sozialgesetzbuch I: *Sozialgeheimnis*
  - §§66–77 Sozialgesetzbuch X: Einschränkungen des Sozialgeheimnisses
- *Teledienstegesetz* mit *Teledienstedatenschutzgesetz*

  (für einheitliche wirtschaftliche Rahmenbedingungen für die verschiedenen Nutzungsmöglichkeiten der elektronischen Informations- und Kommunikationsdienste; z.B. elektronische Bankgeschäfte, elektronischer Handel, Internetzugang)
- ...

H.P. Bull (erster Bundesbeauftragte für den Datenschutz)

"*Datenschutz* soll den

fairen, rechtsstaatlich angemessenen und
in den gebotenen rechtlichen Formen stattfindenden

Umgang mit *personenbezogenen Daten* gewährleisten und damit die

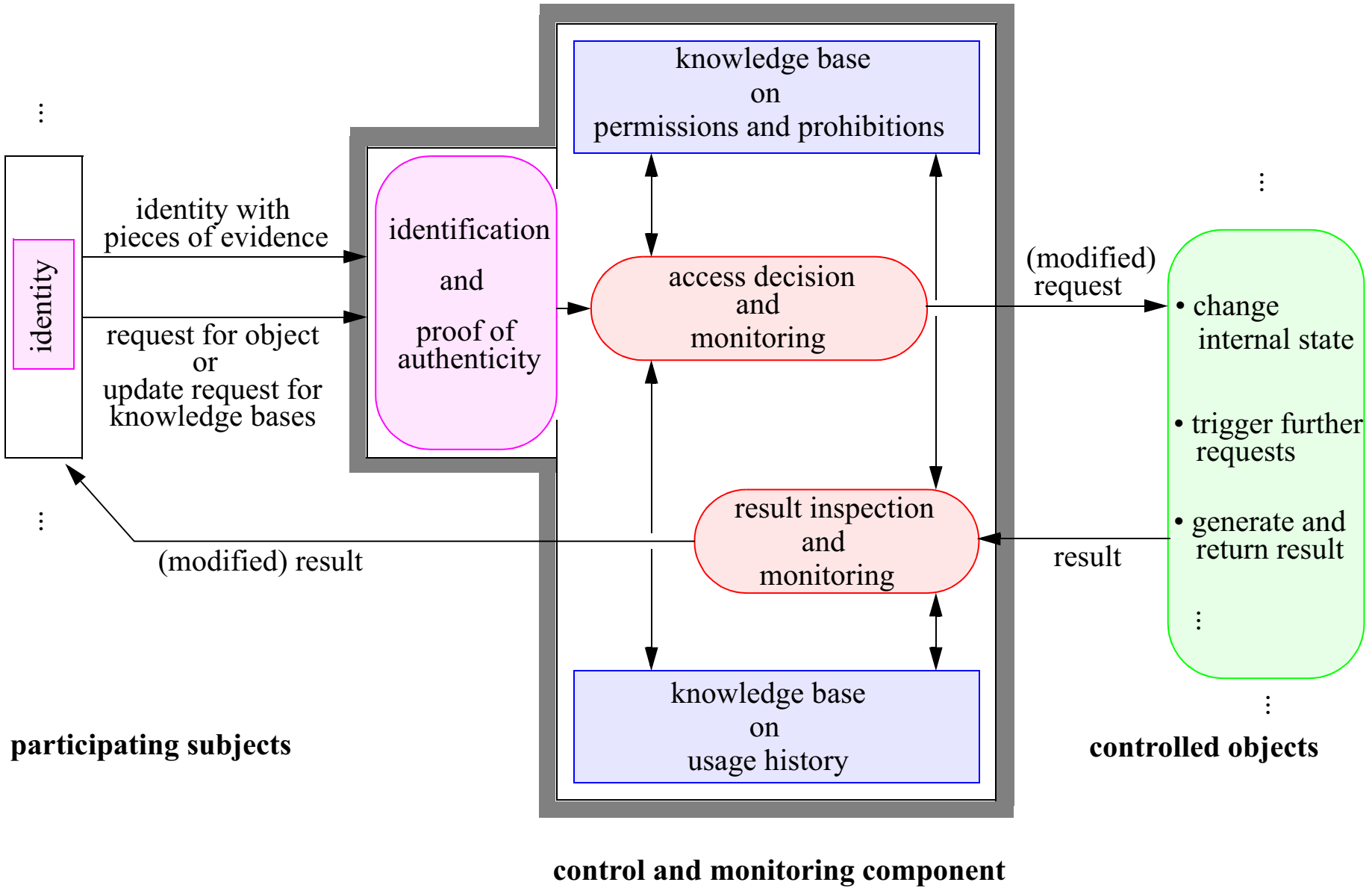
Selbstbestimmung,
Entscheidungsfreiheit und
Persönlichkeitsentfaltung

des Einzelnen im modernen Industrie- und Verwaltungsstaat fördern

und die *Ansammlung übermäßiger Machtpotentiale*
durch Informationskonzentration verhindern."

15.3 Kontrolle und Überwachung als Informationssystem-Aufgabe

ein allgemeines Modell für Kontrolle und Überwachung



Kontrolle und Überwachung: widersprechende Sicherheitsinteressen ausgleichen entsprechend “need-to-know”

- ***Verfügbarkeit*:**

  Erlaubnisse für solche Daten (Informationen) erforderlich,
  die für Aufgabenstellung notwendig sind

- ***Vertraulichkeit, Integrität oder ähnliche Interessen:***

  Verbote für alle anderen Daten (Informationen),
  die für Aufgabenstellung nicht notwendig sind

Kontrolle und Überwachung: Verfügbarkeit gewährleisten

zwei Phasen:

- Administrator vergibt *dauerhaft* Erlaubnisse:
  dargestellt in “Wissensbanken”

- Begünstigter nutzt *wiederholt* seine Erlaubnisse:
  immer dann, wenn für ihn erforderlich

möglicherweise, aber nicht immer:

- Verfügbarkeit kann ausdrücklich beendet werden
  vermöge Entzug der dauerhaften Erlaubnisse
- manche Ansätze ermöglichen auch dynamische Entscheidungen
  vermöge der Benutzungsgeschichte

statische Erlaubnisse und zusätzliche *dynamische* Bedingungen

statische Erlaubnisse:

- immer und “im Grundsatz”

zusätzliche **dynamische Bedingungen:**

- abhängig vom Zustand (Handlungen in der Vergangenheit)
- reichen von “bedingungslos/immer” bis zu ausgeklügelten Zustandsbedingungen (z.B. bezüglich der vorangehenden Ergebnisse von Anfragen)

15.4 Sicherheit in Oracle-SQL

- ACID- Eigenschaften vermöge Transaktionen
- diskretionäre Zugriffskontrolle
- mandatorische Zugriffskontrolle

diskretionäre Zugriffsrechte

- Administrator (`sysdba` oder `sysoper`) darf:
  - Datenbankobjekte erzeugen, ändern und entfernen
  - Benutzer registrieren und entfernen

```
create ...
alter ...
drop ...
```

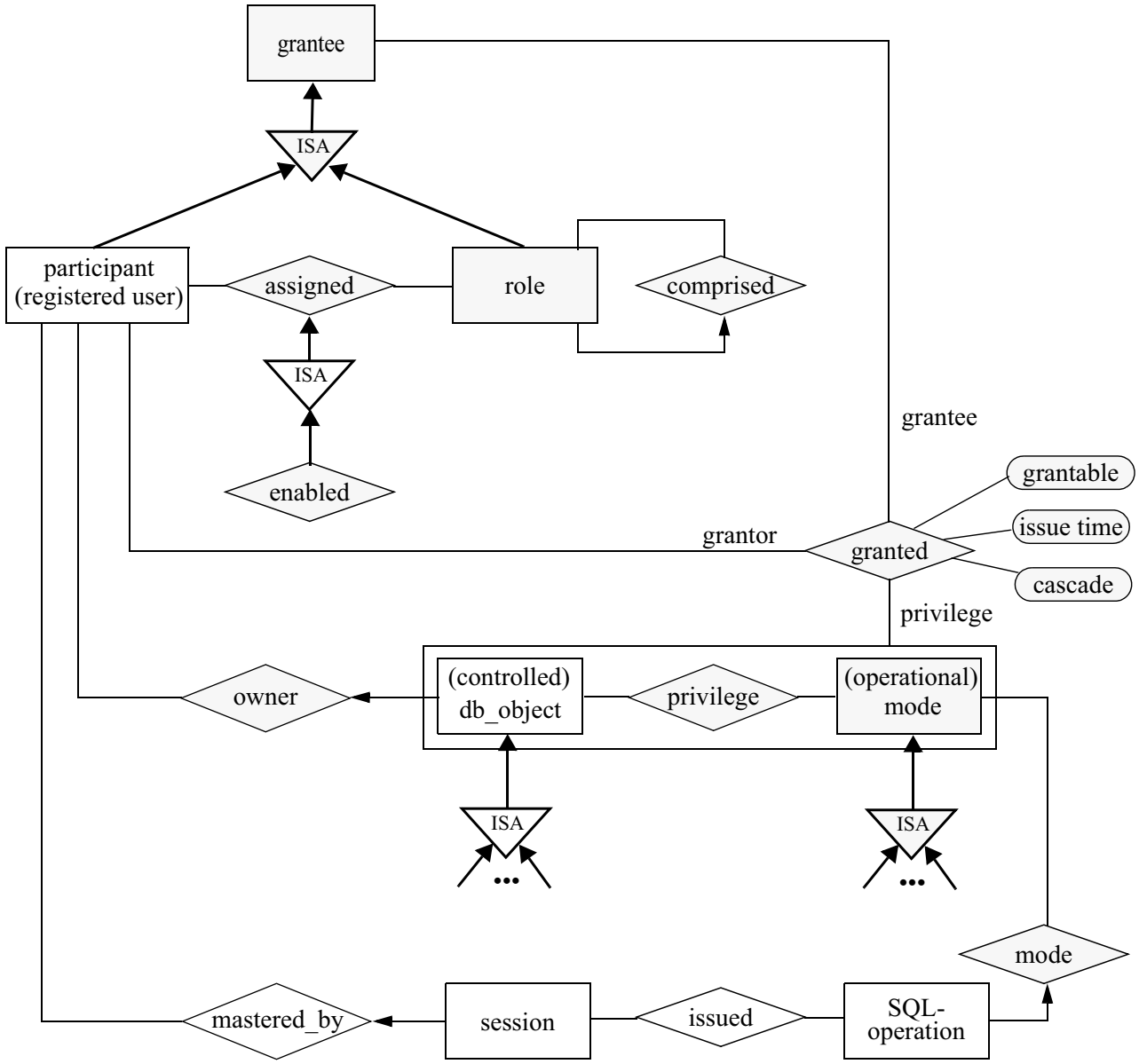
- Administrator als ***Besitzer*** seiner Objekte darf :
  - durch Kontrolloperationen Rechte vergeben und entziehen

```
grant ... to ...
revoke .. from ...
```

  - Objektoperationen ausführen

```
alter ...
delete ...
execute ...
index ...
insert ...
read ...
reference ...
select ...
update ...
```

vereinfachte ER-Modellierung für diskretionäre Zugriffsrechte



vereinfachte Zugriffsentscheidung für diskretionäre Rechte

```
function decide_discret(participant, object, operation):
Boolean;

return
 participant = object.owner

 OR

 [EXISTS granted:
  granted.privilege.mode = operation.mode
  AND
  granted.privilege.db_object = object
  AND
  [granted.grantee = participant
   OR
   [ EXISTS role_1, role_2:
    granted.grantee = role_1
    AND transitively_comprised(role_1, role_2)
    AND enabled(participant, role_2)
   ]
  ]
 ];
```

Oracle-Konzept für "*virtual private database*", VPD

- erlaubt inhaltsabhängige oder zustandsabhängige Abänderungen von Anweisungen oder Ergebnissen

- Beispiel:

  Abänderungen auf Tupelebene durch Vereinbarung einer VPD-*policy*:

  - bewirkt das (konjunktive) Anhängen einer zusätzlichen `where`-Klausel für jede lesende Anfrage an betroffene Relation
  - bewirkt den Aufruf einer gespeicherten Prozedur, um die Rückgabetupel zu filtern

mandatorische Zugriffsrechte

- proprietäre Form von **Sicherheitsmarken** (labels) für

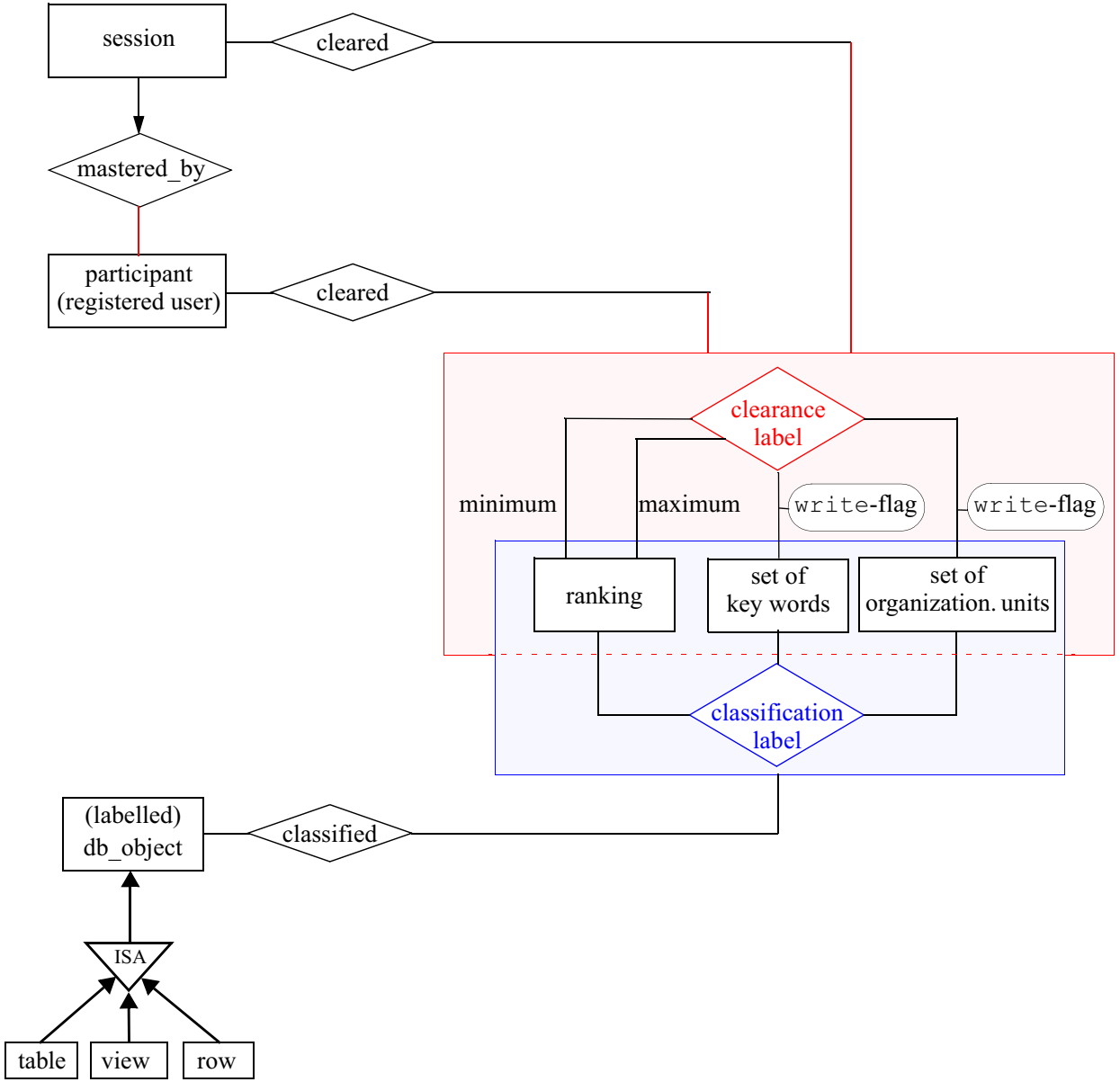
  Freigaben (clearances) von Benutzern und
  Klassifikationen (classifications) von Objekten

- Zugriffssentscheidung:

  Freigabe von anforderndem Benutzer
  muss **passen** (im Allgemeinen bezüglich einer Verbandsordnung) zur Klassifikation des angeforderten Objekts

- besonderer **Produktverband** über $Rank \times \wp\, KW \times \wp\, Org$:
  - *Rank* : Menge von *sensitivity rankings* mit linearer Ordnung $\leq_{Rank}$,
    für "*relative Vertrauenswürdigkeit*" von Benutzern oder
    oder "*relative Schutzwürdigkeit* " von Objekten
  - *KW* : Menge von *keywords*,
    für "*Informationsgehalt*" oder "*beabsichtigte Nutzung*"
  - *Org* : Menge von Bezeichnern für
    "*organisatorische oder geographische Einheiten*"
    mit *Vererbungshierarchie* $\leq_{Org}$

vereinfachte ER-Modellierung für mandatorische Zugriffsrechte



vereinfachte Zugriffsentscheidung für mandatorische Rechte

- statische Zuordnung von Freigaben zu Benutzern

- dynamische Zuordnung einer Freigabe zu einer Session
  im Rahmen der statisch vereinbarten Grenzen für *ranking*:
  - *maximum*: niemals Lesezugriffe darüber
  - *minimum*: niemals Schreibzugriffe darunter

- berücksichtigt `write`-*flag* für Schreibzugriffe

- **Lesezugriff** für ein Tupel erlaubt,
  falls “Freigabe dominiert Klassifikation” (proprietär abgewandelt),
  d.h. “Lesen nur nach unten”:

  - $rank_{row} \leq_{Rank} rank_{sess}$

  - $kws_{row} \subseteq kws_{sess}$

  - $expansion(org_{row}) \cap expansion(org_{sess}) \neq \emptyset$

- **Schreibzugriff** für ein Tupel erlaubt,
  falls “Klassifikation dominiert Freigabe” (proprietär abgewandelt),
  d.h. “Schreiben nur nach oben”:

  - $rank_{row} \leq_{Rank} rank_{sess}$ (auch Lesezugriff erlaubt)

  - für minimum $rank_{user}$ mit $rank_{user} \leq_{Rank} rank_{sess}$,
    $rank_{user} \leq_{Rank} rank_{row}$

  - falls kws_{row} vereinbart,
    $kws_{row} \subseteq kws_{sess}$
    bzw. sogar
    $kws_{row} \subseteq \texttt{write}\text{-}restriction(kws_{sess})$

  - falls org_{row} vereinbart,
    $expansion(org_{row}) \cap expansion(\texttt{write}\text{-}restriction(org_{sess})) \neq \emptyset$

Ausnahmeregeln für Halter von "*label security privileges*"

Beispiele:

- unbeschränkte Lesezugriffe erlauben
- unbeschränkte Lese- und Schreibzugriffe erlauben
- Passendvergleich bezüglich *organizational units* nicht berücksichtigen
- *dynamische Marke* für eine *Session* abändern (insbesondere anheben) zur statischen Marke eines anderen Benutzers
- *ranking* Eintrag für ein Tupel ändern im Rahmen der vereinbarten Grenzen
- *keyword* oder *organizational unit* Eintrag für ein Tupel ändern

Überblick über Sicherheitskonzepte in Oracle

- diskretionäre Zugriffskontrolle
- Besitzer von Objekten
- Weitergabe von Rechten
- Rollen-basierte Rechte mit Rollenhierarchie
- VPD-Politiken
- für mandatorische Zugriffskontrolle
  Sicherheitsmarken: ranking, key words, units
- Freigaben von Benutzern
- Klassifikation von Objekten
- label security privileges für Umgehung der mandatorischen Zugriffskontrolle
- dynamische Rechteübertragung an Sessions

Index