

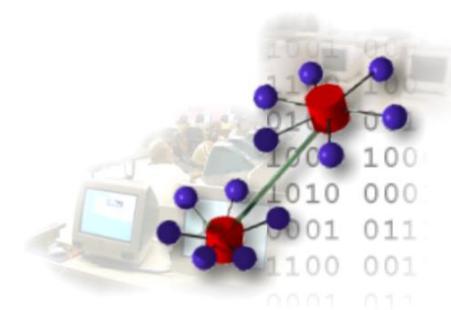


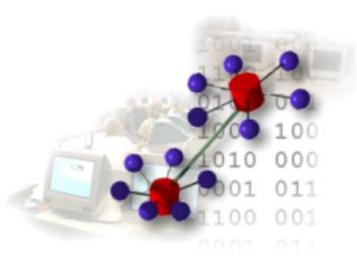
Database
Research Group

Datenbanksysteme

Bernhard Seeger

<http://www.mathematik.uni-marburg.de/~seeger>





Organisation (Vorlesung)

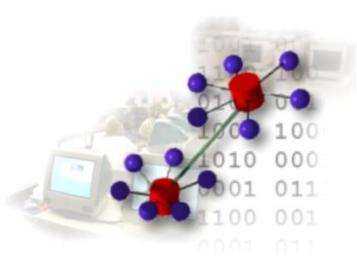
■ Vorlesung

- Fr. 10-14h Hörsaal C im Hörsaalgebäude Lahnberge
 - 10:15 – 11:15 Vorlesung
 - 11:30 – 12:30: Vorlesung
 - 13:00 – 14:00: Vorlesung
- Letzte Vorlesung am 12.7.2019
- Brückentage: 30. Mai, 20. Juni
 - Was sollen wir machen?

*!! Neuer Vorlesungsort !!
!! Neue Vorlesungszeiten !!*

■ Am Samstag, den 18 Mai: SQL-Workshop

- PC-Saal (Ebene D3) neben dem Treppenhaus D im Mehrzweckgebäude Lahnberge



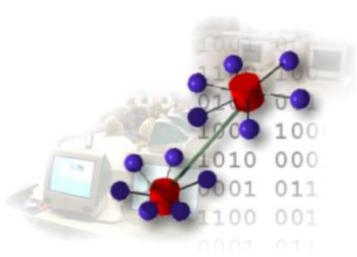
Organisation (Übung)

■ Übungsleiter:

- Jana Holznigenkemper, Andreas Morgen

■ Übungsblätter

- Ausgabe und Abgabe des Übungsblatts am Freitag (10h)
- Übungstermine: Mo 14-16, Di 14-16 und Di 16-18
- Statt einem Übungsblatt wird es am 24.5 einen 30-minütigen Test über den Inhalt des SQL-Workshops geben.
 - Uhrzeit: 10:15 - 10:45.
 - Vorlesungsräume für den Test werden noch angekündigt.



Organisation (Prüfung)

■ Studienleistung

- Mindestens 50% der Übungsaufgaben
- Maximal zwei Übungszettel mit 0 Punkten
- Sonderregelung Lehramt
 - Mindestens 40% der Übungsaufgaben
 - Eine fachdidaktische Zusatzleistung

■ Prüfungsleistung

- Abschlussklausur am Freitag, den 19.7.2019, von 12-14h im 00/0030 der Biegenstraße 14, Hörsaalgebäude.
- Nachholklausur: wird noch bekannt gegeben

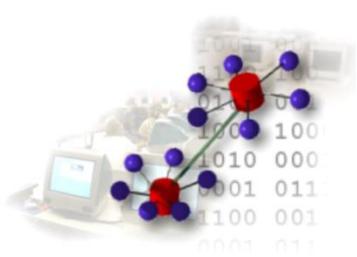


Organisation (Ilias)

■ Nutzung von Ilias

- Alle Materialien werden in Ilias zur Verfügung gestellt.
 - Übungsblätter
 - Folien
- Abgabe der Übungen

- Anmeldung zwingend erforderlich (ab sofort)
- Anmeldung für die Übungen (ab Freitag 18h)

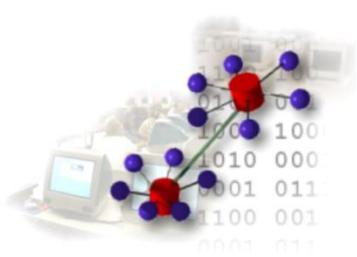


Literatur

Deutsche Bücher

- A. Kemper, A. Eikler: “Datenbanksysteme. Eine Einführung”, De Gruyter Studium, 2015.
 - Frühere Auflagen sind im Oldenbourg-Verlag erschienen.
 - G. Saake, K.-U. Sattler, A. Heuer: “Datenbanken Konzepte und Sprachen”, mitp. 2018.
 - G. Vossen: “Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme”, Oldenbourg, 2008.
- Englische Bücher

- Jeffrey D. Ullman, Jennifer D. Widom: A First Course in Database Systems, Prentice Hall, 2007.
- Raghu Ramakrishnan, Johannes Gehrke: Database Management Systems, McGraw-Hill Professional



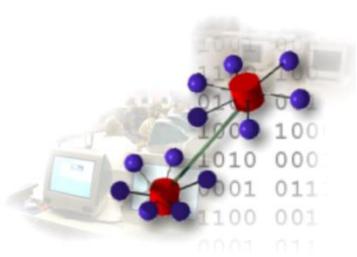
WEB-Quellen

■ Videokanal

- www.datenbankenlernen.de

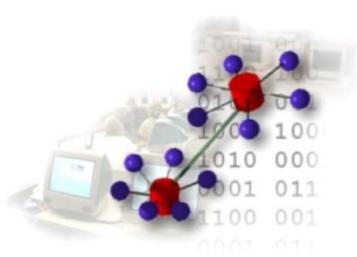
■ Relationale Algebra

- <https://dbis-uibk.github.io/relax/calc.htm>
- [Table API Apache Flink](#)



Inhaltsverzeichnis

- Einführung (26.4)
- Relationales Modell (26.4, 3.5)
 - Relationale Algebra, Tupelkalkül, Erweiterte Relationale Algebra
- SQL: Die relationale Datenbanksprache (10.5, 17.5, 18.5)
- Konzeptioneller Datenbankentwurf (24.5)
- Entwurfstheorie (7.6)
- Transaktionskonzepte u. Fehlerbehandlung (14.6, 21.6)
- Anwendungsprogrammierung (21.6, 28.6)
- Physische Datenorganisation & Implementierung der relationalen Algebra (5.7)
- NotOnly-SQL Systeme (12.7)



Datenbanken und DBMS

■ Datenbanksysteme (DBS)

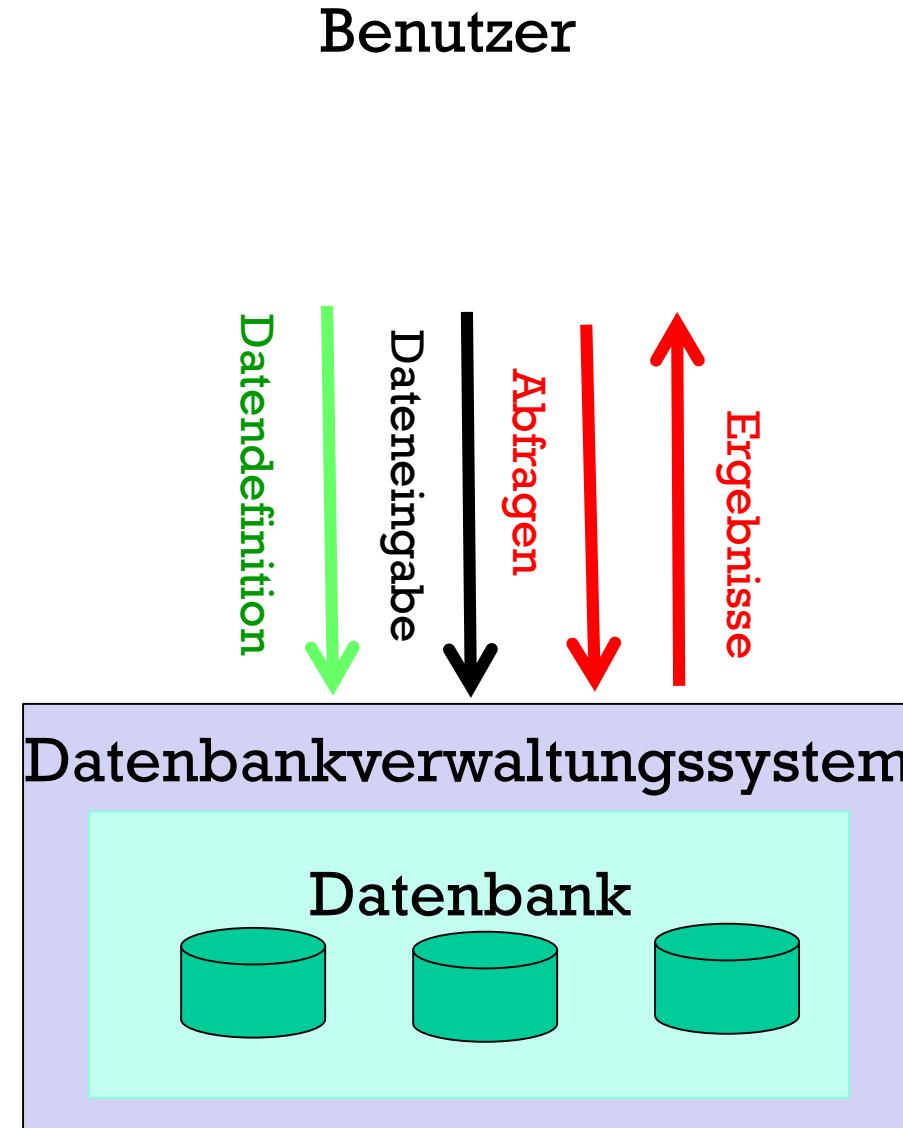
- dienen zur rechnergestützten Verwaltung großer, persistent zu verwaltenden Datenbestände.
 - Lebensdauer der Daten >> Lebensdauer der Programme

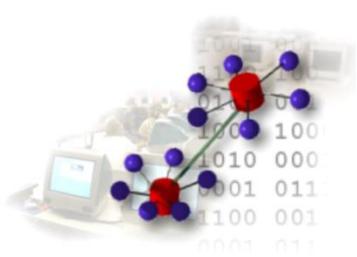
■ Datenbanksysteme bestehen aus

- **Datenbankverwaltungssystem**
(engl. database management system, **DBMS**)
 - Software zur Verwaltung von Daten.
- **Datenbank** repräsentiert eine logisch zusammenhängende Datenmenge bestehend aus
 - den zu verwaltenden Daten,
 - Hilfsdaten (z. B. Indexe, Logdaten, Metadaten).



Benutzerinteraktion





Klassische Anwendungen

- **Bankinformationssystem**
 - Verwaltung der Kunden, ihre Konten, ...
- **Versicherungsinformationssystem**
 - Verwaltung der Kunden, ihre Verträge, ...
- **Telekommunikation**
 - Abrechnung
- **Logistik**
 - Paketversand



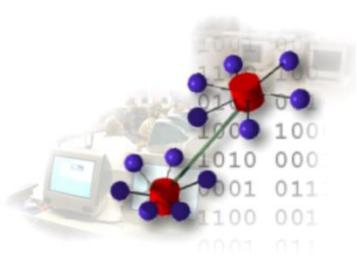
Beispiele

■ Große Datenbanken heute

- Max-Planck-Institut für Meteorologie (2015)
 - Größe
 - 330 TByte (Online-Speicher) + 6PByte (Offline-Speicher)
- AT&T (2015)
 - Anzahl von Datensätzen
 - 1.9 Billionen Datensätze

■ Technologischer Fortschritt im Bereich DBMS

→ Nahezu problemlose Unterstützung aller klassischer Anwendungen (in naher Zukunft)!



Gibt es noch etwas zu tun?

- **Aufzeichnung aller persönlichen Daten in Datenbanken**

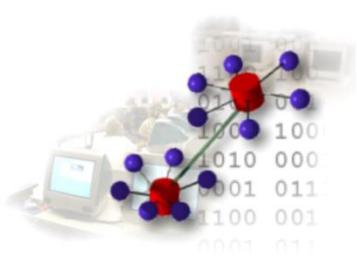
- Banktransaktionen, Email, Bilder, Vitaldaten, Aufzeichnung aller Gespräche
 - Vision
 - Vannevar Bush: „As we may think“ (1945)



- **Technische Realisierung heute möglich**

- Billiger Magnetplattenspeicher
 - Kapazität: 10 TB
 - SSD
 - Kapazität bis zu 2 TB
 - Sehr große Hauptspeicher
 - In Kombination mit NVRAM





Gibt es noch etwas zu tun? (2)

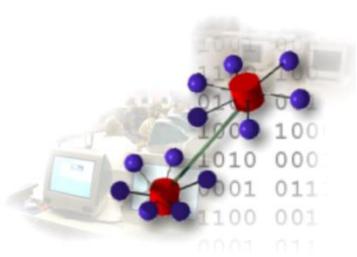
■ Neue Formen der Datenerfassung

- Sensoren
 - Verkehrssensor, Finanzticker



■ Anforderungen

- Verwaltung aller Sensordaten in einer Datenbank
- Neue Arten von Abfragen
 - Unterstützung von historischen Abfragen
 - Wo war ich am 7.7.2009 um 12h?
 - Kontinuierliche Abfragen
 - Informiere mich über Änderungen beim Börsenkurs von AT&T?



Die Vision wird zu Wahrheit!

Soziale Netzwerke

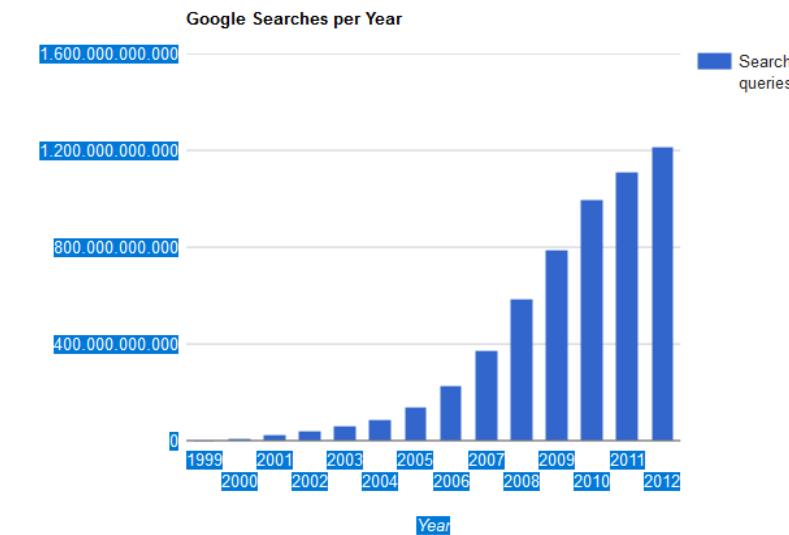
■ Facebook

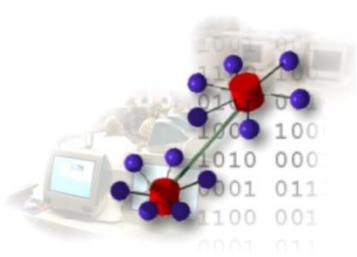
- 3000 neue Bilder pro Sekunde
= 250 Millionen pro Tag
- > 1 Milliarde Nutzer



Web

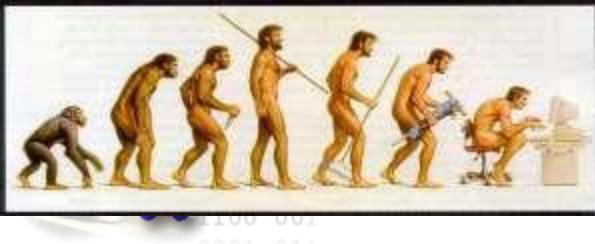
- Google
 - Anzahl von Suchoperationen/Tag
 - 5.5 Milliarden
- Internet Archive
 - 18,5 PByte (2014)
 - 50 Pbyte (2017)





Neue Architekturen für DBMS

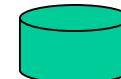
- **DBMS als Ecosystem der Informationsverarbeitung**
 - Entwicklung neuer Architekturen
 - NoSQL-Datenbanken
 - Verteilte Datenbanken
 - Hauptspeicher-Datenbanken
 - Datenstromsysteme
 - Es gibt noch viel zu tun!
- **Diese Entwicklungen werden derzeit unter dem Synonym **BIG DATA** zusammengefasst.**
 - Informatiktechnik mit hoher gesellschaftlicher Relevanz!
 - Internet of Things
 - Industrie 4.0
 -



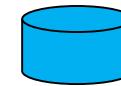
Als es noch keine Datenbanksysteme gab, ...

- Entwicklung von DBS begann vor etwa 50 Jahren.
 - Zuvor wurden vornehmlich einfache Dateisysteme benutzt.
- Beispiel für die Datenverarbeitung in einer Versicherung:
 - Drei Kundenberater **Alfred**, **Beate** und **Carlo**, die je nach Art des Versicherungstyps Kunden betreuen.
 - Jeder der Kundenberater benutzt für den Zugriff auf die Kundendaten ein selbstentwickeltes Programm
 - Jeder Berater hat seine eigene Kundendatei.

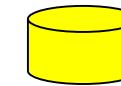
Alfred



Beate



Carlo





Anwendungsprogramme

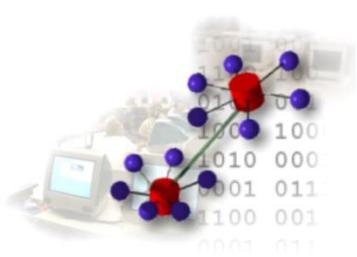
■ Anwendungsprogramm (AWP)

- Ein Programm, das direkt durch den Benutzer oder eine spezifische Anwendungskomponente aufgerufen wird.

■ ~~Java Beispiel (ohne Datenbankzugriff)~~

C#-Beispiel (mit Datenbanksystem)

```
static void Main () {  
    // Verbindung zur Datenbank  
  
    SQLConnection db= new SQLConnection("Data Source = (local); Initial  
    Catalog = MyDatabase; Integrated Security=SSPI");  
  
    // Anfrage  
    var simpleQuery = from tabelle in db  
                      where tabelle.feld > limit  
                      select feld;  
  
    // Zugriff auf die Ergebnisse  
  
    foreach (var f in simpleQuery) { Console.WriteLine(f.feld); }  
}
```



Probleme

- **Direkte Erzeugung und Verarbeitung der Daten erfolgte im AWP unter Verwendung von Dateien**
 - kein standardisiertes Speicherungsformat
 - hoher Aufwand beim Datenaustausch
 - mehrfache und unkoordinierte Verwaltung der Daten
 - häufige Inkonsistenzen im Datenbestand
 - hoher Aufwand bei der Verknüpfung von Daten aus mehreren Dateien
 - Zugriff auf Daten erfolgt explizit im AWP
 - hoher Aufwand bei der AWP-Entwicklung
 - Optimierung des Programmcode durch Entwickler
 - Mehrbenutzerbetrieb nahezu unmöglich
 - Dateninkonsistenzen und Datenverluste
 - Unzureichende Möglichkeiten beim Datenschutz



Anforderungen an Datenbanksysteme (1)

■ Gemeinsame Datenbasis mehrerer Benutzer

- Gemeinsam genutzte, persistente Datenbasis auf schnellem Speicher
 - Direkter Zugriff durch Benutzer
 - Indirekter Zugriff über AWP
- Kontrollierte Datenredundanz
 - Vermeidung von Kopien der gleichen Daten durch integrierte Verwaltung aller Daten.

■ Mehrbenutzerbetrieb

- Gleichzeitiger Zugriff mehrerer Benutzer auf gemeinsame Datenbank
- Virtuelles Einbenutzersystem
 - Keine Beeinflussung durch andere Benutzer



Anforderungen an Datenbanksysteme (2)

- **Sicherstellung der Datenqualität**
 - **Datenintegrität und Datenkonsistenz**
 - Unterstützung von Integritätsbedingungen
 - ➔ Gewährleistung der Korrektheit und Vollständigkeit der Daten
 - Automatische Überprüfung der Bedingungen beim Einfügen, Ändern und Löschen der Daten
 - **Datenschutz**
 - Zugriffskontrolle durch Authentifizierung und Verschlüsselung
 - ➔ Schutz der Datenbank vor nicht-autorisierten Zugriff
 - **Schutz der Daten im Falle eines Systemfehlers**
 - Log-Dateien und Sicherungskopien
 - ➔ Wiederanlauf des Systems und automatisches Wiederherstellen der aktuellen Datenbank



Anforderungen an Datenbanksysteme (3)

- **Bereitstellung von unterschiedlichen Benutzerschnittstellen**
 - Ad-hoc Anfragesprachen für interaktive Benutzer
 - Menügesteuerte, einfach zu benutzende Schnittstellen
 - Spezieller Zugang für Administrator
- **Unterstützung der Softwareentwicklung mit DBMS**
 - Programmierschnittstellen für die Softwareerstellung
 - Schnelle Entwicklung von Software unter Ausnutzung einer mächtigen Infrastruktur
 - Flexible und schnelle Anpassung der Software bei Änderungen in der Datenbank wie z. B.
 - Verteilung der Daten über mehrere Festplatten
 - Änderung der Speicherorganisation
 - Änderung des Typs der Daten



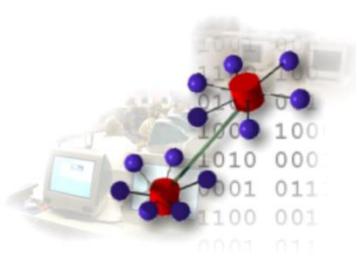
Anforderungen an Datenbanksysteme (4)

■ Hohe Leistungsfähigkeit

- Ziele
 - **Niedrige Antwortzeiten** bei einer Anfrage
 - **Hoher Durchsatz**
 - Maximierung der Anzahl der Anfragen pro Sekunde

■ Lösungen in einem DBMS

- Werkzeuge zur effizienten Speicherung und Anfrageverarbeitung
 - **Indexstrukturen** für große Datenmengen
 - ➔ Logarithmische Zugriffskosten
 - **Effiziente Implementierung** der Algorithmen
 - ➔ Z. B. zum Sortieren großer Datenmengen
- Effektive Anfragebearbeitung
 - Automatische **Optimierung von Anfragen**



Wichtige Konzepte

nicht nur in Datenbanken

■ Datenabstraktion

- Welcher Aspekt einer Anwendung ist relevant und soll in der Datenbank abgebildet werden?
- Einführung von Abstraktionsebenen



■ Datenmodell

- Infrastruktur zur Abbildung der realen Welt



■ Datenunabhängigkeit

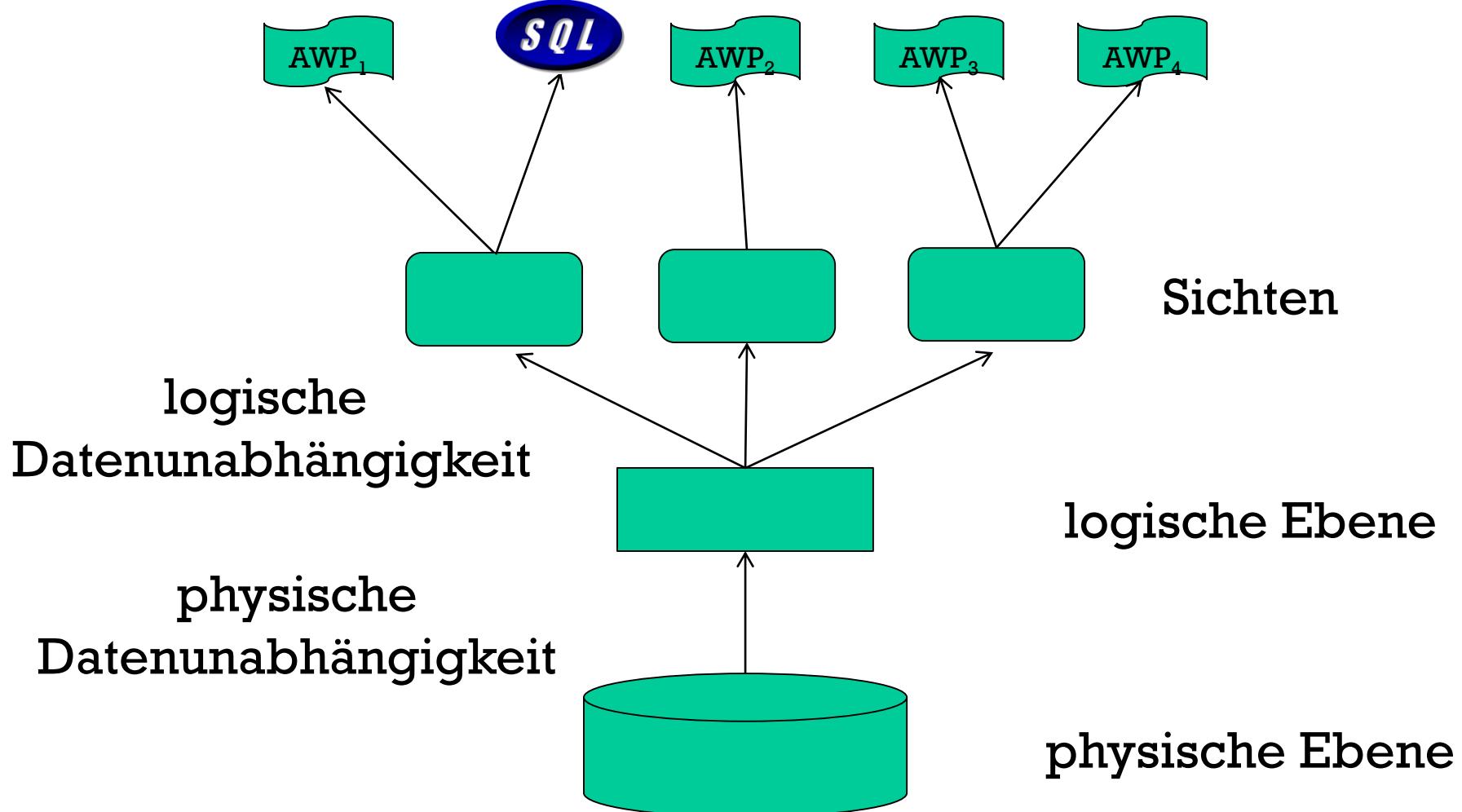


Datenabstraktion

- DBS verfügt über mehrere Abstraktionsstufen
 - **Externe Ebenen (= Sichten)**
 - Beschränkung der Datenbank, der für ein Endbenutzer oder Endbenutzergruppe relevant ist.
→ z. B. Datenschutz
 - **Logische Ebene**
 - Beschreibung aller Daten und deren Beziehungen in der Datenbank
→ z. B. Gemeinsame Datenbasis
 - **Physische Ebene**
 - Festlegung der Speicherstrukturen
→ z. B. Leistungsfähigkeit



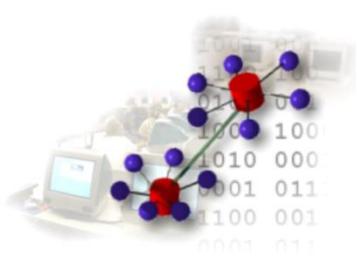
Datenabstraktion





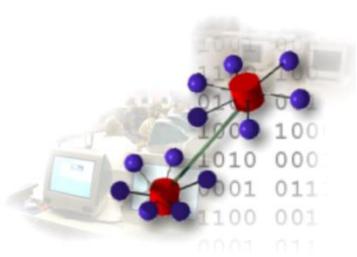
Datenunabhängigkeit

- Änderung einer Ebene beeinflusst nicht die darüber liegenden Ebenen.
- **Logische Datenunabhängigkeit**
 - Änderungen der logischen Ebene haben keinen Einfluss auf die Sichten und damit nicht auf die AWPs.
 - Beispiel:
Kundenkonto soll um das Attribut “Uhrzeit” erweitert werden.
- **Physische Datenunabhängigkeit**
 - Änderungen der physischen Ebene haben keinen Einfluss auf die logische Ebene und damit auch nicht auf die Sichten und die AWPs.
 - Beispiel:
Anlegen eines Suchbaums, um schneller zu suchen.



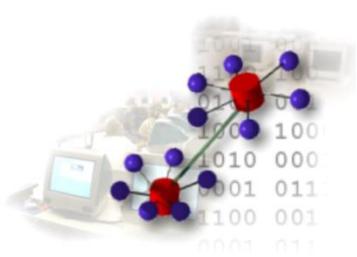
Datenmodell

- Ein Datenmodell bietet eine Infrastruktur zur
 - Strukturbeschreibung von Daten
 - Datendefinitionssprache (DDL)
 - Definition der Syntax und Semantik von Operationen.
 - Datenmanipulationssprache (DML)
 - Einfügen, Ändern und Löschen von Daten
 - Suche nach Daten
- Unterscheidung
 - Datenbankschema
 - Menge aller Strukturbeschreibungen
 - Datenbankinstanz
 - Gültiger Zustand in der Datenbank



Logische Datenmodelle

- DBS besitzen zumindest zwei Datenmodelle:
 - physisches Datenmodell:
zur speicherorientierten Repräsentation der Daten
 - logisches Datenmodell:
zur benutzerorientierten Repräsentation der Daten
- Logische Datenmodelle
 - **relationales Datenmodell**
 - objektorientiertes Modell
 - XML
 - JSON
 - HDF
 - ...

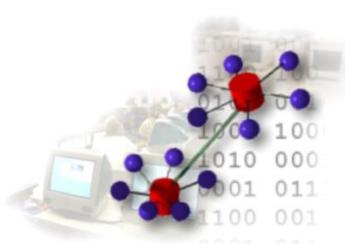


Beispiel (JSON Schema)

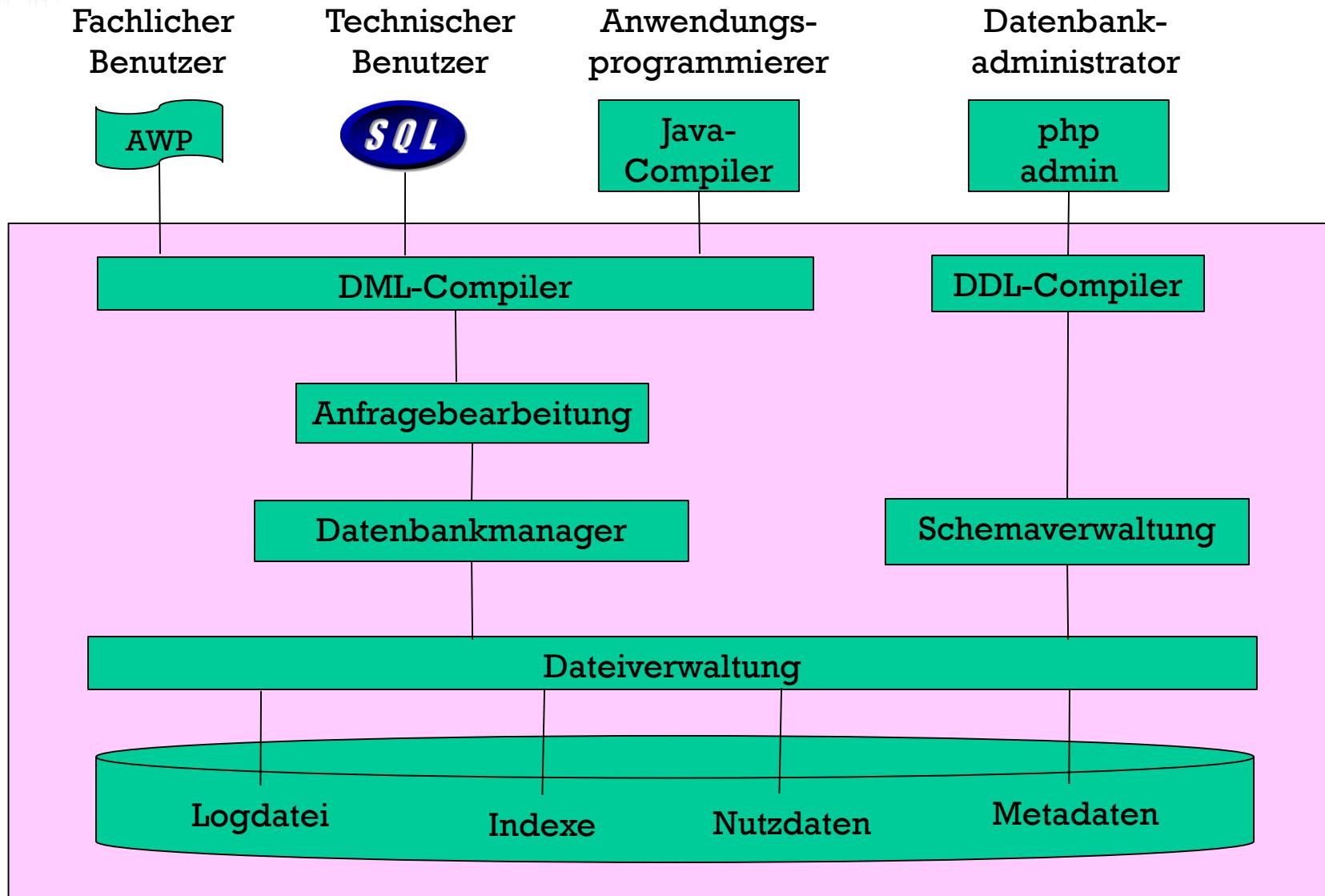
```
{ "name": "Product",  
  "properties": {  
    "id": { "type": "number",  
            "description": "Product identifier",  
            "required": true },  
    "name": { "type": "string",  
              "description": "Name of the product",  
              "required": true },  
    "price": { "type": "number",  
               "minimum": 0,  
               "required": true },  
    "tags": { "type": "array",  
              "items": { "type": "string" } }  
  }  
}
```

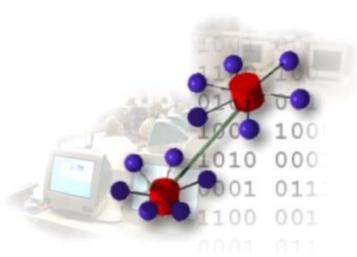
Annotations pointing to specific parts of the JSON schema:

- A blue callout labeled "Name der Eigenschaft" points to the key "name" in the top-level object.
- A blue callout labeled "Typ der Eigenschaft" points to the value "number" under the key "type" for the "id" property.
- A blue callout labeled "Semantische Bedingungen" points to the "description" and "required" fields under the "id" property.



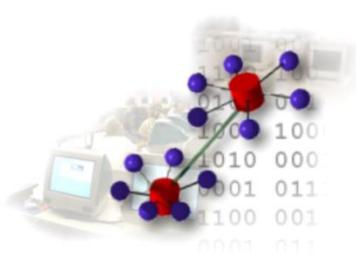
Grobarchitektur eines DBMS



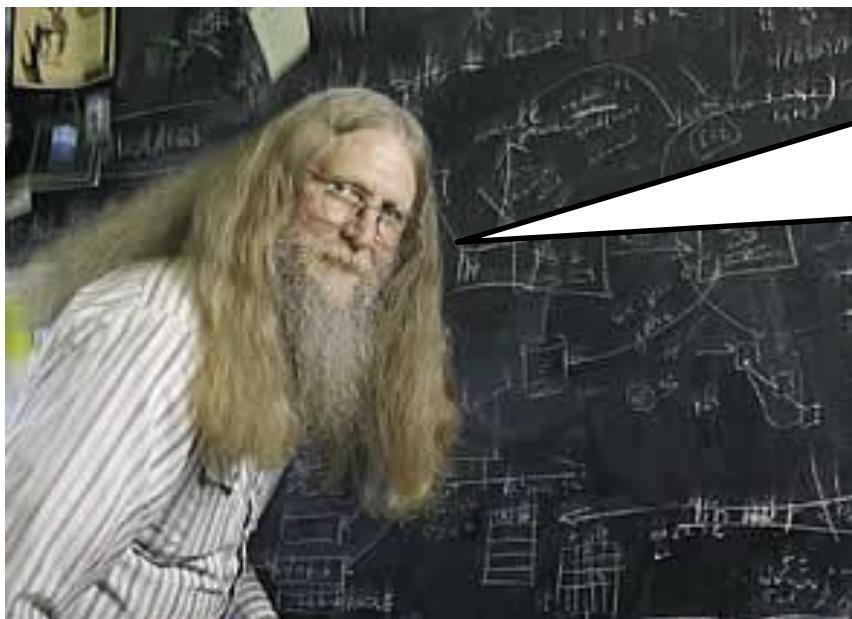


Wichtiges zusammengefasst

- **Informelle Definition eines Datenbanksystems**
- **Gründe, weshalb Dateisysteme sich nicht immer für die Verwaltung von Daten eignen.**
- **Datenunabhängigkeit**
 - logische und physische
- **Datenmodell**
- **Performance**
 - Antwortzeit und Durchsatz

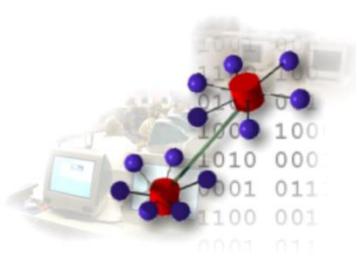


2. Das Relationale Datenmodell



Relational databases
are the foundation of
the western
civilization.

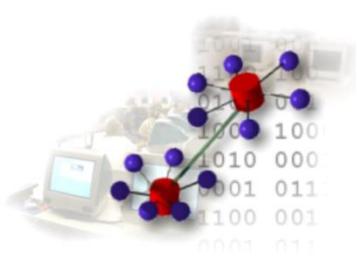
Bruce Lindsay,
IBM Fellow @ IBM Almaden Research Center



Stimmt diese Behauptung?

- **Fast alle kommerziellen DBMS wie z. B.**
 - Oracle, SQL Server, IBM DB2, ...
- und nicht-kommerzielle Systeme wie z. B.**
 - mySQL, PostgreSQL, SQLite H2, ...
- sind relationale Datenbanksysteme und basieren auf dem relationalen Datenmodell.**

- **Nahe zu alle wichtigen Daten unserer „zivilisierten“ Welt werden in relationalen DBMS verwaltet.**
 - Banken
 - Versicherungen
 - Unternehmens- und Wirtschaftsdaten



Auslöser für die Entwicklung relationaler DBMS

■ Mitte der sechziger Jahre

- IBM entwickelt für American Airlines (AA) das Reservierungssystem SABRE
 - Riesiges Prestigeprojekt
 - 200 Datenbankprogrammierer über 4 Jahre
 - Ständige Verzögerungen auf Grund immer wieder neuer Anforderungen (neue Datenfelder) durch AA
 - ➔ Teures Umschreiben und Anpassen der Datenbank mit einer halben Million Programmzeilen

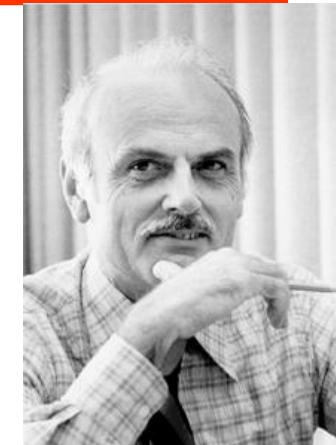
■ IBM damaliges Datenbanksystem IMS hatte substantiell diese Probleme verursacht.

- Insbesondere deshalb hat sich dann IBM entschlossen ein grundlegend neues System zu entwickeln.

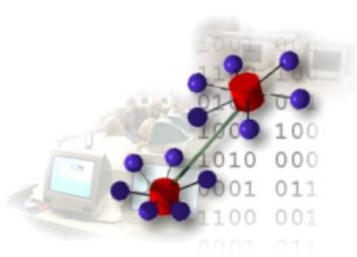


Grundlage der relationalen DBMS

- Edgar Codd schrieb zwei Artikel (1969, 1970) über das relationale Modell.
 - E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Comm. of the ACM 13(6): 377-387(1970)*
- Durch den Einsatz der neuen Technologie konnte das SABRE-Projekt erfolgreich umgesetzt werden.
 - ➔ Heute würde man von einer „disruptive technology“ sprechen.
- Codd erhielt für diese bahnbrechenden Arbeiten 1981 den Turing Award („Nobelpreis der Informatik“)



* Gutachter haben empfohlen diesen Artikel nicht für eine Publikation anzunehmen.



Gründe für den Erfolg des relationalen Modells

“Make things as simple as possible, but not simpler.”

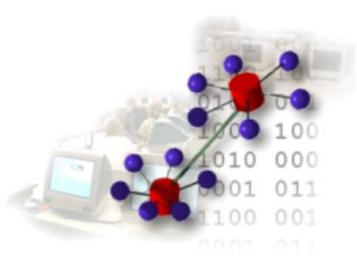
■ Einfachheit

- Relation (entspricht einer Tabelle) als grundlegende Datenstruktur
- Wenige, aber ausdrucksstarke Grundoperationen zur Verarbeitung von Relationen
 - klare Semantik

■ Mengenorientierte Verarbeitung der Daten

■ Formale Grundlagen

- Datenmodell
- Anfragebearbeitung



2.1 Relationen

■ Informelle Beschreibung

- Eine Relation ist eine Tabelle!

■ Beispiel:

- Tabelle **Studierende**

MatNr	Name	Wohnort
7	Bond	London
42	Adams	Cambridge
1527	Philipp	Marburg

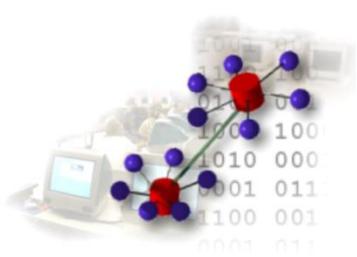
Tabellenname

Attribut

Tabellschema

Tupel

- Implizit gehört zu jedem Attribut ein Wertebereich
 - Wertebereich von MatNr = ganze Zahlen
 - Wertebereich von Wohnort = String



Formale Definition: 1. Versuch

- Sei U eine nicht-leere Menge, das **Universum** der Attribute.
 - Zu jedem **Attribut** $A \in U$ gibt es einen **Wertbereich** $\text{dom}(A)$.
 - Der Wertebereich $\text{dom}(A)$ eines Attributs A ist endlich und besteht aus atomaren Elementen, die keine weitere Struktur besitzen.
 - Beispiele hierfür sind int oder String.
- Ein **Relationenschema** RS ist eine Liste von unterschiedlichen Attributen (A_1, \dots, A_k) aus dem Universum U .
 - Der Parameter k wird **Grad** oder **Stelligkeit** des Schemas genannt.
 - Der **Wertebereich** $\text{dom}(RS)$ des Schema ergibt sich aus:
$$\text{dom}(RS) = \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_k)$$
- Zu einem Relationenschema RS bezeichnet r eine **Relation** falls $r \subseteq \text{dom}(RS)$.
- Für ein **Tupel** t einer Relation r gilt $t \in r$.

Was ist bei diesen Definitionen das grundlegende Problem?



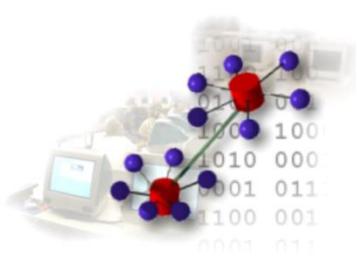
Formale Definition: 2. Versuch

- Sei U eine nicht-leere Menge, das **Universum** der Attribute.
 - Zu jedem **Attribut** $A \in U$ gibt es einen **Wertbereich** $\text{dom}(A)$.
 - Der Wertebereich $\text{dom}(A)$ eines Attributs A ist endlich und besteht aus atomaren Elementen, die keine weitere Struktur besitzen.
 - Beispiele hierfür sind int oder String.
- Ein **Relationenschema** RS ist eine Teilmenge des Universums U .
 - Die Anzahl der Elemente in RS wird **Grad** oder **Stelligkeit** des Schemas genannt.
- Zu einem Relationenschema RS ist die **Relation** $r = r(RS)$ eine **endliche Menge von totalen Abbildungen** t mit .

$$t : RS \rightarrow \bigcup_{A \in RS} \text{dom}(A)$$

und $t(A) \in \text{dom}(A)$ für alle $A \in RS$. t wird auch als **Tupel** bezeichnet.

Was ist der Vorteil dieser Definitionen?

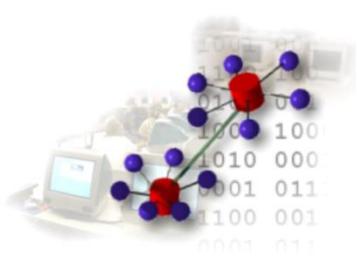


Beispiel

■ Beispiel: Tabelle Studierende

MatNr	Name	Wohnort
7	Bond	London
42	Adams	Cambridge
1527	Philipp	Marburg

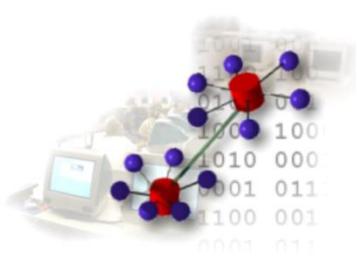
- **Attribute:** MatNr, Name, Wohnort
 - Wertebereiche
 - $\text{dom}(\text{MatNr}) = \text{Menge der ganzen Zahlen}$
- **Relationenschema:** {**MatNr, Name, Wohnort**}
- **Tupel t_1**
 - $t_1(\text{MatNr}) = 7$
 - $t_1(\text{Name}) = \text{"Bond"}$
 - $t_1(\text{Wohnort}) = \text{"London"}$



Gleichheit von zwei Relationen

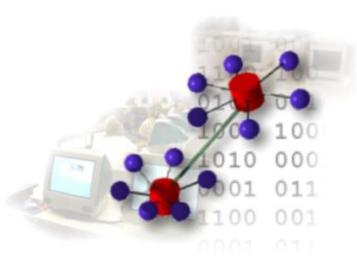
- **Zwei Relationen r und s sind **gleich**, falls**
 - ihre Relationenschemata gleich sind und
 - ihre Mengen der totalen Abbildungen gleich sind.

- **Die Reihenfolge der Attribute hat keine Bedeutung!**



Weitere Begriffe

- Relationen können über einen eindeutigen Namen angesprochen werden.
 - Wenn dies nicht der Fall ist, sprechen wir von einer **temporären Relation**.
 - Eine Relation r hat somit zwei wichtige Bestandteile
 - Relationenschema RS_r
 - Relationenname $Name_r$
- Sei r eine Relation und $t \in r$ ein Tupel. Sei $X \subseteq RS_r$.
 - Dann bezeichnet $t[X]$ das Tupel t eingeschränkt auf X .
 - Ist $X = \{A\}$, so schreiben wir kurz $t[A]$ (statt $t[\{A\}]$).



Integritätsbedingungen

- Zu einem Schema RS können wir nun die Menge der möglichen Relationen betrachten

$$\text{REL}(\text{RS}) = \{r \mid r(\text{RS})\}$$

- Wenn wir die Tabelle Studierende betrachten, sind weitere Einschränkungen dieser Menge sinnvoll.

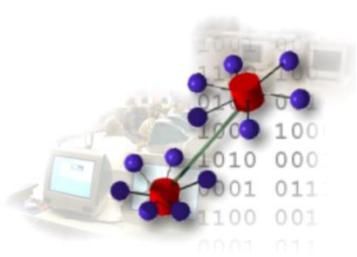
Warum?

- Das Attribut **MatNr** der Tabelle Studierende hat eine besondere Bedeutung.
 - MatNr ist eindeutig!
 - Es gibt keine zwei unterschiedlichen Tupel mit einem gleichen Wert für das Attribut MatNr.
 - Diese Eigenschaft wird als **Integritätsbedingung** bezeichnet. Wir verkleinern damit die Menge der möglichen Relationen REL(RS).



Schlüssel

- Für ein Relationenschema RS wird $X \subseteq RS$ als **Schlüssel** bezeichnet, falls folgende Bedingungen für alle möglichen Relationen $r(RS)$ gelten:
 - **Eindeutigkeit:**
 - Für zwei beliebige Tupel $t_1, t_2 \in r$ gilt:
$$t_1[X] = t_2[X] \Rightarrow t_1 = t_2$$
 - **Minimalität:**
 - Es gibt keine echte Teilmenge $Y \subset X$, so dass die Eindeutigkeit erfüllt ist.
- Es gibt mindestens einen Schlüssel in einem Relationenschema
 - Es kann aber auch mehrere Schlüssel geben.

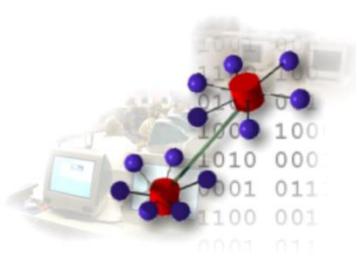


Primärschlüssel

- Der **Primärschlüssel** ist ein ausgezeichneter Schlüssel des Relationenschemas.
 - Es gibt nur einen Primärschlüssel.
 - Der Primärschlüssel wird in einer Datenbank als Stellvertreter für Datensätze genutzt.
 - Die Attribute des Primärschlüssels werden im Schema durch Unterstreichen hervorgehoben.

Kennst Du Max Mustermann aus Musterhausen, der am Fachbereich 12 Mathe(Bachelor) studiert, derzeit Datenbanksysteme I hört, und im siebten Fachsemester ist?

Meinst Du den Studenten mit Matrikelnr. 124223?



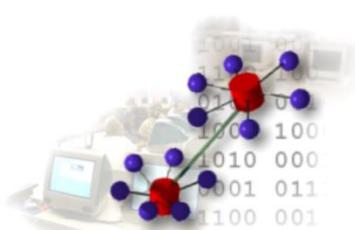
Formale Definition

- Eine (**lokale**) **Integritätsbedingung** lib eines Relationenschemas RS ist eine Boolesche Funktion
$$lib: REL(RS) \rightarrow \{\text{true}, \text{false}\}$$
- Zu einem Relationenschema RS bezeichnet LIB_{RS} die **Menge aller lokalen Integritätsbedingungen**.
- Beispiel (Relationschema Studierende)
 - MatNr ist ein Schlüssel
- Definition (**Konsistente Relation**)
 - Eine Relation r ist in einem konsistenten Zustand, wenn alle dem Schema zugeordneten lokalen Integritätsbedingungen erfüllt sind.



Datenbank

- **Bisher haben wir nur eine einzige Relation/Tabelle betrachtet.**
- **Eine relationale Datenbank besteht aber i. A. aus sehr vielen Relationen.**
 - Beispiele
 - Universitätsinformationssystem
 - Relationen: Studierende, Vorlesung, Belegung
 - Online-Shop
 - Relationen: Kunde, Produkt, Bestellung
 - ERP (Enterprise Resource Management)
 - Relationen: Personal, Abteilung, Maschine,



Beispiel (ERP-Datenbank)

PMZuteilung

<u>pnr</u>	<u>mnr</u>	Note
67	84	3
67	93	2
67	101	3
73	84	5
114	93	5
114	101	3
51	93	2
69	101	2
333	84	3
701	84	2
701	101	2
82	101	2

Personal

<u>pnr</u>	PName	VName	Abt	Lohn
67	Meier	Helmut	B10	65000
73	Müller	Margot	B10	51000
114	Bayer	Martin	A63	60000
51	Daum	Birgit	A64	72000
69	Störmer	Willi	A64	60000
333	Haar	Hans	A63	75000
701	Reiner	Willi	A64	42500
82	Just	Michael	A64	65000

Maschine

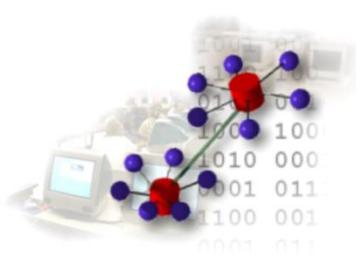
<u>mnr</u>	MName
84	Presse
93	Füllanlage
101	Säge

ALeitung

<u>abtnr</u>	<u>pnr</u>
B10	67
A63	333
A64	333

Abteilung

<u>abtnr</u>	AName
B10	Spielzeug
A63	Computer
A64	Suppen



ERP-Datenbankschema

■ Relation Personal

- Angestellte in einem Unternehmen
- Schema: 5 Attribute, Primärschlüssel: pnr

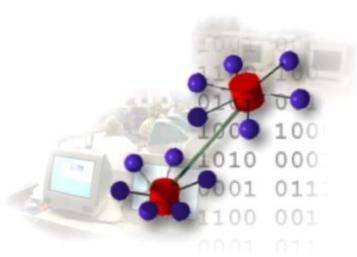
■ Relation Maschine

- Maschinen in einem Unternehmen
- Schema: 2 Attribute, Primärschlüssel: mnr

■ Relation PMZuteilung

- Welche Angestellte können welche Maschine, wie gut bedienen?
- Schema: 3 Attribute, Primärschlüssel: {mnr, pnr}

■ ...



Fremdschlüssel

Beobachtung

- Relation PMZuteilung besitzt Attribute, die in einer anderen Relationen Primärschlüssel sind.
 - Diese Attribute werden als **Fremdschlüssel** bezeichnet.

Fremdschlüsselbedingung

- Jeder Wert eines Fremdschlüssel muss auch in der Relation vorhanden sein, in der das Attribut Primärschlüssel ist.



Formale Definition (Datenbank)

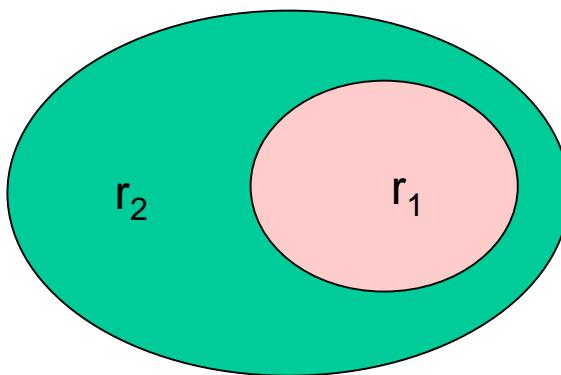
- Ein **Datenbankschema** $DS = \{RS_1, \dots, RS_m\}$ besteht aus einer endlichen Menge von Relationenschemata.
 - Jedem Schema RS_j ist eine Menge von lokalen Integritätsbedingungen $LIB_j = LIB_{RS_j}$ zugeordnet.
- Zu einem Datenbankschema DS ist $d = \{r_1, \dots, r_m\}$ eine **Datenbank**, falls $r_i \in REL(RS_i)$.
 - $DB(DS)$ bezeichnet die **Menge aller Datenbanken** über dem Schema DS .

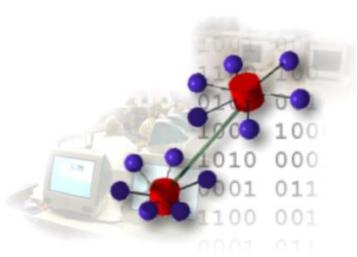


Formale Definition (Fremdschlüssel)

- Sei DS ein Datenbankschema und $RS_1, RS_2 \in DS$ zwei Relationenschemata, wobei X Primärschlüssel von RS_2 ist. Dann ist $Y \subseteq RS_1$ ein **Fremdschlüssel**, wenn für alle Relationen $r_1 \in \text{REL}(RS_1)$ und $r_2 \in \text{REL}(RS_2)$ gilt:

$$\{t[Y] \mid t \in r_1\} \subseteq \{t[X] \mid t \in r_2\}$$





Formale Definition

- Eine globale **Integritätsbedingung** ib eines Datenbankschemas DS ist eine Boolesche Funktion
$$ib: DB(DS) \rightarrow \{\text{true, false}\}$$
- Zu einem Datenbankschema DS bezeichnet IB_{DS} die **Menge aller Integritätsbedingungen**.
- Beispiel (Relationschema PMZuteilung)
 - mnr ist ein Fremdschlüssel

Definition (**Konsistente Datenbank**)

- Eine Datenbank d ist in einem **konsistenten Zustand**, wenn alle dem Schema zugeordneten Integritätsbedingungen erfüllt sind.



2.2 Die relationale Algebra

Motivation

- Definition einer Sprache für die Verarbeitung von Relationen → **SQL**
 - Sprache wird genutzt, um Daten
 1. aus Tabelle(n) zu lesen,
 2. diese zu verarbeiten,
 3. und das Ergebnis wieder in Form einer (temporären) Tabelle zur Verfügung zu stellen.

PMZuteilung

pnr	mnr	Note
67	84	3
67	93	2
67	101	3
73	84	5
...

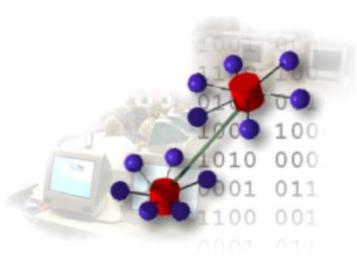
Eingabe

SQL

```
select pnr  
from PMZuteilung  
where Note = 5
```

Ausgabe

pnr
73
114



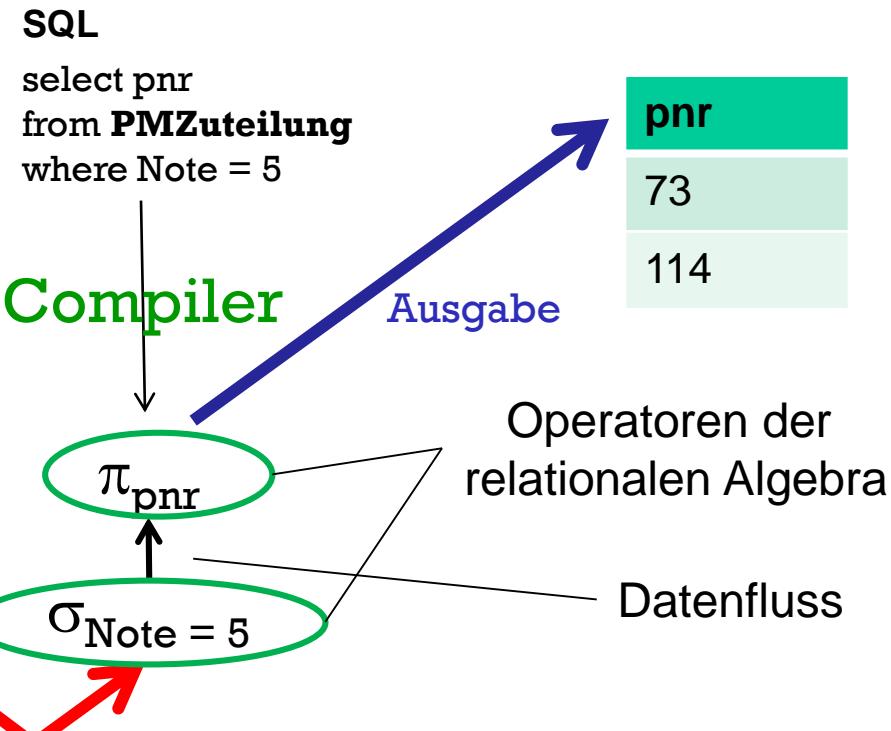
SQL und relationale Algebra

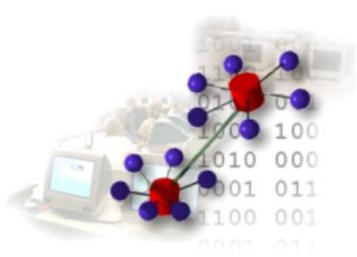
■ Übersetzung einer SQL-Anfrage

- Ausdruck in SQL-Ausdruck wird übersetzt in einen Ausdruck einer **Algebra von Operatoren**.
 - Jeder Operator hat als Eingabe eine Tabelle und erzeugt als Ausgabe ebenfalls eine Tabelle.

PMZuteilung

pnr	mnr	Note
67	84	3
67	93	2
67	101	3
73	84	5
...





Anforderungen an die Algebra

■ Ausdrucksstärke

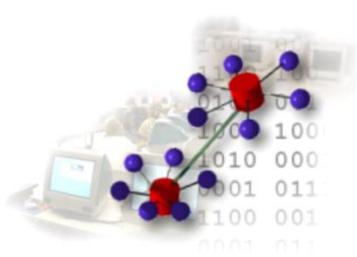
- Was ist mit SQL bzw. der relationalen Algebra alles berechenbar?
→ siehe Theoretische Informatik: Turing-berechenbar

■ Effizienz

- Können die Operatoren der Algebra effizient implementiert werden?
 - In welcher Komplexitätsklasse liegen die Operatoren?
 $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$
→ siehe Praktische Informatik II

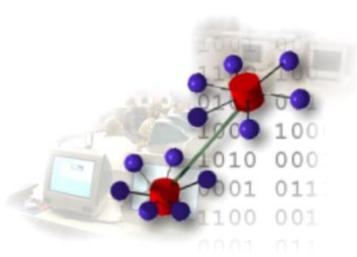
■ Einfachheit

- Was ist die minimale Anzahl von Operatoren für die gewünschte Ausdrucksstärke?



Relationale Algebra

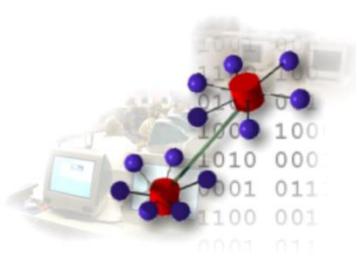
- Algebra
 - Gegeben eine Menge N (“Anker der Algebra”)
 - Menge von Operatoren $OP = \{op_1, \dots, op_n\}$ der Form
$$op_j: N^k \rightarrow N$$
 - Beispiel (Boolesche Algebra)
 - $N = \{\text{true}, \text{false}\}$
 - $OP = \{\neg, \wedge, \vee, 0, 1\}$ mit $\wedge, \vee: N \times N \rightarrow N$ und $\neg: N \rightarrow N$
- Definition (**Relationale Algebra**)
 - Anker ist die Menge aller Relationen
 - Menge von **6 Operatoren**: $\sigma, \pi, \cup, \times, -, \rho$
 - Diese Bezeichnungen sind historisch bedingt und werden heute noch in allen Büchern zu Datenbanken genutzt.



2.2.1 Basisoperatoren

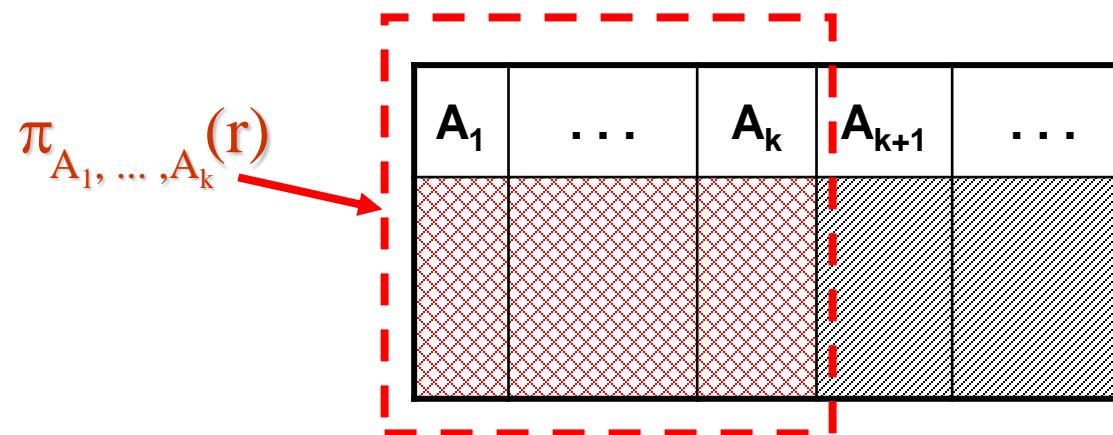
Die Relationale Algebra besitzt 6 Basisoperatoren:

- Projektion
- Selektion
- Umbenennung
- Vereinigung
- Differenz
- Kartesisches Produkt



Projektion π

- **Eingabe**
 - Eine Relation und ein oder mehrere Attribute
- **Ausgabe**
 - Filtern von Spalten aus einer Relation
 - Alle nicht genannten Attribute werden eliminiert.





Formale Definition der Projektion

- Sei $r \in \text{REL}(\text{RS})$ eine Relation und $X \subseteq \text{RS}$. Die Ausgabe von $\pi_X(r)$ ist eine (temporäre) Relation s mit
 - $\text{RS}_s = X$
 - $s = \{t[X] \mid t \in r\}$
- Kurzschreibweise: $s = \pi_X(r)$
- **Man beachte die Mengensemantik!**
 - Mögliche Duplikate in der Ausgaberation werden eliminiert.



Selektion σ

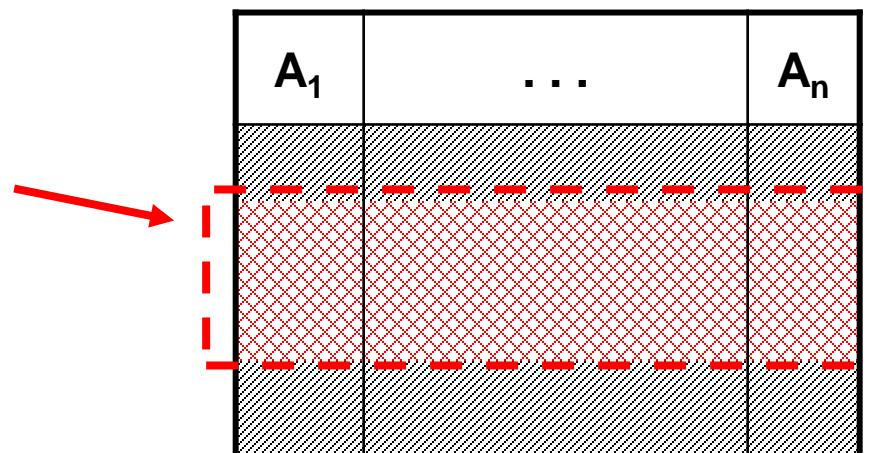
■ Eingabe

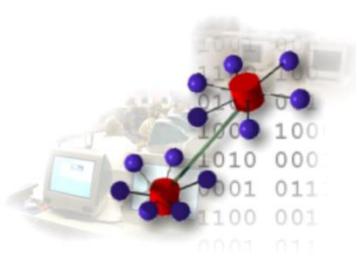
- Eine Relation r und eine Bedingung B

■ Ausgabe

- Eine Relation mit allen Zeilen von r , die B erfüllen.

$\sigma_{cond}(r)$





Formale Definition der Selektion

- Sei $r \in \text{REL}(\text{RS})$ eine Relation und $F: \text{RS} \rightarrow \{\text{true}, \text{false}\}$ eine Boolesche Funktion. Die Ausgabe von $\sigma_F(r)$ ist eine Relation s mit
 - $\text{RS}_s = \text{RS}$
 - $s = \{t \mid F(t) \text{ und } t \in r\}$
- Kurzschreibweise: $s = \sigma_F(r)$
- Boolesche Funktion F besteht aus Attributen von RS, Konstanten, Vergleichsoperatoren ($=, \neq, <, \leq, >, \geq$) und Booleschen Operatoren (\wedge, \vee, \neg).
 - Lohn > 50000
 - Lohn $> 50000 \wedge \text{VName} = \text{"Willi"}$!



Umbenennung ρ

■ Eingabe

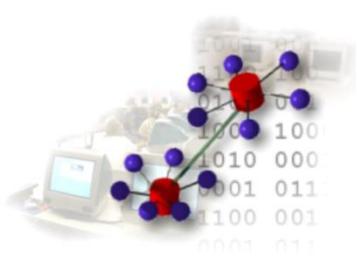
- Relation r und ein Attribut A aus dem Schema und ein Attribut B, das nicht im Schema von r ist.

■ Ausgabe

- Eine Relation mit exakt den gleichen Datensätzen wie r, aber statt dem Attribut A mit dem Attribut B.

$\rho_{Bk \leftarrow Ak}(r)$

A_1	...	A_{k-1}	B_k	A_{k+1}	...	A_n



Formale Definition (Umbenennung)

■ Umbenennung eines Attributs

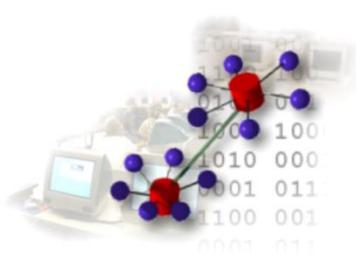
- Sei $r \in \text{REL(RS)}$ eine Relation und sei $A \in \text{RS}$ und $X \notin \text{RS}$ mit $\text{dom}(X) = \text{dom}(A)$. Die Ausgabe von $\rho_{X \leftarrow A}(r)$ ist dann eine (temporäre) Relation s mit
 - $\text{RS}_s = \text{RS} \setminus \{A\} \cup \{X\}$
 - $s = \{t \mid t \in r\}$

■ Umbenennung von zwei (und mehreren) Attributen

- $\rho_{X \leftarrow A, Y \leftarrow B}(r)$

■ Zusätzlich kann auch die Umbenennung einer Relation definiert werden.

- Sei $r \in \text{REL(RS)}$ eine Relation und $Name$ ein eindeutiger Bezeichner. Dann erzeugt $s = \rho_{Name}(r)$ eine neue Relation s mit $\text{Name}_s = Name$.



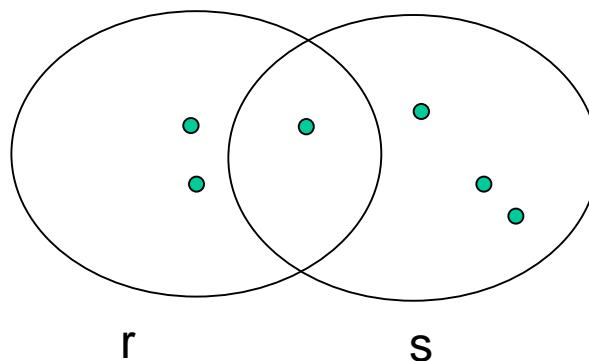
Vereinigung \cup

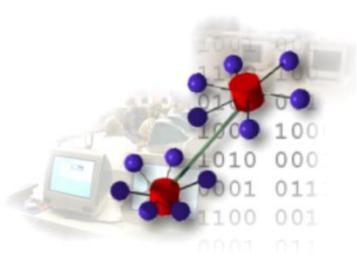
■ Eingabe

- Zwei Relationen r und s mit gleichem Schema ($RS_r = RS_s$).

■ Ausgabe

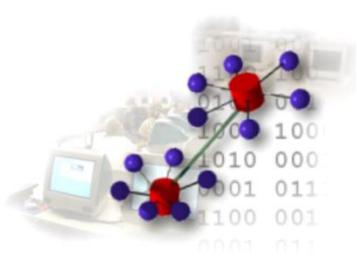
- Eine Relation mit allen Datensätzen aus r und s .





Formale Definition (Vereinigung)

- Seien $r_1, r_2 \in \text{REL}(\text{RS})$ zwei Relationen über dem gleichen Schema RS. Die Ausgabe von $r_1 \cup r_2$ ist dann eine (temporäre) Relation s mit
 - $\text{REL}_s = \text{RS}$
 - $s = \{t \mid t \in r_1 \text{ oder } t \in r_2\}$
- Man beachte dabei, dass die Mengensemantik gilt.
 - Ein Tupel, das in beiden Eingaberelationen vorkommt, wird in der Ausgaberelation nur einmal vorkommen.



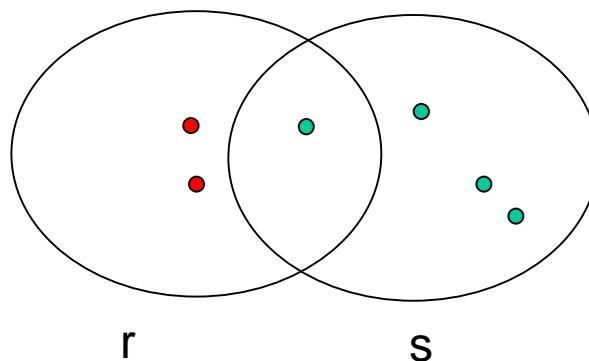
Differenz -

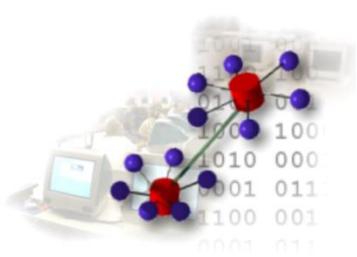
■ Eingabe

- Zwei Relationen r und s mit gleichem Schema.

■ Ausgabe

- Eine Relation mit allen Datensätzen aus r die nicht in s sind.





Formale Definition (Differenz)

- Seien $r_1, r_2 \in \text{REL}(\text{RS})$ zwei Relationen über dem gleichen Schema RS. Die Ausgabe von $r_1 - r_2$ ist dann eine (temporäre) Relation s mit
 - $\text{REL}_s = \text{RS}$
 - $s = \{t \mid t \in r_1 \text{ und } t \notin r_2\}$
- Mengensemantik!



Kartesisches Produkt \times

■ Eingabe

- Zwei Relationen r und s mit disjunkten Schemata.

■ Ausgabe

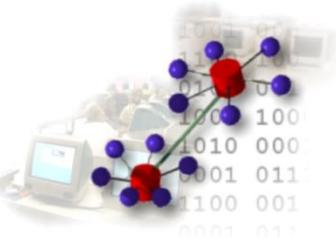
- Eine Relation, in der jedes Tupel aus r mit jedem Tupel aus s verknüpft ist.

r	A	B

s	C	D

$r \times s$

A	B	C	D



Formale Definition (Kartesisches Produkt)

- Seien $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$ zwei Relationen mit $\text{RS}_1 \cap \text{RS}_2 = \emptyset$. Die Ausgabe von $r_1 \times r_2$ ist dann eine (temporäre) Relation s mit
 - $\text{REL}_s = \text{RS}_1 \cup \text{RS}_2$
 - $s = \{t \mid t[\text{RS}_1] \in r_1 \text{ und } t[\text{RS}_2] \in r_2\}$

■ Anmerkungen

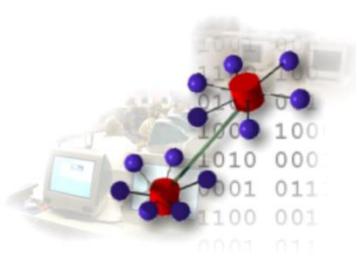
- Bei Gleichheit von Attributen in r_1 und r_2 kann man über Umbenennung dafür sorgen, dass die Schemata disjunkt werden.



Beispiele für Anfragen (1)

Datenbankschema siehe Folie 49

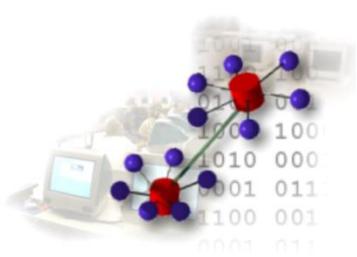
- Bestimme alle Angestellte und deren Lohn, die mehr als 60000 verdienen.



Beispiele für Anfragen (2)

Datenbankschema siehe Folie 49

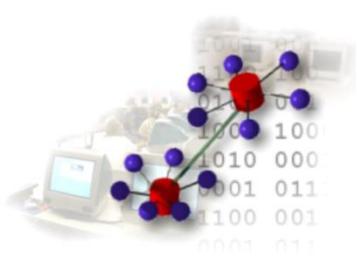
In welcher Abteilung arbeiten die Angestellten mit Nachnamen Müller?



Beispiele für Anfragen (3)

Datenbankschema siehe Folie 49

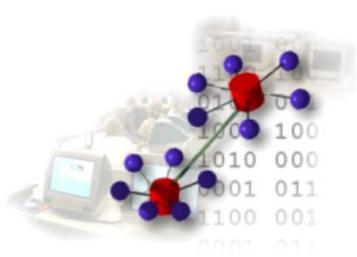
- *Finde die Namen aller Angestellten aus der Abteilung Computer.*



Beispiele für Anfragen (4)

Datenbankschema siehe Folie 49

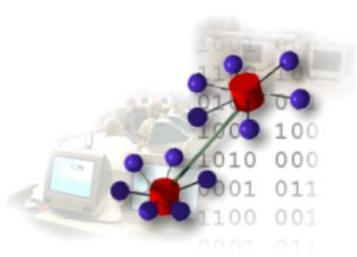
- Bestimme die Angestellten mit gleichem Vornamen.



Beispiele für Anfragen (5)

Datenbankschema siehe Folie 49

- *Finde alle Angestellten, die nur an der Maschine mit Nummer 84 ausgebildet sind.*

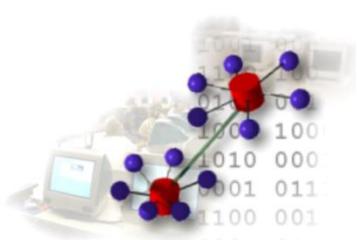


Implementierung einer Algebra als Klasse

- Definition einer **Klasse Table** mit folgenden Methoden:

Relationale Algebra	Methoden
Filter	Table filter (?)
Projektion	Table select (?)
Umbenennung	Table as (?)
Vereinigung	Table union (Table right)
Differenz	Table minus (Table right)
Kartesisches Produkt	Table cross (Table right)

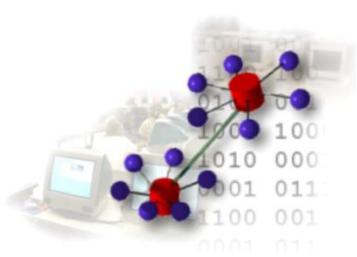
- Bei den ersten drei Operatoren müssen die Eingabeparameter noch angegeben werden.
- Es fehlen noch Methoden, um Table-Objekte zu erstellen.



Verfeinerung des Entwurfs

Relationale Algebra	Methoden
Filter	Table filter(String pred)
Projektion	Table select (String plist)
Umbenennung	Table as (String list)
Vereinigung	Table union(Table right)
Differenz	Table minus(Table right)
Kartesisches Produkt	Table cross(Table right)
Initialisierung	Table scan(String relName)

- **Die Parameter für die Methoden filter, select und as werden als Zeichenkette übergeben.**
 - Hierfür müssen wir noch überlegen, wie die Zeichenkette interpretiert werden soll.
- **Zusätzlich gibt es eine Methode scan, um eine Relation aus der Datenbank zu lesen.**



Umbenennung

■ Anwendung

- Schritt für Schritt:

- Table t = new Table();
t = t.scan("Maschinen");
t = t.as("m, name");

- In einem Ausdruck:

- Table t = (new Table()).scan(("Maschinen").as ("m, name"));

■ Wichtige Eigenschaften

- Jedes Attribut des Table-Objekts bekommt einen neuen Namen.
 - Die Namen werden in eine durch Komma separierte Liste angegeben.
 - Jeder Name in der Liste ist eindeutig.



Projektion

■ Anwendung

- Schritt für Schritt:

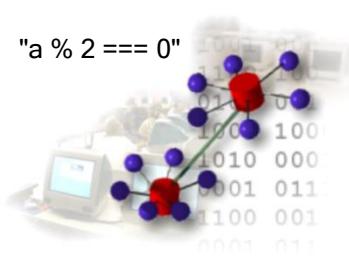
```
Table t = new Table();
t = t.scan("Personal");
t = t.select("pnr, Lohn");
```

- In einem Befehl:

```
Table t = (new Table()).scan(("Personal").select("pnr, Lohn");
```

■ Wichtige Eigenschaften

- Es werden Attribute des Table-Objekts ausgewählt.
- Die Namen müssen in dem Table-Objekt wirklich vorhanden sein.



Filter

■ Anwendung

- Schritt für Schritt:

```
Table t = new Table();
t = t.scan("Personal");
t = filter("Lohn === 60000");
```

- In einem Befehl:

```
Table t = (new Table()).scan(("Personal").filter("Lohn ===
60000");
```

■ Vereinfachende Annahmen

- Für unsere Vorlesung beschränken wir uns zunächst auf Gleichheitsprädikate mit einem Attribut und einem Wert.
 - Wir verwenden === für den Test auf Gleichheit.
 - Der Wert und der Typ des Attributs müssen zusammenpassen.

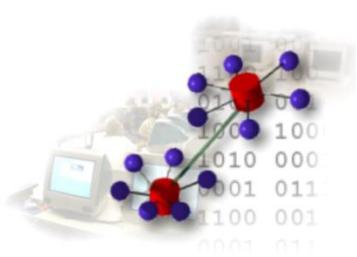


Apache Flink



Apache Flink

- Diese Funktionalität existiert bereits in dem Open-Source Framework [Apache Flink](#).
 - Flink bietet sowohl Funktionalität für die Verarbeitung von Datenströmen als auch Tabellen.
 - Für die Verarbeitung von Tabellen gibt es die [Table-API](#).
- Diese API bietet noch viel mehr an Funktionalität.
 - Prädikate lassen sich noch viel flexibler angeben als wir es bisher in der Vorlesung kennengelernt haben.
 - Auf einige dieser Möglichkeiten werden wir in der Vorlesung noch eingehen.
 - Es gibt weitere Operationen, die wir z. T. noch in der Vorlesung vorstellen werden.



2.2.2 Abgeleitete Operatoren

■ Motivation

- Definition von höherwertigen Operatoren, um Anfragen einfacher zu formulieren.

■ Operatoren

- Verbundoperationen
 - Theta-Verbund (theta join)
 - Natürlicher Verbund (natural join)
 - Semi-Verbund (semi join)
 - Anti-Join (anti join)
- Schnitt
- Division
 - Allquantifizierte Anfragen



Schnitt

■ Eingabe

- Zwei Relation r_1 und r_2 mit gleichem Schema.

■ Ausgabe

- Ergebnisrelation mit Tupeln, die in r_1 **und** in r_2 vorkommen.

■ Formale Definition (**Schnitt**)

- $r_1 \cap r_2 := r_1 - (r_1 - r_2)$

■ Wir können also das Ergebnis auf die bisherigen Operationen zurückführen.

- Eine Angabe des Relationenschemas ist nicht erforderlich.



Theta Join

■ Informell

- Verknüpfung von zwei Relationen bezgl. einem Prädikat θ , dass sich auf Attribute der Relation r_1 und r_2 bezieht.

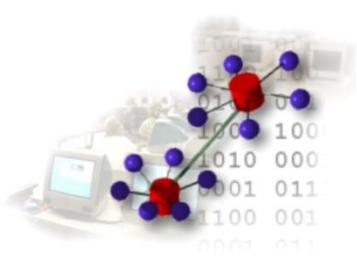
■ Beispiel

- Gegeben eine Menge von POI (Point of Interests) mit Attributen X und Y. Bestimme die Paare von Punkten, die höchstens 100m voneinander entfernt sind.

■ Formale Definition

- Seien $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$ mit $\text{RS}_1 \cap \text{RS}_2 = \emptyset$.

$$r_1 \bowtie_{\theta} r_2 := \sigma_{\theta}(r_1 \times r_2).$$



Theta Join - Eigenschaften

Beispiel für Theta Join:

r_1

A	D
1	a
3	b
2	a

r_2

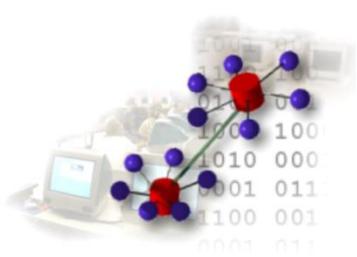
B	C
5	2
0	1
1	1

\bowtie
 $A \leq C \wedge B > 0$

=

A	D	B	C
1	a	5	2
1	a	1	1
2	a	5	2

- Joinbedingungen Θ sind aufgebaut wie Selektionsbedingungen.
- Theta Join und natürlicher Verbund sind **kommutativ** und **assoziativ**, d.h. die Klammerungsreihenfolge bei Mehrfach-Joins ist im Prinzip unwesentlich.



Umsetzung in Flink



Apache Flink

- In Flink gibt es zusätzlich die Methode
 - Table join(Table right)
- Dieser Methode muss noch ein Aufruf der Methode where mit der Joinbedingung folgen.
 - Es werden nur **Bedingungen mit Gleichheit** unterstützt.
- Beispiel
 - Table left = new Table().scan(Rel1, "a, b, c");
 - Table right = new Table().scan(Rel2, "d, e, f");

Table result = left.join(right).where("a == d");



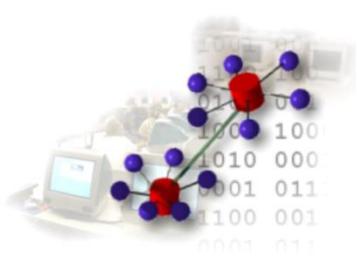
Natürlicher Verbund (1)

■ Eingabe

- Zwei Relationen r_1 und r_2 die gleiche Attribute besitzen.
 - Durch die Verwendung von **PrimärschlüsseIn als Fremdschlüssel** ist dies oft der Fall.
 - Ziel ist es, die Daten aus beiden Relation miteinander zu verknüpfen.

■ Ausgabe (konstruktive Beschreibung)

1. Wir erzeugen das kartesische Produkt (ohne dabei zu fordern, dass die Schemata disjunkt sind).
 - Zur Unterscheidung bekommen gleiche Attribute als Präfix den Relationenname (und einem Punkt).
2. Es werden alle Tupel eliminiert, die in den „gleichen“ Attributpaaren unterschiedliche Werte haben.
3. Danach wird noch für jedes dieser Attributpaare eins der Attribute eliminiert.



Natürlicher Verbund (2)

Beispiele

- Gegeben folgende Relationen r_1 und r_2 .

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

- Natürlicher Verbund zwischen der Relation PMZuteilung und Maschine.

1. Kartesisches Produkt

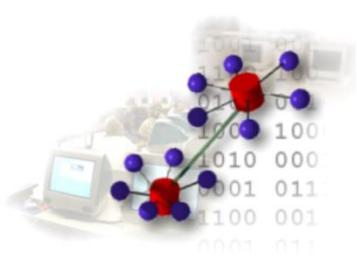
A	$r_1.B$	$r_1.C$	$r_2.B$	$r_2.C$	D
1	2	3	2	3	4
...

2. Selektion

- Erhalte nur die Tupel mit $r_1.B = r_2.B$ und $r_1.C = r_2.C$.

3. Projektion

- Entferne die „doppelten“ Spalte $r_2.B$ und $r_2.C$



Formale Definition (Natürlicher Verbund)

- Gegeben zwei Relationen $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$. Dann ist

$$r_1 \bowtie r_2 := \{t \mid t[\text{RS}_1] \in r_1 \text{ und } t[\text{RS}_2] \in r_2\}$$

- Man beachte, dass bei dieser Definition kein Relationenschema für t angegeben wurde. Dieser ergibt sich implizit aus der kleinsten Menge von Attributen $(\text{RS}_1 \cup \text{RS}_2)$, so dass die **Bedingung** syntaktisch korrekt ist.

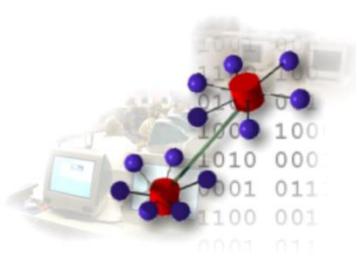
$$r_1 \bowtie r_2$$

$\text{RS}_1 - \text{RS}_2$			$\text{RS}_1 \cap \text{RS}_2$			$\text{RS}_2 - \text{RS}_1$		
A_1	\dots	A_m	B_1	\dots	B_k	C_1	\dots	C_n
				gemeinsame Attribute				



Unterschiede zwischen Theta Join und Natural Join

- Beim Theta-Join ist es erforderlich, dass die Attribute der Relationen unterschiedlich sind.
→ Es gibt **keine** automatische Selektion attributgleicher Tupel
- Beim Theta-Join findet **keine** automatische Projektion auf "relevante" Spalten statt.



Semi-Join

■ Informell

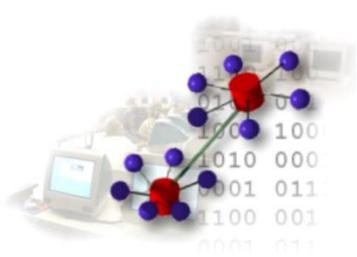
- Der Semi-Join von Relationen r und s berechnet alle Tupel der Relation r, die am Join mit der Relation s beteiligt sind.

■ Beispiel

- Berechne alle Personen, die mindestens eine Maschine bedienen können.

■ Varianten

- Linker und rechter Semi Join
 - Teilrelationenbildung eines der beiden Join-Operanden
 - Nur diejenigen Tupel des ausgewählten Join-Operanden werden ausgewählt, die "einen Joinpartner" besitzen.
- Der Semi-Join kann für alle Join-Varianten definiert werden.



Semi-Join (Definition)

Seien $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$.

$$r_1 \ltimes r_2 := \pi_{\text{RS}_1}(r_1 \bowtie r_2)$$

- Beispiel

$$\begin{array}{l} r_1 \\ \hline \begin{array}{|c|c|} \hline A & B \\ \hline 1 & 2 \\ \hline 3 & 1 \\ \hline 2 & 5 \\ \hline \end{array} \end{array} \times \begin{array}{l} r_2 \\ \hline \begin{array}{|c|c|} \hline B & C \\ \hline 5 & 2 \\ \hline 2 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \end{array} = \begin{array}{|c|c|} \hline A & B \\ \hline 1 & 2 \\ \hline 2 & 5 \\ \hline \end{array}$$



Anti-Join

- "vergessene" Operation (von Codd nicht eingeführt und in Lehrbüchern meist nicht erwähnt): **Anti-Join** (auch Complement (Semi) Join)
- Tupel aus einem der beiden Operanden werden ausgewählt, die keinen Joinpartner besitzen.

- Definition: $r_1 \setminus r_2 := r_1 - (r_1 \bowtie r_2)$

- Beispiel:

$$\begin{array}{c} r_1 \\ \hline \begin{array}{|c|c|} \hline A & B \\ \hline 1 & 2 \\ \hline 3 & 1 \\ \hline 2 & 5 \\ \hline \end{array} \end{array} \quad \setminus \quad \begin{array}{c} r_2 \\ \hline \begin{array}{|c|c|} \hline B & C \\ \hline 5 & 2 \\ \hline 2 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \end{array} = \begin{array}{|c|c|} \hline A & B \\ \hline 3 & 1 \\ \hline \end{array}$$

- Semi Join und Anti Join sind - wie alle Joinvarianten - **ableitbar**.
- Semi Join, Anti Join sind nicht kommutativ !



Division

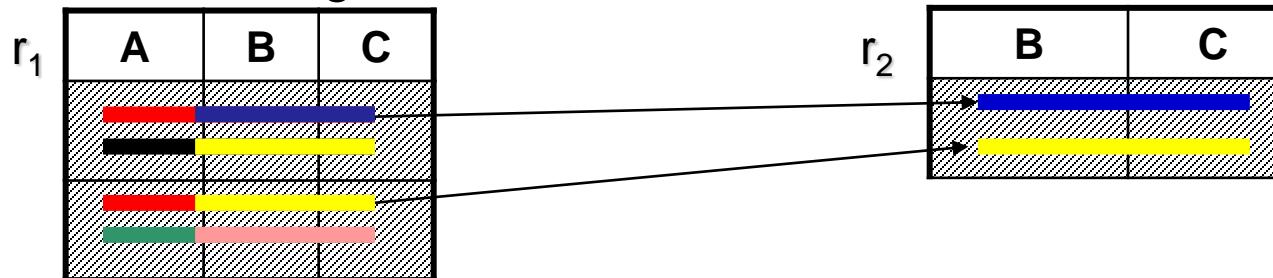
■ Informell

■ Eingabe

- Zwei Relationen r_1 und r_2 , wobei das Schema von r_1 das Schema von r_2 enthält.

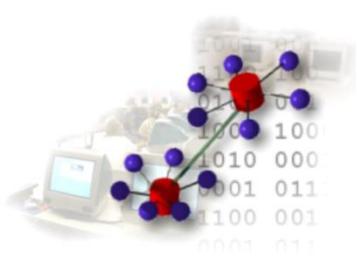
■ Ausgabe

- Tupel aus r_1 , die mit allen Tupeln der Relation r_2 in Beziehung stehen.



■ Beispiel

- Berechne die Angestellten, die alle Maschinen bedienen können.



Division (Formale Definition)

■ Definition

Seien $r_1 \in \text{REL}(\text{RS}_1)$ und $r_2 \in \text{REL}(\text{RS}_2)$ mit $\text{RS}_1 \supseteq \text{RS}_2$.

$$r_1 \div r_2 := \{t \mid \forall u \in r_2 \exists v \in r_1: t = v[\text{RS}_1 - \text{RS}_2] \wedge u[\text{RS}_2] = v[\text{RS}_2]\}$$

- Welche Attribute sind in dem Schema von $r_1 \div r_2$?

■ Beispiel

$$\begin{array}{c} r_1 \\ \begin{array}{|c|c|} \hline A & B \\ \hline a & 1 \\ \hline a & 2 \\ \hline a & 3 \\ \hline b & 2 \\ \hline b & 3 \\ \hline \end{array} \end{array} \quad \div \quad \begin{array}{c} r_2 \\ \begin{array}{|c|} \hline B \\ \hline 1 \\ \hline 2 \\ \hline \end{array} \end{array} = \begin{array}{|c|} \hline A \\ \hline a \\ \hline \end{array}$$



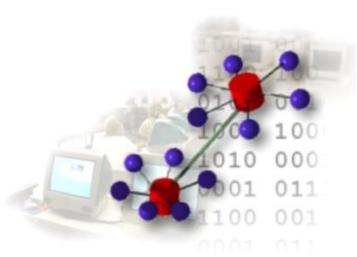
Division (Beispiel)

Das Beispiel zeigt schon die Bedeutung der Division bei der Formulierung allquantifizierter Aussagen:

"Welcher Angestellter ist für **alle** Maschinen ausgebildet?"

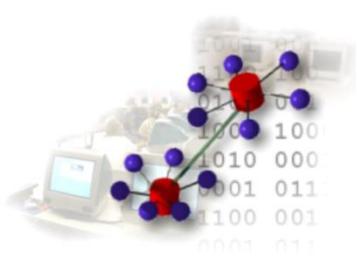
bzgl. dieser Tabellen der ERP-Datenbank:

PMZuteilung			Maschine	
<u>pnr</u>	<u>mnr</u>	Note	<u>mnr</u>	MName
67	84	3	84	Presse
67	93	2	93	Füllanlage
67	101	3	101	Säge
73	84	5		
114	93	5		
114	101	3		
51	93	2		
...				



Eigenschaften der Division

- Man kann die Division als Umkehrfunktion des kartesischen Produkts ansehen.
 - Ist $s = r_1 \div r_2$. Dann gilt nämlich folgende Eigenschaft:
$$s \times r_2 \subseteq r_1 \tag{*}$$
 - Tatsächlich ist s die **größte** Relation, die diese Eigenschaft (*) erfüllt.
- Aus Eigenschaft (*) lässt sich herleiten, wie die Division auf die Grundoperationen der RA zurückgeführt werden kann.



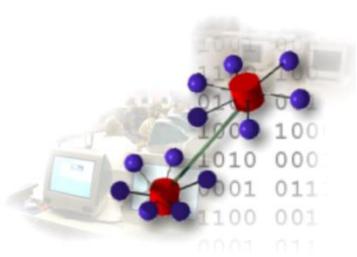
Beispiele (1)

- **Datenbankschema siehe Folie S. 49**
 - Finde alle Namen von Angestellten, die an einer Maschine ausgebildet sind.



Beispiele (2)

- **Datenbankschema siehe Folie S. 49**
 - Finde alle Namen der Angestellten, die an keiner Maschine genügend gut ausgebildet sind (Note ist immer schlechter als 4).



Beispiele (3)

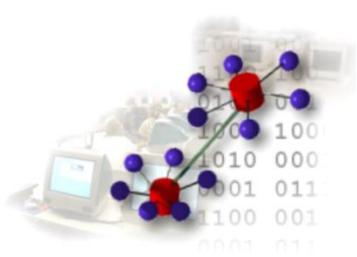
■ Datenbankschema siehe Folie S. 49

- Finde die Namen der Angestellten aus Abteilung “Suppen”, die an der Maschine mit mnr = 93 ausgebildet sind.



Beispiele (4)

- **Datenbankschema siehe Folie S. 49**
 - Finde die Personalnummern der Angestellten, die an der gleichen Maschine ausgebildet sind wie der Angestellte mit pnr = 114.



Zusammenfassung

■ Relationale Algebra

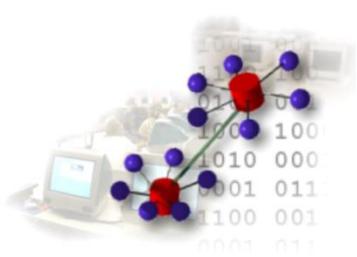
- Menge von 6 Operatoren
 - Selektion, Projektion, Kartesisches Produkt, Vereinigung, Differenz und Umbenennung
 - Jeder Operator berechnet zu einer Relation eine neue (temporäre) Relation

■ Wichtige davon abgeleitete Operatoren

- Join
 - Natürlicher Join, Theta-Join, Semi-Join, Anti-Join
- Division
 - Allquantifizierte Anfragen

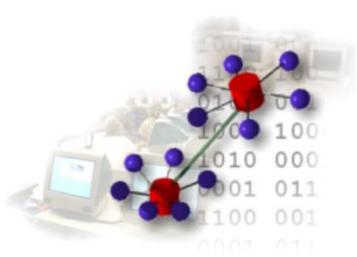
■ Nachteile

- **Imperative Formulierung** durch explizite Vorschrift zur schrittweisen Berechnung → Komplexe aufgebaute Anfragen
- Einschränkung auf die **Mengenverarbeitung**
 - Hoher Aufwand für die Duplikateliminierung
- **Keine Aggregationsmöglichkeiten** für die Daten



2.3 Das Relationenkalkül

- **Bisher**
 - Benutzung einer prozeduralen Anfragesprache
 - Explizite Beschreibung, wie das Ergebnis berechnet wird.
 - Hoher Aufwand bei der Formulierung komplexer Anfragen
- **Zugrundeliegende Idee beim Relationenkalkül (RK)**
 - **Deklarative** Beschreibung der Ergebnisrelation ohne dabei explizit eine Vorschrift für die Konstruktion der Ergebnisrelation zu geben.
- **Zwei bekannte Kalküle**
 - **Tupelkalkül** und Domänenkalkül



Anfragesprachen

- Entwicklung einer Anfragesprache basierend auf dem Relationenkalkül
 - Erheblich einfachere und kompaktere Formulierung komplexer Anfragen.
- SQL
 - Sprache ist eine Mischung zwischen Relationale Algebra (RA) und dem Tupelkalkül (TK)
- Anfragesprachen auf Basis des Relationenkalküls
 - Graphische Anfragesprache **QBE** (Query by Example)
 - Anfragesprache ähnlich zu der von Microsoft Access
 - **Quel** – Anfragesprache im Ingres-System
 - **Datalog**
 - Basiert auf Konzepten der Sprache Prolog
 - Unterstützung von Rekursion



Grundlagen für TK

- **Prädikatenlogik erster Stufe**
 - Konzept der mathematischen Logik → siehe Modul Logik
- **Übertragung auf Datenbanken unter folgenden Annahmen (**Closed World Assumption**):**
 - Tupel in der Datenbank sind **elementare Aussagen**, die wahr sind.
 - Anfragen sind **abgeleitete Aussagen**, die aus anderen Aussagen hergeleitet werden können.
 - Aussagen, die nicht in der Datenbank sind und nicht hergeleitet werden können, **sind falsch!**



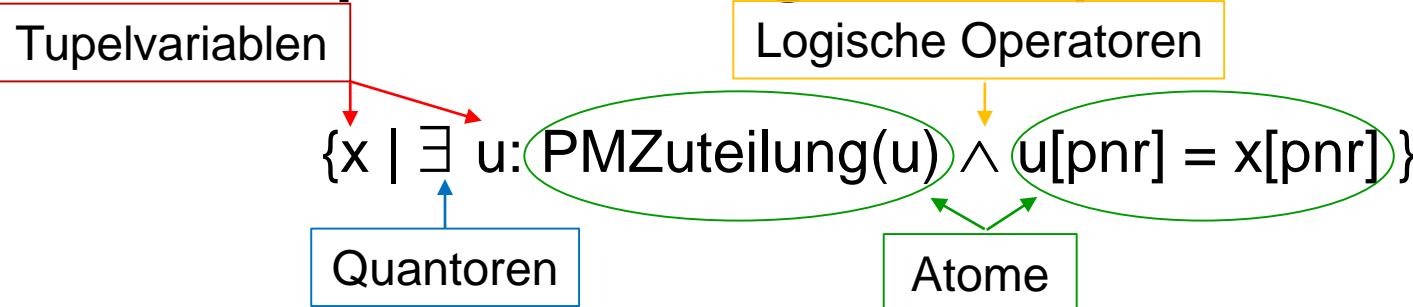
Das Tupelkalkül

■ Tupelkalkül (TK)

- Variablen, die einem Tupel einer Relation entsprechen.
→ **Tupelvariable**

Das Konzept der Tupelvariable ist auch Teil der Anfragesprache SQL.

■ Beispiele für Anfragen im Tupelkalkül



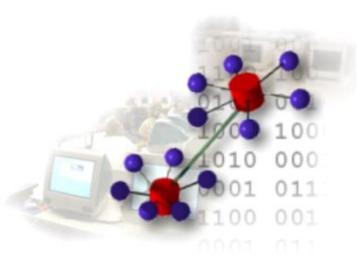
■ Einführung einer Sprache

- Syntax
- Semantik



2.3.1 Syntax des TK

- Eine Formel setzt sich zusammen aus **Atomen** der Form
 - **r(t)**
r ist eine Relation und t ist eine Tupelvariable mit $t \in r$.
 - **t[A] θ u[B]**
t und u sind Tupelvariablen, A und B sind Attribute, θ ein relationaler Operator.
 - **t[A] θ c**
t ist eine Tupelvariable, A ein Attribut, c eine Konstante aus $\text{dom}(A)$ und θ ein relationaler Operator
- **Beispiele:**
 - Personal(t), $t[\text{Note}] > 4$, $t[\text{abtnr}] = u[\text{abtnr}]$



Zusammengesetzte Formeln

- Eine Formel ist entweder

- ein Atom

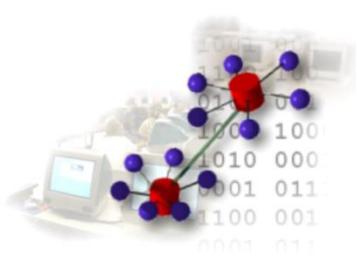
oder rekursiv durch folgende Ausdrücke definiert

(Ann.: *f und g sind Formeln*):

- $f \wedge g$, $f \vee g$, $\neg f$, (f)
 - $\forall t: f(t)$, $\exists t: f(t)$
 - t sind (freie) Variablen in der Formel f

- Beispiele

- $\neg t[\text{Note}] > 4$
 - $(t[\text{pnr}] = u[\text{pnr}]) \vee \neg t[\text{Note}] > 4$
 - $\text{PMZuteilung}(t) \wedge (t[\text{pnr}] = u[\text{pnr}]) \vee \neg t[\text{Note}] > 4$
 - $\exists t: \text{PMZuteilung}(t) \wedge (t[\text{pnr}] = u[\text{pnr}]) \vee \neg v[\text{Note}] > 4$



Freie und gebundene Variablen

■ Informell

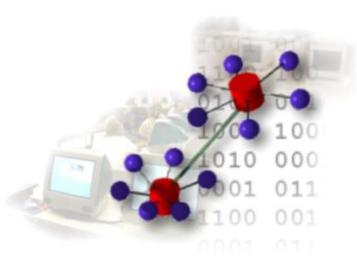
- Durch Angabe eines Quantors direkt vor einer freien Variablen wird diese gebunden.

■ Formale Definition (**freie / gebundene Variablen**)

- Das Auftreten einer Tupelvariablen in einem Atom ist stets frei.
- Für $f := \neg g$ und $f := (g)$ sind alle freien Variablen von g auch frei in f .
- Für $f := g \wedge h$ und $f := g \vee h$ sind die Variablen in f frei, falls sie in g oder h frei sind.
- Sei t eine freie Variable in g . Dann wird durch $f := \forall t: g(t)$, $f := \exists t: g(t)$ t zu einer **gebundenen Variable** in f .

■ Beispiel

$$\forall t: \neg \text{PMZuteilung}(t) \vee t[\text{Note}] > 4$$



Ausdruck im TK

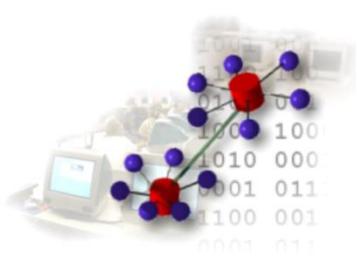
■ Definition

- Ein **Ausdruck im Tupelkalkül** hat die Form

$$\{t \mid f(t)\}$$

wobei t die einzige freie Variable in der Formel f ist.

- Das Schema einer Tupelvariable t ergibt sich implizit dadurch, dass genau die Attribute dazugehören, die von t benötigt werden.
 - Beispiel
 - $\{x \mid \exists u: \text{PMZuteilung}(u) \wedge u[\text{pnr}] = x[\text{pnr}]\}$
Das Schema von v ist {pnr} und von u ist {mnr,pnr,Note}.
 - Man darf auch ein Schema RS explizit zu einer Variable hinzufügen.
$$\{t(\text{RS}) \mid f(t)\}$$



Substitution in einer Formel

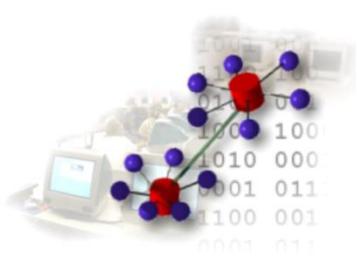
- Sei f eine Formel mit einer **freien Variable x** . Dann ist

$$f(x/t)$$

die **Substitution** der Tupelvariable x in f durch das Tupel t . In jedem Atom, das ein freies Auftreten von x enthält, gehen wir folgendermaßen vor:

- $r(x)$ wird ersetzt durch “wahr”, falls $t \in r$. Andernfalls durch “falsch”.
- $x[A] \theta u[B]$ wird ersetzt durch $c \theta u[B]$ mit $c = t[A]$ (Ann.: $x \neq s$)
- $x[A] \theta c$ wird ersetzt durch “wahr”, falls $t[A] \theta c$ gilt. Andernfalls durch “falsch”.

- Durch Substitution gewinnt man eine Formel, die nur noch die Konstanten “wahr” und “falsch” sowie Atome mit gebundenen Variablen enthält.



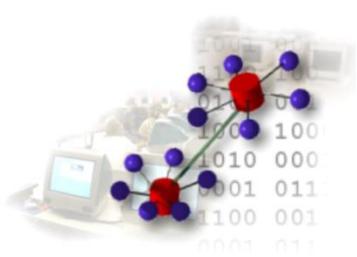
Beispiele

■ Beispiel

- Ein Ausdruck des TK mit Formel:
 $\{t \mid \forall u: \neg \text{PMZuteilung}(u) \vee \neg u[\text{pnr}] = t[\text{pnr}] \vee u[\text{Note}] < t[\text{Note}]\}$
- In der Formel ist Variable u gebunden und Variable t frei.
- Ersetzen von Variable t mit $t[\text{pnr}] = 73$ und $t[\text{Note}] = 5$:
 $\forall u: \neg \text{PMZuteilung}(u) \vee \neg u[\text{pnr}] = 73 \vee u[\text{Note}] < 5$

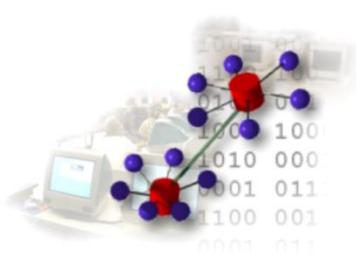
■ Beispiel

- Ein Ausdruck des TK mit Formel:
 $\{u \mid \neg \text{PMZuteilung}(u) \vee \neg u[\text{pnr}] = 73 \vee u[\text{Note}] < 5\}$
- Die Variable u ist frei in der Formel.
- Ersetzen von u durch $u[\text{pnr}] = 51$, $u[\text{mnr}] = 93$ und $u[\text{Note}] = 2$:
 $\neg \text{wahr} \vee \neg \text{falsch} \vee \text{wahr}$



2.3.2 Interpretation einer Formel

- Sei f eine **Formel ohne freie Variablen**. Die **Interpretation** $I(f)$ ist wie folgt definiert:
 - Sei $f = \text{"wahr"}$. Dann ist $I(f) := \text{true}$. Sei $f = \text{"falsch"}$. Dann ist $I(f) := \text{false}$.
 - Sei $f = g$. Dann ist $I(f) := I(g)$.
 - Sei $f = \neg g$. Dann ist $I(f) := \text{true}$ genau dann, falls $I(g) = \text{false}$.
 - Sei $f = g \wedge h$. Dann ist $I(f) := \text{true}$ genau dann, falls $I(g) = \text{true}$ und $I(h) = \text{true}$.
 - Sei $f = g \vee h$. Dann ist $I(f) := \text{true}$ genau dann, falls $I(g) = \text{true}$ oder $I(h) = \text{true}$.
 - Sei $f = \exists x: g(x)$. Dann ist $I(f) := \text{true}$, falls es mindestens ein Tupel t gibt, so dass $I(g(x/t)) = \text{true}$ ist. Andernfalls, $I(f) := \text{false}$
 - Sei $f = \forall x: g(x)$. Dann ist $I(f) := \text{true}$ genau dann, falls für alle t $I(g(x/t)) = \text{true}$ gilt. Andernfalls, $I(f) := \text{false}$.
- Der **Wert des Ausdrucks** $\{ x \mid f(x) \}$ des Tupelkalküls besteht aus allen Tupeln t , die $I(f(x/t)) = \text{true}$ erfüllen.



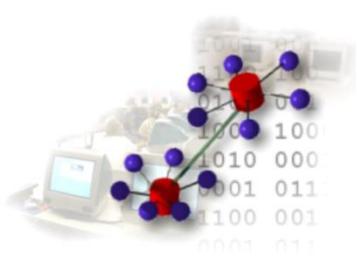
Beispiel (1)

■ Anfrage

- Finde alle Personalnummern von Angestellten, die an einer Maschine ausgebildet sind.
 $\{x \mid \text{PMZuteilung}(x) \wedge x[\text{Note}] < 3\}$

■ Auswertung der Formel

- x ist eine Variable mit dem Schema {pnr, mnr, Note} und der Wertebereich von pnr und mnr ist die Menge der positiven ganzen Zahlen und von Note die Menge {1,2,3,4,5,6}
- Iterative Substitution
 - $x = (1,1,1)$
 - Ist $\text{PMZuteilung}(1,1,1) \wedge 1 < 3$ erfüllt ?
 - $x = \dots$
 -
 - ...
 - $x = (51,93,2)$
 - Ist $\text{PMZuteilung}(51,93,2) \wedge 2 < 3$ erfüllt ?
 - ...



Beispiel (2)

■ Anfrage mit Existenzquantor

- Finde alle Personalnummern von Angestellten, die an einer Maschine ausgebildet sind.

$$\{x \mid \exists u: \text{PMZuteilung}(u) \wedge u[\text{pnr}] = x[\text{pnr}] \}$$

■ Auswertung der Formel

- x ist eine Variable mit dem Schema {pnr} und der Wertebereich von pnr ist die Menge der positiven ganzen Zahlen.
- Iterative Substitution

- x = 1

- Ist $\exists u: \text{PMZuteilung}(u) \wedge u[\text{pnr}] = 1$ erfüllt ?

- x = 2

-

- ...

- x = 51

- Ist $\exists u: \text{PMZuteilung}(u) \wedge u[\text{pnr}] = 51$ erfüllt ?

- ...



Beispiel (3)

Anfrage mit Allquantor

- Finde alle Personalnummern der Angestellten, die an keiner Maschine genügend gut ausgebildet sind.

$$\{x \mid \forall u: \neg \text{PMZuteilung}(u) \vee u[\text{Note}] > 4 \vee \neg u[\text{pnr}] = x[\text{pnr}]\}$$

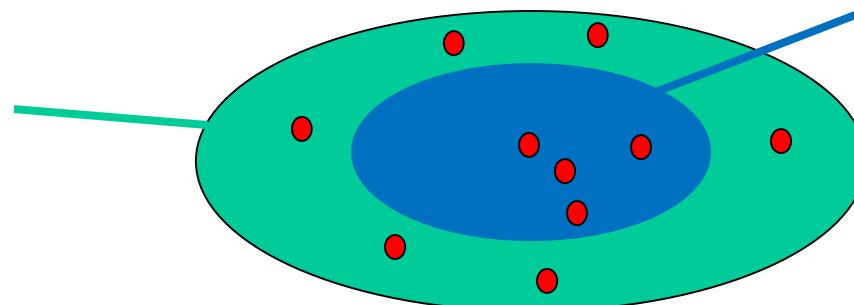
Auswertung der Formel

- x ist eine Variable mit dem Schema {pnr} und der Wertebereich von pnr ist die Menge der positiven ganzen Zahlen.

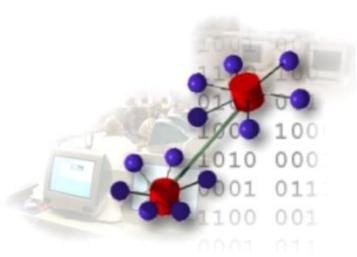
- $x = 1$

 - Ist $\forall u: \neg \text{PMZuteilung}(u) \vee u[\text{Note}] > 4 \vee \neg u[\text{pnr}] = 1$ erfüllt?

alle Werte für u
mit Schema
(pnr,mnr,Note)



alle Tupel aus PMZuteilung



Kurzschreibweisen

■ Einführung von Kurzschreibweisen:

- Relation r und Formel f:

$$\exists t: r(f(t)) := \exists t: r(t) \wedge f(t)$$

- Relation r und Formel f:

$$\forall t: r(f(t)) := \forall t: \neg r(t) \vee f(t)$$

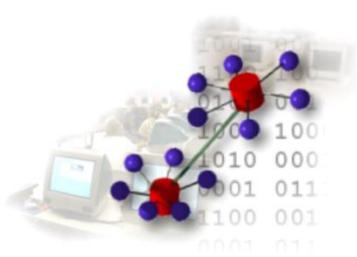
- Formeln f, g:

$$f \Rightarrow g := \neg f \vee g$$

■ Beispiel

- Berechne die pnr der Angestellten, die an keiner Maschine genügend gut ausgebildet sind.

$$\{ u \mid \exists t: \text{PM-Zuteilung}(u[\text{pnr}] = t[\text{pnr}] \wedge \\ \forall v: \text{PM-Zuteilung}(v[\text{pnr}] = t[\text{pnr}] \Rightarrow v[\text{Note}] > 4) \\ \} \\)$$



Ausdrucksstärke der Sprache

■ Satz

- Das Tupelkalkül (TK) ist ausdrucksstärker als die RA.

■ Beweisidee

- Zeige für die Operatoren der relationalen Algebra, dass diese sich durch TK ausdrücken lassen.
 - Projektion, Filter, Vereinigung, Differenz, Kartesisches Produkt, Umbenennung
- Finde eine Anfrage, die sich in TK ausdrücken lässt, aber nicht in RA.
 - $\{ t \mid \neg \text{PMZuteilung}(t) \}$



TK und RA (1)

- Alle Ausdrücke der **Relationenalgebra** lassen sich äquivalent auch durch Anfragen im **Tupelkalkül** ausdrücken:
- Projektion:

RA

TK

$\pi_A(r)$

$\{t \mid \exists u: r(u) \wedge t = u[A]\}$

- Selektion:

$\sigma_F(r)$

$\{v \mid r(v) \wedge F(v)\}$

- Vereinigung (Durchschnitt, Differenz und Produkt analog):

$r \cup s$

$\{v \mid r(v) \vee s(v)\}$



TK und RA (2)

- Natürlicher Verbund
 - Gegeben zwei Relationen $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$.

$$r_1 \bowtie r_2 = \{t \mid t[\text{RS}_1] \in r_1 \text{ und } t[\text{RS}_2] \in r_2\}$$

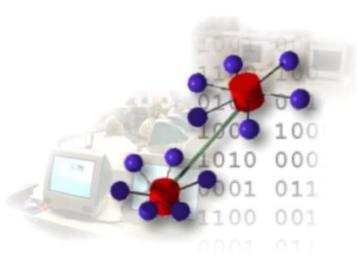
- Division
 - Gegeben zwei Relationen $r_1 \in \text{REL}(\text{RS}_1)$, $r_2 \in \text{REL}(\text{RS}_2)$ mit $\text{RS}_1 \supseteq \text{RS}_2$

$$\begin{aligned} r_1 \div r_2 = \{t \mid & \forall u \in r_2 \exists v \in r_1: \\ & t[v[\text{RS}_1 - \text{RS}_2]] \wedge u[\text{RS}_2] = v[\text{RS}_2]\} \end{aligned}$$



Praktische Relevanz ?

- Es können im TK Anfragen mit unendlich groÙe Ergebnismenge formuliert werden.
 - Benutzer sind an solchen Ergebnissen nicht interessiert.
 - Solche Anfragen sind nicht effizient zu berechnen.
 - Leider ist das Problem, ob eine Anfrage eine endliche Ergebnismenge liefert, **nicht entscheidbar**.



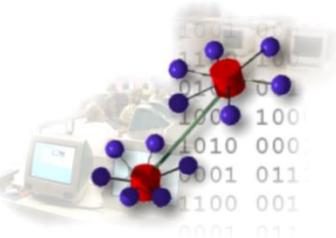
Sichere Ausdrücke

- Syntaktische Einschränkung auf Anfragen mit endlicher Ergebnismenge.
 - Diese Anfragen werden als **sichere Ausdrücke** bezeichnet.
 - Definition ist relativ komplex
→ siehe das Buch „Datenmodelle ...“ von Vossen
 - Sichere Ausdrücke kann man durch Anbinden der Tupelvariablen an eine Relation r erhalten.
 - $\forall t: r(f(t))$
 - $\exists t: r(f(t))$

Dabei muss f ebenfalls eine sichere Formel sein.

Satz

- Das TK eingeschränkt auf sichere Ausdrücke hat die gleiche Ausdrucksstärke wie RA.



RA und SQL

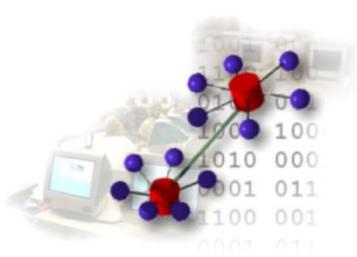
- **SQL:** Prinzipien werden genauer im nachfolgenden Kapitel behandelt.
- Basiskonzept 'SELECT-FROM_WHERE'-Block

```
SELECT ID, Kfz, Einwohner, Land  
FROM stadt , stadt_in_land  
WHERE Einwohner >= 1000 AND Land = 'BY';
```

- Dieser Block kann direkt in ein Ausdruck der RA überführt werden:


$$\pi_{ID, Kfz, Einwohner, Land} (\sigma_{Einwohner \geq 1000 \wedge Land = 'BY'} (stadt \times stadt_in_land))$$

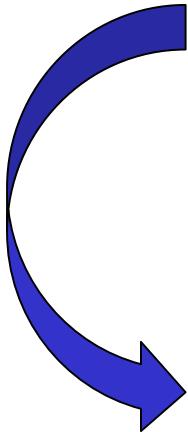
- Weitere "RA-Erbschaft": Operatoren UNION, INTERSECT und MINUS zur Verknüpfung von SFW-Blöcken und von komplexeren Anfragen



TK und SQL

- Aber SQL kann auch als **TK-Sprache** bezeichnet werden.
- Korrespondenz zum TK beim Grundkonzept SFW-Block:

```
SELECT ID, Kfz, Einwohner, Land  
FROM   stadt x, stadt_in_land y  
WHERE  x.Einwohner >= 1000 AND y.Land = 'BY';
```



```
{x| stadt(x) ∧ stadt_in_land(y) ∧  
     x[Einwohner] ≥ 1000 ∧ y[Land] = 'BY' }
```

- Mögliche Verwendung von **Quantoren** im WHERE-Teil einer SQL-Anfrage zeigt, dass wirklich TK gemeint ist und nicht etwa die (syntaktisch ähnlichen) Selektionsbedingungen der RA.



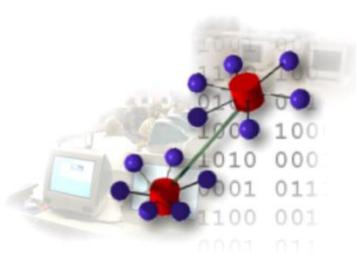
2.4 Erweiterung der relationalen Algebra

- Viele interessante Anfragen sind leider weder in RA noch in TK ausdrückbar.
 - Beispiele:
 - Berechne zu jeder Maschine die Anzahl von Personen, welche die Maschine bedienen können.
 - Liefere die Ergebnisse sortiert.
- Was ist unser Problem?
 - In der Mengensemantik gibt es **keine Ordnung**.
 - **Verdichten** der Datensätze ist nicht möglich.
 - **Erhaltung von Duplikaten** wird nicht unterstützt.



Anforderung für zusätzliche Funktionen

- Bewahrung von Duplikaten (z. B. die durch Projektion entstehen)
 - Unterstützung von **Multimengen** (engl.: bags)
- Verdichtung der Daten einer Relation durch Aggregation
 - Zusätzliche Operatoren in der relationalen Algebra
- Unterstützung für das Sortieren der Daten
 - Unterstützung von Folgen statt nur Mengen



2.4.1 M-Relationen

■ Informelle Vorgehensweise

- In jeder Relation speichern wir uns zu jedem Tupel noch zusätzlich die Vielfachheit des Tupels ab.
- Operationen der RA müssen noch erweitert, so dass jetzt die **Multimengensemantik** unterstützt wird.

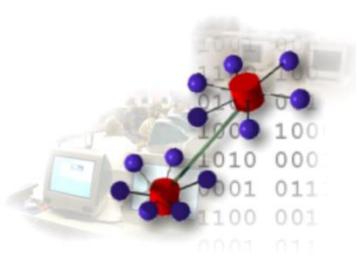
■ Formale Definition

- Sei $V \notin U$ (U ist das Universum der Attribute) mit $\text{dom}(V) = \mathbb{N}$. Zu einem Relationenschema RS ist die **M-Relation** $r = r(RS)$ eine endliche Menge von totalen Abbildungen t mit

$$t: RS \cup \{V\} \rightarrow \bigcup_{A \in RS} \text{dom}(A) \cup \mathbb{N}$$

und $t(A) \subseteq \text{dom}(A)$ für alle $A \in RS$ und $t(V) \in \mathbb{N}$.

- Im Folgenden bezeichnet **MREL(RS)** die Menge aller M-Relationen über dem Schema RS .



Kartesisches Produkt

■ Informell

- Die Häufigkeit im Ergebnis ergibt sich durch das Produkt der Häufigkeiten in den beiden M-Relationen.

■ Formale Definition

- Seien $r_1 \in MREL(RS_1)$, $r_2 \in MREL(RS_2)$ zwei M-Relationen mit $RS_1 \cap RS_2 = \emptyset$. Die Ausgabe von $r_1 \times r_2$ ist eine (temporäre) Relation s mit

- $REL_s = RS_1 \cup RS_2$
 - $s = \{ t \mid t[RS_1] \in r_1[RS_1] \text{ und } t[RS_2] \in r_2[RS_2] \text{ und } t[V] = r_1[V]^*r_2[V]\}$



Kartesisches Produkt

■ Eingabe

- Zwei M-Relationen r und s.

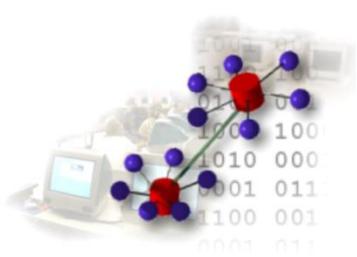
■ Ausgabe

- Eine M-Relation, in der jedes Tupel aus r mit jedem Tupel aus s verknüpft ist.

r	A	B	V
			2
			3

s	C	D	V
			5
			7

$r \times s$	A	B	C	D	V
					10
					14
					15
					21



Summenvereinigung

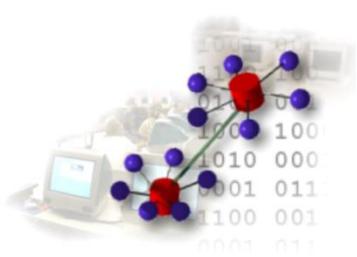
■ Informell

- Die Zeilen von zwei Relationen werden vereinigt und die Häufigkeit ergibt sich aus der Summe der Einzelhäufigkeiten

■ Formale Definition

- Seien $r_1, r_2 \in MREL(RS)$ zwei Relationen über dem gleichen Schema RS. Die Ausgabe von $r_1 \cup r_2$ ist eine (temporäre) Relation s mit
 - $REL_s = RS$
 - $s = \{t \mid t[RS] \in r_1 \text{ oder } t[RS] \in r_2 \text{ und } t[V] = r_1[V] + r_2[V]\}$

Multimengensemantik!



Vereinigung (Beispiel)

■ Eingabe

- Zwei M-Relationen r und s mit gleichem Schema.

■ Ausgabe

- Eine M-Relation, in der jedes Tupel aus r und jedes Tupel aus s vorkommt (Vielfachheit: Summe).

r	A	B	V
		red bar	2
		green bar	3

s	A	B	V
		blue bar	5
		red bar	7

$r \cup s$	A	B	V
		red bar	9
		green bar	3
		blue bar	5



Verallgemeinerung der Projektion

■ Informell

- Statt einer Projektion sollen beliebige Funktionen erlaubt sein, die ein Tupel in ein anderes Tupel überführen.
→ Map-Operator funktionaler Sprachen

■ „Formale“ Definition

- Sei RS ein Schema und $r \in MREL(RS)$ eine M-Relation. Weiterhin sei X ein Relationenschema und $\mu: RS \rightarrow X$ eine Funktion, die zu einem Tupel aus r ein Tupel mit dem Schema X erzeugt. *Dann gilt:*
 - $RS_s = X$
 - $s = \{ t \mid t[X] = \mu(u) \text{ mit } u \in r \text{ und } t[V] = \sum_{t[X]=\mu(u) \wedge u \in r} u[V] \}$



Projektion mit Duplikaterhaltung

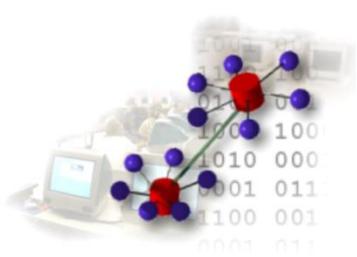
- Relation PMZuteilung projiziert auf Attribut pnr
 - Man beachte dabei, dass die Definition sich auf mengenwertige Relationen direkt übertragen lässt.

PMZuteilung

<u>pnr</u>	<u>mnr</u>	Note
67	84	3
67	93	2
67	101	3
73	84	5
114	93	5
114	101	3
51	93	2
69	101	2
333	84	3
701	84	2
701	101	2
82	101	2

$\pi_{pnr}(\text{PMZuteilung})$

<u>pnr</u>	V
67	3
73	1
114	2
51	1
69	1
333	1
701	2
82	1



Projektion mit arithmetischem Ausdruck

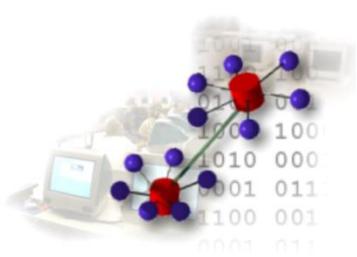
■ Relation PMZuteilung auf Attribut pnr.

- Umwandlung Note in Notenpunkte mit folgender Formel
 - $\text{return } (\text{Note} == 6) ? 0 : 14 - (\text{Note} - 1) * 3$

PMZuteilung		
<u>pnr</u>	<u>mnr</u>	Note
67	84	3
67	93	2
67	101	3
73	84	5
114	93	5
114	101	3
51	93	2
69	101	2
333	84	3
701	84	2
701	101	2
82	101	2

$\pi_{\text{pnr}, \text{mnr}, \text{np} \leftarrow (\text{note} == 6) ? 0 : 14 - (\text{note} - 1) * 3} (\text{PMZuteilung})$

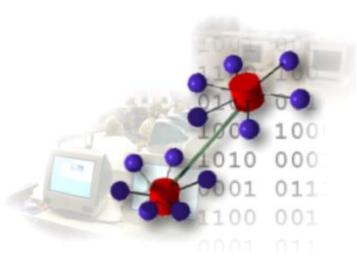
<u>pnr</u>	<u>mnr</u>	np	V
67	84	8	1
67	93	11	1
67	101	8	1
73	84	2	1
114	93	2	1
114	101	8	1
51	93	11	1
69	101	11	1
333	84	8	1
701	84	11	1
701	101	11	1
82	101	11	1



Multimengen in Flink



- Die Klasse Table unterstützt sowohl Mengen als auch Multimengen.
 - Es gibt zusätzliche Operatoren für die Berechnung von Multimengen.
 - Table unionAll(Table right)
 - Diese Methode implementiert die Summenvereinigung.
 - Table minusAll(Table right)
 - Diese Methode implementiert die Summendifferenz.
 - Projektionsoperator select liefert eine Multimenge (ohne Duplikateliminierung)
 - Sollen die Ergebnisse ohne Duplikate geliefert werden, muss noch zusätzlich der Operator Table duplicate() angewendet werden.
 - Zudem kann mit as ein Ausdruck an einen neuen Attributnamen gebunden werden.
 - Beispiele
 - Berechne alle Personen, die eine Maschine bedienen können.
`pmz.select("pn").distinct();`
 - Berechne eine um 1 bessere Note in der Tabelle pmz.
`Pmz.filter("Note > 1").select("Note-1 as nn");`



4.4.2 Aggregation

■ Ziel

- Berechnung wichtiger Kennzahlen einer Multimenge/Menge
 - Summe (*sum*), Durchschnitt (*avg*), Anzahl (*count*), Minimum (*min*) und Maximum (*max*) von Bedeutung.

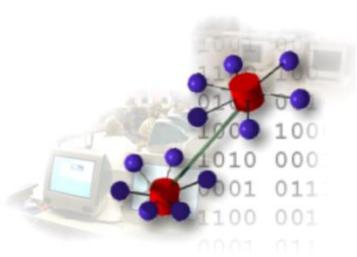
■ Zwei Arten der Aggregation

- Skalare Aggregation
 - Berechnung eines Wertes für eine Multimenge/Menge
- Vektoraggregate
 - Berechnung eines Vektors mit Aggregaten für eine Multimenge/Menge.
 - Dies erfordert zunächst eine disjunkte Aufteilung in Gruppen.



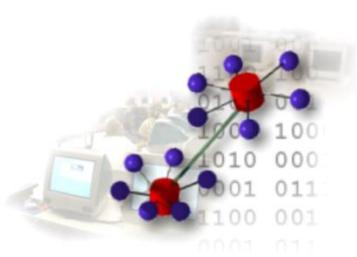
Skalare Aggregate

- Eine Aggregationsfunktion $\text{agg}: MREL(RS) \rightarrow D$ berechnet zu einer M-Relation $r \in MREL(RS)$ einen Wert aus einem Wertebereich D .
 - Die Funktion $\text{count}(r)$ liefert die Anzahl der Tupel in r .
 - Aggregationsfunktionen $\text{avg}_A(r)$, $\text{sum}_A(r)$, $\text{count}_A(r)$ liefern einen numerischen Wert zu einem Attribut $A \in RS$.
 - Aggregationsfunktionen $\text{min}_A(r)$, $\text{max}_A(r)$ den kleinsten bzw. größten Wert zu dem Attribut $A \in RS$.
- Das Ergebnis dieser Operation ist keine Tabelle, sondern ein skalarer Wert.



Beispiele

- Berechne die Anzahl der Tupel in der Relation PMZuteilung.
 - $\text{count}(\text{PMZuteilung})$
- Berechne die Anzahl der Angestellten, die eine Maschine bedienen können.
 - $\text{count}_{\text{pnr}}(\text{PMZuteilung})$
- Berechne die durchschnittlichen Noten der Mitarbeiter
 - $\text{avg}_{\text{Note}}(\text{PMZuteilung})$



Skalare Aggregate in Flink

■ Unterstützung von Aggregaten in der Projektion

- Zur Erinnerung
 - Die Methode select unterstützt als Argumente eine Liste von Attributnamen.
- Zusätzlich kann jetzt noch ein skalares Aggregat angegeben werden.
 - Hierbei ist es zwingend erforderlich mit "as" noch ein Attributname der Ausgabe zu definieren.
 - Beispiele
 - `Table res = pmz.select("avg(Note) as a");`
 - `Table res = pmz.select("sum(Note) as b");`



Vektoraggregate

- Zunächst wird die Tabelle in eine möglichst kleine Anzahl von Gruppen aufgeteilt.
 - Die Aufteilung erfolgt an Hand eines oder mehrerer Attribute der Relation.

1. Partitionierung von PMZuteilung unter Verwendung von {pnr} in Gruppen.

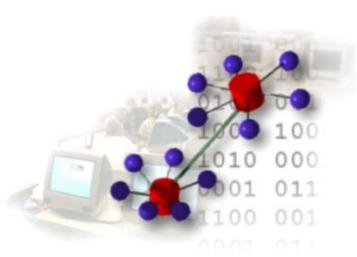
- In einer Gruppe sind die Tupel, die in den ausgewählten Attributen gleich sind.

pnr	mnr	Note	v
67	84	3	1
67	93	2	1
67	101	3	1
73	84	5	1
114	93	5	1
114	101	3	1
51	93	2	1
69	101	2	1
333	84	3	1
701	84	2	1
701	101	2	1
82	101	2	1

2. Danach wird für jede Gruppe unter Verwendung eines Aggregats ein Tupel erzeugt.

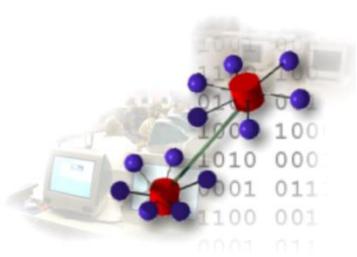
pnr	avg(Note)
67	8/3
73	5
114	4
51	2
69	2
333	3
701	2
82	2

Das Ergebnis ist eine normale Relation!



Formale Definition

- Eine **Gruppe** ist eine Relation, die das gleiche Schema wie die Eingaberelation hat.
 - Sei $X \subseteq RS$ und $r \in MREL(RS)$. Für zwei Tupel $t_1, t_2 \in r$ mit $t_1[X] = t_2[X]$ gilt, dass diese in der gleichen Gruppe von r liegen.
- Zu einer Aggregation (über einem ausgezeichneten Attribut) wird nun **für jede Gruppe eine Kennzahl** berechnet. Diese Kennzahl wird zusammen mit den Werten der Gruppierungsattribute in der Ergebnisrelation eingetragen.
 - Man kann auch mehrere Aggregate pro Gruppe berechnen.
- Operator: $\gamma_{X, A1 \leftarrow agg1, A2 \leftarrow agg2, \dots, An \leftarrow aggN}(r)$



Beispiele

- Berechne für jede Maschine die Anzahl der Angestellten, welche die Maschine bedienen können.

$\gamma_{mnr, c \leftarrow \text{count}()}$ (**PMZuteilung**)

- Berechne für jeden Angestellten seine durchschnittliche Note eine Maschine zu bedienen.

$\gamma_{pnr, c \leftarrow \text{avg}(Note)}$ (**PMZuteilung**)

- Berechne den Notenspiegel.



Vektoraggregate in Flink



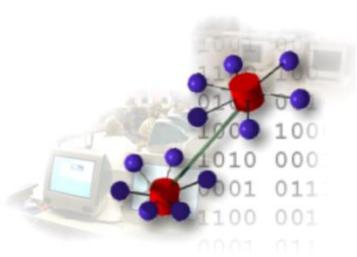
■ Beispiel

- Table counts = orders

```
.groupBy("a")
.select("a, b.count as cnt");
```

■ Erklärungen

- In der groupBy-Methode wird das Gruppierungsattribut bzw. die Gruppierungsattribute angegeben.
- Danach **muss** eine select-Methode folgen, in der für jede Gruppe genau ein Tupel berechnet wird.
 - Dabei **muss** das Gruppierungsattribut in der Liste sein.
 - Ansonsten können noch für die anderen Attribute Aggregate berechnet werden.
 - Das Aggregat wird ähnlich wie ein Methodenaufruf an das Attribut gehängt.



Duplikatbeseitigung

- Bei der Duplikatbeseitigung transformiert man eine M-Relation in eine normale Relation.
 - Streichen der Spalte V.
- Betrachten wir hierzu die M-Relation von Folie 132, die durch $\pi_{\text{pnr}}(\text{PMZuteilung})$ erzeugt wurde.
 - Durch Beseitigung der Duplikate erzeugen wir eine mengenwertige Relation mit einer Spalte.
- Der Operator bekommt wegen seiner Bedeutung das Symbol δ .

$\delta(\pi_{\text{pnr}}(\text{PMZuteilung}))$

pnr

67

73

114

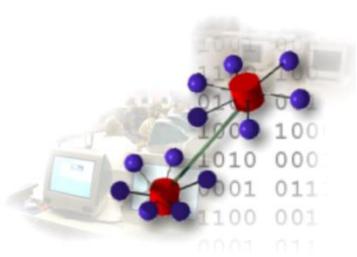
51

69

333

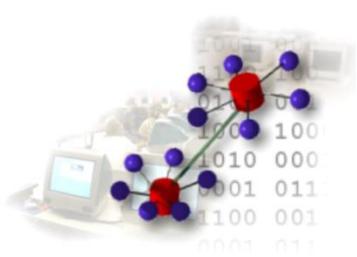
701

82



Sortieren

- Zusätzlich gibt es den **Sortieroperator τ** , der die Tupel einer Relation / M-Relation als Folge ausgibt.
 - Hierzu wird eine Liste $L = (A_1, A_2, \dots, A_k)$ mit Attributen aus dem Schema der Relation verwendet. Die Tupel werden lexikographisch bezgl. der Attribute sortiert.
- Beispiel
 - $\tau_{\text{Note, pnr}}(\text{PMZuteilung})$
- Der Sortieroperator wird bei einer Anfrage immer zuletzt angewendet.
 - Danach kann also kein weiterer Operator folgen.



Sortieren in Flink



■ Beispiel

- Table t = pmz.orderBy("mnr, Note.asc");

- In der Methode orderBy wird als Parameter noch die Liste der Sortierattribute und ggf. noch mitgeteilt, ob absteigend sortiert werden soll.

- **Zusätzlich kann noch angegeben werden, ab welcher Position und wie viele Tupel in der sortierten Folge geliefert werden sollen.**
 - Table t = pmz.orderBy("mnr, Note.asc").fetch(5);
 - Table t = pmz.orderBy("mnr, Note.asc").offset(3);



Zusammenfassung

■ RA & TK

- Verarbeitung von Mengen
- Keine Berechnung von Aggregaten

■ Erweiterung der RA

- Unterstützung einer Multimengensemantik
 - Neudefinition der Operationen der RA
- Unterstützung von weitergehenden Operationen
 - Aggregatberechnung
 - Partitionierung in Gruppen
 - Sortieren