Moduldokumentation

Modul Distributed Data Management (ddm)

Simon Wächter

2019

Inhalt

[1 Einleitung 4](#_Toc2708561)

[1.1 Einleitung 4](#_Toc2708562)

[1.2 Lernziele 4](#_Toc2708563)

[1.3 Prüfungen 4](#_Toc2708564)

[2 Woche 1 5](#_Toc2708565)

[2.1 Gliederung 5](#_Toc2708566)

[2.2 Fachvertiefung 5](#_Toc2708567)

[2.3 Verteiltes Rechnen 5](#_Toc2708568)

[2.3.1 Verteiltes Rechnen (Distributed Computing) 5](#_Toc2708569)

[2.3.2 Was wird verteilt 6](#_Toc2708570)

[2.4 Einordnung der Thematik 6](#_Toc2708571)

[2.4.1 Verteilt, mobil, parallel … 6](#_Toc2708572)

[2.4.2 … Big Data 6](#_Toc2708573)

[2.5 Verteilte Datenbanksysteme 7](#_Toc2708574)

[2.5.1 Verteilte Datenbanksysteme 7](#_Toc2708575)

[2.5.2 Klassifizierung 7](#_Toc2708576)

[2.5.3 DDBS (P2P) Architecture 8](#_Toc2708577)

[2.5.4 MDBS Architecture 9](#_Toc2708578)

[2.5.5 Date’s 12 Regeln 9](#_Toc2708579)

[2.5.6 Aspekte Verteilter Datenbanksysteme 10](#_Toc2708580)

[2.6 Parallele Datenbanksysteme 10](#_Toc2708581)

[2.6.1 Parallele Datenbanksysteme 10](#_Toc2708582)

[2.6.2 Architekturen 10](#_Toc2708583)

[2.6.3 Aspekte paralleler Datenbanksysteme 11](#_Toc2708584)

[2.7 NoSQL Systeme 11](#_Toc2708585)

[2.7.1 NoSQL Systeme 11](#_Toc2708586)

[2.7.2 Konzepte NoSQL Systeme 11](#_Toc2708587)

[2.8 Mobile Datenbanksysteme 12](#_Toc2708588)

[2.9 Semantic Web 12](#_Toc2708589)

[2.10 Gliederung 13](#_Toc2708590)

[2.11 Was sind Trigger? 13](#_Toc2708591)

[2.12 Wozu dienen Trigger? 13](#_Toc2708592)

[2.13 Trigger Konzept 14](#_Toc2708593)

[2.14 Triggertypen 14](#_Toc2708594)

[2.14.1 Triggertypen: Ereignis 14](#_Toc2708595)

[2.14.2 Triggertypen: Timing 14](#_Toc2708596)

[2.14.3 Triggertypen: Granulat 15](#_Toc2708597)

[2.15 DML-Trigger auf Tabellen 15](#_Toc2708598)

[2.16 PL/SQL Block 16](#_Toc2708599)

[2.16.1 Einfacher PL/SQL Block 16](#_Toc2708600)

[2.16.2 Bildschirmausgabe 16](#_Toc2708601)

[2.16.3 Datentypen 17](#_Toc2708602)

[2.16.4 Anweisung 17](#_Toc2708603)

[2.17 1. Erfahrung mit Trigger 18](#_Toc2708604)

[2.17.1 Beispiel-Datenbasis 18](#_Toc2708605)

[2.17.2 1. Erfahrung mit Trigger 19](#_Toc2708606)

[2.18 Konsistent Prüfen 20](#_Toc2708607)

[3 Woche 2 22](#_Toc2708608)

[3.1 Inhalt 22](#_Toc2708609)

[3.2 Einführung 22](#_Toc2708610)

[3.2.1 Ausganglage 22](#_Toc2708611)

[3.2.2 Datenbank Beispiel 22](#_Toc2708612)

[3.2.3 Verteilte Anwendungen 23](#_Toc2708613)

[3.2.4 Entwurfsproblem 23](#_Toc2708614)

[3.2.5 Alternative Entwurfsstrategien 24](#_Toc2708615)

[3.2.6 Top-down Entwurf 24](#_Toc2708616)

[3.3 Verteilungsentwurf 25](#_Toc2708617)

[3.3.1 Kernpunkte des Verteilungsentwurfs 25](#_Toc2708618)

[3.3.2 Grad der Fragmentierung 25](#_Toc2708619)

[3.3.3 Arten der Fragmentierung 25](#_Toc2708620)

[3.3.4 Möglichkeit der Fragmentierung – Horizontal 26](#_Toc2708621)

[3.3.5 Möglichkeit der Fragmentierung – Vertikal 27](#_Toc2708622)

[3.3.6 Korrektheit der Fragmentierung 27](#_Toc2708623)

[3.4 PHF 28](#_Toc2708624)

[3.4.1 PHF – Information über Daten 28](#_Toc2708625)

[3.4.2 PHF – Information über Anwendungen 28](#_Toc2708626)

[3.4.3 Minterm Beispiele 29](#_Toc2708627)

[3.4.4 PHF – Information über Anwendungen 29](#_Toc2708628)

[3.5 Verfahren 29](#_Toc2708629)

[3.5.1 PHF Verfahren, 1. Schritt 29](#_Toc2708630)

[3.5.2 Vollständigkeit und Minimalität 30](#_Toc2708631)

[3.5.3 PHF – Verfahren, 2. Schritt 30](#_Toc2708632)

[3.5.4 PHF – Verfahren, 3. Schritt 30](#_Toc2708633)

[3.5.5 PHF – Beispiel 31](#_Toc2708634)

[3.6 DHF 32](#_Toc2708635)

[3.6.1 Abgeleitete horizontale Fragmentierung 32](#_Toc2708636)

[3.6.2 DHF - Beispiel 33](#_Toc2708637)

[3.7 Tafelmitschrift 33](#_Toc2708638)

[4 Woche 3 35](#_Toc2708639)

[4.1 Inhalt 35](#_Toc2708640)

[4.2 Vertikale Fragmentierung 35](#_Toc2708641)

[4.2.1 Vertikale Fragmentierung 35](#_Toc2708642)

[4.2.2 Datenbank Beispiel 36](#_Toc2708643)

[4.3 Zugriffseigenschaften 36](#_Toc2708644)

[4.3.1 VF – Zugriffseigenschaft 36](#_Toc2708645)

[4.4 Cluster Methode 38](#_Toc2708646)

[4.4.1 VF – Cluster Methode 38](#_Toc2708647)

[4.4.2 VF – Affinität 38](#_Toc2708648)

[4.4.3 VF – Affinitätsmatrix 39](#_Toc2708649)

[4.5 Bond Energy Algorithm 39](#_Toc2708650)

[4.5.1 VF – Bond Energy Algorithm 39](#_Toc2708651)

[4.5.2 VF- Beispiel 40](#_Toc2708652)

[4.6 Splitting 42](#_Toc2708653)

[4.6.1 VF – mögliche Cluster 42](#_Toc2708654)

[4.6.2 VF – Splitting 42](#_Toc2708655)

[4.6.3 VF – Beispiel 43](#_Toc2708656)

[4.6.4 Resultierende Fragment 43](#_Toc2708657)

[4.6.5 VF – Rotation 43](#_Toc2708658)

[4.7 Notizen 44](#_Toc2708659)

[5 Woche 4 45](#_Toc2708660)

# Einleitung

## Einleitung

Dieses Dokument stellt die Moduldokumentation für das Modul ddm dar. Allfällige Unterlagen sind im Modulordner zu finden.

## Lernziele

Skalierbarkeit, Verfügbarkeit und Ausfallstoleranz bilden Schlüsselanforderungen an heutige verteilte Informationssysteme. In diesem Modul vertiefen sich die Studierenden in die grundlegenden Konzepte und Technologien verteilter Datenhaltung und -verarbeitung. Dabei wird sowohl auf die klassischen Ansätze verteilter Datenbanksysteme eingegangen als auch neuartige Paradigmen der NoSQL Systeme besprochen.

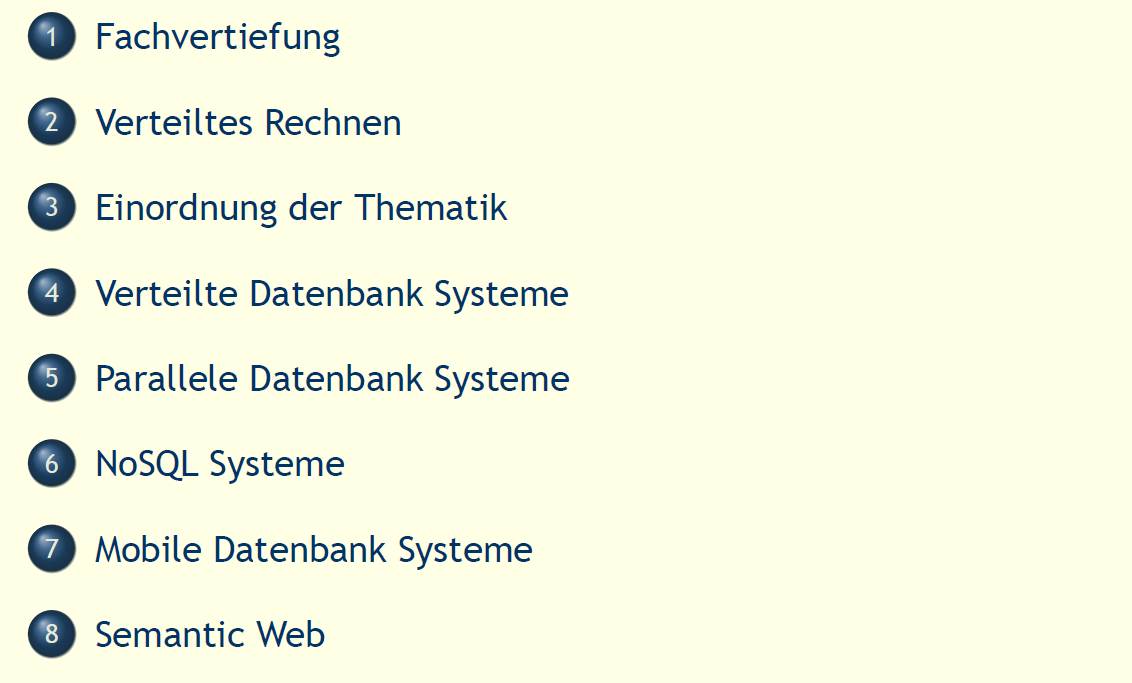
* Aktive Datenbanken:
  + Stored Procedures
  + ECA Prinzip
  + Trigger
* Verteilte Datenbanksysteme:
  + Architektur verteilter Datenbanksysteme
  + verteilter Datenentwurf
  + verteilte Anfrageverarbeitung
  + verteilte Transaktionen
  + Replikation
* NoSQL:
  + Daten-, Speicher- und Anfragemodelle
  + Key/Value, Wide Column, Document, Graph
  + Map/Reduce
  + Partitionierung (Sharding), verteilte Hashtabellen
  + Konsistenz- und Replikationsmodelle
  + CAP Theorem, BASE
* NoSQL Systeme: Cassandra, MongoDB, HBase, Neo4j

## Prüfungen

Die Modulnote setzt sich aus einer Erfahrungsnote zu 50% mit zwei Assessments zu je 25% und einer Modulschlussprüfung zu 50% zusammen.

# Woche 1

## Gliederung

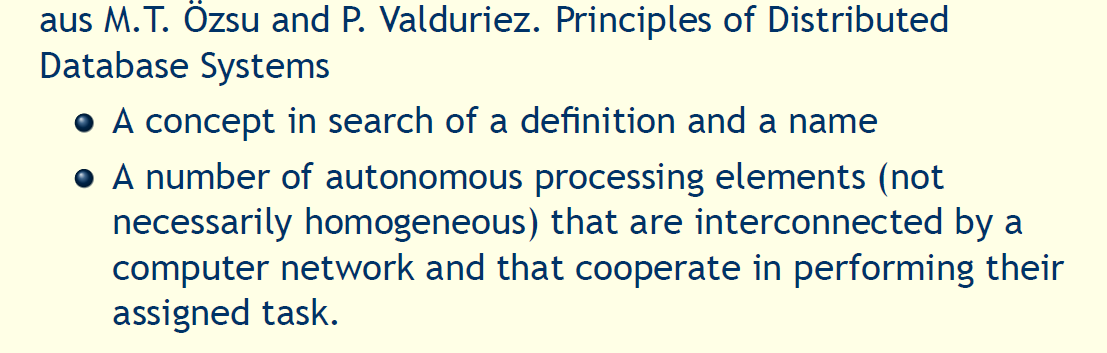


## Fachvertiefung



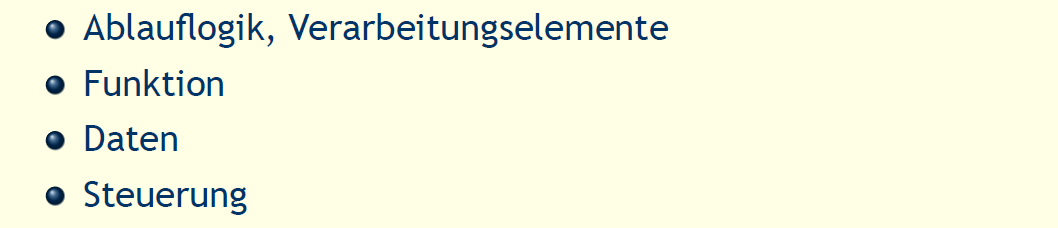
## Verteiltes Rechnen

### Verteiltes Rechnen (Distributed Computing)



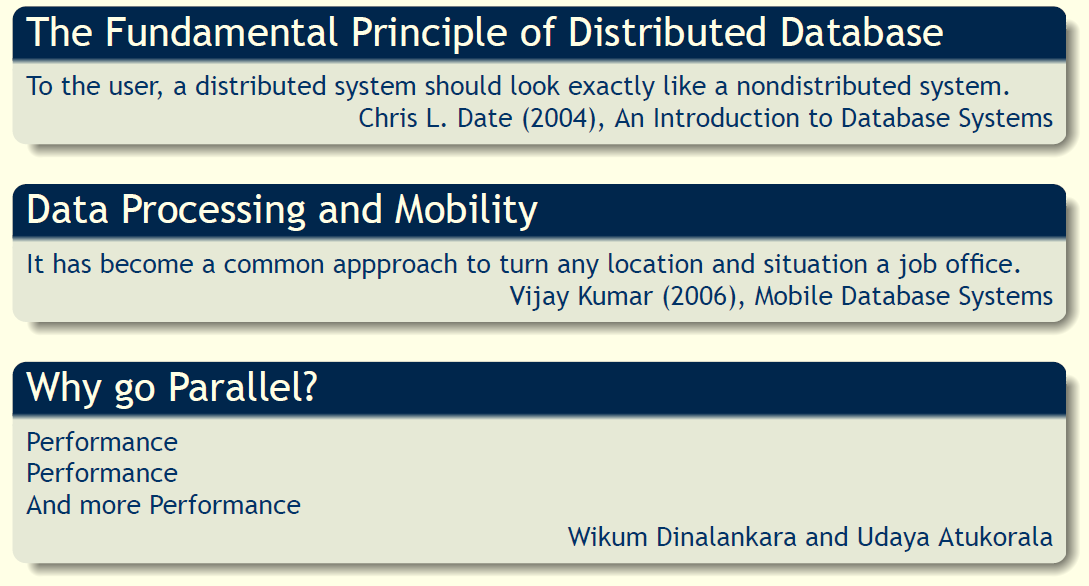
In verteilten Systemen kann die Ablauflogik, Daten, Steuerung verteilt sein. Für einen Benutzer verhält sich ein verteiltes System wie ein nicht verteiltes System. Warum sollte man Parallelisieren? -> Performance, Performance, Performance

### Was wird verteilt

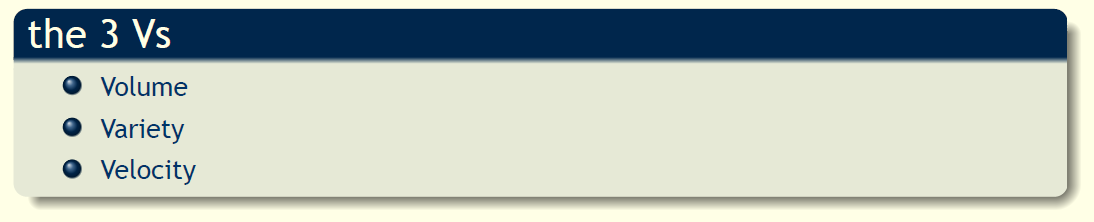


## Einordnung der Thematik

### Verteilt, mobil, parallel …



### … Big Data

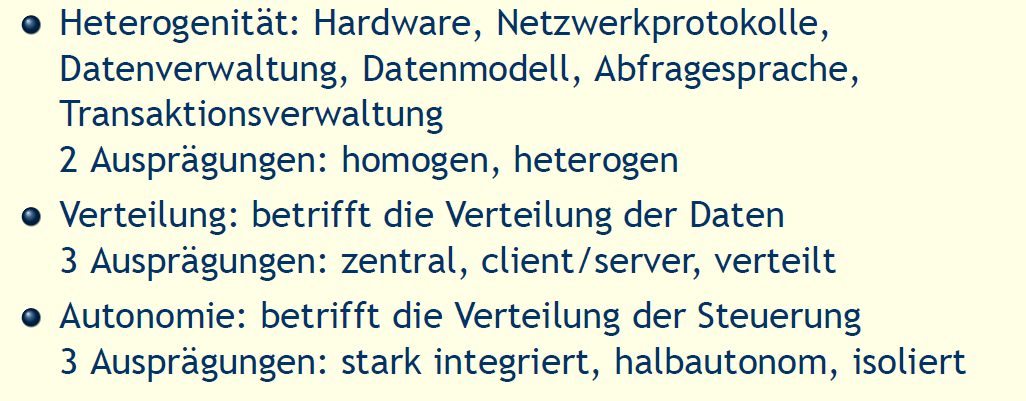


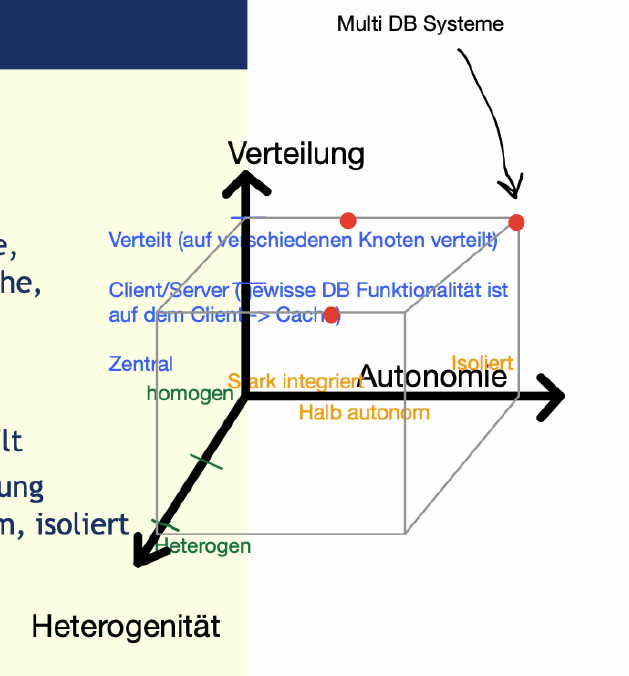
## Verteilte Datenbanksysteme

### Verteilte Datenbanksysteme

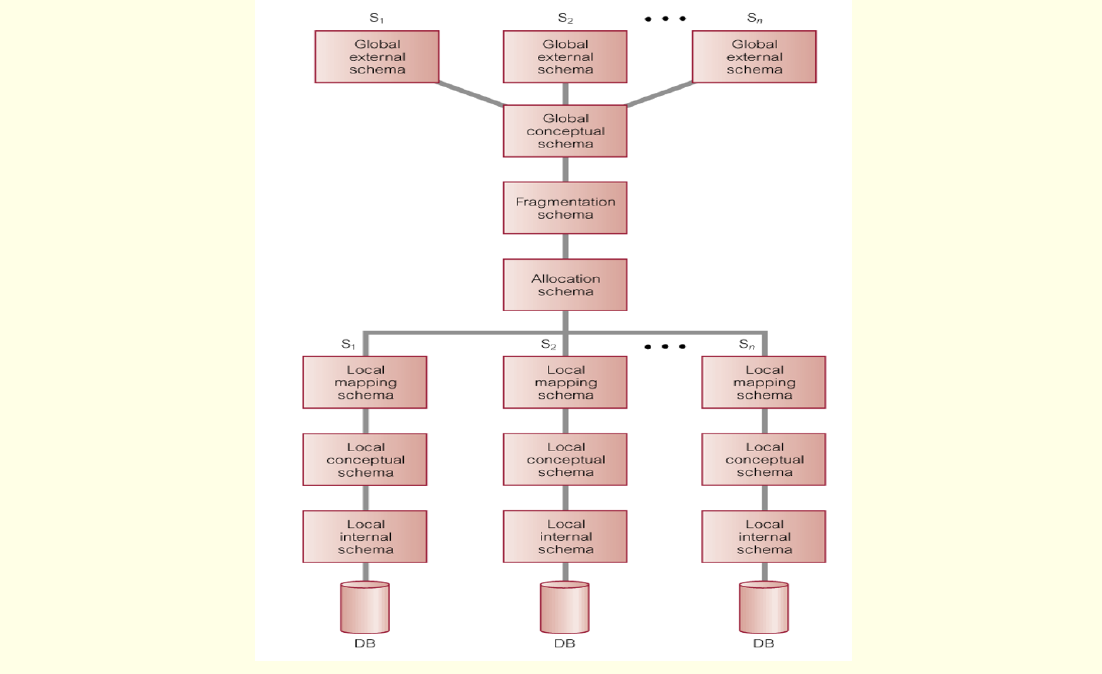


### Klassifizierung

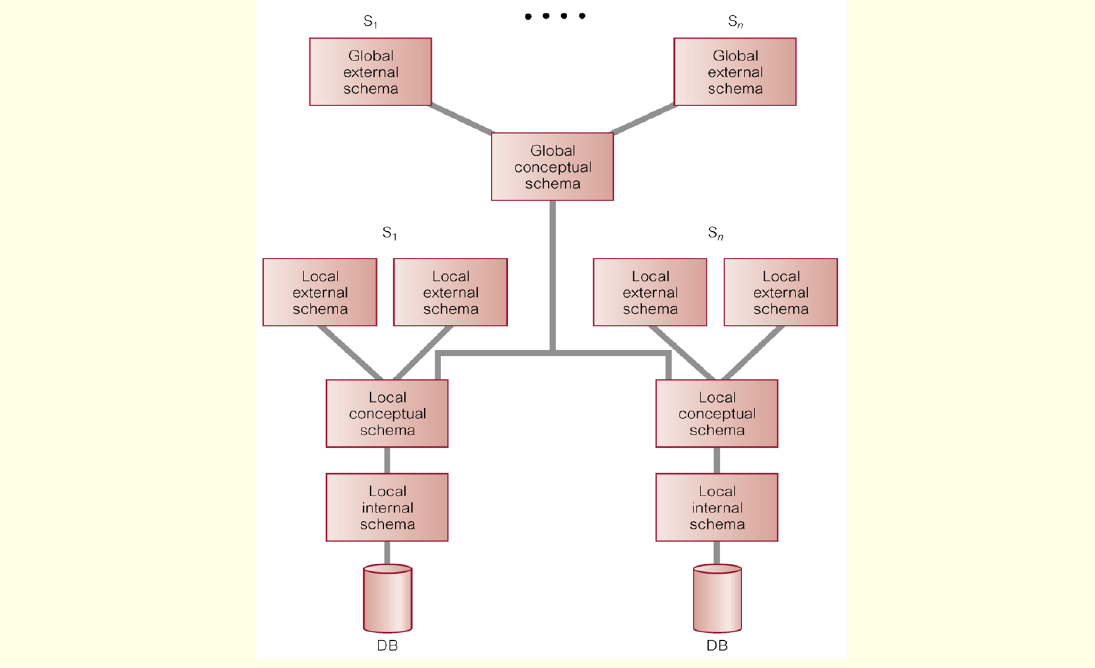




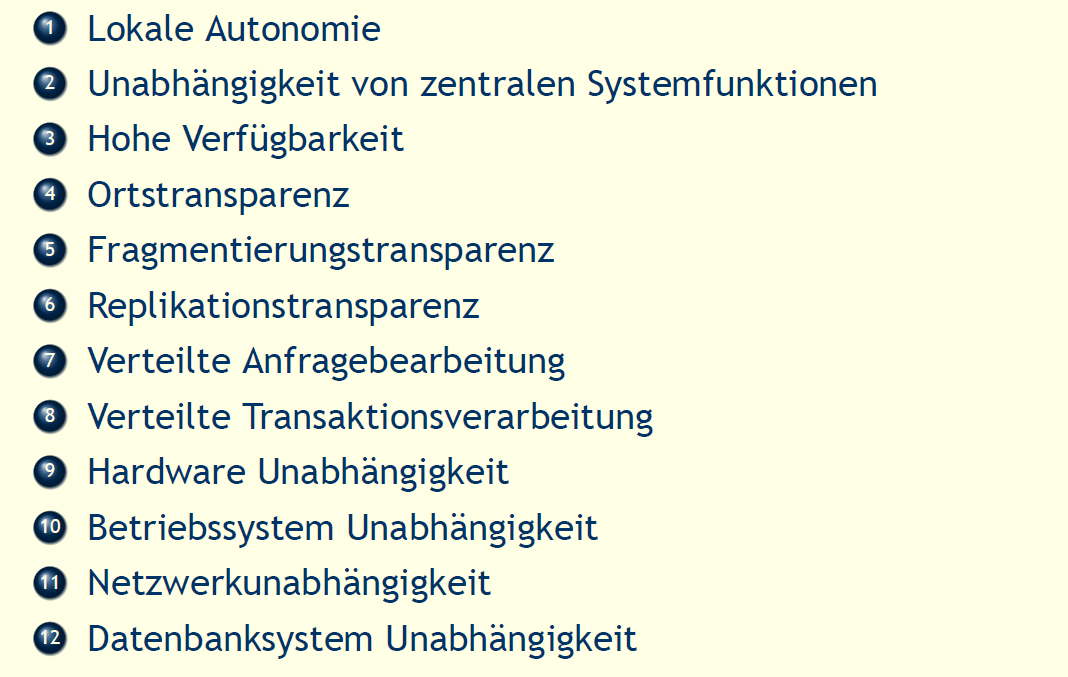
### DDBS (P2P) Architecture



### MDBS Architecture



### Date’s 12 Regeln

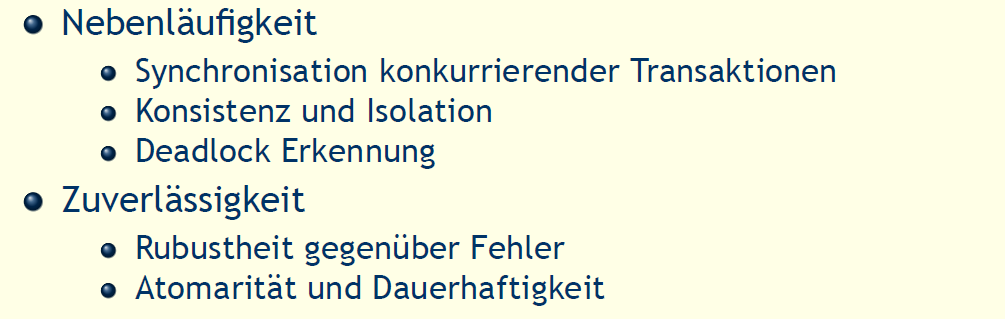


Ortstransparenz und Namenstransparenz:

* Kann mittels View oder Synonymen erstellt werden.
* Damit die Orts- und Namenstransparenz auf jeder DB genutzt werden kann müssen die Links
* symmetrisch sein. Als von jeder DB ein Link auf die anderen erstellt werden.

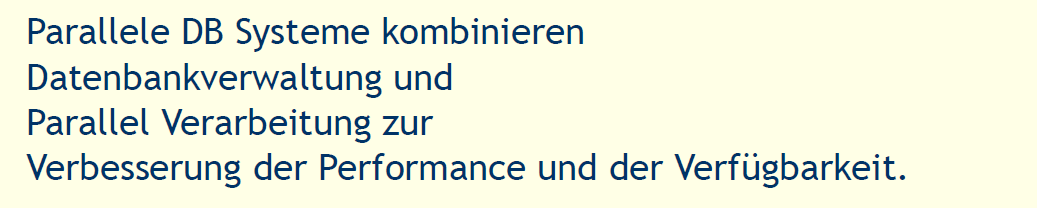
### Aspekte Verteilter Datenbanksysteme





## Parallele Datenbanksysteme

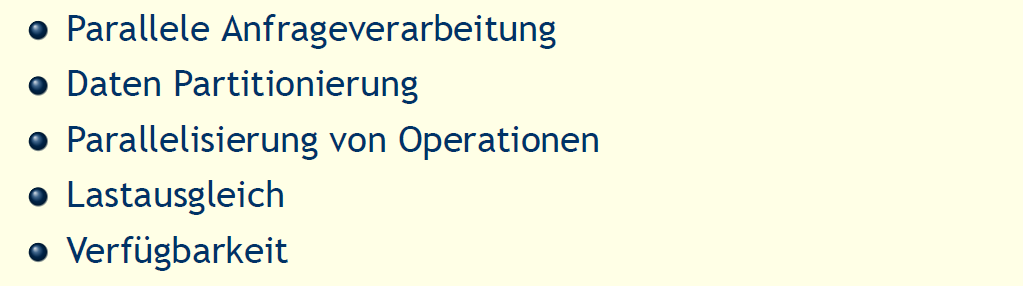
### Parallele Datenbanksysteme



### Architekturen

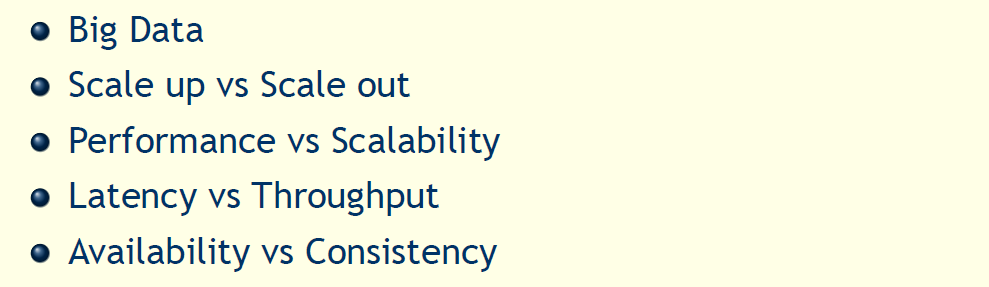


### Aspekte paralleler Datenbanksysteme



## NoSQL Systeme

### NoSQL Systeme



* BigData
* Scale up vs Scale out
* Performance vs Scalability (nicht Fokus auf Perf sondern Skalierbarkeit)
* Latency vs Throughput (Fokus auf Throughput)
* Availability vs Consistency (muss man konsistente Daten haben oder muss es die Verfügbarkeit gewährleistet sein)

### Konzepte NoSQL Systeme



Konzepte NoSQL Systeme

ACID (atomicity, consistency, isolation und durability)

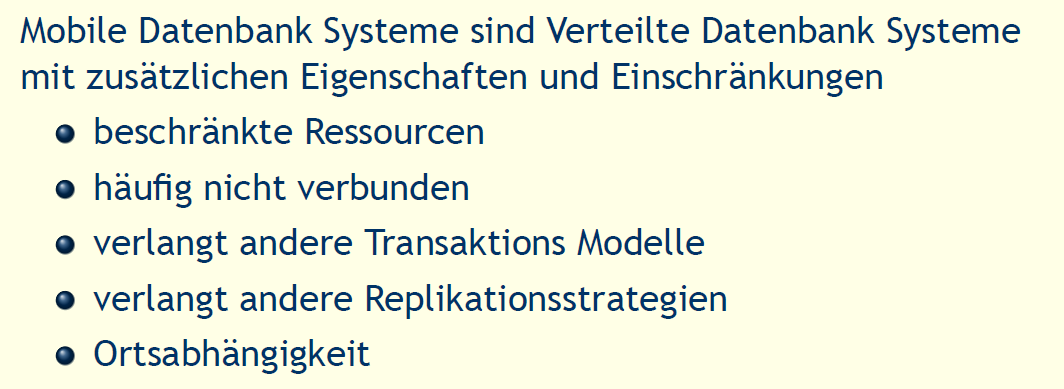
https://de.wikipedia.org/wiki/ACID

BASE

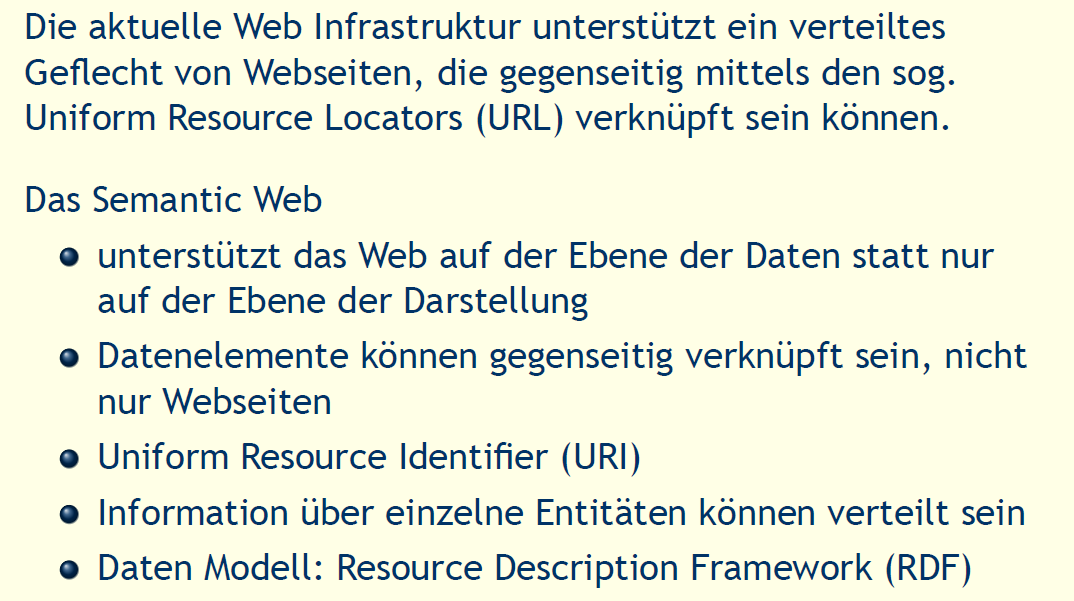
* Basically Available
* Soft state
* Eventual Consistency

https://db-engines.com/de/article/BASE

## Mobile Datenbanksysteme



## Semantic Web



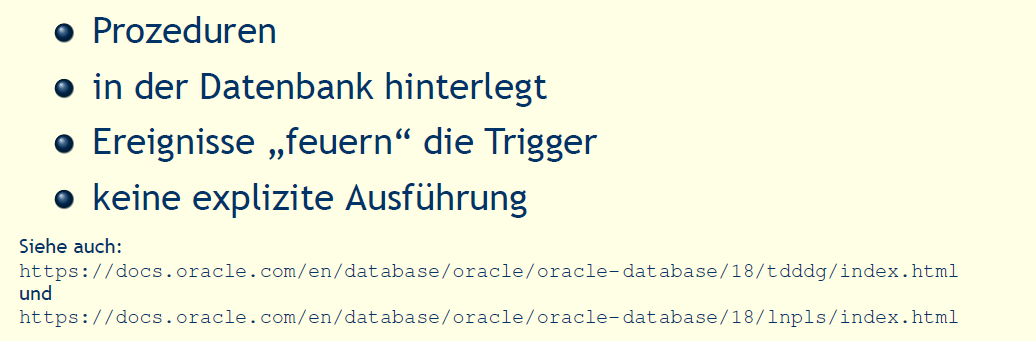
Ontologien dienen als Mittel der Strukturierung und zum Datenaustausch, um

* bereits bestehende Wissensbestände zusammenzufügen
* in bestehenden Wissensbeständen zu suchen und diese zu editieren
* aus Typen von Wissensbeständen neue Instanzen zu generieren.

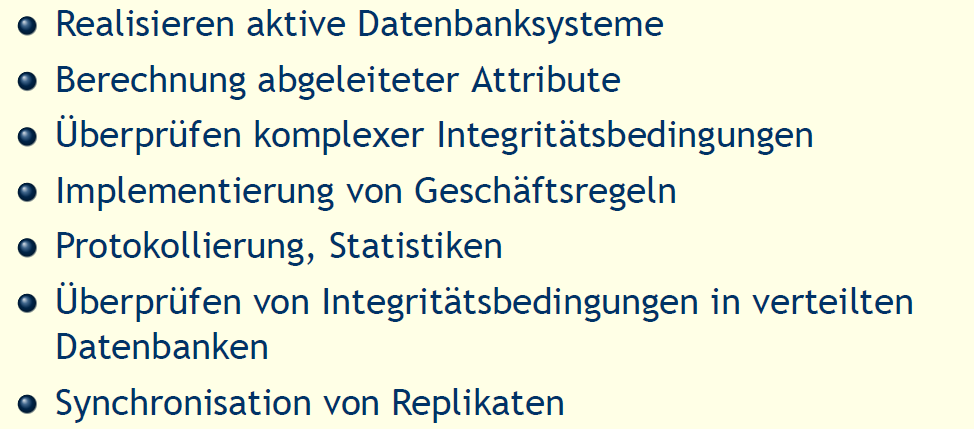
## Gliederung



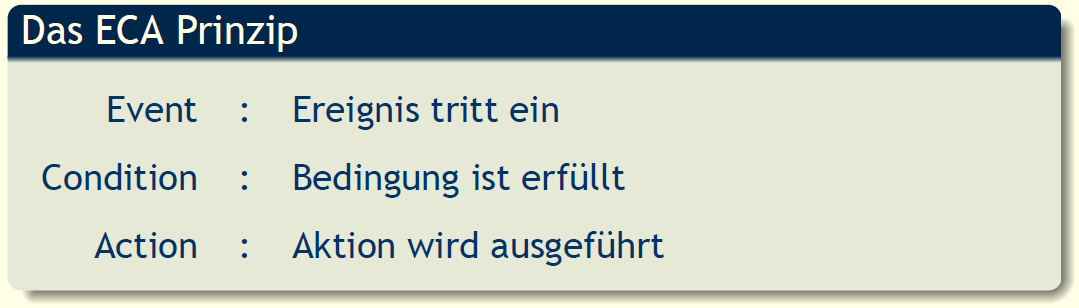
## Was sind Trigger?



## Wozu dienen Trigger?

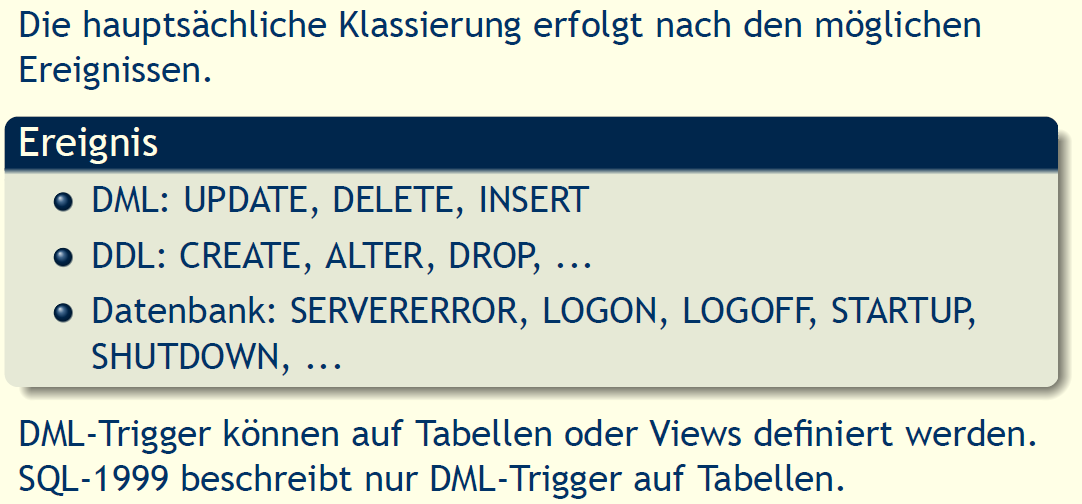


## Trigger Konzept

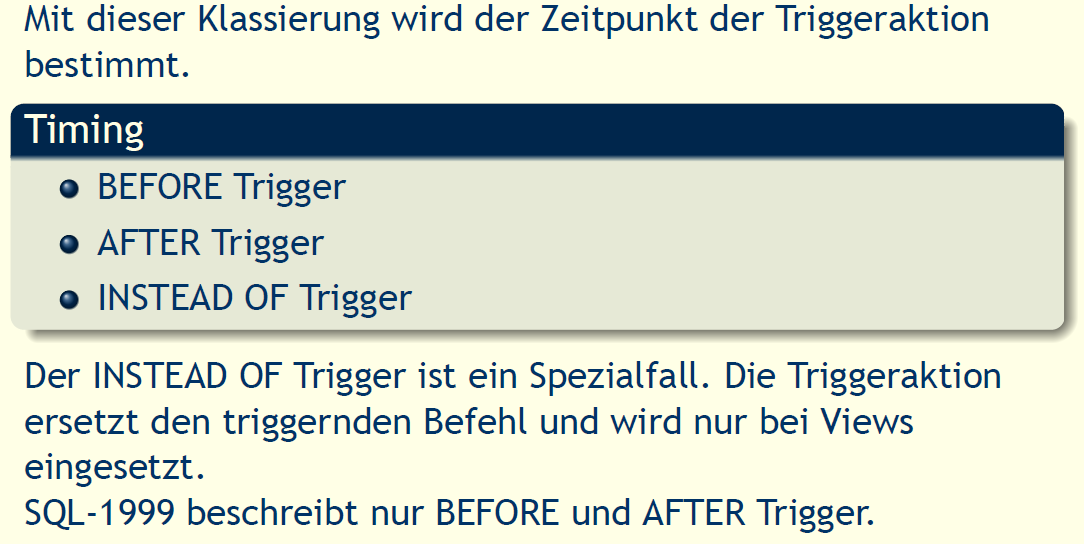


## Triggertypen

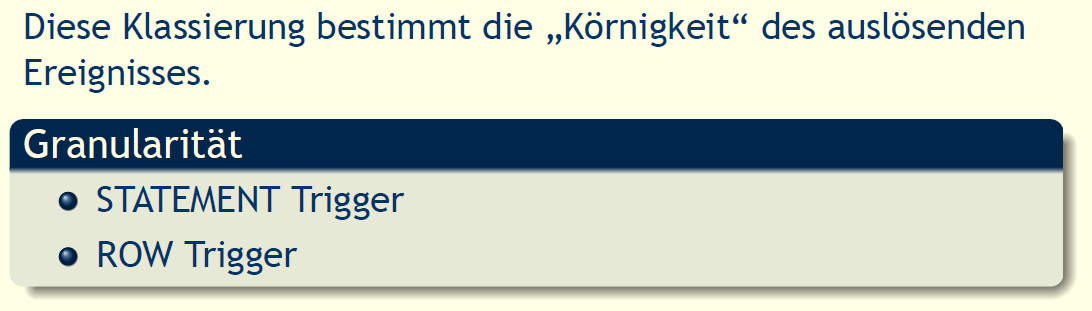
### Triggertypen: Ereignis



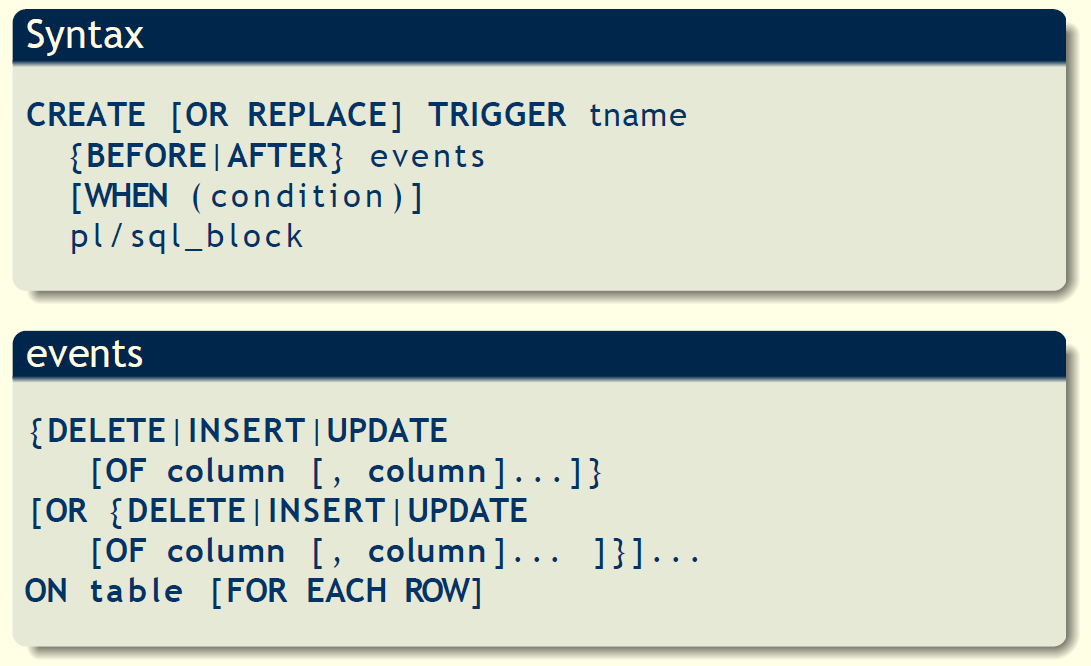
### Triggertypen: Timing



### Triggertypen: Granulat



## DML-Trigger auf Tabellen

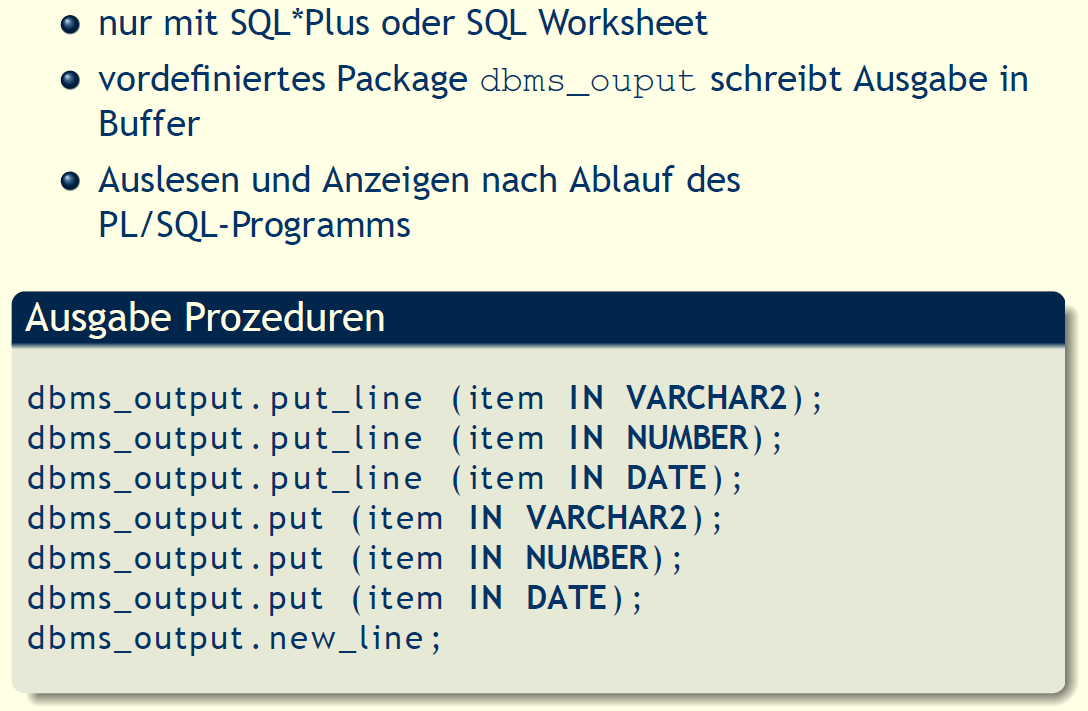


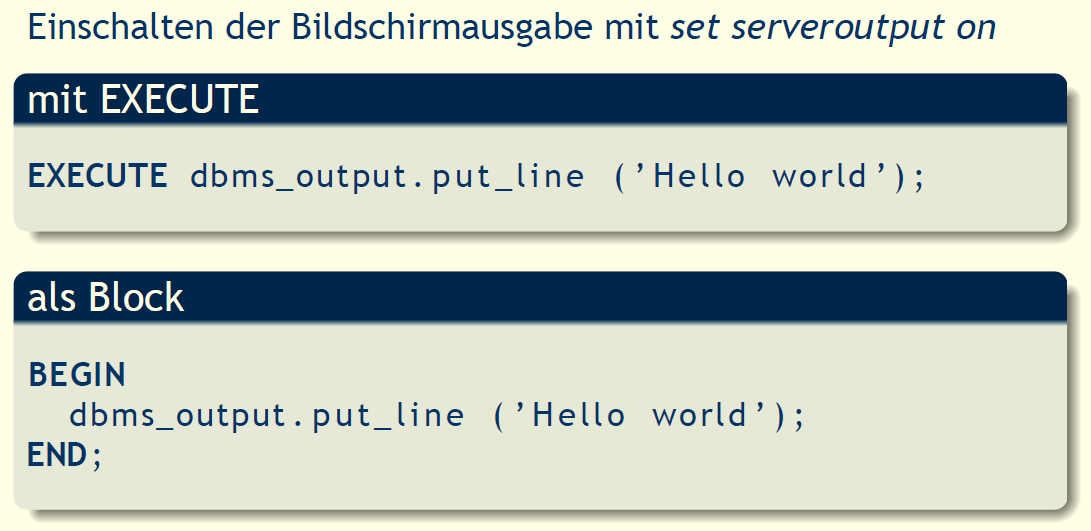
## PL/SQL Block

### Einfacher PL/SQL Block

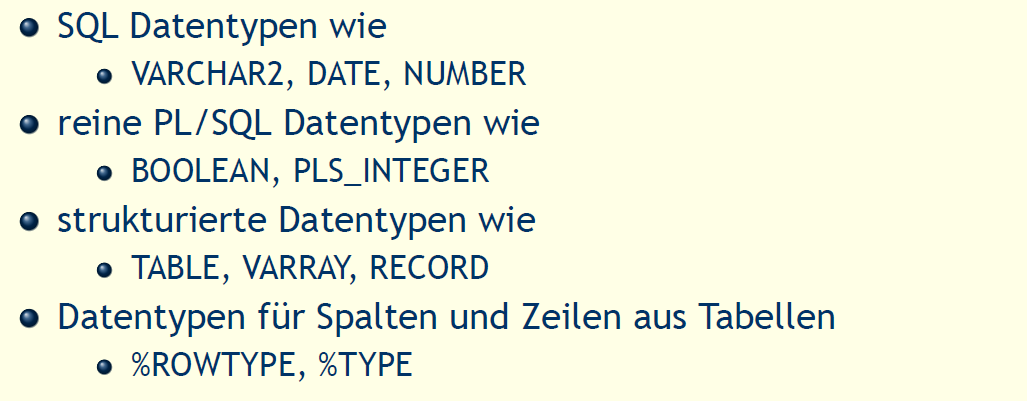


### Bildschirmausgabe

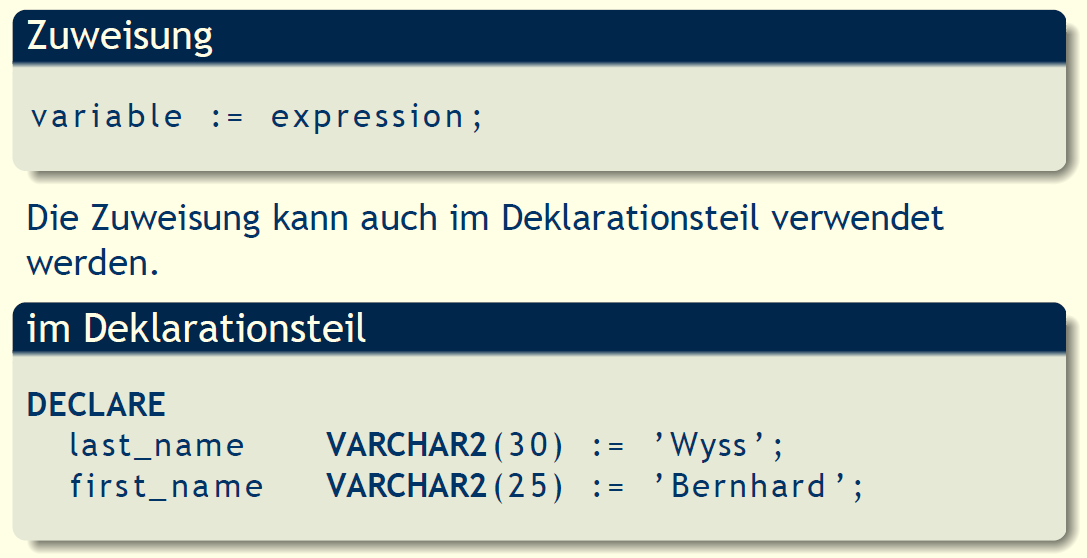


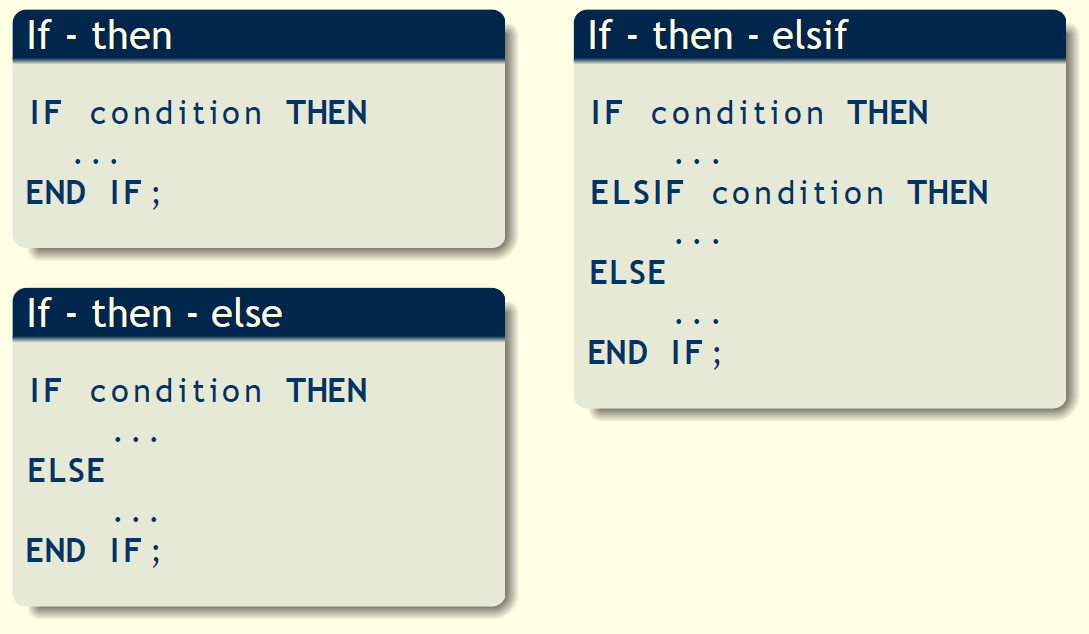


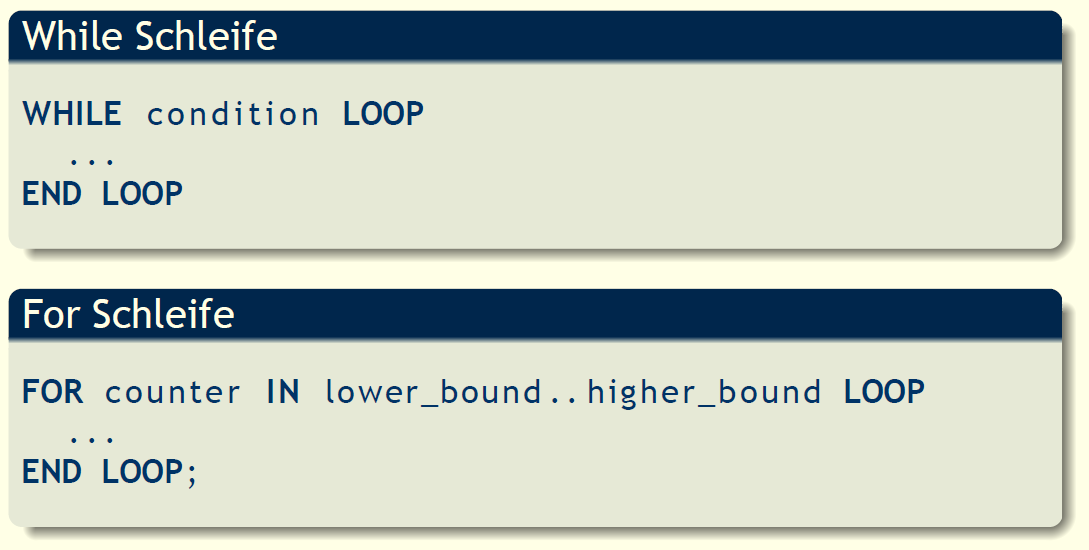
### Datentypen



### Anweisung

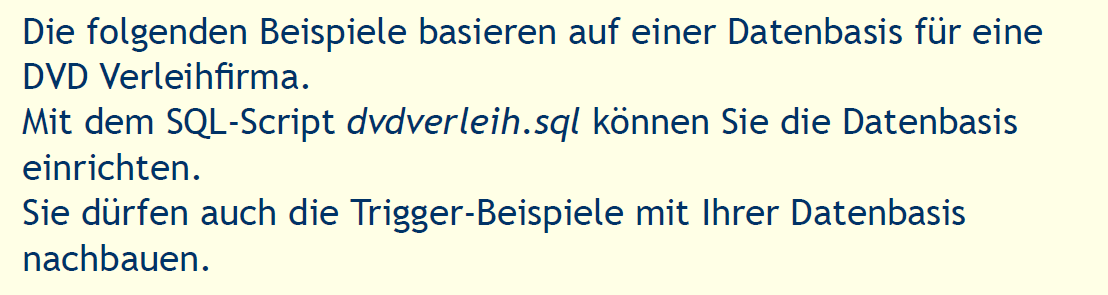




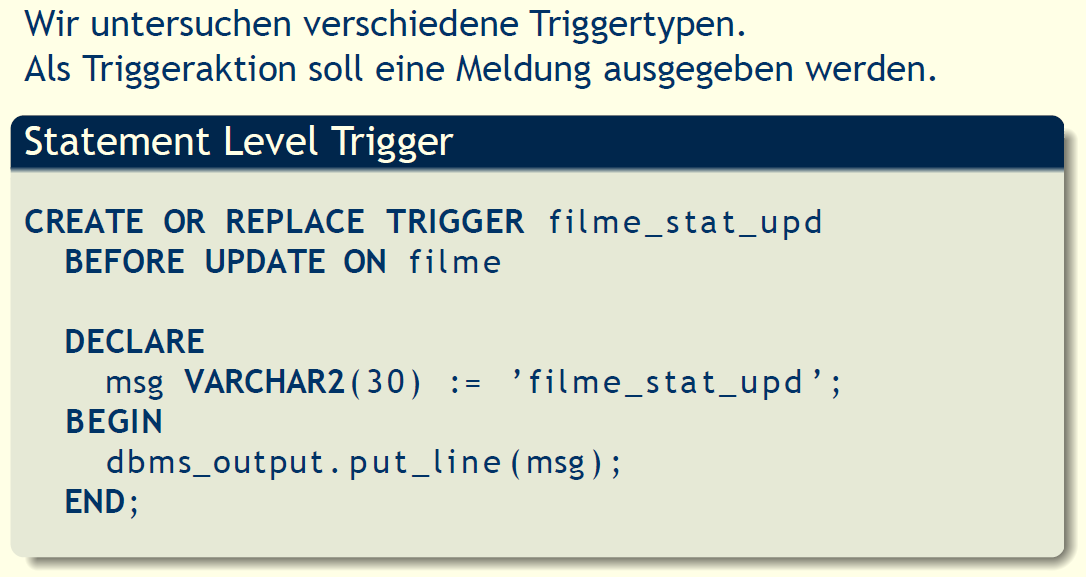


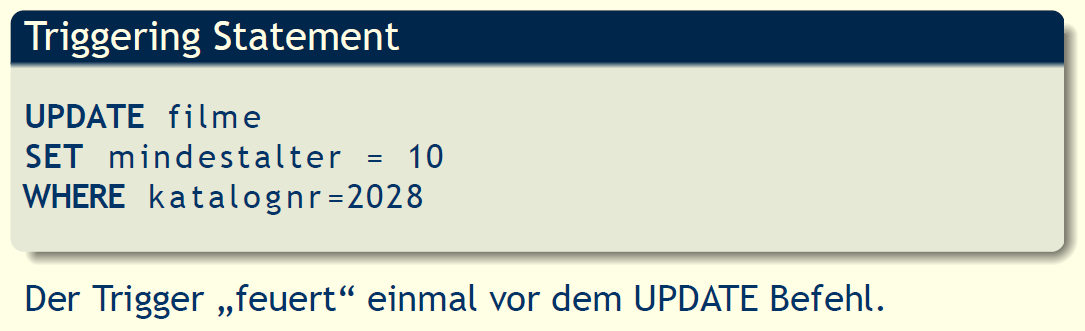
## 1. Erfahrung mit Trigger

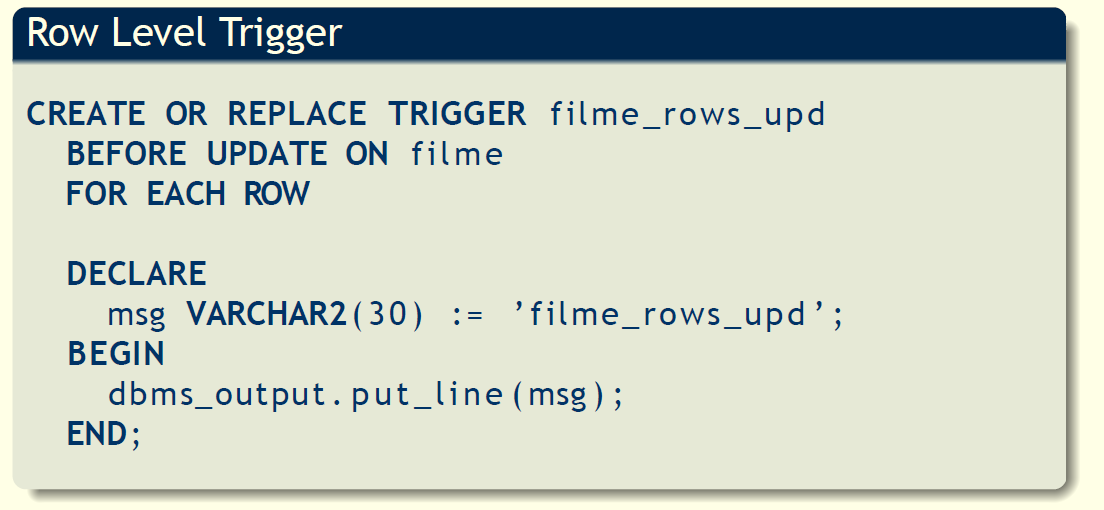
### Beispiel-Datenbasis

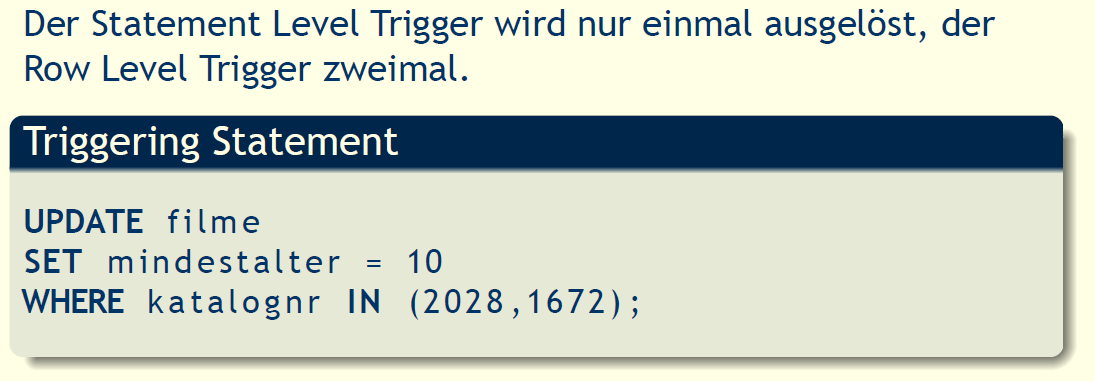


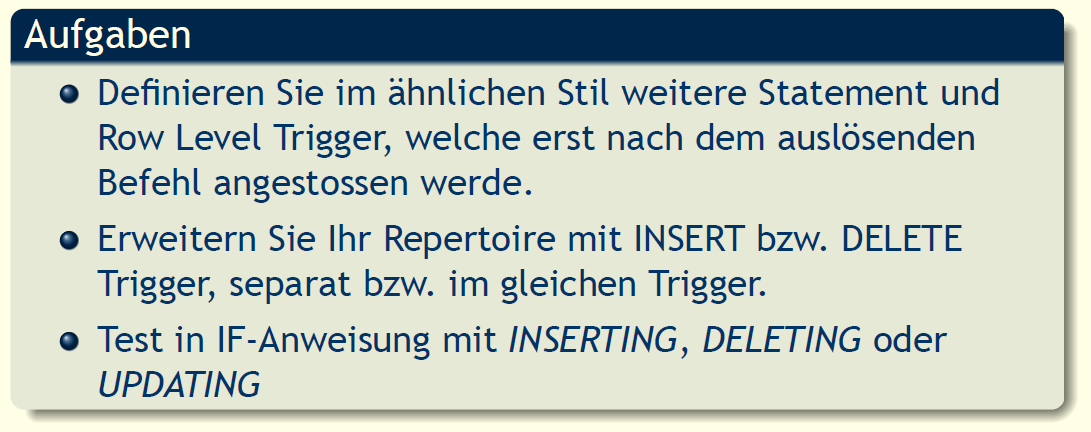
### 1. Erfahrung mit Trigger



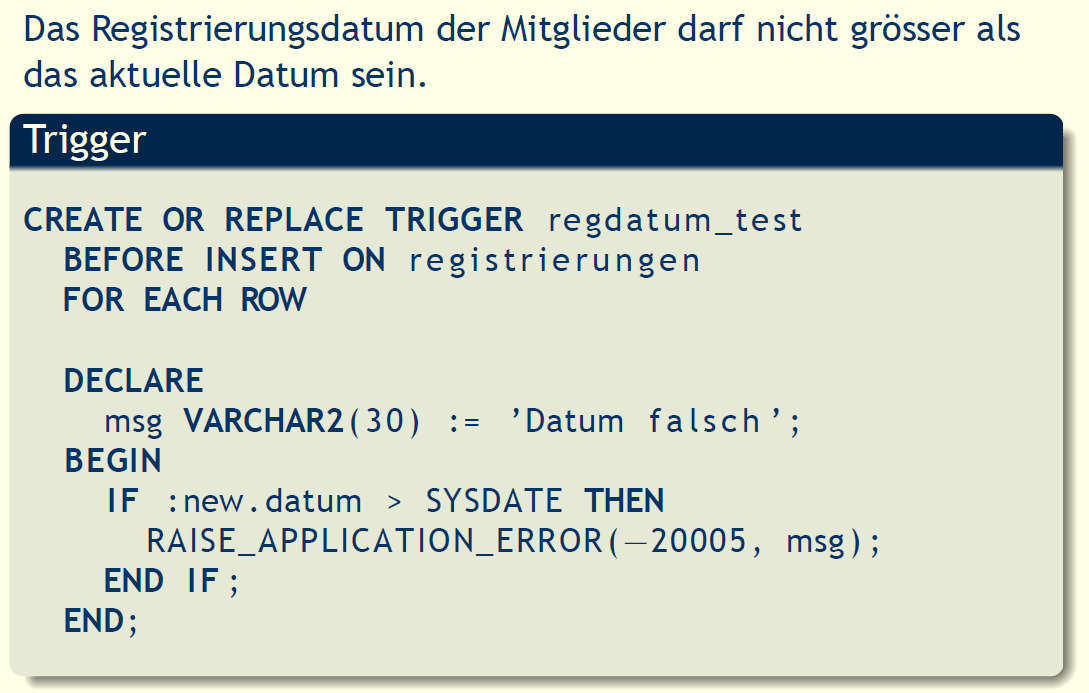


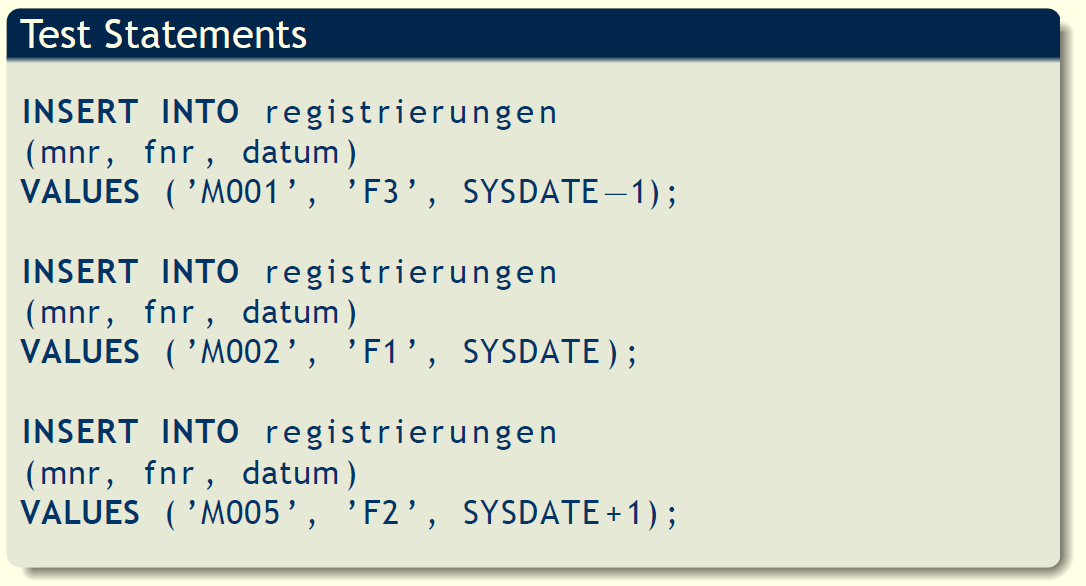


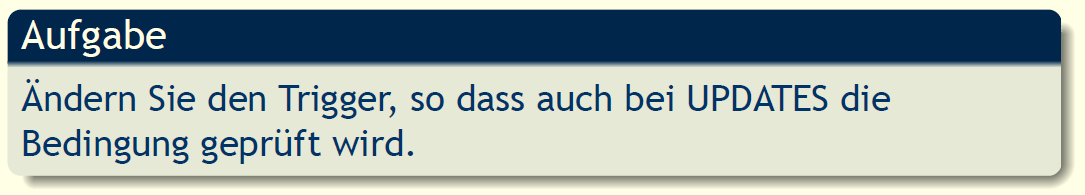




## Konsistent Prüfen

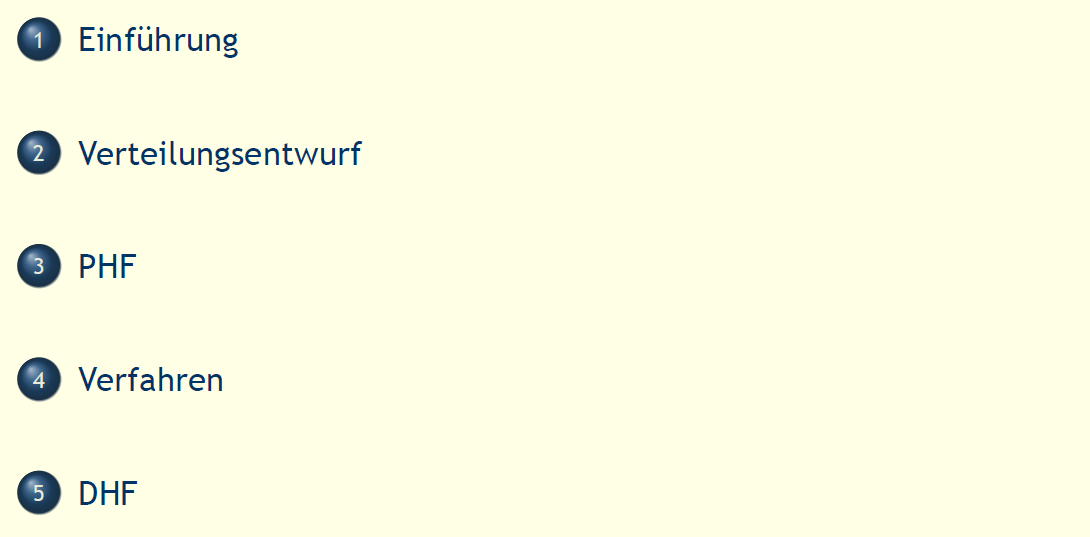






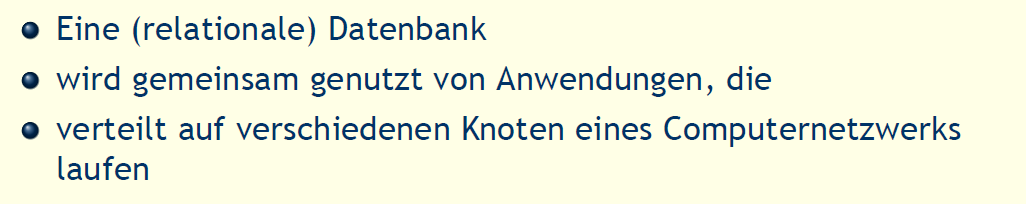
# Woche 2

## Inhalt

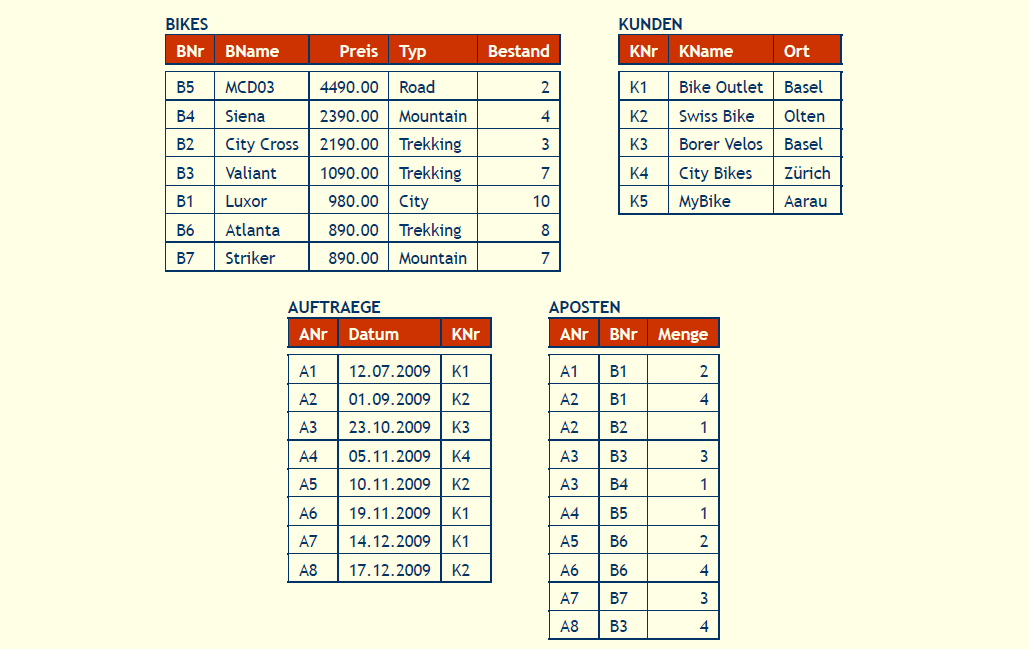


## Einführung

### Ausganglage



### Datenbank Beispiel



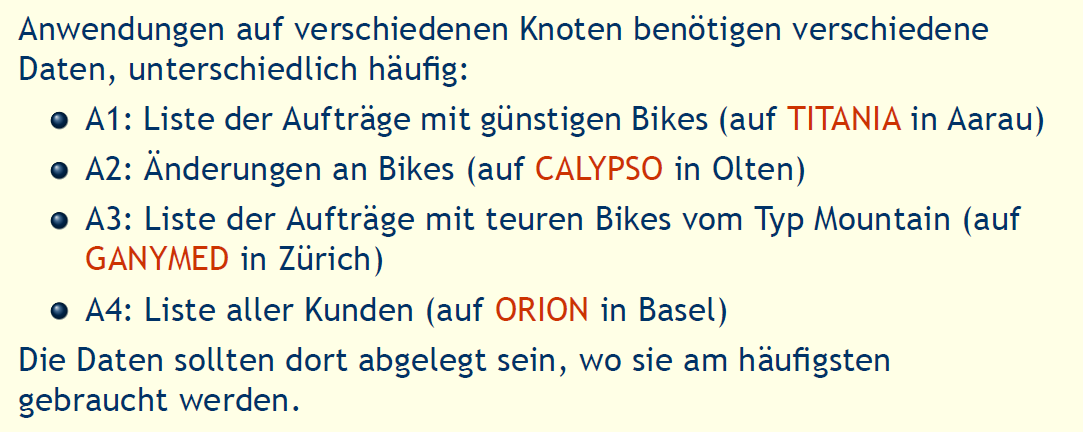
Die Daten sollten dort abgelegt werden wo sie am häufigsten verwendet werden. Der Ort hat Einfluss auf die Performance. Lokaler DB Zugriff ist günstiger als wenn standortübergreifend auf die DB zugegriffen wird. Bei der Aufteilung muss man sich fragen:

* Welche Anwendungen?
* Welche Queries?
* Auf welchen Konten?
* Benötigen welche Daten
* Mit welcher Häufigkeit
* Sharps zufällig
* Fragmentierung gewollt

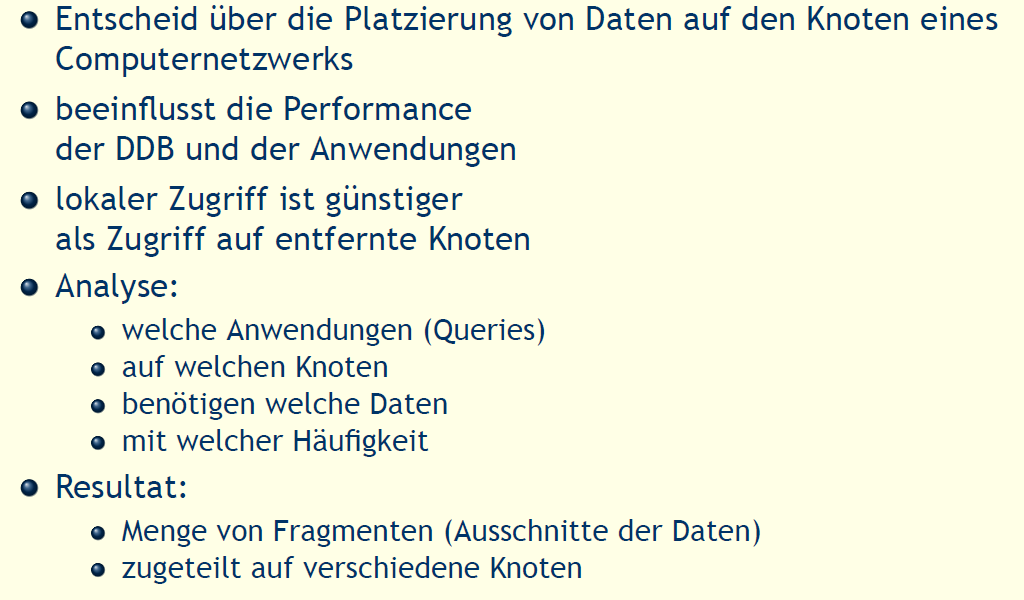
Für den Entwurf gibt es verschiedene Strategien. Top-Down und Bottom-Up

Top-Down eignet sich bei homogenen Systemen und wenn der Entwurf von Grund auf gemacht werden kann.

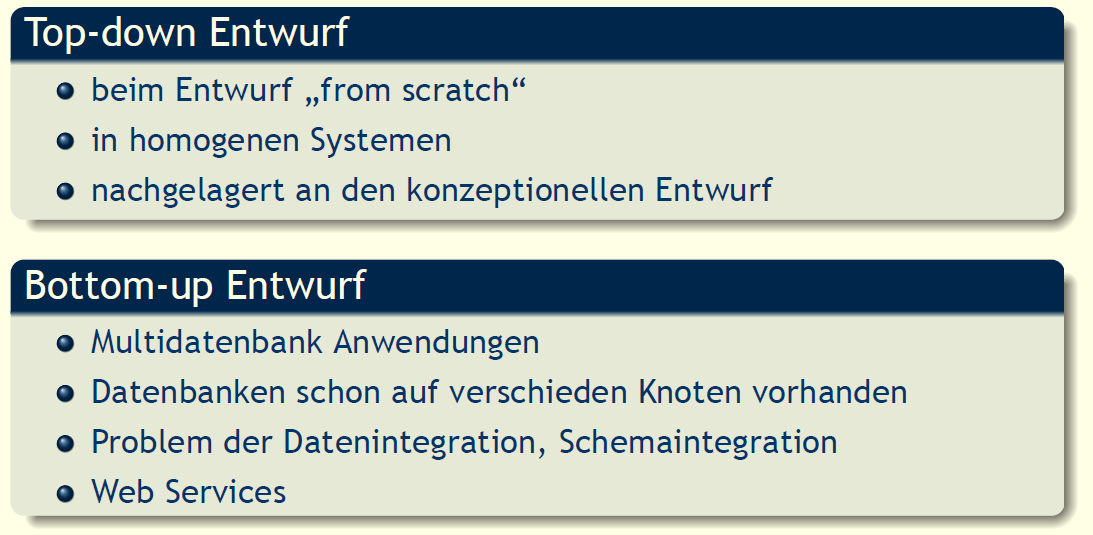
### Verteilte Anwendungen



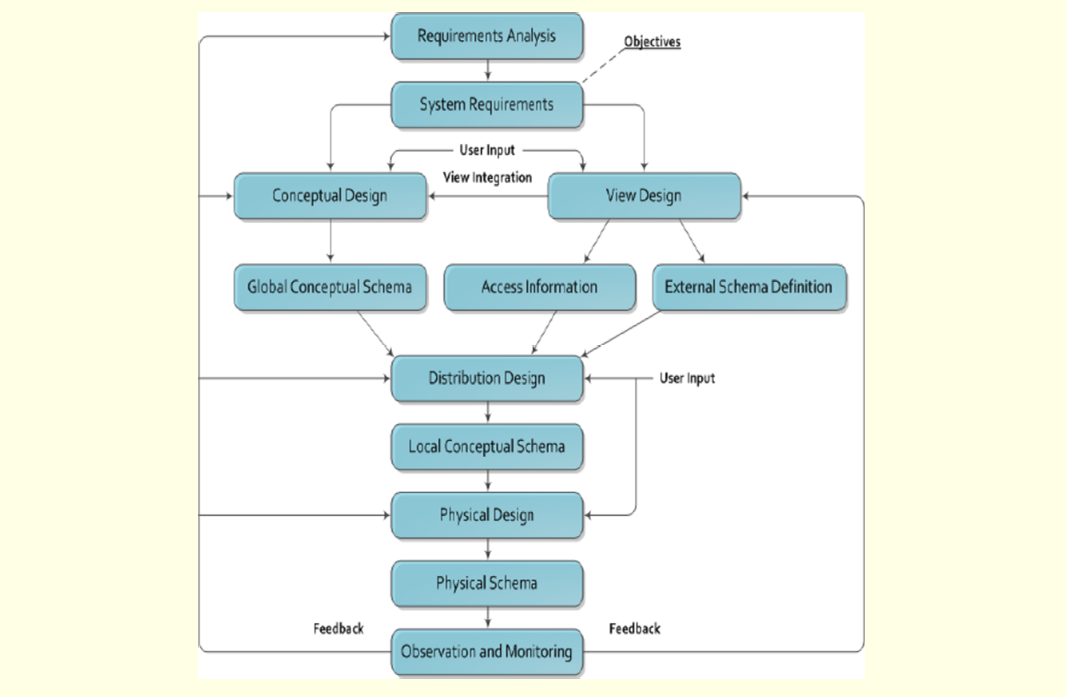
### Entwurfsproblem



### Alternative Entwurfsstrategien

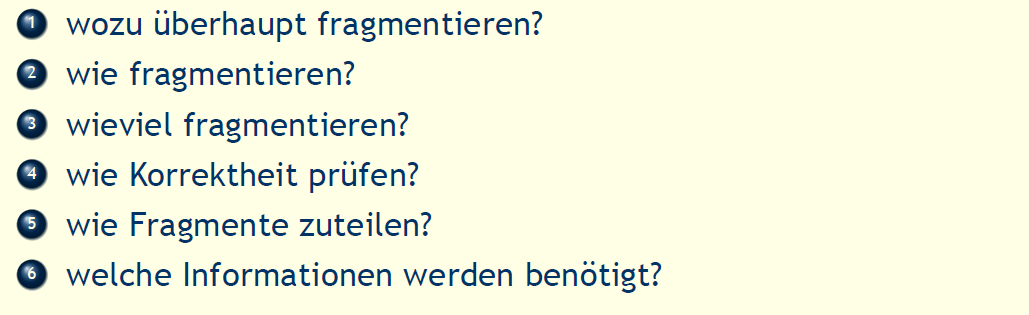


### Top-down Entwurf

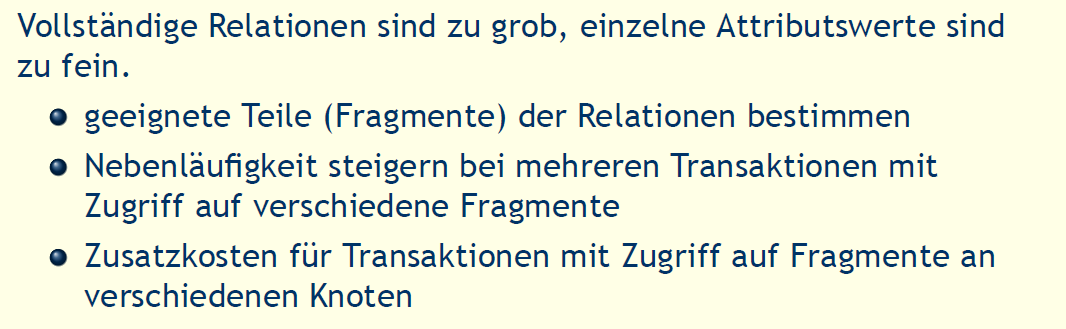


## Verteilungsentwurf

### Kernpunkte des Verteilungsentwurfs



### Grad der Fragmentierung

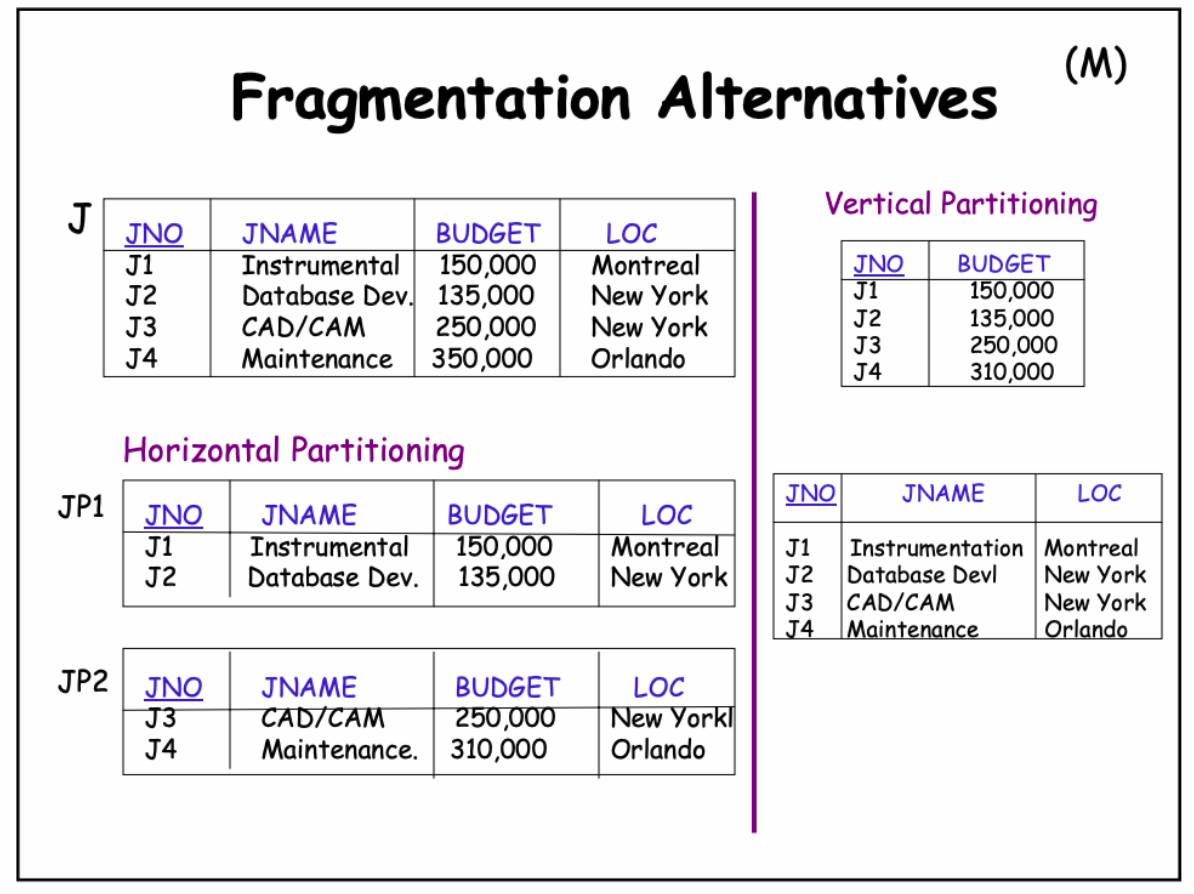


Vollständige Relationen sind zu grob, einzelne Attributswerte sind zu fein.

* Geeignete Teile bestimmen
* Zugriff auf entfernte Fragmente ist teurer

### Arten der Fragmentierung



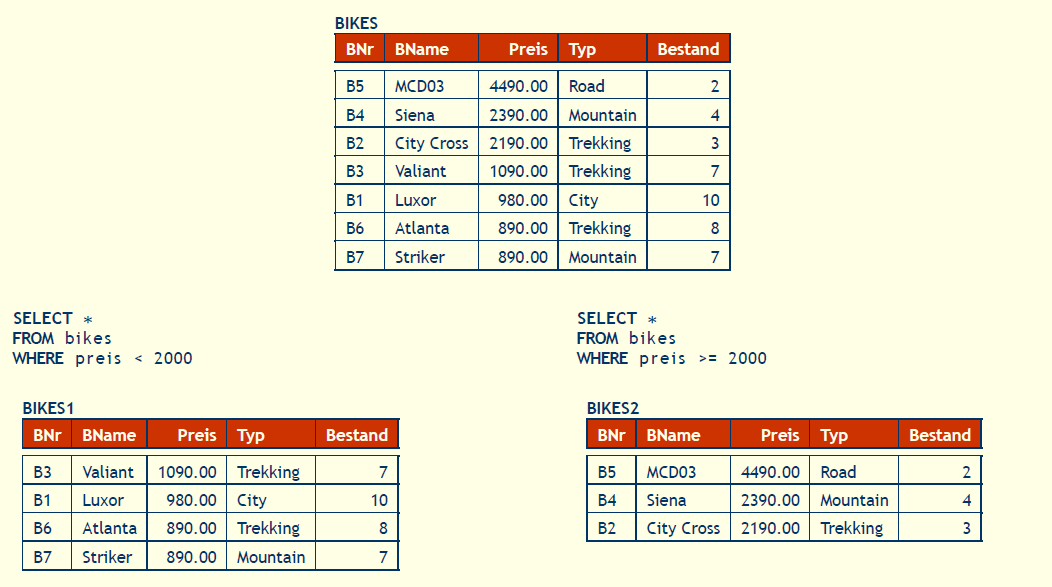


Horizontale Fragmentierung entspricht eher der Normalisierung (Auftrennen nach Tabellen oder Attributen) (UNION)

Alle Bike die Teurer als 2000 sind kommen in Tabelle A und alle günstiger als 2000 in Tabelle B

Vertikale Fragmentierung entspricht eher der Patitioierung (Historisierung) (JOIN)

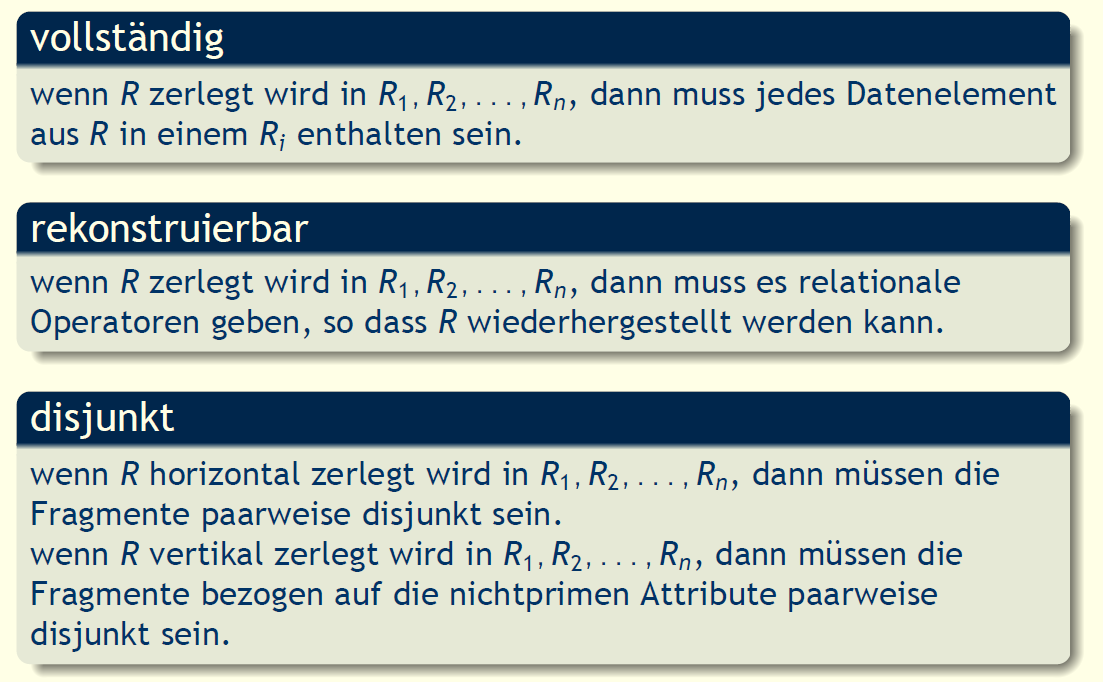
### Möglichkeit der Fragmentierung – Horizontal



### Möglichkeit der Fragmentierung – Vertikal

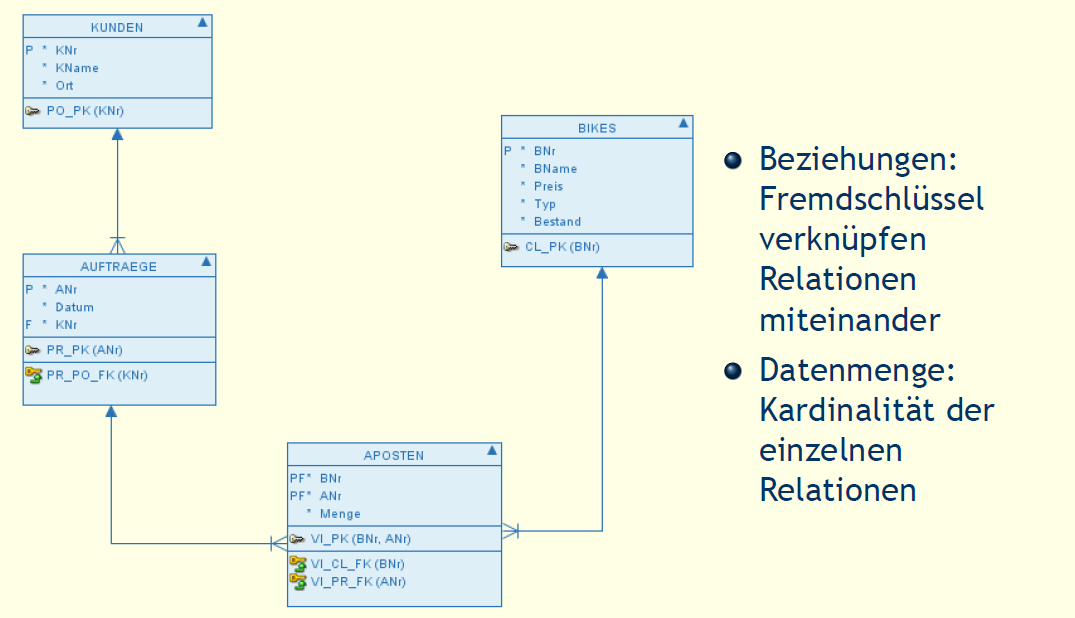


### Korrektheit der Fragmentierung



## PHF

### PHF – Information über Daten



### PHF – Information über Anwendungen



Simple Predicates

P1: Typ=‚Road‘

P3: Typ = City

Minterm Predicates

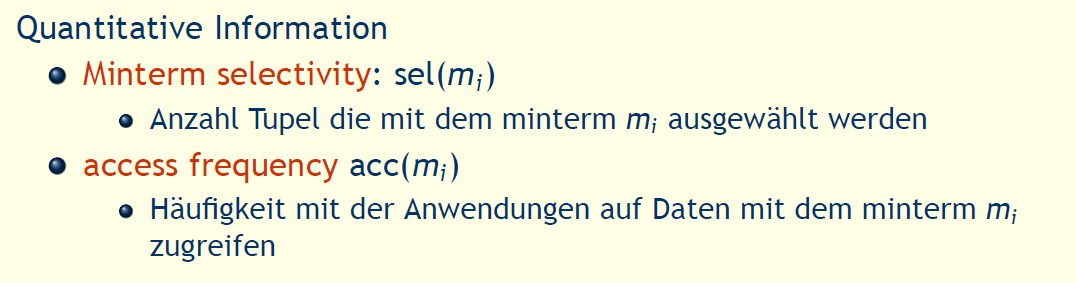
M1: Typ = ‚Road‘ and Typ = City

p1 ^ p2 ^ …

### Minterm Beispiele

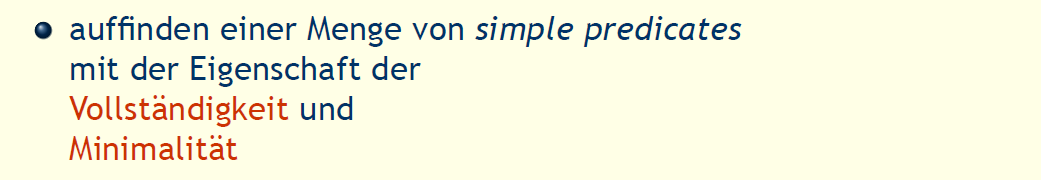


### PHF – Information über Anwendungen

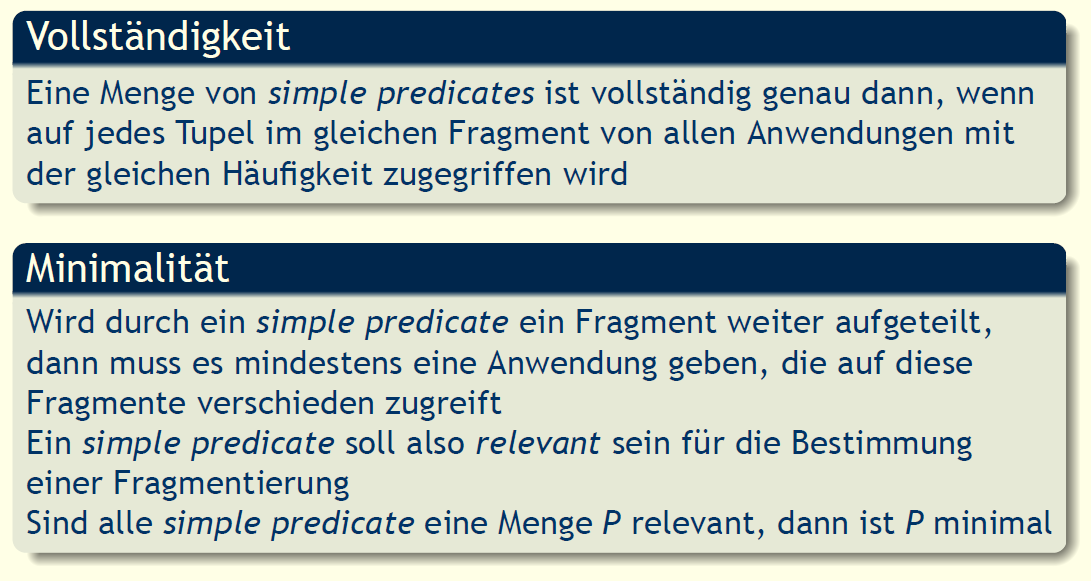


## Verfahren

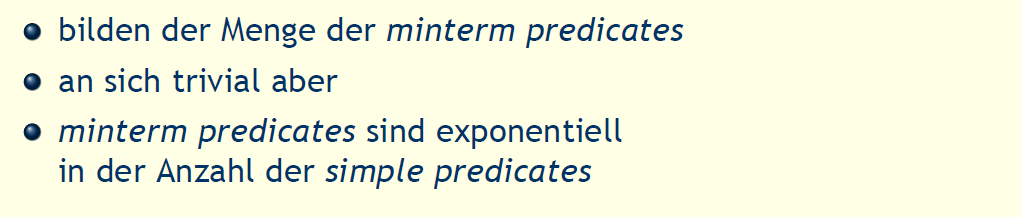
### PHF Verfahren, 1. Schritt



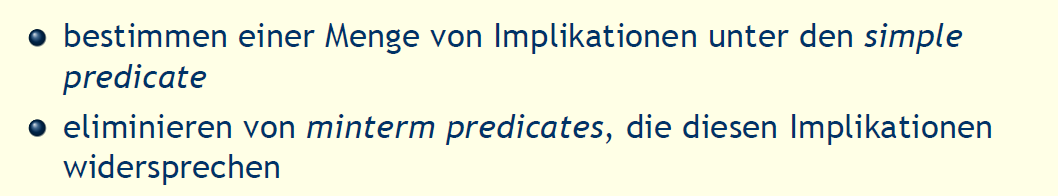
### Vollständigkeit und Minimalität



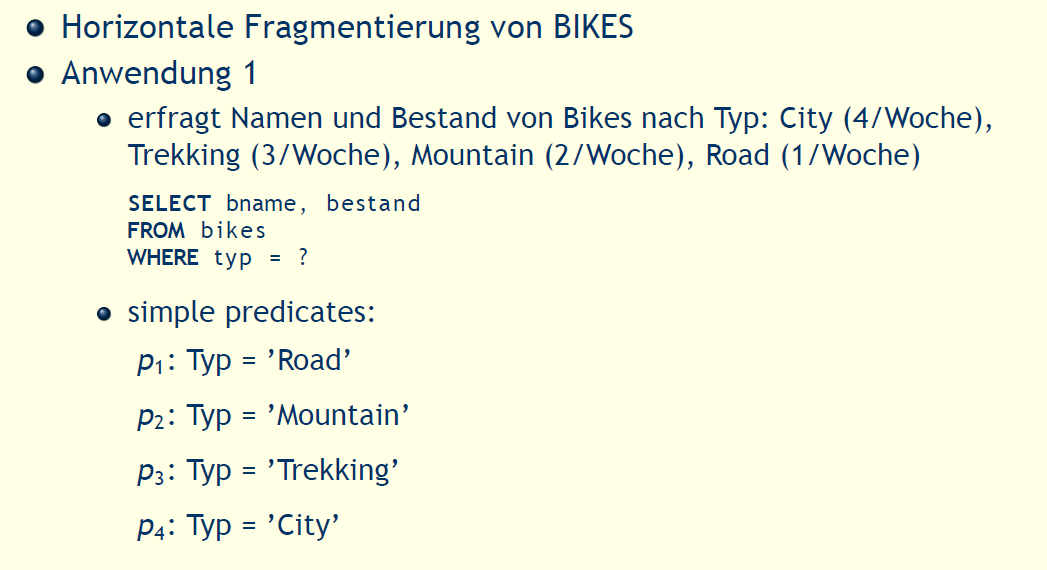
### PHF – Verfahren, 2. Schritt

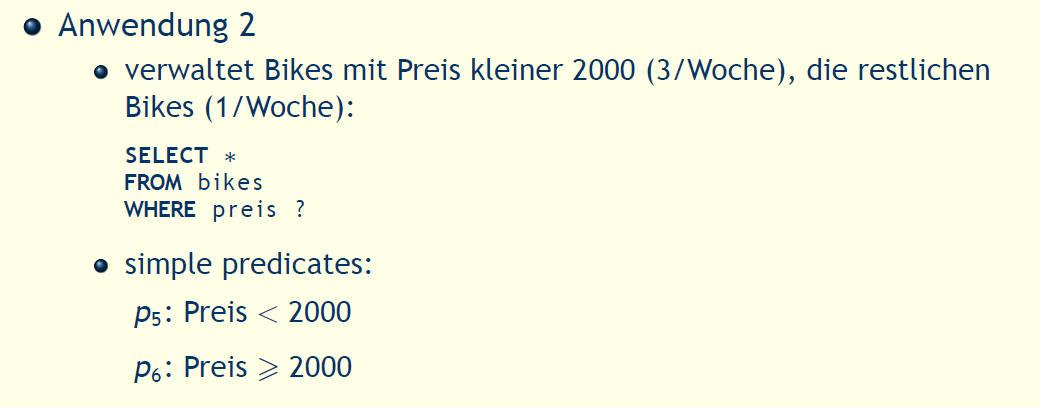


### PHF – Verfahren, 3. Schritt



### PHF – Beispiel



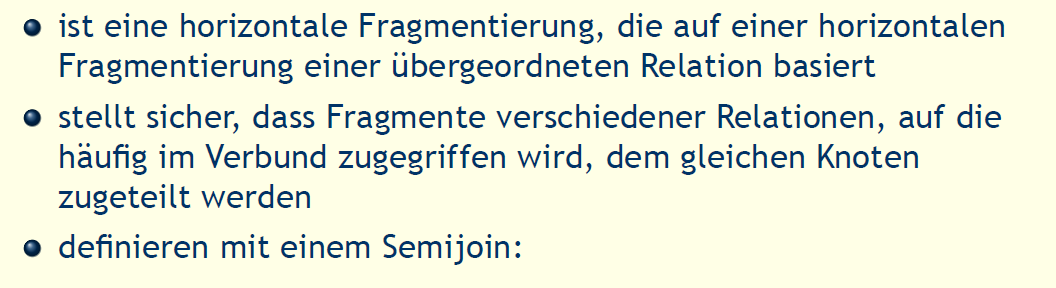




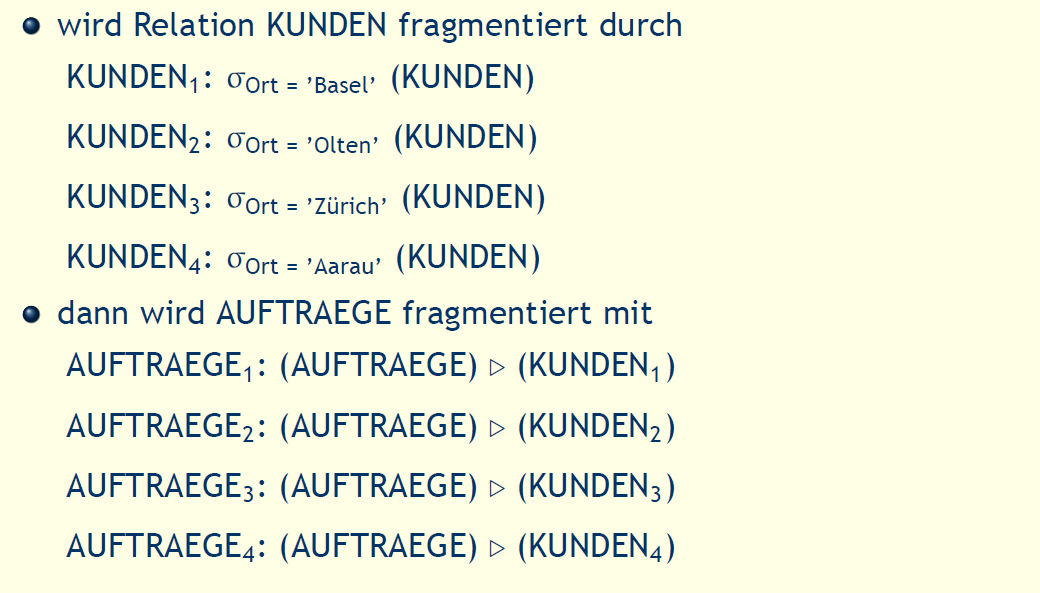


## DHF

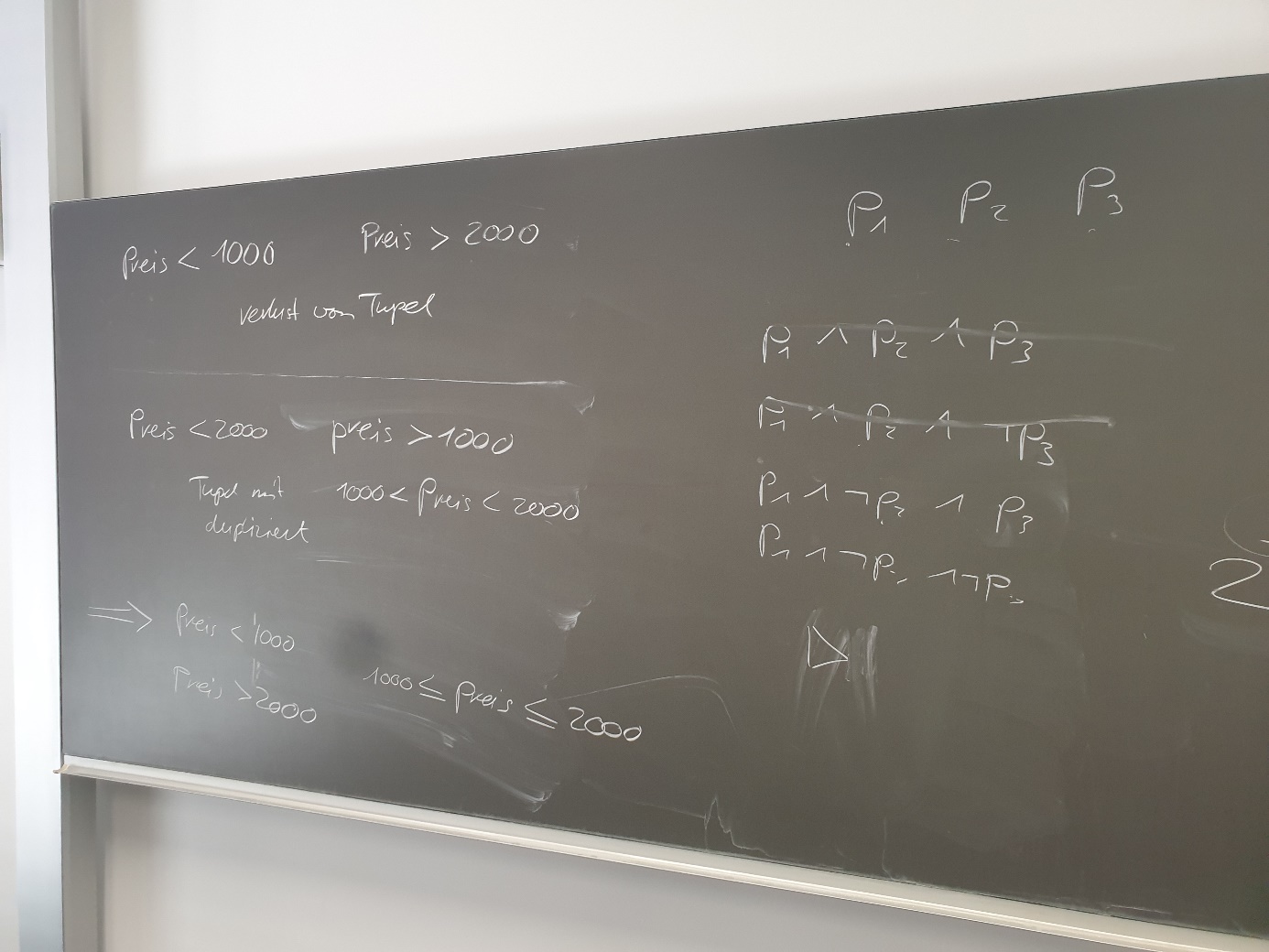
### Abgeleitete horizontale Fragmentierung

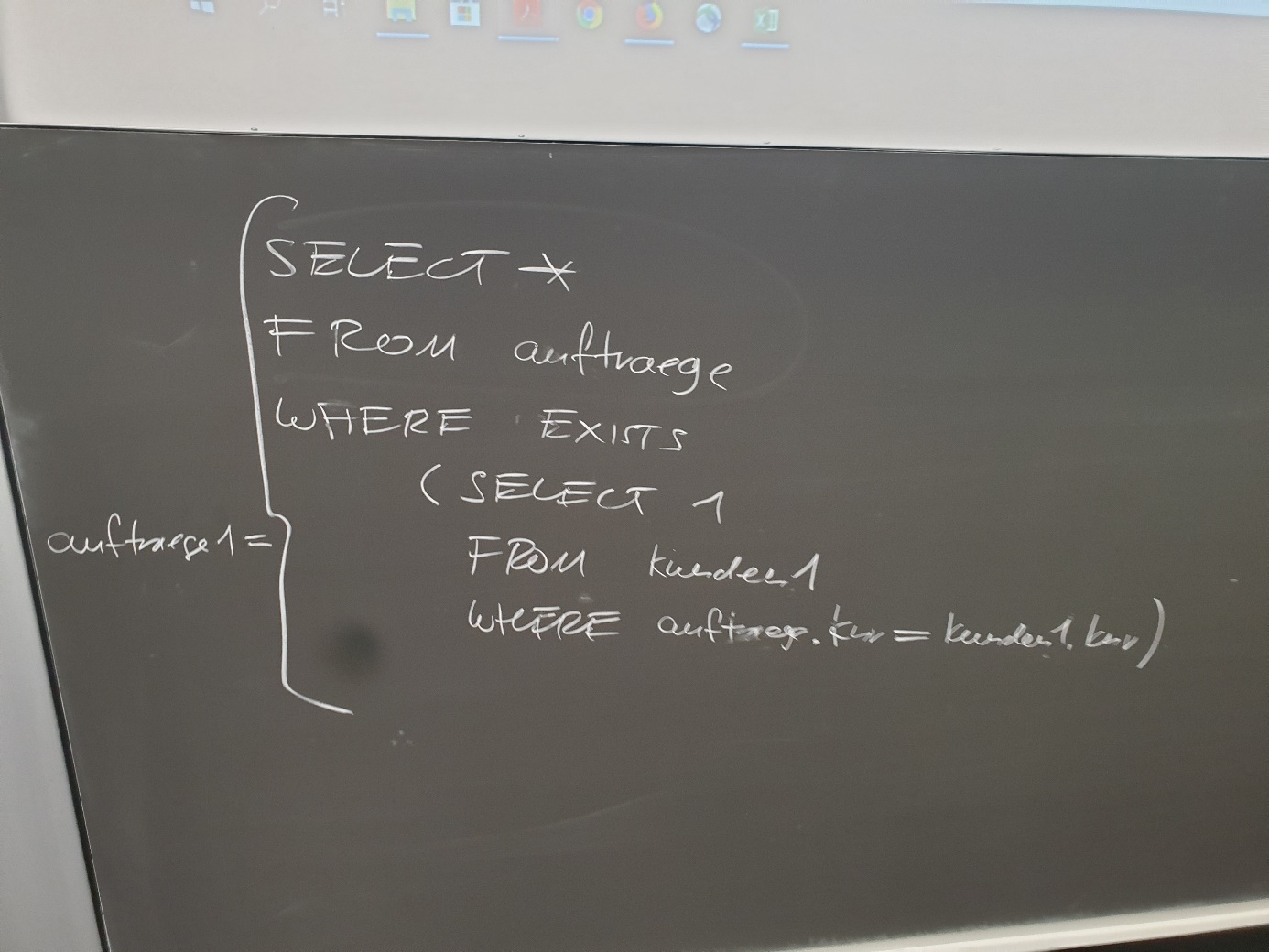


### DHF - Beispiel



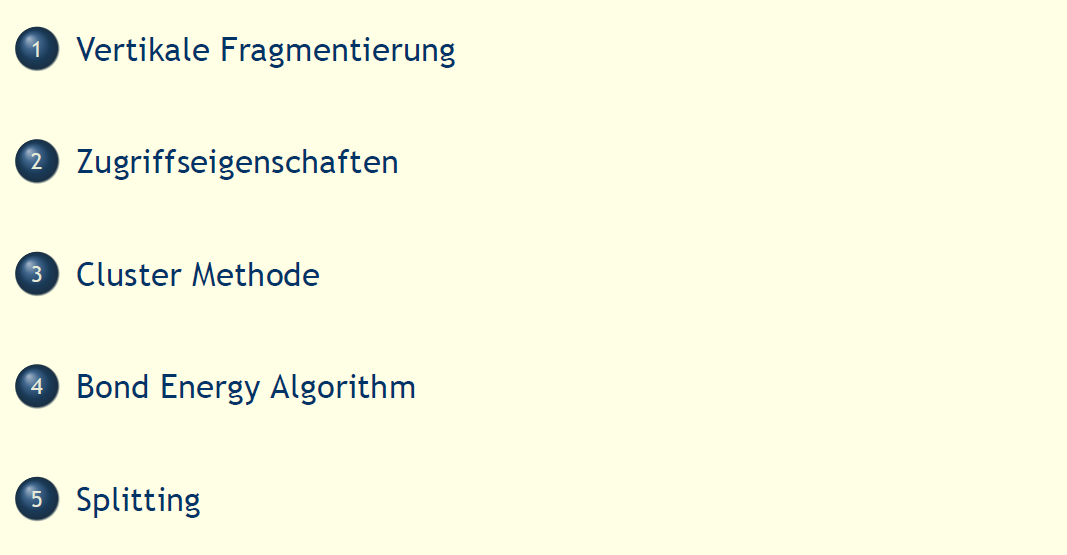
## Tafelmitschrift





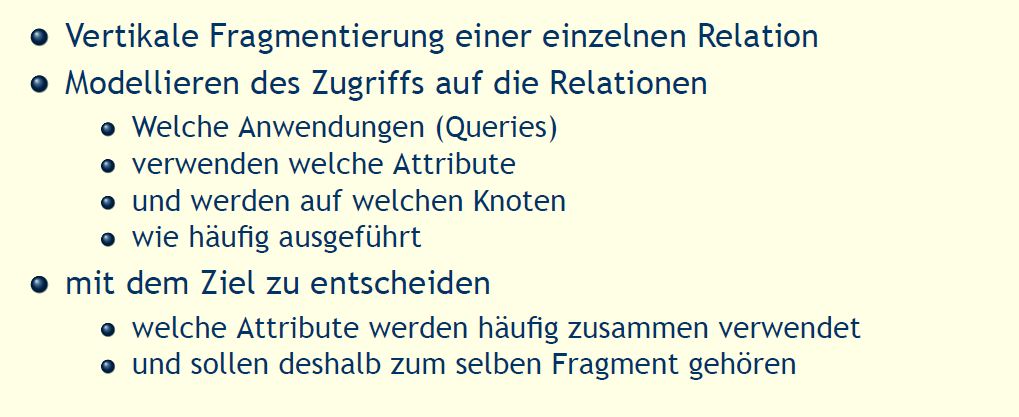
# Woche 3

## Inhalt

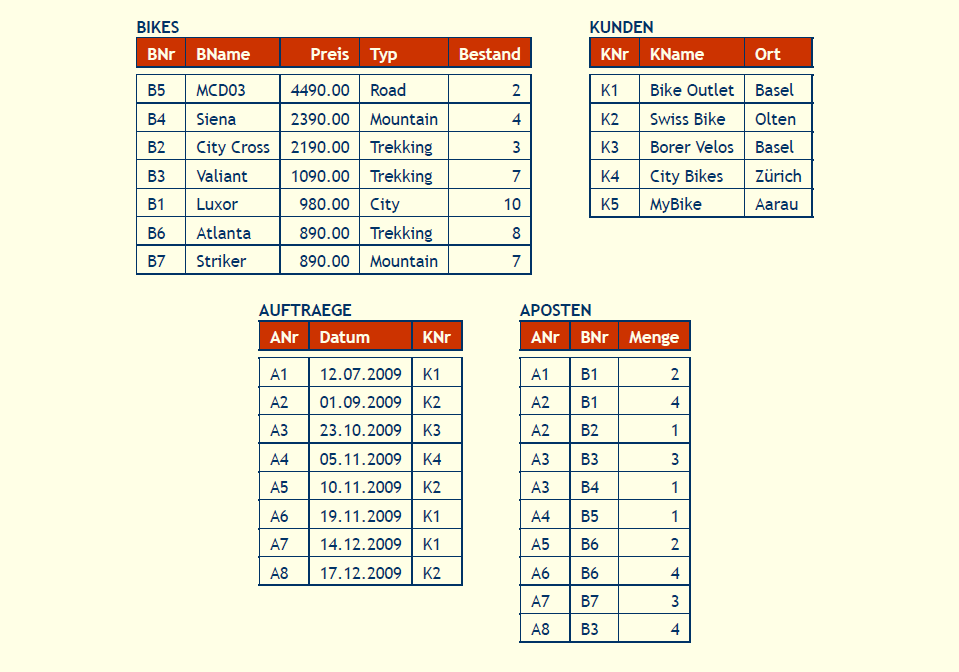


## Vertikale Fragmentierung

### Vertikale Fragmentierung



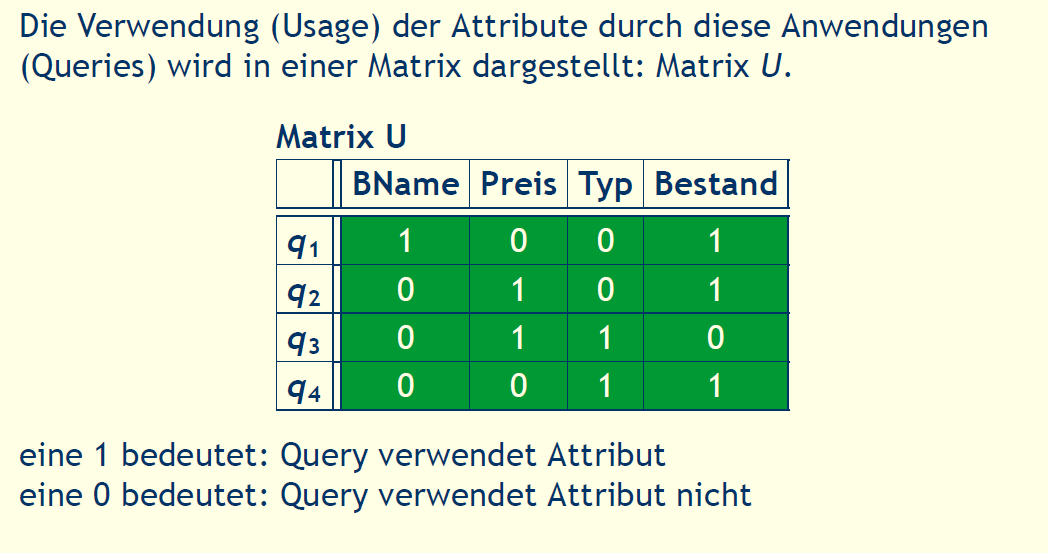
### Datenbank Beispiel

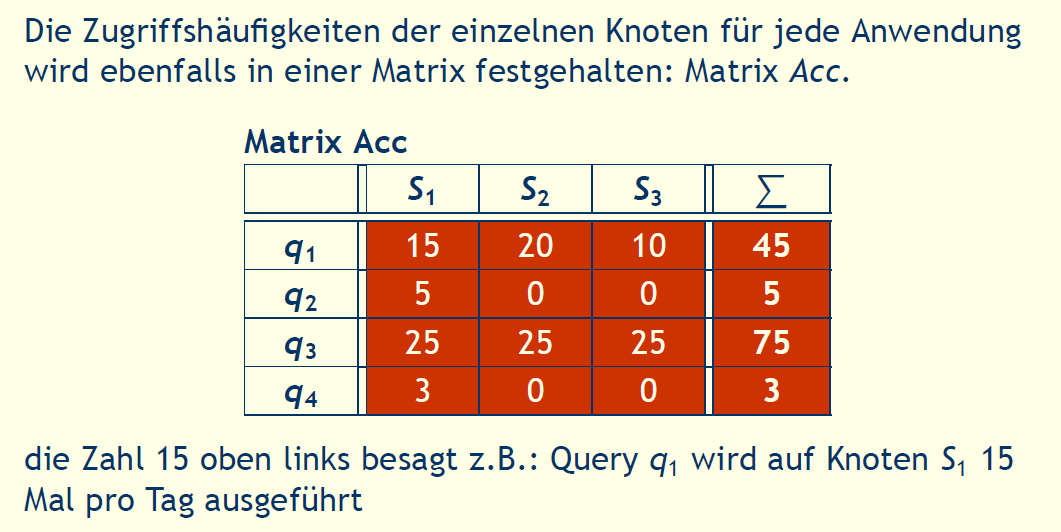


## Zugriffseigenschaften

### VF – Zugriffseigenschaft





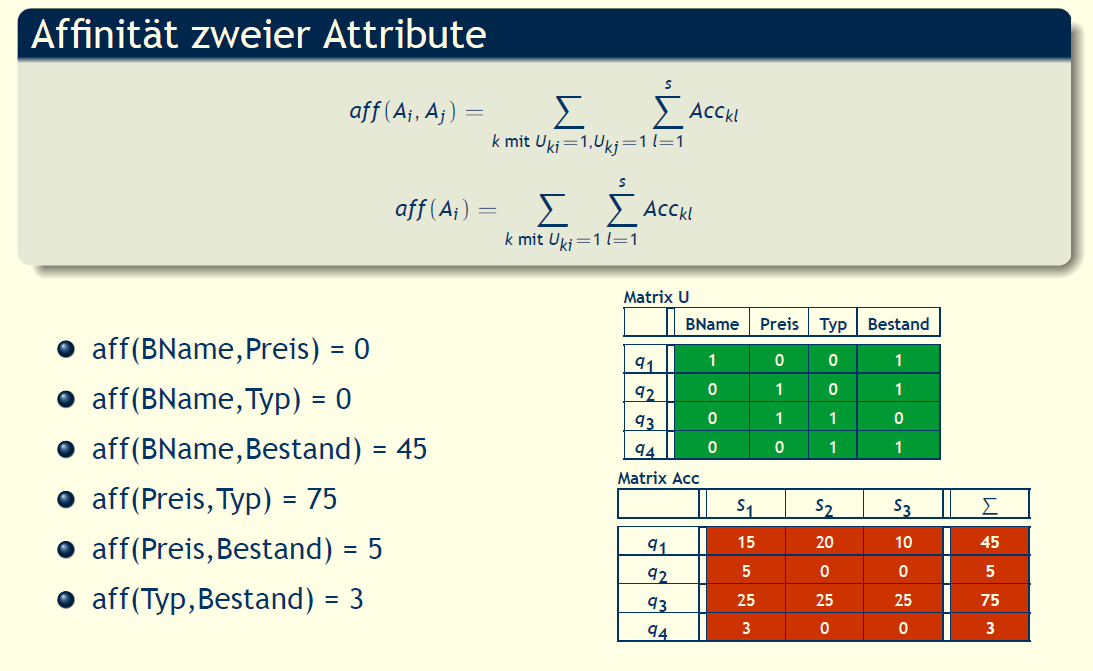


## Cluster Methode

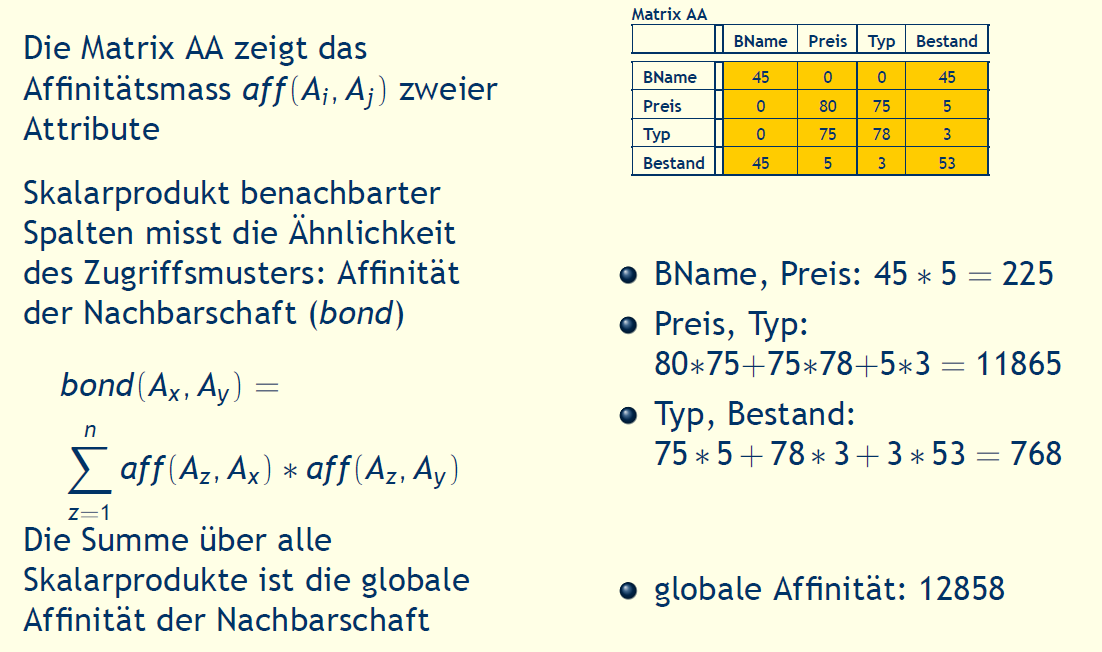
### VF – Cluster Methode



### VF – Affinität

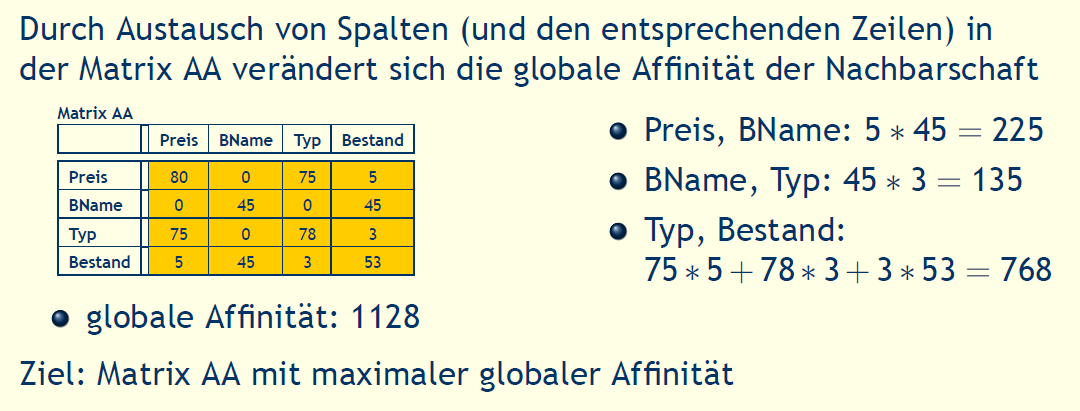


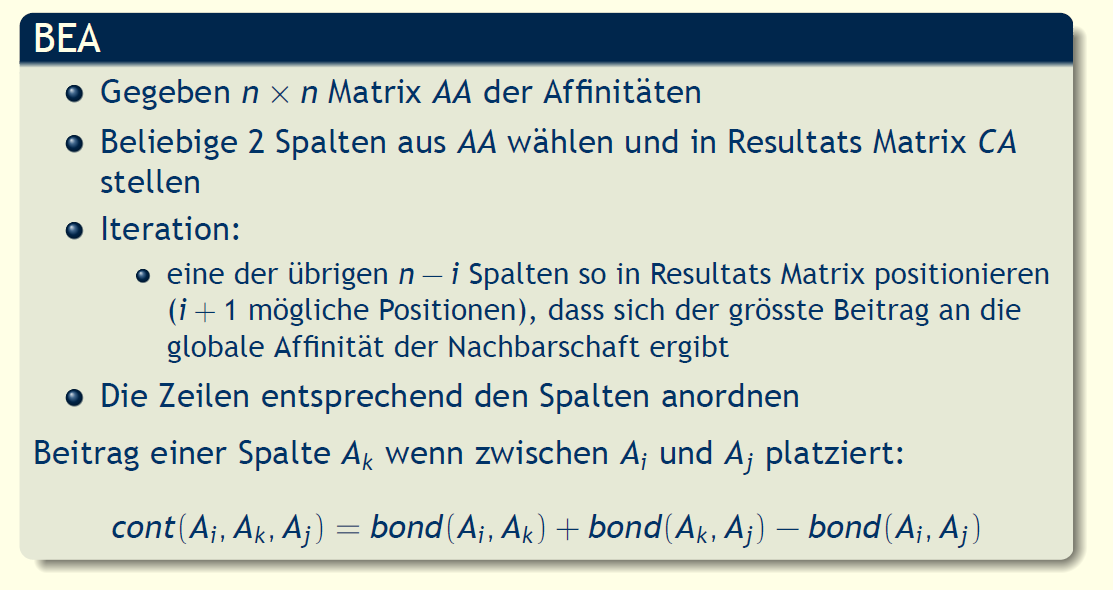
### VF – Affinitätsmatrix



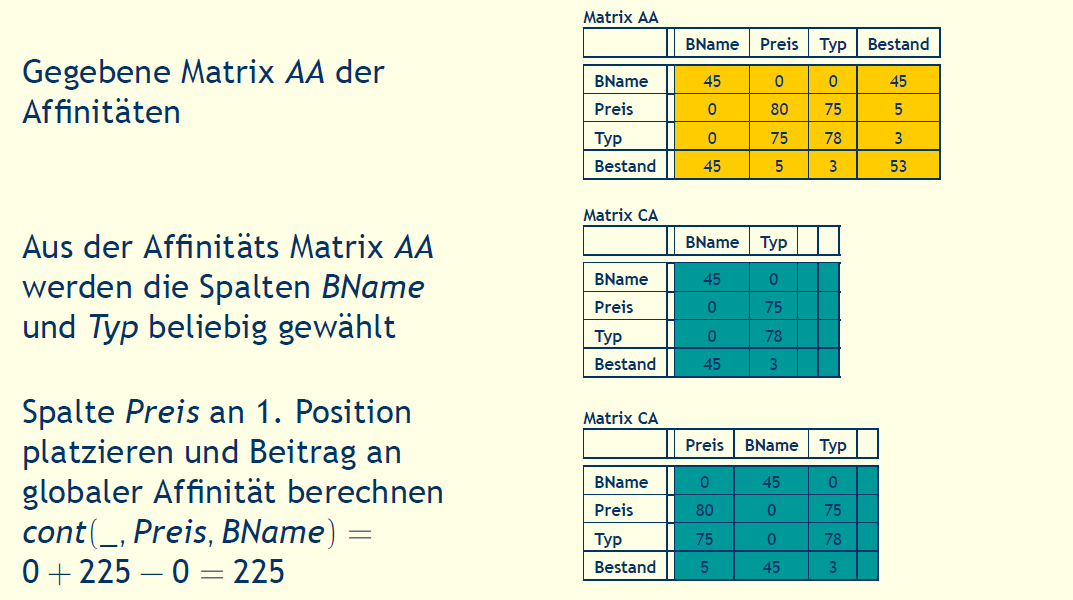
## Bond Energy Algorithm

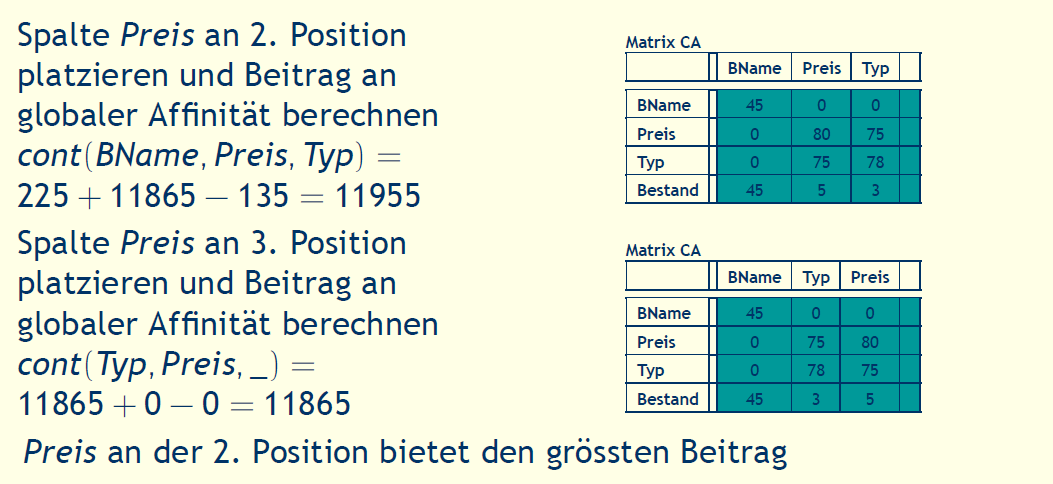
### VF – Bond Energy Algorithm

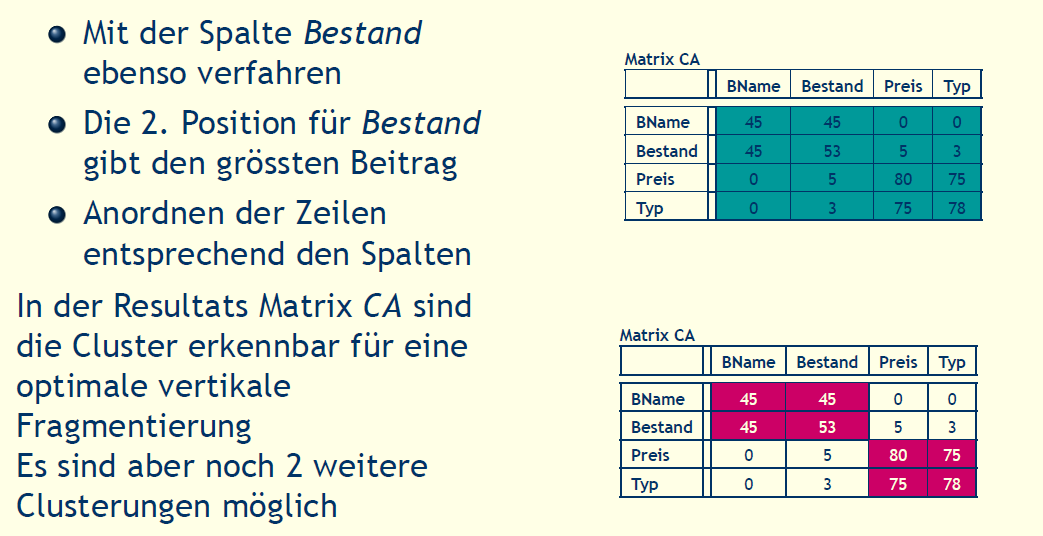




### VF- Beispiel

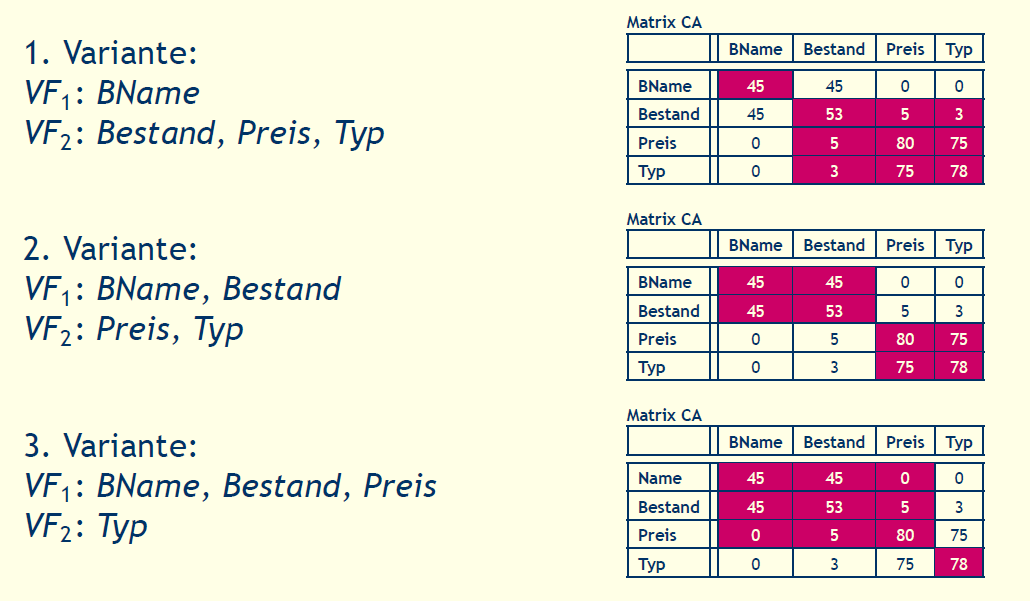




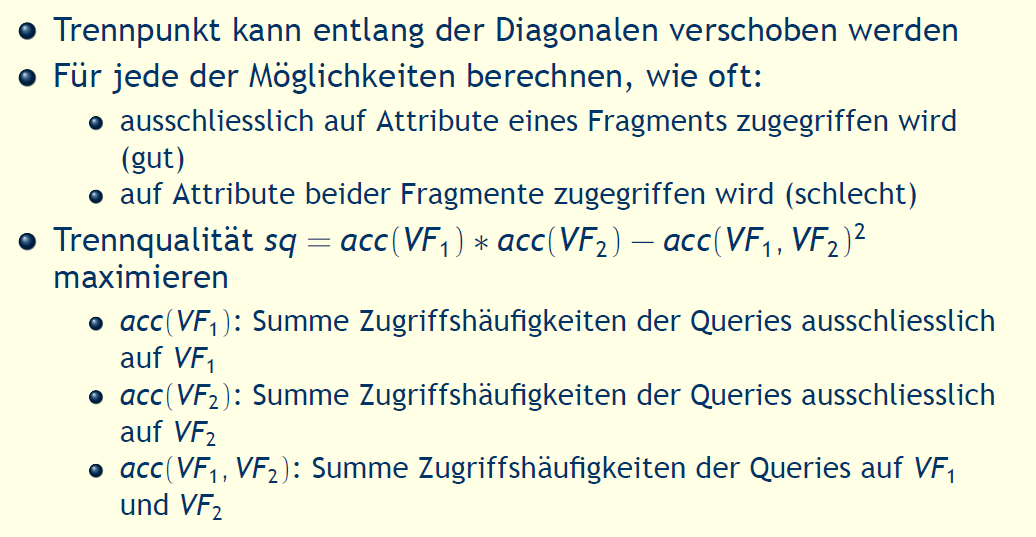


## Splitting

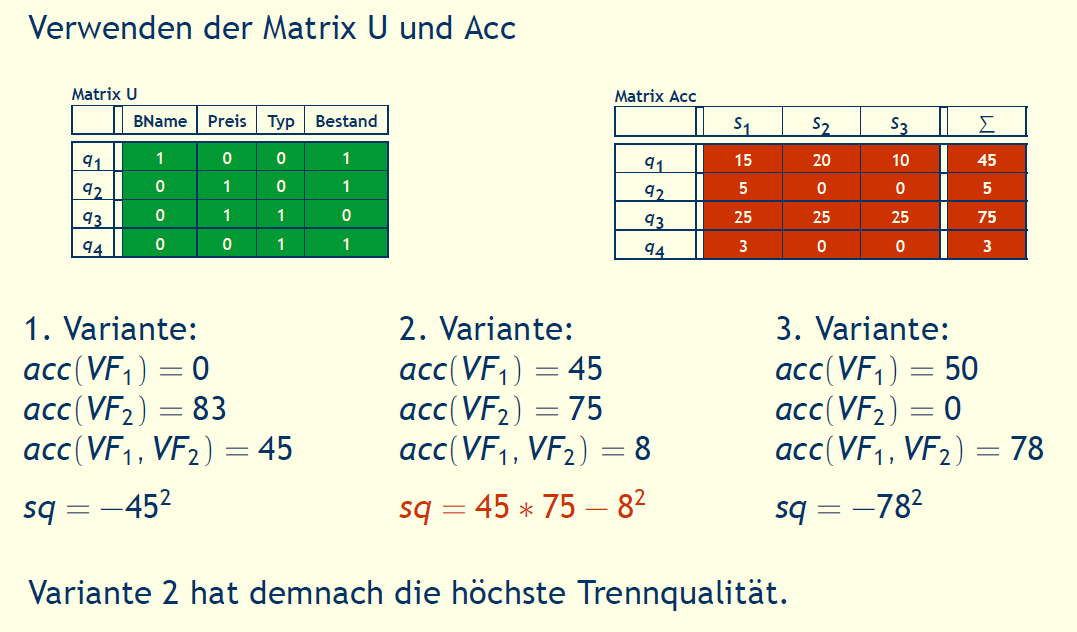
### VF – mögliche Cluster



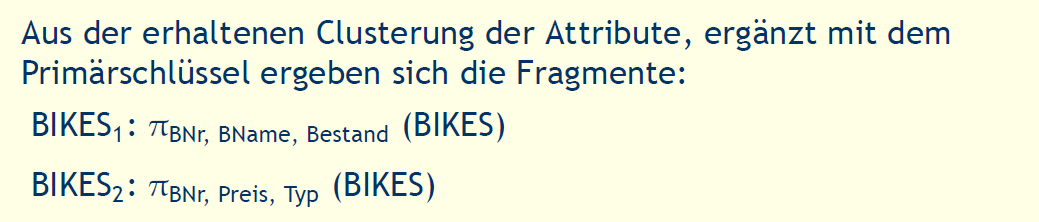
### VF – Splitting



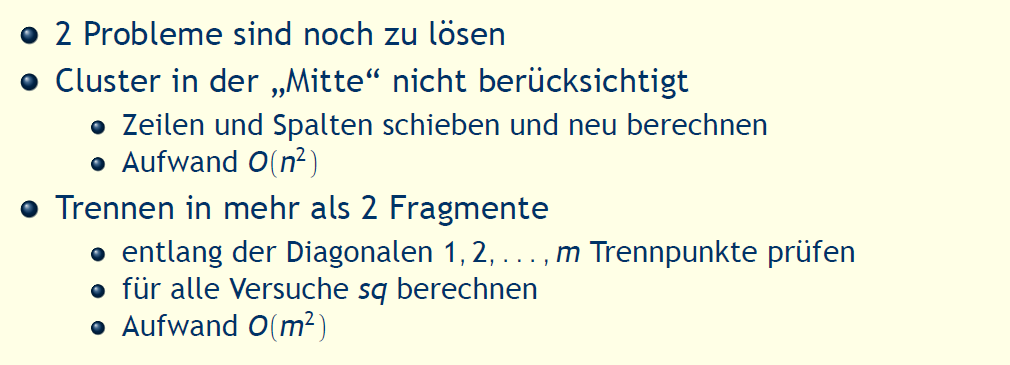
### VF – Beispiel



### Resultierende Fragment

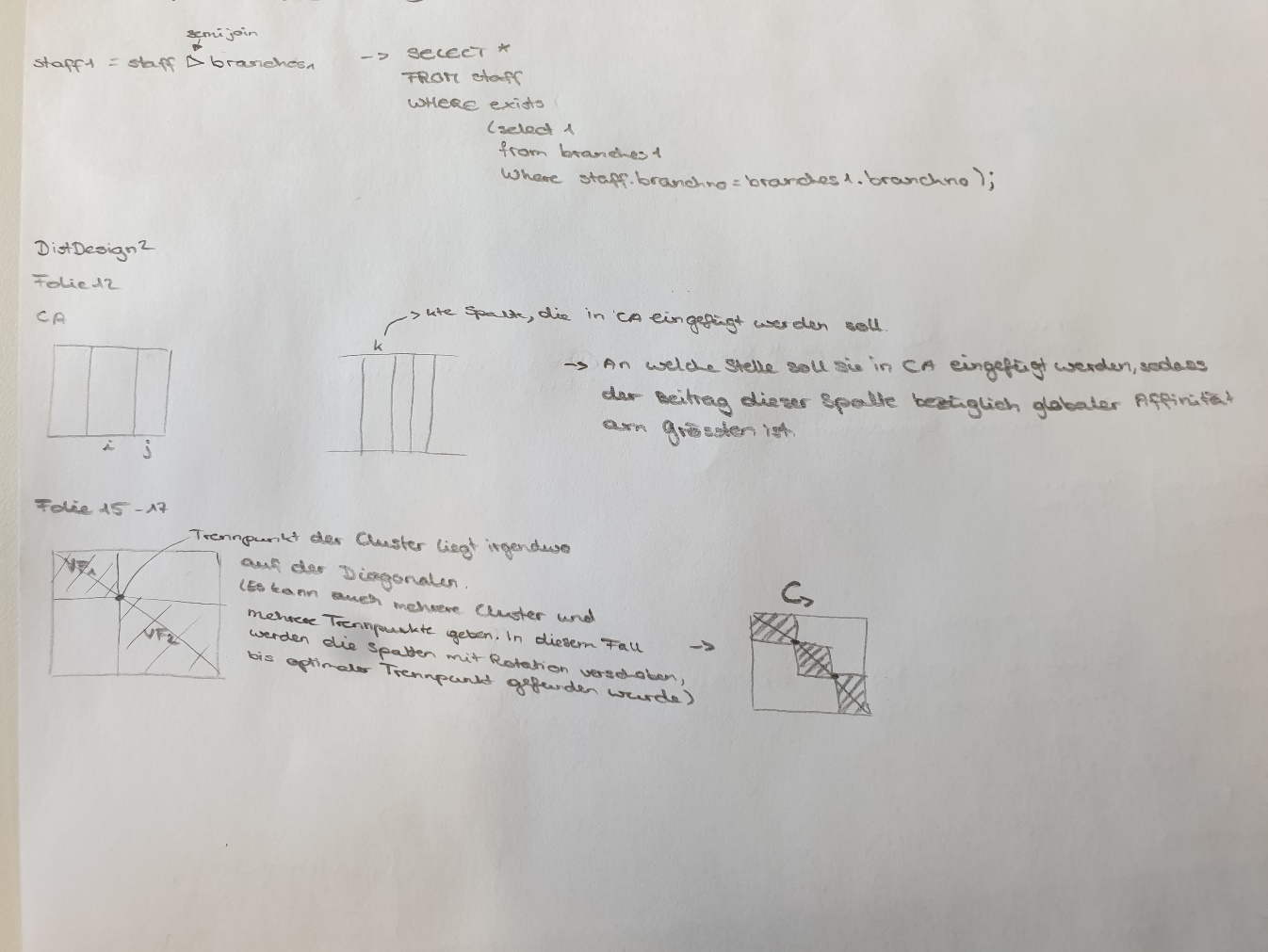


### VF – Rotation



## Notizen





# Woche 4