

---

# **Uni DB1 Notes**

Notes for the DB1 (databases) course at HdM Stuttgart

Felix Pojtinger

2022-02-01

# Inhaltsverzeichnis

<b>1</b>	<b>Aufbau eines DBMS</b>	<b>4</b>
1.1	Typen von Daten . . . . .	4
1.2	Programmaufbau . . . . .	4
1.3	Erweitertes Programm-Modell . . . . .	4
1.4	Definition DBMS . . . . .	4
1.5	Effizienz-Typen . . . . .	5
1.6	Services . . . . .	5
1.7	Metadaten . . . . .	5
1.8	Interne Dateistruktur . . . . .	5
1.9	Hintergrundprozesse . . . . .	6
1.10	Logging . . . . .	6
1.11	Datenbank vs. Schema . . . . .	6
1.12	Verwendungszwecke für Views . . . . .	6
<b>2</b>	<b>Keys</b>	<b>7</b>
2.1	Definition Candidate Key . . . . .	7
2.2	Definition Primary Key . . . . .	7
2.3	Definition Foreign Key . . . . .	7
<b>3</b>	<b>Skripte</b>	<b>7</b>
3.1	Restartfähige Skripte . . . . .	7
3.2	Delta-Skripte . . . . .	7
<b>4</b>	<b>Mengenoperationen</b>	<b>8</b>
4.1	Typen von Multi-Tabellen-Abfragen: . . . . .	8
4.2	Optimierung von Additiven Mengenoperationen . . . . .	8
4.3	Inner- vs Outer-Join . . . . .	8
4.4	Weitere Joins . . . . .	8
<b>5</b>	<b>Modellierung</b>	<b>9</b>
5.1	Datenbankentwurfsablauf . . . . .	9
5.2	Abbildungsprozess . . . . .	9
5.3	Grundsätze der Modellbildung . . . . .	9
5.4	Anforderungsdokument . . . . .	10
5.5	ER-Modell . . . . .	10
5.6	Redundanz-Anomalien . . . . .	10
5.7	Normalformen . . . . .	11

5.8	Ablauf des Schemaentwurfs . . . . .	11
5.9	Indizierung . . . . .	11
5.9.1	Problemfelder von Indizes . . . . .	11
5.9.2	Spaltenwahl für Indizes . . . . .	11
<b>6</b>	<b>Weitere Services</b>	<b>12</b>
6.1	Authorisierungsdienst . . . . .	12
6.2	Mehrnutzerbetrieb . . . . .	12
6.3	Zuverlässigkeit . . . . .	12
6.4	Transaktionen/ACID . . . . .	12
6.5	Transaktionskontrolle . . . . .	12
6.6	Konsistenzsicherung . . . . .	13
6.7	Parallelitätssteuerung . . . . .	13
6.8	Möglichkeiten der Einbindung . . . . .	13
6.9	Impedance Mismatch . . . . .	13
6.10	Definition Cursor . . . . .	14

“The true courage is to admit that the light at the end of the tunnel is probably the headlight of another train approaching” - Slavoj Žižek, *The Courage of Hopelessness*

Mehr Details unter <https://github.com/pojntfx/uni-db1-notes>.

## 1 Aufbau eines DBMS

### 1.1 Typen von Daten

- **Persistent:** Über mehrere Programmläufe verfügbar
- **Temporär:** Nur während der Laufzeit verfügbar

### 1.2 Programmaufbau

Applikation → DBMS → Datenbank

### 1.3 Erweitertes Programm-Modell

- Präsentationsschicht (temporäre Daten)
- Logik-Schicht (temporäre Daten)
- Daten-Zugriffs-Schicht (temporäre Daten)
- API zwischen App und DBMS
- DBMS (persistente Daten)
- Datenbank (persistente Daten)

### 1.4 Definition DBMS

- Sammelt Daten
- Verwaltet Daten
- Definiert Struktur (Modell)
- Definition, Manipulation & Abfrage von Daten
- Services

## 1.5 Effizienz-Typen

Entwickler:

- Effiziente Modellierung
- Einfache Sprache
- Gutes Tooling

Admins:

- Effiziente Ressourcennutzung
- Einbindung in Systemverwaltung
- Monitoring
- Anpassung
- Zugriffssteuerung

## 1.6 Services

SMARASTD:

- Service to ensure Data Integrity
- Multi-User Support
- Authorization Service
- Recovery Service
- Access via Network
- Storage, Query & Manipulation of Data
- Transactional Support
- Data Dictionary & System Catalog

## 1.7 Metadaten

- **Data Dictionary:** Metadaten der DB-Objekte
- **System Catalog:** Status und Konfiguration

## 1.8 Interne Dateistruktur

- $n$  Datendateien ( $n = 1 \dots MAXDATAFILES$ )
- Control-Files
- Logfiles

## 1.9 Hintergrundprozesse

- **DBWR** (Database Write-Prozess): Lesen & Schreiben auf Daten-Dateien
- **LGWR** (Log Write-Prozess): Logging aller Veränderungen
- **PMON** (Process-Monitor): Garbage Collector; führt in konsistenten Zustand nach Abbruch von z.B. einer Transaktion
- **SMON** (System-Monitor): Consistency Check; führt in konsistenten Zustand nach Crash von DBMS, OS oder Hardware
- **ARCH** (Archiv-Prozess): Archivierung von Daten

## 1.10 Logging

Notwendig für ...

- Konsistenz
- Wiederherstellbarkeit

Log-Dateien sind ...

- Groß: Ineffizienter Zugriff
- Wichtig: Verlust muss vermieden werden

→ Round-Robin-Prozess mit Archiv-Prozess

## 1.11 Datenbank vs. Schema

- Datenbank: Objekte zusammen von DBMS **verwaltet**
- Schema: Objekte zusammen von DBMS **betrachtet**

## 1.12 Verwendungszwecke für Views

- Trennung der Anwendungsschicht vom Unternehmensmodell (menhir)
- Sicherheit
- Zugriffsstrukturen
- Vereinfachte Schemaevolutionen
- Einfügen und Löschen einschränken

→ Am besten immer nur via Views auf Daten zugreifen

## 2 Keys

### 2.1 Definition Candidate Key

Ein Key is eine Menge von Spalten.

- **Eindeutigkeit:** Es gibt keine zwei Zeilen mit demselben Candidate Key
- **Irreduzibilität:** Nimmt man eine oder mehrere Spalten aus dem Key, so ist dieser nichtmehr eindeutig.

### 2.2 Definition Primary Key

Ein gewählter Candidate Key (oft der mit der kleinsten Anzahl von Spalten).

### 2.3 Definition Foreign Key

Es werden zwei Tabellen A und B betrachtet.

Der Foreign Key, welcher B aus A referenziert, ist ein Candidate Key von B (meist der Primary Key).

## 3 Skripte

### 3.1 Restartfähige Skripte

1. Löschen Constraints
2. Löschen Objekte
3. Anlegen Objekte
4. Anlegen Constraints

### 3.2 Delta-Skripte

Bei einer Erweiterung des Modells dürfen bestehende Daten nicht ungültig werden.

- `alter table`: Neue Spalte einfügen
- `update`: Default-Werte für alte Zeilen einfügen (falls `not null`)
- `insert`: Fehlende Zeilen anlegen (falls `not null`)
- `alter table`: Foreign Key-Constraint hinzufügen
- `alter table`: `not null`-Constraint hinzufügen

## 4 Mengenoperationen

### 4.1 Typen von Multi-Tabellen-Abfragen:

- **Additive** Mengenoperationen: Mehrere Teilabfragen (`in` etc.)
- **Multiplikative** Mengenoperationen: Kartesisches Produkt (`join` etc.)

### 4.2 Optimierung von Additiven Mengenoperationen

Wenn Abfragen über mehrere Tabellen gemacht werden, so müssen alle Abfragen fertig sein, damit verglichen werden kann. Deshalb `union all` verwenden (Vorsicht: Duplikate werden nicht entfernt!)

### 4.3 Inner- vs Outer-Join

- **Inner Join**: Zeilen in linker Tabelle, für welche in der rechten Tabelle keine entsprechenden Zeilen existieren, werden nicht dargestellt.
- **Outer Join** (+): Zeilen in Tabelle A, für welche in Tabelle B keine entsprechende Zeilen existieren, werden mit `null` gefüllt.
  - **Left Outer Join**: Rechts kann `null`-Werte haben
  - **Right Outer Join**: Links kann `null`-Werte haben
  - **Full Outer Join**: Beide könnten `null`-Werte haben

### 4.4 Weitere Joins

- Bulk Join (Kartesisches Produkt)
- Restricted Join (mit zwei Where-Bedingungen)
- Natural Join (min. ein Attribut gleich)
- Semi Join (nur Attribute einer Tabelle im `select`-Statement)
- Multiple Join (z.B. `join` aus drei Tabellen)
- Auto Join (Tabelle mit sich selbst `joinen`; z.B. Stückliste)



## 5 Modellierung

### 5.1 Datenbankentwurfsablauf

1. **Input:** Reale Welt
2. Anforderungen analysieren
3. Entwurf (konzeptionell) erstellen
4. Entwurf (logischen) erstellen
5. Implementieren
6. **Output:** System

Dabei wird nebenläufig kontinuierlich getestet.

### 5.2 Abbildungsprozess

- **Realwelt**
  - Vielschichtig
  - Unikate
  - Umfangreiche Beziehungen
- **Semantisches Datenmodell**
  - Zusammenfassung zu Gruppen, abstrahiert
  - Integritätsbedingungen
  - Explizit modellierte Beziehungen
- **Relationales Datenbankmodell**
  - Einfach
  - Tabellen
  - Implizit modellierte Beziehungen

### 5.3 Grundsätze der Modellbildung

SSRWKV:

- Syntaktische & semantische Richtigkeit
- Systematischer Aufbau
- Relevanz
- Wirtschaftlichkeit

- Klarheit
- Vergleichbarkeit

## 5.4 Anforderungsdokument

Ein gutes Anforderungsdokument sollte die Eigenschaften haben ...

- Korrektheit
- Vollständigkeit
- Konsistenz
- Einfachkeit
- Eindeutigkeit

Ein gutes Anforderungselement sollte bestehen aus ...

- Informationsanforderungen
- Bearbeitungsanforderungen
- Funktionale Anforderungen
- Dynamische Anforderungen

## 5.5 ER-Modell

- **Atribut:** Datenelement
- **Entität:** Gruppierungselement
- **Beziehung:** Verknüpfung ( $n:m$ -Beziehungen via schwacher Entität)
- **Kardinalität:** Maximale Anzahl an Elementen in Beziehung



## 5.6 Redundanz-Anomalien

Folgende Anomalien treten durch Redundanzen auf:

- Änderungsanomalie
- Löschanomalie
- Einfügeanomalie

## 5.7 Normalformen

- **Erste Normalform:** Spalten sind nicht weiter auftrennbar
- **Zweite Normalform:** Alle Attribute hängen vom Schlüssel ab (keine funktionalen Abhängigkeiten)
- **Dritte Normalform:** Beziehungen werden über Foreign Key-Constraints abgebildet (keine transitiven Abhängigkeiten)

## 5.8 Ablauf des Schemaentwurfs

1. Erheben von Infos
2. Identifikation der Attribute
3. Formalisierung von Infos
4. Gruppierung der Attribute

## 5.9 Indizierung

### 5.9.1 Problemfelder von Indizes

- Im temporären Speichern funktionieren Indizes nicht mehr
- Falsche Anwendung von Indizes kann sogar langsamer als keine Indizes sein. Ohne Indizes werden immer alle Zeilen einer Tabelle evaluiert; bei Indizes wird immer von einer Position ausgehend, bis die *where*-Clause eintritt, evaluiert. Letztere Strategie besitzt damit auch einen Overhead, welcher teurer als die Ersparnis durch das frühere Abbrechen nach dem Eintreten der *where*-Clause sein kann.

### 5.9.2 Spaltenwahl für Indizes

Bei der Erstellung eines Indexes sollte immer die Spalte mit der höchsten Selektivität ( $> 0,8$ ) zuerst angegeben werden, welche sich mit folgender Formel berechnen lässt:

$$\text{Selektivität} = 1 - \frac{n - \text{distinct}(n)}{n}$$

$n$ : Anzahl von Elementen

$\text{distinct}(n)$ : Anzahl von eindeutigen Elementen

## 6 Weitere Services

### 6.1 Authorisierungsdienst

Nutzt eine Allowlist.

- Beschränkung von Nomen (i.e. “Nutzer x darf auf Tabelle `products` zugreifen”): **Objektprivilegien**
- Beschränkung von Prädikaten (i.e. “Nutzer x darf `updaten`”): **Systemprivilegien**

### 6.2 Mehrnutzerbetrieb

- Sichtbarkeit von Daten
- Änderbarkeit von Daten
- Trennung in Anwender und Admins
- Schonung von Ressourcen
- Einfache Verwaltung

### 6.3 Zuverlässigkeit

Daten dürfen weder physisch noch semantisch fehlerhaft sein, weshalb folgende Dinge existieren müssen:

- Transaktionen
- Virtueller Single-User-Betrieb

### 6.4 Transaktionen/ACID

Aktionen werden entweder vollständig oder gar nicht ausgeführt.

- **Atomicity**: Alles oder nichts
- **Consistency**: Zustand 1 → Zustand 2 (Unterbrechung: Zustand 2 = Zustand 1)
- **Isolation**: Virtueller Single-User-Betrieb
- **Durability**: Zustand 2 bleibt erhalten, egal was passiert

### 6.5 Transaktionskontrolle

- `begin`: Start einer Transaktion (SQL: Nicht definiert)

- **end**: Ende einer Transaktionen (SQL: `commit`)
- **undo**: Verwerfen offener Transaktionen (SQL: `rollback`)
- **redo**: Wiederherstellung abgeschlossener Transaktionen (SQL: Nicht definiert)
- **savepoint**: Sub-Transaktionen (SQL only)

## 6.6 Konsistenzsicherung

- **Constraints**: In Tabellen
- **Transaktionen**: In Ablaufebene
- **Trigger**: In Prozeduralen Erweiterungen

## 6.7 Parallelitätssteuerung

Verhindern von ...

- **Lost Update**: Verlorengegangenen Änderungen
- **Dirty Read/Write**: Zugriff auf "schmutzige" Daten

Umsetzung durch ...

- Lese-, Schreib- und Exklusiv-Sperren (**Funktionale Sperr-Ebene**)
- Table-, Page- und Row-Level-Sperren (**Physische Sperr-Ebene**)

→ Z.B. durch `select ... for update of ...`

## 6.8 Möglichkeiten der Einbindung

- Low Code-Umgebungen (z.B. LibreOffice Base, IFTTT)
- Embedding
- APIs

## 6.9 Impedance Mismatch

- `too_many_rows`: Mehr als ein Datensatz
- `no_data_found`: Null Datensätze (nicht streng genommen ein Impedance Mismatch)

### **6.10 Definition Cursor**

Ergebnis einer Abfrage wird in einer Tabelle abgelegt, von welcher dann  $n$ -mal gefetched werden kann.