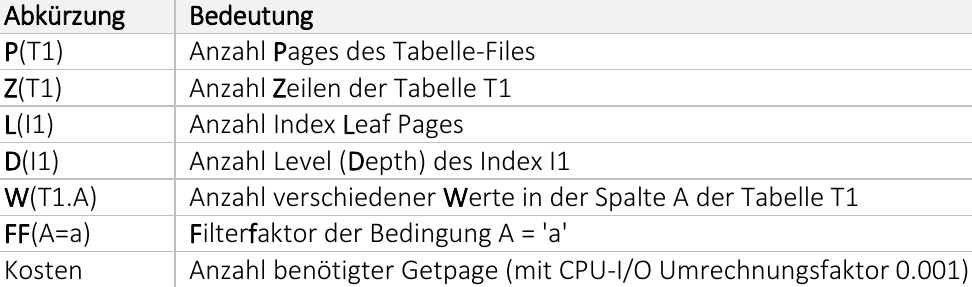
**Zusammenfassung DBF**

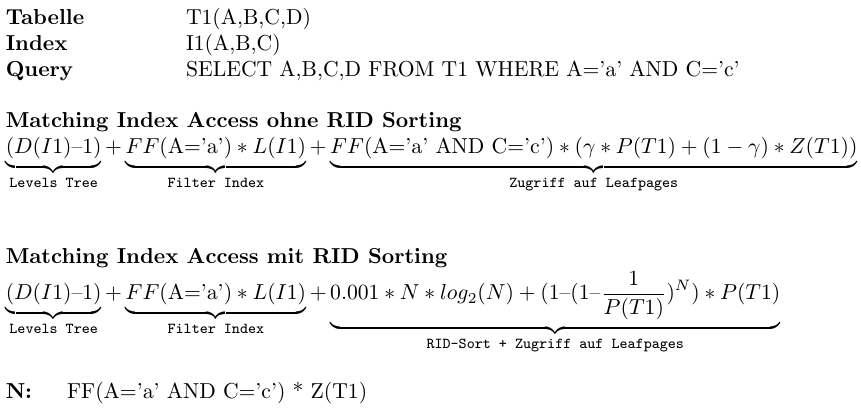
René Bernhardsgrütter – 22.06.2013

**Credits: Christoph Rehmann, Roland Heusser, Rodrigo Benz, Simon Flüeli, Martin Thürer**

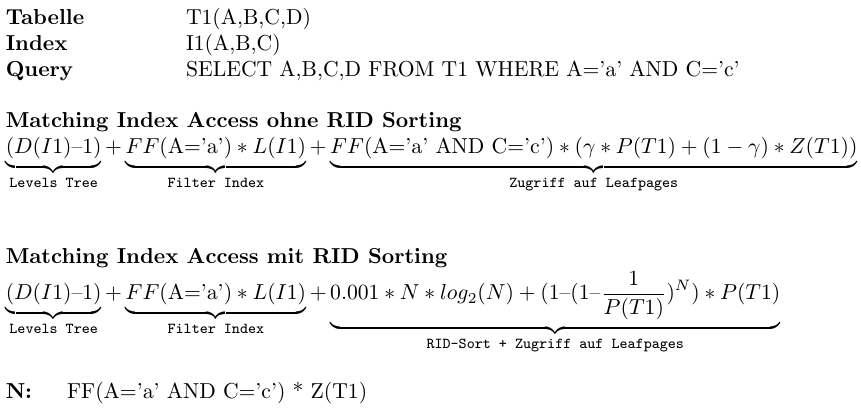
# Optimizer



## Gesamtkosten



Hier nur erster und letzter FF des Indexes verwenden!

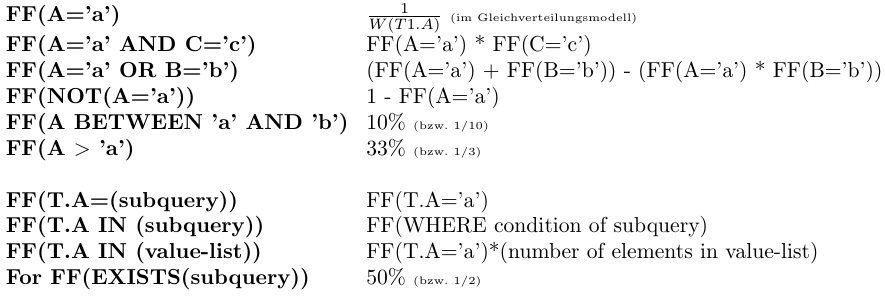


Bei 1 Index: Nur ein FF verwenden! Wenn 2 oder mehr Indices verwendet, jeweils den ersten und den letzten Index als FF verwenden!

**Wichtig**: Oben wurde via **2 Indices zugegriffen**! Wenn man nur   
**1 Index** verwenden möchte, verwendet man als **FF alleinig**:

und als **N**:

bzw.:



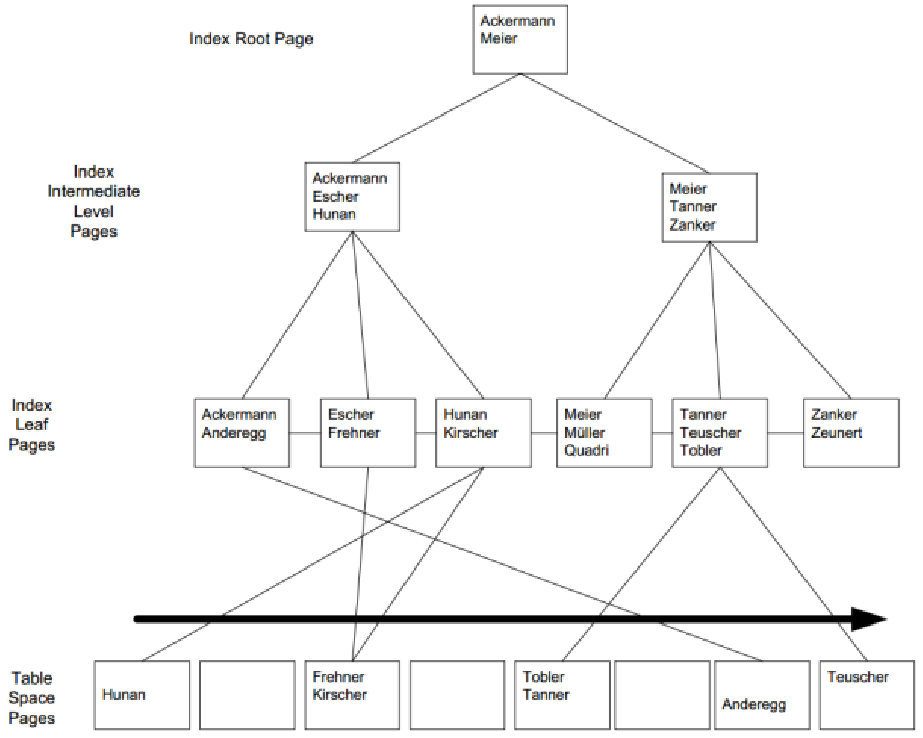
## Sortierung

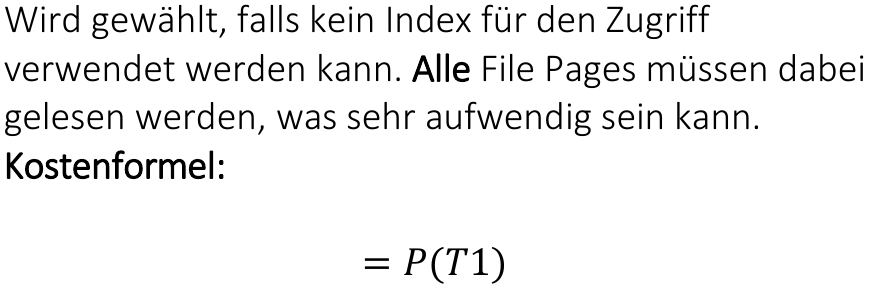
K = Kosten für das Sortieren von N RIDs

N = Anzahl RIDs

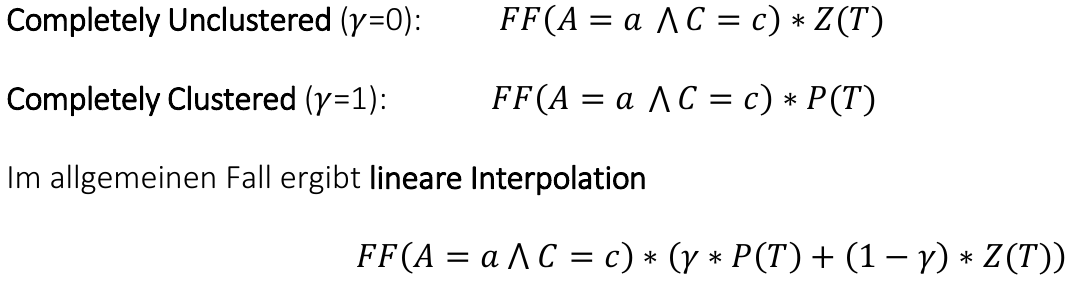
P = Porportionalitätsfaktor = 0.001

## Tablespace Scan





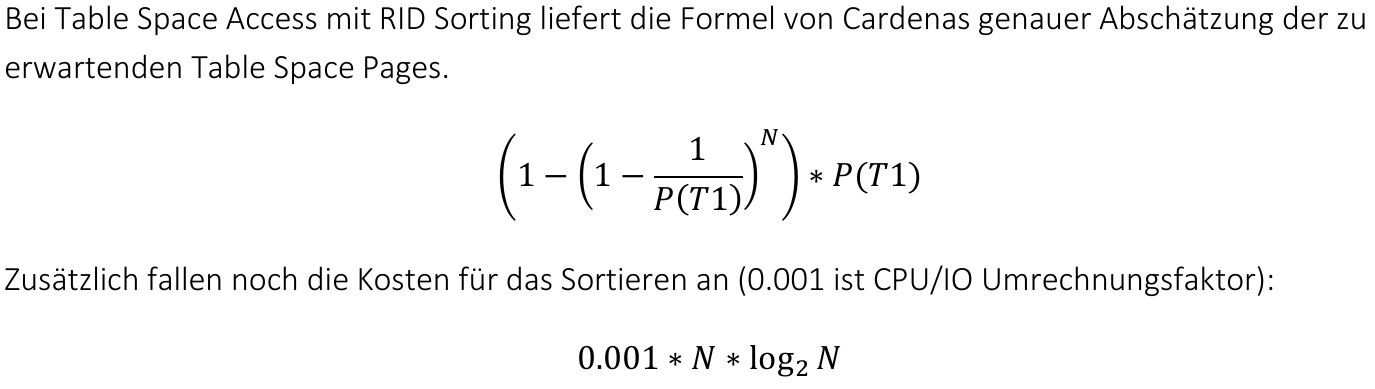
### Erwartete Table Space Pages mit linearer Interpolation



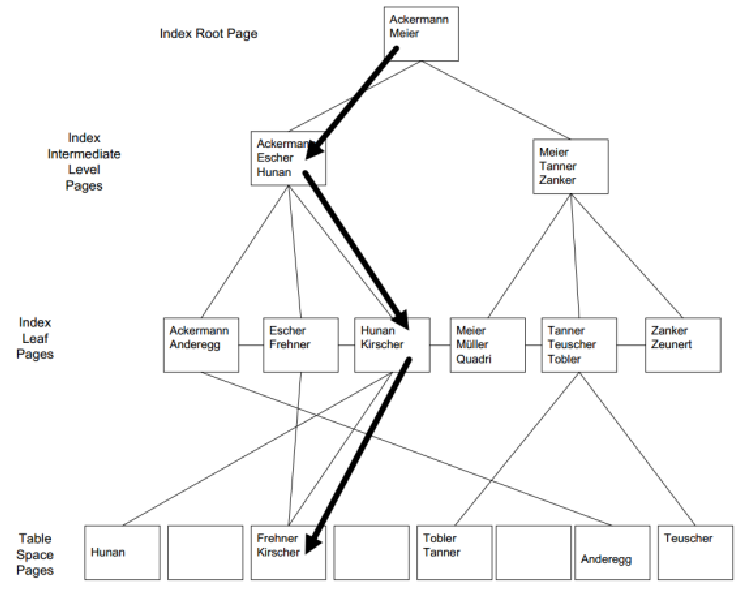
Wenn beide/alle Filterfaktoren FF **unabhängig** sind, folgt daraus:

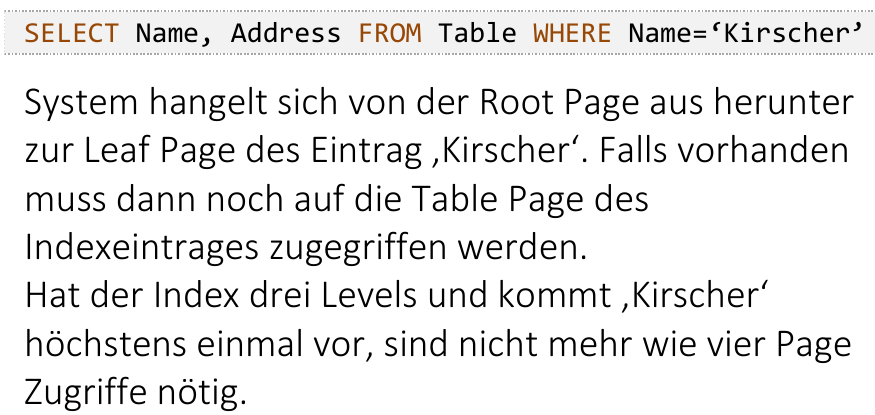


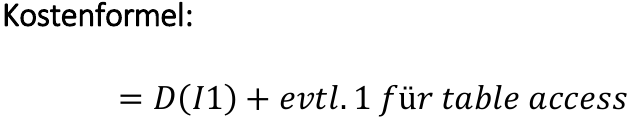
### Erwartete Table Space Pages mit Cardenas



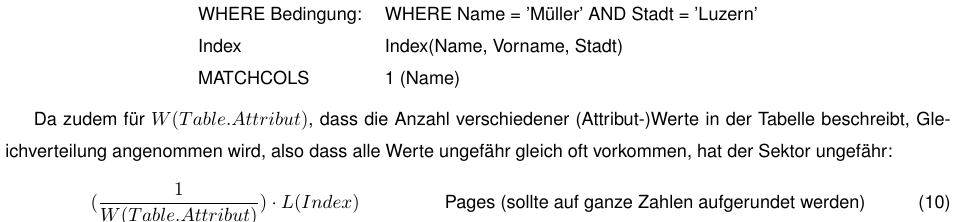
## One Fetch UNIQUE Index Access

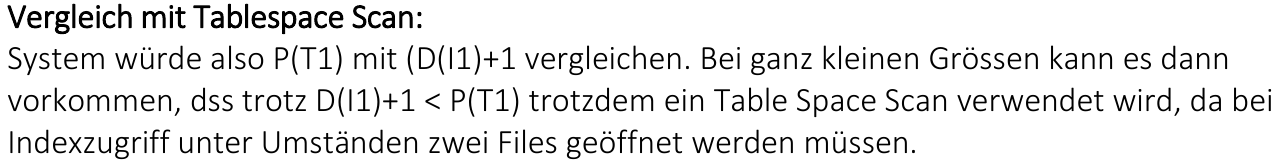




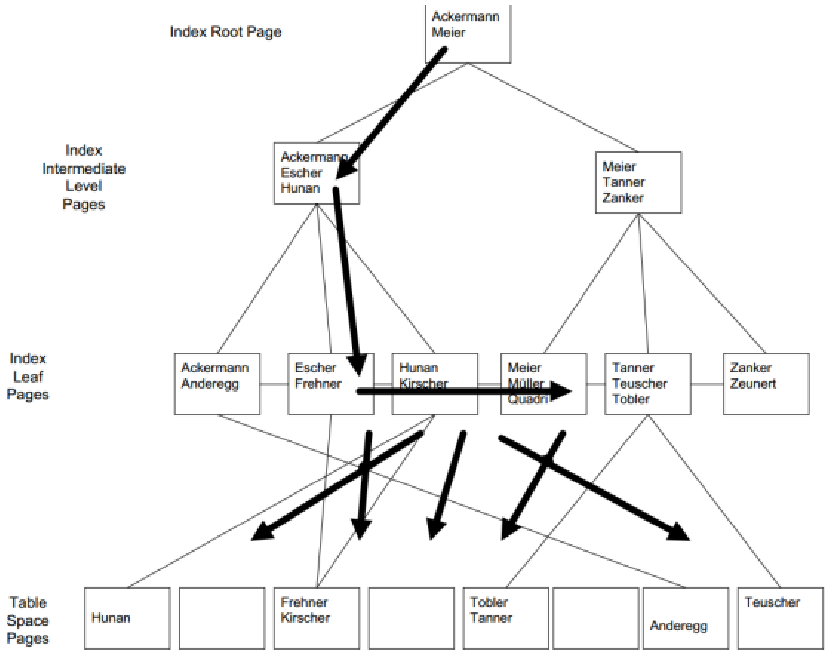


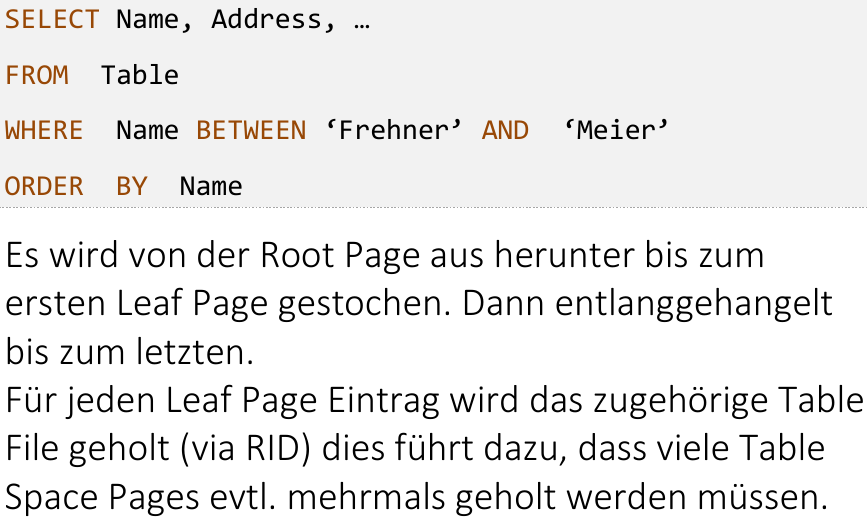
**Beispiel:**

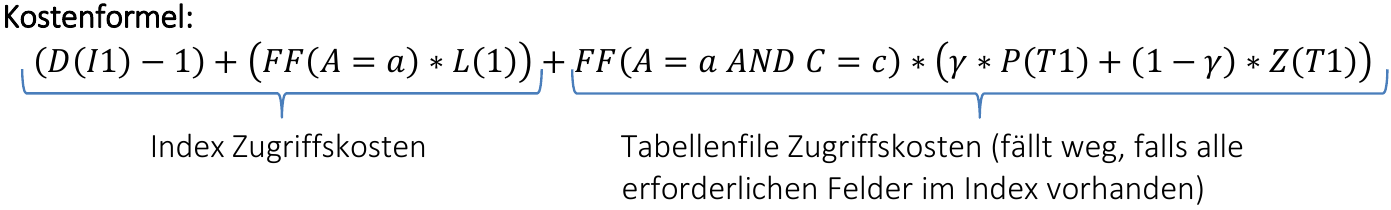




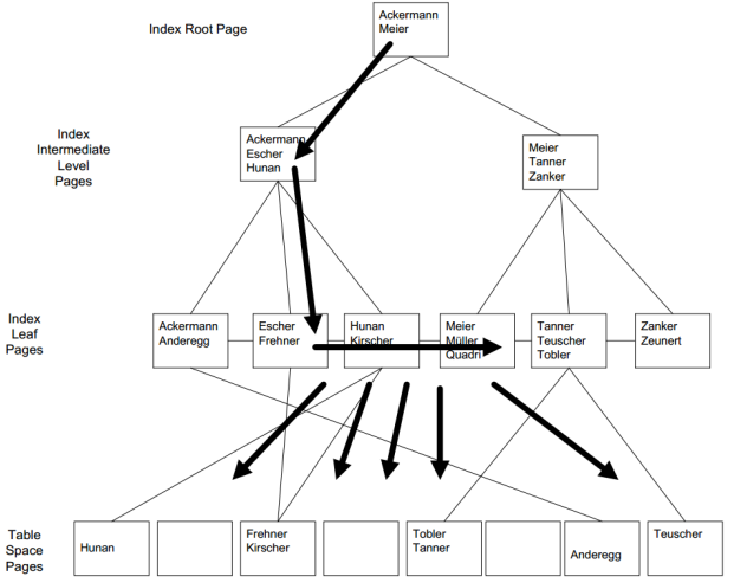
## Matching Index Access without RID Sorting

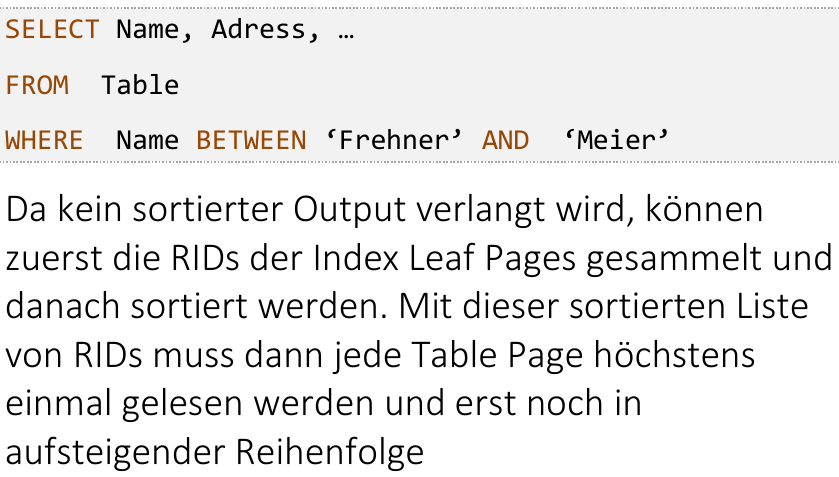


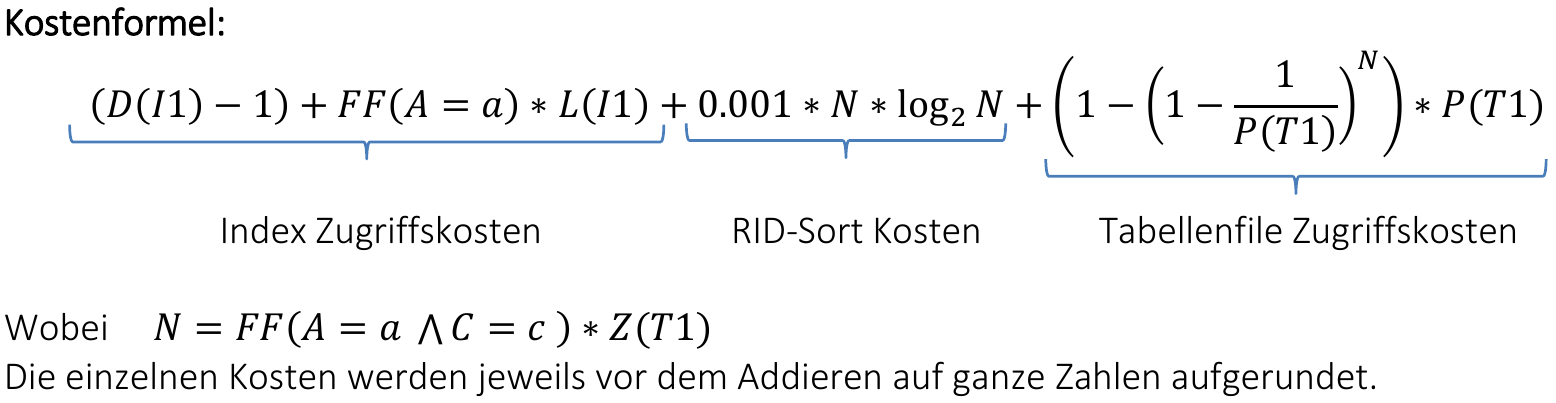




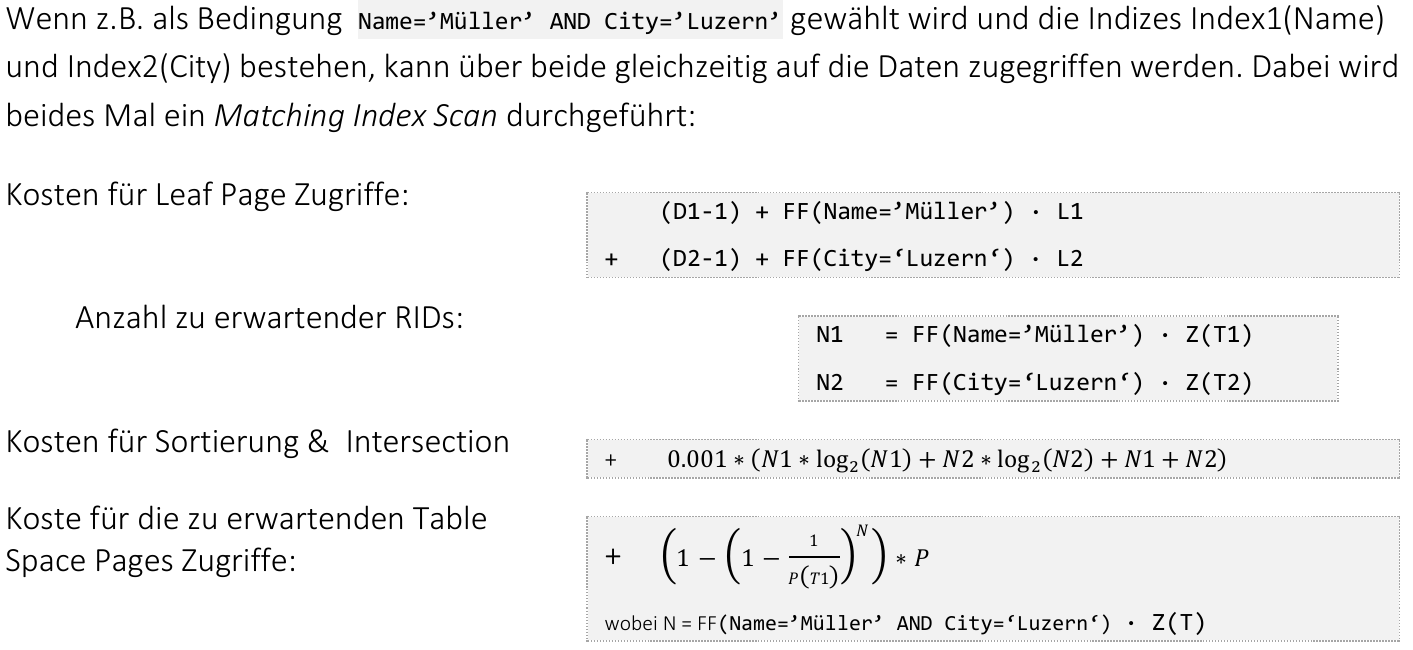
## Matching Index Access with RID Sorting



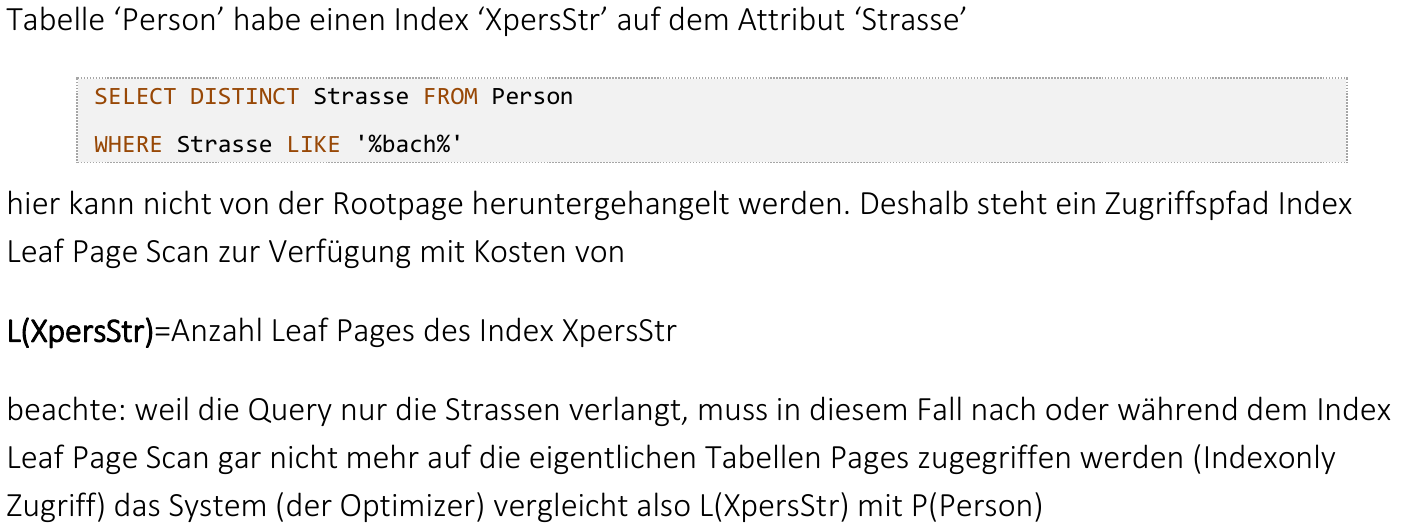




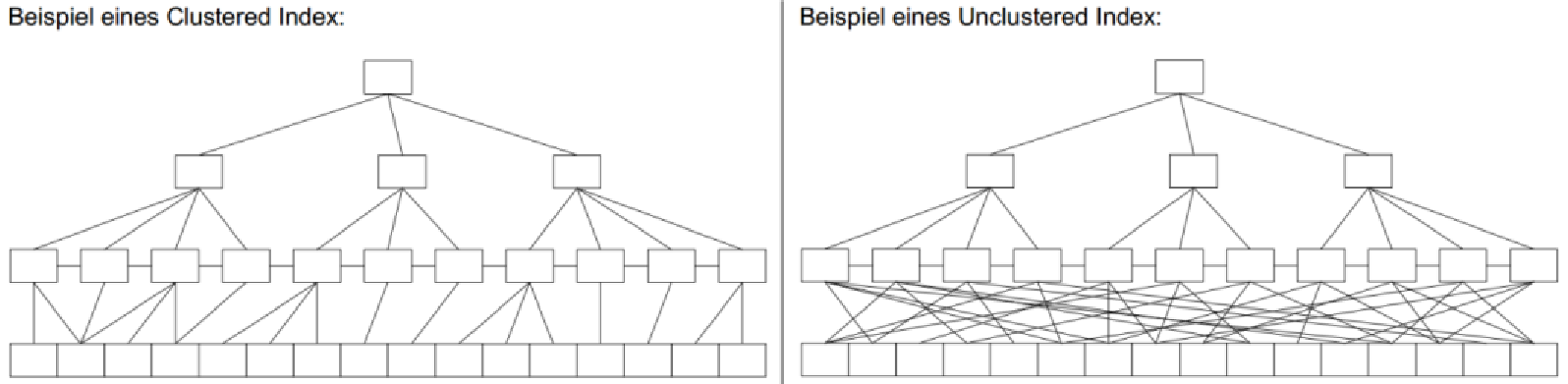
## Matching Index with Multiple Single Indices

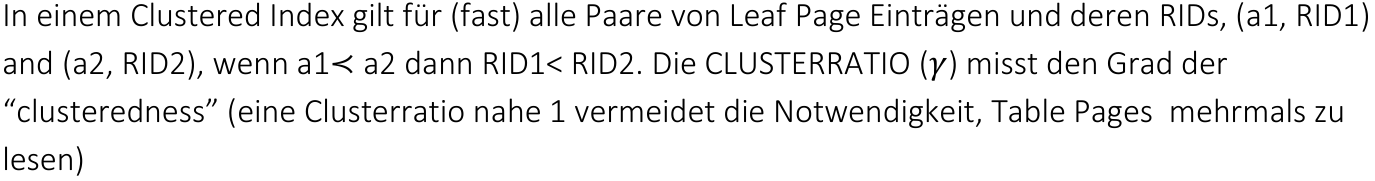


## Index Leaf Page Scan

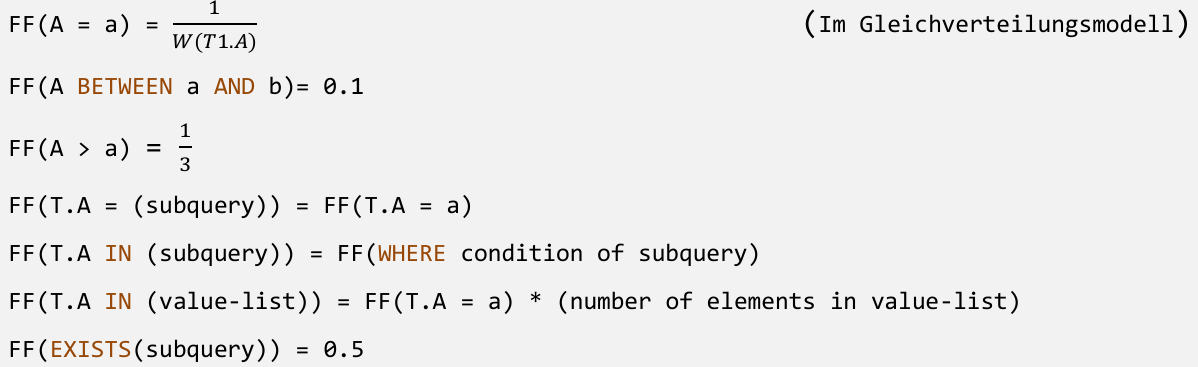


## Clustered and Unclustered Indices

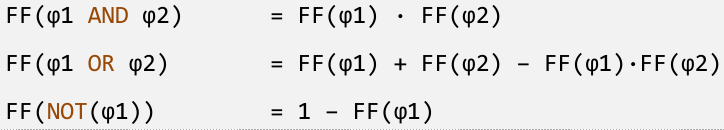




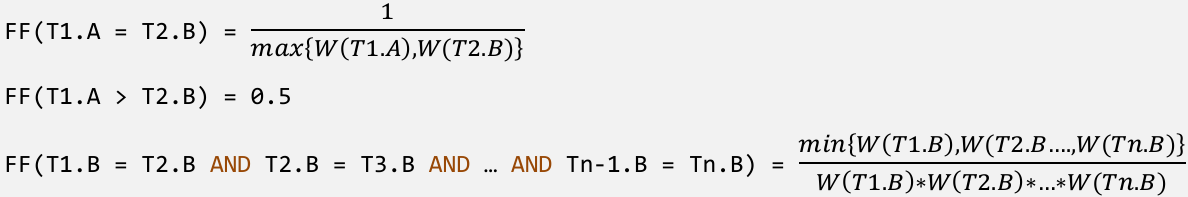
## Filterfaktor FF als Zahlenwert



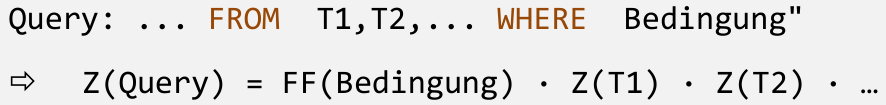
Weiter gilt (Unabhängigkeit angenommen):



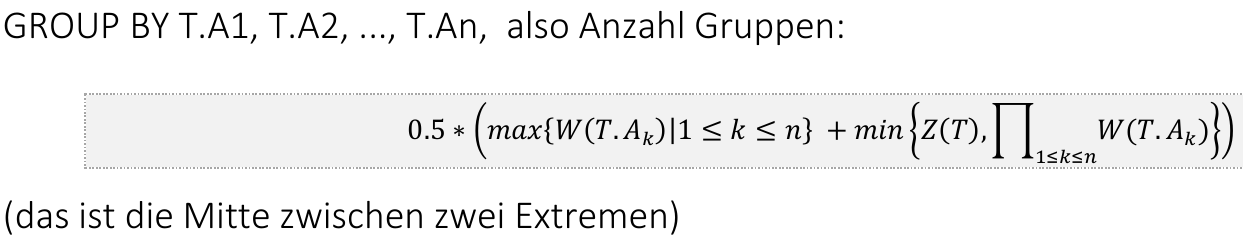
### Filterfaktoren bei JOIN Prädikaten



### Anzahl Outputzeilen bei JOIN Prädikaten



### GROUP BY



## Join Algorithmen

B = Anzahl garantiert verfügbare Bufferseiten. **Annahme: B ≥ 2.**

#### Nested Loop

Oftmals ist es günstiger die **Tabelle mit der kleineren Anzahl Seiten als äussere Tabelle** zu verwenden.

#### Nested Loop with indexed inner Table



Gegeben: **Index I2(A) über Tabelle T2**

Wenn Index **I2 UNIQUE**:

#### Sort Merge

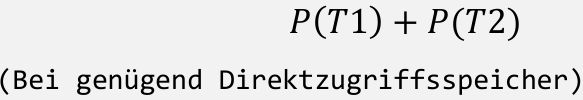
#### Sort Merge with indexed inner Table



Gegeben: **Index I2(A) über Tabelle T2**

Kosten für Sortierung der äusseren Tabelle

#### Hash Join



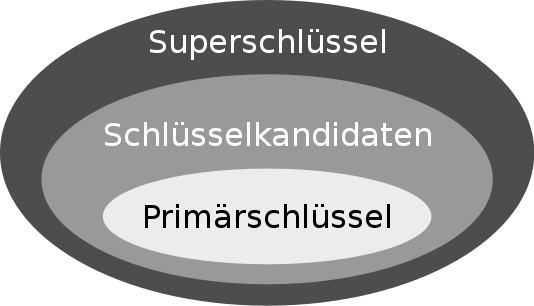
# Normalisierung

Relation: Angabe der Felder und der FDs:

## Schlüsseldefinitionen

**Schlüssel**: Minimale Attributmenge, mit der alle Attribute der Relation über die FDs erreichbar sind (min. bezieht sich dabei nicht auf die Schlüssellänge sondern darauf, dass nicht ein kürzerer Schlüssel mit Teilen davon existiert).

**Minimal**: Ein Superschlüssel/Schlüssel/ Schlüsselkandidat ist **minimal**, falls die minimale Anzahl Attribute verwendet wird, um ein Tupel (≃Tabellenzeile) eindeutig zu identifizieren.

**Superschlüssel**: Es existieren semantische und syntaktische Superschlüssel. *Sie können alle anderen Attribute generieren (via funktionale Abhängigkeiten). Sie sind nicht zwingend minimal*.

**Superschlüsselkandidat**: Möglicher Schlüssel + Minimal.

**Primärschlüssel**: Besteht aus **einem Attribut**.

**Semantischer Schlüssel**: Zeit- und inhaltsabhängiger Schlüssel.

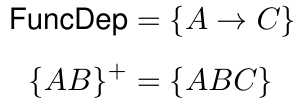
**Syntaktischer Schlüssel**: Zeitunabhängiger Schlüssel, der immer über die Relation gültig ist. Wird von DB erwzungen.

## Funktionale Abhängigkeit FD

Semantische Folgerung

Syntaktische Folgerung

Bei folgender **FD** ist der Abschluss folgendermassen:



## Vereinfachung von Mengen von FDs

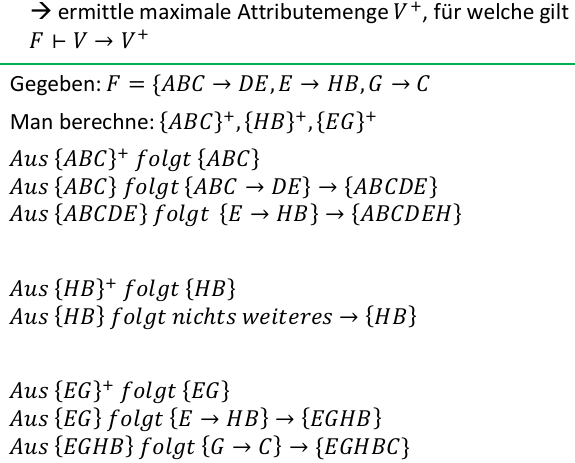
Beispiel gegeben:

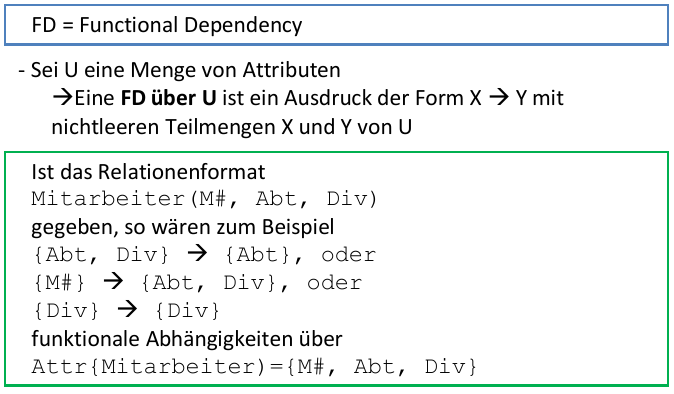
1. Frage: Redundanz enthalten?
2. Mengen der FDs durch äquivalente Mengen ersetzen, sodass **auf der rechten Seite nur noch Attribute, keine Mengen mehr** stehen:
3. Man sieht: Bei kann das weggelassen werden, denn . Man kann die ganze FD streichen.
4. Neue Menge

## Abgeschlossenheit

V+ = Maximale Attributmenge von einer Menge V

Von einer nichtleeren Attributmenge ausgehen und ermitteln,

welche Attribute daraus folgen:  




## Syntaktischer Schlüssel zu Relation finden

Ist jederzeit gültig.

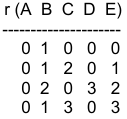
**Bsp:**

R(A,B,C,D,E); {ABC → DE, D → E}

1. Vereinfachen: {ABC → D, ABC → E, D → E}
2. Möglichst wenige Attribute finden, die die gesamte Relation abdecken: Schlüssel = {A,B,C}
3. Achtung: Es kann mehrere geben, wenn sie Minimallänge haben!

## Semantischer Schlüssel zu Relation finden

Ist abhängig von Daten gültig (kann also später ungültig sein).

**Bsp:**

Relation r gegeben:

Alle semantischen Schlüssel finden:

1. Jede Zeile muss mit dem Schlüssel eindeutig adressiert werden können.
2. Schauen, ob eine Spalte nur unterschiedliche Werte hat (E).
3. Schauen, ob mehrere Spaltenkombinationen vertikal eindeutige Werte hat.
4. Semantische Schlüssel = {E}, {B,C}, {C,D}

## Minimale Überdeckung

heisst eine **minimale Überdeckung** (minimal cover) für , falls aus durch Anwendung des folgenden Algorithmus hervorgeht:

1. Ersetze alle FDs durch solche der Form (**rechts ein einzelnes Attribut**)
2. Für alle und ,

- Falls äquivalent zu , dann ersetze mit

- 2) So lange wiederholen, bis alle minimal sind

1. Für alle

- Falls äquivalent ist zu , dann streiche aus

## Normalformen

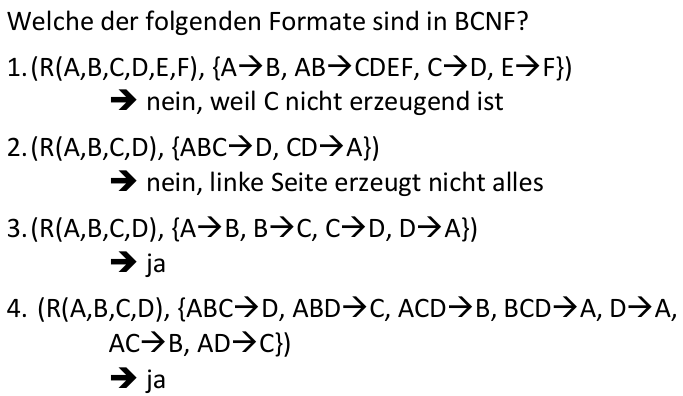
Die Normalformen sind mit unterschiedlicher Stärke definiert.

Für die verschiedenen Normalformen **muss für jede FD mindestens eine der Definitionen zutreffen**:

### BCNF

→ Linke Seite (je-)der FD ist ein Superschlüssel

**Wenn BCNF, dann auch 3NF!**

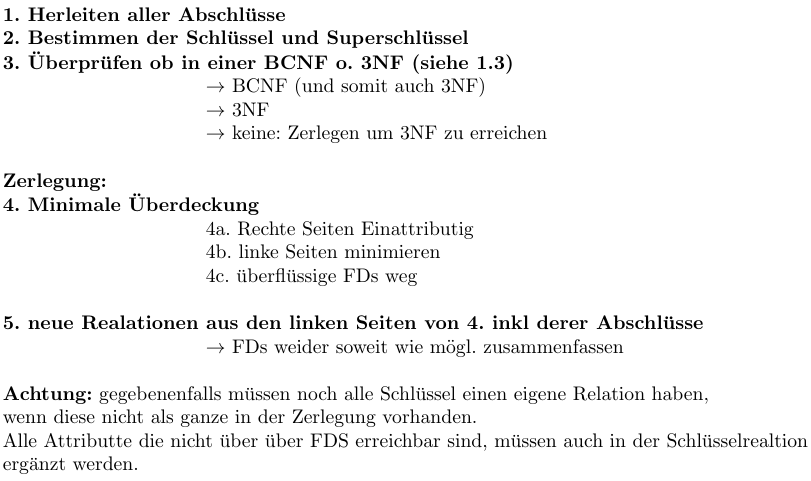


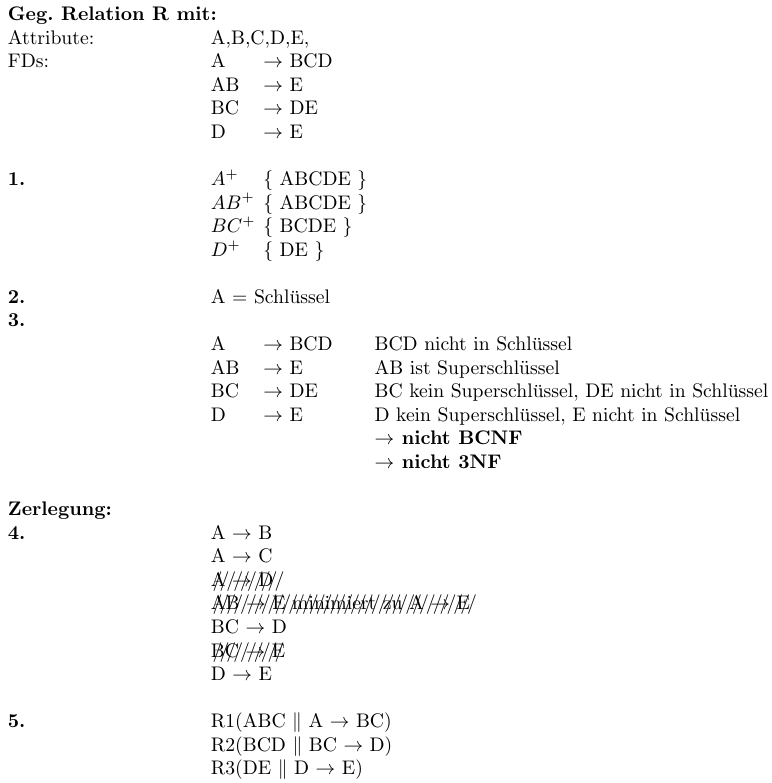
### 3NF

→ Linke Seite (je-)der FD ist Superschlüssel

→ Rechte Seite (je-)der FD ist direkt in einem Superschlüssen enthalten

**Wenn nicht 3NF, dann auch nicht BCNF!**

**Herleitung:  
**

**Bsp:**

### IDFN

→ Gegebene Relationenmenge {(R1, F1), (R2,F2), (R3, F3) IND}

→ Jede einzelne Relation in BCNF

→ Alle INDs Schlüsselbasiert

→ Keine zirkulären INDs über die Relationen

# Recovery

**ACID**:

* **Atomicity**: Als Ganzes oder gar nicht.
* **Consistency**: Nach einer Transaktion wieder in korrektem (syntaktisch), konsistenten Zustand
* **Isolation**: Transaktionen nicht durch andere Transaktionen störbar
* **Durability**: Daten der Transaktion nicht verlieren

## Logical Unit of Recovery, LUR

Transaktionen hinterlassen nach ihrer Ausführung auf einer Datenbank eine bis mehrere LUR’s. Eine LUR entspricht aus der Sicht der Datenbank (der Modellvorstellung) der ACID-Transaktion.

***Für Details, siehe Roli-Recovery-ZUF.***

## Zurücksetzen / UNDO-Information

Bei einem System-Restart oder -Rollback müssen nicht fertige LURs zurückgesetzt werden. Die UNDO-Information kann im selben Log-Record stehen.

## Write Ahead Log (WAL)

= UNDO-Informationen in das Log speichern bevor Update auf DB durchgeführt wird (damit Rollback funktioniert, wenn etwas schief geht)

# Concurrency

***Siehe Siflu ZUF.***