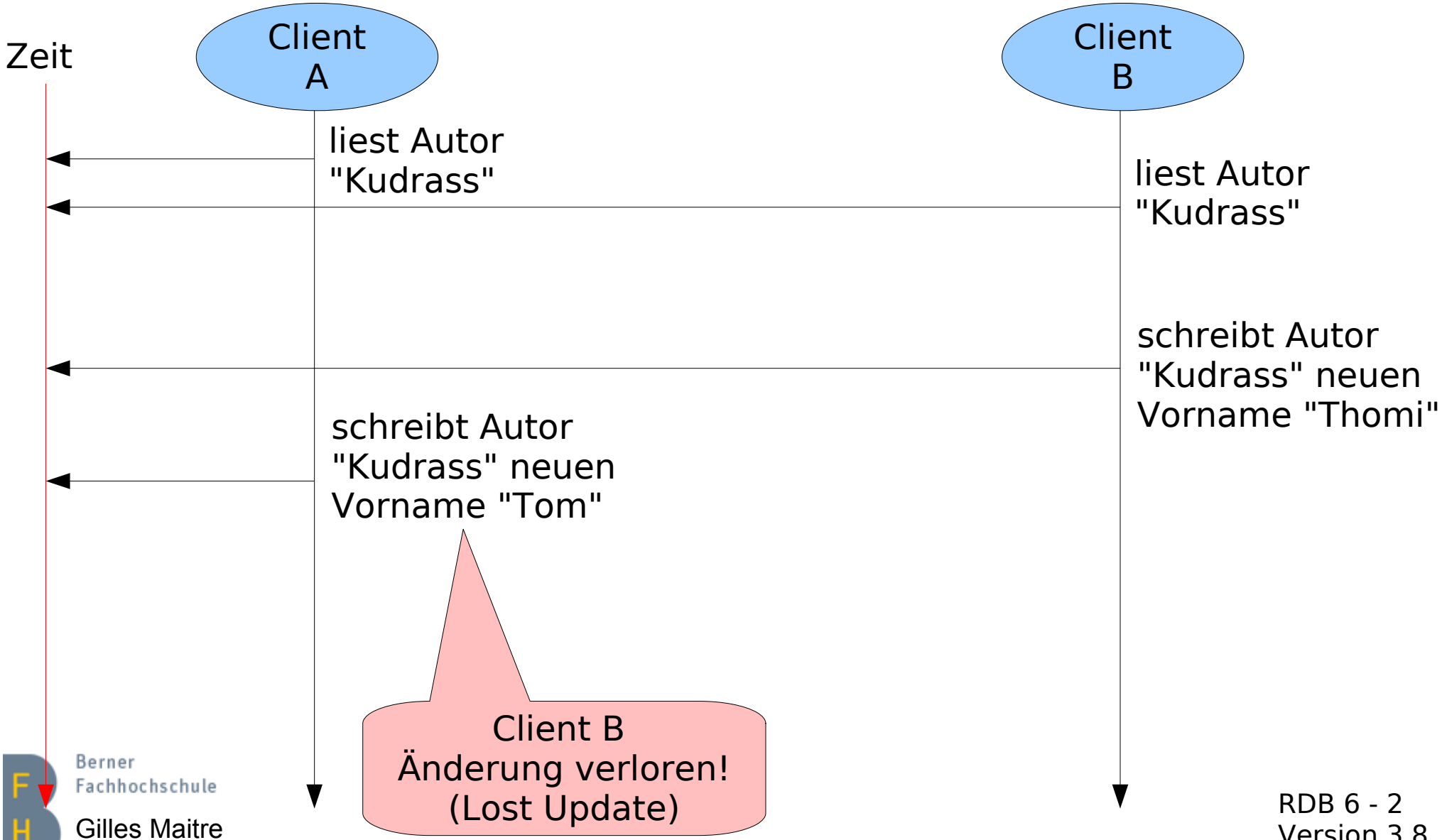
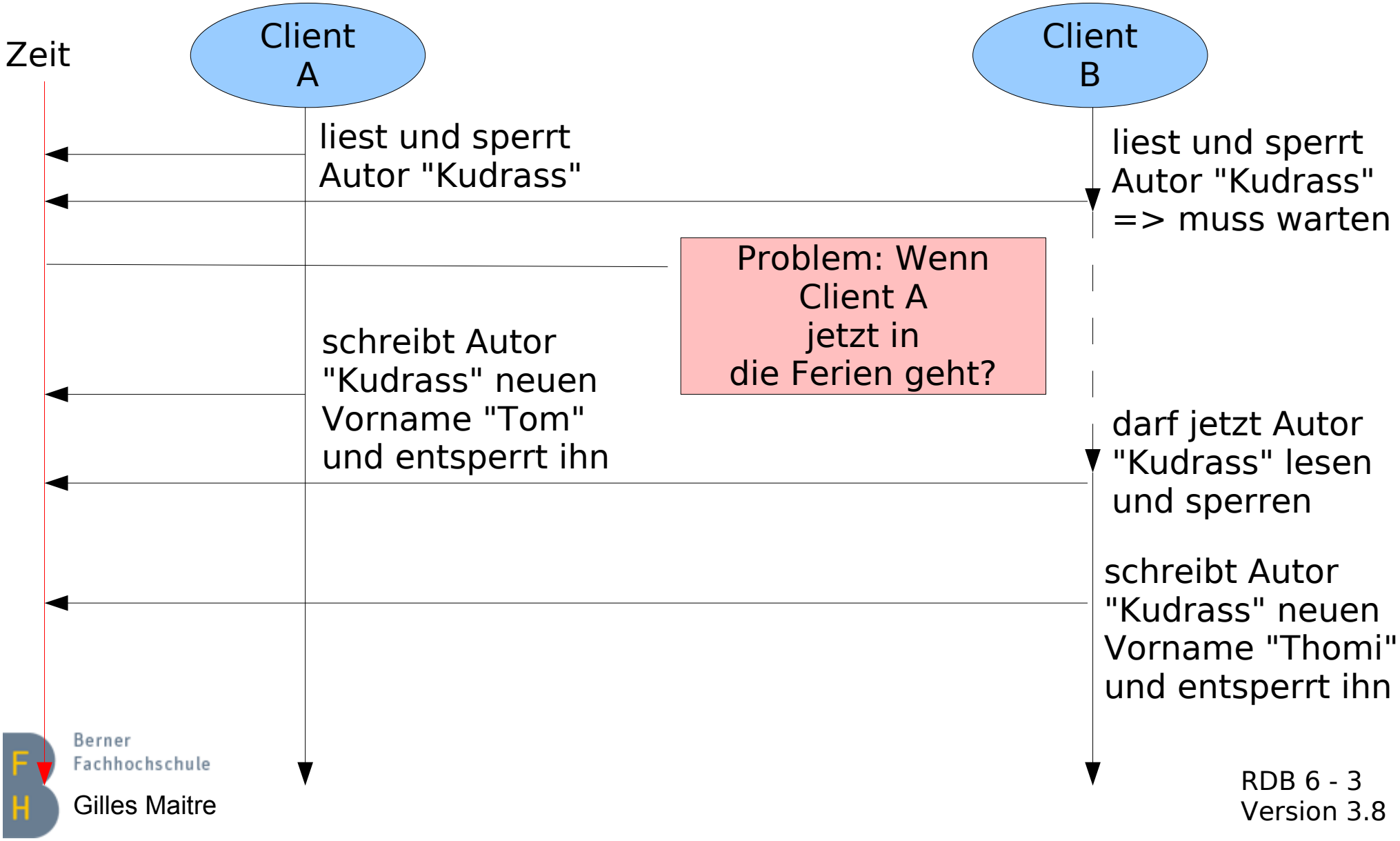
Tag 6
Inhaltsverzeichnis

- Zwei einfache Sperrverfahren in Java
  - Pessimitic, optimistic Locking
- Performance
  - System, DB, Applikation
- DB Zugriffsberechtigungen
- SQL Injection, Gefahr und Gegenmassnahmen
- Metadaten
- Views
- Stored Programs
  - Stored Procedures (SP)
  - Triggers
- Die Wikipedia Architecture
- Übungen

Zwei einfache Sperrverfahren
Problem wenn keine Synchronisation...



Zwei einfache Sperrverfahren
Pessimistic Locking



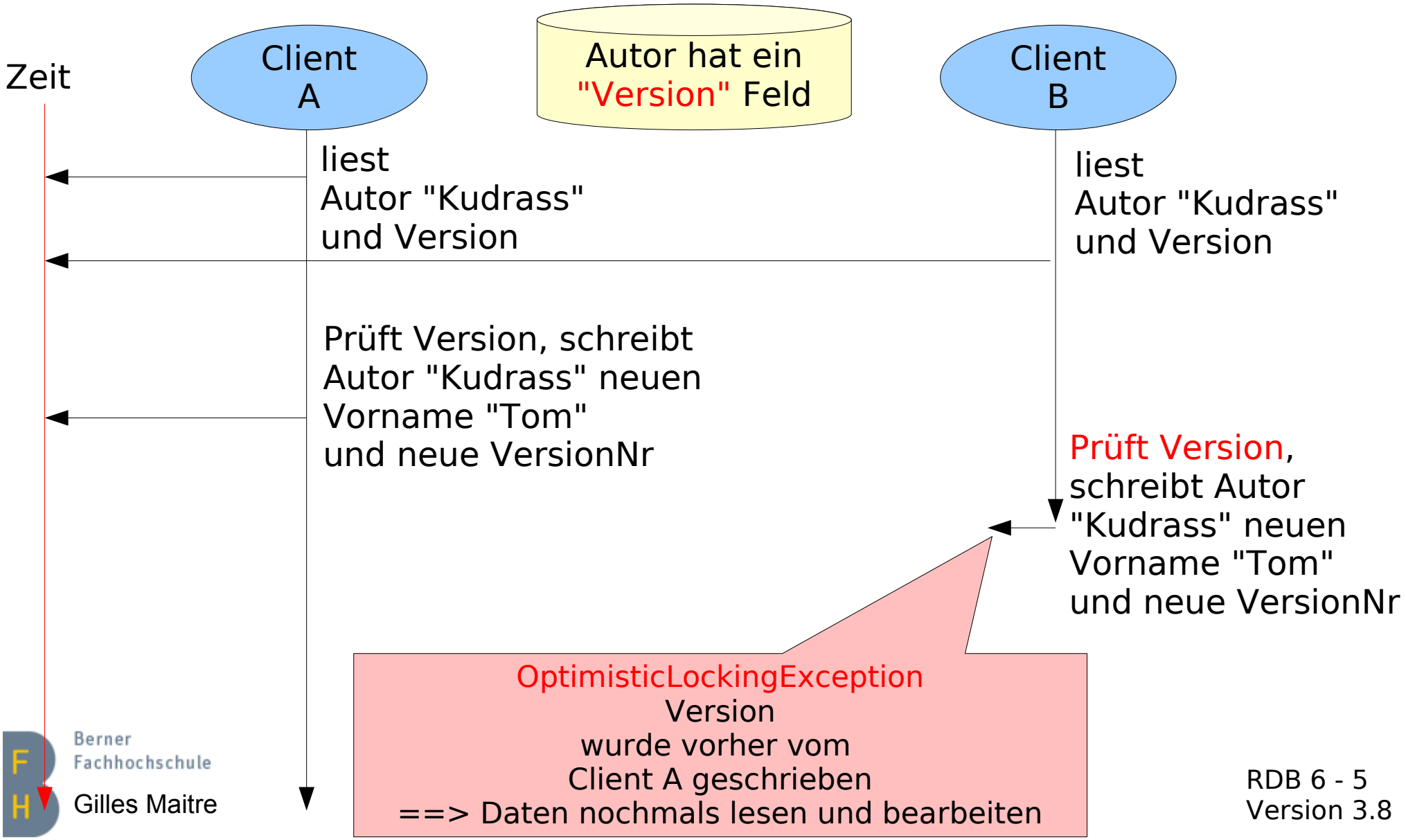
Pessimistic Locking
Einfache Implementation mit JDBC

```
// Annahme:
// In der Variable "autor"
// (vom Typ Autor, mit den Feldern PersNr, Name und Vorname)
// habe ich vorher den Autor "Kudrass" gelesen
connection.setAutoCommit(false);
String selectQuery =
    "SELECT * FROM Autor WHERE PersNr=? FOR UPDATE";
PreparedStatement selectStm = connection.prepareStatement(selectQuery);

// Select for update ausführen
selectStm.setString(1, autor.getPersNr());
ResultSet rs = selectStm.executeQuery();
// Hier muss ich warten, falls jemand diese Zeile schon gelockt hat...
// Dann
// Update vorbereiten
String updateQuery =
    "UPDATE Autor SET Vorname=? WHERE PersNr=?";
PreparedStatement updateStm = connection.prepareStatement(updateQuery);

// Update ausführen
updateStm.setString(1, autor.getVorname());
updateStm.setString(2, autor.getPersNr());
int nbrOfModifiedRecords = updateStm.executeUpdate();
connection.commit();
connection.setAutoCommit(true);
```

Zwei einfache Sperrverfahren
Optimistic Locking



Optimistic Locking
Einfache Implementation mit JDBC

```
// Annahme:
// In der Variable "autor"
// (vom Typ Autor, mit den Feldern PersNr, Version, Name und Vorname)
// habe ich vorher den Autor "Kudrass" gelesen

// Update vorbereiten
String updateQuery =
    "UPDATE Autor SET Vorname=?, Version=? WHERE PersNr=? AND Version=?";
PreparedStatement updateStm = connection.prepareStatement(updateQuery);

// Update ausführen
updateStm.setString(1, autor.getVorname());
updateStm.setInt(2, autor.getVersion() + 1); // Version wird inkrementiert
updateStm.setInt(3, autor.getPersNr());
updateStm.setInt(4, autor.getVersion()); // Meine Ursprungs-Version

// Klappt nur wenn kein Update von "Kudrass" in der Zwischenzeit
int nbrOfModifiedRecords = updateStm.executeUpdate();
if (nbrOfModifiedRecords == 0) {
    throw new OptimisticLockingException();
}
```

Pessimistic und Optimistic Locking
Anwendungsfälle

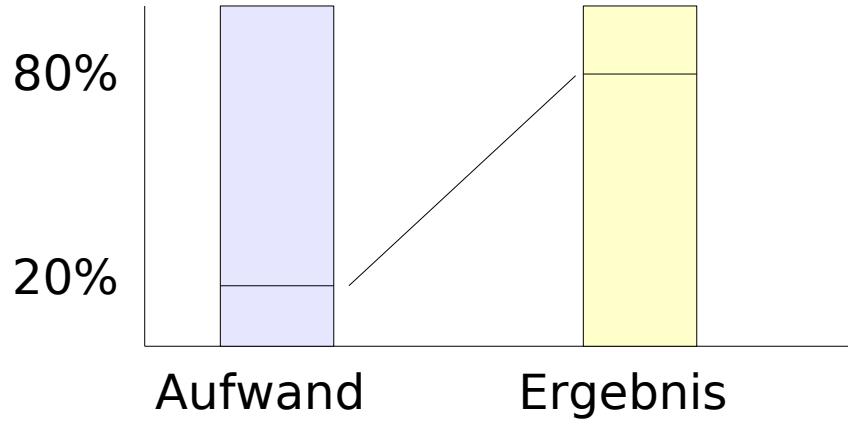
- Wenn... wenig Zeilen eine kurze absehbare Zeit gesperrt werden müssen und kleines Codefragment
  → Pessimistic Locking genügt

- Wenn... diverse Zeilen und/oder Dauer der Sperrung nicht absehbar (z.B. GUI Client/Server Applikation)
  → Optimistic Locking besser geeignet

- Zur Info: Ab Oracle 10g, Optimistic Locking vereinfacht dank "ora_rowscn" Pseudo-Column

  http://robertgfreeman.blogspot.com/2005/06/orarowscn-new-10g-pseudo-column.html

Performance
Das Pareto-Prinzip (80-zu-20-Regel)



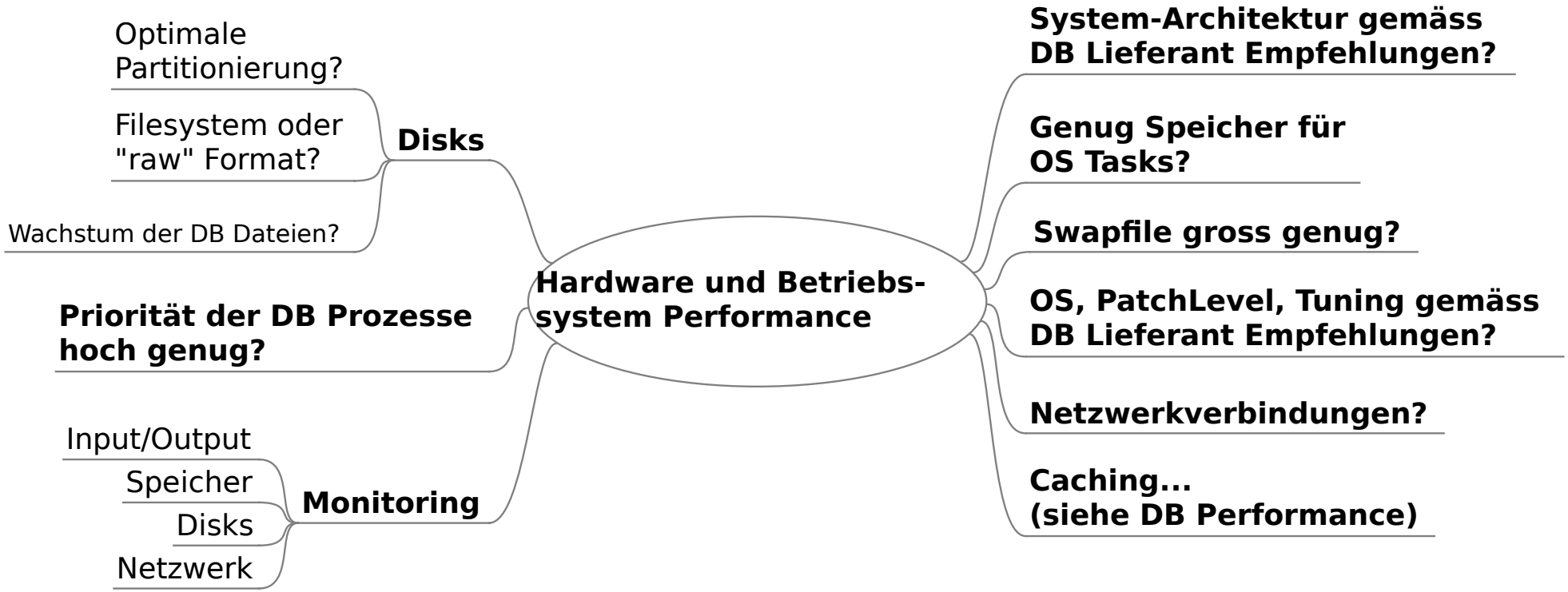
- Im Durchschnitt erzeugen 20% der Kunden einer Firma 80% des Umsatzes

  => Firma muss sich *vor allem* um diese 20% kümmern

- Im Computer-Bereich...

  => Nur optimieren, wo es sich lohnt!

System Performance
Hardware und Betriebssystem



DB Performance
Indexierung

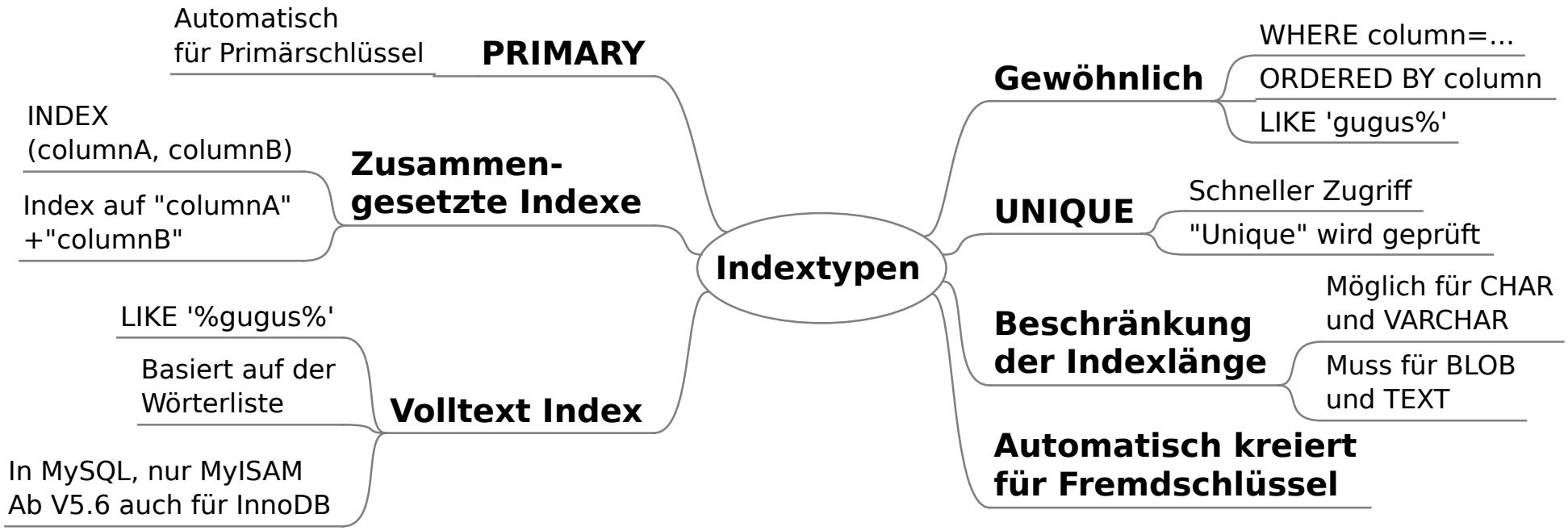
- Extra Daten, um Zugriff auf DB Inhalt zu optimieren
  - Werte in Spalten
  - Joins
  - Sortierung / Aggregation

- Pro: Datenzugriff *schneller*
  Kontra: Datenänderungen *aufwendiger*

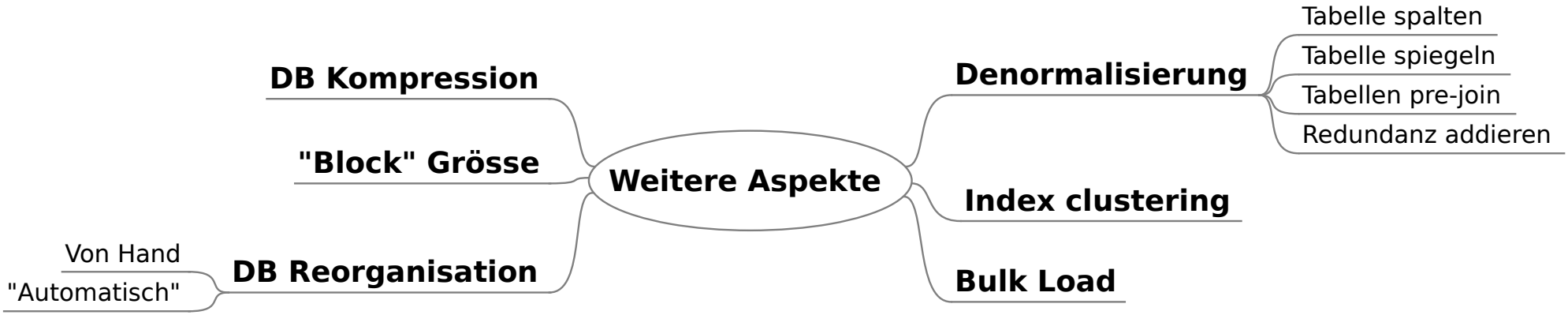
  => Kompromiss *suchen*

- Implementation
  - B-Bäume
  - Hashtabellen (*nicht implementiert in MySQL V5*)

DB Performance (1)
Indextypen



DB Performance (2)
Weitere DB-Aspekte



DB Performance
Indexeinschränkungen

Indexe werden nicht verwendet, wenn

- Ungleichsoperator
  z.B.: `WHERE column != ...`
- Funktionsaufruf im Ausdruck
  z.B.: `WHERE DATE(column)...`
- Beim Join: Ungleicher Typ für Primär- und Fremdschlüssel
- LIKE Operator der Form '%gugus'
- ORDER BY + andere Kriterien
- Index auf Spalte mit vielen ähnlichen Werten, z.B. boolean

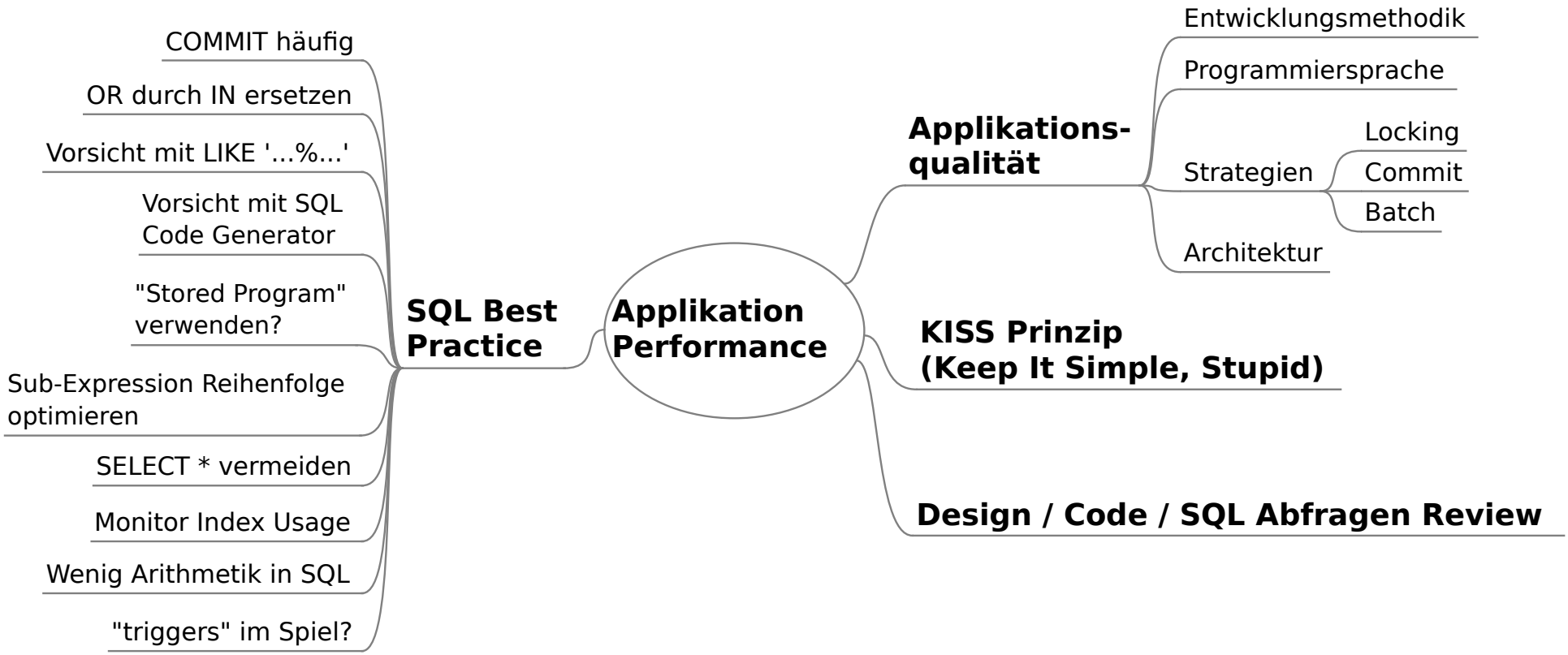
DB Performance
SQL Indexbefehle

```
-- Erstellen
mysql> CREATE INDEX 'index'...
mysql> ALTER TABLE 'table'...

-- Anzeigen
mysql> SHOW CREATE TABLE 'table';
mysql> SHOW INDEX FROM 'table';

-- Löschen
mysql> DROP INDEX 'index';
```

Applikation Performance
Ein paar Aspekte



Applikation Performance
MySQL Monitoring

- Befehl "SHOW ENGINE INNODB STATUS\G" verwenden
  (*siehe MySQL Dokumentation*)
  - => Detailinfos über InnoDB Engine
- Befehl EXPLAIN SELECT... verwenden
  - => Detailinfos über SELECT Statement (Indexverwendung, ...)
- SHOW FULL PROCESSLIST
- Analyse der Logdatei
- Warten auf Tag 7 ;-)

DB Zugriffsberechtigungen
SQL GRANT (vereinfacht)

GRANT <priviledgeList>
ON <database>.<table>
TO <user>@<host> [IDENTIFIED BY 'password']

```
mysql> GRANT Select, Insert ON MeineCDs.* TO gmaitre@'%';

mysql> GRANT All ON *.* TO ''@localhost; -- never do that
```

DB Zugriffsberechtigungen
SQL REVOKE (vereinfacht)

REVOKE <priviledgeList>
ON <database>.<table>
FROM <user>@<host> [IDENTIFIED BY 'password']

```
mysql> REVOKE Select, Insert ON MeineCDs.* FROM gmaitre@'%';

mysql> REVOKE All ON *.* FROM admin@'%';  -- never do that
```

MySQL Zugriffsberechtigungen
SQL SHOW GRANTS

```
mysql> SHOW GRANTS FOR gmaitre@'%'\G

*************************** 1. row ***************************
Grants for gmaitre@%: GRANT USAGE ON *.* TO 'gmaitre'@'%' IDENTIFIED BY PASSWORD
'*A170434C682FB67394D0F469BB2236F6B62F3A85'
*************************** 2. row ***************************
Grants for gmaitre@%: GRANT ALL PRIVILEGES ON `MeineCDs`.* TO 'gmaitre'@'%'
WITH GRANT OPTION
*************************** 3. row ***************************
Grants for gmaitre@%: GRANT ALL PRIVILEGES ON `AutorBuch`.* TO 'gmaitre'@'%'
WITH GRANT OPTION
```

SQL Injection
Gefahr und Gegenmassnahmen

- **Was ist das?**
  Manipulation von SQL-Anweisungen, durch Parameter, die *bösartigen SQL-Code enthalten*

- **Beispiele**
  - https://de.wikipedia.org/wiki/SQL-Injection#Vorgang
  - https://xkcd.com/327/

- **Gegenmassnahmen**
  - Programmatische Überprüfung der Eingaben
  - Systematisches Code-Review
    (die das Problem angehen! => Checklisten verwenden)
  - Verwendung von SQL Prepared-Statements

Quellen:
- Kudraß Seite 146
- https://de.wikipedia.org/wiki/SQL-Injection

Metadaten
Anwendung in DB

- Siehe RDB Tag 5, Slide 21 ;-)
- Metadaten ::= Daten, die die Daten beschreiben
- Beispiel von einem Buch:
  - Daten ::= Buchtext
  - Metadaten ::= Titel, Autor, Verlag, ISBN, Text
- Anwendung für relationale Datenbanken:
  - Daten ::= Tabellen-Schemas und Daten
  - Metadaten ::= Das DataDictionary =>
    Die Daten von MySQL selber
    - Tabellen, ihre Schemas, Zugriffsrechte, ...

MySQL
Metadaten und Grenzen

- Alle Metadaten in DB information_schema gespeichert (nur Lesezugriff)
- **`mysql> USE information_schema`** funktioniert!
- SHOW & DESCRIBE brauchen diese Information...
- Tabellenstruktur nach ANSI/ISO SQL:2003
- Wichtige Tabellen:
  - TABLES, COLUMNS, VIEWS, TRIGGERS, ROUTINES, ...
- Äquivalente Befehle:

```
SHOW databases;
SELECT schema_name FROM schemata;
SELECT DISTINCT table_schema FROM tables;
```

- Siehe auch **"Understanding the MySQL Information Schema Database"**

Views
Was ist das?

- Wirkt wie eine logische Tabelle oder Tabellengruppe
- Basiert (*im Prinzip*) auf einer SELECT Anweisung (oder einer anderen View)
- Spalten und/oder Zeilen können eingeschränkt werden

Views
Was bringt das?

- Eingeschränkter Zugriff auf Spalten und Linien
- Komplexität des Datenzugriffs kann versteckt werden
- Abfragen können wiederverwendet werden
- Tabellen- und/oder Spaltenname können umbenannt werden

Views
Einschränkungen

- Viele Einschränkungen
- Vor allem in INSERT, UPDATE oder DELETE Befehle (und im Bezug zu Joins)
- Im SELECT kein:
  - GROUP BY
  - DISTINCT
  - LIMIT
  - UNION
  - HAVING

Views
Anwendungen

```
mysql> CREATE VIEW V_AutornameBuchTitel AS
       SELECT a.Name, b.Titel
       FROM Autor a, Autor_Buch ab, Buch b
       WHERE a.PersNr = ab.PersonNr AND b.ISBN = ab.ISBN;

mysql> SELECT Name, Titel FROM V_AutornameBuchTitel;

mysql> SHOW CREATE VIEW V_AutornameBuchTitel;

mysql> RENAME VIEW v1 TO v2;

mysql> DROP VIEW v2;        -- nur das View wird gelöscht!
```

Stored Programs
Prinzipien (1)

- Parametrierbar und auf dem DB Server *gespeichert*
  - Erlaubt Vorarbeiten...
  - Vermeidung von SQL Code Redundanz...
  - Bessere Sicherheit...
  - Weniger Netzwerkverkehr...
- Ausgeführt auf dem DB Server
- SP Programmiersprache
  - Kontrollanweisungen ("if", "loop", etc...)
  - Variablen
  - "CURSOR"-Elemente (wirkt wie JDBC ResultSet)
- Nachteil: Diverse DBMS Implementationen

Stored Programs
Prinzipien (2)

- Typen
  - Stored Procedures (SP)
  - Triggers

- MySQL: Untermenge von ANSI SQL:2003 SQL/PSM
  - Ähnlich zu Oracle's PL/SQL und SQL Server's Transact-SQL
  - Starke Einschränkungen in der Anwendung

Stored Programs
Triggers, was ist das?

- Ereignis-orientiertes Programmieren in DB
- Automatischer Aufruf eines "Stored Program's" bei einer Veränderung von Daten
- Anwendungsbeispiele
  - DB Integrität garantieren
  - Journal
- Gefahren
  - Nicht voraussehbare *Kettenreaktion* <=> Einfluss auf Performance
  - DB Zustandswechsel schwer zu *debuggen*

Die Wikipedia Architecture
Datenbankperspektive

- Aktuell:
  https://meta.wikimedia.org/wiki/Wikimedia_servers
- Weniger:
  https://meta.wikimedia.org/wiki/File:Wikimedia-servers-2010-12-28.svg

Übungen

1) In der GMCD DB, Tabelle "CD", schauen Sie ob Indexes definiert sind und analysieren Sie das Verhalten folgender Abfragen:
   explain select * from CD where Beschreibung = "Dresden"
   explain select * from CD where Datum = "1968-01-13"
   explain select * from CD where Datum != "1968-01-13"
   explain select * from CD where Datum like "19%"

2) Mit einem SQL-Befehl basierend auf die Metadaten-DB "information_schema" suchen Sie alle Datenbanken, die auf Ihrem Computer definiert sind und listen Sie die Tabellen der Datenbank "GMCD".