NewElements

IT-Schulungen.com

SQL - Kurs

MySQL für Entwickler

Dr.sc.nat. Michael J.M. Wagner, New Elements*

Revision 327-2022

*michael@wagnertech.de

Inhaltsverzeichnis

IT-Schulungen.com
Portfolio

IT-Schulungen.com ist eines der führenden, herstellerunabhängigen Seminarportale von Schulungen rund um die Informationstechnologie (IT) und das IT-Management. Seit über 15 Jahren ist IT-Schulungen.com eine anerkannte Anlaufstelle für viele Unternehmen und Behörden, wenn es um die Durchführung von DACH-weiten Schulungen geht.

- Applikationsserver / Middleware
- Business Intelligence
- Business-Skills und Führung
- Cloud
- CRM
- Datenbanken
- eBusiness
- ERP-Systeme
- IT Management
- IT-Recht / Lizenzierung
- ITIL
- Mobile
- Multimedia
- Office
- Open Source
- Portale
- SAP®
- Security
- Serversysteme
- Softwareentwicklung
- Systemmanagement

Zertifizierungen & Partnerschaften

1 Grundlagen

1.1 Einführung Datenbanken

Entwicklung der Datenbanken

Historische Entwicklung:

- Datenhaltung in Dateien
- Dateien mit wahlfreiem Zugriff
- Datenbanken

.[Fuchs: S. 6f.]

Datenbankmodelle

Datenbanken entwickelten sich in der EDV-Geschichte nach und nach mit dem Ansteigen paralleler Nutzung der Systeme. In den 50er-Jahren waren größete Datenmengen meist auf *Magnetbändern* abgelegt. Ein sequentieller Zugriff funktionierte ganz gut, der wahlfreie Zugriff war schwierig und zeitintensiv.

Einen Fortschritt brachte die Einführung von *Festplattensystemen*. Sie erlaubten den wahlfreien Zugriff auf einzelne Datensätze (60er Jahre).

Durch das Aufkommen von Terminals und damit der parallelen Bearbeitung von Daten, kamen in den 70er Jahren Systeme, die erstmals Merkmale heutiger Datenbanksysteme aufwiesen. Die Struktur der Daten war wie bei den Festplattensystemen hierarchisch. Es war die Zeit der *hierarchischen Datenbanksysteme*.

Durch den Forschritt in der IT-Technik konnte in den 80er Jahren mehr Logik in das Datenbank-Managementsystem (DBMS) gelegt werden, die Datenstrukturen konnten einfacher gehalten werden. Es entwickelten sich die *relationalen Datenbanksysteme*.

Mit dem Hype der Objektorientierung der 90er Jahre wurden *objektorientierte Datenbanken* entwickelt. Diese Entwicklungen scheiterten aber daran, dass die neu entwickelten Datenbanken geben die großen relationlen „Platzhirschen" keine nennenswerten Marktanteile erringen konnten.

Die Anpassung zwischen der objektorientierten Welt der Programmiersprachen und der relationalen Welt der Datenbanken überbrücken heute sogenannte objekt-relationale Mapper (OR-Mapper).

Die relationalen Datenbanken leisten heute den Hauptteil der Datenspeicherung. Objektorientierte Datenbanken und NoSQL-Datenbanken führen ein Nischendasein für spezielle Anwendungen. .[Fuchs: S. 8ff.]

Aufbau und Organisation einer Datenbank

Für die Datenbankmodellierung gilt das 3-Ebenen-Modell (s. Abb. 1[1]).

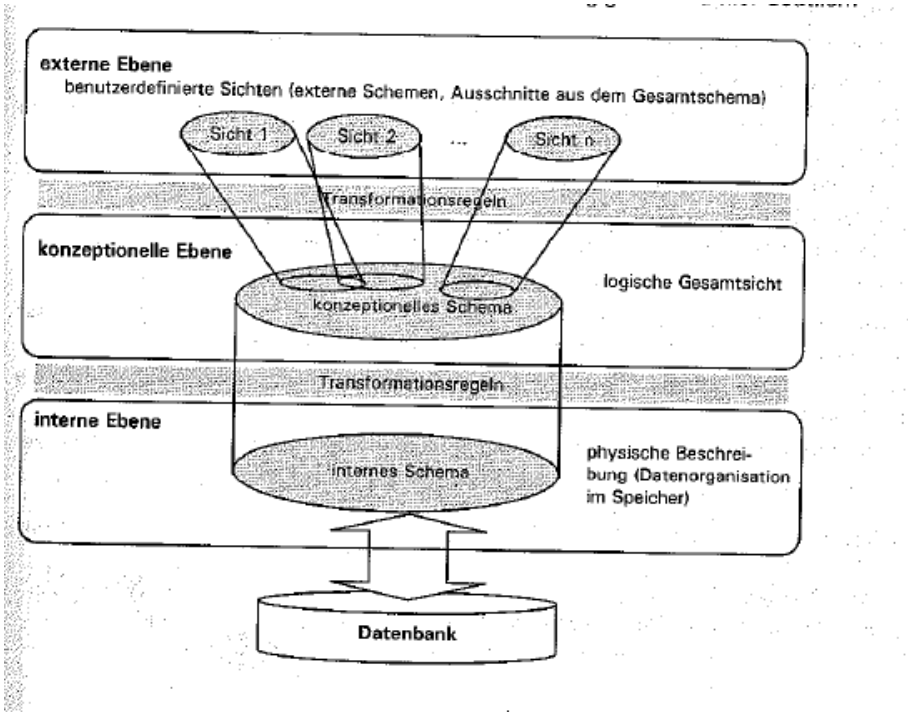

Abbildung 1: 3-Ebenen-Modell

- Externe Ebene: Der Bereich der Anwendungsprogrammierung. Ein Anwendung für einen bestimmten Zweck hat eine bestimmte Teilsicht auf die Datenwelt.
- Konzeptinonelle Ebene: Logische Gesamtsicht auf die Unternehmensdaten („Unternehmensmodell"). Üblicherweise als ER-Modell
- Interne Ebene: Realisierung des Unternehmensmodells in einer konkreten Datenbank

.[Fuchs: S. 16]

Datenbankmanagementsystem (DBMS): [Fuchs: 17f.]

Weitere Komponenten des DBMS: [Fuchs: S. 19]

Physische Datenbankarchitektur

Datenbanken werden in verschiedenen Architekturen betrieben[2]:

- Zentralisierte DBMS: Anwendungen, DBMS und Datenspeicher liegen auf einem Serversystem (Abb. 2).

[1] Herdt (2004): S. 13.
[2] Herdt (2004): S. 16ff.

- Verteilte DBMS: Anwendungen, DBMS und Datenspeicher liegen auf verteilten Serversystemen (Abb. 3).
- Client-Server DBMS: Die Anwendung ist zwischen einem benutzerseitigen Client und dem Server verteilt (Abb. 4).
- Parallele DBMS: Das Serversystem ist parallelisiert. Die Kommunikation zwischen den Systemen kann dabei auf verschiedenen Ebenen erfolgen. Ein Beispiel ist die *shared disk*-Architektur (Abb. 5).

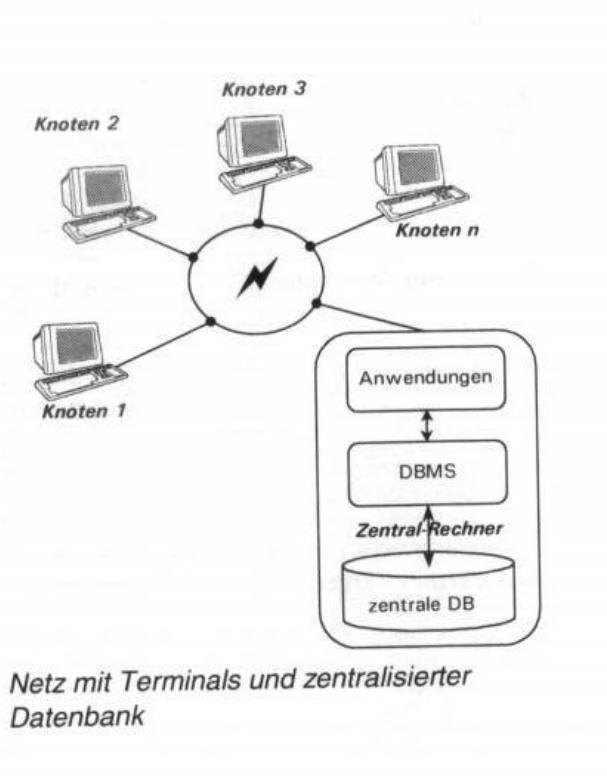

Abbildung 2: Zentralisierte DBMS

.[Fuchs: S. 20ff.]

1.2 Der Datenbankentwurf

Für die Datenmodellierung hat sich das Entity-Relationship-Modell (ER) durchgesetzt. Es ist durch folgende Eingenschaften gekennzeichnet:

- DB-unabhängig
- Fachliche Schlüssel
- Beziehungen
- Kardinalitäten
- Generalisierung/Spezialisierung

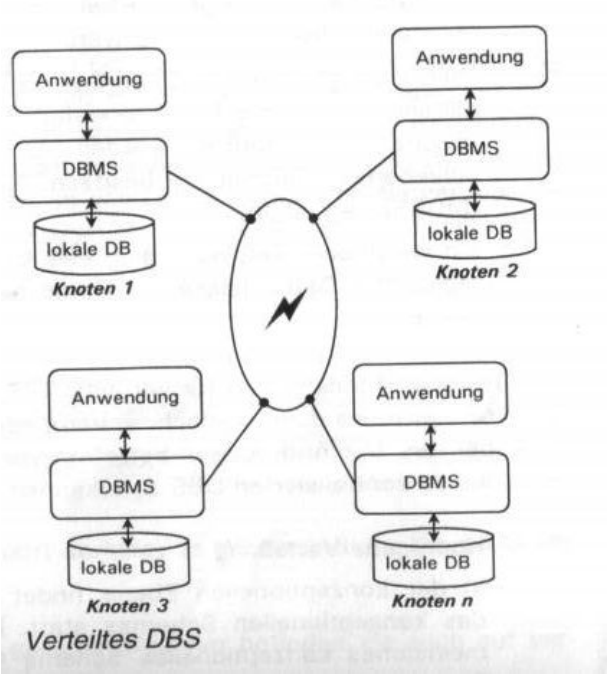

Abbildung 3: Verteilte DBMS

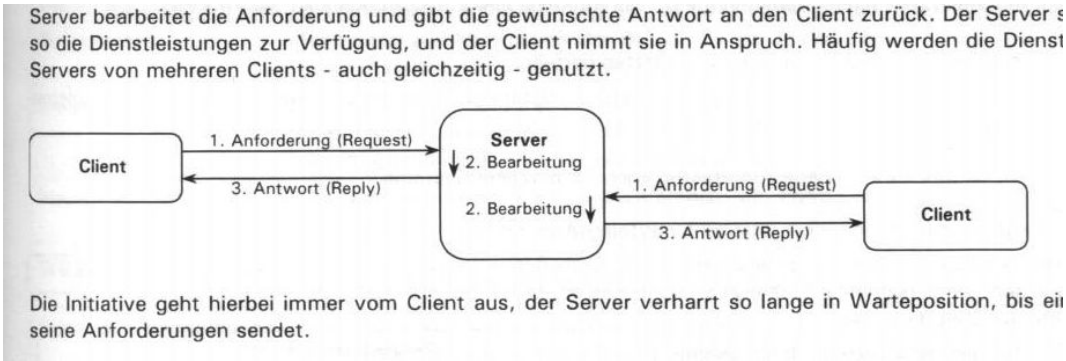

Abbildung 4: Client-Server DBMS

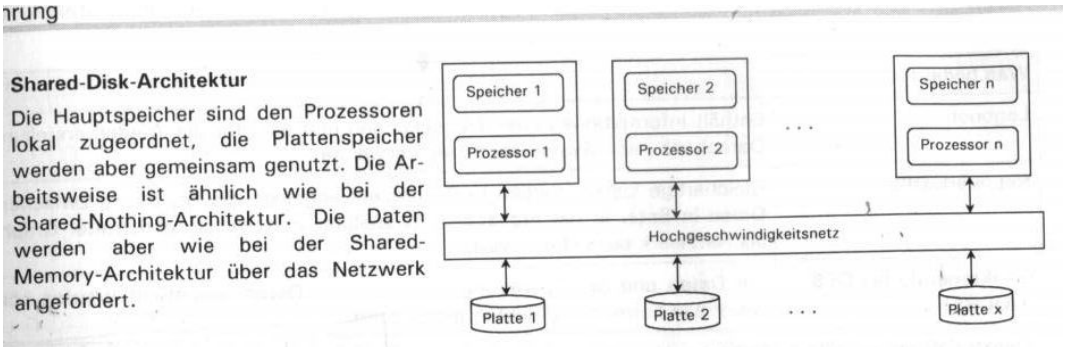

Abbildung 5: Parallele DBMS

Im ER-Modell kann die Beziehung selbst Träger von Information sein (Abb. 6).

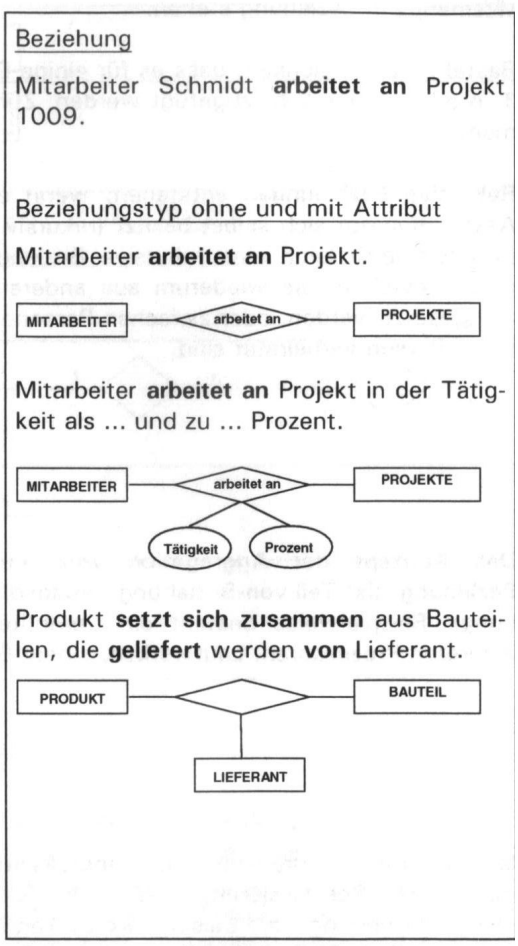

Abbildung 6: Beziehung und Beziehungstyp

Eine besondere Beziehung ist die `is-a`-Beziehung. Je nach Betrachtungsstandpunkt drückt sie eine Generalisierung oder eine Spezialisierung aus (Abb. 7).

.[Fuchs: Kap. 2.4]

Aufgabe:

Entwerfen Sie ein ER-Modell für eine Bücherei:

- Die Basisentitäten sind *Medien*, *Exemplare* und *Benutzer*.
- *Medien* können *Bücher, CDs, DVDs* sein.
- *Exemplare* sind von einem bestimmten *Medium*.
- *Benutzer* können *Exemplare* ausleihen.
- Bei der Rückgabe sollen Ausleihen nicht gelöscht, sondern als „rückgegeben" gekennzeichnet werden.

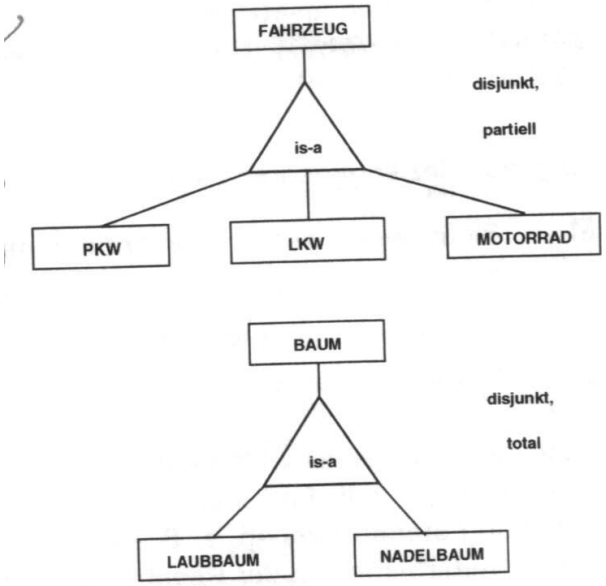

Abbildung 7: Is-a Beziehungen

Die Entitäten/Relationen haben mindestens folgende Attribute:

`Buch`	`ISBN`
`Benutzer`	`vorname`
	`nachname`
`Exemplar`	`signatur`
`Medium`	`erscheinungsjahr`

1.3 Das relationale Modell

Das Datenbankmodell ist grundsätzlich unabhängig vom Typus der realisierenden Datenbank. Sollen die Daten in einer relationalen Datenbank gespeichert werden, so muss das ER-Modell in relationales Modell überführt werden. Dabei gelten folgende Regeln:

- Entitäten werden durch Normalisierung in eine oder mehrere Tabellen überführt (Relation = Tabelle).
- Einfache 1:n-Beziehungen werden durch Fremdschlüsselbeziehungen abgebildet.
- Kompliziertere Beziehungen werden durch Beziehungstabellen realisiert.
- Für die Realisierung einer `is-a`-Beziehung gibt es drei Möglichkeiten:
  - eine Tabelle für die Hierarchie
  - eine Tabelle für jedes Blatt
  - eine Tabelle für jeden Knoten

Normalisierung

Die Normailsierung läuft in drei Schritten ab:

- 1. Normalform
  - Eine Tabelle hat nur Zeilen und Spalten.
  - Eine Zelle ist nicht weiter strukturiert.
- 2. Normalform
  - Jede Zelle ist nur vom Primärschlüssel abhängig.
- 3. Normalform
  - Auflösung transitiver Abhängigkeiten.

Ein weiteres Augenmerk muss den Attributen geschenkt werden, ob sie einen NULL-Wert annehmen dürfen. Aspekte hierbei sind:

- Performance

  Der NULL-Wert ist ein weiterer Freiheitsgrad, der insbesondere bei Datenabfragen Zeit kostet.

- Gibt es einen "natürlichen"NULL-Wert?

  Wenn es einen "natürlichen"NULL-Wert (0, Leerstring) gibt, muss sich ein NULL semantisch unterscheiden.

- Wie sieht das Zusammenspiel mit der darüber liegenden Programmiersprache aus. Kann diese ein NULL abbilden?

.[Fuchs: Kap. 3]

Datentypen

MySQL kennt folgende Datentypen. Je nach Anwendungsfall sollte der je geeignetste gewählt werden.[3]

- Zahlen

  Für Ganzzahlwerte stehen in MySQL folgende Typen zur Verfügung:
  `TINYINT`, `SMALLINT`, `MEDIUMINT`, `INT`, dazu die *unsigned* Varianten.

  Benutzen Sie den Typ, der gerade sicher ausreicht.

- Geldbeträge werden in `DECIMAL` abgelegt

[3] https://www.developwebsites.net/choose-optimal-mysql-data-type/

- Zeichenketten

  Hier gibt es `CHAR`, `VARCHAR`, `TEXT`.

  `TEXT` ist dasselbe wie `VARCHAR(65535)`, sollte also nur bei entsprechend großen Zeichenketten verwendet werden. `CHAR` benötigt bei der Ablage im Speicher immer dieselbe Größe, ist aber schneller bei Suchvorgängen.

- Datum
  - UNIX-Zeitstempel als `INT`: Kleiner Platzbedarf, nicht direkt lesbar
  - `DATETIME`: Direkt lesbar, Funktionen verfügbar

Aufgabe:

Entwicklen Sie aus den ER-Modell ein relationales Modell. Sehen Sie numerische (zusätzliche) Schlüssel vor.

2 Structured Query Language (SQL)

Geschichte und Aufbau: [Fuchs: Kap. 4.1]

2.1 Datenbanken

Werden mit dem Begriff „Datenbanken" oft DBMS (z.B. MySQL, Oracle) gemeint, ist jetzt „Datenbank" im engeren Sinne, die Organisation innerhalb unseres DBMS, das unsere Tabellen enthält, gemeint.

- Datenbank erstellen [Fuchs: Kap. 4.2]
- Datenbank anzeigen und auswählen [Fuchs: Kap. 4.3]
- Datenbank löschen [Fuchs: Kap. 4.4]

Aufgabe:

- Legen Sie eine Datenbank `Bucherei` an.
- Zeigen Sie alle bestehenden Datenbanken an.
- Wählen Sie die Datenbank `Bucherei` aus.

2.2 Tabellen

- Tabellen erstellen [Fuchs: Kap. 5.1]
- Datentypen festlegen [Fuchs: Kap. 5.2]
- Constraints in Tabellen verwenden [Fuchs: Kap. 5.3]
- Domänen (vordefinierte Constraints) werden von MariaDB/MySQL nicht unterstützt.
- Vorhandene Tabellen anzeigen, ändern und löschen [Fuchs: Kap. 5.5]

Aufgabe:

Legen Sie in der Datenbank `Bucherei` die Tabellen an.

2.3 Daten einfügen, aktualisieren, löschen

- Daten einfügen [Fuchs: Kap. 6.1]
- Daten aktualisieren [Fuchs: Kap. 6.2]
- Bedingtes Einfügen / Aktualisieren von Daten

  Mit dem Architekturprinzip der „losen Kopplung" ist es nötig „idempotente" Aufrufe zu implementieren. Ein doppeltes Anlegen eines Datensatzes soll also nicht zu einer Fehlermeldung, sondern (falls nötig) zu einer Aktualisierung führen. [Fuchs: Kap. 6.3]
- Daten löschen [Fuchs: Kap. 6.4]

Aufgabe:

- Fügen Sie in die Datenbank `Bucherei` Datensätze ein.
- Ändern Sie das Erscheinungsjahr eines Mediums.
- Löschen Sie einen Nutzer

2.4 Abfragen über eine Tabelle

- Das SELECT-Statement [Fuchs: Kap. 7.1]
- Bedingungen [Fuchs: Kap. 7.2]
- Abfrageergebnisse gruppieren → Kap. 4.3
- Abfrageergebnisse sortieren [Fuchs: Kap. 7.4]

2.5 Funktionen in Abfragen

- Aggregatfunktionen [Fuchs: S. 131]
- Mathematische Funktionen [Fuchs: S. 134f.]
- Funktionen für Zeichenketten [Fuchs: S. 136]

  Weitere Funktionen für Zeichenketten:

  - Verbinden von Zeichenketten: `concat(str1, str2)`
  - Defaultwert für `NULL`: `coalesce(spalte,defaultwert)`
  - Bedingungen: `if(bedingung, truewert, falsewert)`

  Anmerkung: Die Werte können auch Bezüge auf andere Spalten sein.

3 Beispieldatenbank

In Ihrem System befindet sich die Datenbank `FlughafenDB`. Sie enthält folgende Tabellen (Abb. 8):

`flughafen`	Flughafen-Daten
`flughafen_geo`	Geo-Daten eines Flughafen
`passagier`	Basisdaten eines Passagiers
`passagierdetails`	weiter Passagierdaten
`flugplan`	ein Flug von/nach mit Uhrzeit und den Flugtagen (Mo-So)
`flug`	ein konkreter Flug an einem bestimmten Tag
`flug_log`	Änderungen an Flügen
`fluglinie`	Fluglinie
`flugzeug_typ`	Flugzeugtyp
`flugzeug`	ein konkretes Flugzeug
`wetterdaten`	Wetterdaten zu einem bestimmten Zeitpunkt an einem Ort
`mitarbeiter`	Flughafenmitarbeiter
`buchung`	verbindet einen Passagier mit einem Flug

Die Datenbank liegt zwei Varianten vor, einer kleineren und einer großen mit 50 Mio. Datensätzen.

Aufgabe:

Starten Sie die *MySQL workbench* und sehen Sie sich in der Datenbank um.

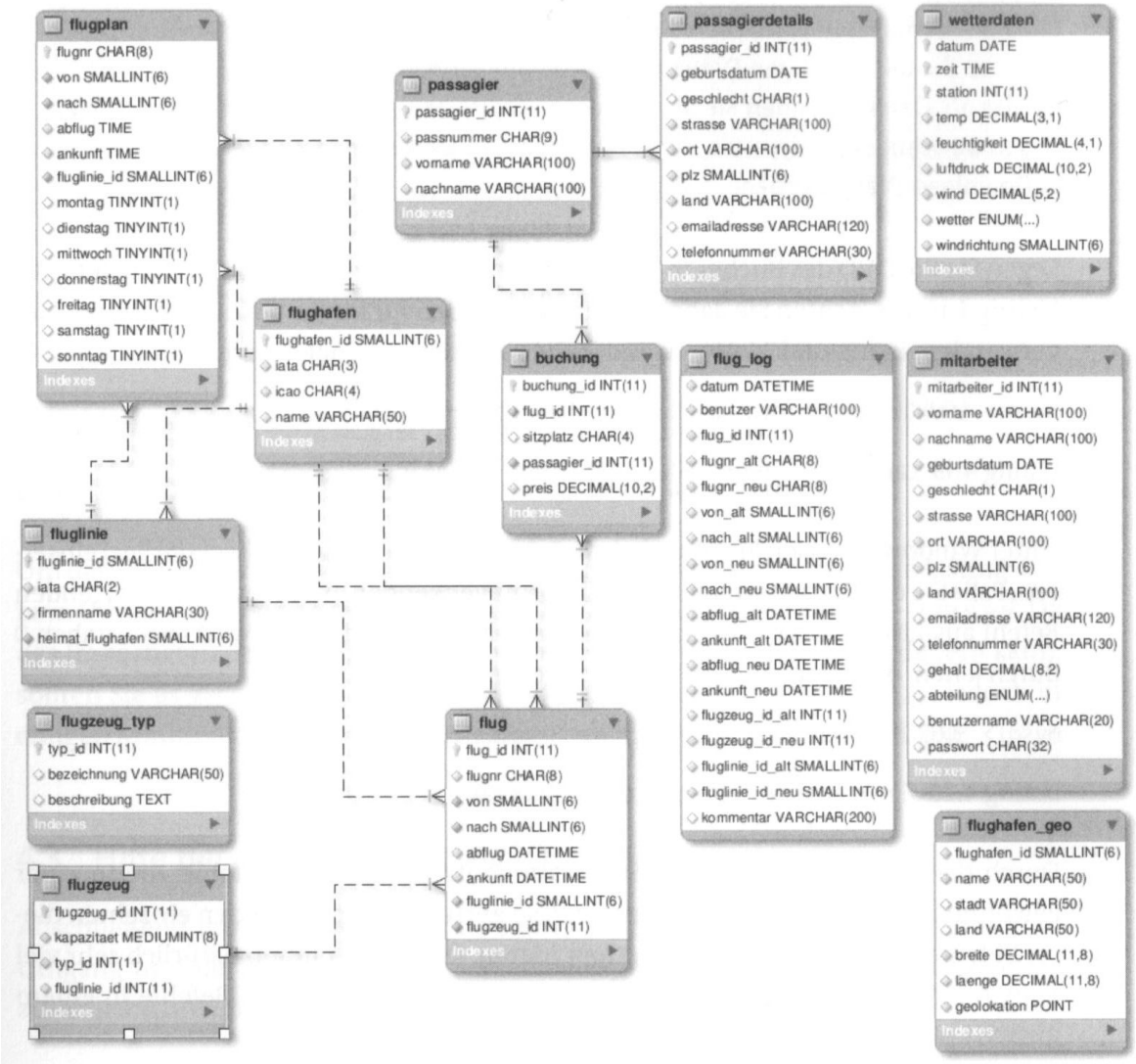

Abbildung 8: Das Flughafenmodell

4 Fortgeschrittenes SQL

4.1 Schlüssel und Indizes

Durch einen Schlüssel wird ein Datensatz in einer Tabelle eindeutig identifiziert. Man unterscheidet dabei

- technische Schlüssel

  Ein technischer Schlüssel ist üblicherweise in Ganzzahlwert, der den Datensatz eindeutig identifiziert, daher oft *id* genannt. Technische Schlüssel sind gerade beim Einsatz von OR[4]-Mappern unerlässlich.

- fachlische Schlüssel

  Fachliche Schlüssel identifizieren einen Datensatz in fachlicher Weise eindeutig. Sie können sich über mehrere Felder erstrecken.

Eine weitere Unterscheidung von Schlüsseln ist die Einteilung in einen *Primärschlüssel* und weiteren *Sekundärschlüsseln*. Der Primärschlüssel bestimmt oft die Reihenfolge der Datenablage in der Datenbank. Im Allgemeinen wird bei der Existenz eines technischen Schlüssels dieser auch als Primärschlüssel verwendet.

Primär- und Sekundärschlüssel stellen auch stets *Indizes* dar. Ein Index ist ein Verzeichnis, das hilft, einen bestimmten Datensatz schnell zu finden. Daher sind Indizes performance-kritisch.

Des weiteren können auch Indizes für Nicht-Schlüssel-Felder eingerichtet werden. In diesem Fall kann der Zugriff über den Index zu mehr als einem Datensatz führen.

Des weiteren existieren in einer Datenbank *Fremdschlüssel*. Diese verweisen von einer Tabelle zu einer anderen und führen zu einer Sicherung der Integrität. Fremdschlüssel können aber auch nachteilig sein Fremdschlüssel führen zu Performance-Einbußen, weil bei Datenänderungen

- ständig Prüfungen durchgeführt werden.
- zur Integritätssicherung die referenzierten Tabellen mit gelockt werden.

Es ist zu überlegen, auf Fremdschlüsselsicherung bei Referenzen auf sehr statische Tabellen zu verzichten. Im Flughafen-Datenmodell könnte also auf eine Fremdschlüsselsicherung auf die Tabelle `flughafen` verzichtet werden. In diesem Fall ist die Sicherung der Fremdschlüsselrelation durch die darüber liegende Software zu leisten.

Indizes sind Verzeichnisse für Nicht-Schlüssel-Felder und beschleunigen Suche und Sortierung. Indizes müssen bei Datenänerungen aktualisert werden.

.[Fuchs: Kap. 8]

Aufgabe:

- Untersuchen Sie in der Beispieldatenbank die existierenden Schlüssel und Indizes.

[4] objektrelational

4.2 Abfragen über mehrere Tabellen

Tabellen verknüpfen

Mit der SELECT-Abfrage können Daten von mehreren Tabellen geholt werden. Die Ausgestaltung der Abfrage bewirkt, in welcher Weise die Datensätze zueinander in Verbindung stehen. Die Verbindung von Tabellen wird als *join* bezeichnet. Folgende Joins sind von Bedeutung:

- *cross join*: Jedes Element der einen Tabelle wird mit jedem Element der anderen Tabelle verbunden.
- *inner join*: Die Tabellen werden so verbunden, dass eines oder mehrere Felder beider Tabellen denselben Wert haben.
- *left outer join*: Von der ersten Tabelle werden alle Einträge verwendet. Einträge der zweiten Tabelle werden dann hinzugefügt, wenn wie beim *inner join* Felder mit denselben Werten vorhanden sind.

Cross und *inner join* unterscheiden sich durch unterschiedliche Bedingungen in der WHERE-Clause:

```
> SELECT * FROM tab1, tab2 WHERE ...;
```

Für einen *inner join* lautet die Bedingung

```
WHERE tab1.feld1 = tab2.feldx AND ....
```

.[Fuchs: Kap. 10.1, 10.2]

Aufgabe (Bücherei-DB):

- Erstellen Sie eine einfache Verknüpfung der Tabellen `Nutzer` und `Verleih`, um für jeden Leser die ausgeliehenen Bücher zu ermitteln. Lassen Sie sich in der Abfrage die Vor- und Familiennamen der Leser und die `signatur` anzeigen.
- Verwenden Sie die gleiche Abfrage, und ergänzen Sie diese um eine Sortierung nach Familiennamen und Vornamen der Leser.
- Erstellen Sie einen *cross join* der beiden Tabellen.

Tabellen verbinden mit JOIN

Ein *inner join* lässt sich auch mit dem Schlüsselwort JOIN erstellen. Die Syntaxlautet dabei:

```
> SELECT <datenfelder> FROM tab1 [INNER] JOIN tab2 ON tab1.feld1 = tab2.feldx AND ...
-> [WHERE ...];
```

Die Syntax für den *left outer join* ist sehr ähnlich. Er wird im Statemant als LEFT [OUTER] JOIN bezeichnet.

Eine Tabelle lässt sich auch mit sich selbst verknüpfen. Dies kommt bei teschnischer Software vor: Container, verkettete Listen.

Mit dem Schlüsselwort UNION lassen sich die Ergebnisse zweier SELECT-Abfragen vereinigen. Dies geht nur, wenn diese Abfragen dieselbe Anzahl von Felder zurückliefern.

.[Fuchs: Kap. 10.3]

Aufgaben:

Lösen Sie diese Aufgabe unter Verwendung des Schlüsselworts `JOIN`.

- Verwenden Sie die Datenbank `Bibliothek`.
- Erweitern Sie das SQL-Statement der letzen Aufgabe. Zeigen Sie zusätzlich die ISBN der Bücher an, wenn es sich um ein solches handelt.

Weitere Aufgaben in der Datenbank `Uebungen`:

- Erstellen Sie eine Abfrage über die Tabellen `t_lager` und `t_artikel`, um alle im Lager vorrätigen Artikel aufzullisten. Zeigen Sie die Attribute Artikelnummer, Artikelnamen, Liefernt und Stückzahl an.
- Wandeln Sie die Abfrage so um, dass Sie alle Artikel erhalten, egal ob sie am Lager sind oder nicht. Falls vorhanden, sollen neben der Artikelnummer, dem Artikelnamen und der Lieferantennummer auch die Stückzahl und der Preis angezeigt werden.
- Ergänzen Sie die Abfrage: Für Artikel, die nicht im Lager vorrätig sind, soll die Stückzahl „0" und „kein Preis" als Preis ausgegeben werden.

Abfrageergebnisse vereinigen

Abfrageergebnisse mit gleich Spalten können zu einem Gesamtergebnis vereinigt werden: [Fuchs: Kap. 10.4]

Aufgabe:

Erstellen Sie aus der Flughafen-DB eine Gesamtliste (Name, Vorname, Geburtsdatum) aller Personen, die mit dem Flughafen zu tun haben (Passagiere und Mitarbeiter).

Unterabfragen

- Unterabfragen zur Ermittlung eines Schlüssles

  Diese Abfragen können leicht in einen `JOIN` umgewandelt werden.

- Unterabfragen mit `EXISTS/NOT EXISTS`

.[Fuchs: Kap. 10.6]

Aufgabe (Bücherei-DB):

- Suchen Sie alle Ausleihen für ein bestimmtes `Exemplar`, von dem Sie aber nur die `signatur` kennen. Formulieren Sie die Abfrage mit `EXISTS` und als `JOIN`.

- Suchen Sie alle *Exemplare*, die noch nie ausgeliehen wurden.

4.3 Abfrageergebnisse gruppieren

Beim Gruppieren wird eine Aggregatfunktion auf eine Teiltreffermenge angewendet, die eine gemeinsame Eigenschaft haben. Es kann beispielsweise ausgegeben werden wieviele (Aggregatfunktion COUNT) Nutzer unserer Bibliothek am selben Wohnort (gemeinsame Eigenschaft) wohnen. Soll auf den Wert der Aggregatfunktion eine Bedingung gesetzt werden, erfolgt dies mit HAVING. Die Syntax sieht dabei so aus:

```
> SELECT <gem_eigenschaft>, <Aggregat> FROM ... GROUP BY <gem_eigenschaft>
->  HAVING <Aggregat> =|<|> ...; .
```
[Fuchs: Kap. 7.3]

Aufgabe (Bücherei-DB):

Beantworten Sie durch Abfragen auf die Tabelle `Exemplar` folgende Fragen:

- Wieviele Medien hat die Bücherei?
- Wieviele davon sind Bücher, CDs, DVDs?
- Wieviele Medien hat die Bücherei, aufgeschlüsselt nach Erscheinungsjahr?
- Wie sehen die Zahlen für Medien jünger als 1980 aus?
- Aus welchen Erscheinungsjahren sind mehr 2 Medien aufgenommen worden?

4.4 Sichten / Views

> Sichten sind gespeicherte SELECT-Anweisungen, die bis auf ihre Definition keinen zusätzlichen Speicherplatz benötigen. Sie verwenden automatisch die aktuellen Datensätze der zugrunde liegenden Tabellen.[5]

Im Bereich der Anwendungsentwicklung ist die Verwendung eher unüblich, da die SQL-Anweisungen im Code gekapselt sind.

Bei manuellem Zugriff auf die Datenbank stellen Sichten eine Erleichterung der Arbeit dar. Darüberhinaus kann der DBA über eine entsprechende Rechtevergabe erreichen, dass ein bestimmter Nutzer nur einen passenden Ausschnitt der Daten sehen kann.

- Anlegen von Sichten:

  ```
  CREATE VIEW <view> AS SELECT ... ;
  ```

  Achtung: Die Abfragen dürfen die Klauseln GROUP BY, HAVING, ORDER BY, UNION nicht enthalten.

[5] Herdt (2004): S. 132

- Anwenden von Sichten:

  `SELECT ... FROM <view> ... ;`

- Löschen von Sichten: `DROP VIEW <view>;`

.[Fuchs: Kap. 11]

Aufgabe:

- Speichern Sie die erste Abfrage der Übungen aus Abschnitt 4.2 als Sicht und führen die diese aus.

4.5 Transaktionen

Datenbanksysteme ermöglichen parallelen Mehrbenutzer-Betrieb. Sie stellen sicher, dass sich die verschiedenen Benutzer auf Datenebene nicht in die Quere kommen. Der zentrale Begriff dabei ist die „Atomarität" und die „Serialisierbarkeit". Alle Abfragen und Änderungen einer Transaktion werden als zu einem Zeit stattfindend betrachtet. Und wenn diese gedachten Zeitpunkte in eine konsistente Reihenfolge gebracht werden können, ist das ganze System konsistent, die Transaktionen werden durchgefürt. Ist dies nicht möglich, so muss eine Transaktion zurückgerollt werden. Das Transaktionswesen reägt also elementar dazu bei, dass Datenbanksysteme die ACID-Eigenschaften aufweisen:

Atomicity - Atomarität	Alle Abfragen und Änderungen einer Transaktion werden als zu einem Zeit stattfindend betrachtet.
Consitency - Konsistenz	Zum Ende einer Transaktion befindet sich die Datenbank in einem konsistenten Zustand.
Isolation	Die Datenbank versucht die Benutzer so gegeneinader abzuschirmen, dass die im Idealfall nichts davon spüren.
Durability - Dauerhaftigkeit	Mit dem Ende einer Transaktion sind die Daten persisent abgelegt.

Die Grad der Isalation wird durch den *isolation level* bestimmt. Der SQL-Standard definiert folgende Grade:

`READ UNCOMMITTED`	Die Trennung ist schwach ausgeprägt. Andere Nutzer können Datenänderungen sehen, bevor die festgeschrieben wurden.
`READ COMMITTED`	Datensätze, die bearbeitet werden, werden gesperrt. Probleme kann es geben, wenn Datensätze von anderen Transaktionen hinzu- oder wegkommen
`REPEATABLE READ`	Durch das Einfrieren des Datenstands zu Trasaktionsbeginn kann es zu Phantom-Ergebnissen kommen.
`SERIALIZABLE`	Höchster Isolationsgrad. Aber Sperren können zu Blockaden und Verzögerungen führen.

Für das Datenbanksystem ist der Standardwert `REPEATABLE READ` (V5.6). Um für eine Session einen anderen Isalationgrad einzustellen, kann folgendes eingegeben werden:

`SET SESSION TRANSACTION ISOLATION LEVEL <level>;`

.[Fuchs: Kap. 14]

Folgende Sequenzen demonstrieren die jeweiligen Probleme[6]:

- *uncommitted read*

Transaktion 1	Transaktion 2
`START TRANSACTION;`	`START TRANSACTION;`
`INSERT INTO buchung(flug_id, sitzplatz,`	
`-> passagier_id, preis) VALUES`	
`-> (172, '34C', 38, 429.00);`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`
`ROLLBACK;`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`

- *non-repeatable read*

Transaktion 1	Transaktion 2
`START TRANSACTION;`	`START TRANSACTION;`
`INSERT INTO buchung(flug_id, sitzplatz,`	
`-> passagier_id, preis) VALUES`	
`-> (172, '34C', 38, 429.00);`	
`SELECT COUNT(*) FROM buchung`	
`-> WHERE flug_id = 172;`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`
`COMMIT;`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`

- *phantom read*

Transaktion 1	Transaktion 2
`START TRANSACTION;`	`START TRANSACTION;`
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`
`DELETE FROM buchung WHERE`	
`-> buchung_id = 55099799;`	
`SELECT COUNT(*) FROM buchung`	
`-> WHERE flug_id = 172;`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`
`COMMIT;`	
	`SELECT COUNT(*) FROM buchung`
	`-> WHERE flug_id = 172;`
	`SELECT * FROM buichung`
	`-> WHERE buchung_id = 55099799;`

[6]Pröll (2015): S. 183ff.

Aufgabe:

- Führen Sie die drei Beispielszenarien für die verschiedenen Isolationsgrade durch.

Anmerkung: Sie können mit `SHOW engine innodb status;` die laufenden Transaktionen anschauen. Eine Transaktion wird erst mit dem ersten ihr zugeordneten Befehl gestartet.

5 Die Datenbank MySQL / MariaDB

5.1 Einführung

Bei MySQL gibt es drei Linien

- Kommerziell: *MySQL Enterprise Edition*, *MySQL Cluster CGE*
- OpenSource: *MySQL Community Edition* unter GPL

Anmerkung: *My* soll der Vorname der Tochter des Co-Gründers Michael Widenius sein[7].

Dazu gibt es kompatible Alternativen:

- MariaDB: Abspaltung der ursprünglichen Gründer
  - Weiterentwicklung von `MyISAM`
  - Standard von vielen Distributionen (z.B. Debian)
- Galera: Clusterbetieb mit Galera-Protokoll
- Percona-DB: mit Xtra-Engine und Percona-Backup

Die Vorteile von MySQL im Überblick[8]:

- einfach zu erlernen und zu bedienen
- flexibel und vielseitig
- skalier- und anpassbar
- hochleistungsfähig
- hochverfügbar
- stabil
- sicher
- einsatzerprobt
- Open Source
- kostenlos bzw. günstig in der Anschaffung

[7] Pröll (2015): S. 40.
[8] Pröll (2015): S. 43.

- ständige Weiterentwicklung durch die Firma (Oracle) und die Community

Geschichte von MySQL

1995 riefen die Schweden D. Axmark und A. Larsson und der Finne M. Wildenius das Projekt ins Leben. Ausgangslage:

- ISAM-Speicher-Engine bei einem schwedischen Telekommunikationsunternehmen
- mSQL Datanbank: flexibel, einfach zu bedienen

ABER: Die Systeme ließen sich nicht zusammenbringen. Daher entschieden sich die Gründer zu einer Eigenimplementierung. 1996 erscheint die Version 3.11.1 für Solaris. Diese Versionsnummer wurde gewählt, weil sich der Funktionsumfang an der mSQL-Version derselben Versionsnummer orientierte. Da das Projekt selbst OpenSource-Elemente benutzte, stand es von Anfang an als OpenSouce-Software zur Verfügung. Hinter MySQL stand die dafür gegründete Firma MySQL SG[9]. Mittlerweile ist MySQL über Aufkäufe bei Oracle gelandet.

MySQL-Versionen bestehen aus drei Ziffern (z.B. 5.7.6). Diese bedeuten:

1. Hauptversion: Wird nur bei Änderungen im Speicherformat geändert.
2. Release-Serie: Wird bei wichtigen neuen Features hochgezählt.
3. Versionsnummer innerhalb der Release-Serie

Während die V3 wegen der fehlenden ACID[10]-Eigenschaften noch kaum „Datenbank" genannt werden konnte, hat sich MySQL mit der V5 zu einem System entwickelt, das eine vollentwickelte, den SQL-Standard (weitgehend) erfüllende Datenbank darstellt.

- 2008 Verkauf an Sun
- 2009 Weiterverkauf an Oracle
- Gründung von MariaDB von Monty Widenius[11]

  Eigenschaften von Version 10:

  - User-Rollen
  - Volltext für Chinesisch, ...
  - Global Transaction ID
  - neue Storage Engines (Aria)
  - „Galera ready"
  - Datenverschlüsselung
  - InnoDB 5.7 (default)
  - GIS, Json-Funktionen

[9] Aktiengesellschaft
[10] s. Kap. 4.5
[11] Linux Magazin 01/19

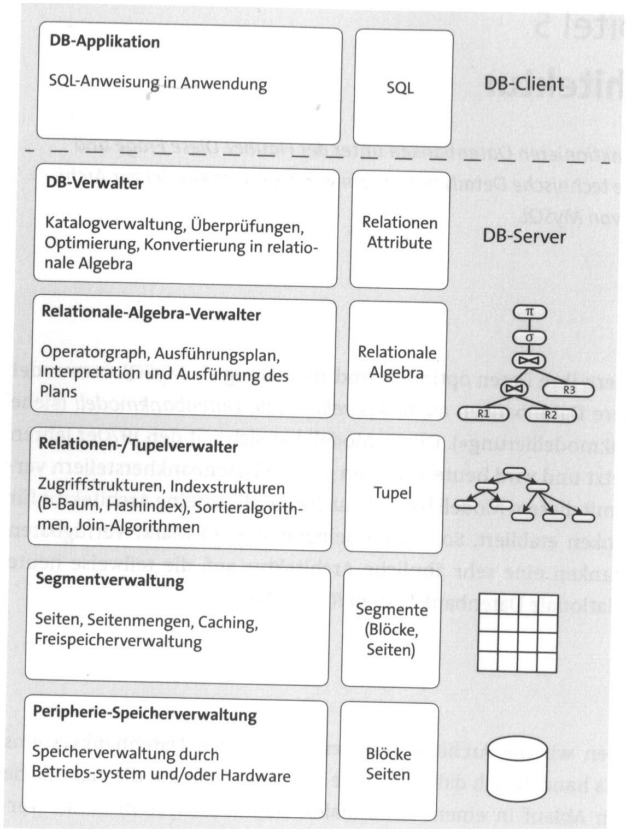

Abbildung 9: Fünf-Schichten-Modell eines Datenbanksystems

- Konnektoren unter LGPL
- Mariabackup
- Oracle-Modus
  - Oracle Style Sequences
  - Oracle PL/SQL
- Versionierte Tabellen
- Open Query Graph
- „Wie MySQL wird auch bei MariaDB aufs Schreiben hin optimiert."

5.2 Fünf-Schichten-Modell

Im Folgenden wird die Architektur eines Datenbanksystems näher beschrieben. Es handelt sich dabei um eine abstrakte Sicht auf die eigentliche Architektur, was den Ablauf in einem relationalen Datenbanksystem besser verständlich macht[12].

Abbildung 9 zeigt die fünf Schichten eines Datanbanksystems, die im Folgenden näher erklärt werden.

[12]Pröll (2015): S. 151ff.

- Datenbankapplikation

  Datenbanken sind kein Selbstzweck. Sie versorgen eine Anwendung mit Daten. Diese Anwendungsschicht stellt die oberste Schicht dieses Modells dar. Die Anwendung sendet SQL-Befehle an das Datenbanksystem.

- Datenbankverwaltung

  Diese oberste Schicht des Datenbanksystems prüft eingenende SQL-Anfragen auf syntaktische Korrektheit und grundsätzliche Sinnhaftigkeit (sind die genannten Tabellen und Attribute überhaupt im System?). Hier erfolgt auch die Umsetzung in eine der relationalen Algebra gehorchende Notation.

- Relationale Algebra

  Im Rahmen der relationalen Algebra können Zugriffsregeln äquivalent umgeformt werden, so dass ihre konkrete Ausführung möglichst performant ist. Diese Umformungen sind die Aufgabe des Optimizers. Ein guter Optimizer war stets das gut gehütete Geheimnis einer kommerziellen Datenbank. Damit ein Optimizer gute Arbeit leisten kann, benötigt er statistische Werte der Daten, was deren absolute Menge und Werteverteilung angeht.

- Relationen-/Tupelverwaltung

  Diese Schicht realisiert die Speicherung der Daten und den Zugriff auf dieselben auf Relationsebene. Hier ist auch die Verwaltung der Indizes untergebracht.

- Segmentverwaltung

  Die Segmantverwaltung organisiert die Gruppierung von Datensätzen zu Segmenten. Ein Segment ist die Einheit, die auf einmal von der Platte gelesen wird, respektive auf die Platte geschrieben wird.

- Peripherie-Speicherverwaltung

  Diese Schicht organisiert die Ablage im Speicher, sei er füchtig oder persistent. In früheren Zeiten, als RAM sehr knapp war, lag in dieser Schicht eine Stärke kommerzieller Datenbanksysteme. Oracle, IBM und Co. investierten in eigene Dateisysteme um den Festplattenzugriff möglichst performant zu gestalten. MySQL verwendet das Dateisystem des Betriebssystems. Das wird heute durch die, durch den meißt großen verfügbaren RAM, geschaffene Möglichkeit der Pufferung mehr als ausgeglichen.

5.3 MySQL-Architektur

Betrachtet man die MySQL-Architektur in Abbildung 10, so fällt auf, dass vom Relationen-/Tupelverwalter abwärts verschiedene Implementierungen zur Verfügung stehen. Die verschiedenen *storage engines* haben ihre je eigenen Stärken und Schwächen, was MySQL als Ganzes zu vielseitig macht.

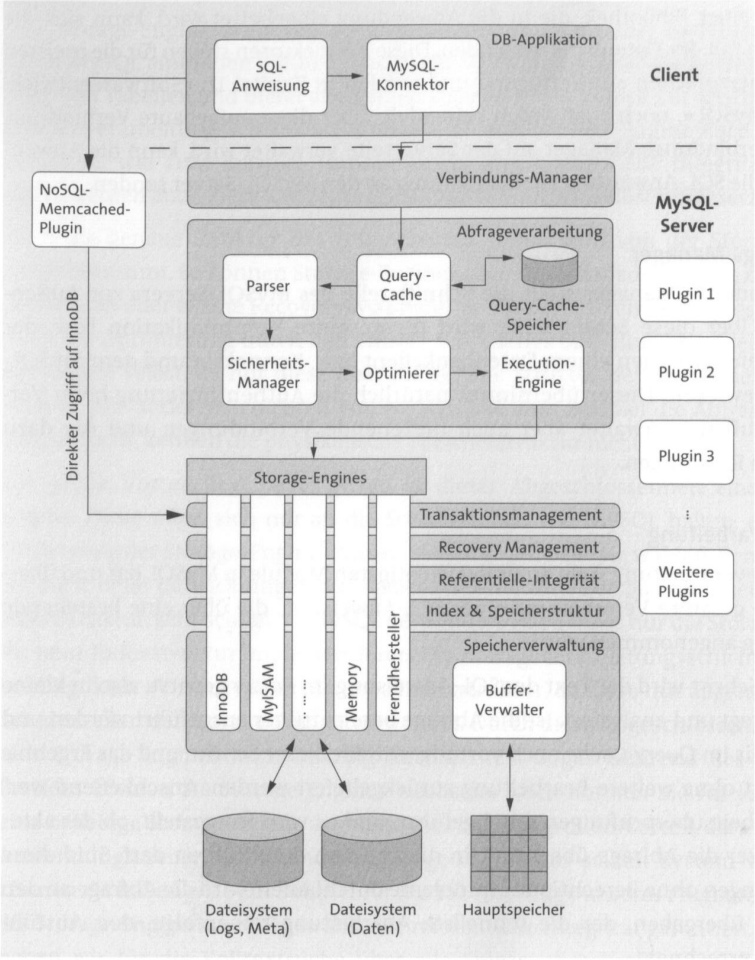

Abbildung 10: MySQL-Server-Architektur

Um Daten mit dem mySQL-Server auszutauschen, muss sich ein Client mit dem Server verbinden. mySQL stellt dazu verschiedene Schnittstellen zur Verfügung. Diese unterteilen sich in Konnektoren und APIs.

- Konnektoren = Nachrichtenschnittstelle
  - ODBC, .NET, JDBC, C++, C, Python
- API = Programmbibliothek
  - C, PHP, Perl, Python, Ruby

Und noch mehr!

Der Standard-MySQL-Kommandozeilen-Client nutzt die C-API.
Ein Hinweis: Mit `pager less -S` lässt sich eine Darstellung ohne Zeilenumbruch innerhalb einer Zeile einstellen.

Aufgabe:

Überprüfen Sie in der Paketverwaltung, welche Konnektoren installiert sind (`dpkg -l |grep mysql`).

Abfrageverarbeitung

In der Abfrageverwaltung sind die Schichten *Datenbankverwaltung* und *Relationale Algebra* realisiert. Hier sind auch der *query cache* und der Optimizer, „die Intelligenz von MySQL"[13] zu finden.

Der *Optimierer* ist das Herzstück einer Datenbank. Doch auch Herzen können mal falsch schlagen. Daher muss, falls es zu Performace-Engpässen kommt, das Ergebnis des Optimierers, als der Ausführungsplan, kritisch bereachtet werden. Dies wird im Abschnitt 7.5 näher betrachtet.

5.4 Storage Engine

MySQL liefert folgende *storage engines* aus:

- InnoDB: Die Standardmaschine mit gutem Transaktionsverhalten
- MyISAM: Stärke im Zugriff, aber kein Transaktionsschutz
- MEMORY: Transiente Daten: schnell im Zugriff, aber flüchtig

InnoDB

Die InnoDB ist die Engine mit der größten Innovation in den letzen Jahren. Hier wurden all die Merkmale entwickelt, die eine standardkonforme Datenbank bieten muss. Zu ihren Stärken gehören

- ACID-konformität
- referentielle Integrität (*foreign keys*)
- *concurrency control* für die Abfrage-Parallelisierung
- B-Baum als Hauptindexstruktur
- Daten werden nach Primärindex sortiert abgelegt
- adaptive Hash-Indizes

[13] Pröll (2015): S. 168.

Zu den Optimierungsmöglichkeiten der InnoDB gehören Speichergröße, die die Engine zur Verfügung bekommt (`innodb_buffer_pool_size`), die Größe der Logfiles (`innodb_log_file_size`), sowie der zugehörige Logbuffer (`innodb_log_buffer_size`). Letztere dienen zur Transaktionsverwaltung, damit bei einem Rollback der alte Zustand wieder hergestellt werden kann.

Bei Versionen vor 5.6 werden alle Tabellen, die durch die InnoDB verwaltet werden, in einer einzigen Datei (`ibdata1`) abgelegt. Bei diesen Systemen sollte der Paramter `innodb_file_per_table` in `my.cnf` gesetzt werden[14]. Die einzelnen Dateien tragen dann die Endung `.ibd`.

InnoDB bringt auch einige Einschränkungen mit:

- Indizes können nur für die ersten 767 Bytes einer Spalte erstellt werden. Nur diese liegen sequentiell in der Datei. Ist die Spalte breiter, wird der Rest an anderer Stelle gespeichert.
- maximale Tabellengröße: 64 TB
- maximale Spaltenzahl: 1017
- maximale Zeilengröße mit/ohne `VARCHAR`: 8 kB / 65 kB
- maximale Schlüsselgröße: 3072 By
- NULL in indizierten Spalten: ja
- Index auf `BLOB` und `TEXT`: eingeschränkt (s. Bedingung 1)

Aufgabe:

- Prüfen Sie den Wert von `innodb_file_per_table`
- Besichtigen Sie die Dateien, in denen die Daten der InnoDB liegen.

MyISAM

MyISAM ist die älteste Engine von MySQL. Sie brachte durch ihre enorme Performanz „das Internet zum Laufen". Bis zur Version 5.6 war nur hier der Volltextindex möglich. Weitere Merkmale sind die GIS[15]-Erweiterung und der R-Raum-Index. Eine spezielle Sortierung der Datensätze innerhalb der Datei existiert nicht.

Hinweis zur Datenmodellierung: Enthält die Tabelle `VARCHAR` oder `TEXT` führt dies zu variablen Zeilenlängen. Dabei kann es bei Löschungen und Wiedereinfügungen zu ungenutzem Speicherplatz kommen. Solche Tabellen müssen daher von Zeit zu Zeit defragmentiert werden (`OPTIMIZE TABLE`).

Weitere Besonderheiten und Einschränkungen:

- tabellenweiter Sperrmechanismus
- maximale Tabellengröße: 256 TB
- maximale Zeilenzahl: 2^{64}

[14] Pröll (2015): S. 197.
[15] Geoinformationssystem

- maximale Anzahl von Indizes: 64
- maximale Anzahl von Spalten im Index: 16
- maximale Schlüsselgröße: 1000 By
- NULL in indizierten Spalten: ja
- Index auf `BLOB` und `TEXT`: ja

MEMORY

- B-Baum und Hashindizes
- transient

Zusammenfassende Übersicht s. Abbildung 11.

Eigenschaft	Innodb	MyISAM	MEMORY
maximale Größe der Tabelle	64 TB	256 TB	Hauptspeicher
MVCC (wichtig, um ACID-Eigenschaft zu gewährleisten)	ja	nein	nein
Transaktionen	ja	nein	nein
Sperrebene	Zeile	Tabelle	Tabelle
referentielle Integrität (Foreign Keys)	ja	nein	nein

Eigenschaft	Innodb	MyISAM	MEMORY
Index	B-Baum, adaptiver Hash	B-Baum	Hash, B-Baum
Clustered Primärindex	ja	nein	nein
Kompression	ja	möglich (nur lesend)	nein
GIS-Unterstützung	ja (Version 5.7)	ja	nein
Volltextindex	ja	ja	nein
Replikation	ja	ja	ja
Query-Cache	ja	ja	ja
Delayed Inserts (seit Version 5.6 nicht mehr unterstützt)	nein	ja	ja

Abbildung 11: Wichtige Eigenschaften der Storage-Engines

Weitere Engines

- MERGE
  - Zusammenfassung von struturell gleichen MyISAM-Tabellen zu einer virtuellen Tabelle

- ARCHIVE
  - Komprimierte Tabelle ohne Löschmöglichkeit
  - keine Indizes
  - kein `BLOB`
- FEDERATED
  - Virtuelle Tabelle wird durch Datenabfragen bei weiteren MySQL-Servern realisiert.
- CSV
  - kein *auto increment*, Index, Schlüssel
  - Vorteil: direkt im Dateisystem lesbar
- BLACKHOLE
  - keine Datenablage
  - Anwendung: Mitschreiben von Änderungslogs bei verteilten Datenbanken

Darüber hinaus gibt es weitere Fremdhersteller-Engines[16]. Beispiele:

- XtraDB: Basierend auf InnoDB mit Performanceverbesserungen
- Aria: Nachfolger der MyISAM-Engine. Entwickelt im Rahmen des Forks MariaDB.
- Spider: Verteilung mit XA-Transaktionsmonitor, partitionierte Verteilung[17]

Für InnoDB und XtraDB gibt es das Galera-Plugin für den Master-Master-Clusterbetrieb (s. Abschnitt ??). Galera vermeidet die Nachteile des Zweiphasenkommits.

5.5 Information-Schema

Viele Informationen über den Zustand der Datenbank lassen sich über das *Information Schema* erhalten. Auf dem *Information Schema* arbeiten auch die `SHOW`-Befehle. Hier ein paar Beispiele für nützliche Abfragen[18]:

- Welche Berechtigungen hat der Benutzer *hugo*?

```
> SELECT * FROM user_privileges WHERE grantee LIKE "'hugo'@'localhost'";
```

- Wieviele Verbindungen wurden abgebrochen?

```
> SHOW STATUS LIKE 'Aborted_connects';
```

- Wie lange läuft der Server?

```
> SHOW STATUS LIKE 'Uptime';
```

[16] Pröll (2015): S. 211
[17] http://spiderformysql.com (2.11.2022)
[18] Pröll (2015): S. 303ff.

- Wieviele Abfragen wurden gestellt?

  ```
  > SHOW STATUS LIKE 'Queries';
  ```

- Welche Tabellen haben die InnoDB-Engine?

  ```
  > SELECT table_schema, table_name FROM tables WHERE engine='InnoDB';
  ```

- Weitere Informationen zu den InnoDB-Tabellen wie Anzahl der Einträge, Details über Indizes:

  `> SELECT * FROM innodb_sys_tablestats;` (eine Sicht im *Information Schema*, ab 5.6)

Verschiedene Metriken können in der Tabelle `INNODB_METRICS` gesammelt werden. Welche Art von Metriken zur Verfügung stehen, kann über `SELECT name FROM INNODB_METRICS;` in Erfahrung gebracht werden.

Performancekritisch sind beispielsweise Deadlocks. Ihr Auftreten kann durch das Einschalten einer entsprechenden Metrik überwacht werden. Das Einschalten einer Metrik erfolgt mit dem Befehl
`SET GLOBAL innodb_monitor_enable = '<werkzeugnamen>';` (z.B. `lock_deadlocks`)

Da das Erstellen von Metriken selbst auch Rechenzeit benötigt, sollten im Normalfall alle Zähler deaktivert sein. Die Deaktiviereung erfolgt mit dem Befehl
`SET GLOBAL innodb_monitor_disable = 'all';`

Aufgabe:

- Führen Sie diese Abfragen im *Information Schema* durch.
- Schalten Sie die Deadlock-Metrik ein und fragen Sie die Werte ab.

5.6 Volltextsuche

MySQL stellt für effektive Suche in großen Textmengen den *Volltextindex* zur Verfügung. Damit können Suchabfragen, die ein `LIKE` enthalten beschleunigt werden. Der Volltextindex steht aktuell nur für MyISAM-Tabellen zur Verfügung. In der Flughafen-DB hat die Tabelle `flugzeug_typ` einen solchen Index. Soll bei einer Abfrage dieser Index benutzt werden, geschieht das mit dieser Syntax[19]:
`> SELECT * FROM tab WHERE MATCH(<spalte/n>) AGAINST('<suchstring>');`

Die Suche kann dabei in folgender Weise modifiziert werden:

- `IN NATURAL LANGUAGE` führt zu einer Veroderung der Suchwörter
- `IN BOOLEAN MODE`: Hier können die Wörter mit Operatoren versehen werden:
  `+Wort`: Wort muss enthalten sein
  `-Wort`: Wort darf nicht enthalten sein
  `Wort*`: Wort beginnt mit „Wort"
  `<Wort`: Wort wird mit geringerer Relevanz gesucht
  `>Wort`: Wort wird mit höherer Relevanz gesucht

[19] Pröll (2015): S. 412.

`"Wort1 Wort2"`: Es wird nach exakt dieser Wortkombination gesucht
kein Operator: Wort ist optional
Runde Klammern dienen der Gruppierung

Bei der Erstellung des Index werden all zu oft vorkommende Wörter, sogenannte Stoppwörter, nicht berücksichtigt. Diese Stoppwörter finden sich in einer Datei, die durch den Konfigurationseintrag `ft_stopword_file` festgelegt ist. Darüber hinaus muss die Länge der indizierten Wörter zwischen `ft_min_word` und `ft_max_word` liegen.

Der Volltextindex ist sehr leistungsfähig, hat aber auch Nachteile[20]:

- großer Speicherbedarf
- Performanceverluste bei Datenänderungen

Aufgabe:

- Suchen Sie in der Tabelle `flugzeug_typ` nach Flugzeugen, deren Bezeichnung oder Beschreibung 'A380' und/oder 'Airbus' enthält. Arbeiten Sie dabei auch mit den Modifikationen.

5.7 Benutzerverwaltung

Bei der Benutzerverwaltung können zwei Szenarien unterschieden werden. Arbeitet eine Anwendung auf der Datenbank, so gibt es für diese Anwendung einen Nutzer, der mit all den Rechten ausgestattet ist, die die Anwendung auf der Datenbank benötigt. Dies umfasst meist auch das Anlegen und Löschen von Tabellen. Die Rechte einzelner Nutzer innerhalb der Anwendung verwaltet die Anwendung selbst.

Arbeiten hingegen Benutzer direkt auf der Datenbank, so wird jedem dieser Nutzer ein eigener Zugang zum System eingerichtet. Die Verwaltung der Nutzerrechte erfolgt in diesem Fall mit Mitteln der Datenbank.

Für beide Arten von Nutzern kann unterschieden werden, ob es sich um einen lokalen Nutzer (`@localhost`), um einen Nutzer handelt, der nur von einem bestimmten Rechner aus zugreifen kann (`@<host>`) oder um einen globalen Benutzer.

- Anlegen eines neuen Nutzers:

```
CREATE USER <user>[@...] IDENTIFIED BY '<password>';
```

Ein so angelegter Nutzer hat aber noch keinerlei Rechte. Einer Anwendung werden üblicherweise alle Rechte auf einer bestimmten Datenbank eingeräumt.

- Einräumen von Rechten auf eine Datenbank:

```
GRANT ALL ON <database>.* TO <user>;
```

Desweiteren:

[20] Pröll (2015): S. 420.

- Löschen eines Nutzers:

  ```
  DROP USER <user>[@...];
  ```

- Löschen der Rechten eines Benutzers:

  ```
  REVOKE GRANT ALL FROM <user>;
  ```

Sollen Rechte feingranularer vergeben werden, bietet MySQL die Verwaltung auf folgenden Ebenen an:

- `*.*`: global
- `<db>.*`: Datenbankebene
- `<db>.<table>`: Tabellenebene

Auch die vergebenen Rechte können näher spezifiziert werden. Die einzelnen Rechte sind:

`SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, ...`

Die Rechte `SELECT, UPDATE` können dabei noch auf einzelne Spalten beschränkt werden.

.[Fuchs: Kap. 13]

Aufgabe:

Legen Sie einen Benutzer `kurs` an, die alle Rechte auf die Datenbanken `Bucherei` und `Flug` hat.

5.8 Stored Procedures

Stored procedures kapseln mehrere Datenbankzugriffe zu einem einzigen Aufruf. [Fuchs: Kap. 15]

Aufgabe (Bucherei-DB):

- Legen Sie eine Tabelle `Alte_Ausleihen` an.
  - Diese Tabelle hat keine Fremdschlüssel.
  - Für die Benutzer werden Vor- und Nachname in die Tabelle aufgenommen, für das Exemplar die Signatur.
- Schreiben Sie eine *stored procedure* `benutzer_loschen`, die als Parameter eine Benutzer-ID bekommt. Diese Prozedur soll:
  - Prüfen, ob der Benutzer noch offene Ausleihen hat. Falls ja, wird das Löschen abgebrochen.
  - Alle Ausleihen, die diesen Benuzter betreffen, samt Nachname, Vorname und Signatur in die neue Tabelle kopieren,

  - diese Ausleihen löschen,
  - den Benutzer löschen
- Schreiben Sie eine *stored procedure* `exemplar_loschen`, die als Parameter eine Signatur bekommt und entsprechend funktioniert.

5.9 Trigger

Trigger sind *stored procedures*, die aber nicht durch den Benutzer, sondern durch ein Ereignis in der Datenbank selbst aufgerufen werden. [Fuchs: Kap. 16]

Aufgabe (Bucherei-DB):

In unserer Datenbank soll eine Statistik geführt werden, wie oft ein Medium (nicht Exemplar!) ausgeliegen wird.

- Legen Sie eine Tabelle `Ausleihstat` mit folgenden Spalten an (neben dem technischen Primärschlüssel):
  - `medium_id`: Fremdschlüssel auf `Medium`
  - `anzahl`: Zähler für die Ausleihen
- Schreiben Sie einen Trigger, der bei einer Neuanlage in der Tabelle `Ausleihe`
  - prüft, ob es in `Ausleihstat` schon einen Eintrag für das zugehörige Medium gibt,
  - falls nein, einen Eintrag mit `anzahl = 1` anlegt,
  - falls ja, die `anzahl` um eins erhöht.

6 MySQL-Administration

MySQL bietet eine Reihe von hauseigenen Administrationstools. Für die Konsole sind dies[21]:

- `mysql`: Datenbank-Universalclient
- `mysqladmin`: Werkzeug für die Administration
- `mysqlimport`: Werzeug für den Datenimport
- `mysqldump`: Export und Backup von Daten
- `mysqlcheck`: Überprüft und repariert Tabellen
- `mysqlslap`: Benchmarking-Programm

[21] Pröll (2015): S. 261.

Anmerkung: Die meisten Operationen lassen sich auch innerhalb des Universalclients ausführen. Im Batchbetrieb kann die Formulierung mit einem der Spezialbefehle einfacher zu formulieren sein.

Als grafisches Werkzeug steht die *MySQL Workbench* zur Verfügung.

6.1 Backup, Restore, Upgrade

Backup

Bei der Planung von Backups müssen folgende Randbedingungen berücksichtigt werden:

- Wie oft muss ein Backup stattfinden?
  Suchen Sie das Optimum zwischen den Kosten eines Datenverlusts und den Backupkosten.
- Wann soll ein Backup stattfinden?
  Während eines Backups müssen die Tabellen für Änderungen gesperrt werden, damit der Backup nicht inkonsistent ist.
- Planen Sie „Feueralarm-Übungen".
  Der beste Backup ist wertlos, wenn Sie nicht auch die Wiederherstellung getestet haben.

Grundsätzlich wird beim Backup das logische Backup und das physische Backup unterschieden. Beim physischen werden einfach die Dateien des Dateisystems mit Mitteln des Betriebssystems gesichert. Dieses Verfahren hat aber Nachteile:

- Strukturen, die nur im Speicher ihr Abbild haben (`MEMORY`-Engine) werden nicht gesichert.
- Ein Zurückspeichern eines solchen Backups ist nur in eine identische Umgebung möglich.
- Backup und Restore sind nur bei heruntergefahrenem Server möglich.

Darüber hinaus gibt es das Werkzeug `xtrabackup`, das es ermöglicht, die Dateien der InnoDB bei laufendem Betrieb herauszukopieren[22].

Bei einem logischen Backup legt MySQL die Daten in Form von SQL-Statements ab, mit denen die Datenbank rekonstruiert werden kann. Dieses Verfahren hat den Vorteil, dass damit auch Upgrades gefahren werden können. Doch auch diese Backup-Methode hat Nachteile:

- Höherer Ressourcenbedarf, damit längere Zeitdauer
- Höherer Platzbedarf der Backupdateien:
  Dies kann allerding durch eine anschließende Kompression der Daten abgemildert werden.

Das Standard-Werkzeug für den logischen Backup ist bei MySQL der `mysqldump`. Als Optionen nimmt er neben den Login-Daten auch die Datenbank, die er sichern soll.

Sichern und Wiedereinspielen einer Sicherung erfolgt also mit folgenden Befehlen:

```
mysqldump -u<user> -p<password> --opt --single-transaction \
  --dump-date <database> | gzip > <file>
cat <file> | mysql -u<user> -p<password> <database>
```

[22] Pröll (2015): S. 291f.

Bedeutung der Optionen → MySQL8.0.[23]

Zwei Anmerkungen dazu:

- Eine ständige Quelle des Ärgernisses sind imkompatibilitäten beim verwendeten Zeichensatz. Die Zeichensätze des MySQL-Servers, des Betriebssystems, der Befehle `mysqldump` und `mysql`, sowie der Sicherungsdatei (physisch und logisch) müssen aufeinander abgestimmt sein.
- Zu einer kompletten Datenbanksicherung gehören auch die Konfigurationsdateien in `/etc/mysql`.

Aufgabe:

- Verbinden Sie sich mit der Bücherei-DB und prüfen Sie, ob im MySQL-Client die Umlaute richtig angezeigt werden.
- Führen Sie den Datenbankexport durch und prüfen Sie, ob in der Exportdatei die Umlaute richtig angezeigt werden.
- Legen Sie eine neue Datenbank an und importieren Sie die Exportdatei. Prüfen Sie im MySQL-Client, ob die Umlaute richtig angezeigt werden.

Anmerkung: Die MySQL-Kommandozeilen-Befehle kennen den zusätzlichen Parameter `--default-character-set`.

Recovery von Bedienfehlern[24]

Bei Datenbanken, auf denen Nutzer direkt arbeiten, kann es vorkommen, dass es zu gravierenden Bedienfehlern kommt. In diesem Fall ist es nützlich ein feingranulares Recovery zu haben. Dies ermöglicht das Binärlog. In einem Binärlog werden alle datenändernden Transaktionen aufgezeichnet. Im Fehlerfall ist folgendes auszuführen:

- Schließen des aktuellen Binärlogs mit `mysql> FLUSH LOGS;`
- Letztes Komplettbackup einspielen
- Alle Datenänderungen bis zum „Unfall" und danach nachziehen

Zum Einschalten des Binärlogs muss die Variable `log_bin` auf die Binärlogdatei zeigen. Anschaun und Einspielen eines Teils eines Binärlogs erfolgt mit:

```
mysqlbinlog <bin-file>
mysqlbinlog --stop-date "2015-03-15 17:34:09" <bin-file> | mysql -uroot -p
mysqlbinlog --start-date "2015-03-15 17:34:15" <bin-file> | mysql -uroot -p
```

[23] https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html#mysqldump-option-summary (2.11.2022)
[24] Pröll 2015, S. 293ff.

Backup über Replikation

Grundsätzlich stellt auch die Replikation eine Form des Backups dar. Zu beachten ist aber, dass auch alle Fehler mitrepliziert werden. Daher ist es auch in diesem Fall ratsam, dann und wann ein Komplettbackup zu schreiben. Weitere Informationen dazu in Kapitel **??**.

6.2 Die zentrale Konfigurationsdatei my.cfn[25]

Die Zentrale Konfigurationsdatei findet sich in Linux-Systemen unter `/etc/my.cnf`. MySQL bietet viele Einstellungsmöglichkeiten, die sich aber auch negativ auf die Performance auswirken können. MySQL passt sich gut an die zur Verfügung stehenden Resourcen an, so dass für die Konfiguration gilt: Weniger ist mehr!

Grundsätzlich gilt:

> Da es keine allgemeingültigen Empfehlungen gibt, die jeder MySQL-Installation zu ungeahnten Geschwindigkeiten verhilft und alle Performance-Engpässe beseitigt, müssen Sie die betreffenden Einstellungen auf Ihrem eigenen System ausprobieren und testen. Zu viele Parameter beeinflussen sich gegenseitig und sind immer von den koonkreten Anforderungen abhängig, um allgemeingütige Werte angeben zu können. Beispielsweise ist es ein Unterschied, ob Sie nur MyISAM-Tabellen einsetzen oder einen Mix aus InnoDB- und MyISAM-Engines verwenden.[26]
>
> A good rule of thumb is if you can't provide a valid reason for increasing any of these buffers, keep them set to the default values.[27]

Der Startbefehl
`mysqld --help --verbose` gibt Auskunft, welche Konfigurationsdateien in welcher Reihenfolge geladen werden und zeigt die daraus resultierende Startconfiguration. Der Befehl
`mysqladmin variables -uroot -p<password>` gibt die aktuelle Konfiguration aus.

Die Konfigurationsdatei ist in Abschnitte geteilt, die durch echige Klammern eingeleitet werden. Die Wertzuweisung an einzelne Parameter ergolgt durch „=". Beispiel:

```
[mysqld]
server_id = 25
```

Einige Paramter können die Performance beeinflussen:

- `skip-external-locking`: Prüft keine externe Sperren von MyISAM-Tabellen. Dies macht den Zugriff auf MyISAM schneller, es darf aber kein MyISAM-Zugriff parallel (z.B. über `myisamchk`) erfolgen.
- `key_buffer_size`: Hauptspeicher für die Verwaltung der Inidizes von MyISAM-Tabellen. Faustregel: 1/4 des Arbeitsspeichers bei intensiver MyISAM-Nutzung. Ob der Wert groß genug gewählt ist, kann aus dem Verhältnis `key_reads` zu `key_read_requests` ermittelt werden. Dieser Wert sollte 1% nicht überschreiten.

[25] Pröll (2015): S. 242ff.
[26] Pröll (2015): S. 325.
[27] https://haydenjames.io/my-cnf-tuning-avoid-this-common-pitfall (2.11.2022)

- `innodb_buffer_pool_size`: Abfragespeicher für die InnoDB. Bis zu 80% des Hauptspeichers, falls nur InnoDB-Tabellen verwendet werden.
- `max_connections`: Eine Erhöhung dieser Einstellung lässt mehr parallele Clients zu. Damit werden mehr Verbindungen gespeichert, was den Speicherbedarf erhöht. Zur Optimierung dieses Paramters können folgende Variablen betrachtet werden:
  - `max_connections` soll deutlich größer als `max_used_connections` sein.
  - Das Verhältnis von `threads_created` zu `connections` soll kleiner 1% sein.

  Der aktuelle Zustand kann mit
  `mysql> SHOW GLOBAL STATUS LIKE 'threads%';` erfragt werden.
- `max_allowed_packet`: Dieser Paramter kommt sowohl in der Client, als auch in der Servereinstellung vor. Es sollte an die Größe der Datensätze der Datenbank angepasst sein.

Parameter wie `sort_buffer_size`, `read_buffer_size`, ... werden pro Request vergeben und sind daher deutlich kleiner zu bemessen.

Caches

Die Leistungsfähigkeit eines Datenbanksystems erhöht sich „mit allem", was sich im Speicher vorhalten lässt. Es sollte aber nicht mehr im Speicher vorgehalten werden, als auch wirklich physischer Speicher dem DBMS zur Verfügung steht, weil es sonst zu zeitintensiven Swap-Vorgängen kommt. MySQL hat (u.a.) diese Caches:

- *sort cache*: Dient zum Sortieren von Ergebnissen, wenn also wiederholt Abfragen mit `ORDER BY` auftreten. Mit
  `mysql> SHOW GLOBAL STATUS like 'Sort_merge_passes';` kann abegfragt werden, wieviele Durchläufe eine Sortierung braucht. Erscheint einem dieser Wert zu hoch, kann mit Erhöhung der `sort_buffer_size` versucht werden, diesen Wert zu reduzieren.
- `read_buffer_size`: Dieser *cache* wird für die Optimierung von *full table scans* eingesetzt. Lassen sich diese also nicht verhindern, so kann über diesen Paramter eine Optimierung erreicht werden.
- `table_open_cache` gibt die maximale Anzahl der offen gehaltenen Tabellen an. Die aktuell offenen Tabellen können mit
  `mysql> SHOW OPEN TABLES;` abgefragt werden.

Mit `mysqltuner` existiert ein Werkzeug, das einige dieser Werte automatiert prüft. Profis sehen aber den einen beschränkten Nutzen in diesen Werkzeugen.

Speichermessung

Grundsätzlich gilt für alle Speicherzuweisungen: Sind sie zu klein bemessen, arbeitet die Datenbank ineffizient. Sind sie zu groß bemessen, muss das Betriebssystem Speicher auslagern.

Die Speicherbelegung des Systems wird mit
`free -m` abgefragt.

Die Speicherbelegung eines einzelnen Prozesses kann mit
`cat /proc/<PID>/status` abgefragt werden.

Aufgabe:

- Betrachten Sie Ihre aktuelle Konfigurationsdatei
- Betrachten Sie die aktuell gültigen (Default-) Einstellungen.
- Prüfen Sie, ob das System *swap* verwendet.
- Prüfen Sie, ob MySQL *swap* verwendet.
- Prüfen Sie die oben beschriebenen Einstellungen.
- Rufen Sie
  `mysqltuner --user=root --pass=<password> 2>tuner.err` auf.

6.3 Tabellenwartung

MyISAM-Tabellen

Bei einem Absturz des Servers kann es zu Inkonsistenzen in den Tabellen kommen. Insbesondere bei den nicht-transaktionssicheren MyISAM-Tabellen kann es zu Problemen kommen. Diese können durch einen Tabellen-Check festgestellt werden. Im Client lautet der Befehl dafür:

```
> CHECK TABLE <tabelle> [<umfang>];
```

`<umfang>` kann dabei folgende Werte haben: `QUICK FAST CHANGED MEDIUM EXTENDED`. Defaultwert ist `MEDIUM`[28].

Mit dem Kommandozeilentool können ganze Datenbanksysteme geprüft werden:

```
# mysqlcheck <root-login> --all-databases --medium-check
```

Werden Fehler festgestellt, kann versucht werden, diese mit der Anweisung

```
> REPAIR TABLE <tabelle>;
```

zu reparieren.

[28] Pröll (2015): S. 298.

Bei der Vorstellung der MyISAM-Engine wurde bereits erwähnt, dass diese fragmentieren kann. Die Defragmentierung erfolgt über

```
> OPTIMIZE TABLE <tabelle>;
```

Auf der Kommandozeile lautet dieser Befehl:

```
# mysqlcheck <root-login> --optimize --all-databases
```

Sollte nach einem Crash das DBMS so angeschlagen sein, dass MySQL gar nicht mehr richtig startet, so kann versucht werden, eine DB-Reparatur direkt auf Datei-Ebene durchzuführen. Der Befehl dazu lautet `myisamchk`.[29]

InnoDB-Tabellen

Der `OPTIMIZE`-Befehl kann auch auf InnoDB-Tabellen angewandt werden. Bei diesen Tabellen werden über diesen Befehl die Daten reorganisiert und die Indizes neu berechnet, was der Performance dienen kann.[30]

Sollte nach einem Crash das DBMS so angeschlagen sein, dass MySQL gar nicht mehr richtig startet, so kann versucht werden, so kann versucht werden, MySQL trozt korrupter InnoDB-Tabellen zu starten. Dazu wird der Konfigrtationsparameter `innodb_force_recovery = 1` gesetzt und abschließend versucht, durch einen lesenden Zugriff die Daten zu retten.[31]

Aufgabe:

- Überprüfen Sie die in der Datenbank vorhandenen MyISAM-Tabellen.
- Führen Sie an allen Tabellen die vorgestellten Wartungsarbeiten durch.

7 Abfrageoptimierung

Die Abfrageoptimierung ist *der* Hebel für die Verbesserung der Performance.

7.1 Allgemeines Vorgehen

Zur Performance-Optimierung haben sich best practices gebildet:

- Performance-Optimierung erfolgt reaktiv: Wenn es in einer Anwendung zu Problemen kommt, analysieren und beheben wir diese.
- Bei betrieblichen Anwendungen spielt die Datenbank eine wichtige Rolle.

[29] Pröll (2015): S. 300.
[30] https://dev.mysql.com/doc/refman/8.0/en/optimizing-innodb-storage-layout.html (2.11.2022)
[31] Pröll 2015): S. 301.
[32] MySQL-Vortag beim Münchner Linux-Stammtisch am 20.1.2020

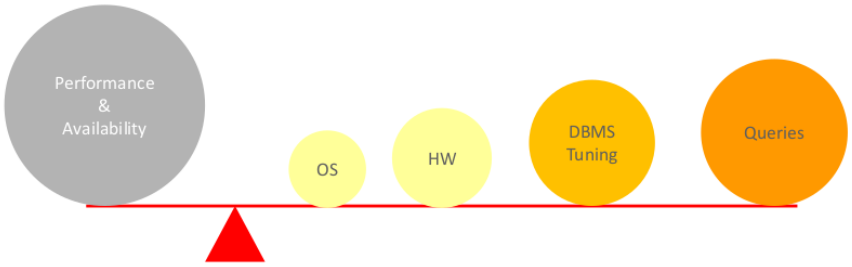

Abbildung 12: Abfrageoptimierung[32]

Vorgehen zur Performance-Optimierung:

- Analyse der kritischen Operationen
- Häufig rentiert sich gleich ein Blick auf die Datenbank.
  - Mit dem Query-Log kann festgestellt werden, ob häufig dieselben Anfragen gesendet werden. In diesem Fall sollte in der Anwendung das Ergebnis vorgehalten werden.

    Um das Logging einzuschalten muss die Variable `general_log` gesetzt werden.
  - Durch Performance-Messungen können langlaufende Anfragen identifiziert und optimiert werde. Hier ist zu beachten, dass sich die Laufzeiten von Testsystem und Produktivsystem aufgrund unterschiedlicher Datenmengen stark unterscheiden können.
- Führen diese Maßnahmen zu keinem eindeutigen Ergebnis, kann das Problem durch komplexe Lastzustände verursacht sein. Diese können nur durch geeignete Tools oder den Einsatz des `performance_schema` analysiert und behoben werden.

7.2 Messungen

Lasttests stellen bei Performance-Analysen eine besondere Herausforderung dar. Die Erzeugung einer Last ist meist mit hohem Aufwand verbunden. Hier hilft das Werkzeug `mysqlslap`. Es ist fähig für die Datenbanken des Systems selbständig SQL-Abfragen zu erzeugen und die parallel auf den Server zu geben. Die Syntax lautet:

```
# mysqlslap <login> --create-schema=test --auto-generate-sql
   --concurrency=15 --number-of-queries=10000
```

In diesem Fall werden 10000 Abfragen über 15 parallele Clients durchgeführt.

Um im laufenden Betrieb beobachten zu können, welche Clients eine hohe Last erzeugen, kann das Werkzeug `mytop` (ggf. nachinstallieren) verwendet werden.

Zur Performance eines Datenbankservers trägt insbesondere der Zugriff auf im RAM gefindliche Daten bei. Nach einem Neustart des Servers sind aber erst mal alle Caches leer. Daher muss ein Server der „warm laufen" bevor sich reproduzierbare Performance-Daten messen lassen.

Soll ein bestimmtes SQL-Statement auf Performanz geprüft werden, bietet uns das Kommando `BENCHMARK`[33] die Möglichkeit dieses zu messen. Da es wenig aussagekräftig ist, dasselbe Statement ständig zu wiederholen, sollte das Kommando mit der `RAND()`-Funktion variiert werden.

Aufgaben:

- Starten Sie `mytop -uroot -p<password>`
- Starten Sie einen Lasttest für 5 Clients und 1000 Abfragen. Beobachten Sie das Geschehen im `mytop`
- Messen Sie die Ausführungszeit:

```
 > SELECT BENCHMARK (10,
-> ( SELECT max(flug_id) FROM flug f, flughafen h
->        WHERE f.von = h.flughafen_id AND f.abflug BETWEEN
->          DATE(DATE_ADD(NOW(),
->          INTERVAL FLOOR((RAND()*30)) DAY)) AND
->          DATE(DATE_ADD(NOW(),
->          INTERVAL FLOOR((RAND()*60)) DAY))LIMIT 1));
```

- Führen Sie das Statement auch für größere Ausführungszahlen durch.

7.3 Performance-Schema[34]

Mit MySQL 5.7 wurde das Performance-Schema weiter ausgebaut. Im MySQL-Code sind hart Eventgeber implementiert, deren Ereignisse in temporären Tabellen gehalten werden.

Folgende Tabellen seien an dieser Stelle exemplarisch vorgestellt:

- Tabllen, die auf `_current` enden: Sie enthalten aktuelle Messergebisse. Beispiel: `events_statements_curre` enthält die letzten ausgeführten Statenments.
- Tabllen, die auf `_instances` enden: Sie enthalten Informationen zu Dateien, Sockets, Locks, ... Beispiel: `file_instances`
- Tabellen, die `_summary` enthalten: Hier sind Daten bereits aggrgiert. Z.B. `events_waits_summary_by_insta`

Die Tabellen im Performance-Schema können mit
`TRUNCATE <TAB>;` rückgesetzt werden.

Interessant sind die Analysen auf den Daten des Performance-Schemas. Da die Inormationen aus verschiedenen Tabellen zusammengetragen werden müssen, sind das längliche SQL-Statements:

- Threads und ihre Ausfürungszeiten[35]

[33] Pröll (2015): S. 316.
[34] Pröll (2015): S. 341ff.
[35] Pröll (2015): S. 355., `src/process_details.sql`

```
SELECT t.thread_id AS THREAD_ID,
t.processlist_id AS CONN_id,
t.processlist_db AS DB,
t.processlist_command AS Operation,
t.processlist_state AS Aktino,
t.processlist_time AS Aktionsdauer,
IF(s.timer_wait IS NOT NULL, (s.sql_text), 'NULL (kein SQL-Statement)') AS 'Let
(s.lock_time/POW(10,12)) AS 'Lock-Wartezeit (s)',
s.rows_examined AS 'Zeilen gelesen',
s.rows_sent AS 'Zeilen zur"uckgegeben',
s.rows_affected AS 'Zeilen betroffen',
IF(s.no_index_used > 0, 'Nein', 'Ja') AS 'Index verwendet',
e.event_name AS 'Instrumentenname',
IF(e.timer_wait IS NULL AND e.event_name IS NOT NULL, 'Event wartet noch ...',
  e.timer_wait/POW(10,12)) AS 'Eventdauer'
FROM threads AS t
  LEFT JOIN events_waits_current AS e USING (thread_id)
  LEFT JOIN events_statements_current as s USING (thread_id)
ORDER BY t.processlist_time DESC, Eventdauer DESC \G;
```

- Abfragen ohne Verwendung eines Index[36]

```
SELECT DIGEST_TEXT AS 'Query',
COUNT_STAR AS 'Anzahl Abfragen',
SUM_NO_INDEX_USED AS 'Ohne Index',
SUM_NO_GOOD_INDEX_USED AS 'Suboptimaler Idx'
FROM events_statements_summary_by_digest
WHERE (SUM_NO_INDEX_USED > 0 OR
  SUM_NO_GOOD_INDEX_USED > 0) AND
  (SCHEMA_NAME IS NOT NULL)
ORDER BY COUNT_STAR DESC LIMIT 15\G;
```

- Statistiken über das Sortieren[37]

```
SELECT DIGEST_TEXT AS 'Query',
COUNT_STAR AS 'Anzahl Abfragen',
SUM_SORT_MERGE_PASSES AS 'Sortierdurchl"aufe',
SUM_SORT_SCAN AS 'Full Table Scans',
SUM_SORT_RANGE AS 'Range Sorts',
SUM_SORT_ROWS AS 'Sortierte Zeilen',
SUM_ROWS_SENT AS 'Zeilen zur"uckgegeben',
SUM_ROWS_EXAMINED AS 'Untersuchte Zeilen'
FROM events_statements_summary_by_digest
WHERE SUM_SORT_ROWS > 0
ORDER BY SUM_SORT_MERGE_PASSES DESC, SUM_SORT_SCAN\G;
```

7.4 Profiling

Profiling ist die detailierte Betrachtung der Systemresourcen. Mit Version 5.7 wird diese Funktionalität mit in das `performance_schema` verschoben.

[36] Pröll (2015): S. 356., `src/query_no_idx.sql`
[37] Pröll (2015): S. 357., `src/query_sort.sql`

Das Profiling wird mit `SET profiling = 1` eingeschaltet. Danach werden für die ausgeführten Abfragen Performance-Daten mitgeschrieben. Diese können mit

```
> SHOW PROFILES;
```

abgefragt werden. Das Profil einer speziellen Abfrage lässt sich mit

```
> SHOW PROFILE FOR QUERY <n>;
```

abfragen, wobei `<n>` die `Query_id` aus der Abfrage zuvor ist.

Aufgabe:

- Schalten Sie das Profiling ein.
- Führen Sie das SQL des `BENCHMARK`-Befehls fünf mal aus.
- Führen Sie das SQL des `BENCHMARK`-Befehls fünf mal identisch aus (d.h. ohne `RAND`).
- Führen Sie `SELECT COUNT(*) FROM buchung;` fünf mal aus.
- Analysieren Sie die Profile.

7.5 Abfrageoptimierung

Ein einfacher, erster Ansatz zur Perfomance-Messung und -Optimiertung funktioniert so:

- Identifikation kritischer Anfragen im Query Log

  MySQL unterstützt die Suche zusätzlich mit dem `slow_query_log`. Dieses wird in der Datei `my.cnf` mit folgenden Einträgen eingeschaltet[38]:

  ```
  slow_query_log = ON
  slow_query_log_file = /var/log/mysql_slow.log
  long_query_time = 1.5   # in Sekunden
  ```

  Diese Einstellung wird erst in einer neuen Session aktiv.

- Da oftmals *full table scans* für ein schlechtes Laufzeitverhalten verantwortlich sind, kann explizit nach diesen gesucht werden. Der Konfigurationseintrag dafür lautet:

  ```
  log_queries_not_using_indexes = ON
  ```

  Diese Queries werden gleichfalls im `slow_query_log_file` festgehalten.

- Messung der Laufzeit im Kommandozeilen-Client
- Analyse des DB-Zugriffs mit `EXPLAIN`
- Optimierung des Zugriffs
  - Alternative Formulierung
  - Indizes
    Anlegen: `mysql> CREATE INDEX <name> ON <table> (<key>)`

[38] Pröll (2015): S. 386.

Aufgabe:

- Schalten Sie das Query-Log, `slow_query_log` und das Logging von *full table scans* ein.
- Verfolgen Sie die Logs im Terminal (`$ tail -f <log-datei>`).
- Führen Sie einige Abfragen im Client durch, auch über mehrere Tabellen.
- Führen Sie auf eine interessante Abfrage ein `EXPLAIN SELECT ...` aus.
- Wenn bei einem Schritt `key=NULL` steht, erfolgt hier ein *full table scan*, was bei großen Tabellen viel Zeit kostet. Führen Sie für diesen Zugriff einen Index ein und führen Sie das `EXPLAIN` erneut aus.

Ist man sich nicht mehr klar, welche Indizes für eine Tabelle existieren, kann dies mit
`> SHOW INDEX FROM <tabelle>;`
nachgeschaut werden.

Indizes über mehrere Spalten können nur in der angegeben Reihenfolge ausgewertet werden. Ist für die Tabelle `flug` ein Index `ON (nach, flugzeug_id)` definiert, kann dieser von Abfragen verwendet werden, die nur nach `nach` einschränken, oder nach `nach` und `flugzeug_id`, nicht aber in Abfragen die nur nach `flugzeug_id` einschränken.

Bei größeren Abfragen wird die Ausgabe des `EXPLAIN` schnell unübersichtlich. Um die Ausgabe zu visualisieren eigent sich der Befehl `pt-visual-explain`. Auf einem UNIX/Linux-System kann die Ausgabe des `EXPLAIN` mit dem Pipe-Symbol dem `pt-visual-explain` übergeben werden:
`$ mysql <login> <db> -e "EXPLAIN ...;" | pt-visual-explain`

Aufgabe:

- Suchen Sie sich ein längeres SQL-Statement aus Abschnitt 4.2 aus und führen Sie ein `EXPLAIN` durch.
- Installieren Sie `pt-visual-explain` und visualisieren Sie das `EXPLAIN`.

Allgemeine Tipps:

- `JOIN` ist einer Subquery vorzuziehen[39].
- Verzicht auf `ORDER BY`[40]
- Vermeidung von `LIKE`[41]
- Vermeidung von Berechnungen in der `WHERE`-Clause[42]

[39] Pröll (2015): S. 434.
[40] Pröll (2015): S. 438.
[41] Pröll (2015): S. 442.
[42] Pröll (2015): S. 444.

Darüber hinaus ist es möglich dem Optimizer Hinweise zur Verwendung von Indizes zu geben. Dazu kann in einer Abfrage hinter den Tabellennamen nach einem der folgenden Schlüsselwörter ein Index angegeben werden[43]:

- `USE`: Empfehlung den genannten Index zu verwenden
- `FORCE`: Dringende Empfehlung den genannten Index zu verwenden
- `IGNORE`: Empfehlung den genannten Index nicht zu verwenden

7.6 Partitionierung

Werden Tabellen in einem System sehr groß, so können diese in Teiltabellen aufgeteilt werden. Man unterscheidet hierbei zwischen vertikaler und horizontaler Partitionierung (s. Abb. 13[44])

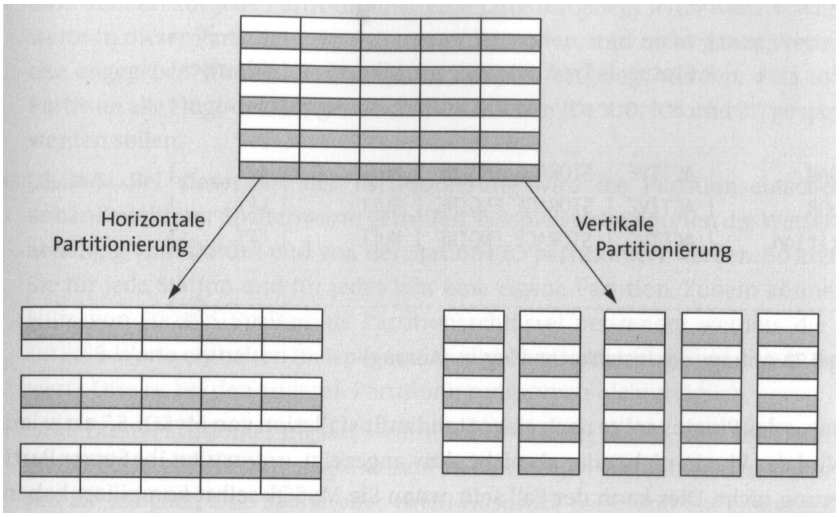

Abbildung 13: Horizontale und vertikale Partitionierung

Welche Art der Partitionierung in einem konkreten Fall geeignet ist, hängt von den fachlichen Anforderungen der Anwendungen ab. Es muss vermieden werden, dass die Tabellen häufig wieder zu einer großen Tabelle zusammengefasst werden müssen. Ob eine Trennung nach Attributgruppen (vertikal) oder nach Gruppen von Datensätzen erfolgt, ist fachlich zu entscheiden.

8 Quellen

Böttcher	Ulrike Böttcher, Peter Teich: SQL. Grundlagen und Datenbankdesign. Herdt 2004[2].
Fuchs	Elmar Fuchs: SQL. Grundlagen und Datenbankdesign. Herdt 2021[1].
MySQL5.7	Online-Dokumentation: https://dev.mysql.com/doc/refman/5.7/en/
Pröll (2015)	Stefan Pröll et al.: MySQL. Das umfassende Handbuch. Reinwerk 2015[3].

[43] Pröll (2015): S. 453.
[44] Pröll (2015): S. 457.