DB: Grundlagen des Datenbankdesigns

Ziele

- Die wichtigsten Tabellentypen von MySQL und ihre Eigenschaften verstehen und einsetzen können.
- Die verschiedensten MySQL-Datentypen kennen, ihre Eigenschaften erkennen und diese einsetzen können.
- Mit dem Aufzähltyp `ENUM` umgehen können.
- Die Optionen `DEFAULT`, `UNIQUE` und `NOT NULL` einsetzen können.
- Die Wichtigkeit von Primärschlüsseln erkennen und ihre Definition beherrschen.
- Die Funktionsweise von Indexen verstehen, diese definieren können.
- Mit Volltextindexen umgehen können.
- Die wichtigen Konzepte des Fremdschlüssels und der referentiellen Integrität verstehen und einsetzen können.
- 1:n- und 1:1-Beziehungen definieren können.
- Sichten definieren und einsetzen können.

MySQL-Tabellentypen im Vergleich

MyISAM	InnoDB
☺ Standardtabellenformat	☺ Transaktionsmanagement
☺ Ausgereift und stabil	☺ Sicherer
☹ Nur *Table Locking*	☺ Ideal wenn viele Anwender gleichzeitig Änderungen machen
☺ Geringer Platzbedarf	☺ *Row Level Locking* mit automatischer Erkennung von *Deadlocks*
☺ *Volltextindexe*[1]	☺ *Fremdschlüssel*
☺ *GIS-Daten* speicherbar	☺ Schnelles *Crash Recovery*
	☹ Großer Platzbedarf
	☹ Lizenzkosten größer für kommerzielles Produkt

In einer Datenbank können Tabellen unterschiedlichen Typs angelegt werden.

HEAP- und temporäre Tabellen

Nur im Arbeitsspeicher abgelegt. Gehen verloren, wenn Server beendet bzw. Verbindung getrennt wird. Gut für kleine Datenmengen mit äußerst schnellem Zugriff.

[1] Funktioniert ab Version 5.6 auch bei InnoDB

MySQL-Datentypen

Integerzahlen (xxxINT)

TINYINT[(m)]	1 Byte -128 bis +127
SMALLINT[(m)]	2 Byte -32.768 bis +32.767
MEDIUMINT[(m)]	3 Byte -8.388.608 bis +8.388.607
INT[(m)] oder **INTEGER**[(m)]	4 Byte -2.147.483.648 bis +2.147.483.647
BIGINT[(m)]	8 Byte $\pm 9{,}22 \cdot 10^{18}$

SQL-Standard

- xxx**INT**(m) m legt die maximale Ausgabebreite fest, beeinflusst aber nicht den für den Datentyp zulässigen Zahlenbereich.
- xxx**INT UNSIGNED** verändert den Zahlenbereich z. B. von -32.768 bis +32.767 auf 0 bis 65.535
- Datentyp **BOOL** ist Synonym für **TINYINT**.

AUTO_INCREMENT bei Integerzahlen

- Beim Einfügen eines neuen Datensatzes erhält Datenfeld automatisch den um 1 erhöhten Wert des derzeit enthaltenen größten Wertes.
- Nur zulässig wenn gleichzeitig **NOT NULL** und **PRIMARY KEY** oder **UNIQUE** definiert wurde.
- Wenn beim Einfügen spezifischer Wert für Datenfeld übergeben wird, dann wird dieser eingetragen.

Dezimalzahlen

FLOAT[(m,d)]	4 Byte 8 Stellen Genauigkeit
DOUBLE[(m,d)]	8 Byte 16 Stellen Genauigkeit
DECIMAL[(m[,d])]	Festkommadarstellung m <= 65, d <= 30

m Gesamtstellenanzahl, d Anzahl der Stellen nach Dezimalpunkt
bewirkt eine Rundung!!
z. B. **FLOAT**(5,3) speichert 0.1235 als 0.124 und 1000 als 99.999

Datum und Uhrzeit

DATE	3 Byte im Format 2006-12-31 Bereich 1000-01-01 bis 9999-12-31
TIME	3 Byte im Format 23:59:59 Bereich ±838:59:59
DATETIME	5 Byte im Format 2006-12-31 23:59:59 Bereich 1000-01-01 00:00:00 bis 9999-12-31 23:59:59'
TIMESTAMP	4 Byte Bereich 1970 bis 2038
YEAR	1 Byte Bereich 1900 bis 2155

Erstes **TIMESTAMP**-Datenfeld wird bei jeder Änderung des Datensatzes automatisch aktualisiert und bei neuem Datensatz automatisch gesetzt.

Zeichenketten

CHAR(n)	Zeichenkette mit fixer Länge, n maximal 255
VARCHAR(n)	Zeichenkette mit variabler Länge, n maximal 65.535
TINYTEXT	Zeichenkette mit variabler Länge, maximal 255
TEXT	Variable Länge, maximal 65.535
MEDIUMTEXT	Variable Länge, maximal 16.777.215
LONGTEXT	Variable Länge, maximal 4.294.967.295

- Länge hängt vom verwendeten Zeichensatz ab. Es existieren Zeichensätze die mehr als ein Byte zur Darstellung eines Zeichens verwenden.
- MySQL ändert in manchen Fällen die Datenfelddefinition ab, weil andere für MySQL effizienter ist (*Silent Column Changes*).

BLOBs (Binary Large Objects)

TINYBLOB	Variabler Länge, maximal 255 Byte
BLOB	Variable Länge, maximal 65.535 Byte
MEDIUMBLOB	Variable Länge, maximal 16.777.215 Byte
LONGBLOB	Variable Länge, maximal 4.294.967.295 Byte

Optionen und Attribute

NOT NULL	Datenfeld muss Wert haben
DEFAULT	Standardwert der beim Einfügen eines neuen Datensatzes in das leere Datenfeld eingetragen wird

Spaltentyp ENUM

```
CREATE TABLE personen(
  pnummer INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
  pname VARCHAR(50) NOT NULL,
  pbeziehungsstatus ENUM ("Single", "Verlobt",
     "Verheiratet", "Geschieden",
     "Es ist kompliziert") NOT NULL);
INSERT INTO personen(pname)
  VALUES ("Sepp");
INSERT INTO personen(pname, pbeziehungsstatus)
  VALUES ("Elsa", "Getrennt");
SELECT *
  FROM personen
  WHERE pbeziehungsstatus IN ("Single", "");
oder
SELECT *
  FROM personen
  WHERE pbeziehungsstatus IN (1, 0);
```

Ergebnis

personen	(pnummer,	pname,	pbeziehungsstatus)
	1	Sepp	Single
	2	Elsa	""

- 65.535 unterschiedliche Werte möglich.
- Bei ungültigem Wert wird "" eingetragen.
- Ist enum-Datenfeld auf **NOT NULL** gesetzt, wird automatisch erster Wert der Aufzählung in Datenfeld eingetragen.

Wert	Index
NULL	NULL
""	0
"Single"	1
"Verlobt"	2
"Verheiratet"	3
"Geschieden"	4
"Es ist kompliziert"	5

- Jeder Werte der Aufzählung hat eigenen Index mit dem gerechnet werden kann, "" hat Index 0.

```
DATE personen SET pbeziehungsstatus = 3
  WHERE pbeziehungsstatus = 1;
```

Primärschlüssel

ist ein Datenfeld oder mehrere Datenfelder einer Tabelle, welche jeden Datensatz eindeutig identifizieren.

z. B. knummer 1024, anummer IS-03-8639,
bisbn 978-3-8273-2774-1, anummer AG670XL,
pnummer WMKMXR87T01B220U,
{mnachname, mvorname, mgebdatum} {"Resch", "Kurs", "1987-01-31"}

```
CREATE TABLE kunden(
   knummer INTEGER NOT NULL AUTO_INCREMENT,
   …
   PRIMARY KEY (knummer));
CREATE TABLE pruefungskandidaten(
   mnachname VARCHAR(50) NOT NULL,
   mvorname VARCHAR(50) NOT NULL,
   mgebdatum DATE NOT NULL,
   …
   PRIMARY KEY (mnachname, mvorname, mgebdatum));
```

- Primärschlüssel muss *zeitinvariant* sein.
- Primärschlüsselinhalte sollten sich für Datensatz nicht mehr ändern.
- Primärschlüssel sollte *möglichst kompakt* aus wenigen Datenfeldern und wenn möglich nur aus einer Zahl bestehen.
- Primärschlüsseldatenfelder müssen **NOT NULL** sein.
- Jede Tabelle sollte einen Primärschlüssel besitzen.
- Anhand von Primärschlüsseln werden Beziehungen zw. Tabellen dargestellt (siehe hinten).
- Für jeden Primärschlüssel wird automatisch ein *Index* erstellt (siehe hinten) ➔ dadurch schnelle Suche nach Primärschlüsselfeldern möglich.

Indexe oder Sekundärschlüssel

ist eine zusätzliche verborgene Tabelle mit sortierten Querverweisen auf die Datensätze der Tabelle ähnlich einem *Stichwortverzeichnis* eines Buches.

```
CREATE TABLE messwerte(
   mnummer BIGINT NOT NULL AUTO_INCREMENT,
   snummer INTEGER NOT NULL,                    für Stationsnummer
   mdatum TIMESTAMP NOT NULL,
   mwert DECIMAL(5,2) NOT NULL,
   KEY (snummer, mdatum),
   PRIMARY KEY (mnummer));

SELECT *
   FROM messwerte                    wird schnell abgearbeitet
   WHERE snummer = 1093;

SELECT *
   FROM messwerte                    kein Geschwindigkeitsvorteil
   WHERE mdatum <= now();
```

- Index nutzlos bei

```
WHERE snummer <> 1093
WHERE mnachname LIKE "%u%"
```

- Indexe nur bei vielen Datensätzen und bei Definition der referentiellen Integrität interessant.
- Beschleunigen das Suchen, Änderungen werden langsam.
- Benötigen zusätzlichen Speicherplatz auf Datenträger.
- Auch mehrere Indexe pro Tabelle möglich.

```
ALTER TABLE messwerte
   ADD UNIQUE KEY (mdatum);
```

- Kein Datensatz darf im mit **UNIQUE** indizierten Datenfeld denselben Wert wie ein anderer Datensatz haben.
- Zur Realisierung von *1:1-Beziehungen* verwendet (siehe hinten).

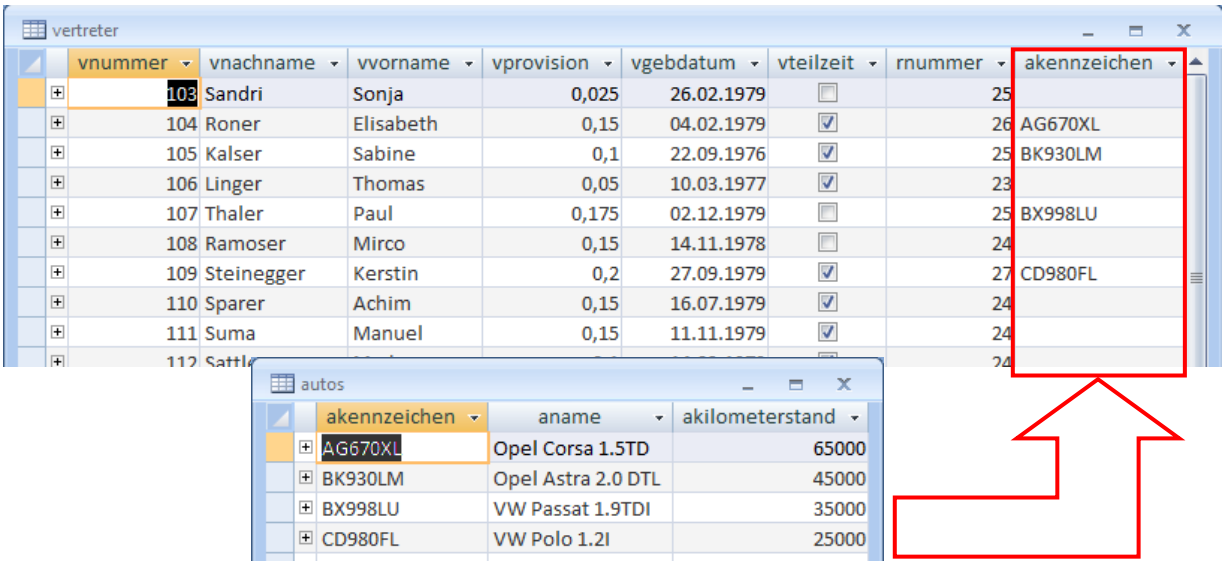
Volltextindexe

```
CREATE TABLE forenbeitraege(
   bnummer INTEGER NOT NULL AUTO_INCREMENT,
   btitel TINYTEXT NOT NULL,
   btext TEXT,
   FULLTEXT KEY (btitel, btext),
   PRIMARY KEY (bnummer))
   ENGINE=MyISAM bzw. InnoDB;

SELECT bnummer, btitel, btext, MATCH (btitel, btext)
   AGAINST ("Sichten MySQL Views")
   FROM forenbeitraege
   ORDER BY 4 DESC
   LIMIT 5;
```

- Es wird ein Index aller Wörter erstellt, welche im Text vorkommen.
- Wörter müssen länger als 3 Buchstaben sein und Tabelle mind. 3 Datensätze enthalten. Je mehr Datensätze umso besser.
- Reihenfolge und Groß-/Kleinschreibung irrelevant.
- MATCH errechnet einen Zahlenwert für die Relevanz, liefert 0 falls keines der Wörter gefunden oder wenn Suchbegriffe in sehr vielen Datensätzen auftreten.

Fremdschlüssel (engl. Foreign Keys)

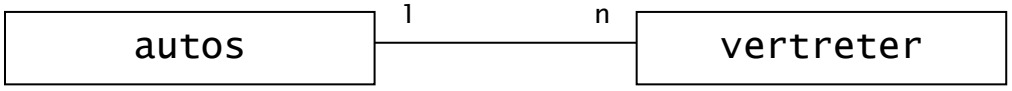


vertreter

vnummer	vnachname	vvorname	vprovision	vgebdatum	vteilzeit	rnummer	akennzeichen
103	Sandri	Sonja	0,025	26.02.1979	☐	25	
104	Roner	Elisabeth	0,15	04.02.1979	☑	26	AG670XL
105	Kalser	Sabine	0,1	22.09.1976	☑	25	BK930LM
106	Linger	Thomas	0,05	10.03.1977	☑	23	
107	Thaler	Paul	0,175	02.12.1979	☐	25	BX998LU
108	Ramoser	Mirco	0,15	14.11.1978	☐	24	
109	Steinegger	Kerstin	0,2	27.09.1979	☑	27	CD980FL
110	Sparer	Achim	0,15	16.07.1979	☑	24	
111	Suma	Manuel	0,15	11.11.1979	☑	24	
112	Sattl…					24	

autos

akennzeichen	aname	akilometerstand
AG670XL	Opel Corsa 1.5TD	65000
BK930LM	Opel Astra 2.0 DTL	45000
BX998LU	VW Passat 1.9TDI	35000
CD980FL	VW Polo 1.2I	25000



Mastertabelle Detailtabelle

1:n-Beziehung mit *referentieller Integrität*

- Ein Auto kann mehreren Vertretern zugewiesen werden.
- Ein Vertreter hat immer nur ein Auto.
- Ein Vertreter kann auch kein Auto haben.
- In `vertreter` nur Autos eintragen die in `autos` vorhanden sind.
- Was passiert, wenn Auto in `autos` gelöscht wird in `vertreter`?
- Was passiert, wenn Auto in `autos` sein Kennzeichen ändert?

```
CREATE TABLE vertreter(
   vnummer INTEGER NOT NULL,
   …
   akennzeichen VARCHAR(7),
   KEY (akennzeichen),
   FOREIGN KEY (akennzeichen)
      REFERENCES autos(akennzeichen)
      ON DELETE SET NULL ON UPDATE CASCADE,
   PRIMARY KEY (vnummer)) ENGINE=InnoDB;
```

FRAGE: Was würde passieren, wenn umgekehrt in `autos` die `vnummer` als Fremdschlüssel definiert würde?

- **ON DELETE/UPDATE SET NULL**, **CASCADE** oder **RESTRICT** Standardvorgabe ist **RESTRICT**.
- Nur auf InnoDB-Tabellentypen definierbar.
- Datenfeldtyp von Fremdschlüssel und verbundenem Primärschlüssel müssen übereinstimmen.
- Fremdschlüssel kann auch aus mehreren Datenfeldern bestehen.
- Fremdschlüssel kann auch **NULL** sein.
- Fremdschlüssel kann auch auf dieselbe Tabelle verweisen ➔ *Hierarchische Strukturen*
- **ACHTUNG**: *Zirkuläre Referenzen*

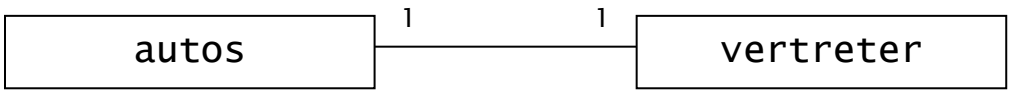
```
SELECT v.vnummer, v.vnachname, a.akennzeichen, a.aname,
   a.akilometerstand
   FROM vertreter v, autos a
   WHERE v.akennzeichen = a.akennzeichen;
```

vertreterautos

vnummer	vnachname	vvorname	akennzeichen	aname	akilometerstand
104	Roner	Elisabeth	AG670XL	Opel Corsa 1.5TD	65000
105	Kalser	Sabine	BK930LM	Opel Astra 2.0 DTL	45000
109	Steinegger	Kerstin	CD980FL	VW Polo 1.2I	25000
107	Thaler	Paul	BX998LU	VW Passat 1.9TDI	35000

1:1-Beziehung mit *referentieller Integrität*

- Ein Auto kann nur einem Vertreter zugewiesen werden.
- Ein Vertreter hat immer nur ein Auto.
- Einem Vertreter kann auch kein Auto zugewiesen werden.



Mastertabelle Detailtabelle

```
ALTER TABLE vertreter
   ADD UNIQUE (akennzeichen);
```

Setze Fremdschlüsseldatenfeld auf **UNIQUE**.

Sichten (engl. Views)

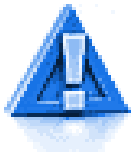
ermöglichen spezielle Darstellungen von Tabellen zu definieren.

```
CREATE VIEW vertreterautos AS
   SELECT v.vnummer, v.vnachname, a.akennzeichen,
      a.aname, a.akilometerstand
      FROM vertreter v, autos a
      WHERE v.akennzeichen = a.akennzeichen;

SELECT *
   FROM vertreterautos
   WHERE akilometerstand > 250000
   ORDER BY akilometerstand DESC;
```

- Views verhalten sich wie Tabellen (**INSERT**, **UPDATE**, **DELETE** unter bestimmten Umständen möglich).
- Hilfsmittel zum Datenschutz.
- Komfortable, *verborgene Joins*.

Programmierregeln



- Datenbank- und Tabellen- und Datenfeldnamen in Kleinbuchstaben.
- Tabellennamen im Plural benennen (z. B. kunden).
- Einheitliche Namensvergabe bei Datenfeldnamen anwenden (z. B. knummer, kname, kstrassenr, kplzort, kemail, …).