

SQL: DDL
Data Definition Language



Was ist die DDL?

- Die Data Definition Language (DDL) ist eine Datenbanksprache, die verwendet wird, um Datenstrukturen und verwandte Elemente zu beschreiben, zu ändern oder zu entfernen.
- Die wichtigsten Befehle der SQL DDL sind
 - CREATE TABLE, RENAME TABLE und DROP TABLE
 - ALTER TABLE
 - CREATE INDEX und DROP INDEX
 - CREATE VIEW und DROP VIEW

DDL – CREATE TABLE

```
CREATE TABLE `rechnung` (  
  `rechnungsnummer` bigint(20) NOT NULL,  
  `datum` date NOT NULL,  
  `kundennummer` bigint(20) NOT NULL,  
  `rechnungsstatus` tinyint(4) UNSIGNED NOT NULL,  
  `zahlungsart` tinyint(4) UNSIGNED DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `zahlungsart` (  
  `id` tinyint(4) UNSIGNED NOT NULL,  
  `name` varchar(30) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `kunde` (  
  `kundennummer` bigint(20) NOT NULL,  
  `id_adresse` bigint(20) NOT NULL,  
  `kreditkarteninformation` bigint(20) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `kreditkarteninformation` (  
  `id` bigint(20) NOT NULL,  
  `kundennummer` bigint(20) NOT NULL,  
  `kreditkartennummer` bigint(20) NOT NULL,  
  `monat` tinyint(3) UNSIGNED NOT NULL,  
  `jahr` tinyint(3) UNSIGNED NOT NULL,  
  `typ` tinyint(4) UNSIGNED NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Was ist ein Index?

- Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.
- Ein Index besteht aus einer Ansammlung von Zeigern (Verweisen), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren.
 - Wird eine indizierte Spalte als Suchkriterium herangezogen, so sucht das DBMS die gewünschten Datensätze anhand dieser Zeiger.
 - In der Regel finden hier B+ - Bäume Anwendung:
 - Bitte schauen Sie sich für die Klausur die Datenstrukturen B - Baum und B+ - Baum an!
 - Ohne Index müsste die Spalte sequenziell durchsucht werden, während eine Suche mit Hilfe des Baums nur logarithmische Komplexität hat.

Was ist ein Index?

- Meistens wird eine einzelne Spalte indiziert, doch auch zusammengesetzte Indizes sind in den meisten Datenbanksystemen möglich.
- Auf Spalten, die Primärschlüssel enthalten (SQL-Klausel PRIMARY KEY im Befehl CREATE TABLE), wird automatisch ein Index gelegt.
- Häufig reicht die Sortierung der Daten nach einem Primärindex nicht aus, so dass weitere Indizes erforderlich sind.
 - Wird nun für diese weiteren Erschließungen ein Gesamtinhaltsverzeichnis aufgebaut, entsteht ein Sekundärindex.

Index in SQL

```
CREATE INDEX Indexname ON Tabellename (Spaltenname (n) )
```

- Keiner der verschiedenen SQL-Standards definiert Befehle für Indizes.
- Die Befehle zum Anlegen und Entfernen von Indizes sind daher immer datenbank-spezifisch.
- Allerdings haben sich die Befehle **CREATE INDEX** und **DROP INDEX** weitestgehend durchgesetzt.
 - PhpMyAdmin verwendet beim DB-Export **KEY** statt **INDEX**

DDL – Primärschlüssel & Indizes/Keys

```
ALTER TABLE `rechnung`  
  ADD PRIMARY KEY (`rechnungsnummer`),  
  ADD KEY `kundennummer` (`kundennummer`),  
  ADD KEY `rechnungsstatus` (`rechnungsstatus`),  
  ADD KEY `zahlungsart` (`zahlungsart`);
```

```
ALTER TABLE `zahlungsart`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `name` (`name`);
```

```
ALTER TABLE `kunde`  
  ADD PRIMARY KEY (`kundennummer`),  
  ADD UNIQUE KEY `id_adresse` (`id_adresse`),  
  ADD UNIQUE KEY `kreditkarteninformation` (`kreditkarteninformation`);
```

```
ALTER TABLE `kreditkarteninformation`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `kundennummer` (`kundennummer`),  
  ADD UNIQUE KEY `kreditkartennummer` (`kreditkartennummer`),  
  ADD KEY `typ` (`typ`);
```

DDL – DROP <TABLE / INDEX / VIEW>

```
DROP TABLE IF EXISTS 'kunde'
```

- Allgemein formuliert:
 - `DROP TABLE Relation`
 - `DROP INDEX Index-Name`
 - `DROP VIEW Sicht`

MyISAM vs. InnoDB

- MyISAM (My Indexed Sequential Access Method) ist eine Storage-Engine des Datenbankverwaltungssystems MySQL.
- Ab Version 5.5 wurde sie durch InnoDB als Standard-Storage-Engine abgelöst.
- MyISAM zeichnet sich durch hohe Effizienz im Vergleich zu anderen von MySQL unterstützten Tabellentypen aus und unterstützt eine leistungsfähige Volltextsuche.
- MyISAM ist daneben für Tabellen empfehlenswert, die deutlich häufiger gelesen werden, als in sie geschrieben wird.
- MyISAM unterstützt keine Transaktionen, so dass im Fehlerfall inkonsistente Daten in der Datenbank zurückbleiben können, falls von mehreren zusammengehörigen Queries einige bereits ausgeführt wurden und andere nicht.
- Auch bietet MyISAM keine referenzielle Integrität.

MyISAM vs. InnoDB

- InnoDB ist ein freies Speichersubsystem für das Datenbank-managementsystem MySQL.
- Sein Hauptvorteil gegenüber anderen Speichersubsystemen für MySQL ist, dass Transaktionssicherheit und referenzielle Integrität über Fremdschlüssel gewährleistet werden.
- Innobase Oy, der Hersteller von InnoDB, wurde im Oktober 2005 von Oracle Systems übernommen.

Charset und Collate

- Wichtig ist zum einen, dass die Codierung der einzelnen String-Typ-Felder
 - CHAR,
 - VARCHAR,
 - TEXT, ENUM,
 - SET

richtig eingestellt ist: Charset!

- Zum anderen sollte nach jedem Verbindungsaufbau dem MySQL-Server mitgeteilt werden, welche Codierung für die zu sendenden und empfangenen Daten verwendet werden soll:
- Diese Einstellungen sorgen auf jeden Fall für Klarheit und einen reibungslosen Datenaustausch, egal was sonst noch für Konfigurationswerte auf dem MySQL-Server eingestellt sind.
- Die Codierung der Felder legen Sie beim Erstellen der Tabelle fest.
- Sie kann auch nachträglich geändert werden, wobei die bereits enthaltenen Daten umcodiert werden.
- Voraussetzung für ein reibungsloses Umcodieren ist natürlich, dass die Daten in den Felder der bisher konfigurierten Codierung entsprechen.

Charset und Collate

- Neben dem "Character Set" gibt es auch noch den Begriff "Collation", der oft in unmittelbarer Nähe zu "Character Set" zu finden ist.
- Collation hat dem Zweck, Regeln für das Vergleichen von Zeichen (beispielsweise zum Sortieren) zu definieren.
- Für Probleme mit der Codierung kann man den Collation-Wert unberücksichtigt lassen.
- Übrigens ist der Begriff Character Set (Zeichensatz) im Prinzip falsch, es müsste eigentlich Character Encoding (Zeichencodierung) heißen.

Charset und Collate: Beispiel der nachträglichen Anpassung

```
ALTER TABLE Tabellename  
CONVERT TO CHARACTER SET utf8  
COLLATE utf8_general_ci
```

- Die Sortierung utf8-general-ci ist schneller, aber sortiert die Sonderzeichen nicht ganz präzise nach dem Duden.
- Die Sortierung utf8_unicode_ci ist exakter, aber leider auch langsamer.
- Das ci steht bei der Sortierung übrigens für case insensitive und bedeutet, dass Gross- und Kleinschreibung nicht unterschieden werden.

Was ist eine View?

- Eine Sicht / View ist eine virtuelle Relation bzw. virtuelle Tabelle in einem Datenbanksystem.
- Diese logische Relation wird über eine im DBMS gespeicherte Abfrage definiert.
- Der Datenbankbenutzer kann eine Sicht wie eine normale Tabelle abfragen.
- Wann immer eine Abfrage diese Sicht benutzt, wird diese zuvor durch das DBMS berechnet.
- Eine Sicht stellt im Wesentlichen einen Alias für eine Abfrage dar.

Erzeugen einer View

```
SELECT *  
FROM (  
    SELECT v.kaeufer, v.verkaeufer FROM produkte p, verkaeufe v  
    WHERE p.produkt_id = v.produkt_id AND p.produkt = "Software";  
) AS SoftwareVerkaeufe;
```

...als Alias erzeugt das gleiche Ergebnis wie

```
CREATE VIEW SoftwareVerkaeufe AS  
SELECT v.kaeufer, v.verkaeufer FROM produkte p, verkaeufe v  
WHERE p.produkt_id = v.produkt_id AND p.produkt = "Software";
```

- Eine nachfolgende Abfrage **SELECT verkaeufer FROM SoftwareVerkaeufe** würde sich auf das Ergebnis dieser Sicht beziehen und würde nur die Verkäufer auflisten, die Software verkauft haben.

Der Sinn von Views

- Die Aufgabe einer Sicht ist es, den Zugriff auf das Datenbankschema zu vereinfachen.
 - Normalisierte Datenbankschemata verteilen Daten auf zahlreiche Tabellen mit komplexen Abhängigkeiten.
 - Dies führt zu aufwändigen SQL-Abfragen.
 - Außerdem wird ein hohes Maß an Wissen über das Schema vorausgesetzt, um solche Abfragen zu erstellen.
 - Das Bereitstellen geeigneter Sichten erlaubt einen einfachen Zugriff, ohne Kenntnis des darunter liegenden Schemas und ohne Aufweichung der Normalisierung.

Der Sinn von Views

- Ein weiterer Vorteil von Sichten ist, dass das DBMS keinen zusätzlichen Aufwand zur Vorbereitung der Abfrage benötigt.
 - Die Sicht-Abfrage wurde vom Parser bereits bei der Erstellung syntaktisch zerlegt und vom Anfrageoptimierer vereinfacht.
- Ein Nachteil von Sichten kann sein, dass die Komplexität der dahinter liegenden Abfrage unterschätzt wird.
 - Der Aufruf einer Sicht kann zu sehr aufwändigen Abfragen führen und der unbedachte Einsatz solcher dann zu erheblichen Performanceproblemen.