THEMENWOCHE 1 · LERNFELD 8

Relationale Datenbanken und SQL

Von Rohdaten zur administrierten Datenbank

Nachschlagewerk für Lernende

Dieses Nachschlagewerk begleitet alle fünf Aufgabenpakete. Es erklärt die Begriffe und Befehle, liefert aber keine fertige Lösung für die Verkaufsdaten.

Lernziele und Arbeitsweise

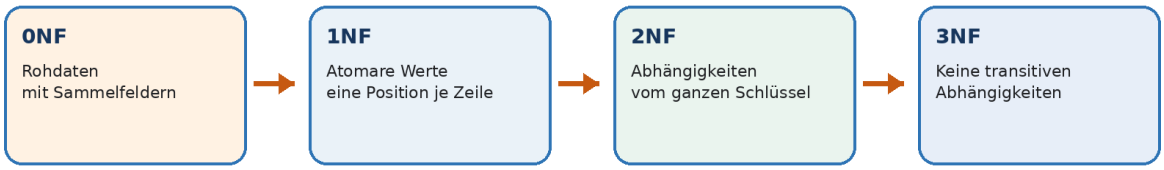
Am Ende der Themenwoche kannst du …

- unstrukturierte Daten schrittweise bis zur dritten Normalform überführen und Entscheidungen begründen,
- ein ER-Modell in Chen-Notation und ein erweitertes EER-Modell erstellen,
- ein lokales MySQL- oder MariaDB-System einrichten und seine Funktion prüfen,
- SQL-Befehle den Ebenen DDL, DML, DQL, DCL und TCL zuordnen und anwenden,
- ein relationales Modell als Datenbank mit Tabellen, Schlüsseln und Integritätsregeln umsetzen,
- Benutzerkonten nach dem Prinzip der minimalen Rechte einrichten und kontrollieren.

Aufgabenpaket	Thema	Zentrales Ergebnis
1	Normalisierung	Dokumentierte 1NF, 2NF und 3NF
2	ER und EER	ER-Modell (Chen) und EER-Modell
3	DBMS	Lauffähige lokale Entwicklungsumgebung
4.1	SQL verstehen und Datenbank aufbauen	Befehlsübersicht und zwei SQL-Skripte für Tabellen und Integritätsregeln
4.2	Mit Daten arbeiten und Ergebnisse prüfen	SQL-Skript für Testdaten und geprüfte SQL-Anweisungen mit tee-Protokollen
5	Benutzer und Rechte	Getestetes Berechtigungskonzept

1 Normalisierung von Daten

Normalisierung zerlegt eine unübersichtliche Datensammlung in Tabellen mit eindeutigen Aufgaben. Das reduziert Redundanzen und verhindert Änderungs-, Einfüge- und Löschanomalien.



Stufe	Prüffrage	Typische Maßnahme
0NF	Enthält ein Feld mehrere Werte oder zusammengesetzte Angaben?	Rohdaten sichern und Probleme markieren
1NF	Ist jeder Attributwert atomar und gibt es einen Schlüssel?	Sammelfelder aufteilen, eine Position je Zeile
2NF	Hängt jedes Nichtschlüsselattribut vom gesamten Schlüssel ab?	Teilabhängige Daten in eigene Tabellen verschieben
3NF	Hängen Nichtschlüsselattribute nur vom Schlüssel ab?	Transitive Abhängigkeiten auflösen

Funktionale Abhängigkeiten

Die Schreibweise A → B bedeutet: Ein Wert von A bestimmt genau einen Wert von B. Beispiel: Artikel_ID → Artikelbezeichnung. Bei Position mit dem Schlüssel (Verkaufs-ID, Positionsnummer) gilt: Verkaufs-ID → Verkaufsdatum ist eine Teilabhängigkeit und wird in der 2NF ausgelagert. Positionsschlüssel → Artikel_ID → Artikelbezeichnung ist dagegen transitiv und wird in der 3NF aufgelöst. Artikel_ID wird dann Primärschlüssel in Artikel und bleibt als Fremdschlüssel in Position. Entscheidend ist stets der Schlüssel der gerade untersuchten Tabelle.

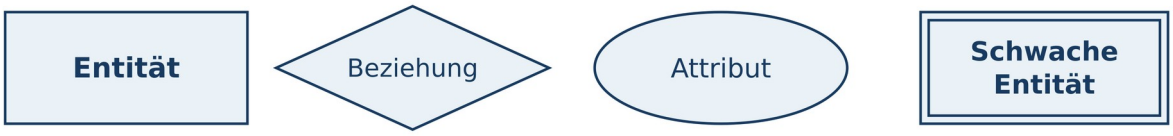
Anomalien erkennen

Anomalie	Beispiel	Folge
Änderung	Ein Herstellerland steht in vielen Zeilen.	Eine einzelne Korrektur lässt widersprüchliche Daten zurück.
Einfügen	Ein neuer Artikel ist noch nicht verkauft worden.	Der Artikel kann in einer reinen Verkaufsliste nicht gespeichert werden.
Löschen	Der letzte Verkauf eines Artikels wird gelöscht.	Dabei verschwinden auch die Artikelinformationen.

Wichtig: Bereinige offensichtliche Schreibweisen getrennt von der Normalisierung. Eine fachlich unbekannte Information darfst du nicht erfinden.

2 ER- und EER-Modelle

Ein ER-Modell beschreibt Fachobjekte, ihre Merkmale und Beziehungen unabhängig von einer konkreten Datenbank. Das EER-Modell erweitert diese Sicht um zusätzliche Konzepte wie schwache Entitäten. In dieser Themenwoche erstellst du beide Fassungen; im EER-Modell steht die abhängige Identifikation der Verkaufsposition im Mittelpunkt.



Kardinalitäten stehen an den Beziehungslinien: 1:1 · 1:n · m:n

Schwache Entitäten werden durch den Schlüssel der besitzenden Entität und einen partiellen Schlüssel gemeinsam identifiziert.

Element	ER-Modell (Chen)	Bedeutung
Entitätstyp	Rechteck	Menge gleichartiger Fachobjekte
Beziehungstyp	Raute	Fachlicher Zusammenhang zwischen Entitäten
Attribut	Ellipse	Eigenschaft eines Entitäts- oder Beziehungstyps
Schlüsselattribut	Unterstrichener Attributname	Identifiziert eine Entität eindeutig
Mehrwertiges Attribut	Doppelte Ellipse	Kann mehrere Werte je Entität besitzen
Schwache Entität siehe Zusatzabschnitt Z1	Doppeltes Rechteck	Benötigt eine besitzende Entität zur Identifikation

Kardinalitäten und Teilnahme

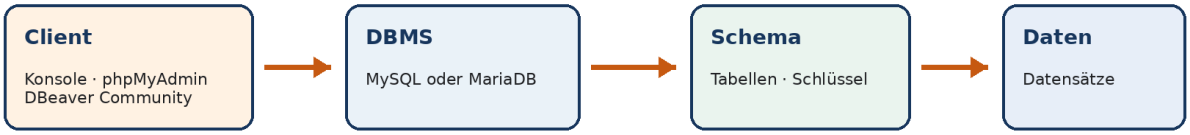
1:1, 1:n und m:n beschreiben die maximale Anzahl beteiligter Entitäten. Mit Min-Max-Angaben wie (0,1), (1,1), (0,n) oder (1,n) wird zusätzlich sichtbar, ob die Teilnahme optional oder verpflichtend ist.

Vom Modell zu Tabellen

- Jeder starke Entitätstyp wird in der Regel zu einer Tabelle.
- Eine 1:n-Beziehung wird meist durch einen Fremdschlüssel auf der n-Seite umgesetzt.
- Eine m:n-Beziehung benötigt eine Zwischentabelle. Deren Schlüssel besteht häufig aus den beiden Fremdschlüsseln oder aus einer zusätzlichen ID.
- Das relationale Tabellenmodell ist eine eigene Entwurfsstufe. Für das ER-Modell in Chen-Notation wird ein geeignetes Zeichenwerkzeug verwendet.

3 Datenbankmanagementsystem einrichten

Ein Datenbankmanagementsystem verwaltet Schemata, Tabellen, Datensätze, Benutzer, Rechte, Transaktionen und Sicherungen. Für die Themenwoche genügt eine lokale Entwicklungsumgebung.



Baustein	Aufgabe	Prüfung
Datenbankserver	SQL verarbeiten und Daten speichern	Server läuft, Version und Port sind bekannt
Client	Verbindung aufbauen und Befehle senden	Anmeldung über die Konsole an localhost funktioniert
Verwaltungsoberfläche	Objekte und Daten grafisch verwalten	phpMyAdmin oder dBeaver Community öffnet
Protokoll	Schritte und Fehlermeldungen festhalten	TEE-Protokoll und SQL-Skript sind nachvollziehbar

UniServer Zero und XAMPP

UniServer Zero und XAMPP sind die unterstützten Entwicklungsumgebungen dieser Themenwoche. XAMPP liefert unter Windows eine Entwicklungsumgebung mit MariaDB und phpMyAdmin. UniServer Zero ist eine portable Alternative. Arbeite nur lokal, öffne keine Ports ins Schul- oder Internet und installiere Dienste nur mit Erlaubnis. XAMPP ist laut Hersteller für Entwicklungsumgebungen gedacht, nicht für den produktiven Einsatz.

Microsoft SQL Server als eigenverantwortliche Alternative

Lernende mit vorhandener Erfahrung in Microsoft SQL Server dürfen diese Umgebung verwenden. Sie wird in der Themenwoche nicht unterstützt. Einrichtung, Fehlerbehebung sowie die notwendige Anpassung der MySQL- und MariaDB-Beispiele an Transact-SQL liegen vollständig in der Verantwortung der Lernenden.

Zugelassene Verwaltungswerkzeuge

Für die Arbeit können phpMyAdmin und die Community-Version von dBeaver verwendet werden. MySQL Workbench soll nicht eingesetzt werden, weil die Unterstützung von MariaDB nicht vollständig kompatibel ist und dadurch bei Modellierung, Metadaten oder Verwaltung abweichendes Verhalten auftreten kann.

Funktionsprüfung

```
SELECT VERSION();
SHOW DATABASES;
CREATE DATABASE db_probe CHARACTER SET utf8mb4;
DROP DATABASE db_probe;
```

Wenn der Server nicht startet, prüfe zuerst Port 3306, bereits laufende MySQL- oder MariaDB-Dienste, Schreibrechte des Datenverzeichnisses und die Serverprotokolle.

Falls Probleme auftreten: Beschreibe das beobachtete Problem, deine Prüfschritte und die gefundene Lösung. Nutze dazu die oben genannten Hinweise zur Fehlersuche. Wenn die Einrichtung problemlos funktioniert, ist keine zusätzliche Fehleranalyse erforderlich.

4 Grundlagen der Sprache SQL

SQL ist die Standardsprache relationaler Datenbanken. Die Einteilung in Befehlsgruppen erleichtert die Orientierung. In einzelnen Quellen wird SELECT der DML zugerechnet; in diesem Unterricht wird SELECT als DQL getrennt betrachtet.



Ebene	Zweck	Beispiele
DDL	Strukturen definieren	CREATE, ALTER, DROP
DML	Datensätze verändern	INSERT, UPDATE, DELETE
DQL	Datensätze auswählen	SELECT, WHERE, ORDER BY
DCL	Zugriffsrechte steuern	GRANT, REVOKE
TCL	Transaktionen steuern	START TRANSACTION, COMMIT, ROLLBACK

Grundmuster für Datenbank und Tabelle

```
CREATE DATABASE db_training
 CHARACTER SET utf8mb4;
USE db_training;

CREATE TABLE artikel (
 artikel_id INT PRIMARY KEY AUTO_INCREMENT,
 bezeichnung VARCHAR(80) NOT NULL,
 preis DECIMAL(10,2) NOT NULL
) ENGINE=InnoDB;
```

Grundmuster für Datenänderungen und Auswahl

```
INSERT INTO artikel (bezeichnung, preis)
VALUES ('Beispielartikel', 19.90);

UPDATE artikel
SET preis = 21.50
WHERE artikel_id = 1;

SELECT artikel_id, bezeichnung, preis
FROM artikel
WHERE preis >= 20.00
ORDER BY bezeichnung;
```

Sicherheitsregel: Führe UPDATE und DELETE zuerst als SELECT mit derselben WHERE-Bedingung aus. Ohne WHERE kann der Befehl alle Zeilen betreffen.

5 Datenbank und Tabellen aus dem Modell erzeugen

Die Umsetzung folgt einer festen Reihenfolge: Schema anlegen, Elterntabellen erzeugen, Kindtabellen erzeugen, Fremdschlüssel ergänzen, Testdaten einfügen und Ergebnisse prüfen.

Entscheidung	Empfehlung	Begründung
Zeichensatz	utf8mb4	Speichert den vollständigen Unicode-Zeichenraum.
Text variabler Länge	VARCHAR(n)	Verbraucht nur die benötigte Länge bis zur Grenze.
Postleitzahl	VARCHAR(10)	Führende Nullen und internationale Formate bleiben erhalten.
Geldbetrag	DECIMAL(10,2)	Exakte Dezimalwerte und ausreichender Wertebereich.
Fremdschlüssel	Passender Datentyp auf beiden Seiten	Voraussetzung für referenzielle Integrität.
Storage Engine	InnoDB	Unterstützt Transaktionen und Fremdschlüssel.

Fremdschlüssel definieren

```
ALTER TABLE kindtabelle
 ADD CONSTRAINT fk_kind_eltern
 FOREIGN KEY (eltern_id)
 REFERENCES elterntabelle (eltern_id)
 ON UPDATE CASCADE
 ON DELETE RESTRICT;
```

Option	Wirkung	Wann sinnvoll
RESTRICT oder NO ACTION	Verhindert Löschen der Elternzeile bzw. Ändern ihres referenzierten Schlüssels, solange Kindzeilen darauf verweisen.	Abhängige Daten sollen erhalten bleiben. Andere Attribute der Elternzeile bleiben änderbar.
CASCADE	Überträgt Änderung oder Löschung auf Kindzeilen.	Kindzeilen haben ohne Elternzeile keine Bedeutung.
SET NULL	Setzt den Fremdschlüssel auf NULL.	Die Beziehung ist optional und die Spalte erlaubt NULL.

Kontrollbefehle

```
SHOW TABLES;
DESCRIBE tabellenname;
SHOW CREATE TABLE tabellenname;
SELECT COUNT(*) FROM tabellenname;
```

6 Benutzer und Rechte verwalten

Dieses Anwendungsbeispiel zeigt den vollständigen Weg von der Testdatenbank über Benutzerkonten und Rechte bis zu deren Prüfung und Entfernung. Du arbeitest in der MySQL- oder MariaDB-Konsole. Alle benötigten Anweisungen stehen in diesem Abschnitt.

6.1 Ausgangssituation und Vorbereitung

In der lokalen Datenbank dbtest werden zwei Personen als Testdaten gespeichert. Ein Konto soll die Datenbank bearbeiten dürfen, ein zweites ausschließlich lesen. Die Beispieldaten und Passwörter dienen nur dieser Übung.

Konto	Aufgabe	Rechte auf dbtest
dbuser@localhost	Datenbank bearbeiten	Alle Rechte auf dieser Datenbank
dbuser_1@localhost	Daten lesen	Nur SELECT

Verwende eine neue Übungsdatenbank. Falls dbtest oder eines der Konten bereits existiert, prüfe zunächst dessen Verwendung und wähle gegebenenfalls andere Namen. Führe Einrichtung und Verwaltung als root oder mit einem entsprechend berechtigten Administrationskonto aus. Öffne die Konsole aus deiner Entwicklungsumgebung oder im Verzeichnis des Clients:

```
mysql -u root -p
```

Das Passwort wird nach dem Start abgefragt. Bei MariaDB kann der Client auch mariadb heißen. Mit SELECT VERSION(); ermittelst du Produkt und Version. Für ein Protokoll kannst du vor den folgenden Schritten tee einschalten; die Bedienung steht in Abschnitt 6.8.

Testdatenbank und Tabelle anlegen

```
CREATE DATABASE dbtest
  CHARACTER SET utf8mb4 COLLATE utf8mb4_german2_ci;
USE dbtest;
CREATE TABLE tblperson (
  id_person INT PRIMARY KEY,
  nachname VARCHAR(30) NOT NULL,
  vorname VARCHAR(30) NOT NULL
) ENGINE=InnoDB;
INSERT INTO tblperson (id_person, nachname, vorname)
VALUES (1, 'Weber', 'Anna'), (2, 'Yilmaz', 'Deniz');
SHOW TABLES;
SELECT * FROM tblperson ORDER BY id_person;
```

CHARACTER SET legt den Zeichensatz fest. COLLATE bestimmt Vergleichs- und Sortierregeln für Text. Die beiden SELECT-Ergebniszeilen bilden die Ausgangsbasis für die späteren Rechteprüfungen.

6.2 Konten anlegen und Rechte vergeben

Ein Konto besteht aus Benutzername und Host. 'dbuser'@'localhost' gilt für lokale Verbindungen. Ein Konto mit demselben Namen und einem anderen Host ist ein eigenständiges Konto. Der Host '%' steht grundsätzlich für beliebige Hosts; er öffnet jedoch nicht automatisch Netzwerk, Serverport oder Firewall. Für dieses Beispiel werden beide Konten ausschließlich mit localhost angelegt.

```
CREATE USER 'dbuser'@'localhost'
  IDENTIFIED BY 'LF8_Uebung_Admin_2026!';

CREATE USER 'dbuser_1'@'localhost'
  IDENTIFIED BY 'LF8_Uebung_Lesen_2026!';

GRANT ALL PRIVILEGES ON dbtest.*
  TO 'dbuser'@'localhost';

GRANT SELECT ON dbtest.*
  TO 'dbuser_1'@'localhost';
```

CREATE USER legt das Konto an. GRANT erteilt die angegebenen Rechte. dbtest.* begrenzt die Vergabe auf die Datenbank dbtest und deren Tabellen. ALL PRIVILEGES auf dieser Ebene gewährt keine globale Serveradministration, kein globales CREATE USER und nicht automatisch das Recht, Rechte weiterzugeben. Dafür wäre zusätzlich GRANT OPTION nötig.

In einer Anwendung erhält jedes Konto nur die tatsächlich benötigten Rechte. Das weitgehend berechtigte Konto dient hier dazu, den Unterschied zum reinen Lesekonto zu untersuchen.

Verwendung	Typischer Rechtebedarf	Nicht erforderlich
Lesen	SELECT	INSERT, UPDATE, DELETE, DROP
Datenpflege	SELECT, INSERT, UPDATE, DELETE	CREATE USER, GRANT OPTION
Schemaentwicklung	CREATE, ALTER, INDEX, REFERENCES und benötigte Datenrechte	Globale Serveradministration

6.3 Vergebene Rechte anzeigen

```
SHOW GRANTS FOR 'dbuser'@'localhost';
SHOW GRANTS FOR 'dbuser_1'@'localhost';
SHOW CREATE USER 'dbuser_1'@'localhost';
```

SHOW GRANTS zeigt die vergebenen Rechte. Unten steht ein typischer relevanter Auszug für das neue Lesekonto. Darstellung und zusätzliche Zeilen können je nach Version abweichen:

```
GRANT USAGE ON *.* TO `dbuser_1`@`localhost`
GRANT SELECT ON `dbtest`.* TO `dbuser_1`@`localhost`
```

USAGE bedeutet hier, dass keine globalen Rechte vergeben wurden. SELECT auf dbtest.* ist das eigentliche Leserecht. SHOW CREATE USER beschreibt die Kontodefinition und die Authentifizierung; es zeigt das Passwort nicht im Klartext und ersetzt keine Rechteprüfung.

6.4 Speicherorte von Konten und Rechten

Die folgenden Abfragen führt das Administrationskonto aus. Ein normales Anwendungskonto benötigt keinen Zugriff auf das Systemschema mysql. Die Systemobjekte dienen hier zur Einsicht; ändere sie nicht direkt mit INSERT, UPDATE oder DELETE.

Systemobjekt	Welche Information steht dort
MySQL: mysql.user	Konten, Authentifizierung und statische globale Rechte
mysql.db	Rechte auf Datenbankebene, beispielsweise Select_priv und Update_priv
mysql.tables_priv und mysql.columns_priv	Rechte für einzelne Tabellen beziehungsweise Spalten
MariaDB ab 10.4: mysql.global_priv	Globale Kontodaten und Rechte; mysql.user ist hier eine Kompatibilitätsansicht

Vorhandene Konten finden

```
SELECT Host, User
FROM mysql.user
WHERE User IN ('dbuser', 'dbuser_1')
ORDER BY User, Host;
```

Nach dem Anlegen beider Konten enthält der gefilterte Auszug diese Einträge:

Host	User
localhost	dbuser
localhost	dbuser_1

Mit USE mysql; und anschließend SELECT Host, User FROM user; erhältst du denselben Zugriff auf das Systemobjekt. Die vollständig qualifizierte Schreibweise mysql.user funktioniert unabhängig von der aktuell gewählten Datenbank und vermeidet Verwechslungen.

Datenbankbezogene Rechte nachvollziehen

```
SELECT Host, Db, User, Select_priv, Update_priv
FROM mysql.db
WHERE Db = 'dbtest'
  AND User IN ('dbuser', 'dbuser_1')
ORDER BY User, Host;
```

Host	Db	User	Select_priv	Update_priv
localhost	dbtest	dbuser	Y	Y
localhost	dbtest	dbuser_1	Y	N

Y bedeutet, dass das betreffende Recht auf dieser Ebene erteilt wurde; N bedeutet, dass es dort nicht erteilt wurde. dbuser darf aufgrund von ALL PRIVILEGES lesen und ändern. dbuser_1 darf lesen, aber nicht ändern. Ein N allein beweist jedoch keinen generellen Rechteentzug: Globale Rechte, andere Vergabeebenen oder Rollen können ebenfalls Zugriffe ermöglichen. Deshalb prüfst du zusätzlich SHOW GRANTS und das tatsächliche Verhalten.

6.5 Anmeldung und Zugriffe praktisch prüfen

Lass die Administrationskonsole geöffnet. Verwende für jeden Test eine zusätzliche, eindeutig zugeordnete Konsolensitzung. Änderungen an Konten oder Rechten werden weiterhin in der Administrationssitzung ausgeführt.

Mit dem Lesekonto anmelden

```
mysql -h localhost -u dbuser_1 -p
```

```
SELECT USER(), CURRENT_USER();
USE dbtest;
SELECT id_person, nachname, vorname
FROM tblperson ORDER BY id_person;
```

USER() zeigt den bei der Verbindung angegebenen Benutzer samt Verbindungsherkunft. CURRENT_USER() zeigt das vom Server für die Berechtigungsprüfung verwendete Konto. Im Beispiel muss das Lesekonto dbuser_1@localhost verwendet werden. Das SELECT muss die beiden Ausgangsdatensätze liefern.

id_person	nachname	vorname
1	Weber	Anna
2	Yilmaz	Deniz

Einen unzulässigen Schreibzugriff testen

```
SELECT * FROM tblperson WHERE id_person = 1;
UPDATE tblperson SET nachname = 'Becker'
WHERE id_person = 1;
```

Die SELECT-Prüfung ist erlaubt. Das UPDATE muss abgewiesen werden, typischerweise mit ERROR 1142 und dem Hinweis „UPDATE command denied“. Der Nachname bleibt Weber. Eine solche erwartete Fehlermeldung belegt, dass die Rechtebegrenzung greift. Funktioniert das UPDATE dennoch, prüfe CURRENT_USER(), zusätzliche Rechte und aktive Rollen.

Mit dem Bearbeitungskonto vergleichen

Beende die Lesesitzung mit quit und melde dich als dbuser an:

```
mysql -h localhost -u dbuser -p
```

```
USE dbtest;
SELECT * FROM tblperson WHERE id_person = 1;
UPDATE tblperson SET nachname = 'Becker'
WHERE id_person = 1;
SELECT * FROM tblperson WHERE id_person = 1;
```

Jetzt muss das UPDATE erfolgreich sein. Die anschließende Abfrage zeigt für id_person 1 den Nachnamen Becker und den Vornamen Anna. Dokumentiere jeweils Konto, Anweisung, erwartetes Ergebnis und tatsächliche Ausgabe.

6.6 Rechte entziehen und erneut prüfen

Wechsle zur Administrationssitzung. Entziehe dem Lesekonto sein SELECT-Recht, ohne das Konto zu löschen:

```
REVOKE SELECT ON dbtest.*
  FROM 'dbuser_1'@'localhost';
SHOW GRANTS FOR 'dbuser_1'@'localhost';
```

Die zuvor angezeigte SELECT-Vergabe muss verschwunden sein. Melde das Lesekonto neu an und prüfe mit SELECT * FROM dbtest.tblperson; erneut. Ohne weitere Rechte muss der Zugriff nun scheitern. Das Konto kann weiterhin existieren und sich anmelden; seine Berechtigung zum Lesen wurde entfernt.

Soll das Konto wieder lesen dürfen, erteile das Recht in der Administrationssitzung erneut mit GRANT SELECT ON dbtest.* TO 'dbuser_1'@'localhost';. Kontrolliere danach SHOW GRANTS und den Zugriff in einer neuen Lesesitzung.

CREATE USER, GRANT, REVOKE und DROP USER aktualisieren die Rechteverwaltung selbst. Ein zusätzliches FLUSH PRIVILEGES ist danach normalerweise nicht erforderlich. Das manuelle Neuladen ist für besondere Administrationsfälle vorgesehen, beispielsweise nach direkten Änderungen an Grant-Tabellen. Solche Änderungen gehören nicht zu dieser Übung.

6.7 Übungskonten entfernen

Beende zuerst die Testsitzungen. Lösche anschließend im Administrationskonto die beiden ausschließlich für die Übung angelegten Konten und kontrolliere das Ergebnis:

```
DROP USER 'dbuser'@'localhost';
DROP USER 'dbuser_1'@'localhost';
SELECT Host, User FROM mysql.user
WHERE User IN ('dbuser', 'dbuser_1');
```

Die gefilterte Abfrage muss nun keine dieser beiden Konten mehr liefern. DROP USER entfernt ein Konto mit seinen Vergaben, aber nicht die Datenbank. Erst wenn die Übungsdaten nicht mehr benötigt werden, kannst du ausschließlich die hier angelegte Testdatenbank mit DROP DATABASE dbtest; entfernen.

6.8 Konsolensitzung mit tee protokollieren

Lege unter Windows zuerst einen Ordner (Beispiel) C:\LF8\protokolle an. In der interaktiven MySQL- oder MariaDB-Konsole startest du dann die Mitschrift. tee und notee sind Clientkommandos, keine SQL-Anweisungen; schreibe sie jeweils in eine eigene Zeile ohne Semikolon:

```
tee C:/LF8/protokolle/rechte.txt
SELECT VERSION();
SHOW GRANTS FOR CURRENT_USER;
notee
```

Der Ordner muss vorhanden und beschreibbar sein. Die Textdatei wird bei Bedarf angelegt; bei einer bestehenden Datei hängt tee neue Ein- und Ausgaben an. notee beendet die Mitschrift. Starte tee in jeder Sitzung, deren Ergebnisse du nachweisen möchtest, und nutze zur Zuordnung gegebenenfalls getrennte Dateinamen je Konto.

Das Protokoll enthält SQL-Anweisungen und Ausgaben. Eine wiederverwendbare .sql-Datei enthält dagegen nur ausführbare Anweisungen und SQL-Kommentare. Wie du beide für dein ePortfolio verwendest, steht zentral unter „Dokumentationsstandard im ePortfolio“ in Abschnitt 7.

7 Prüfen, dokumentieren und reflektieren

Gleiche Modell, SQL-Skripte und Datenbank anhand der Aufgaben ab. Für Fremdschlüssel genügt in dieser Themenwoche die Kontrolle der Definitionen mit SHOW CREATE TABLE sowie die Planung der Testfälle. Ausführliche praktische Fremdschlüsseltests folgen in Themenwoche 2.

Prüfbereich	Nachweis
Normalisierung	Schlüssel und funktionale Abhängigkeiten je Stufe
Modell	ER-Modell: Chen-Symbole, Beziehungstypen, Kardinalitäten und Teilnahmebedingungen
Struktur	SHOW CREATE TABLE für alle Tabellen
Daten	Zeilenzahlen und gezielte SELECT-Abfragen
Integrität	Fremdschlüsseldefinitionen kontrollieren; Lösch- und Änderungsregeln begründen und Testfälle planen
Rechte	SHOW GRANTS und Anmeldung mit jedem Testkonto

Dokumentationsstandard im ePortfolio

SQL-Anweisungen und ihre Ergebnisse werden nicht als Screenshots dokumentiert. Konsolenbilder mit heller Schrift auf dunklem Hintergrund sind schlecht lesbar, nicht durchsuchbar und lassen sich für die Prüfungsvorbereitung nicht sinnvoll weiterverwenden. Arbeite deshalb vorrangig in der MySQL- oder MariaDB-Konsole und protokolliere die Sitzung mit TEE und NOTEE, wie in Abschnitt 6 beschrieben.

- Übernimm aus dem TEE-Protokoll die ausgeführte SQL-Anweisung und die dazugehörige Ausgabe als auswählbaren Text in das ePortfolio.
- Kennzeichne zu jedem Ausschnitt die Aufgabe, den Zweck des Befehls und das geprüfte Ergebnis.
- Für eine spätere Wiederverwendung kannst du ausgewählte SQL-Anweisungen zusätzlich in einer SQL-Datei sammeln. Ab A4.2.2 genügt das tee-Protokoll mit Anweisungen und Ergebnissen; ein weiteres ausführbares Skript ist nicht erforderlich.
- Verwende keine Screenshots als Nachweis für SQL-Befehle, Tabellenstrukturen, Abfrageergebnisse oder Fehlermeldungen. Erforderliche Modelle und Diagramme werden als sauber exportierte Abbildungen eingefügt.
- Übe für die AP2 ausgewählte SQL-Anweisungen zusätzlich mit Stift und Papier. Vergleiche die handschriftliche Lösung anschließend mit deinem geprüften SQL-Bestand.

Empfohlene ePortfolio-Struktur

- 01 Start und BBK
- 02 Normalisierung
- 03 ER- und EER-Modelle
- 04 DBMS-Einrichtung
- 05 SQL Grundlagen
- 06 Datenbank und Tabellen
- 07 Benutzer und Rechte
- 08 Abschluss und Reflexion

Leistungsnachweis

Du präsentierst dein ePortfolio und erläuterst deine Ergebnisse und Lösungswege in einem Fachgespräch. Die Präsentationsform ist frei; der Termin in der folgenden Schulwoche wird mit der Lehrperson vereinbart (zusammen ca. 20 bis 30 Minuten). Bewertungsgrundlage ist das BBK. Ein schriftlicher Test entfällt.

Quellen und weiterführende Dokumentation

- MySQL 8.4 Referenzhandbuch CREATE USER: https://dev.mysql.com/doc/refman/8.4/en/create-user.html
- MySQL 8.4 Referenzhandbuch GRANT: https://dev.mysql.com/doc/refman/8.4/en/grant.html
- MySQL 8.4 Referenzhandbuch SHOW GRANTS: https://dev.mysql.com/doc/refman/8.4/en/show-grants.html
- MySQL 8.4 Referenzhandbuch Grant Tables: https://dev.mysql.com/doc/refman/8.4/en/grant-tables.html
- MariaDB Dokumentation CREATE USER: https://mariadb.com/docs/server/reference/sql-statements/account-management-sql-statements/create-user
- MariaDB Dokumentation mysql.global_priv: https://mariadb.com/docs/server/reference/system-tables/the-mysql-database-tables/mysql-global_priv-table
- Apache Friends XAMPP FAQ für Windows: https://www.apachefriends.org/faq_windows.html
- Uniform Server Projektseite: https://sourceforge.net/projects/miniserver/
- dBeaver Dokumentation zu Datenbanktreibern: https://dbeaver.com/docs/dbeaver/Database-drivers/
- MySQL Workbench Referenzhandbuch: https://dev.mysql.com/doc/workbench/en/
- Microsoft Transact-SQL Referenz: https://learn.microsoft.com/en-us/sql/t-sql/language-reference

Zusatzabschnitt Z1 · Schwache Entitäten

Ergänzung zu Abschnitt 2 „ER- und EER-Modelle“. Eine schwache Entität besitzt keinen eigenen vollständigen Schlüssel. Sie wird erst durch den Schlüssel einer besitzenden Entität zusammen mit einem partiellen Schlüssel eindeutig identifiziert.

Z1.1 Ein klassisches Beispiel

Stell dir ein Datenmodell für ein Unternehmen vor, das Mitarbeiter und deren Kinder für familienbedingte Zuschläge erfasst:

Konzept	Beschreibung	Beispiel
Starke Entität	Mitarbeiter wird durch Mitarbeiter_ID eigenständig identifiziert.	101 · Max Mustermann
Schwache Entität	Kind hat unter anderem Vorname und Geburtsdatum. Der Vorname allein ist nicht eindeutig.	Kind „Anna“
Identifikation	Mitarbeiter_ID und der partielle Schlüssel Vorname identifizieren das Kind gemeinsam.	(101, Anna)

Modellannahme: Innerhalb der zu einem Mitarbeiter erfassten Kinder ist jeder Vorname eindeutig. Nur unter dieser Annahme eignet sich Vorname als partieller Schlüssel. Andernfalls wird ein anderer oder zusätzlicher Diskriminator benötigt.

Z1.2 Darstellung in Chen-Notation

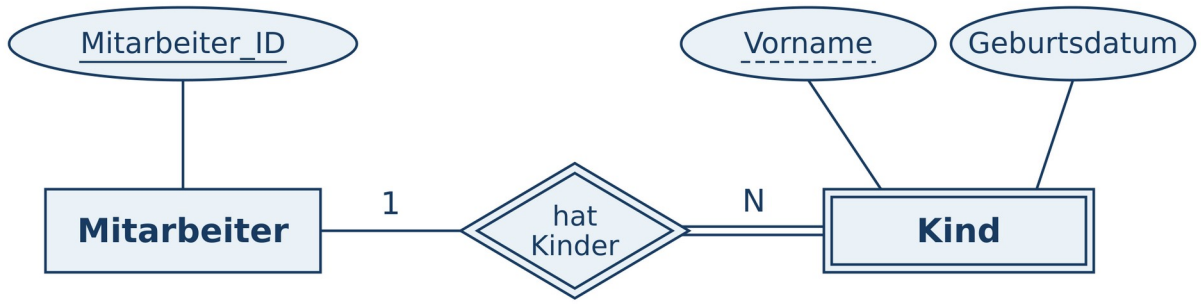


Abbildung Z1 · Ausschnitt mit den für die Identifikation wesentlichen Attributen. Weitere Mitarbeiterattribute enthält der SQL-Entwurf.

Mitarbeiter: Ein einfaches Rechteck kennzeichnet den starken Entitätstyp. Mitarbeiter_ID ist als Schlüssel durchgehend unterstrichen.

hat Kinder: Die doppelte Raute kennzeichnet die identifizierende Beziehung. Das doppelte Rechteck kennzeichnet den schwachen Entitätstyp Kind; Vorname ist als partieller Schlüssel gestrichelt unterstrichen.

Teilnahme und Kardinalität: Die doppelte Verbindungslinie auf der Kind-Seite bedeutet totale Teilnahme: Jedes erfasste Kind gehört in diesem Modell zu genau einem Mitarbeiter. Ein Mitarbeiter kann kein, ein oder mehrere Kinder haben. Die Beziehung ist 1:N.

Z1.3 Umsetzung als relationales Schema

Bei der Übersetzung in SQL erhält die Tabelle Kind einen zusammengesetzten Primärschlüssel aus Mitarbeiter_ID und Vorname. Mitarbeiter_ID ist zugleich Fremdschlüssel zur Tabelle Mitarbeiter. Eine zusätzliche künstliche Kind_ID ist für dieses Beispiel nicht erforderlich.

```
-- Starke Entität
CREATE TABLE Mitarbeiter (
    Mitarbeiter_ID INT PRIMARY KEY,
    Nachname VARCHAR(50) NOT NULL,
    Vorname VARCHAR(50) NOT NULL,
    Abteilung VARCHAR(50)
) ENGINE=InnoDB;

-- Schwache Entität
CREATE TABLE Kind (
    Mitarbeiter_ID INT NOT NULL,
    Vorname VARCHAR(50) NOT NULL,
    Geburtsdatum DATE NOT NULL,
    PRIMARY KEY (Mitarbeiter_ID, Vorname),
    FOREIGN KEY (Mitarbeiter_ID)
        REFERENCES Mitarbeiter(Mitarbeiter_ID)
        ON DELETE CASCADE
) ENGINE=InnoDB;
```

Die Besonderheiten dieser SQL-Struktur

Zusammengesetzter Primärschlüssel: PRIMARY KEY (Mitarbeiter_ID, Vorname) verhindert, dass dieselbe Kombination doppelt vorkommt. Mitarbeiter 101 und Mitarbeiter 102 können jeweils ein Kind namens „Anna“ haben; die beiden Datensätze bleiben eindeutig unterscheidbar.

Identifikation und Existenzabhängigkeit: Ein Kind kann nur mit einem vorhandenen Mitarbeiter verknüpft werden. Der Fremdschlüssel sichert den Verweis; NOT NULL verhindert eine fehlende Zuordnung. Der Mitarbeiter-Schlüssel ist zugleich Teil des Kind-Schlüssels.

Gewählte Löschregel: Wird die Zeile eines Mitarbeiters gelöscht, entfernt ON DELETE CASCADE automatisch auch die zugehörigen Kind-Zeilen. Das folgt aus der ausdrücklich gewählten SQL-Regel, nicht allein aus der Bezeichnung „schwache Entität“.

Alternative: Mit ON DELETE RESTRICT würde die Löschung eines Mitarbeiters verhindert, solange zugehörige Kinder gespeichert sind. Welche Regel passt, wird fachlich entschieden.

Merksatz

Schwache Entität = Identifikation durch den Schlüssel der besitzenden Entität plus einen partiellen Schlüssel. Im Beispiel wird „Anna“ erst zusammen mit Mitarbeiter_ID eindeutig.

Weiterführend: Chen-Symbole und schwache Entitäten · MySQL Fremdschlüssel und Löschregeln

Zusatzabschnitt Z2 · Von der n:m-Beziehung zur Detailtabelle

Am Beispiel Kunde und Artikel lernst du, wie eine n:m-Beziehung durch eine Detailtabelle abgebildet wird. Aus einer direkten Beziehung entstehen zwei 1:n-Beziehungen.

Z2.1 Fachliches Modell und Tabellenmodell unterscheiden

Im fachlichen ER-Modell kann die Beziehung „Kunde kauft Artikel“ direkt als n:m-Beziehung dargestellt werden. Das relationale Tabellenmodell beschreibt dagegen die konkrete Umsetzung mit Tabellen, Spalten, Datentypen sowie Primär- und Fremdschlüsseln.

Für das relationale Tabellenmodell gilt: Die n:m-Beziehung wird über eine zusätzliche Detail- oder Zwischentabelle aufgelöst. Ein einzelner Fremdschlüssel zwischen Kunde und Artikel reicht dafür nicht aus. Die Detailtabelle stellt zwei 1:n-Beziehungen her.

Begriff EER: Einige Modellierungswerkzeuge nennen auch ein relationales Tabellenmodell „EER-Diagramm“. In diesem Sinn ist hier die Detailtabelle erforderlich. Das konzeptionelle Enhanced Entity-Relationship-Modell ist dagegen ein erweitertes fachliches Modell und erlaubt weiterhin n:m-Beziehungen. Entscheidend ist daher die Modellebene, nicht allein die Bezeichnung EER.

Z2.2 Kunde und Artikel als n:m-Beziehung

Ein Kunde kann mehrere Artikel kaufen. Derselbe Artikel kann von mehreren Kunden gekauft werden. Kunde und Artikel können auch ohne einen bisherigen Kauf gespeichert sein.



Abbildung Z2.1 · n:m-Beziehung in Chen-Notation, mit PlantUML erstellt.

Modellannahme: Wir betrachten je Kunde und Artikel die insgesamt gekaufte Menge. Wiederholte Käufe werden zusammengefasst; einzelne Bestellungen und Kaufzeitpunkte werden nicht erfasst.

Beispiel: Anna kauft eine Tastatur und zwei Mäuse. Ben kauft ebenfalls eine Maus. Die Gesamtmenge gehört zur Kombination aus Kunde und Artikel, nicht allein zum Kunden oder zum Artikel.

Übergang: Jede dieser Kombinationen wird im nächsten Schritt zu einem eigenen Kaufdetail. Die fachliche n:m-Beziehung bleibt dadurch vollständig abbildbar.

Z2.3 Relationales Tabellenmodell

Kaufdetail verbindet jeweils genau einen Kunden mit genau einem Artikel. Ein Kunde und ein Artikel können jeweils in beliebig vielen Kaufdetails vorkommen. Es entstehen Kunde–Kaufdetail mit 1:n und Artikel–Kaufdetail mit 1:n.

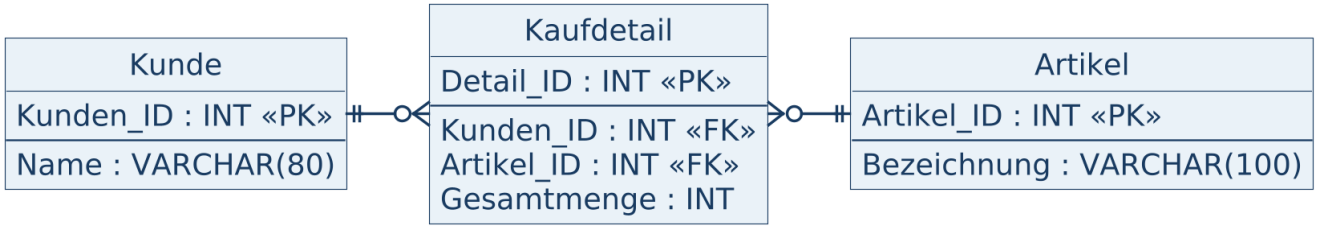


Abbildung Z2.2 · Relationales Tabellenmodell mit Attributen, Datentypen und Schlüsseln; Beziehungen in Krähenfußnotation. Mit PlantUML erstellt.

Notation: PK = Primärschlüssel, FK = Fremdschlüssel. Zwei Striche bedeuten genau einen Datensatz; Kreis und Krähenfuß bedeuten null bis viele Datensätze. Der Krähenfuß steht jeweils auf der Seite von Kaufdetail.

Datentypen: INT speichert ganze Zahlen, VARCHAR(n) Texte mit höchstens n Zeichen. Alle dargestellten Spalten sind NOT NULL.

Z2.4 Beispieldaten und Schlüsselregeln

Kunde behält Kunden_ID als Primärschlüssel, Artikel behält Artikel_ID. Die Tabelle Kaufdetail enthält zusätzlich beide IDs als Fremdschlüssel. Kaufdetail.Kunden_ID verweist auf Kunde.Kunden_ID; Kaufdetail.Artikel_ID verweist auf Artikel.Artikel_ID.

Detail_ID (PK)	Kunden_ID (FK)	Artikel_ID (FK)	Gesamtmenge
1	101	201	1
2	101	202	2
3	102	202	1

Zuordnung: 101 = Anna, 102 = Ben; 201 = Tastatur, 202 = Maus.

Schlüsselregel: Detail_ID identifiziert jede Zeile. UNIQUE (Kunden_ID, Artikel_ID) verhindert doppelte Kombinationen; beide Fremdschlüssel sind NOT NULL. Alternativ kann die Kombination selbst den Primärschlüssel bilden.

Merksatz: Die Detailtabelle speichert die Zuordnung und deren Eigenschaften. Für einzelne Kaufvorgänge benötigt das Modell zusätzlich beispielsweise Bestellung und Bestellposition.

Weiterführend: PlantUML Chen-Notation · PlantUML Krähenfußnotation · Relationale Umsetzung