EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

© 2014 cherezoff/Shutterstock

1. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 · 42781 Haan-Gruiten

Europa-Nr.: 33457

Verfasser:
Dirk Hardy, 46049 Oberhausen

Die in diesem Lehr- und Übungsbuch genannten Software-, Hardware- und Handelsnamen sind in der Mehrzahl auch eingetragene Warenzeichen.
Unter Verwendung von Screenshots aus:
XAMPP (© The Apache Software Foundation)

1. Auflage 2025

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Korrektur von Druckfehlern identisch sind.

ISBN 978-3-7585-3345-7

Alle Rechte vorbehalten. Das Werk ist urheberrechtlich geschützt. Jede Verwertung außerhalb der gesetzlich geregelten Fälle muss vom Verlag schriftlich genehmigt werden.

© 2025 by Verlag Europa-Lehrmittel, Nourney, Vollmer GmbH & Co. KG, 42781 Haan-Gruiten
www.europa-lehrmittel.de
Satz: Typework Layoutsatz & Grafik GmbH, 86153 Augsburg
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagbilder: yongqiang – stock.adobe.com; refresh(PIX) – stock.adobe.com
Druck: Plump Druck & Medien GmbH, 53619 Rheinbreitbach

Vorbemerkung

Das Speichern und Verarbeiten von Daten ist seit Beginn der Informationstechnologie ein beherrschendes Thema. Dazu wurde bereits in den 1970er-Jahren das relationale Datenbankmodell entwickelt, das bis heute als Standard in der Welt der Datenbankmodelle betrachtet werden kann. Diesem Modell liegen Relationen (Tabellen) zugrunde. Damit diese Relationen geeignet organisiert (erstellt, abgefragt, verändert) werden können, entstand im gleichen Zeitraum die strukturierte Abfragesprache ***SQL*** (**S**tructured **Q**uery **L**anguage). ***SQL*** hat sich im Laufe der Jahre immer weiterentwickelt und ist inzwischen ein mächtiges Instrument zur Verwaltung von relationalen Datenbanken. Innerhalb der Familie der Programmiersprachen kann ***SQL*** als deklarative (beschreibende) Sprache der 4. Generation bezeichnet werden. Der Stellenwert dieser Sprache für die Informationstechnologie wird auch in den nächsten Jahren sehr hoch bleiben.

Aufbau des Buches

Der Aufbau des Buches orientiert sich an dem Einsatz einer relationalen Datenbank in der Praxis. Neben den Erläuterungen zu den Grundlagen von relationalen Datenbanken wird eine praxisnahe Beispieldatenbank konzipiert, die als Grundlage für die Abfragen mit der Sprache ***SQL*** in allen weiteren Kapiteln des Buches genutzt wird. Somit wird durchgehend mit einer Datenbank gearbeitet. Die Sprache ***SQL*** wird dabei schrittweise gelernt. Aus motivationalen Gründen wird mit den reinen Abfragen begonnen und erst später wird der Teil von ***SQL*** betrachtet, mit dem Tabellen und weitere Datenbankobjekte erstellt werden können. Abgerundet wird das Buch einerseits durch einen Einstieg in komplexe Themen wie ***Events*** und ***Trigger***, aber auch durch einen Workshop, der alle wichtigen Themen innerhalb einer übergeordneten Aufgabenstellung behandelt.

Das Buch ist als Grundlagenbuch für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen, aber auch von den informationstechnischen Assistenten/-innen genutzt werden.

Als Entwicklungswerkzeuge wird in diesem Buch das **Softwarepaket XAMPP** genutzt. Dieses Paket ist kostenfrei als Download im Internet verfügbar. Im Anhang des Buches befinden sich neben der Auflistung der SQL-Schlüsselworte auch Hinweise zur Nutzung verschiedener Datenbankmanagementsysteme. Ebenso ist das komplette SQL-Skript zur Erstellung der Tabellen abgedruckt. Das Skript ist kostenfrei über das digitale Medienregal EUROPATHEK (www.europathek.de) als Download erhältlich. Den benötigten Freischaltcode finden Sie auf der Umschlaginnenseite des Buches.

Dirk Hardy Frühjahr 2025

Inhaltsverzeichnis

1 Grundlagen der Sprache SQL

Das folgende Kapitel bietet Hintergrundinformationen zu relationalen Datenbanken, der Sprache SQL und zu einer Datenbankkonzeption. Um SQL zu erlernen sind diese Informationen nicht zwingend, aber sie helfen beim Verständnis der Sprache. Das Kapitel soll deshalb auch keine vollständige Einführung in relationale Datenbanken und deren Konzeption liefern, sondern nur ein Basisverständnis schaffen, damit die Sprache SQL besser eingeordnet und genutzt werden kann.

1.1 Entwicklung relationaler Datenbanken und der Sprache SQL

In den 1970-er Jahren entwickelte E. F. Codd[1] bei der Firma IBM das relationale Datenmodell. Dieses Modell basiert auf Relationen (vergleichbar mit Tabellen) und war das richtungsweisende Modell für die darauffolgenden Datenbanksysteme bzw. Datenbankmanagementsysteme. Das erste relationale Datenbankmanagementsystem wurde aber 1979 von der Firma Oracle und nicht von IBM herausgebracht – IBM stellte erst 1983 mit DB2 sein relationales Datenbankmanagementsystem vor. Lange Zeit waren Oracle und IBM mit ihren Systemen marktführend, Oracle ist es aktuell immer noch, aber IBM wurde von MySQL (inzwischen auch in Oracle-Besitz) und Microsoft verdrängt. Für die ersten relationalen Datenbanken wurde eine eigene Sprache entwickelt, mit der die Datenbank abgefragt und administriert werden konnte – diese Sprache hieß ***SEQUEL*** (für **S**tructured **E**nglisch **Que**ry **L**anguage). Aus rechtlichen Gründen musste die Sprache umbenannt werden und heißt seitdem SQL (für **S**tructured **Q**uery **L**anguage). Inzwischen ist SQL ein eigenständiger Name für die Sprache (nach ISO/IEC-Standard[2]). Die folgende Grafik zeigt die Entwicklung der relationalen Datenbanken und der Sprache SEQUEL/SQL im Überblick:

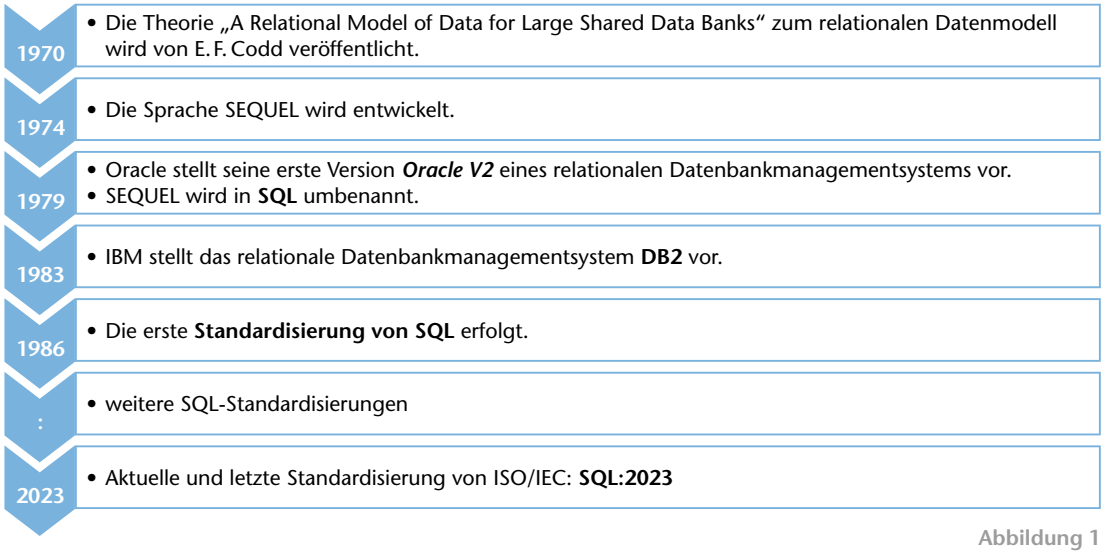


Abbildung 1

1 Codd (1923–2003) war ein englischer Mathematiker und Datenbankentwickler/Datenbanktheoretiker.

2 ISO ist die „Internationale Organisation für Normung“ und IEC ist die „Internationale Elektrotechnische Kommission“.

1.2 Einordnung der Sprache SQL

SQL ist eine Abfragesprache. Sie dient zur Abfrage von Datenbanken, aber auch zur Administration von Datenbanken. Das Ergebnis einer Abfrage (engl. query) ist dabei eine Teilmenge der Daten aus der Datenbank. Deshalb ist SQL auch keine klassische Programmiersprache wie C oder Basic, sondern gehört zu den **deklarativen Programmiersprachen**. Mit diesen Programmiersprachen wird das Ergebnis beschrieben und nicht der Weg, um das Ergebnis zu erhalten (so wie in einer klassischen Programmiersprache wie C). Es wird also das „WAS" beschrieben und nicht das „WIE". Um das „WIE" kümmert sich dann die Logik im Hintergrund der Sprache. SQL ist deshalb auch leichter lesbar. Die Anweisungen sind dem alltäglichen Sprachgebrauch damit deutlich näher, als bei einer Programmiersprache wie C. Das folgende Beispiel soll diesen Unterschied verdeutlichen. Dabei werden Grundkenntnisse in der Programmierung vorausgesetzt.

Gegeben ist eine Tabelle „Kunde" in einer relationalen Datenbank:

Tabelle Kunde:

Name	Vorname	Telefon
Hansen	Fred	1220201
Kaiser	Maik	2304023
Maier	Frauke	2373272
Knudsen	Rolf	2383838

Nun sollen alle Namen der Kunden angezeigt werden. In SQL würde die Anweisung so aussehen:

```
SELECT Name FROM Kunde;
```

Das ist fast selbsterklärend: Selektiere (**SELECT**) die Namen (**Name**) aus (**FROM**) der Tabelle Kunde.

In einer Sprache wie C könnte es hingegen so aussehen:

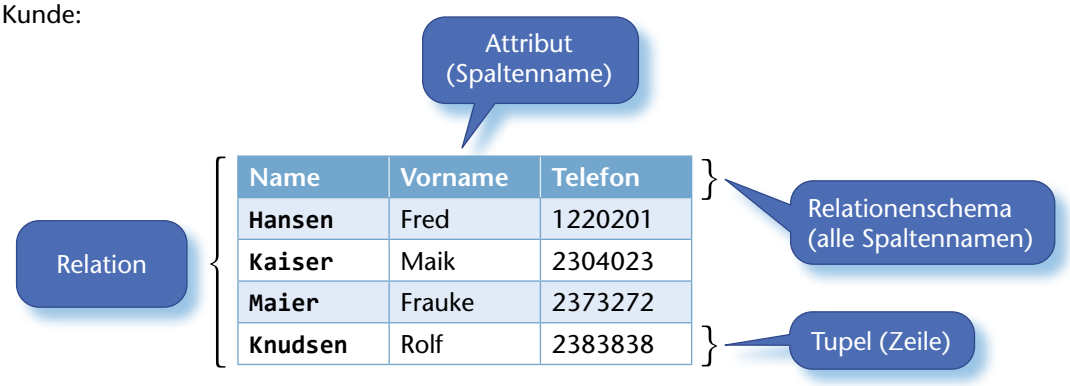
```
struct TKunde
{
        char Name[40];
        char Vorname[40];
        char Telefon[20];

};

int i = 0;

struct TKunde Kunde[4] = {      "Hansen","Fred","1220201",
                                "Kaiser","Maik","2304023",
                                "Maier","Frauke","2373272",
                                "Knudsen","Rolf","2383838"
                        };

for (i = 0; i < 4; i++)
{
        printf("%s\n", Kunde[i].Name);
}
```

Der Unterschied ist enorm. Für versierte C-Programmierer ist der Quelltext einfach zu verstehen, aber für einen Laien praktisch nicht lesbar. Die SQL-Anweisung hingegen ist aber auch für einen Laien durchaus nachvollziehbar. Das soll allerdings nicht bedeuten, dass SQL eine Sprache ist, die nur auf diesem leicht verständlichen Niveau arbeitet. Im Laufe der Kapitel wird noch deutlich, dass SQL sehr kompliziert sein kann, wenn das Ergebnis (das „WAS") entsprechend kompliziert ist.

1.3 Relationenalgebra als Basis von SQL

Die Relationenalgebra ist die theoretische Grundlage von Abfragesprachen für Datenbanken. Eine solche Relationenalgebra basiert auf Relationen und Operationen auf diesen Relationen. Relationen sind vergleichbar mit Tabellen und bestehen aus Attributen und Tupeln. Im Vergleich mit Tabellen sind die Attribute dann die Spaltennamen und die Tupel die einzelnen Zeilen einer Tabelle.

Das folgende Beispiel zeigt die Begrifflichkeiten im Zusammenhang mit der bereits bekannten Tabelle Kunde:



Name	Vorname	Telefon
Hansen	Fred	1220201
Kaiser	Maik	2304023
Maier	Frauke	2373272
Knudsen	Rolf	2383838

Folgende Operationen sind auf solchen Relationen definiert:

Bezeichnung	Beschreibung	Beispiel
Projektion	Unter Projektion wird das Ausblenden von Attributen verstanden, so dass bei einer Abfrage nur die gewünschten Attribute angezeigt werden.	Es soll nur der Name ausgewählt werden: Name Hansen Kaiser Maier Knudsen
Selektion	Die Selektion filtert einzelne Tupel aus der Relation. **Hinweis:** Die Selektion und die Projektion bilden die Basis-Abfragebefehle von SQL.	Alle Kunden sollen ausgewählt werden, die alphabetisch vor dem Kunden *Knudsen* liegen: Name | Vorname | Telefon Hansen | Fred | 1220201 Kaiser | Maik | 2304023
Kreuzprodukt	Verschiedene Relationen werden miteinander verknüpft. Dadurch entsteht eine neue Relation, die aus allen Kombinationen der Tupeln besteht. Jedes Tupel der einen Relation wird mit jedem Tupel der anderen Relation verknüpft. Bei einer Relation **A mit n Tupeln** und einer **Relation B mit m Tupeln** ist das Ergebnis des Kreuzprodukts eine Relation mit **m*n Tupeln.**	Gegeben sind zwei Relationen A und B **A:** Name | Telefon Hansen | 1220201 Kaiser | 2304023 Maier | 2373272 **B:** Abteilung EDV Einkauf Vertrieb

Hinweis:

Das Kreuzprodukt ist die Basis der verschiedenen `JOIN`-Befehle, die in späteren Kapiteln eine wichtige Rolle spielen.

Das Kreuzprodukt sieht dann so aus:

Name	Telefon	Abteilung
Hansen	1220201	EDV
Kaiser	2304023	EDV
Maier	2373272	EDV
Hansen	1220201	Einkauf
Kaiser	2304023	Einkauf
Maier	2373272	Einkauf
Hansen	1220201	Vertrieb
Kaiser	2304023	Vertrieb
Maier	2373272	Vertrieb

Mengenoperationen

Die Mengenoperationen basieren auf den bekannten mathematischen Operationen der Mengenlehre – eine Relation kann im Prinzip wie eine Menge betrachtet werden.[3]

Die wichtigsten Operationen sind:

- Vereinigungsmenge
- Schnittmenge
- Differenzmenge

Hinweis:

Diese Operationen bilden die Basis der SQL-Befehle `UNION` oder `INTERSECT`.

Hinweis:

Diese Operation bildet die Basis des SQL-Befehls `MINUS`.

Gegeben sind zwei Relationen, die die Teilnehmer eines Mathematikkurses und eines Informatikkurses auflisten:

Mathe-Kurs:

Name
König
Hansen
Kaiser
Maier

Info-Kurs:

Name
Knudsen
Laufer
Maier
König

Vereinigung:

Name
König
Hansen
Kaiser
Maier
Knudsen
Laufer

Die Relationen werden *vereinigt* – mehrfaches Auftreten von Tupeln wird verhindert.

Schnitt:

Name
König
Maier

Die Relationen werden *geschnitten* – nur die Tupel, die in beiden Relationen enthalten sind, werden aufgeführt.

Differenz:

Name
Hansen
Kaiser

Die Relationen werden *subtrahiert* – aus der ersten Relation werden alle Tupel entfernt, die auch in der zweiten Relation enthalten sind.

3 Der berühmte Mathematiker Cantor definierte eine Menge so: „Unter einer Menge M verstehen wir jede Zusammenfassung von bestimmten wohlunterschiedenen Objekten unserer Anschauung oder unseres Denkens (welche die Elemente von M genannt werden) zu einem Ganzen.“ Streng genommen ist eine mathematische Relation eine Teilmenge des Kreuzproduktes (kartesischen Produktes) zweier Mengen.

1.4 Sprachbestandteile von SQL

Eine Abfragesprache für Datenbanken hat ganz allgemein drei große Bestandteile – einen Sprachteil, der Daten manipuliert (beispielsweise ändert, einfügt oder löscht), einen Sprachteil, der die Datenstruktur (beispielsweise Tabellen) anlegt und einen Sprachteil, mit dem die Datenbank administriert werden kann (beispielsweise Benutzerrechte vergeben). SQL hat ebenfalls diese drei Bestandteile, die im Detail so aussehen:

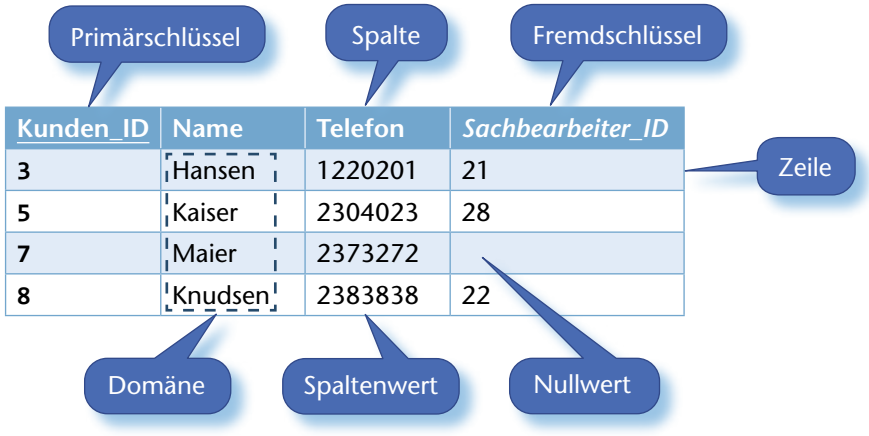
Abbildung 2

In manchen Erläuterungen zu SQL werden die Abfrage der Datenbank und die Steuerung von Transaktionen in eigenen Sprachteilen angeben (**DQL** für **D**ata **Q**uery **L**anguage und **TCL** für **T**ransaction **C**ontrol **L**anguage). Im allgemeinen Standard werden die Abfragen jedoch unter der **DML** und die Transaktionssteuerung unter der **DCL** eingeordnet.

1.5 Grundbegriffe des relationalen Datenmodells

Vor der Erarbeitung einer konkreten Konzeption einer relationalen Datenbank werden noch einige wichtige Grundbegriffe erläutert, die für das Verständnis der Konzeption und der Umsetzung nötig sind. Wie in den vorigen Unterkapiteln bereits angesprochen, basiert ein relationales Datenmodell auf Tabellen. Diese Tabellen haben Spalten, Zeilen, Werte, Domänen und Schlüssel:

Tabelle Kunde:



Kunden_ID	Name	Telefon	*Sachbearbeiter_ID*
3	Hansen	1220201	21
5	Kaiser	2304023	28
7	Maier	2373272	
8	Knudsen	2383838	22

Diese wichtigen Begriffe werden nun kurz dargestellt:

Bezeichnung	Beschreibung	Beispiel
Spalte	Eine Spalte in einer Tabelle beschreibt eine Eigenschaft des Objektes, das in der Tabelle abgebildet werden soll.	Die Tabelle Kunde soll konkrete Kunden speichern. Kunden haben einen Namen, der unter der ***Spalte Name*** eingetragen wird.
Zeile	Eine Zeile in einer Tabelle repräsentiert eines der gespeicherten Objekte.	In der Tabelle Kunde ist der *Kunde Hansen* in einer Zeile gespeichert. Damit sind alle wichtigen Daten des Kunden erfasst.
Spaltenwert	Ein Spaltenwert ist ein konkreter Eintrag unter einer Spalte in einer bestimmten Zeile.	In der Tabelle Kunde hat der *Kunde Hansen* unter der Spalte Telefon den konkreten Eintrag „1220201“.
Domäne	Die Domäne legt fest, welche Werte in einer Spalte eingetragen werden dürfen.	In der Tabelle Kunde muss unter der Spalte Telefon eine Nummer eingetragen werden. Die Domäne ist damit Menge gültiger Telefonnummern – also einer Folge von Ziffern zwischen 0 und 9.
Nullwert	Der Nullwert bezeichnet das Fehlen eines Eintrags in einer Spalte. Er ist nicht zu verwechseln mit der Zahl Null oder eine leeren Zeichenkette.	In der Tabelle Kunde hat der *Kunde Maier* keinen Eintrag in der Spalte *Sachbearbeiter_ID*. Er hat damit einen Nullwert an dieser Stelle. Das bedeutet, dass er (noch) keinen Verweis auf einen für ihn zuständigen Sachbearbeiter hat.
Primär-schlüssel	Der Primärschlüssel identifiziert eine Zeile in der Tabelle eindeutig. Das bedeutet, dass der Primärschlüssel-Eintrag einmalig ist. Der Primärschlüssel kann auch eine Kombination verschiedener Spalten sein, um eindeutig zu werden. Dabei ist die so genannte *Minimaleigenschaft* zu beachten. Sobald ein Schlüssel eindeutig ist, dürfen keine weiteren Spalten hinzugefügt werden. **Hinweis:** Primärschlüssel werden in der Regel durch eine Unterstreichung kenntlich gemacht.	In der Tabelle Kunde ist der Primärschlüssel die **Kunden_ID**. Dieser Wert ist einmalig und identifiziert einen Kunden eindeutig, auch, wenn es mehrere Kunden mit dem Namen „Maier“ gäbe.
Fremd-schlüssel	Von einem Fremdschlüssel spricht man, wenn eine Tabelle eine Spalte enthält, die in einer anderen Tabelle der Primärschlüssel ist. In dieser Spalte werden dann Werte aus dem Primärschlüssel der anderen Tabelle eingetragen. Mithilfe des Fremdschlüssels und des Primärschlüssels können dann zusammengehörige Zeilen miteinander verknüpft werden. **Hinweis:** *Fremdschlüssel* werden oftmals durch Kursivdruck kenntlich gemacht.	In der Tabelle Kunde ist die Spalte Sachbearbeiter_ID ein Fremdschlüssel. In dieser Spalte werden Werte eingetragen, die in einer weiteren Tabelle Sachbearbeiter im Primärschlüssel enthalten sind. Damit erhält der Kunde einen eindeutigen Verweis auf einen Datensatz (Zeile) in der Tabelle Sachbearbeiter. Dem Kunden wird damit eindeutig ein Sachbearbeiter zugeordnet.

1.6 Konzeption einer relationalen Datenbank

Eine relationale Datenbank basiert auf Tabellen (Relationen) und deren Beziehungen untereinander. Anhand eines konkreten Beispiels wird eine solche Konzeption vorgestellt. Dabei wird in der Planung das so genannte **Entity-Relationship-Model ERM** (Objekt-Beziehungs-Modell) eingesetzt, mit dem eine Datenmodellierung für ein gegebenes Problem durchgeführt werden kann. Anschließend wird dieses Modell in Tabellen übersetzt und die Beziehungen zwischen den Tabellen mithilfe des **Primär-Fremdschlüsselkonzeptes** umgesetzt. Die Realisierung in der Datenbank erfolgt dann mit SQL-Befehlen (einem SQL-Skript). Diese Konzeption (Tabellenstruktur) soll dann die Grundlage für die Erarbeitung der SQL-Befehle ab dem zweiten Kapitel sein.

Problemstellung:

Die Softwareentwicklungsfirma ***ProLQS*** realisiert verschiedene Software-Projekte für ihre Kunden. Bislang wurde die Projektverwaltung mithilfe eines Tabellenkalkulations-Programmes organisiert. Aufgrund eines schnellen Wachstums der Firma hat die Geschäftsleitung beschlossen, dass die Organisation in Zukunft mithilfe einer relationalen Datenbank erfolgen soll. In einem ersten Schritt wurde ein ER-Modell entwickelt, das die relevanten Daten erfasst und miteinander in Beziehung setzt. Das ER-Modell liegt in Form eines ER-Diagramms vor.

`ER-Diagramm:`

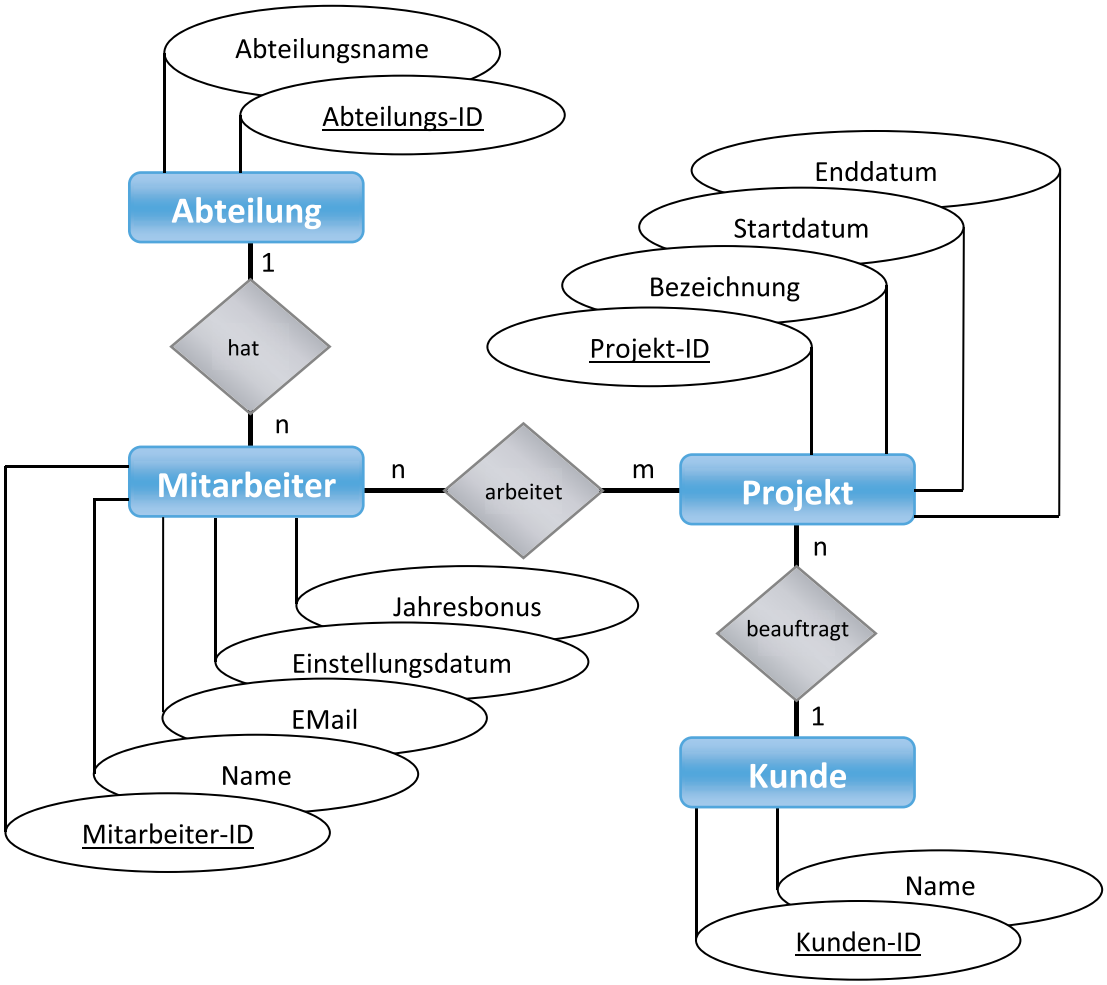


Abbildung 3

Das ER-Modell in Form des ER-Diagramms beschreibt die Problemstellung auf einer abstrakteren Ebene. Ein Mitarbeiter mit seinen persönlichen Daten wie ID, Name oder E-Mail ist jeweils einer (1) Abteilung zugeordnet und kann aber an verschiedenen (m) Projekten mitarbeiten. Ein Projekt kann verschiedene (n) Mitarbeiter einbinden, ist aber genau einem (1) Kunden zugeordnet. Ein Kunde kann hingegen verschiedene (n) Projekte in Auftrag geben. Mitarbeiter, Abteilung, Projekt und Kunde werden in dem Modell als **Entity-Typen** bezeichnet. Die Beziehungen zwischen den Entitäten sind die **Relationships** und werden durch die **Kardinalitäten** (1, n, m) weiter qualifiziert.

Überführung in Tabellen:

Das ***ER-Diagramm*** bietet eine perfekte Vorlage für den Aufbau der Tabellen und deren Beziehungen. In einem ersten Schritt werden die einzelnen Entity-Typen zu Tabellen und deren Attribute zu Spaltennamen. Als Grundlage für die konkreten Abfragen ab Kapitel 2 werden bereits Daten in die Tabellen eingepflegt.

Abteilung:

Abteilungs_ID	Name
100	Web-Design
110	QA
150	Personalabteilung
:	:

Mitarbeiter:

Mitarbeiter_ID	Name	EMail	Einstellungsdatum	Jahresbonus
1	Hansen	Hansen@ProLQS.de	01.10.2015	1800
2	Kaiser	Kaiser@ProLQS.de	01.01.2010	*NULL*
3	Maier	Maier @ProLQS.de	01.04.2019	750
:	:	:	:	:

Projekt:

Projekt_ID	Bezeichnung	Startdatum	Enddatum
1001	ERP-Modul 1	16.09.2025	06.12.2025
1021	CMS Einkauf	06.01.2026	20.03.2026
1029	Paypal-Integration	10.10.2025	17.10.2025
:	:	:	:

Kunde:

Kunden_ID	Name
811	Büro-Treff Langenfeld
822	Zahnklinik Düsseldorf
901	Maklerbüro Schmittke
:	:

Nun müssen die Beziehungen umgesetzt werden. Damit ein Mitarbeiter genau einer Abteilung zugeordnet ist, die Abteilung aber beliebig viele Mitarbeiter haben kann, wird der Primärschlüssel der Abteilungstabelle als Fremdschlüssel in der Mitarbeitertabelle eingefügt. Damit ist die 1:n-Beziehung umgesetzt. Analog wird mit Kunde und Projekt verfahren. Damit sehen die Tabellen Mitarbeiter und Projekt so aus:

Mitarbeiter:

Mitarbeiter_ID	Name	EMail	Einstellungs-datum	Jahres-bonus	*Abteilungs_ID*
1	Hansen	Hansen@ProLQS.de	01.10.2015	1800	***110***
:	:	:	:	:	:

Projekt:

Projekt_ID	Bezeichnung	Startdatum	Enddatum	*Kunden_ID*
1001	ERP-Modul 1	16.09.2025	06.12.2025	***811***
:	:	:	:	:

Zum Schluss bleibt die n:m-Beziehung zwischen Mitarbeiter und Projekten zu klären. Diese Beziehung muss mit einer so genannten **Beziehungstabelle** aufgelöst werden. Das gegenseitige Einfügen von Primärschlüsseln als Fremdschlüsseln würde diese Art der Beziehung nicht abbilden und zu Widersprüchen in den Daten führen. Die Beziehungstabelle nimmt dabei beide Primärschlüssel der Tabellen auf und realisiert dadurch die **n:m-Beziehung**.

Mitarbeiter_Projekt:

MP_ID	*Mitarbeiter_ID*	*Projekt_ID*
1	*1*	*1001*
2	*1*	*1029*
3	:	:

1.7 Implementieren der Beispieldatenbank

Die Datenbankkonzeption aus Kapitel 1.6 dient in den weiteren Kapiteln als Grundlage für die Abfragen. Das Erstellen der Datenbank kann mithilfe des Skriptes im Anhang erfolgen. Das Skript ist im SQL-Standard angelegt und soll in einer **MariaDB-Datenbank** (weiterentwickelte MySQL-Datenbank) implementiert werden. Durch Modifizierungen kann es auch auf anderen Datenbanksystemen laufen (Hinweise dazu sind im Anhang). Die Installation der Datenbank und die Implementierung der Beispieldatenbank mit der Web-Anwendung **phpMyAdmin** wird nun kurz dargestellt:

Schritt 1: Herunterladen und installieren des Softwarepaketes XAMPP

Dieses Programmpaket steht kostenfrei im Internet zur Verfügung. Es beinhaltet den Web-Server ***A**pache,* die Datenbank ***M**ariaDB* sowie ***P**HP* und ***P**erl* als Skriptsprachen.

Die Installation von XAMPP ist einfach gehalten. Das Programm fragt, welche Komponenten installiert werden sollen. Wahlweise können die Skriptsprachen oder weitere Programme eingebunden werden. In der Basisvariante reicht es aus, wenn der Apache-Webserver, die Datenbank MariaDB (***MySQL anklicken)*** und PHP als Skriptsprache installiert werden.

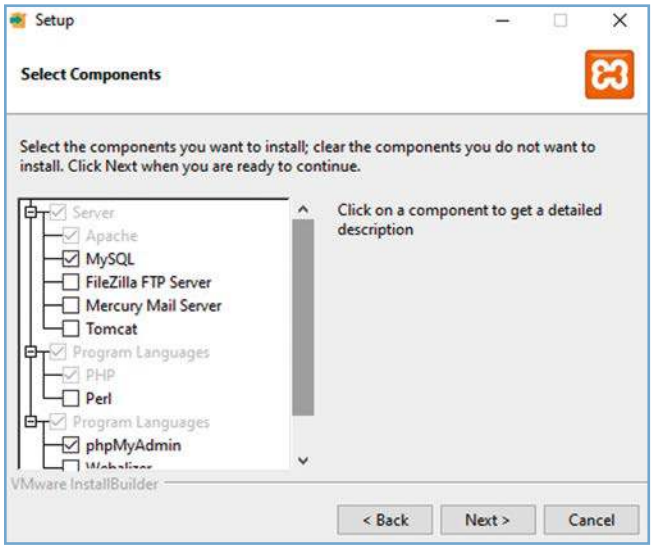


Abbildung 4

Nach der Installation kann das *Control Panel* gestartet werden. Hier können alle Dienste aktiviert oder deaktiviert werden. Für das Nutzen der Datenbank reichen die Aktivierung von Webserver (Apache) und Datenbank (MySQL bzw. MariaDB):

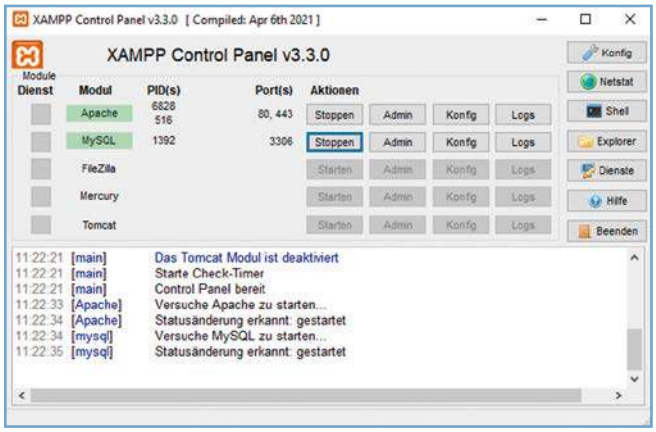


Abbildung 5

Schritt 2: Administration über den Browser

Nach dem erfolgreichen Starten kann die weitere Administration über den Browser erfolgen. Dazu einfach „localhost/“ in der Befehlszeile des Browsers eintragen:

Abbildung 6

Nach einem Klick auf „phpMyAdmin“ öffnet sich eine Web-Anwendung, mit der die Datenbank komfortabel administriert werden kann:

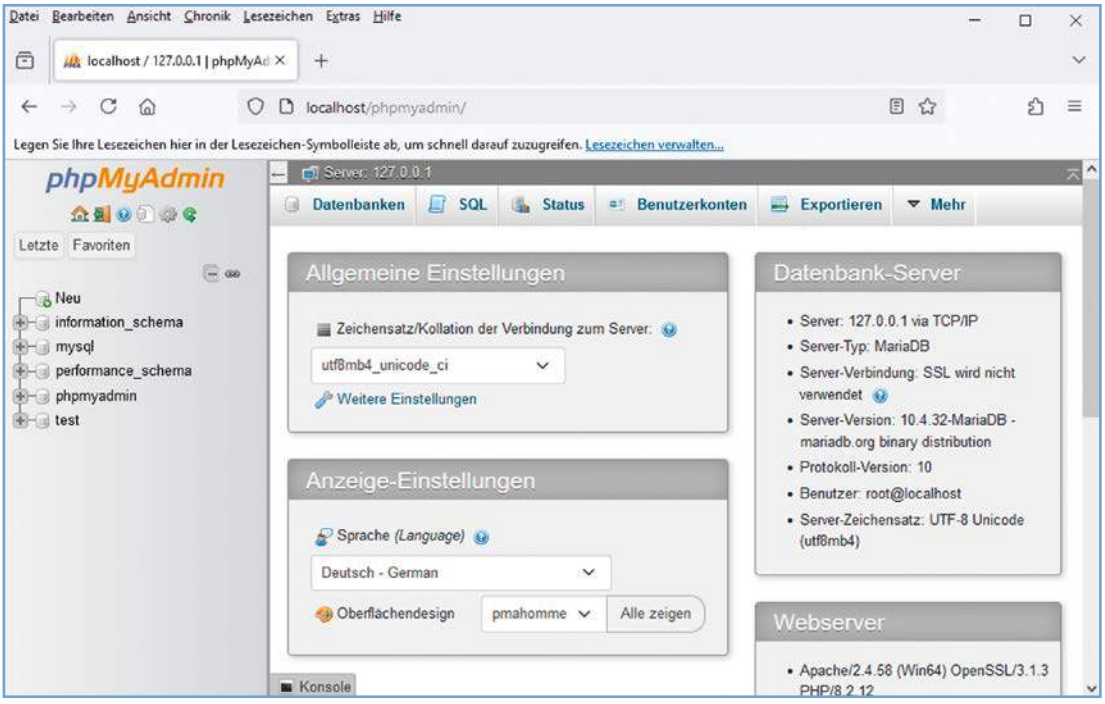


Abbildung 7

Schritt 3: Anlegen einer neuen Datenbank

Über das Register „Datenbanken“ kann dann eine neue Datenbank angelegt werden:

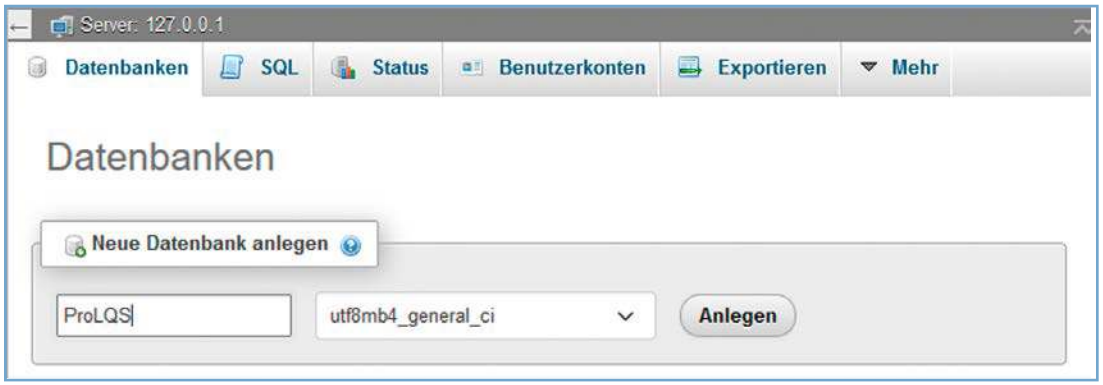


Abbildung 8

Anschließend können Tabellen und Einträge vorgenommen, oder ein vorhandenes Skript eingelesen werden. Für die weiteren Kapitel wird das Skript aus dem Anhang eingelesen, mit dem die Beispiel-Datenbank aus Kapitel 1.6 implementiert wird:

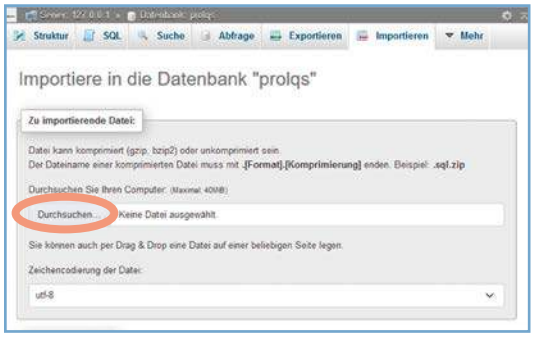


Abbildung 9

Abbildung 10

Nach dem erfolgreichen Einlesen sind die Beispieltabellen korrekt angelegt:

Abbildung 11

Hinweis:

Die Skript-Datei muss im UTF-8-Format gespeichert sein, sonst werden die Umlaute nicht korrekt importiert. Alternativ kann das ganze Skript auch per Copy&Paste in den SQL-Editor von phpMyAdmin kopiert und ausgeführt werden.

1.8 Aufgaben zu Kapitel 1

Aufgabe 1:

Welche der folgenden Aussagen sind korrekt?

- SQL ist eine objektorientierte Programmiersprache.
- SQL bedeutet *Standard Query Language.*
- SQL hat 3 Bereiche (DML, DCL und DDL).
- Die Rechtevergabe für Benutzer erfolgt in SQL mit DDL-Befehlen.
- SQL ist eine deklarative Sprache.
- Eine Domäne legt den Spaltennamen fest.
- Ein Primärschlüssel verweist auf einen Fremdschlüssel einer anderen Tabelle.
- Ein Fremdschlüssel verweist auf einen Primärschlüssel der eigenen oder einer anderen Tabelle.
- Die Minimaleigenschaft eines Primärschlüssels bedeutet, dass maximal zwei Spaltennamen den Primärschlüssel bilden.

Aufgabe 2:

Führen Sie die folgenden Operationen mit den angegebenen Relationen durch.

Relation R1: Mathematik-Grundkurs

Vorname	Nachname
Stefanie	Schellwig
Britta	Weinbrand
Natascha	Herrlich
Sigmund	Kochwein
Roger	Herzlich
Matthias	Naubert
Alexandra	Höflich
Nicole	Huber
Axel	Webster
Stefanie	Werning

Relation R2: Kunst-Leistungskurs

Vorname	Nachname
Britta	Weinbrand
Meike	Hoswig
Natascha	Herrlich
Daniela	Caspar
Matthias	Naubert
Alexandra	Höflich
Daniela	Lohmann
Nicole	Huber
Axel	Webster
Stefanie	Werning

a) `R2 ∩ R1`

b) `R1 ∪ R2 – ( R1 ∩ R2 )`

∩ = Vereinigung
∪ = Schnitt
– = Differenz

2 Daten abfragen mit SELECT

In den folgenden Kapiteln wird in den Erklärungen und Beispielen eine relationale Datenbank abgefragt, die im ersten Kapitel konzipiert wurde. Zur Implementierung der Datenbank kann das Skript im Anhang genutzt werden. Das Skript ist für die relationale Datenbank MariaDB (oder MySQL) konzipiert, ist aber mit kleinen Anpassungen auch unter einer Oracle-Datenbank oder in ACCESS einsetzbar (Hinweise dazu finden sich im Anhang).

Hier sind die Tabellen im Überblick:

Abteilung:

Abteilungs_ID	Name
100	Web-Design
110	QA
150	Personalabteilung
:	:

Mitarbeiter:

Mitarbeiter_ID	Name	EMail	Einstellungsdatum	Jahresbonus	*Abteilungs_ID*
1	Hansen	Hansen@ProLQS.de	01.10.2015	1800	110
2	Kaiser	Kaiser@ProLQS.de	01.01.2010	*NULL*	100
3	Maier	Maier @ProLQS.de	01.04.2019	750	150
:	:	:	:	:	:

Projekt:

Projekt_ID	Bezeichnung	Startdatum	Enddatum	*Kunden_ID*
1001	ERP-Modul 1	16.09.2025	06.12.2025	811
1021	CMS Einkauf	06.01.2026	20.03.2026	901
1029	Paypal-Integration	10.10.2025	17.10.2025	822
:	:	:	:	:

Mitarbeiter_Projekt:

MP_ID	*Mitarbeiter_ID*	*Projekt_ID*
1	*1*	*1001*
2	*1*	*1029*
:	:	:

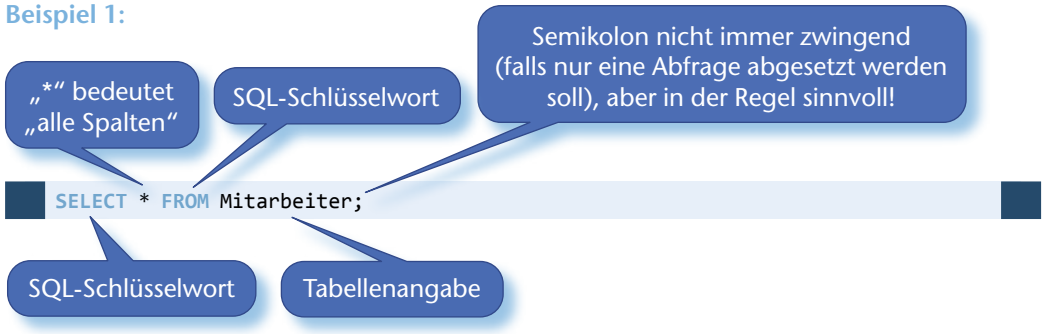
Kunde:

Kunden_ID	Name
811	Büro-Treff Langenfeld
822	Zahnklinik Düsseldorf
901	Maklerbüro Schmittke
:	:

2.1 Aufbau der SELECT-Anweisung

Die SELECT-Anweisung ist die grundlegende Anweisung in SQL und gehört in die Kategorie der Datenmanipulation (**DML**). Die Abfrage startet mit dem Schlüsselwort SELECT. Danach wird angegeben, welche Spalten (alternativ alle Spalten) abgefragt werden sollen (das entspricht der Projektion aus Kapitel 1.3). Zum Schluss wird die Tabelle angegeben, die abgefragt werden soll.

Beispiel 1:



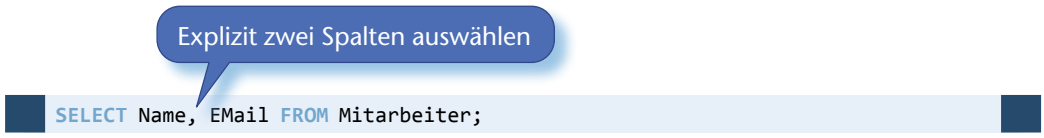
Ergebnis der Abfrage:

Mitarbeiter_ID	Name	EMail	Einstellungsdatum	Jahresbonus	Abteilungs_ID
1	Hansen	Hansen@ProLQS.de	01.10.2015	1800	110
2	Kaiser	Kaiser@ProLQS.de	01.01.2010	*NULL*	100
3	Maier	Maier @ProLQS.de	01.04.2019	750	150
:	:	:	:	:	:

Hinweis:

SELECT und FROM sind Schlüsselworte[4] (SQL-Befehle). Diese und weitere Schlüsselworte werden in den folgenden Kapiteln immer in Großbuchstaben und in blauer Farbe gedruckt. Das ist nicht vorgeschrieben, aber erhöht die Lesbarkeit der Anweisungen.

Beispiel 2:



Ergebnis der Abfrage:

Name	EMail
Hansen	Hansen@ProLQS.de
Kaiser	Kaiser@ProLQS.de
Maier	Maier @ProLQS.de
:	:

4 Eine Übersicht aller Schlüsselworte (reservierte Worte) befindet sich im Anhang.