

Lösungen der Übungsaufgaben

E.1 Lösungen zu Kapitel 1

Übung 1-1 Datenbanksysteme verfügen über folgende Eigenschaften und Fähigkeiten:

- Effiziente Verwaltung großer Datenmengen
- komfortable Anfrageformulierung
- Paralleles Arbeiten ohne Nebeneffekte (Transaktionskonzept)
- Datenunabhängigkeit durch ein Drei-Ebenen-Konzept
- Zugriffskontrolle und Datensicherheit

Übung 1-2 Für die physische Datenunabhängigkeit besteht eine konzeptuelle Sicht auf einen Datenbestand unabhängig von der für die Speicherung der Daten gewählten Datenstruktur. Bei der logischen Datenunabhängigkeit koppelt die Datenbank Änderungen und Erweiterungen von den Anwendungsschnittstellen ab.

Übung 1-3 Klassische Datenbanksystemen eignen sich vorrangig zur Verwaltung vieler Objekte, wobei die zu verwaltenden Objekte relativ wenigen, immer gleich strukturierten Objekttypen angehören. Herkömmliche Datenbanksysteme sind mit sehr heterogenen Objekten, die zu vielen unterschiedlichen Objekttypen gehören, sowie stark strukturierten Objekten überfordert. Ebenso bereitet die Verwaltung semistrukturierter Daten Probleme.

Übung 1-4 Datenbanksysteme kommen in allen Lebensbereichen vor, beispielsweise als Komponente in Vergleichsportalen und Online-Shops. Die Systeme werden vorrangig im Hintergrund eingesetzt und werden meist über eine grafische Benutzeroberfläche verwendet.

E.2 Lösungen zu Kapitel 2

Übung 2-1 Folgende Attributkombinationen können als Schlüssel für die Relation R verwendet werden:

- A
- B, D

Übung 2-2 Die Anfragen lassen sich mittels der relationalen Algebra wie folgt formulieren:

1. Geburtsjahre der Studierenden im dritten Semester:

$$\pi_{\text{Geburtsjahr}}(\sigma_{\text{Semester}=3}(\text{STUDIERENDER}))$$

2. Fächer, bei denen die Studentin Merle Mutig die Prüfung mit einer Note besser als 2.0 abgelegt hat:

$$\pi_{\text{Fach}}(\sigma_{\text{Vorname}='Merle' \wedge \text{Nachname}='Mutig' \wedge \text{Note} < 2.0}(\text{STUDIERENDER} \bowtie \text{MODULPRÜFUNG}))$$

3. Vor- und Nachname der Studenten, die eine Prüfung im Fach Datenbanken, aber nicht im Fach Statistik abgelegt haben:

$$\begin{aligned} &\pi_{\text{Vorname}, \text{Nachname}}(\sigma_{\text{Fach}='Datenbanken'}(\text{STUDIERENDER} \bowtie \text{MODULPRÜFUNG})) - \\ &\pi_{\text{Vorname}, \text{Nachname}}(\sigma_{\text{Fach}='Statistik'}(\text{STUDIERENDER} \bowtie \text{MODULPRÜFUNG})) \end{aligned}$$

Übung 2-3 Die erste Anfrage liefert das folgende Ergebnis:

	Vorname	Name
ERGEBNIS_1	Maria	Meyer
	Tamara	Jagellovsk

Die Anfragen 2 und 3 sind äquivalent und liefern das gleiche Ergebnis, da Vereinigung \cup und logisches Oder \vee das Gleiche ausdrücken:

	Vorname
ERGEBNIS_2_3	Maria
	Hugo
	Tamara
	Kevin
	Merle

Übung 2-4 Die folgende Tabelle gibt einen kurzen Überblick zu den Begriffen und deren informale Bedeutung.

Attribut	Spalte einer Tabelle
Attributwert	Element eines Wertebereichs
Datenbank	Menge von Relationen
Datenbankschema	Menge von Relationenschemata
Fremdschlüssel	Attributmenge, die in einer anderen Relation Schlüssel ist
Fremdschlüsselbedingung	Alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf
Primärschlüssel	Ein beim Datenbankentwurf ausgewählter Schlüssel
Relation	Menge von Zeilen einer Tabelle
Relationenschema	Menge von Attributen
Schlüssel	Minimale Menge von Attributen, deren Werte ein Tupel einer Tabelle eindeutig identifizieren
Tupel	Zeile einer Tabelle
Wertebereich	Mögliche Werte eines Attributs

E.3 Lösungen zu Kapitel 3

Übung 3-1 Ein Datenbankmodell ist ein System von Konzepten zur Beschreibung von Datenbanken. Es legt Syntax und Semantik von Datenbankbeschreibungen für ein Datenbanksystem fest, die sogenannten Datenbankschemata.

Statische Eigenschaften beschreiben die Objekte sowie deren Daten, Datentypen und Beziehungen. Zu den dynamischen Eigenschaften zählen die Operationen auf den Objekten sowie Beziehungen und Abhängigkeiten zwischen Operationen. Integritätsbedingungen werden über Objekten und Operationen definiert, um die Korrektheit der Anwendung zu sichern.

Übung 3-2 Entity-Relationship-Modelle bestehen im Kern aus drei Komponenten:

1. Entites: Die in einer Datenbank zu repräsentierenden Informationseinheiten
2. Beziehungen: Modellieren das Verhältnis zwischen Entities
3. Attribute: Eigenschaften von Entities oder auch Beziehungen

Übung 3-3 Die Intervallnotation gibt an, wie oft ein einzelnes Entity eines Entitätstyps minimal und maximal in Beziehung zu Entities aus einem zweiten Entitytyp steht.

Übung 3-4 Ein weiteres Beispiel für eine abhängige Entity sind Fakultäten, die ohne eine zugeordnete Universität nicht existieren können. Universitäten sind wie Fachhochschulen spezielle Hochschulen.

Löcher auf Golfplätzen können ohne den Golfplatz nicht existieren. Mini-golfplätze sind spezielle Golfplätze.

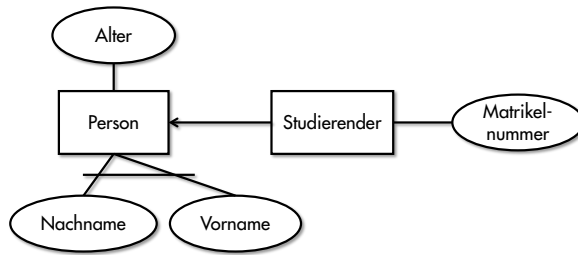
Übung 3-5 Die zweite Variante ist für die eindeutige Identifizierung einer Person am geeignetsten. Variante 1 entfällt, da Vor- und Nachname mehrfach auftreten können. Durch eine ID wie in Variante 2 wird ein eindeutiges Identifizierungsmerkmal festgelegt. In Variante 3 stellen ID und der Name einer Person alternative Schlüssel dar, wodurch der Name auch eindeutig sein muss, was, wie in Variante 1 erklärt, nicht der Fall sein sollte.

Übung 3-6 Die folgende Abbildung E.1 zeigt die ERM's für die einzelnen Teilaufgaben.

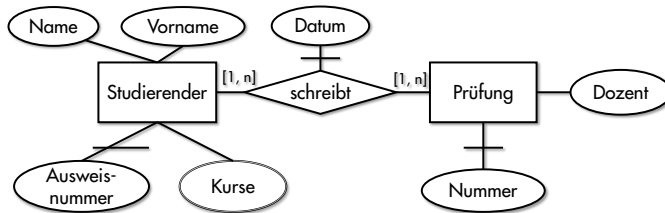
Übung 3-7 Abbildung E.2 zeigt das Entity-Relationship-Modell für die Aufgabenstellung „Band-Info“.

Übung 3-8 Abbildung E.3 zeigt das Entity-Relationship-Modell für die Aufgabenstellung „Firma“.

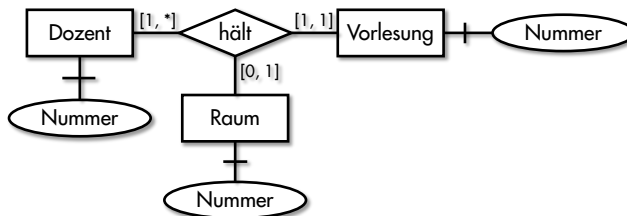
Übung 3-9 Abbildung E.4 zeigt das Entity-Relationship-Modell für die Aufgabenstellung „Biertrinker“.



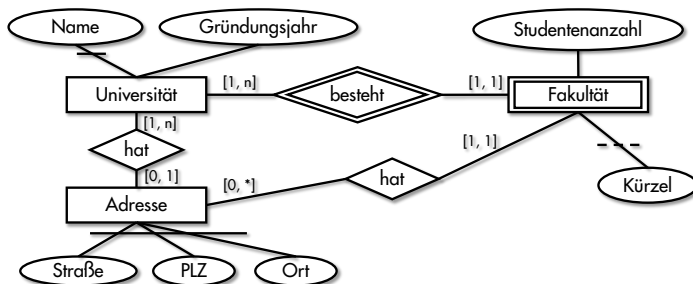
(a) IST-Beziehung



(b) Mehrwertiges Attribut



(c) Ternäre Beziehung



(d) Abhängiger Entity-Typ

Abbildung E.1: ER-Schemata „Universität“

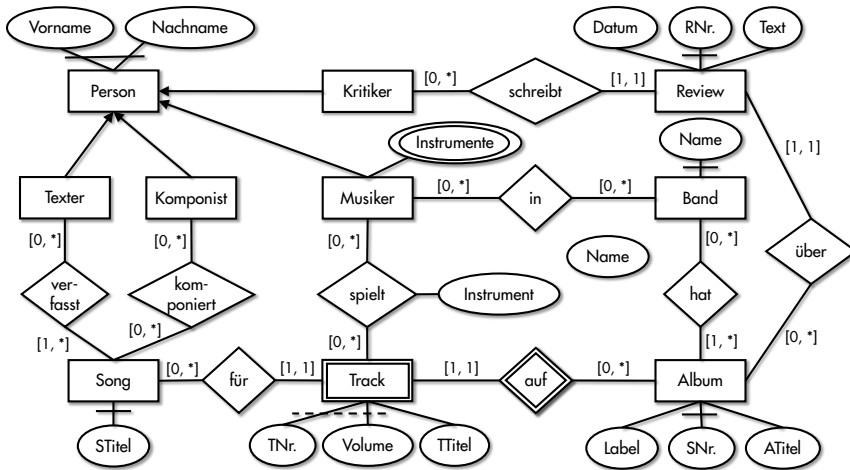


Abbildung E.2: ER-Schema „Band-Info“

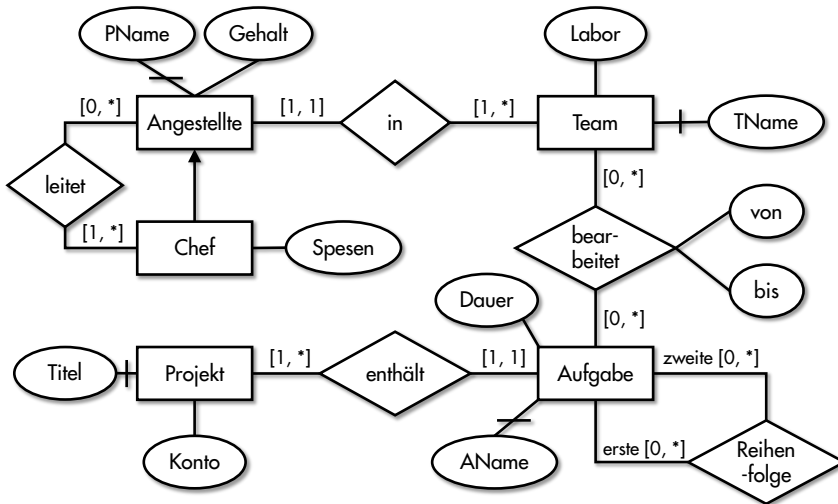


Abbildung E.3: ER-Schema „Firma“

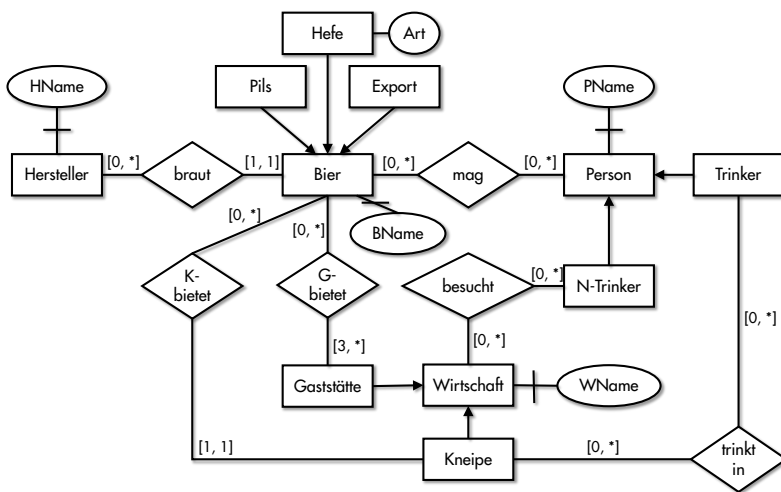


Abbildung E.4: ER-Schema „Biertrinker“

E.4 Lösungen zu Kapitel 4

Übung 4-1 Die fehlenden Phasen sind (von oben nach unten):

- der konzeptionelle Entwurf,
- der logische Entwurf und
- die Datendefinition.

Übung 4-2 Es folgen die Lösungen zu den einzelnen Aufgabenstellungen:

Aufgabenstellung Universität

- **Teilaufgabe IST-Beziehung**
 - $Person(\{Vorname, Nachname, Alter\}, \{\{Vorname, Nachname\}\})$
 - $Studierender(\{Vorname, Nachname, Matrikelnummer\}, \{\{Vorname, Nachname\}\})$
mit $Studierender(Vorname, Nachname)$
→ $Person(Vorname, Nachname)$
- **Teilaufgabe Mehrwertiges Attribut**
 - $Studierender(\{Vorname, Nachname, Ausweisnummer\}, \{\{Ausweisnummer\}\})$
 - $Kurse(\{Ausweisnummer, Kurs\}, \{\{Ausweisnummer, Kurs\}\})$
mit $Kurse(Ausweisnummer) \rightarrow Studierender(Ausweisnummer)$
 - $Prüfung(\{Nummer, Dozent\}, \{\{Nummer\}\})$
 - $schreibt(\{Ausweisnummer, Nummer, Datum\}, \{\{Ausweisnummer, Nummer\}\})$
mit $schreibt(Ausweisnummer) \rightarrow Studierender(Ausweisnummer)$
und $schreibt(Nummer) \rightarrow Prüfung(Nummer)$
- **Teilaufgabe Ternäre Beziehung**
 - $Dozent(\{DNummer\}, \{\{DNummer\}\})$
 - $Raum(\{RNummer\}, \{\{RNummer\}\})$
 - $Vorlesung(\{VNummer, DNummer, RNummer\}, \{\{VNummer\}\})$
mit $Vorlesung(RNummer) \rightarrow Raum(RNummer)$
und $Vorlesung(DNummer) \rightarrow Dozent(DNummer)$
- **Teilaufgabe Abhängiger Entity-Typ**
 - $Universität(\{Name, Gründungsjahr\}, \{\{Name\}\})$
 - $Adresse(\{Straße, PLZ, Ort\}, \{\{Straße, PLZ, Ort\}\})$
 - $hat1(\{Straße, PLZ, Ort, Name\}, \{\{Straße, PLZ, Ort\}\})$
mit $hat1(Straße, PLZ, Ort) \rightarrow Adresse(Straße, PLZ, Ort)$
und $hat1(Name) \rightarrow Universität(Name)$

- *Fakultät*($\{Name, Kürzel, Studentenzahl, Straße, PLZ, Ort\}, \{\{Name, Kürzel\}\})$
mit *Fakultät*(*Name*) \rightarrow *Universität*(*Name*)
und *Fakultät*(*Straße, PLZ, Ort*) \rightarrow *Adresse*(*Straße, PLZ, Ort*)

Aufgabenstellung Band-Info

- *Person*($\{Vorname, Nachname\}, \{\{Vorname, Nachname\}\})$
- *Texter*($\{Vorname, Nachname\}, \{\{Vorname, Nachname\}\})$
mit *Texter*(*Vorname, Nachname*) \rightarrow *Person*(*Vorname, Nachname*)
- *Komponist*($\{Vorname, Nachname\}, \{\{Vorname, Nachname\}\})$
mit *Komponist*(*Vorname, Nachname*)
 \rightarrow *Person*(*Vorname, Nachname*)
- *Musiker*($\{Vorname, Nachname\}, \{\{Vorname, Nachname\}\})$
mit *Musiker*(*Vorname, Nachname*) \rightarrow *Person*(*Vorname, Nachname*)
- *Kritiker*($\{Vorname, Nachname\}, \{\{Vorname, Nachname\}\})$
mit *Kritiker*(*Vorname, Nachname*) \rightarrow *Person*(*Vorname, Nachname*)
- *Song*($\{STitle\}, \{\{STitel\}\})$
- *Album*(*SNr., Label, ATitle, SNr.*)
- *Track*($\{SNr., Volume, TNr., TTitle, SongTitle\}, \{\{SNr., Volume, TNr.\}\})$
mit *Track*(*SNr.*) \rightarrow *Album*(*SNr.*)
und *Track*(*SongTitle*) \rightarrow *Song*(*Title*)
- *Review*($\{RNR., Vorname, Nachname, Datum, Text, SNr.\}, \{\{RNR.\}\})$
mit *Review*(*Vorname, Nachname*) \rightarrow *Kritiker*(*Vorname, Nachname*)
und *Review*(*SNr.*) \rightarrow *Album*(*SNr.*)
- *Band*($\{BName\}, \{\{BName\}\})$
- *Instrumente*($\{Vorname, Nachname, Instrument\}, \{\{Vorname, Nachname, Instrument\}\})$
mit *Instrument*(*Vorname, Nachname*) \rightarrow *Musiker*(*Vorname, Nachname*)
- *verfasst*($\{Vorname, Nachname, STitle\}, \{\{Vorname, Nachname, STitle\}\})$
mit *verfasst*(*Vorname, Nachname*) \rightarrow *Texter*(*Vorname, Nachname*)
und *verfasst*(*STitle*) \rightarrow *Song*(*STitle*)
- *komponiert*($\{Vorname, Nachname, STitle\}, \{\{Vorname, Nachname, STitle\}\})$
mit *komponiert*(*Vorname, Nachname*)
 \rightarrow *Komponist*(*Vorname, Nachname*)
und *komponiert*(*STitle*) \rightarrow *Song*(*STitle*)
- *spielt*($\{Vorname, Nachname, SNr., Volume, TNr., Instrument\}, \{\{Vorname, Nachname, SNr., Volume, TNr.\}\})$
mit *spielt*(*Vorname, Nachname*) \rightarrow *Musiker*(*Vorname, Nachname*)
und *spielt*(*SNr., Volume, TNr.*) \rightarrow *Track*(*SNr., Volume, TNr.*)

- *in*({Vorname, Nachname, BName}, {{Vorname, Nachname, BName}})
mit *in*(Vorname, Nachname) → *Musiker*(Vorname, Nachname)
und *in*(BName) → *Band*(BName)
- *hat*({BName, SNr.}, {{BName, SNr.}})
mit *hat*(BName) → *Band*(BName)
und *hat*(SNr.) → *Album*(SNr.)

Aufgabenstellung Firma

- *Angestellte*({PName, Gehalt, TName}, {{PName}})
- *Chef*({PName, Spesen}, {{PName}})
mit *Chef*(PName) → *Angestellte*(PName)
- *leitet*({APName, CPName}, {{APName}})
mit *leitet*(APName) → *Angestellte*(PName)
und *leitet*(CPName) → *Chef*(PName)
- *Team*({TName, Labor}, {{TName}})
- *Projekt*({PTitel, Konto}, {{PTitel}})
- *Aufgabe*({AName, Dauer, PTitel}, {{AName}})
mit *Aufgabe*(PTitel) → *Projekt*(PTitel)
- *bearbeitet*({TName, AName, von, bis}, {{TName}, {AName}})
mit *bearbeitet*(TName) → *Team*(TName)
und *bearbeitet*(AName) → *Aufgabe*(AName)
- *Reihenfolge*({erster AName, zweiter AName}, {{erster AName, zweiter AName}})
mit *Reihenfolge*(erster AName) → *Aufgabe*(AName)
und *Reihenfolge*(zweiter AName) → *Aufgabe*(AName)

Aufgabenstellung Biertrinker

- *Wirtschaft*({WName}, {{WName}})
- *Gaststätte*({WName}, {{WName}})
mit *Gaststätte*(WName) → *Wirtschaft*(WName)
- *Kneipe*({WName, BName}, {{WName}})
mit *Kneipe*(WName) → *Wirtschaft*(WName)
und *Kneipe*(BName) → *Bier*(BName)
- *Person*({PName, Alter}, {{PName}})
- *Trinker*({PName}, {{PName}})
mit *Trinker*(PName) → *Person*(PName)
- *Nichttrinker*({PName}, {{PName}})
mit *Nichttrinker*(PName) → *Person*(PName)

- $Hersteller(\{HName\}, \{\{HName\}\})$
- $Bier(\{BName, HName, Vol\% \}, \{\{BName\}\})$
mit $Bier(HName) \rightarrow Hersteller(HName)$
- $Pils(\{BName\}, \{\{BName\}\})$
mit $Pils(BName) \rightarrow Bier(BName)$
- $Export(\{BName\}, \{\{BName\}\})$
mit $Export(BName) \rightarrow Bier(BName)$
- $Hefe(\{BName\}, \{\{BName\}\})$
mit $Hefe(BName) \rightarrow Bier(BName)$
- $besucht(\{WName, PName\}, \{\{WName, PName\}\})$
mit $besucht(WName) \rightarrow Wirtschaft(WName)$
und $besucht(PName) \rightarrow Nichttrinker(PName)$
- $trinkt_in(\{WName, PName\}, \{\{WName, PName\}\})$
mit $trinkt_in(WName) \rightarrow Kneipe(WName)$
und $trinkt_in(PName) \rightarrow Trinker(PName)$
- $mag(\{PName, BName\}, \{\{PName, BName\}\})$
mit $mag(PName) \rightarrow Person(PName)$
und $mag(BName) \rightarrow Bier(PName)$
- $Gbietet(\{WName, BName\}, \{\{WName, BName\}\})$
mit $Gbietet(WName) \rightarrow Gastst\ddot{a}tte(WName)$
und $Gbietet(BName) \rightarrow Bier(BName)$

Übung 4-3 Es ergeben sich folgende Relationenschemata aus den gegebenen ERMs:

- **IST-Beziehung**
 - $A(\{a, b, c\}, \{\{a\}\})$
 - $B(\{a, d\}, \{\{a\}\})$
mit $B(a) \rightarrow A(a)$
- **Mehrwertiges Attribut**
 - $A(\{a, c\}, \{\{a\}\})$
 - $Ab(\{a, b\}, \{\{a, b\}\})$
mit $Ab(a) \rightarrow A(a)$
 - $B(\{a, d, e\}, \{\{a, e\}\})$
mit $B(a) \rightarrow A(a)$
und $B(e) \rightarrow C(e)$
 - $C(\{e, h, g\}, \{\{e\}, \{h\}\})$

- Ternäre Beziehung
 - $A(\{a, b, c\}, \{\{a\}\})$
mit $A(b) \rightarrow B(b)$
und $A(c) \rightarrow D(c)$
 - $D(\{c\}, \{\{c\}\})$
 - $B(\{b\}, \{\{b\}\})$
- Abhängiger Entity-Typ
 - $A(\{a\}, \{\{a\}\})$
 - $C(\{a, b\}, \{\{a, b\}\})$
mit $C(a) \rightarrow A(a)$
 - $D(\{c\}, \{\{c\}\})$
 - $E(\{c, a, b\}, \{\{c\}\})$
mit $E(c) \rightarrow D(c)$
und $E(a, b) \rightarrow C(a, b)$

E.5 Lösungen zu Kapitel 5

Übung 5-1 Die erste Normalform erlaubt nur atomare Attribute in den Relationenschemata. Für die Erfüllung der zweiten Normalform dürfen keine partiellen Abhängigkeiten zwischen Schlüsseln des Relationenschemas und weiteren Attributen vorliegen. Zur Einhaltung der dritten Normalform dürfen zudem keine transitiven Abhängigkeiten in einem Relationenschema existieren.

Übung 5-2 Die in den Aufgaben 4.2 und 4.3 entwickelten Datenbankschemata sollten die dritte Normalform erfüllen. Die meisten partiellen und transitiven Abhängigkeiten sollten bereits beim Entwerfen der ERMs bzw. bei der Transformation in das relationale Modell entdeckt worden sein. Ein nachträgliches Prüfen, insbesondere unter Einbindung eines Domänenexperten, ist aber unabdingbar für einen guten Datenbankentwurf.

Übung 5-3 Sind nur die Relationenschemata gegeben, nicht jedoch die Relationen, so wird davon ausgegangen, dass die erste Normalform implizit erfüllt ist. Die zweite / Normalform ist in den gegebenen Schemata

- erfüllt / erfüllt,
- erfüllt / nicht erfüllt,
- nicht erfüllt / nicht erfüllt,
- erfüllt / erfüllt und
- erfüllt / erfüllt.

Übung 5-4 Die Abhängigkeitstreue ist nicht gegeben, da z.B. die funktionale Abhängigkeit $\{A\} \rightarrow \{B, C\}$ nicht durch die Schlüsselabhängigkeiten darstellbar ist.

Übung 5-5 Durch die Verbundtreue kann eine Originalrelation aus den aus ihr entstandenen zerlegten Relationen durch den natürlichen Verbund zurückgewonnen werden.

Übung 5-6 Die Verbundtreue ist für die Teilaufgaben erfüllt:

- erfüllt,
- erfüllt,
- nicht erfüllt,
- nicht erfüllt und
- nicht erfüllt.

Übung 5-7 Mögliche kleinere Datenbankschemata sind:

1. $\{(\{A, B, C, D, E\}, \{\{A, D\}\}), (\{D, F\}, \{\{D\}\})\}$,
2. $\{(\{A, B, C\}, \{\{A\}\}), (\{A, D, E, F\}, \{\{A, D\}\})\}$,
3. $\{(\{A, B, C, D, F\}, \{\{A, D\}\}), (\{A, D, E\}, \{\{A, D\}\})\}$ und
4. $\{(\{A, B, C, D, E, F\}, \{\{A, D\}\})\}$.

Alle Schemata erfüllen nicht die zweite Normalform. Dies liegt jeweils an den folgenden funktionalen Abhängigkeiten:

1. Partielle Abhängigkeit $A \rightarrow BC$ im ersten Relationenschema,
2. partielle Abhängigkeit $D \rightarrow DF$ im zweiten Relationenschema,
3. partielle Abhängigkeit $A \rightarrow BC$ im ersten Relationenschema sowie
4. den beiden obigen partiellen Abhängigkeiten im alleinigen Relationenschema.

Übung 5-8 Folgende Eigenschaften sind in den einzelnen Datenbankschemata gegeben:

Schema-#	3. NF	Abhängigkeitstreue	Verbundtreue	Minimalität
1	nein	nein	ja	nein
2	ja	nein	ja	nein
3	ja	ja	ja	ja
4	nein	nein	nein	nein

E.6 Lösungen zu Kapitel 6

Übung 6-1 Die SQL-Anweisungen werden wie folgt formuliert:

```
create table MITARBEITERIN (  
    Name varchar(100),  
    Geburtstag date not null,  
    Gehalt numeric(8,2),  
    primary key (Name)  
)  
  
create table FIRMA (  
    Firmenname varchar(100),  
    Handelsregisternummer integer,  
    primary key (Firmenname),  
    unique (Handelsregisternummer)  
)  
  
create table ABTEILUNG (  
    Firmenname varchar(100),  
    Abteilungsname varchar(100),  
    Budget numeric(10,2) not null,  
    primary key (Firmenname, Abteilungsname),  
    foreign key (Firmenname) references Firma(Firmenname)  
)  
  
create table ARBEITET_IN (  
    Name varchar(100),  
    Firmenname varchar(100),  
    Abteilungsname varchar(100),  
    primary key (Name, Firmenname),  
    foreign key (Firmenname, Abteilungsname)  
        references ABTEILUNG(Firmenname, Abteilungsname),  
    foreign key (Name) references MITARBEITERIN(Name)  
)
```

Übung 6-2 Für das Datenbankschema „Universität“ ergeben sich folgende **create table**-Anweisungen:

- Teilaufgabe IST-Beziehung

```
create table PERSON (  
    Vorname varchar(50),  
    Nachname varchar(50),  
    Alter integer not null,  
    primary key (Vorname, Nachname)  
)
```

```

create table STUDIERENDER (
    Vorname varchar(50),
    Nachname varchar(50),
    Matrikelnummer integer not null,
    primary key (Vorname, Nachname),
    foreign key (Vorname, Nachname)
        references PERSON(Vorname, Nachname)
)

```

- Teilaufgabe Mehrwertiges Attribut

```

create table STUDIERENDER (
    Vorname varchar(50) not null,
    Nachname varchar(50) not null,
    Ausweisnummer integer,
    primary key (Ausweisnummer)
)

```

```

create table KURSE (
    Ausweisnummer integer,
    Kurs varchar(100),
    primary key (Ausweisnummer, Kurs),
    foreign key (Ausweisnummer)
        references STUDIERENDER(Ausweisnummer)
)

```

```

create table PRÜFUNG (
    Nummer integer,
    Dozent varchar(100) not null,
    primary key (Nummer)
)

```

```

create table SCHREIBT (
    Ausweisnummer integer,
    Nummer integer,
    Datum date not null,
    primary key (Ausweisnummer, Nummer),
    foreign key (Ausweisnummer)
        references STUDIERENDER(Ausweisnummer),
    foreign key (Nummer) references PRÜFUNG(Nummer)
)

```

- Teilaufgabe Ternäre Beziehung

```

create table DOZENT (
    DNummer integer primary key
)

```

```

create table RAUM (
    RNummer integer primary key
)

create table VORLESUNG (
    VNummer integer,
    RNummer integer,
    DNummer integer,
    primary key (VNummer),
    foreign key (DNummer) references DOZENT(DNummer),
    foreign key (RNummer) references RAUM(RNummer)
)

```

- Teilaufgabe Abhängiger Entity-Typ

```

create table UNIVERSITÄT (
    Name varchar(100),
    Gründungsjahr date not null,
    primary key (Name)
)

create table ADRESSE (
    Straße varchar(50),
    PLZ char(5),
    Ort varchar(50),
    primary key (Straße, PLZ, Ort)
)

create table HAT1 (
    Straße varchar(50),
    PLZ char(5),
    Ort varchar(50),
    Name varchar(100),
    primary key (Name),
    unique (Straße, PLZ, Ort),
    foreign key (Name) references UNIVERSITÄT(Name),
    foreign key (Straße, PLZ, Ort)
        references ADRESSE(Straße, PLZ, Ort)
)

```



```

create table FAKULTÄT (
    Name varchar(100),
    Kürzel varchar(3),
    Studentenzahl integer not null,
    Straße varchar(50),
    PLZ char(5),
    Ort varchar(50),
    primary key (Name, Kürzel),
    foreign key (Name) references UNIVERSITÄT(Name),
    foreign key (Straße, PLZ, Ort)
        references ADRESSE(Straße, PLZ, Ort)
)

```

E.7 Lösungen zu Kapitel 7

Übung 7-1 Folgende vier Kriterien gelten für Anfragesprachen:

- **Ad-hoc-Formulierung:** Der Benutzer soll eine Anfrage formulieren können, ohne ein vollständiges Programm schreiben zu müssen.
- **Deskriptivität:** Der Benutzer soll formulieren, *Was will ich haben?*, und nicht, *Wie komme ich an das, was ich haben will?*
- **Weitere Kriterien:** z.B. Mengenorientiertheit, Abgeschlossenheit, ...

Übung 7-2 Die SQL-Anfragen auf dem Beispiel „Hotel“ lautet wie folgt:

1. Welche Hotels verfügen über Einzel- oder Doppelzimmer?

```
select distinct HName from zimmer
where zimmer.ztyp= „EZ“
or zimmer.ztyp= „DZ“
```

2. Welche Hotels verfügen über Einzel- und Doppelzimmer?

```
select distinct HName
from zimmer
where zimmer.ztyp= „DZ“
and HName in (
    select distinct HName
    from zimmer
    where zimmer.ztyp= „EZ“
)
```

3. Welche Hotels verfügen nur über Einzelzimmer?

```
select distinct HName
from zimmer
where zimmer.ztyp= „EZ“
and HName not in (
    select distinct HName
    from zimmer
    where zimmer.ztyp != „EZ“
)
```

4. Welche Ausstattung bietet Hotel Neptun zu welchem Preis an?

```
select distinct AName, Preis
from bietet
where HName= „Neptun“
```

5. Was kosten Einzelzimmer in 5-Sterne-Hotels?

```
select distinct hotel.HName, Preis
from hotel
  join zimmertyp on (hotel.HName = zimmertyp.HName)
where kategorie = 5
and ztyp= „EZ“
```

6. In welchen Hotels hat Franz Mueller reserviert?

```
select distinct Hname
from reserviert
where Pname= „Mueller“
and Pvorname= „Franz“
```

7. In welchen 2-Sterne-Hotels hat Franz Mueller Doppelzimmer reserviert?

```
select distinct hotel.Hname
from hotel
  join reserviert on (hotel.HName = reserviert.HName)
where Kategorie = 2
and zimmertyp.ztyp= „DZ“
and Pname= „Mueller“
and Pvorname= „Franz“
```

8. In welchen Rostocker Hotels hat Franz Mueller Zimmer für eine Nacht gebucht?

```
select distinct hotel.Hname
from reserviert natural join hotel
where Pname= „Mueller“
and Pvorname = „Franz“
and Ort = „Rostock“
and bis-von = 1
```

9. Welche Gäste haben noch offene Rechnungen?

```
select distinct pvorname, pname
from rechnung
where bezahlt = false
```

10. In welchen Hotels wurden alle gestellten Rechnungen bezahlt?

```
select distinct Hname
from hotel
where HName not in (
  select Hname
  from rechnung
  where bezahlt = false
)
```

11. Welche Personen sind keine Gäste?

```
select Pvorname, Pname
from person
where (Pvorname, Pname) not in (
    select Pvorname, Pname
    from gast
)
```

12. Welches Hotel bietet das billigste Zimmer an?

```
select HName
from zimmertyp
where Preis = (
    select min(Preis) as mini
    from zimmertyp
)
```

13. Welches Hotel bietet die meisten Einzelzimmer an?

```
select HName
from zimmertyp
where anzahl = (
    select max(anzahl) as maxi
    from zimmertyp
    where Ztyp = „EZ“
)
```

14. Welches Hotel bietet die durchschnittlich billigsten Zimmer an?

```
select HName
from zimmertyp
group by Hname
having avg(Preis) = (
    select min(avgPreis)
    from (
        select Hname, avg(Preis) as avgPreis
        from zimmertyp
        group by Hname
    )
)
```

15. Wer hat die höchste Hotelrechnung?

```
select Pvorname, Pname
from rechnung
where Summe >= (
    select max(Summe)
    from Rechnung
)
```

16. Wer hat mehr als eine Hotelrechnung bezahlt?

```
select Pvorname, Pname, count(bezahlt) as cnt
from rechnung
where bezahlt = 'Y'
group by Pvorname, Pname
having cnt > 1
```

17. Welche Hotels bieten dieselben Zimmertypen wie das Hotel Huebner?

```
select distinct HName
from zimmertyp
where (HName, ztyp) not in (
    select HName, ztyp
    from zimmertyp
    where (HName, ztyp) not in (
        select distinct HName, huebner.ztyp
        from zimmertyp, (
            select distinct ztyp
            from zimmertyp
            where Hname = „Huebner“
        ) as huebner
    )
)
and HName != „Huebner“
```

Übung 7-3 Die SQL-Anfragen auf dem Beispiel „Maler“ lautet wie folgt:

1. Welche Bilder (Titel) hängen in der Galerie „Alte Meister“ in Dresden?

```
select distinct Titel
from Bild
    join Galerie using (GalerieName)
where Ort = „Dresden“
and GalerieName = „Alte Meister“
```

2. Wessen (MalerName) Titel hängen in Dresden, aber nicht in Berlin?

```
select distinct MalerName
from Bild
    join Galerie using (GalerieName)
where Ort = „Dresden“
and MalerName not in (
    select MalerName
    from Bild
        join Galerie using (GalerieName)
    where Ort = „Berlin“
)
```

3. Welche Maler wurden älter als 60 Jahre? (Hinweis: Zur Vereinfachung der Altersberechnung ziehen wir nur die Jahresangaben heran.)

```
select distinct MalerName
from Maler
where Todesjahr-Gebjahr > 60
```

4. Von welchen Malern hängen in mindestens zwei Galerien Bilder?

```
select distinct MalerName
from Bild
group by MalerName
having count (distinct GalerieName) >= 2
```

5. Wann wurde das älteste Bild gemalt?

```
select min(Jahr)
from Bild
```

6. Wann wurde das zweitälteste Bild gemalt? (Hinweis: Wir nehmen an, dass jedes Jahr nur einmal vorkommt.)

```
select min(Jahr)
from Bild
where Jahr > (
    select min(Jahr)
    from Bild
)
```

Übung 7-4 Die SQL-Anfragen auf dem Beispiel „Werkstatt“ lautet wie folgt:

1. Welche Artikel (Name) lagern in Rostock oder Laage?

```
select distinct Name
from Lager
natural join Artikel
where Ort in („Rostock“, „Laage“)
```

2. Welche Artikel (Namen) gibt es in keinem Lager?

```
select distinct Name
from Artikel
where ArtNr not in (
    select distinct ArtNr
    from Lager
)
```

Übung 7-5 Die Ergebnisrelationen sehen wie folgt aus:

ERGEBNIS_1	A	B	C
	2	4	5
	2	7	8
	2	7	2

	B
ERGEBNIS_2	4
	7

	A	C
	2	8
ERGEBNIS_3	1	5
	2	5
	2	2

	D
ERGEBNIS_4	5

	B	
ERGEBNIS_5	4	1
	7	2

E.8 Lösungen zu Kapitel 8

Übung 8-1 Die folgende SQL-Anweisung erzeugt die Sicht DURCHSCHNITTSPREIS mit den Durchschnittspreisen über alle Zimmertypen pro Hotel:

```
create view DURCHSCHNITTSPREIS as (
  select HName, avg(Preis) as Schnitt
  from ZIMMERTYP
  group by HName
)
```

Übung 8-2 Die folgende SQL-Anweisung gibt der Gruppe GAST das Recht, die zuvor erzeugte Sicht anzufragen.

```
grant select
on DURCHSCHNITTSPREIS
to group GAST
```

Übung 8-3 Das Ändern auf dieser Sicht ist nicht sinnvoll, da diese einen aggregierten Wert enthält. Die Änderungen können nicht ohne Weiteres auf die Basisrelationen übertragen werden.

Ein Einfügen ist ebenso unsinnig, da sonst die Basisrelationen, sofern überhaupt möglich, mit Nullwerten gefüllt werden müssten. Durch das Löschen eines Tupels aus der Sicht müsste entweder die Basisrelation ZIMMERTYP mit verändert werden oder es käme zu Inkonsistenzen zwischen Sicht und den Basisrelationen.

Übung 8-4 Die folgende SQL-Anweisung erlaubt dem Chef, seine Rechte an der Sicht ZIMMERSTATISTIK weiterzugeben. Dazu wird das bestehende Recht um die **with grant option**-Klausel erweitert:

```
grant select
on ZIMMERSTATISTIK
to Chef
with grant option
```

Übung 8-5 Folgende Attributkombinationen zählen zu personenbezogenen Daten nach dem Bundesdatenschutzgesetz:

- Geburtsdatum, Geschlecht und Studiengang aller Rostocker Studenten
- IBAN und BIC
- Straße, Postleitzahl, Wohnort und Telefonnummer aller Rostocker Professoren
- Vorname aller studentischen Hilfskräfte am Lehrstuhl DBIS

Übung 8-6 Für die Sicht MITARBEITER_WEB gehen wir davon aus, dass die Informationen für alle einsehbar sein sollen:

```
grant select
on MITARBEITER_WEB
to public
```

Der Prorektor Studium und Lehre (MNr. = 4711) darf sich die Durchschnittsnoten pro Studiengang anschauen, nicht aber Einzelnoten der Studierenden:

```
grant select
on NOTEN_STUDIENGANG
to 4711
```

Der Studiendekan der Fakultät Ingenieurwissenschaften (MNr. = 1234) darf das nur für die Studiengänge seiner Fakultät sehen:

```
grant select
on NOTEN_ING_FAK
to 1234
```

Jeder Studierende darf seine eigenen Noten sehen, hier die Studentin Merle Mutig:

```
grant select
on NOTEN_MERLE
to 19367
```


Übung 8-7 Das Lösen dieser Aufgabe an einem selbstgewählten Datenbestand sei dem Leser überlassen.

E.9 Lösungen zu Kapitel 9

Übung 9-1 Statische Integritätsbedingungen sind Bedingungen, die Datenbankzustände einschränken. Sie werden unabhängig vom zeitlichen Ablauf der Datenbankänderungen betrachtet.

Übung 9-2 Durch Trigger können dynamische Integritätsbedingungen realisiert werden. Zudem lassen sich komplexe Anwendungsaufgaben ereignisorientiert und zentral im Datenbanksystem sowie rekursive Anfragen umsetzen. Dazu zählen die Typ-, Schlüssel- und referentielle Integrität.

Übung 9-3 Beispiele für dynamische Integritätsbedingungen, sind das Einfügen, Löschen und Ändern von Tupeln und Werten in einer Relation.

Übung 9-4 Trigger können ggf. nicht terminieren. Dies tritt z.B. dann auf wenn ein Trigger ein neues Tupel erzeugt, dadurch aber ein zweiter Trigger aufgerufen wird, der dieses Tupel wieder löscht, wodurch wieder der erste Trigger aufgerufen wird. Zudem kann die Abarbeitungsfolge von parallel ausgeführten Triggern nicht sichergestellt werden.

Übung 9-5 Der Trigger lässt sich wie folgt formulieren:

```
create trigger TRIGGER_EINSCHREIBUNG
after insert on EINSCHREIBUNG
referencing new as inserted
when (
    exists (
        select count(*)
        from EINSCHREIBUNG
        where Vorlesungs_ID = inserted.Vorlesungs_ID
        having count(*) > 100
    )
)
begin atomic
    rollback;
end
```

Statt dem **rollback** kann auch das neu eingefügte Tupel **inserted** mittels **delete** wieder gelöscht werden. Soll die Prüfung bereits vor Einfügen des Tupels realisiert werden, so sieht der Trigger wie folgt aus:

```

create trigger TRIGGER_EINSCHREIBUNG_2
before insert on EINSCHREIBUNG
referencing new as inserted
when (
    exists (
        select count(*)
        from EINSCHREIBUNG
        where Vorlesungs_ID = inserted.Vorlesungs_ID
        having count(*) = 100
    )
)
begin atomic
    do instead nothing;
end

```

E.10 Lösungen zu Kapitel 10

Übung 10-1 Für den schnellen Zugriff auf die Rechnungen eines Gastes legen wir einen Index auf dem Attribut GNachname an, welches wir zudem aufsteigend sortieren:

```

create index ridx
on RECHNUNG (
    GNachname asc
)

```

Übung 10-2 Der B-Baum eignet sich als Indexstruktur für alle Relationen über dem jeweiligen Primärschlüssel. Diese Struktur eignet sich speziell für Punktanfragen, bei denen gezielt nach einzelnen Tupeln gesucht wird. Der Bitmax-Index eignet sich vor allem für Attribute mit nur wenigen möglichen Werten, wie dem Semester der Studierenden oder dem Fach einer Modulprüfung. Hash-Tabellen eignen sich speziell für Daten mit wenig Dynamik, wie beispielsweise die Zuordnung eines Mitarbeiters zu einem Raum.

Übung 10-3 Als Dateiorganisationsformen existieren neben dem Heap sequenzielle, Hash-organisierte und baumartige Speicherverfahren für Relationen. Bei den Zugriffspfaden unterscheidet man in Verfahren für Primär- und Sekundärschlüssel, ein- und mehrdimensionale Verfahren, sowie in statische und dynamische Verfahren.

Dynamische Verfahren ermöglichen schnelle Zugriffe bei sich häufig ändernden Daten. Die Nachteile liegen in aufwendigeren Änderungsoperationen und einem erhöhten Speicherplatzbedarf.

Übung 10-4 Eine Transaktion ist eine Folge von Operationen, die die Datenbank von einem konsistenten Zustand in einen konsistenten, eventuell veränderten Zustand überführt, wobei das ACID-Prinzip eingehalten werden muss. Zu den ACID-Eigenschaften zählen die Atomarität, die Konsistenz, die Isolation und die Dauerhaftigkeit.

Übung 10-5 Zur Wahrung der Atomarität erfolgt für nicht abgeschlossene Transaktionen ein **UNDO**. Dabei werden in umgekehrter Reihenfolge alle *Before image*-Einträge der jeweiligen Transaktion zurückgespeichert.

Die Dauerhaftigkeit erfolgreich abgeschlossener Transaktionen, das heißt, wenn sich im Log ein Commit-Eintrag befindet, kann durch ein **REDO** (Wiederholen) sichergestellt werden. Dazu erfolgt ein Auslesen aller *After images* und deren Einbringen in die Datenbank.

E.11 Lösungen zu Kapitel 11

Übung 11-1 Die Datenbank sieht mit dem Snowflake-Modellierungsstil wie folgt aus:

TRACKS					
Titel	AlbumID	GenreID	TagID	Länge	Track#
Worried Life Blues	A1	G1	D1	4:16	8
I Want To Break Free	A2	G2	D2	4:15	5
I Want It All	A2	G2	D3	4:02	4
Child In Time	A3	G2	D4	12:15	2
Highway Star	A3	G2	D5	6:43	1
Answers	A4	G3	D6	7:08	13
Dragonsong	A4	G3	D7	5:43	8
Torn from the Heavens	A5	G3	D8	4:24	14
Metal	A5	G3	D9	5:30	58
Piece of Mind	A6	G3	D10	2:52	5

ALBEN	AlbumID	Album
	A1	Riding with the King
	A2	Greatest Hits II
	A3	Made in Japan
	A4	Distant Worlds
	A5	Heavensward
	A6	The Far Edge of Fate

GENRES	GenreID	Genre
	G1	Blues
	G2	Rock
	G3	OST

TAGE	TagID	Tag	MonatID
	D1	19	M1
	D2	19	M2
	D3	02	M3
	D4	03	M4
	D5	14	M5
	D6	25	M5
	D7	20	M6
	D8	02	M7
	D9	23	M8
	D10	25	M9

MONATE	MonatID	Monat	JahrID
	M1	06	Y1
	M2	04	Y1
	M3	05	Y1
	M4	03	Y2
	M5	03	Y3
	M6	07	Y4
	M7	04	Y4
	M8	05	Y4
	M9	03	Y4

JAHRE	JahrID	Jahr
	Y1	2020
	Y2	2017
	Y3	2018
	Y4	2019

Übung 11-2 Die Anfragen lassen sich wie folgt formulieren:

1. Letztes Musikstück pro Album und CD:

```
select Album, CD, max(track#) as LetztesStück
from TRACKS
  natural join ALBEN
group by Album, CD
```

2. Durchschnittliche Länge der Musikstücke pro Jahr und pro Album im jeweiligen Jahr:

```
select Jahr, Album, avg(length) as Schnitt
from TRACKS
  natural join ALBEN
  natural join JAHRE
group by rollup(Jahr, Album)
order by Jahr, Album
```

3. Speicherplatz der Musikstücke insgesamt, pro Album, pro Jahr, pro Komponist, pro Album+Jahr, pro Album+Komponist, pro Jahr+Komponist und pro Album+Jahr+Komponist:

```
select Jahr, Komponist, SAlbum, sum(filesize) as Speicher
from TRACKS
    natural join KOMPONIST
    natural join JAHRE
    natural join ALBEN
group by cube(Jahr, Komponist, Album)
order by Jahr, Komponist, Album
```

E.12 Lösungen zu Kapitel 12

Übung 12-1 Für die 13.000 Studenten werden bei der zeilenorientierten Speicherung insgesamt 130.000 kB Speicher benötigt. Dies entspricht knapp 127 MB, also 127 Seiten je 1 MB.

In der zeilenorientierten Speicherung benötigen die Attribute Matrikelnummer und Geburtsdatum 260.000 Bytes, also ≈ 254 kB bzw. $\approx 0,25$ MB. Es wird somit nur eine Seite zur Speicherung benötigt.