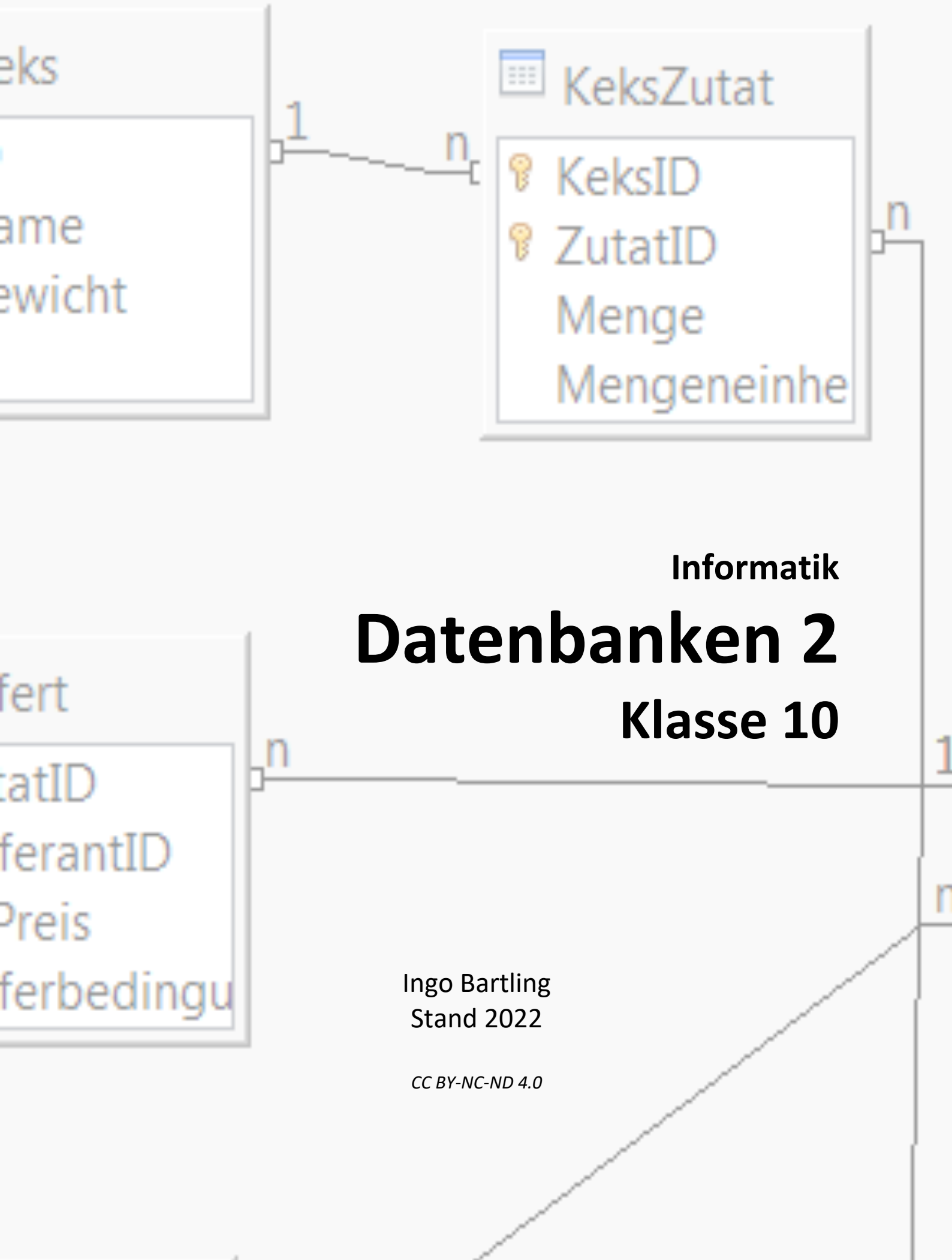


Informatik

Datenbanken 2

Klasse 10

Ingo Bartling

Stand 2022

Inhaltsverzeichnis

Einleitung

Dieses Arbeitsheft orientiert sich am LehrplanPLUS (www.lehrplanplus.bayern.de) des bayerischen Gymnasiums.

*Aufgaben mit Sternchen sind für Schüler*Innen gedacht, die sich noch tiefer mit dem Thema auseinandersetzen oder die benutzte Software besser kennenlernen wollen.*

Lehr- & Lernprinzipien

Ziel des Arbeitsheftes ist es, dich in der Erarbeitung der Unterrichtsinhalte zu unterstützen. Dabei gelten folgende Unterrichtsprinzipien:

> *Wissen kann nicht übertragen werden; es muss im Gehirn eines jeden Lernenden neu geschaffen werden.*

Daraus folgt, dass der Lehrer sozusagen kocht, aber der Lernende selber essen muss. Er kann nicht gefüttert werden. Dabei sollte verpflichtet sich der Lehrende spannende, interessante Gerichte zu anzubieten und der Lernende verpflichtet sich alles zu probieren.

> *Das Gehirn erkennt Muster und Regeln selbst.*

Damit das optimal funktioniert, muss man üben, üben, üben. Sinnvoll und zielgerichtet. An vielen Beispielen. Erst langsam und an einfachen Aufgaben, dann an immer komplexeren. Wie bei einem Instrument.

> *Neue Inhalte werden immer an Vorwissen angeknüpft.*

Es wie bei Weihnachtsbaum (altes Wissen) schmücken: Ich kann die Kugeln (neues Wissen) einfach drauf werfen und hoffen, dass sie hängenbleiben oder ich gebe mir Mühe und hänge die Kugeln zielgerichtet an den Baum („das ist doch wie“, Eselsbrücken). Je mehr Wissen vorhanden ist (dichter Baum) desto leichter fällt es einem auch, sich Neues zu merken.

Links

Hier noch ein paar Links, die dir „beim Essen“ helfen und die regelmäßig im Unterricht genutzt werden:

Link	**Beschreibung**
www.dbiu.de	Online-Datenbank passend zum Cornelsen Schulbuch
www.klassenkarte.de/kdm	Online-Editor zum Erstellen von Klassendiagrammen.
blog.ingo-bartling.de	Auf meinem Blog findest du die aktuellen Arbeitshefte und viele Aufgabe rund um die Informatik.
www.ingo-bartling.de	Hier finden sich viele Arbeitsblätter zu meinen drei Fächern Mathematik, Physik und Informatik. Manche Arbeitsblätter haben auch hier Eingang gefunden.

Viel Erfolg und Spaß beim Erlernen des Gebiets der Datenbanken.

Wiederholung 9. JgSt. - Grundwissen

Um eine gemeinsame Basis zu schaffen, wird das Wissen der neunten Jahrgangstufe hier nochmal zusammengefasst geübt. Auch hier gilt: Es ist wichtig, dass **DU** dich mit dem Thema auseinandersetzt. Das bedeutet aber nicht, dass du es alleine machen musst. Frage deine Platznachbarn oder recherchiere im Internet, wenn du Probleme hast. Diskutiere und vergleiche deine Ergebnisse mit anderen. Informatik ist ganz viel Teamwork, denn Programme entstehen in der Regel immer in Teamwork.

Aufgabe 1 – Geschenkdatenbank

Erstelle eine Datenbank für `Geschenke` an deine Familie und Freunde. Folgende Informationen sollen *mindestens* gespeichert werden: `Person, Anlass, Datum, Beschreibung, Preis`.

a) Gib die 4 wichtigsten[1] Datentypen mit einem Beispiel an. Ergänze die Tabelle.

Datentyp	**Beispiel**	**Erklärung**
boolean	true	Wahrheitswert
int	1234	Ganze Zahlen
float	3.141 oder 3.141f	Dezimalzahl
String	”Informatik”	Zeichenkette

b) Entwerfe eine Klassenkarte für die Klasse `Geschenk` in UML-Syntax mit Hilfe der Software www.klassenkarte.de/kdm. Speichere die Grafik.

c) Gib das Relationenschema für die Relation `Geschenk` an.
Unterstreiche den Primärschlüssel und erkläre, warum bzw. ob du einen natürlichen oder künstlichen Primärschlüssel gewählt hast.

Geschenk = {ID, Person, Anlass, Datum, Beschreibung, Preis} Künstlicher Schlüssel

Geschenk = {Person, Anlass, Datum, Beschreibung, Preis} Natürlicher Schlüssel, da keine Person ein Geschenk zweimal bekommen soll.

d) Setze die Relation in einem Tabellenkalkulationsprogramm um und ergänze 10 Datensätze. Hier MUSS ein Geschenk mindestens doppelt vorkommen („Kinogutschein“) und ein Name mindestens doppelt vorkommen – wähle z.B. Familienmitglied2.

e) Übernehme die Daten in ein Datenbanksystem.

[1] Was als wichtig erachtet wird, ist natürlich relativ und hängt auch von den IDEs und den Sprachen ab.
[2] Das doppelte Vorkommen ist wichtig für die Herleitung der kommenden Inhalte

f) Vervollständige die Tabelle für die Struktur von SQL-Abfragen:

SQL	Beschreibung
`SELECT`	Angabe der auszugebenen Spalte (Projektion)
`FROM`	Angabe der Tabellen: Tab1, Tab2,…
`WHERE`	Bedingung an ein oder mehrere Attribute
`GROUP BY`	Stapel bilden
`HAVING`	Bedingung an Aggregatfunktionen
`ORDER BY`	Sortieren nach Attribute(n): ASC, DESC

g) Führe folgende Datenbankabfragen an deiner Datenbank durch[3]:

SQL	Interpretation
`SELECT *` `FROM Geschenk`	Listet alle Information der Tabelle Geschenk auf.

`SELECT DISTINCT(Person)`
`FROM Geschenk`
`ORDER BY Person ASC`
	Listet alle Personennamen *einmalig* aufsteigend sortiert auf.

`SELECT Beschreibungen, Preis`
`FROM Geschenk`
`ORDER BY Preis DESC`
	Listet alle Geschenke absteigend sortiert nach dem Preis auf.

`SELECT Person, Beschreibungen`
`FROM Geschenk`
`WHERE Preis>5`
	Listet alle Beschreibungen, die Person und den Preis auf, die ein Geschenk bekommt, das echt mehr als 5€ kostet.

`SELECT COUNT(*) AS Anzahl` `FROM Geschenk`	Zählt alle Datensätze und benennt die Ausgabespalte in Anzahl um.

`SELECT Person, SUM(Preis) AS Wert`
`FROM Geschenk`
`GROUP BY Person`
	Gibt den Gesamtwert der Geschenke pro Person absteigen bzgl. des Wertes in der Form: „`Person Wert`“ aus.

`SELECT Beschreibung,`
`  COUNT(*), SUM(Preis)`
`FROM Geschenk`
`GROUP BY Beschreibung`
`HAVING SUM(Preis)>20`
`ORDER BY SUM(Preis)`
	Listet alle Geschenke, die Anzahl und deren Kosten aufsteigend sortiert aus, wenn die Kosten mehr als 20€ pro Geschenkegruppierung sind.

[3] Eventuelle Spalten- und Tabellennamen müssen an deine Daten angepasst werden.

Evaluierung: Aufgabe 1

Nach dem Bearbeiten einer Aufgabe solltest du deine Arbeiten reflektieren[4], um mögliches Verbesserungspotential zu finden.

Was lief gut?

Gibt an, was du bereits gut umsetzen oder bei was du helfen konntest.

Was lief nicht so gut? - Unbekannte Fachbegriffe

Oftmals fängt das Problem schon damit an, dass du bestimmte Begriffe wie *Relationenschema* nicht kennst. Liste alle Fachbegriffe auf, die vorkamen und die du nicht kanntest. Frage eine(n) Mitschüler*In und lasse dir den Begriff erläutern. Erkläre den Fachbegriff mit eigenen Worten und verknüpfe ihn dadurch mit bereits bekanntem Fachwissen. Kann dir niemand helfen, dann schreibe das Wort an die Tafel. Dann sehen andere, dass es ihnen genauso geht und der Lehrer erklärt den Begriff für alle.

Was lief nicht so gut? - Ungeschicktes Arbeiten

Hast du keine Sicherungen gemacht und musstest immer von vorne anfangen? War dir das Programm neu? Oder fühlst du dich unsicher bei der Benutzung – bei was genau? Gib an:

Welche Note würdest du dir nun selbst geben?

Note	**1**	**2**	**3**	**4**	**5**	**6**
Beschreibung	Du konntest alles, warst schnell, hast vielleicht anderen geholfen oder hast neue Aufgaben gefunden.	Du konntest alles richtig machen, warst aber vielleicht nicht sehr schnell.	Ich habe alles machen können, musste aber manches recherchieren oder nachfragen.	Ich habe manches falsch, weil ich noch Lücken haben, aber das Wesentliche habe ich verstanden.	Ich habe fast alles falsch und habe vieles nicht verstanden.	Ich habe nichts oder alles falsch, habe mir keine Mühe gegeben
Meine Note						

[4] https://blog.ingo-bartling.de/2018/03/05/effizient-lernen/

Fehler in Datenbanken

Beispieldatenbank: Geschenkeliste

ID	Person	Anlass	Datum	Beschreibung	Preis
1	Mama	Geburtstag	04.11.22	Theaterkarten	60,00
2	Oma	Xmas	24.12.22	Buch4, Bild	20,00
3	Papa	weinachten	24.12.22	Kinokarte	15,00
4	Freund	Geburtstag	24.03.22	2 Kinokarten	15,00
5	Bruder	Geburtstag	08.06.22	Buch1	19,90
6	Schwester	Geburttag	04.07.22	Buch2	14,90
7	oma	Geburtstag	24.12.22	Buch3	7,99
8	Opa	Geburtstag	24.12.22	Foto	2,49
9	Mama	Muttertag	08.05.22	Blumen	30,00
10	Mama	Weihnachten	24.12.22	Kinokarte, Kerzen	15,00

Untersucht man die Datensätze genau, so fällt auf, dass leicht falsche Daten entstehen können. Die Ursache für diesen Fehler ist in der Regel

Redundanz also mehrfachgespeicherte Informationen.

Gib Beispiele hierfür an:

Oma, Mama, Geburtstag, …

Durch falsche Daten können bei Abfragen merkwürdigen Ergebnissen entstehen:

Inkonsistenzen sind sich widersprechende Daten.

Gib ein Beispiel an:

Da bei ID=3 „weinachten“ geschrieben wurde, würde dieser Datensatz bei der Auflistung aller Weihnachtsgeschenke nicht ausgegeben werden.

Falscher Datensätze können durch folgende Datenbank-Operationen auftreten:

Operation	Fehlername
Einfügen neuer Daten	Insert-Anomalie
Aktualisieren von Daten	Update-Anomalie
Löschen von Daten	Delete-Anomalie

Erkläre an der obigen Relation, warum es hier beispielsweise beim Löschen von Datensätzen zu Fehlern kommen kann.

Es gehen wichtige Informationen verloren. So kann es hier passieren, dass jemand ein Geschenk doppelt bekommt.

1. Normalform

Die drei Anomalien können dadurch reduziert werden, dass die ursprüngliche Relation in mehrere Relationen zerlegt wird. Je nachdem, wie die neuen Relationen strukturiert sind, sind diese dann in der x-ten[5] Normalform.

Beispielrelation: Geschenk

ID	Person	Anlass	Datum	Beschreibung	Preis
1	Mama	Geburtstag	04.11.22	Theaterkarten	60,0
2	Oma	Xmas	24.12.22	Buch4, Bild	20,0
3	Papa	weinachten	24.12.22	Kinokarte	15,0
4	Freund	Geburtstag	24.03.22	2 Kinokarten	20,0
5	Bruder	Geburtstag	08.06.22	Buch1	19,9
6	Schwester	Geburttag	04.07.22	Buch2	14,9
7	oma	Geburtstag	24.12.22	Buch3	7,99
8	Opa	Geburtstag	24.12.22	Foto	2,49
9	Mama	Muttertag	08.05.22	Blumen	30,0
10	Mama	Weihnachten	24.12.22	Kinokarte, Kerzen	15,0

Die dargestellte Relation *Geschenk* in der DB *Geschenkeliste* ist für alle Abfragen noch nicht optimal. Eine Auflistung aller Personen, die eine Kinokarte bekommen haben, ist nur mit Hilfe der Operatoren _____like_____oder _____contains_____ möglich.

Welche Idee hast du, um das Problem zu lösen:

z.B. mehrere Spalten Geschenk1, Geschenk2, Geschenk3,…

oder mehrere Datensätze, die sich nur in „Beschreibung“ unterscheiden (Redundanz!)

Definition – 1. Normalform

Eine Relation ist in 1. Normalform (NF), wenn alle Attribute atomar sind,

also nicht weiter zerlegt werden können bzw. nur genau einen Wert haben.

Die neuen Tabellen sind oft größer, allerdings sind die Daten _____einfacher nutzbar_____.

Störend sind immer noch die _____mehrfach gespeicherten_____Daten (_____Redundanzen_____),

die leicht zu _____sich widersprechenden_____ Daten (_____Inkonsistenzen_____) führen können.

[5] Mehr dazu auf https://de.wikipedia.org/wiki/Normalisierung_(Datenbank)

Aufgabe 2 – Playlists bei Streaming-Diensten

Du möchtest die Verwaltung von Musikalben bei Streaming-Diensten nachvollziehen und legst dir hierfür eine eigene Datenbank mit der folgenden Tabelle an:

Album	ID	Titel	Jahr	Titelliste
	4637	"Billie Eilish", "Happier than ever"	2021	"Getting older", "My future", "Goldwing"
	8524	"The Beatles", "A hard day's night"	1964	"A hard day's night"
	7398	"Billie Eilish", "When we all fall asleep, where do we go"	2017	"Bad guy", "8", "Xanny"

a) Wie lautet der Relationen-Name? Album

b) Was lautet der Primärschlüssel, ist er natürlich oder künstlich und woran kann man ihn an der obigen Darstellung erkennen? ID, künstlich, unterstrichen

c) Gib das Relationenschema an:

Album = {ID:int, Titel:String, Jahr:int, Titelliste: String }

d) Warum ist die Tabelle nicht in 1. NF? Titel und Titelliste sind nicht atomar

e) Welche Abfragen wären schwer möglich? Welche Probleme gäbe es? Gib Beispiele an.

- Auflistung aller Alben von Billie Eilish wäre nur schwer möglich.
- Es ist immer nur die gesamte Titelliste verfügbar.
- Wer hat das Lied „Getting older" gesungen?

f) Gib die Tabelle in 1. NF an! Erstelle dazu in einem DBMS[6] eine entsprechende Tabelle `Album` in 1. NF. Ergänze die Tabelle um Felder, wie die Dauer eines Liedes auf Minuten gerundet.

g*) Erstelle ein Formular.

h) Ergänze 10 Datensätze aus 3 Alben deiner Wahl ein oder benutze obige Relation.
Hinweis: Informationen können aus dem Internet bezogen werden.

i) Erstelle Abfragen: „Spieldauer eines Albums", „Anzahl der Songs eines Albums", „Liste aller Alben eines Interpreten", „alle Alben auf denen der gleiche Song ist",…
Ist dein Relationen-Entwurf gut? Woran erkennst du das? Gib Beispiele an.

Antworten hängen vom jeweiligen Relationenschema ab.

j*) Gib die Ergebnisse der Abfragen aus i) als Bericht (Textdokument) aus.

[6] **DBMS**: Ein Datenbankmanagementsystem wie Base in OpenOffice oder Access, verwaltet mehrere Datenbanken und deren Tabellen, Abfragen etc.

Aufgabe 3* – Geschenkdatenbank

Benutze deine Geschenkeliste aus der Tabellenkalkulation und wandle die Daten so um, dass sie atomar sind. Importiere die Daten in eine Datenbank und versuche nun, alle Personen auszugeben, die mindestens eine Kinokarte bekommen haben.

Hinweis: Falls du bei deinen Daten diese Situation nicht hast, so vergebe zusätzlich an mindestens zwei Personen mindestens zwei Geschenke.

Aufgabe 4* – Asiatische Menükarte

Asiatische Menükarten haben oft eine auffällige Struktur: Der Unterschied zwischen vielen Gerichten besteht nur aus der Proteinbeilagen (Huhn, Fisch, Ente, Tofu), während die Soßen und Beilagen immer die gleichen sind: rote Currysoße, gelbe Currysoße, süßsaure Soße, etc.

a) Erstelle eine Datenbank mit nur 1 Relation in 1. Normalform inkl. Formular zum Anzeigen der Daten. Jeder Datensatz entspricht einem Gericht.

   Gib ein mögliches Relationenschema an:

   Gericht = {ID, Name, Protein, Kohlenhydrate, Beilage, Soße, Art, Preis}

   ID ist vom Datentyp int, der Rest vom Datentyp String.

   Art bedeutet Vorspeise, Hauptgericht, etc.

b) Erstelle eine Abfrage, die die Menükarte möglichst sinnvoll ausgibt.

c) Notiere, welche positiven bzw. negativen Aspekte dir bei deiner Lösung auffallen.

   Positiv: Die Menükarte könnte auch nach Protein, Soße oder Preis sortiert ausgegeben werden.

   Negativ: Es gibt viele Redundanzen (Soße), wodurch es zu Inkonsistenzen kommen kann

2. Normalform

Die Relation *Geschenk* wurde nun in die 1. Normalform umgewandelt und ergänzt.
Dies ist an den atomaren Attributen zu erkennen. Es treten aber immer noch viele Redundanzen auf, wodurch es zu Inkonsistenzen kommen kann.

Relation *Geschenk* in 1. Normalform

ID	Person	Anlass	GeschenkID	Datum	Beschreibung	Preis	Anzahl	Gesamt
1	Mama	Geburtstag	1	04.11.20	Theaterkarte	30,00	2	60,00
2	Oma	Weihnachten	2	24.12.22	Buch4	20,00	1	20,00
2	Oma	Weihnachten	3	24.12.22	Buch3	7,99	1	0,00
4	Papa	Weihnachten	4	24.12.22	Kinokarte	15,00	1	15,00
5	Freund	Geburtstag	5	24.03.22	Kinokarte	15,00	2	30,00
6	Bruder	Geburtstag	6	08.06.22	Buch1	19,90	1	19,90
7	Schwester	Geburtstag	7	04.07.21	Buch2	14,90	1	14,90
2	Oma	Geburtstag	3	24.12.22	Buch3	7,99	1	7,99
8	Opa	Geburtstag	9	24.12.22	Foto	2,49	1	2,49
1	Mama	Muttertag	10	08.05.22	Blumen	30,00	1	30,00
1	Mama	Weihnachten	4	24.12.22	Kinokarte	15,00	1	15,00
1	Mama	Weihnachten	12	24.12.22	Kerzen	5,00	1	5,00
1	Mama	Weihnachten	13	24.12.21	Schal	15,00	1	15,00

Der alte Primärschlüssel ID genügt nicht mehr, da der Datensatz zu beispielsweise ID=2 und ID=1 nicht mehr eindeutig gefolgert werden kann.

Bilde aus den obigen Attributen einen neuen Primärschlüssel: ID, Anlass, GeschenkID

Um die Redundanzen zu verringern, wird die Tabelle in mehrere Tabellen aufgeteilt.

Definition – Funktionale Abhängigkeit

Lassen sich aus Attributen a_i **eindeutig** die Werte anderer Attribute b_j folgern,

so liegt eine **funktionale Abhängigkeit** vor:

$$a_i \rightarrow b_j$$

Gib die funktionalen Abhängigkeiten in der obigen Relation an:

ID -> Person

GeschenkID -> Datum, Beschreibung, Preis, Anzahl

Anlass (Keine Abhängigkeit, aber Teil des Schlüssels)

Aus den funktionalen Abhängigkeiten kann man erkennen, dass folgende 3 Relationen sinnvoll wären: Person, Anlass, Geschenk

Merke: Jede Relation stellt genau nur 1 Sachverhalt dar!

Definition - 2. Normalform

Eine Relation ist in 2. Normalform, wenn sie

a) in 1. Normalform ist UND

b) jedes nicht-Schlüsselattribut vom ganzen(!) Schlüssel abhängt, nicht nur von einem Teil.

Aufgabe 5 – Warum künstliche Schlüssel?

Erläutere, warum nur Relationen mit einem zusammengesetzten Schlüssel die 2. Normalform verletzen können!

Ist der Schlüsselkandidat einelementig, so müssen sämtliche Attribute zwangsläufig voll funktional von diesem Schlüsselkandidaten abhängig sein. Dies ist genau die Voraussetzung für die NF 2.

Aufgabe 6 – Geschenkeliste in 2. NF

Gib die Relationen-Schemata der neuen Geschenke-DB an. Die Spalte *Gesamt* braucht nicht berücksichtigt zu werden, da diese auch berechnet werden kann.

Person = {<u>ID</u>, Name}

Anlass = {<u>ID</u>, Bedeutung}

Geschenk = {<u>ID</u>, Beschreibung, Preis, Anzahl, Datum}

Welches Problem tritt auf?

Es geht der Zusammenhang zwischen der Person, dem Anlass und dem Geschenk verloren.

Definition: Fremdschlüssel

Ist ein Attribut a in einer anderen Relation Primärschlüssel,

so ist a ein **Fremdschlüssel (foreign key)**.

a wird im Relationenschema unterstrichelt.

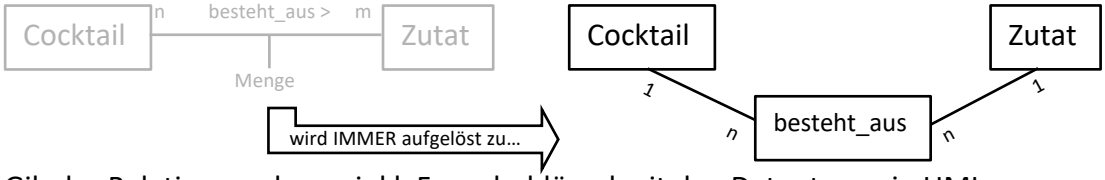
Mit Hilfe von Fremdschlüsseln lassen sich die ursprüngliche Datensätze wieder zusammensetzen. Betrachte hierzu den
.

Wintertime
10 ml Grenadine
20 ml Mandelsirup
70 ml Orangensaft
40 ml Ananassaft
20 ml Zitronensaft

Beispielaufgabe – Alkoholfreie Cocktails in 2. NF

Entwerfe eine Datenbank für die Speicherung von Rezepten alkoholfreier Cocktails.

a) Zeichne das Klassendiagramm mit allen Kardinalitäten.



b) Gib das Relationenschema inkl. Fremdschlüssel mit den Datentypen in UML-Schreibweise an.[7]

Cocktail = {ID : int, Titel : String}

Zutat = {ID : int, Titel : String}

besteht_aus = {ID:int, Cocktail_ID:int , Zutat_ID:int, Menge:int}

c) Teste deine Datenbank für dein Lieblingscocktail oder benutze das obige Rezept.

Cocktail

ID	Titel
1	Wintertime
2	Sunrise

Zutat

ID	Titel
1	Grenadine
2	Mandelsirup
3	Orangensaft
4	Ananassaft
5	Zitronensaft

besteht_aus

ID	CocktailID	ZutatID	Menge
1	1	1	10
2	1	2	20
3	1	3	70
4	1	4	40
5	1	5	20
6	2	1	40
7	2	4	50
8	2	5	20
9	2	3	150

[7] Eine Feldnamen *Name* ist in Access leider nicht möglich, da es ein reserviertes Wort ist.

Aufgabe 7 – Klassendiagramme

Zeichne bei allen Aufgaben das Klassendiagramm mit allen Kardinalitäten auf ein separates Blatt und notiere hier die zugehörigen Relationenschemata inklusive der Datentypen in UML-Schreibweise an.[8]

a) Die Mitglieder[9] eines Sportvereins und ihre Teilnahme an einer Sportart (Fußball, Tennis, Turnen)[10] sollen verwaltet werden.

n:m-Beziehung
Mitglied = {ID : int, Vorname : String, Nachname : String}
Sportart = {ID : int, Titel : String}
ist_in = { ID : int , SportartID : int, MitgliedID : int}

b) Die Bestellungen (nur ID als Attribut) mit den ausgewählten Produkten (Hamburger, Cheeseburger, Pommes etc.) an den Eingabeterminals einer FastFood-Kette soll verwaltet werden.

n:m-Beziehung
Bestellung = {ID : int }
Produkt = {ID : int, Titel : String}
ist_in = { ID : int , BestellungID : int, ProduktID : int, Anzahl : int}

c) Für jeden Tag darf eine SchülerIn bei unserer Mensa eine der Essensvarianten Menü1, Menü2, Nudeln bzw. Bowl bestellen.

1:n-Beziehung
Schueler = {ID : int, EssenID : int }
Essen = {ID : int, Titel : String}

d) Der eindeutige Zusammenhang zwischen einer SchülerIn und dem personalisierten Fahrausweis soll verwaltet werden.

1:1-Beziehung

Schueler = {ID : int, FahrausweisID : int }
Fahrausweis = {ID : int, Titel : String}

ODER
Schueler = {ID : int, Vorname : String}
Fahrausweis = {ID : int, Titel : String, SchuelerID:int}

e) Die Abschlussfahrt mit mehreren Zielen der Abiturienten soll verwaltet werden.

1:n-Beziehung
Schueler = {ID : int, ZielID : int }
Ziel = {ID : int, Titel : String}

f) Die Zimmerverteilung der SchülerInnen beim Schullandheim soll verwaltet werden.

1:n-Beziehung
Schueler = {ID : int, ZimmerID : int }
Zimmer = {ID : int, Titel : String}

[8] Aus Darstellungs- und Platzgründen werden bei der Musterlösung nur die Relationenschemata angegeben.
[9] Bei Personen genügt i.d.R der Name bzw., weil Name in Access nicht zugelassen ist, der Vorname als Attribut.
[10] Da der Schwerpunkt auf den Klassendiagramm liegt, werden meist nur wenige Beispielattribute angegeben.

g) Die Zuordnung der Bücher[11] einer lehrmittelfreien Bücherei zu den SchülerInnen soll verwaltet werden.

1:1-Beziehung
BuchExemplar = {ID : int, Titel : String, JgSt : int, Fach : String }
Schueler = {ID : int, Vorname : String, BuchExemplarID : int}

h) Die Zuordnung der roten Rosen, die man bei der SMV für eine MitschülerIn am Valentinstag bestellen kann, möchte die SMV mit Hilfe einer DB verwalten.

n:m-Beziehung
Schueler = {ID : int, Vorname : int }
Rose = {ID : int, SchuelerID_von : int , SchuelerID_an : int }

i*) Es soll verwaltet werden, was es an einem bestimmten Tag (Datum) in der Mensa als Menü1, Menü2, Vegetarisch bzw. als Bowl. gibt.

n:m-Beziehung
Tag = {ID : int, Datum : String }
Essen = {ID : int, Titel : String, ArtID : int}
Art = {ID : int, MenuArt : String }
auswahl = { ID : int, tagID : int, EssensID : int}

Es sind auch andere Lösungen möglich:

Tag = {ID : int, Datum : String }
Menu1 = {ID : int, Titel : String}
Menu2 = {ID : int, Titel : String}
Nudeln = {ID : int, Titel : String}
Bowl = {ID : int, Titel : String}
auswahl = { ID : int, TagID : int, menu1ID : int, menu2ID : int, NudelnID : int, BowlID : int }

Hier könnten keine weitere Arten (Suppen, Dessert, Menu3,…) ergänzt werden.

[11] Bei Büchern werden i.d.R. die Exemplare verwaltet, da ein Buchtitel öfters vorhanden ist.

SQL-Abfragen über mehrere Tabellen

SQL-Abfragen über mehrere Tabellen bzw. Relationen sind in der Grundform nicht sehr effizient, da in der Regel zunächst das sogenannte Kreuzprodukt bestimmt wird.

Definition: Kreuzprodukt

Bei einem Kreuzprodukt wird jede Zeile der 1. Relation (n Datensätze, i Spalten) mit jeder Zeile der 2. Relation (m Datensätze, j Spalten) verbunden. Es ergeben sich n*m Zeilen (Datensätze) und i+j Spalten.

Beispielaufgabe 8 – Kreuzprodukt

Gib das Kreuzprodukt der beiden angegebenen Relationen an.

R1

ID	Klasse
1	10a
2	10b

×

R2

ID	Name
1	Albert
2	Berta
3	Caesar
4	Doro
5	Emil
6	Fritz

R1 x R2

Da „ID“ nicht eindeutig wäre, muss der Relationen-Name mit angegeben werden

R1.ID	Klasse	R2.ID	Name
1	10a	1	Albert
1	10a	2	Berta
1	10a	3	Caesar
1	10a	4	Doro
1	10a	5	Emil
1	10a	6	Fritz
2	10b	1	Albert
2	10b	2	Berta
2	10b	3	Caesar
2	10b	4	Doro
2	10b	5	Emil
2	10b	6	Fritz

Gib die zugehörige SQL-Abfrage an:

```
SELECT *
FROM R1, R2
```

Beispielaufgabe – Fortsetzung alkoholfreie Cocktails in 2. NF

Gegeben waren folgende Relationen:

Cocktail

ID	Titel
1	Wintertime
2	Sunrise

Zutat

ID	Titel
1	Grenadine
2	Mandelsirup
3	Orangensaft
4	Ananassaft
5	Zitronensaft

besteht_aus

ID	CocktailID	ZutatID	Menge
1	1	1	10
2	1	2	20
3	1	3	70
4	1	4	40
5	1	5	20
6	2	1	40
7	2	4	50
8	2	5	20
9	2	3	150

a) Gib die ersten drei Datensätze der nebenstehen SQL-Abfrage an.
Welches Problem entsteht?
Wie wird es gelöst?

```
SELECT *
FROM besteht_aus, Cocktail, Zutat;
```

b) Wie viele Zutaten benötigt der Cocktail „Sunrise“? Gib eine SQL-Abfrage an.

c) Welche Zutaten werden für die Cocktails Sunrise bzw. Wintertime benötigt? Jede Zutat soll nur einmal genannt werden. Erstelle zwei passende SQL-Abfragen.

d) Welches Volumen in ml hat der Cocktail Wintertime?

e*) Erstelle eine SQL-Abfrage, sodass auf Basis der Ergebnistabelle ein Cocktailbuch nach Cocktails aufsteigend sortiert als Bericht ausgegeben werden könnte.

f*) Erstelle Formulare zum Eingeben weiteren Cocktails bzw. Zutaten.

g*) Gib weitere alkoholfreie(!) Cocktails ein.

Aufgabe 9* – Bundesjugendspiele in 2. NF

Entwerfe eine Datenbank für die Ergebnisverwaltung der Bundesjugendspiele. Hierzu möchtest du von jedem Schüler den Vornamen, Nachnamen, Klasse, Geburtsjahr speichern. Zudem möchtest du die Ergebnisse des Laufs (50m, 75m, 100m), Wurfs (80g, 200g) und des Weitsprungs speichern.

a) Zeichne das Klassendiagramm mit allen Kardinalitäten.

b) Gib das Relationenschema inkl. Fremdschlüssel mit den Datentypen in UML-Schreibweise an.

Person = {ID:int, Vorname:String, Nachname:String, , Klasse:String, Geburtsjahr:int }

Lauf = {ID : int, Strecke : int}

Wurf = {ID : int, Masse:int}

Wettbewerb = {ID:int, Person_ID:int , LaufID:int, L_Erg:float, Geschenk_ID:int,

Anzahl:int}

Aufgabe 10 – Geschenkeliste in 2. NF mit Fremdschlüsseln

a) Zeichne mit Bleistift das Klassendiagramm für alle benötigten Klassen inkl. der Kardinalitäten.

b) Gib die Relationenschemata der neuen Geschenke-DB **inklusive Fremdschlüssel** an. Die Spalte *Gesamt* braucht nicht berücksichtigt zu werden.
Gib auch die Datentypen in UML-Schreibweise an.

Person = {ID : int, Name : String}

Anlass = {ID : int, Anlass : String, Datum : String}

Geschenk = {ID : int, Beschreibung : String, Preis : float}

wer_was_wann = {ID:int, Person_ID:int , Anlass_ID:int, Geschenk_ID:int, Anzahl:int}

c) Zeichne die Tabellen und trage beispielhafte Werte ein, um deine Relationenschemata zu testen.

d) Setze deine Relationenschemata in einer DB um und gebe die ursprünglichen Daten ein.

e) Setze folgende SQL-Abfragen um:
   I. Was bekommt „Oma“ alles an Geschenken?
   II. Wieviel Geld musst du an Weihnachten ausgeben?
   III. Wer bekommt am meisten Geschenke (höchster Gesamtpreis oder Geschenkeanzahl)?

f) Ergänze einen Bericht *Einkaufsliste* bei der die Geschenke nach Person gruppiert angezeigt werden.

g*) Ergänze in deiner Datenbank ein Formular zum Eingeben von Geschenken bzw. Personen bzw. Anlässen.

h*) Ergänze in deiner Datenbank Berichte auf Basis der Abfragen aus d).

¡*****)
Ergänze ein Formular für die Relation *wer_was_wann* mit Hilfe von Kombinationsfeldern.

Aufgabe 11 – Cafeteria

In unserer Cafeteria können sich Schülerinnen und Schüler online für einen bestimmten Tag ein Mittagessen vorbestellen. Sie haben die Wahl zwischen „Menü 1“, „Menü 2“, „Nudeln mit Soße“ und „Salat Bowl“. An jedem Tag kann es ein anderes Menü oder andere Nudeln mit Soße oder eine Salat Bowl geben. Wenn die Schüler ihr Mittagessen abholen, identifizieren sie sich mit einem NFC-Chip[12], worauf ihr Name eingeblendet wird. Die Schülerinnen und Schüler können mit verfolgen, wie viele Essen und welche Essen schon ausgegeben wurden, wie viele noch reserviert sind und wie viele noch da sind – es werden immer ein paar mehr vorbereitet, wenn man mal spontan Hunger bekommt.

a) Zeichne das Klassendiagramm in UML und mit allen Kardinalitäten.

b) Gib nun das vollständige Relationenschema mit Datentypen und Fremdschlüsseln an.

Person = {ID : int, Name : String, Guthaben : float}

Angebot = {Datum : String, Menu1_ID:int, Menu2_ID:int, Nudeln_ID:int, Salat_ID:int, }

Gericht = {ID : int, Beschreibung : String, Preis : float, Kategorie_ID : int,
veg : boolean}

hat_gebucht = {ID : int, Person_ID : int, Tag_Datum : String, Gericht_ID}

Kategorie = {ID : int, Art}

[12] https://de.wikipedia.org/wiki/Near_Field_Communication

3. Normalform*

Ein Problem kann, je nach Zerlegung in die 2. Normalform, in der Geschenke-Datenbank immer noch auftreten[13]:

ID	Person
1	Mama
2	Oma
4	Papa

ID	PersonID	AnlassID	GeschenkID
1	1	2	1
2	2	2	2
3	4	1	3

ID	Anlass	Datum
1	Geburtstag	04.11.20
2	Weihnachten	24.12.22
3	Muttertag	

ID	Beschreibung	Preis	Anzahl	Gesamt
1	Theaterkarte	30,00	2	60,00
2	Buch4	20,00	1	20,00
3	Buch3	7,99	1	0,00

Der Wert des Nichtschlüsselattributs *Gesamt* hängt von den Werten der Nichtschlüsselattribute *Preis* und *Anzahl* ab. Warum können solch verkettete Abhängigkeiten (_____) zu Problemen führen?

Wird der Eintrag in *Preis* und/oder *Anzahl* geändert, ohne auch den Wert in *Gesamt* zu ändern, kommt es zu Inkonsistenten Daten.

Definition – 3. Normalform

Eine Relation ist in 3. Normalform, wenn sie

a) 2. Normalform ist UND

b) jedes Nichtschlüsselattribut direkt von Schlüssel abhängt

[13] Die Spalte *Gesamt* wurde extra nochmal angelegt, um das Problem aufzuzeigen.

Wiederholungsprojekt

Im Rahmen dieses Projekts wiederholst du nochmals alle Elemente, die du jetzt können solltest. Dies soll dein Wissen vor dem Abschlussprojekt festigen.

Aufgabe

Du möchtest eine Schülerfirma gründen, die den Mittagessensverkauf an deiner Schule organisieren soll. Die Verwaltung der Bestellung soll dabei mit Hilfe einer Datenbank erfolgen. Von den Schülern werden mindestens die Daten „Benutzername“, „Vorname“, „Nachname“ und „Klasse“ benötigt.

a) Zunächst sollen die Daten der Schüler gespeichert werden.

Gib die **Klassenkarte** und das zugehörige Relationenschema an. Die Zeichnung wird später ergänzt, lasse genügend Platz oder benutze https://klassenkarte.de/kdm/index.html

Begründe mit Hilfe von **Fachbegriffen**, dass die Relation *Schueler* in **2. Normalform** ist.

Setze deine Lösung mit Hilfe eines Datenbankprogramms um. Erstelle zunächst die zugehörige Tabelle bzw. Relation im Entwurfsmodus. Gib dann mindestens 5 Beispieldatensätze mit Hilfe eines schönen **Formulars** ein.

b) Im nächsten Schritt soll die Gerichte erfasst werden. Folgende Elemente sollen **mindestens** gespeichert werden: „Name“, „Preis“, „normal/vegan“[14].

Erstelle eine Relation *Essen*. Notiere hierzu in a) das Relationenschema. Ergänze die Relation *Essen* in deiner Datenbank. Gib über ein schön gestaltetes Formular mindestens 3 normale (Hamburger, Pizza, Pommes) und 2 vegane Essen (Kässpätzle, Salat) ein.

[14] Wenn möglich kann auch ein Foto des Essens gespeichert werden. Überlege gut, wie du die Eigenschaft *vegan*, *normal* oder auch *vegetarisch* speichern könntest.

c) Im letzten Schritt soll die Bestellung der Schüler durch eine Relation *hat_bestellt* verwaltet werden, die unter anderem das *Datum* der Bestellung speichern soll.

   Ergänze in a) das Klassendiagramm mit allen Assoziationen und Kardinalitäten und gib das Relationenschema für die Relation *hat_bestellt* an.

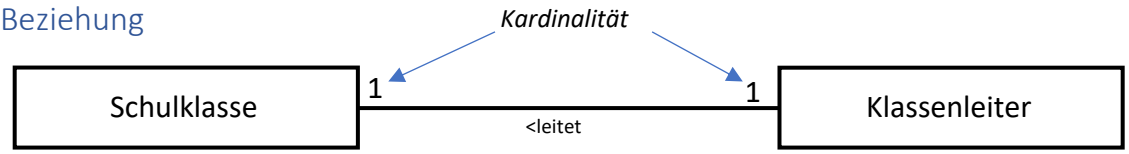
   Ergänze die Relation *hat_bestellt* in deinem Datenbankprojekt, gib mindestens 10 mögliche Bestellungen ein. Benutze ein **Kombinationsfeld**.

d) Auf Basis der eingegebenen Daten sollten folgende Abfragen bzw. Berichte durchgeführt werden. Nicht alles kann einfach als SQL-Abfrage erfolgen.
   a. Wie viele Schüler sind in jeder Klasse
   b. Wie viele Schüler haben an einem Datum etwas bestellt?
   c. Welches Gericht wurde wie oft bestellt?
   d. Welche Klasse hat welche Gerichte wie oft bestellt? (Klasse, Gericht, Anzahl)
   e. Wieviel Geld ist in jeder Klasse einzusammeln?
   f. Was hat jeder Schüler bestellt?
   g. Wie viel muss jeder Schüler bezahlen?

e) Ein guten Datenbankdesign zeigt sich erst, wenn die Datenbank erweitert oder ergänzt wird.
   SchülerInnen aus der Oberstufe möchten im Rahmen des Unterrichts herausfinden, ob es Unterschiede bei den Bestellungen zwischen Männern und Frauen gibt. Die folgenden Fragen sollen beantwortet bzw. die Behauptungen sollen belegt oder widerlegt werden können:
   a. Männer geben mehr Geld für Essen aus.
   b. Frauen kaufen eher gesundes/veganes Essen.
   c. Männer kaufen sich mehr Essen.
   d. Die Schüler der Unterstufe (Klassen 5,6,7) geben weniger Geld aus als Schüler der Klasse 8-10.

Anhang 1: Beziehungen zwischen Relationen & Kardinalitäten

1:1-Beziehung

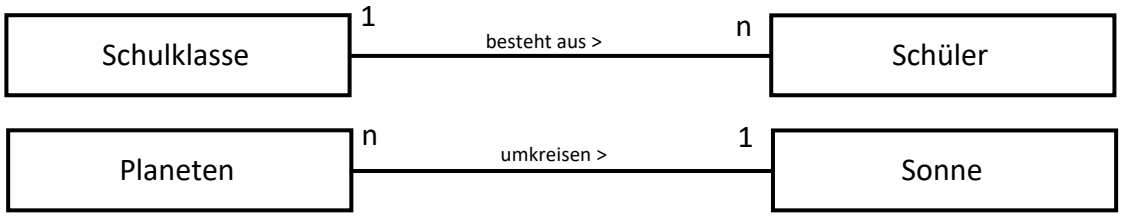


Umsetzung: Schulklasse={ID, Name, KlassenleiterID}
Klassenleiter={ID, Name, Unterrichtsfach}

oder

Schulklasse={ID, Name}
Klassenleiter={ID, Name, Unterrichtsfach, SchulklassenID}

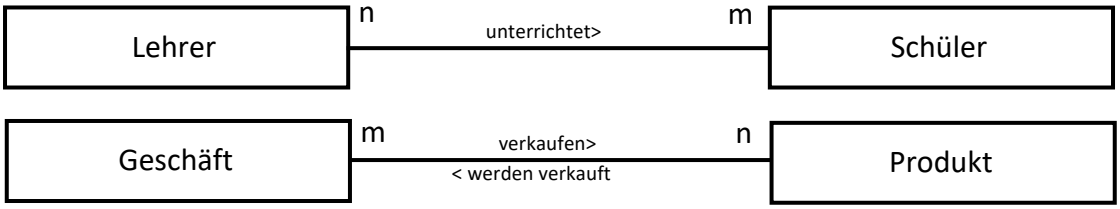
1:n-Beziehung



Umsetzung: Schulklasse={ID, Name }
Schüler={ID, Name, SchulklasseID}

Der Fremdschlüssel kommt immer dahin, wo „n“ war.

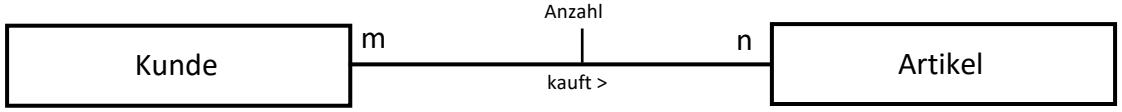
n:m-Beziehung



Umsetzung: Lehrer={ID, Name }
Schüler={ID, Name}
unterrichtet={ID, LehrerID, SchülerID}

Diese Beziehungen können nur durch **Vermittlungstabellen** aufgelöst werden.

Die Beziehung selbst kann ebenfalls Eigenschaften besitzen, wie folgendes Beispiel zeigt:

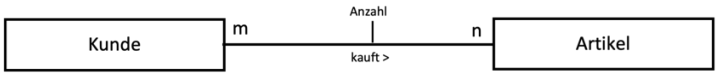


Umsetzung: Kunde={ID, Name, Adresse }
Artikel={ID, Name}
kauft={ID, KundenID, ArtikelID, Anzahl}

Anhang 2: Fach- & Grundbegriffe

Ergänze hier nach eigenem Ermessen dir unbekannte Begriffe und wichtige neue Vokabeln.

Fachbegriff	**Erklärung**
Modellierung	Analysieren und Übertragen einer Aufgabe in die Informatik: Substantiv -> Klasse oder Attribut Adjektiv -> Attribut Verb -> Methode
Klassendiagramm	Klassendiagramm bilden den Zusammenhang zwischen Klassen ab. Klassen stehen immer in Rechtecken. Im Themenbereich DB werden idR. nur Klassennamen angegeben zzgl. Kardinalität und Beziehungsname inkl. Leserichtung
Relation	„Relation“ kann synonym zu „Tabelle“ benutzt und als Karteikasten mit Karteikarten (Datensatz) vorgestellt werden. Eine Datenbank besteht aus mindestens einer Relation. Diese muss einen Primärschlüssel besitzen.
Relationenschema	Eine Relationenschema gibt den Relationenname zzgl. der Feld-/Attributnamen eventuell mit Datentyp und Primär- bzw. Fremdschlüsseln an: Tier = {Nr:int, Name:String, GehegeNr:int}
Datentypen	Die vier wichtigsten Datentypen lauten: int (ganze Zahl), float (Dezimalbruch), boolean (Wahrheitswert), String (Zeichenkette) Es gibt weitere Datentypen: double, Date, char,…
Redundanzen	Identische Attributwerte; sollte unbedingt vermieden werden, da sie zu Inkonsistenzen führen können.
Inkonsistenzen	Attributwerte, die eigentlich identisch sein sollten, sich jedoch unterscheiden.
Anomalien	Es gibt drei Möglichkeiten, um „Fehler“ (Anomalien) zu erzeugen: Beim erstmaligen Anlegen (Insert-Anomalie), beim Aktualisieren (Update-Anomalie), beim Löschen noch benötigter Daten (Delete-Anomalie)
Kardinalität	(dt. Vielfachheit). Wird benötigt, um in Klassendiagrammen die Art der Beziehung (1:1, 1:n, n:m) darzulegen.
1. Normalform	Jeder Attributwert kann nicht weiter sinnvoll zerlegt werden, ist somit atomar.
2. Normalform	Die Relation ist in 1. NF und jeder Attributwert ist voll funktional abhängig vom Primärschlüssel.

Kreuzprodukt	Stehen bei WHERE in einer SQL-Abfrage mehrere Relationen, so werden alle Datensätze miteinander kombiniert, was zu sehr großen Ergebnistabellen führt mit vielen inkonsistenten Daten.
Primärschlüssel	Ein Primärschlüssel identifiziert einen Datensatz eindeutig. Eine Primärschlüssel kann aus mehreren Attributen der Relation bestehen (natürlich) oder eine zusätzliche Nummer oä sein (künstlich).
Fremdschlüssel	Bezieht sich ein Attribut auf einen Primärschlüssel einer anderen Relation so spricht man von einem Fremdschlüssel.
Funktionale Abhängigkeit	Aus einem Attributwert kann ein anderer gefolgert werden: Ort -> PLZ
Atomar	Die Werte eines Attributs können nicht weiter zerlegt werden. Das muss nicht immer eindeutig sein und hängt vom Einsatz der DB ab (Datum, Straße+Hausnummer)
1:1-Beziehung	 Fremdschlüssel kann entweder bei „Schulklasse“ oder bei „Klassenleiter“ stehen.
1:n-Beziehung	Häufigste Art der Beziehung:  Der Fremdschlüssel steht immer bei der Relation mit dem „n“, hier also bei „Schüler“.
n:m-Beziehung	 Werden immer zu zwei 1:n-Beziehungen zu einer zusätzlichen Relation kauft = {ID, Kunde_ID, Artikel_ID, Anzahl} aufgelöst.

Anhang 3: Ausgewählte Musterlösungen

Wiederholungsprojekt

a. Wie viele Schüler sind in jeder Klasse?
b. Wie viele Schüler haben an einem Datum etwas bestellt?
c. Welches Gericht wurde wie oft bestellt?
d. Welche Klasse hat welche Gerichte wie oft bestellt?
e. Wieviel Geld ist in jeder Klasse einzusammeln?
f. Was haben die Schüler mit der ID = 1 und ID = 2 alles bestellt?
g. Welcher Schüler muss am meisten bezahlen?

a) Wie viele Schüler sind in jeder Klasse?

```
SELECT Klasse, COUNT(Klasse) AS Schüleranzahl
FROM Schueler
GROUP BY Klasse
```

b) Wie viele Schüler haben an einem Tag (Datum) etwas bestellt?

```
SELECT Tag_ID, COUNT(Tag_ID) AS Buchungsanzahl
FROM Schueler, hat_gebucht, Tag
WHERE Tag.ID = Tag_ID
GROUP BY Tag_ID
```

c) Welches Gericht wurde wie oft bestellt?

```
SELECT Count(*) AS Buchungsanzahl, Beschreibung
FROM Essen, hat_gebucht
WHERE wer_hat_was_wann_gebucht.Gericht_ID = Essen.ID
GROUP BY Beschreibung
```

d) Welches Klasse hat welche Gerichte wie oft bestellt (Klasse, Gericht, Anzahl)

```
SELECT Klasse, Beschreibung, COUNT(Beschreibung) AS Anzahl
FROM Essen, Schueler, hat_gebucht
WHERE hat_gebucht.Schueler _ID = Schueler.ID AND
hat_gebucht.Gericht_ID = Essen.ID
GROUP BY Klasse, Beschreibung
Order BY Klasse
```

e) Wieviel Geld ist in jeder Klasse einzusammeln?

```
SELECT Klasse, SUM(Preis) AS Kosten
FROM Schueler, hat_gebucht, Essen
WHERE Schueler_ID = Schueler.ID AND Gericht_ID = Gericht.ID
GROUP BY Klasse
```

f) Was haben die Schüler mit der ID = 1 und ID = 2 alles bestellt?

```
SELECT Distinct(Beschreibung)
FROM Schueler, hat_gebucht, Essen
WHERE Schueler_ID = Schueler.ID AND Essen_ID = Essen.ID AND
   (Schueler.ID = 1 OR Schueler.ID = 2)
```

g) Welcher Schüler muss am meisten bezahlen?

```
SELECT Benutzername, SUM(Preis) AS Kosten
FROM Schueler, hat_gebucht, Essen
WHERE Schueler_ID = Schueler.ID AND Essen_ID = Essen.ID
GROUP BY Benutzer
ORDER BY Sum(Preis)
```