

Lernfeld 08

Daten systemübergreifend bereitstellen

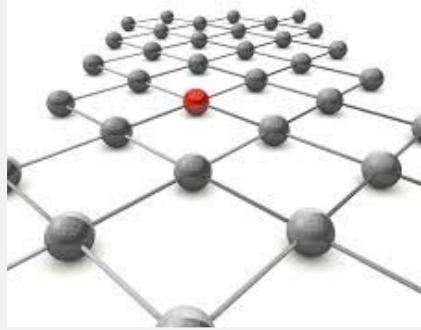
Themen und Lernziele



Aufträge im Rahmen
von Softwareprojekten
bearbeiten

Lernziel

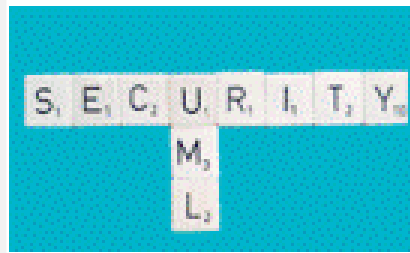
Projektmanagement und
Vorgehensmodelle vertiefen



Daten bewerten und
Daten zusammen-
führen

Lernziel

Qualität von Daten
bewerten können



Objektorientierung
und Informations-
sicherheit planen

Lernziel

Anwendungen planen
mittels UML



Softwareergonomie von
Benutzerschnittstellen

Lernziel

Anforderungen an das
Design moderner
Benutzerschnittstellen
erlernen



Anwendungen in Java
implementieren

Lernziel

Grundlagen von Java
erlernen

Themen und Lernziele



Anwendungen in
Python
implementieren

Lernziel

Objektorientierung und
GUI's in Python
implementieren



Datenbanklösungen
bedarfsgerecht
entwickeln

Lernziel

Basiswissen zu
Datenbanken und SQL
erweitern



Software testen und
dokumentieren

Lernziel

Wissen über das Testen
und Dokumentieren von
Software vertiefen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

Einleitung

Sie sollen Ihr bisheriges Wissen über Datenbanken und SQL festigen und erweitern.

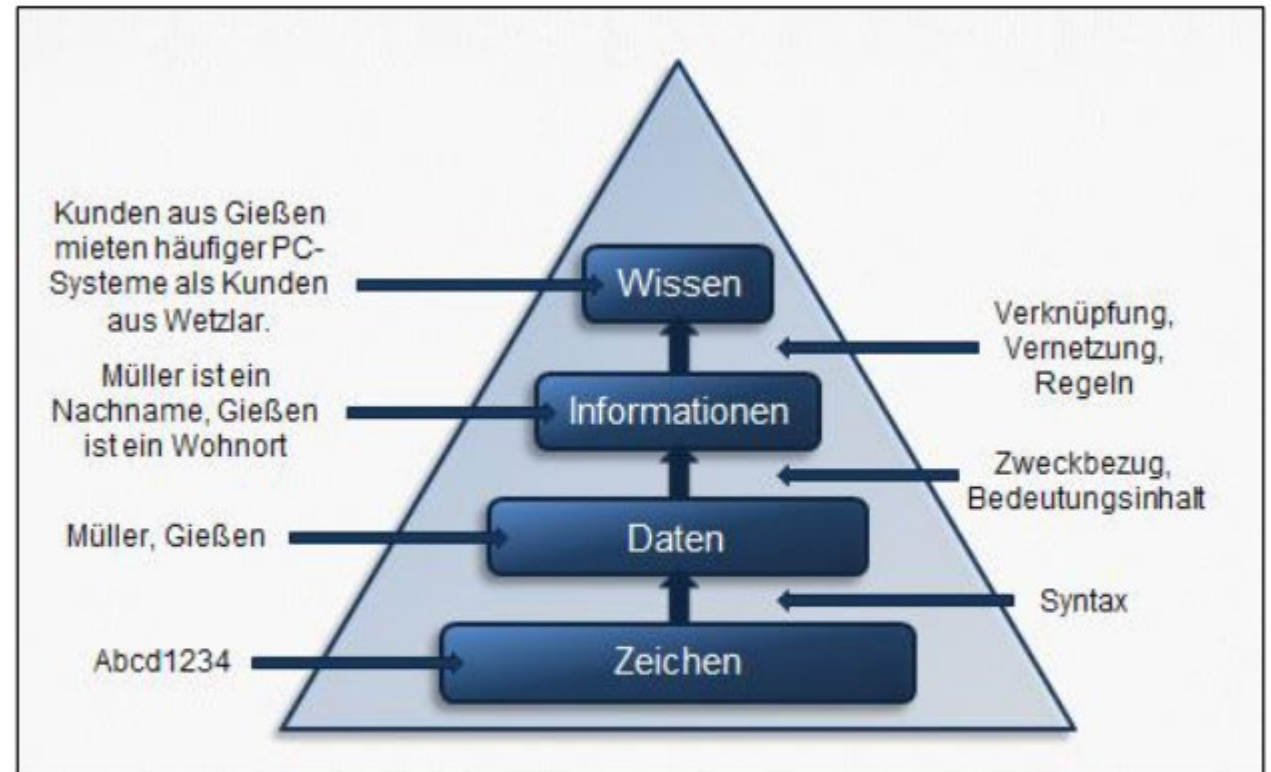
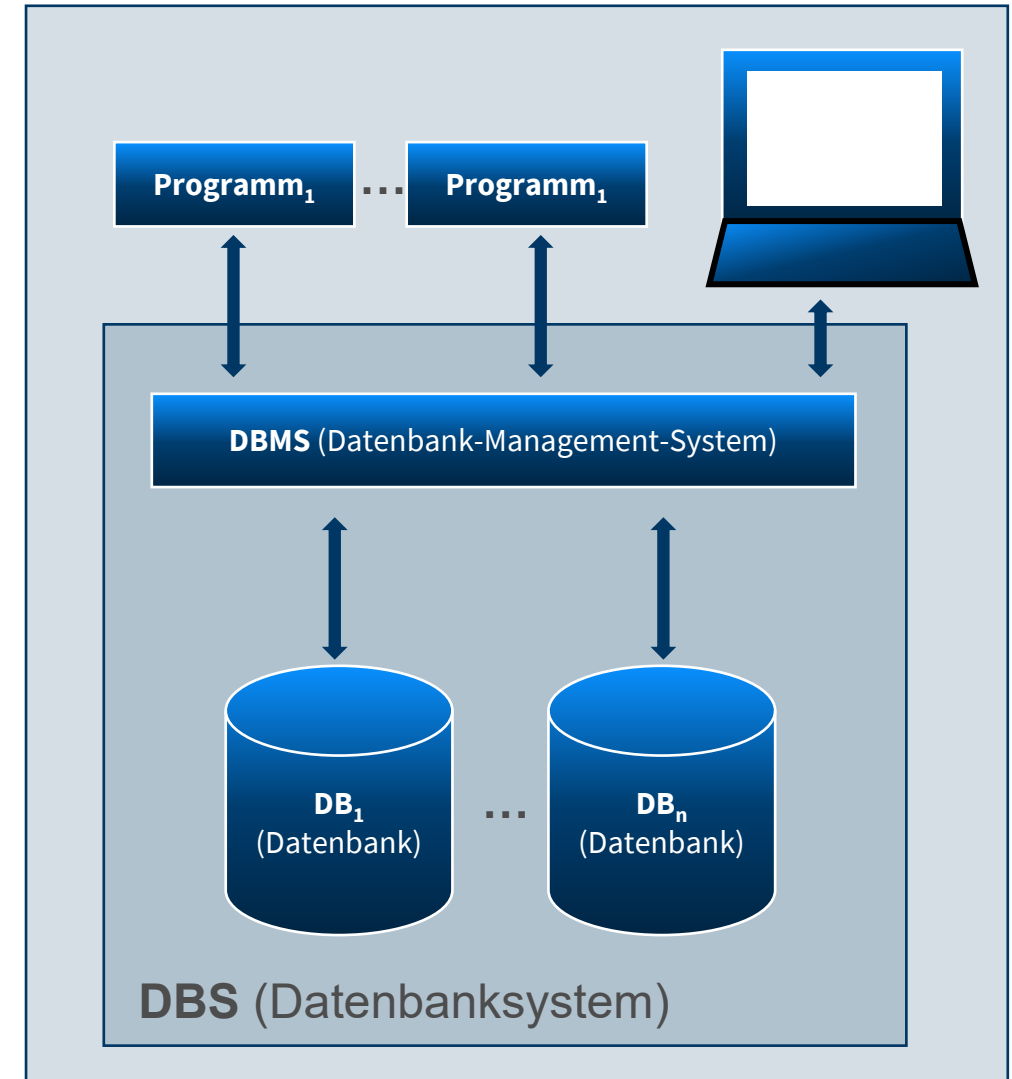


Abb. 1: Zusammenhang zwischen Wissen, Informationen, Daten und Zeichen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7 Datenbanken

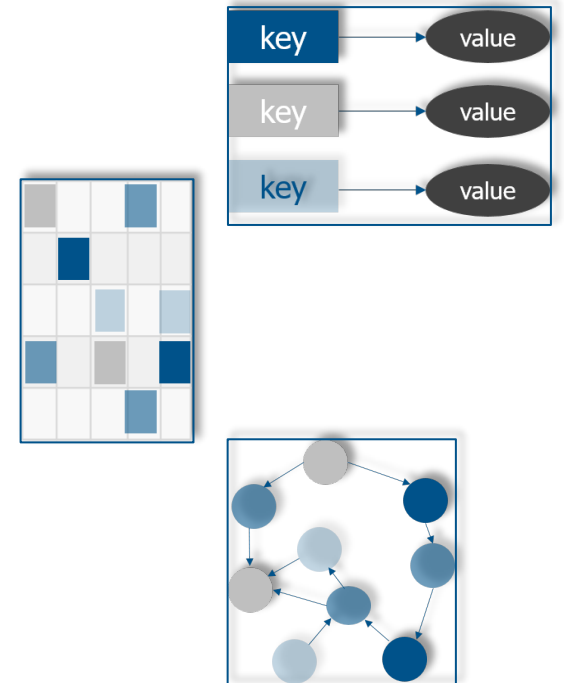
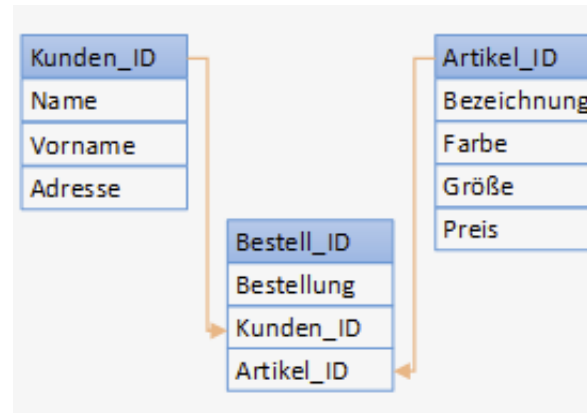
- Daten werden systemübergreifend bereitgestellt.
- Möglichkeit durch Datenmodelle und Datenmanagementsysteme den Ort der Speicherung festlegen.
- Daten werden in Datenbanksystemen (DBS) gespeichert.
- Ein DBS umfasst Datenbanken (DB) und das Datenbank-Management-System (DBMS).
- Unter einem Datenbank-Management-System versteht man ein Programmsystem, das die notwendige Systemsoftware zur Datenverwaltung bereitstellt.
- Es handelt sich um eine Software zur Speicherung und Bearbeitung von großen Datenmengen.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Sie sollen verschiedene Datenmodelle unterscheiden können und ein geeignetes Modell für Ihre Datenbanklösung auswählen.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Beispiele für unstrukturierte und teilweise strukturierte Daten

Text:	Fließtext, strukturierter Text, Textsammlungen, Dokumente usw.
Grafik:	Landkarten, Kataster, Seekarten, technische Zeichnungen, 3-D-Modell usw.
Bild:	Fotos, Röntgenbilder, MRT- und CT-Bilder, biometrisches Foto
Audio:	Musik, Sprache, Geräusche usw.
Video:	Film, Videoaufzeichnung, Videokonferenz, Animation usw.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

Big Data 5 Vs

 **Volume**
Datenmenge

- Weltweite Datenmenge 2020: 50 Zettabyte
- Teilchenbeschleuniger erzeugt 1 Petabyte Daten jährlich.
- Facebook belegt 1 Petabyte Speicherplatz.
- Ca. 100 Terabyte Daten pro US-Unternehmen
- Weltweit werden täglich 2,5 Trillionen Bytes an Daten erzeugt (2017).
- Geschätzte 6 Mrd. Menschen weltweit besitzen ein Mobiltelefon.

 **Velocity**
Geschwindigkeit der Datenverarbeitung

- NY Stock Exchange generiert 1 Terrabyte Handelsdaten pro Tag.
- 2016: 20 Mrd. Netzwerkverbindungen
- 2019: 3,3 Mrd. Smartphones weltweit

 **Value**
Mehrwert der Daten
für das Unternehmen

 **Variety**
Vielfalt der Datenarten

- Online-Videos: 100 Mio. Videos; mehr als 600 Jahre Laufzeit.
- Facebook: 200 Mrd. Fotos
- Globale Gesundheitsdaten: 150 Exabyte; Wachstum: 2,4 Exabyte pro Jahr
- Twitter: 500 Mio. Tweets pro Tag

 **Veracity**
Glaubwürdigkeit der Daten

- 33 % aller Führungskräfte misstrauen den Informationen, die sie verwenden, um Entscheidungen zu treffen.
- Mangelnde Datenqualität kostet Unternehmen jährlich 600 Mrd. Euro.
- 30 % der von Vermarktern gesammelten Daten sind für Entscheidungsfindung in Echtzeit unbrauchbar.
- Fehlerhafte Daten in allen Unternehmen und der Regierung kosten die US-Wirtschaft 3,1 Mrd. Dollar pro Jahr.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Die fünf Big Data Vs

Volumen

beschreibt die immer größer werdende Menge an Daten, welche zu speichern und zu verarbeiten ist.

Variety

beschreibt die Vielfalt der Daten. Rund 90 % der gespeicherten Daten sind den unstrukturierten Formaten wie Texten, Bildern oder Videos zuzuordnen. Durch Big Data werden diese Daten anhand von Machine Learning analysierbar.

Velocity

kennzeichnet die erhöhte Geschwindigkeit, mit der Daten produziert und verarbeitet werden müssen. Für viele Anwendungsfälle spielt die Echtzeitverarbeitung der Daten eine große Rolle und kann für den entscheidenden Wettbewerbsvorteil sorgen.

Veracity

ist die Vertrauenswürdigkeit und die Qualität der Daten, welche aus vielen sehr unterschiedlichen Quellen zusammengeführt werden.

Value

beschreibt den Mehrwert für das Unternehmen, der sich durch die Auswertung der riesigen Datenmengen ergibt. Das ist der wichtigste Punkt für Unternehmen im Big-Data-Umfeld.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

ACID-Modell

Atomicity

Alle Aktionen einer Transaktion werden komplett oder gar nicht ausgeführt.

Consistency

Ist eine Transaktion erfolgreich abgeschlossen, muss sie in der zuvor konsistenten Datenbank einen wieder konsistenten Zustand hinterlassen.

Isolation

Stellt sicher, dass die parallele Nutzung der Datenbank durch mehrerer Anwender keine negativen Auswirkungen nach sich zieht.

Durability

Ist eine Transaktion ausgeführt und konsistent, sind ihre Informationen dauerhaft in der Datenbank gespeichert.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Um die Vorteile der NoSQL-Datenbanken zu nutzen, muss man auf die Zuverlässigkeit des ACID-Modells verzichten.

NoSQL-Datenbanker folgen dem BASE-Modell (Basic Availability, Soft State, Eventual Consistency); somit ist eine unmittelbare Konsistenz und Zuverlässigkeit der Daten nicht gegeben.

Analogie:



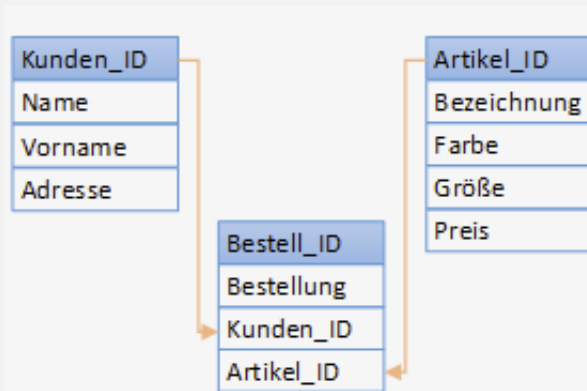
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Relationale Datenbank

Relationales Datenmodell

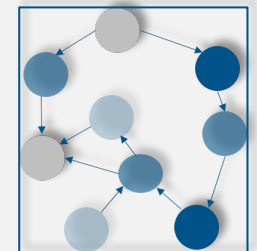
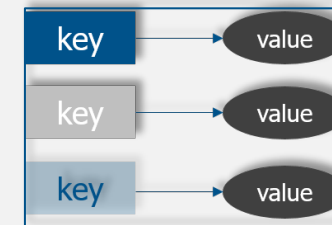
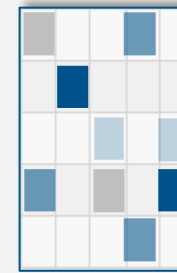
Daten werden in separaten Tabellen gespeichert und bei komplexen Suchanfragen vom System zusammengeführt.



NoSQL-Datenbank

Verschiedene Datenbankmodelle

Z. B. Dokumenten-, Graph- oder Key-Value- Datenmodell



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Relationale Datenbank

Schema

Datentyp und -struktur sind festgelegt.
Aufwendige Anpassungen;
um neue Attribute hinzuzufügen, muss teilweise die gesamte Datenbank angepasst werden.

Skalierung

Vertikale Skalierung; ein einzelner Server muss die Leistung des kompletten Datenbanksystems tragen. Die Leistungsfähigkeit des Servers kann bedingt gesteigert werden.

NoSQL-Datenbank

Schema

Flexibel; strukturierte, semistrukturierte und unstrukturierte Daten können zusammen abgespeichert werden, eine vorherige Konvertierung ist nicht notwendig.

Skalierung

Horizontale Skalierung; durch Hinzufügen von neuen Datenbankservern kann das Datenbanksystem fast beliebig skaliert werden.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Relationale Datenbank	NoSQL-Datenbank
<p>ACID-Regeln</p> <p>Bei SQL-Datenbanken werden alle ACID-Regeln wie Atomicity, Consistency, Isolation, Durability umgesetzt.</p>	<p>ACID-Regeln</p> <p>Sie werden bei NoSQL-DBs meistens nicht unterstützt. Stattdessen wird das BASE-Modell verwendet (Basically Available, Soft State, Eventually Consistent). Es gilt: Verfügbarkeit vor Konsistenz.</p>
<p>Leistung</p> <p>Um die Leistung zu erhöhen, müssen Abfragen, Indizes und Struktur optimiert werden.</p>	<p>Leistung</p> <p>Durch das Nutzen von Cloud-Servern und Hardware-Cluster haben NoSQL-Datenbanken eine deutlich höhere Leistungsstärke.</p>

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.1 Ein geeignetes Datenmodell auswählen

Relationale Datenbank	NoSQL-Datenbank
<p>Populäre RDBMs</p> <p>MySQL, PostgreSQL, Oracle, MS-SQL</p>	<p>Populäre NoSQL</p> <p>MongoDB, Apache HBase Amazon DynamoDB, Redis, Couchbase Cassandra, Elasticsearch</p>
<p>Einsatzbereich</p> <p>Datenbanklösungen, welche</p> <ul style="list-style-type: none">• einen ACID- Support erfordern und• bei denen sich die Struktur nicht wesentlich verändert• oder die Datenmenge nicht unkontrolliert wächst. <p>Vor allem für Daten, welche sich leicht strukturieren lassen.</p>	<p>Einsatzbereich</p> <p>Datenbanklösungen, welche</p> <ul style="list-style-type: none">• eine Echtzeitverarbeitung von Daten erfordern,• Daten verarbeiten, die sich schlecht strukturieren lassen,• große, schnell wachsende Datenmengen zu bewältigen haben.

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.1 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Videodaten werden zu den strukturierten Daten gezählt.
- b) E-Mails zählen zu den semistrukturierten Daten.
- c) Den Großteil der Daten im Big-Data-Bereich machen unstrukturierte Daten aus.
- d) Echtzeitdatenverarbeitung ist ein Merkmal von Big-Data.
- e) Relationale Datenbanklösungen eignen sich hervorragend für Big-Data-Anwendungen.
- f) Die Skalierung von NoSQL-Datenbanklösungen kann vertikal durch Hinzufügen neuer Server erfolgen.

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.1 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

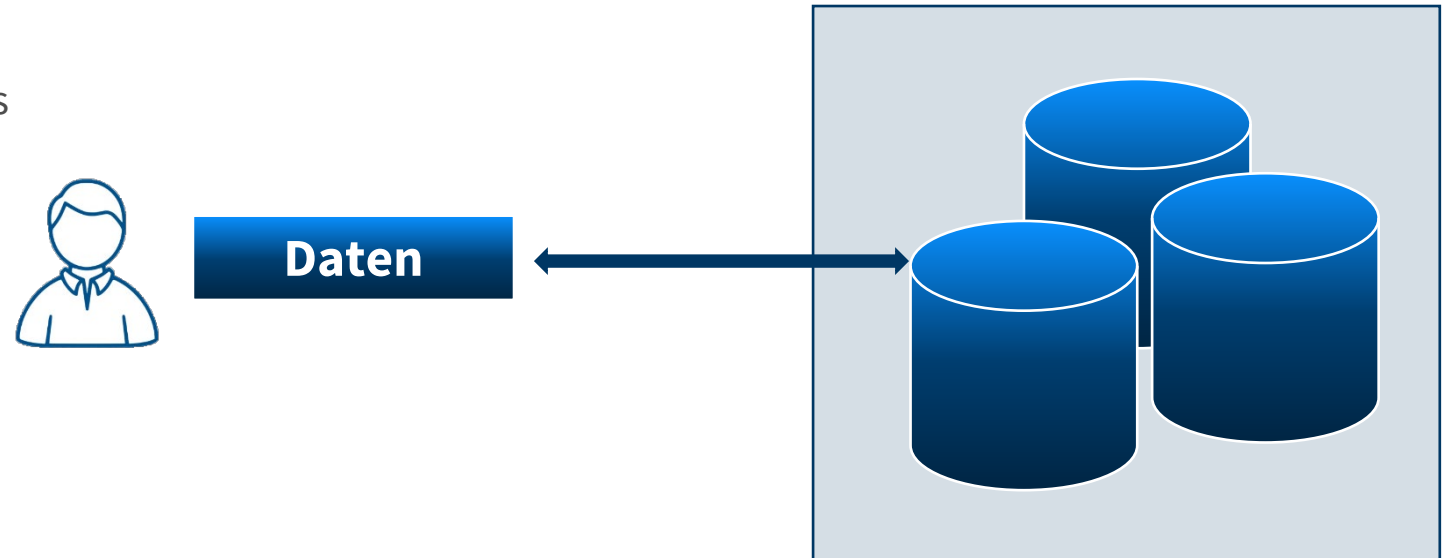
Welche Aussage ist richtig?

- g) Relationale Datenbanken folgen dem ACID-Modell.
- h) Graphdatenbanken zählen zu den NoSQL-Datenbanken.
- i) MySQL ist eine Key-Value-Datenbank.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess des Designs relationaler Datenbanken beschreiben

Sie sollen sich die Phasen des Designs einer relationalen Datenbank erarbeiten und im Anschluss präsentieren.

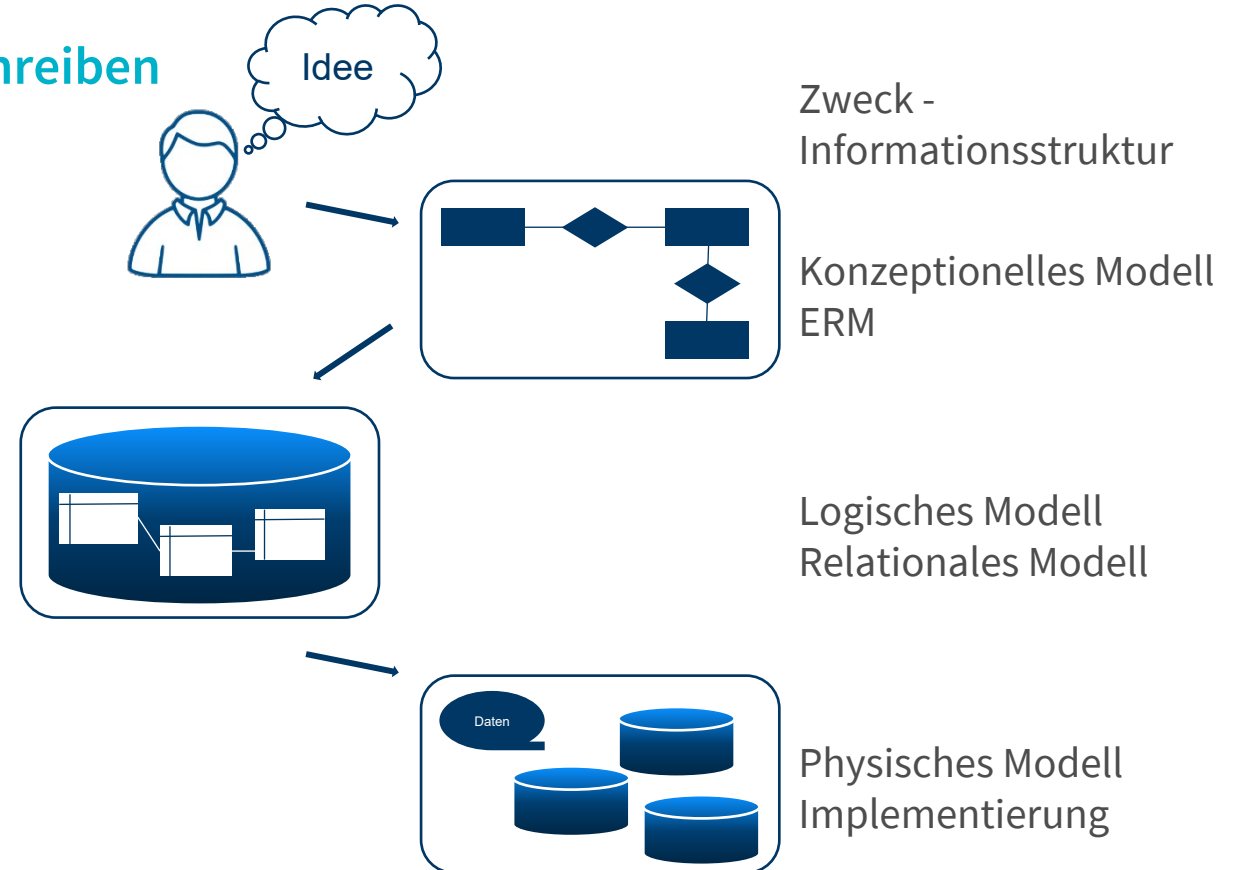


8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

4 Phasen des Designprozesses

- Analysephase
verbal, textuell, visuell
- Konzeptionelle Phase
formal, vollständig, grafisch
Entitäten, Attribute, Beziehung (ERM)
- Logische Phase
Relationen, Attribute, Primärschlüssel,
Fremdschlüssel (relationales Datenmodell)
- Implementationsphase
DDL – Data Definition Language



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

Analysephase

- Ermittlung der Informationsstruktur
- Informelle Beschreibung:



Zweck -
Informationsstruktur

- Ein Mitarbeiter hat genau eine Personalakte und jede Personalakte ist genau einem Mitarbeiter zugeordnet
- Eine Klasse hat mehrere Schüler. Jede Klasse hat eine Klassenbezeichnung und eine Klassennummer. Jeder Schüler hat einen Vornamen und Nachnamen.
- Ein Kunde kauft mehrere Artikel. Aber ein Artikel kann auch von mehreren Kunden gekauft werden.

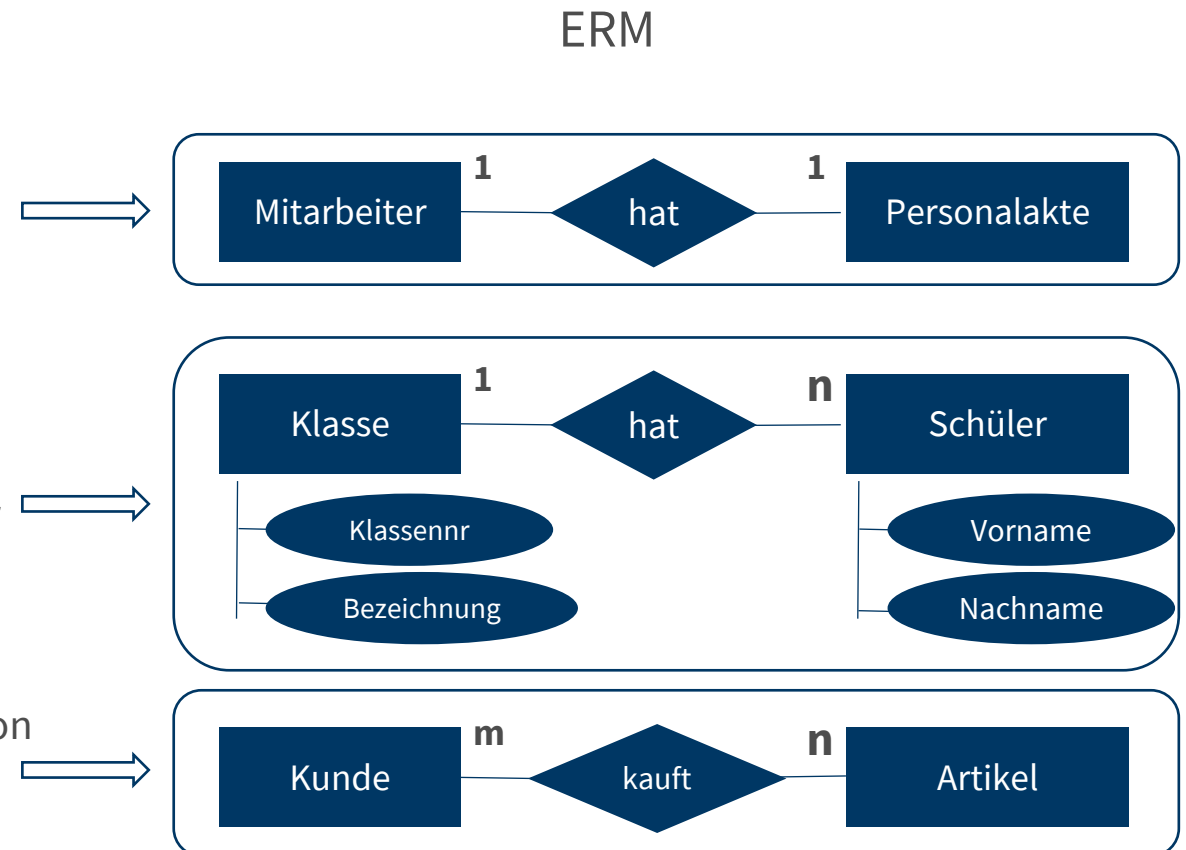
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

Konzeptionelle Phase

Informelle Beschreibung → Entity Relationship Modell (ERM)

- Ein Mitarbeiter hat genau eine Personalakte und jede Personalakte ist genau einem Mitarbeiter zugeordnet.
- Eine Klasse hat mehrere Schüler. Jede Klasse hat eine Klassenbezeichnung und eine Klassennummer. Jeder Schüler hat einen Vornamen und einen Nachnamen.
- Ein Kunde kauft mehrere Artikel. Aber ein Artikel kann auch von mehreren Kunden gekauft werden.



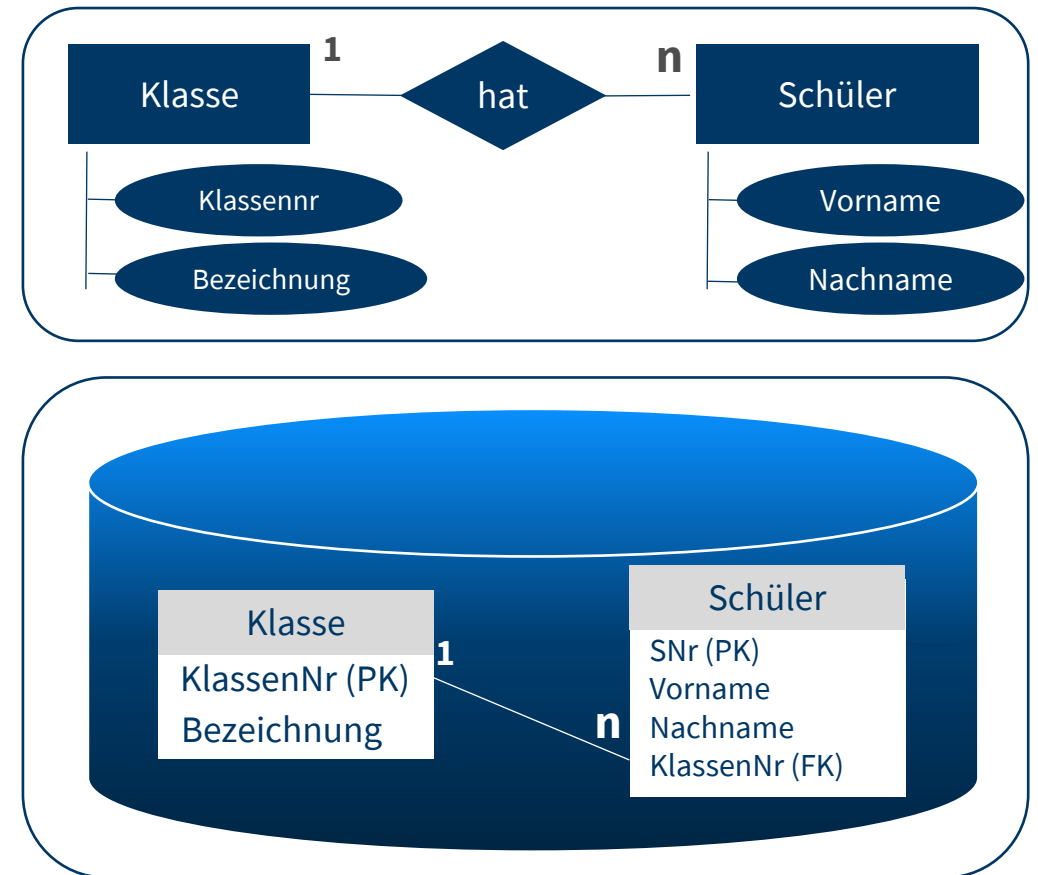
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

Logische Phase

Umsetzung des ERM in ein **relationales Datenmodell**

- Primärschlüssel (PK)
- Fremdschlüssel (FK)
- Anwenden der Transformationsregeln
- Anwenden der Normalisierung



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

Transformationsregeln:

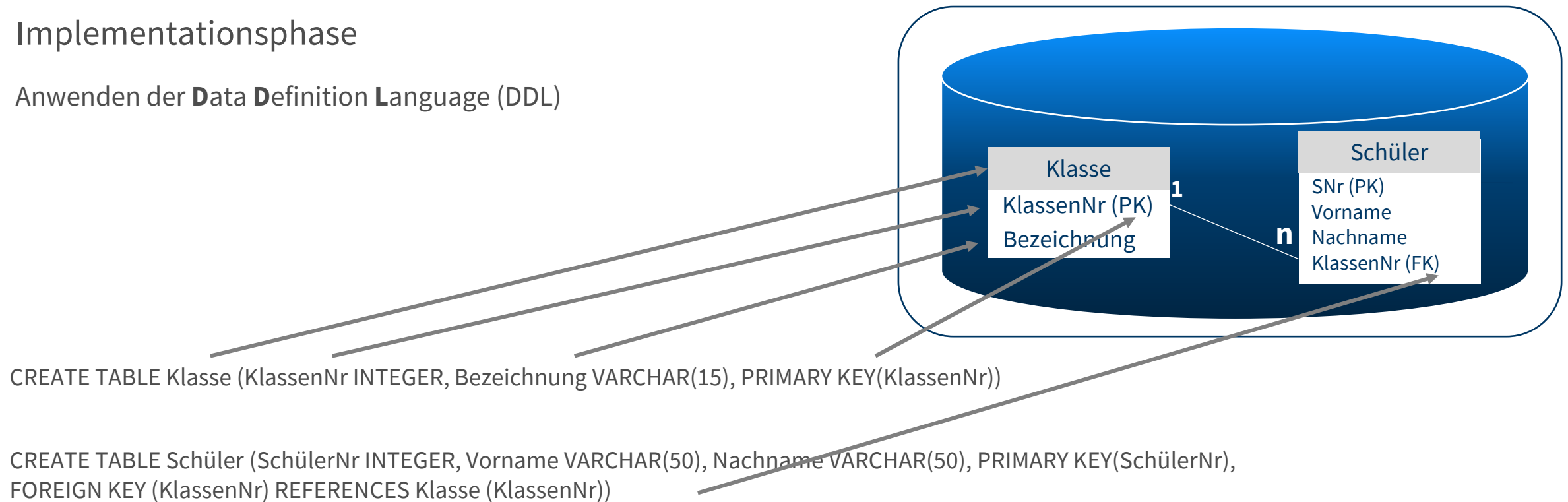
1. Ein Entitätstyp wird mit all seinen Attributen zu jeweils einer Tabelle zusammengefasst.
2. Jede Tabelle erhält einen Primärschlüssel aus einem Attribut oder einer Kombination aus Attributen, welche im ER-Modell gekennzeichnet wurden. Sind solche Attribute nicht vorhanden, dann wird ein künstlicher Primärschlüssel eingeführt.
3. Jede $m : n$ -Beziehung wird in einer eigenen Tabelle abgebildet. Diese Tabelle enthält die Primärschlüssel der beteiligten Entitätstypen als Fremdschlüssel und die Attribute, welche der Beziehung direkt zugeordnet sind.
4. Die Umsetzung der $1 : n$ -Beziehungen erfolgt, indem der Primärschlüssel der einen Tabelle als Fremdschlüssel in die andere Tabelle (n -Beziehung) eingefügt wird.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Den Prozess relationaler Datenbanken beschreiben

Implementationsphase

Anwenden der **Data Definition Language (DDL)**



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Kompetenzcheck ☒

Führen Sie folgende Aufgabe durch:

8.7.2_ Aufg_1-Designprozessphasen beschreiben



8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.2 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Der Prozess des Designs von relationalen Datenbanken verläuft in mehreren Phasen.
- b) Es darf keine Phase des Designprozesses übersprungen werden.
- c) Am Ende der letzten Phase soll eine funktionstüchtige Datenbank stehen.
- d) Das ER-Modell wird in der logischen Phase erstellt.
- e) Die Normalisierung des relationalen Datenmodells ist ein Bestandteil der konzeptionellen Phase.
- f) Aus einem schlechten logischen Design resultiert auch immer eine schlecht physikalische Umsetzung.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.2 Kompetenzcheck ☒

Führen Sie folgende Aufgaben im Lehrbuch durch:

Seite 270, Aufgabe 2 und 3



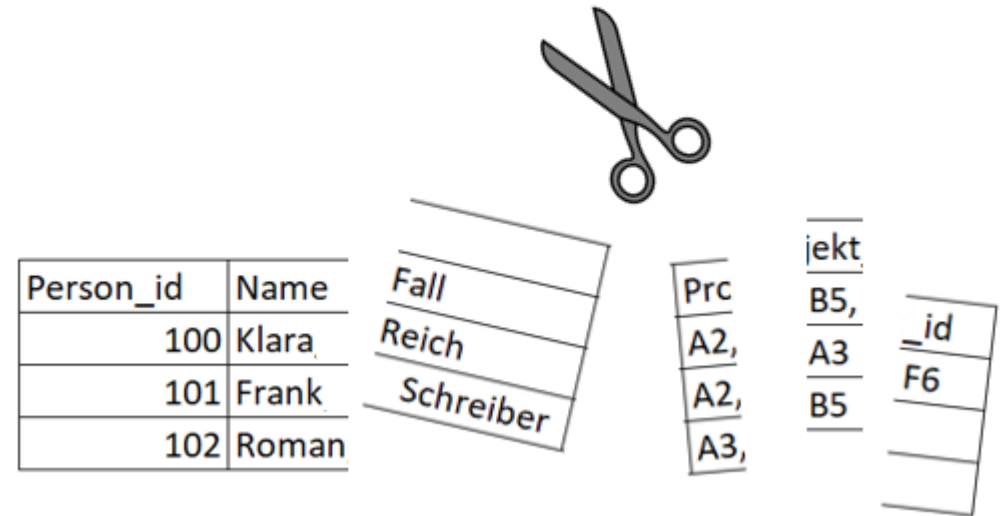
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Normalisierung

- Warum?
- Anomalien
- Normalisierungsschritte

Person_id	Name	Projekt_id
100	Klara, Fall	A2, B5, F6
101	Frank, Reich	A2, A3
102	Roman, Schreiber	A3, B5



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Datenkonsistenz

- Ist einer der Qualitätsansprüche für Datenbanken.
- Die Konsistenz ist als die Korrektheit von Daten innerhalb eines Datenbanksystems zu verstehen.
- Daten sind konsistent, wenn sie widerspruchsfrei sind.
- Es werden möglichst eindeutige Speicherorte festgelegt sowie die Verfahren wie neue Informationen bzw. Daten eingepflegt werden.



Re·dun·danz

/redunˈdants, Redundanz/

Substantiv, feminin [die] **BILDUNGSSPRACHLICH**

das Vorhandensein von eigentlich überflüssigen, für die Information nicht notwendigen Elementen;
Überladung mit Merkmalen



In·te·gri·tät

/Integrität/

Substantiv, feminin [die]

1. Makellosigkeit, Unbescholtenheit, Unbestechlichkeit
"die Integrität dieses Mannes ist unbestreitbar"
2. **POLITIK • RECHTSSPRACHE**
Unverletzlichkeit [eines Staatsgebietes]
"die territoriale Integrität eines Staates anerkennen, garantieren"

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Redundanz

Wiederholung derselben Daten ohne tatsächlichen Informationsgewinn

MitarbeiterNr	Name	AbteilungsNr	Abteilungsname
1	Maier	2	Buchhaltung
2	Schulz	2	Buchhaltung
3	Müller	1	Einkauf
4	Winkelmann	2	Buchhaltung
5	Günter	3	Produktion

Redundanzen können zu Anomalien führen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Anomalien

Mutationsanomalie

(Mutation der Daten)
Die Mutationsanomalie tritt auf, wenn die redundanten Daten (z. B. durch versehentliches Falschschreiben) verändert werden („mutieren“).

Einfügeanomalie

(Ungewolltes Dateneinfügen)
Von einer Einfügeanomalie spricht man, wenn durch das Einfügen von Daten ungewollt weitere Daten eingefügt werden müssen.

Löschanomalie

(Unbeabsichtigtes Datenlöschen)
Eine Löschanomalie liegt vor, wenn durch das Löschen von Daten weitere Informationen verloren gehen, die aber gar nicht gelöscht werden sollten.

Änderungsanomalie

(Datenänderung macht Folgenänderung notwendig)
Wenn das Ändern eines Datensatzes zwangsläufig das Ändern weiterer Datensätze nach sich zieht, spricht man von einer Änderungsanomalie.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Mutationsanomalie

(Mutation der Daten)

Die Mutationsanomalie tritt auf, wenn die redundanten Daten (z. B. durch versehentliches Falschschreiben) verändert werden („mutieren“).

<u>MitarbeiterNr</u>	Name	AbteilungsNr	Abteilungsname
1	Maier	2	Buchhaltung
2	Schulz	2	Buchh.
3	Müller	1	Einkauf
4	Winkelmann	2	Buchhaltung
5	Günter	3	Produktion

Der Name der Abteilung „mutierte“ auf Grund eines Schreibfehlers von der richtigen Schreibweise 'Buchhaltung' zu 'Buchh.'.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Einfügeanomalie

(Ungewolltes Dateneinfügen)

Von einer Einfügeanomalie spricht man, wenn durch das Einfügen von Daten ungewollt weitere Daten eingefügt werden müssen.

MitarbeiterNr	Name	AbteilungsNr	Abteilungsname
1	Maier	2	Buchhaltung
2	Schulz	2	Buchhaltung
3	Müller	1	Einkauf
4	Winkelmann	2	Buchhaltung
5	Günter	3	Produktion
6	?	4	Vertrieb

Die Abteilung 'Vertrieb' ist bereits bekannt und soll in die Datenbank eingetragen werden.

Wenn der Abteilung noch kein Mitarbeiter zugewiesen worden ist, wird ein Dummy-Mitarbeiter-Datensatz zum Eintragen der Abteilung notwendig.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Löschanomalie

(Unbeabsichtigtes Datenlöschen)

Eine Löschanomalie liegt vor, wenn durch das Löschen von Daten weitere Informationen verloren gehen, die aber gar nicht gelöscht werden sollten.

MitarbeiterNr	Name	AbteilungsNr	Abteilungsname
1	Maier	2	Buchhaltung
2	Schulz	2	Buchhaltung
3	Müller	1	Einkauf
4	Winkelmann	2	Buchhaltung
5	Günter	3	Produktion

Löscht man den letzten Mitarbeiter aus der Abteilung 'Einkauf', dann geht die Information über diese Abteilung ebenfalls verloren.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Änderungsanomalie

(Datenänderung macht Folgenänderung notwendig)

Wenn das Ändern eines Datensatzes zwangsläufig das Ändern weiterer Datensätze nach sich zieht, spricht man von einer Änderungsanomalie.

MitarbeiterNr	Name	AbteilungsNr	Abteilungsname
1	Maier	2	Buchhaltung
2	Schulz	2	Buchhaltung
3	Müller	1	Einkauf
4	Winkelmann	2	Buchhaltung
5	Günter	3	Produktion

Die Abteilung 'Buchhaltung' wird umbenannt in 'Rechnungswesen'
Diese eine Namensänderung führt zu sehr vielen Folgenänderungen.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Redundanzen

.... führen zu Anomalien!

Anomalien

.... führen zu inkonsistenten Daten!

Inkonsistenzen

.... sind der Tod jeder Datenbank!

Redundanzen müssen vermieden werden

Die Normalformen helfen beim Vermeiden und Beseitigen von
Redundanzen aus den Tabellen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Normalisierung

ist ein Verfahren zur Verringerung von Datenredundanz in relationalen Datenmodellen mit den Zielen, Anomalien zu vermeiden und die Datenkonsistenz zu erhöhen.

- Normalisieren ist die sinnvolle Aufteilung von Attributen in mehrere Tabellen.
- Über Beziehungen (PK→FK) werden die Tabellen miteinander verknüpft mit dem Ziel, Redundanzen, Inkonsistenzen und Anomalien zu vermeiden und beseitigen.
- Um eine Datenbankstruktur auf diese Weise zu optimieren, existieren Regeln zur sogenannten Normalisierung.
- In der Praxis beschränkt man sich auf die erste bis dritte Normalform. Die vierte und fünfte wird selten angewendet.
- Man wendet sie i. d. R. der Reihe nach an.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Abhängigkeiten

Funktionale
Zu jedem a genau ein b

Voll-Funktionale
Abhängig von allen
Schlüsselkandidaten (2. Form)

Transitive (indirekte)
a von b abhängig und b von c. So ist a
von c transitiv abhängig (3. Form)

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Normalformen

1. Normalform (1. NF)

Eine Tabelle liegt dann in der ersten Normalform vor, wenn alle Attribute der Tabelle nur einfache Werte aufweisen, d. h. wenn die Werte atomar vorliegen.

2. Normalform (2. NF)

Eine Tabelle liegt dann in zweiter Normalform vor, wenn sie der ersten Normalform genügt und alle Nichtschlüsselattribute vom gesamten Primärschlüssel abhängig sind.

3. Normalform (3. NF)

Eine Tabelle liegt dann in der dritten Normalform vor, wenn sie der zweiten Normalform genügt und kein Nichtschlüsselattribut transitiv abhängig ist.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

1. Normalform

Eine Tabelle liegt dann in der ersten Normalform vor, wenn alle Attribute der Tabelle nur einfache Werte aufweisen, d. h. wenn die Werte **atomar** vorliegen.

Der Begriff **atomar** (griechisch: *atomos* „unteilbar“) kennzeichnet ein Element, eine Struktur oder einen Sachverhalt, der nicht weiter zerlegbar/unterteilbar ist. Im Gegensatz dazu stehen die komplexen/zusammengesetzten Strukturen, die aus mehreren Teilbestandteilen aufgebaut sind.

[Quelle: wikipedia.org]

Bestellungen (unnormalisiert)

BestellNr	Datum	Kunde	Bestellposition
1	12.12.2021	Maier - KNr. 71	1. 2 Tische Nr. 12 2. 3 Schränke Nr. 88
2	14.12.2021	Maier - KNr. 71	1. 4 Stühle Nr. 67
3	14.12.2021	Schulz - KNr. 33	1. 4 Tische Nr. 12 2. 8 Stühle Nr. 67 3. 1 Schrank Nr. 88

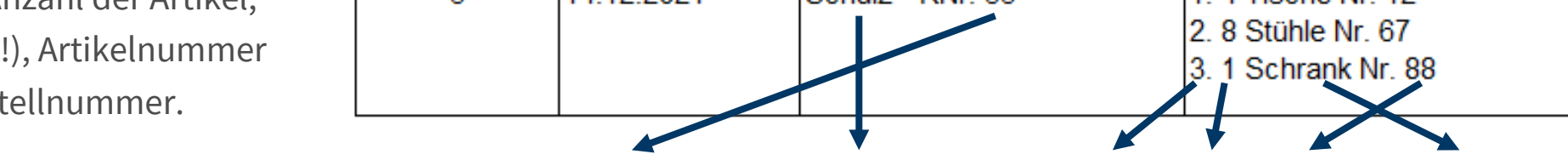
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

- Spalte *Kunde* enthält 2 Informationen:
Name und Kundennummer
→ **neue Spalten bilden**
- Spalte *Bestellposition* enthält in der ersten vier Informationen:
Position in der Bestellung, Anzahl der Artikel,
Artikelbezeichnung (Einzahl!), Artikelnummer
sowie 2 Zeilen zu dieser Bestellnummer.
→ **neue Zeilen bilden
pro Bestellposition**

Bestellungen (unnormalisiert)

BestellNr	Datum	Kunde	Bestellposition
1	12.12.2021	Maier KNr. 71	1. 2 Tische Nr. 12 2. 3 Schränke Nr. 88
2	14.12.2021	Maier - KNr. 71	1. 4 Stühle Nr. 67
3	14.12.2021	Schulz - KNr. 33	1. 4 Tische Nr. 12 2. 8 Stühle Nr. 67 3. 1 Schrank Nr. 88



BestellNr	Datum	KundenNr	KundenName	Position	Anzahl	ArtikelNr	Artikelname
1	12.12.2021	71	Maier	1	2	12	Tisch
1	12.12.2021	71	Maier	2	3	88	Schrank
2	14.12.2021	71	Maier	1	4	67	Stuhl
3	14.12.2021	33	Schulz	1	4	12	Tisch
3	14.12.2021	33	Schulz	2	8	67	Stuhl
3	14.12.2021	33	Schulz	3	1	88	Schrank

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

- Primärschlüssel ermitteln
- Bestellnummer ist nicht mehr eindeutig
- Zusammengesetzter Primärschlüssel
BestellNr, Position, KundenNr und ArtikelNr

BestellNr	Datum	KundenNr	KundenName	Position	Anzahl	ArtikelNr	Artikelname
1	12.12.2021	71	Maier	1	2	12	Tisch
1	12.12.2021	71	Maier	2	3	88	Schrank
2	14.12.2021	71	Maier	1	4	67	Stuhl
3	14.12.2021	33	Schulz	1	4	12	Tisch
3	14.12.2021	33	Schulz	2	8	67	Stuhl
3	14.12.2021	33	Schulz	3	1	88	Schrank

1. Normalform

<u>BestellNr</u>	<u>Position</u>	<u>Datum</u>	<u>KundenNr</u>	KundenName	Anzahl	<u>ArtikelNr</u>	Artikelname
1	1	12.12.2021	71	Maier	2	12	Tisch
1	2	12.12.2021	71	Maier	3	88	Schrank
2	1	14.12.2021	71	Maier	4	67	Stuhl
3	1	14.12.2021	33	Schulz	4	12	Tisch
3	2	14.12.2021	33	Schulz	8	67	Stuhl
3	3	14.12.2021	33	Schulz	1	88	Schrank

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

1. Normalform – zusammengefasst

Eine Relation (Tabelle) befindet sich
in 1. Normalform, wenn ...

- sie zweidimensional ist, d. h. ein Gebilde aus Zeilen und Spalten
- sich in jedem Datensatz nur Daten befinden, die zu einem Objekt der realen Welt gehören und jeder Datensatz nur einmal vorkommt,
- sich in jeder Spalte nur Daten befinden, die einem Attribut entsprechen und das Attribut nur einmal in der Relation vorkommt,
- für jedes Attribut nur **ein** Wert (atomar) eingetragen ist.

1. Normalform

<u>BestellNr</u>	<u>Position</u>	<u>Datum</u>	<u>KundenNr</u>	<u>KundenName</u>	<u>Anzahl</u>	<u>ArtikelNr</u>	<u>Artikelname</u>
1	1	12.12.2021	71	Maier	2	12	Tisch
1	2	12.12.2021	71	Maier	3	88	Schrank
2	1	14.12.2021	71	Maier	4	67	Stuhl
3	1	14.12.2021	33	Schulz	4	12	Tisch
3	2	14.12.2021	33	Schulz	8	67	Stuhl
3	3	14.12.2021	33	Schulz	1	88	Schrank

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Probleme

- Die Relation weist Redundanzen auf, z. B. treten Kundennamen, Datum und Artikeldaten im Beispiel mehrfach auf.
- Die Relation enthält voneinander unabhängige Sachverhalte, wie zum Beispiel Kunden und Artikel.

1. Normalform

<u>BestellNr</u>	<u>Position</u>	<u>Datum</u>	<u>KundenNr</u>	<u>KundenName</u>	<u>Anzahl</u>	<u>ArtikelNr</u>	<u>Artikelname</u>
1	1	12.12.2021	71	Maier	2	12	Tisch
1	2	12.12.2021	71	Maier	3	88	Schrank
2	1	14.12.2021	71	Maier	4	67	Stuhl
3	1	14.12.2021	33	Schulz	4	12	Tisch
3	2	14.12.2021	33	Schulz	8	67	Stuhl
3	3	14.12.2021	33	Schulz	1	88	Schrank

Jede Relation sollte nur einen Sachverhalt abbilden!

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren


2. Normalform

Eine Tabelle liegt dann in zweiter Normalform vor, wenn sie der ersten Normalform genügt und alle Nichtschlüsselattribute vom **gesamten** Primärschlüssel abhängig sind.

Wird immer angewendet, wenn es zusammengesetzte Primärschlüssel (aus mehr als einem Attribut) gibt.

1. Normalform

<u>BestellNr</u>	<u>Position</u>	Datum	<u>KundenNr</u>	KundenName	Anzahl	<u>ArtikelNr</u>	Artikelname
1	1	12.12.2021	71	Maier	2	12	Tisch
1	2	12.12.2021	71	Maier	3	88	Schrank
2	1	14.12.2021	71	Maier	4	67	Stuhl
3	1	14.12.2021	33	Schulz	4	12	Tisch
3	2	14.12.2021	33	Schulz	8	67	Stuhl
3	3	14.12.2021	33	Schulz	1	88	Schrank



Primärschlüssel

Nichtschlüsselattribute

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

2. Normalform - Vorgehen

- Zerlegen Sie die Relation in kleinere Relationen, sodass in jeder Relation alle Nicht-schlüsselfelder nur noch vom Primärschlüssel abhängen.
- Besteht ein Primärschlüssel aus mehreren Attributen, ist zu prüfen, ob es Attribute gibt, die eigentlich nur von einem Teil des Primärschlüssels abhängen. Ist das der Fall, so sind dieser Teil des Primärschlüssels und die zugehörigen Attribute in eine neue Relation zu bringen.

1. Normalform

<u>BestellNr</u>	<u>Position</u>	Datum	<u>KundenNr</u>	KundenName	Anzahl	<u>ArtikelNr</u>	Artikelname
1	1	12.12.2021	71	Maier	2	12	Tisch
1	2	12.12.2021	71	Maier	3	88	Schrank
2	1	14.12.2021	71	Maier	4	67	Stuhl
3	1	14.12.2021	33	Schulz	4	12	Tisch
3	2	14.12.2021	33	Schulz	8	67	Stuhl
3	3	14.12.2021	33	Schulz	1	88	Schrank

Primärschlüssel

Nichtschlüsselattribute

8.7 Datenbanklösungen bedarfsgerecht entwickeln

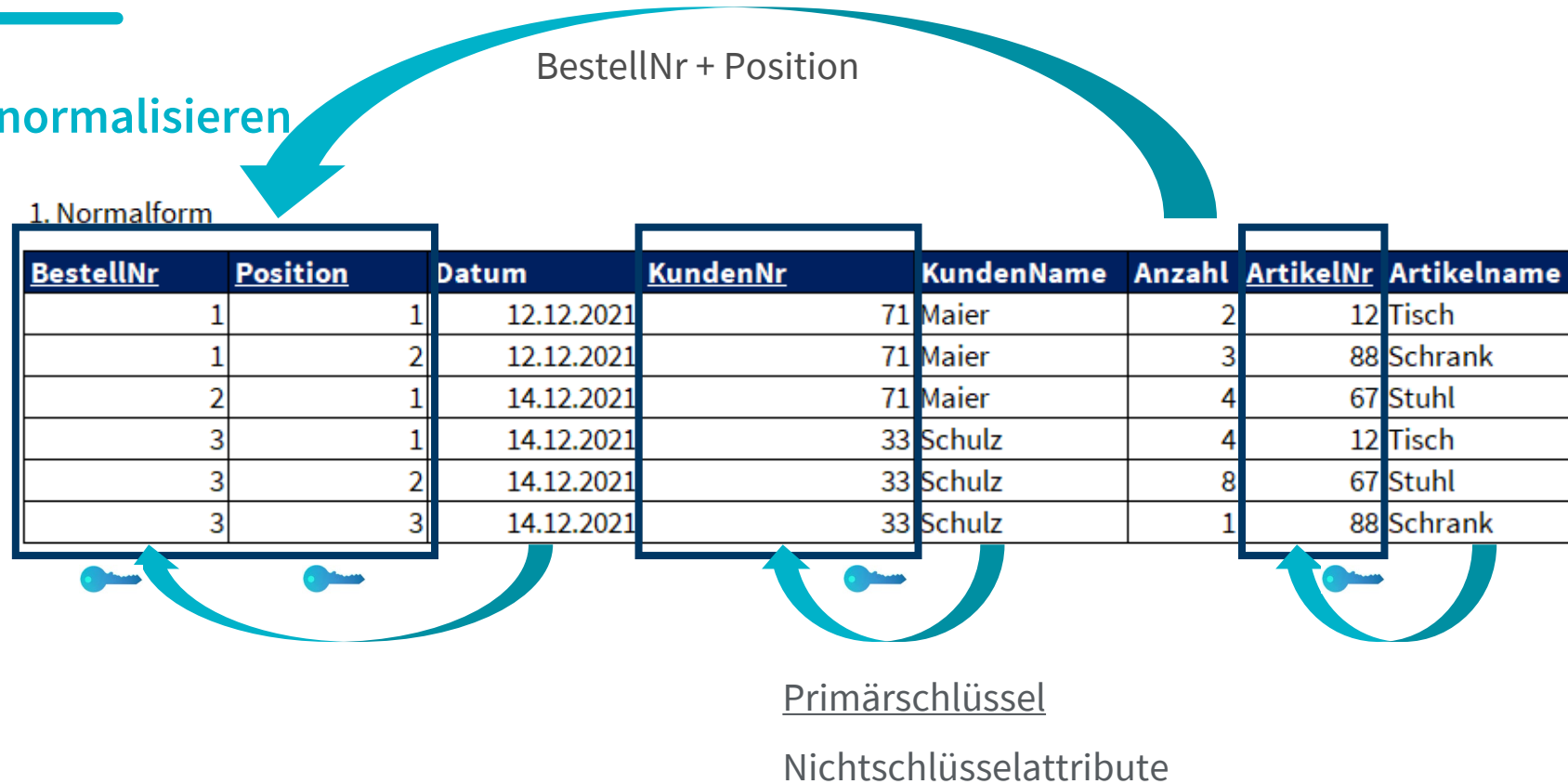
8.7.3 Relationale Datenbanken normalisieren

2. Normalform

- Abhängigkeiten der Nichtschlüselfelder von Teilen des Primärschlüssels?

BestellNr:

Datum



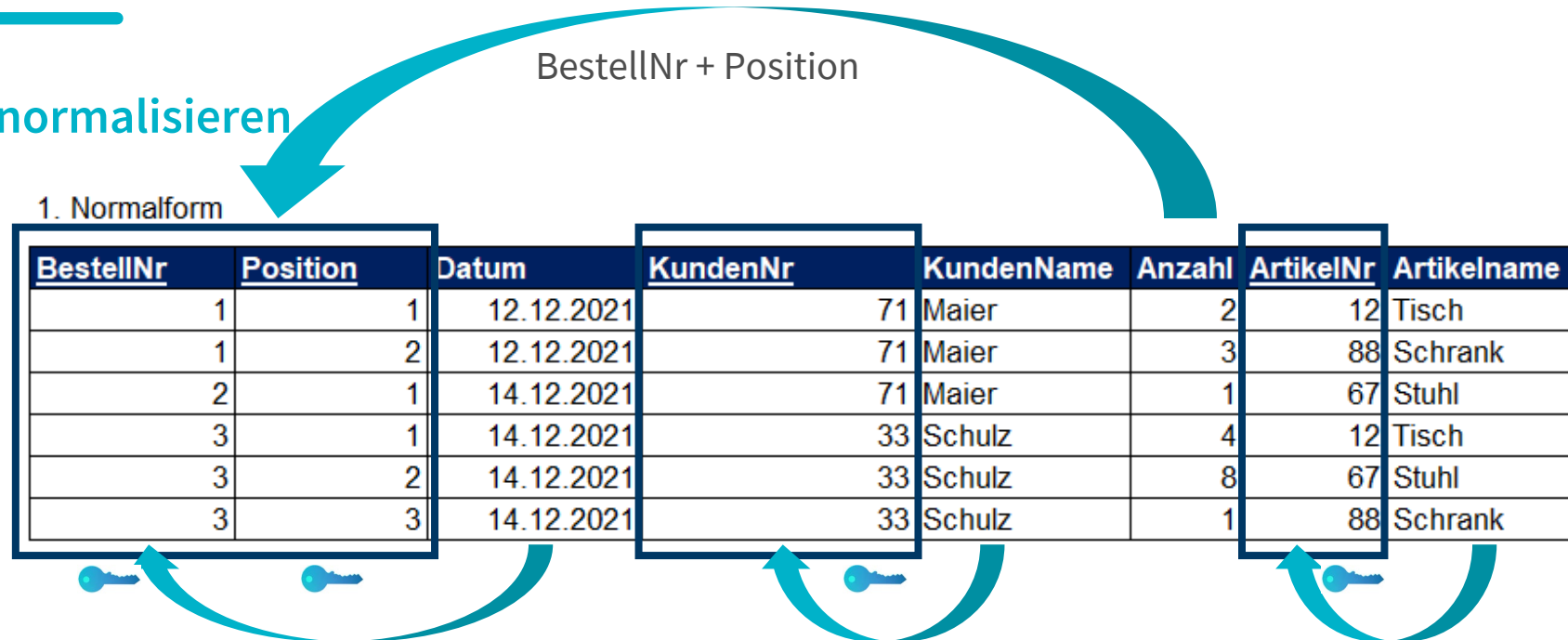
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

2. Normalform

- Abhängigkeiten der Nichtschlüselfelder von Teilen des Primärschlüssels?

- **BestellNr:**
Datum
- **KundenNr:**
Name
- **ArtikelNr:**
Artikelname
- **BestellNr, Position:**
Anzahl



Bestellung(BestellNr, Datum)
Kunde(KundenNr, KundenName)
Artikel(ArtikelNr, Artikelname)
Bestellposition(BestellNr, Position, Anzahl)

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

2. Normalform

Redundanzen pro Bestellung
sind beseitigt

Bestellung

<u>BestellNr</u>	Datum
1	12.12.2021
2	14.12.2021
3	14.12.2021



Kunde

<u>KundenNr</u>	KundenName
71	Maier
33	Schulz



Artikel

<u>ArtikelNr</u>	Artikelname
12	Tisch
67	Stuhl
88	Schrank



Bestellung(BestellNr, Datum)

Kunde(KundenNr, KundenName)

Artikel(ArtikelNr, Artikelname)

Bestellposition(BestellNr, Position, Anzahl)

Bestellposition

<u>BestellNr</u>	<u>Position</u>	Anzahl
1	1	2
1	2	3
2	1	4
3	1	4
3	2	8
3	3	1



? Beziehung

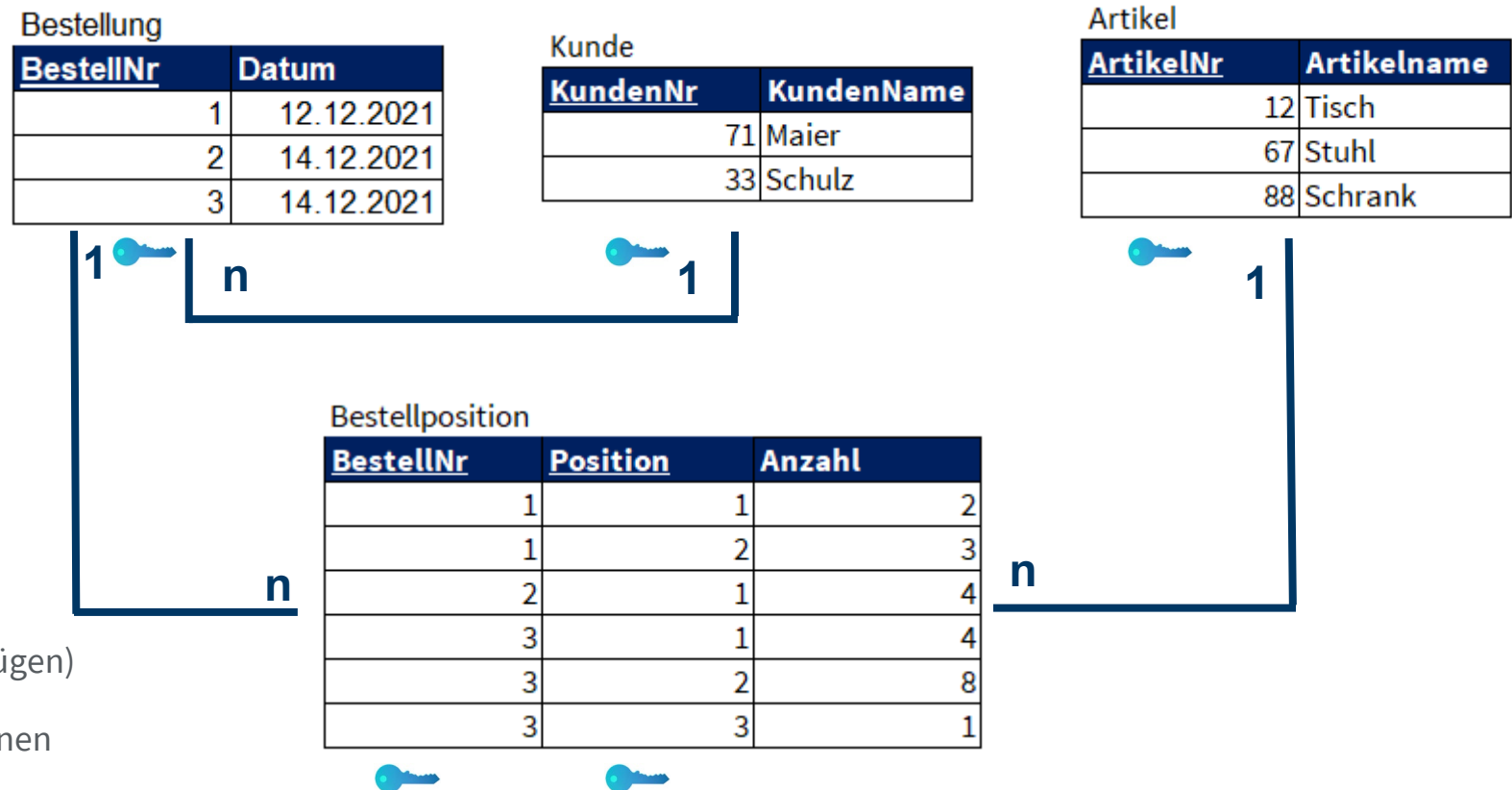
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

2. Normalform

Fremdschlüssel einfügen, um Beziehungen abzubilden

- Ein Kunde kann mehrere Bestellungen haben
(KundenNr in Tabelle Bestellungen hinzufügen)
- Ein Artikel ist in mehreren Bestellungen und dort in den jeweiligen Positionen aufgeführt.
(ArtikelNr in Tabelle Bestellposition hinzufügen)
- Eine Bestellung hat mehrere Bestellpositionen
(BestellNr in Tabelle Bestellposition)



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

2. Normalform

Bestellung		FK
<u>BestellNr</u>	Datum	KundenNr
1	12.12.2021	71
2	14.12.2021	71
3	14.12.2021	33



Kunde	
<u>KundenNr</u>	KundenName
71	Maier
33	Schulz



Artikel	
<u>ArtikelNr</u>	Artikelname
12	Tisch
67	Stuhl
88	Schrank



Bestellung(BestellNr, Datum, KundenNr)
Kunde(KundenNr, KundenName)
Artikel(ArtikelNr, Artikelname)
Bestellposition(BestellNr, Position, Anzahl, ArtikelNr)

Primärschlüssel

Nichtschlüsselattribute

Bestellposition			
<u>BestellNr</u>	<u>Position</u>	Anzahl	ArtikelNr
1	1	2	12
1	2	3	88
2	1	4	67
3	1	4	12
3	2	8	67
3	3	1	88



8.7 Datenbanklösungen bedarfsgerecht entwickeln

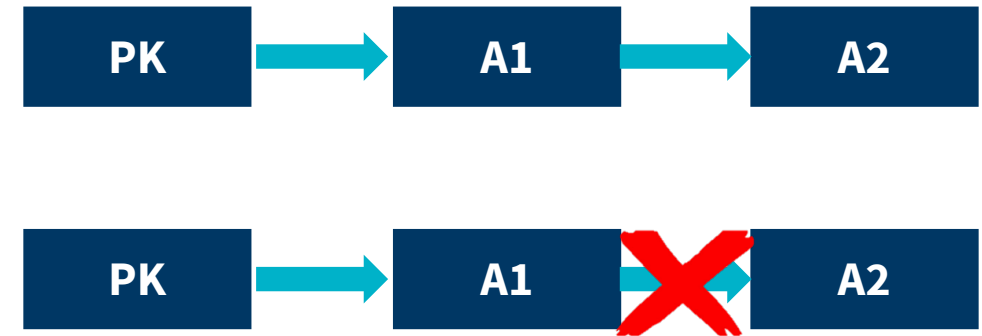
8.7.3 Relationale Datenbanken normalisieren

3. Normalform

Eine Tabelle liegt dann in der dritten Normalform vor, wenn sie der zweiten Normalform genügt und kein Nichtschlüsselattribut transitiv abhängig ist.

Sobald ein Nichtschlüselfeld nur über ein anderes Nichtschlüselfeld identifizierbar ist, wird von transitiver Abhängigkeit gesprochen.

Transitive Abhängigkeiten verursachen ebenfalls Datenredundanz und -inkonsistenz.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

3. Normalform

Eine Tabelle liegt dann in der dritten Normalform vor, wenn sie der zweiten Normalform genügt und kein Nichtschlüsselattribut transitiv abhängig ist.

Sobald ein Nichtschlüselfeld nur über ein anderes Nichtschlüselfeld identifizierbar ist, wird von transitiver Abhängigkeit gesprochen

Transitive Abhängigkeiten verursachen ebenfalls Datenredundanz und -inkonsistenz

Lieferanten

LieferantenNr	Name	Strasse	PLZ	Ort
1	Bit & Byte GmbH	Neugasse 12	12345	Irgendwo
2	Surf KG	Bahnhofstr. 25	24680	Geldheim
3	SWE AG	Alter Weg 93	12345	Irgentwo
4	DEV GmbH	Hauptstrasse 31	69469	Weinheim
5	DB-Develop GbR	Referenzweg 3	13579	IT-Hausen



1. NF:



3. NF:



2. NF:



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

3. Normalform

Redundanzen beseitigt

Lieferanten(LieferantenNr, Name, Strasse, *PLZ*)

Orte(PLZ, Ort)

Lieferanten

<u>LieferantenNr</u>	Name	Strasse	FK PLZ
1	Bit & Byte GmbH	Neugasse 12	12345
2	Surf KG	Bahnhofstr. 25	24680
3	SWE AG	Alter Weg 93	12345
4	DEV GmbH	Hauptstrasse 31	69469
5	DB-Develop GbR	Referenzweg 3	13579

Orte

<u>PLZ</u>	Ort
12345	Irgendwo
24680	Geldheim
69469	Weinheim
13579	IT-Hausen



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

3. Normalform

Kunde

<u>KundenNr</u>	Vorname	Nachname	Herkunftsland
1	Frank	Reich	Frankreich
2	Klara	Fall	Deutschland
3	Karl	Auer	Deutschland
4	Roman	Schreiber	Oesterreich

Kunde

<u>KundenNr</u>	Vorname	Nachname	FK
1	Frank	Reich	1
2	Klara	Fall	2
3	Karl	Auer	2
4	Roman	Schreiber	3

Land

<u>HLNr</u>	Herkunftsland
1	Frankreich
2	Deutschland
3	Oesterreich

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Normalformen

BCNF Normalform

(Boyce-Codd-Normalform)

Nur Abhängigkeiten vom Schlüssel
zugelassen

4. Normalform

Keine mehrwertigen
Abhängigkeiten

5. Normalform

Nur triviale Verbundabhängigkeit

8.7 Datenbanklösungen bedarfsgerecht entwickeln

Vorteile

- Im Idealfall enthält eine normalisierte Datenbank keine vermeidbaren Redundanzen mehr und ist in sich vollständig konsistent.
- Die Datenbank ist durch die Normalisierung effizienter organisiert, flexibler und belegt keinen unnötigen Speicherplatz mehr.
- Gute Performance, da Tabellen einfach zu durchsuchen sind.
- Die Datensätze lassen sich gut miteinander verbinden.
- Die Datenbank ist leichter wartbar.
- Es treten keine Anomalien mehr auf.

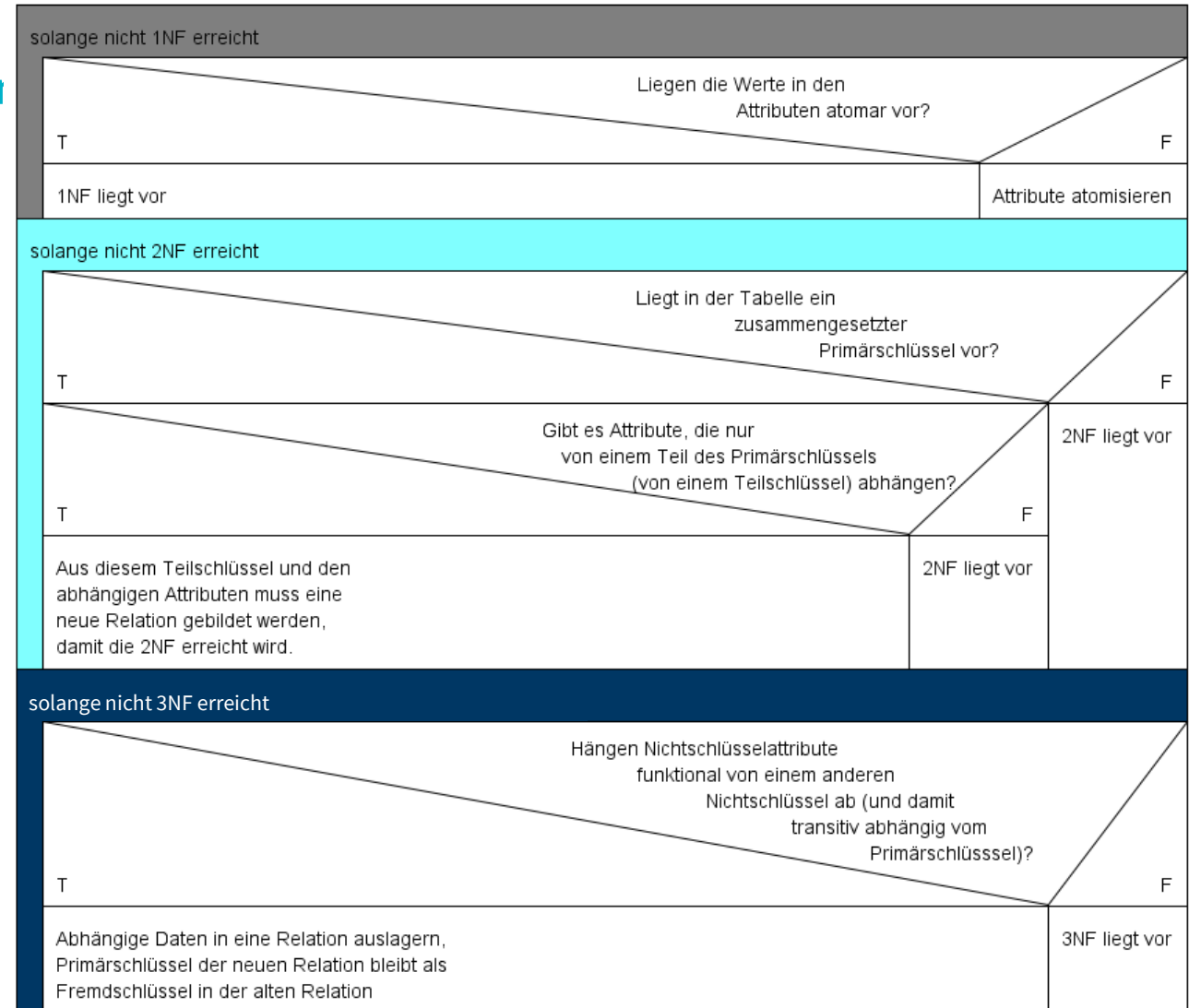
Nachteile

- Es entstehen viele kleine Tabellen, die die Leistung der Datenbank negativ beeinflussen können.
- Aufgrund der vielen künstlichen Schlüssel und der erforderlichen zusätzlichen Verknüpfungen wird das System komplexer, was zu größerer Fehleranfälligkeit führen kann.
- Zusätzliche Schlüssel erfordern zusätzlichen Speicherplatz, stellen also auch eine Art von Redundanz dar.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Relationale Datenbanken normalisieren

Normalisieren bis zur 3. Normalform



8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.3 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Normalisierung ist ein Verfahren, um Datenredundanz in relationalen Datenmodellen zu beseitigen.
- b) Eine Tabelle liegt dann in der 1. NF vor, wenn alle Werte der Tabelle atomar sind.
- c) Eine Tabelle liegt dann in der 2. NF vor, wenn sie der 1. NF genügt und alle Nichtschlüsselattribute vom gesamten Primärschlüssel abhängig sind.
- d) Eine Tabelle liegt dann in der 3. NF vor, wenn sie der 2. NF genügt und alle Nichtschlüselfelder transitiv unabhängig sind.
- e) Es gibt drei Normalformen.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.3 Kompetenzcheck ☒



Lösen Sie die Aufgaben im Lehrbuch Seite 274/275.

Aufgabe 2, 4

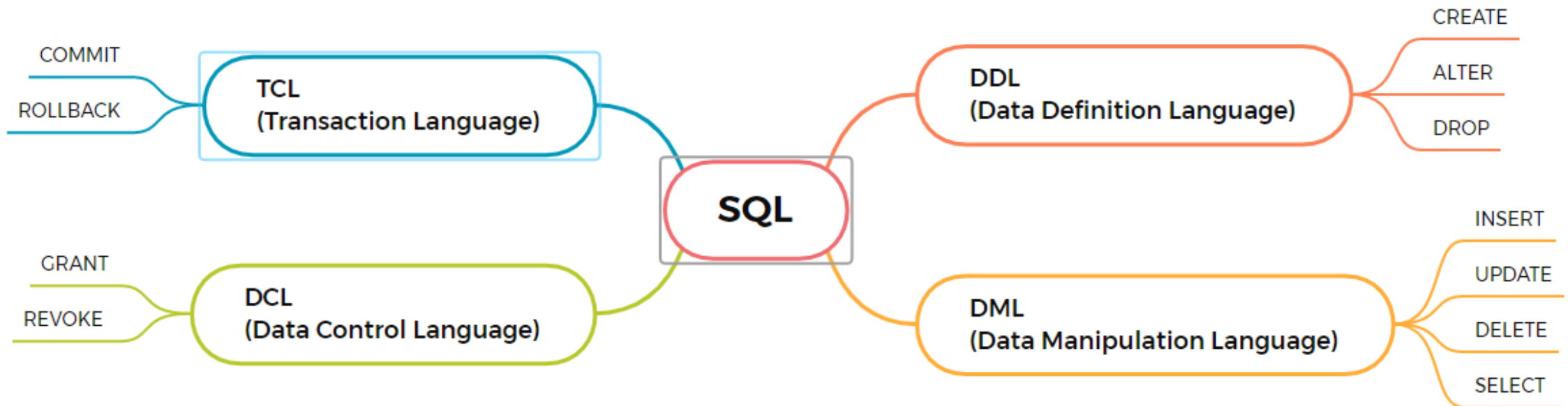
OPTIONAL:

8.7.3_Aufg_3_Normalisieren.xlsx

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Sie sollen Ihr Wissen über SQL um neue Anweisungen erweitern.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

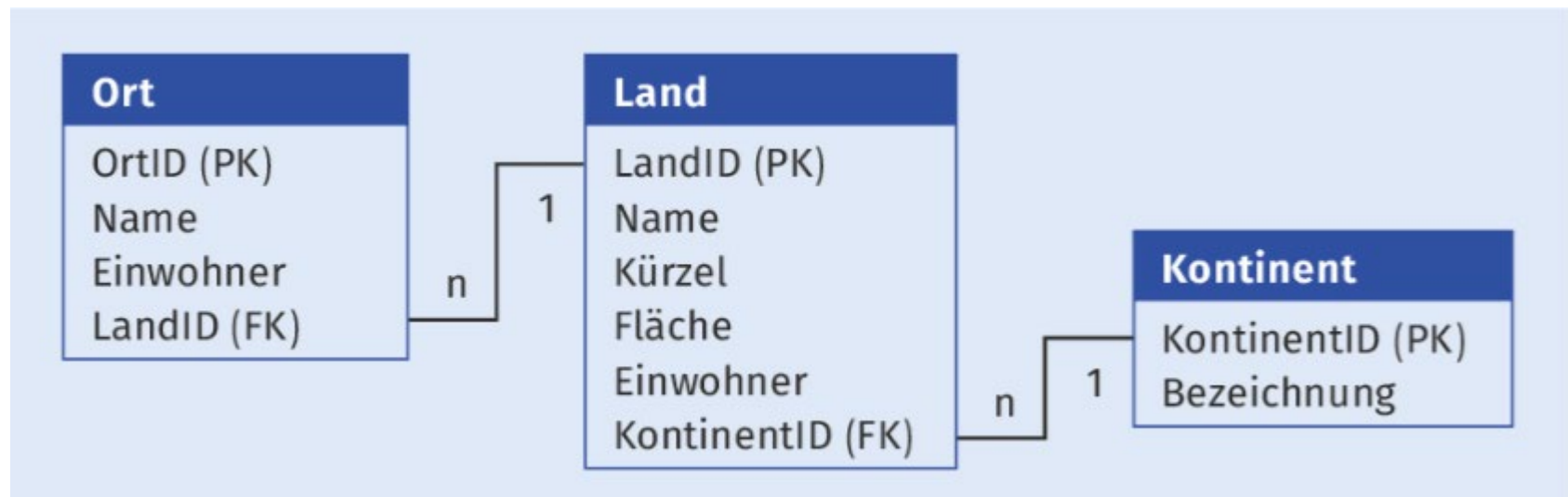
8.7.4 Das Basiswissen über SQL erweitern und anwenden

Sie sollen Ihr Wissen über SQL um neue Anweisungen erweitern.

Einleitung: wie gehen wir vor,

was wissen wir noch?

- Basistabellen
- Datenbank anlegen
- Tabellen Anlegen
- Daten erfassen
- Abfragen



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Anlegen einer Tabelle

Syntax	CREATE TABLE Tabellename (Spaltenname1 Datentyp1 [NOT NULL] [AUTOINCREMENT] [PRIMARY KEY] Spaltenname2 Datentyp2 [FOREIGN KEY REFERENCES] ...);
Erläuterung	Tabellename = Name der Tabelle, Spaltenname = Name der jeweiligen Spalte Datentyp = Datentyp der jeweiligen Spalte (z . B . INT, VARCHAR (50) etc .) NOT NULL = Wert der Spalte darf nicht NULL sein, AUTOINCREMENT = Zahl wird bei jedem neuen Datensatz automatisch um eins erhöht PRIMARY KEY = Spalte wird zum Primärschlüssel, FOREIGN KEY = Spalte wird zum Fremdschlüssel REFERENCES = Verknüpfung mit dem Primärschlüssel einer anderen Tabelle
Beispiel	CREATE TABLE Person (PersonID INT NOT NULL AUTOINCREMENT PRIMARY KEY, Name VARCHAR (50) NOT NULL, Vorname VARCHAR (50), Groesse FLOAT, Gewicht FLOAT, Geburtsdatum DATE OrtID INT FOREIGN KEY REFERENCES Ort (OrtID));

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Ändern einer Tabelle: Datentyp einer Spalte ändern	
Syntax	ALTER TABLE Tabellename MODIFY COLUMN Spaltenname <i>Datentyp Optionen</i> ;
Erläuterung	Tabellename = Name der Tabelle, in welcher eine neue Spalte bearbeitet werden soll Spaltenname = Name der zu bearbeitenden Spalte Datentyp = neuer Datentyp der ausgewählten Spalte (z. B. INT, VARCHAR (50) etc.)
Beispiel	ALTER TABLE Person MODIFY COLUMN Bemerkungen VARCHAR(250);

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Ändern einer Tabelle: Spalte löschen

Syntax	ALTER TABLE Tabellename DROP COLUMN Spaltenname;
Erläuterung	Tabellename = Name der Tabelle, in welcher eine neue Spalte gelöscht werden soll Spaltenname = Name der zu löschenden Spalte
Beispiel	ALTER TABLE Person DROP COLUMN Bemerkungen;

Löschen einer Tabelle

Syntax	DROP TABLE Tabellename;
Erläuterung	Tabellename = Name der Tabelle, die gelöscht werden soll
Beispiel	DROP TABLE Person;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Einfügen von Datensätzen	
Syntax	INSERT INTO Tabellename (Spalte1, Spalte2, ... VALUES (Wert1, Wert2, ...);
Erläuterung	Tabellename = Name der Tabelle, in die ein Datensatz eingetragen werden soll Spalte = Name der Spalte, in die ein Wert geschrieben werden soll Wert = Wert, der in die entsprechende Spalte geschrieben werden soll
Beispiel	INSERT INTO Person (Vorname, Name, Gewicht) VALUES ("Hans", "Müller", 80.5);

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Ändern von Datensätzen	
Syntax	UPDATE Tabellename SET Spalte = Wert WHERE Bedingungen;
Erläuterung	Tabellename = Name der Tabelle, welche aktualisiert werden soll Spalte = Name der Spalte, in welcher der Wert geändert werden soll Wert = neuer Wert, der in die entsprechende Spalte geschrieben werden soll Bedingungen = Einschränkung der Datensätze, die aktualisiert werden sollen
Beispiel	UPDATE Person SET Gewicht = 79.50 WHERE PersonID = 21 OR PersonID = 43;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Löschen von Datensätzen	
Syntax	DELETE FROM Tabellenname WHERE <i>Bedingungen</i> ;
Erläuterung	Tabellenname = Name der Tabelle aus der Datensätze gelöscht werden sollen Spalte = Name der Spalte, in die ein Wert geschrieben werden soll Bedingungen = Einschränkung auf jene Datensätze, die gelöscht werden sollen
Beispiel	DELETE FROM Person WHERE Name IS NULL;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Datenabfrage mit SELECT	
Syntax	SELECT [DISTINCT] Spalte1 [AS Alias], Spalte2 , FROM Tabelle1 [Alias], Tabelle2 [Alias] ... [WHERE Bedingungen] [GROUP BY Spalte [HAVING Bedingungen]] [ORDER BY Spalte1 [ASC DESC], Spalte2 ...];
Erläuterung	<p>Spalte1..n = Namen der Spalten, deren Werte angezeigt werden sollen</p> <p>Alias = Temporärer Name der Spalte oder der Tabelle</p> <p>Tabelle = Angabe der Tabellen, aus denen die Datensätze abgefragt werden</p> <p>WHERE Bedingung = dient zum Filtern der Datensätze</p> <p>GROUP BY und HAVING = dient zum Gruppieren der Datensätze</p> <p>ORDER BY = dient zum Sortieren der Datensätze</p>
einfaches Beispiel	SELECT Person.Name , Person.Vorname , Person.Geburtsdatum FROM Person ;
alle Spalten	SELECT * FROM Person ;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Datenabfrage mit SELECT

Syntax	SELECT [DISTINCT] Spalte1 [AS Alias], Spalte2 , FROM Tabelle1 [Alias], Tabelle2 [Alias] ... [WHERE Bedingungen] [GROUP BY Spalte [HAVING Bedingungen]] [ORDER BY Spalte1 [ASC DESC], Spalte2 ...];
--------	---

einfaches Beispiel	SELECT Person.Name, Person.Vorname, Person.Geburtsdatum FROM Person;
-----------------------	---

alternativ mit Alias	SELECT P.Name, P.Vorname, P.Geburtsdatum FROM Person AS P;
-------------------------	--

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Datenabfrage mit SELECT	
Syntax	SELECT [DISTINCT] Spalte1 [AS Alias], Spalte2 , FROM Tabelle1 [Alias], Tabelle2 [Alias] ... [WHERE Bedingungen] [GROUP BY Spalte [HAVING Bedingungen]] [ORDER BY Spalte1 [ASC DESC], Spalte2 ...];
jeder Name nur einmal	SELECT DISTINCT Person.Name FROM Person;
mit Bedingungen	SELECT P.Name, P.Vorname FROM Person AS P WHERE P.Groesse > 1.80 AND P.Gewicht < 100 AND P.Name = 'Müller';
mit Wertebereich	SELECT P.Name, P.Vorname FROM Person AS P WHERE P.Groesse BETWEEN 1.60 AND 1.80;

Mit **BETWEEN** sind auch die Datensätze in der Ergebnismenge enthalten, bei denen das Feld der Where-Bedingung exakt die Unter- und Obergrenze als Werte enthält.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Platzhalter mit Operator LIKE

%	Prozentzeichen: steht für kein Zeichen, ein Zeichen oder mehrere Zeichen. Entspricht in DOS dem Wildcard-Zeichen *
----------	---

_	Unterstrich: repräsentiert immer genau ein Zeichen. Entspricht in DOS dem Wildcard-Zeichen ?
----------	---

Beispiele	SELECT P.Name, P.Vorname FROM Person P WHERE P.Name LIKE '%m%'; Liefert alle Namen, in denen ein 'm' enthalten ist.
------------------	--

	SELECT P.Name, P.Vorname FROM Person P WHERE P.Name LIKE 'm%'; Liefert alle Namen, die mit 'm' beginnen.
--	---

	SELECT P.Name, P.Vorname FROM Person P WHERE P.Name LIKE '_m%'; Liefert alle Namen die an zweiter Stelle ein 'm' haben.
--	--

	SELECT P.Name, P.Vorname FROM Person P WHERE P.Name LIKE '%m'; Liefert alle Namen, die mit einem 'm' enden.
--	--

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

In SQL-Statements können arithmetische Ausdrücke zur Berechnung mehrerer Felde enthalten sein.

Hinter der Formel sollte ein Alias definiert werden, der in der Anzeige als Spaltenüberschrift erscheint. Ohne Alias würde die gesamte Formel als Spaltenüberschrift angezeigt.

Abfrage mit Berechnung

Aufgabe

Es sollen alle Namen, Vornamen und der BMI angezeigt werden.
BMI ist der Bodymaßindex: $BMI = \text{Gewicht [kg]} / (\text{Größe[m]} * \text{Größe[m]})$

```
SELECT P.Name, P.Vorname,  
  ( P.Gewicht / ( P.Groesse * P.Groesse )) AS BMI  
FROM Person P;
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Sortierung der Ergebnismenge	
Syntax	Sortieranweisungen werden immer am Ende des SQL-Statements formuliert. . . . [ORDER BY Spalte1 [ASC DESC], Spalte2 ...];
ASC	ASCENDING -- Aufsteigende Sortierung (default)
DESC	DESCENDING -- Absteigende Sortierung
Beispiel	SELECT P.Name, P.Vorname FROM Person P ORDER BY P.Name DESC ; Absteigend sortierte Ausgabe

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Aggregatfunktion	Bedeutung	Beispiel
COUNT(Spalte)	Liefert die Anzahl der Datensätze, die in der Spalte einen definierten Wert enthalten. Werte, die NULL sind, werden nicht gezählt.	SELECT COUNT (P.PersonID) FROM Person P;
	Liefert die Anzahl der Zeilen der Tabelle.	SELECT COUNT (*) FROM Person P;
SUM(Spalte)	Liefert die Summe der Werte der Spalte Gewicht.	SELECT SUM (P.Gewicht) FROM Person P;
MIN(Spalte)	Liefert den kleinsten Wert der Spalte Gewicht.	SELECT MIN (P.Gewicht) FROM Person P;
MAX(Spalte)	Liefert den größten Wert der Spalte Gewicht.	SELECT MAX (P.Gewicht) FROM Person P;
AVG(Spalte) average	Liefert den Durchschnitt über alle Werte der Spalte Groesse.	SELECT AVG (P.Groesse) FROM Person P;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

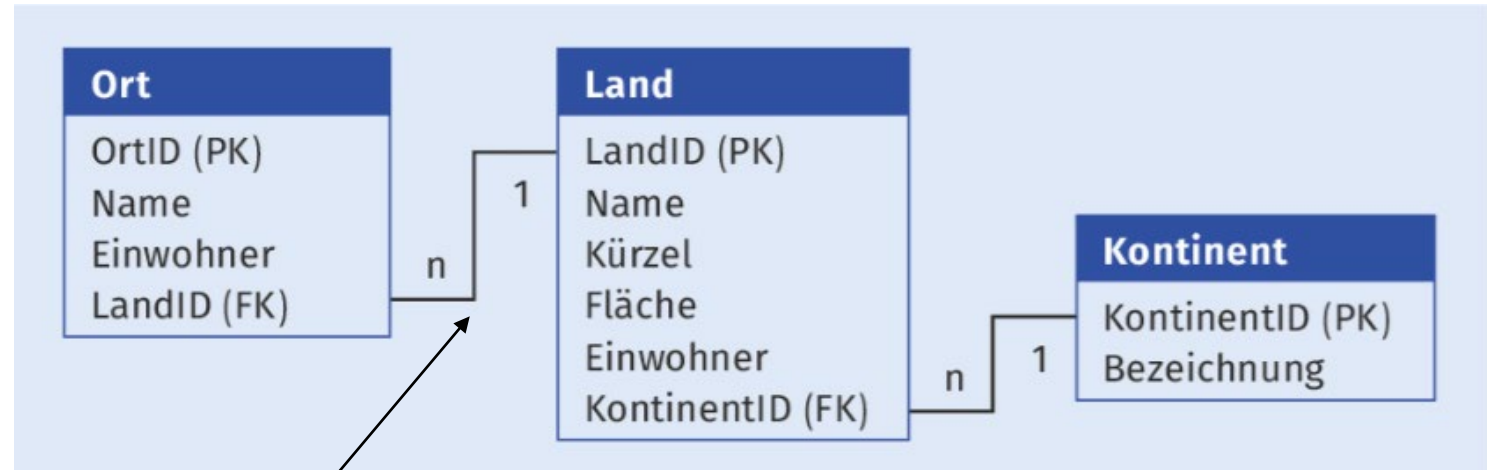
8.7.4 Das Basiswissen über SQL erweitern und anwenden

Gruppierung der Ergebnismengen	
Syntax	SELECT ... FROM ... GROUP BY Spalte HAVING Bedingungen ;
Erläuterung	Spalte = Spalte, nach der gruppiert werden soll Bedingungen = Bedingungen, nach denen die Gruppen gefiltert werden
Beispiel	SELECT AVG (P.Groesse) FROM Person P GROUP BY P.OrtID ; Mit GROUP BY soll die Durchschnittsgröße von allen Personen je Wohnort ermittelt werden.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 (1) Abfragen über mehrere Tabellen

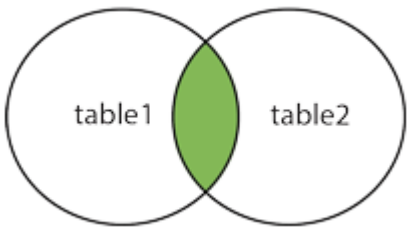
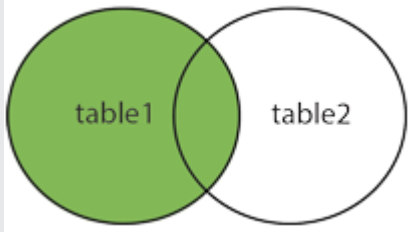
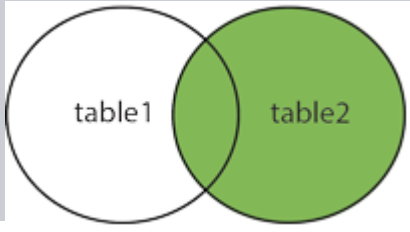
Wenn die Verbindung zwischen zwei Tabellen definiert ist, kann auf alle Informationen der beteiligten Tabellen zugegriffen werden.



SELECT Ort.Name, Ort.Einwohner,
Land.Name, Land.Kürzel, Land.Fläche,
Land.Einwohner
WHERE Ort.LandID = Land.LandID

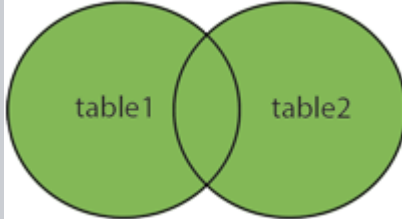
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Schlüsselwort	Beschreibung		Beispiel
INNER JOIN	Liefert die Werte der linken Tabelle und die Werte der rechten Tabelle, welche miteinander verknüpft sind.		SELECT ... FROM table1 A INNER JOIN table2 B ON A.ID = B.ID
LEFT JOIN	Liefert alle Werte der linken Tabelle und alle Werte der rechten Tabelle, die eine Verknüpfung mit der linken Tabelle besitzen.		SELECT ... FROM table1 A LEFT JOIN table2 B ON A.ID = B.ID
RIGHT JOIN	Liefert alle Werte der rechten Tabelle und alle Werte der linken Tabelle, die eine Verknüpfung mit der rechten Tabelle besitzen.		SELECT ... FROM table1 A RIGHT JOIN table2 B ON A.ID = B.ID

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Schlüsselwort	Beschreibung		Beispiel
FULL JOIN	Liefert alle Werte der rechten und der linken Tabelle, unabhängig davon, ob eine Verknüpfung besteht oder nicht.		SELECT ... FROM table1 A FULL JOIN table2 B ON A.ID = B.ID

FULL JOIN wird von MySQL nicht unterstützt

UNION	Alternative Lösung UNION verbindet die Ergebnismengen von zwei (oder mehreren) Abfragen zu einer Ausgabe. Doppelte Werte werden ignoriert. Einschränkung: die Anzahl der Ergebnisspalten müssen übereinstimmen.	SELECT ... FROM table1 A LEFT JOIN table2 B ON A.ID = B.ID UNION SELECT ... FROM table1 A RIGHT JOIN table2 B ON A.ID = B.ID
-------	--	---

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Unterabfrage (Subquery) mit einem Ergebniswert

Alle Orte anzeigen, deren Einwohnerzahl über der durchschnittlichen Einwohnerzahl aller Orte liegt.

```
SELECT O.Name, O.Einwohner  
FROM Ort O  
WHERE O.Einwohner  
>  
(SELECT AVG(O.Einwohner) FROM Ort O);
```

Mögliche Vergleichsoperatoren

=, >, >=, <, <=

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden

Unterabfrage (Subquery) mit Ergebnisliste

Suche Orte, deren Name auch als Ländername in der Ländertabelle enthalten ist.

```
SELECT Name FROM Ort WHERE Name  
IN  
(SELECT Name FROM Land);
```



Mengenoperatoren

IN	Prüft, ob ein bestimmter Wert in einer Menge von Werten der Unterabfrage enthalten ist.
ALL	Prüft, ob eine Bedingung für alle Ergebnisse der Unterabfrage erfüllt ist.
ANY	Prüft, ob eine Bedingung für ein beliebiges Ergebnis der Unterabfrage erfüllt ist.
EXISTS	Gibt den Wert „wahr“ zurück, wenn die Unterabfrage einen oder mehrere Datensätze liefert.

Terra Datenbank Ergebnis:

62 Ortsnamen gibt es auch als Ländername

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 (4) Rechtevergabe

Benutzer anlegen	
Syntax	CREATE USER Benutzername IDENTIFIED BY <i>Passwort</i> ;
Erläuterung	Benutzername = Zugangsname des Benutzers Passwort = Zugangscode des Benutzers
Beispiel 1	CREATE USER Schulz IDENTIFIED BY '15WHx%5a';
Beispiel 2	CREATE USER S 'fmueller'@'localhost' IDENTIFIED BY 'geheim';
Benutzer fmueller sind nur Zugriffe vom lokalen Computer aus erlaubt.	

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 (4) Rechtevergabe

Benutzer löschen

Syntax	DROP USER Benutzername;
Erläuterung	Benutzername = Name des Benutzers, der gelöscht werden soll
Beispiel	DROP USER Schulz;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 (4) Rechtevergabe

Rechte erteilen	
Syntax	GRANT Rechteliste ON Objektname TO Benutzername [WITH GRANT OPTION];
Erläuterung	Rechteliste = Aufzählung aller Rechte, die der Benutzer erhalten soll (mit Komma getrennt) Objektname = Name des Datenbankobjektes Benutzername = Name des Benutzers WITH GRANT OPTION = Damit darf der Benutzer seine Rechte weitergeben.
Beispiel	GRANT INSERT, UPDATE ON Ortsverwaltung.Ort TO Schulz;

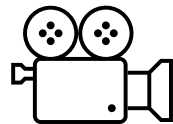
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 (4) Rechtevergabe

Rechte entziehen	
Syntax	REVOKE Rechteliste ON Objektname FROM Benutzername
Erläuterung	Rechteliste = Aufzählung aller Rechte, die dem Benutzer entzogen werden sollen Objektname = Name des Datenbankobjektes Benutzername = Name des Benutzers
Beispiel	REVOKE INSERT, UPDATE ON Ortsverwaltung.Ort FROM Schulz;

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Das Basiswissen über SQL erweitern und anwenden



animierte Präsentation



Lernziel
JOIN

**Verwaltung der
Daten mithilfe von
Datenbanken**

GFN 1

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.4 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Mit INNER JOIN werden nur die Datensätze angezeigt, bei denen eine Verknüpfung zwischen den beiden Tabellen besteht.
- b) Mit der Anweisung RIGHT JOIN werden alle Datensätze der linken und der rechten Tabelle angezeigt.
- c) Mit UNION werden die Ergebnismengen von zwei oder mehreren Abfragen miteinander verbunden.
- d) ANY, ALL und IN gehören zu den sogenannten Mengenoperatoren.
- e) Durch IN wird überprüft, ob ein bestimmter Wert in einer Menge von Werten einer Unterabfrage enthalten ist.

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.4 Kompetenzcheck ☒

- f) Unterabfragen dürfen keine weiteren Unterabfragen enthalten.
- g) REVOKE dient dem Erteilen von Rechten für Datenbankbenutzer.
- h) Damit ein Benutzer seine Rechte weitergeben kann, muss bei dem Erteilen der Rechte der Zusatz WITH GRANT OPTION angegeben werden.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.4 Kompetenzcheck ☒



Lösen Sie die Aufgaben 2 im Lehrbuch Seite 280

OPTIONAL:

8.7.4_Aufg_1-Lehrbuch Seite 280 Aufgabe 2.docx

Eine Übungsdatenbank „terra.zip“ steht zur Verfügung.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Sie sollen lernen, eine MySQL-Datenbank mit einem Java-Programm zu verbinden und SQL-Anweisungen aus dem Programm heraus auszuführen.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

MySQL ist ein relationales Datenbankmanagementsystem.

Wurde seit 1994 vom schwedischen Unternehmen MySQL AB entwickelt

2010 Übernahme durch Oracle

Es ist als Open-Source-Software sowie
als kommerzielle Enterprise Version
für verschiedene Betriebssysteme verfügbar.

Nach der Übernahme durch Oracle steht
das Datenbanksystem immer häufiger in der Kritik.

Alternativ wird auch MariaDB genutzt, eine von Oracle unabhängige Entwicklung durch
Ulf Michael Widenius, der Hauptautor der Originalversion des Open-Source-DBMS MySQL.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Voraussetzungen für die Verbindung einer Java-Anwendung mit einer MySQL-Datenbank

- MySQL-Datenbankserver
- JDBC-Treiber herunterladen
- Treiber in Klassenpfad einbinden
- Connection String mit
Hostname, Portnummer,
Datenbasename,
Username und Password
festlegen



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Testumgebung über die Console einrichten

- MySQL Datenbank ist gestartet, z. B. über XAMPP
- JDBC Driver for MySQL (**Connector/J**) herunterladen
Link: <https://www.mysql.com/products/connector/>
- Für dauerhafte Verfügbarkeit des Treibers
die Datei in Unterverzeichnis **lib** der Java Distribution ablegen oder ...
- Einen Testordner anlegen
die Datei [mysql-connector-java-8.0.29.jar](#) in den Testordner kopieren;
das folgende Testprogramm auch dort speichern
- Fortsetzung folgt ...

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Testumgebung über die Console einrichten (Fortsetzung)

```
import java.sql.*;
public class SimpleTest {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/TestDB","schueler","lf08");
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeQuery("show databases;");
            System.out.println("Connected");
        } catch (Exception e)
        { System.out.println(e); }
    }
}
```

Datei speichern im Testordner als **SimpleTest.java**

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

In der Console in den Testordner wechseln und folgende Befehle ausführen:

```
mysql -h localhost -u root -p
```

```
Enter password:
```

```
create database testdb;
```

```
create user "schueler" identified by "lf08";
```

```
use testdb;
```

```
grant all on testdb to "schueler";
```

```
exit
```

```
javac SimpleTest.java
```

```
java -cp ./mysql-connector-java-8.0.29.jar SimpleTest
```

Datenbank anlegen

Benutzerdaten anlegen

Zugriffsrechte festlegen

Compiler Aufruf

Programm mit Classpath Parameter ausführen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Klasse MySqlConnection

Notwendige Klassen importieren

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

Objektvariablen anlegen

```
public class MySqlConnection{  
    private Connection con = null;  
    private Statement sqlStatement = null;  
    private String dbHost = "localhost"; // Hostname  
    private String dbPort = "3306";      // Port -- Standard: 3306  
    private String dbName = "terra";     // Datenbankname  
    private String dbUser = "trainer";   // Datenbankuser  
    private String dbPass = "test2022";  // User-Passwort
```

Zugriffsparameter initialisieren,
die mit den Einstellungen der
Datenbank übereinstimmen müssen

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Datenbankverbindung
herstellen

Exceptions abfangen
und auswerten

```
public void create() {  
    try {  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        // Verbindung zur JDBC-Datenbank herstellen.  
        con = DriverManager.getConnection("jdbc:mysql://" + dbHost + ":" + dbPort + "/"  
                                         + dbName + "?" + "user=" + dbUser + "&" + "password=" + dbPass);  
        sqlStatement = con.createStatement();  
        System.out.println("Verbindung erfolgreich hergestellt.");  
    } catch (ClassNotFoundException e) {  
        System.out.println("Verbindung nicht möglich.");  
    } catch (SQLException e) {  
        System.out.println("Verbindung nicht möglich");  
        System.out.println("SQLException: " + e.getMessage());  
        System.out.println("SQLState: " + e.getSQLState());  
        System.out.println("VendorError: " + e.getErrorCode());  
    }  
}
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Datenbankverbindung schließen

Exceptions abfangen
und auswerten

```
public void close() {  
    try {  
        if (sqlStatement != null) { sqlStatement.close(); }  
        if (con != null) {  
            con.close();  
            System.out.println("Datenbankverbindung beendet");  
        }  
    } catch (SQLException e) {  
        System.out.println("SQLException: " + e.getMessage());  
        System.out.println("SQLState: " + e.getSQLState());  
        System.out.println("VendorError: " + e.getErrorCode());  
    }  
}
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Eine MySQL-Datenbank mit Java ansprechen

Es werden Daten aus der Datenbank mit SELECT-Anweisung abgefragt.

Die Ergebnismenge wird als ResultSet an das Programm zurückgegeben und kann dann weiterverarbeitet werden.

```
public ResultSet getData(String sql) {  
    try {  
        return sqlStatement.executeQuery(sql);  
    } catch (SQLException e) {  
        System.out.println("SQLException: " + e.getMessage());  
        System.out.println("SQLState: " + e.getSQLState());  
        System.out.println("VendorError: " + e.getErrorCode());  
        return null;  
    }  
}
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln

**SELECT
Statement**

**ResultSet
auswerten**

```
import java.sql.ResultSet;
public class Program {
    public static void main(String[] args) {
        MySqlConnection mySqlConnection = null;
        try {
            mySqlConnection = new MySqlConnection();
            mySqlConnection.create();
            String sqlStatement = "SELECT O.Name, O.Einwohner, L.Name " +
                                "FROM Ort O INNER JOIN Land L ON O.LNR = L.LNR LIMIT 10";
            System.out.println("Ortsname, Einwohner, Land");
            System.out.println("-----");
            ResultSet rs = mySqlConnection.getData(sqlStatement);
            while (rs.next()) {
                String ortsName = rs.getString("Name");
                int anzahlEinwohner = rs.getInt("Einwohner");
                String landName = rs.getString("Name");
                // Ausgabe der Werte
                System.out.format("%s,\t %s,\t %s\n", ortsName, anzahlEinwohner, landName);
            }
        } catch (Exception e) {e.printStackTrace();}
    } finally {if (mySqlConnection != null) { mySqlConnection.close(); }}}
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.5 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Eine MySQL-Verbindung kann durch die Methode "getConnection(...)" der Klasse "Driver-Manager" hergestellt werden.
- b) Bevor eine Datenbankverbindung erzeugt wird, muss ein entsprechender Treiber geladen werden.
- c) Ausnahmen, welche Datenbanken betreffen, werden als SQL-Exception zurückgegeben.
- d) UPDATE-SQL-Anweisungen werden mit der Methode "executeQuery(...)" ausgeführt.
- e) Mit der SELECT-Anweisung ermittelte Daten werden in einem ResultSet-Objekt zurückgegeben.
- f) Mit der Methode "close()" wird die Datenbank geschlossen.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.5 Kompetenzcheck ☒

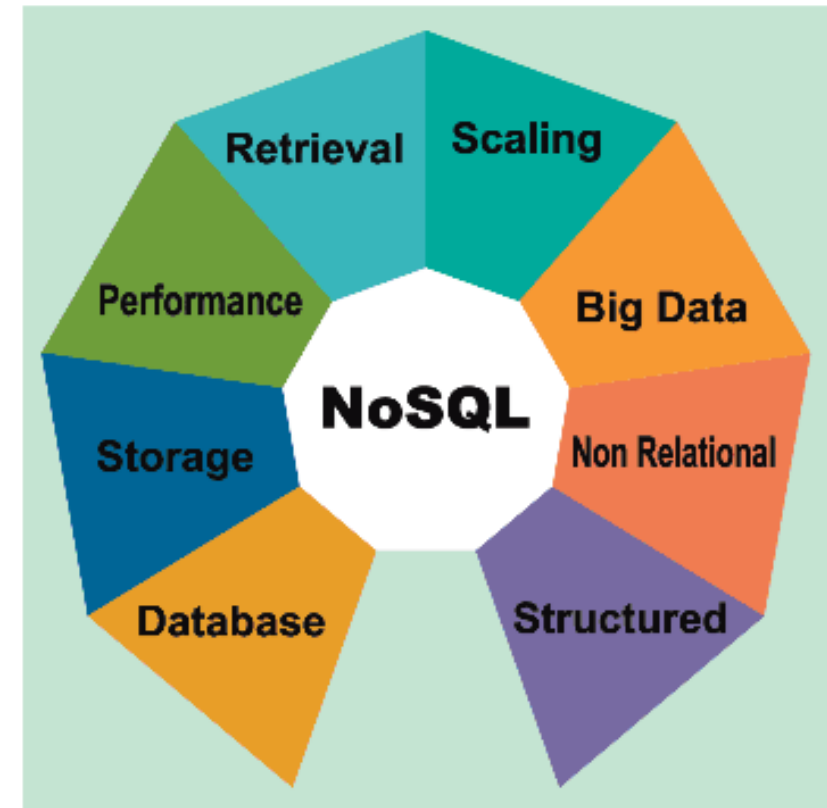


- g) Mit dem Schlüsselwort "new" werden Objekte angelegt.
- h) Der finally-Zweig wird nur ausgeführt, wenn eine Exception aufgetreten ist.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

Sie sollen NoSQL-Datenbanken und deren zugrunde liegende Datenmodelle unterscheiden.

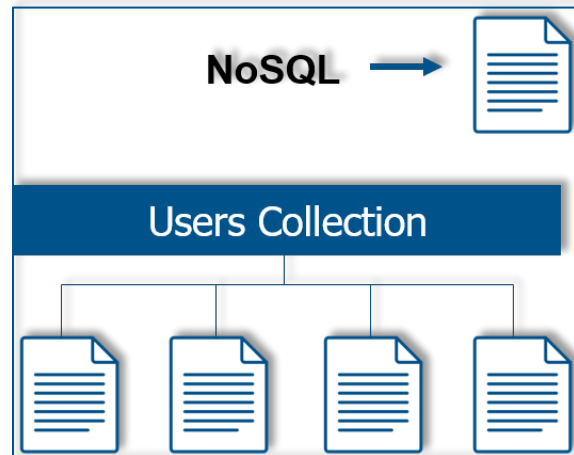


8.7 Datenbanklösungen bedarfsgerecht entwickeln

NoSQL

- Nimmt vom relationalen Modell Abstand
- Kein starr definiertes Schema
- Unstrukturierte und halb strukturierte Daten können gespeichert und bearbeitet werden
- Varianten:

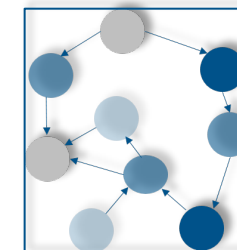
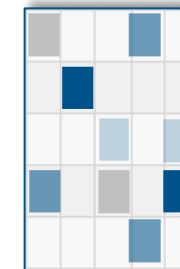
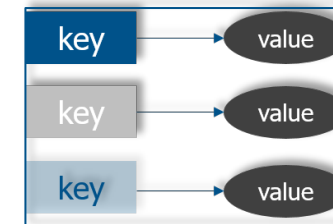
- **Document**
(BaseX, MongoDB)



NoSQL

Varianten:

- **Key-Value**
(Amazon Dynamo, Google BigTable)
- **Column**
(Apache, Cassandra, Sybase IQ)
- **Graph**
(Neo4j, HyperGraphDB)



8.7 Datenbanklösungen bedarfsgerecht entwickeln

Vorteile NoSQL

- Im Big-Data-Bereich stoßen relationale Datenbanken an ihre Grenzen.
- Ein Verbund von Datenbanken kann gebildet werden.
- Schemalos und tabellenfrei
- Bietet ein hohes Maß an Flexibilität bei Datenmodellen
- Lässt sich einfach und kostengünstig skalieren, da NoSQL auf verteilten Systemen arbeitet.
- Der Open-Source Gedanke:
 - Andere Entwicklungen sind frei einsehbar
 - Die Dokumentationen sind öffentlich verfügbar

Nachteile NoSQL

- Manche Entwicklungen und Dokumentationen sind nicht vollständig.
- Wenn Konsistenz ein Muss ist und sich das Datenvolumen nicht wesentlich ändert, ist die Verwendung von NoSQL-Datenbanken ungeeignet.
- Es können keine ACID-Eigenschaften gewährleistet werden.
- Wenig Community-Unterstützung.
- Keine Standardisierung; führt zu Problemen bei einer Migration.

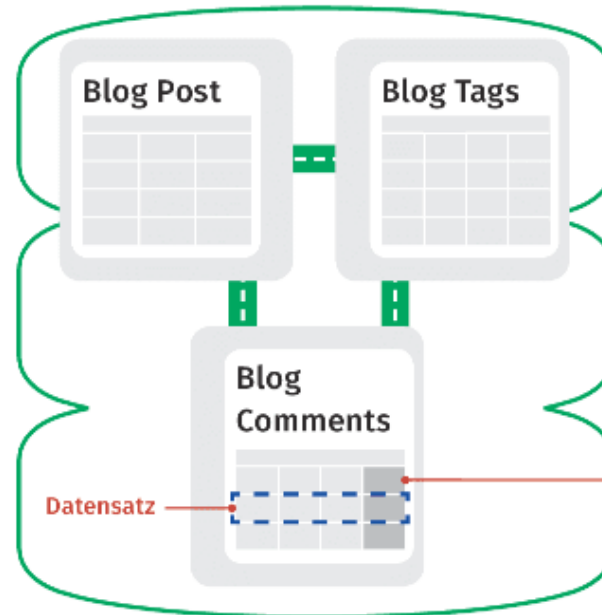
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(1) Dokumentorientierte Datenbank

- Daten werden in Form von Dokumenten abgespeichert.
- Jedes Dokument kann hat einen eindeutigen Identifier
- Daten sind unstrukturiert.
- Es können verschieden Attribute zugeordnet werden, um sie durchsuchen zu können.

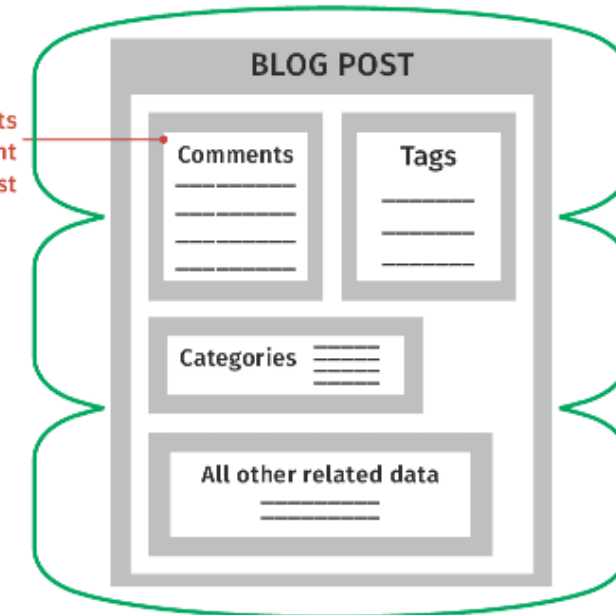
Relationale Datenbank



Alle Daten eines Blog Posts werden in einem Dokument zusammengefasst

Alle Daten eines Blog Posts werden unter verschiedenen Tabellen aufgeteilt

Dokumentorientierte Datenbank

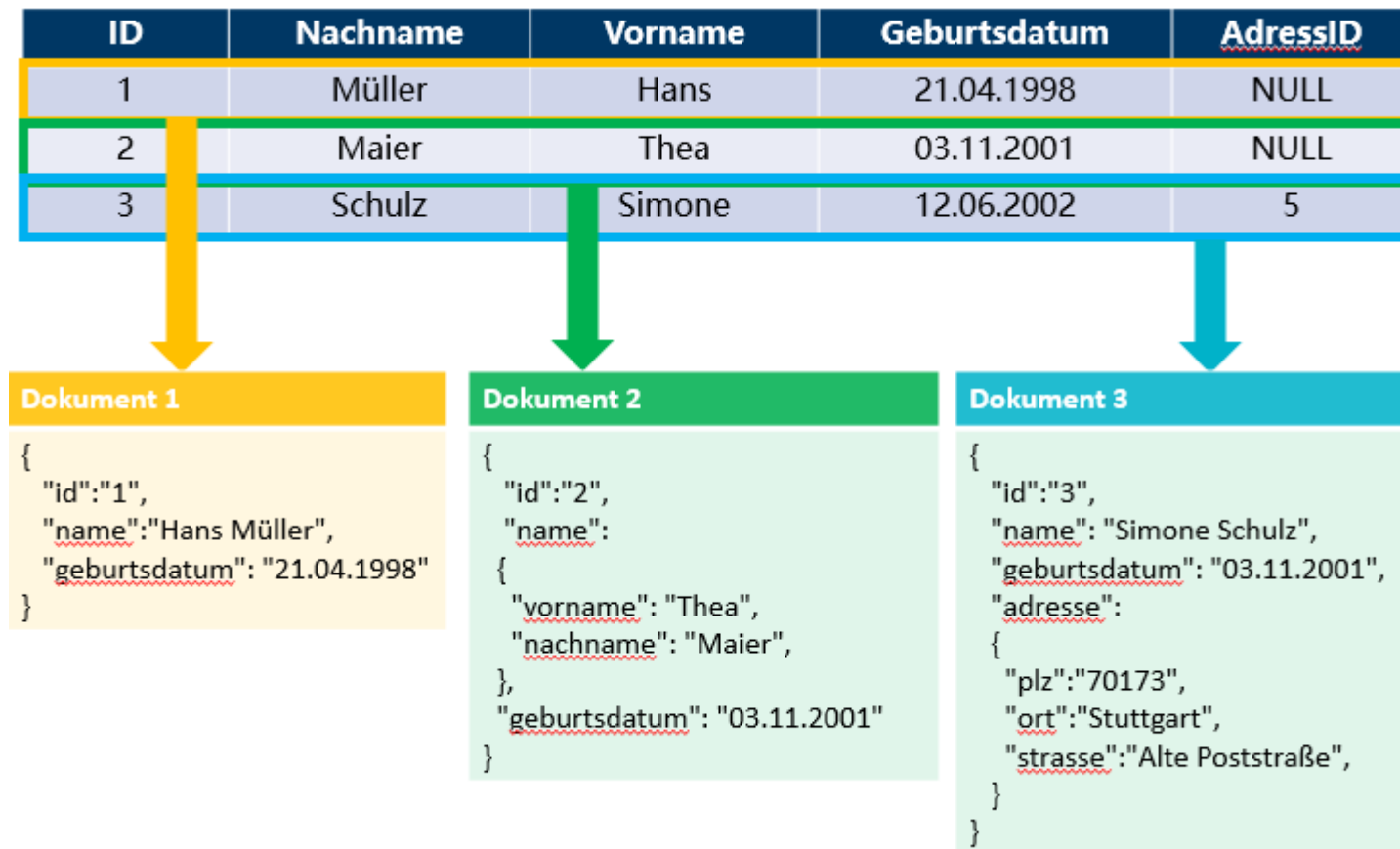


8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(1) Dokumentorientierte Datenbank

- Daten können keinem gemeinsamen Typ zugeordnet werden.
- Gespeicherte Informationen unterscheiden sich in vielen Datensätzen.
- Eignet sich besonders gut für CMS-Systeme (Content-Management-Systeme).
- Datenformat: JSON
- (Terrastore, Amazon Document DB, Google Cloud Firestore, Apache CouchDB)

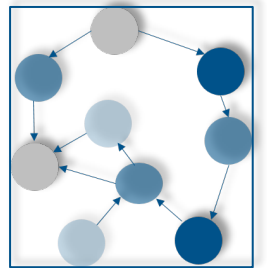
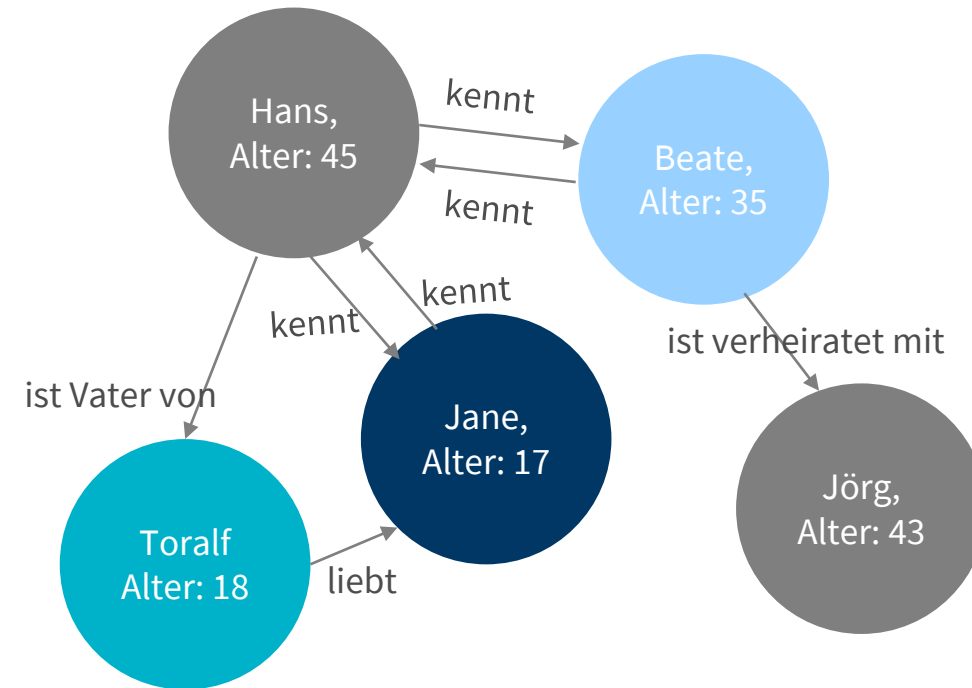


8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(2) Graphdatenbank

- Speichern der Daten mithilfe von Knoten, Kanten und Eigenschaften
- Eine Beziehung stellt eine Verbindung zwischen zwei Knoten dar. Das wie wird beschrieben.
- Beziehungen besitzen einen Start- und einen Endknoten, einen Typ und eine Richtung
- Knoten und Kanten können Attribute besitzen
- Wird bspw. in Social-Media Bereichen eingesetzt
- (Neptune, OrientDB, Neo4j)



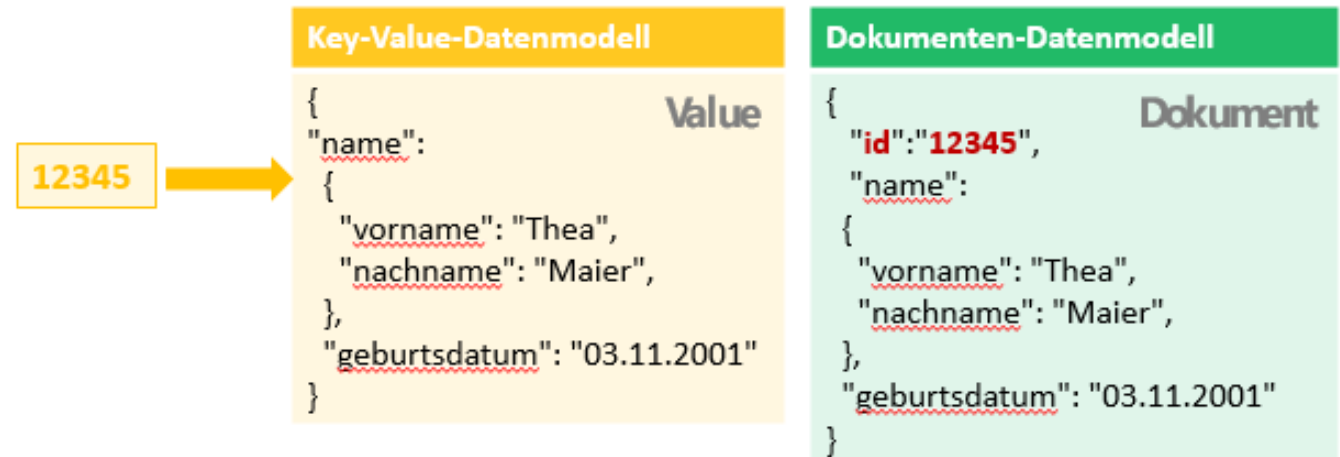
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(3) Key-Value-Datenbank

- Daten werden in Form von Schlüssel-Werte-Paaren in einer Hash-Tabelle gespeichert
- Schlüssel ist eindeutig
- Schlüssel nicht im Datensatz vorhanden, dient als "Referenz"
- Schneller, einfacher Zugriff möglich
- Hohe Skalierbarkeit
- Last verteilbar auf mehreren Servern möglich
- (Amazon Dynamo, Google BigTable)

Key	Value
12345	12.45, 345.66, 445.99
12246	{vorname: "Uwe", nachname: "Meier"}
33444	Zitat: "Das hat folgende Bedeutung..."
50078	"Anzahl Mitglieder : 51"



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(3) Key-Value-Datenbank

Nachteile:

- Abfragen sehr eingeschränkt, weil Zugriff nur über den Schlüssel möglich.
- Verzicht auf Indizes und komplexe Suchalgorithmen, um schnellere Schreib- und Lesegeschwindigkeit zu gewährleisten.
- Umfangreiche Abfragen benötigen eine API.
- Bei vielen Beziehungen wird es unübersichtlich und komplex.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.6 NoSQL-Datenbanken und deren Datenmodelle unterscheiden

(4) Spaltenorientierte Datenbank

- Daten werden in Spalten gespeichert
- Sehr schneller Zugriff, wenn Daten bestimmter Spalten ausgelesen werden sollen
- Aufwendig beim Ergänzen oder Lesen von Zeilen
- Einsatz für Data-Warehouse oder Data-Mining und Analyseprogrammen (OLAP)
- (Apache, Cassandra oder Sybase IQ)

ID	Nachname	Vorname	Geburtsdatum	Gehalt
1	Müller	Hans	21.04.1998	2700
2	Maier	Thea	03.11.2001	2220
3	Schulz	Simone	12.06.2002	1970

Zeilenorientierte Speicherung

1, Müller, Hans, 21.04.1998, 2700; 2, Maier, Thea, 03.11.2001, 2220; 3, Schulz, Simone, 12.06.2002, 1970 ;

Spaltenorientierte Speicherung

1, 2, 3; Müller, Maier, Schulz; Hans, Thea, Simone; 21.04.1998, 03.11.2001, 12.06.2002; 2700, 2220, 1970 ;

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.6 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Für Dokumente von dokumentenorientierten Datenbanken wird oft das Datenformat von JSON verwendet.
- b) Die Struktur der Dokumente bei dokumentenorientierten Datenbanken muss immer gleich sein.
- c) Graphdatenbanken werden im Social-Media-Bereich eingesetzt.
- d) Konten und Kanten sind Modellierungsbestandteile von Graphdatenbanken.
- e) Key-Value-Datenbanken eignen sich hervorragend für die Speicherung von Daten, welche eine komplexe Beziehungsstruktur aufweisen.

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.6 Kompetenzcheck ☒

- f) Durch den Schlüssel kann bei Key-Value-Datenbanken sehr schnell auf die Daten zugegriffen werden.
- g) Key-Value-Datenbanken lassen sich sehr gut vertikal skalieren.
- h) Komplexe Datenabfragen bei Key-Value-Datenbanken lassen sich einfach durch Formulierung entsprechender SQL-Anweisungen durchführen.
- i) Bei spaltenorientierten Datenbanken werden die Daten intern trotzdem zeilenweise gespeichert.
- j) Bei der Analyse von Daten und in dem Bereich des Data-Minings kommen u. a. spaltenorientierte Datenbanken zum Einsatz.
- k) MySQL zählt zu den spaltenorientierten Datenbanken.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.7 Die NoSQL-Datenbank "Mongo-DB" mit Python ansprechen

Sie sollen lernen, mithilfe eines Python-Programms Daten in MongoDB zu lesen und zu schreiben.



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.7 Die NoSQL-Datenbank "Mongo-DB" mit Python ansprechen

MongoDB

- Universelle, dokumentenbasierte, meist genutzte, weit verbreitete NoSQL Datenbank
- Format der Dokumente JSON ähnlich
- Für moderne Anwendungsentwicklung und die Cloud



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.7 Die NoSQL-Datenbank "Mongo-DB" mit Python ansprechen

Vorgehen beim Speichern:

- MongoClient importieren
- Datenbank erzeugen
- Collection erzeugen
- Dokument erzeugen
- Erzeugtes Dokument in die DB speichern

```
1 from pymongo import MongoClient
2
3 database = MongoClient('mongodb://localhost:27017') ['artikelverwaltung_db']
4 collection = database['artikel']
5
6 schrank_daten = {
7     'artikel_nr': '002348',
8     'name': 'Holzschrank HS900',
9     'hersteller': 'Möbel GmbH',
10    'preis': '588.00 Euro'
11 }
12 collection.insert_one(schrank_daten)
```

```
{
  "name": "Georg",
  "alter": 47,
  "verheiratet": false,
  "beruf": null,
  "kinder": [
    {
      "name": "Lukas",
      "alter": 19,
      "schulabschluss": "Gymnasium"
    },
    {
      "name": "Lisa",
      "alter": 14,
      "schulabschluss": null
    }
  ]
}
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.7 Die NoSQL-Datenbank "Mongo-DB" mit Python ansprechen

Vorgehen beim Lesen:

○ MongoClient importieren

○ Datenbank erzeugen

○ Collection erzeugen

○ Einzelnes Dokument lesen

○ Ausgeben der Daten

○ Mehrere Dokumente lesen

Resultset erzeugen,
durch die Ergebnisliste iterieren und ausgeben

```
1 from pymongo import MongoClient
2
3 database = MongoClient('mongodb://localhost:27017')['artikelverwaltung_db']
4 collection = database['artikel']
5
6 daten = collection.find_one({"artikel_nr": "1234"})
7 print("Daten des Artikels mit der Nr. 1234")
8 print (daten)
9
10 print("Alle Daten des Herstellers -Möbel GmbH- ")
11 for daten in collection.find({"hersteller": "Möbel GmbH"}):
12     print (daten)
```

8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.7 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) MongoDB ist eine spaltenorientierte Datenbank,
- b) Um MongoDB in Python ansprechen zu können, muss der MongoClient importiert werden.
- c) Das Paket "PyMongo" ist Bestandteil der Standardbibliothek von Python.
- d) Eine Collection ist in MongoDB eine Gruppe von Dokumenten.
- e) Jede Collection kann nur ein Dokument enthalten.
- f) Mit der Funktion "write_one(...)" einer Collection kann ein Dokument in die Datenbank geschrieben werden.
- g) Mit der Funktion "find(...)" können mehrere Dokumente angefragt werden.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.7 Kompetenzcheck ☒

Lösen Sie die Aufgaben im Lehrbuch Seite 288

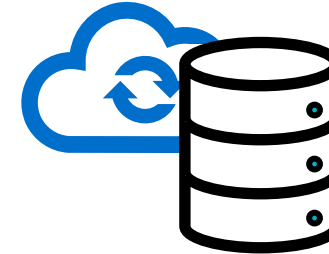
Aufgabe 2



8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.8 Cloud-basierte Datenbanklösungen unterscheiden

Sie sollen sich grundlegendes Wissen über cloud-basierte Datenbank aneignen und diese voneinander unterscheiden können.



Definition

Aus struktureller und gestalterischer Sicht unterscheidet sich eine Cloud-Datenbank nicht von einer Datenbank, die auf den eigenen Servern eines Unternehmens betrieben wird. Der entscheidende Unterschied liegt darin, wo die Datenbank gespeichert ist.

Im Gegensatz zu einer On-Premises-Datenbank, auf die die Nutzer über das lokale Netzwerk (LAN) eines Unternehmens zugreifen, befindet sich eine Cloud-Datenbank auf Servern und Storage, die von einem Cloud- oder Database-as-a-Service-Anbieter (**DBaaS**) bereitgestellt werden. Der Zugriff erfolgt ausschließlich über das Internet.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

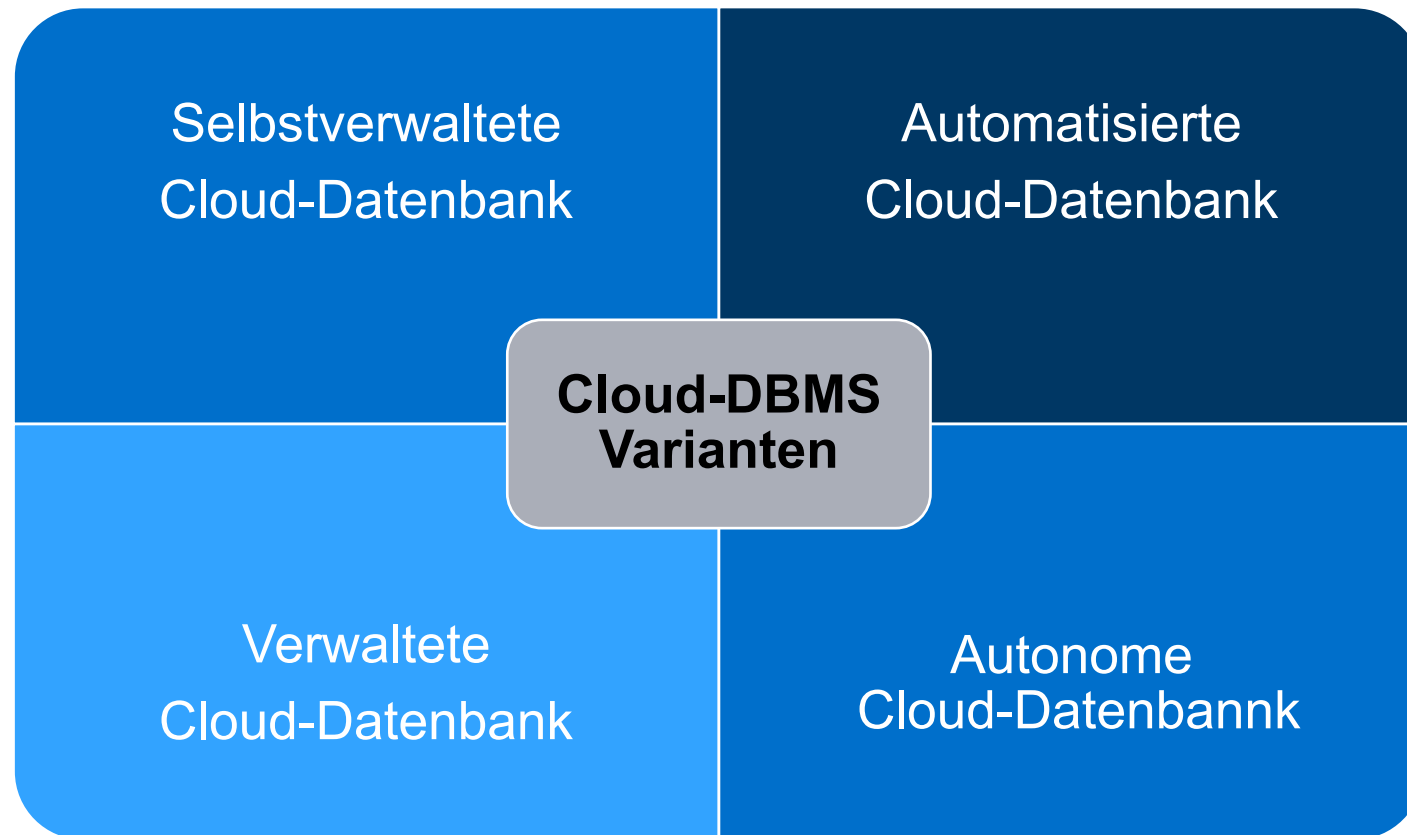
8.7.8 Cloud-basierte Datenbanklösungen unterscheiden

Vorteile von cloud-basierten Datenbanklösungen

Flexibilität	Schnelle und kurzfristige In- und Außerbetriebsetzung
Schnelle Markteinführung	Schnelle Reaktion auf geänderte oder neue Geschäftsbedingungen, da keine Hardware besorgt werden muss.
Geringe Risiken	Durch Nutzung der DBaaS-Modelle können Ausfallzeiten reduziert oder ganz vermieden werden. Beliebige auch kurzfristige Skalierbarkeit verringert den Planungsaufwand.
Niedrige Kosten	Payper-Use-Abonnementmodelle und dynamische Skalierung ermöglichen es, z. B. in Spitzenlastzeiten die Datenbankkapazitäten hoch zu skalieren und danach wieder auf ein normales Maß herunter zu skalieren. Dieses ist weitaus kostengünstiger als eigene Server für die Bewältigung von Lastspitzen vorzuhalten.

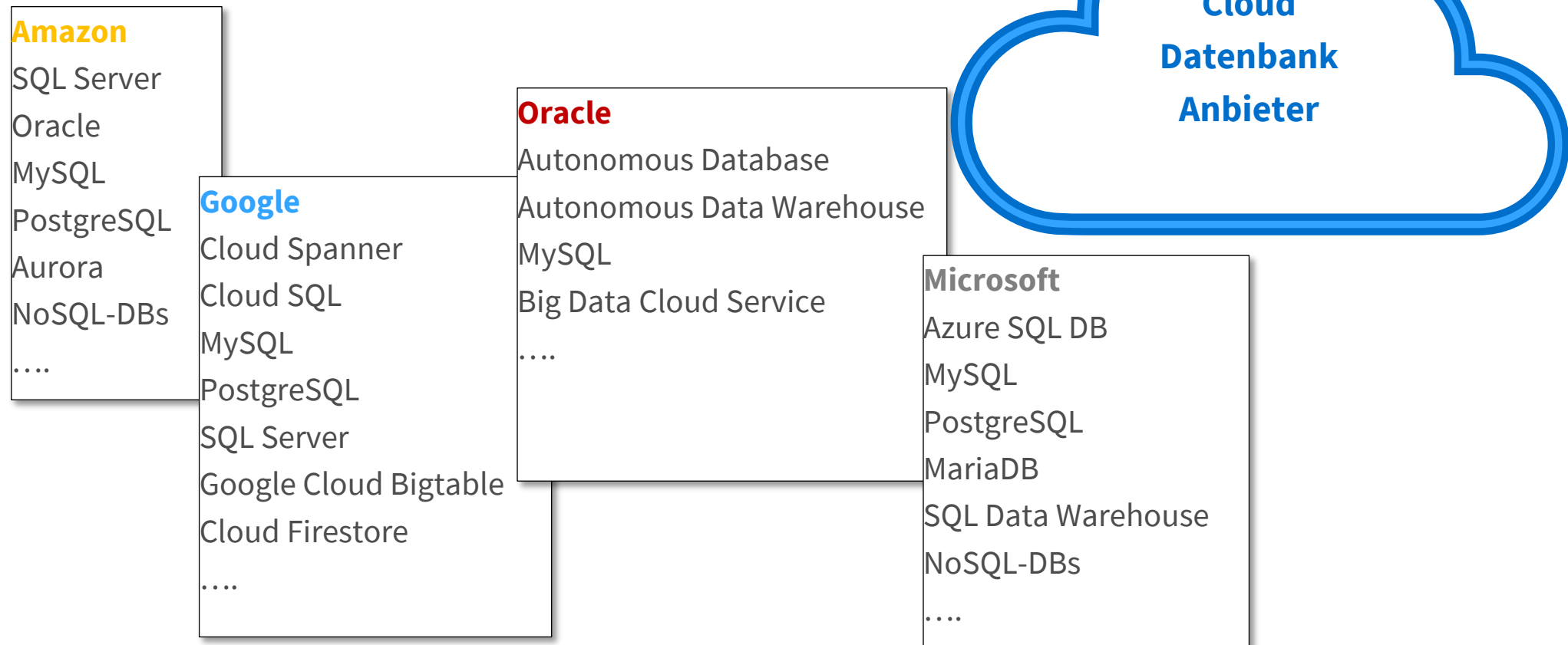
8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.8 Cloud-basierte Datenbanklösungen unterscheiden

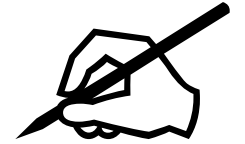


8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.8 Cloud-basierte Datenbanklösungen unterscheiden



8.7 Datenbanklösungen bedarfsgerecht entwickeln



8.7.8 Kompetenzcheck ☒

Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- a) Cloud-basierte Datenbanklösungen helfen den Unternehmen, Kosten zu sparen.
- b) Cloud-basierte Datenbanklösungen können nur schwer an sich ändernde Rahmenbedingungen angepasst werden.
- c) Bei selbstverwalteten Cloud-Datenbanken kümmert sich ein Datenbankadministrator des Unternehmens um alle Belange, welche mit dem Datenbankmanagement zusammenhängen.
- d) Bei autonomen Cloud-Datenbanken verwaltet das Unternehmen, welches das DBaaS-Produkt erworben hat, seine Datenbank selbst.

8.7 Datenbanklösungen bedarfsgerecht entwickeln

8.7.8 Kompetenzcheck ☒



Beantworten Sie die Kompetenzcheckfragen im Buch zu dem gerade durchgearbeiteten Kapitel.

Welche Aussage ist richtig?

- e) Bei automatisierten Datenbanken ist kein separater Datenbankadministrator notwendig.
- f) Amazon stellt eigene Werkzeuge zur Datenmigration in die Cloud-Datenbank zur Verfügung
- g) ORACLE hat von allen Cloud-Datenbank-Anbietern die größte Palette an DBaaS-Produkten.



Vielen Dank.

GFN GmbH
Tel: 0800 436 436 436
zentrale-kundenberatung@gfn.de