



Datenbanken

34. Logisches Schema: Relationales Datenmodell

Thomas Weise (汤卫思)
twaise@hfuu.edu.cn

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Databases



Dies ist ein Kurs über Datenbanken an der Universität Hefei (合肥大学).

Die Webseite mit dem Lehrmaterial dieses Kurses ist <https://thomasweise.github.io/databases> (siehe auch den QR-Kode unten rechts). Dort können Sie das Kursbuch (in Englisch) und diese Slides finden. Das Repository mit den Beispielen finden Sie unter <https://github.com/thomasWeise/databasesCode>.



Outline

1. Einleitung
2. Definitionen
3. Schlüssel
4. Relationale Datenbankmanagementsysteme
5. Zusammenfassung





Einleitung



Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.



Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten.



Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke).

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen).

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).



Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).
- In der kompakten Notation verschwanden dann die Beziehungstypen als Elemente und sie wurden zu einfachen geraden Linien.

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).
- In der kompakten Notation verschwanden dann die Beziehungstypen als Elemente und sie wurden zu einfachen geraden Linien.
- Das hat zwei Gründe.

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).
- In der kompakten Notation verschwanden dann die Beziehungstypen als Elemente und sie wurden zu einfachen geraden Linien.
- Das hat zwei Gründe: Rauten verschwenden Platz.

Einleitung



- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).
- In der kompakten Notation verschwanden dann die Beziehungstypen als Elemente und sie wurden zu einfachen geraden Linien.
- Das hat zwei Gründe: Rauten verschwenden Platz und Beziehungen existieren nicht als einzelne Objekte im relationalen Modell.

- Die nächste Phase des Datenbankdesigns ist die Transformation eines Entity-Relationship-Modells in ein logisches Schema passend zum selektierten Datenmodell⁶³.
- Wir haben dafür das relationale Datenmodell ausgewählt.
- In den ERDs die wir gemalt haben *bevor* wir die kompakte Notation kennengelernt haben, hatten wir drei visuelle Komponenten: Entitätstypen (Rechtecke), Attribute (Ellipsen) und Beziehungstypen (Rauten).
- In der kompakten Notation verschwanden dann die Beziehungstypen als Elemente und sie wurden zu einfachen geraden Linien.
- Das hat zwei Gründe: Rauten verschwenden Platz und Beziehungen existieren nicht als einzelne Objekte im relationalen Modell.
- Beziehungen werden stattdessen als Fremdschlüssel (also spezielle Attribute) und Einschränkungen realisiert.

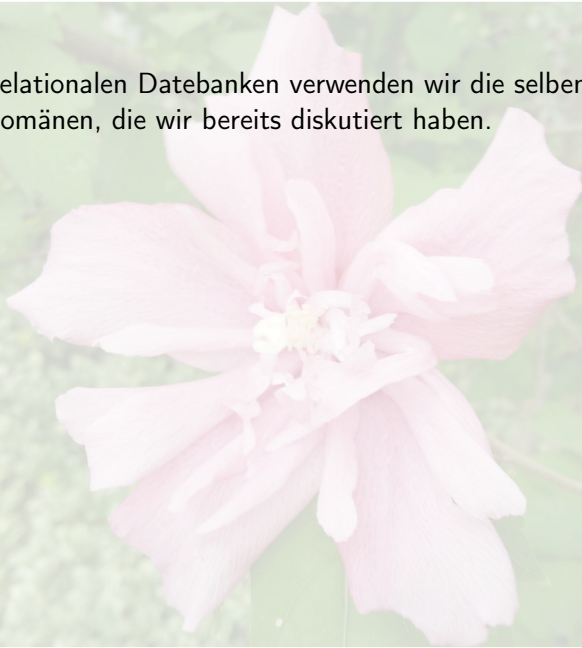


Definitionen



Definitionen

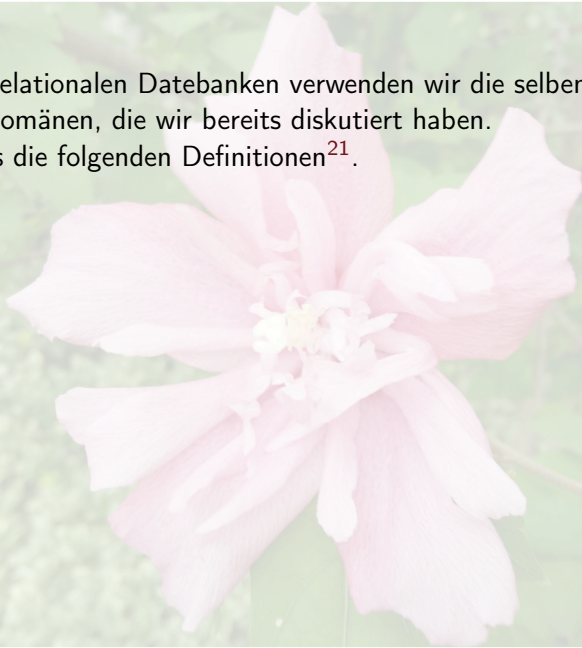
- Im Kontext von relationalen Datebanken verwenden wir die selben Definitionen von Attributen und Domänen, die wir bereits diskutiert haben.



Definitionen



- Im Kontext von relationalen Datebanken verwenden wir die selben Definitionen von Attributen und Domänen, die wir bereits diskutiert haben.
- Zusätzlich gibt es die folgenden Definitionen²¹.



Definitionen



- Im Kontext von relationalen Datebanken verwenden wir die selben Definitionen von Attributen und Domänen, die wir bereits diskutiert haben.
- Zusätzlich gibt es die folgenden Definitionen²¹:

Definition: Relationsschema

Ein *Relationsschema* (EN: *relation schema*) Σ ist eine geordnete Sequenz von n Attributen $(a_1, a_2, \dots, a_n)$, also eine Sequenz von Attributnamen und Domänen.

Definitionen



- Im Kontext von relationalen Datebanken verwenden wir die selben Definitionen von Attributen und Domänen, die wir bereits diskutiert haben.
- Zusätzlich gibt es die folgenden Definitionen²¹:

Definition: Relationsschema

Ein *Relationsschema* (EN: *relation schema*) Σ ist eine geordnete Sequenz von n Attributen $(a_1, a_2, \dots, a_n)$, also eine Sequenz von Attributnamen und Domänen.

Definition: Relation

Eine *Relation* R ist eine Menge von n -Tupeln $R \subseteq \text{dom}(a_1) \times \text{dom}(a_2) \times \dots \times \text{dom}(a_n)$ mit der ein Relationsschema $\Sigma(R)$ das die Attribute $(a_1, a_2, \dots, a_n)$ spezifiziert assoziiert ist⁶².

Definitionen



- Zusätzlich gibt es die folgenden Definitionen²¹:

Definition: Relationsschema

Ein *Relationsschema* (EN: *relation schema*) Σ ist eine geordnete Sequenz von n Attributen $(a_1, a_2, \dots, a_n)$, also eine Sequenz von Attributnamen und Domänen.

Definition: Relation

Eine *Relation* R ist eine Menge von n -Tupeln $R \subseteq \text{dom}(a_1) \times \text{dom}(a_2) \times \dots \times \text{dom}(a_n)$ mit der ein Relationsschema $\Sigma(R)$ das die Attribute $(a_1, a_2, \dots, a_n)$ spezifiziert assoziiert ist⁶².

- Die Definition eines Relationsschemas im relationalen Modell ist daher in etwa Äquivalent zur Definition eines Entitätstyps im konzeptuellen Modell.



Definition: Relationsschema

Ein *Relationsschema* (EN: *relation schema*) Σ ist eine geordnete Sequenz von n Attributen $(a_1, a_2, \dots, a_n)$, also eine Sequenz von Attributnamen und Domänen.

Definition: Relation

Eine *Relation* R ist eine Menge von n -Tupeln $R \subseteq \text{dom}(a_1) \times \text{dom}(a_2) \times \dots \times \text{dom}(a_n)$ mit der ein Relationsschema $\Sigma(R)$ das die Attribute $(a_1, a_2, \dots, a_n)$ spezifiziert assoziiert ist⁶².

- Die Definition eines Relationsschemas im relationalen Modell ist daher in etwa Äquivalent zur Definition eines Entitätstyps im konzeptuellen Modell.
- Wenn wir unser konzeptuelles Modell in ein logisches, relationales Modell übersetzen, dann wird ein Entitätstyp zu einem Relationsschema.



Definition: Relationsschema

Ein *Relationsschema* (EN: *relation schema*) Σ ist eine geordnete Sequenz von n Attributen $(a_1, a_2, \dots, a_n)$, also eine Sequenz von Attributnamen und Domänen.

Definition: Relation

Eine *Relation* R ist eine Menge von n -Tupeln $R \subseteq \text{dom}(a_1) \times \text{dom}(a_2) \times \dots \times \text{dom}(a_n)$ mit der ein Relationsschema $\Sigma(R)$ das die Attribute $(a_1, a_2, \dots, a_n)$ spezifiziert assoziiert ist⁶².

- Wenn wir unser konzeptuelles Modell in ein logisches, relationales Modell übersetzen, dann wird ein Entitätstyp zu einem Relationsschema.
- Der Unterschied zum konzeptuellen Modell ist, dass wir Relationen sowohl für Entitäten als auch Beziehungen benutzen.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.
- Wir könnten denken, das Relationen als Tabellen implementiert werden.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.
- Wir könnten denken, das Relationen als Tabellen implementiert werden.
- Aber das stimmt nur zum Teil.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.
- Wir könnten denken, das Relationen als Tabellen implementiert werden.
- Aber das stimmt nur zum Teil:
- Relationen können auch das Ergebnis von einem **SELECT**-Statement in SQL sein.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.
- Wir könnten denken, das Relationen als Tabellen implementiert werden.
- Aber das stimmt nur zum Teil:
- Relationen können auch das Ergebnis von einem **SELECT**-Statement in SQL sein.
- Relationen können auch die Parameter eines **INSERT INTO**-Statements sein.

Relationen: Mehr als Tabellen



- Auf den ersten Blick könnte man nun denken „Relationen = Tabellen“ in einer Datenbank.
- Wir könnten denken, das Relationen als Tabellen implementiert werden.
- Aber das stimmt nur zum Teil:
- Relationen können auch das Ergebnis von einem **SELECT**-Statement in SQL sein.
- Relationen können auch die Parameter eines **INSERT INTO**-Statements sein.
- Relationen sind also ein sehr allgemeines Konzept für Daten.

Relationen: Mengen



- Beachten Sie, das eine Relation eine *Menge* von Tupeln ist.

Relationen: Mengen



- Beachten Sie, dass eine Relation eine *Menge* von Tupeln ist.
- Weil eine Menge das selbe Element nicht mehrfach enthalten kann, sind Duplikate, also gleiche Tupel (Zeilen, Datensätze), nicht zulässig¹⁵.

Relationen: Mengen



- Beachten Sie, dass eine Relation eine *Menge* von Tupeln ist.
- Weil eine Menge das selbe Element nicht mehrfach enthalten kann, sind Duplikate, also gleiche Tupel (Zeilen, Datensätze), nicht zulässig¹⁵.
- Abweichend davon erlaubt die Sprache SQL doch Duplikate in Tabellen und Abfrageergebnissen¹⁵.

Relationen: Mengen



- Beachten Sie, dass eine Relation eine *Menge* von Tupeln ist.
- Weil eine Menge das selbe Element nicht mehrfach enthalten kann, sind Duplikate, also gleiche Tupel (Zeilen, Datensätze), nicht zulässig¹⁵.
- Abweichend davon erlaubt die Sprache SQL doch Duplikate in Tabellen und Abfrageergebnissen¹⁵.
- Mengen sind auch ungeordnet, also gibt es keine Standardordnung für die Tupel in einer Relation.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.
- In den ersten Arbeiten über relationale Datenbanken²¹ war die Reihenfolge von Spalten in einer Relation noch wichtig und es war erlaubt, dass mehrere Spalten die selben Namen haben durften.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.
- In den ersten Arbeiten über relationale Datenbanken²¹ war die Reihenfolge von Spalten in einer Relation noch wichtig und es war erlaubt, dass mehrere Spalten die selben Namen haben durften.
- Das ist heute **nicht mehr so**.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.
- In den ersten Arbeiten über relationale Datenbanken²¹ war die Reihenfolge von Spalten in einer Relation noch wichtig und es war erlaubt, dass mehrere Spalten die selben Namen haben durften.
- Das ist heute **nicht mehr so**.
- Heute ist die Reihenfolge der Spalten egal und jede Spalte einer Tabelle hat einen eindeutigen/einmaligen Namen⁶⁷.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.
- In den ersten Arbeiten über relationale Datenbanken²¹ war die Reihenfolge von Spalten in einer Relation noch wichtig und es war erlaubt, dass mehrere Spalten die selben Namen haben durften.
- Das ist heute **nicht mehr so**.
- Heute ist die Reihenfolge der Spalten egal und jede Spalte einer Tabelle hat einen eindeutigen/einmaligen Namen⁶⁷.
- Die Werte von Attributen sind atomar, also unteilbar.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Alle Attribute (Spalten) müssen Namen haben, es gibt also keine anonymen Spalten⁶⁷.
- In den ersten Arbeiten über relationale Datenbanken²¹ war die Reihenfolge von Spalten in einer Relation noch wichtig und es war erlaubt, dass mehrere Spalten die selben Namen haben durften.
- Das ist heute **nicht mehr so**.
- Heute ist die Reihenfolge der Spalten egal und jede Spalte einer Tabelle hat einen eindeutigen/einmaligen Namen⁶⁷.
- Die Werte von Attributen sind atomar, also unteilbar.
- Es gibt keine mehrwertigen oder zusammengesetzten Attribute^{63,67}.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Der Grad einer Relation wird wie folgt definiert — verwechseln Sie das nicht mit dem Grad einer Beziehung!

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Der Grad einer Relation wird wie folgt definiert — verwechseln Sie das nicht mit dem Grad einer Beziehung!

Definition: Grad einer Relation

Der Grad (EN: *degree*) einer Relation ist die Anzahl n ihrer Attribute.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Der Grad einer Relation wird wie folgt definiert — verwechseln Sie das nicht mit dem Grad einer Beziehung!

Definition: Grad einer Relation

Der Grad (EN: *degree*) einer Relation ist die Anzahl n ihrer Attribute.

- Relationen sind der Kern einer relationalen Datenbank.

Relationen: Spalten mit Namen, Ungeordnet, Attribute



- Der Grad einer Relation wird wie folgt definiert — verwechseln Sie das nicht mit dem Grad einer Beziehung!

Definition: Grad einer Relation

Der Grad (EN: *degree*) einer Relation ist die Anzahl n ihrer Attribute.

- Relationen sind der Kern einer relationalen Datenbank.

The totality of data in a data bank may be viewed as a collection of time-varying relations. These relations are of assorted degrees. As time progresses, each n -ary relation may be subject to insertion of additional n -tuples, deletion of existing ones, and alteration of components of any of its existing n -tuples.

— Edgar Frank „Ted“ Codd [21], 1970



Schlüssel



Schlüssel

- Wir haben Schlüssel bereits beim konzeptuellen Modellieren besprochen.



Schlüssel



- Wir haben Schlüssel bereits beim konzeptuellen Modellieren besprochen.
- Jetzt haben wir gesagt, dass Relationen Mengen von einmaligen Datensätzen sind.

Schlüssel



- Wir haben Schlüssel bereits beim konzeptuellen Modellieren besprochen.
- Jetzt haben wir gesagt, dass Relationen Mengen von einmaligen Datensätzen sind.
- Schlüssel machen diese Datensätze eindeutig bzw. einmalig.

Schlüssel



- Wir haben Schlüssel bereits beim konzeptuellen Modellieren besprochen.
- Jetzt haben wir gesagt, dass Relationen Mengen von einmaligen Datensätzen sind.
- Schlüssel machen diese Datensätze eindeutig bzw. einmalig.
- Schlüssel spielen daher eine wichtige Rolle im relationalen Datenbankdesign.

Schlüssel



- Wir haben Schlüssel bereits beim konzeptuellen Modellieren besprochen.
- Jetzt haben wir gesagt, dass Relationen Mengen von einmaligen Datensätzen sind.
- Schlüssel machen diese Datensätze eindeutig bzw. einmalig.
- Schlüssel spielen daher eine wichtige Rolle im relationalen Datenbankdesign.
- Wir haben schon gelernt, dass ein *Schlüssel(kandidat)* ein minimaler Superschlüssel ist, also eine minimale Menge von Attributen, mit denen eine Entität identifiziert werden kann.

Schlüssel



- Schlüssel machen diese Datensätze eindeutig bzw. einmalig.
- Schlüssel spielen daher eine wichtige Rolle im relationalen Datenbankdesign.
- Wir haben schon gelernt, dass ein *Schlüssel(kandidat)* ein minimaler Superschlüssel ist, also eine minimale Menge von Attributen, mit denen eine Entität identifiziert werden kann.
- Im relationalen Datenmodell kann man das auch so ausdrücken:⁶³

Definition: Schlüssel

Eine Menge von Attributen $K \subseteq \Sigma(R)$ gegeben als $K = \{k_i : i \in 1..m \wedge k_i \in \Sigma(R)\}$ einer Relation R ist ein *Schlüssel*, wenn und nur wenn:

1. K identifizierend (EN: *identifying*) wirkt: Definieren wir $v_{1,i}$ und $v_{2,i}$ für alle $i \in 1..m$ als die Werte der Attribute k_i für zwei Zeilen r_1 und r_2 in R . Wenn gilt das $r_1 \neq r_2$, dann gibt es mindestens ein $j \in 1..m$ mit $v_{1,j} \neq v_{2,j}$ **UND**
2. Es gibt keine Untermenge $\{k_1, \dots, k_m\}$ mit diese Eigenschaft, also der Schlüssel ist minimal.

Superschlüssel und Primärschlüssel



- Für Superschlüssel S gilt dann einfach das $K \subseteq S \subseteq \Sigma(R)$.

Superschlüssel und Primärschlüssel



- Für Superschlüssel S gilt dann einfach das $K \subseteq S \subseteq \Sigma(R)$.
- Jede Relation muss mindestens einen Schlüssel haben, denn die Dateinsätze müssen ja eindeutig / einmalig sein.

Superschlüssel und Primärschlüssel



- Für Superschlüssel S gilt dann einfach das $K \subseteq S \subseteq \Sigma(R)$.
- Jede Relation muss mindestens einen Schlüssel haben, denn die Dateinsätze müssen ja eindeutig / einmalig sein.
- Einer der Schlüssel wird dann als Primärschlüssel ausgewählt.

Superschlüssel und Primärschlüssel



- Für Superschlüssel S gilt dann einfach das $K \subseteq S \subseteq \Sigma(R)$.
- Jede Relation muss mindestens einen Schlüssel haben, denn die Dateinsätze müssen ja eindeutig / einmalig sein.
- Einer der Schlüssel wird dann als Primärschlüssel ausgewählt.
- Wir benutzen oftmals aber einen Ersatzschlüssel (EN: *surrogate key*), also einen Wert, der automatisch von DBMS für jeden Datensatz generiert wird, als Primärschlüssel.

Fremdschlüssel

- Wir wissen auch schon, dass die Datensätze einer Relation Datensätze in anderen Relationen referenzieren können.



Fremdschlüssel

- Wir wissen auch schon, dass die Datensätze einer Relation Datensätze in anderen Relationen referenzieren können.
- Das geht über Fremdschlüssel (EN: *foreign keys*)⁶³.



Fremdschlüssel



- Wir wissen auch schon, dass die Datensätze einer Relation Datensätze in anderen Relationen referenzieren können.
- Das geht über Fremdschlüssel (EN: *foreign keys*)⁶³.

Definition: Fremdschlüssel

Eine Menge von Attributen F im Schema $\Sigma(R_1)$ der Relation R_1 wird Fremdschlüssel (EN: *foreign key*) genannt, wenn

1. die Attribute von F die selben Domänen wie die Attribute des Primärschlüssels P einer anderen Relation R_2 haben **und**
2. jeder Wert von F in einem Tupel $r_1 \in R_1$ entweder als Wert von P für ein Tupel $r_2 \in R_2$ auftaucht oder **NULL** ist.

Fremdschlüssel



- Wir wissen auch schon, dass die Datensätze einer Relation Datensätze in anderen Relationen referenzieren können.
- Das geht über Fremdschlüssel (EN: *foreign keys*)⁶³.

Definition: Fremdschlüssel

Eine Menge von Attributen F im Schema $\Sigma(R_1)$ der Relation R_1 wird Fremdschlüssel (EN: *foreign key*) genannt, wenn

1. die Attribute von F die selben Domänen wie die Attribute des Primärschlüssels P einer anderen Relation R_2 haben **und**
2. jeder Wert von F in einem Tupel $r_1 \in R_1$ entweder als Wert von P für ein Tupel $r_2 \in R_2$ auftaucht oder **NULL** ist.

- Damit haben wir die Definitionen von Schlüssel(kandidaten), Primärschlüsseln und Fremdschlüsseln aus der konzeptuellen Ebene in die logische Ebene für das relationale Modell gehoben.



Relationale Datenbankmanagementsysteme



Relationale Datenbankmanagementsysteme



- In den 1980ern gab es viele Anbieter von DBMSen, die die Definition des relationalen Datenmodells von Codd nicht richtig implementiert hatten.

Relationale Datenbankmanagementsysteme



- In den 1980ern gab es viele Anbieter von DBMSen, die die Definition des relationalen Datenmodells von Codd nicht richtig implementiert hatten.
- Diese hatten oft Mechaniken eingebaut, mit denen die relationalen Eigenschaften umgehen werden konnten.

Relationale Datenbankmanagementsysteme



- In den 1980ern gab es viele Anbieter von DBMSen, die die Definition des relationalen Datenmodells von Codd nicht richtig implementiert hatten.
- Diese hatten oft Mechaniken eingebaut, mit denen die relationalen Eigenschaften umgehen werden konnten.
- Als Antwort darauf hat Codd 13 Regeln definiert, die jedes relationale DBMS einhalten muss^{22,23,67,72,73}.

Relationale Datenbankmanagementsysteme



- In den 1980ern gab es viele Anbieter von DBMSen, die die Definition des relationalen Datenmodells von Codd nicht richtig implementiert hatten.
- Diese hatten oft Mechaniken eingebaut, mit denen die relationalen Eigenschaften umgehen werden konnten.
- Als Antwort darauf hat Codd 13 Regeln definiert, die jedes relationale DBMS einhalten muss^{22,23,67,72,73}.
- Weil die erste Regel *Rule 0* genannt wird, werden die 13 Regeln auch manchmal als *the twelve rules* bezeichnet.

Die 13 Regeln

0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.



Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?
 - Wir hatten ein neues Benutzerkonto namens `boss` für unser DBMS erstellt.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?
 - Wir hatten ein neues Benutzerkonto namens `boss` für unser DBMS erstellt.
 - Als wir die Liste existierender Benutzer geprüft haben, hatten wir `SELECT username FROM pg_catalog.pg_user` geschrieben.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?
 - Wir hatten ein neues Benutzerkonto namens `boss` für unser DBMS erstellt.
 - Als wir die Liste existierender Benutzer geprüft haben, hatten wir `SELECT username FROM pg_catalog.pg_user` geschrieben.
 - `pg_catalog.pg_user` ist eine Tabelle im Systemkatalog.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Das beinhaltet auch die Namen von Tabellen und Spalten, die Spaltentypen, usw., die auch in Tabellen gespeichert werden müssen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?
 - Wir hatten ein neues Benutzerkonto namens `boss` für unser DBMS erstellt.
 - Als wir die Liste existierender Benutzer geprüft haben, hatten wir `SELECT username FROM pg_catalog.pg_user` geschrieben.
 - `pg_catalog.pg_user` ist eine Tabelle im Systemkatalog.
 - Als wir uns mit LibreOffice Base auf unsere Datenbank verbunden hatten, haben wir viele komische Tabellen gesehen.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
 - Solche speziellen Tabellen speichern die Struktur einer Datenbank und sind normalerweise Teil eines fest eingebauten Systemkatalogs.
 - Dieser Systemkatalog speicher die Metadaten eines Systems und ist selbst eine (oder ist Teil einer) relationalen Datenbank.
 - Erinnern Sie sich an unsere ersten einfachen Beispiele als wir mit PostgreSQL gearbeitet hatten?
 - Wir hatten ein neues Benutzerkonto namens `boss` für unser DBMS erstellt.
 - Als wir die Liste existierender Benutzer geprüft haben, hatten wir `SELECT username FROM pg_catalog.pg_user` geschrieben.
 - `pg_catalog.pg_user` ist eine Tabelle im Systemkatalog.
 - Als wir uns mit LibreOffice Base auf unsere Datenbank verbunden hatten, haben wir viele komische Tabellen gesehen.
 - Sie gehörten zum Systemkatalog.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellennamen, Primärschlüsselwert und Spaltennamen.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellennamen, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellennamen, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellename, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.
 - Echte Daten beinhalten unspezifizierten Elemente.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellennamen, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.
 - Echte Daten beinhalten unspezifizierten Elemente.
 - Es kann Adressen ohne Hausnummern oder Leute ohne Telefonnummer geben.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellename, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.
 - Echte Daten beinhalten unspezifizierten Elemente.
 - Es kann Adressen ohne Hausnummern oder Leute ohne Telefonnummer geben.
 - Also müssen wir mit diesen Situationen umgehen können.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellename, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.
 - Echte Daten beinhalten unspezifizierten Elemente.
 - Es kann Adressen ohne Hausnummern oder Leute ohne Telefonnummer geben.
 - Also müssen wir mit diesen Situationen umgehen können.
 - Allerdings verletzen undefinierte oder fehlende Werte auch die Definition von Tupeln in Relationen.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellename, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
 - Diese Regel hat über die Jahre zu vielen Diskussionen geführt¹⁵.
 - Echte Daten beinhalten unspezifizierten Elemente.
 - Es kann Adressen ohne Hausnummern oder Leute ohne Telefonnummer geben.
 - Also müssen wir mit diesen Situationen umgehen können.
 - Allerdings verletzen undefinierte oder fehlende Werte auch die Definition von Tupeln in Relationen.
 - Wir könnten uns aber zumindest vorstellen, dass jede Domäne jedes Attributs den zusätzlichen Wert **NULL** beinhaltet.

Die 13 Regeln



0. Ein relationales DBMS muss die Datenbank ausschließlich durch relationale Methoden verwalten.
1. Alle Informationen in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Art repräsentiert: Als Werte in Tabellen.
2. Jedes Datum in einer relationalen Datenbank muss zugreifbar sein über eine Kombination von Tabellennamen, Primärschlüsselwert und Spaltenname.
3. **NULL**-Werte werden unterstützt, um fehlende und unpassende Information auf eine systematische Art darstellen zu können, unabhängig vom Datentyp.
4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),
 - Einschränkungen definieren, um Datenintegrität sicherzustellen (z. B. Grenzen für Wertebereiche, Fremdschlüssen, ...),

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),
 - Einschränkungen definieren, um Datenintegrität sicherzustellen (z. B. Grenzen für Wertebereiche, Fremdschlüssen, ...),
 - Autorisation (z. B. Benutzermanagement) und

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),
 - Einschränkungen definieren, um Datenintegrität sicherzustellen (z. B. Grenzen für Wertebereiche, Fremdschlüssen, ...),
 - Autorisation (z. B. Benutzermanagement) und
 - Transaktionsgrenzen (Beginn, Commit und Rollback).

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),
 - Einschränkungen definieren, um Datenintegrität sicherzustellen (z. B. Grenzen für Wertebereiche, Fremdschlüssen, ...),
 - Autorisation (z. B. Benutzermanagement) und
 - Transaktionsgrenzen (Beginn, Commit und Rollback).

Im Falle unseres Kurses ist die Sprache SQL.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können und die die folgenden Elemente unterstützt:
 - Daten definieren (z. B. Erstellen von Tabellen),
 - Sichten definieren,
 - Daten manipulieren (z. B. hinzufügen und löschen von Daten),
 - Einschränkungen definieren, um Datenintegrität sicherzustellen (z. B. Grenzen für Wertebereiche, Fremdschlüssen, ...),
 - Autorisation (z. B. Benutzermanagement) und
 - Transaktionsgrenzen (Beginn, Commit und Rollback).

Im Falle unseres Kurses ist die Sprache SQL. Natürlich könnte man sich aber auch eine andere Sprache ausdenken.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten. Die Operationen gelten also nicht nur für einzelne Datensätze (Zeilen), sondern können mehrere Zeilen auf einmal bearbeiten, denn ihre Eingabedaten sind auch Relationen (Tabellen, Ergebnisse von `SELECT`, ...).

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden. Die Art wie ein DBMS Daten speichert hat keinen Einfluss darauf, wie Anwendungen auf die Daten mit Hilfe der text-basierten Sprache zugreifen.

Die 13 Regeln



4. Die Datenbankbeschreibung wird auf logischer Ebene genau wie normale Daten repräsentiert, so dass autorisierte Benutzer mit der selben relationalen Sprache auf sie zugreifen können, die sie auch auf normale Daten anwenden würden.
5. Ein relationales System muss mindestens eine Sprache unterstützen, deren Statements mit einer wohldefinierten Syntax als Text ausgedrückt werden können.
6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedatet.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.

Die 13 Regeln



6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedated.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
 - Sagen wir, dass wir eine Tabelle in zwei Tabellen aufteilen und Zeilen auf beide verteilen, wobei wir Spalten und Primärschlüssel unberührt lassen.

Die 13 Regeln



6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedated.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
 - Sagen wir, dass wir eine Tabelle in zwei Tabellen aufteilen und Zeilen auf beide verteilen, wobei wir Spalten und Primärschlüssel unberührt lassen.
 - Wir können eine Sicht entwickeln, die die beiden Tabellen (mit **UNION**) zusammenführt.

Die 13 Regeln



6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedated.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
 - Sagen wir, dass wir eine Tabelle in zwei Tabellen aufteilen und Zeilen auf beide verteilen, wobei wir Spalten und Primärschlüssel unberührt lassen.
 - Wir können eine Sicht entwickeln, die die beiden Tabellen (mit **UNION**) zusammenführt.
 - Eine Applikation, die darauf zugreift, sieht keine Veränderung.

Die 13 Regeln



6. Alle Sichten, die theoretisch upgedatet werden können, werden auch vom System upgedated.
7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).

Die 13 Regeln



7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen). Wir haben das mehrmals gemacht, z. B. mit der **PRIMARY KEY**-Einschränkung, der **REFERENCES**-Einschränkung und der **CHECK**-Einschränkung.

Die 13 Regeln



7. Die Fähigkeit eine Relation als Operand zu benutzen gilt nicht nur für das Auslesen von Daten, sondern auch für das Einfügen, Ändern, oder Löschen von Daten.
8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen). Wir haben das mehrmals gemacht, z. B. mit der **PRIMARY KEY**-Einschränkung, der **REFERENCES**-Einschränkung und der **CHECK**-Einschränkung. Alle wurden in SQL definiert.

Die 13 Regeln



8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.

Die 13 Regeln



8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
 - Ein DBMS *kann* unterstützen, dass Daten über mehrere Computer (Knoten) oder einem Cluster verteilt gespeichert werden.

Die 13 Regeln



8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
 - Ein DBMS *kann* unterstützen, dass Daten über mehrere Computer (Knoten) oder einem Cluster verteilt gespeichert werden.
 - *Wenn* es das unterstützt, dann darf das keinen Einfluss auf die SQL-Programme haben, die auf es zugreifen.

Die 13 Regeln



8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
 - Ein DBMS *kann* unterstützen, dass Daten über mehrere Computer (Knoten) oder einem Cluster verteilt gespeichert werden.
 - *Wenn* es das unterstützt, dann darf das keinen Einfluss auf die SQL-Programme haben, die auf es zugreifen.
 - In diesem Fall muss das DBMS die Anfragen auf die entsprechenden Knoten verteilen und die Ergebnisse wieder zusammensetzen.

Die 13 Regeln



8. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen, die auf die physische Speicher- oder Zugriffsmethode angewendet werden.
9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
 - Ein DBMS *kann* unterstützen, dass Daten über mehrere Computer (Knoten) oder einem Cluster verteilt gespeichert werden.
 - *Wenn* es das unterstützt, dann darf das keinen Einfluss auf die SQL-Programme haben, die auf es zugreifen.
 - In diesem Fall muss das DBMS die Anfragen auf die entsprechenden Knoten verteilen und die Ergebnisse wieder zusammensetzen.
 - Wenn ein DBMS verteiltes Speichern nicht unterstützt, dann erfüllt es diese Bedingung automatisch.

Die 13 Regeln



9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
12. Falls ein relationales DBMS zusätzlich eine low-level Sprache unterstützt, dann darf es nicht möglich sein, damit die Integritätsregeln/Einschränkungen der high-level Sprache (wie SQL) zu umgehen.

Die 13 Regeln



9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
12. Falls ein relationales DBMS zusätzlich eine low-level Sprache unterstützt, dann darf es nicht möglich sein, damit die Integritätsregeln/Einschränkungen der high-level Sprache (wie SQL) zu umgehen. Ein DBMS muss mindestens eine relationale Sprache unterstützen, aber es kann auch weitere, andere Sprachen unterstützen, vielleicht aus Performancegründen.

Die 13 Regeln



9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
12. Falls ein relationales DBMS zusätzlich eine low-level Sprache unterstützt, dann darf es nicht möglich sein, damit die Integritätsregeln/Einschränkungen der high-level Sprache (wie SQL) zu umgehen. Ein DBMS muss mindestens eine relationale Sprache unterstützen, aber es kann auch weitere, andere Sprachen unterstützen, vielleicht aus Performancegründen. Keine der Zugriffsmethoden darf aber die Integrität der relationalen Daten brechen.

Die 13 Regeln



9. Anwendungsprogramme und Terminalaktivitäten bleiben logisch unberührt von allen Veränderungen der zugrundeliegenden Tabellen, die theoretisch ein Unberührtbleiben zulassen.
10. Integritäts-Einschränkungen für eine Datenbank müssen in einer Sprache definiert werden können und im Systemkatalog gespeichert werden (nicht in Anwendungen).
11. Ein relationales DBMS hat Verteilungs-Unabhängigkeit.
12. Falls ein relationales DBMS zusätzlich eine low-level Sprache unterstützt, dann darf es nicht möglich sein, damit die Integritätsregeln/Einschränkungen der high-level Sprache (wie SQL) zu umgehen. Ein DBMS muss mindestens eine relationale Sprache unterstützen, aber es kann auch weitere, andere Sprachen unterstützen, vielleicht aus Performancegründen. Keine der Zugriffsmethoden darf aber die Integrität der relationalen Daten brechen.

Diese Regeln werden zum größten Teil von modernen DBMSen implementiert.

Aus ihnen können wir auch bereits die Features lernen, die solche DBMSe unterstützen.



Zusammenfassung



Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.
- Wir können diese Definitionen mit den Definitionen, die wir auf der konzeptuellen Ebene diskutiert haben, gut in Einklang bringen.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.
- Wir können diese Definitionen mit den Definitionen, die wir auf der konzeptuellen Ebene diskutiert haben, gut in Einklang bringen.
- Wir kennen auch die grundlegenden Anforderungen an relationale DBMSs.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.
- Wir können diese Definitionen mit den Definitionen, die wir auf der konzeptuellen Ebene diskutiert haben, gut in Einklang bringen.
- Wir kennen auch die grundlegenden Anforderungen an relationale DBMSs.
- Wenn wir das mit dem, was wir am Beispiel der Fabrikdatenbank gelernt haben, kombinieren, dann haben wir schon ein recht klares Verständnis davon, was ein relationales DBMS wie PostgreSQL uns bietet.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.
- Wir können diese Definitionen mit den Definitionen, die wir auf der konzeptuellen Ebene diskutiert haben, gut in Einklang bringen.
- Wir kennen auch die grundlegenden Anforderungen an relationale DBMSs.
- Wenn wir das mit dem, was wir am Beispiel der Fabrikdatenbank gelernt haben, kombinieren, dann haben wir schon ein recht klares Verständnis davon, was ein relationales DBMS wie PostgreSQL uns bietet.
- Und wir können uns in etwa vorstellen, wie wir ein konzeptuelles Modell in ein logisches Modell überführen könnten.

Zusammenfassung



- Wie haben nun ein gutes Verständnis davon, wie das relationale Datenmodell funktioniert.
- Wir kennen die grundlegenden Definitionen.
- Wir können diese Definitionen mit den Definitionen, die wir auf der konzeptuellen Ebene diskutiert haben, gut in Einklang bringen.
- Wir kennen auch die grundlegenden Anforderungen an relationale DBMSs.
- Wenn wir das mit dem, was wir am Beispiel der Fabrikdatenbank gelernt haben, kombinieren, dann haben wir schon ein recht klares Verständnis davon, was ein relationales DBMS wie PostgreSQL uns bietet.
- Und wir können uns in etwa vorstellen, wie wir ein konzeptuelles Modell in ein logisches Modell überführen könnten.
- Und genau das machen wir als nächstes.



谢谢你们！
Thank you!
Vielen Dank!



References I



- [1] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume I: Introduction to SQL Queries*. Tetras Publishing, Okt. 2018. ISBN: **978-1-9996172-4-0**. See also² (siehe S. **123**, **134**).
- [2] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume II: In-Depth Querying*. Tetras Publishing, Okt. 2018. ISBN: **978-1-9996172-5-7**. See also¹ (siehe S. **123**, **134**).
- [3] Richard Barker. *Case*Method: Entity Relationship Modelling (Oracle)*. 1. Aufl. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., Jan. 1990. ISBN: **978-0-201-41696-1** (siehe S. **133**).
- [4] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: **978-1-0981-1340-7** (siehe S. **134**, **136**).
- [5] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Okt. 2019. ISBN: **978-1-4842-5514-8** (siehe S. **134**).
- [6] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: **978-1-83898-747-3** (siehe S. **134**).
- [7] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) und Newsgroups: alt.hypertext, 6. Aug. 1991. URL: <https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt> (besucht am 2025-02-05) (siehe S. **136**).
- [8] Alex Berson. *Client/Server Architecture*. 2. Aufl. Computer Communications Series. New York, NY, USA: McGraw-Hill, 29. März 1996. ISBN: **978-0-07-005664-0** (siehe S. **133**).
- [9] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: **978-1-80324-028-2** (siehe S. **134**).
- [10] Silvia Botros und Jeremy Tinley. *High Performance MySQL*. 4. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Nov. 2021. ISBN: **978-1-4920-8051-0** (siehe S. **135**).
- [11] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: **978-0-13-769132-6** (siehe S. **135**).

References II



- [12] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 133).
- [13] Ben Brumm. "A Guide to the Entity Relationship Diagram (ERD)". In: *Database Star*. Armadale, VIC, Australia: Elevated Online Services PTY Ltd., 30. Juli 2019–23. Dez. 2023. URL: <https://www.databasestar.com/entity-relationship-diagram> (besucht am 2025-03-29) (siehe S. 133).
- [14] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 134, 135).
- [15] Donald D. Chamberlin. "50 Years of Queries". *Communications of the ACM (CACM)* 67(8):110–121, Aug. 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/3649887. URL: <https://cacm.acm.org/research/50-years-of-queries> (besucht am 2025-01-09) (siehe S. 30–33, 77–82, 136).
- [16] Peter Pin-Shan Chen. "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned". In: *Software Pioneers: Contributions to Software Engineering*. Hrsg. von Manfred Broy und Ernst Denert. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, Feb. 2002, S. 296–310. doi:10.1007/978-3-642-59412-0_17. URL: http://bit.csc.lsu.edu/%7Echen/pdf/Chen_Pioneers.pdf (besucht am 2025-03-06) (siehe S. 133).
- [17] Peter Pin-Shan Chen. "The Entity-Relationship Model – Toward a Unified View of Data". *ACM Transactions on Database Systems (TODS)* 1(1):9–36, März 1976. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0362-5915. doi:10.1145/320434.320440 (siehe S. 124, 133).
- [18] Peter Pin-Shan Chen. "The Entity-Relationship Model: Toward a Unified View of Data". In: *1st International Conference on Very Large Data Bases (VLDB'1975)*. 22.–24. Sep. 1975, Framingham, MA, USA. Hrsg. von Douglas S. Kerr. New York, NY, USA: Association for Computing Machinery (ACM), 1975, S. 173. ISBN: 978-1-4503-3920-9. doi:10.1145/1282480.1282492. See¹⁷ for a more comprehensive introduction. (Siehe S. 133).
- [19] Christmas, FL, USA: Simon Sez IT. *Microsoft Access 2021 – Beginner to Advanced*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2023. ISBN: 978-1-83546-911-8 (siehe S. 134).
- [20] David Clinton und Christopher Negus. *Ubuntu Linux Bible*. 10. Aufl. Bible Series. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 10. Nov. 2020. ISBN: 978-1-119-72233-5 (siehe S. 136).

References III



- [21] Edgar Frank „Ted“ Codd. “A Relational Model of Data for Large Shared Data Banks”. *Communications of the ACM (CACM)* 13(6):377–387, Juni 1970. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: **0001-0782**. doi:10.1145/362384.362685. URL: <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf> (besucht am 2025-01-05) (siehe S. 17–21, 34–43, 135).
- [22] Edgar Frank „Ted“ Codd. “An Evaluation Scheme for Database Management Systems that are claimed to be Relational”. In: *Second International Conference on Data Engineering (ICDE'1986)*. 5.–7. Feb. 1986, Los Angeles, CA, USA. Los Alamitos, CA, USA: IEEE Computer Society, 1986, S. 720–729. ISBN: **978-0-8186-0655-7**. doi:10.1109/ICDE.1986.7266284. URL: [https://www.fsmwarden.com/Codd/An%20evaluation%20scheme%20\(1986\).pdf](https://www.fsmwarden.com/Codd/An%20evaluation%20scheme%20(1986).pdf) (besucht am 2025-04-10). Keynote Address. ©1985 CW Communications, Inc., excerpted from *Computerworld*²³ (siehe S. 60–63, 125).
- [23] Edgar Frank „Ted“ Codd. “Is your DBMS really relational? (Part 1)”. *Computerworld* 19:ID1, 14. Okt. 1985. Framingham, MA, USA: CW Communications, Inc. and Needham, MA, USA: Foundry (formerly IDG Communications, Inc.) ISSN: **0010-4841**. URL: <https://thaumatorium.com/articles/the-papers-of-ef-the-coddfather-codd/1985a-is-your-dbms-really-relational> (besucht am 2025-04-10). See also the Keynote Address²² (siehe S. 60–63, 125).
- [24] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 136).
- [25] Matt David und Blake Barnhill. *How to Teach People SQL*. San Francisco, CA, USA: The Data School, Chart.io, Inc., 10. Dez. 2019–10. Apr. 2023. URL: <https://dataschool.com/how-to-teach-people-sql> (besucht am 2025-02-27) (siehe S. 136).
- [26] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 136).
- [27] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide WebW[?]: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: **978-0-13-299045-5** (siehe S. 136).
- [28] Pooyan Doozandeh und Frank E. Ritter. “Some Tips for Academic Writing and Using Microsoft Word”. *XRDS: Crossroads, The ACM Magazine for Students* 26(1):10–11, Herbst 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: **1528-4972**. doi:10.1145/3351470 (siehe S. 135).

References IV



- [29] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2015. ISBN: 978-1-4493-6290-4 (siehe S. 134, 135).
- [30] Steve Fanning, Vasudev Narayanan, „flywire“, Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffrers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen und Randolph Gam. *Base Guide 7.3. Revision 1. Based on LibreOffice 7.3 Community*. Berlin, Germany: The Document Foundation, Aug. 2022. URL: <https://books.libreoffice.org/en/BG73/BG73-BaseGuide.pdf> (besucht am 2025-01-13) (siehe S. 134).
- [31] Luca Ferrari und Enrico Pirozzi. *Learn PostgreSQL*. 2. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2023. ISBN: 978-1-83763-564-1 (siehe S. 135).
- [32] Jonas Gamalielsson und Björn Lundell. "Long-Term Sustainability of Open Source Software Communities beyond a Fork: A Case Study of LibreOffice". In: *8th IFIP WG 2.13 International Conference on Open Source Systems: Long-Term Sustainability OSS'2012*. 10.–13. Sep. 2012, Hammamet, Tunisia. Hrsg. von Imed Hammouda, Björn Lundell, Tommi Mikkonen und Walt Scacchi. Bd. 378. IFIP Advances in Information and Communication Technology (IFIPACT). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2012, S. 29–47. ISSN: 1868-4238. ISBN: 978-3-642-33441-2. doi:10.1007/978-3-642-33442-9_3 (siehe S. 134).
- [33] Dawn Griffiths. *Excel Cookbook – Receipts for Mastering Microsoft Excel*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2024. ISBN: 978-1-0981-4332-9 (siehe S. 134).
- [34] Terry Halpin und Tony Morgan. *Information Modeling and Relational Databases*. 3. Aufl. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Juli 2024. ISBN: 978-0-443-23791-1 (siehe S. 135).
- [35] Jan L. Harrington. *Relational Database Design and Implementation*. 4. Aufl. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Apr. 2016. ISBN: 978-0-12-849902-3 (siehe S. 135).
- [36] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 134).
- [37] Matthew Helmke. *Ubuntu Linux Unleashed 2021 Edition*. 14. Aufl. Reading, MA, USA: Addison-Wesley Professional, Aug. 2020. ISBN: 978-0-13-668539-5 (siehe S. 134, 136).

References V



- [38] Manuel Hoffmann, Frank Nagle und Yanuo Zhou. *The Value of Open Source Software*. Working Paper 24-038. Boston, MA, USA: Harvard Business School, 1. Jan. 2024. URL: https://www.hbs.edu/ris/Publication%20Files/24-038_51f8444f-502c-4139-8bf2-56eb4b65c58a.pdf (besucht am 2025-06-04) (siehe S. 135).
- [39] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (siehe S. 135).
- [40] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Juni 2023. URL: [https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_\(en\).zip](https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip) (besucht am 2025-01-08). Consists of several parts, see <https://modern-sql.com/standard> for information where to obtain them. (Siehe S. 136).
- [41] Shannon Kempe und Paul Williams. *A Short History of the ER Diagram and Information Modeling*. Studio City, CA, USA: Dataversity Digital LLC, 25. Sep. 2012. URL: <https://www.dataversity.net/a-short-history-of-the-er-diagram-and-information-modeling> (besucht am 2025-03-06) (siehe S. 133).
- [42] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 135).
- [43] Joan Lambert und Curtis Frye. *Microsoft Office Step by Step (Office 2021 and Microsoft 365)*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Juni 2022. ISBN: 978-0-13-754493-6 (siehe S. 134).
- [44] Kent D. Lee und Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (siehe S. 135).
- [45] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: <https://www.libreoffice.org> (besucht am 2024-12-12) (siehe S. 134).
- [46] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra und William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 3. Jan. 2025. URL: <https://www.britannica.com/technology/client-server-architecture> (besucht am 2025-01-20) (siehe S. 133).

References VI



- [47] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 135).
- [48] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: <https://mariadb.com/kb/en/documentation> (besucht am 2025-04-24) (siehe S. 134).
- [49] Ron McFadyen und Cindy Miller. *Relational Databases and Microsoft Access*. 3. Aufl. Palatine, IL, USA: Harper College, 2014–2019. URL: <https://harpercollege.pressbooks.pub/relationaldatabases> (besucht am 2025-04-11) (siehe S. 134).
- [50] Jim Melton und Alan R. Simon. *SQL: 1999 – Understanding Relational Language Components*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Juni 2001. ISBN: 978-1-55860-456-8 (siehe S. 136).
- [51] *Microsoft Word*. Redmond, WA, USA: Microsoft Corporation, 2024. URL: <https://www.microsoft.com/en-us/microsoft-365/word> (besucht am 2024-12-12) (siehe S. 135).
- [52] Cameron Newham und Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (siehe S. 133).
- [53] Regina O. Obe und Leo S. Hsu. *PostgreSQL: Up and Running*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Okt. 2017. ISBN: 978-1-4919-6336-4 (siehe S. 135).
- [54] Robert Orfali, Dan Harkey und Jeri Edwards. *Client/Server Survival Guide*. 3. Aufl. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 25. Jan. 1999. ISBN: 978-0-471-31615-2 (siehe S. 133).
- [55] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Kononov, Robert M. Flight, Kai Blin und Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), 14. Juli 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (siehe S. 133).
- [56] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 135).
- [57] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6 (siehe S. 133).

References VII



- [58] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: **978-1-78398-154-0** (siehe S. **134**).
- [59] Mike Reichardt, Michael Gundall und Hans D. Schotten. "Benchmarking the Operation Times of NoSQL and MySQL Databases for Python Clients". In: *47th Annual Conference of the IEEE Industrial Electronics Society (IECON'2021)*. 13.–15. Okt. 2021, Toronto, ON, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2021, S. 1–8. ISSN: **2577-1647**. ISBN: **978-1-6654-3554-3**. doi:[10.1109/IECON48115.2021.9589382](https://doi.org/10.1109/IECON48115.2021.9589382) (siehe S. **135**).
- [60] Mark Richards und Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: **978-1-4920-4345-4** (siehe S. **133**).
- [61] Heinz Schweppe und Manuel Scholz. *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V> (besucht am 2025-01-08).
- [62] Heinz Schweppe und Manuel Scholz. "Schema Definition with SQL / DDL (II)". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 4. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-06-DDLSQL-2-2.pdf> (besucht am 2025-05-15) (siehe S. **17–23**, **137**).
- [63] Heinz Schweppe und Manuel Scholz. "Schema Design: Logical Design using the Relational Data Model". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 3. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-04-RDM1-2.pdf> (besucht am 2025-04-04) (siehe S. **5–15**, **34–39**, **45–50**, **55–58**).
- [64] Winfried Seimert. *LibreOffice 7.3 – Praxiswissen für Ein- und Umsteiger*. Blaufelden, Schwäbisch Hall, Baden-Württemberg, Germany: mitp Verlags GmbH & Co. KG, Apr. 2022. ISBN: **978-3-7475-0504-5** (siehe S. **134**).
- [65] Yuriy Shamshin. "Conceptual Database Model. Entity Relationship Diagram (ERD)". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. Kap. 04. URL: https://dbs.academy.lv/lection/dbs_LS04EN_erd.pdf (besucht am 2025-03-29) (siehe S. **133**).
- [66] Yuriy Shamshin. *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. URL: <https://dbs.academy.lv> (besucht am 2025-01-11).

References VIII



- [67] Yuriy Shamshin. "Logical Data Models. Relation Model. Relation Algebra". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. Kap. 05. URL: https://dbs.academy.lv/lection/dbs_LS05EN_rm.pdf (besucht am 2025-04-10) (siehe S. 34–39, 60–63).
- [68] Ellen Siever, Stephen Figgins, Robert Love und Arnold Robbins. *Linux in a Nutshell*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2009. ISBN: 978-0-596-15448-6 (siehe S. 134).
- [69] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 133).
- [70] John Miles Smith und Philip Yen-Tang Chang. "Optimizing the Performance of a Relational Algebra Database Interface". *Communications of the ACM (CACM)* 18(10):568–579, Okt. 1975. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/361020.361025 (siehe S. 135).
- [71] "SQL Commands". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), 20. Feb. 2025. Kap. Part VI. Reference. URL: <https://www.postgresql.org/docs/17/sql-commands.html> (besucht am 2025-02-25) (siehe S. 136).
- [72] Ryan K. Stephens und Ronald R. Plew. *Sams Teach Yourself SQL in 21 Days*. 4. Aufl. Sams Tech Yourself. Indianapolis, IN, USA: SAMS Technical Publishing und Hoboken, NJ, USA: Pearson Education, Inc., Okt. 2002. ISBN: 978-0-672-32451-2 (siehe S. 60–63, 130, 136).
- [73] Ryan K. Stephens, Ronald R. Plew, Bryan Morgan und Jeff Perkins. *SQL in 21 Tagen. Die Datenbank-Abfragesprache SQL vollständig erklärt (in 14/21 Tagen)*. 6. Aufl. Burghann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of ⁷² (siehe S. 60–63, 136).
- [74] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 135, 136).
- [75] Alkin Tezuysal und Ibrar Ahmed. *Database Design and Modeling with PostgreSQL and MySQL*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2024. ISBN: 978-1-80323-347-5 (siehe S. 135).
- [76] Linus Torvalds. "The Linux Edge". *Communications of the ACM (CACM)* 42(4):38–39, Apr. 1999. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/299157.299165 (siehe S. 134).

References IX



- [77] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: **979-8-8688-0215-7** (siehe S. **133**, **136**).
- [78] Laurie A. Ulrich und Ken Cook. *Access For Dummies*. Hoboken, NJ, USA: For Dummies (Wiley), Dez. 2021. ISBN: **978-1-119-82908-9** (siehe S. **134**).
- [79] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: **978-0-13-792931-3** (siehe S. **134**).
- [80] Thomas Weise (汤卫恩). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: <https://thomasweise.github.io/databases> (besucht am 2025-01-05) (siehe S. **133–135**).
- [81] Thomas Weise (汤卫恩). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: <https://thomasweise.github.io/programmingWithPython> (besucht am 2025-01-05) (siehe S. **135**).
- [82] Matthew West. *Developing High Quality Data Models*. Version: 2.0, Issue: 2.1. London, England, UK: Shell International Limited und European Process Industries STEP Technical Liaison Executive (EPISTLE); Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, 8. Dez. 1995–Dez. 2010. ISBN: **978-0-12-375107-2**. URL: <https://www.researchgate.net/publication/286610894> (besucht am 2025-03-24). Edited by Julian Fowler (siehe S. **133**).
- [83] *What is a Relational Database?* Armonk, NY, USA: International Business Machines Corporation (IBM), 20. Okt. 2021–12. Dez. 2024. URL: <https://www.ibm.com/think/topics/relational-databases> (besucht am 2025-01-05) (siehe S. **135**).
- [84] Ulf Michael „Monty“ Widenius, David Axmark und Uppsala, Sweden: MySQL AB. *MySQL Reference Manual – Documentation from the Source*. Sebastopol, CA, USA: O'Reilly Media, Inc., 9. Juli 2002. ISBN: **978-0-596-00265-7** (siehe S. **135**).
- [85] Kinza Yasar und Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., Juni 2024. URL: <https://www.techtarget.com/searchdatamanagement/definition/database-management-system> (besucht am 2025-01-11) (siehe S. **133**).

References X



- [86] Pavlo V. Zahorodko und Pavlo V. Merzlykin. "An Approach for Processing and Document Flow Automation for Microsoft Word and LibreOffice Writer File Formats". In: *4th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW'2021)*. 18. Dez. 2021, Virtual Event and Kryvyi Rih, Ukraine. Hrsg. von Arnold E. Kiv, Serhiy O. Semerikov, Vladimir N. Soloviev und Andrii M. Striuk. Bd. 3077 der Reihe CEUR Workshop Proceedings ([CEUR-WS.org](https://ceur-ws.org)). Aachen, Nordrhein-Westfalen, Germany: CEUR-WS Team, Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen, 2022, S. 66–82. ISSN: 1613-0073. URL: <https://ceur-ws.org/Vol-3077/paper12.pdf> (besucht am 2025-10-04) (siehe S. 134, 135).
- [87] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 133).

Glossary (in English) I



Bash is a the shell used under Ubuntu Linux, i.e., the program that „runs“ in the terminal and interprets your commands, allowing you to start and interact with other programs^{12,52,87}. Learn more at <https://www.gnu.org/software/bash>.

client In a client-server architecture, the client is a device or process that requests a service from the server. It initiates the communication with the server, sends a request, and receives the response with the result of the request. Typical examples for clients are web browsers in the internet as well as clients for database management systems (DBMSes), such as psql.

client-server architecture is a system design where a central server receives requests from one or multiple clients^{8,46,54,57,60}. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as psql, or the different applications that access the DBs.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*⁸⁰.

DBMS A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB⁸⁵.

DBS A *database system* is the combination of a DB and a the corresponding DBMS, i.e., basically, an installation of a DBMS on a computer together with one or multiple DBs. DBS = DB + DBMS.

ERD Entity relationship diagrams show the relationships between objects, e.g., between the tables in a DB and how they reference each other^{3,13,16–18,41,65,82}

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes^{69,77}. Learn more at <https://git-scm.com>.

GitHub is a website where software projects can be hosted and managed via the Git VCS^{55,77}. Learn more at <https://github.com>.

Glossary (in English) II



IT information technology

- LAMP Stack** A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP^{14,37}.
- LibreOffice** is an open source office suite^{32,45,64} which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See⁸⁰ for more information and installation instructions.
- LibreOffice Base** is a DBMS that can work on stand-alone files but also connect to other popular relational databases^{30,64}. It is part of LibreOffice^{32,45,64} and has functionality that is comparable to Microsoft Access^{6,19,78}.
- LibreOffice Calc** is a spreadsheet software that allows you to arrange and perform calculations with data in a tabular grid. It is a free and open source spreadsheet software^{45,64}, i.e., an alternative to Microsoft Excel. It is part of LibreOffice^{32,45,64}.
- LibreOffice Writer** is a free and open source text writing program⁸⁶ and part of LibreOffice^{32,45,64}. It is a good alternative to Microsoft Word.
- Linux** is the leading open source operating system, i.e., a free alternative for Microsoft Windows^{4,36,68,76,79}. We recommend using it for this course, for software development, and for research. Learn more at <https://www.linux.org>. Its variant Ubuntu is particularly easy to use and install.
- MariaDB** An open source relational database management system that has forked off from MySQL^{1,2,5,29,48,58}. See <https://mariadb.org> for more information.
- Microsoft Access** is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases^{6,19,49,78}. It is part of Microsoft Office. A free and open source alternative to this commercial software is LibreOffice Base.
- Microsoft Excel** is a spreadsheet program that allows users to store, organize, manipulate, and calculate data in tabular structures^{9,33,43}. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc^{45,64}.
- Microsoft Office** is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access⁴³. LibreOffice is a free and open source alternative.

Glossary (in English) III



- Microsoft Windows** is a commercial proprietary operating system¹¹. It is widely spread, but we recommend using a Linux variant such as Ubuntu for software development and for our course. Learn more at <https://www.microsoft.com/windows>.
- Microsoft Word** is one of the leading text writing programs^{28,51,86} and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.
- MySQL** An open source relational database management system^{10,29,59,75,84}. MySQL is famous for its use in the LAMP Stack. See <https://www.mysql.com> for more information.
- OSS** Open source software, i.e., software that can freely be used, whose source code is made available in the internet, and which is usually developed cooperatively over the internet as well³⁸. Typical examples are Python, Linux, Git, and PostgreSQL.
- PostgreSQL** An open source object-relational DBMS^{31,53,56,75}. See <https://postgresql.org> for more information.
- psql** is the client program used to access the PostgreSQL DBMS server.
- Python** The Python programming language^{39,44,47,81}, i.e., what you will learn about in our book⁸¹. Learn more at <https://python.org>.
- relational database** A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other^{21,34,35,70,74,80,83}.
- server** In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carrying the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers¹⁴ in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the „server computer“⁴².

Glossary (in English) IV



SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases^{15,24–26,40,50,71–74}. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference⁷¹.

terminal A terminal is a text-based window where you can enter commands and execute them^{4,20}. Knowing what a terminal is and how to use it is very essential in any programming- or system administration-related task. If you want to open a terminal under Microsoft Windows, you can Druck auf  + , dann Schreiben von `cmd`, dann Druck auf . Under Ubuntu Linux,  +  +  opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux^{20,37}. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at <https://ubuntu.com>. If you are in China, you can download it from <https://mirrors.ustc.edu.cn/ubuntu-releases>.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code⁷⁷. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

WWW World Wide Web^{7,27}

$i..j$ with $i, j \in \mathbb{Z}$ and $i \leq j$ is the set that contains all integer numbers in the inclusive range from i to j . For example, $5..9$ is equivalent to $\{5, 6, 7, 8, 9\}$

$\text{dom}(a)$ the domain of an attribute a .

$\mathbb{R}$ the set of the real numbers.

Glossary (in English) V



$\Sigma(R)$ the relation schema of relation R ⁶².

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., $\dots, -3, -2, -1, 0, 1, 2, 3, \dots$, and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.