合肥大學
HEFEI UNIVERSITY

Datenbanken

13. Fabrik-Datenbank: Daten ändern und löschen

Thomas Weise (汤卫思)
tweise@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

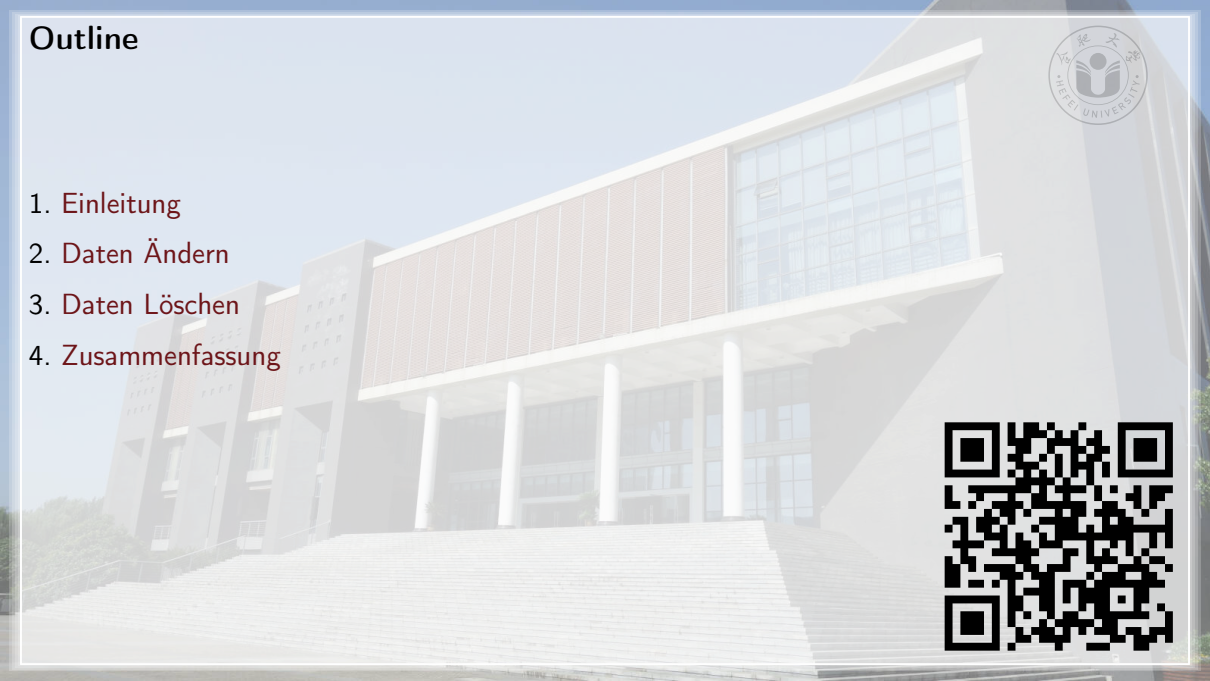
Version: 2025-08-29

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



Einleitung

Einleitung

- Wir haben nun eine Datenbank mit drei Tabellen erstellt (via `CREATE DATABASE` und `CREATE TABLE`).

Einleitung

- Wir haben nun eine Datenbank mit drei Tabellen erstellt (via `CREATE DATABASE` und `CREATE TABLE`).
- Wir haben Daten in die Tabellen eingefügt (via `INSERT INTO`).

Einleitung

- Wir haben nun eine Datenbank mit drei Tabellen erstellt (via `CREATE DATABASE` und `CREATE TABLE`).
- Wir haben Daten in die Tabellen eingefügt (via `INSERT INTO`).
- Und wir haben die Daten wieder ausgelesen (via `SELECT FROM`) und sogar Daten von verschiedenen Tabellen zusammengeführt (via `LEFT JOIN` und `INNER JOIN`).

Einleitung

- Wir haben nun eine Datenbank mit drei Tabellen erstellt (via `CREATE DATABASE` und `CREATE TABLE`).
- Wir haben Daten in die Tabellen eingefügt (via `INSERT INTO`).
- Und wir haben die Daten wieder ausgelesen (via `SELECT FROM`) und sogar Daten von verschiedenen Tabellen zusammengeführt (via `LEFT JOIN` und `INNER JOIN`).
- Was wir noch nicht gemacht haben, ist Daten zu ändern und zu löschen.

Einleitung

- Wir haben nun eine Datenbank mit drei Tabellen erstellt (via `CREATE DATABASE` und `CREATE TABLE`).
- Wir haben Daten in die Tabellen eingefügt (via `INSERT INTO`).
- Und wir haben die Daten wieder ausgelesen (via `SELECT FROM`) und sogar Daten von verschiedenen Tabellen zusammengeführt (via `LEFT JOIN` und `INNER JOIN`).
- Was wir noch nicht gemacht haben, ist Daten zu ändern und zu löschen.
- Auch dafür gibt es einfache SQL-Befehle.

Daten Ändern

Daten Ändern

- Unsere Fabrik hat einen neuen Anbieter für Schuhschachteln gefunden.

Daten Ändern

- Unsere Fabrik hat einen neuen Anbieter für Schuhschachteln gefunden.
- Die neuen Schachteln sind 5mm niedriger und daher billiger.

Daten Ändern

- Unsere Fabrik hat einen neuen Anbieter für Schuhschachteln gefunden.
- Die neuen Schachteln sind 5mm niedriger und daher billiger.
- Wir müssen also die Verpackungsdimensionen der Produkte, die in Schuhschachteln ausgeliefert werden, verringern.

Daten Ändern

- Unsere Fabrik hat einen neuen Anbieter für Schuhschachteln gefunden.
- Die neuen Schachteln sind 5mm niedriger und daher billiger.
- Wir müssen also die Verpackungsdimensionen der Produkte, die in Schuhschachteln ausgeliefert werden, verringern.
- Das geht mit einer `UPDATE <table> SET <fields>`-Anweisung[73].

Daten Ändern

- M

```
1  -- Syntax des UPDATE-Kommandos.
2
3  -- Wir wollen die Werte bestimmter Spalten der Tabelle 'tableName'
4  -- ändern, in den Zeilen, die die angegebene Bedingung erfüllen.
5  -- Wir spezifizieren den Tabellennamen direkt nach UPDATE.
6
7  -- Im 'SET'-Teil geben wir Ausdrücke an für die neuen Werte der Spalten,
8  -- die wir ändern wollen. Diese Ausdrücke können einfach Konstanten wie
9  -- '1' sein, aber auch komplexe mathematische Berechnungen basierend auf
10 -- den alten Werten enthalten, z.B. so etwas wie
11 -- 'column_1 = column_1 * 7'.
12
13 -- Wenn wir nicht alle Zeilen ändern wollen, dann können wir eine
14 -- 'WHERE'-Klausel angeben. Die Änderung wird dann nur auf die Zeilen
15 -- angewandt, für die 'WHERE'-Bedingung wahr/True ist.
16
17 -- Manche Systeme, wie PostgreSQL, erlauben uns die 'RETURNING'-Klausel
18 -- zu verwenden, die wie ein 'SELECT' wirkt, das *nach* dem Update und
19 -- nur auf die betreffenden Zeilen angewandt wird.
20 UPDATE tableName
21     SET column_1 = expression_1, column_2 = expression_2, ...
22     WHERE <condition to select rows that should be changed>
23     RETURNING *; -- optional/PostgreSQL: Daten geänderter Zeilen holen
```

Produkte in Schuhschachteln: Finden und Ändern

- Finden wir zuerst alle Produkte in Schuhschachteln.

```
/** Get all the products that ship in shoe boxes. */

-- We know what product comes in a shoe box by the box dimensions of
-- 350mm * 250mm * 130mm. If it has this size, it's a shoe box.
SELECT * FROM product  -- new height = old height - 5
    WHERE (width = 350) AND (height = 250) AND (depth = 130);
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
  ↪ =1 -ebf update_table_product_1.sql
 id |      name       | price  | weight | width | height | depth
----+-----------------+--------+--------+-------+--------+-------
  1 | Shoe, Size 36 | 150.99 |   1300 |   350 |    250 |   130
  2 | Shoe, Size 37 | 152.99 |   1325 |   350 |    250 |   130
  3 | Shoe, Size 38 | 154.99 |   1350 |   350 |    250 |   130
  4 | Shoe, Size 39 | 156.99 |   1375 |   350 |    250 |   130
  5 | Shoe, Size 40 | 158.99 |   1400 |   350 |    250 |   130
  6 | Shoe, Size 41 | 160.99 |   1425 |   350 |    250 |   130
  7 | Shoe, Size 42 | 162.99 |   1450 |   350 |    250 |   130
  8 | Shoe, Size 43 | 164.99 |   1475 |   350 |    250 |   130
  9 | Small Purse   | 100.00 |    500 |   350 |    250 |   130
(9 rows)

# psql 16.10 succeeded with exit code 0.
```

Produkte in Schuhschachteln: Finden und Ändern

- Jetzt nutzen wir die selbe `WHERE`-Klausel in unserer `UPDATE`-Anweisung[73].

```
/* We change entries in the table 'product' in our factory database. */

-- We want to reduce the height of all show boxes by 5mm.
-- We know what product comes in a shoe box by the box dimensions of
-- 350mm * 250mm * 130mm. If it has this size, it's a shoe box.
UPDATE product SET height = height - 5  -- new height = old height - 5
    WHERE (width = 350) AND (height = 250) AND (depth = 130);
-- Get the updated rows.
SELECT * FROM product
    WHERE (width = 350) AND (height = 245) AND (depth = 130);
-- Now the shoe boxe are 245mm high. Before they were 250mm high.
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪=1 -ebf update_table_product_2.sql
UPDATE 9
 id |      name       | price  | weight | width | height | depth
----+-----------------+--------+--------+-------+--------+-------
  1 | Shoe, Size 36 | 150.99 |   1300 |   350 |    245 |   130
  2 | Shoe, Size 37 | 152.99 |   1325 |   350 |    245 |   130
  3 | Shoe, Size 38 | 154.99 |   1350 |   350 |    245 |   130
  4 | Shoe, Size 39 | 156.99 |   1375 |   350 |    245 |   130
  5 | Shoe, Size 40 | 158.99 |   1400 |   350 |    245 |   130
  6 | Shoe, Size 41 | 160.99 |   1425 |   350 |    245 |   130
  7 | Shoe, Size 42 | 162.99 |   1450 |   350 |    245 |   130
  8 | Shoe, Size 43 | 164.99 |   1475 |   350 |    245 |   130
  9 | Small Purse   | 100.00 |    500 |   350 |    245 |   130
(9 rows)

# psql 16.10 succeeded with exit code 0.
```

Produkte in Schuhschachteln: Finden und Ändern

- Mit der PostgreSQL-spezifischen `RETURNING`-Anweisung geht es noch besser[74].

```
/* We change entries in the table 'product' in our factory database. */

-- We want to reduce the height of all show boxes by 5mm.
-- We know what product comes in a shoe box by the box dimensions of
-- 350mm * 250mm * 130mm. If it has this size, it's a shoe box.
UPDATE product SET height = height - 5  -- new height = old height - 5
    WHERE (width = 350) AND (height = 250) AND (depth = 130)
    RETURNING *;  -- Same as SELECT * FROM product WHERE ...;
-- Now the shoe boxe are 245mm high. Before they were 250mm high.
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf update_table_product_4.sql
 id |      name       | price  | weight | width | height | depth
----+-----------------+--------+--------+-------+--------+-------
  1 | Shoe, Size 36 | 150.99 |   1300 |   350 |    245 |   130
  2 | Shoe, Size 37 | 152.99 |   1325 |   350 |    245 |   130
  3 | Shoe, Size 38 | 154.99 |   1350 |   350 |    245 |   130
  4 | Shoe, Size 39 | 156.99 |   1375 |   350 |    245 |   130
  5 | Shoe, Size 40 | 158.99 |   1400 |   350 |    245 |   130
  6 | Shoe, Size 41 | 160.99 |   1425 |   350 |    245 |   130
  7 | Shoe, Size 42 | 162.99 |   1450 |   350 |    245 |   130
  8 | Shoe, Size 43 | 164.99 |   1475 |   350 |    245 |   130
  9 | Small Purse   | 100.00 |    500 |   350 |    245 |   130
(9 rows)

UPDATE 9
# psql 16.10 succeeded with exit code 0.
```

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir den `id`-Wert einer Zeile in der Tabelle `product` *ändern*, so dass die entsprechenden Bestellungen nicht mehr "passen"?

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir den `id`-Wert einer Zeile in der Tabelle `product` *ändern*, so dass die entsprechenden Bestellungen nicht mehr "passen"?
- Wir wissen: Herr Bibbo hat 12 Einheiten des Produkts "Shoe, Size 42" bestellt.

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir den `id`-Wert einer Zeile in der Tabelle `product` *ändern*, so dass die entsprechenden Bestellungen nicht mehr "passen"?
- Wir wissen: Herr Bibbo hat 12 Einheiten des Produkts "Shoe, Size 42" bestellt. "Shoe, Size 42" hat `id = 7`.
- Was passiert, wenn wir die `product` Tabelle so ändern, dass "Shoe, Size 42" dann `id = 20` hat?

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir den `id`-Wert einer Zeile in der Tabelle `product` *ändern*, so dass die entsprechenden Bestellungen nicht mehr "passen"?
- Wir wissen: Herr Bibbo hat 12 Einheiten des Produkts "Shoe, Size 42" bestellt. "Shoe, Size 42" hat `id = 7`.
- Was passiert, wenn wir die `product` Tabelle so ändern, dass "Shoe, Size 42" dann `id = 20` hat? Die Zeile mit der Bestellung in Tabelle `demand` würde dann ja nicht mehr auf ein existierendes Produkt verweisen...

Referentielle Integrität während Updates

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir den `id`-Wert einer Zeile in der Tabelle `product` *ändern*, so dass die entsprechenden Bestellungen nicht mehr "passen"?
- Wir wissen: Herr Bibbo hat 12 Einheiten des Produkts "Shoe, Size 42" bestellt. "Shoe, Size 42" hat `id = 7`.
- Was passiert, wenn wir die `product` Tabelle so ändern, dass "Shoe, Size 42" dann `id = 20` hat? Die Zeile mit der Bestellung in Tabelle `demand` würde dann ja nicht mehr auf ein existierendes Produkt verweisen...
- Wir fragen also: Können wir mit `UPDATE` die referentielle Integrität unserer Daten verletzen?

Referentielle Integrität während Updates

- Wir fragen also: Können wir mit `UPDATE` die referentielle Integrität unserer Daten verletzen?

```
1 /* Try to change the id of a product referenced elsewhere. */
2 -- Product id = 7 is used in one demand record.
3 -- It therefore cannot be changed to 20.
4 UPDATE product SET id = 20 WHERE id = 7 RETURNING *;
```

Referentielle Integrität während Updates

- Wir fragen also: Können wir mit `UPDATE` die referentielle Integrität unserer Daten verletzen?

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf update_table_product_error.sql
psql:factory/update_table_product_error.sql:4: ERROR:  update or delete
    ↪ on table "product" violates foreign key constraint "
    ↪ demand_product_fkey" on table "demand"
DETAIL:  Key (id)=(7) is still referenced from table "demand".
psql:factory/update_table_product_error.sql:4: STATEMENT:  /* Try to
    ↪ change the id of a product referenced elsewhere. */
-- Product id = 7 is used in one demand record.
-- It therefore cannot be changed to 20.
UPDATE product SET id = 20 WHERE id = 7 RETURNING *;
# psql 16.10 failed with exit code 3.
```

Referentielle Integrität während Updates

- Wir fragen also: Können wir mit `UPDATE` die referentielle Integrität unserer Daten verletzen?
- Wir können es offenbar nicht.

Referentielle Integrität während Updates

- Wir fragen also: Können wir mit `UPDATE` die referentielle Integrität unserer Daten verletzen?
- Wir können es offenbar nicht. Und das ist gut so!

Daten Löschen

Daten Löschen

- Wir stellen fest, dass wir noch nie ein Paar “Shoe, Size 36” verkauft haben!

Daten Löschen

- Wir stellen fest, dass wir noch nie ein Paar “Shoe, Size 36” verkauft haben!
- Daher wollen wir dieses Produkt aus unserem Sortiment streichen.

Daten Löschen

- Wir stellen fest, dass wir noch nie ein Paar “Shoe, Size 36” verkauft haben!
- Daher wollen wir dieses Produkt aus unserem Sortiment streichen.
- Um Daten zu löschen, verwenden wir das Kommando `DELETE FROM ... WHERE ...;`.[21]

Daten Löschen

- Wir stellen fest, dass wir noch nie ein Paar “Shoe, Size 36” verkauft haben!
- Daher wollen wir dieses Produkt aus unserem Sortiment streichen.
- Um Daten zu löschen, verwenden wir das Kommando `DELETE FROM ... WHERE ...;`.[21]

```
1 -- Lösche alle Zeilen in Tabelle 'tableName' auf die die
2 -- 'WHERE'-Bedingung zutrifft.
3 -- Eine DBMSe wie PostgreSQL bieten zudem die RETURNING-Klausel an, die
4 -- wie ein SELECT funktioniert, das nur auf die gelöschten Zeilen (bevor
5 -- sie gelöscht werden) angewendet wird.
6 DELETE FROM tableName WHERE <condition>
7 RETURNING column1, column2, ...;
```

Löschen des Produkts mit `id = 1`

- "Shoe, Size 36" hat `id = 1` in Tabelle `product`.

Löschen des Produkts mit `id = 1`

- "Shoe, Size 36" hat `id = 1` in Tabelle `product`.
- Löschen wir also diese Zeile!

Löschen des Produkts mit `id = 1`

- Löschen wir also diese Zeile!

```sql
/* We delete an entry from the table 'product' in our factory database. */

-- We got 11 products.
SELECT COUNT(*) as number_of_products from product;

-- Delete the 'Shoe, Size 36' ... nobody ever bought it.
DELETE FROM product WHERE id = 1  -- The id of 'Shoe, Size 36' is 1.
RETURNING id, name;  -- get the id and name of the deleted row.

-- We now we got only 10 products.
SELECT COUNT(*) as number_of_products from product;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP=1 -ebf delete_from_table_product.sql
 number_of_products
--------------------
                 11
(1 row)

 id |     name
----+--------------
  1 | Shoe, Size 36
(1 row)

DELETE 1
 number_of_products
--------------------
                 10
(1 row)

# psql 16.10 succeeded with exit code 0.
```

Löschen des Produkts mit `id = 1`

- Die PostgreSQL-spezifische `RETURNING`-Klausel gibt uns die gelöschten Daten zurück.

```sql
/* We delete an entry from the table 'product' in our factory database. */

-- We got 11 products.
SELECT COUNT(*) as number_of_products from product;

-- Delete the 'Shoe, Size 36' ... nobody ever bought it.
DELETE FROM product WHERE id = 1  -- The id of 'Shoe, Size 36' is 1.
RETURNING id, name;  -- get the id and name of the deleted row.

-- We now we got only 10 products.
SELECT COUNT(*) as number_of_products from product;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP=1 -ebf delete_from_table_product.sql
 number_of_products
--------------------
                 11
(1 row)

 id |     name
----+--------------
  1 | Shoe, Size 36
(1 row)

DELETE 1
 number_of_products
--------------------
                 10
(1 row)

# psql 16.10 succeeded with exit code 0.
```

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir eine Zeile aus der Tabelle `customer` löschen, auf die Zeilen in der Tabelle `demand` verweisen?

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir eine Zeile aus der Tabelle `customer` löschen, auf die Zeilen in der Tabelle `demand` verweisen?
- Wir wissen: Herr Bebbo hat vier Bestellungen aufgegeben.

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir eine Zeile aus der Tabelle `customer` löschen, auf die Zeilen in der Tabelle `demand` verweisen?
- Wir wissen: Herr Bebbo hat vier Bestellungen aufgegeben. Herr Bebbo hat `id = 2` Tabelle `customer`.

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir eine Zeile aus der Tabelle `customer` löschen, auf die Zeilen in der Tabelle `demand` verweisen?
- Wir wissen: Herr Bebbo hat vier Bestellungen aufgegeben. Herr Bebbo hat `id = 2` Tabelle `customer`.
- Was passiert, wenn wir diesen Eintrag aus der Tabelle `customer` löschen?

Referentielle Integrität während Löschen

- Wir hatten herausgefunden, dass wir keine Zeilen in die Tabelle `demand` eintragen können, für die es keine entsprechenden Zeilen in den Tabellen `customer` und `product` gibt.
- Aber was passiert, wenn wir eine Zeile aus der Tabelle `customer` löschen, auf die Zeilen in der Tabelle `demand` verweisen?
- Wir wissen: Herr Bebbo hat vier Bestellungen aufgegeben. Herr Bebbo hat `id = 2` Tabelle `customer`.
- Was passiert, wenn wir diesen Eintrag aus der Tabelle `customer` löschen? Die Zeilen mit seinen Bestellungen in Tabelle `demand` würde dann ja nicht mehr auf einen existierendes Kunden verweisen...

Referentielle Integrität während Löschen

- Wir fragen also: Können wir mit `DELETE` die referentielle Integrität unserer Daten verletzen?

Referentielle Integrität während Löschen

- Wir fragen also: Können wir mit `DELETE` die referentielle Integrität unserer Daten verletzen?

```
1 /* Try to delete customer Bebbo. */
2 -- Customer Bebbo has id = 2. However, he as a demand record referencing
3 -- him. So he cannot be deleted.
4 DELETE FROM customer WHERE id = 2  RETURNING id, name;
```

Referentielle Integrität während Löschen

- Wir fragen also: Können wir mit `DELETE` die referentielle Integrität unserer Daten verletzen?

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf delete_from_table_customer_error.sql
psql:factory/delete_from_table_customer_error.sql:4: ERROR:  update or
    ↪ delete on table "customer" violates foreign key constraint "
    ↪ demand_customer_fkey" on table "demand"
DETAIL:  Key (id)=(2) is still referenced from table "demand".
psql:factory/delete_from_table_customer_error.sql:4: STATEMENT:  /* Try
    ↪ to delete customer Bebbo. */
-- Customer Bebbo has id = 2. However, he as a demand record referencing
-- him. So he cannot be deleted.
DELETE FROM customer WHERE id = 2  RETURNING id, name;
# psql 16.10 failed with exit code 3.
```

Referentielle Integrität während Löschen

- Wir fragen also: Können wir mit `DELETE` die referentielle Integrität unserer Daten verletzen?
- Wir können es offenbar nicht.

Referentielle Integrität während Löschen

- Wir fragen also: Können wir mit `DELETE` die referentielle Integrität unserer Daten verletzen?
- Wir können es offenbar nicht. Und das ist gut so!

Zusammenfassung

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt: Während eigentlich alle wichtigen relationalen DBMSe die Sprache SQL unterstützen, gibt es trotzdem Unterschiede!

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt: Während eigentlich alle wichtigen relationalen DBMSe die Sprache SQL unterstützen, gibt es trotzdem Unterschiede!
- PostgreSQL bietet z. B. zusätzlich das Statement `RETURNING` für die `INSERT`, `UPDATE`, und `DELETE`-Befehle an, welches sehr nützlich ist und uns jeweils die betroffenen Zeilen der Tabelle zurückgibt.

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt: Während eigentlich alle wichtigen relationalen DBMSe die Sprache SQL unterstützen, gibt es trotzdem Unterschiede!
- PostgreSQL bietet z. B. zusätzlich das Statement `RETURNING` für die `INSERT`, `UPDATE`, und `DELETE`-Befehle an, welches sehr nützlich ist und uns jeweils die betroffenen Zeilen der Tabelle zurückgibt.
- Andere DBMSe unterstützen dieses Statement teilweise[22,38,58] oder gar nicht.

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt: Während eigentlich alle wichtigen relationalen DBMSe die Sprache SQL unterstützen, gibt es trotzdem Unterschiede!
- PostgreSQL bietet z. B. zusätzlich das Statement `RETURNING` für die `INSERT`, `UPDATE`, und `DELETE`-Befehle an, welches sehr nützlich ist und uns jeweils die betroffenen Zeilen der Tabelle zurückgibt.
- Andere DBMSe unterstützen dieses Statement teilweise[22,38,58] oder gar nicht.
- Selbst SQL-basierte DBMSe sind also nicht zwangsläufig kompatibel.

Zusammenfassung

- Jetzt können wir auch Daten in Tabellen verändern und löschen.
- Damit können wir nun aktiv mit Datenbanken arbeiten.
- Wir können nun alles machen, was wir auch mit Spreadsheets wie Microsoft Excel oder LibreOffice Calc machen können ... und viel mehr!
- Wir haben auch einen anderen wichtigen Aspekt gelernt: Während eigentlich alle wichtigen relationalen DBMSe die Sprache SQL unterstützen, gibt es trotzdem Unterschiede!
- PostgreSQL bietet z. B. zusätzlich das Statement `RETURNING` für die `INSERT`, `UPDATE`, und `DELETE`-Befehle an, welches sehr nützlich ist und uns jeweils die betroffenen Zeilen der Tabelle zurückgibt.
- Andere DBMSe unterstützen dieses Statement teilweise[22,38,58] oder gar nicht.
- Selbst SQL-basierte DBMSe sind also nicht zwangsläufig kompatibel.
- Sie können also nicht einfach gegeneinander ausgetauscht werden, und man muss gut überlegen, welches DBMS man für ein Projekt einsetzen will.

谢谢您们！

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 also[2] (siehe S. 61, 71).

[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 also[1] (siehe S. 61, 71).

[3] 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. 71, 73).

[4] 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. 71).

[5] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 71).

[6] 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. 73).

[7] 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. 70).

[8] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (siehe S. 71).

[9] 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. 72).

[10] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 71).

[11] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 70).

References II

[12] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 71, 72).

[13] 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. 72, 73).

[14] 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. 71).

[15] 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. 73).

[16] 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. 72).

[17] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 72).

[18] 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. 72).

[19] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 72).

[20] 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. 73).

[21] "DELETE". 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-delete.html (besucht am 2025-03-07) (siehe S. 30–33).

References III

[22] "DELETE". In: *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/docs/server/reference/sql-statements/data-manipulation/changing-deleting-data/delete (besucht am 2025-07-06) (siehe S. 50–59).

[23] 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. 72).

[24] 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. 71, 72).

[25] Steve Fanning, Vasudev Narayanan, "flywire", Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffers, 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. 71).

[26] 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. 72).

[27] Kevin P. Gaffney, Martin Prammer, Laurence C. Brasfield, D. Richard Hipp, Dan R. Kennedy und Jignesh M. Patel. "SQLite: Past, Present, and Future". *Proceedings of the VLDB Endowment (PVLDB)* 15(12):3535–3547, Aug. 2022. Irvine, CA, USA: Very Large Data Bases Endowment Inc. ISSN: 2150-8097. doi:10.14778/3554821.3554842. URL: https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf (besucht am 2025-01-12). All papers in this issue were presented at the *48th International Conference on Very Large Data Bases (VLDB 2022)*, 9 5-9, 2022, hybrid/Sydney, NSW, Australia (siehe S. 73).

[28] 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 (IFIPAICT). 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. 71).

References IV

[29] 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. 71).

[30] 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. 72).

[31] 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. 72).

[32] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 71).

[33] 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. 71, 73).

[34] D. Richard Hipp u. a. "Well-Known Users of SQLite". In: *SQLite*. Charlotte, NC, USA: Hipp, Wyrick & Company, Inc. (Hwaci), 2. Jan. 2023. URL: https://www.sqlite.org/famous.html (besucht am 2025-01-12) (siehe S. 73).

[35] 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. 72).

[36] 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. 72).

[37] *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 (besucht am 2025-01-08). Consists of several parts, see https://modern-sql.com/standard for information where to obtain them. (Siehe S. 72).

References V

[38] "`INSERT...RETURNING`". In: *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/docs/server/reference/sql-statements/data-manipulation/inserting-loading-data/insertreturning (besucht am 2025-07-06) (siehe S. 50–59).

[39] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 72).

[40] 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. 71).

[41] 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. 72).

[42] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: https://www.libreoffice.org (besucht am 2024-12-12) (siehe S. 71).

[43] 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. 70).

[44] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 72).

[45] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (besucht am 2025-04-24) (siehe S. 71).

[46] 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. 71).

[47] 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. 72).

References VI

[48] *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. 72).

[49] 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. 70).

[50] 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. 72).

[51] 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. 70).

[52] 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 Konovalov, 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. 70).

[53] *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), Feb. 2025. URL: https://www.postgresql.org/docs/17/index.html (besucht am 2025-02-25).

[54] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 72).

[55] 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. 70).

[56] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: 978-1-78398-154-0 (siehe S. 71).

[57] 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 (siehe S. 72).

References VII

[58] "`RETURNING`". In: *SQLite*. Charlotte, NC, USA: Hipp, Wyrick & Company, Inc. (Hwaci), 8. Mai 2024. URL: https://sqlite.org/lang_returning.html (besucht am 2025-04-24) (siehe S. 50–59).

[59] 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. 70).

[60] 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. 71).

[61] 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. 71).

[62] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 70).

[63] 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. 72).

[64] *SQLite*. Charlotte, NC, USA: Hipp, Wyrick & Company, Inc. (Hwaci), 2025. URL: https://sqlite.org (besucht am 2025-04-24) (siehe S. 73).

[65] "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. 72).

[66] 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. 67, 72).

[67] 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. Burgthann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of[66] (siehe S. 72).

References VIII

[68] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 72).

[69] 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. 72).

[70] 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. 71).

[71] 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. 70, 73).

[72] 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. 71).

[73] "`UPDATE`". 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-update.html (besucht am 2025-03-07) (siehe S. 11–14, 16, 17).

[74] ."What does standard SQL or any of the non-PostgreSQL SQLs do instead of `RETURNING`?" In: *Database Administrators*. Hrsg. von user210271. New York, NY, USA: Stack Exchange Inc., 7.–8. Juni 2020. URL: https://dba.stackexchange.com/questions/268664 (besucht am 2025-04-23) (siehe S. 16–18, 37).

[75] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 71).

[76] Thomas Weise (汤卫思). *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. 70–72).

[77] Thomas Weise (汤卫思). *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. 72).

References IX

[78] *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. 72).

[79] 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. 72).

[80] Marianne Winslett und Vanessa Braganholo. "Richard Hipp Speaks Out on SQLite". *ACM SIGMOD Record* 48(2):39–46, Juni 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0163-5808. doi:10.1145/3377330.3377338 (siehe S. 73).

[81] 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. 70).

[82] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 70).

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[11,49,82]. 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[7,43,51,55,59]. 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*[76].

DBA A *database administrator* is the person or group responsible for the effective use of database technology in an organization or enterprise.

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 DB[81].

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[62,71]. Learn more at https://git-scm.com.

GitHub is a website where software projects can be hosted and managed via the Git VCS[52,71]. Learn more at https://github.com.

IT information technology

Glossary (in English) II

LAMP Stack A system setup for web applications: Linux, Apache (a webserver), MySQL, and the server-side scripting language PHP[12,33].

LibreOffice is on open source office suite[28,42,60] which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Calc and LibreOffice Base. See[76] for more information and installation instructions.

LibreOffice Base is a DBMSes that can work on stand-alone files but also connect to other popular relational databases[25,60]. It is part of LibreOffice[28,42,60] and has functionality that is comparable to Microsoft Access[5,14,72].

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 spread sheet software[42,60], i.e., an alternative to Microsoft Excel. It is part of LibreOffice[28,42,60].

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows[3,32,61,70,75]. 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[1,2,4,24,45,56]. See https://mariadb.org for more information.

Microsoft Access is a DBMSes that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases[5,14,46,72]. 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[8,29,40]. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc[42,60].

Microsoft Office is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access[40]. LibreOffice is a free and open source alternative.

Microsoft Windows is a commercial proprietary operating system[10]. 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.

Glossary (in English) III

Microsoft Word is one of the leading text writing programs[23,48] and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.

MySQL An open source relational database management system[9,24,57,69,79]. 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 availabe in the internet, and which is usually developed cooperatively over the internet as well[35]. Typical examples are Python, Linux, Git, and PostgreSQL.

PostgreSQL An open source object-relational DBMS[26,50,54,69]. See https://postgresql.org for more information.

psql is the client program used to access the PostgreSQL DBMS server.

Python The Python programming language[36,41,44,77], i.e., what you will learn about in our book[77]. 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[16,30,31,63,68,76,78].

server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carring 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 servers[12] 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"[39].

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases[13,17–19,37,47,65–68]. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference[65].

Glossary (in English) IV

SQLite is an relational DBMS which runs as in-process library that works directly on files as opposed to the client-server architecture used by other common DBMSes. It is the most wide-spread SQL-based DB in use today, installed in nearly every smartphone, computer, web browser, television, and automobile[13,27,34,80]. Learn more at `https://sqlite.org`[64].

terminal A terminal is a text-based window where you can enter commands and execute them[3,15]. 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 ⊞+R, dann Schreiben von `cmd`, dann Druck auf ↵. Under Ubuntu Linux, Ctrl+Alt+T opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux[15,33]. 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 code[71]. 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[6,20]