合肥大學
HEFEI UNIVERSITY

Datenbanken

23. Fabrik-Datenbank: Aufräumen

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: 2025-10-10

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. Das Kommando `DROP`
3. Löschen der Beispiels
4. Zusammenfassung

Einleitung

Einleitung

- Wir kommen nun langsam an das Ende unserer kurzen Reise quer durch die Domäne der relationalen Datenbanken.

Einleitung

- Wir kommen nun langsam an das Ende unserer kurzen Reise quer durch die Domäne der relationalen Datenbanken.
- Wir haben einen groben Eindruck, was Datenbanken sind und wie wir sie verwenden können.

Einleitung

- Wir kommen nun langsam an das Ende unserer kurzen Reise quer durch die Domäne der relationalen Datenbanken.
- Wir haben einen groben Eindruck, was Datenbanken sind und wie wir sie verwenden können.
- Um unsere Reise, also unser Fabrik-Beispiel, zu beenden, wollen wir alle Dinge, die wir erstellt haben, auch wieder löschen.

Einleitung

- Wir kommen nun langsam an das Ende unserer kurzen Reise quer durch die Domäne der relationalen Datenbanken.
- Wir haben einen groben Eindruck, was Datenbanken sind und wie wir sie verwenden können.
- Um unsere Reise, also unser Fabrik-Beispiel, zu beenden, wollen wir alle Dinge, die wir erstellt haben, auch wieder löschen.
- Natürlichen machen wir das in der umgekehrten Reihenfolge, in der wir die Objekte erstellt haben.

Das Kommando DROP

Das Kommando DROP

- Die SQL Syntax um verschiedene Datenbank-Objekte zu löschen ist sehr einfach[22–25].

```
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

- Die SQL Syntax um verschiedene Datenbank-Objekte zu löschen ist sehr einfach[22–25].
- Sie schreiben `DROP` gefolgt vom Objekttyp gefolgt vom Objektname.

```sql
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

- Die SQL Syntax um verschiedene Datenbank-Objekte zu löschen ist sehr einfach[22–25].
- Sie schreiben `DROP` gefolgt vom Objekttyp gefolgt vom Objektname.
- Der Objekttyp könnte z. B. `TABLE`, `VIEW`, `USER` oder `DATABASE` sein.

```
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

- Die SQL Syntax um verschiedene Datenbank-Objekte zu löschen ist sehr einfach[22–25].
- Sie schreiben `DROP` gefolgt vom Objekttyp gefolgt vom Objektname.
- Der Objekttyp könnte z. B. `TABLE`, `VIEW`, `USER` oder `DATABASE` sein.
- Diese Kommandos werden fehlschlagen, wenn das Objekt noch in Benutzung ist, also wenn es noch von anderen Objekten referenziert wird.

```
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

- Die SQL Syntax um verschiedene Datenbank-Objekte zu löschen ist sehr einfach[22–25].
- Sie schreiben `DROP` gefolgt vom Objekttyp gefolgt vom Objektname.
- Der Objekttyp könnte z. B. `TABLE`, `VIEW`, `USER` oder `DATABASE` sein.
- Diese Kommandos werden fehlschlagen, wenn das Objekt noch in Benutzung ist, also wenn es noch von anderen Objekten referenziert wird.
- Sie können z. B. keine Tabelle löschen, die noch Teil eine Fremdschlüssels ist.

```sql
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

- Sie schreiben `DROP` gefolgt vom Objekttyp gefolgt vom Objektname.
- Der Objekttyp könnte z. B. `TABLE`, `VIEW`, `USER` oder `DATABASE` sein.
- Diese Kommandos werden fehlschlagen, wenn das Objekt noch in Benutzung ist, also wenn es noch von anderen Objekten referenziert wird.
- Sie können z. B. keine Tabelle löschen, die noch Teil eine Fremdschlüssels ist.
- Das Kommando wird auch fehlschlagen, wenn das Objekt nicht existiert.

```
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Das Kommando DROP

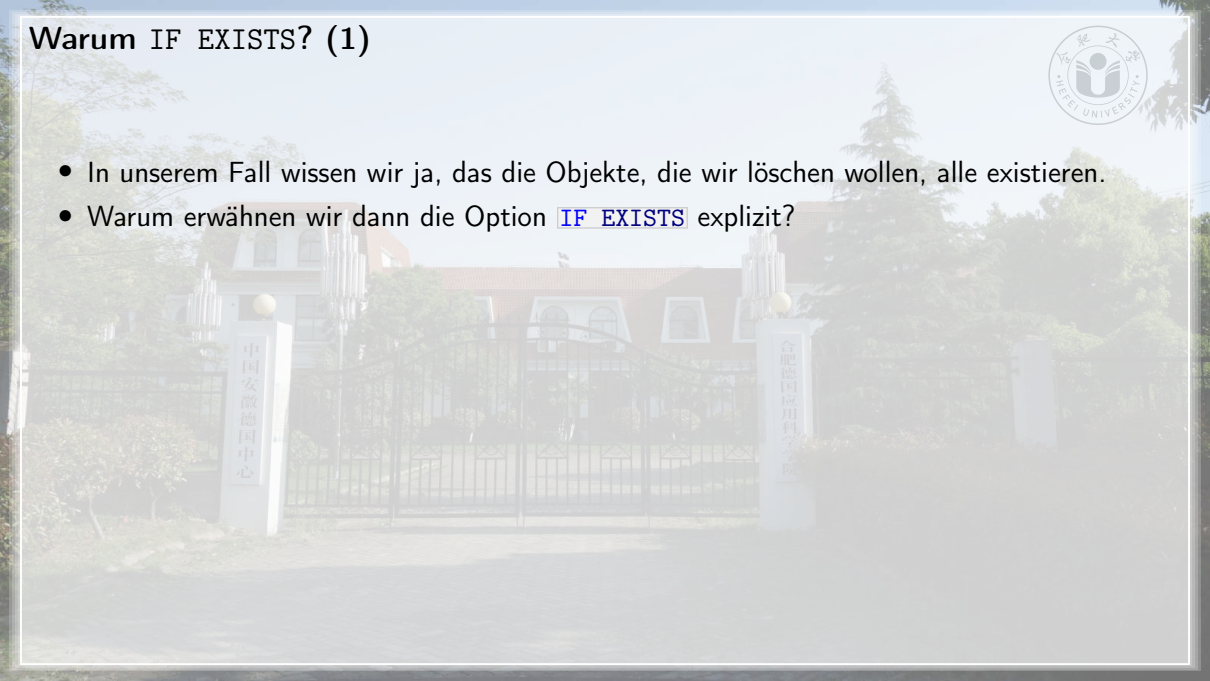
- Der Objekttyp könnte z. B. `TABLE`, `VIEW`, `USER` oder `DATABASE` sein.
- Diese Kommandos werden fehlschlagen, wenn das Objekt noch in Benutzung ist, also wenn es noch von anderen Objekten referenziert wird.
- Sie können z. B. keine Tabelle löschen, die noch Teil eine Fremdschlüssels ist.
- Das Kommando wird auch fehlschlagen, wenn das Objekt nicht existiert.
- Letzteres kann man umgehen, in dem mann `IF EXISTS` zwischen dem Objekttyp und dem Objektname einfügt.

```
/* The syntax for deleting elements: Using the 'DROP' Command.
 *
 * We can only delete elements that are not referenced by other
 * elements. Users can only be deleted if they do not own anything
 * anymore. Tables can only be deleted if they are not part of any
 * foreign key constraint, not used in any view, etc. And so on.
 */

-- Delete table "table_name": Fails if the table does not exist.
DROP TABLE table_name;
-- Delete table "table_name": It is OK if the table does not exist.
DROP TABLE IF EXISTS table_name;

-- Delete view "view_name": Fails if the view does not exist.
DROP VIEW view_name;
-- Delete view "view_name": It is OK if the view does not exist.
DROP VIEW IF EXISTS view_name;

-- Delete database "database_name": Fails if the DB does not exist.
DROP DATABASE database_name;
-- Delete database "database_name": It is OK if the DB does not exist.
DROP DATABASE IF EXISTS database_name;

-- Delete user "user_name": Fails if the user does not exist.
DROP USER user_name;
-- Delete user "user_name": It is OK if user does not exist.
DROP USER IF EXISTS user_name;
```

Warum `IF EXISTS`? (1)

- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.

Warum `IF EXISTS`? (1)



- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.
- Warum erwähnen wir dann die Option `IF EXISTS` explizit?

Warum `IF EXISTS`? (1)

- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.
- Warum erwähnen wir dann die Option `IF EXISTS` explizit?
- Weil sie einige interessante Verwendungen hat.

Warum `IF EXISTS`? (1)

- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.
- Warum erwähnen wir dann die Option `IF EXISTS` explizit?
- Weil sie einige interessante Verwendungen hat.
- Wenn wir z. B. ein Backup unserer gesamten DB machen wollen, dann könnten wir die ganze DB als ein großes SQL-Skript exportieren.Because it has some interesting use cases[67].

Warum `IF EXISTS`? (1)

- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.
- Warum erwähnen wir dann die Option `IF EXISTS` explizit?
- Weil sie einige interessante Verwendungen hat.
- Wenn wir z. B. ein Backup unserer gesamten DB machen wollen, dann könnten wir die ganze DB als ein großes SQL-Skript exportieren.Because it has some interesting use cases[67].
- Dieses Skript wäre im Grunde eine Aneinanderreihung aller unserer Listings, mit den `CREATE ...`- und `INSERT INTO...`-Kommandos und allem.

Warum `IF EXISTS`? (1)

- In unserem Fall wissen wir ja, das die Objekte, die wir löschen wollen, alle existieren.
- Warum erwähnen wir dann die Option `IF EXISTS` explizit?
- Weil sie einige interessante Verwendungen hat.
- Wenn wir z. B. ein Backup unserer gesamten DB machen wollen, dann könnten wir die ganze DB als ein großes SQL-Skript exportieren.Because it has some interesting use cases[67].
- Dieses Skript wäre im Grunde eine Aneinanderreihung aller unserer Listings, mit den `CREATE ...`- und `INSERT INTO...`-Kommandos und allem.
- Wenn wir dieses Skript ausführen, würden wir im Grunde die Datenbank wiederherstellen.Running this script would re-create the DB in exactly its present state.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.
- Wir können ja kein Objekt erstellen, das schon existiert.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.
- Wir können ja kein Objekt erstellen, das schon existiert.
- Oder vielleicht hatten wir ja einen Crash oder ein Benutzer hat ein fehlerhaftes SQL-Kommando ausgeführt und die Datenbank kaput gemacht.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.
- Wir können ja kein Objekt erstellen, das schon existiert.
- Oder vielleicht hatten wir ja einen Crash oder ein Benutzer hat ein fehlerhaftes SQL-Kommando ausgeführt und die Datenbank kaput gemacht.
- Dann existieren vielleicht noch ein paar Tabellen und Views und andere sind weg.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.
- Wir können ja kein Objekt erstellen, das schon existiert.
- Oder vielleicht hatten wir ja einen Crash oder ein Benutzer hat ein fehlerhaftes SQL-Kommando ausgeführt und die Datenbank kaput gemacht.
- Dann existieren vielleicht noch ein paar Tabellen und Views und andere sind weg.
- Eine Methode, um solche Skripte robuster zu machen, ist ein `DROP ... IF EXISTS` auszuführen, bevor die Tabellen und Sichten wieder erstellt werden.

Warum `IF EXISTS`? (2)

- Manchmal aber auch nicht.
- Wenn die Datenbank schon existiert, dann schlägt das SQL-Skript fehl.
- Wir können ja kein Objekt erstellen, das schon existiert.
- Oder vielleicht hatten wir ja einen Crash oder ein Benutzer hat ein fehlerhaftes SQL-Kommando ausgeführt und die Datenbank kaput gemacht.
- Dann existieren vielleicht noch ein paar Tabellen und Views und andere sind weg.
- Eine Methode, um solche Skripte robuster zu machen, ist ein `DROP ... IF EXISTS` auszuführen, bevor die Tabellen und Sichten wieder erstellt werden.
- Dann können wir die Datenbank oder Teile von ihr auch wiederherstellen, wenn sie noch Bruchstückhaft existiert.

Löschen der Beispiels

Löschen der Objekte in der Datenbank

- Nun wollen wir aber endlich alle Objekte wieder löschen, die wir erstellt haben.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Nun wollen wir aber endlich alle Objekte wieder löschen, die wir erstellt haben.
- Zuerst löschen wir die Sicht `sale`.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Nun wollen wir aber endlich alle Objekte wieder löschen, die wir erstellt haben.
- Zuerst löschen wir die Sicht `sale`.
- Das geht mit dem Kommando `DROP VIEW sale;`[25].

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Nun wollen wir aber endlich alle Objekte wieder löschen, die wir erstellt haben.
- Zuerst löschen wir die Sicht `sale`.
- Das geht mit dem Kommando `DROP VIEW sale;`[25].
- `DROP` ist das SQL-Kommando um Dinge zu löschen.

```sql
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Nun wollen wir aber endlich alle Objekte wieder löschen, die wir erstellt haben.
- Zuerst löschen wir die Sicht `sale`.
- Das geht mit dem Kommando `DROP VIEW sale;`[25].
- `DROP` ist das SQL-Kommando um Dinge zu löschen.
- `VIEW` wird spezifiziert, um klar zu machen das wir eine Sicht löschen wollen.

```sql
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Zuerst löschen wir die Sicht `sale`.
- Das geht mit dem Kommando `DROP VIEW sale;`[25].
- `DROP` ist das SQL-Kommando um Dinge zu löschen.
- `VIEW` wird spezifiziert, um klar zu machen das wir eine Sicht löschen wollen.
- Das verhindert, das wir aus Versehen etwas anderes löschen.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Das geht mit dem Kommando `DROP VIEW sale;`[25].
- `DROP` ist das SQL-Kommando um Dinge zu löschen.
- `VIEW` wird spezifiziert, um klar zu machen das wir eine Sicht löschen wollen.
- Das verhindert, das wir aus Versehen etwas anderes löschen.
- Wir geben den Namen der Sicht, die wir löschen wollen, an – nämlich `sale`.

```
1  /* Cleanup after the example: Delete all Tables and Views. */
2
3  -- This must be run inside the database, by user 'boss'.
4
5  -- Delete the views.
6  DROP VIEW IF EXISTS sale;
7
8  -- Delete the tables, in the inverse order of creation.
9  DROP TABLE IF EXISTS demand;
10 DROP TABLE IF EXISTS product;
11 DROP TABLE IF EXISTS customer;
```

```
1  $ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
       ↪ =1 -ebf cleanup_inside_database.sql
2  DROP VIEW
3  DROP TABLE
4  DROP TABLE
5  DROP TABLE
6  # psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- `DROP` ist das SQL-Kommando um Dinge zu löschen.
- `VIEW` wird spezifiziert, um klar zu machen das wir eine Sicht löschen wollen.
- Das verhindert, das wir aus Versehen etwas anderes löschen.
- Wir geben den Namen der Sicht, die wir löschen wollen, an – nämlich `sale`.
- Wir fügen das `IF EXISTS` zwischen dem Objekttyp und dem Namen der Sicht ein.

```
1  /* Cleanup after the example: Delete all Tables and Views. */
2
3  -- This must be run inside the database, by user 'boss'.
4
5  -- Delete the views.
6  DROP VIEW IF EXISTS sale;
7
8  -- Delete the tables, in the inverse order of creation.
9  DROP TABLE IF EXISTS demand;
10 DROP TABLE IF EXISTS product;
11 DROP TABLE IF EXISTS customer;
```

```
1  $ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
       ↪ =1 -ebf cleanup_inside_database.sql
2  DROP VIEW
3  DROP TABLE
4  DROP TABLE
5  DROP TABLE
6  # psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- `VIEW` wird spezifiziert, um klar zu machen das wir eine Sicht löschen wollen.
- Das verhindert, das wir aus Versehen etwas anderes löschen.
- Wir geben den Namen der Sicht, die wir löschen wollen, an – nämlich `sale`.
- Wir fügen das `IF EXISTS` zwischen dem Objekttyp und dem Namen der Sicht ein.
- Wenn die Sicht existiert, wird sie gelöscht.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Das verhindert, das wir aus Versehen etwas anderes löschen.
- Wir geben den Namen der Sicht, die wir löschen wollen, an – nämlich `sale`.
- Wir fügen das `IF EXISTS` zwischen dem Objekttyp und dem Namen der Sicht ein.
- Wenn die Sicht existiert, wird sie gelöscht.
- Wenn nicht, dann passiert nichts.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪=1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir geben den Namen der Sicht, die wir löschen wollen, an – nämlich `sale`.
- Wir fügen das `IF EXISTS` zwischen dem Objekttyp und dem Namen der Sicht ein.
- Wenn die Sicht existiert, wird sie gelöscht.
- Wenn nicht, dann passiert nichts.
- Ohne das `IF EXISTS` würde dieser Fall einen Fehler auslösen.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir fügen das `IF EXISTS` zwischen dem Objekttyp und dem Namen der Sicht ein.
- Wenn die Sicht existiert, wird sie gelöscht.
- Wenn nicht, dann passiert nichts.
- Ohne das `IF EXISTS` würde dieser Fall einen Fehler auslösen.
- Mit `IF EXISTS` brauchen wir uns nicht darum zu kümmern.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wenn die Sicht existiert, wird sie gelöscht.
- Wenn nicht, dann passiert nichts.
- Ohne das `IF EXISTS` würde dieser Fall einen Fehler auslösen.
- Mit `IF EXISTS` brauchen wir uns nicht darum zu kümmern.
- Wir können das Kommando zweimal ausführen, und es wäre auch OK.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wenn nicht, dann passiert nichts.
- Ohne das `IF EXISTS` würde dieser Fall einen Fehler auslösen.
- Mit `IF EXISTS` brauchen wir uns nicht darum zu kümmern.
- Wir können das Kommando zweimal ausführen, und es wäre auch OK.
- Wir benutzen die selbe Methode, um alle drei Tabellen zu löschen.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Ohne das `IF EXISTS` würde dieser Fall einen Fehler auslösen.
- Mit `IF EXISTS` brauchen wir uns nicht darum zu kümmern.
- Wir können das Kommando zweimal ausführen, und es wäre auch OK.
- Wir benutzen die selbe Methode, um alle drei Tabellen zu löschen.
- Wir führen aus `DROP TABLE IF EXISTS demand;`, `DROP TABLE IF EXISTS product;` und dann `DROP TABLE IF EXISTS customer;`[23].

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Mit `IF EXISTS` brauchen wir uns nicht darum zu kümmern.
- Wir können das Kommando zweimal ausführen, und es wäre auch OK.
- Wir benutzen die selbe Methode, um alle drei Tabellen zu löschen.
- Wir führen aus `DROP TABLE IF EXISTS demand;`, `DROP TABLE IF EXISTS product;` und dann `DROP TABLE IF EXISTS customer;`[23].
- Wir müssen Tabelle `demand` zuerst löschen und danach die Tabellen `customer` und `product`.

```
1  /* Cleanup after the example: Delete all Tables and Views. */
2
3  -- This must be run inside the database, by user 'boss'.
4
5  -- Delete the views.
6  DROP VIEW IF EXISTS sale;
7
8  -- Delete the tables, in the inverse order of creation.
9  DROP TABLE IF EXISTS demand;
10 DROP TABLE IF EXISTS product;
11 DROP TABLE IF EXISTS customer;
```

```
1 $ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
2 DROP VIEW
3 DROP TABLE
4 DROP TABLE
5 DROP TABLE
6 # psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir können das Kommando zweimal ausführen, und es wäre auch OK.
- Wir benutzen die selbe Methode, um alle drei Tabellen zu löschen.
- Wir führen aus `DROP TABLE IF EXISTS demand;`, `DROP TABLE IF EXISTS product;` und dann `DROP TABLE IF EXISTS customer;`[23].
- Wir müssen Tabelle `demand` zuerst löschen und danach die Tabellen `customer` und `product`.
- Das ist wegen der `REFERENCES`-Einschränkungen.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪=1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir benutzen die selbe Methode, um alle drei Tabellen zu löschen.
- Wir führen aus `DROP TABLE IF EXISTS demand;`, `DROP TABLE IF EXISTS product;` und dann `DROP TABLE IF EXISTS customer;`[23].
- Wir müssen Tabelle `demand` zuerst löschen und danach die Tabellen `customer` und `product`.
- Das ist wegen der `REFERENCES`-Einschränkungen.
- Wir können kein Objekt löschen, das noch referenziert wird.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪=1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir führen aus
  `DROP TABLE IF EXISTS demand;`,
  `DROP TABLE IF EXISTS product;`
  und dann
  `DROP TABLE IF EXISTS customer;`[23].
- Wir müssen Tabelle `demand` zuerst löschen und danach die Tabellen `customer` und `product`.
- Das ist wegen der `REFERENCES`-Einschränkungen.
- Wir können kein Objekt löschen, das noch referenziert wird.
- So lange diese Einschränkungen existieren, können die Tabellen `customer` und `product` nicht gelöscht werden.

```
1  /* Cleanup after the example: Delete all Tables and Views. */
2
3  -- This must be run inside the database, by user 'boss'.
4
5  -- Delete the views.
6  DROP VIEW IF EXISTS sale;
7
8  -- Delete the tables, in the inverse order of creation.
9  DROP TABLE IF EXISTS demand;
10 DROP TABLE IF EXISTS product;
11 DROP TABLE IF EXISTS customer;
```

```
1 $ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
2 DROP VIEW
3 DROP TABLE
4 DROP TABLE
5 DROP TABLE
6 # psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Wir müssen Tabelle `demand` zuerst löschen und danach die Tabellen `customer` und `product`.
- Das ist wegen der `REFERENCES`-Einschränkungen.
- Wir können kein Objekt löschen, das noch referenziert wird.
- So lange diese Einschränkungen existieren, können die Tabellen `customer` und `product` nicht gelöscht werden.
- In dem wir Tabelle `demand` zuerst löschen verschwinden auch die beiden Einschränkungen.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Objekte in der Datenbank

- Das ist wegen der `REFERENCES`-Einschränkungen.
- Wir können kein Objekt löschen, das noch referenziert wird.
- So lange diese Einschränkungen existieren, können die Tabellen `customer` und `product` nicht gelöscht werden.
- In dem wir Tabelle `demand` zuerst löschen verschwinden auch die beiden Einschränkungen.
- Die Tabellen `customer` und `product` werden dann nicht mehr referenziert und können gelöscht werden.

```
/* Cleanup after the example: Delete all Tables and Views. */

-- This must be run inside the database, by user 'boss'.

-- Delete the views.
DROP VIEW IF EXISTS sale;

-- Delete the tables, in the inverse order of creation.
DROP TABLE IF EXISTS demand;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS customer;
```

```
$ psql "postgres://boss:superboss123@localhost/factory" -v ON_ERROR_STOP
    ↪ =1 -ebf cleanup_inside_database.sql
DROP VIEW
DROP TABLE
DROP TABLE
DROP TABLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Alle diese „Löschungen“ wurden unter dem Benutzer `boss` und dessen Passwort `superboss123` gemacht.

```sql
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Alle diese „Löschungen“ wurden unter dem Benutzer `boss` und dessen Passwort `superboss123` gemacht.
- Löschen wir nun auch die Datenbank und diesen Benutzer.

```sql
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Alle diese „Löschungen“ wurden unter dem Benutzer `boss` und dessen Passwort `superboss123` gemacht.
- Löschen wir nun auch die Datenbank und diesen Benutzer.
- Wir löschen zuerst die Datenbank durch `DROP DATABASE IF EXISTS factory;`

```
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Alle diese „Löschungen“ wurden unter dem Benutzer `boss` und dessen Passwort `superboss123` gemacht.
- Löschen wir nun auch die Datenbank und diesen Benutzer.
- Wir löschen zuerst die Datenbank durch `DROP DATABASE IF EXISTS factory;`[22]
- Dann können wir auch den Benutzer mit `DROP USER IF EXISTS boss;` löschen[24].

```
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Alle diese „Löschungen“ wurden unter dem Benutzer `boss` und dessen Passwort `superboss123` gemacht.
- Löschen wir nun auch die Datenbank und diesen Benutzer.
- Wir löschen zuerst die Datenbank durch `DROP DATABASE IF EXISTS factory;`[22]
- Dann können wir auch den Benutzer mit `DROP USER IF EXISTS boss;` löschen[24].
- Beachten Sie, dass wir uns als DBA unter Benutzer `postgres` einloggen, um diese Schritte durchzuführen.

```
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Löschen der Datenbank und des Benutzers

- Löschen wir nun auch die Datenbank und diesen Benutzer.
- Wir löschen zuerst die Datenbank durch `DROP DATABASE IF EXISTS factory;`[22]
- Dann können wir auch den Benutzer mit `DROP USER IF EXISTS boss;` löschen[24].
- Beachten Sie, dass wir uns als DBA unter Benutzer `postgres` einloggen, um diese Schritte durchzuführen.
- Damit sind alle Überreste unseres Beispiels von DBMS Server verschwunden.

```
/* Cleanup after the example: Delete the Database and User */

-- This must be run by the administrative user 'postgres'.

-- If the database 'factory' exists, we delete it.
DROP DATABASE IF EXISTS factory;

-- If the user 'boss' already exists, we delete it.
-- We can only delete the user after all objects associated with it,
-- e.g., the databases, have been deleted.
DROP USER IF EXISTS boss;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ cleanup_database_and_user.sql
DROP DATABASE
DROP ROLE
# psql 16.10 succeeded with exit code 0.
```

Zusammenfassung

Zusammenfassung: Löschen

- Alle Objekte, die wir in einem DBMS erstellen können, können wir auch wieder löschen.

Zusammenfassung: Löschen

- Alle Objekte, die wir in einem DBMS erstellen können, können wir auch wieder löschen.
- Das Löschen von DB-Objekten kommt eher selten vor.

Zusammenfassung: Löschen

- Alle Objekte, die wir in einem DBMS erstellen können, können wir auch wieder löschen.
- Das Löschen von DB-Objekten kommt eher selten vor.
- In einigen Situationen, z. B. dem Einspielen von Backups, ist es aber sinnvoll.

Zusammenfassung: Beispiel

- Nun sind wir mit unserem Beispiel fertig.

Zusammenfassung: Beispiel

- Nun sind wir mit unserem Beispiel fertig.
- Was haben wir gelernt?

Zusammenfassung: Beispiel

- Nun sind wir mit unserem Beispiel fertig.
- Was haben wir gelernt?
- Zuersteinmal haben wir Hands-On Erfahrung mit einem der weltweit führenden DBMSe, PostgreSQL, gesammelt.

Zusammenfassung: Beispiel

- Nun sind wir mit unserem Beispiel fertig.
- Was haben wir gelernt?
- Zuersteinmal haben wir Hands-On Erfahrung mit einem der weltweit führenden DBMSe, PostgreSQL, gesammelt.
- Wir haben viele Kommandos in der Sprache SQL and das DBMS geschickt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.

Zusammenfassung: SQL-Kommandos

- Was für SQL-Kommandos haben wir denn ausprobiert?
- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar.

Zusammenfassung: SQL-Kommandos

- Nun, wir haben einen Benutzer (eine Rolle) und eine Datenbank erstellt.
- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.

Zusammenfassung: SQL-Kommandos

- Wir haben Tabellen in der DB erstelt.
- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.

Zusammenfassung: SQL-Kommandos

- Wir haben Daten in die Tabellen eingefügt und auch wieder ausgelesen.
- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.
- Wäre das eine effiziente oder elegante DB?

Zusammenfassung: SQL-Kommandos

- Wir haben die Daten mehrerer Tabellen mit Anfragen zusammengesetzt.
- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.
- Wäre das eine effiziente oder elegante DB?
- Mit ziemlicher Sicherheit nicht.

Zusammenfassung: SQL-Kommandos

- Wir haben eine Sicht erstellt und Anfragen oben auf die Sicht draufgesetzt.
- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.
- Wäre das eine effiziente oder elegante DB?
- Mit ziemlicher Sicherheit nicht.
- Wir haben ja noch gar nichts über Datenbankdesign gelernt.

Zusammenfassung: SQL-Kommandos

- Wir haben auch die Daten in einer Tabelle verändert.
- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.
- Wäre das eine effiziente oder elegante DB?
- Mit ziemlicher Sicherheit nicht.
- Wir haben ja noch gar nichts über Datenbankdesign gelernt.
- Aber wir haben jetzt bereits schon ein wenig Erfahrung.

Zusammenfassung: SQL-Kommandos

- Und am Schluss haben wir wieder alles gelöscht.
- Wir haben also einige der wichtigsten SQL-Kommandos kennengelernt.
- Na klar, wir haben nur mit ihnen herumgespielt.
- Wir sind nicht mal in die Nähe davon gekommen, ihr volles Verhalten, ihre Sonderfälle, oder ihre Performanz zu verstehen, geschweige denn, was sie eigentlich genau machen.
- Aber eins ist klar: Wenn uns jemand sagen würde, wir dass wir eine DB für eine bestimmte Anwendung erstellen sollen ... dann könnten wir das wahrscheinlich.
- Wir könnten etwas zusammenbauen, das in etwa funktioniert.
- Wäre das eine effiziente oder elegante DB?
- Mit ziemlicher Sicherheit nicht.
- Wir haben ja noch gar nichts über Datenbankdesign gelernt.
- Aber wir haben jetzt bereits schon ein wenig Erfahrung.
- Wir haben durchaus schon rohe und ungeschlieffene Fähigkeiten auf diesem Gebiet!

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.
- Was es nicht machen kann, können wir mit Python implementieren.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.
- Was es nicht machen kann, können wir mit Python implementieren.
- Mit Python könnten wir komplexe Datenvarbeitungsschritte durchführen oder auch Daten von Sensoren an die Datenbank schicken.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.
- Was es nicht machen kann, können wir mit Python implementieren.
- Mit Python könnten wir komplexe Datenvarbeitungsschritte durchführen oder auch Daten von Sensoren an die Datenbank schicken.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.
- Was es nicht machen kann, können wir mit Python implementieren.
- Mit Python könnten wir komplexe Datenvarbeitungsschritte durchführen oder auch Daten von Sensoren an die Datenbank schicken.
- Wir könnten ML oder AI-Modelle in Python basierend auf Daten aus einer Datenbank trainieren.

Zusammenfassung: Datenbankzugriff via Programmiersprache

- Wir verstehen im Prinzip auch, wie wir von einer Programmiersprache aus auf eine Datenbank zugreifen kann.
- Wir wissen in etwa, was ein DBMS mit SQL machen kann und was nicht.
- Damit können wir eigentlich schon ziemlich komplexe Applikationen bauen.
- Was ein DBMS machen kann, geht in die Datenbank.
- Was es nicht machen kann, können wir mit Python implementieren.
- Mit Python könnten wir komplexe Datenvarbeitungsschritte durchführen oder auch Daten von Sensoren an die Datenbank schicken.
- Wir könnten ML oder AI-Modelle in Python basierend auf Daten aus einer Datenbank trainieren.
- Dabei kann der Programmkode mit Hilfe einer Bibliothek wie psycopg mit der Datenbank kommunizieren.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.
- Dann verwenden wir doch lieber eine GUI wie LibreOffice Base als Frontend für die Datenbank.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.
- Dann verwenden wir doch lieber eine GUI wie LibreOffice Base als Frontend für die Datenbank.
- Zumindest das Eingeben, Anzeigen, und Ändern der Daten wird dann viel einfacher als über den psql-Client.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.
- Dann verwenden wir doch lieber eine GUI wie LibreOffice Base als Frontend für die Datenbank.
- Zumindest das Eingeben, Anzeigen, und Ändern der Daten wird dann viel einfacher als über den psql-Client.
- Während wir DBAs immer noch alle Features von PostgreSQL benutzen können, kann der Sekretär im Verkaufsdepartment neue Kunden und Bestellungen viel bequemer eingeben.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.
- Dann verwenden wir doch lieber eine GUI wie LibreOffice Base als Frontend für die Datenbank.
- Zumindest das Eingeben, Anzeigen, und Ändern der Daten wird dann viel einfacher als über den psql-Client.
- Während wir DBAs immer noch alle Features von PostgreSQL benutzen können, kann der Sekretär im Verkaufsdepartment neue Kunden und Bestellungen viel bequemer eingeben.
- Wir haben nämlich gelernt, wie man Formulare zum Eingeben von Daten in Tabellen verwenden kann, die in komplexen Beziehungen zu anderen Tabellen stehen.

Zusammenfassung: Datenbankzugriff via GUI

- Vielleicht arbeiten wir auch in einer kleinen Firma, in der nur drei, vier Leute mit der Datenbank arbeiten.
- Wir wollen wahrscheinlich trotzdem die Power von PostgreSQL nutzen.
- Aber ein separates Programm zu entwickeln, das mit den Daten arbeitet, könnte vielleicht zu viel Aufwandt sein.
- Dann verwenden wir doch lieber eine GUI wie LibreOffice Base als Frontend für die Datenbank.
- Zumindest das Eingeben, Anzeigen, und Ändern der Daten wird dann viel einfacher als über den psql-Client.
- Während wir DBAs immer noch alle Features von PostgreSQL benutzen können, kann der Sekretär im Verkaufsdepartment neue Kunden und Bestellungen viel bequemer eingeben.
- Wir haben nämlich gelernt, wie man Formulare zum Eingeben von Daten in Tabellen verwenden kann, die in komplexen Beziehungen zu anderen Tabellen stehen.
- Berichtet dagegen erlauben uns, Daten ansprechend auszulesen, darzustellen, und auszudrucken.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?
- Vielleicht wollen Sie ja Ihre Bücher oder Musiksammlung katalogisieren?

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?
- Vielleicht wollen Sie ja Ihre Bücher oder Musiksammlung katalogisieren?
- Vielleicht wollen Sie eine Datenbank mit Ihren Vorfahren anlegen?

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?
- Vielleicht wollen Sie ja Ihre Bücher oder Musiksammlung katalogisieren?
- Vielleicht wollen Sie eine Datenbank mit Ihren Vorfahren anlegen?
- Vielleicht wollen Sie eine Bibliographie von Literaturreferenzen anlegen?

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?
- Vielleicht wollen Sie ja Ihre Bücher oder Musiksammlung katalogisieren?
- Vielleicht wollen Sie eine Datenbank mit Ihren Vorfahren anlegen?
- Vielleicht wollen Sie eine Bibliographie von Literaturreferenzen anlegen?
- Es gibt nichts was Ihnen mehr beim Erlernen von Datenbanken helfen könnte, als so ein eigenes Projekt auszuführen.

Zusammenfassung

- An dieser Stelle in unserem Kurs haben Sie also mehrere Standardwerkzeuge und Abstraktionen kennengelernt, und zwar während Sie mit einer echten Datenbank auf einem professionellen DBMS gearbeitet haben.
- Sie haben noch keine tiefen theoretischen Kenntnisse.
- Aber dafür ein gutes Gefühl dafür, welches Werkzeug für welche Fragestellung geeignet seien könnte.
- Und ich hoffe, dass ich Ihre Neugier wecken konnte.
- Vielleicht haben Sie ja irgendein Problem, für das Sie selbst gerne eine Datenbank entwickeln wollen?
- Vielleicht wollen Sie ja Ihre Bücher oder Musiksammlung katalogisieren?
- Vielleicht wollen Sie eine Datenbank mit Ihren Vorfahren anlegen?
- Vielleicht wollen Sie eine Bibliographie von Literaturreferenzen anlegen?
- Es gibt nichts was Ihnen mehr beim Erlernen von Datenbanken helfen könnte, als so ein eigenes Projekt auszuführen.
- Das Zweitbeste ist vielleicht, den Rest des Kurses anzuschauen (oder irgendein Buch zu lesen).

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. 113, 123).

[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. 113, 123).

[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. 123, 125).

[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. 123).

[5] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 123).

[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. 125).

[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. 122).

[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. 124).

[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. 124).

[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. 124).

[11] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 122).

References II

[12] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 123, 125).

[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. 125).

[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. 123).

[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. 125).

[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. 125).

[17] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 125).

[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. 125).

[19] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 125).

[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. 125).

[21] 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. 124).

References III

[22] "DROP DATABASE". 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-dropdatabase.html (besucht am 2025-03-05) (siehe S. 10–14, 52–57).

[23] "DROP TABLE". 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-droptable.html (besucht am 2025-03-05) (siehe S. 10–14, 31–49).

[24] "DROP USER". 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-dropuser.html (besucht am 2025-03-05) (siehe S. 10–14, 52–57).

[25] "DROP VIEW". 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-dropview.html (besucht am 2025-03-05) (siehe S. 10–14, 31–37).

[26] 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. 123, 124).

[27] 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. 123).

[28] 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. 124).

References IV

[29] 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. 123).

[30] Michael Goodwin. *What is an API?* Armonk, NY, USA: International Business Machines Corporation (IBM), 9. Apr. 2024. URL: https://www.ibm.com/topics/api (besucht am 2024-12-12) (siehe S. 122).

[31] 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. 124).

[32] 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. 125).

[33] 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. 125).

[34] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 123).

[35] 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. 123, 125).

[36] 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. 124).

[37] 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. 124).

References V

[38] *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. 125).

[39] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 125).

[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. 124).

[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. 124).

[42] Marc-André Lemburg. *Python Database API Specification v2.0*. Python Enhancement Proposal (PEP) 249. Beaverton, OR, USA: Python Software Foundation (PSF), 12. Apr. 1999. URL: https://peps.python.org/pep-0249 (besucht am 2025-02-02) (siehe S. 124).

[43] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: https://www.libreoffice.org (besucht am 2024-12-12) (siehe S. 123, 124).

[44] 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. 122).

[45] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 124).

[46] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (besucht am 2025-04-24) (siehe S. 123).

[47] 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. 123).

References VI

[48] 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. 125).

[49] *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. 124).

[50] 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. 122).

[51] 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. 124).

[52] 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. 122).

[53] 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. 123).

[54] *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).

[55] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 124).

[56] 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. 122).

[57] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: 978-1-78398-154-0 (siehe S. 123).

References VII

[58] 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. 124).

[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. 122).

[60] Stuart J. Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach (AIMA)*. 4. Aufl. Hoboken, NJ, USA: Pearson Education, Inc. ISBN: 978-1-292-40113-3. URL: https://aima.cs.berkeley.edu (besucht am 2024-06-27) (siehe S. 122).

[61] 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. 123, 124).

[62] Shai Shalev-Shwartz und Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, England, UK: Cambridge University Press (CUP), Juli 2014. ISBN: 978-1-107-05713-5. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning (besucht am 2024-06-27) (siehe S. 124).

[63] 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. 123).

[64] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 122).

[65] 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. 125).

[66] "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. 125).

[67] "SQL Dump". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), 20. Feb. 2025. Kap. 25.1. URL: https://www.postgresql.org/docs/17/backup-dump.html (besucht am 2025-03-05) (siehe S. 17–22).

References VIII

[68] 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. 120, 125).

[69] 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[68] (siehe S. 125).

[70] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 125).

[71] 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. 124).

[72] 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. 123).

[73] 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. 122, 123, 125).

[74] 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. 123).

[75] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 123).

[76] Daniele „dvarrazzo" Varrazzo, Federico „fogzot" Di Gregorio und Jason „jerickso" Erickson. *Psycopg*. London, England, UK: The Psycopg Team, 2010–2023. URL: https://www.psycopg.org (besucht am 2025-02-02) (siehe S. 124).

[77] 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. 122, 123, 125).

References IX

[78] 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. 124).

[79] *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. 125).

[80] 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. 124).

[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. 122).

[82] 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). 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. 123, 124).

[83] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 122).

Glossary (in English) I

AI Artificial Intelligence, see, e.g.,[60]

API An *Application Programming Interface* is a set of rules or protocols that enables one software application or component to use or communicate with another[30].

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[11,50,83]. 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 DBMSes, such as psql.

client-server architecture is a system design where a central server receives requests from one or multiple clients[7,44,52,56,59]. 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*[77].

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[64,73]. Learn more at https://git-scm.com.

Glossary (in English) II

GitHub is a website where software projects can be hosted and managed via the Git VCS[53,73]. Learn more at https://github.com.

GUI graphical user interface

IT information technology

LAMP Stack A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP[12,35].

LibreOffice is on open source office suite[29,43,61] which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See[77] 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[27,61]. It is part of LibreOffice[29,43,61] and has functionality that is comparable to Microsoft Access[5,14,74].

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[43,61], i.e., an alternative to Microsoft Excel. It is part of LibreOffice[29,43,61].

LibreOffice Writer is a free and open source text writing program[82] and part of LibreOffice[29,43,61]. It is a good alternative to Microsoft Word.

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows[3,34,63,72,75]. 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[1,2,4,26,46,57]. 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[5,14,47,74]. It is part of Microsoft Office. A free and open source alternative to this commercial software is LibreOffice Base.

Glossary (in English) III

Microsoft Excel is a spreadsheet program that allows users to store, organize, manipulate, and calculate data in tabular structures[8,31,40]. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc[43,61].

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.

Microsoft Word is one of the leading text writing programs[21,49,82] and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.

ML Machine Learning, see, e.g.,[62]

MySQL An open source relational database management system[9,26,58,71,80]. 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[36]. Typical examples are Python, Linux, Git, and PostgreSQL.

PostgreSQL An open source object-relational DBMS[28,51,55,71]. See https://postgresql.org for more information.

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

psycopg or, more exactly, `psycopg 3`, is the most popular PostgreSQL adapter for Python, implementing the Python DB API 2.0 specification[42]. Learn more at https://www.psycopg.org[76].

Python The Python programming language[37,41,45,78], i.e., what you will learn about in our book[78]. Learn more at https://python.org.

Glossary (in English) IV

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[16,32,33,65,70,77,79].

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[13,17–19,38,48,66,68–70]. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference[66].

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[15,35]. 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[73]. 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]