



合肥大學
HEFEI UNIVERSITY



Datenbanken

8. Fabrik-Datenbank: Benutzer und Datenbank

Thomas Weise (汤卫思)
twaise@hfuu.edu.cn

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

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Databases



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

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



Outline



1. Einleitung
2. Erstellen eines Benutzerkontos
3. Erstellen einer Datenbank
4. Zusammenfassung





Einleitung



Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.
- Praktische Beispiele werden die Übung verlagert.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.
- Praktische Beispiele werden die Übung verlagert.
- Das ist OK.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.
- Praktische Beispiele werden die Übung verlagert.
- Das ist OK.
- Als ich Student war, habe ich alles, was ich über Datenbanken wissen musste aber anders gelernt.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.
- Praktische Beispiele werden die Übung verlagert.
- Das ist OK.
- Als ich Student war, habe ich alles, was ich über Datenbanken wissen musste aber anders gelernt.
- Durch Ausprobieren und Herumspielen.

Einleitung



- In vielen Kursen über *Datenbanken* wird das Gebiet in einzelnen Stücken diskutiert.
- Themen wie der Datenbank-Entwurfsprozess, Datenmodellierung, Entity-Relationship-Diagramme, σ -Algebra, SQL-Befehle und Normalisierung werden besprochen.
- Praktische Beispiele werden die Übung verlagert.
- Das ist OK.
- Als ich Student war, habe ich alles, was ich über Datenbanken wissen musste aber anders gelernt.
- Durch Ausprobieren und Herumspielen.
- Und deshalb machen wir das jetzt auch.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.
- Relationale Datenbanken speichern Tabellen haben wir gesagt.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.
- Relationale Datenbanken speichern Tabellen haben wir gesagt.
- Aber irgendwie ganz anders als ein vernünftiges Microsoft Excel-Spreadsheet.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.
- Relationale Datenbanken speichern Tabellen haben wir gesagt.
- Aber irgendwie ganz anders als ein vernünftiges Microsoft Excel-Spreadsheet.
- Wahrscheinlich ist Ihnen immer noch nicht klar, was Datenbanken eigentlich sind.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.
- Relationale Datenbanken speichern Tabellen haben wir gesagt.
- Aber irgendwie ganz anders als ein vernünftiges Microsoft Excel-Spreadsheet.
- Wahrscheinlich ist Ihnen immer noch nicht klar, was Datenbanken eigentlich sind.
- Und wie man die benutzt.

Was sind Datenbanken und wie benutzt man sie?



- Bisher sind *Datenbanken* für Sie nur irgendwelche abstrakten Dinge zum Speichern von Daten, auf die irgendwie über ein Klienten-Programm zugegriffen wird.
- Alles sehr eigenartig.
- Relationale Datenbanken speichern Tabellen haben wir gesagt.
- Aber irgendwie ganz anders als ein vernünftiges Microsoft Excel-Spreadsheet.
- Wahrscheinlich ist Ihnen immer noch nicht klar, was Datenbanken eigentlich sind.
- Und wie man die benutzt.
- Lassen Sie uns das ändern.

Fabrik-Datenbank



- Wir haben folgendes Fantasie-Szenario



Fabrik-Datenbank



- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.

- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.
 - Sie wurden als die IT-Abteilung dieser Fabrik angestellt.

- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.
 - Sie wurden als die IT-Abteilung dieser Fabrik angestellt.
 - An Ihrem ersten Arbeitstag kommt ihr Chef und sagt: *Mache bitte eine Datenbank, um alle unsere Produktvarianten, Kundendaten und Bestellungen zu speichern.*

- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.
 - Sie wurden als die IT-Abteilung dieser Fabrik angestellt.
 - An Ihrem ersten Arbeitstag kommt ihr Chef und sagt: *Mache bitte eine Datenbank, um alle unsere Produktvarianten, Kundendaten und Bestellungen zu speichern.*
- Wir benutzen dazu PostgreSQL, weil es ein sehr weit verbreitetes DBMS ist.

- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.
 - Sie wurden als die IT-Abteilung dieser Fabrik angestellt.
 - An Ihrem ersten Arbeitstag kommt ihr Chef und sagt: *Mache bitte eine Datenbank, um alle unsere Produktvarianten, Kundendaten und Bestellungen zu speichern.*
- Wir benutzen dazu PostgreSQL, weil es ein sehr weit verbreitetes DBMS ist. Im es war auf Rang 1 im "Stack Overflow 2024 Developer Survey"⁶⁸ und auf Rang 2 in⁵⁵.

- Wir haben folgendes Fantasie-Szenario
 - Sie arbeiten für eine kleine Fabrik, die Schuhe und Handtaschen herstellt.
 - Sie wurden als die IT-Abteilung dieser Fabrik angestellt.
 - An Ihrem ersten Arbeitstag kommt ihr Chef und sagt: *Mache bitte eine Datenbank, um alle unsere Produktvarianten, Kundendaten und Bestellungen zu speichern.*
- Wir benutzen dazu PostgreSQL, weil es ein sehr weit verbreitetes DBMS ist. Im es war auf Rang 1 im "Stack Overflow 2024 Developer Survey"⁶⁸ und auf Rang 2 in⁵⁵.

Nützliches Werkzeug

PostgreSQL^{29,53,58,72} ist ein fortschrittliches relationales DBMS. Es ist kostenfrei, Open Source und die Grundlage für alle Beispieldatenbanken in diesem Kurse.

Was wird passieren?

- In diesem Beispiel werden wird einiges lernen.



Was wird passieren?

- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.



Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.



Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigste Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?
 - Ja.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?
 - Ja. Wir verwenden die Sprache SQL, die von all diesen Datenbanken unterstützt wird. ^{15,20,21,23,39,50,67,69-71}

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?
 - Ja. Wir verwenden die Sprache SQL, die von all diesen Datenbanken unterstützt wird. ^{15,20,21,23,39,50,67,69-71}
- Natürlich können wir keines dieser Themen vollständig oder tiefgreifend bearbeiten.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?
 - Ja. Wir verwenden die Sprache SQL, die von all diesen Datenbanken unterstützt wird. ^{15,20,21,23,39,50,67,69-71}
- Natürlich können wir keines dieser Themen vollständig oder tiefgreifend bearbeiten.
- Aber wir können ein Gefühl für sie entwickeln, auf dem man aufbauen kann.

Was wird passieren?



- In diesem Beispiel werden wir einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.
 - DBs bestehen aus Tabellen, die ihrerseits aus Zeilen und Spalten bestehen.
 - Die Zeilen sind die in den Tabellen gespeicherten Datensätze.
 - Jeder Datensatz einer Tabelle hat die selben Attribute, die den Spalten entsprechen.
 - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern und wieder laden können.
- Wie geht das?
 - Lernen wir eine Methode, die nur bei PostgreSQL-Datenbanken funktioniert?
 - Oder werden die Dinge, die wir anschauen, auch bei MariaDB, Microsoft SQL Server, Oracle Database, MySQL, SQLite, usw. funktionieren?
 - Ja. Wir verwenden die Sprache SQL, die von all diesen Datenbanken unterstützt wird. ^{15,20,21,23,39,50,67,69-71}
- Natürlich können wir keines dieser Themen vollständig oder tiefgreifend bearbeiten.
- Aber wir können ein Gefühl für sie entwickeln, auf dem man aufbauen kann.
- Also: Los geht's.



Erstellen eines Benutzerkontos



Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.
- Er ist aber kein Datenbankadministrator,

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.
- Er ist aber kein Datenbankadministrator,
- Wir wollen ihm also Zugriff auf die neue Datenbank geben, aber lieber nicht die Kontrolle über das gesamte DBMS.

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.
- Er ist aber kein Datenbankadministrator,
- Wir wollen ihm also Zugriff auf die neue Datenbank geben, aber lieber nicht die Kontrolle über das gesamte DBMS.
- Deshalb ist der erste Schritt ein anderer

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.
- Er ist aber kein Datenbankadministrator,
- Wir wollen ihm also Zugriff auf die neue Datenbank geben, aber lieber nicht die Kontrolle über das gesamte DBMS.
- Deshalb ist der erste Schritt ein anderer: Wir erstellen ein neues Benutzerkonto (ein Rolle) speziell für die geplante Datenbank auf unserem DBMS.

Benutzerkonto Erstellen



- Der erste Schritt, diese Anforderung zu erfüllen, wäre es, eine neue Datenbank anzulegen.
- Wir haben ja PostgreSQL schon installiert, das wird sich also machen lassen.
- Unser Chef möchte vollen Zugriff auf diese neue Datenbank haben.
- Er ist aber kein Datenbankadministrator,
- Wir wollen ihm also Zugriff auf die neue Datenbank geben, aber lieber nicht die Kontrolle über das gesamte DBMS.
- Deshalb ist der erste Schritt ein anderer: Wir erstellen ein neues Benutzerkonto (ein Rolle) speziell für die geplante Datenbank auf unserem DBMS.
- Wenn unter diesem Benutzer etwas schief geht, oder ein Angreifer dessen Passwort irgendwie erhält, dann ist der Schaden zumindest auf diese eine Datenbank limitiert.

Erstellen eines Benutzerkontos: CREATE USER



- Das SQL-Kommando, um einen neues Benutzerkonto auf den Datenbankserver zu erstellen, lautet `CREATE USER`.

Erstellen eines Benutzerkontos: CREATE USER



- Das SQL-Kommando, um einen neues Benutzerkonto auf den Datenbankserver zu erstellen, lautet **CREATE USER**.
- Es hat die folgende Syntax:

```
1  --_Erstellen_eines_neuen_Benutzerkontos.
2  --
3  --_userName:_der_Name_des_Benutzerkontos,_das_wir_anlegen_wollen.
4  --_password:_dass_Passwort,_dass_wir_dem_Benutzer_zuweisen_wollen.
5  CREATE_USER_username_WITH_ENCRYPTED_PASSWORD_'password';
```

Erstellen eines Benutzerkontos: CREATE USER



- Das SQL-Kommando, um einen neues Benutzerkonto auf den Datenbankserver zu erstellen, lautet `CREATE USER`.
- Nun kennen wir das passende Kommando.

```
1  --_Erstellen_eines_neuen_Benutzerkontos.
2  --
3  --_userName:_der_Name_des_Benutzerkontos,_das_wir_anlegen_wollen.
4  --_password:_dass_Passwort,_dass_wir_dem_Benutzer_zuweisen_wollen.
5  CREATE_USER_username_WITH_ENCRYPTED_PASSWORD_'password';
```

Erstellen eines Benutzerkontos: CREATE USER



- Das SQL-Kommando, um einen neues Benutzerkonto auf den Datenbankserver zu erstellen, lautet `CREATE USER`.
- Nun kennen wir das passende Kommando...
- ...aber wie können wir dieses Kommando eingeben?

```
1  --_Erstellen_eines_neuen_Benutzerkontos.
2  --
3  --_userName:_der_Name_des_Benutzerkontos,_das_wir_anlegen_wollen.
4  --_password:_dass_Passwort,_dass_wir_dem_Benutzer_zuweisen_wollen.
5  CREATE_USER_username_WITH_ENCRYPTED_PASSWORD_'password';
```

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.
- Nehmen Sie an, dass das Passwort des Datenbankadministratorbenutzers `textilpostgres` auf `XXX` gesetzt ist.

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.
- Nehmen Sie an, dass das Passwort des Datenbankadministratorbenutzers textilpostgres auf `xxx` gesetzt ist. Benutzen Sie niemals etwas wie `xxx` als Passwort.

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.
- Nehmen Sie an, dass das Passwort des Datenbankadministratorbenutzers textilpostgres auf `XXX` gesetzt ist. Benutzen Sie niemals etwas wie `XXX` als Passwort.
- Wir öffnen ein Terminal (unter Ubuntu Linux via `Ctrl` + `Alt` + `T`; unter Microsoft Windows durch Druck auf `Windows` + `R`, dann Schreiben von `cmd`, dann Druck auf `↵`).

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.
- Nehmen Sie an, dass das Passwort des Datenbankadministratorbenutzers textilpostgres auf `XXX` gesetzt ist. Benutzen Sie niemals etwas wie `XXX` als Passwort.
- Wir öffnen ein Terminal (unter Ubuntu Linux via `Ctrl` + `Alt` + `T`; unter Microsoft Windows durch Druck auf `Windows` + `R`, dann Schreiben von `cmd`, dann Druck auf `↵`).
- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.

Verbindung to PostgreSQL via psql



- Wir sitzen an der Tastatur vor dem Computer, auf dem der Datenbank-Server läuft.
- Nehmen Sie an, dass das Passwort des Datenbankadministratorbenutzers `textilpostgres` auf `XXX` gesetzt ist. Benutzen Sie niemals etwas wie `XXX` als Passwort.
- Wir öffnen ein Terminal (unter Ubuntu Linux via `Ctrl` + `Alt` + `T`; unter Microsoft Windows durch Druck auf `Windows` + `R`, dann Schreiben von `cmd`, dann Druck auf `↵`).
- Wir wollen nun den Klienten `psql` mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.

Nützliches Werkzeug

`psql` ist ein textbasiertes Konsolenprogramm mit dem man sich auf den PostgreSQL-Server verbinden kann. Von der `psql`-Konsole können wir SQL-Befehle an den PostgreSQL-Server schicken und dessen Ausgaben empfangen.

Verbinden to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.

Verbindung to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`

Verbindung to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
 - `postgres://` identifiziert den parameter als Verbindungs-URI.
 - Das zweite „`postgres`“ ist der Benutzername.
 - Der Doppelpunkt „`:`“ trennt den Benutzername von dem Password `XXX`.

Verbindung to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
 - `postgres://` identifiziert den parameter als Verbindungs-URI.
 - Das zweite „`postgres`“ ist der Benutzername.
 - Der Doppelpunkt „`:`“ trennt den Benutzername von dem Passwort `XXX`. Benutzen Sie niemals etwas wie `XXX` als Passwort.

Verbindung to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
 - `postgres://` identifiziert den parameter als Verbindungs-URI.
 - Das zweite „`postgres`“ ist der Benutzername.
 - Der Doppelpunkt „`:`“ trennt den Benutzername von dem Passwort `XXX`. Benutzen Sie niemals etwas wie `XXX` als Passwort.
 - Nach dem „`@`“ kommt die Netzwerkadresse, auf der der PostgreSQL-Server läuft.

Verbindung to PostgreSQL via psql



- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI⁵⁷ starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
 - `postgres://` identifiziert den parameter als Verbindungs-URI.
 - Das zweite „`postgres`“ ist der Benutzername.
 - Der Doppelpunkt „`:`“ trennt den Benutzername von dem Passwort `XXX`. Benutzen Sie niemals etwas wie `XXX` als Passwort.
 - Nach dem „`@`“ kommt die Netzwerkadresse, auf der der PostgreSQL-Server läuft. `localhost` steht für den aktuellen Computer.

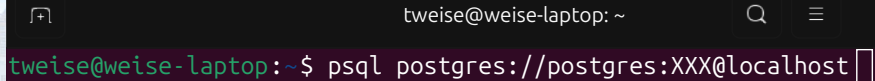
Erstellen des Benutzers via Terminal

- Wir loggen uns jetzt in unseren PostgreSQL-Server ein und erstellen das neue Benutzerkonto Schritt-für-Schritt.



Erstellen des Benutzers via Terminal

- Wir starten psql mit der passenden Verbindungs-URI.



A terminal window with a dark background. The title bar shows 'tweise@weise-laptop: ~' and standard window controls. The command prompt is 'tweise@weise-laptop:~\$' and the command entered is 'psql postgres://postgres:XXX@localhost'. A cursor is visible at the end of the command.

```
tweise@weise-laptop:~$ psql postgres://postgres:XXX@localhost
```

Erstellen des Benutzers via Terminal

- psql läuft und ist mit dem PostgreSQL DBMS verbunden.



```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ psql postgres://postgres:XXX@localhost  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, comp  
ression: off)  
Type "help" for help.  
  
postgres=#
```

Erstellen des Benutzers via Terminal



- Wir erstellen das Benutzerkonto `boss` mit dem (verschlüsselt zu speichernden) Passwort `superboss123` mit Hilfe des SQL-Befehls `CREATE USER`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ psql postgres://postgres:XXX@localhost  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, comp  
ression: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';
```

Erstellen des Benutzers via Terminal



- Der Befehl war erfolgreich. psql zeigt dies durch Ausgabe von `CREATE ROLE` an.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ psql postgres://postgres:XXX@localhost  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, comp  
ression: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=#
```

Erstellen des Benutzers via Terminal



- Zurück zur psql-Session: Der Befehl war erfolgreich. psql zeigt dies durch Ausgabe von `CREATE ROLE` an.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ psql postgres://postgres:XXX@localhost  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, comp  
ression: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=#
```

Erstellen des Benutzers via Terminal



- Wir wollen uns die Liste aller Benutzerkonten ausgeben lassen. Dazu wenden wir den SQL-Befehl `SELECT ... FROM` auf die entsprechende Systemtabelle an.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ psql postgres://postgres:XXX@localhost  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, comp  
ression: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=# SELECT username FROM pg_catalog.pg_user; 
```

Erstellen des Benutzers via Terminal



- Es gibt zwei Benutzerkonten: Das Systemadministratorkonto `postgres` und den neuen Benutzer `boss`.

```
tweise@weise-laptop: ~  
session: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=# SELECT username FROM pg_catalog.pg_user;  
username  
-----  
postgres  
boss  
(2rows)  
  
postgres=#
```

Erstellen des Benutzers via Terminal



- Wir beenden die psql-Session durch `\q` und `↵`.

```
tweise@weise-laptop: ~  
session: off)  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=# SELECT username FROM pg_catalog.pg_user;  
username  
-----  
postgres  
boss  
(2rows)  
  
postgres=# \q
```

Erstellen des Benutzers via Terminal



- Wir sind zurück im normalen Terminal.

```
tweise@weise-laptop: ~  
Type "help" for help.  
  
postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';  
CREATE ROLE  
postgres=# SELECT username FROM pg_catalog.pg_user;  
username  
-----  
postgres  
boss  
(2rows)  
  
postgres=# \q  
tweise@weise-laptop :~$
```

SELECT FROM

- Und nun kennen wir das zweite SQL-Kommando: SELECT...FROM.



SELECT FROM



- Und nun kennen wir das zweite SQL-Kommando: **SELECT...FROM**.
- Es hat die folgende Syntax:

```
1  -- Werte bestimmter Spalten einer Tabelle auslesen.
2  --
3  -- Dieses Kommando gibt die Werte der Spalte 'columnName' der Tabelle
4  -- 'tableName' für alle Zeilen zurück.
5  -- Das Ergebnis sieht wie eine (temporäre) Tabelle aus, die nur die
6  -- aus Tabelle 'tableName' ausgewählte(n) Spalte(n) hat.
7  --
8  -- columnName: Name der Spalte, die wir auswählen wollen
9  -- tableName: Name der Tabelle, zu der die Spalte gehört.
10 SELECT columnName FROM tableName;
11
12 -- Sie können auch mehrere Spalten auf einmal auswählen.
13 -- Dieser Befehl selektiert N Spalten aus der Tabelle.
14 -- Die Namen der Spalten sind durch Kommas (",") getrennt.
15 SELECT columnName1, columnName2, ..., columnNameN FROM tableName;
16
17 -- Diese Anweisung gibt alle Werte aus allen Spalten der Tabelle zurück.
18 SELECT * FROM tableName;
```

Erstellen des Benutzers via Script



- Alternativ können wir auch die SQL-Kommandos einfach alle in eine Text-Datei schreiben.

Erstellen des Benutzers via Script



- Alternativ können wir auch die SQL-Kommandos einfach alle in eine Text-Datei schreiben.
- Diese können wir dann mit psql an den PostgreSQL-Server schicken und in einem Schritt ausführen.

Erstellen des Benutzers via Script



- Alternativ können wir auch die SQL-Kommandos einfach alle in eine Text-Datei schreiben.
- Diese können wir dann mit psql an den PostgreSQL-Server schicken und in einem Schritt ausführen.
- Das ist viel angenehmer, besonders wenn wir mehrere komplexe Befehle ausführen wollen.

Erstellen des Benutzers via Script



- Alternativ können wir auch die SQL-Kommandos einfach alle in eine Text-Datei schreiben.
- Diese können wir dann mit psql an den PostgreSQL-Server schicken und in einem Schritt ausführen.
- Das ist viel angenehmer, besonders wenn wir mehrere komplexe Befehle ausführen wollen.

```
psql "postgres://user:password@host:port/database" -v ON_ERROR_STOP=1 -ebf script.sql
```

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm `psql`

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Servers, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten
- `-v ON_ERROR_STOP=1`: Das Programm soll bei Fehlern sofort abbrechen.

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten
- `-v ON_ERROR_STOP=1`: Das Programm soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf stdout⁴⁰.

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten
- `-v ON_ERROR_STOP=1`: Das Programm soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf stdout⁴⁰.
- `-b`: Drucke gescheiterte Befehle auf stderr.

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten
- `-v ON_ERROR_STOP=1`: Das Programm soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf stdout⁴⁰.
- `-b`: Drucke gescheiterte Befehle auf stderr.
- `-f filename`: Lese Kommandos aus Datei `filename`.

Erstellen des Benutzers via Script



```
psql "postgres://user:password@host:port/database"-v ON_ERROR_STOP=1 -ebf script.sql
```

- `psql`: Das Programm psql
- Verbindungs-URI, in Anführungszeichen.
 - `postgres://` zeigt an, dass das eine PostgreSQL Verbindungs-URI ist
 - `user:password`: Benutzername und Passwort
 - `host`: Netzwerkadresse des PostgreSQL-Server, localhost = unser Computer
 - `port`: der Port, an dem der Server auf Verbindungen wartet, weglassen für Default (5432)
 - `database`: Name der Datenbank, weglassen wenn wir direkt auf dem System arbeiten
- `-v ON_ERROR_STOP=1`: Das Programm soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf stdout⁴⁰.
- `-b`: Drucke gescheiterte Befehle auf stderr.
- `-f filename`: Lese Kommandos aus Datei `filename`.
- `-ebf filename`: Das selbe wie `-e -b -f filename`.

Erstellen des Benutzers via Script



```
1  /* In this example, we create a new user for our database. */
2
3  -- On PostgreSQL, there is a table `pg_catalog.pg_user` with all users.
4  -- We print the column `username` with the user names.
5  SELECT username FROM pg_catalog.pg_user;
6
7  -- Create the user 'boss'.
8  -- He will be the owner of the database that we will create.
9  CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';
10
11 -- Now there is a new user: 'boss'.
12 SELECT username FROM pg_catalog.pg_user;
```

```
1  $ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
   ↪ create_user.sql
2  username
3  -----
4  postgres
5  (1 row)
6
7  CREATE ROLE
8  username
9  -----
10 postgres
11 boss
12 (2 rows)
13
14 # psql 16.11 succeeded with exit code 0.
```

Erstellen des Benutzers via Script



```
1  /* In this example, we create a new user for our database. */
2
3  -- On PostgreSQL, there is a table `pg_catalog.pg_user` with all users.
4  -- We print the column `username` with the user names.
5  SELECT username FROM pg_catalog.pg_user;
6
7  -- Create the user 'boss'.
8  -- He will be the owner of the database that we will create.
9  CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';
10
11 -- Now there is a new user: 'boss'.
12 SELECT username FROM pg_catalog.pg_user;
```

Erstellen des Benutzers via Script



```
1 $ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
   ↪ create_user.sql
2
3 username
4 -----
5 postgres
6 (1 row)
7
8 CREATE ROLE
9 username
10 -----
11 postgres
12 boss
13 (2 rows)
14
15 # psql 16.11 succeeded with exit code 0.
```

Ein Wort zu Stilrichtlinien

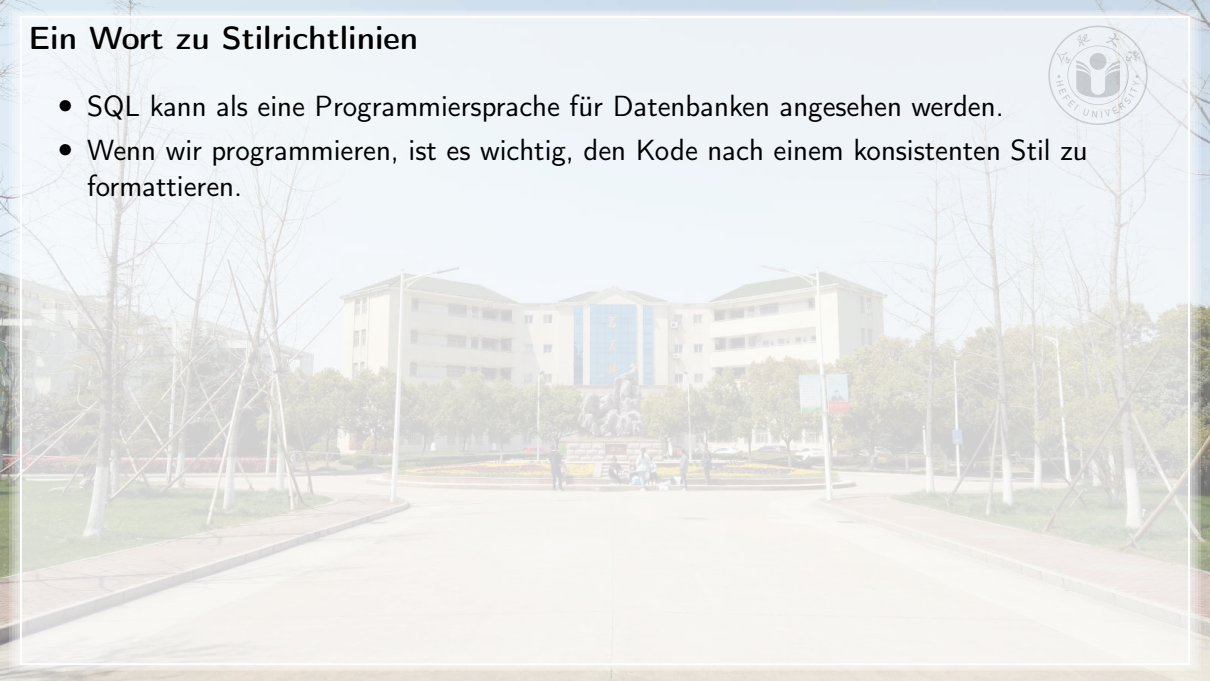
- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.



Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.



Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Code und Skripte nach nach einem konsistenten Stil zu formatieren, eine konsistentes Namensschema zu verwenden und den generell akzeptierten Best Practices und Normen für diese Programmiersprache zu folgen.

Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Code und Skripte nach nach einem konsistenten Stil zu formatieren, eine konsistentes Namensschema zu verwenden und den generell akzeptierten Best Practices und Normen für diese Programmiersprache zu folgen.

- Nur so kann die Zusammenarbeit in einem Team gut klappen.

Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Code und Skripte nach nach einem konsistenten Stil zu formatieren, eine konsistentes Namensschema zu verwenden und den generell akzeptierten Best Practices und Normen für diese Programmiersprache zu folgen.

- Nur so kann die Zusammenarbeit in einem Team gut klappen.
- Für viele Programmiersprachen gibt es klare Styleguides (Stilrichtlinien).

Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Code und Skripte nach nach einem konsistenten Stil zu formatieren, eine konsistentes Namensschema zu verwenden und den generell akzeptierten Best Practices und Normen für diese Programmiersprache zu folgen.

- Nur so kann die Zusammenarbeit in einem Team gut klappen.
- Für viele Programmiersprachen gibt es klare Styleguides (Stilrichtlinien).
- Für SQL gibt es so etwas nicht, bzw. gibt es viele sich widersprechende Styleguides.

Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Code und Skripte nach nach einem konsistenten Stil zu formatieren, eine konsistentes Namensschema zu verwenden und den generell akzeptierten Best Practices und Normen für diese Programmiersprache zu folgen.

- Nur so kann die Zusammenarbeit in einem Team gut klappen.
- Für viele Programmiersprachen gibt es klare Styleguides (Stilrichtlinien).
- Für SQL gibt es so etwas nicht, bzw. gibt es viele sich widersprechende Styleguides.
- Trotzdem werden wir versuchen, einige genrelle Regeln aufzustellen und zu befolgen.

Ein Wort zu Stilrichtlinien



- SQL kann als eine Programmiersprache für Datenbanken angesehen werden.
- Wenn wir programmieren, ist es wichtig, den Code nach einem konsistenten Stil zu formatieren.
- Nur so kann die Zusammenarbeit in einem Team gut klappen.
- Für viele Programmiersprachen gibt es klare Styleguides (Stilrichtlinien).
- Für SQL gibt es so etwas nicht, bzw. gibt es viele sich widersprechende Styleguides.
- Trotzdem werden wir versuchen, einige genrelle Regeln aufzustellen und zu befolgen.

Gute Praxis

SQL-Schlüsselworte (wie `SELECT`, `CREATE`, ...) sollten immer durchgängig mit Großbuchstaben geschrieben werden²².



Erstellen einer Datenbank



Erstellen einer Datenbank: CREATE DATABASE



- Nachdem wir den Benutzer „boss“ erstellt haben, können wir die Datenbank „factory“ für diesen Benutzer anlegen.



Erstellen einer Datenbank: CREATE DATABASE



- Nachdem wir den Benutzer „boss“ erstellt haben, können wir die Datenbank „factory“ für diesen Benutzer anlegen.
- Das machen wir genauso wie vorhin das Erstellen des Benutzers.



Erstellen einer Datenbank: CREATE DATABASE



- Nachdem wir den Benutzer „boss“ erstellt haben, können wir die Datenbank „factory“ für diesen Benutzer anlegen.
- Das machen wir genauso wie vorhin das Erstellen des Benutzers.
- Diesmal lautet das Kommando **CREATE DATABASE**:

```
1  --_Erstellen_einer_Datenbank.
2  --
3  --_'databaseName':_der_Name_für_die_neue_Datenbank.
4  --_'userName':_der_Benutzer,_dem_die_neue_Datenbank_"gehören"_soll.
5  CREATE_DATABASE_databaseName_OWNER_userName;
```

Erstellen der Datenbank via Terminal

- Wir loggen uns jetzt in unseren PostgreSQL-Server ein und erstellen die neue Datenbank.



Erstellen der Datenbank via Terminal



- Wir starten psql mit der passenden Verbindungs-URI. Wir benutzen immer noch das Systemadministratorenkonto `postgres`.

```
tweise@weise-laptop: ~$ psql "postgres://postgres:XXX@localhost"
```

Erstellen der Datenbank via Terminal



- psql läuft und ist mit dem PostgreSQL DBMS verbunden.

```
tweise@weise-laptop: ~  
twaise@weise-laptop :~$ psql "postgres://postgres:XXX@localhost"  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression:  
off)  
Type "help" for help.  
  
postgres=#
```

Erstellen der Datenbank via Terminal



- Wir erstellen die Datenbank `factory` für den Benutzer `boss` mit Hilfe des SQL-Befehls `CREATE DATABASE ... OWNER`.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ psql "postgres://postgres:XXX@localhost"  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression:  
off)  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss; 
```

Erstellen der Datenbank via Terminal



- Der Befehl war erfolgreich. psql zeigt dies durch Ausgabe von `CREATE DATABASE` an.

```
tweise@weise-laptop: ~  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=#
```

Erstellen der Datenbank via Terminal



- Wir können alle Elemente einer Datenbank durch Kommentare dokumentieren. Exemplarisch speichern wir einen Kommentar zur Datenbank.

```
tweise@weise-laptop: ~  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data about our factory and products.'; 
```

Erstellen der Datenbank via Terminal



- Der SQL-Befehl `COMMENT ON ... IS` war erfolgreich, was die Ausgabe `COMMENT` anzeigt.

```
tweise@weise-laptop: ~  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data ab  
out our factory and products.';  
COMMENT  
postgres=#
```

Erstellen der Datenbank via Terminal



- Wir wollen uns nun die Liste aller Datenbanken auf dem Server ausgeben lassen und fragen die entsprechende Systemtabelle mit `SELECT ... FROM` ab.

```
tweise@weise-laptop: ~  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data ab  
out our factory and products.';  
COMMENT  
postgres=# SELECT datname FROM pg_database; 
```

Erstellen der Datenbank via Terminal

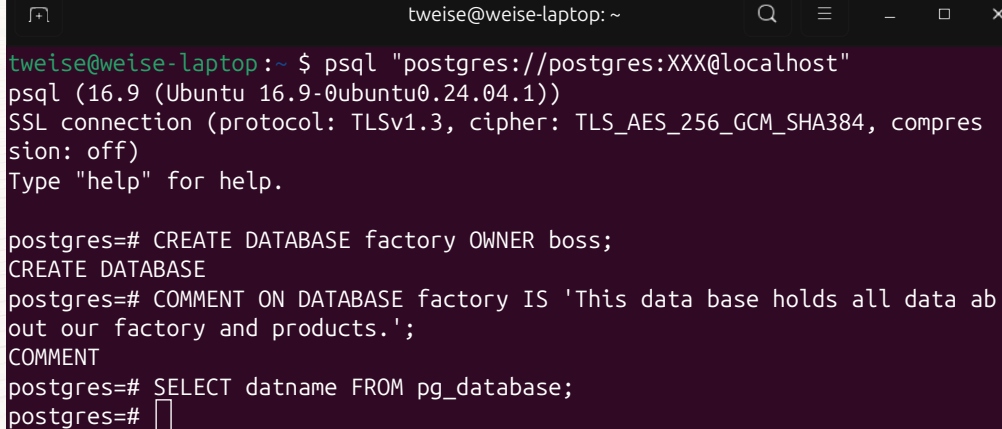


- Die Ausgabe zeigt unsere neue Datenbank mit an, ist aber ggf. im seitenweisen Anzeigemodus.

```
tweise@weise-laptop: ~  
datname  
-----  
postgres  
template1  
template0  
violation  
fixed  
factory  
test  
(7rows)  
  
~  
~  
(END)
```

Erstellen der Datenbank via Terminal

- Falls wir im seitenweisen Anzeigemodus sind, verlassen wir diesen durch Druck auf .



A terminal window titled 'tweise@weise-laptop: ~' with standard window controls. The terminal shows the execution of the 'psql' command to connect to a PostgreSQL database. The connection is successful, and the user is prompted to enter SQL commands. The commands shown are: creating a database named 'factory', adding a comment to it, and querying the 'pg_database' system catalog. The cursor is currently at the 'postgres=#' prompt.

```
tweise@weise-laptop:~ $ psql "postgres://postgres:XXX@localhost"
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=# CREATE DATABASE factory OWNER boss;
CREATE DATABASE
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data about our factory and products.';
COMMENT
postgres=# SELECT datname FROM pg_database;
postgres=#
```

Erstellen der Datenbank via Terminal



- Wir beenden die psql-Session durch `\q` und `↵`.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~ $ psql "postgres://postgres:XXX@localhost"  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data about our factory and products.';  
COMMENT  
postgres=# SELECT datname FROM pg_database;  
postgres=# \q
```

Erstellen der Datenbank via Terminal



- Wir sind zurück im normalen Terminal.

```
tweise@weise-laptop: ~  
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)  
Type "help" for help.  
  
postgres=# CREATE DATABASE factory OWNER boss;  
CREATE DATABASE  
postgres=# COMMENT ON DATABASE factory IS 'This data base holds all data about our factory and products.';  
COMMENT  
postgres=# SELECT datname FROM pg_database;  
postgres=# \q  
tweise@weise-laptop:~$
```

Erstellen der Datenbank via Script



```
1  /* In this example, we create a new database named 'factory'. */
2
3  -- On PostgreSQL, there is a table `pg_databases` listing all databases.
4  -- We print the column `datname` with the names of the databases.
5  SELECT datname FROM pg_database;
6
7  -- Create the database 'factory', owned by user 'boss'.
8  CREATE DATABASE factory OWNER boss;
9
10 -- Store a comment about the purpose of our database.
11 COMMENT ON DATABASE factory is
12     'This database holds all data about our factory and products.';
13
14 -- Now there is a new database in the list, namely 'factory'.
15 SELECT datname FROM pg_database;
```

```
1  $ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
   ↳ create_database.sql
2  datname
3  -----
4  postgres
5  template1
6  template0
7  (3 rows)
8
9  CREATE DATABASE
10 COMMENT
11   datname
12  -----
13  postgres
14  factory
15  template1
16  template0
17  (4 rows)
18
19 # psql 16.11 succeeded with exit code 0.
```

Erstellen der Datenbank via Script



```
1  /* In this example, we create a new database named 'factory'. */
2
3  -- On PostgreSQL, there is a table `pg_databases` listing all databases.
4  -- We print the column `datname` with the names of the databases.
5  SELECT datname FROM pg_database;
6
7  -- Create the database 'factory', owned by user 'boss'.
8  CREATE DATABASE factory OWNER boss;
9
10 -- Store a comment about the purpose of our database.
11 COMMENT ON DATABASE factory is
12     'This database holds all data about our factory and products.';
13
14 -- Now there is a new database in the list, namely 'factory'.
15 SELECT datname FROM pg_database;
```

Erstellen der Datenbank via Script



```
1 $ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
   ↪ create_database.sql
2     datname
3     -----
4     postgres
5     template1
6     template0
7 (3 rows)
8
9 CREATE DATABASE
10 COMMENT
11     datname
12     -----
13     postgres
14     factory
15     template1
16     template0
17 (4 rows)
18
19 # psql 16.11 succeeded with exit code 0.
```



Zusammenfassung



Zusammenfassung

- Wir haben soeben mit einem echten DBMS interagiert.



Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.
- Wir haben gesehen, dass das Schwierigste an der ganzen Sache ist, PostgreSQL und psql erstmal zum Laufen zu bekommen.

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.
- Wir haben gesehen, dass das Schwierigste an der ganzen Sache ist, PostgreSQL und psql erstmal zum Laufen zu bekommen.
- Das eigentliche Arbeiten mit dem DBMS war gar nicht so schwer...

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.
- Wir haben gesehen, dass das Schwierigste an der ganzen Sache ist, PostgreSQL und psql erstmal zum Laufen zu bekommen.
- Das eigentliche Arbeiten mit dem DBMS war gar nicht so schwer...
- ...und wir kennen nun zwei Möglichkeiten, mit dem PostgreSQL DBMS via psql zu interagieren

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.
- Wir haben gesehen, dass das Schwierigste an der ganzen Sache ist, PostgreSQL und psql erstmal zum Laufen zu bekommen.
- Das eigentliche Arbeiten mit dem DBMS war gar nicht so schwer...
- ...und wir kennen nun zwei Möglichkeiten, mit dem PostgreSQL DBMS via psql zu interagieren:
 - entweder über eine interaktive Session, bei der wir Kommandos in ein Terminal eintippen.

Zusammenfassung



- Wir haben soeben mit einem echten DBMS interagiert.
- Wir haben die SQL-Befehle `CREATE USER` und `CREATE DATABASE` zum Erstellen von Benutzerkonten und Datenbanken benutzt.
- Wir haben auch zum ersten Mal den SQL-Befehl `SELECT ... FROM` benutzt, um Daten aus einer Tabelle auszulesen.
- Wir haben gesehen, dass das Schwierigste an der ganzen Sache ist, PostgreSQL und psql erstmal zum Laufen zu bekommen.
- Das eigentliche Arbeiten mit dem DBMS war gar nicht so schwer...
- ... und wir kennen nun zwei Möglichkeiten, mit dem PostgreSQL DBMS via psql zu interagieren:
 - entweder über eine interaktive Session, bei der wir Kommandos in ein Terminal eintippen.
 - oder, indem wir die Kommandos in eine Datei schreiben und diese direkt an den Server schicken.



谢谢你们！
Thank you!
Vielen Dank!



References I



- [1] Dirk Angermann. *T-SQL-Abfragen für Microsoft SQL-Server 2022*. Blaufelden, Schwäbisch Hall, Baden-Württemberg, Germany: mitp Verlags GmbH & Co. KG, Juni 2024. ISBN: **978-3-7475-0633-2** (siehe S. **143**).
- [2] 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 also³ (siehe S. **133, 143**).
- [3] 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 also² (siehe S. **133, 143**).
- [4] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: **978-1-0981-1340-7** (siehe S. **143, 145**).
- [5] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Okt. 2019. ISBN: **978-1-4842-5514-8** (siehe S. **143**).
- [6] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: **978-1-83898-747-3** (siehe S. **142, 143**).
- [7] Peter Belknap, John Beresiewicz, Benoît Dageville, Karl Dias, Yakov Shafranovich und Khaled Yagoub. "A Decade of Oracle Database Manageability". 34(4):20–27, Dez. 2011. URL: http://sites.computer.org/debull/A11dec/DODM%5C_V2.pdf (siehe S. **144**).
- [8] 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. **145**).
- [9] 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. **142**).
- [10] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: **978-1-80324-028-2** (siehe S. **143**).
- [11] 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. **144**).

References II



- [12] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 143).
- [13] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 142).
- [14] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 142, 144).
- [15] Donald D. Chamberlin. "50 Years of Queries". *Communications of the ACM (CACM)* 67(8):110–121, Aug. 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/3649887. URL: <https://cacm.acm.org/research/50-years-of-queries> (besucht am 2025-01-09) (siehe S. 26–39, 144).
- [16] Stuart Cheshire und Marc Krochmal. *Special-Use Domain Names*. Request for Comments (RFC) 6761. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Feb. 2013. URL: <https://www.ietf.org/rfc/rfc4253.txt> (besucht am 2025-02-27) (siehe S. 143).
- [17] 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. 142, 143).
- [18] 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. 145).
- [19] 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. 144).
- [20] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 26–39, 144).
- [21] 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. 26–39, 144).

References III



- [22] Matt David und Blake Barnhill. "Syntax Conventions". In: *How to Teach People SQL*. San Francisco, CA, USA: The Data School, Chart.io, Inc., 28. Juli 2020. URL: <https://dataschool.com/how-to-teach-people-sql/syntax-conventions> (besucht am 2025-02-27) (siehe S. 103).
- [23] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 26–39, 144).
- [24] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide Web: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (siehe S. 145).
- [25] 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. 143).
- [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. 143, 144).
- [27] Donald E. Eastlake 3rd und Aliza R. Panitz. *Reserved Top Level DNS Names*. Request for Comments (RFC) 2606. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Juni 1999. URL: <https://www.ietf.org/rfc/rfc2606.txt> (besucht am 2025-02-27) (siehe S. 143).
- [28] 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. 142).
- [29] 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. 19–25, 144).

References IV



- [30] Kevin P. Gaffney, Martin Prammer, Laurence C. Brasfield, D. Richard Hipp, Dan R. Kennedy und Jignesh M. Patel. "SQLite: Past, Present, and Future". *Proceedings of the VLDB Endowment (PVLDB)* 15(12):3535–3547, Aug. 2022. Irvine, CA, USA: Very Large Data Bases Endowment Inc. ISSN: 2150-8097. doi:10.14778/3554821.3554842. URL: <https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf> (besucht am 2025-01-12). All papers in this issue were presented at the 48th International Conference on Very Large Data Bases (VLDB 2022), 9 5-9, 2022, hybrid/Sydney, NSW, Australia (siehe S. 144).
- [31] Jonas Gamalielsson und Björn Lundell. "Long-Term Sustainability of Open Source Software Communities beyond a Fork: A Case Study of LibreOffice". In: *8th IFIP WG 2.13 International Conference on Open Source Systems: Long-Term Sustainability OSS'2012*. 10.–13. Sep. 2012, Hammamet, Tunisia. Hrsg. von Imed Hammouda, Björn Lundell, Tommi Mikkonen und Walt Scacchi. Bd. 378. IFIP Advances in Information and Communication Technology (IFIPACT). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2012, S. 29–47. ISSN: 1868-4238. ISBN: 978-3-642-33441-2. doi:10.1007/978-3-642-33442-9_3 (siehe S. 142, 143).
- [32] 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. 143).
- [33] 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. 144).
- [34] 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. 144).
- [35] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 143).
- [36] 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. 142, 145).
- [37] D. Richard Hipp u. a. "Well-Known Users of SQLite". In: *SQLite*. Charlotte, NC, USA: Hipp, Wyrick & Company, Inc. (Hwaci), 2. Jan. 2023. URL: <https://www.sqlite.org/famous.html> (besucht am 2025-01-12) (siehe S. 144).
- [38] 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. 144).

References V



- [39] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Juni 2023. URL: [https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_\(en\).zip](https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip) (besucht am 2025-01-08). Consists of several parts, see <https://modern-sql.com/standard> for information where to obtain them. (Siehe S. 26–39, 144).
- [40] „stderr, stdin, stdout – Standard I/O Streams”. In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Hrsg. von Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) und San Francisco, CA, USA: The Open Group, 8. Aug. 2024. URL: <https://pubs.opengroup.org/onlinepubs/9799919799/functions/stdin.html> (besucht am 2024-10-30) (siehe S. 77–92, 145).
- [41] Darl Kuhn und Thomas Kyte. *Expert Oracle Database Architecture: Techniques and Solutions for High Performance and Productivity*. 4. Aufl. New York, NY, USA: Apress Media, LLC, Nov. 2021. ISBN: 978-1-4842-7499-6 (siehe S. 144).
- [42] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 144).
- [43] Joan Lambert und Curtis Frye. *Microsoft Office Step by Step (Office 2021 and Microsoft 365)*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Juni 2022. ISBN: 978-0-13-754493-6 (siehe S. 143).
- [44] Kent D. Lee und Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (siehe S. 144).
- [45] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: <https://www.libreoffice.org> (besucht am 2024-12-12) (siehe S. 142, 143).
- [46] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra und William L. Hosch. “Client-Server Architecture”. In: *Encyclopaedia Britannica*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 3. Jan. 2025. URL: <https://www.britannica.com/technology/client-server-architecture> (besucht am 2025-01-20) (siehe S. 142).
- [47] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 144).

References VI



- [48] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: <https://mariadb.com/kb/en/documentation> (besucht am 2025-04-24) (siehe S. 143).
- [49] Ron McFadyen und Cindy Miller. *Relational Databases and Microsoft Access*. 3. Aufl. Palatine, IL, USA: Harper College, 2014–2019. URL: <https://harpercollege.pressbooks.pub/relationaldatabases> (besucht am 2025-04-11) (siehe S. 143).
- [50] Jim Melton und Alan R. Simon. *SQL: 1999 – Understanding Relational Language Components*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Juni 2001. ISBN: 978-1-55860-456-8 (siehe S. 26–39, 144).
- [51] *Microsoft Word*. Redmond, WA, USA: Microsoft Corporation, 2024. URL: <https://www.microsoft.com/en-us/microsoft-365/word> (besucht am 2024-12-12) (siehe S. 143).
- [52] Cameron Newham und Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (siehe S. 142).
- [53] Regina O. Obe und Leo S. Hsu. *PostgreSQL: Up and Running*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Okt. 2017. ISBN: 978-1-4919-6336-4 (siehe S. 19–25, 144).
- [54] Robert Orfali, Dan Harkey und Jeri Edwards. *Client/Server Survival Guide*. 3. Aufl. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 25. Jan. 1999. ISBN: 978-0-471-31615-2 (siehe S. 142).
- [55] Camila A. Paiva, Raquel Maximino, Frederico Paiva, Rafael Accetta Vieira, Nicole Espanha, João Felipe Pimentel, Igor Wiese, Marco Aurélio Gerosa, Igor Steinmacher, Leonardo Murta und Vanessa Braganholo. "Analyzing the Adoption of Database Management Systems throughout the History of Open Source Projects". *Empirical Software Engineering: An International Journal* 30(3):71, Feb. 2025. London, England, UK: Springer Nature Limited. ISSN: 1382-3256. doi:10.1007/S10664-025-10627-Z. URL: <https://www.authorea.com/users/677798/articles/674742> (besucht am 2025-06-04) (siehe S. 19–25).
- [56] Dušan Petković. *Microsoft SQL Server 2019: A Beginner's Guide*. 7. Aufl. New York, NY, USA: McGraw-Hill, Jan. 2020. ISBN: 978-1-260-45888-6 (siehe S. 143).

References VII



- [57] *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) (siehe S. 53–64).
- [58] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 19–25, 144).
- [59] 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. 142).
- [60] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: 978-1-78398-154-0 (siehe S. 143).
- [61] 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. 144).
- [62] 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. 142).
- [63] 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. 142, 143).
- [64] 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. 143).
- [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. 144).
- [66] *SQLite*. Charlotte, NC, USA: Hipp, Wyrick & Company, Inc. (Hwaci), 2025. URL: <https://sqlite.org> (besucht am 2025-04-24) (siehe S. 144).

References VIII



- [67] "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. 26–39, 144).
- [68] "Stack Overflow 2024 Developer Survey". In: *Stack Overflow*. New York, NY, USA: Stack Exchange Inc., Mai–Juni 2024. URL: <https://survey.stackoverflow.co/2024> (besucht am 2025-06-01) (siehe S. 19–25).
- [69] 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. 26–39, 140, 144).
- [70] Ryan K. Stephens, Ronald R. Plew, Bryan Morgan und Jeff Perkins. *SQL in 21 Tagen. Die Datenbank-Abfragesprache SQL vollständig erklärt (in 14/21 Tagen)*. 6. Aufl. Burghann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of⁶⁹ (siehe S. 26–39, 144).
- [71] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 26–39, 144).
- [72] 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. 19–25, 144).
- [73] 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. 143).
- [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. 142, 143).
- [75] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 143).
- [76] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: <https://thomasweise.github.io/databases> (besucht am 2025-01-05) (siehe S. 142, 144).

References IX



- [77] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: <https://thomasweise.github.io/programmingWithPython> (besucht am 2025-01-05) (siehe S. 144).
- [78] *What is a Relational Database?* Armonk, NY, USA: International Business Machines Corporation (IBM), 20. Okt. 2021–12. Dez. 2024. URL: <https://www.ibm.com/think/topics/relational-databases> (besucht am 2025-01-05) (siehe S. 144).
- [79] Peter Whyte. *Microsoft SQL Server DBA Blog*. Edinburgh, Scotland, UK, 2018–2025. URL: <https://peter-whyte.com/sql-dba-blog> (besucht am 2025-06-03) (siehe S. 143).
- [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. 144).
- [81] Marianne Winslett und Vanessa Braganholo. "Richard Hipp Speaks Out on SQLite". *ACM SIGMOD Record* 48(2):39–46, Juni 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0163-5808. doi:10.1145/3377330.3377338 (siehe S. 144).
- [82] 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. 142).
- [83] 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. 143).
- [84] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 142).

Glossary (in English) I



Bash is a the shell used under Ubuntu Linux, i.e., the program that „runs“ in the terminal and interprets your commands, allowing you to start and interact with other programs^{13,52,84}. Learn more at <https://www.gnu.org/software/bash>.

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

client-server architecture is a system design where a central server receives requests from one or multiple clients^{9,46,54,59,62}. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as psql, or the different applications that access the DBs.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*⁷⁶.

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 DB⁸².

IT information technology

LAMP Stack A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP^{14,36}.

LibreOffice is on open source office suite^{31,45,63} which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See⁷⁶ 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^{28,63}. It is part of LibreOffice^{31,45,63} and has functionality that is comparable to Microsoft Access^{6,17,74}.

Glossary (in English) II



- LibreOffice Calc** is a spreadsheet software that allows you to arrange and perform calculations with data in a tabular grid. It is a free and open source spreadsheet software^{45,63}, i.e., an alternative to Microsoft Excel. It is part of LibreOffice^{31,45,63}.
- LibreOffice Writer** is a free and open source text writing program⁸³ and part of LibreOffice^{31,45,63}. It is a good alternative to Microsoft Word.
- Linux** is the leading open source operating system, i.e., a free alternative for Microsoft Windows^{4,35,64,73,75}. 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.
- localhost** is the hostname of the current computer^{16,27}. It is equivalent to the IP address 127.0.0.1. Any message or package sent to localhost will be sent to the current computer itself.
- MariaDB** An open source relational database management system that has forked off from MySQL^{2,3,5,26,48,60}. See <https://mariadb.org> for more information.
- Microsoft Access** is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases^{6,17,49,74}. It is part of Microsoft Office. A free and open source alternative to this commercial software is LibreOffice Base.
- Microsoft Excel** is a spreadsheet program that allows users to store, organize, manipulate, and calculate data in tabular structures^{10,32,43}. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc^{45,63}.
- Microsoft Office** is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access⁴³. LibreOffice is a free and open source alternative.
- Microsoft SQL Server** The Microsoft SQL Server is a successful commercial relational/Structured Query Language (SQL)-based DBMS^{1,56,79}. Learn more at <https://www.microsoft.com/sql-server> and <https://learn.microsoft.com/en-us/sql>.
- Microsoft Windows** is a commercial proprietary operating system¹². 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^{25,51,83} and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.

Glossary (in English) III



- MySQL** An open source relational database management system^{11,26,61,72,80}. MySQL is famous for its use in the LAMP Stack. See <https://www.mysql.com> for more information.
- Oracle Database** The Oracle Database was the first commercial SQL-based relational database¹⁵. It is still a highly successful proprietary product with many features^{7,41}. Learn more at <https://www.oracle.com/database>.
- PostgreSQL** An open source object-relational DBMS^{29,53,58,72}. See <https://postgresql.org> for more information.
- psql** is the client program used to access the PostgreSQL DBMS server.
- Python** The Python programming language^{38,44,47,77}, i.e., what you will learn about in our book⁷⁷. Learn more at <https://python.org>.
- relational database** A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other^{19,33,34,65,71,76,78}.
- server** In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carrying the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers¹⁴ in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the „server computer“⁴².
- SQL** The *Structured Query Language* is basically a programming language for querying and manipulating relational databases^{15,20,21,23,39,50,67,69–71}. It is understood by many DBMSes. You find the SQL commands supported by PostgreSQL in the reference⁶⁷.
- SQLite** is an relational DBMS which runs as in-process library that works directly on files as opposed to the client-server architecture used by other common DBMSes. It is the most wide-spread SQL-based DB in use today, installed in nearly every smartphone, computer, web browser, television, and automobile^{15,30,37,81}. Learn more at <https://sqlite.org>⁶⁶.

Glossary (in English) IV



- stderr** The *standard error stream* is one of the three pre-defined streams of a console process (together with the standard input stream (stdin) and the stdout)⁴⁰. It is the text stream to which the process writes information about errors and exceptions. If an uncaught **Exception** is raised in Python and the program terminates, then this information is written to stderr. If you run a program in a terminal, then the text that a process writes to its stderr appears in the console.
- stdin** The *standard input stream* is one of the three pre-defined streams of a console process (together with the stdout and the stderr)⁴⁰. It is the text stream from which the process reads its input text, if any. The Python instruction `input` reads from this stream. If you run a program in a terminal, then the text that you type into the terminal while the process is running appears in this stream.
- stdout** The *standard output stream* is one of the three pre-defined streams of a console process (together with the stdin and the stderr)⁴⁰. It is the text stream to which the process writes its normal output. The `print` instruction of Python writes text to this stream. If you run a program in a terminal, then the text that a process writes to its stdout appears in the console.
- terminal** A terminal is a text-based window where you can enter commands and execute them^{4,18}. 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^{18,36}. 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>.
- WWW** World Wide Web^{8,24}