合肥大學
HEFEI UNIVERSITY

Datenbanken

8. Fabrik-Datenbank: Benutzer und Datenbank

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: 2025-07-10

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

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.

Fabrik-Datenbank

- 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.

Fabrik-Datenbank

- 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.*

Fabrik-Datenbank

- 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.

Fabrik-Datenbank

- 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"[14] und auf Rang 2 in[10].

Fabrik-Datenbank

- 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"[14] und auf Rang 2 in[10].

Nützliches Werkzeug

PostgreSQL[6,9,12,18] 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 wird einiges lernen.
- Wir lernen die wichtigsten Struktur und Komponenten von relationalen Datenbanken kennen.

Was wird passieren?

- In diesem Beispiel werden wird 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 wird 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.

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.
  - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern, und wieder laden können.

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.
  - Wir lernen, wie wir Tabellen erstellen, Datensätze speichern, und wieder laden können.
- Wie geht das?

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.
  - 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 wird 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 entsprechn.
  - 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 wird 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 entsprechn.
  - 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 wird 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 entsprechn.
  - 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 Spache SQL, die von all diesen Datenbanken unterstützt wird.[1–3,5,7,8,13,15–17]

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.
  - 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 Spache SQL, die von all diesen Datenbanken unterstützt wird.[1–3,5,7,8,13,15–17]
- Natürlich können wir keines dieser Themen vollständig oder tiefgreifend bearbeiten.

Was wird passieren?

- In diesem Beispiel werden wird 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 entsprechn.
  - 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 Spache SQL, die von all diesen Datenbanken unterstützt wird.[1–3,5,7,8,13,15–17]
- 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 wird 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 entsprechn.
  - 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 Spache SQL, die von all diesen Datenbanken unterstützt wird.[1–3,5,7,8,13,15–17]
- 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 Benuterkontos

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:

```
-- Erstellen eines neuen Benutzerkontos.
--
-- userName: der Name des Benutzerkontos, das wir anlegen wollen.
-- password: dass Passwort, dass wir dem Benutzer zuweisen wollen.
CREATE USER userName WITH 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 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 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; under Microsoft Windows durch Druck auf ⊞ + 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; under Microsoft Windows durch Druck auf ⊞+R, dann Schreiben von `cmd`, dann Druck auf ↵).
- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] 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; under Microsoft Windows durch Druck auf ⊞ + R, dann Schreiben von `cmd`, dann Druck auf ↵).
- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] 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.

Verbinding to PostgreSQL via psql

- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.

Verbinding to PostgreSQL via psql

- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`

Verbinding to PostgreSQL via psql

- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
  - `postgress://` identifiziert den parameter als Verbindungs-URI.
  - Das zweite "`postgres`" ist der Benutzername.
  - Der Doppelpunkt "`:`" trennt den Benutzername von dem Password `XXX`.

Verbinding to PostgreSQL via psql

- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
  - `postgress://` identifiziert den parameter als Verbindungs-URI.
  - Das zweite "`postgres`" ist der Benutzername.
  - Der Doppelpunkt "`:`" trennt den Benutzername von dem Password `XXX`. Benutzen Sie niemals etwas wie `XXX` als Passwort.

Verbinding to PostgreSQL via psql

- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
  - `postgress://` identifiziert den parameter als Verbindungs-URI.
  - Das zweite "`postgres`" ist der Benutzername.
  - Der Doppelpunkt "`:`" trennt den Benutzername von dem Password `XXX`. Benutzen Sie niemals etwas wie `XXX` als Passwort.
  - Nach dem "`@`" kommt die Netzwerkadresse, auf der der PostgreSQL-Server läuft.

Verbinding to PostgreSQL via psql

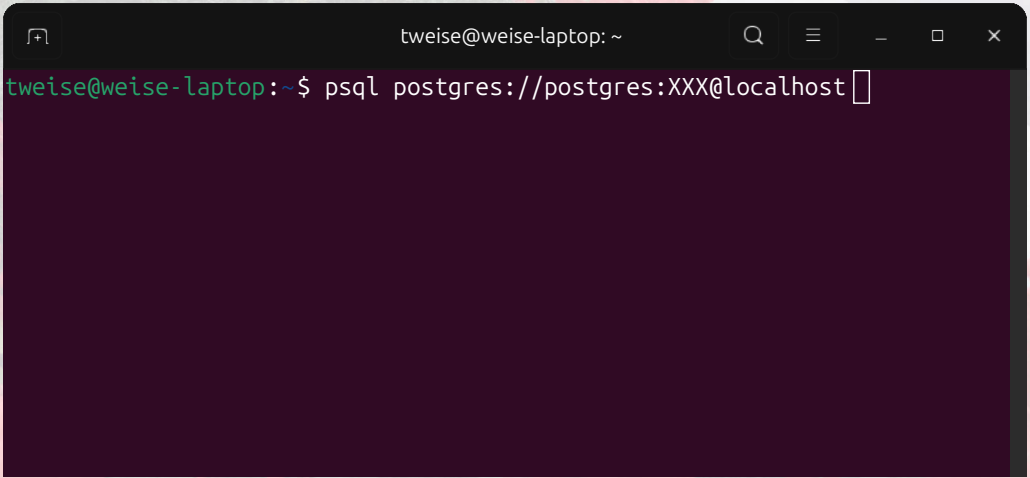
- Wir wollen nun den Klienten psql mit der passenden Verbindungs-URI[11] starten, um auf den PostgreSQL-Server zuzugreifen.
- Die Verbindungs-URI ist `postgres://postgres:XXX@localhost`, wobei
  - `postgress://` identifiziert den parameter als Verbindungs-URI.
  - Das zweite "`postgres`" ist der Benutzername.
  - Der Doppelpunkt "`:`" trennt den Benutzername von dem Password `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.

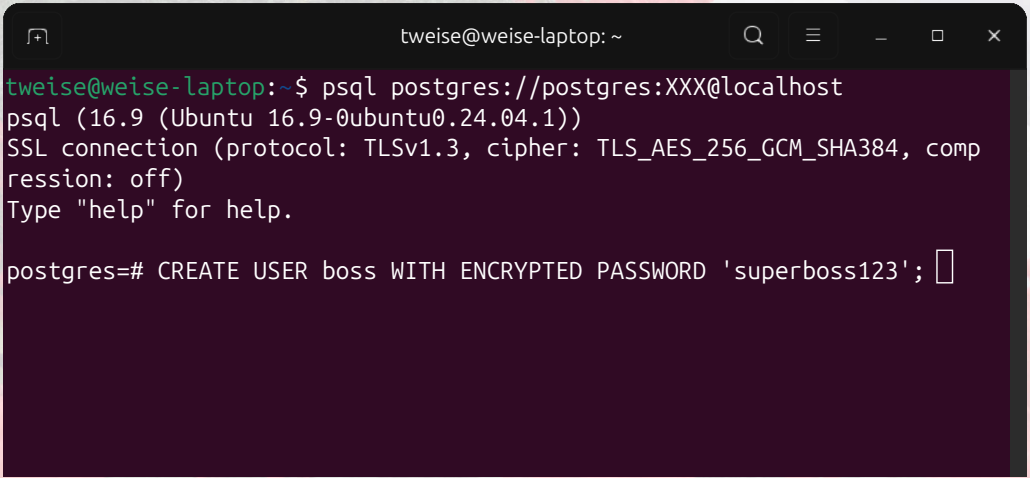


Erstellen des Benutzers via Terminal

- psql läuft und ist mit dem PostgreSQL DBMS verbunden.

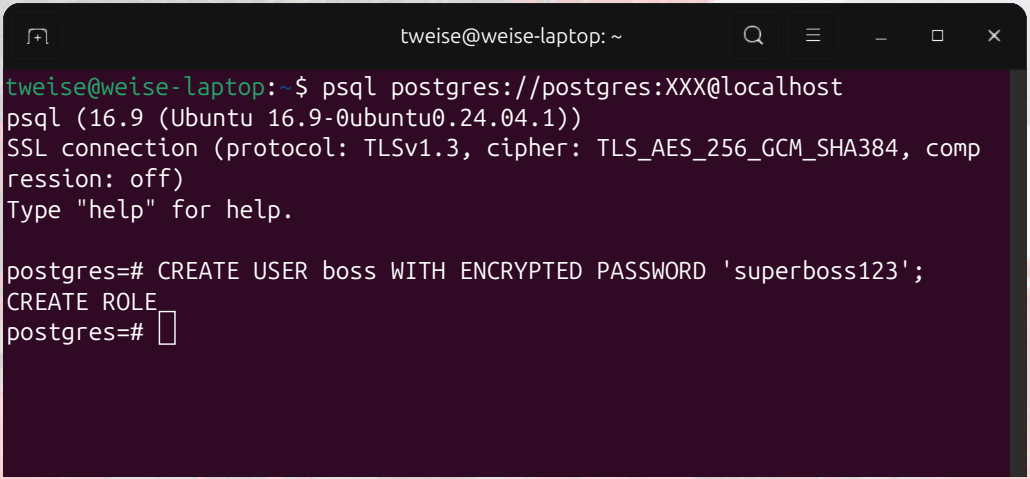
Erstellen des Benutzers via Terminal

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



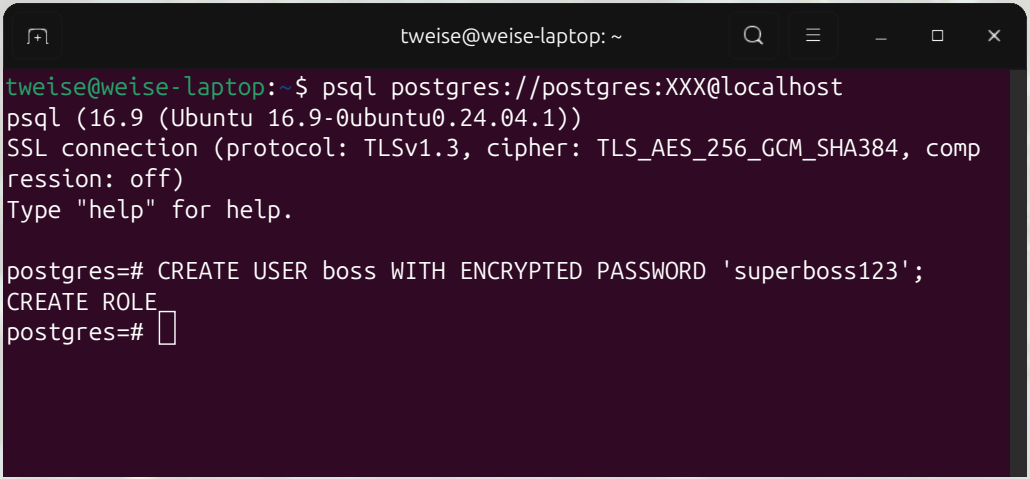
Erstellen des Benutzers via Terminal

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



Erstellen des Benutzers via Terminal

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



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: ~
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';
CREATE ROLE
postgres=# SELECT usename FROM pg_catalog.pg_user;
```

Erstellen des Benutzers via Terminal

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

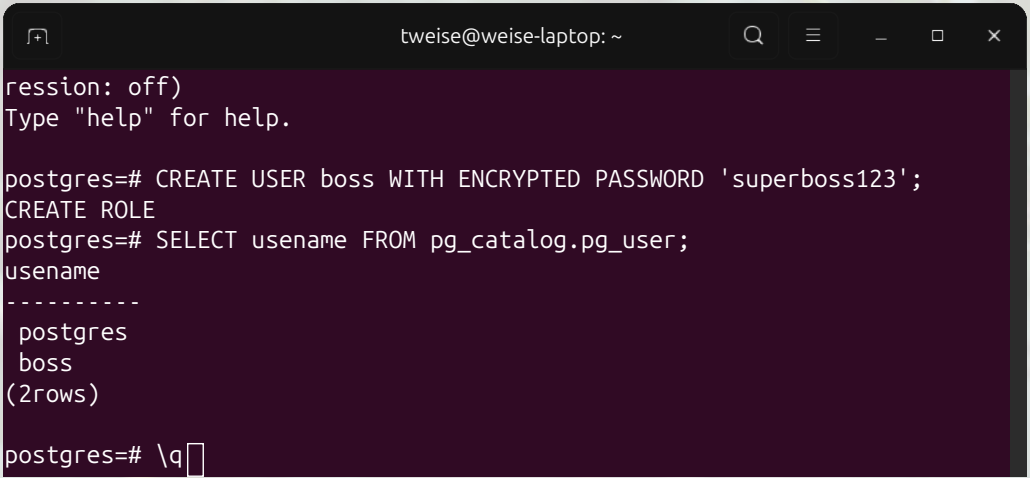
```
tweise@weise-laptop: ~
ression: off)
Type "help" for help.

postgres=# CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';
CREATE ROLE
postgres=# SELECT usename FROM pg_catalog.pg_user;
usename
----------
 postgres
 boss
(2rows)

postgres=#
```

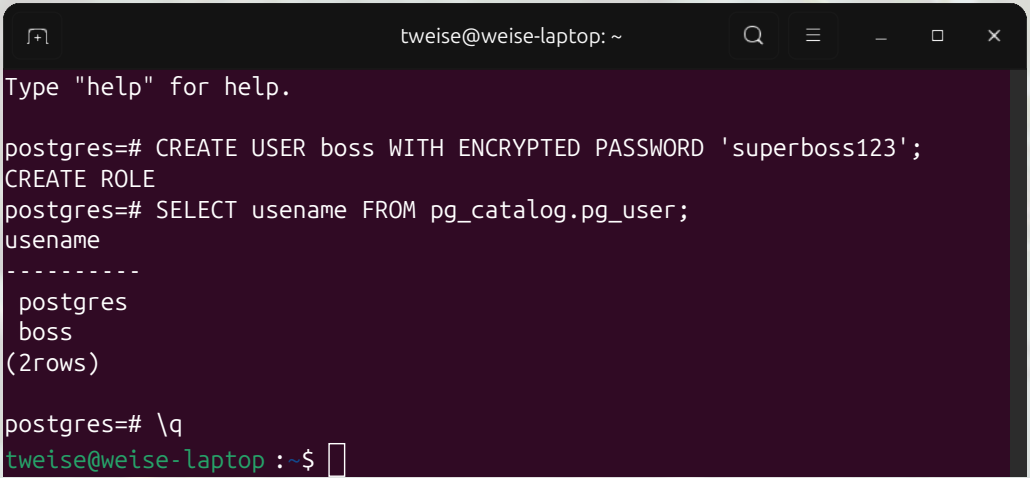
Erstellen des Benutzers via Terminal

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



Erstellen des Benutzers via Terminal

- Wir sind zurück im normalen Terminal.



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:

```sql
-- Werte bestimmter Spalten einer Tabelle auslesen.
--
-- Dieses Kommando gibt die Werte der Spalte 'columnName' der Tabelle
-- 'tableName' für alle Zeilen zurück.
-- Das Ergebnis sieht wie eine (temporäre) Tabelle aus, die nur die
-- aus Tabelle 'tableName' ausgewählte(n) Spalte(n) hat.
--
-- columnName: Name der Spalte, die wir auswählen wollen
--  tableName: Name der Tabelle, zu der die Spalte gehört.
SELECT columnName FROM tableName;

-- Sie können auch mehrere Spalten auf einmal auswählen.
-- Dieser Befehl selektiert N Spalten aus der Tabelle.
-- Die Namen der Spalten sind durch Kommas (",") getrennt.
SELECT columnName1,columnName2,...,columnNameN FROM tableName;

-- Diese Anweisung gibt alle Werte aus allen Spalten der Tabelle zurück.
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 auführen 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 auführen 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 das 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 das 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 das 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 das 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 Program 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 das 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 Program soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf `stdout`

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 das 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 Program soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf `stdout`
- `-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 das 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 Program soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf `stdout`
- `-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 das 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 Program soll bei Fehlern sofort abbrechen.
- `-e`: Drucke alle erfolgreichen Befehle auf `stdout`
- `-b`: Drucke gescheiterte Befehle auf `stderr`.
- `-f filename`: Lese Kommandos aus Datei `filename`.
- `-ebf filename`: Das selbe wir `-e -b -f filename`.

Erstellen des Benutzers via Script


```sql
/* In this example, we create a new user for our database. */

-- On PostgreSQL, there is a table `pg_catalog.pg_user` with all users.
-- We print the column `usename` with the user names.
SELECT usename FROM pg_catalog.pg_user;

-- Create the user 'boss'.
-- He will be the owner of the database that we will create.
CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';

-- Now there is a new user: 'boss'.
SELECT usename FROM pg_catalog.pg_user;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ create_user.sql
 usename
----------
 postgres
(1 row)

CREATE ROLE
 usename
----------
 postgres
 boss
(2 rows)

# psql 16.9 succeeded with exit code 0.
```

Erstellen des Benutzers via Script

```
/* In this example, we create a new user for our database. */

-- On PostgreSQL, there is a table `pg_catalog.pg_user` with all users.
-- We print the column `usename` with the user names.
SELECT usename FROM pg_catalog.pg_user;

-- Create the user 'boss'.
-- He will be the owner of the database that we will create.
CREATE USER boss WITH ENCRYPTED PASSWORD 'superboss123';

-- Now there is a new user: 'boss'.
SELECT usename FROM pg_catalog.pg_user;
```

Erstellen des Benutzers via Script

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ create_user.sql
 usename
----------
 postgres
(1 row)

CREATE ROLE
 usename
----------
 postgres
 boss
(2 rows)

# psql 16.9 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 Kode nach einem konsistenten Stil zu formattieren.

Ein Wort zu Stilrichtlinien

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

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Kode und Skripte nach nach einem konsistenten Stil zu formattieren, 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 Kode nach einem konsistenten Stil zu formattieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Kode und Skripte nach nach einem konsistenten Stil zu formattieren, 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 Kode nach einem konsistenten Stil zu formattieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Kode und Skripte nach nach einem konsistenten Stil zu formattieren, 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 Kode nach einem konsistenten Stil zu formattieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Kode und Skripte nach nach einem konsistenten Stil zu formattieren, 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 Kode nach einem konsistenten Stil zu formattieren.

Gute Praxis

Egal welche Programmiersprache Sie benutzen, es ist wichtig, Kode und Skripte nach nach einem konsistenten Stil zu formattieren, 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 Kode nach einem konsistenten Stil zu formattieren.
- 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 werden[4].

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`:

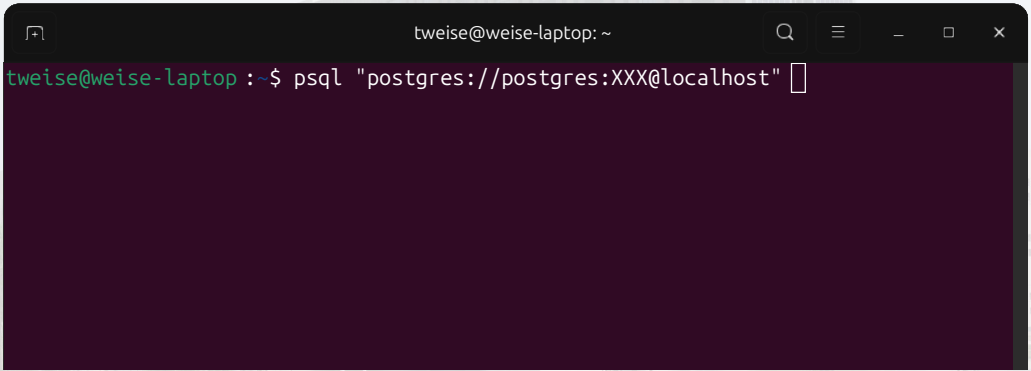
```
1 -- Erstellen einer Datenbank.
2 --
3 -- 'databaseName': der Name für die neue Datenbank.
4 -- 'userName':     der Benutzer, dem die neue Datenbank "gehören" soll.
5 CREATE DATABASE databaseName OWNER userName;
```

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`.

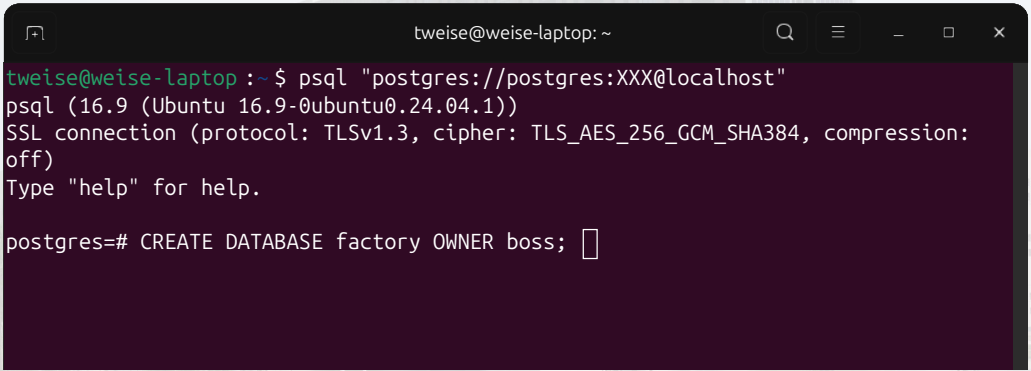


Erstellen der Datenbank via Terminal

- psql läuft und ist mit dem PostgreSQL DBMS verbunden.

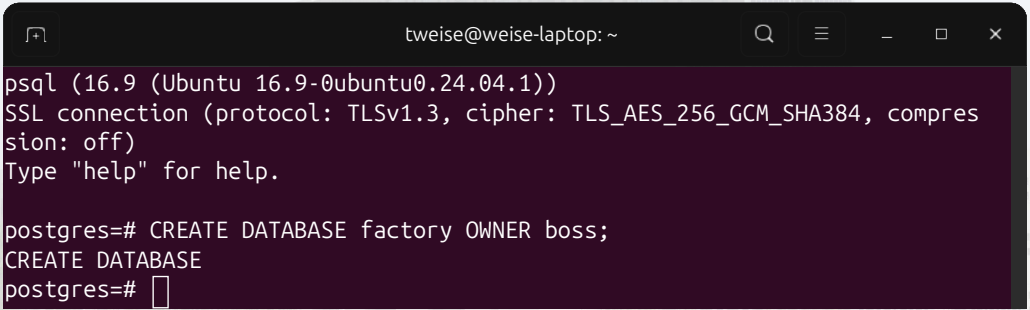
Erstellen der Datenbank via Terminal

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



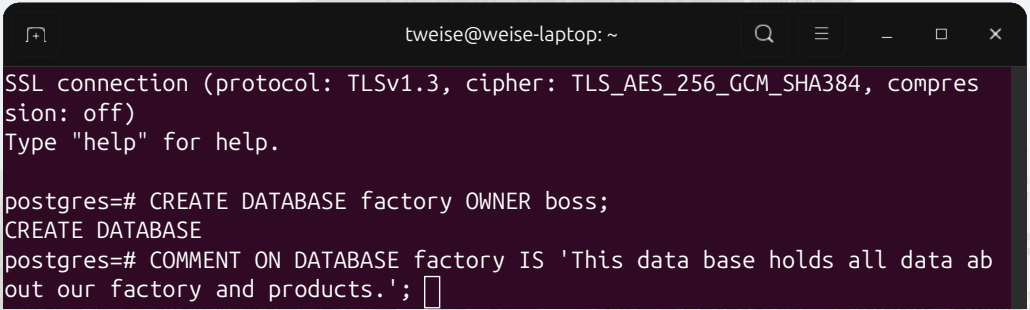
Erstellen der Datenbank via Terminal

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



Erstellen der Datenbank via Terminal

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



Erstellen der Datenbank via Terminal

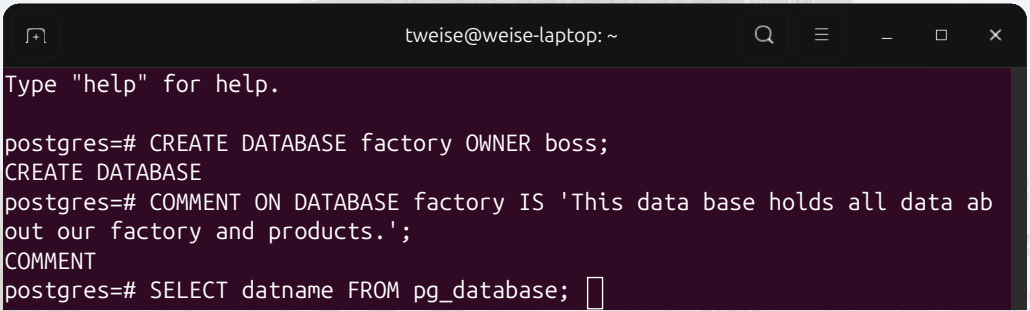
- Der SQL-Befehl `COMMENT ON ... IS` war erfolgreich, was die Ausgabe `COMMENT` anzeigt.

```
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.

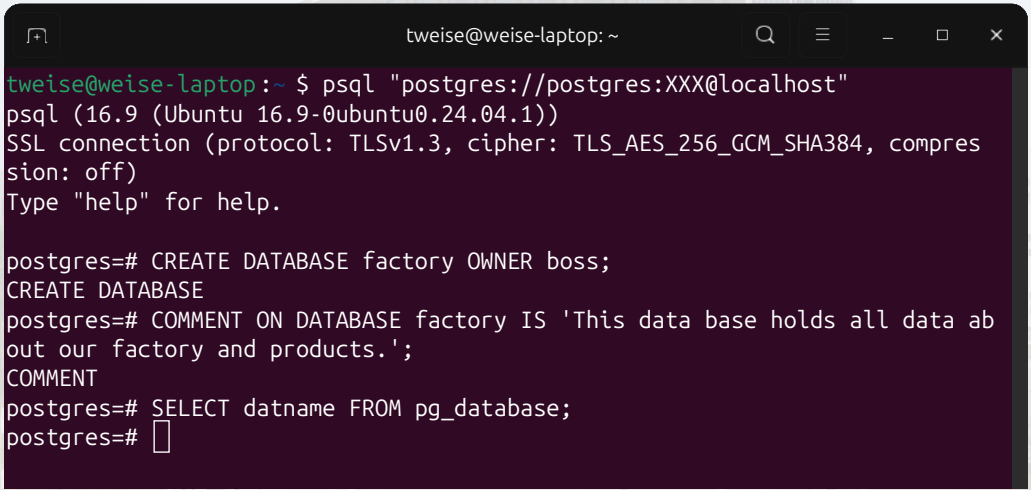


Erstellen der Datenbank via Terminal

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

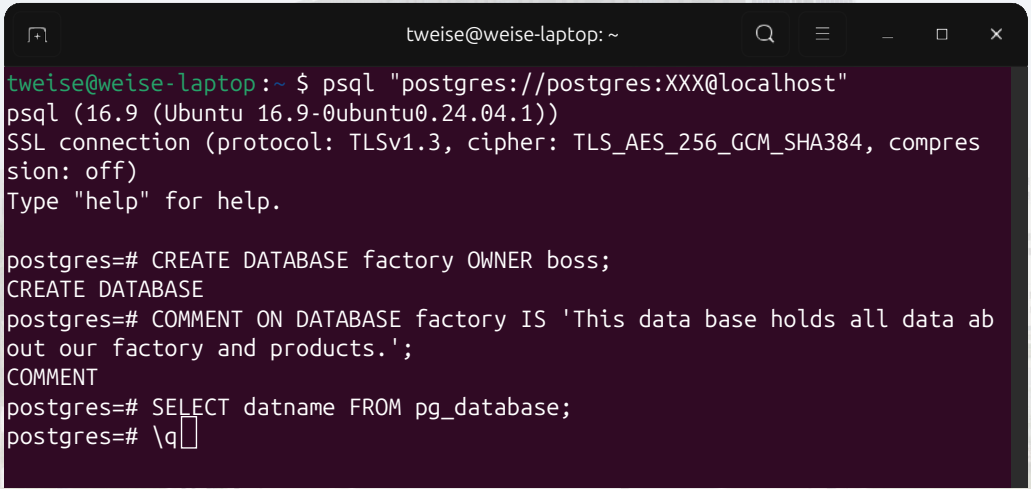
Erstellen der Datenbank via Terminal

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



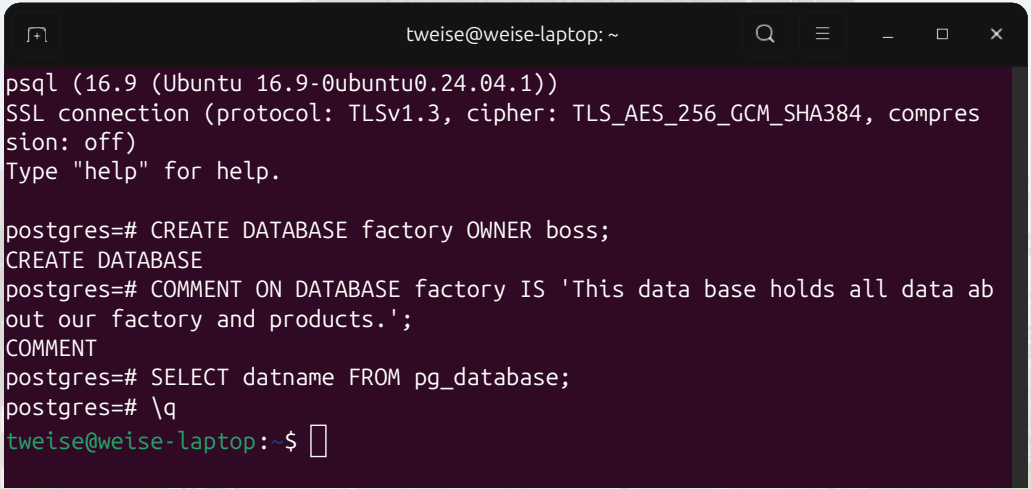
Erstellen der Datenbank via Terminal

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



Erstellen der Datenbank via Terminal

- Wir sind zurück im normalen Terminal.



Erstellen der Datenbank via Script

```sql
/* In this example, we create a new database named 'factory'. */

-- On PostgreSQL, there is a table `pg_databases` listing all databases.
-- We print the column `datname` with the names of the databases.
SELECT datname FROM pg_database;

-- Create the database 'factory', owned by user 'boss'.
CREATE DATABASE factory OWNER boss;

-- Store a comment about the purpose of our database.
COMMENT ON DATABASE factory is
    'This database holds all data about our factory and products.';

-- Now there is a new database in the list, namely 'factory'.
SELECT datname FROM pg_database;
```

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ create_database.sql
  datname
-----------
 postgres
 template1
 template0
(3 rows)

CREATE DATABASE
COMMENT
  datname
-----------
 postgres
 factory
 template1
 template0
(4 rows)

# psql 16.9 succeeded with exit code 0.
```

Erstellen der Datenbank via Script


```sql
/* In this example, we create a new database named 'factory'. */

-- On PostgreSQL, there is a table `pg_databases` listing all databases.
-- We print the column `datname` with the names of the databases.
SELECT datname FROM pg_database;

-- Create the database 'factory', owned by user 'boss'.
CREATE DATABASE factory OWNER boss;

-- Store a comment about the purpose of our database.
COMMENT ON DATABASE factory is
    'This database holds all data about our factory and products.';

-- Now there is a new database in the list, namely 'factory'.
SELECT datname FROM pg_database;
```

Erstellen der Datenbank via Script

```
$ psql "postgres://postgres:XXX@localhost" -v ON_ERROR_STOP=1 -ebf
    ↪ create_database.sql
  datname
-----------
 postgres
 template1
 template0
(3 rows)

CREATE DATABASE
COMMENT
  datname
-----------
 postgres
 factory
 template1
 template0
(4 rows)

# psql 16.9 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.

谢谢您们！

Thank you!

Vielen Dank!

References I

[1] 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).

[2] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, DC, USA: American National Standards Institute (ANSI), 1986 (siehe S. 26–39).

[3] 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).

[4] 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).

[5] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 26–39).

[6] 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).

[7] *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. 26–39).

[8] 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).

[9] 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).

References II

[10] 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).

[11] *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).

[12] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 19–25).

[13] "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).

[14] "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).

[15] 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, 134).

[16] 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[15] (siehe S. 26–39).

[17] 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).

[18] 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).