



Python Komplet

5-Tages Seminar

Tag 3+4

Timm Gieger



Agenda – Tag 3+4

Ausstehende Themen von Tag 1+2

Klassen / Objektorientierte Programmierung

SQL Grundlagen / SQLAlchemy / Pandas / numpy

Feedback & Fragerunde

Agenda – Tag 5

Ausstehende Themen von Tag 3+4

GUI Programmierung

Testing / OS-Befehle

Ausblick / eigene Themen

Feedback & Fragerunde

Installation von benötigten Packages

1. Anaconda Navigator öffnen (oder direkt Anaconda Prompt über Windows-Suche)
2. Anaconda Prompt installieren und starten (falls noch nicht installiert)
3. Navigieren in Verzeichnis von requirements.txt (Schulungsunterlagen)
4. „pip install -r requirements.txt“



03 Funktion / Methoden

Funktionen / Methoden

Syntax

- Kann von Programm aufgerufen werden
- Mehr Flexibilität durch Parameter
- Rückgabewerte möglich

Syntax:

Definition:

```
def funktionsbezeichnung(Parameter):  
    Anweisungen  
    return Rückgabewert
```

Aufruf:

```
funktionsbezeichnung(Parameter)
```

Funktionen / Methoden

Docstring

- Ermöglicht es Funktionen/Methoden direkt zu dokumentieren

Syntax:

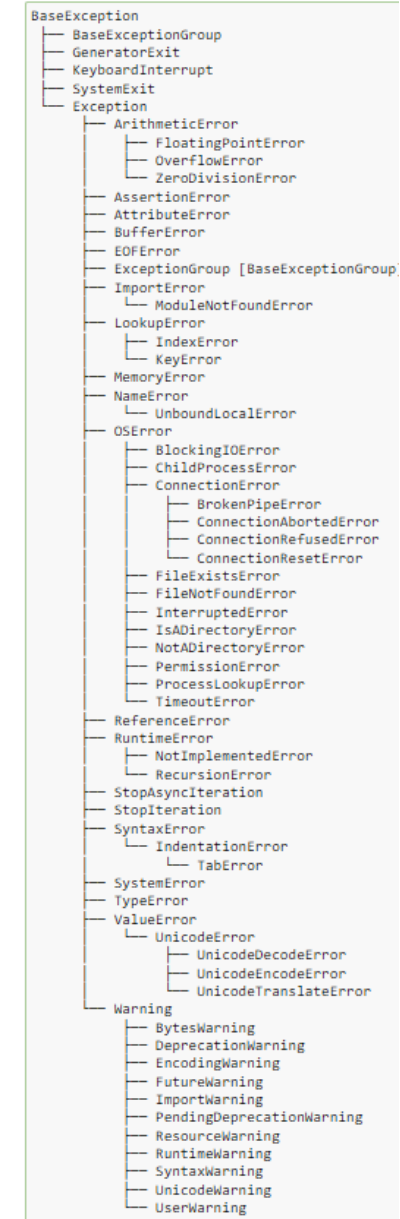
```
>>> def fahrenheit(T_in_celsius):  
...     """ returns the temperature in degrees Fahrenheit """  
...     return (T_in_celsius * 9 / 5) + 32  
...
```

Funktionen / Methoden mit Python

04 Exceptions

Exceptions

- Fehlerbehandlung in Python-Programmen
- Ermöglicht es unbehandelte Fehler aufzufangen ohne Programmabbruch
- Es gibt verschiedene Arten von Exceptions die aufgefangen werden können



**Built-In
Exceptions**

Exceptions

Basic Syntax:

try:

 Anweisung

except:

 Anweisung wenn Fehler in try-Block

Exceptions mit Python

05 Dateien einlesen & schreiben

Lesen & schreiben von Textfiles

- Lesen von textfiles mit `open()`-Methode
- Möglichkeit files komplett einzulesen mit `read()` oder `readlines()`
- Schreiben von Strings mit `write()` und `writelines()`

Dateien einlesen & schreiben

Pickle-Modul

- Persistenter Datenspeicher
- Ermöglicht Serialisierung von Objekten
- Persistierung über Laufzeit von Python-Programmen hinaus



Dateien einlesen & schreiben

Shelf-Modul

- Ebenfalls persistenter Datenspeicher
- Ermöglicht Dictionary-ähnliche Speicherung von Daten (ähnlich wie Regal)
- Shelf-Daten können nach einlesen wieder über Key/Value Paare referenziert werden



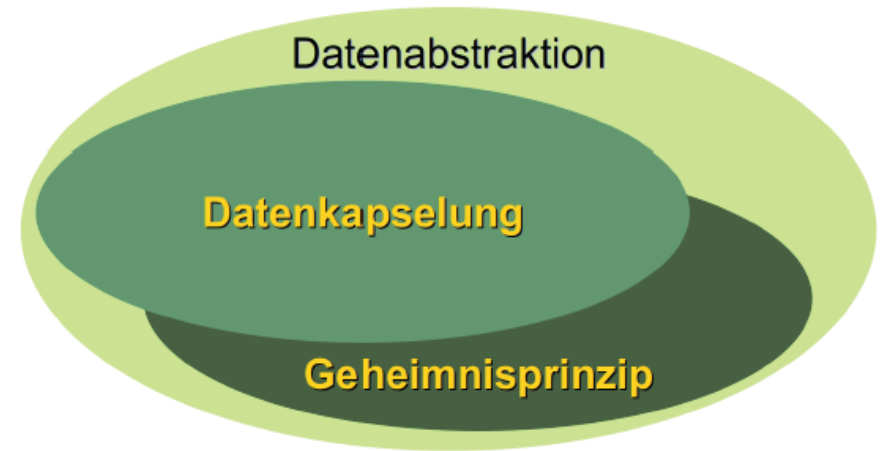
Dateien einlesen & schreiben mit Python

Objektorientierte Programmierung

Objektorientierte Programmierung

- Aufgeteilt in Objekte & Klassen:
 - Klasse bildet das Konzept eines Objekts
 - Objekt ist eine Instanz einer Klasse
- Geheimnisprinzip – Daten sind nach außen nicht sichtbar
→ Beispiel: pop und append-Methode der list-Klasse (interne Struktur verborgen)
- Datenkapselung – Schutz der Daten/Attributen vor unmittelbarem Zugriff

Datenabstraktion = Datenkapselung + Geheimnisprinzip



Objekte

- Jedes Objekt ist eine Instanz einer Klasse (Bauplan)
- Haben die gleichen Attribute und Methoden wie andere Objekte der gleichen Klasse zur Verfügung

Beispiel Konto:

Jedes Konto besitzt folgende Attribute:

- IBAN, Kontostand, Kontoinhaber

...

Jedes Konto besitzt folgende Methoden:

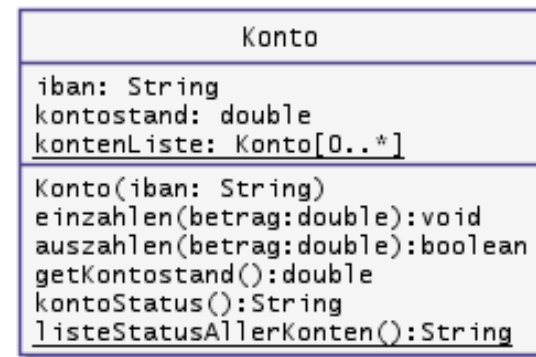
- Einzahlen, auszahlen, Kontostand abfragen etc...

UML-Klassendiagramme

(Unified Modeling Language)

- Grafische Darstellung von Objekten und deren Beziehungen zueinander
- UML als Standard für Objektorientierte Modellierung
- Technologie/Programmiersprachenunabhängig!!!

Beispiel für Klassendiagramm für die Klasse Konto:



UML-Klassendiagramme

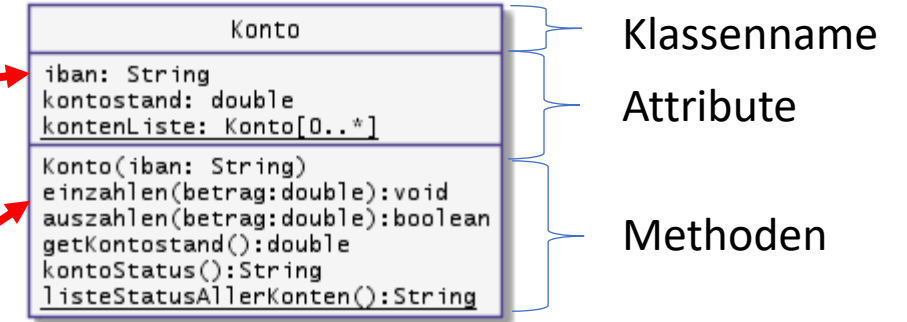
Aufbau

(Instanz-)Attribute:

- Eigenschaften eines Objekts
- Jedes Instanzattribut gilt nur für die eigene Instanz
- Notation in Form von
Name: Datentyp

Operationen/Methoden:

- Verhalten von Objekten
- Notation in Form von
Methodenname (parameter:
Parametertyp) : TypDesRückgabewerts



UML-Klassendiagramme

Aufbau

statische Attribute:

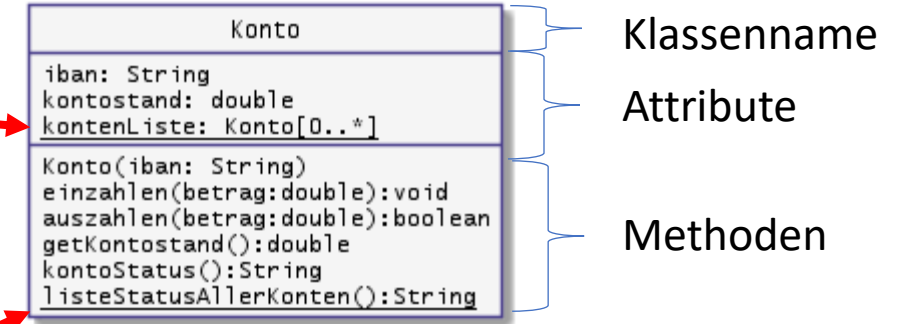
- Gelten für alle Instanzen einer Klasse
- Notation in Form von

Name: Datentyp

statische Methoden:

- Hat keinen Zugriff auf Instanzattribute (nur Klassenattribute)
- Benötigen keine Objektinstanzen um aufgerufen werden zu können
- Notation in Form von

Methodenname (parameter:
Parametertype) : TypDesRückgabewerts



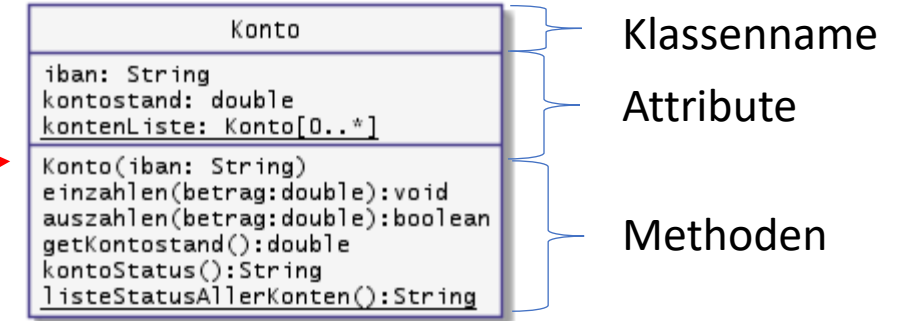
UML-Klassendiagramme

Aufbau

Konstruktor/init-Methode:

- Konstruktor beschreibt die Erstellung einer Instanz der Objektklasse
- Manchmal durch <<create>> gekennzeichnet
- Notation in Form von:

Klassenname(parameter:Datentyp)



Kapselung & Datenabstraktion

Beispiel

Zugriff mit „gettern“ und „settern“ auf private oder geschützte Instanzattribute

Möglich mit __ (doppeltem underscore)

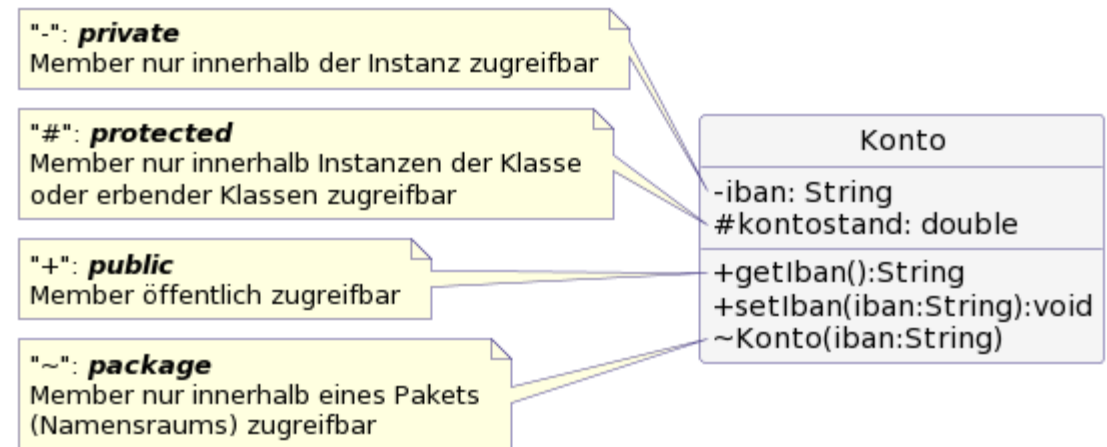
`__var1 = None`

Möglich mit _ (einzelnem underscore)

`_var1 = None`

default in Python

`var1 = None`



CC BY 4.0 Hannes Stein

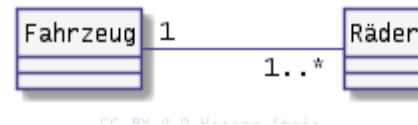
Objektbeziehungen

Assoziation: Klassen benutzen
Methoden/Attribute anderer Klassen



Kardinalität & Multiplizität:

Wie viele Instanzen einer Klasse können mit
einer anderen in Beziehung stehen



Objektbeziehungen

Aggregation vs. Komposition:

Aggregation:

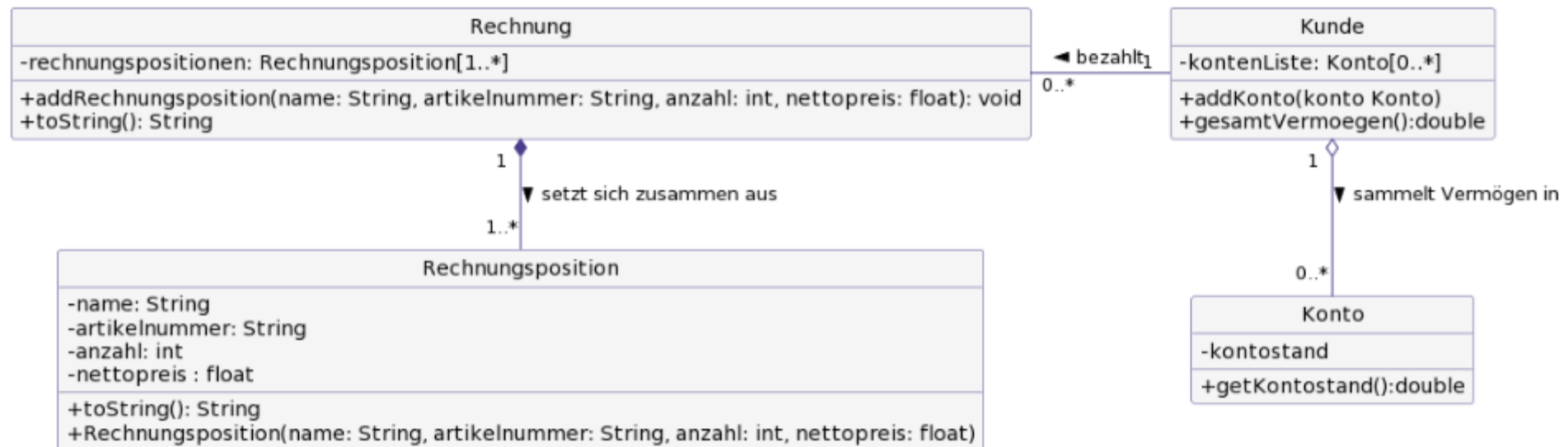
- Komponente kann auch unabhängig vom Ganzen existieren (z.B.: Kunde & Konto)
- Wird in eigener Klasse instanziiert
- Notation als unausgefüllte Raute

Komposition:

- Komponenten können nicht alleine existieren (z.B.: Baum & Blätter)
- Werden in verbundener Klasse instanziiert sobald ein Objekt dieser Klasse erstellt wird
- Notation als ausgefüllte Raute

Objektbeziehungen

Aggregation vs. Komposition:

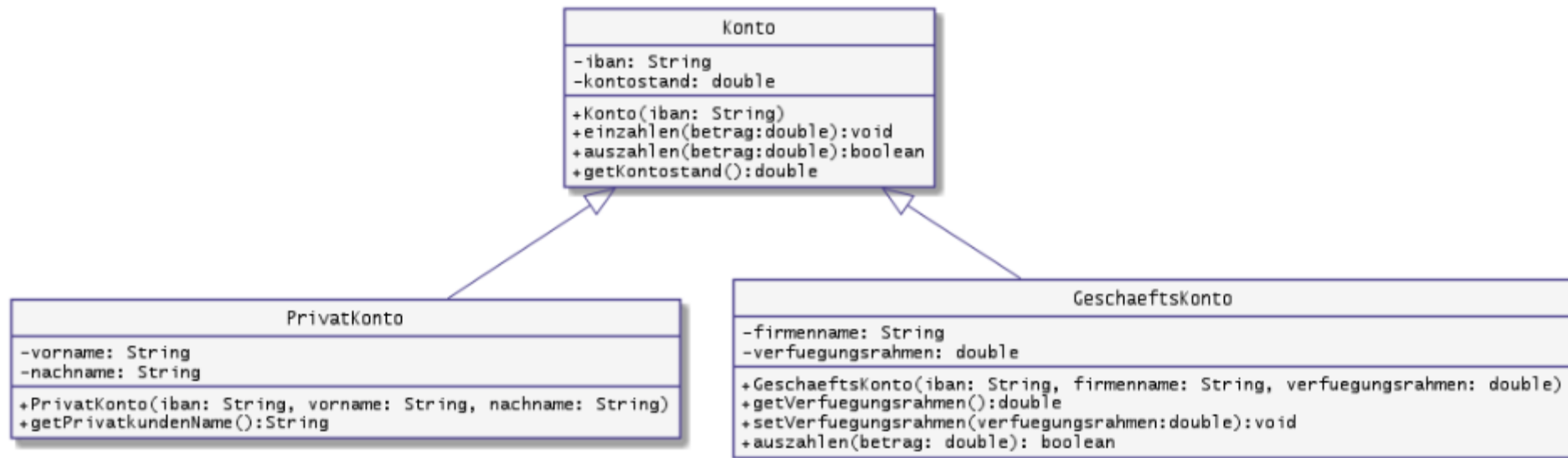


Vererbung

- macht Mehrfachnutzung von Klassenattributen und Methoden möglich
- Bezeichnung als Eltern-Kind oder Super-Subklasse
- Wird als unausgefüllter gerichteter Pfeil von der erbendenden auf die vererbende Klasse dargestellt
- Methoden können „überschrieben“ werden wenn diese neue Funktionalitäten benötigen/hinzufügen

Vererbung

Beispiel



Objektorientierte Programmierung mit Python

Datenbanken / SQLAlchemy

Typen von Datenbanken

Hierarchische Datenbanken:

- Hierarchische Baumstruktur (Parent-Child Relationships)
- Veraltetes Datenbankprinzip

Objektorientierte Datenbanken:

- Meist auf Java und .Net-Plattformen zu finden
- Speicherung durch Objekte und Methoden

Relationale Datenbanken:

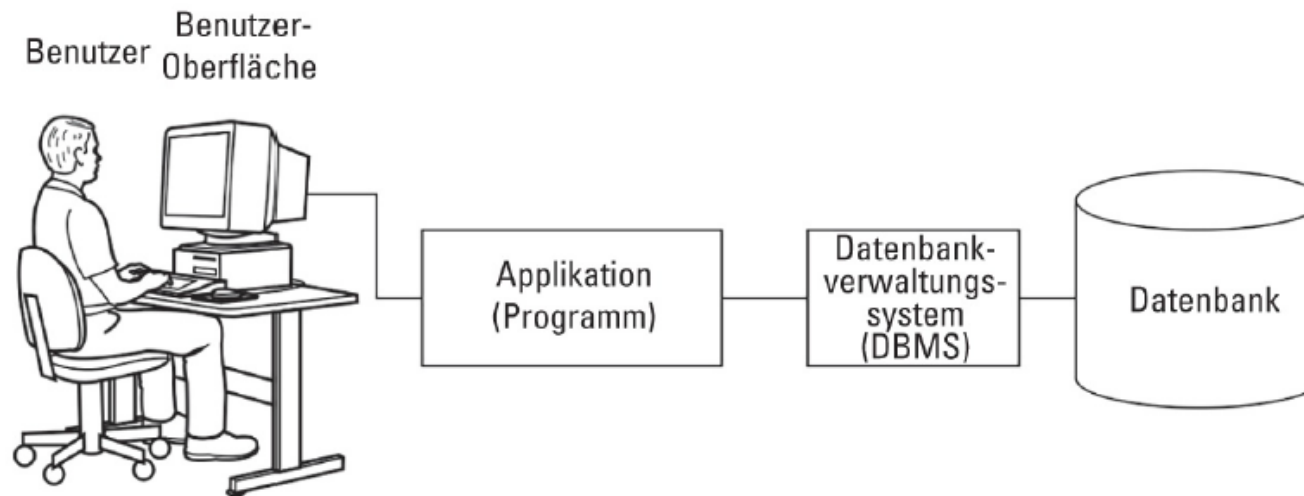
- Verfügt über RDBMS
- Hält ACID-Prinzipen + Referentielle Integrität
- Über Tabellen und Relationen definiert (Schlüsselprinzip)
- Über SQL „programmierbar“

NoSQL Datenbanken:

- SQL teilweise nutzbar
- Es liegt meist kein festes Schema zugrunde
- Zeitreihendatenbanken (InfluxDB)
- Dokumentenbasierte Datenbank / Key-Value Stores (MongoDB)
- Graphdatenbanken (Neo4j)

RDBMS

- Softwareapplikation
- Überwacht ACID Prinzipen und Referentielle Integrität
- Ermöglicht vereinfachte Einrichtung von Datenbanken/Usern/Rechten etc.



ACID

Atomicity (A): Transaktionen oder Befehle werden entweder ganz oder garnicht ausgeführt – der User bekommt erst den Stand nach vollständiger Ausführung zu sehen

Consistency (C): Ein konsistenter Zustand wird beibehalten → jeder verfügt über die gleichen Informationen

Isolation (I): Eine Ausführung mehrerer gleichzeitiger Transaktionen führt immer zum gleichen Endzustand, als würden diese einzeln ausgeführt werden

Durability (D): Daten dürfen nur durch Transaktionen geändert werden und nicht durch äußere Einflüsse

➡ NoSQL Datenbanken oder Cluster können ACID meist nicht komplett umsetzen (verteilte Systeme)

Datenbankdesign

Primary-/Foreign- Key Prinzip

- Primärschlüssel werden benötigt, um Eindeutigkeit von Daten umzusetzen
 - Beispiele für Primärschlüssel: Steueridentifikationsnummer, Produkt-id
 - Bei Mehrdeutigkeit können auch zusammengesetzte Schlüssel benannt werden (z.B.: id und timestamp für Zeitreihen)
 - Partielle oder transitive Abhängigkeit sollte vermieden werden (siehe 2te + 3te Normalform)
- Tabellen/Entitäten können durch Fremdschlüssel verknüpft werden
 - Wenn in einer Tabelle *Rechnung* (Primärschlüssel *rechnungsid* aus Tabelle *Rechnung*) die Kundennr. des Kunden hinterlegt werden soll (Fremdschlüssel *kundennr.* aus Tabelle *Kunde*)
 - Fremdschlüssel werden in ihrer eigenen Tabelle als Primärschlüssel behandelt

→ Aufgabe des RDBMS ist es bspw. zu schauen, dass Primärschlüssel unique bleiben und miteinander verknüpfte Informationen nicht einfach gelöscht werden können

Datenbankdesign

Normalformen

Was ist eine Normalform und warum braucht man diese?

Normalformen sorgen dafür dass Datenbanktabellen so entworfen werden, dass keine Änderungsanomalien auftreten können und so alle Daten zuverlässig gespeichert werden

Was bedeutet das?

Datenbankdesign

Normalformen

1te Normalform

- Daten müssen atomar vorliegen
- Daten müssen im selben Datentyp vorliegen

Negativbeispiel:

Adresse: „Musterstraße 1, 12345 Musterhausen“ (nicht atomar)

Richtig:

Straße: „Musterstraße“, Hausnummer: 1, Plz: 12345, Ort: „Musterhausen“


Datenbankdesign

Normalformen

2te Normalform:

- Tabelle muss in 1NF vorliegen
- keine partiellen Abhängigkeiten von einem zusammengesetzten Primärschlüssel → alle nicht-schlüssel Attribute müssen vom gesamten Schlüssel abhängig sein

Partielle Abhängigkeiten



BestellID	ProduktID	ProduktName	KundeName	BestellDatum	Menge
1	101	Apfel	Müller	2023-06-01	3
2	102	Banane	Meier	2023-06-02	2
1	102	Banane	Müller	2023-06-01	1
3	103	Orange	Schulz	2023-06-03	5

ProduktName ist partiell von ProduktID abhängig und KundenName partiell von BestellID

Datenbankdesign

Normalformen

2te Normalform

Auflösung:

Tabelle: Bestellungen

BestellID	KundeName	BestellDatum
1	Müller	2023-06-01
2	Meier	2023-06-02
3	Schulz	2023-06-03

Tabelle: BestellDetails

BestellID	ProduktID	Menge
1	101	3
1	102	1
2	102	2
3	103	5

Tabelle: Produkte

ProduktID	ProduktName
101	Apfel
102	Banane
103	Orange

← Zuordnungstabelle (ähnlich wie Warenkorb)

Datenbankdesign

Normalformen

3te Normalform:

- Tabelle muss in 2NF vorliegen
 - Tabelle darf keine transitiven Abhängigkeiten enthalten
- Wenn ein nicht-Schlüsselattribut von einem anderen nicht-Schlüsselattribut abhängig ist, welches vom Primärschlüssel abhängig ist

direkte Abhängigkeit		transitive Abhängigkeit	
MitarbeiterID	Name	Abteilung	Abteilungsleiter
1	Max Müller	Verkauf	Hans Meier
2	Anna Meier	Marketing	Julia Schmidt
3	Peter Schulz	Verkauf	Hans Meier
4	Laura Klein	IT	Karl Braun

Wenn Mitarbeiter Abteilung wechselt
würde Abteilungsleiter bestehen bleiben!
→ Änderungsanomalie

Datenbankdesign

Normalformen

3te Normalform

Auflösung:

Tabelle: Mitarbeiter

MitarbeiterID	Name	AbteilungsID
1	Max Müller	1
2	Anna Meier	2
3	Peter Schulz	1
4	Laura Klein	3

Tabelle: Abteilungen

AbteilungsID	Abteilung	Abteilungsleiter
1	Verkauf	Hans Meier
2	Marketing	Julia Schmidt
3	IT	Karl Braun

Datenbankdesign

Um Änderungsanomalien vorzubeugen, sollten Datenbanken mindestens nach der zweiten Normalform entworfen werden (3te Normalform optional aber empfohlen)!

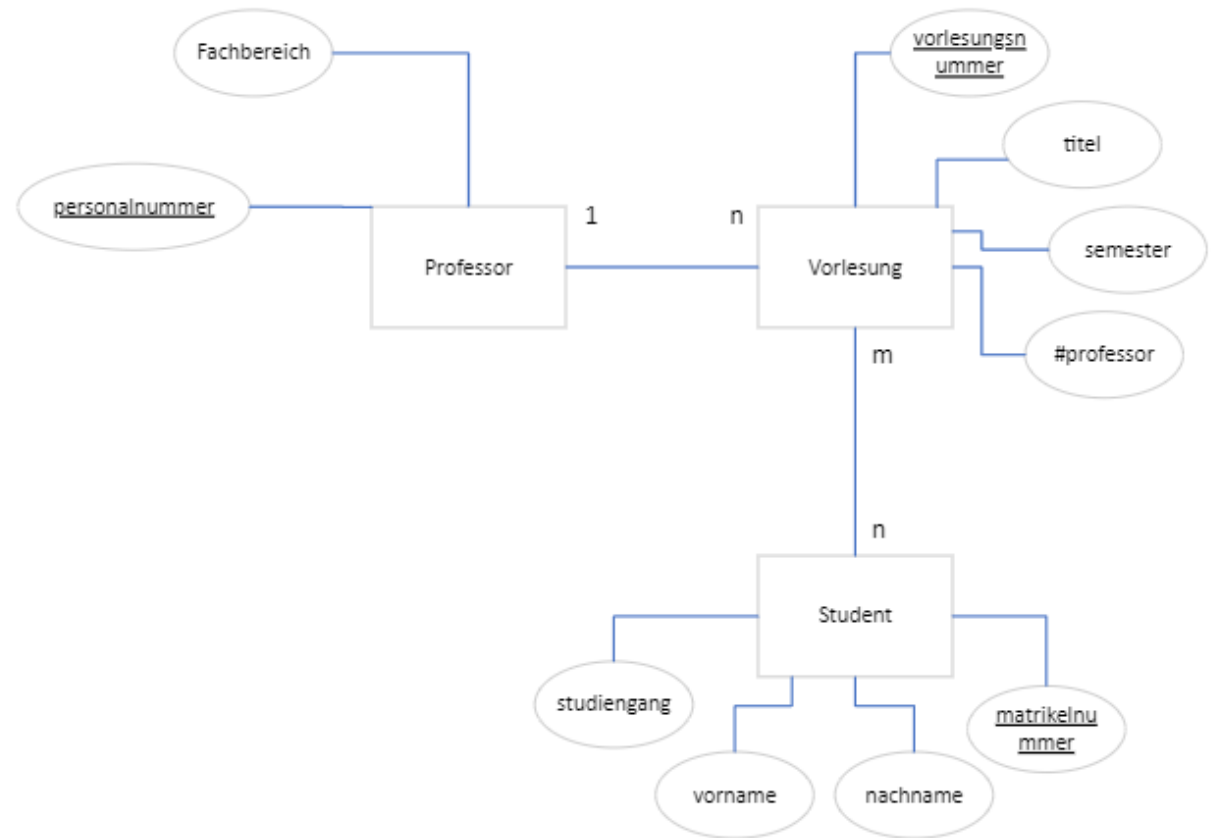
KundenID	KundenName	Adresse	BestellID	Bestelldatum	Produkte	Menge
1	Max Müller	Hauptstraße 1, Berlin	1001	2023-06-01	Apfel, Banane	3, 2
2	Anna Meier	Nebenweg 5, Hamburg	1002	2023-06-02	Orange	5
1	Max Müller	Hauptstraße 1, Berlin	1003	2023-06-03	Banane, Kirsche, Birne	1, 10, 4

Was sollte man an der obigen Datenbanktabelle wohl optimieren und warum?

ERM

Entity-Relationship-Model

- Modellierungstool für Datenbankschemata
- Enthält Entitäten, Relationen und Eigenschaften/Attribute
- Kardinalität bestimmt wie viele Beziehung eine Entität eingehen kann (1:1, 1:n oder n:m)
- Primärschlüssel und Fremdschlüssel können dargestellt werden



ERM

Notation

Entities:

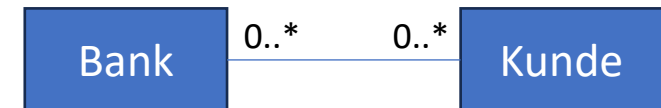
- Werden als Objekte modelliert, welche durch einen rechteckigen Rahmen erkennbar sind. Grundsätzlich wird immer die Einzahl benutzt. z.B.: Auto, Bank, Vorlesung etc...

Relationen:

- Sind Beziehungen, die eine Entität mit einer anderen Entität verbindet.
- Kardinalität bestimmt wie viele Beziehungen eine Entität mit einer anderen eingehen kann
- Hierfür wird entweder das Minimum und Maximum genannt (Min-Max Notation) oder nur das Maximum bezeichnet (Chen-Notation)



Min-Max Notation



Chen-Notation

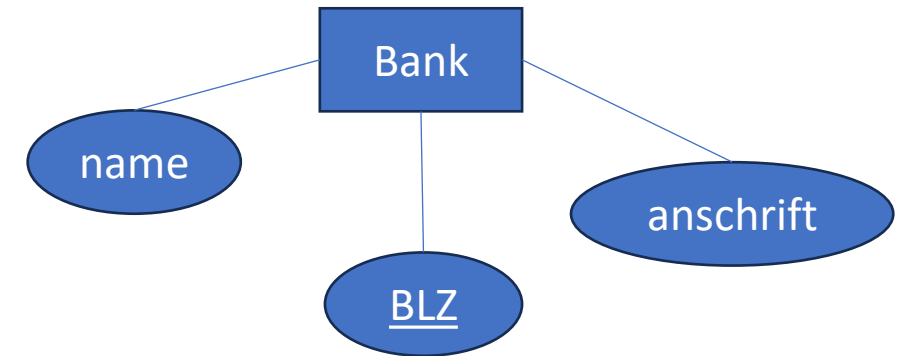


ERM

Notation

Eigenschaften/Attribute:

- Werden als Elipse mit Verbindung zur Entität modelliert
- Jedes Attribut steht für ein Feld in einer Datenbanktabelle
- Unterstrichene Attribute sind Primärschlüssel
- Attribute mit # davor sind Fremdschlüssel



Schema definieren mit PgAdmin

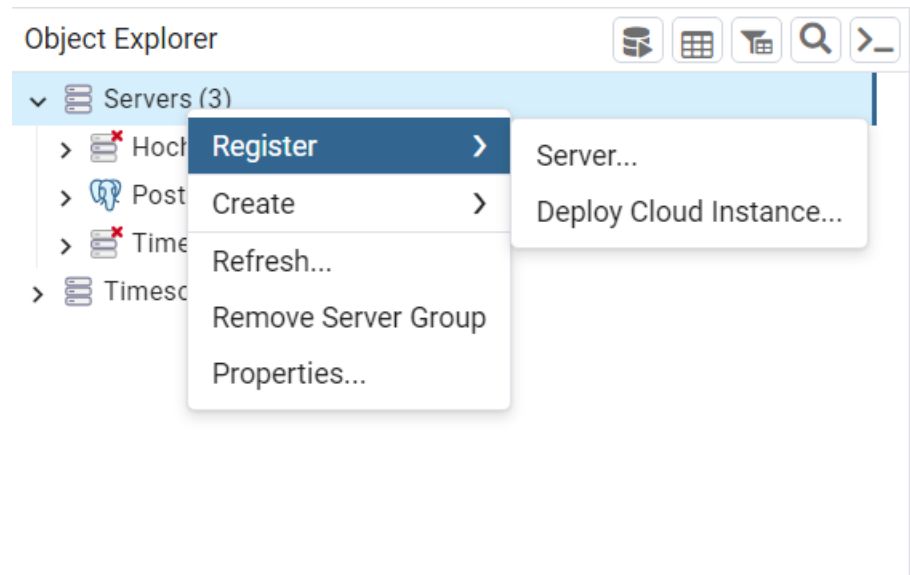
Schemadefinition

- Schemata sind voneinander abgegrenzte Bereiche, welche eigene Tabellen/Prozeduren etc. enthalten können, ohne andere Bereiche zu beeinflussen
- Schemata können einem User hinzugefügt werden
- Gut, wenn mehrere Personen über den gleichen User auf eine Datenbank zugreifen

Schematadefinition

PgAdmin

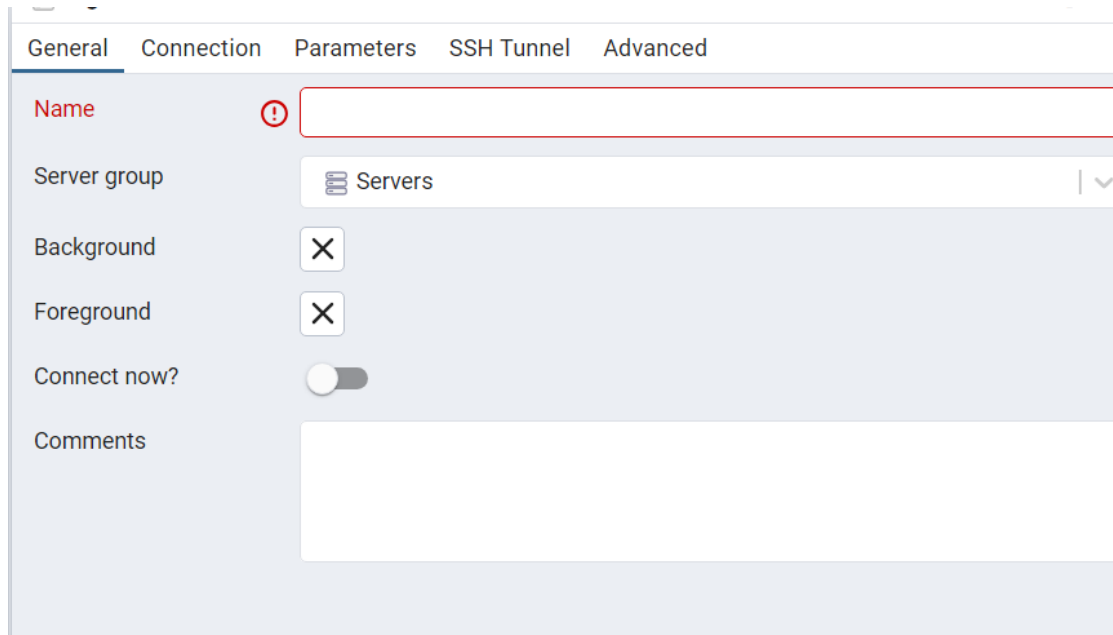
1. Öffnen Sie Ihr PgAdmin-Tool, indem Sie es entweder in der Suchfunktion suchen oder über das Desktopicon starten
2. Klicken Sie auf das Icon „Server“ mit Rechtsklick und wählen dann „Register“ → „Server“



Schematadefinition

PgAdmin

3. Geben Sie im folgenden Fenster einen frei gewählten Namen ein und vergewissern Sie sich, dass die Auswahl für „Connect now“ **deaktiviert** ist.



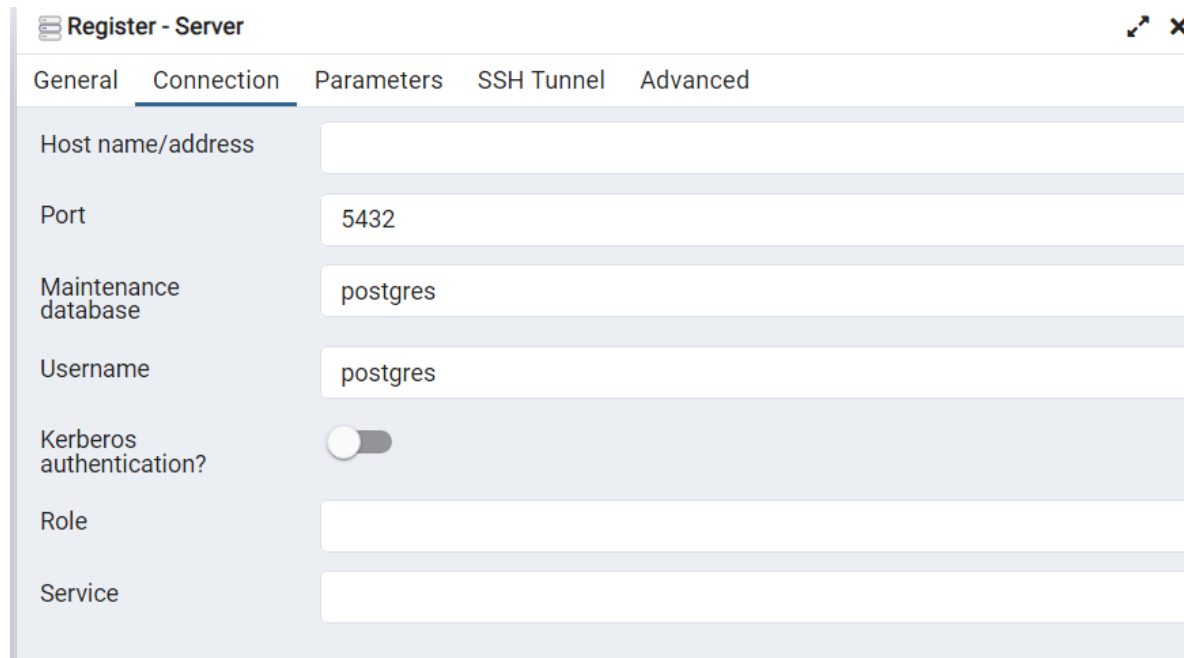
The screenshot shows the 'General' tab of the PgAdmin connection configuration window. The 'Name' field is empty and highlighted with a red border and a red warning icon. The 'Server group' dropdown is set to 'Servers'. The 'Background' and 'Foreground' options are both disabled, indicated by 'X' icons. The 'Connect now?' toggle switch is turned off. The 'Comments' field is empty.

Field	Value / State
Name	Empty (Warning icon)
Server group	Servers
Background	Disabled (X icon)
Foreground	Disabled (X icon)
Connect now?	Disabled (Toggle off)
Comments	Empty

Schematadefinition

PgAdmin

4. Wechseln Sie nun in den Tab „Connection“ und geben Sie dort IP/hostname des Postgresql Servers an erster Stelle ein. Der Standardport für Postgres-Installationen ist 5432 und muss deshalb meistens nicht geändert werden. Geben Sie bei Username den Benutzernamen an mit dem Sie sich auf der Datenbank anmelden wollen. Der defaultuser ist „postgres“.



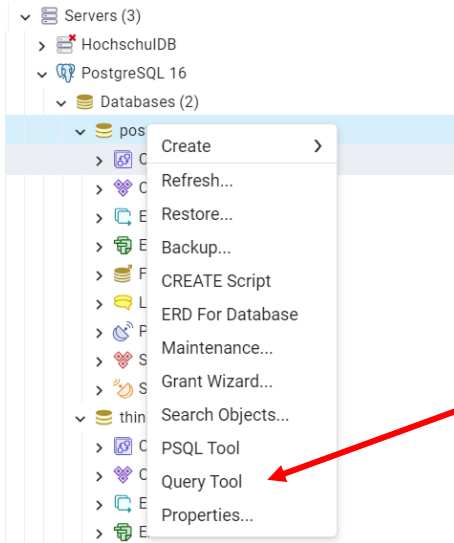
The screenshot shows the 'Register - Server' dialog box in PgAdmin, with the 'Connection' tab selected. The dialog has a title bar with a menu icon, the text 'Register - Server', and window control buttons. Below the title bar are five tabs: 'General', 'Connection', 'Parameters', 'SSH Tunnel', and 'Advanced'. The 'Connection' tab is active, showing a list of fields for server configuration. The fields and their values are: 'Host name/address' (empty), 'Port' (5432), 'Maintenance database' (postgres), 'Username' (postgres), 'Kerberos authentication?' (disabled toggle), 'Role' (empty), and 'Service' (empty).

Field	Value
Host name/address	
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	<input type="checkbox"/>
Role	
Service	

Schematadefinition

PgAdmin

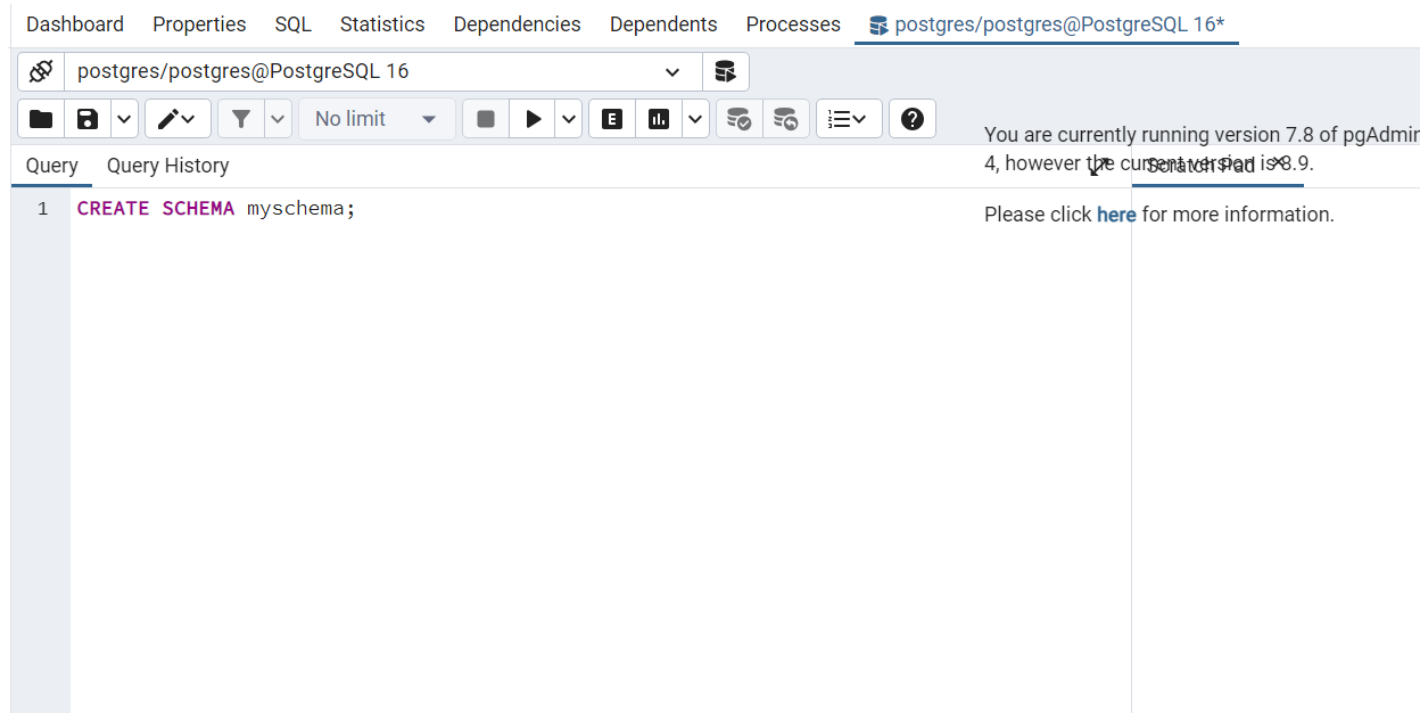
5. Klicken Sie unten rechts nun auf „Save“ und versuchen Sie sich mit dem Server zu verbinden, indem Sie mit „Rechtsklick“ auf das neue Servericon mit dem vergebenen Namen und dann auf „Connect to Server“ klicken
6. Sie sollten nun aufgefordert werden ein Passwort einzugeben. Geben Sie das passende Passwort zu dem erstellten User ein.
7. Klicken Sie mit Rechtsklick auf die „postgres“-Datenbank und wählen Sie „Query Tool“ aus



Schematadefinition

PgAdmin

8. Geben Sie nun in dem geöffneten Query Tool auf der rechten Seite den Befehl „CREATE SCHEMA vorname_nachname“ ein → Ersetzen Sie „vorname_nachname“ bitte durch Ihre Namen (vorname_nachname)

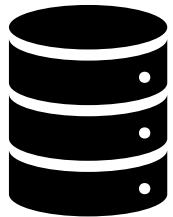
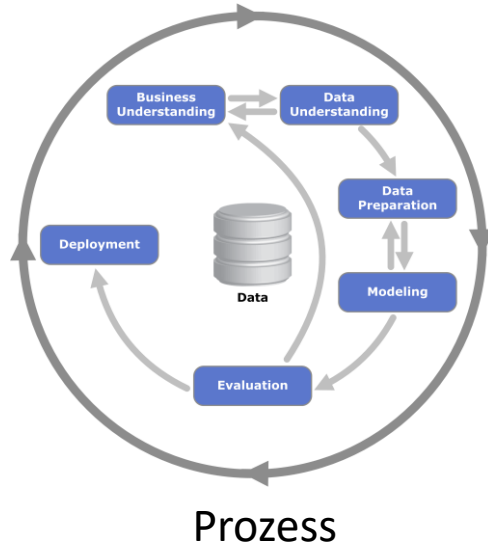


sqlAlchemy mit Python

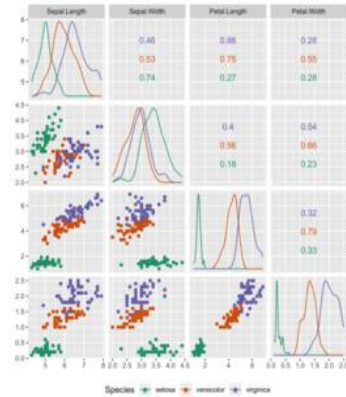
Exkurs Data Science

Einführung pandas / numpy

Was ist Data Science?



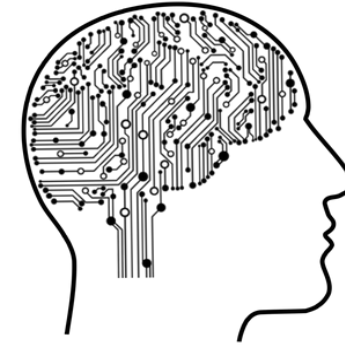
Daten



EDA

Data Science

...



Machine Learning / AI

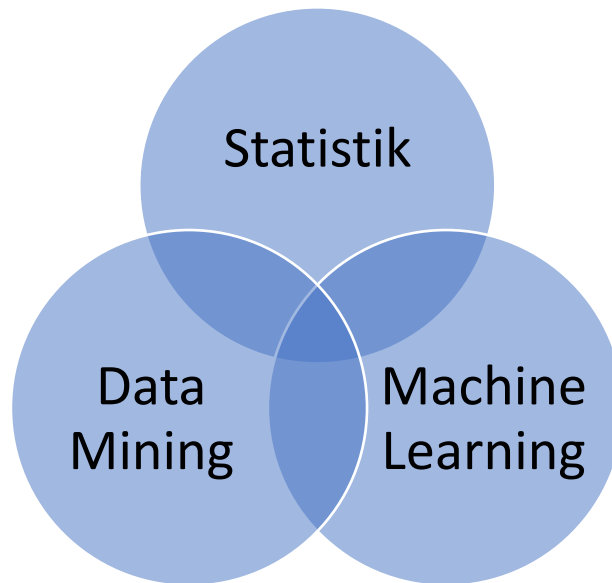


Analytics

Was ist Data Science

Data Science Definition

„...Extraktion nützlichen Wissens und aussagekräftiger Informationen aus großen Datenmengen, um geschäftliche Entscheidungsfindungen zu verbessern.“¹



Hauptgebiete von Data Science, welche nicht eindeutig voneinander separierbar sind und sich teils überlappen.

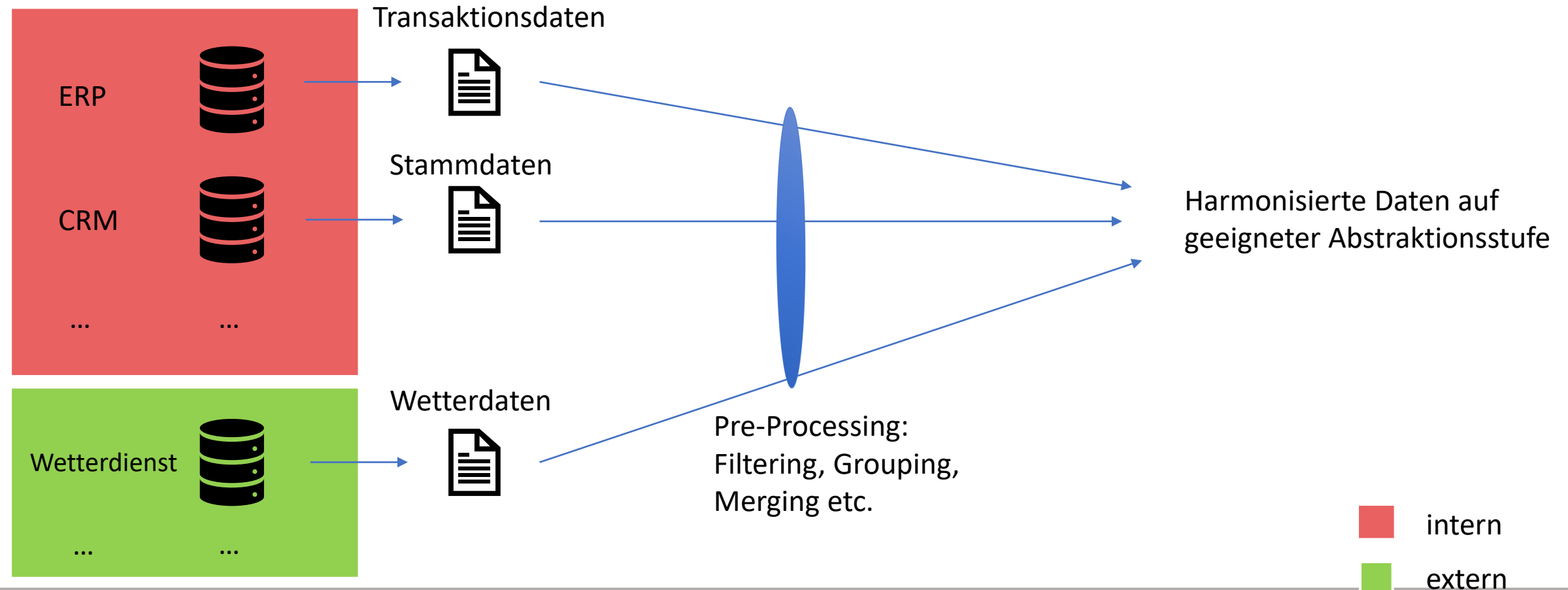
Ausblick Data Science mit Python

- Grouping
- Merging
- Filtering
- Sorting

Warum Merging/Grouping/Filtering...?

- Daten werden selten (eher nie) in geeigneter Abstraktionsstufe ausgeliefert
- Aufgabe des Data Scientist ist mit Auswahl geeigneter Methoden passende Datenform für spätere Analyse zu wählen
- Auch für EDA kann eine höhere/niedrigere Abstraktionsstufe größeren Einblick in die vorliegenden Daten geben
- Teil des Pre-Processing neben anderen notwendigen Schritten wie beispielsweise der Umgang mit Fehlwerten.

Warum Merging/Grouping/Filtering...?



Beispiel Corona-Daten

IdBundesland	Bundesland	Landkreis	Altersgruppe	Geschlecht	AnzahlFall	AnzahlTode...	Meldedatum	IdLandkreis	Datenstand	NeuerFall	NeuerTodes...	Refdatum
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Sep 30, 2020	1001	06.04.2021, 0...	0	-9	Sep 30, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Oct 29, 2020	1001	06.04.2021, 0...	0	-9	Oct 29, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Nov 3, 2020	1001	06.04.2021, 0...	0	-9	Nov 3, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Nov 20, 2020	1001	06.04.2021, 0...	0	-9	Nov 19, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Nov 23, 2020	1001	06.04.2021, 0...	0	-9	Nov 18, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Dec 18, 2020	1001	06.04.2021, 0...	0	-9	Dec 14, 2020
1	Schleswig-H...	SK Flensburg	A00-A04	M	2	0	Jan 6, 2021	1001	06.04.2021, 0...	0	-9	Jan 6, 2021
1	Schleswig-H...	SK Flensburg	A00-A04	M	1	0	Jan 8, 2021	1001	06.04.2021, 0...	0	-9	Jan 6, 2021

→ Daten auf niedrigster Abstraktionsebene sind oftmals nicht geeignet für prädiktives Modell

Beispiel Corona Daten

Problemstellung:

Extraktion von Features für prädiktives Modell zur Vorhersage von Knappheit/Engpässen für Intensivbetten pro Stadt/Landkreis

- Sie wollen die Todesrate (prozentual) und die aktuelle ICU-Bettenauslastung (prozentual) bestimmen.

Wie sieht die Formel zur Berechnung dieser aus?

Formel Todesrate:

Formel ICU-Bettenauslastung:

Data Science mit Python

Feedback & Fragerunde

- Haben Sie Fragen zu den behandelten Themen?
- Welche Themen haben Sie vermisst?
- Feedback an mich?

Ihr Ansprechpartner



Timm Gieger

mobile: +49176 40566154

mail: tim.gieger@geekit-ds.de