Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Übungsklausur

Name:
Vorname:
Matrikel-Nr:

Maximale Bonuspunkte: 10
Punktzahl der Klausur: 100
Zum Bestehen sind 50 Punkte notwendig.

Name:

+ Möglichkeiten zur Bearbeitung der Übungsklausur
- Sie beantworten die Fragen direkt im PDF-Dokument
- Sie beantworten die Fragen auf Papier

Name:

1) Welche Eigenschaften hat Big Data?

+ 2 Punkte

siehe VL1, Folie 5

2) Wie wird Smart Data definiert?

+ 2 Punkte

siehe VL1, Folie 5

Name:

3) Genauigkeit der Zeitstempel

- \+ 3 Punkte `siehe VL1, Folie 8`
- \+ python datetime:
- \+ pandas:
- \+ numpy:

Name:

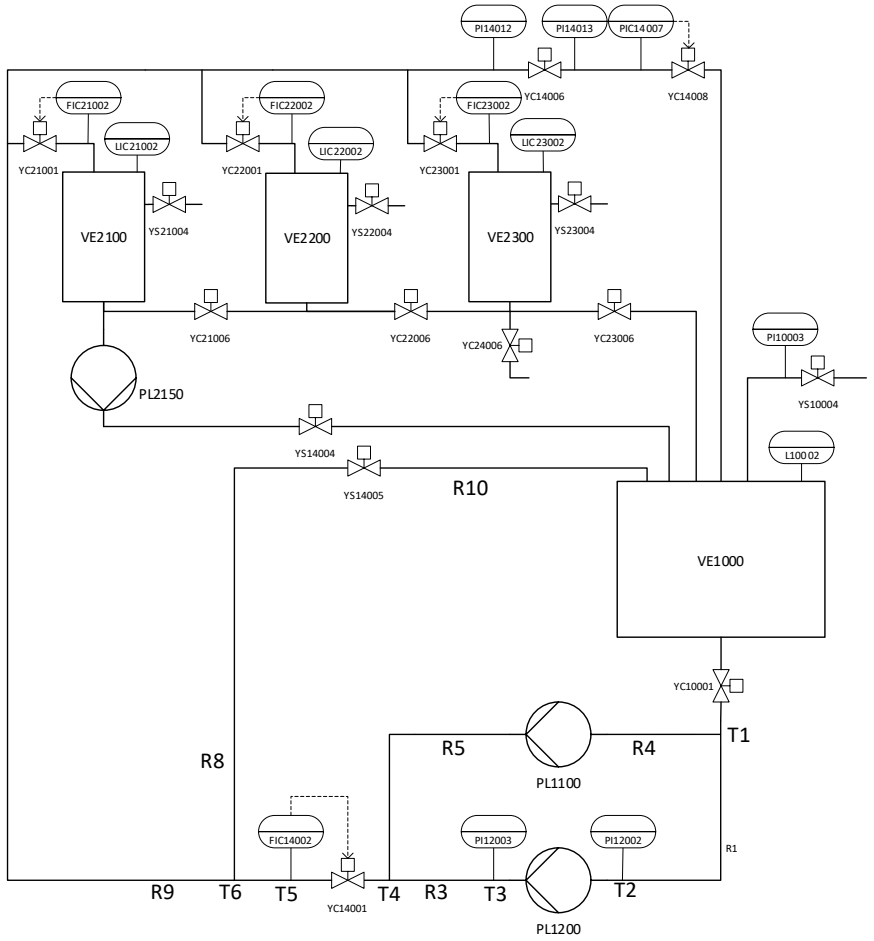
4) Graph-Datenbank

kürzester Weg von VE1000 nach YC14001,
also
VE1000->YC1001->T1->R4->PL1100->R5->T4->YC14001

+ 4 Punkte

+ Welche Ausgabe wird durch folgendes Skript erzeugt:

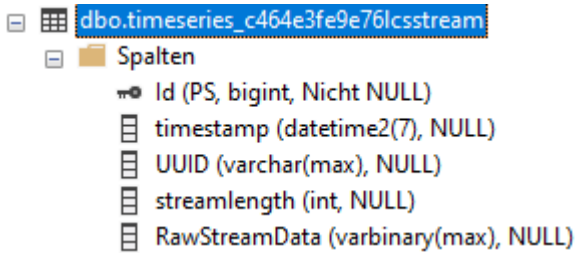
```
SELECT AKZ1, ConnectedAssets
FROM (
SELECT
asset1.AKZ AS AKZ1,
STRING_AGG(asset2.AKZ, '->') WITHIN GROUP (GRAPH PATH) AS ConnectedAssets,
LAST_VALUE(asset2.AKZ) WITHIN GROUP (GRAPH PATH) AS LastNode
FROM
Assets AS asset1,
connectedTo FOR PATH AS cTo,
Assets FOR PATH  AS asset2
WHERE MATCH(SHORTEST_PATH(asset1(-(cTo)->asset2)+))
AND asset1.AKZ = 'VE1000'
) AS Q
WHERE Q.LastNode = 'YC14001'
```



Name:

5) Auslesen und Umwandeln von Binärdaten aus einem SQL-Server

- \+ 4 Punkte
- \+ In einer Tabelle sind Messwerte (RawStreamData) eines IIoT-Sensors im Binärformat gespeichert. Der Sensor erzeugt bei jeder Messung 512 Messwerte im Datenformat Int16. Diese 512 Messwerte werden binär als Stream in der Spalte [RawStreamData] gespeichert.



- \+ Aus wie vielen Bytes besteht ein Stream?

```
Ein Messwert hat das Datenformat Int16. Int16 besteht aus 16 Bit, also 2 Bytes.
Ein Stream besteht aus 512 Messwerten->
512 * 2 Bytes = 1024 Bytes
```

- \+ Schreiben Sie ein SQL-Statement zum Auslesen einer beliebigen Zeile aus der Tabelle [timeseries_c464e3fe9e76lcsstream] mit anschließender Umwandlung der Binärdaten in ein Array von Messwerten.

```
SQL-Statement zum Auslesen einer beliebigen Zeile:
SELECT TOP 1 RawStreamData
FROM timeseries_c464e3fe9e76lcsstream

Umwandeln der Binärdaten (Variable rawdata):
values=numpy.frombuffer(rawdata,numpy.int16)
```

Name:

6) SPARQL

+ 2 Punkte
+ Welche Informationen werden durch folgende Abfrage ausgegeben?
- Q11344: chemical element
- P1086: atomic number

```
SELECT ?element ?elementLabel ?symbol ?number
WHERE
{
    ?element wdt:P31 wd:Q11344;
             wdt:P246 ?symbol;
             wdt:P1086 ?number.
    SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE]". }
}
ORDER BY ?number
```

Es wird eine Liste aller chemischen Elemente geordnet nach der Ordnungszahl ausgegeben.

Name:

7) Beschreiben Sie die MQTT Architektur

+ 4 Punkte

```
Pub/Sub-Architektur:
Architektur mit Client / Server (Broker)
Clients senden (publishen) Nachrichten zu bestimmten Topics an den Server.
Clients können sich zu Topics subscriben.
Server verteilt empfangene Nachrichten an alle Clients, die subscribed sind.
Last Will: Fällt ein Client aus, so schickt der Server eine Nachricht an das vorab bestimmte Topic.
Authentifizierung möglich
Verschlüsselung über TLS möglich
keine Semantik
```

Name:

8) AMQP

- \+ 4 Punkte
- \+ Folgender (unvollständiger) Code implementiert das Empfangen einer Nachricht mit RabbitMQ
- \+ Was bewirkt die Zeile „ch.basic_ack(delivery_tag=method.delivery_tag)”?

Empfänger schickt ein Acknowledgement an den Server.
Erhält der Server innerhalb eines festgelegten Timeouts kein Acknowledgement, so wird die Nachricht an den nächsten Empfänger geschickt, bis genau ein Empfänger ein Acknowledgement geschickt hat.

```
[…]
cnxn= pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor=cnxn.cursor()
connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))
channel = connection.channel()

channel.queue_declare(queue='task_queue', durable=True)
print(' [*] Waiting for messages. To exit press CTRL+C')

def callback(ch, method, properties, body):
    print(" [x] {} Received {}".format(datetime.datetime.now(),body.decode()))
    # save work to SQL database
    try:
       cursor.execute("INSERT INTO MessageFromAMQP(timestamp,message) VALUES (?,?)",timestamp,body.decode("UTF-8"))
       cnxn.commit()
       ch.basic_ack(delivery_tag=method.delivery_tag)
       print(" [x] {} Saved message {} to SQL server.".format(datetime.datetime.now(),body.decode("UTF-8")))
    except:
       print("Insert to SQL Server failed.")

# give one message to a worker at a time, i.e. don't dispatch a new message to a worker until it has processed and acknowledged the previous one
channel.basic_qos(prefetch_count=1)
channel.basic_consume(queue='task_queue', on_message_callback=callback)
channel.start_consuming()
```

Name:

9) Vergleichen Sie MQTT, AMQP und OPC UA

\+ 6 Punkte

`siehe VL4, Folie 3`

Name:

10) Beschreiben Sie die OPC UA-Struktur

\+ 3 Punkte

`siehe VL4, Folie 6`

Name:

11) ETL

- \+ 8 Punkte
- \+ In welche Teilprozesse kann der ETL-Prozess unterteilt werden? `siehe VL5, Folie 4`
- \+ Wozu dienen die einzelnen Teilprozesse? `siehe VL5, Folie 5, 6, und 7`

Name:

12) Airflow - DAG

+ 3 Punkte
+ In Airflow können z.B. ETL-Prozesse mit Directed Acyclic Graphs (DAGs) beschrieben werden.
+ In einem DAG werden drei Tasks t1, t2 und t3 definiert.
+ Was bedeutet der Befehl t1 >> [t2,t3]

```
Zuerst wird der Befehl t1 ausgeführt.
Danach werden die Befehle t2 und t3 parallel ausgeführt.
Anmerkung: Ob die Ausführung tatsächlich parallel erfolgt, hängt von der Konfiguration von Airflow
und der Systemauslastung ab.
```

Name:

13) Airflow - Parallele Ausführung der Tasks / Remote Execution

+ 3 Punkte
+ Welche Möglichkeiten bestehen mit Airflow, Tasks parallel bzw. remote ausführen zu lassen?

```
Airflow bietet verschiedene Executor an:
- Local Executor
- Kubernetes Executor
- Celery Executor
- CeleryKubernetes Executor
```

Name:

14) Verteilte Web-Anwendung – Fehlerbehandlung

+ 8 Punkte
+ In einer verteilten Web-Anwendung speichert ein Microservice das Ergebnis einer Modell-Berechnung in einer SQL-Datenbank:

```
def gpmodel_todb(config,function_id:int,fnc:str,rmse:float):
    drv = pyodbc.drivers()
    DRIVER_NAME=drv[3]
    SQL_STR="Driver={"+DRIVER_NAME+"};SERVER="+config['SERVER']+';DATABASE='+config['DBGP']+';UID='+config['UID']+';PWD='+config['PWD']
    cnxn=pyodbc.connect(SQL_STR,autocommit=True)
    cnxn.cursor().execute("UPDATE gpresult SET [scheduled]=?,[ready]=?,[function]=?,[root_mean_squared_error]=? WHERE [FunctionId]=?",\
      False,True,fnc,rmse,function_id)
    cnxn.close()
```

+ Zählen Sie mindestens zwei mögliche Fehlerfälle auf und beschreiben Sie, wie diese behandelt werden können.

1) Die Eingabeparameter können None sein -> Überprüfung auf None.
2) Die Verbindung zum SQL-Server schlägt fehl -> erneut versuchen die Verbindung aufzubauen; nach einer bestimmten Anzahl von Fehlschlägen eine Ausnahme auslösen.
3) Der UPDATE-Befehl schlägt fehl, weil keine Zeile mit der FunctionId "function_id" existiert -> Vor dem UPDATE-Befehl prüfen, ob eine entsprechende Zeile vorhanden ist.

Name:

15) Verteilte Web-Anwendung – Cache

+ 4 Punkte
+ Wie kann ein In-Memory-Cache bzw. ein Distributed Cache die Performance einer Web-Anwendung steigern?

Komplexe Anfragen eines Users können z.B. vom Backend-for-Frontend (BFF) im In-Memory-Cache oder im Distributed Cache gespeichert werden.
Für die Art der Anfrage wird ein Hash gebildet, um die Anfrage eindeutig zuordnen zu können.
Das BFF bildet aus der Anfrage einen Hash, schaut im Cache nach, ob die Antwort existiert. Falls ja, kann die Antwort direkt aus dem Cache ausgelesen und an den Users zurückgegeben werden.

Name:

16) gRPC

+ 4 Punkte
+ Welche API-Varianten bietet gRPC?

siehe VL7, Folie 8

Name:

17) Web-Architektur mit Microservices

\+ 12 Punkte

A) Beschreiben Sie die Web-Architektur mit Microservices und gehen Sie dabei auf Unterschiede zum monolithischen Ansatz ein.

B) Wie können die Grenzen von Microservices bestimmt werden?

C) Welche Möglichkeiten gibt es, den Grad der Kopplung von Microservices zu bewerten?

zu A) Jeder Microservice implementiert eine end-to-end-domain Fähigkeit (capability) innerhalb einer context boundary
Jeder Service kann autonom entwickelt und unabhängig ausgeführt werden.
Jeder Microservice bestimmt über seine Daten
Im Gegensatz dazu werden beim monolithischer Ansatz verschiedene Schichten (tier) implementiert, die voneinander abhängig sind, z.B. gibt es eine Datenbank für das gesamte System, in dem alle Daten gespeichert sind.

zu B) Die Grenzen können mittels Bounded Context ermittelt werden.

zu C) Ermittlung der Abhängigkeiten des Microservices zu anderen Services und Ermittlung der Abhängigkeiten anderer Services zu diesem Microservice.

Name:

17) Web-Architektur mit Microservices (Fortsetzung)

Name:

18) Data Warehouse

+ 24 Punkte
+ Ein Data Warehouse für eine Spezialchemie-Anlage hat
- fünf Dimensionen: Lieferant, Kunde, Herstellungsdatum, Produkttyp, Qualitätsstufe (Ganze Zahl zwischen 1 und 3) und
- drei assoziierte numerische Werte (Measures): Anzahl produzierte Chargen, Produktionsdauer und produzierte Menge (Masse in Tonnen)
+ Erstellen Sie Queries für folgende Fragestellungen:

Queries: siehe 04-UeKlausur_A18_queries.sql

- Gesamte produzierte Menge nach Produkttyp im Jahr 2021
- Year-to-date produzierte Menge nach Produkttyp in den Jahren 2019 bis 2021 aufgeschlüsselt nach Monaten
- Produzierte Menge aufgeschlüsselt nach a) Quartalen, b) Kunde, c) Produkttyp und d) Qualitätsstufe

+ Welche Hierarchie(n) ist/sind in den Fragestellungen enthalten?

Explizit ist in den Fragestellungen die Hierarchie Jahr -> Quartal -> Monat enthalten.

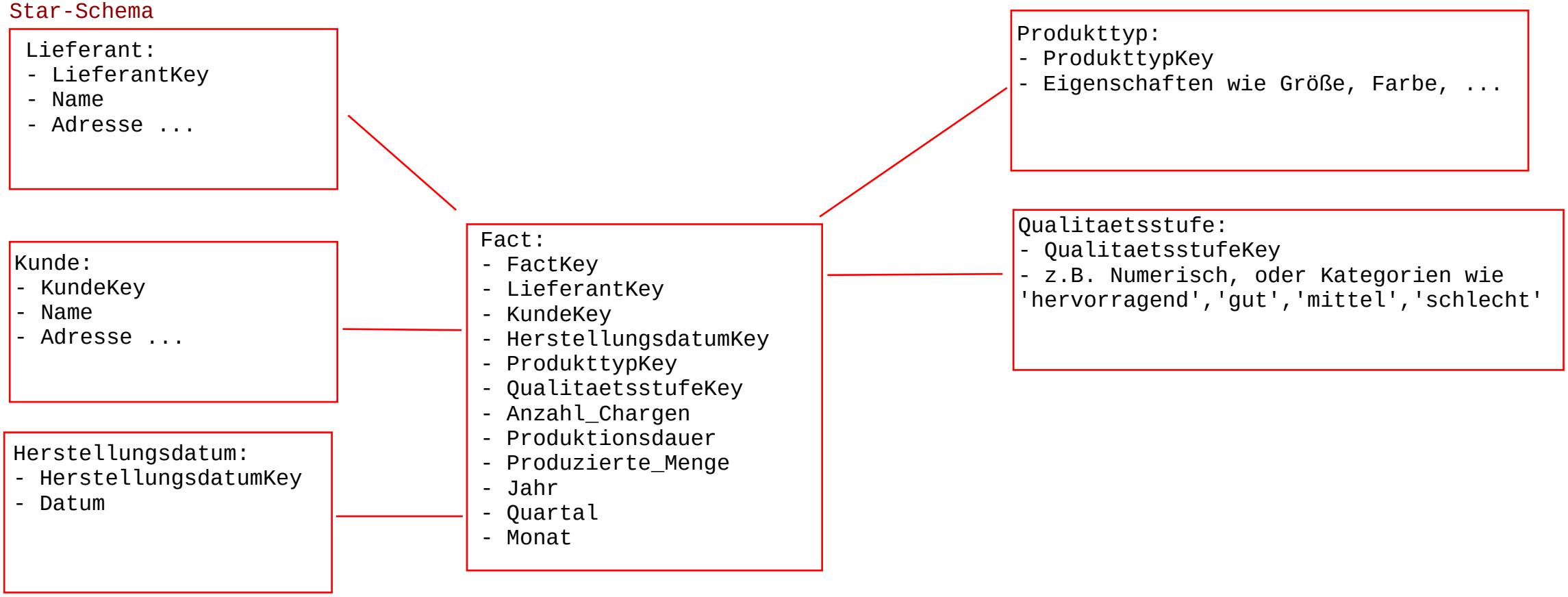
+ Zeichnen Sie ein Star-Schema für das Data Warehouse.

siehe nächste Seite

Name:

18) Data Warehouse (Fortsetzung 1)

Star-Schema



Name:

18) Data Warehouse (Fortsetzung 2)