Datenmanagement, Vorlesung @DHBW Übung

Andreas Buckenhofer

Daimler TSS

Ein Unternehmen der Daimler AG

Andreas Buckenhofer

Senior DB Professional

Since 2009 at Daimler TSS
Department: Performance Management
Business Unit: Vehicle Platforms

vcard

Contact/Connect

xing

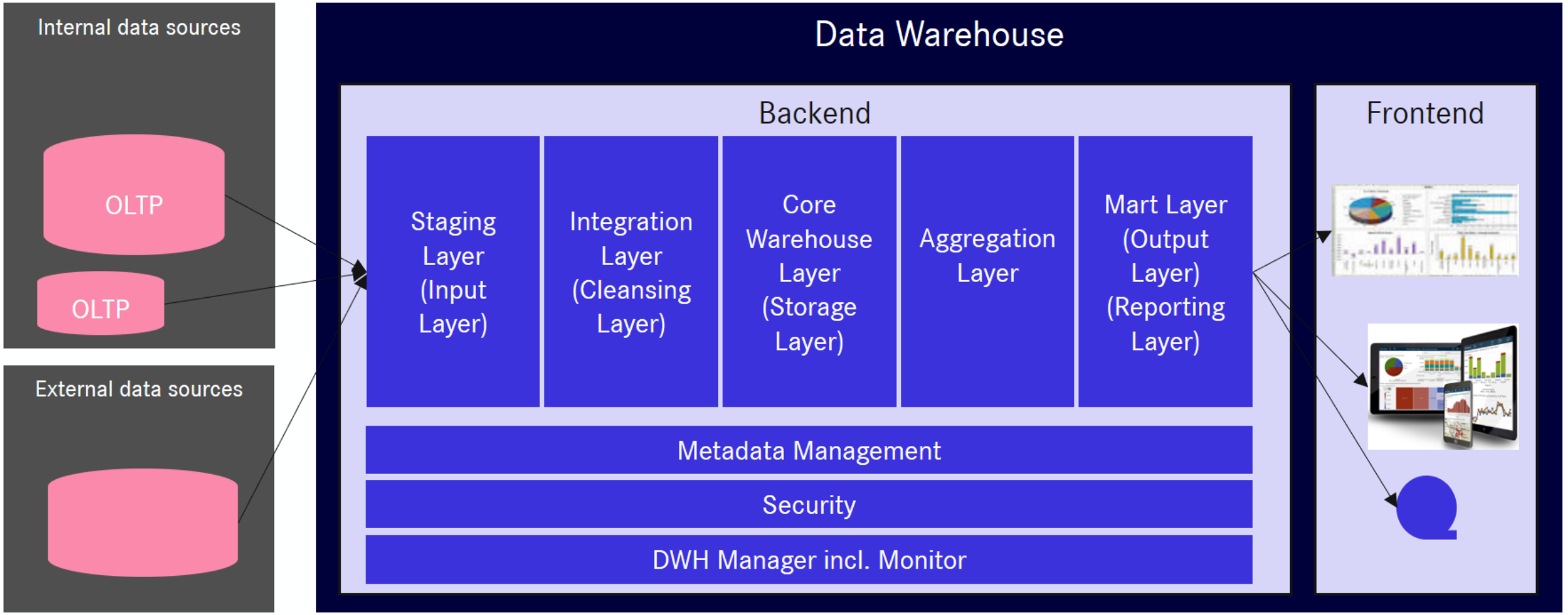
DOAG

Blog

- Over 20 years experience with database technologies
- Over 20 years experience with data-intense applications
- International project experience

- Oracle ACE
- DOAG delegate and DBC member
- Lecturer at DHBW
- Certified Data Vault Practitioner 2.0
- Certified Oracle Professional
- Certified IBM Big Data Architect

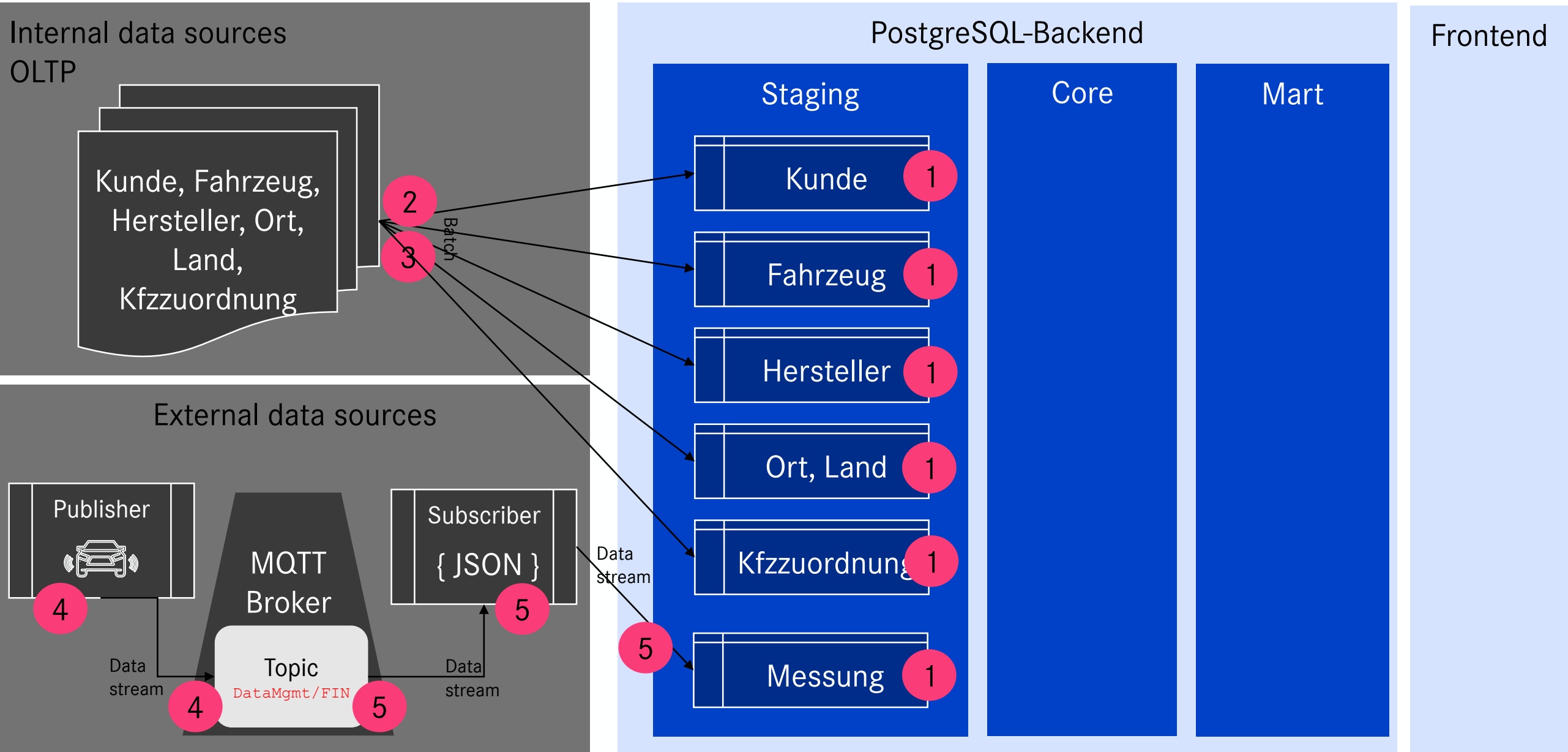
Standard DWH architecture



Fallstudie

- Erstellen Sie ein DWH aus internen und externen Daten
  - Interne Daten: Kunden, Fahrzeuge, u.ä
  - Externe Daten: Messdaten / Fahrzeugdaten
- Komponenten
  - Spaltenorientierte Datenbank (PostgreSQL mit cstore_fdw) zum Aufbau des DWH
  - Python, Javascript, SQL für Datenerzeugung, Datentransfer, Datentransformation
  - Batch: csv-Dateien (hier: SQL-Skripte) für interne Daten
  - Stream: Sensordaten, die über MQTT (Message Queue Telemetry Transport) übertragen werden, für externe Daten

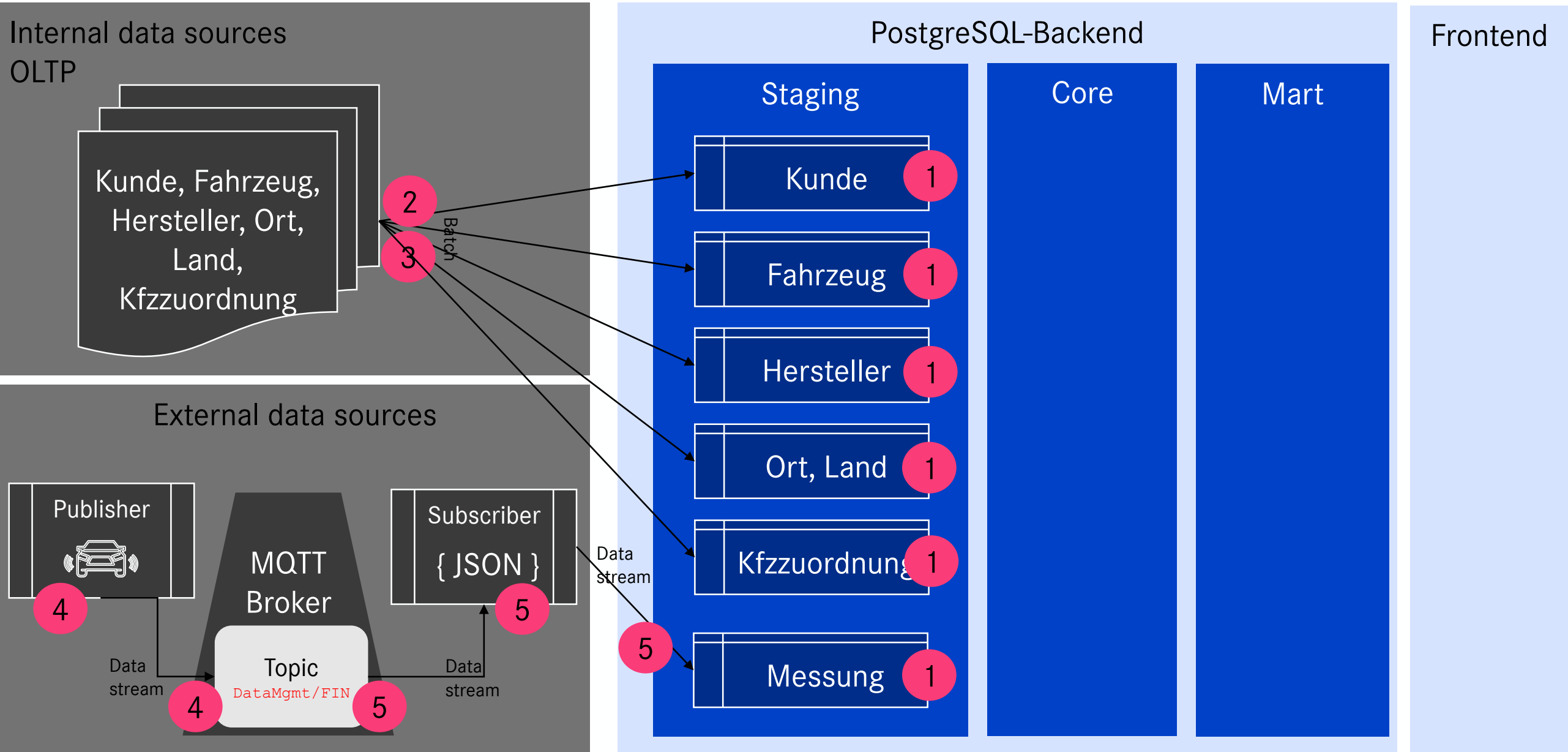
Architektur Fallstudie



Übung 1

1. Erstellen Sie die Tabellen im staging-Layer
2. Laden Sie die Daten in den staging Layer
3. Legen Sie Ihre eigenen Daten an und laden Sie diese in den staging-Layer
4. Erstellen Sie einen MQTT Publisher
5. Erstellen Sie einen MQTT Subscriber

Architektur Fallstudie



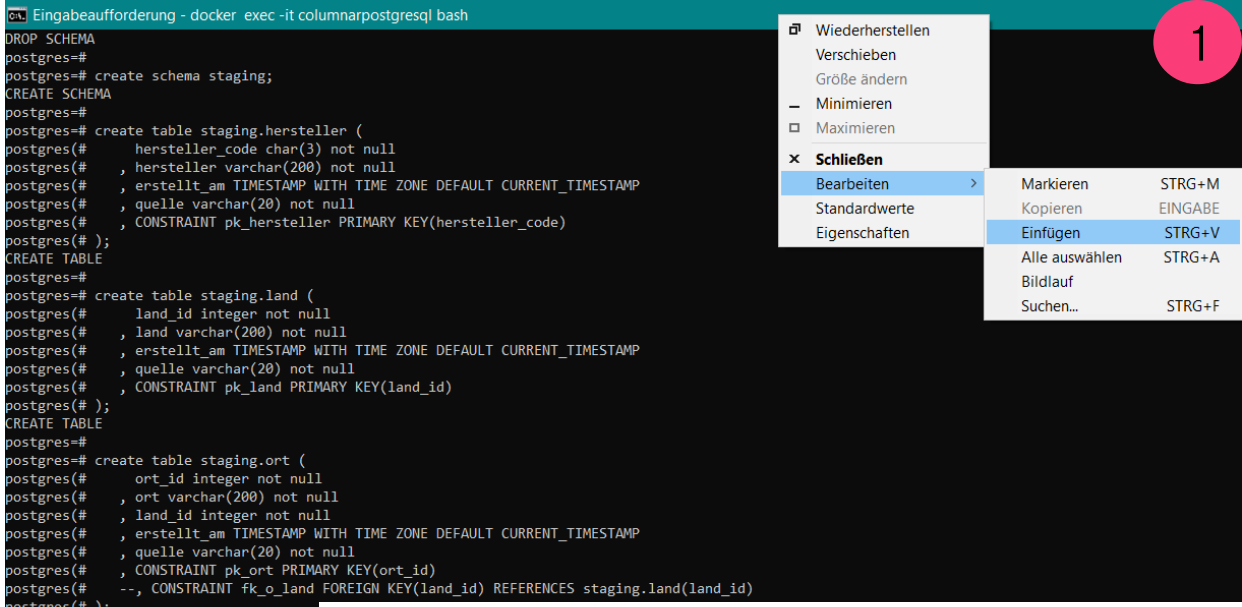
Tabellen im staging Layer erstellen

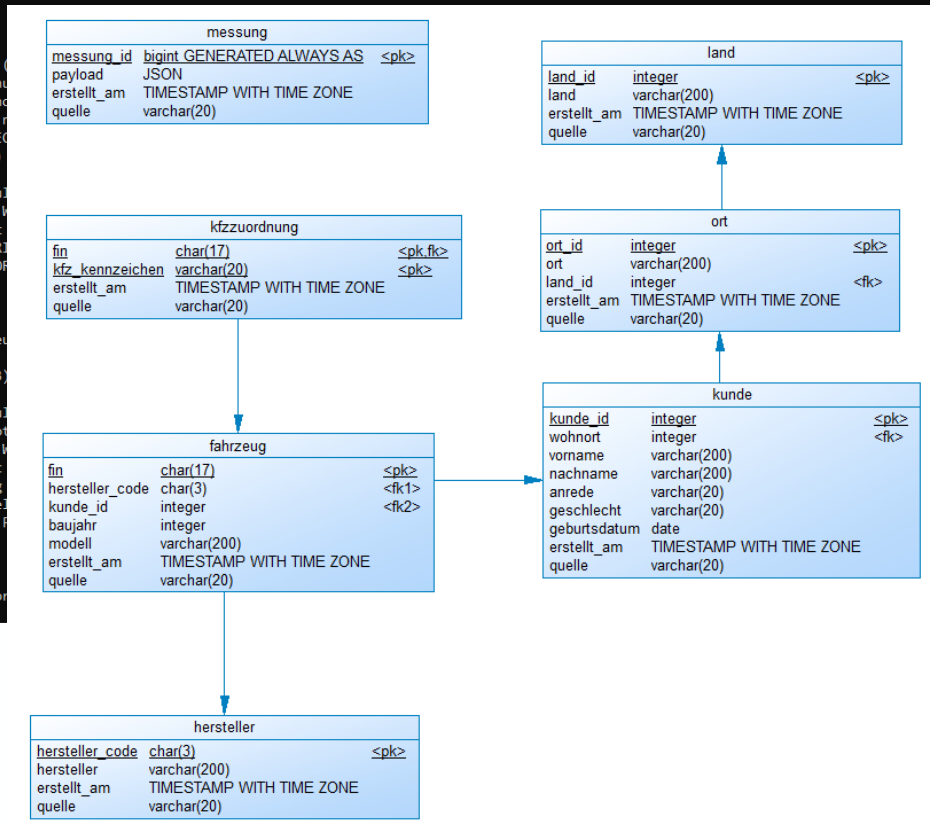
Führen Sie die zur Verfügung gestellten Skripte aus:

- 01_create_staging.sql

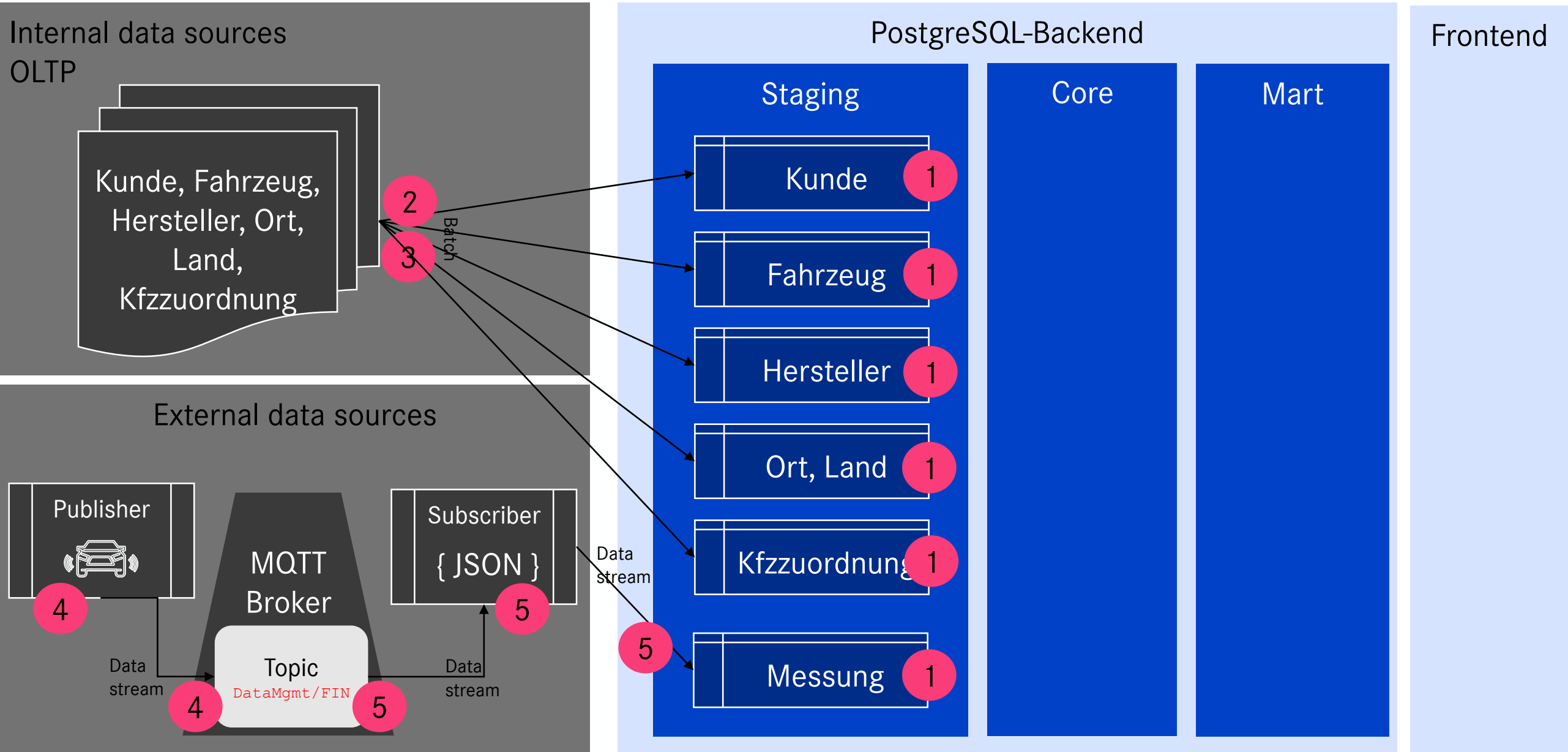
Kopieren Sie das Skript in den Container oder alternativ mittels copy & paste

Anmerkung: Die Skripte enthalten constraints. Üblicherweise werden keine constraints im staging-Layer angelegt. Für die Übung werden jedoch einige Constraints verwendet, um die erzeugten Daten zu validieren.





Architektur Fallstudie



Daten in den staging Layer laden

Führen Sie die zur Verfügung gestellten Skripte aus:

- 02_load_staging.sql

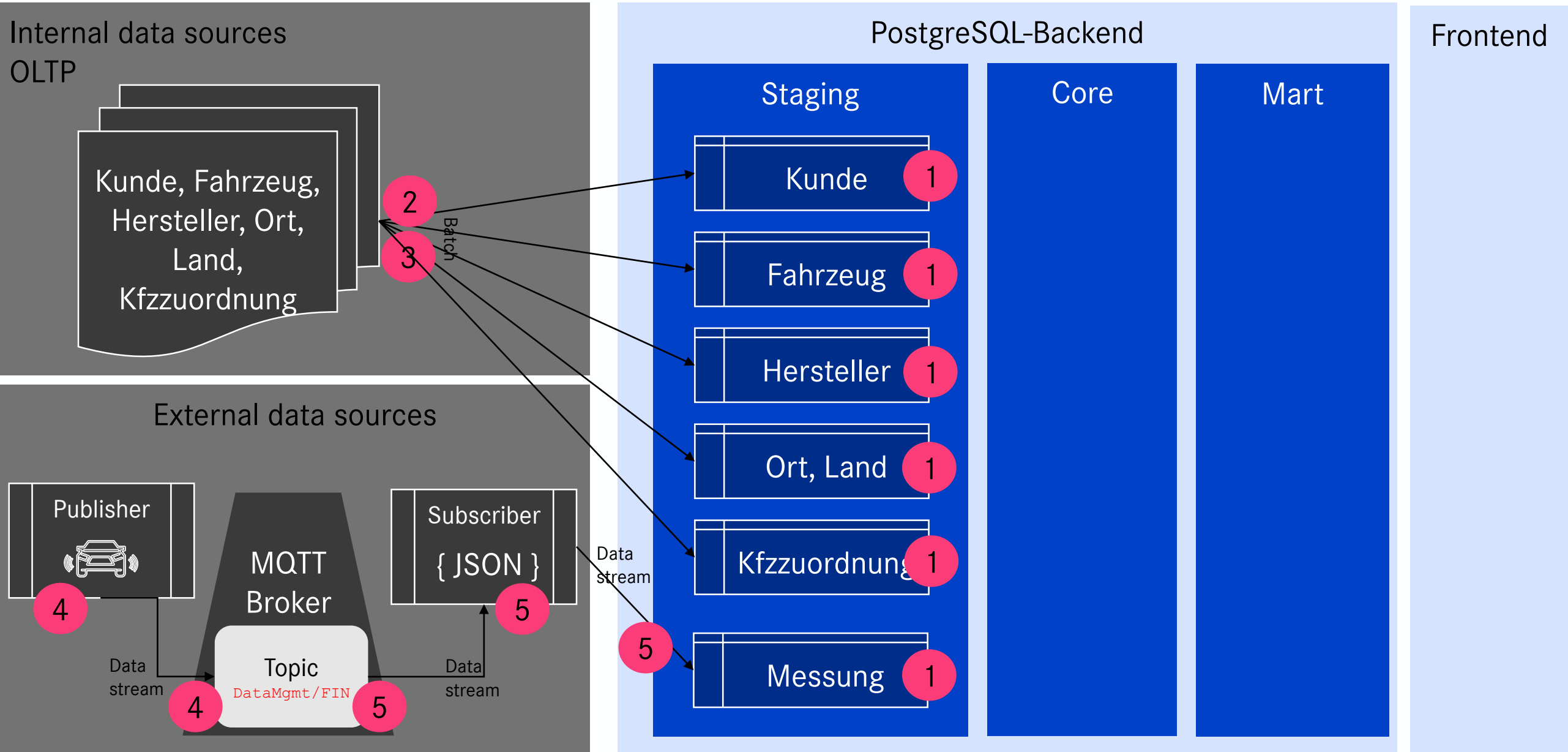
Anmerkung: üblicherweise werden csv-Dateien in den staging-Layer geladen. In der Übung werden insert-Befehle genutzt. csv-Dateien können mittels Bulk-Befehlen geladen werden für eine deutlich bessere Performanz.

Außerdem enthalten die Tabellen im staging-layer in der Übung bereits Datentypen. Sonst werden häufig varchar-Datentypen verwendet, um ggf. fehlerhafte Daten aus den csv-Dateien laden zu können.

2

```
insert into staging.hersteller (hersteller_code, hersteller, quelle)
  values
    ('WDD', 'Mercedes-Benz', 'WMI database')
    , ('WVW', 'Volkswagen', 'WMI database')
    , ('WBA', 'BMW', 'WMI database')
    , ('WB1', 'BMW Motorrad', 'WMI database')
    , ('1FM', 'Ford', 'WMI database')
    , ('ZFF', 'Ferrari', 'WMI database')
    , ('WMA', 'MAN', 'WMI database')
    , ('WAU', 'Audi', 'WMI database')
    , ('WPO', 'Porsche', 'WMI database')
    , ('SCC', 'Lotus', 'WMI database')
    , ('WOL', 'Opel', 'WMI database')
    , ('VF1', 'Renault', 'WMI database')
    , ('SNT', 'Sachsenring Automobilwerke Zwickau GmbH', 'WMI database')
;


insert into staging.land (land_id, land, quelle) values
    (101, 'Deutschland', 'Geo System')
    , (102, 'Österreich', 'Geo System')
    , (103, 'Mittelerde', 'Geo System')
    , (104, 'Schweiz', 'Geo System')
    , (105, 'China', 'Geo System')
;

insert into staging.ort (ort_id, ort, land_id, quelle) values
    (1, 'Stuttgart', 101, 'Geo System')
    , (2, 'München', 101, 'Geo System')
    , (3, 'Frankfurt', 101, 'Geo System')
    , (4, 'Türmli', 104, 'Geo System')
    , (5, 'Xian', 105, 'Geo System')
    , (6, 'Peking', 105, 'Geo System')
    , (7, 'Valmar', 103, 'Geo System')
    , (8, 'Númenor', 103, 'Geo System')
    , (9, 'Wien', 102, 'Geo System')
    , (10, 'Rot a.d. Rot', 100, 'Geo System')
;



insert into staging.kunde (kunde_id, vorname, nachname, anrede, geschlecht, geb
  values (171893, 'Darth', 'Vader', 'Herr', 'männlich', to_date('12.02.1990', '

insert into staging.fahrzeug (fin, kunde_id, baujahr, modell, quelle)
  values ('SNTU411STM9032150', 171893, 1985, 'Trabant 601 de luxe', 'Fahrzeug D

insert into staging.kfzzuordnung (fin, kfz_kennzeichen, quelle)
  values ('SNTU411STM9032150', 'UL-DV 111
```

Architektur Fallstudie



Eigene Daten anlegen und in den staging Layer laden

Legen Sie jeweils einen Datensatz für

- einen Kunden
- ein Fahrzeug
- eine Fahrzeugzuordnung

neu an.

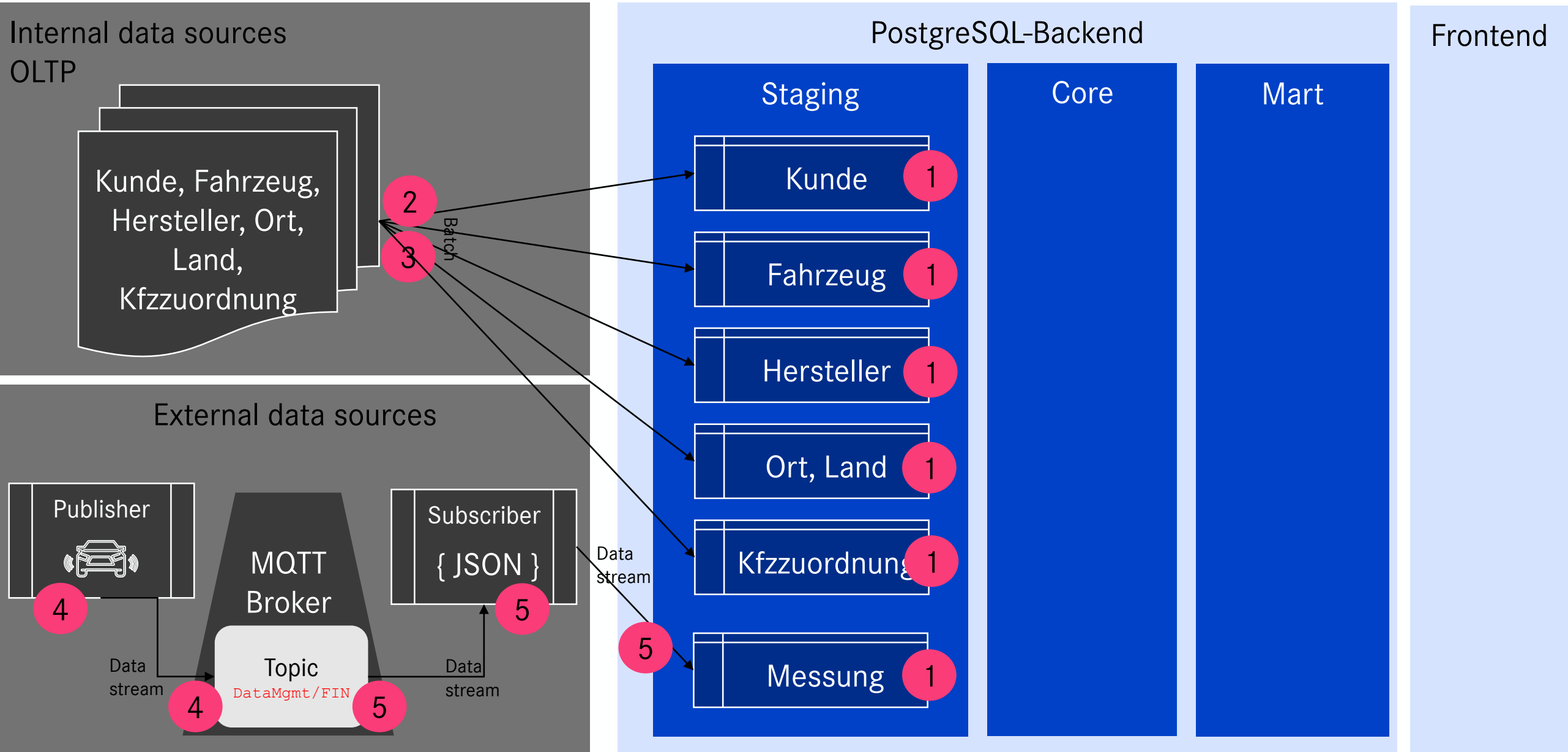
Erstellen Sie SQL Befehle nach dem unten aufgeführten Muster. Nutzen Sie eigene Daten.

Führen Sie die insert Befehle aus.

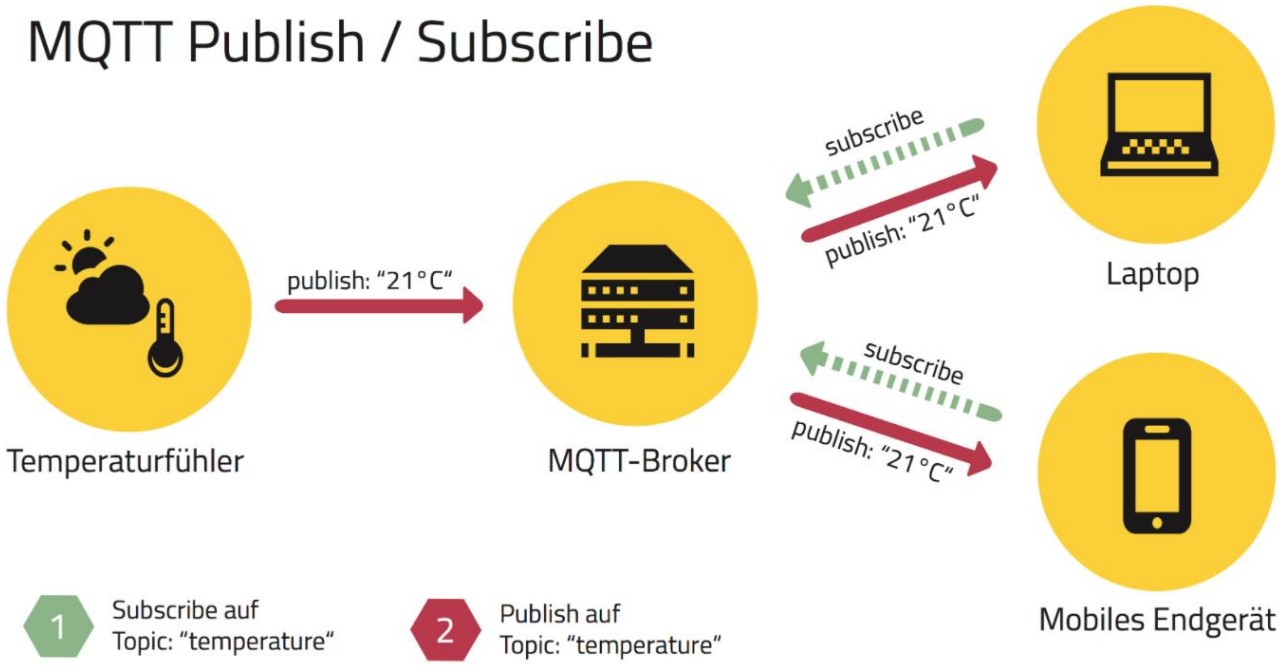
Laden sie die Befehle im Skript 03_staging.sql in moodle hoch!

```
insert into staging.kunde (kunde_id, vorname, nachname, anrede, geschlecht, geburtsdatum, wohnort, quelle)
  values (171893, 'Darth', 'Vader', 'Herr', 'männlich', to_date('12.02.1990', 'DD.MM.YYYY'), 7, 'CRM');

insert into staging.fahrzeug (fin, kunde_id, baujahr, modell, quelle)
  values ('SNTU411STM9032150', 171893, 1985, 'Trabant 601 de luxe', 'Fahrzeug DB');

insert into staging.kfzzuordnung (fin, kfz_kennzeichen, quelle)
  values ('SNTU411STM9032150', 'UL-DV 111', 'Fahrzeug DB');
```

Architektur Fallstudie



MQTT Protokoll im Internet der Dinge



Leichtgewichtiges Protokoll für den Transport von Nachrichten über ein unzuverlässiges Netzwerk

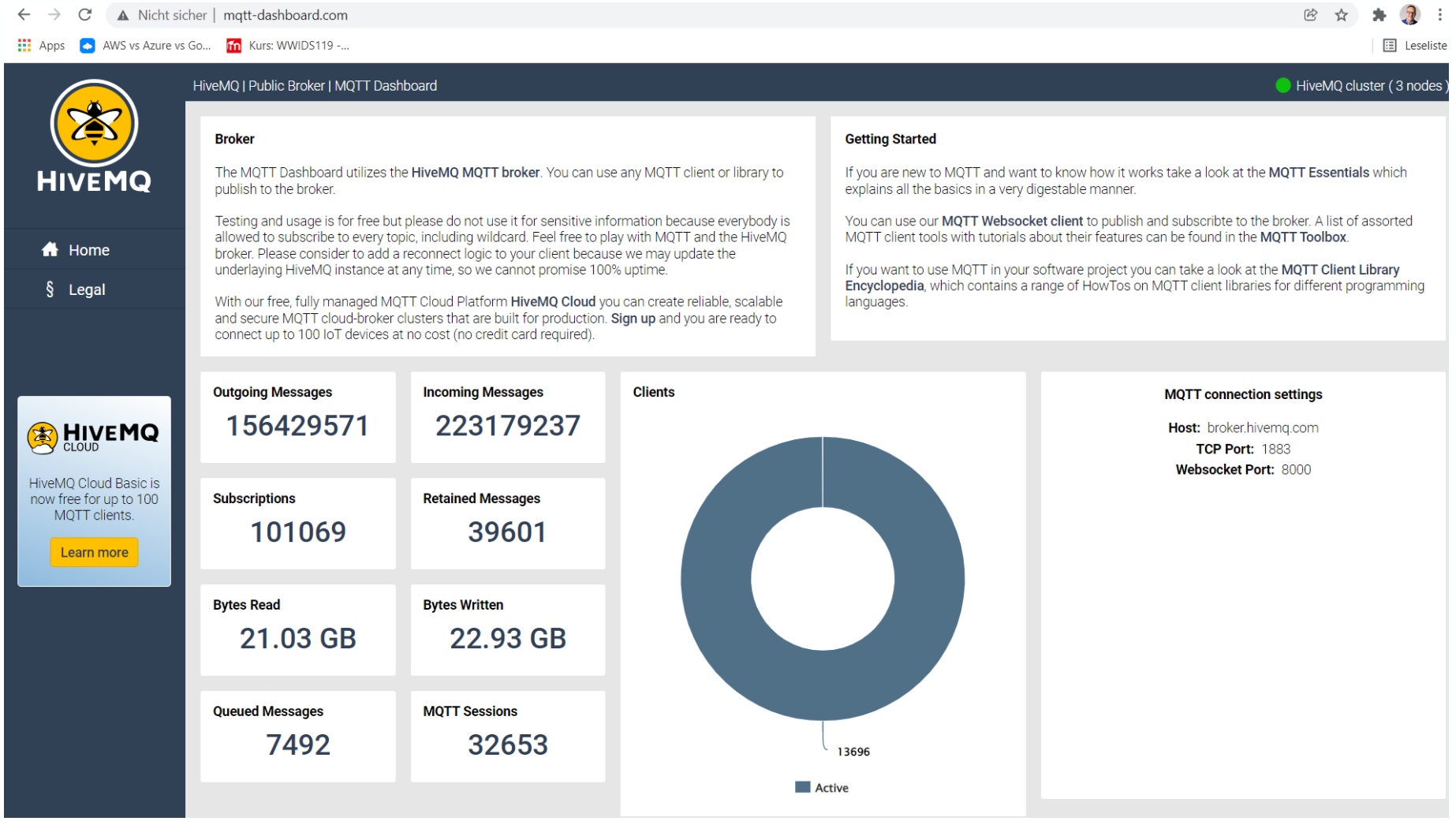
MQTT Broker

- Aufgaben: Speicherung, Verwaltung und Verteilung von Informationen (Topics)
- Ausfallsicherer Betrieb notwendig

MQTT Client

- Publisher: sendet Nachrichten an ein Topic
- Subscriber: empfängt/abonniert Nachrichten

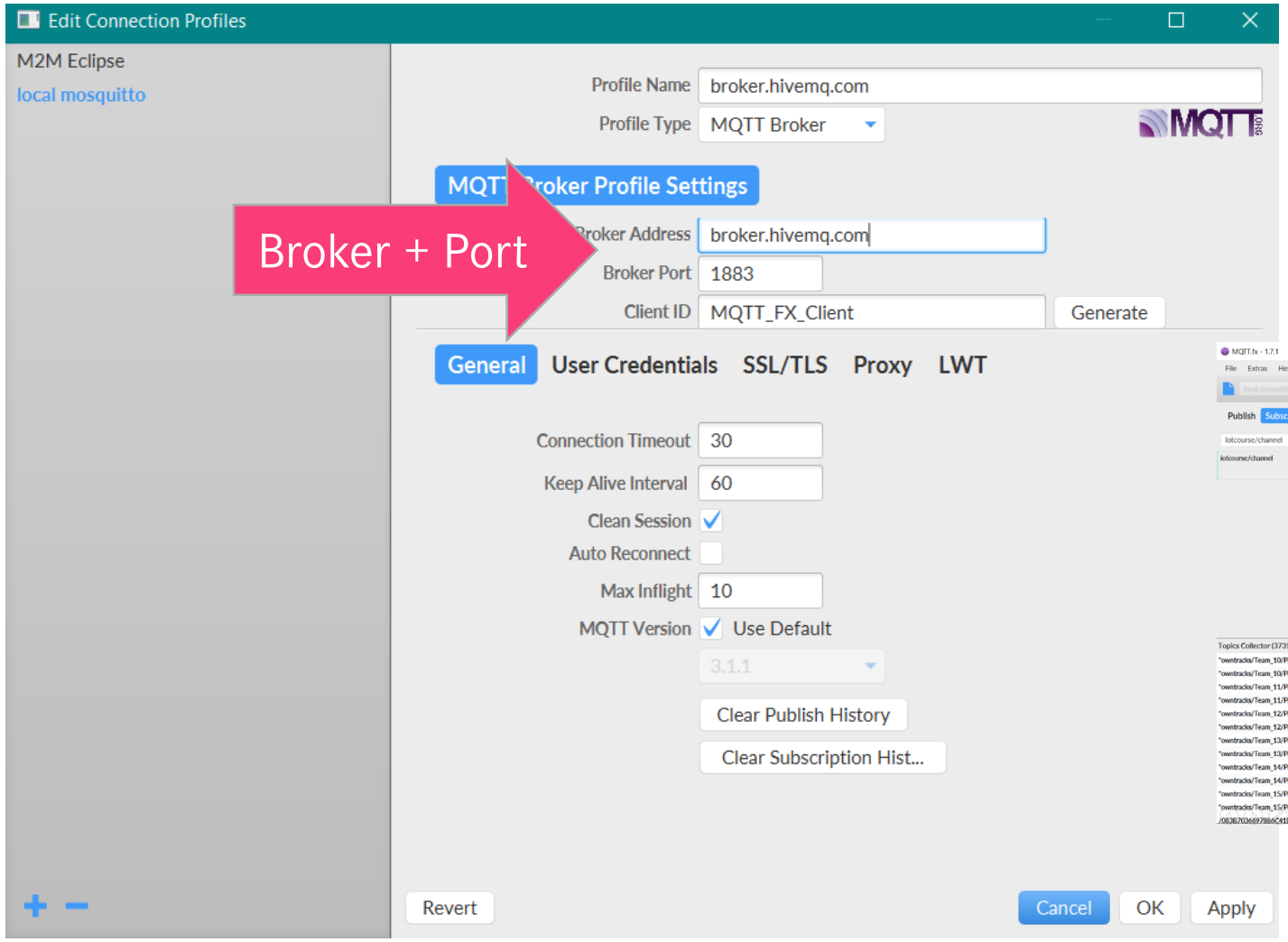
Öffentlicher MQTT Broker zum Testen



Host: broker.hivemq.com

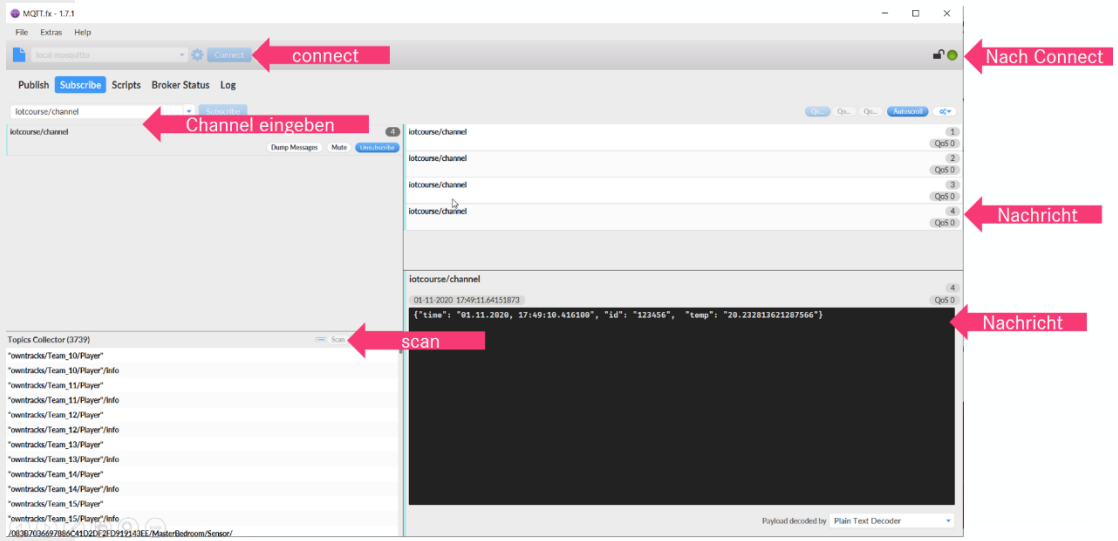
Port: 1883

MQTT.fx



Optional, kann für Tests hilfreich sein:

MQTT.fx
http://www.jensd.de/apps/mqttfx/1.7.1/
(neue Versionen lizenzpflichtig)



MQTT Client - Beispiele

Qualitätsstufen von Nachrichten
- QoS=0: eine Nachricht wird maximal einmal geliefert
- QoS=1: mindestens einmal geliefert
- QoS=2: Garantie, dass eine Nachricht "exakt einmal geliefert" wurde

Beispiele und Anleitungen, z.B.:
http://www.steves-internet-guide.com/into-mqtt-python-client/
https://www.emqx.com/en/blog/mqtt-js-tutorial

```
pip install paho-mqtt
```

Publisher

```
import paho.mqtt.client as mqtt
broker_address="broker.hivemq.com"
client = mqtt.Client("P1", clean_session=False) #use your own unique ID
client.connect(broker_address)
client.publish("DataMgmt/FIN", <Ihr json>, qos=1)
```

Subscriber

```
import paho.mqtt.client as mqtt
broker_address="broker.hivemq.com"

def on_message(client, userdata, message):
    print("message read: ", str(message.payload.decode("utf-8")))
    <JSON Message in DB schreiben>

client = mqtt.Client("S1", clean_session=False) #use your own unique ID
client.on_message=on_message
client.connect(broker_address)
client.subscribe("DataMgmt/FIN", qos=1)
client.loop_forever()
```

Persistente Sitzungen
- Client verbindet sich mit MQTT Broker und erstellt abonniert Topics (subscriptions)
- Falls Verbindung unterbrochen wird, gehen Nachrichten verloren
- Der Verlust von Nachrichten kann vermieden werden durch persistente Sitzungen durch Setzen von cleanSession flag beim Verbindungsaufbau
  - True: Client möchte keine persistente Sitzung
  - False: Client fordert persistente Sitzung an

MQTT Publisher erstellen

Erstellen Sie einen MQTT-Publisher. Erzeugen Sie Daten der Form

```
{
   „fin":"SNTU411STM9032150",
   "zeit":"01.01.2020 15:34:05.123",
   "geschwindigkeit":60,
   "ort":10
}
```

```
{
   „fin":"<FIN>",
   "zeit":"<DD.MM.YYYY HH24:MI:SS.MS>",
   "geschwindigkeit":<Geschwindigkeit>,
   "ort":<Ort-ID>
}
```

Schreiben Sie alle 5 Sekunden Daten in das Topic `DataMgmt/FIN`

- FIN: die von Ihnen festgelegte FIN (siehe Folien zu „Daten in staging Layer laden“)
- Zeit: aktuelle Systemzeit (Millisekunden oder Mikrosekunden)
- Geschwindigkeit: zufälliger Integer-Wert im Bereich 0 bis 200
- Ort: ID (Feld ort_id aus der Tabelle staging.ort; verwendeter Wert darf konstant sein)

Achten Sie auf valide JSON-Dokumente! (JSON überprüfen: https://jsonformatter.curiousconcept.com/)

Sie können optional weitere Key-Value-Paare hinzufügen.

MQTT Subscriber erstellen

Erstellen Sie einen MQTT-Subscriber

- Der Subscriber liest Nachrichten aus dem MQTT Topic `DataMgmt/FIN` aus (d.h. nicht nur Ihre eigenen!)
- Die Nachrichten werden in der PostgreSQL Datenbank in die Tabelle staging.messung geschrieben

Für den PostgreSQL-Container muss das Passwort des Users gesetzt werden!

```
C:\>docker exec -it columnarpostgresql bash
postgres@a03231f13afb:/usr/src/dwh_course$ psql -U postgres
psql (12.4 (Debian 12.4-1.pgdg100+1))
Type "help" for help.

postgres=# \password
Enter new password:
Enter it again:
```

Verbindungsdaten sind üblicherweise:

```
pip install psycopg2-binary
import psycopg2
conn = psycopg2.connect("dbname='postgres' user='postgres' password='…' host='localhost' port='5432'")
```

Abgabe

- Abgabe der Skripte über moodle oder github (Abgabe in moddle: txt-Datei mit dem github-Link)

3 Abgabe eines sql-Skripts 03_staging.sql

4 5 Abgabe Skript(e)

Daimler TSS GmbH
Wilhelm-Runge-Straße 11, 89081 Ulm / Telefon +49 731 505-06 / Fax +49 731 505-65 99
tss@daimler.com / Internet: www.daimler-tss.com
Sitz und Registergericht: Ulm / HRB-Nr.: 3844 / Geschäftsführung: Martin Haselbach (Vorsitzender), Steffen Bäuerle