

# **- Bus bunching - Datenimport, Verarbeitung und Darstellung von reellen Daten**

Justin Sprenger (s0556255), Kai Thummerer (s0545266)

8. Januar 2018

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Problem- und Aufgabenstellung . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>3</b>
2.1	BVG-Datenbank . . . . .	3
2.2	JavaFX . . . . .	3
2.3	JDBC-API . . . . .	4
<b>3</b>	<b>Erstellen der Anwendung</b>	<b>5</b>
3.1	MySQL-Import einer Oracle Dumpfile . . . . .	5
3.2	Analyse der Datensätze aus der Dumpfile . . . . .	5
3.3	Implementieren der Hauptanwendung . . . . .	7
3.4	Implementieren einer Datenbank-Schnittstelle . . . . .	8
3.5	Berechnung der Position der einzelnen Busse(Zeitabstand) . . . . .	8
<b>4</b>	<b>Ergebnis</b>	<b>8</b>
<b>5</b>	<b>Literaturverzeichnis</b>	<b>9</b>
<b>6</b>	<b>Anhang</b>	<b>9</b>
6.1	Alle ausgearbeiteten SQL-Abfragen . . . . .	9

# 1 Einleitung

## 1.1 Problem- und Aufgabenstellung

Ein bekanntes Problem im öffentlichen Verkehr ist das Bus bunching. Das bedeutet, dass der zeitlich geplante Verkehrsfluss aus dem Takt gerät und es somit zu Zeitverzögerungen und veränderten An- und Abfahrten kommt. Um diesem Problem vorzubeugen wird eine Software entwickelt, welche dem Zweck dient die aktuelle Position der Fahrzeuge bzw. auch den Abstand der Fahrzeuge darstellt, woraufhin schnell erkannt werden kann wo es zu Problemen führen kann. Das Programm wird hierbei auf aufgezeichnete Daten zugreifen.

## 2 Grundlagen

### 2.1 BVG-Datenbank

Zur Bewältigung der Belegarbeit wurde eine Datenbank (in Form einer Dumpfile) der BVG inklusive der dazugehörigen Dokumentation bereitgestellt. In der Datenbank sind alle wichtigen Informationen der einzelnen Busse und der Bus-Linien abgespeichert.

### 2.2 JavaFX

Die JavaFx , eine Java-Spezifikation von Oracle ist Bestandteil sämtlicher Java Plattformen. Beispielsweise JRE (Java SE Runtime Environment) und des JDK (Java Development Kit). Sie enthält Klassen und Interfaces, geschrieben in Java Code. JavaFx löste die veralteten GUI-Toolkits AWT und Swing ab. Mit dieser wird die Anfertigung , sowie Verteilung von multimedialen, interaktiven Inhalten und GUI's (grafischen Benutzeroberflächen) vereinfacht.

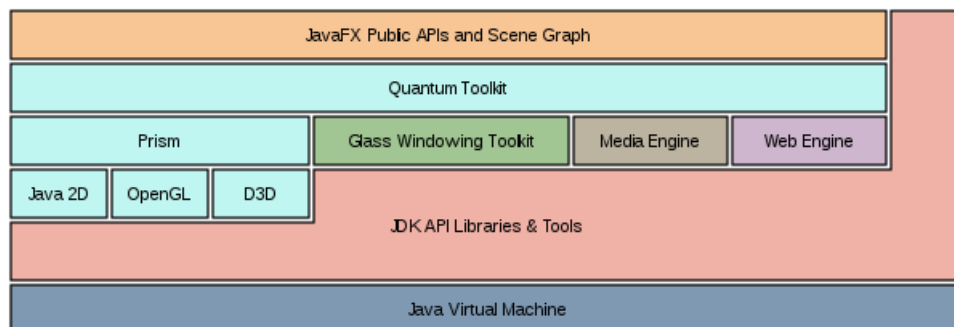


Abbildung 1: JavaFx Architektur

## 2.3 JDBC-API

Die Java Database Connectivity steht für Java Datenverbindungsfähigkeit. Seit 1996 fungiert diese API als Standard einer unabhängigen Schnittstelle zwischen der Programmiersprache Java zu relationalen Datenbanken differierender Hersteller.

JDBC übermittelt SQL-Anfragen zwischen der Java Anwendung und der Datenbank und gibt deren Ergebnisse, sofern vorhanden zurück.

Für jede spezifische DB ist ein spezieller JDBC Treiber erforderlich, der meist vom Anbieter des DB-Systems zur Verfügung gestellt wird.

JDBC Treiber unterteilen sich in 4 Kategorien :

- Typ 1- JDBC-ODBC Bridge Driver
- Typ 2- Native-API Partly-Java Driver
- Typ 3- Net-Protocol All-Java Driver
- Typ 4- Native-Protocol All-Java Driver

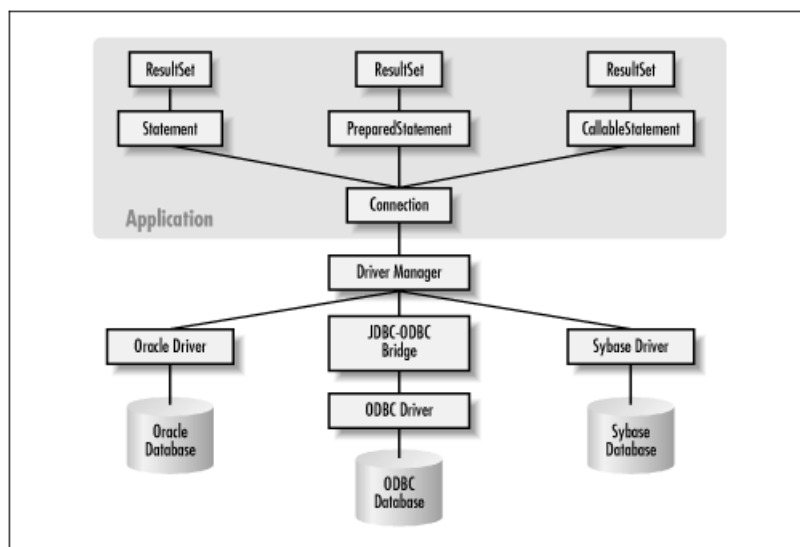
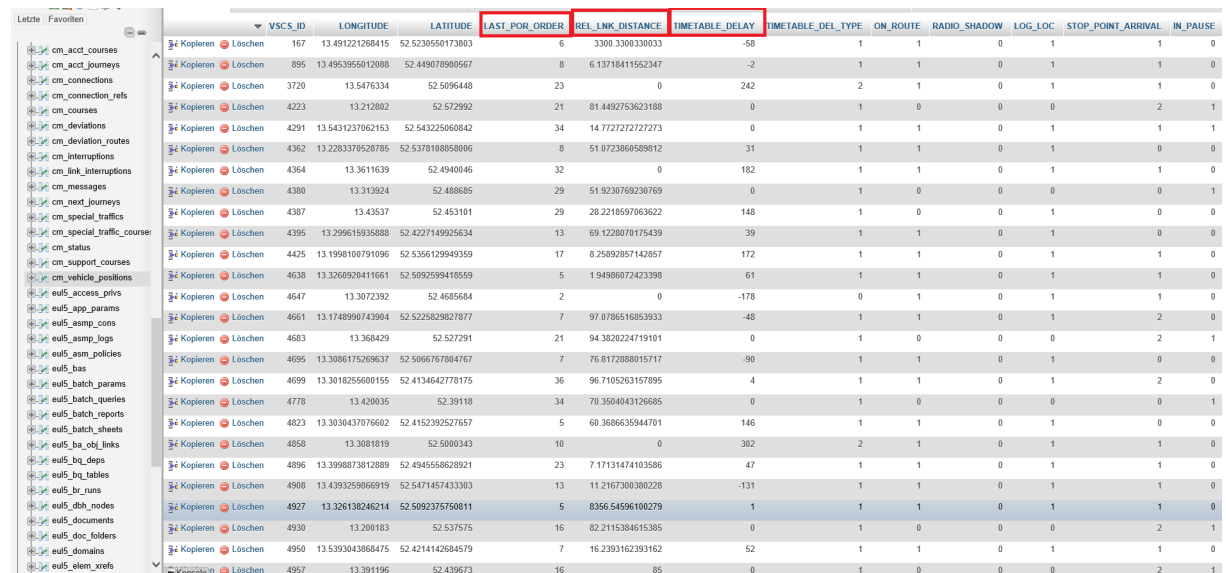


Abbildung 2: JDBC API

## 3 Erstellen der Anwendung

### 3.1 MySQL-Import einer Oracle Dumpfile

Mit Hilfe eines Dumpfile-Konverters ist es möglich die bereits bestehenden Oracle-Dumpfiles in MySQL-Dumpfiles zu konvertieren. Durch die Verwendung der kostenlosen Version des Konverters enthalten die Dumpfiles der Datenbanken nur noch 50 Einträge pro Tabelle. Die umgewandelten Dumpfiles können nun in MySQL-Datenbanken importiert werden.



	VSCS_ID	LONGITUDE	LATITUDE	LAST_POR_ORDER	REL_LNK_DISTANCE	TIMETABLE_DELAY	TIMETABLE_DEL_TYPE	ON_ROUTE	RADIO_SHADOW	LOG_LOC	STOP_POINT_ARRIVAL	IN_PAUSE
Kopieren	167	13.491221268415	52.5230550173803	6	3300.3300330033	-58	1	1	0	1	1	0
Kopieren	895	13.4953955012088	52.44907890567	8	6.13718411552347	-2	1	1	0	1	1	0
Kopieren	3720	13.5476334	52.5096448	23	0	242	2	1	0	1	1	0
Kopieren	4223	13.212802	52.572992	21	81.4492753623188	0	1	0	0	0	2	1
Kopieren	4291	13.5431237062153	52.543225060842	34	14.7727272727273	0	1	1	0	1	1	1
Kopieren	4362	13.2283378528785	52.5378108858006	8	51.072308659812	31	1	1	0	1	0	0
Kopieren	4364	13.3611639	52.4940046	32	0	182	1	1	0	1	1	0
Kopieren	4380	13.313924	52.488685	29	51.9230769230769	0	1	0	0	0	0	1
Kopieren	4387	13.43537	52.453101	29	28.2218597063622	148	1	0	0	1	0	0
Kopieren	4395	13.299615935888	52.4227149925634	13	69.1228078175439	39	1	1	0	1	0	0
Kopieren	4425	13.1998100791096	52.5356129949359	17	8.25892857142857	172	1	1	0	1	1	0
Kopieren	4638	13.3269920411661	52.5092599418559	5	1.94986072423398	61	1	1	0	1	1	0
Kopieren	4647	13.3072392	52.4685684	2	0	-178	0	1	0	1	1	0
Kopieren	4661	13.1748990743904	52.5225829827877	7	97.0786516853933	-48	1	1	0	1	2	0
Kopieren	4683	13.368429	52.527291	21	94.3828224719101	0	1	0	0	0	2	1
Kopieren	4695	13.3086175269637	52.5066767084767	7	76.8172888015717	-90	1	1	0	1	0	0
Kopieren	4699	13.3018255600155	52.4134642778175	36	96.7105263157895	4	1	1	0	1	2	0
Kopieren	4778	13.428035	52.39118	34	70.3504043126685	0	1	0	0	0	0	1
Kopieren	4823	13.3038437076602	52.4152392527657	5	60.3686635944701	146	1	1	0	1	0	0
Kopieren	4858	13.3081819	52.5000343	10	0	382	2	1	0	1	1	0
Kopieren	4896	13.3998873812889	52.4945558628921	23	7.17131474103586	47	1	1	0	1	1	0
Kopieren	4908	13.4393259866919	52.5471457433383	13	11.2167308380228	-131	1	1	0	1	1	0
Kopieren	4927	13.326138246214	52.5092375750811	5	8356.54596100279	1	1	1	0	1	1	0
Kopieren	4930	13.200183	52.537575	16	82.2115384615385	0	1	0	0	0	2	1
Kopieren	4950	13.5393043868475	52.4214142684579	7	16.2393162393162	52	1	1	0	1	1	0
Kopieren	4957	13.391196	52.439673	16	85	0	1	0	0	0	2	1

Abbildung 3: Übersicht einer relevanten Tabelle

### 3.2 Analyse der Datensätze aus der Dumpfile

Bei der Formulierung der Abfragen gab es einige Probleme mit der Identifizierung der wichtigen Attribute der Tabellen. Einige der Attribute waren zwar in der Dokumentation der Datenbank enthalten, jedoch waren viele der relevanten nicht beschrieben.

Die für die Abfragen relevanten Tabellen lauten :

- Lines
- Links
- Points\_On.Route
- Network\_Points
- Routes

- Courses\_On\_Journey
- cm\_vehicle\_positions
- cm\_acct\_journey

Durch das importieren der Dumpfile in MySQL wurde die Konsistenz der Struktur bzw. die Beziehungen zwischen/in den Tabellen aufgehoben(Primär/Fremd-Schlüssel). Dadurch haben sich massive Probleme bei der Gestaltung der Abfragen ergeben.

Die Abfragen um den aktuellen Bus zu ermitteln(mit Beschreibung) lauten wie folgt:

// Gibt die LDI-ID zurück, welche dazu dient die Routennummer und somit die Bushaltestellen-ID zu ermitteln

```
SELECT LDI_ID
FROM Lines
WHERE NO like 192
```

// Gibt die Routennummer aus

```
SELECT ROU_NO
FROM Points_On_Route
WHERE LDI_ID like 10606661
```

// Ermittelt aus den gelieferten Routennummern, die ID der Route, welche der Richtung entspricht

```
SELECT NO
FROM ROUTES
WHERE DIRECTION like 'variable Richtung' and NO like 118
```

// Liefert die Bushaltestellen-ID

```
SELECT NP_ID
FROM Points_On_Route
WHERE LDI_ID like 10606661 and ROU_NO like 118
```

// Ermittelt anhand der Bushaltestellen-ID und der Haltestelle die Busnummer

```
SELECT VSCS_ID
FROM NETWORK_POINTS
WHERE ID like 3564322920 and Name like 'U Osloer Str.'
```

// Gibt die Zeitabweichung und die aktuelle Position in der Reihenfolge des Busses aus

```
SELECT TIMETABLE_DELAY, LAST_POR_ORDER
FROM CM_VEHICLE_POSITIONS
WHERE VSCS_ID Like '167'
```

### 3.3 Implementieren der Hauptanwendung

Zur Gestaltung der Anwendung wurde auf die JavaFX Api zurrückgegriffen. In der Grafischen Oberfläche hat der Nutzer die Möglichkeit die Buslinie anhand der Nummer, die Haltestelle und die Richtung anzugeben. Durch Klick auf den Button Start wird das Ergebnis in einer Tableview ausgegeben. In der TableView enthalten sind folgende Informationen: Busliniennummer, aktuelle Haltestelle, Zeitabstand des Busses zur aktuellen Haltestelle.

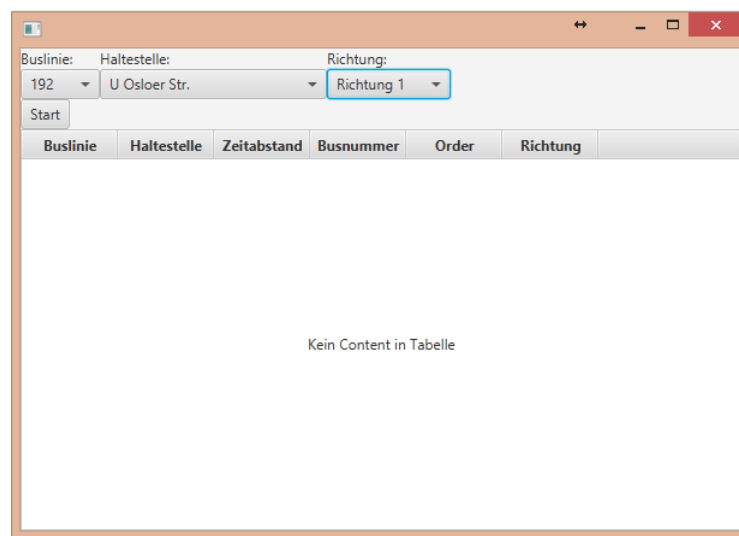


Abbildung 4: Hauptanwendung

### 3.4 Implementieren einer Datenbank-Schnittstelle

Um eine Verbindung mit der Datenbank herstellen zu können wurde die JDBC-API verwendet. In dem Programm wurde eine Datenbank Klasse erstellt, welche diese Schnittstelle importiert und in welcher die SQL-Abfragen deklariert sind.

```
66      * @param ids Busstationsnummer
67      * @param name Name der Haltestelle
68      * @return
69      */
70  public String[][] getVscsid(long ids,String name){
71      String[][] result = new String[1][1];
72
73      try {
74          conn = DriverManager.getConnection("jdbc:mysql://" + address + ":" + port + "/" + database + "?" + "user=" + user + "&" + "password=" + pass);
75          Statement stmt=conn.createStatement();
76          ResultSet rs=stmt.executeQuery("SELECT VSCS_ID FROM NETWORK_POINTS WHERE ID like " + ids + " and Name like '" + name + "'");
77          while(rs.next()){
78              result[result.length-1][0] = rs.getString(1);
79
80              String[][] temp = new String[result.length+1][1];
81              for(int i=0;i<result.length;i++){
82                  temp[i] = result[i];
83              }
84              result = temp;
85          }
86          conn.close();
87      } catch (SQLException e) {
88          e.printStackTrace();
89      }
90      return result;
91  }
```

Abbildung 5: JDBC DB Example

### 3.5 Berechnung der Position der einzelnen Busse(Zeitabstand)

Die Abstände müssen nicht direkt berechnet werden. in der Tabelle cm\_vehicle\_positions existiert die Spalte timedelay, welche die Verspätungen der einzelnen Busse zur nächsten Haltestelle in Sekunden angibt. Durch eine Abfrage, welche die ID des nächsten/vorherigen Busses ausgibt, kann der zeitliche Abstand zwischen dem Bus und der nachfolgenden und vorangegangenen Haltestelle ermittelt werden. Die Zeiten müssen gegebenenfalls zusammen addiert werden. Dies ist in diesem Fall jedoch nicht erforderlich, da jeweils nur ein Bus vorher und nachher ausgegeben wird.

## 4 Ergebnis

Aufgrund des Fehlerhaften Datenbank Imports in MySQL war es nicht möglich geschachtelte Abfragen zu formulieren. Aufgrund von unvollständigen Datensätzen ergeben sich aus den zusammenhängenden/passenden Abfragen keine vollständigen Resultate.

In dieser Anwendung musste aus gegebenen Umständen nicht zusammenhängende Datensätze verarbeitet werden.

Mittels einer funktionsfähigen Datenbank würden die im Anhang gezeigten SQL-Abfragen vermutlich zusammenhängende Ausgaben liefern.



## 5 Literaturverzeichnis

### Literatur

- [1] Christian Ullenboom *Java ist auch eine Insel (Auflage 12)* 2016.
- [2] JavaFX Overview (Release 8) <https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784>
- [3] Java JDBC API <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>
- [4] Java SE Technologies <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>
- [5] JAXenter 6 März 2017 <https://jaxenter.de/java-tutorial-javafx-53878>
- [6] Java Servlet Programming [https://docstore.mik.ua/orelly/java-ent/servlet/ch09\\_02.htm](https://docstore.mik.ua/orelly/java-ent/servlet/ch09_02.htm)
- [7] Oracle Java Documentation <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>

## 6 Anhang

### 6.1 Alle ausgearbeiteten SQL-Abfragen

— Aktueller Bus —

siehe Punkt 3.2 - Analyse der Datensätze aus der Dumpfile

— Nachfolgender Bus —

```
SELECT NP_ID_TO  
FROM LINKS  
WHERE NP_ID like 3564322669
```

```
SELECT NP_ID  
FROM POINTS_ON_ROUTE  
WHERE LDI_ID like 10606661 and ROU_NO like 118
```

```
SELECT ID  
FROM NETWORK_POINTS  
WHERE ID like 3564322920 and Name like 'U Osloer Str.'
```

```
SELECT VSCS_ID  
FROM NETWORK_POINTS  
WHERE ID like 3564322920 and Name like 'U Osloer Str.'
```

```
SELECT TIMETABLE_DELAY, LAST_POR_ORDER
FROM CM_VEHICLE_POSITIONS
WHERE VSCS_ID Like '167'
```

—— **Voriger Bus** ——

```
SELECT NP_ID
FROM LINKS
WHERE NP_ID_TO like 3564322669 and ID like 728
```

```
SELECT NP_ID
FROM POINTS_ON_ROUTE
WHERE LDI_ID like 10606661 and ROU_NO like 118
```

```
SELECT ID
FROM NETWORK_POINTS
WHERE ID like 3564322920 and Name like 'U Osloer Str.'
```

```
SELECT VSCS_ID
FROM NETWORK_POINTS
WHERE ID like 3564322920 and Name like 'U Osloer Str.'
```

```
SELECT TIMETABLE_DELAY, LAST_POR_ORDER
FROM CM_VEHICLE_POSITIONS
WHERE VSCS_ID Like '167'
```

—— **Richtung** ——

```
SELECT REMARK
FROM LINES
WHERE NO like 255
```