

## Projektdokumentation: Iteration No. 4

Christoph van Heteren-Frese (Matr.-Nr.: 4465677),  
Sven Wildermann (Matr.-Nr.: 4567553)

---

### 1 Rahmenbedingungen

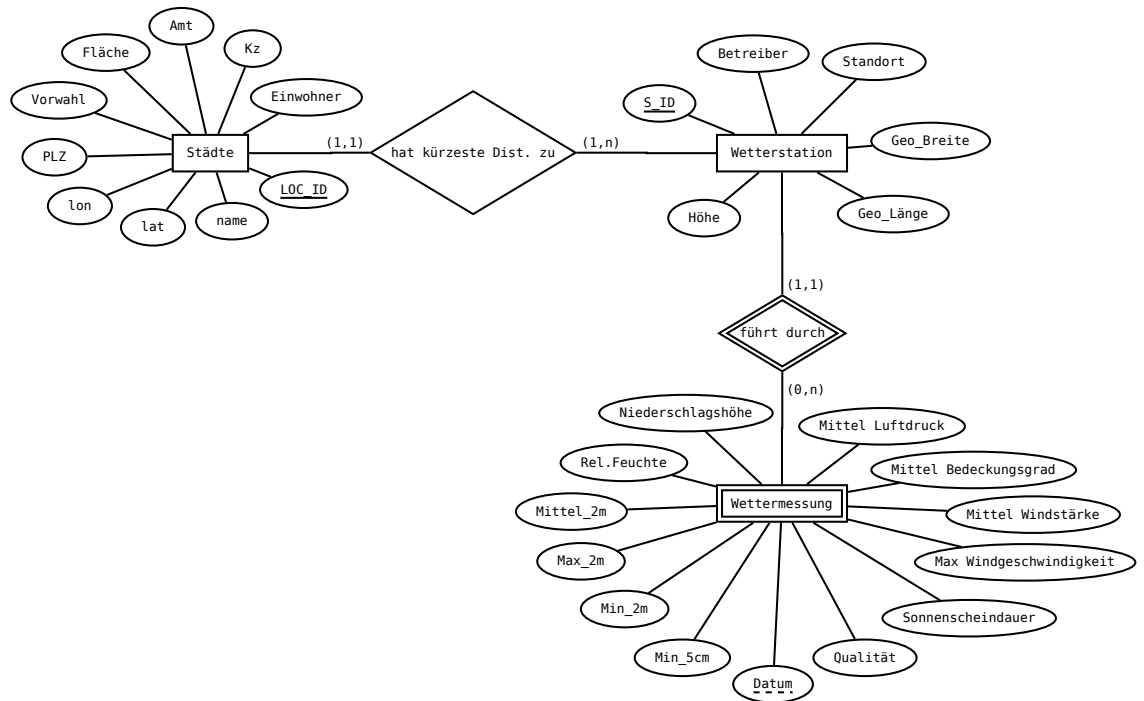
- DBS: PostgreSQL
- JavaServer Pages (JSP): Apache Tomcat 7

### 2 Modellierung

Wir modellieren unser Datenbankschema wie folgt:

Da wir davon ausgehen, dass nicht regelmäßig neue Wetterstationen entstehen oder abgebaut werden, haben wir die Relation “hat kürzeste Distanz zu” gebildet. Diese muss aktualisiert werden, sobald sich die Wetterstationen ändern. Wenn eine neue Stadt (bzw. “location”) hinzu kommt oder entfernt wird, reicht es entsprechend einen Eintrag hinzuzufügen bzw. zu entfernen.

Da die Wetterdaten über einen Schlüssel eindeutig mit den Wetterstationen verknüpft sind, haben wir hier eine Relation “führt durch”. Im Endeffekt wird hier jede Verknüpfung zwischen einer Wettermessung und einem Standort hergestellt. Aktualisieren sich die Wettermessungen, so muss auch diese Relation aktualisiert werden.



### 3 Herkunft der Daten

Die Daten der Städte stammen aus der frei zugänglichen Datenbank GeoDB. Die Wetterdaten werden von der Uni Bayreuth zur Verfügung gestellt.

### 4 Erzeugen der Relationen/Tabellen

Die erzeugten Relationen weichen teilweise von den in [1] dokumentierten Schemata ab.

#### Beispiel: geodb\_intdata

```

1 CREATE TABLE geodb_intdata (
2     loc_id integer,
3     int_type integer,
4     int_val integer,
5     valid_since date,

```

```

6     date_type_since integer,
7     valid_until date,
8     date_type_until integer
9 );

```

### Beispiel: staedte

```

1 CREATE TABLE staedte (
2     id serial NOT NULL PRIMARY KEY,
3     plz VARCHAR ( 10 ) NOT NULL ,
4     name VARCHAR ( 255 ) NOT NULL ,
5     vorwahl text,
6     einwohner integer,
7     flaeche float,
8     amt text,
9     kz varchar (3),
10    lat float,
11    lon float
12 );

```

## 5 Datenimport

Zeilen, die der JOIN-Bedingung nicht genügen, werden durch 'OUTER LEFT JOIN' nur dann in die Zieltabelle eingefügt, wenn keine weiteren Bedingungen dies verhindern. Letzteres ist genau dann der Fall, wenn der gewünschte Texttyp im WHERE-Statement 'selektiert' wird. Daher muss dies bereits in der Joinbedingung geschehen.

### Beispiel: Städte-Relation

```

1 INSERT INTO staedte (plz, name, vorwahl, einwohner, flaeche, amt, kz, lat, lon)
2 SELECT plz.text_val1 as plz, name.text_val1 as name, vorwahl.text_val1 as vorwahl,
3     einw.int_val as einwohner, flaeche.float_val as flaeche, amt.text_val1 as amt,
4     kz.text_val1 as kz, coord.lat, coord.lon
5 FROM geodb_textdata plz
6 LEFT OUTER JOIN geodb_textdata name
7     ON plz.loc_id = name.loc_id
8 LEFT OUTER JOIN geodb_textdata vorwahl
9     ON (plz.loc_id = vorwahl.loc_id AND vorwahl.text_type = 500400000)
10 LEFT OUTER JOIN geodb_floatdata flaeche
11     ON (plz.loc_id = flaeche.loc_id AND flaeche.float_type = 610000000)
12 LEFT OUTER JOIN geodb_intdata einw
13     ON (plz.loc_id = einw.loc_id AND einw.int_type = 600700000)
14 LEFT OUTER JOIN geodb_textdata amt
15     ON (plz.loc_id = amt.loc_id AND amt.text_type = 500700000)
16 LEFT OUTER JOIN geodb_textdata kz
17     ON (plz.loc_id = kz.loc_id AND kz.text_type = 500500000)
18 LEFT OUTER JOIN geodb_locations gl
19     ON (plz.loc_id = gl.loc_id AND gl.loc_type IN (100600000 ,100700000))
20 LEFT OUTER JOIN geodb_coordinates coord
21     ON plz.loc_id = coord.loc_id
22 WHERE plz.text_type = 500300000
23     AND name.text_type = 500100000 ORDER by name, plz;

```

## 6 Berechnung der Relation "hat kürzeste Distanz zu"

Für die Berechnung der Relation "hat kürzeste Distanz zu", die zwischen den Städten und den Wetterstationen vorliegt, wurden eigene SQL-Funktionen implementiert.

Die erste Funktion berechnet den Abstand zwischen zwei Koordinaten in km auf einer Kugel, so dass nicht nur die Distanz innerhalb Deutschlands korrekt berechnet wird, sondern auch die Entfernungen zwischen zwei weiter voneinander entfernten Punkten.

```

1 CREATE OR REPLACE FUNCTION distance (Lat1 IN double precision,
2                                     Lon1 IN double precision,
3                                     Lat2 IN double precision,
4                                     Lon2 IN double precision,
5                                     Radius IN double precision DEFAULT 6387.7)
6 RETURNS double precision AS '
7
8 DECLARE
9     DEGTORAD double precision := 57.29577951;
10 BEGIN
11     RETURN(COALESCE(Radius,0) * ACOS((sin(COALESCE(Lat1,0) / DEGTORAD) *
12     SIN(COALESCE(Lat2,0) / DEGTORAD)) +
13     (COS(COALESCE(Lat1,0) / DEGTORAD) * COS(COALESCE(Lat2,0) / DEGTORAD) *
14     COS(COALESCE(Lon2,0) / DEGTORAD - COALESCE(Lon1,0)/ DEGTORAD))));
15 END;'
16 LANGUAGE 'plpgsql';

```

Die zweite Funktion iteriert die Funktion distance auf allen Städten zu allen Wetterstationen, nimmt nur den kürzesten Eintrag einer Stadt zu einer Wetterstation und fügt diesen Eintrag in die Tabelle shortestdistance hinzu. Wenn diese Berechnung für alle Datensätze erneut durchgeführt werden muss, muss zuerst geprüft werden welchen Index der letzte Datensatz in der Relation städte hat (um diesen in der Funktion bei Abweichung zu ersetzen) und die Relation shortestdistance muss vorher geleert werden.

```

1 CREATE OR REPLACE FUNCTION shortest_distance() RETURNS VOID AS '
2
3 DECLARE
4 BEGIN
5
6     FOR i in 1..61832 LOOP
7         -- for all the data
8         INSERT INTO shortest_distance (select wetterstation.s_id, wetterstation.standort,
9         locations.id, locations.name, "distance"(locations.lat, locations.lon,
10         wetterstation.geo_breite, wetterstation.geo_laenge)
11         as Distanz from locations, wetterstation where
12         locations.id=i ORDER BY Distanz LIMIT 1);
13
14     END LOOP;
15
16
17 END;'
18 LANGUAGE 'plpgsql';

```

## 7 Web-Anwendung

Die Web-Anwendung wurde in JSP realisiert. Der Zugriff auf die Datenbank, die Erstellung der Tabellen und die Verknüpfung der eingegebenen Werte mit denen in der Datenbank wird aus der Datei auf Grund konsequenter Kommentierung deutlich.

```

1 <%--
2 Webanwendung Stadtwetter
3 - ermoglicht eine einfache Datenbankabfrage der StadtwetterDb
4 - genutzte Technologien:

```

```

5      - JSP (JSTL, EL)
6      - CSS, JavaScript (nur Kosmetik)
7
8      - Autoren: Christoph van Heteren-Frese, Sven Wildermann
9      - Erstellt in Berlin am 19.06.2013
10     - dokumentierter Stand vom 24. Juni 2013
11 --%>
12
13 <%-- Expression language einschalten --%>
14 <%@ page isELIgnored="false" %>
15
16 <%-- JSTL einbinden --%>
17 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
18 <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
19
20 <%-- Beans einbinden --%>
21 <jsp:useBean id="date" class="java.util.Date" />
22
23 <!DOCTYPE html>
24 <html>
25     <head>
26         <%-- Design und Titel einbinden --%>
27         <link href="design.css" type="text/css" rel="stylesheet">
28         <title>DBS 2013 Stadtwetter Web-App</title>
29     </head>
30     <body>
31         <h1>Stadtwetter Wep-App</h1>
32
33         <%-- Formularfelder fuer die Nutzereigabe erzeugen--%>
34         <form method="get">
35             <h4>Bitte Auswahl treffen:</h4>
36             <input type="text" name="cond" value="Plz, Ort oder Ortsteil" size=30
37             onFocus="if(this.value=='Plz, Ort oder Ortsteil') this.value=''"><br>
38             Datum festlegen:<br>
39             <input type="date" name="date" id="date" value="2013-01-01"><br>
40             Zeitraum angeben &nbsp; <input onclick="document.getElementById('auswahl').disabled = false"
41             type="checkbox" name="type"><br>
42             <input type="date" name="date2" id="auswahl" value="2013-02-01" disabled="disabled"><br>
43             Optionen:<br>
44             Detaillierte Darstellung &nbsp; <input type="checkbox" name="details"><br>
45             <input type="submit" value="Suchen">
46         </form>
47
48         <%-- Falls eine Eingabe gemacht wurde, zeige das Ergebnis --%>
49         <c:if test="${not empty param.cond}">
50             <c:set var="date2" value="${param.date}" />
51             <c:if test="${not empty param.date2}">
52                 <c:set var="date2" value="${param.date2}" />
53             </c:if>
54
55             <%-- Datenbank einbinden --%>
56             <sql:setDataSource driver="org.postgresql.Driver"
57                 url="jdbc:postgresql://localhost/stadtwetter"
58                 user="jdbc"
59                 password="dbs2013"
60                 scope="request"/>
61
62             <%-- Datenbankabfrage --%>
63             <sql:query var="ablage" sql="SELECT DISTINCT wm.datum, wm.sonnenscheindauer,
64                 wm.mittel_windstaerke, wm.mittel_bedeckungsgrad,
65                 wm.niederschlagshoehe, wm.mittel_2m
66                 FROM (SELECT * FROM locations WHERE name like '${param.cond}' OR

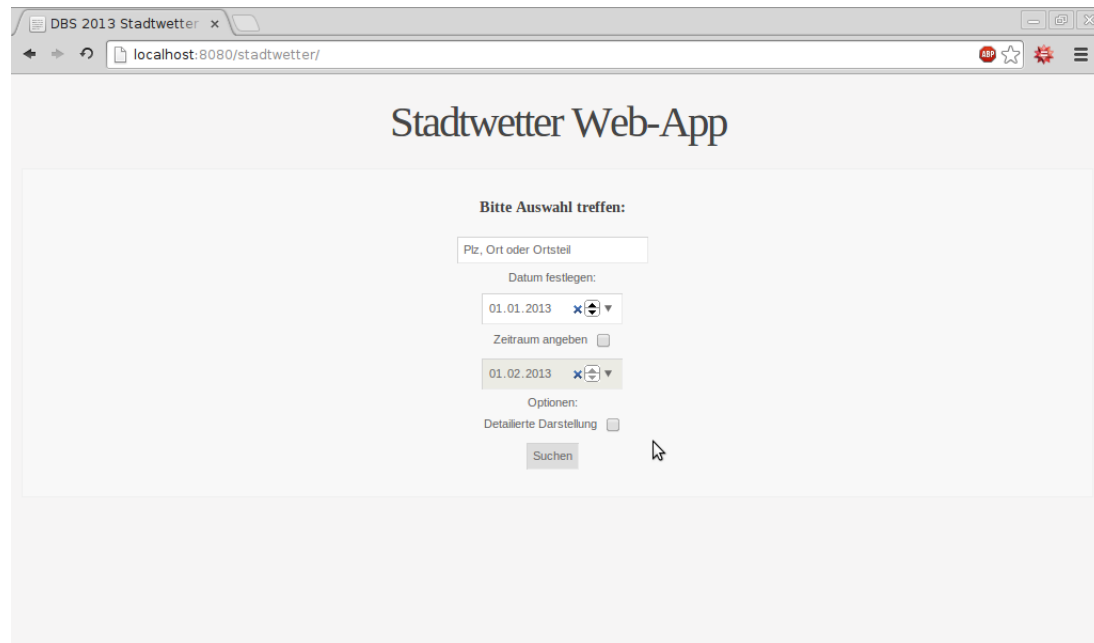
```

```

67         plz='${param.cond}' LIMIT 1) AS loc
68         JOIN shortest_distance sd ON (loc.id = sd.locationid)
69         JOIN wetterstation ws ON (ws.s_id = sd.wetterstationsid)
70         JOIN fuehrt_durch fd ON (ws.s_id = fd.s_id )
71         JOIN (SELECT DISTINCT * FROM wettermessung) AS wm
72         ON (fd.s_id = wm.stations_id) WHERE wm.datum BETWEEN '${param.date}'
73         AND '${date2}' ORDER BY wm.datum;"/>
74 <!-- ueberschrift erzeugen -->
75 <h2>Ergebnisse f&uuml;r ${param.cond}:</h2>
76
77 <!-- Konstruktion einer einfachen Tabelle inklusive Werte-->
78 <!-- die ueberschriften werden direkt aus der Datenbank geholt -->
79
80 <c:forEach items="${ablage.rows}" var="currRow">
81 <div class="CSSTableGenerator">
82 <table cellpadding="0" cellspacing="0" >
83 <tr>
84 <td>Wetter in ${param.cond} am ${currRow[ablage.columnNames[0]]}</td><td></td>
85 </tr><tr>
86 <td>Sonnenscheindauer</td>
87 <td>${currRow[ablage.columnNames[1]]} h</td>
88 </tr><tr>
89 <td>Mittlere Windst&auml;rke</td>
90 <td>${currRow[ablage.columnNames[2]]} bfa</td>
91 </tr><tr>
92 <td>Mittlerer Bedeckungsgrad</td>
93 <td>${currRow[ablage.columnNames[3]]} Achtel</td>
94 </tr><tr>
95 <td>Mittlere Temperatur</td>
96 <td>${currRow[ablage.columnNames[5]]} &deg;C</td>
97 </tr>
98 <c:if test="${not empty param.details}">
99 <tr>
100 <td>Niederschlagsh&ouml;e</td>
101 <td>${currRow[ablage.columnNames[4]]} mm</td>
102 </tr>
103 </c:if>
104 </table>
105 </div>
106 </c:forEach>
107
108 </c:if>
109 </body>
110 </html>

```

Das Design und das Verhalten der Webpage selbst wird über eine .css-Datei und eine xml-Datei bestimmt. Der Inhalt dieser Datei ist an sich trivial und hat für diese Dokumentation keine weitere Bedeutung.



**Abb. 1: Web-App vor der Benutzereingabe:** Es kann nach einem Ort, einem Stadtteil, oder einer Postleitzahl gesucht werden. Das Wetter lässt sich entweder für ein bestimmtes Datum, oder für einen Zeitraum anzeigen. Des Weiteren kann zwischen einer einfachen und einer detaillierten Ansicht gewählt werden.



**Abb. 2: Web-App nach einer Benutzereingabe:** Es wurde nach dem Wetter für den Berliner Stadtteil Tiergarten im Zeitraum 01.05.2013 bis 07.05.2013 gesucht.

Ergebnisse für Tiergarten:

Wetter in Tiergarten am 2013-05-01	
Sonnenscheindauer	6.1 h
Mittlere Windstärke	2.0 bfa
Mittlerer Bedeckungsgrad	4.4 Achtel
Mittlere Temperatur	12.3 °C
Niederschlagshöhe	0.0 mm

Wetter in Tiergarten am 2013-05-02	
Sonnenscheindauer	0.0 h
Mittlere Windstärke	2.0 bfa
Mittlerer Bedeckungsgrad	7.6 Achtel
Mittlere Temperatur	12.6 °C
Niederschlagshöhe	0.0 mm

Wetter in Tiergarten am 2013-05-03	
------------------------------------	--

**Abb. 3: Ergebnisse der Abfrage:** Die Web-App bereitet die Datensätze auf und zeigt die Ergebnisse in Tabellarischer Form an.

## 8 Datamining mit Weka

Es wurden nur die Daten der Wetterstation Berlin-Tegel betrachtet. Die für die Analyse benötigten Daten wurden über die integrierte Datenbankschnittstelle des Weka Explorers importiert (Details zum Setup sind in [2] zu finden). Die entsprechende SQL-Abfrage dafür lautet:

```
1 select * from wettermessung where stations_id=10382;
```

Mit Hilfe von Filtern wurde die so erzeugte Datenbasis aufbereitet (z.B. numerische Werte in nominale konvertiert) und anschließend mit dem Naive Bayes sowie dem J48 Algorithmus analysiert.

## Literatur

- [1] OpenGeoDb – Dokumentation. URL [http://opengeodb.org/wiki/Datenbank\\\_der\\\_OpengeoDB](http://opengeodb.org/wiki/Datenbank\_der\_OpengeoDB).
- [2] Weka Wiki – Databases. URL <http://weka.wikispaces.com/Databases>.