Friedrich-Schiller-Universität Jena

Lehrstuhl für Geoinformatik

WiSe 2017

Abschlussbericht GEO 413: Geodatenbanken

Erstellung einer räumlichen Datenbank zur Verwaltung von Messdaten

Vorgelegt von:

Krüger, Eric

Studiengang: M. Sc. Geoinformatik

Matrikel: 142645

Adresse: Jenergasse 15

07743 Jena

Email: eric.krueger@uni-jena.de

Abgabedatum: 30.11. 2017

Modulverantwortlicher: Carsten Busch

Inhalt

Abbildungsverzeichnis:

Abkürzungen

DB	Datenbank
DWD	deutscher Wetterdienst
ERM	Enity Relationship model
TLUG	Thüringer Landesamt für Umwelt und Geologie
GIS	Geoinformationssystem

1 Aufgabenstellung

Ziel des Geodatenbankkurses ist die Erstellung einer PostgreSQL/PostGIS Datenbank zur Verwaltung von Messdaten und deren zugehörigen Metadaten.

Folgenden Informationen sollen in der Datenbank verwaltet werden:

Messdaten von Pegelstationen, welche Wasserstand und Durchfluss umfassen, sowie Messdaten von Klimastationen, welche Temperatur, Niederschlag, Sonnenscheindauer und max. Windgeschwindigkeit beinhalten. Dazu gehörenden sollen die Metadaten von Klima- und Pegelstationen abgespeichert werden, welche an den Stationen erhoben werden, sowie die Stammdaten dieser. Für die Qualitätssicherung sind Informationen wie der Ansprechpartner, die verantwortliche Organisation, Aussagen zu Datenqualität und den Aufnahmeverfahren vorzusehen.

In der Datenbank sind weiterhin die Daten zu den Bundesländern und deren Geometrien zu verwalten. Des Weiteren soll ein entsprechendes E/R Modell mit geeigneten Attributen entwickelt werden. Dieses Modell soll in eine PostgreSQL/PostGIS Datenbank überführt werden. Dafür können Sie SQL Skripte, die PG-Admin3 Oberfläche für die Datenmodellierung/Tabellenerstellung nutzen. Danach soll die Datenbank mit Beispieldaten der TLUG (Stationen und fünf Zeitreihen zu Wasserstand/Durchfluss) und des DWD (Zeitreihen zu Klimadaten Temperatur, Niederschlag, Sonnenscheindauer, max. Windgeschwindigkeit) befüllt werden. Überlegen Sie sich bei dem Design, inwieweit sich die Möglichkeit der Vererbung von Tabellen nutzen lässt. Dokumentieren Sie ausführlich Ihre Schritte bei der Übertragung des E/R Modells in die Tabellenstruktur im PostgreSQL

Folgende Quellen sollen für die Datenakquisition genutzt werden:

TLUG: http://www.tlug-jena.de/hw/datenladen.html

Die Stationen sollen aus der Ausgangsdate Pegel.csv übernommen werden.

DWD: Einladen der DWD Stationen:

https://www.dwd.de/DE/leistungen/klimadatendeutschland/statliste/statlex_rich.txt?view=nasPublication&nn=16102

DWD: Auswahl von jeweils drei Beispielstationen und den vier Parametern:

http://www.dwd.de/klimadaten (aktuelle Tageswerte)

Überführen Sie diese Daten in Ihre Datenbank. Erstellen Sie für die TLUG Daten Sichten (views), welche die jeweiligen wöchentlichen Durchschnittswerte darstellen. Für die DWD Daten ist eine Übersicht (view) mit dem Jahresmittel für Niederschlag und Temperatur zu erstellen.

Aus Gründen der Nachvollziehbarkeit, sollen sämtliche Änderungen an Stationsdaten nochmals in einer „Historientabelle“ (Station_log) mitgeführt werden. Erstellen Sie dafür einen Trigger, der sämtliche Änderungen dieser Tabelle in Station_log protokolliert, zur Kontrolle nehmen Sie selbst einfache Änderungen mit dem update Befehl vor

2 Methodisches Vorgehen

Die wichtigsten Punkte des methodischen Vorgehens im Geodatenbankprojekt, können aus Abbildung 1 entnommen werden. Zu Beginn werden die Rohdaten der Klima- und Pegelmessstation von den jeweiligen Einrichtungen (TLUG, DWD) Online bezogen. Nach der Aufbringung aller Rohdaten, welche die Daten der Messstationen und deren Messwerte umfassen, wurde ein erstes konzeptionelles Datenbankmodell erstellt.

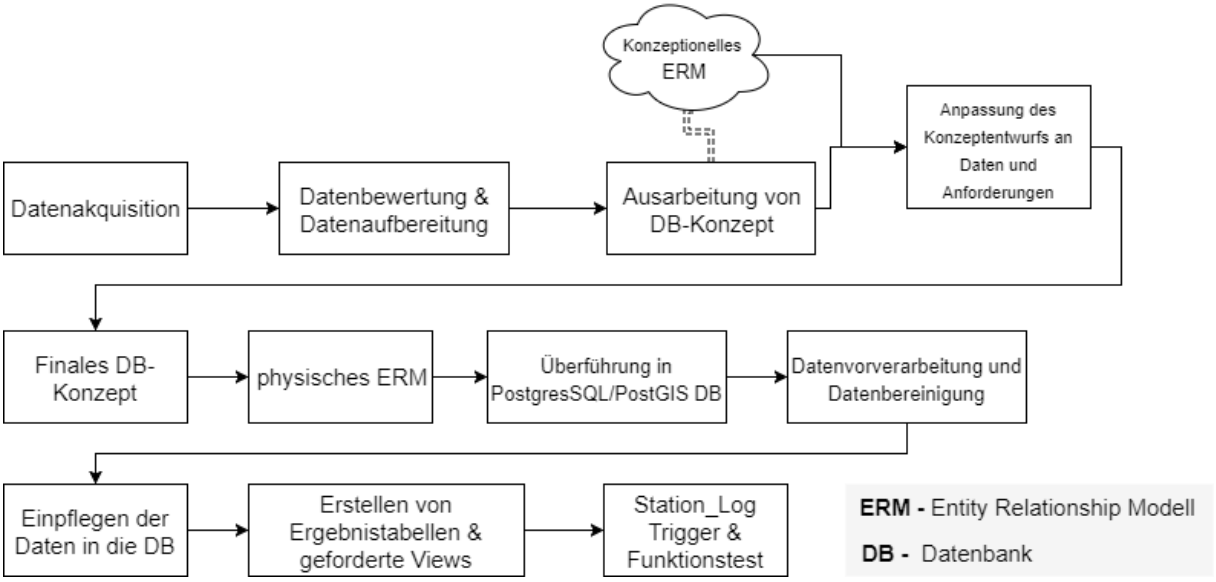

Abbildung 1) Methodisches Vorgehen

Im Laufe der Ausarbeitung des Konzeptionellen Datenbankmodells, wurden die Eingangsdaten iterativ auf das Datenbankmodell angepasst, sowie das Datenbankmodell auf die Anforderung der geforderten Datenbank. Im nächsten Schritt wurde das Final Datenbankmodell konkretisiert, da das konzeptionelle Entity-Relationshipmodel nur als grobe Vorlage dienen kann und es für die Erstellung der eigentlichen Datenbank nicht die nötige Detailtiefe besitzt.

Bei der Erstellung des physischen Entity-Relationshipmodels werden alle Entitäten der Datenbank, sowie deren Attribute, Datentypen der Attribute und Relation innerhalb der Datenbank festgelegt. Dabei sind grundlegende Konzepte wie, Normalisierung der Datenbank in zweite und dritte Normalform, die Atomarisierung der Daten innerhalb einer Entität und die Festlegung von Primär- und Fremdschlüssel zu berücksichtigen (Elmasri & Navathe, 2009). Durch die Erstellung des physischen ERM liegt ein genauer Plan zur Erstellung der Datenbank vor welcher nun in PostgreSQL übernommen werden kann.

Bevor die beschafften Messdaten in die Datenbank eingepflegt werden, wurden diese bereinigt und für den Import in die PostgreSQL-Datenbank angepasst. Im letzten Schritt wurden die geforderten Sichten erstellt welche die wöchentlichen und jährlichen Mittelwerte der Messstationen beinhalten. Darüber hinaus wurde ein Datenbanktrigger erstellt, welcher alle Änderungen an der Entität der Messstationen dokumentiert.

3 Entity Relationship Modell

Dieses Kapitel beschäftigt sich zu Beginn mit dem ersten Datenbankkonzept (siehe Abbildung 2) und im zweiten Teil mit dem Konkreten Datenbankmodell (ERM, siehe Abbildung 3), welches schließlich auch als physische Datenbank in PostgreSQL/PostGIS umgesetzt wurde. Das konzeptionelle Datenbankmodell (Abb. 2) dient dabei für einen ersten Überblick über möglich Beziehungen und Entitäten innerhalb der Datenbank. Im physischen Datenbankmodell (Abb. 3) werden die Ideen des konzeptionellen Modells konkretisiert und soweit ausgeführt das es als Vorlage zur Umsetzung der Datenbank in PostgresSQL dienen kann.

3. 1 Konzeptionelles ERM

Das konzeptionelle Datenbankmodell ist aus Abbildung 2 zu entnehmen. Es wurde sich dafür entschieden das die Tabellen Messstationen und Messdaten im Mittelpunkt stehen sollen, da diese primär in der Datenbank verwaltet werden. Die Tabelle Messstationen wird dabei von den Entitäten Einrichtung, Historientabelle und Meta-Daten verwaltet, dokumentiert und beschrieben. Ein wichtiger Punkt bei der Verknüpfung von Messstationen und Messdaten ist das beliebig viele Messdaten in Relation zu einer Messstation stehen können. In der späteren

Umsetzung werden die Tabellen Messdaten und Messstationen nach Klima und Pegel unterschieden und mittels Vererbung an übergeordnete Tabellen weitergegeben.

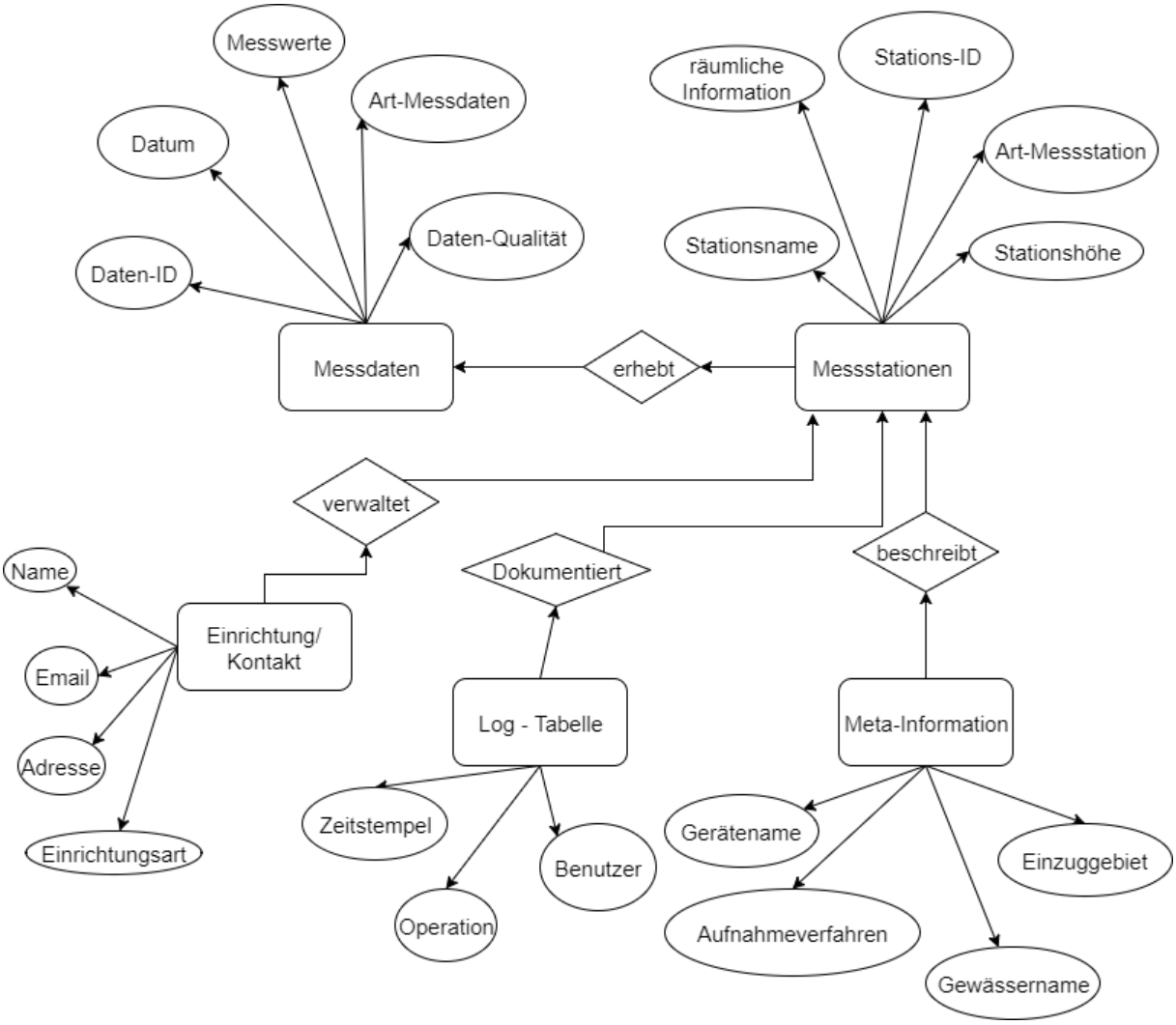

Abbildung 2) Konzeptionelles Datenbankmodell

3. 2 Physisches Entity-Relationshipmodel

Das physische ERM stellt die Finalisierung des Datenbankkonzeptes aus 3.1 dar (siehe Abbildung 3). Dabei sind, wie schon im Konzept erarbeitet, die Entitäten Stationen und Messdaten die wichtigsten Tabellen in der Datenbank. Die Unterscheidung zwischen Klima und Pegel Messstation/-daten wurde über die Art ID (AID) getroffen. Dabei kommt zur Umsetzung der Messdaten auch Vererbung zum Einsatz, in welcher die Tabelle Messdaten die Daten ID (DID), Art ID (AID), Stations_ID und das Messdatum von den Entitäten Klima_Messdaten und Pegel_Messdaten vererbt bekommt. Für eine lückenlose Dokumentation der Messverfahren und zur Kontaktaufnahme mit verwaltenden Parteien der Messstationen wurden Metainformation für Pegel- und Klimamessstationen als Entitäten eingefügt (Meta_Pegel, Meta_Klima). Diese stehen wiederum in Relation mit Stations_Personal und den

dazugehörigen Einrichtungen (DWD oder TLUG). Darüber hinaus wurde die Tabelle VG2500_BLD hinzugefügt, welche die räumlichen Informationen der deutschen Bundesländer beinhaltet (BKG, 2011). Auch wurden die Tabellen für die Ergebnissichten Jahreswerte_Klima und Wochenwerte_Pegel hinzugefügt, welche für die Erstellung der Ergebnissichten benötigt werden und so die Wochen- und Jahresmittel archivieren.

Bei der Umsetzung der Datenbank wurde auf die Einhaltung der dritten Normalform geachtet, dies meint die Abhängigkeit der Datensätze vom Primär Schlüssel (Elmasri & Navathe, 2009). Dies wurde zu einem großenteil in der Datenbank berücksichtigt, lediglich bei der Umsetzung der Tabellen von Metadaten, Einrichtung und Personal wurde diese Normalisierung für die Übersichtlichkeit der Datenbank vernachlässigt.

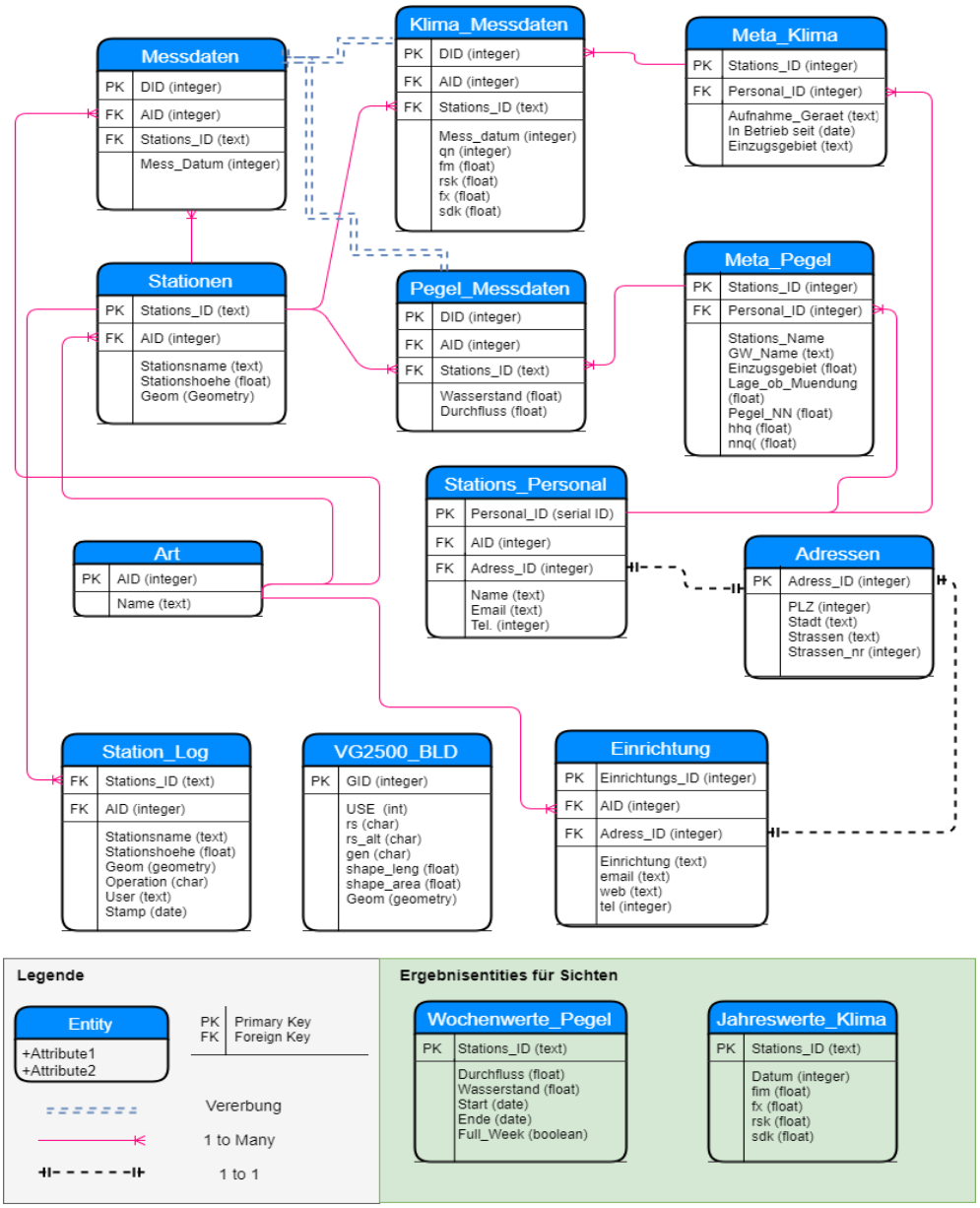

Abbildung 3) physisches ERM

Ferner wurden innerhalb der Datenbank nur „one to many“ (1:n) oder 1:1 Beziehungen geknüpft, zur Vermeidung von unhandlichen „many to many“ (m:n) Beziehungen.

4 Überführung in PostgreSQL/PostGIS Messdatenbank

Für die Erstellung der Datenbank wurde PostgreSQL 10 mit der Erweiterung PostGIS in der Version 2.4.0 genutzt. Darüber hinaus wurde die graphische Benutzerschnittstelle pgAdmin 3 genutzt um die PostgreSQL Datenbank zu bearbeiten. Alle Programme wurden unter dem Betriebssystem Windows 10 von Microsoft ausgeführt. Das detaillierte ERM diente als Vorlage bei der Datenbankerstellung.

Damit die Erweiterung PostGIS nutzbar wird muss diese explizit im SQL-Code angegeben werden (siehe Abbildung 4).

```
create extension postgis;
```

Abbildung 4) PostGIS Extension

Für die Überführung des geforderten Shapefiles der Bundesländer in die Datenbank wurde die Anwendung shp2psql genutzt, welche mit Installation der PostGIS-Erweiterung verfügbar ist.

Für die Erstellung von Entitäten innherhalb der Datenbank wurde der Befehl „create table“ genutzt. Zur Vermeidung von Datenredundanz/-operationen und für die übersichtliche Gestaltung der Datenbank wurden Vererbungsregeln auf die Tabellen Messdaten, Klima_Messdaten und Pegel_Messdaten angewandt (siehe Abbildung 5).

```
create table messdaten (DID bigint Primary key, stations_id float, mess_datum bigint, aid integer);
create table klima_messdaten (QN float, FM float, RSK float, FX float, SDK float) inherits(messdaten);
create table pegel_messdaten (wasserstand float, durchfluss float) inherits(messdaten);
```

Abbildung 5) Vererbung an Messdaten

Durch diese Vererbungsregeln musste kein expliziter „insert“ Befehl an die Tabelle Messdaten übergeben werden, denn an die Tabelle Messdaten wurden automatische die Variablen DID, Stations_ID, Mess_Datum und AID vererbt.

Für die Dokumentation aller Änderungen der Tabelle Stationen war es nötig eine Datenbank-trigger einzufügen, welcher die Änderungen (update - verändern, delete – löschen und insert – einfügen) in der Tabelle Station_Log dokumentiert (siehe Abbidlung 6; PostgreSQL-Wiki, 2012).

```
create or replace function change_log_stations() RETURNS trigger as $station_log$
 BEGIN
        IF (TG_OP = 'DELETE') THEN
            INSERT INTO station_log SELECT 'D', now(), user, OLD.*;
            RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
            INSERT INTO station_log SELECT 'U', now(), user, NEW.*;
            RETURN NEW;
        ELSIF (TG_OP = 'INSERT') THEN
            INSERT INTO station_log SELECT 'I', now(), user, NEW.*;
            RETURN NEW;
        END IF;
        RETURN NULL; -- result is ignored since this is an AFTER trigger
    END;
$station_log$ LANGUAGE plpgsql;


drop trigger if exists station_log on stationen;

CREATE TRIGGER station_log
AFTER INSERT OR UPDATE OR DELETE on stationen
    FOR EACH ROW EXECUTE PROCEDURE change_log_stations();
```

Abbildung 6) Datenbank Trigger

Die Möglichkeit Relation innerhalb der Datenbank zu knüpfen wurde durch Primär- und Fremdschlüssel ermöglicht, wie bereits im ERM dargestellt. Zur Veranschaulichung soll Abbildung 7 dienen. Dabei dienen die Primärschlüssel zur eindeutigen Kennzeichnung von Datensätzen innerhalb einer Entität (Elmasri & Navathe, 2009). Fremdschlüssel stellen dabei die Primärschlüssel der in Relation stehenden Entität dar und dienen zur Kontrolle des Dateneintrags (Elmasri & Navathe, 2009).

```
alter table messdaten add constraint stations_id foreign key (stations_id)
       references stationen
       on delete cascade;
```

Abbildung 7) Beispielabfrage "inner join"

5 Datenquellen und Vorbereitung

Für dieses Datenbankprojekt wurden Daten des Deutschenwetterdienstes und Daten der Thüringer Landesanstalt für Umwelt und Geologie genutzt. Aus beiden Datenquellen wurden Messstation und deren Lage bezogen, die Messdaten in Form von Zeitreihen, sowie die Metadaten der Messstationen. Die genutzten Daten teilen sich wie folgt auf:

DWD:

- 3 verschieden Klimamessstationen des
- Deutschen Wetterdienstes (DWD, 2017b)

- Messdaten der 3 Stationen zu Sonnenscheindauer, max. Windgeschwindigkeit, Temperatur und Niederschlag (DWD, 2017a)
- 1 Jahr aufgeteilt in 12 Monate
- Metadaten der Klimastationen

TLUG:

- Pegelmessstationen (Breitgestellt durch Modulleiter)
- Messdaten von 5 verschiedenen Pegelstation zu Wasserstand und Durchfluss
- Zeitraum von 4 Wochen (TLUG, 2017)
- Metadaten und Stammdaten der Pegelstationen

Die von DWD und TLUG bezogenen Daten zu Messstationen und Messwerten wurden als Tabstopp getrennte Textdaten heruntergeladen und mit der Programmiersprache R (Version 3.4.2) für den Import in die Datenbank vorbereitet. Dies ermöglichte es die Datenvorbereitung zu automatisieren. Im Hinblick darauf das die Datenbank theoretisch weiter als Messdatenbank genutzt werden kann ist so eine schnelle und generische Vorverarbeitung der Rohdaten möglich. Zu erwähnen ist das für die Pegeldaten im Rahmen der Datenvorbereitung eine Spalte zur Verwaltung der Stations-ID, hinzugefügt wurde, darüber hinaus wurde die Struktur des Aufnahmedatums an das Datumsformat der Klimawerte angepasst. Für Klima- sowie Pegeldaten wurde in der Datenvorbereitung die Art-ID (AID) hinzugefügt, welche definiert ob es sich um Klima- oder Pegeldaten handelt, ferner wurde eine Daten-ID (DID) für jeden Datensatz erstellt, welche sich aus Stations-ID, Datum und Uhrzeit zusammensetzt. Dies ermöglicht es allein an Hand der DID Aufnahmestation und Zeitpunkt der Aufnahme abzuleiten.

Die vorbereiten Daten wurden als Semikolon getrennte Textdateien abgespeichert. Alle für den Import vorbereiten Daten sind auf dem beigelegten Datenträger zu finden, dies schließt die genutzten Skripte zur Vorbereitung ein. Für die Bearbeitung des Geodatenprojektes wurde des Weiteren ein Reposetory auf Github angelegt, welche unter dem folgenden Link einzusehen ist: https://github.com/EricKrg/geodb dies ermöglicht den Prozess zur fertigen Datenbank nachzuvollziehen. Die Datenbank ist über die Wiederherstellungsdatei „geodb_dump“ im Ordner sql_sheets zu rekonstruieren, diese Backupdatei muss dafür in einer leeren Datenbank mit pgAdmin 3 wiederhergestellt werden.

6 Datenimport & Überführung in die Messdatenbank

Das Shapefile der Bundesländer wurde mithilfe des PostGIS Shapefile Import/Export Managers shp2psql importiert. Für den Import der Messdaten wurden die zuvor aufbereiteten Daten vorerst in temporäre Tabellen importiert, um diese im nächsten Schritt unter der Nutzung des Insert-Befehls in die Zieltabelle einzufügen (siehe Abbildung 7).

```
create table temp_brocken (did bigint, stations_id float, mess_datum bigint, aid int, qn int, fm float, rsk float, fx float, sdk float);
create table temp_fehmarn (did bigint, stations_id float, mess_datum bigint, aid int, qn int, fm float, rsk float, fx float, sdk float);
create table temp_zugspitze (did bigint, stations_id float, mess_datum bigint, aid int, qn int, fm float, rsk float, fx float, sdk float);

copy temp_brocken from 'C:/Users/Eric/Documents/geodb/klima_daten/brocken_raw_klima.txt' Delimiter ';';
copy temp_fehmarn from 'C:/Users/Eric/Documents/geodb/klima_daten/fehmarn_raw_klima.txt' Delimiter ';';
copy temp_zugspitze from 'C:/Users/Eric/Documents/geodb/klima_daten/Zugspitze_raw_klima.txt' Delimiter ';';

create table klima_messdaten (QN float, FM float, RSK float, FX float, SDK float) inherits(messdaten);
alter table klima_messdaten add primary key(did);

Insert into klima_messdaten  select did, stations_id, mess_datum , aid , qn , fm , rsk , fx , sdk  from temp_fehmarn;
Insert into klima_messdaten  select did, stations_id, mess_datum , aid , qn , fm , rsk , fx , sdk  from temp_brocken;
Insert into klima_messdaten  select did, stations_id, mess_datum , aid , qn , fm , rsk , fx , sdk  from temp_zugspitze;
```

Abbildung 8) Beispiel Datenimport

Das einladen der Messstation gestaltet sich ähnlich zum einladen der Messdaten, jedoch muss im Fall der Messstationen eine weitere Spalte vom Datentyp Geometry hinzugefügt werden, damit es möglich ist räumliche Operationen auf diese Entität anzuwenden. Dies ist umzusetzen mit dem Befehl AddGeometryColumn unter der Angabe des Koordinatensystems als geodätische Parameter ID (EPSG) und dem gewünschten Vektorentyp, in diesem Fall PointZ (siehe Abbildung 9).

```
Select AddGeometryColumn('stationen','geom',4326,'POINT',3);
update stationen set geom= GeomFromEWKT ('SRID=4326;POINT('||lon||' '||lat||' '||STATIONSHOEHE_METERN||')');
```

Abbildung 9) Hinzufügen der Geometrie

Da die Koordinaten der Pegelstationen nicht im Koordinatensystem WGS84 vorlagen mussten diese im Zuge des Imports in das Zielkoordinatensystem transformiert werden und nach Transformation zur Haupttabelle der Messstationen hinzugefügt werden. Die Transformation in das Koordinatensystem WGS84 wurde durch den Befehl ST_Transform umgesetzt (siehe Abbildung 10).

```
SELECT AddGeometryColumn('pegel_temp','geomgg',31468,'POINT',3); --srid 31468, punkt 3 dimensional
Update pegel_temp set geomgg= GeomFromEWKT ('SRID=31468;POINT('||rw||' '||hw||' '||hoehe||')'); -- set geom to srid to gg

SELECT AddGeometryColumn('pegel_temp','geomwgs',4326,'POINT',3);
Alter Table pegel_temp alter column geomwgs Type geometry(PointZ,4326) using ST_Transform(geomgg,4326); -- transform to wgs
```

Abbildung 10) Beispiel ST_Transform

Neben Messstationen und Messdaten wurden auch die Metadaten der Messstationen, Information zu den verwaltenden Einrichtungen und dem Stations-Personal eingeladen. Dabei ist zu berücksichtigen das die Tabelle Stations-Personal und die Damit Verknüpfte Entität Adressen, sowie die Metadaten der Klimastationen noch ungefüllt sind und bei Bedarf mit den jeweiligen Kontaktdetails gefüllt werden kann. Damit bildet die Datenbank eine strukturierte Grundlage welche flexibel erweiterbar ist.

7 Notwendige Sichten

Nach dem importieren der Daten in die Datenbank sollten Sichten zu den wöchentlichen Durchschnittswerten (Wasserstand, Durchfluss) der Pegelstationen und den jährlichen Durchschnittswerten der Klimastationen (Temperatur, max. Windgeschwindigkeit, Niederschlag, Sonnenscheindauer) erstellt werden.

Für die Berechnung der Durchschnittswerte von Pegel- und Klimamesswerten wurde die Möglichkeit genutzt die erstellte Datenbank mit der Programmiersprache R anzusprechen. Umgesetzt wurde dies mit dem Software-Pakete RPostgreSQL, welche über einer offenen Datenbank Verbindung (ODBC) mit der Messdatenbank kommunizierte (Conway, Eddelbuettel, Nishsiyama, Prayaga & Tiffin, 2017; siehe Abbildung 11).

```
# loads the PostgreSQL driver
drv <- dbDriver("PostgreSQL")
# creates a connection to the postgres database
# note that "con" will be used later in each connection to the database
con <- dbConnect(drv, dbname = "Messdaten",
                 host = "localhost", port = 5432,
                 user = "postgres", password = pw)
rm(pw) # removes the password

qry <- {"select * from klima_messdaten inner join messdaten on klima_messdaten.did = messdaten.did"
} #Pegelmessdaten query
```

Abbildung 11) Verbindung zur Datenbank

Nach dem einladen der benötigten Daten aus der Datenbank, konnten die Jahre- und Wochenmittel für Pegel- und Klimamessstationen automatisiert berechnet werden (siehe Abbildung 12). Bei der Berechnung wurden die Werte nach Stationen gruppiert.

```
for (i in klimate_stations){ #month
  print(i)
  df_list = list()
  n = 1
  for (j in mon_duration){  #station
    print(j)
    temp <- relevant[which(relevant$mon == j & relevant$stations_id == i),,drop = FALSE]
    df_list[[n]] <- temp
    n = n + 1
      if (j == mon_duration[12]){
        temp2 <- rbind_list(df_list)
        fm <- sum(temp2$fm)/NROW(temp2)
        rsk <- sum(temp2$rsk)/NROW(temp2)
        fx <- sum(temp2$fx)/NROW(temp2)
        sdk <- sum(temp2$sdk)/NROW(temp2)
        station <- temp2$stations_id[1]
        date <- paste0(substr(temp2[1,]$datum,3,4),"/",
                       substr(temp2[NROW(temp2),]$datum,3,4),
                       "_",temp2$mon[1],
                       "_",temp2$mon[NROW(temp2)])
        year_list[[q]] <- assign(paste0("klima", i),
               tibble(fm,rsk,fx,sdk,station,date))
        q = q +1
        print("full year")
        break
      }
    }
  }
```

Abbildung 12) Berechnung der Jahresmittel - Klima

Zu erwähnen ist das für die Ergebnisstabelle der Wochenmittelwerte der Pegelstationen, die Variablen für Start- und Endzeitpunkt der Berechnungen je Datensatz hinzugefügt wurde. Zusätzlich wurde die Variable Full_Week angefügt, welche angibt ob die Berechnung über den Zeitraum einer vollen Woche durchgeführt wurde, da es im beim einpflegen aktueller Messdaten dazukommen kann das noch keine vollständigen Wochendaten vorliegen.

Nach der Berechnung wurden die Ergebnistabellen über die offene Datenbankverbindung in die Messdatenbank eingepflegt (siehe Abbildung 13).

```
dbWriteTable(con, "jahreswerte_klima", rbind_list(year_list))
```

Abbildung 13) Speicherung der Mittelwerte

Die Programmiersprache R wurde wegen ihrer Stärke in der statischen Auswertung von Daten für diese Aufgabe gewählt. Alle erstellten Skripte zur Berechnung der Mittelwerte befinden sich auf dem beigelegten Datenträge und können darüber hinaus in der Github Reposetory eingesehen werden.

Ergebnissichten

Nach dem einpflegen der Jahres- und Wochenmittelwerte mussten lediglich die Sichten für Klima Jahresmittel und Pegel Wochenmittel erstellt werden (siehe Abbildung 14).

```
Drop view if exists pegel_wochenwerte;
Drop view if exists klima_jahreswerte;

create view pegel_wochenwerte as select * from wochenwerte_pegel where full_week = TRUE;
create view klima_jahreswerte as select * from jahreswerte_klima;
```

Abbildung 14) Erstellung der Sichten

	row.names text	fm double precision	rsk double precision	fx double precision	sdk double precision	station text	date text
1	1	6.20710382513661	1.62513661202186	12.8854280510018	5.42272859744991	5516	16/17_04_03
2	2	9.81638733705773	4.48957169459963	18.5608938547486	4.30618249534451	722	16/17_04_03
3	3	6.69889502762431	5.3609576427256	17.7145488029466	5.39069797421731	5792	16/17_04_03

Abbildung 15) Sicht Jahresmittel-Klima

	row.names text	station double precision	durchfluss double precision	wasserstand double precision	start date	end date	full_week boolean
1	1	57521	1.30797872340426	30.8936170212766	2017-10-11	2017-10-17	TRUE
2	2	42012	26.068085106383	96.2340425531915	2017-10-11	2017-10-17	TRUE
3	3	57027	20.848829787234	69.2446808510638	2017-10-11	2017-10-17	TRUE
4	4	57521	1.30797872340426	30.8936170212766	2017-10-18	2017-10-24	TRUE
5	5	42012	26.068085106383	96.2340425531915	2017-10-18	2017-10-24	TRUE
6	6	57027	20.848829787234	69.2446808510638	2017-10-18	2017-10-24	TRUE
7	7	57521	1.30797872340426	30.8936170212766	2017-10-25	2017-10-31	TRUE
8	8	42012	26.068085106383	96.2340425531915	2017-10-25	2017-10-31	TRUE
9	9	57027	20.848829787234	69.2446808510638	2017-10-25	2017-10-31	TRUE
10	10	57521	1.30797872340426	30.8936170212766	2017-11-01	2017-11-07	TRUE

Abbildung 16) Sicht Wochenmittel-Pegel

8 Zusammenfassung

In diesem Geodatenbankprojekt wurde ein PostrgeSQL/PostGIS Datenbank erstellt. Dies umfasste den Planungsprozess, mit der Erstellung eines konzeptionellen Datenbankmodels und dem daraus folgenden physischen Entity-Relationshipmodel, sowie die Datenbeschaffung und Vorverarbeitung. Das finalisierte Model wurden schließlich in PostgreSQL umgesetzt und mit den aufbereiten Daten befüllt. Im letzten Schritt wurde die Funktionalität der Datenbank und des Datenbank-Triggers durch Testabfragen und der Erstellung von Datenbanksichten kontrolliert.

Literatur

BKG – Bundesamt für Kartographie und Geodäsie (2011).

Verwaltungsgebiete und Verwaltungsgrenzen Deutschland. Abgerufen von: https://www.bkg.bund.de/DE/Produkte-und-Services/Shop-und-Downloads/Digitale-Geodaten/Verwaltungsgebiete-Verwaltungsgrenzen/verwaltungsgebiete_cont.html

DWD. (2017a).

Klimadaten Deutschland. Abgerufen von: http://www.dwd.de/DE/leistungen/klimadatendeutschland/klimadatendeutschland.html

DWD. (2017b).

Klimastationen Deutschland. Abgerufen von:

https://www.dwd.de/DE/leistungen/klimadatendeutschland/statliste/statlex_rich.txt?view

=nasPublication&nn=16102

Elmasri, R. A. & S.B. Navathe S.B. (2009³).

Grundlagen von Datenbanksystemen. München: Pearson Education Deutschland GmbH.

J. Conway, D. Eddelbuettel, T. Nishiyama, S. K. Prayaga & N. Tiffin (2017).

RPostgreSQL: R Interface to the 'PostgreSQL' Database System. R package version 0.6-2.

Abgerufen von: https://CRAN.R-project.org/package=RPostgreSQL

PostgreSQL-Wiki (2012).

Audit trigger. Abgerufen von: https://wiki.postgresql.org/wiki/Audit_trigger

TULG. (2017).

Pegeldaten. Abgerufen von: http://hnz.thueringen.de/hw2.0/datenladen.html