Friedrich-Alexander Universität Erlangen-Nürnberg
Institut für Geographie
Methodikseminar Geodatenbanken
Dozent: Christian Bittner
Wintersemester 2014/2015

Umsetzung von Prozessen der Stadtentwicklung in OSM

Dokumentation Arbeitsschritte

15.03.2015

Sebastian Fischer
MA Kulturgeographie
2. Semester
Sebastian.Fischer@fau.de

Christoph Götz
MA Kulturgeographie
3. Semester
christoph.goetz@fau.de

Inhalt

Die Reihenfolge der Abfragen ist grundsätzlich chronologisch angeordnet. Lediglich das Erstellen des Datensatzes mit allen Objekten wurde der Übersicht halber vorgezogen. Die Analyseabfragen sind nach der Fragestellung angeordnet. Sofern nicht anders angegeben, wurden alle grau hinterlegten Abfragen in PostgreSQL 9.1 durchgeführt.

1. Aktueller Datensatz zu Erlangen vorbereiten

Für erste Tests wurde ein Datensatz mit den aktuellen Daten Erlangens angelegt. Dieser wurde anschließend aber nicht weiter verwendet.

1.1. Daten von Overpass-Turbo herrunterladen

Ausgabe der Open Street Map Daten zu Erlangen als .osm Datei von der Web-Applikation Overpass-Turbo (http://overpass-turbo.eu/).

http://overpass-api.de/api/interpreter?data=((area[name="Erlangen"];node(area););(area[name="Erlangen"];(way(area);>;););(area[name="Erlangen"];(rel(area);>;);););out meta;

1.2. Import nach PostgreSQL durch ein Osmosis Import Script

Anpassen des Osmosis Import Skripts (Dateipfade und Datenbanknamen anpassen) und Ausführen des Skripts.

1.3. Verbindung in QuantumGIS einrichten

Im Menü: Layer → PostGIS-Layer hinzufügen →Neue Verbindung einrichten:

- Name: erlangen
- Dienst: localhost
- Port: 5432
- Datenbank: erlangen
- Benutzername: postgres
- Passwort: pg2014

→Mit Datenbank verbinden

→ Layer auswählen und hinzufügen

→ OpenLayers Plugin aktivieren für BaseMap

1.4. Polygon der Gemeindegrenzen erstellen

um herauszufinden, ob die Daten nur die kreisfreie Stadt Erlangen oder auch den Landkreis Erlangen-Höchstadt umfassen (Polygon wurde nicht weiter verwendet).

```
create table city_ways as
select * from
(select * from relations where tags @> 'boundary=>administrative')
as relations
cross JOIN
(Select * FROM relation_members Where member_type = 'W')
as members
where relations.id = members.relation_id;

create table city_ways_geo as
Select * from city_ways
left JOIN
(Select linestring, id as way_id from ways) as line_geom
on member_id = line_geom.way_id;
create table poly_temp as
select relation_id, (st_polygonize(linestring)) as poly_gc from city_ways_geo
group by relation_id;

create table city_boundary_polygon as
```

```
Select polygons.relation_id, polygons.polygon, relations.version, relations.user_id,
relations.tstamp, relations.changeset_id, relations.tags from
(select relation id, ST CollectionExtract(poly gc, 3) as polygon from poly temp) as polygons
LEFT JOIN
(select id, version, user_id, tstamp, changeset_id, tags from relations) as relations
ON polygons.relation_id = relations.id;
drop table city_ways;
drop table city_ways_geo;
drop table poly_temp;
select Populate_Geometry_Columns();
```

1.5. Timeslide für Gebäude

Für einen ersten Eindruck: Erstellung einer Tabelle mit allen als Gebäude getaggten Objekten zur weiteren Visualisierung in QGIS mit Hilfe des TimeManager-Plugins.

```
create table building_time as
Select v.*, LOCALTIMESTAMP from ways v
where tags ? 'building';
select populate_geometry_columns()
```

In QGIS das TimeManager Plugin aktivieren (vorher mussten die Einstellungen angepasst werden, damit auch experimentelle Plugins angezeigt werden).

2. History Datensatz zu Erlangen vorbereiten

2.1. Bounding Box für history Datensatz festlegen

mit Hilfe der Web-Applikation http://boundingbox.klokantech.com/ wurden die folgenden Koordinaten als Eckpunkte der bounding box festgelegt. Alle Objekte, welche innerhalb dieser Box liegen wurden anschließend aus einem globalen history-Datensatz ausgeschnitten. Diese Box entspricht nicht parzellenscharf, aber relativ genau den Gemeindegrenzen Erlangens.

```
westlimit=10.926178; southlimit=49.53277; eastlimit=11.064676; northlimit=49.621987
```

2.2. Import der sql-Datei nach PostgreSQL

Anlegen einer neuen Datenbank in der Eingabeaufforderung.

```
createdb -p 5432 -U postgres erlangen
```

Import des history Datensatzes in PostgreSQL über die Eingabeaufforderung

```
psql -U postgres -d erlangen -f D:\Geodatenbanken_Goetz_Fischer\erlangen.sql
```

2.3. Shapefile erstellen in QGIS

Erstellen von shapefiles der jeweiligen Untersuchungsgebiete Büchenbach und Röthelheimpark in QGIS; das shapefile dient als „Schablone“ zum Ausschneiden eines Datensatzes zu den jeweiligen Untersuchtungsgebieten (2.7); als Grundlage der Gebiete dienen die Informationen der Stadt Erlangen, bzw. die Grenzen der jeweiligen Baugebiete.

2.4. Shapefile importieren

Importieren der shapefiles in PostgreSQL

```
shp2pgsql -I -s 4326 D:\Geodatenbanken_Goetz_Fischer\buechen_test.shp | psql -d erlangen -U
postgres

shp2pgsql -I -s 4326 D:\Geodatenbanken_Goetz_Fischer\roethel_test.shp | psql -d erlangen -U
postgres
```

2.5. Geometrien der Tabellen umprojizieren

um die shapefiles mit den Datensätzen verbinden zu können, müssen diese in derselben Projektion vorliegen. Zuerst muss der bestehende constraint „enforce_srid_the_geom" (unter [Name der Tabelle] > Contraints) gelöscht/gedroppt werden, anschließend wird die Geometrie der Tabellen transformiert.

```
UPDATE hist_point
SET geom = ST TRANSFORM (geom, 4326);
select Populate_Geometry_Columns();

UPDATE hist_line
SET geom = ST_TRANSFORM (geom, 4326);
select Populate_Geometry_Columns();

UPDATE hist_polygon
SET geom = ST TRANSFORM (geom, 4326);
select Populate_Geometry_Columns();
```

2.6. Id Spalte aus Tabellen entfernen

Entfernen der nicht benötigten Spalte „id" aus den shapefile Tabellen, da der Schritt 2.7. nicht möglich ist, wenn in beiden Tabellen eine Spalte mit demselben Namen vorliegt (die „id" wird lediglich für die Anzeige in einem Geoinformationssystem benötigt).

```
alter Table roethel test
drop column id;

alter Table buechen_test
drop column id;
```

2.7. Datensatz zu den Untersuchungsgebieten erstellen

Tabellen mit allen points, lines bzw. polygons, welche im Raumausschnitt des jeweiligen shapefiles liegen.

Büchenbach

```
create table buechen_point as
select * from hist point,buechen test where
st_contains(buechen_test.the_geom,hist_point.geom)=true;

create table buechen_line as
select * from hist line,buechen test where
st_contains(buechen_test.the_geom,hist_line.geom)=true;

create table buechen_polygon as
select * from hist_polygon,buechen_test where
st_contains(Buechen_test.the_geom,hist_polygon.geom)=true;

select Populate_Geometry_Columns();
```

Röthelheimpark

```
create table roethel_point as
select * from hist_point,Roethel_test where
st_contains(Roethel_test.the_geom,hist_point.geom)=true;

create table roethel line as
select * from hist_line,Roethel_test where
st_contains(Roethel_test.the_geom,hist_line.geom)=true;

create table roethel polygon as
select * from hist_polygon,Roethel_test where
st_contains(Roethel_test.the_geom,hist_polygon.geom)=true;

select Populate_Geometry_Columns();
```

2.8. Umbenennen der shapefiles

Umbenennen der shapefiles in PostgreSQL, um Verwechslungen oder versehentliches Löschen zu vermeiden.

```
ALTER TABLE buechen_test RENAME TO buechen_shp;
ALTER TABLE roethel_test RENAME TO roethel_shp;
```

2.9. Tabelle mit allen Objekten der Untersuchungsgebieten erstellen

Erstellen einer Tabelle mit allen Objekten (points, lines und polygons) für die jeweiligen Gebiete zur Erleichterung der Abfragen (sonst müssten diese für alle drei Typen getrennt durchgeführt werden und anschließend zusammengeführt werden). Da für den join alle Tabellen dieselben Spalten (Anzahl, Typ und Name) haben müssen, werden die Points und Lines zuerst um weitere leere Spalten erweitert.

Büchenbach

```
ALTER TABLE buechen point
ADD COLUMN minor smallint;

ALTER TABLE buechen_point
ADD COLUMN z_order Integer;

ALTER TABLE buechen_point
ADD COLUMN area real;

ALTER TABLE buechen_line
ADD COLUMN area real;

create table all_buechen_objects as
select id, version, minor, visible, user id, user name, valid from, valid to, tags, z order,
area, 'point' as type from
buechen_point
UNION
select id, version, minor, visible, user id, user name, valid from, valid to, tags, z order,
area, 'line' as type from
buechen_line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
buechen_polygon;

select Populate Geometry Columns();
```

Röthelheimpark

```
ALTER TABLE roethel_point
ADD COLUMN minor smallint;

ALTER TABLE roethel_point
ADD COLUMN z_order Integer;

ALTER TABLE roethel point
ADD COLUMN area real;

ALTER TABLE roethel_line
ADD COLUMN area real;

create table all_roethel_objects as
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'point' as type from
roethel_point
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'line' as type from
roethel line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
```

```
roethel_polygon;

select Populate_Geometry_Columns();
```

Erlangen

```
ALTER TABLE hist_point
ADD COLUMN minor smallint;

ALTER TABLE hist point
ADD COLUMN z_order Integer;

ALTER TABLE hist_point
ADD COLUMN area real;

ALTER TABLE hist_line
ADD COLUMN area real;

create table all_erlangen_objects as
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'point' as type from
buechen_point
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'line' as type from
hist_line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
hist_polygon;

select Populate_Geometry_Columns();
```

3. Analyse der History Datensatzes

3.1. Timeslide für die Gebäude

Erstellen einer Tabelle mit allen als Gebäude getaggten Objekten zur weiteren Visualisierung in QGIS mit Hilfe des TimeManager-Plugins.

Büchenbach

```
create table buechen_building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from buechen_polygon v
where tags ? 'building';
select populate_geometry_columns()
```

Röthelheimpark

```
create table roethel_building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from roethel_polygon v
where tags ? 'building';
select populate_geometry_columns()
```

Erlangen

```
create table building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from hist_polygon v
where tags ? 'building';
select populate_geometry_columns()
```

3.2. Timeslide für die Gebäude – Anpassungen valid_to

Anpassen der bei 3.1 erstellten Tabellen: im history-Datensatz gibt es die Zeitfelder „valid_from“ (Zeitpunkt der Erstellung) und „valid_to“ (Zeitpunkt der Deaktivierung); Objekte, welche aktuell noch aktiv sind, besitzen keinen Wert bei „valid_to“ und würden folglich bei einer Visualisierung im TimeManager mit Startpunkt „valid_from“ und Endpunkt „valid_to“ nicht angezeigt. Um die Anzeige

möglich zu machen, wird, wenn für die Spalte „valid_to“ kein Wert angegeben ist, in dieses Feld das Datum „2014-12-31“ eingesetzt. Um die ursprüngliche Tabelle zu erhalten, wurde erst eine Kopie angelegt.

Büchenbach

```
create table buechen_building_time_hist_polygon_to as
select * from buechen_building_time_hist_polygon

update buechen_building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

Anschließend Ausgabe des Ergebnisses als Bilderserie (Funktion „export video“ im TimeManger) im Intervall zehn Tage (weiter verwendet) und im Intervall ein Tag.

Röthelheimpark

```
create table roethel_building_time_hist_polygon_to as
select * from roethel_building_time_hist_polygon

update roethel_building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

Anschließend Ausgabe des Ergebnisses als Bilderserie (Funktion „export video“ im TimeManger) im Intervall zehn Tage (weiter verwendet) und im Intervall ein Tag.

Erlangen

```
create table building_time_hist_polygon_to as
select * from building_time_hist_polygon

update building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

Anschließend Ausgabe des Ergebnisses als Bilderserie (Funktion „export video“ im TimeManger) im Intervall zehn Tage (weiter verwendet) und im Intervall ein Tag.

3.3. Zeitreihen aller Objekte erstellen

Erstellen einer Tabelle mit allen Objekten pro Monat und einer Tabellen mit allen Monaten und anschließend Zusammenfügen beider Tabellen und Einsetzen einer „0“ bei leeren Monaten um Visualisierung möglich zu machen; abschließend Zwischenergebnisse löschen und als .csv Tabelle für weitere Berechnung und Visualisierung in Excel exportieren.

Büchenbach

```
create table months count temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_buechen_objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
    date_trunc('month',(select min(valid_from) from all_buechen_objects)),
    date_trunc('month',(select max(valid_from) from all_buechen_objects)),
    '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time series temp.series = months count temp.date trunc;

CREATE Table time_series_buechen AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
```

```
 END
from
series count temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_buechen
TO 'D:\time_series_buechen.csv' DELIMITER ',' CSV HEADER;
```

Röthelheimpark

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_roethel_objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
    date_trunc('month',(select min(valid_from) from all_roethel_objects)),
    date_trunc('month',(select max(valid_from) from all_roethel_objects)),
    '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months count temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_roethel AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_roethel
TO 'D:\time_series_roethel.csv' DELIMITER ',' CSV HEADER;
```

Erlangen

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_erlangen_objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_erlangen_objects)),
       date_trunc('month',(select max(valid_from) from all_erlangen_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_erlangen AS
select series,
```

```
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_erlangen
TO 'D:\time_series_erlangen.csv' DELIMITER ',' CSV HEADER;
```

3.4. Zeitreihen der Erstversionen erstellen

wie 3.3 Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version

Büchenbach

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_buechen_objects where version=1
GROUP BY date trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_buechen_objects)),
       date_trunc('month',(select max(valid_from) from all_buechen_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months count temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_buechen_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months count temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time series buechen v1
TO 'D:\time_series_buechen_v1.csv' DELIMITER ',' CSV HEADER;
```

Röthelheimpark

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all roethel objects where version=1
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_roethel_objects)),
       date_trunc('month',(select max(valid_from) from all_roethel_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
```

```
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_roethel_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_roethel_v1
TO 'D:\time_series_roethel_v1.csv' DELIMITER ',' CSV HEADER;
```

Erlangen

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_erlangen_objects where version=1
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_erlangen_objects)),
       date_trunc('month',(select max(valid_from) from all_erlangen_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_roethel_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_erlangen_v1
TO 'D:\time_series_erlangen_v1.csv' DELIMITER ',' CSV HEADER;
```

3.5. Edits zählen – alle Versionen

Auszählen der Edits nach Usernamen zusammengefasst und nach der Anzahl der Edits absteigend sortiert; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL.

Büchenbach

```
create table anzahl_edits_buechen as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_buechen_objects
GROUP BY user_name
```

```
ORDER BY anzahl_edits DESC;

COPY anzahl edits buechen
TO 'D:\anzahl_edits_buechen.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_buechen;
```

Röthelheimpark

```
create table anzahl_edits_roethel as
SELECT COUNT (id) AS anzahl edits, user name
FROM all_roethel_objects
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_roethel
TO 'D:\anzahl_edits_roethel.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_roethel;
```

Erlangen

```
create table anzahl_edits_erlangen as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_erlangen_objects
GROUP BY user_name
ORDER BY anzahl edits DESC;

COPY anzahl_edits_erlangen
TO 'D:\anzahl_edits_erlangen.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_erlangen;
```

3.6. Edits zählen – erste Version

wie 3.5 Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version.

Büchenbach

```
create table anzahl_edits_buechen_v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_buechen_objects where version=1
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_buechen_v1
TO 'D:\anzahl_edits_buechen_v1.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_buechen_v1;
```

Röthelheimpark

```
create table anzahl edits roethel v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_roethel_objects where version=1
GROUP BY user_name
ORDER BY anzahl edits DESC;

COPY anzahl_edits_roethel_v1
TO 'D:\anzahl_edits_roethel_v1.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_roethel_v1;
```

Erlangen

```
create table anzahl_edits_erlangen_v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_erlangen_objects where version=1
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_erlangen_v1
TO 'D:\anzahl_edits_erlangen_v1.csv' DELIMITER ',' CSV HEADER;

drop table anzahl_edits_erlangen_v1;
```

3.7. Maximale Versionsnummer ausgeben

Maximaler Versionsnummer aller Objekte ausgeben.

Büchenbach

```
select max(version)
from all_buechen_objects
```

Röthelheimpark

```
select max(version)
from all_roethel_objects
```

Erlangen

```
select max(version)
from all_erlangen_objects
```

3.8. Durchschnittliche Versionsnummer ausgeben

Durchschnittliche Versionsnummer aller Objekte ausgeben.

Büchenbach

```
select avg(version)
from all_buechen_objects
```

Röthelheimpark

```
select avg(version)
from all_roethel_objects
```

Erlangen

```
select avg(version)
from all_erlangen_objects
```

3.9. Anzahl der Tag & Value Kombinationen

Auszählen der Tag & Value Kombinationen (z.B. „building=yes“) und Anordnung nach der Anzahl absteigend; außerdem Berechnen der Summe der vergebenen Tag & Value Kombinationen insgesamt.

Büchenbach

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all buechen objects v
group by werte
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_buechen_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

Röthelheimpark

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_roethel_objects v
group by werte
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_roethel_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

Erlangen

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_erlangen_objects v
group by werte
```

```
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_erlangen_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

3.10. Anzahl der Tags pro Objekt

Auszählen der Anzahl der Tags für jedes Objekt; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL

Büchenbach

```
create table tag_count_buechen as
Select *, array_length(akeys(v.tags),1) as tag_count from all_buechen_objects as v
order by tag_count desc;

COPY tag_count_buechen
TO 'D:\tag_count_buechen.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_buechen;
```

Röthelheimpark

```
create table tag_count_roethel as
Select *, array_length(akeys(v.tags),1) as tag_count from all_roethel_objects as v
order by tag_count desc;

COPY tag_count_roethel
TO 'D:\tag_count_roethel.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_roethel;
```

Erlangen

```
create table tag count erlangen as
Select *, array_length(akeys(v.tags),1) as tag_count from all_erlangen_objects as v
order by tag_count desc;

COPY tag_count_erlangen
TO 'D:\tag_count_erlangen.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_erlangen;
```

4. Weitere Berechnungen und Visualisierung

Timeslide in Adobe Premiere

- Erstellen eines Videos aus den durch das TimeManagerPlugin ausgegebenen Screenshots (da die Funktion „export video“ nur diese Bilder ausgibt, aber kein Video)
- Hinzufügen der fehlenden Datumsangaben
- Export als Full-HD .mp4

Zeitreihen in Excel (jeweils für alle und nur erste Versionen)

- Visualisierung der Zeitreihe als Diagramm (nur von allen Versionen weiterverwendet)

Anzahl der Edits in Excel (jeweils für alle und nur erste Versionen)

- Berechnung der Summe der Objekte
- Berechnung des Mittelwerts der Edits (Edits pro User)
- Berechnung des Medians der Edits
- Berechnung der Prozent der User, welche mehr als 50% editiert haben
- Berechnung der Prozent der User, welche mehr als 90% editiert haben
- Berechnung wie viel der vorhergehende User jeweils mehr Objekte editiert hat als der nachfolgende

Anzahl der Tags & Value Kombinationen

- Berechnung Verhältnis Anzahl Werte zu Anzahl Objekte (händisch)

Anzahl der Tags pro Objekt in Excel

- Berechnung der Häufigkeitsverteilung

Karten der Untersuchungsgebiete in ArcGis

- mit Hilfe der erstellten shapefiles
- eine Erstellung in QGIS war auf Grund eines bekannten und dokumentierten Bugs des OpenLayersPlugins beim PDF Export nicht möglich (http://hub.qgis.org/issues/2117 http://hub.qgis.org/issues/5827)

5. Abfragen, welche in der Analyse nicht weiter verwendet wurden

Timeslide Gebäude

aus points und lines; nicht verwendet, da nur wenige (zudem falsch) getaggte Gebäude

```
create table building_time_hist_point as
Select v.*, LOCALTIMESTAMP from hist_point v
where tags ? 'building';
select populate_geometry_columns()

create table building_time_hist_line as
Select v.*, LOCALTIMESTAMP from hist_line v
where tags ? 'building';
select populate_geometry_columns()

create table buechen_building_time_hist_point as
Select v.*, LOCALTIMESTAMP from buechen_point v
where tags ? 'building';
select populate_geometry_columns()

create table buechen_building_time_hist_line as
Select v.*, LOCALTIMESTAMP from buechen_line v
where tags ? 'building';
select populate_geometry_columns()

create table roethel_building_time_hist_point as
Select v.*, LOCALTIMESTAMP from roethel_point v
where tags ? 'building';
select populate_geometry_columns()

create table roethel_building_time_hist_line as
Select v.*, LOCALTIMESTAMP from roethel_line v
where tags ? 'building';
select populate_geometry_columns()
```

Timeslide Landuse

für Landnutzung; nicht verwendet, da nur sehr wenige Objekte und nur einzelne mit Tag „construction“

```
create table landuse_roethel_time as
Select v.*, LOCALTIMESTAMP from roethel_polygon v
where tags ? 'landuse';

create table landuse_buechen_time as
Select v.*, LOCALTIMESTAMP from buechen_polygon v
where tags ? 'landuse';
```

```
select populate_geometry_columns();
```

Timeslide Straßen

für Straßen als Alternative zu Gebäuden; nicht verwendet, da Gebäude weit häufiger vorhanden sind und Straßen eher eine Grundausstattung der Karte sind

```
create table roethel highway time hist line as
Select v.*, LOCALTIMESTAMP from roethel_line v
where tags ? 'highway';
```

Edits Zählen – alle Versionen

für points, lines und polygons; nicht verwendet, da später Tabelle mit allen Objekten erstellt

```
create table anzahl_polygons_buechen as
SELECT COUNT (id) AS anzahl polygons, user_name
FROM buechen_polygon
GROUP BY user_name
ORDER BY anzahl_polygons DESC;

COPY anzahl_polygons_buechen
TO 'D:\anzahl_polygons_buechen.csv' DELIMITER ',' CSV HEADER;

drop table anzahl_polygons_buechen;


create table anzahl_points_buechen as
SELECT COUNT (id) AS anzahl_points, user_name
FROM buechen_point
GROUP BY user_name
ORDER BY anzahl_points DESC;
COPY anzahl points buechen
TO 'D:\anzahl_points_buechen.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_points_buechen;


create table anzahl_lines_buechen as
SELECT COUNT (id) AS anzahl_lines, user_name
FROM buechen_line
GROUP BY user_name
ORDER BY anzahl_lines DESC;

COPY anzahl_lines_buechen
TO 'D:\anzahl_lines_buechen.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_lines_buechen;


create table anzahl_polygons_roethel as
SELECT COUNT (id) AS anzahl polygons, user_name
FROM roethel_polygon
GROUP BY user_name
ORDER BY anzahl_polygons DESC;

COPY anzahl_polygons_roethel
TO 'D:\anzahl_polygons_roethel.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_polygons_roethel;


create table anzahl_points_roethel as
SELECT COUNT (id) AS anzahl_points, user_name
FROM roethel point
GROUP BY user_name
ORDER BY anzahl_points DESC;

COPY anzahl_points_roethel
TO 'D:\anzahl points roethel.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_points_roethel;


create table anzahl_lines_roethel as
```

```
SELECT COUNT (id) AS anzahl_lines, user_name
FROM roethel_line
GROUP BY user name
ORDER BY anzahl_lines DESC;

COPY anzahl_lines_roethel
TO 'D:\anzahl_lines_roethel.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_lines_roethel;
```

Maximale Versionsnummer

für Points, Lines und Polygons; nicht verwendet, da später Tabelle mit allen Objekten erstellt

```
select max(version)
from roethel_point

select max(version)
from roethel_line

select max(version)
from roethel_polygon


select max(version)
from buechen_point

select max(version)
from buechen_line

select max(version)
from buechen_polygon
```

durchschnittliche Versionsnummer

für Points, Lines und Polygons; nicht verwendet, da später Tabelle mit allen Objekten erstellt

```
select avg(version)
from roethel_point

select avg(version)
from roethel_line

select avg(version)
from roethel_polygon


select avg(version)
from buechen_point

select avg(version)
from buechen_line

select avg(version)
from buechen_polygon
```

6. Analysen weiterer Vergleichsgebiete

Beschreibung Vorgehen Shapefileserstellung

Erstellen von shapefiles der jeweiligen Vergleichsgebiete Büchenbach-Gesamt und Erlangen-Innenstadt in ArcGIS; das shapefile dient als „Schablone“ zum Ausschneiden eines Datensatzes zu den jeweiligen Vergleichsgebieten (6.1); als Grundlage der Gebiete dienen die Informationen der Stadt Erlangen (statistische Bezirke)

Importieren der shapefiles in PostgreSQL

```
shp2pgsql -I -s 4326 D:\GEOFG\Ueberarbeitung\SHPs\bbach.shp | psql -p 5434 -d erlangen_remake -U
postgres

shp2pgsql -I -s 4326 D:\GEOFG\Ueberarbeitung\SHPs\elong.shp | psql -p 5434 -d erlangen_remake -U
postgres
```

6.1. Datensatz zu den Vergleichsgebieten erstellen

Tabellen mit allen points, lines bzw. polygons, welche im Raumausschnitt des jeweiligen shapefiles liegen.

Büchenbach-Gesamt

```
create table bbach_point as
select * from hist_point,bbach
where st_contains(bbach.the_geom,hist_point.geom)=true;

create table bbach_line as
select * from hist_line,bbach
 where st_contains(bbach.the_geom,hist_line.geom)=true;

create table bbach_polygon as
select * from hist_polygon,bbach
 where st_contains(bbach.the_geom,hist_polygon.geom)=true;

select Populate_Geometry_Columns();
```

Erlangen Innenstadt

```
create table elong_point as
select * from hist_point,elong
where st_contains(elong.the_geom,hist_point.geom)=true;

create table elong_line as
select * from hist_line,elong_test
where st_contains(elong.the_geom,hist_line.geom)=true;

create table elong_polygon as
select * from hist_polygon,elong
where st_contains(elong.the_geom,hist_polygon.geom)=true;

select Populate_Geometry_Columns();
```

6.2. Tabelle mit allen Objekten der Vergleichsgebieten erstellen

Erstellen einer Tabelle mit allen Objekten (points, lines und polygons) für die jeweiligen Gebiete zur Erleichterung der Abfragen (sonst müssten diese für alle drei Typen getrennt durchgeführt werden und anschließend zusammengeführt werden).

Büchenbach-Gesamt

```
create table all_bbach_objects as
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'point' as type from
bbach_point
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'line' as type from
bbach line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
bbach_polygon;

select Populate_Geometry_Columns();
```

Erlangen Innenstadt

```
create table all_elong_objects as
```

```
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'point' as type from
elong point
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'line' as type from
elong_line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
elong polygon;

select Populate_Geometry_Columns();
```

6.3. Timeslide für die Gebäude

Erstellen einer Tabelle mit allen als Gebäude getaggten Objekten zur weiteren Visualisierung in QGIS mit Hilfe des TimeManager-Plugins.

Büchenbach-Gesamt

```
create table bbach_building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from bbach_polygon v
where tags ? 'building';
select populate_geometry_columns();
```

Erlangen Innenstadt

```
create table elong_building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from elong_polygon v
where tags ? 'building';
select populate_geometry_columns();
```

6.4. Timeslide für die Gebäude – Anpassungen valid_to

Anpassen der bei 6.3. erstellten Tabellen: im history-Datensatz gibt es die Zeitfelder „valid_from“ (Zeitpunkt der Erstellung) und „valid_to“ (Zeitpunkt der Deaktivierung); Objekte, welche aktuell noch aktiv sind, besitzen keinen Wert bei „valid_to“ und würden folglich bei einer Visualisierung im TimeManager mit Startpunkt „valid_from“ und Endpunkt „valid_to“ nicht angezeigt. Um die Anzeige möglich zu machen, wird, wenn für die Spalte „valid_to“ kein Wert angegeben ist, in dieses Feld das Datum „2014-12-31“ eingesetzt. Um die ursprüngliche Tabelle zu erhalten, wurde erst eine Kopie angelegt.

Büchenbach-Gesamt

```
create table bbach_building_time_hist_polygon_to as
select * from bbach_building_time_hist_polygon

update bbach_building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

Erlangen Innenstadt

```
create table elong_building_time_hist_polygon_to as
select * from elong_building_time_hist_polygon

update elong_building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

6.5. Zeitreihen aller Objekte erstellen

Erstellen einer Tabelle mit allen Objekten pro Monat und einer Tabellen mit allen Monaten und anschließend Zusammenfügen beider Tabellen und Einsetzen einer „0“ bei leeren Monaten um Visualisierung möglich zu machen; abschließend Zwischenergebnisse löschen und als .csv Tabelle für weitere Berechnung und Visualisierung in Excel exportieren.

Büchenbach-Gesamt

```
create table months_count_temp as
```

```
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_bbach_objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
    date_trunc('month',(select min(valid_from) from all_bbach_objects)),
    date_trunc('month',(select max(valid_from) from all_bbach_objects)),
    '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_bbach AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months count temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time series bbach
TO 'D:\time_series_bbach.csv' DELIMITER ',' CSV HEADER;
```

Erlangen Innenstadt

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all elong objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
    date_trunc('month',(select min(valid_from) from all_elong_objects)),
    date_trunc('month',(select max(valid_from) from all_elong_objects)),
    '1 month') as series;

create table series_count_temp as
select * from
time series temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_elong AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_elong
TO 'D:\time_series_elong.csv' DELIMITER ',' CSV HEADER;
```

6.6. Zeitreihen der Erstversionen erstellen

wie 6.5. Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version

Büchenbach-Gesamt

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_bbach_objects where version=1
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_bbach_objects)),
       date_trunc('month',(select max(valid_from) from all_bbach_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_bbach_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_bbach_v1
TO 'D:\time_series_bbach_v1.csv' DELIMITER ',' CSV HEADER;
```

Erlangen Innenstadt

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_elong_objects where version=1
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_elong_objects)),
       date_trunc('month',(select max(valid_from) from all_elong_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_elong_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;
```

```
drop table months_count_temp;
drop table time_series_temp;
drop table series count temp;

COPY time_series_elong_v1
TO 'D:\time_series_elong_v1.csv' DELIMITER ',' CSV HEADER;
```

6.7. Edits zählen – alle Versionen

Auszählen der Edits nach Usernamen zusammengefasst und nach der Anzahl der Edits absteigend sortiert; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL.

Büchenbach-Gesamt

```
create table anzahl_edits_bbach as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_bbach_objects
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_bbach
TO 'D:\anzahl_edits_bbach.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_bbach;
```

Erlangen Innenstadt

```
create table anzahl edits elong as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_elong_objects
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_elong
TO 'D:\anzahl_edits_elong.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_elong;
```

6.8. Edits zählen – erste Version

wie 6.7. Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version.

Büchenbach-Gesamt

```
create table anzahl_edits_bbach_v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_bbach_objects where version=1
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_bbach_v1
TO 'D:\anzahl_edits_bbach_v1.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_bbach_v1;
```

Erlangen Innenstadt

```
create table anzahl_edits_elong_v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_elong_objects where version=1
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_elong_v1
TO 'D:\anzahl_edits_elong_v1.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_elong_v1;
```

6.9. Maximale Versionsnummer ausgeben

Maximaler Versionsnummer aller Objekte ausgeben.

Büchenbach-Gesamt

```
select max(version)
from all_bbach_objects
```

Erlangen Innenstadt

```
select max(version)
from all_elong_objects
```

6.10. Durchschnittliche Versionsnummer ausgeben

Durchschnittliche Versionsnummer aller Objekte ausgeben.

Büchenbach

```
select avg(version)
from all_bbach_objects
```

Erlangen Innenstadt

```
select avg(version)
from all_elong_objects
```

6.11. Anzahl der Tag & Value Kombinationen

Auszählen der Tag & Value Kombinationen (z.B. „building=yes“) und Anordnung nach der Anzahl absteigend; außerdem Berechnen der Summe der vergebenen Tag & Value Kombinationen insgesamt.

Büchenbach-Gesamt

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_bbach_objects v
group by werte
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_bbach_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

Erlangen Innenstadt

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_elong_objects v
group by werte
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_elong_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

6.12. Anzahl der Tags pro Objekt

Auszählen der Anzahl der Tags für jedes Objekt; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL

Büchenbach-Gesamt

```
create table tag_count_bbach as
Select *, array_length(akeys(v.tags),1) as tag_count from all_bbach_objects as v
order by tag_count desc;

COPY tag_count_bbach
TO 'D:\tag_count_bbach.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_bbach;
```

Erlangen Innenstadt

```
create table tag_count_elong as
Select *, array_length(akeys(v.tags),1) as tag_count from all_elong_objects as v
```

```
order by tag_count desc;

COPY tag count elong
TO 'D:\tag_count_elong.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_elong;
```

7. Überarbeitung Erlangen Gesamt

Beschreibung Vorgehen Shapefileserstellung

Erstellen von einem shapefile des Vergleichsgebietes Erlangen-Gesamt in ArcGIS; das shapefile dient als „Schablone" zum Ausschneiden eines Datensatzes des Vergleichsgebietes (7.1); als Grundlage der Gebiete dienen die Informationen der Stadt Erlangen (statistische Bezirke)

Importieren der shapefiles in PostgreSQL

```
shp2pgsql -I -s 4326 D:\GEOFG\Ueberarbeitung\SHPs\erlangen_polygon.shp | psql -p 5434 -d
erlangen_remake -U postgres
```

7.1. Datensatz zum Vergleichsgebiet erstellen

Tabellen mit allen points, lines bzw. polygons, welche im Raumausschnitt des jeweiligen shapefiles liegen.

```
create table erlangen as
select ST_GeometryN(geom,generate_series(1,ST_NumGeometries(geom))) from erlangen_polygon;

create table erlangen_point as
select h.* from hist_point h ,erlangen e
where st_within(h.geom, e.st_gometryn);

create table erlangen line as
select h.* from hist_line h ,erlangen e
where st_within(h.geom, e.st_gometryn);

create table erlangen_polygon_all as
select h.* from hist polyogn h ,erlangen e
where st_within(h.geom, e.st_gometryn);

select Populate_Geometry_Columns();
```

7.2. Tabelle mit allen Objekten des Vergleichsgebiets erstellen

Erstellen einer Tabelle mit allen Objekten (points, lines und polygons) für die jeweiligen Gebiete zur Erleichterung der Abfragen (sonst müssten diese für alle drei Typen getrennt durchgeführt werden und anschließend zusammengeführt werden).

```
create table all_erlangen_objects as
select id, version, minor, visible, user id, user name, valid from, valid to, tags, z order,
area, 'point' as type from
erlangen_point
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'line' as type from
erlangen_line
UNION
select id, version, minor, visible, user_id, user_name, valid_from, valid_to, tags, z_order,
area, 'polygon' as type  from
erlangen_polygon_all;

select Populate_Geometry_Columns();
```

7.3. Timeslide für die Gebäude

Erstellen einer Tabelle mit allen als Gebäude getaggten Objekten zur weiteren Visualisierung in QGIS mit Hilfe des TimeManager-Plugins.

```
create table erlangen_building_time_hist_polygon as
Select v.*, LOCALTIMESTAMP from erlangen_polygon_all v
where tags ? 'building';
select populate_geometry_columns();
```

7.4. Timeslide für die Gebäude – Anpassungen valid_to

Anpassen der bei 7.3. erstellten Tabellen: im history-Datensatz gibt es die Zeitfelder „valid_from“ (Zeitpunkt der Erstellung) und „valid_to“ (Zeitpunkt der Deaktivierung); Objekte, welche aktuell noch aktiv sind, besitzen keinen Wert bei „valid_to“ und würden folglich bei einer Visualisierung im TimeManager mit Startpunkt „valid_from“ und Endpunkt „valid_to“ nicht angezeigt. Um die Anzeige möglich zu machen, wird, wenn für die Spalte „valid_to“ kein Wert angegeben ist, in dieses Feld das Datum „2014-12-31“ eingesetzt. Um die ursprüngliche Tabelle zu erhalten, wurde erst eine Kopie angelegt.

```
create table erlangen_building_time_hist_polygon_to as
select * from erlangen_building_time_hist_polygon

update erlangen_building_time_hist_polygon_to set valid_to ='2014-12-31' where valid_to is NULL;
```

7.5. Zeitreihen aller Objekte erstellen

Erstellen einer Tabelle mit allen Objekten pro Monat und einer Tabellen mit allen Monaten und anschließend Zusammenfügen beider Tabellen und Einsetzen einer „0“ bei leeren Monaten um Visualisierung möglich zu machen; abschließend Zwischenergebnisse löschen und als .csv Tabelle für weitere Berechnung und Visualisierung in Excel exportieren.

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_erlangen_objects
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
    date_trunc('month',(select min(valid_from) from all_erlangen_objects)),
    date_trunc('month',(select max(valid_from) from all_erlangen_objects)),
    '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months count temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_erlangen AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series_count_temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_erlangen
TO 'D:\time_series_erlangen.csv' DELIMITER ',' CSV HEADER;
```

7.6. Zeitreihen der Erstversionen erstellen

wie 7.5. Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version

```
create table months_count_temp as
SELECT COUNT(*), date_trunc('month', valid_from)
FROM all_erlangen_objects where version=1
GROUP BY date_trunc
ORDER BY date_trunc;

create table time_series_temp as
select generate_series(
       date_trunc('month',(select min(valid_from) from all_erlangen_objects)),
       date_trunc('month',(select max(valid_from) from all_erlangen_objects)),
       '1 month') as series;

create table series_count_temp as
select * from
time_series_temp
Left Join
months_count_temp
on time_series_temp.series = months_count_temp.date_trunc;

CREATE Table time_series_erlangen_v1 AS
select series,
 CASE
  WHEN count IS NULL THEN 0
  ELSE count
 END
from
series count temp;

drop table months_count_temp;
drop table time_series_temp;
drop table series_count_temp;

COPY time_series_erlangen_v1
TO 'D:\time_series_erlangen_v1.csv' DELIMITER ',' CSV HEADER;
```

7.7. Edits zählen – alle Versionen

Auszählen der Edits nach Usernamen zusammengefasst und nach der Anzahl der Edits absteigend sortiert; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL.

```
create table anzahl_edits_erlangen as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all erlangen objects
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl edits erlangen
TO 'D:\anzahl_edits_erlangen.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_erlangen;
```

7.8. Edits zählen – erste Version

wie 7.7. Zeitreihen für alle Versionen, jedoch nur für alle Objekte in ihrer ersten Version.

```
create table anzahl edits erlangen v1 as
SELECT COUNT (id) AS anzahl_edits, user_name
FROM all_erlangen_objects where version=1
GROUP BY user_name
ORDER BY anzahl_edits DESC;

COPY anzahl_edits_erlangen_v1
TO 'D:\anzahl_edits_erlangen_v1.csv' DELIMITER ',' CSV HEADER;
drop table anzahl_edits_erlangen_v1;
```

7.9. Maximale Versionsnummer ausgeben

Maximaler Versionsnummer aller Objekte ausgeben.

```
select max(version)
from all_erlangen_objects
```

7.10. Durchschnittliche Versionsnummer ausgeben

Durchschnittliche Versionsnummer aller Objekte ausgeben.

```
select avg(version)
from all_erlangen_objects
```

7.11. Anzahl der Tag & Value Kombinationen

Auszählen der Tag & Value Kombinationen (z.B. „building=yes“) und Anordnung nach der Anzahl absteigend; außerdem Berechnen der Summe der vergebenen Tag & Value Kombinationen insgesamt.

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_erlangen_objects v
group by werte
order by anzahl desc;
```

```
SELECT sum(anzahl) FROM
(SELECT each(v.tags) as werte, count(*) AS anzahl FROM all_erlangen_objects v
group by werte
order by anzahl desc) as werte_pro_item;
```

7.12. Anzahl der Tags pro Objekt

Auszählen der Anzahl der Tags für jedes Objekt; exportieren als .csv Tabelle für weitere Berechnung und Visualisierung in Excel und löschen der Tabelle in PostgreSQL

```
create table tag count erlangen as
Select *, array_length(akeys(v.tags),1) as tag_count from all_erlangen_objects as v
order by tag_count desc;

COPY tag count erlangen
TO 'D:\tag_count_erlangen.csv' DELIMITER ',' CSV HEADER;
drop table tag_count_erlangen;
```

8. Abschließende kleinere Vergleichsanalysen

Berechnen der Flächengröße der einzelnen Gebiete; Umrechnen in km^2 für die Berechnung der Anzahl der Objekte pro km^2

```
select ST_Area(the_geom, true) from elong
select ST_Area(the_geom, true) from bbach
select ST_Area(st_geometryn, true) from erlangen
select ST_Area(the_geom, true) from buechen_shp
select ST_Area(the_geom, true) from roethel_shp
```

Tabellen mit allen Edits im Röthelheimpark im Jan 14, bis Ende Dez 13 und bis Ende Jan 14 erstellen und Abfragen zu Tags durchführen

```
create table roethel_jan14 as
SELECT * FROM all_roethel_objects
WHERE valid_from>='2014-01-01' AND valid_from<='2014-01-31'

create table roethel_dec13 as
SELECT * FROM all_roethel_objects
WHERE valid_from<='2013-12-31'

create table roethel_jan142 as
SELECT * FROM all_roethel_objects
WHERE valid_from<='2014-01-31'
```

```
SELECT each(v.tags) as werte, count(*) AS anzahl FROM roethel_dec13 v
group by werte
order by anzahl desc;

SELECT each(v.tags) as werte, count(*) AS anzahl FROM roethel_jan14 v
group by werte
order by anzahl desc;

SELECT each(v.tags) as werte, count(*) AS anzahl FROM roethel_jan142 v
group by werte
order by anzahl desc;




SELECT * FROM all_roethel_objects
WHERE valid_from>='2014-01-01' AND valid_from<='2014-01-31' and version=1 and minor=0

select COUNT (id) As anzahl_edits, user_name
from roethel_jan14
Group by user_name
order by anzahl_edits DESC;
```

Tabellen mit allen Edits in Büchenbach im Aug 13 erstellen und Abfragen zu Usern durchführen

```
create table bbach_aug13 as
SELECT * FROM all_bbach_objects
WHERE valid_from>='2013-08-01' AND valid_from<='2013-08-31'

select COUNT (id) As anzahl_edits, user_name
from bbach_aug13
Group by user_name
order by anzahl_edits DESC;
```