Eva Zimmermann
Raimund Schnürer

Course: Geospatial Databases (Exercise)
Lecturers: Christian Timm, Catharina Riedemann
WS 2010/2011

Project report - A cadastral system for Alcazarpolis

Content

1. Requirements for the system
2. Conceptual model
3. Importing some data
4. Implementation of the data model
5. Implementation of processes
6. Using the data in a GIS-Client

1. Requirements for the system

The cadastral system for the city Alcazarpolis describes the relationship between parcels, buildings which lie on these parcels, and parties which own parcels and buildings. Each parcel has a specific usage. Depending on the usage type of a parcel and a ground value, the party has to pay taxes. A party can borrow on mortgage which is registered at a court. The city Alcazarpolis is organized in a hierarchical structure: it consists of two sub cities, boroughs and several blocks.

In the following, properties, relationships and operations of all objects are described:

Parties
- All parties are registered in the system
- Parties have a name.

Parcels
- Parcels are identified by an ID.
- Parcels are geometric objects (polygons).
- Parcels must not overlap.
- Each parcel has one usage type.
- Each usage type can appear several times.
- Usage type and geometry of a parcel can change.
- A parcel can be split into two parcels and two parcels can be merged into one parcel.

Buildings
- Buildings are identified by an ID.
- Buildings are geometric objects (polygons).
- Buildings must not overlap.
- Buildings cannot be split or merged. But the geometry of buildings can be changed.

Ground areas
- It is assumed that the ground values and their areas do not change often. Ground values are decimal numbers, they are between 1 and 100 and they are not unique.
- Ground areas are geometric objects (polygons).

Administrative boundaries
- There are different categories of administrative boundaries.
- Administrative boundaries of one category form a partition.
- Administrative boundaries are geometric objects (polygons).

Title documents

- Title documents are identified by a number.

Relationship parcel – party

- Each parcel has at least one owner (party).
- One party can possess no, one or more parcels.
- A parcel belongs to the city if there is not another owner.
- If a party buys such a parcel, the city loses its ownership.
- Ownerships of parcels can be transferred between parties.
- An ownership of a parcel is registered in a title document.
- If a parcel is deleted, all owners lose their ownership.

Relationships parcel – party – tax

- As there can be more than one owner of a parcel, each owner only owns a part (share). This share is only calculated when the taxes are calculated.
- The taxes are calculated once a year. Then each party must pay the taxes.

Relationship building – party

- Each building has at least one owner (party).
- One party can possess no, one or more buildings.
- A building belongs to the city if there is not another owner.
- If a party buys such a building, the city loses its ownership.
- Ownerships of buildings can be transferred between parties.
- An ownership of a building is registered in a title document.
- If a building is deleted, all owners lose their ownership.

Relationship parcel – building

- A building must lie on at least one parcel. But it can also lie on several parcels.
- A building or a part of a building must not lie on an area which is not a parcel.
- There can be none, one or more buildings on a parcel.
- A parcel can only be deleted if there is no building or a part of a building on it.

Relationships parcel – building – party

- A party can possess a parcel without possessing a building on this parcel. This is called a "direct ownership". If the party sells all buildings which lie on this parcel, it still owns the parcel.
- A party can possess a building without possessing the parcels the building is lying on. By possessing this building, the party indirectly possesses the parcels. If the party sells this building, it automatically loses the indirect ownership of the corresponding parcels.

Relationship party – mortgage

- A party can raise mortgages.
- A mortgage can only be raised by one party.

Relationship mortgage – court

- Each mortgage must be registered at a court.
- There can be courts no mortgage is registered at.

2. Conceptual model

After defining the requirements for the system, the conceptual model was made. The objects/entities and their relationships are modeled in an entity-relationship model (see appendix).

An important entity is the entity “party”. This entity has the minimal number of attributes. There is only one attribute for the party name. Alternatively there could be two attributes for the forename and the surname. Additionally one could add address attributes.

The main relationships are the relationships between the entities “parcel”, “building”, and “party”. The ownership is expressed by title documents. So each party which possesses a parcel or a part of a parcel, holds a parcel title document (“parcel_title_document”). If there is more than one owner of a parcel, there are several parcel title documents. There is also a building title document (“building_title_document”) to model the ownership of a building. Alternatively there could have been one title document for each ownership group. The relationship between buildings and parcels is expressed through the relation “buildings_parcels”. This relationship says which buildings lie on which parcels.

As there are pre-defined usage types with usage factors (used to calculate the taxes), they are modeled in an own entity (“usage”). Each parcel references one usage type.

In this model, there are two entities which describe the taxes. On the one hand there is the entity “tax_per_party” which lists all taxes the parties have to pay per year. There is the attribute “paid” to mark if a party has paid the taxes for a specified year. On the other hand there is the entity “tax_per_parcel”. This entity shows the taxes which have to be paid for parcels.

As a party can raise a mortgage to purchase a building, there are the entities “mortgage” and “court”. The entity “mortgage” is connected to the party and the buildings through relationships.

The administrative boundaries and the ground areas are modeled separately from all other entities. The relation to the other geometry objects is only expressed implicit through the geographical position. All geometry objects have the attribute “geometry”.

As the administrative boundaries are only distinguished between different hierarchical levels, they are modeled in one entity. Alternatively each hierarchical level could have been modeled in its own entity or by using an ID which shows the hierarchy.

3. Importing some data

To test if the system and implemented procedures work, some data was imported. Most of the data was provided by the lecturer in the form of a dump file. After extracting this data, the imported tables were analyzed. There are tables for the objects:

- T_BLD (buildings)
- T_ADM (administrative boundaries)
- T_PRCL (parcels)
- T_GV (ground areas)
- T_PARTY_IMP (parties)

and two relationship tables:

- T_PARTY_TO_BLD (relationship between parties and buildings)
- T_PARTY_TO_PRCL (relationship between parties and parcels)

The importing process follows the following procedure:

1. Deleting old data
2. Converting and adding the given and created data.

If available, the implemented procedures for deleting and inserting data, is used. This applies to the following processes:

• Deleting buildings: delete_building()
• Deleting parcels: delete_parcel()
• Deleting parties: delete_party()
• Adding parties: register_party()
• Adding parcel owners: add_parcel_owner()
• Creating parcels: create_parcel()
• Adding building owners: add_buidlings_owner()
• Creating buildings: create_buildings()
• Calculating taxes: calculate_taxes()

There aren't any procedures for the following processes because they are not executed very often:

• Deleting tax datasets (per party and per parcel)
• Deleting ground areas
• Deleting administrative boundaries
• Adding usage values
• Adding ground areas
• Adding administrative boundaries

Not all data, which was provided, was used. So the address of the parties was ignored. As there is only one name attribute for a party, the first and the second name had to be concatenated. For simplicity reasons, we left out composed IDs for parcels as well.

Since the provided data only contains ownerships with one party, some more data was added with ownership groups. So Tick, Trick, and Track own a building. They are co-owners. The direct and indirect ownership of a parcel can be seen: Donald Duck, Tick, Trick, and Truck possess buildings on the parcel 24. So they are indirect owners of this parcel. Additionally Donald Duck and Tick own this parcel directly, too.

4. Implementation of the data model

All in all, we could translate our entity-relationship-model quite straightforward into an implementation scheme for Oracle. We mapped entities to tables and attributes of the entities to columns of the tables. For 1:1 and 1:n relationships, we inserted foreign keys into tables as references. Only the building-parcel relationship, which is an n:m relationship, got its own table. As types for attributes, we used numbers at different scales and precisions, represented text with varying character arrays and stored geometries in the internal type of Oracle Spatial. To do spatial queries, we put spatial indices on the geometry columns and updated the metadata table. Besides this, we created tables to temporarily store IDs for complex queries in some procedures.

For each table, we applied different kinds of constraints. We used the primary key constraint for unique identifiers and the ‘not null’ assertion for obligatory values. For specific types (e.g. paid taxes or administrative boundary categories), we created a set of allowed values. Alternatively, we could have used here 1 and 0 instead of true and false and numbers for administrative boundary levels. In general, we could have even set up more constraints (like triggers) to ensure integrity of our model. But we decided to handle modifications of the database (e.g. inserting new parcels) and integrity checks (e.g. parcels cannot overlap) via procedures. This approach we chose in analogy to object-oriented programming where data should be encapsulated and only be accessible via functions.

In the following, we listed all tables we implemented.
The corresponding SQL statements can be found in the file “Schema.sql”.

PARTY

In this table, different parties can be stored.

Name	Type	Description
pk_party_id	NUMBER(38)	Technical ID
name	VARCHAR2(127)	Name of the party

BUILDING

Here, all buildings are stored.

Name	Type	Description
pk_building_id	NUMBER(38)	Technical ID
geometry	SDO_GEOMETRY	Geometry of the building

BUILDING_TITLE_DOCUMENT

All ownerships of buildings are registered here.

Name	Type	Description
pk_title_document_id	NUMBER(38)	Technical ID
fk_building_id	NUMBER(38)	Building
fk_party_id	NUMBER(38)	Building owner

USAGE

This table contains different types of parcel usages.

Name	Type	Description
pk_usage_id	NUMBER(38)	Technical ID
usage_type	VARCHAR2(16)	Usage of a Parcel: • residential area • commercial area • industrial area • combination • waste land • community land
usage_factor	NUMBER(2,1)	Corresponding factor for tax valuation

PARCEL

In this table, all parcels are stored.

Name	Type	Description
pk_parcel_id	NUMBER(38)	Technical ID
fk_usage_id	NUMBER(38)	Usage of the parcel
geometry	SDO_GEOMETRY	Geometry of the parcel

parcel_size	NUMBER(9,2)	Size of the parcel

PARCEL_TITLE_DOCUMENT

All ownerships of parcels are registered here.

Name	Type	Description
pk_title_document_id	NUMBER(38)	Technical ID
fk_parcel_id	NUMBER(38)	Parcel
fk_party_id	NUMBER(38)	Parcel owner

BUILDINGS_PARCEL

In this table, you can see which building lies on which parcel.

Name	Type	Description
fk_building_id	NUMBER(38)	Building
fk_parcel_id	NUMBER(38)	Parcel

ADMINISTRATIVE_BOUNDARY

This table contains different categories of administrative boundaries.

Name	Type	Description
pk_administrative_boundary_id	NUMBER(38)	Technical ID
name	VARCHAR2(127)	Name of the administrative boundary
category	VARCHAR2(7)	Administrative boundary level: 'city', 'subcity', 'borough', 'block'
geometry	SDO_GEOMETRY	Geometry of the administrative boundary

GROUND

In this table, ground values for different regions are stored.

Name	Type	Description
pk_ground_id	NUMBER(38)	Technical ID
value	NUMBER(5,2)	Ground value
geometry	SDO_GEOMETRY	Geometry of the region with that ground value

TAX_PER_PARTY

This table summarizes for different years the total amount of taxes which a party has to pay.

Name	Type	Description
pk_tax_per_party_year	NUMBER(4)	Tax year
fk_party_id	NUMBER(38)	Party
amount	NUMBER(9,2)	Tax amount to pay
paid	VARCHAR2(5)	Indicates if the tax has been paid, can be 'true' or false'

TAX_PER_PARCEL

For each parcel, the total amount of taxes is stored for different years.

Name	Type	Description
pk_tax_per_parcel_year	NUMBER(4)	Tax year
fk_parcel_id	NUMBER(38)	Parcel
amount	NUMBER(9,2)	Tax value of the parcel

COURT

This table could store different courts.

Name	Type	Description
pk_court_id	NUMBER(38)	Technical ID
name	NUMBER(5,2)	Name of the Court

MORTGAGE

This table could contain mortgages of buildings which can be borrowed by a party. Mortgages have different properties (like an interest rate) and are managed by a court.

Name	Type	Description
pk_mortgage_id	NUMBER(38)	Technical ID
fk_building_id _	NUMBER(38)	Building
fk_party_id	NUMBER(38)	Party
fk_court_id	NUMBER(38)	Court
interest_rate	NUMBER(4,2)	Interest rate of the mortgage
value	NUMBER(12,2)	Value of the mortgage
term	NUMBER(3)	Mortgage duration

TMP_PARTY_ID

In this table, party IDs can be stored temporarily.

Name	Type	Description
pk_party_id	NUMBER(38)	Technical ID

TMP_PARCEL_ID

In this table, parcel IDs can be stored temporarily.

Name	Type	Description
pk_parcel_id	NUMBER(38)	Technical ID

TMP_BUILDING_ID

In this table, building IDs can be stored temporarily.

Name	Type	Description
pk_building_id	NUMBER(38)	Technical ID

5. Implementation of processes

Underneath, descriptions are given to all procedures we realized.
You can find their source code in the folder “Procedures”.
All procedures are neatly tested in the file “TestProcedures.sql”.

1. Parcel mutation

CREATE_PARCEL

This procedure is used to create a new parcel. The ownership of the parcel will be registered in a title document. If there is no owner given, the city will be the owner. If there is no ID given, the ID will be auto generated.

Parameter	Type	Description
p_parcel_id	NUMBER	ID of the parcel
p_geometry	SDO_Geometry	Geometry of the parcel
usage_id	NUMBER	Usage of the parcel
p_party_id	NUMBER	Owner of the parcel

Exceptions are thrown when the parcel ID already exists, when there is no geometry given, when the geometry is not a polygon, when the usage ID or party ID does not exist or when the parcel would overlap with other parcels.

CHANGE_PARCEL_USAGE

With this procedure, the usage type of a parcel can be changed.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel
usage_id	NUMBER	Usage of the parcel

Exceptions are thrown when the parcel ID or usage ID does not exist or when the usage would not change.

CHANGE_PARCEL_GEOMETRY

With this procedure, the geometry of a parcel can be changed.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel
p_geometry	SDO_Geometry	Geometry of the parcel

Exceptions are thrown when there is no parcel ID or geometry given, when the geometry is not a polygon, when the parcel would overlap now with other parcels or when a building would not be entirely on the parcel anymore.

DELETE_PARCEL

With this procedure, you can delete a parcel. All ownerships of the parcel will be released.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel

Exceptions are thrown when the parcel ID does not exist, when there are still buildings on the parcel or when there are still taxes to pay for the parcel.

MERGE_PARCELS

With this procedure, you can merge two parcels. When buildings are on the parcels, the building-parcel relationship will be updated. The merged parcel gets the ID and usage type of the first parcel.

Parameter	Type	Description
parcel_id1	NUMBER	ID of the first parcel
parcel_id2	NUMBER	ID of the second parcel

Exceptions are thrown when one of the parcel IDs does not exist, when the parcels do not touch each other, when the parcels do not have the same owners or when there are still taxes to pay for those parcels.

SPLIT_PARCEL

This procedure can be used to split a part of a parcel. When buildings are on the parcel, the building-parcel relationship will be possibly updated. Usage and ownerships remain the same. If there is no ID given, the ID for the new parcel will be auto generated.

Parameter	Type	Description
p_parcel_id	NUMBER	ID of the parcel
s_geometry	SDO_Geometry	Geometry of the parcel to be split of
s_parcel_id	NUMBER	ID of the parcel to be split of

Exceptions are thrown when the parcel ID or geometry does not exist, when the geometry is not a polygon, when the parcel to be split of is not in a covered by relationship to the parcel, when the ID

of the parcel to be split of already exists or when there are still taxes which have not been paid for the parcel yet.

2. Owner transfer

TRANSFER_BUILDING_OWNERSHIP

With this procedure, you can transfer the ownership of a building from one party to another one.

Parameter	Type	Description
building_id	NUMBER	ID of the building
old_party_id	NUMBER	ID of the old owner
new_party_id	NUMBER	ID of the new owner

Exceptions are thrown when either the building ID, new or old party ID does not exist, when the old party does not possess the building, when the old party tries to transfer the building to itself or to the city or when the new party already possesses the building.

TRANSFER_PARCEL_OWNERSHIP

With this procedure, you can transfer the ownership of a parcel from one party to another one.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel
old_party_id	NUMBER	ID of the old owner
new_party_id	NUMBER	ID of the new owner

Exceptions are thrown when either the parcel ID, new or old party ID does not exist, when the old party does not possess the parcel or still has buildings on the parcel, when the old party tries to transfer the parcel to itself or to the city, when the parties still have taxes to pay or when the new party already possesses the parcel.

3. Registration of immovable property

REGISTER_PARTY

With this procedure, you can register a new party. If there is no ID given, the ID will be auto generated.

Parameter	Type	Description
p_party_id	NUMBER	ID of the party
name	VARCHAR2	Name of the party

An exception is thrown when the party ID already exists.

DELETE_PARTY

This procedure can be used to delete a party.

Parameter	Type	Description
p_party_id	NUMBER	ID of the party

Exceptions are thrown when the party ID does not exist, when the party has still taxes to pay or when the party still owns a building or a parcel.

ADD_BUILDING_OWNER

With this procedure, you can add an owner to a building. When the city has owned the building before, the ownership is completely transferred to the new owner.

Parameter	Type	Description
building_id	NUMBER	ID of the building
party_id	NUMBER	ID of the owner

Exceptions are thrown when the building ID or party ID does not exist or when the party already owns the building.

DELETE_BUILDING_OWNERSHIP

This procedure is used to delete the ownership of a building. If there is no ownership left, the city will own the building.

Parameter	Type	Description
building_id	NUMBER	ID of the building
party_id	NUMBER	ID of the owner

Exceptions are thrown when the building ID or party ID does not exist, when the party does not possess the building or when the city is the owner of the building.

ADD_PARCEL_OWNER

With this procedure, you can add an owner to a parcel. When the city has owned the parcel before, the ownership is completely transferred to the new owner.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel
party_id	NUMBER	ID of the owner

Exceptions are thrown when the parcel ID or party ID does not exist, when the party already owns the parcel or when the party has still taxes to pay.

DELETE_PARCEL_OWNERSHIP

This procedure is used to delete a parcel ownership. If there is no ownership left, the city will own the parcel.

Parameter	Type	Description
parcel_id	NUMBER	ID of the parcel

party_id	NUMBER	ID of the owner

Exceptions are thrown when the parcel ID or party ID does not exist, when the party does not possess the parcel, when the city is the owner of the parcel or when the party has still taxes to pay.

4. Property valuation and taxation

The tax calculation is a complex process. The tax is calculated for one year and all parcels. If it was already calculated, it cannot be calculated anymore for this year. In the following, the calculating process is described in detail:

The tax per parcel is based on:

- the size of the parcel
- its usage factor
- the ground values of the ground areas

The base formula for the taxation rate of a parcel is:

usage factor * size of area * ground value

But a parcel can lie on several ground areas. So, the ground value of each underlying ground area only contributes to the tax according to the shared size.

Example:

A parcel lies on the ground areas A, B, C, and D. The shared sizes between the ground areas and the parcel are:

A: 1000
B: 500
C: 250
D: 100

The ground values are:

A: 50
B: 60
C: 10
D: 80

The usage factor is 0.5.
So, the calculation is as follows:

```
usage factor * ((share size of A * ground value of A)
                + (share size of B * ground value of B)
                + (share size of C * ground value of C)
                + (share size of D * ground value of D))

0.5 * ((1000 * 50)
       + (500 * 60)
       + (250 * 10)
       + (100 * 80))
= 45 250
```

After having calculated the tax for a parcel, it is divided by the number of distinct direct and indirect owners.

Here, the corresponding procedures:

CALCULATE_TAXES

This procedure calculates the taxes for all parties and parcels.

Parameter	Type	Description
year	NUMBER	Tax year

Exceptions are thrown when the year is too small or too big or when the taxes have been already calculated for the given year.

PAY_TAXES

This procedure can be used to pay the taxes of a party.

Parameter	Type	Description
party_id	NUMBER	ID of the party
year	NUMBER	Tax year

Exceptions are thrown when the party ID does not exist, when there are no taxes to pay for the given year or when the taxes have been already paid.

5. Building mutation

CREATE_BUILDING

This procedure is used to create a new building. The ownership of the building will be registered in a title document. If there is no owner given, the city will be the owner. If there is no ID given, the ID will be auto generated.

Parameter	Type	Description
b_building_id	NUMBER	ID of the building
b_geometry	SDO_Geometry	Geometry of the building
p_party_id	NUMBER	Owner of the building

Exceptions are thrown when the building ID already exists, when there is no geometry given, when the geometry is not a polygon, when the party ID does not exist, when the building would overlap with other buildings or when the building would be not entirely on a parcel.

CHANGE_BUILDING_GEOMETRY

With this procedure, the geometry of a building can be changed.

Parameter	Type	Description
building_id	NUMBER	ID of the building
b_geometry	SDO_Geometry	Geometry of the building

Exceptions are thrown when there is no building ID or geometry given, when the geometry is not a polygon, when the building would overlap now with other buildings or when the building would be now not entirely on a parcel.

DELETE_BUILDING

With this procedure, you can delete a building. All ownerships of the building will be released.

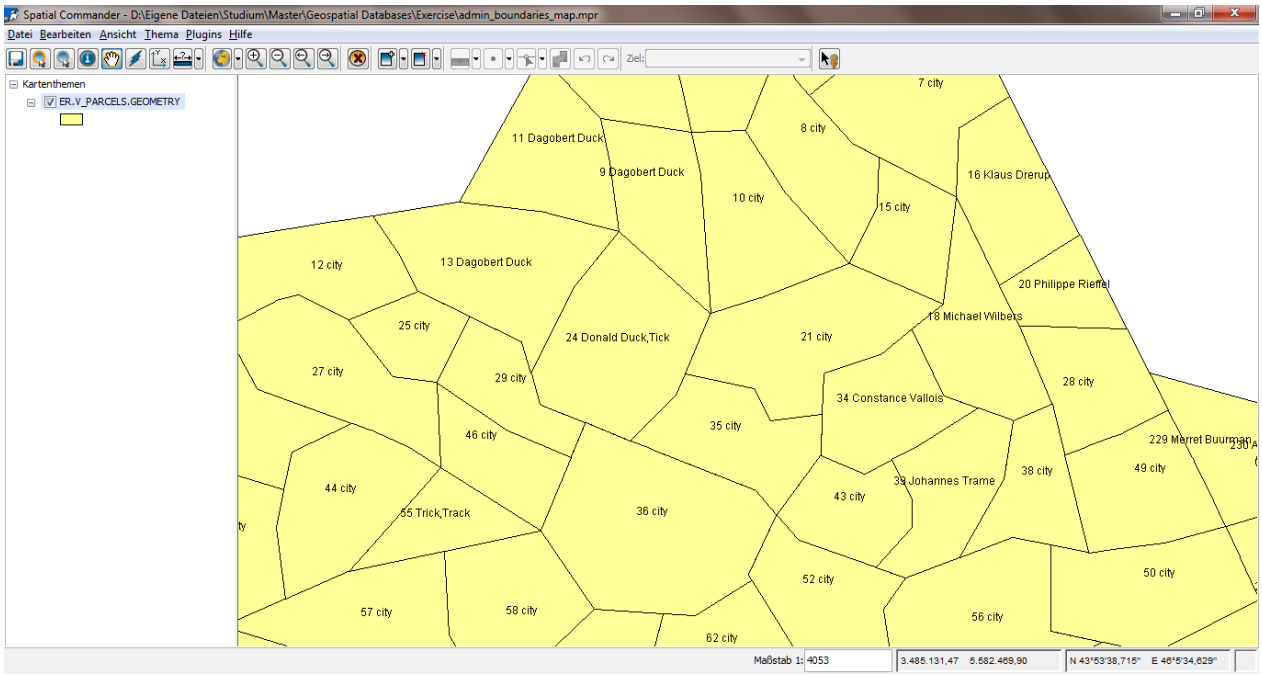
Parameter	Type	Description
building_id	NUMBER	ID of the building

Exceptions are thrown when there is no building ID given.

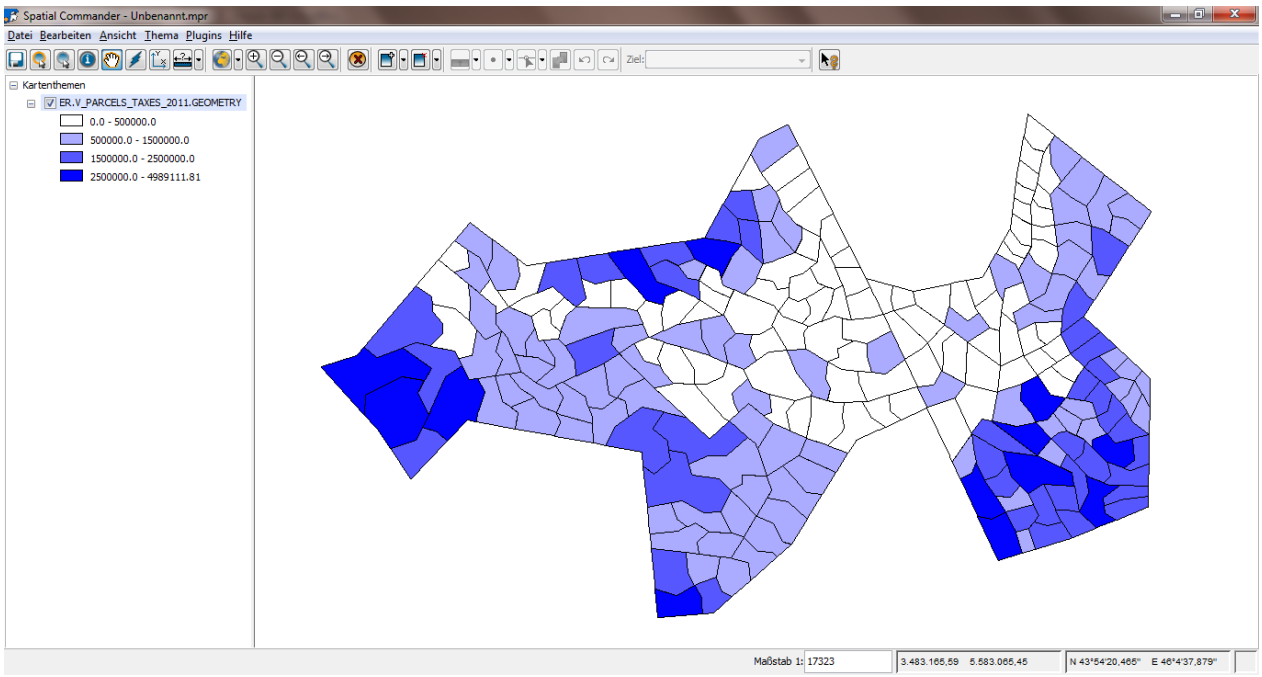
6. Using the data in a GIS-Client

Finally, we used the Spatial Commander to visualize our data. On the one hand, we could directly access tables with geometric columns; on the other hand, we had to create views for more enhanced operations. For example, we joined the parcel title document with the parcel and party table and applied a concatenate function to aggregate the names of the owners. To display those views in the Spatial Commander, it was also important to insert an entry the geometry metadata table and to commit the data to the database.

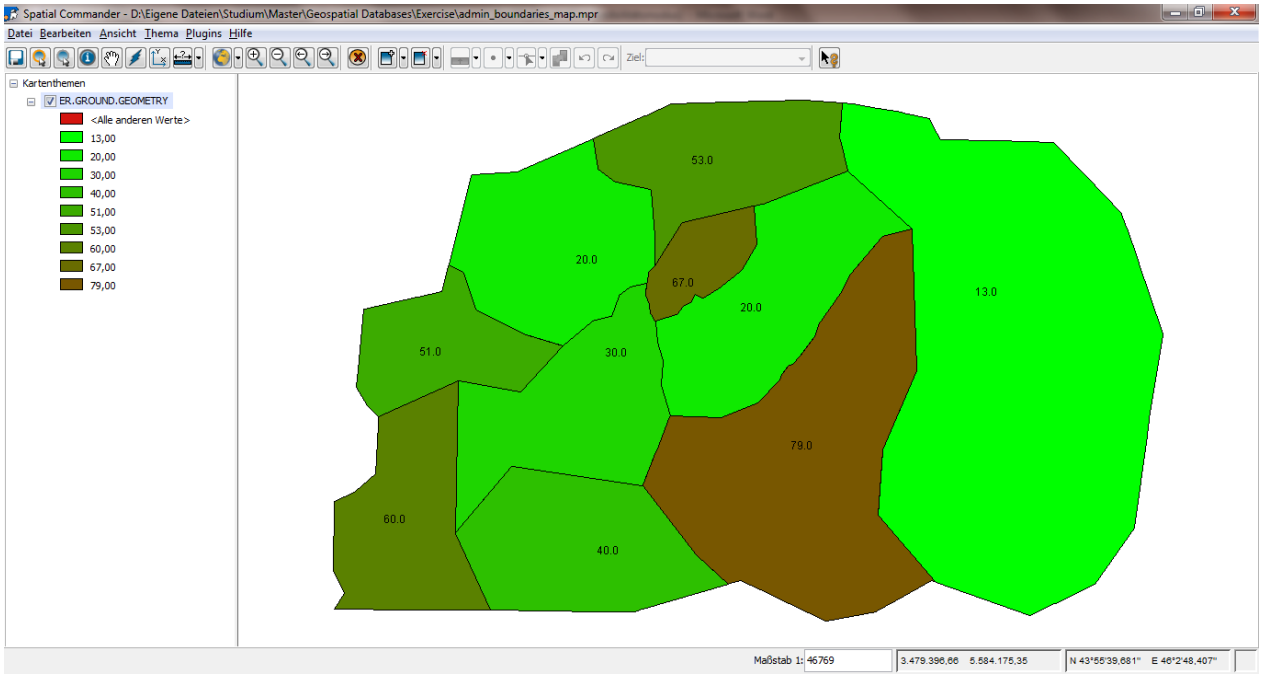
Next, you see five maps to different topics.
SQL statements defining the views can be found in the file “Schema.sql”.



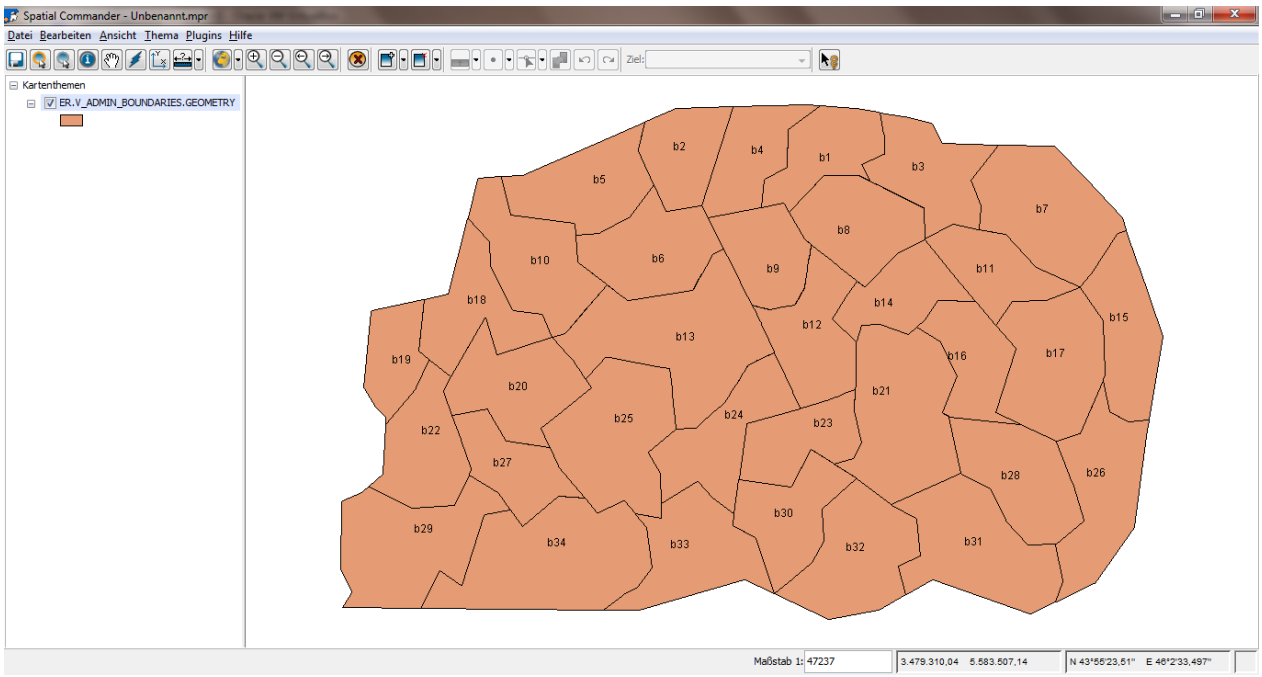
Map 1: Parcels with their IDs and owners



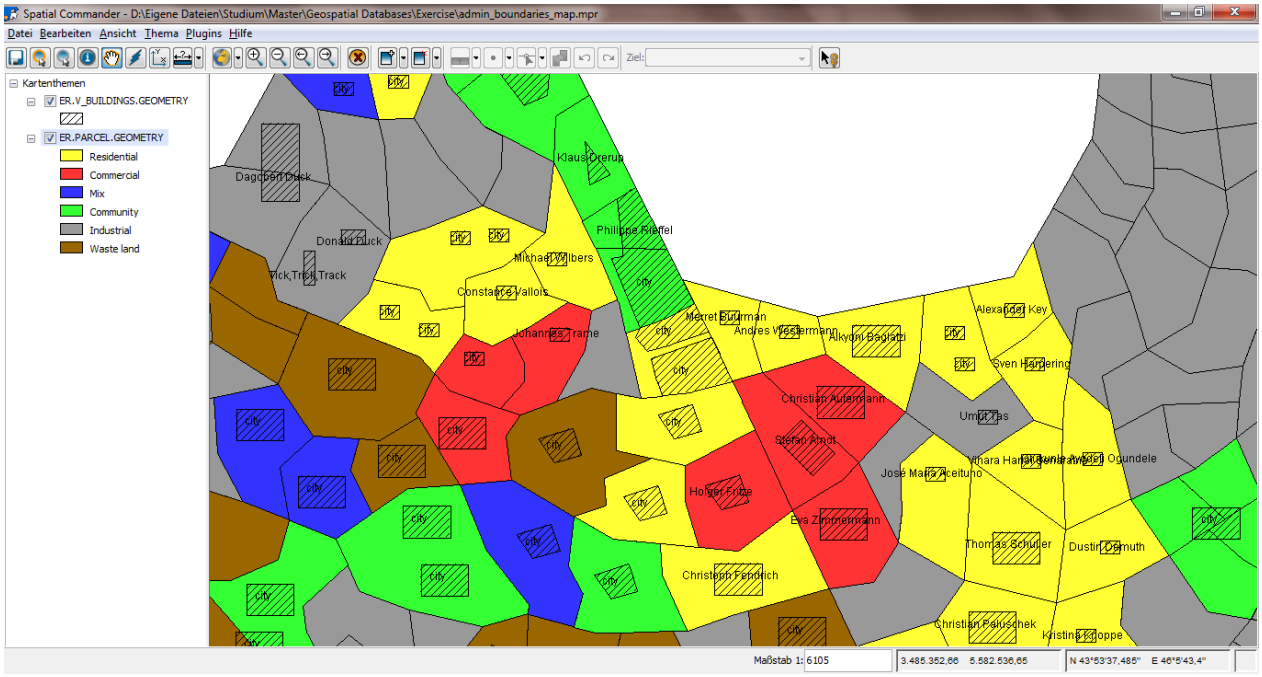
Map 2: Range value classification of the tax valuation



Map 3: Ground values



Map 4: Administrative boundaries on block level



Map 5: Parcels with their usages and buildings with their owners