Datenbankkonzept für bathymetrische Daten

Version 1.0

Grundkonzeptionen der smile consult GmbH Klassenbibliothek

Verfasser:
Christoph Lippert

Redaktionelle Bearbeitung:

Hannover, den 20.02.2017

Dipl.-Ing. Christoph Lippert

smile consult GmbH
Vahrenwalder Straße 4
30165 Hannover

Tel.: 0511 / 54 36 17 - 40
Fax: 0511 / 54 36 17 - 66

info@smileconsult.de
www.smileconsult.de

Inhaltsverzeichnis

1. Einleitung

Die Datenbankschnittstelle stellt ein Basismodul der smile consult GmbH Softwarebibliothek dar. Im vorliegenden Dokument soll die Grundkonzeption der Datenbankschnittstelle und deren technische Realisierung vorgestellt werden. Besonderes Augenmerk wird in diesem Zusammenhang auf die vollständige Abbildung des geometrischen Datenmodells der smile consult Klassenbibliothek in die Datenbankstruktur gelegt. Weiterer Schwerpunkt der konzeptionellen und technischen Realisierung ist die nahtlose Integration der Datenbankschnittstelle in die Methoden der Klassenbibliothek, was die Formulierung datenbankgestützter Methoden ermöglicht.

In Kapitel 4 wird auf Aspekte der Konfiguration des Datenbankservers und des Werkzeugs Gismo als Voraussetzung der Nutzung der Datenbankschnittstelle in den grafischen Frontends eingegangen.

Kapitel 5 erläutert grundlegende Konzeptionen für die Organisation und Pflege des Datenbestandes mit der Funktionalität der Datenbankschnittstelle von Gismo. Zielstellung dieses Kapitels ist es, einen Überblick der in Gismo verfolgten konzeptionellen Ansätze zur vollständigen Abbildung eines Bearbeitungskonzeptes für bathymetrische Daten, beginnend mit dem Daten-Import, über das Editieren der Daten und Metadaten bis hin zu möglichen Ausgabeschnittstellen zu geben. Im Besonderen soll auf die variablen Navigations- und Zugriffsmechanismen der verfolgten Bearbeitungsstrategien eingegangen werden.

2. Basiskonzept der smile software Klassenbibliothek

2.1. Softwarekomponenten und Architekturmuster

Abbildung 1 zeigt schematisch die grundlegenden Softwarekomponenten und deren Kommunikation untereinander in der Klassenbibliothek der smile consult GmbH.

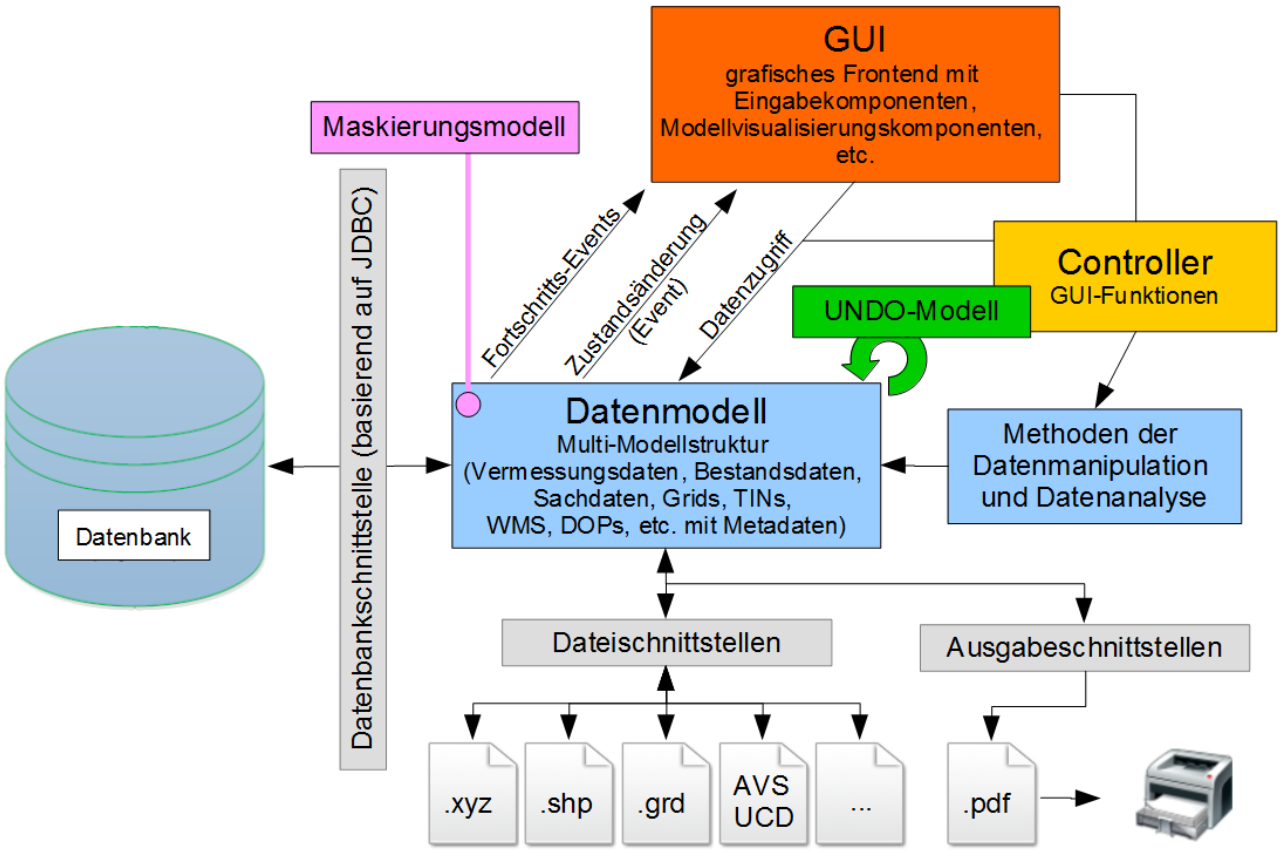


Abbildung 1: Softwarekomponenten der smile consult Klassenbibltiothek

Grundlegender Softwarebaustein ist das Datenmodell, welches die Vermessungsdaten, die digitalen Modelle, die georeferenzierten Orthophotos, etc. in einer Multi-Modellstruktur verwaltet. Basierend auf dem Datenmodell ist die umfangreiche Methodenbibliothek für die Datenmanipulation und die Datenanalyse formuliert. Beide Softwarekomponenten beinhalten ausschließlich nicht-grafische, und von den Komponenten der grafischen Benutzeroberfläche separierte Datenstrukturen und Methoden. Integraler Bestandteil des Datenmodells ist ein Maskierungsmodell, welches die Beschränkung von Methoden auf räumliche Teilbereiche oder auf durch Filtermechanismen maskierte geometrische Objekte ermöglicht. Das Datenmodell kommuniziert mit den Komponenten der grafischen Frontends Gismo/Janet/Davit (Schaubild: GUI) ausschließlich über einen Event-Mechanismus, d.h Zustandsänderungen des Modells oder auch die Fortschrittsanzeige von Bearbeitungsprozessen werden durch Auslösen von Ereignissen an die GUI-Komponenten weitergereicht. Die grafischen Komponenten sind mit den Controller-Funktionen verknüpft. Über die Controllerfunktionalität werden aus den Benutzeroberflächen der Werkzeuge heraus die Methoden der Datenmanipulation / -analyse ausgelöst. In die Controllerkomponente ist ein UNDO-Modell integriert, welches einen generellen 1-Schritt-UNDO-Mechanismus im Softwaredesign verankert.

Das grundlegende Architekturmuster der smile consult Klassenbibliothek folgt somit dem Model–View–Controller-Prinzip. Durch die strikte Trennung des Datenmodells, der Präsentationsschicht und der Steuerung wird eine weitreichende Modularisierung der Softwarekomponenten erzielt, die eine Wiederverwendbarkeit der einzelnen Softwarebausteine sowie deren Pflege und Erweiterbarkeit begünstigt.

Das zweite grundlegende Designprinzip der Klassenbibliothek ist das Schnittstellenkonzept. Das transiente Datenmodell kommuniziert ausschließlich über definierte Schnittstellen mit den Softwaremodulen zur persistenten Speicherung der Daten. Die Klassenbibliothek verfügt über eine Vielzahl von Dateischnittstellen, die einen Im- und Export der Daten in und aus unterschiedlichen Dateiformaten ermöglichen. Die Datenbankschnittstelle hat die Aufgabe, das Schreiben von Daten in die Datenbank sowie die Extraktion von Daten aus der Datenbank und die Übernahme dieser Informationen in das Datenmodell zu organisieren. Ausgabeschnittstellen dienen der Speicherung von Reports, Plotprodukten, Screenshots, etc. in Ausgabeformate wie PDF, Rasterbild- oder Vektorgrafikformate. Analog zum Model-View-Controller-Prinzip unterstützt das Schnittstellenkonzept die weitreichende Modularisierung der Softwarekomponenten mit den bereits genannten Vorteilen hinsichtlich Pflege, Wiederverwendbarkeit und Erweiterbarkeit. Darüber hinaus ist über das Schnittstellenkonzept ein flexibler Einsatz der Softwarebausteine für unterschiedliche Datenquellen und Anwendungsbereiche gewährleistet.

2.2. Datenmodell

Die Verarbeitung bathymetrischer Daten unterschiedlicher Aufnahmeverfahren in ihrer originären Struktur ist ein Grundkonzept der smile consult Softwarebibliothek. Vermessungsdaten aus unterschiedlichen Verfahren, wie Single-Beam, Multi-Beam, ALS-Daten, terrestrischen Aufnahmen, Querprofilmessungen, etc. können mit den Methoden der Softwarebibliothek prozessiert werden. Für eine verlustfreie Datenübernahme aus den Importformaten sollen darüber hinaus alle begleitenden Informationen des Messverfahrens (z.B. Übernahme der „Intensität“ bei Laserscan-Vermessungen) zumindest optional mitgeführt werden können.

Aufgabenabhängig können diese Daten im Rahmen der Modellierung entweder in ihrer originären Struktur belassen, durch Vermaschung in TINs (Triangulated Irregular Network) oder aber auch in strukturierte Modelle überführt werden. Die einzelnen Modellierungsansätze stehen gleichberechtigt nebeneinander und können nach aufgabenspezifischen Gesichtspunkten (z.B. begrenzt zur Verfügung stehende Speicherresourcen) frei gewählt werden.

Ermöglicht wird dieser Ansatz durch das generalisierte Datenmodell der smile consult Klassenbibliothek. Das Datenmodell sieht Datenstrukturen für

- unstrukturierte Daten und
- strukturierte Daten

vor.

2.2.1. Unstrukturierte Daten

Ein unstrukturierter Datensatz setzt sich im Datenmodell aus den geometrischen Objekten

- `Punkt` mit (optionalen) Attributen

- `Element` (Drei- oder Viereck) mit (optionalen) Attributen
- `Polygon` (offen oder geschlossen) mit (optionalen) Attributen

zusammen. Im Datenmodell besitzen die geometrischen Objekte `Element` und `Polygon` ausschließlich topologische Informationen, d.h. in den Datenstrukturen werden lediglich die Datenpunkte (oder auch allgemein: Modellstützstellen) des Datensatzes referenziert.

Als Attribute der geometrischen Objekte sind sowohl einfache Datentypen (Ganzzahl, Fließkommazahl, Zeichenkette, etc.) als auch komplexe Datentypen (z.B. Zeitreihen) vorgesehen.

Durch beliebige Kombination dieser attributierten, geometrischen Objekte in einem Datensatz-Container können sämtliche für die Modellierung relevanten unstrukturierten Datenarten und Modelltypen abgebildet werden. Bei einer Kombination der geometrischen Objekte `Punkt`, `Element` und `Polygon` in einem gemeinsamen Datensatz werden die Polygone als Strukturkanten interpretiert, d.h. die Kantenlagen der Dreieckselemente werden gezielt an den vorgegebenen Polygonkanten ausgerichtet.

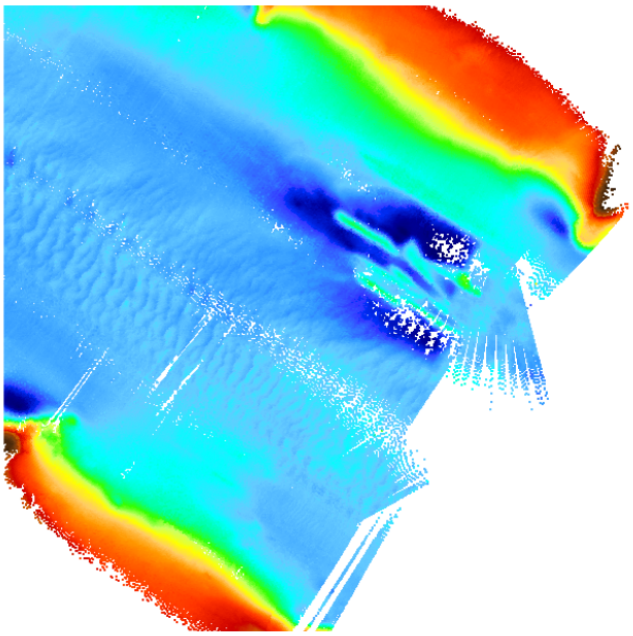
Beispielhaft sind die Abbildungen nachfolgend aufgeführter Vermessungsdaten mit den geometrischen Objekten des Datenmodells zu nennen:

- Multi-Beam-Daten: Liste mit Objekten vom Datentyp `Punkt`
- Single-Beam-Daten mit Peillinenbezug: Listen mit Objekten vom Datentyp `Punkt` und `Polygon` (Polygone bilden die einzelnen Peillinien ab)
- ALS-Daten: Liste mit Objekten vom Datentyp `Punkt`, Attribute `intensity`, `classification`, `scananglerank`, ...
- Deichlinien und einzelne Höhenkoten, trianguliertes Modell (TIN) mit den Deichlinien als integrierte Strukturlinien: Listen mit Objekten vom Datentypen `Punkt`, `Element` und `Polygon`
- etc.

Der Ansatz für unstrukturierte Datentypen besitzt darüber hinaus die grundlegende Flexibilität auch künftige Aufnahmeverfahren (z.B. Laserbathymetrie) unterstützen zu können.

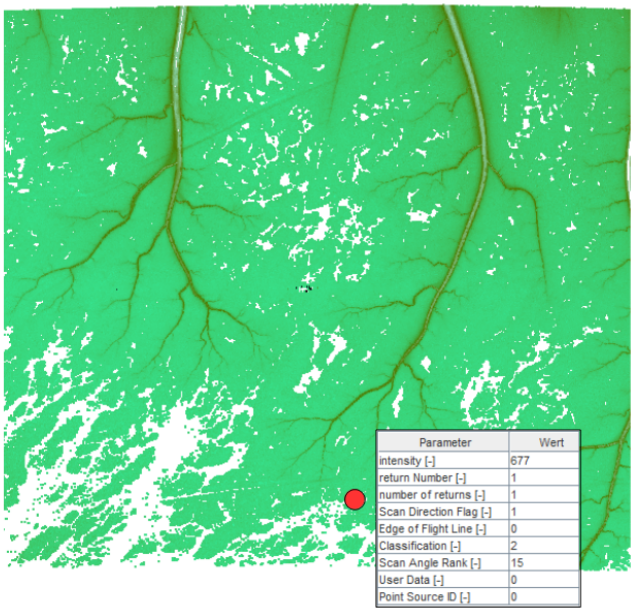
Beispiele unterschiedlichster Datenarten und Modelltypen, die mit den Methoden der Klassenbibliothek verarbeitet werden können, zeigen nachfolgende Abbildungen.

unstrukturierte Daten



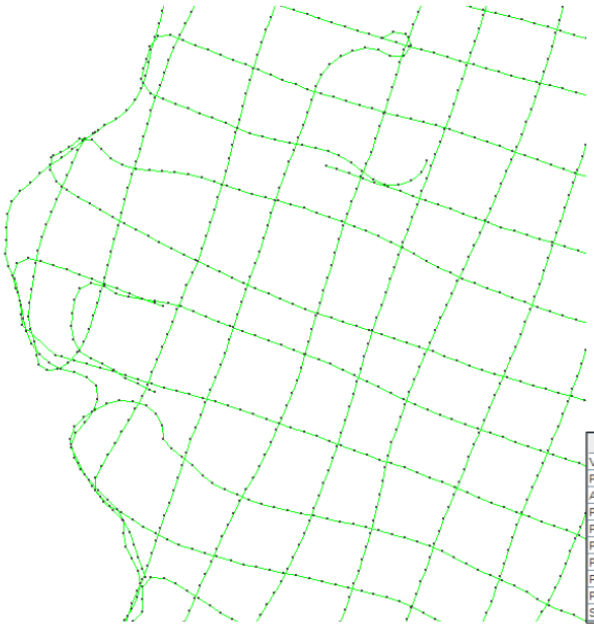
Fächerecholot-Daten

unstrukturierte Daten



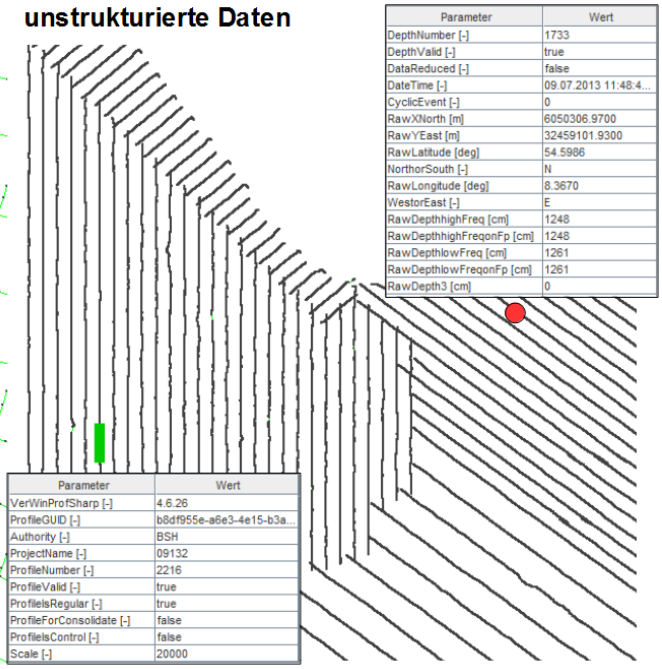
Lidar-Daten mit Attributen

unstrukturierte Daten



Kreuzprofile

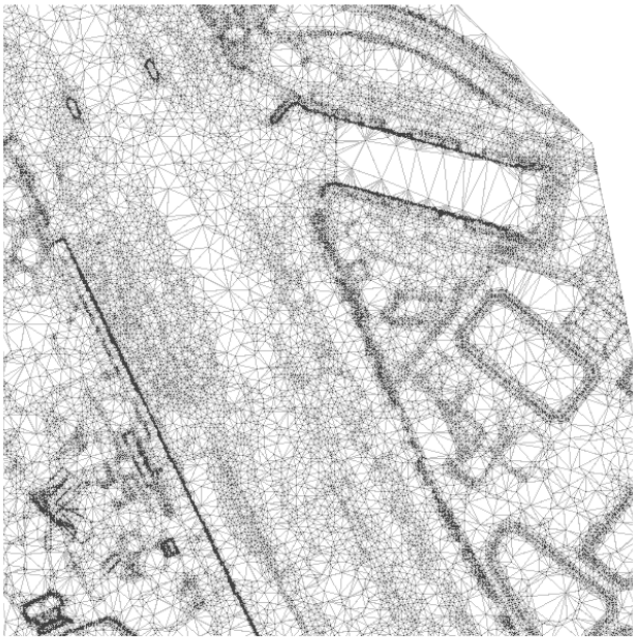
unstrukturierte Daten



Linienpeilungen mit Attributen an den Vermessungspunkten und den Peillinien

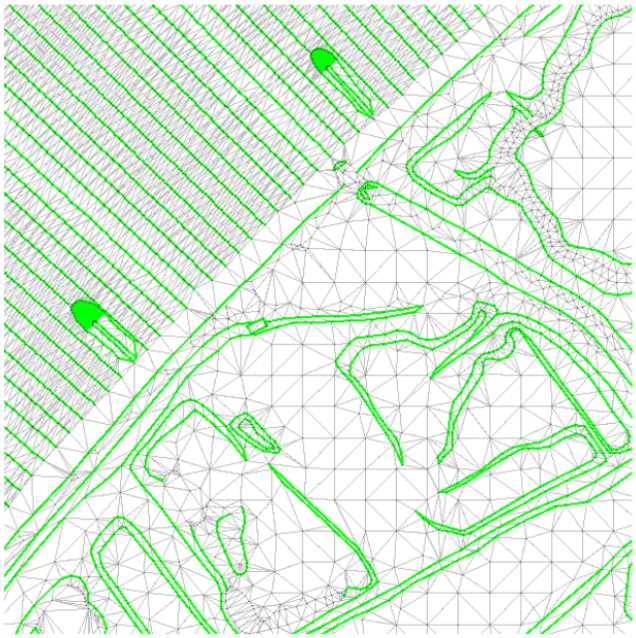
Abbildung 2: Beispiele unstrukturierter Vermessungsdaten

unstrukturierte Daten



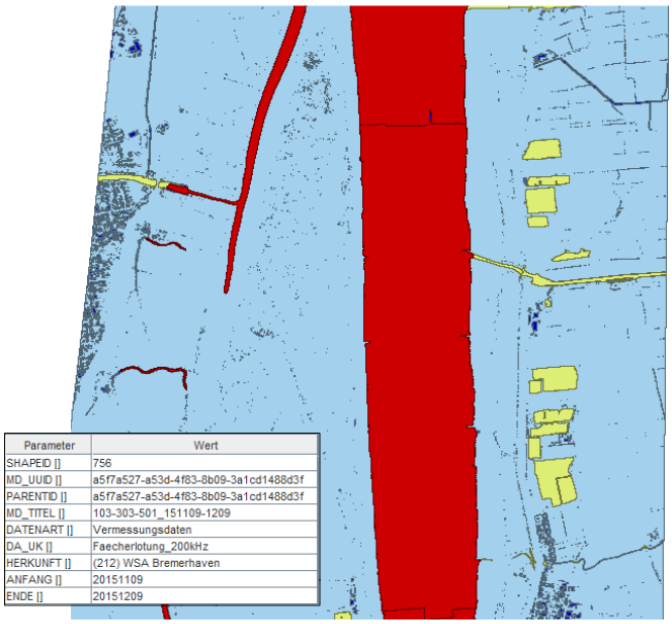
TIN

unstrukturierte Daten



TIN mit Strukturkanten

unstrukturierte Daten



Parameter	Wert
SHAPEID []	756
MD_UUID []	a5f7a527-a53d-4f83-8b09-3a1cd1488d3f
PARENTID []	a5f7a527-a53d-4f83-8b09-3a1cd1488d3f
MD_TITEL []	103-303-501_151109-1209
DATENART []	Vermessungsdaten
DA_UK []	Faecherlotung_200kHz
HERKUNFT []	(212) WSA Bremerhaven
ANFANG []	20151109
ENDE []	20151209

Datenquellenkarte (Polygonflächen mit Attributen)

Abbildung 3: Beispiele unstrukturierter Modelle und Bestandsdaten

2.2.2. Strukturierte Daten

Strukturierte Datensätze werden im Datenmodell über die räumliche Ausdehnung (Bounding-Box) und der Auflösung der Rasterdaten beschrieben. Als Daten werden die einzelnen Höhenwerte (bzw. beliebige skalare Größen) an den Rasterstützstellen abgelegt. Auch für die strukturierten Modelle ist die Definition beliebiger Attribute an den Rasterstützstellen konzeptionell vorgesehen.

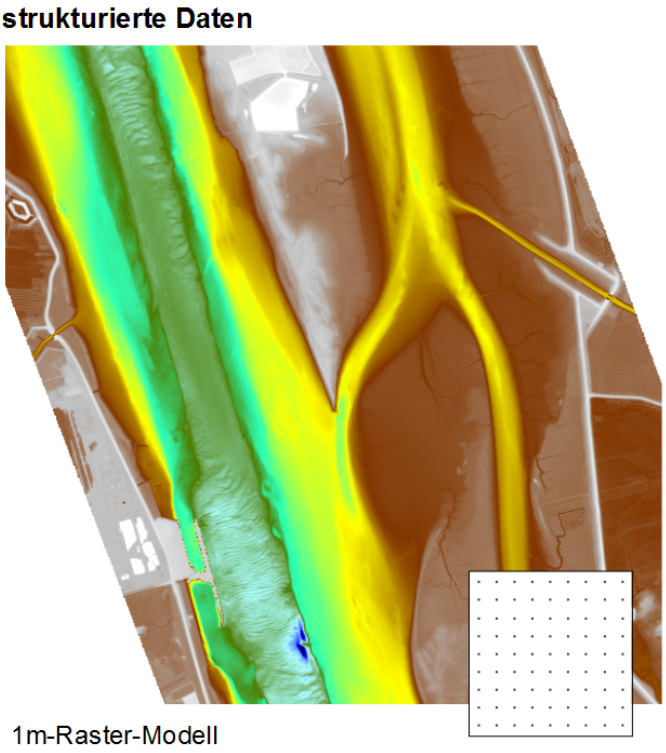


Abbildung 4: Beispiel eines Raster-Modells

2.2.3. Metadaten

Metadaten stellen einen integralen Bestandteil des Datenmodells dar. Die Metadaten enthalten weitergehende Informationen zur Herkunft der Daten, Beschreibung der Datenart, Informationen zur Bearbeitungshistorie, etc. Darüber hinaus kommt den Metadaten im Rahmen der Bearbeitungsstrategie für bathymetrische Daten mit den Methoden der smile consult Softwarebibliothek eine besondere Bedeutung zu, da in den Metadaten Information zur Steuerung von Methoden abgelegt werden („Prozeßmetadaten“, z.B. das räumliche Interpolationsverfahren für den Datensatz). Weiterhin werden die Metadaten genutzt, um technische Beschreibungen über die Struktur und die Bedeutung von Attributen geometrischer Objektklassen zu hinterlegen.

Zur Vereinheitlichung der Beschreibung von Daten durch Metadaten wurden diverse Metadatenstandards konzipiert. Neben diesen Standards existieren eine Vielzahl von proprietären Metadatenformaten,teils auch als einfache Ascii- oder Excel-Dateien. Für Vermessungsdaten aus dem Küsten- und Binnenbereich können exemplarisch folgende Standards bzw. Formate genannt werden:

- ISO19115/19139 in unterschiedlichen Profilen:
  - MDI-DE
  - DMQS (BAW)
  - NOKIS
- WSV3DTop (WSV-3D-Datenarchiv)
- S57 (Internationale Hydrographische Organisation, IHO)
- Climate and Forecast (CF) Metadata Conventions

Die unterschiedlichen Konventionen erhalten eine praktische Relevanz für die Konzeption und Realisierung des Datenmodells, da eine generelle Harmonisierung der Metadatenstandards noch nicht abgeschlossen ist und somit Metadaten abhängig vom Datenanbieter in den unterschiedlichen Metadatenformaten bereitgestellt werden.

Um den unterschiedlichen Metadatenquellen und Spezifikationen Rechnung zu tragen, wurde beim Entwurf der Datenstruktur eines Metadatums im Datenmodell der smile consult Klassenbibliothek auf eine 1:1-Abbildung eines Metadatenstandards verzichtet. Diese Designentscheidung ist auch darin begründet, dass in den genannten Spezifikationen Prozessmetadaten entweder gar nicht berücksichtigt sind oder nicht flexibel genug an die Anforderungen ihrer Verwendung in der Klassenbibliothek angepasst werden können.

Vielmehr wurde im Design der Datenstruktur für Metadaten des Datenmodells der Fokus auf eine dynamische und flexibel für eine spezifische Aufgabenstellung erweiterbare Umsetzung gelegt. Die Datenstruktur ist bewusst einfach gehalten und verfolgt einen generischen Ansatz.

Das Grundelement der Metadatenstruktur besitzt die Informationen

- Schlüsselwort des Metadatenelementes (z.B. „srid“ für den EPSG-Code)
- Elementset für eine einfache Filterung der Metadaten (z.B. Core-Element)
- Inhaltstyp zur Beschreibung des Datentyps (z.B. Integer)
- Inhalt vom Typ `Object` (z.B. 25832)

Verwaltet werden die einzelnen Metadaten-Elemente in einer dynamischen Listenstruktur, die beliebig um weitere Metainformationen ergänzt werden kann.

Die Unterstützung der oben genannten Metadatenformate erfolgt über ein Schnittstellenkonzept für Metadaten. In den einzelnen Metadatenschnittstellen sind die Abbildungsvorschriften für den Im- oder Export aus dem internen Metadatumsformat in die jeweiligen Formate hinterlegt.

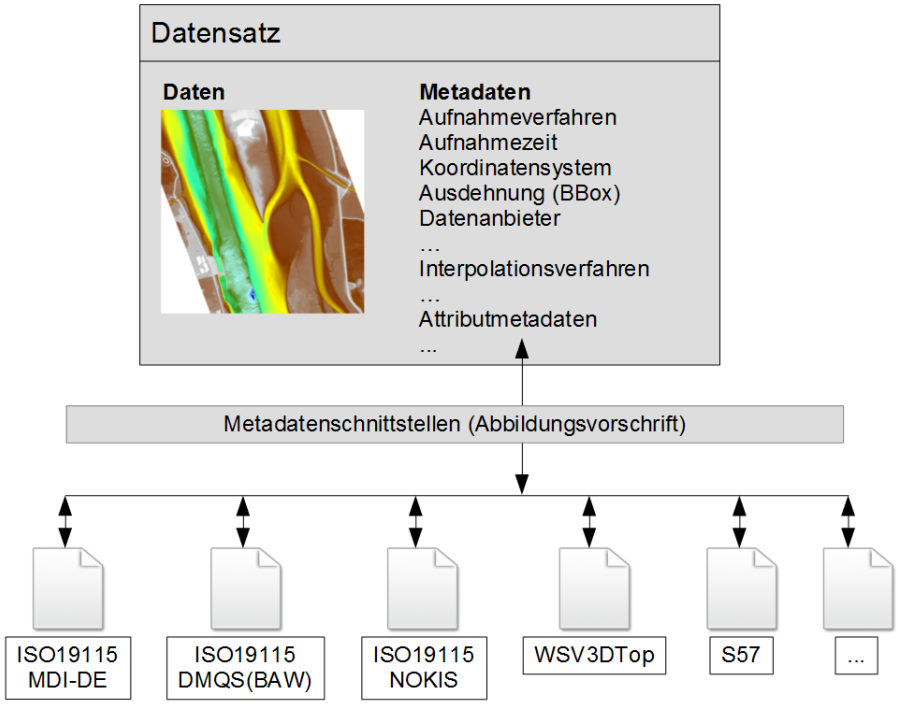


Abbildung 5: Grundschema der Metadatenschnittstelle

2.2.4. Multi-Modellstruktur

Die Multi-Modellstruktur des Datenmodells erlaubt die Verwaltung einer beliebigen Anzahl unstrukturierter und strukturierter Datensätze bzw. Modelle (im folgendem auch Layer genannt). Ergänzt wird der Multi-Modell-Ansatz durch die Integration von Layern mit speziellen Aufgabenbereichen.

- `Geoimage-Layer` für georeferenzierte Rasterbilder (Geo-Tiff, Rasterbilder mit World-File)
- `WMS-Layer` (Zugriff auf einen WebMap-Service)
- `WFS-Layer` (Zugriff auf einen WebFeature-Service)
- `Metadaten-Layer` (Zusammenstellung der Metadaten von Datensätzen einer Datenbankanfrage)

Über den layer-basierten Ansatz der Multi-Modellstruktur wird die methodische Verknüpfung von einem oder mehreren Modellen ermöglicht. Beispielhaft kann in diesem Zusammenhang die Differenzenbildung zwischen zwei Datensätzen genannt werden.

Im Hinblick auf die Datenbankschnittstelle erlangt der `Metadaten-Layer` des Multi-Modell-Ansatzes eine zentrale Bedeutung. Über diesen speziellen Layer wird die Navigation über den Datenbestand einer Datenbank, sowie die Formulierung datenbankgestützter Methoden und Analysen ermöglicht (vgl. Kapitel 3.9).

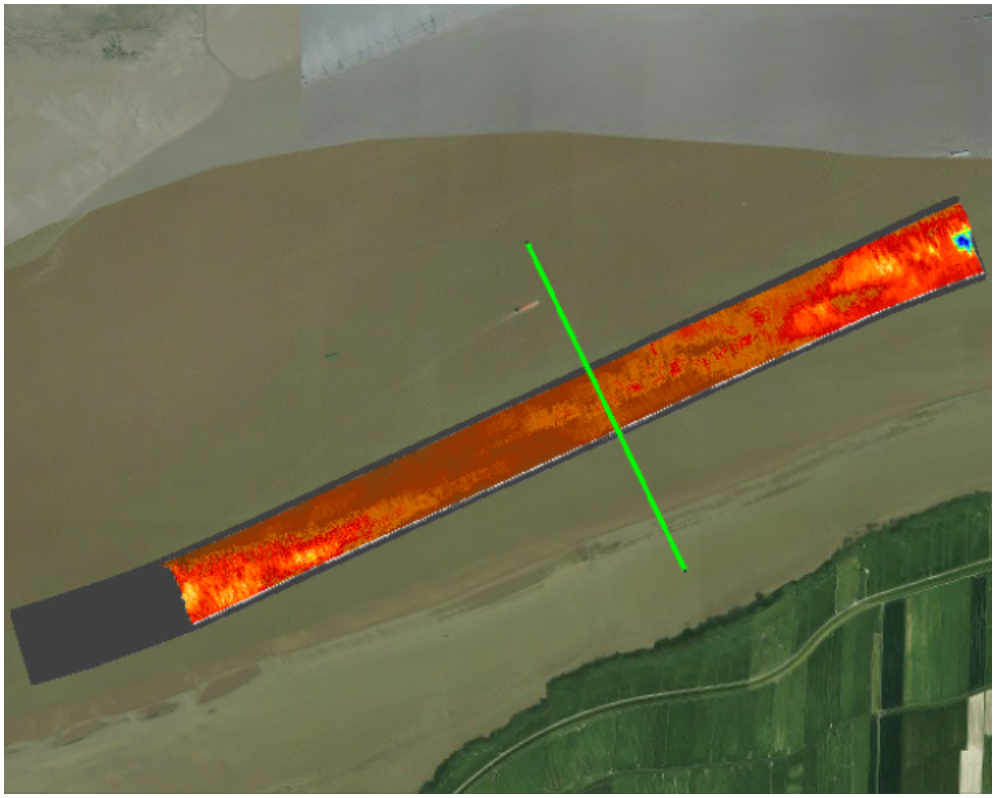


Abbildung 6: Beispiel einer Multi-Modellstruktur mit Vermessungsdaten, einer Querprofillage und einem digtalen Orthophoto in Form eines WebMap-Service

3. Konzeption und Realisierung der Datenbankstruktur und der Datenbankschnittstelle

3.1. Designkriterien

Eine zentrale Anforderung an das Design der Datenbankstruktur ist in der Kompatibilität der zu speichernden Daten mit dem Datenmodell der smile consult Klassenbibliothek gegeben. Die Modellierungsmöglichkeiten für bathymetrische Daten mit den Methoden der Klassenbibliothek sollen dementsprechend nicht durch die Datenbankstruktur limitiert werden. Der Ansatz einer Vereinheitlichung der Struktur der Daten (z.B. durch eine generelle Rasterung) für die Ablage in der Datenbank wird somit ausgeschlossen.

Als Anwendungsbereich für die Datenbankschnittstelle wird nicht allein die Archivierung von Vermessungs- und Modelldaten gesehen. Vielmehr soll die Datenbankschnittstelle direkt in die Bearbeitungs- und Analysemethoden der Klassenbibliothek integriert werden, um die Formulierung datenbankgestützter Methoden zu erlauben. Im Rahmen dieser datenbankgestützten Methoden soll die Datenbankschnittstelle den selektiven Zugriff auf einzelne Datensätze eines umfangreichen Datenbestandes, sowie auf Teildatenbereiche innerhalb eines Datensatzes organisieren. Durch Nutzung eines selektiven Datenzugriffs auf eine oder mehrere Datenbanken ist eine effiziente Ausbalancierung des Aufwandes für den Datenbankzugriff, dem Speicherverbrauch für die Haltung der importierten Daten im Hauptspeicher sowie der Ausführungsperformance für komplexe Methoden auf großen Datenbeständen möglich. Beispielhaft ist in diesem Zusammenhang die Generierung eines konsistenten Digitalen Geländemodells für den aktuellen Zeitpunkt auf Basis einer Vielzahl einzelner Vermessungsdatensätze zu nennen, wobei die Datenbankschnittstelle innerhalb der datenbankgestützten Methodik die Organisation der zeitlichen und räumlichen Überdeckung der Basisdatensätze übernimmt.

Neben den fachlichen Anforderungen an die Datenbankstruktur spielen auch technische Rahmenbedingungen für die Designentscheidungen eine Rolle. Generell werden die Werkzeuge der smile consult GmbH von einem heterogenen Anwenderkreis genutzt, zu denen Behörden der öffentlichen Verwaltung, Universitätseinrichtungen, Einzelunternehmen, etc. angehören. In der Regel werden Entscheidungen bezüglich der IT-Infrastruktur unabhängig von den speziellen Anforderungen einzelner Fachanwendungen getroffen, so dass für das Datenbankdesign die mögliche Nutzung vorhandener Datenbankmanagementsysteme als weitere Anforderung formuliert werden kann.

In diesem Zusammenhang ist als weiteres technisches Designkriterium eine produktunabhängige Konzeption der Datenbankschnittstelle zu nennen. Ermöglicht wird die Einhaltung dieser Forderung durch Verwendung der JDBC-Schnittstelle (Java Database Connectivity), welche integraler Bestandteil der API der Programmiersprache Java ist und auf die Verwendung relationaler Datenbanksysteme ausgerichtet ist. In der Konzeption von JDBC wird die Unterstützung verbreiteter Datenbanksysteme (Oracle, PostgreSQL, MySQL, etc.) durch JDBC-konforme Datenbanktreiber der einzelnen Hersteller gewährleistet.

3.2. Voruntersuchungen zu räumlichen Datentypen

Alle weit verbreiteten Datenbankmanagementsysteme unterstützen räumliche Datentypen für Anwendungen aus dem Geodaten-Bereich. Im Rahmen einer Voruntersuchung sollte ergründet werden, in wie weit die verfügbaren Ansätze mit den speziellen Designanforderungen an die

Datenbankschnittstelle vereinbar sind. Die Untersuchungen wurden seinerzeit mit den Geometrie-Datentypen für MySQL vorgenommen, sind in den Grundergebnissen jedoch auf andere Datenbanksysteme übertragbar.

3.2.1. Abbildung des Datenmodells mit räumlichen Datentypen

Eine Analyse der verfügbaren räumlichen Datentypen zeigt, dass sich die Geometrietypen in der Regel eng an die SimpleFeature Spezifikation der OGC (Open Geospatial Consortium) orientieren. Das impliziert, dass der Fokus des Anwendungsbereiches für räumliche Datentypen auf dem Einsatz in GIS-Systemen liegt. Im Hinblick auf das Datenmodell der smile consult zeigt sich insbesondere, dass für die Abbildung von unstrukturierten Gittern mit Drei- und Viereckselementen keine uneingeschränkt geeigneten Geometrie-Datentypen zur Verfügung stehen.

3.2.2. Abbildung von Triangulationen mit einem polygonalen Geometrietyp für die Dreieckselemente

Um dennoch die Möglichkeiten der räumlichen Suche über Geometrie-Datentypen ausloten zu können, wurde zu Untersuchungszwecken die Abbildungen von Dreiecksgittern mit dem räumlichen Datentyp für ein Polygon zur Speicherung der Gitterelemente verfolgt. Über diesen Ansatz ist dann die Formulierung einfacher `Select`-Statements für Punkt-In-Polygon-Tests möglich, die für die Durchführung von linearen Interpolationen auf den gefundenen Dreiecken (bzw. den Polygonen mit drei Stützstellen) im Testszenario verwendet wurden. Generell bietet dieser Ansatz jedoch den grundlegenden Nachteil, dass die Gittertopologie eines Dreiecksgitters nur sehr aufwändig aus den Einzelpolygonen (besitzen nur Stützstellen-Koordinaten) wiederhergestellt werden kann.

Als Testszenario wurde ein unstrukturiertes, trianguliertes DGM für das Jade-Weser-Ästuar mit 3560000 Knoten und 7110000 Dreieckselementen herangezogen. Die Abbildung mit dem Datentyp `Polygon` ergab folgende Datenbankgröße für das Beispiel:

Datentabellen und Indextabellen: 1941 MB

(zum Vergleich: Dateigröße im Janet-Binär-Format 197 MB, Faktor DB/Datei = 9,7)

Auf diesem Digitalen Geländemodell wurden die Stützstellen eines numerischen Modells mit 201.000 Knoten interpoliert, wobei die räumliche Suche für jede Interpolationsstützstelle jeweils über ein Select-Statement mit Contains-Logik auf den Polygon-Objekten der Datenbank durchgeführt wurde. Die Ausführungszeit betrug im Mittel

1492 sec bzw. 24 min 52 sec = 7,4 msec/Elementsuche

3.2.3. Fazit der Voruntersuchungen

Generell zeigt die Voruntersuchung, dass mit den verfügbaren räumlichen Datentypen der gängigen Datenbankprodukte, die sich in der Regel nah an der SimpleFeature Spezifikation der OGC orientieren, eine 1:1-Abbildung des Datenmodells der smile consult Klassenbibliothek nicht geeignet umgesetzt werden kann. Für die Umsetzung der Tabellenstrukturen des Datenbankmodells wird auf Standard-SQL-Datentypen zurückgegriffen. Nichts desto trotz stellen räumliche Datentypen mit den einfach und effizient zu formulierenden Datenbankanfragen für räumliche Operationen einen interessanten Ansatz dar. Hierbei sollte die weitere Entwicklung der verfügbaren Datentypen für die unterstützten Datenbanksysteme verfolgt werden.

Darüber hinaus kann aus dem Anwendungsbeispiel für das Jade-Weser-Ästuar abgeleitet werden, dass bei der Verwendung räumlicher Datentypen für Massendaten erhöhte Datenbankgrößen akzeptiert werden müssen. Die generelle Eignung von räumlichen Datentypen für umfangreiche Massendaten sollte jedoch für einen bestimmten Anwendungsfall stets kritisch geprüft werden.

Die Ausführungszeiten des Anwendungsbeispieles legen darüber hinaus nahe, dass die Elementsuche mit durchschnittlich 7,4 msec zwar ein akzeptabler Wert ist, jedoch für Anwendungsszenarien wie der DGM-W-Modellierung mit z.B. 330 Kacheln a 1 Mio. Stützstellen, die auf einem umfangreichen Bestand an Basisdaten interpoliert werden, zu enorm hohen Ausführungszeiten führen würde (hier vereinfacht approximiert: mind. 28,3 Tage). Hieraus wird gefolgert, dass die Umsetzung der Datenbankstruktur und der Zugriffsschnittstelle mit einem effizienten Ansatz erfolgen muss, so dass die resultierenden Ausführungszeiten für den praktischen Projekteinsatz geeignet sind.

3.3. Dynamisches Tabellenkonzept

Die Speicherung unterschiedlicher Datenarten und Modelltypen in einer Datenbank wird über die Integration unterschiedlicher Tabellenstrukturen zur geeigneten Abbildung der jeweiligen unstrukturierten und strukturierten Datentypen des (transienten) Datenmodells realisiert. Dieses Vorgehen impliziert, dass keine starre Tabellenstruktur der Datenbankschnittstelle zu Grunde gelegt, sondern ein dynamisches Tabellenkonzept verfolgt wird. Innerhalb des Tabellenkonzeptes werden die Metadaten in einer festen Tabellenstruktur gespeichert, die sich eng an der Umsetzung der Datenstruktur eines Metadatums im Datenmodell orientiert. In den Metadaten werden insbesondere die notwendigen Informationen über die Struktur der verwendeten Daten-Tabellen und der Speicherort der Daten abgelegt (Basis-Tabellenname).

Die Tabellenverwaltung und die Übernahme der notwendigen Struktur- und Speicherort-Informationen in die Metadaten wird innerhalb der Datenbankschnittstelle vom `TableManager` übernommen. Dieser zentrale Softwarebaustein der Datenbankschnittstelle ist dem eigentlichen Schreiben der Daten vorgeschaltet und hat folgende Aufgabenbereiche:

- Erzeugung der Metadaten-Tabellen beim Anlegen einer Datenbank (PostgreSQL, MySQL, …) bzw. beim erstmaligem Verbindungsaufbau mit dem Datenbankserver (Oracle)
- Analyse der Tabellenstruktur, ob die zu importierenden Geometrietypen des Datensatzes in den vorhandenen Tabellen gespeichert werden können
- Analyse der Spaltenstruktur von Tabellen hinsichtlich der zu speichernden Attribute der Geometrien eines Datensatzes
- Anlegen von Daten-Tabellen, wenn keine geeigneten Tabellen im Analyseprozess gefunden wurden

- Aktualisierung der Metadaten bzw. der Metadaten-Tabelle mit den Ergebnissen des Analyse-Prozesses (Basistabellenname, Tabellentyp)
- Managing konkurrierender Schreibzugriffe: der `TableManager` organisiert den Schreibzugriff derart, dass nur ein Prozess parallel in eine Tabelle schreibt

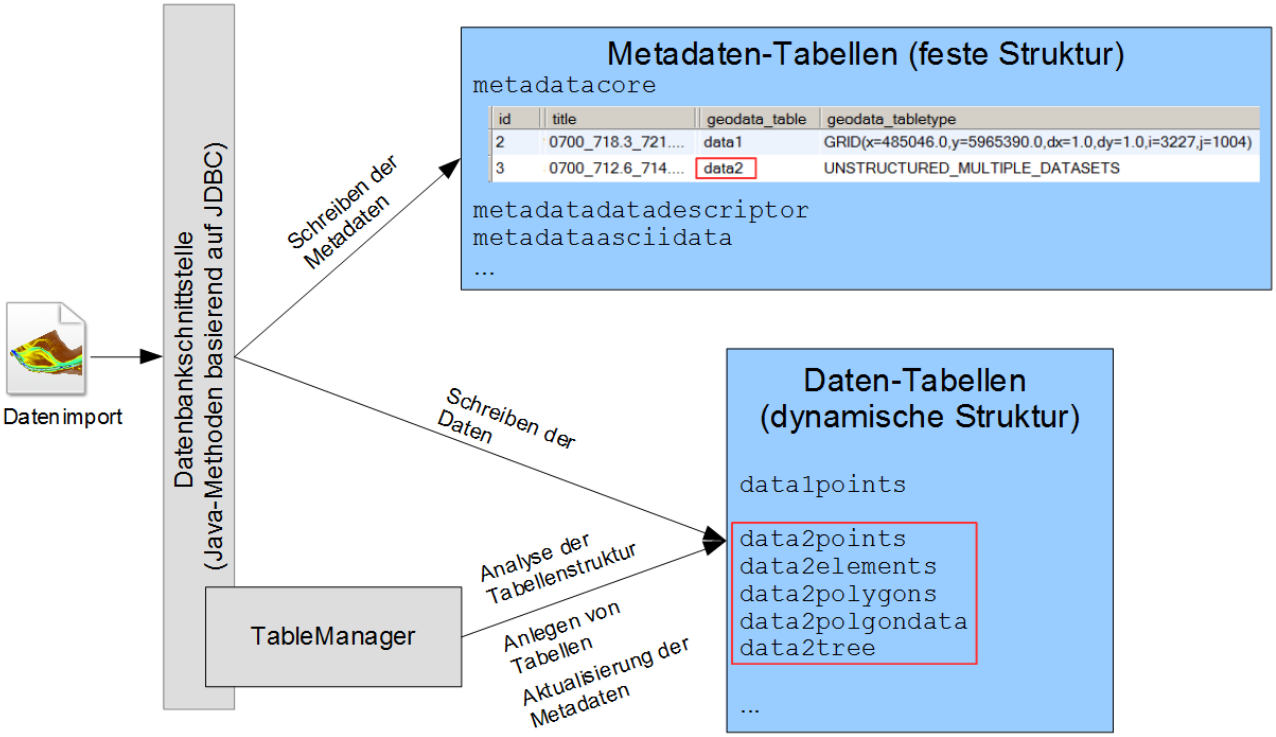


Abbildung 7: Grundschema des dynamischen Tabellenkonzeptes

Die Struktur der Daten-Tabellen ist wiederum eng an das Datenmodell der Klassenbibliothek angelehnt. Hierbei ist sowohl für die unstrukturierten als auch für die strukturierten Datenarten die Erzeugung einer Punktdatentabelle obligatorisch. In der Punktdatentabelle werden bei unstrukturierten Daten die Koordinaten der Stützstellen abgelegt, im Falle strukturierter Daten ist der 1D-Index des Datenpunktes im Raster und der Z-Wert ausreichend. Die Namenskonvention der Datenbankschnittstelle sieht vor, dass sich der Name der Punktdatentabelle aus dem Basis-Tabellennamen + dem Postfix „`points`“ zusammensetzt. In Abhängigkeit von der Verfügbarkeit weiterer Geometrien innerhalb des Datensatzes werden je nach Bedarf zusätzlich die Tabellen

- Basis-Tabellenname + „`polygons`“ zur segment-weisen Speicherung der Polygondaten
- Basis-Tabellenname + „`polygondata`“ zur Speicherung der Attribute von Polygonen
- Basis-Tabellenname + „`elements`“ zur Speicherung der Drei- und Viereckselemente
- Basis-Tabellenname + „`tree`“ zur Speicherung einer Baumstruktur für die räumliche Suche auf unstrukturierten Daten

für einen Datensatz angelegt. Die (optionalen) Attribute für die Punkt- und Elementobjekte (Dreiecke, Vierecke) werden direkt als weitere Spalten in die entsprechenden Datentabellen eingefügt. Die besondere Struktur der segmentweisen Speicherung der Polygondaten erfordert eine eigene Tabelle zur Ablage der Attribute von Polygonen.

In den Abbildungen 8 und 9 sind schematisch für zwei ausgewählte Beispiele die

Tabellenstrukturen eines unstrukturierten und eines strukturierten Datensatzes sowie auszugsweise der Inhalt der jeweiligen Tabellen dargestellt.

Tabelle `data?points` (Basis-Tabellenname + `points`)

id	md_id	nr	x	y	z	status	spi	depthnumber	depthva	datared	datetime	cyclice	rawxnorth	rawyeast	rawlatitude
1	1	0	529685.99	6075004.97	1.705	1	0	161	1	0	2012-09-19 07...	0	6075004.97	529685.99	54.82130881
2	1	1	529855.03	6075145.61	1.205	1	0	290	1	0	2012-09-19 06...	0	6075145.61	529855.03	54.82256262
3	1	2	529612.79	6074869.85	1.405	1	0	252	1	0	2012-09-19 07...	0	6074869.85	529612.79	54.8200989

Tabelle `data?elements` (Basis-Tabellenname + `elements`)

id	md_id	nr	n0	n1	n2	n3	status	spi
1	3	0	4206	3992	3983	3983	128	110
2	3	1	3934	4206	3935	3935	128	110
3	3	2	3886	4230	3920	3920	128	110

Punktnummern !

Tabelle `data?polygons` (Basis-Tabellenname + `polygons`)

id	md_id	nr	poly_edge_nr	p0	p1	spi
1	3	0	0	427	963	8
2	3	0	1	963	1463	10
3	3	0	2	1463	2016	33

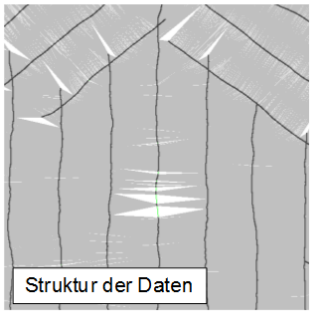


Tabelle `data?polygondata` (Basis-Tabellenname + `polygondata`)

id	poly_id	md_id	verwinprofsha	profileguid	authority	projectname	profilenumber	profilevalid	profileisregula	profileforconsc	profileiscontrol	scale
1	0	1	3.11.13	93b21fa9-4...	BSH	0312	1	1	0	1	0	20000
2	1	1	3.11.13	e5b6988e-b...	BSH	0312	2	1	0	1	0	20000
3	2	1	3.11.13	16fcdd52-c0...	BSH	0312	3	1	0	1	0	20000

Tabelle `data?tree` (Basis-Tabellenname + `tree`)

id	md_id	spi	depth	minx	miny	maxx	maxy	pointcount	edgecount	polygonedgecount
2	3	0	6	529694.5585540001	6075010.048557	529723.481446	6075038.911443001	356	0	0
3	3	2	7	529690.578996	6075027.6091665	529710.6610040001	6075044.2808335	96	0	0
4	3	3	7	529706.188919	6075037.549109999	529727.8110810001	6075055.3508899985	51	0	0

Abbildung 8: Beispiel der Tabellenstruktur für triangulierte Linienpeilungen mit Attributen an den Vermessungspunkten und den Peillinien (d.h. an den Polygonen)

Tabelle `data?points` (Basis-Tabellenname + `points`)

nr	z	status
0	-0.76	1
1	-0.75	1
2	-0.76	1

Strukturinformation des Rasters
(Referenzkoordinate, Auflösung, Anzahl Zeilen und Spalten)
steht in den Metadaten:

GRID(x=470000.5,y=5929000.5,dx=1.0,dy=1.0,i=1000,j=1000)

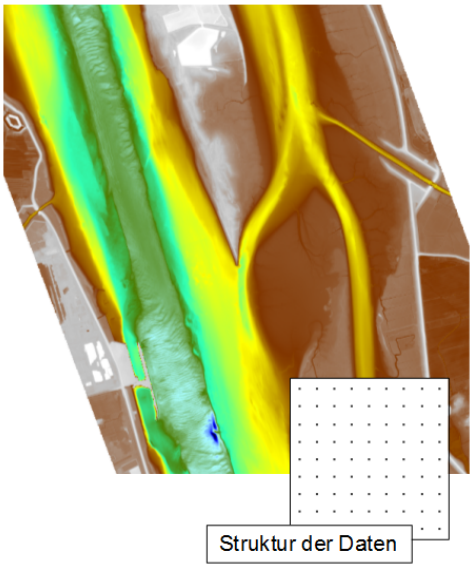
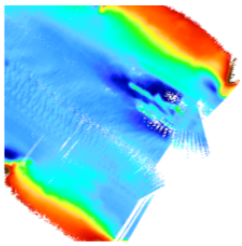
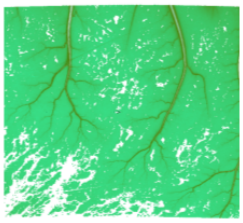


Abbildung 9: Beispiel der Tabellenstruktur für einen strukturierten Datensatz

Nachfolgende Abbildung zeigt anhand weiterer Beispiel die erforderlichen Tabellen für die Speicherung unterschiedlicher bathymetrischer Datenarten und Modelltypen.

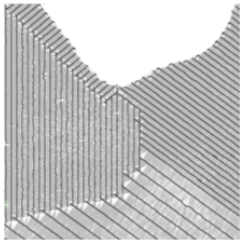


Fächerecholot-Daten
data?points
data?tree

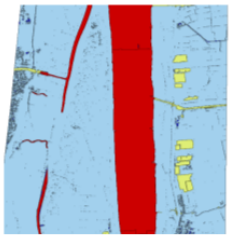


Lidar-Daten mit Attributen
data?points
data?tree

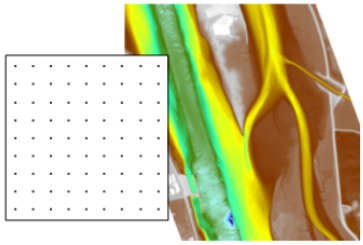
Kreuzprofile
data?points
data?polygons
data?polygondata
data?tree



Linienpeilungen mit Attributen an den Vermessungspunkten und den Peillinien
data?points
data?polygons
data?polygondata
data?elements
data?tree



Datenquellenkarte
(Polygonflächen mit Attributen)
data?points
data?polygons
data?polygondata
data?tree



1m-Raster-Modell
data?points

Abbildung 10: Tabellenstrukturen für unterschiedliche Datenarten und Modelltypen

Die dynamische Struktur der Tabellenverwaltung erlaubt darüber hinaus die Integration von Sonderlösungen für spezielle Rahmenbedingungen. Im Falle der Nutzung eines PostgreSQL Datenbankservers mit PostGIS-Extension wird die Speicherung von strukturierten Datensätzen im räumlichen Datentyp `raster` verfolgt. Hierbei wird die serverabhängige Auswahl der Speicherstrategie wiederum vom Tabellen-Manager übernommen.

Eine technische Beschreibung der Tabellenstrukturen ist in Kapitel 3.7 zu finden.

3.4. Rechte- und Rollenkonzept

Das dynamische Tabellenkonzept erfordert insbesondere, dass ein Datenbank-Account mit den entsprechenden Berechtigungen zum Anlegen und Löschen von Datenbanken, Tabellen und Indices verfügbar ist. Die Weitergabe dieser Login-Daten an den Endanwender ist nicht immer erwünscht, so dass ein dreistufiges Konfigurationsmodell für die Datenbankschnittstelle bzw. für das Werkzeug Gismo als Endanwenderschnittstelle konzipiert ist.

- **Standard-Konfiguration**: es werden keine Voreinstellungen für einen automatisierten Verbindungsaufbau beim Programmstart von Gismo hinterlegt. Die Nutzung der Datenbankschnittstelle durch den Anwender erfolgt durch aktive Anmeldung an einem oder mehreren Datenbankservern. Hierfür ist die Kenntnis der Zugangsdaten für die entsprechenden Server erforderlich.

  Für den Anwender gibt es in der Standard-Konfiguration keine Einschränkungen

hinsichtlich der Modifikation der Datenbank(en) und des Datenbankinhalts.

- **Automatisierter Verbindungsaufbau**: in die Installationsstruktur des Werkzeugs Gismo wird eine Konfigurationsdatei mit den Login-Daten zu einem oder mehreren Datenbankservern integriert (vgl. Kapitel 4.4). Beim Programmstart von Gismo werden die Serververbindungen automatisch hergestellt, wobei die Verbindungsdaten aus der verschlüsselten Konfigurationsdatei gelesen und erst innerhalb der Applikation entschlüsselt werden. Der Endanwender kann somit ohne Kenntnis von Login-Daten zum Datenbankserver die Datenbankschnittstelle nutzen.

  Für den Anwender gibt es jedoch auch in dieser Konfiguration keine Einschränkungen hinsichtlich der Modifikation der Datenbank(en) und des Datenbankinhalts.

- **Automatisierter Verbindungsaufbau und Nutzung einer applikations-eigenen Nutzerverwaltung**: für das Werkzeug Gismo wird anlog zur vorherigen Konfiguration der Mechanismus des automatisierten Verbindungsaufbaus beim Programmstart genutzt. In den Datenbankverbindungen ist eine Verbindung als Master-Verbindung konfiguriert, in der die Tabellenstruktur der internen Nutzerverwaltung erwartet wird (vgl. Kapitel 4.5). Die Ausführung des Werkzeugs Gismo erfordert nunmehr stets die Authentifizierung des Anwenders. Beim Anmeldeprozess werden die Eingabedaten mit der Nutzerverwaltung abgeglichen und der Anwender arbeitet fortan in Gismo mit der in seinen Nutzerdaten hinterlegten Rolle. Als Rollen sind (derzeit) umgesetzt:
  - Rolle *Nutzer*: ein Nutzer darf auf Daten und Metadaten einer Datenbank nur lesend zugreifen
  - Rolle *Bearbeiter:* mit der Rolle Bearbeiter können Daten/Metadaten in die Datenbank eingepflegt und editiert werden
  - Rolle *Prüfer*: ein Prüfer hat lesenden Zugriff auf die Daten und ist berechtigt den Freigabestatus in den Metadaten zu ändern (Werte „in Prüfung“, „geprüft“).
  - Rolle Sachgebietsleiter: ein Sachgebietsleiter hat lesenden Zugriff auf die Daten und ist berechtigt den Freigabestatus in den Metadaten zu ändern (Wert „freigegeben“)
  - Rolle *Administrator*: für die Rolle Administrator bestehen keine Einschränkungen

Die Zugriffssteuerung / Anwendungsbeschränkung erfolgt im Rollenkonzept nicht auf Datenbankebene, sondern wird applikationsseitig in den Komponenten der grafischen Benutzeroberflächen über einen Aktivierungs-/Deaktivierungs-Mechanismus vorgenommen. Über diesen Ansatz können auch komplexe logische Abhängigkeiten in die Verfügbarkeit bzw. die Freischaltung von Funktionen der Werkzeuge eingearbeitet werden, welche auf Datenbankebene nur schwer bis gar nicht umzusetzen wären.

Grundsätzlich kann ein Anwender auch in der zweiten und dritten Konfigurationsstufe (mit automatisiertem Verbindungsaufbau) die Anmeldung an weiteren Datenbankservern (z.B. eine lokale Datenbank) aktiv durchführen. Für diese Server hat der Nutzer dann die Berechtigungen des Punktes „Standard-Konfiguration“. Damit werden komplexe Anwendungsszenarien möglich, in denen sich ein Anwender beispielsweise aus einem zentralen Datenbestand (zugriffsbeschränkt über Nutzerverwaltung, hier hat der Nutzer nur lesenden Zugriff) Daten für ein Projekt auf den lokalen Datenbankserver kopiert, diese Daten nach projektbezogenen Gesichtspunkten reorganisiert, modelliert und über die datenbankgestützten Methoden in die Projektbearbeitung einbindet.

3.5. Konzeption der räumlichen Suche im Datenbankmodell

Innerhalb des Datenbankkonzeptes kommt der räumlichen Suche eine besondere Bedeutung zu, da der selektive Zugriff auf Teildatenbereiche innerhalb eines Datensatzes direkt im Zugriffskonzept der Datenbankschnittstelle verankert wird.

Der Konzeption der räumlichen Suche liegen folgende Grundüberlegungen zu Grunde:

- die räumliche Suche mit den Methoden des transienten Datenmodells ist durch die Verwendung von Suchbäumen (anwendungsbezogen KD-, Quad- oder R-Tree) effizient möglich
- die Datenbankzugriffe zur Extraktion von Daten aus der Datenbank sind im Vergleich zur räumlichen Suche im transienten Datenmodell „teuer“
- der Hauptspeicher der Rechner zur Haltung des Datenmodells ist stets begrenzt (wachsende Hauptspeicherkapazitäten stehen stetig wachsenden Vermessungsdatenvolumina gegenüber)

Die verfolgte Lösungsstrategie wird aus einer geeigneten Ausbalancierung der genannten Aspekte hergeleitet:

- die Bounding-Box-Information der Metadaten eines Datensatzes wird in die räumliche Suche integriert
- die Anzahl der Datenbankzugriffe wird minimiert, d.h. die Extraktion von einzelnen Geometrien pro Datenbankzugriff soll vermieden werden (vgl. auch Kapitel 3.2.2, Nutzung räumlicher Datentypen)
- die Zusammenfassung von Geometrien für einen Datenbankzugriff wird durch eine Baumstruktur organisiert, d.h. die kleinste Zugriffseinheit pro Datenbankanfrage ist ein Blatt des Suchbaumes (welches die Geometrien lokal bündelt)
- die Hauptspeicherkapazität soll optimal ausgenutzt werden, um redundante Datenbankzugriffe zu minimieren, d.h. eine Caching-Strategie für bereits importierte Blätter eines Suchbaumes wird zur Optimierung der Hauptspeichernutzung umgesetzt

Das Grundschema der räumlichen Suche auf Basis der genannten Anforderungen für einen einzelnen Datensatz über die Datenbankschnittstelle zeigt Abbildung 11.

Hinsichtlich der Performance von datenbankgestützten Such-Operationen mit dem gewählten Ansatz sind zwei wesentliche Szenarios zu unterscheiden:

- die aufeinanderfolgenden Such-Operationen haben geringe lokale Abstände (z.B. für die Interpolation eines 1m-Rasters, DGM-W-Modellierung):
  in diesem Fall wird die räumliche Suche cache-dominant und die Performance unterscheidet sich nur unwesentlich von der Performance der Interpolation im transienten Datenmodell (d.h. alle Daten für die Interpolation befinden sich im Hauptspeicher)
- die aufeinanderfolgenden Such-Operationen haben lokale Abstände, die wesentlich über die räumliche Ausdehnung der Suchbaum-Blätter hinausgehen bzw. in unterschiedlichen Datensätzen vorgenommen werden müssen:
  in diesem Szenario wird der selektive Datenbankzugriff auf die Geometrien der Suchbaum-Blätter zum dominierenden Faktor und die Such-Operationen (z.B. für die Interpolation) liegen wesentlich über denen im transienten Datenmodell

Im Umkehrschluss kann aus den zwei Szenarien gefolgert werden, dass ein einfacher

Optimierungsansatz für eine Vielzahl von Such-Operationen darin besteht, eine räumliche Vorsortierung der Such-Anfragen durchzuführen. Diese Optimierungsstrategie wird z.B. in den Interpolationsmethoden der Klassenbibliothek per default verfolgt.

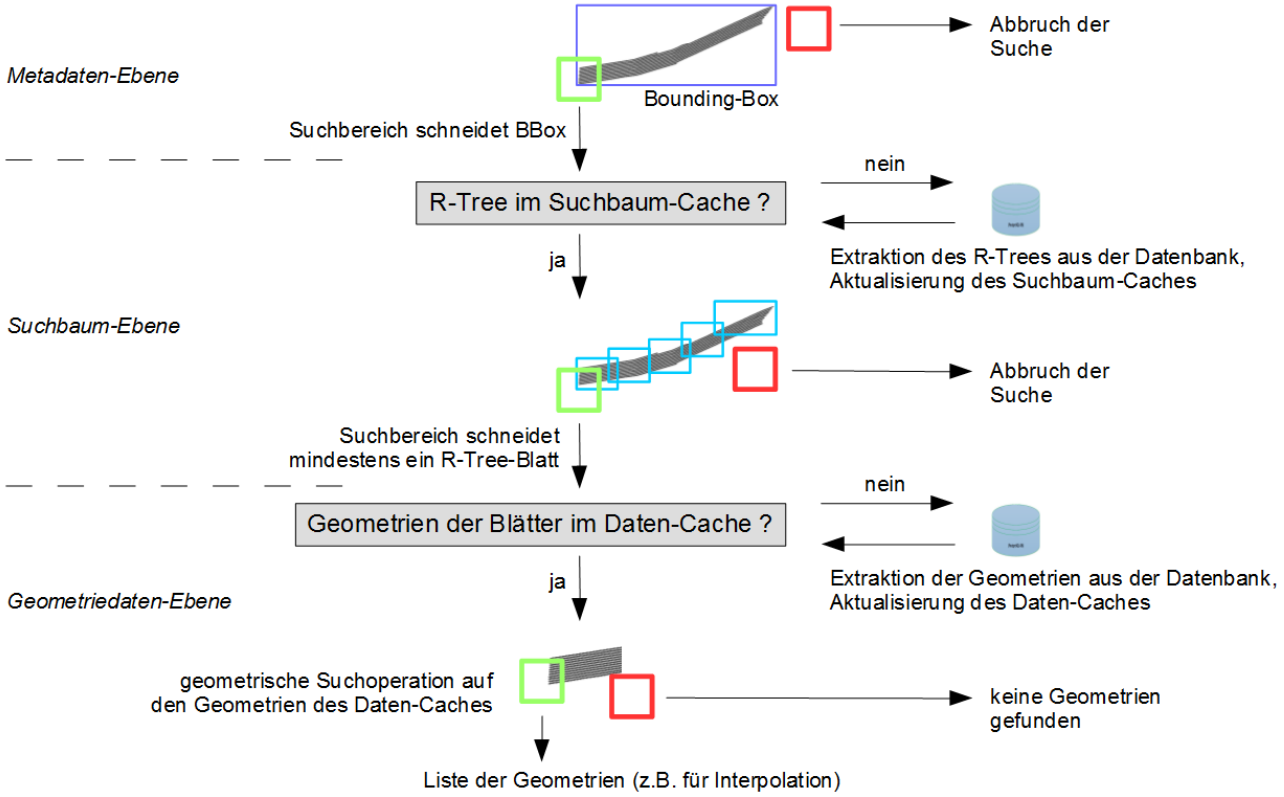
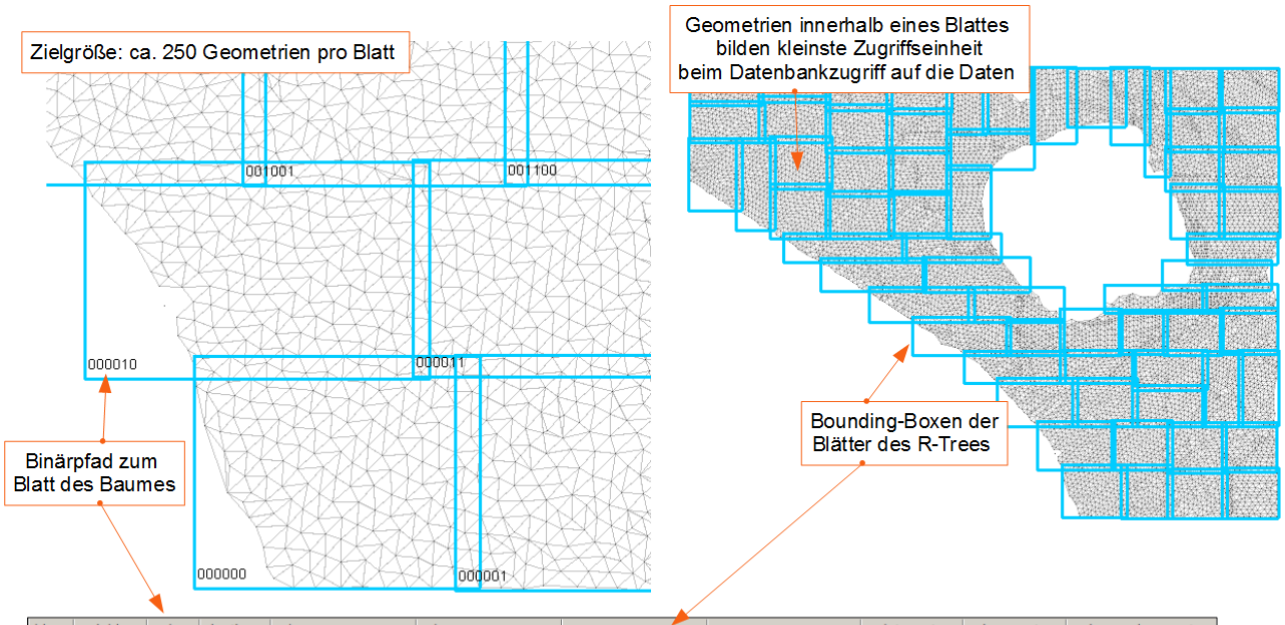


Abbildung 11: Konzept der räumlichen Suche auf einem einzelnen Datensatz

3.5.1. Räumlicher Index

Die praktische Umsetzung der Zugriffslogik wird mit einem R-Tree vorgenommen. Der R-Tree wird in einer eigenen Tabelle für jeden unstrukturierten Datensatz gespeichert und enthält die Informationen zu den Blättern des Baumes (Bounding-Box, Binärpfad, etc. pro Blatt). Die Blätter des Suchbaumes sind derart angelegt, dass die Geometrien Punkt, Element und Polygonsegment des Datensatzes mindestens einem Blatt genau zugeordnet werden können (d.h. die Geometrie fällt vollständig in die Bounding-Box des Blattes). Damit kann jeder Geometrie des Datensatzes der Binärpfad des Blattes als räumlicher Index angefügt werden. Mittels geeigneter `Select`-Statements mit dem räumlichen Index als Bedingung können somit die Geometrien eines Blattes aus dem Gesamtdatensatz gefiltert werden (Abbildung 13).

Die in Abbildung 12 angegebene Zielgröße von ca. 250 geometrischen Objekten pro Blatt des Suchbaumes ist hierbei nicht aus Erwägungen der Zugriffsperformance auf die Datenbank gewählt, sondern wurde im Hinblick auf die Ausführungsgeschwindigkeit der geometrischen Such-Operationen festgelegt. Bei Suchoperationen auf der Geometriedaten-Ebene (vgl. Abbildung 11) werden die Listen der geometrischen Objekte letztendlich linear durchlaufen, was eine Größenbeschränkung für akzeptable Ausführungszeiten erfordert.



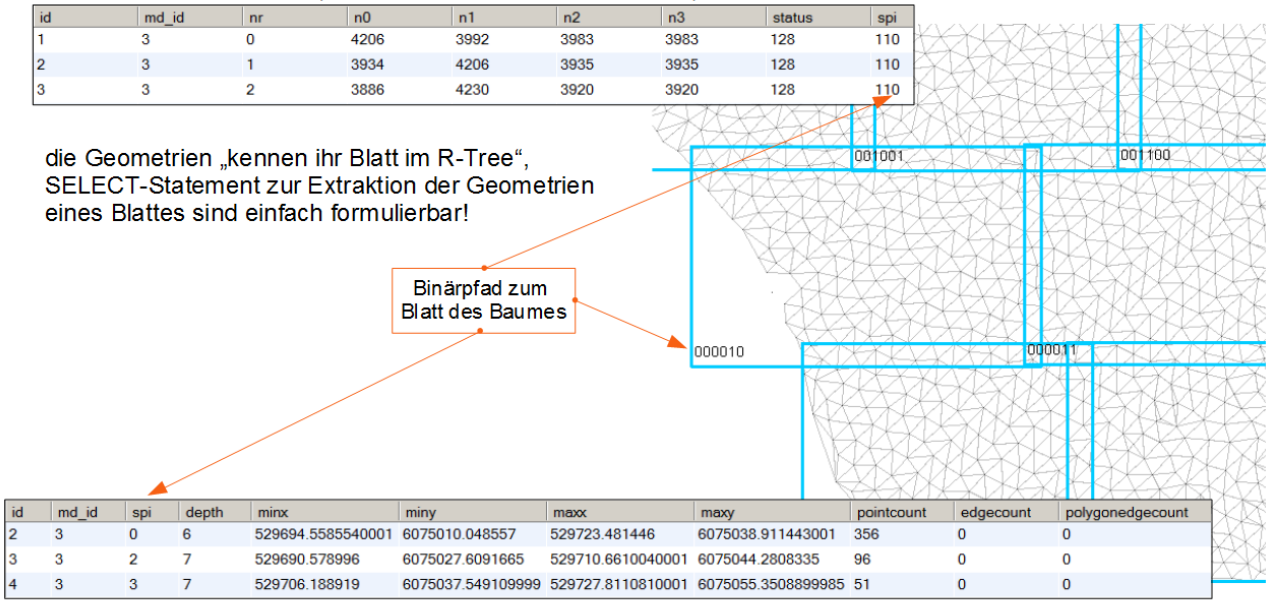
id	md_id	spi	depth	minx	miny	maxx	maxy	pointcount	edgecount	polygonedgecount
2	3	0	6	529694.5585540001	6075010.048557	529723.481446	6075038.911443001	356	0	0
3	3	2	7	529690.578996	6075027.6091665	529710.6610040001	6075044.2808335	96	0	0
4	3	3	7	529706.188919	6075037.549109999	529727.8110810001	6075055.3508899985	51	0	0

Tabelle `data?tree` (Basis-Tabellenname + `tree`)

Abbildung 12: Speicherung der Baumstruktur des R-Trees in einer eigenständigen Tabelle

Tabelle `data?elements` (Basis-Tabellenname + `elements`)

id	md_id	nr	n0	n1	n2	n3	status	spi
1	3	0	4206	3992	3983	3983	128	110
2	3	1	3934	4206	3935	3935	128	110
3	3	2	3886	4230	3920	3920	128	110



id	md_id	spi	depth	minx	miny	maxx	maxy	pointcount	edgecount	polygonedgecount
2	3	0	6	529694.5585540001	6075010.048557	529723.481446	6075038.911443001	356	0	0
3	3	2	7	529690.578996	6075027.6091665	529710.6610040001	6075044.2808335	96	0	0
4	3	3	7	529706.188919	6075037.549109999	529727.8110810001	6075055.3508899985	51	0	0

Tabelle `data?tree` (Basis-Tabellenname + `tree`)

Abbildung 13: Binärpfad zum Blatt des Suchbaumes wird jeder Geometrie zugeordnet

3.6. Konzeption der zeitlichen Suche im Datenbankmodell

Für die zeitliche Suche von Datensätzen im Datenbestand der Datenbank(en) wird von der Modellvorstellung ausgegangen, dass die Daten eines Datensatzes quasi-synoptisch aufgenommen bzw. als Modell für einen Zeitpunkt / Zeitraum erzeugt wurden. Die zeitliche Suche

erfolgt somit stets auf Metadaten-Ebene und wertet die im Metadatum hinterlegte zeitliche Bounding-Box aus.

Die Implementierung der zeitlichen Suche wird derzeit linear auf der zeitlich sortierten Liste der Datensätze des Datenbestandes durchgeführt. Für umfangreiche Datenbestände zeigen Anwendungsszenarios, dass die lineare Suche zu Performanceeinbußen führt, so das auch hier erste Ansätze mit Suchbäumen für eine räumliche und zeitliche Strukturierung der Metadaten verfolgt werden.

3.7. Technische Beschreibung der Tabellenstruktur

3.7.1. Datentyp-Mapping für unterschiedliche DBMS

Die Umsetzung des dynamischen Tabellenkonzeptes inklusive der Realisierung der Datenbankschnittstelle verwendet für alle unterstützten Datenbankmanagementsysteme eine identische Logik und Implementierung. Die Unterstützung der unterschiedlichen DBMS wird durch ein internes Datentyp-Mapping zur Berücksichtigung produktspezifischer Eigenschaften gewährleistet. Die Tabelle 14 zeigt die Abbildungsvorschrift der wesentlichen Datentypen. Für die Datenbanksysteme sind die in den SQL-Statements beim Anlegen der Tabellen verwendeten Typen angegeben.

Java-Typ	MySQL	PostgreSQL	Oracle
long	BIGINT	BIGINT	NUMBER(19,0)
int	INTEGER	INTEGER	NUMBER(10,0)
short	SMALLINT	SMALLINT	NUMBER(5,0)
byte[]	LONGBLOB	BYTEA	BLOB
boolean	CHAR(1)	CHAR(1)	CHAR(1)
double	DOUBLE	DOUBLE	BINARY_DOUBLE
float	FLOAT	REAL	BINARY_FLOAT
char	CHAR(1)	CHAR(1)	CHAR(1)
char[]	VARCHAR	VARCHAR	VARCHAR2
String	LONGTEXT	TEXT	CLOB
Calendar	DATETIME	TIMESTAMP	TIMESTAMP

Tabelle 14: Abbildung der Java-Datentypen des Datenmodells für unterschiedliche DBMS

3.7.2. Metadaten-Tabellen

Die Metadaten werden in einer festen Tabellenstruktur abgelegt, die folgende Tabellen umfasst:

- `metadatacore`
- `metadatadatadescriptor`
- `metadataasciidata`
- `metadatadatetimedata`

- `metadatadoubledata`
- `metadataintdata`
- `metadatabinarydata`

Diese Tabellen werden vom Tabellen-Manager der Datenbankschnittstelle automatisch angelegt, wobei für die `Create Table` – Statements das in Kapitel 3.7.1 dargestellte Typ-Mapping verwendet wird. Die Struktur der Tabellen orientiert sich eng an der Umsetzung der Metadaten-Datenstruktur des Datenmodells, so dass ein einfacher, informations-verlustfreier Datenaustausch zwischen dem Datenmodell und der Datenbank ermöglicht wird.

Produktspezifische Besonderheiten

Für MySQL wird explizit die `MyISAM` – Engine verwendet, wobei die Nutzung dieser Engine im `Create Table` – Statement eingearbeitet ist.

Tabelle `metadatacore`

Die Core-Metadatentabelle enthält alle Basis-Informationen, die als Minimalsatz an Metainformationen innerhalb der Methoden, der GUI-Komponenten, etc. der Klassenbibliothek benötigt wird. Der Inhalt der Tabelle ist somit weniger nach fachlichen Gesichtspunkten gestaltet, sondern enthält insbesondere auch technische Informationen für den internen Datenfluss. Die Tabelle beinhaltet

- Informationen zur Identifikation des Datensatzes (Titel, UUID)
- Zeitliche und Räumliche Ausdehnung
- Koordinatensystem-Information (EPSG-Code)
- Datensatzgrößen (Datenpunkte, Elemente, …)
- Daten zur grafischen Präsentation der räumlichen Ausdehnung (Hüllpolygone)
- Interpolationsmethodik und Aktivitätsflag
- Struktur und Speicherort der Daten (Layertyp, Tabellentyp, Basistabellenname)

Die Spalten- und Indexstruktur der Tabelle ist in Abbildung 19 am Beispiel der Verwendung mit MySQL dargestellt.

Besondere Bedeutung erlangen im Rahmen des dynamischen Tabellenkonzeptes die Felder `geodata_table` und `geodata_tabletype`, da hier der Speicherort für die Daten als Basis-Tabellenname (z.B. `data0`) und die Information über die Tabellenstruktur der Datentabelle hinterlegt sind. In der derzeitigen Umsetzung sind folgende Tabellentypen integriert:

- `UNSTRUCTURED_MULTIPLE_DATASETS`: Tabellenstruktur für mehr als einen unstrukturierten Datensatz
- `UNSTRUCTURED`: Tabellenstruktur für einen unstrukturierten Datensatz. Aus Performancegründen im Hinblick einer direkten Interpolation auf einer Datenbank wurde dieser Tabellentyp als Sonderfall für große Datensätze aufgenommen
- `GRID(x=?,y=?,dx=?,dy=?,i=?,j=?)`: Tabellenstruktur für einen strukturierten Datensatz mit Angabe der Referenzkoordinate, den Rasterweiten und Zeilen-, Spaltenangaben
- `POSTGISRASTERGRID(x=?,y=?,dx=?,dy=?,i=?,j=?)`: Tabellenstruktur für einen

strukturierten Datensatz mit Angabe der Referenzkoordinate, den Rasterweiten und Zeilen-, Spaltenangaben bei Verwendung von PostgreSQL/PostGIS

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `metadatacore` (
  `id` int(11) NOT NULL,
  `metadata_id` varchar(80) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
  `temporalextent_begin` datetime NOT NULL,
  `temporalextent_end` datetime NOT NULL,
  `srid` int(11) NOT NULL,
  `minx` double NOT NULL,
  `maxx` double NOT NULL,
  `miny` double NOT NULL,
  `maxy` double NOT NULL,
  `minz` double NOT NULL,
  `maxz` double NOT NULL,
  `pointcount` int(11) NOT NULL,
  `polygoncount` int(11) NOT NULL,
  `stored_edgecount` int(11) NOT NULL,
  `elementcount` int(11) NOT NULL,
  `hullpolygoncount` int(11) DEFAULT NULL,
  `hullpolygons` longblob,
  `active` char(1) DEFAULT NULL,
  `interpolation_function` varchar(255) DEFAULT NULL,
  `layertype` int(11) NOT NULL,
  `geodata_server` varchar(255) DEFAULT NULL,
  `geodata_database` varchar(255) DEFAULT NULL,
  `geodata_table` varchar(255) DEFAULT NULL,
  `geodata_tabletype` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
                                        KEY                    `mdcoreidx`
(`temporalextent_begin`,`temporalextent_end`,`minx`,`miny`,`maxx`,`maxy`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle `metadatadatadescriptor`

Die Metadatentabelle `metadatadatadescriptor` enthält beschreibende Informationen über die Attributstruktur eines Datensatzes. Als Informationen sind abgelegt:

- ID des zugehörigen Datensatzes
- Attributindex (der Datenstruktur des Datenmodells)

- zugeordnete Geometrie (`Punkt`, `Polygon`, `Element`)
- Informationen zur Bedeutung (Beschreibung, Kurzbeschreibung, Einheit, etc.)
- Speicherort (Tabellenname)

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `metadatadatadescriptor` (
  `id` int(11) NOT NULL,
  `md_id` int(11) NOT NULL,
  `content_type` varchar(80) DEFAULT NULL,
  `table_name` varchar(80) DEFAULT NULL,
  `table_field` varchar(80) DEFAULT NULL,
  `geometry` varchar(10) DEFAULT NULL,
  `description` varchar(80) DEFAULT NULL,
  `short_description` varchar(40) DEFAULT NULL,
  `unit` varchar(40) DEFAULT NULL,
  `unit_si` varchar(40) DEFAULT NULL,
  `param_values` longtext,
  PRIMARY KEY (`id`),
  KEY `mddescidx` (`md_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle `metadataasciidata`

Die Metadatentabelle `metadataasciidata` enthält die variablen Metainformationen der Metadaten-Datenstruktur, die dem Java-Typ `String` im Datenmodell zugeordnet werden können. Eine Zeile der Tabelle beschreibt ein einzelnes Metadatenelement, welches mit einem internen Schlüsselwort versehen ist. Die internen Schlüsselwörter werden für die Abbildungsvorschriften in unterschiedliche Metadatenformate, für die Verwendung im Metadaten-Editor, etc. genutzt. Im Einzelnen werden folgende Informationen abgelegt:

- ID des zugehörigen Datensatzes
- Elementset-Information für die Umsetzung eines Extraktionslevels (*Core, Brief, Summary, Full*) als Filtermechanismus der Metadaten-Elemente
- Inhaltstyp in Anlehnung an den Mime-Type (z.B. „text/plain“)
- internes Schlüsselwort
- Inhalt einer in der Länge unlimitierten Zeichenkette

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `metadataasciidata` (
  `id` int(11) NOT NULL,
  `md_id` int(11) NOT NULL,
```

```
  `elementset` smallint(6) NOT NULL,
  `content_type` varchar(80) NOT NULL,
  `keyword` varchar(80) NOT NULL,
  `text` longtext,
  PRIMARY KEY (`id`),
  KEY `mdascidx` (`md_id`,`elementset`,`keyword`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle `metadataintdata`

Tabelle `metadatadoubledata`

Tabelle `metadatadatetimedata`

Tabelle `metadatabinarydata`

Die Metadatentabellen `metadataintdata`, `metadatadoubledata`, `metadatadatetimedata` und `metadatabinarydata` enthalten analog zur Tabelle `metadatasciidata` die variablen Metainformationen des Datenmodells für die Java-Typen int, double, Calendar, sowie für beliebige binär codierte Daten. Die Binärdaten-Tabelle wird hierbei vorrangig zur Speicherung komplexer Datentypen (z.B. Polygon-Sets zur Definition von Gültigkeitsbereichen) verwendet. Der Aufbau der Tabelle entspricht der Metadatentabelle für die Textdaten.

SQL-Statement der Binär-Datentabelle beispielhaft für MySQL:

```
CREATE TABLE `metadatabinarydata` (
  `id` int(11) NOT NULL,
  `md_id` int(11) NOT NULL,
  `elementset` smallint(6) NOT NULL,
  `content_type` varchar(80) NOT NULL,
  `keyword` varchar(80) NOT NULL,
  `data` longblob,
  PRIMARY KEY (`id`),
  KEY `mdbinidx` (`md_id`,`elementset`,`keyword`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

3.7.3. Daten-Tabellen für Tabellentyp `UNSTRUCTURED_MULTIPLE_DATASETS`

In Abhängigkeit der Struktur der Daten wird die Ablage in unterschiedlichen Tabellen zur Erfassung der Punkt, Polygon und Element-Daten organisiert. Die Tabellenstruktur ist dafür ausgelegt eine beliebige Anzahl an Datensätzen (mit gleicher Struktur) zu verwalten.

Tabelle Basistabellenname + `points`

Die Tabelle enthält die Punkt-Informationen:

- ID des zugehörigen Datensatzes
- Punkt-Nummer
- Koordinaten
- Status-Information (interne Flags. Randpunkt, Polygonpunkt, etc.)
- räumlicher Index (Binärpfad zum Blatt des Suchbaumes)
- optionale Attributspalten

SQL-Statement beispielhaft für MySQL (ohne Attributspalten):

```
CREATE TABLE `data0points` (
  `id` int(11) NOT NULL DEFAULT '0',
  `md_id` int(11) DEFAULT NULL,
  `nr` int(11) DEFAULT NULL,
  `x` double DEFAULT NULL,
  `y` double DEFAULT NULL,
  `z` float DEFAULT NULL,
  `status` tinyint(4) DEFAULT '1',
  `spi` smallint(6) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `spidx` (`spi`),
  KEY `mdidx` (`md_id`),
  KEY `nridx` (`nr`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle Basistabellenname + `polygons`

Die Tabelle enthält die segment-bezogenen Polygondaten:

- ID des zugehörigen Datensatzes
- Polygon-Nummer
- Polygonsegment-Nummer
- Punkt-Nummern der Segmentstützstellen
- räumlicher Index

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `data0polygons` (
  `id` int(11) NOT NULL DEFAULT '0',
  `md_id` int(11) DEFAULT NULL,
  `nr` int(11) DEFAULT NULL,
  `poly_edge_nr` int(11) DEFAULT NULL,
  `p0` int(11) DEFAULT NULL,
  `p1` int(11) DEFAULT NULL,
  `spi` smallint(6) DEFAULT NULL,
  PRIMARY KEY (`id`),
```

```
  KEY `nridx` (`nr`),
  KEY `spidx` (`spi`),
  KEY `mdidx` (`md_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle Basistabellenname + `polygondata`

Die Tabelle enthält Attributdaten für Polygone:

- ID des zugehörigen Datensatzes
- Polygon-Nummer
- Text- und Typ-Attribut (Default-Attribute für Polygone)

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `data0polygondata` (
  `id` int(11) NOT NULL DEFAULT '0',
  `poly_id` int(11) DEFAULT NULL,
  `md_id` int(11) DEFAULT NULL,
  `text` varchar(130) DEFAULT NULL,
  `type` varchar(130) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle Basistabellenname + `elements`

Die Tabelle enthält die Element-Informationen:

- ID des zugehörigen Datensatzes
- Element-Nummer
- Knotennummern (bei Dreiecken ist die vierte Nummer identisch mit der dritten)
- Status-Information (interne Flags. Randpunkt, Polygonpunkt, etc.)
- räumlicher Index (Binärpfad zum Blatt des Suchbaumes)
- optionale Attributspalten

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `data0elements` (
  `id` int(11) NOT NULL DEFAULT '0',
  `md_id` int(11) DEFAULT NULL,
  `nr` int(11) DEFAULT NULL,
  `n0` int(11) DEFAULT NULL,
  `n1` int(11) DEFAULT NULL,
  `n2` int(11) DEFAULT NULL,
  `n3` int(11) DEFAULT NULL,
  `status` smallint(6) DEFAULT '1',
  `spi` smallint(6) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `spidx` (`spi`),
```

```
  KEY `mdidx` (`md_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabelle Basistabellenname + `tree`

Die Tabelle enthält die Suchbaum-Informationen:

- ID des zugehörigen Datensatzes
- räumlicher Index (Binärpfad zum Blatt des Suchbaumes)
- maximale Tiefe des Baumes
- Bounding-Box des Blattes
- Anzahl der Geometrien, die das Blatt umfasst

SQL-Statement beispielhaft für MySQL:

```
CREATE TABLE `data0tree` (
  `spi` smallint(6) DEFAULT NULL,
  `depth` int(11) DEFAULT NULL,
  `minx` double DEFAULT NULL,
  `miny` double DEFAULT NULL,
  `maxx` double DEFAULT NULL,
  `maxy` double DEFAULT NULL,
  `pointcount` int(11) DEFAULT NULL,
  `edgecount` int(11) DEFAULT NULL,
  `polygonedgecount` int(11) DEFAULT NULL,
  `elementcount` int(11) DEFAULT NULL,
  `id` int(11) NOT NULL DEFAULT '0',
  `md_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idx` (`md_id`,`spi`,`minx`,`miny`,`maxx`,`maxy`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

3.7.4. Daten-Tabellen für Tabellentyp `UNSTRUCTURED`

Der Informationsgehalt und die Struktur der Tabellen ist grundlegend identisch mit denjenigen des Typs `UNSTRUCTURED_MULTIPLE_DATASETS`, jedoch sind diese Tabellen aus Performancegründen für die räumlichen Suche nur für einen einzelnen Datensatz ausgelegt. Damit wird keine Information über die ID des zugehörigen Datensatzes benötigt und es ergeben sich Tabellenstrukturen vergleichbar zu denen des vorangegangenen Abschnitts, jedoch bildet die Punkt-Nummer den Primärindex und die Spalte `md_id` kann entfallen. Beispielhaft ist nachfolgend die Tabellenstruktur der Punktdaten-Tabelle als MySQL.Statement dargestellt:

```
CREATE TABLE `data0points` (
  `nr` int(11) NOT NULL DEFAULT '0',
  `x` double DEFAULT NULL,
  `y` double DEFAULT NULL,
  `z` float DEFAULT NULL,
```

```
  `status` tinyint(4) DEFAULT '1',
  `spi` int(11) DEFAULT NULL,
  PRIMARY KEY (`nr`),
  KEY `spidx` (`spi`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Der Aufbau der Tabellen für die Polygon-, Element- und Suchbaumdaten folgt diesem Grundschema (keine Information der ID des zugehörigen Datensatzes erforderlich) und folglich wird auf eine Einzeldarstellung der Tabellenstruktur an dieser Stelle verzichtet.

3.7.5. Daten-Tabellen für Tabellentyp `GRID`

Strukturierte Datensätze werden jeweils einzeln in einer Tabelle abgelegt. Hierbei steht die Strukturinformation (Referenzkoordinate, Rasterweiten, Zeilen und Spalten) unmittelbar im Tabellentyp, z.B. `GRID(x=470000.5,y=5929000.5,dx=1.0,dy=1.0,i=1000,j=1000)` In der Tabelle sind nunmehr lediglich die 1D-Rasterindizes (nr = Spaltenindex + Zeilenindex * j), die Z-Koordinate und das Status-Flag abgelegt. Aus den Strukturinformationen des Rasters und den 1D-Indizes können die Lage-Koordinaten der Rasterstützstellen effizient berechnet werden. Ein Suchbaum ist für Rasterdaten ebenfalls nicht erforderlich, da geometrische Suchoperation durch simple Indexrechnungen auf den Rasterstützstellen durchgeführt werden können.

SQL-Statement zur Erzeugung der Punktdaten-Tabelle eines Rasters beispielhaft für MySQL:

```
CREATE TABLE `data0points` (
  `nr` int(11) NOT NULL DEFAULT '0',
  `z` float DEFAULT NULL,
  `status` tinyint(4) DEFAULT '1',
  PRIMARY KEY (`nr`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

3.7.6. Daten-Tabellen für Tabellentyp `POSTGISRASTER` (nur PostgreSQL)

Speziell für die Verwendung einer PostgreSQL-Datenbank mit PostGIS-Aufsatz wurde die Unterstützung des Datentyps `raster` in die Datenbankschnittstelle integriert. Der Datentyp bietet eine effiziente Speicherung von Rasterdaten und wird durch eine umfangreiche, serverseitige Bibliothek von Rasteroperationen (auf Basis der GDAL-Bibliothek) unterstützt. Die angebotenen Rasteroperation erfüllen die in den Designanforderungen genannten Kriterien bezüglich eines selektiven Zugriffs auf Teildatenbereiche, so dass die Unterstützung des Rasterdatentyps nahtlos in die Datenbankschnittstelle als produktspezifische Sonderlösung aufgenommen werden konnte. Die Unterscheidung welche Ablagestrategie für Rasterdaten (Typ `GRID` oder `POSTGISRASTER`) verwendet wird, trifft automatisch der Tabellen-Manager der Datenbankschnittstelle.

Die Rasterdaten werden ausnahmslos in einer festen Tabelle gespeichert, die nachfolgende Struktur besitzt und neben dem Rastertyp, die ID des zugehörigen Datensatzes enthält:

```
CREATE TABLE public.pg_rasterdata
(
  id integer NOT NULL DEFAULT nextval('pg_rasterdata_id_seq'::regclass),
```

```
  md_id integer,
  grid raster,
  CONSTRAINT pg_rasterdata_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.pg_rasterdata
  OWNER TO postgres;

-- Index: public.pg_rasterdatamdidx

-- DROP INDEX public.pg_rasterdatamdidx;

CREATE INDEX pg_rasterdatamdidx
  ON public.pg_rasterdata
  USING btree
  (md_id);
```

3.8. Basisoperationen der Datenbankschnittstelle

Die Methoden der Datenbankschnittstelle sind in der Programmiersprache Java unter ausschließlicher Verwendung der JDBC-Schnittstelle implementiert. Generell werden die SQL-Statements derzeit im `AUTOCOMMIT`-Mode ausgeführt, d.h. eine Rollback-Option ist in diesem Modus nicht möglich. Für das Speichern der Geometriedaten in den Datentabellen werden „prepared statements“ bzw. „parameterized statements“ verwendet, die im Batch-Execution-Modus ausgeführt werden.

Hinsichtlich der Konfiguration der JDBC-Treiber werden dokumentierte Optionen ausgenutzt (z.B. `rewriteBatchedStatements=true` zur Beschleunigung des Batch-Inserts unter MySQL).

Nachfolgend sind die grundlegenden Operation der Datenbankschnittstelle mit technischen und fachlichen Details ihrer Realisierung aufgeführt.

3.8.1. Anlegen einer Datenbank

Das Anlegen von Datenbanken wird für alle Datenbanksysteme über die Datenbankschnittstelle ermöglicht, die mit einem „einfachen“ `CREATE DATABASE` – Statement ohne weitere Konfigurationsfiles, bestimmter Servermodi, etc. das Erzeugen von Datenbanken erlauben. Dies wird aktuell von PostgreSQL, MySQL, HSQL, Derby, etc. erfüllt. Für Oracle ist diese Option nicht verfügbar.

3.8.2. Löschen einer Datenbank

Das Löschen von Datenbanken erfolgt über `DROP DATABASE` -Statements. Lösch-Operation können jedoch über die Methoden der Datenbankschnittstelle nur auf Gismo-eigene Datenbanken durchgeführt werden. Die Filterung der Datenbanken wird anhand der Verfügbarkeit der Metadatentabellen vorgenommen.

3.8.3. Speichern eines Datensatzes

Das Speichern eines Datensatzes in die Datenbank ist eine der grundlegenden Datenbank-Operationen. Hierbei werden alle vorbereitenden Schritte von der Datenbankschnittstelle automatisiert übernommen. Diese sind im Einzelnen:

- Prüfung der topologischen Konsistenz der Geometrien
- Synchronisierung der Metadaten mit den Geometriedaten, u.a.
  - Aktualisierung der Bounding-Box
  - Aktualisierung der Geometriezahlen
  - Initialisierung der Interpolationsmethode mit einem für die Struktur der Daten geeigneten Verfahren (Nächster-Nachbar-Interpolation bei Punktmengen, lineare/bilineare Interpolation bei Triangulationen/Rastern)
- automatische Generierung von Hüllpolygonen (nicht-konvexe Hülle der Datenpunkte) zur grafischen Präsentation der Metadaten mit einer Methodik nach Nutzervorgabe
- Prüfung der zeitlichen Ausdehnung, andernfalls Initialisierung mit dem aktuellem Importdatum
- Prüfung des Lage-Bezugssystems (EPSG-Code), gegebenenfalls Setzen von -1 (nicht initialisiert)
- Prüfung des Titels, gegebenenfalls Übernahme des Titels aus Importfilenamen (wenn verfügbar)
- Prüfung/Vergabe eines Datensatz-Identifikators in Form einer UUID (Feld `metadata_id` der Core-Metadatentabelle)
- Setzen aller relevanten Zeitstempel (Importdatum = Datum der letzten Modifikation der Metadaten = letzte Modifikation der Daten)
- Prüfung und gegebenenfalls Anpassung der Datentabellenstruktur für die zu importieren den Daten

Bei der Prüfung/Vergabe des Datensatz-Identifikators (UUID) wird nach folgenden Regeln vorgegangen:

- Generierung einer UUID beim Import, wenn keine ID im Datensatz vorhanden ist
- Generierung einer neuen UUID, wenn in der Datenbank die UUID des Datensatzes bereits von einem anderen Datensatz verwendet wird
- Übernahme der UUID, wenn kein Datensatz in der Datenbank diese UUID besitzt

Die erweiterte Logik zur Übernahme der Metadaten-ID erfolgt in Anbetracht der Tatsache, dass in ausgewählten Dateiformaten der komplette Umfang an Metadaten eines Datensatzes gespeichert werden kann (UGRID-NetCDF) und somit beim Re-Import eines derartigen Datensatzes durchaus identische UUIDs auftreten können. Weiterhin wird auf eine generelle Neu-Vergabe der ID beim Import verzichtet, um das Anwendungsszenario einer kompletten Spiegelung der Daten eines Servers auf einem anderen Datenbankserver zu erlauben. Über die dann auf beiden Servern identischen Metadaten-IDs der Datensätze können Synchronisationsmechanismen für die Datenbestände konzipiert werden.

Das Schreiben des Datensatzes erfolgt mehrschrittig durch Schreiben der Metadaten, der Punktdaten, der Polygondaten, der Elementdaten und der Baumstruktur.

Nach erfolgtem Import werden die Informationen der Datenbankverbindung (Server, Datenbank, DB-ID) in die Metainformationen des Datensatzes übernommen, d.h. der Datensatz „kennt“ fortan seine Herkunft. Im gleichen Zuge wird er für die mögliche Bearbeitung durch andere Nutzer gesperrt. Die Sperrung erfolgt durch Eintragung eines Sperrvermerks in die Metadaten des Datensatzes in der Datenbank. Die Sperre wird dann zurückgesetzt, wenn der Datensatz aus Gismo (bzw. aus dem Datenmodell) entfernt wird.

3.8.4. Extraktion eines Datensatzes

Für die Extraktion eines Datensatzes werden drei grundlegende Modi unterschieden:

- **Extraktion zur Bearbeitung**: der Datensatz wird vollständig aus der Datenbank in das Datenmodell transferiert, wobei die Geometriedaten und Metadaten bearbeitet und in die Datenbank zurückgeschrieben werden können. Dafür werden die Information der Datenbankverbindung (Server, Datenbank, DB-ID) mitgeführt und der Datensatz für die parallele Bearbeitung durch Dritte gesperrt.

  Im Rahmen der Bearbeitung des Datensatzes kann auch nur eine Modifikation der Metadaten vorgenommen werden. Nach dem Editieren und Schließen des Metadaten-Editors erhält der Anwender die Möglichkeit, die modifizierten Metadaten in die Datenbank zurückzuschreiben. Hierbei werden nur Metadaten und keine Geometriedaten geschrieben. Erfolgt keine weitere Bearbeitung der Geometriedaten, ist auch keine Aktualisierung des Gesamtdatensatzes in der Datenbank mehr erforderlich.

- **Lesende Extraktion**: der Datensatz wird vollständig aus der Datenbank in das Datenmodell transferiert, die Informationen zur Datenbankverbindung werden jedoch nicht übernommen und der Datensatz wird folglich nicht gesperrt. Innerhalb von Gismo stehen dem Anwender alle Optionen zur Datenmanipulation, der Datenanalyse oder auch des Exports in das Filesystem zur Verfügung, jedoch ist das Zurückschreiben des Datensatzes in die Datenbank nicht möglich.

- **Nachladbare Extraktion**: die nachladbare Extraktion ist ein ausschließlich lesender Zugriffsmodus. Hierbei werden zunächst nur die Metadaten aus der Datenbank extrahiert (grafische Präsentation über die Hüllpolygone in der GUI). Die Nachladelogik der Datenbankschnittstelle erlaubt dann die Extraktion der Geometriedaten innerhalb einer räumlichen Bounding-Box (Zoombereich in Gismo), wobei im Falle unstrukturierter Daten die räumliche Strukturierung des Datensatzes mit dem gespeicherten Suchbaum und bei Rasterdaten die strukturierte Anordnung für die räumliche Filterung ausgenutzt wird.

  Im Zuge der Nachlade-Operation werden bei unstrukturierten Daten die Geometrien innerhalb der Bounding-Box in das Datenmodell transferiert, wobei eventuell bereits vorhandene Geometrien im Datensatz des Datenmodells vollständig ersetzt werden (z.B. bei wiederholt ausgeführten Nachlade-Operationen). Die importierten Teildaten können daraufhin mit den Methoden des grafischen Frontends visualisiert, analysiert, modifiziert oder auch in Dateien exportiert werden. Lediglich das Zurückschreiben in die Datenbank ist nicht möglich.

  Die Nachladelogik für strukturierte Daten ist darüber hinaus mit einem Level-Of-Detail-

Ansatz kombiniert, so dass über die Ausdehnung der Bounding-Box des Nachladebereichs eine automatisierte „Vergröberung“ der Rasterdaten automatisiert durchgeführt werden kann. Die Vergröberung der Rasterdaten erfolgt hierbei nach der Logik „Extrahiere nur jeden x-ten Rasterdatenpunkt“. Die technische Realisierung erfolgt bei der Tabellenstruktur vom Typ `GRID` durch ein `SELECT`-Statement mit eingearbeiteter Modulo-Operation in der Bedingung zur Filterung der Rasterindizes. Im Fall der PostGIS-Rasterdatenstruktur wird eine serverseitige geometrische Operation zur Re-Interpolation auf ein gröberes Raster genutzt. Diese Operation wird mit einem einzelnen `SELECT`-Statement ausgeführt.

Ein generelles Merkmal bei der Extraktion von Datensätzen aus der Datenbank ist darin gegeben, dass die einzelnen Geometrien die Werte ihrer Primärschlüssel aus den Datentabellen „verlieren“.

Zur Realisierung der unterschiedlichen Extraktionsmodi existieren in der Datenbankschnittstelle jeweils unterschiedliche Methoden zum Laden des Gesamtdatensatzes sowie des Teildatensatzes.

3.8.5. Aktualisierung eines Datensatzes

Die Aktualisierung eines Datensatz ist auf Basis der Konzeption der Extraktionsmodi nur für Datensätze möglich, die „zur Bearbeitung“ aus der Datenbank geladen werden. Hinsichtlich der Modifikation der Daten mit den Methoden der Klassenbibliothek sind hierbei jedoch keine Beschränkungen aktiv. Das heißt insbesondere, dass grundlegende, strukturelle Änderungen an einem Datensatz, der aus der Datenbank geladen wurde, durchgeführt werden können, z.B.:

- Löschen / Vergröberung der Datenpunkte
- Vermaschung der Datenpunkte
- Hinzunahme von Polygonen als „Strukturpolygone“
- Zusammenfügen/Trennen von Datensätzen
- Änderung der Attributstruktur
- etc.

Die möglichen Strukturänderungen müssen beim Zurückschreiben der Daten zwingend beachtet werden, so dass die Aktualisierung von Datensätzen die komplexeste Teilaufgabe der Datenbankschnittstelle darstellt. Hierzu ist anzumerken, dass ein detailliertes Modifikations-Tracking bezüglich geometrischer und / oder struktureller Veränderungen innerhalb des Datenmodells derzeit nicht unterstützt wird. Aus diesem Grund wird der Ansatz verfolgt, die Aktualisierung von Datensätzen als

- Update der Metadaten und
- Re-Import der Geometriedaten

aufzufassen. Für die Realisierung der Datenbankschnittstelle bedeutet dies, dass beim Aktualisieren eines Datensatzes zunächst die vorhandenen Geometriedaten aus den Datentabellen vollständig entfernt werden, dann die Tabellenstruktur durch den Tabellen-Manager geprüft und gegebenenfalls angepasst wird und im Abschluss Metadaten und Geometriedaten aktualisiert bzw. reimportiert werden. Im Worst-Case-Szenario hat diese Strategie bei minimalen Daten-Änderungen (z.B. Modifikation einzelner Z-Werte) den aufwändigen Re-Import umfangreicher Geometriedaten zur Folge. Bezüglich eines verbesserten Modifikations-Trackings innerhalb des Datenmodells wurden bereits Studien unternommen, die beispielsweise topologische und geometrische Checksums bewerten. Diese Ansätze wurden jedoch bislang nicht als ausreichend fail-safe erachtet, um die Datenkonsistenz der zu speichernden Daten zu

gewährleisten. Generell könnte jedoch ein spezieller Ansatz aus Nutzung des Status-Flags der Punkt-Geometrien zur Erfassung von Änderungen des Z-Wertes (Modifikations-Tracking verankert in den Bearbeitungsmethoden), sowie der Prüfung von geometrischen und topologischen Checksums für den Datensatz (d.h. haben sich Anzahl und Lage der Datenpunkte verändert) eine verbesserte Verfolgung von Nutzeränderungen im Datenmodell ermöglichen. Auf Basis dieser Analysen wäre eine effizientere Aktualisierungs-Logik für Datensätze umsetzbar.

Weiterhin sind innerhalb der Methoden der Klassenbibliothek bezüglich des Trennens / Zusammenfügens von Datensätzen folgende Regeln implementiert:

- beim Trennen eines Datensatzes behält der Datensatz außerhalb der Maskierungspolygone die Verbindung zur Datenbank, die getrennten Geometrien können als Neu-Import in die Datenbank eingepflegt werden
- beim Zusammenfügen zweier Datensätze behält derjenige Datensatz seine Verbindung zum Datenbankserver, welcher der Ziel-Layer bei der Operation war. Der Einfüge-Layer wird aus dem Datenmodell entfernt (existiert u.U. aber noch in der Datenbank!).

Generell bedeutet die Ausführung der Trenn- bzw. Zusammenfüge-Operationen im Datenmodell keine automatische Aktualisierung der Datenbank. Die Änderung der Daten in der Datenbank muss dann aktiv durch den Anwender ausgeführt werden.

3.8.6. Löschen von Datensätzen

Beim Löschen von Datensätzen werden die Metadateneinträge eines Datensatzes, sowie die Geometriedateneinträge aus den Datenbanktabellen komplett entfernt. Im Fall der Tabellenstrukturen `UNSTRUCTURED` und `GRID`, wird diese Operation als `DROP TABLE` - Statement ausgeführt (effiziente Operation), da in diesen Datentabellen nur die Daten des einzelnen Datensatz gespeichert sind. Für die Tabellenstrukturen mit multiplen, unstrukturierten Datensätzen erfolgt die Lösch-Operation durch Entfernung der Daten in Abhängigkeit ihrer DB-ID (Tabellentypen `UNSTRUCTURED_MULTIPLE_DATASETS` und `POSTGISRASTER`).

3.8.7. Extraktion eines Metadaten-Layers

Ein Metadaten-Layer wird durch eine Recherche-Anfrage an einen oder mehreren Datenbankservern erzeugt. Innerhalb der Recherche wird der Datenbestand nach benutzerdefinierten Regeln gefiltert, z.B. durch

- räumliche und zeitliche Filterung
- Filterung nach Aufnahmeverfahren / Datenart
- Filterung nach Datenbanken / Ablageort
- etc.

Ergebnis der Anfrage ist eine geordnete Liste von Metadaten, wobei jedes Metadatum jeweils als „Repäsentant seines Datensatzes“ aufzufassen ist. Die Sortierung der Metadaten wird im Zuge der Recherche-Anfrage festgelegt, in der Regel erfolgt eine zeitliche Sortierung. Für bestimmte Anwendungsszenarios kann auch eine vom Anwender definierte Prioritätenfolge (z.B. Sortierschema nach Aufnahmeverfahren: Lidardaten vor Fächerecholotdaten vor Single-Beam-Daten vor ...) sinnvoll sein.

Die Liste der Metadaten wird in einem Layer zusammengefasst und der Multi-Modellstruktur des

Datenmodells hinzugefügt. Im weiteren werden diese speziellen Layer als *Metadaten-Layer* bezeichnet. Die grafische Präsentation eines Metdaten-Layers erfolgt in den Visualisierungskomponenten über die Hüllpolygone (nicht-konvexe Ausdehnungsbeschreibung des Datensatzes) der Metadaten.

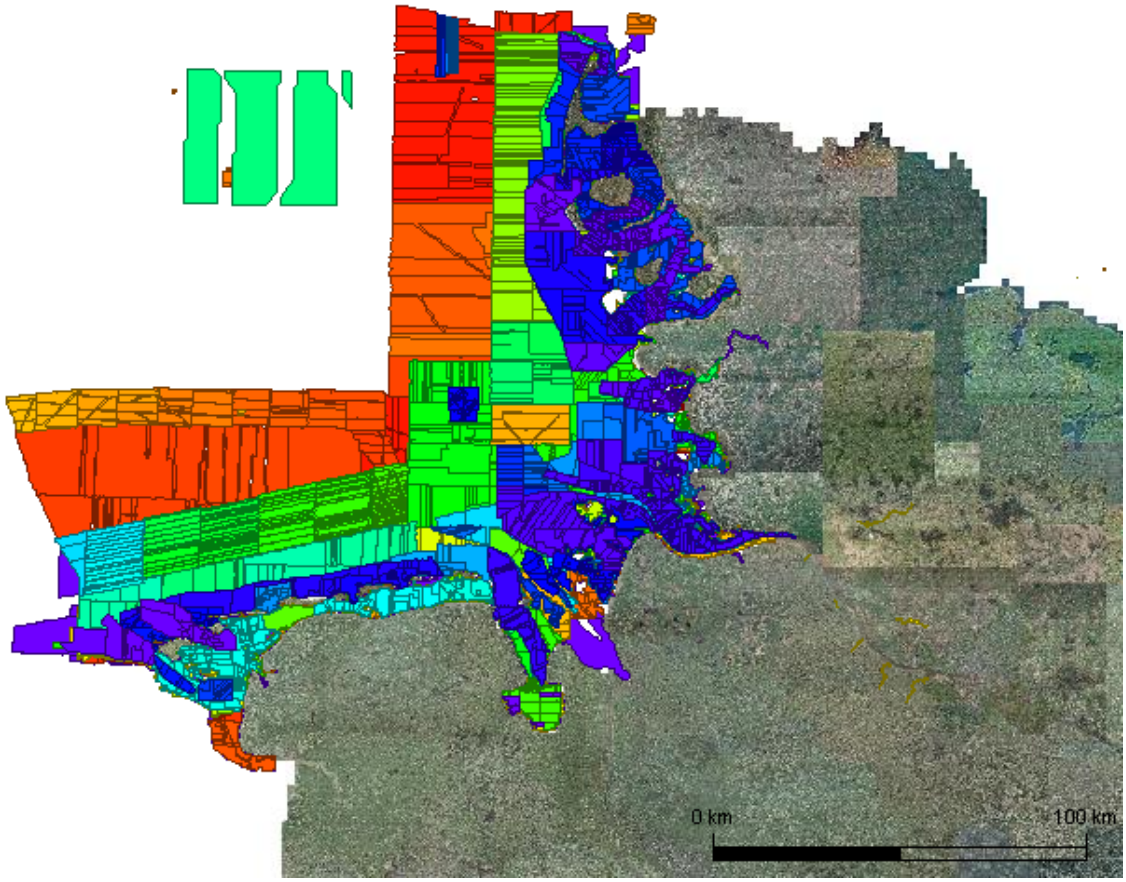


Abbildung 15: Metadaten-Layer mit 3528 Metadaten bzw. Datensätzen und georeferenziertem Hintergrundbild

Generell enthält der Metadaten-Layer keine Geometriedaten, sondern ist strikt dafür ausgelegt, beliebigen Methoden den selektiven Zugriff auf die Geometriedaten über die Metadaten zu organisieren. Zur Optimierung redundanter Zugriffe auf die Geometriedaten ist der Metadaten-Layer mit der in Kapitel 3.5 beschrieben Caching-Strategie für den Datendurchgriff ausgestattet.

3.9. Datenbankgestützte Methoden

Unter datenbankgestützten Methoden wird die Nutzung des Datenbankinhalts über die Zugriffsmethoden der Datenbankschnittstelle direkt in den Methoden zur Datenmanipulation und Datenanalyse der Klassenbibliothek verstanden. Zentrale Rolle spielt hierbei das Konzept des Metadaten-Layers, welcher die abstrahierte Datenzugriffsschnittstelle für die Methoden des Datenmodells darstellt. Als einfaches Beispiel kann hier die Interpolation von Stützstellen (z.B. eines numerischen Modells) auf einem Digitalen Geländemodell angeführt werden. Im

“klassischen“ Ansatz der Multi-Modellstruktur des Datenmodells, werden die Daten des DGMs vollständig in den physikalischen Speicher importiert und dann die Stützstellen aus dem Modell nach Vorgabe einer Interpolationsvorschrift generiert. Über den Ansatz der datenbankgestützten Methoden wird nunmehr die Datenbank mit der Zugriffsschnittstelle als „Digitales Geländemodell“ aufgefasst. Die „Vermittlerrolle“ spielt in diesem Fall der Metadaten-Layer, d.h. in der Multi-Modellstruktur kann die Interpolation von Stützstellen auf einem Metadaten-Layer durchgeführt werden. Die besonderen Regeln der Interpolation werden nutzerdefiniert festgelegt (Raum-Zeit-Interpolation, Interpolation nach Prioritätenvorgabe, etc.) und bestimmte Informationen (z.B. räumliche Interpolationsmethodik des Datensatzes) werden in den Metadaten erwartet.

Innerhalb der Analyse- und Bearbeitungsmethoden erlauben folgende Funktionen die Angabe von Metadaten-Layern als Eingangsparameter:

- Rastergenerierung: Metadaten-Layer als DGM
- Interpolation beliebiger Stützstellen (numerische Modelle, etc.): Metadaten-Layer als DGM
- Differenzenbildung zwischen zwei Metadaten-Layer an vorgegebenen Stützstellen (Raster, unstrukturiert)
- Generierung bathymetrischer Zeitreihen (zeitlicher Durchstich am Ort): Metadaten-Layer als DGM
- Profilschnittanalyse auf einem Metadaten-Layer
- Datenpunkte löschen auf einem Metadaten-Layer (Durchlauf über alle Datensätze für Zuschneide-Operation mit Maskierung)
- Z-Koordinaten von Datenpunkten editieren auf einem Metadaten-Layer (Durchlauf über alle Datensätze für Z-Modifikation mit Maskierung)

Die Integration von Metadaten-Layern in weitere Bearbeitungs- und Analysemethoden wird sukzessive verfolgt und ausgebaut.

4. Administration der Datenbankserver und Konfiguration von Gismo

4.1. Hardwarevoraussetzungen

Die Datenbankschnittstelle von Gismo stellt keine speziellen Mindestanforderungen an die zu verwendende Hardware. Im aktuellen Betrieb der Datenbankschnittstelle von Gismo sind von Server-Lösungen mit täglichem Backup-Mechanismen bis zum Einsatz handelsüblicher PCs unterschiedlichste Hardwarekonfigurationen erfolgreich im Einsatz.

Generell ist jedoch eine ausreichende Speicherausstattung für den Datenbankserver sicherzustellen, um die zu erwartenden Datenmengen verwalten zu können. Die Datenmengen richten sich sowohl nach der Struktur der abgelegten Daten (strukturiert / unstrukturiert, vermascht / unvermascht, mit / ohne Attribute), als auch nach dem eingesetzten Datenbanksystem.

Als Richtwerte für einen triangulierten Datensatz (ohne Attribute) mit 3,2 Mio. Datenpunkten, 75 Polygonen und 6,4 Mio. Dreiecken sind produktabhängig folgende Datensatz- / Datenbankgrößen zu erwarten:

JBF, Datei [MB]	MySQL [MB]	PostgreSQL [MB]	Oracle [MB]
188	628	1066	942

Die Datengrößen lassen sich für weitere Datensätzen mit gleicher Struktur näherungsweise linear fortschreiben.

4.2. Einrichtung des Datenbankservers

Die Konzeption der Datenbankschnittstelle von Gismo ist darauf ausgerichtet, bestehende relationale Datenbanksysteme einer IT-Infrastruktur nutzen zu können. Für ein verfügbares Datenbanksystem ist lediglich die Einrichtung eines Master-Datenbank-Users für Gismo mit den entsprechenden Privilegien zum Anlegen und Löschen von Datenbanken, Tabellen und Indizes als Voraussetzung für die Inbetriebnahme der Datenbankschnittstelle von Gismo erforderlich (vgl.4.2.1).

Ist kein relationales Datenbankmanagementsystem ad hoc verfügbar, so wird trotz des generischen, produktunabhängigen Ansatzes die Verwendung der Systeme MySQL, PostgreSQL oder Oracle empfohlen, da hier die umfangreichsten Erfahrungen bezüglich einer Nutzung der Datenbankschnittstelle von Gismo vorliegen.

Bei einer Neu-Einrichtung des Datenbankservers hat sich die Verwendung einer Standard-Konfiguration für die Systeme PostgreSQL und MySQL in der Regel als ausreichend erwiesen. Hierzu stehen umfangreiche Tutorials und Anleitungen im Internet zur Verfügung.

4.2.1. Einrichtung eines Master-Datenbank-Users für Gismo

Der Master-Datenbank-User wird in Gismo zum Verbindungsaufbau zu einem Datenbankserver benötigt. Hierbei ist zwingend zu beachten, dass dieser DB-Zugang die für Gismo erforderlichen Rechte erhält, andernfalls ist die Datenbankschnittstelle nicht bzw. nicht vollumfänglich nutzbar.

Nachfolgend ist beispielhaft die Einrichtung eines Accounts für die Systeme MySQL und Oracle gezeigt.

MySQL mit Workbench

Bei der Administration der Nutzer und deren Privilegien mit der grafischen Oberfläche *MySQL Workbench* wird die Konfiguration des Nutzers über die Auswahl globaler Berechtigungen vorgenommen (Reiter *Administrative Roles*). Als Rolle wird dann *DBManager* ausgewählt (Abbildung 16)

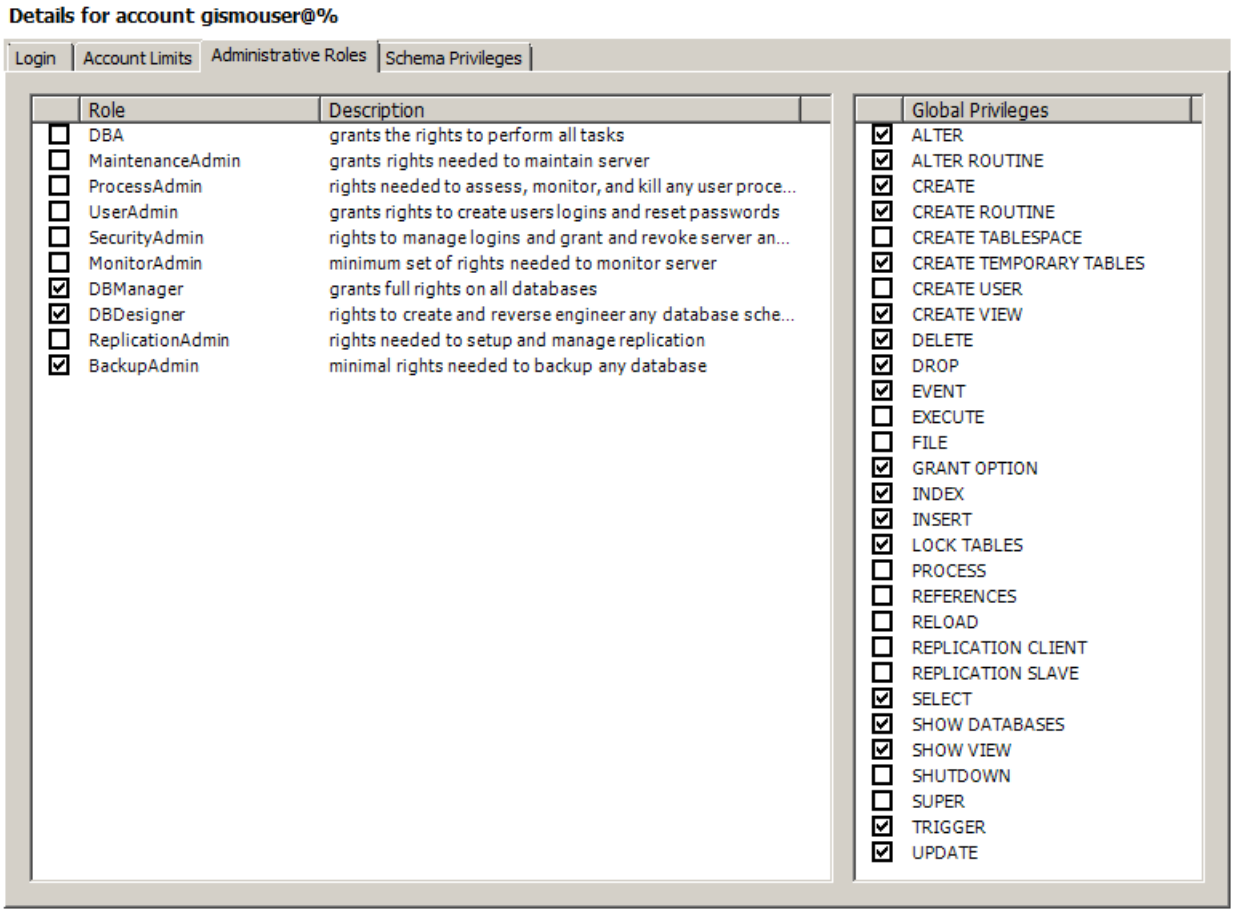


Abbildung 16: Auswahl der Rolle DBManager in MySQL Workbench zur Definition der notwendigen Berechtigungen eines Gismo-Datenbank-Users

Oracle

Die Einrichtung eines Gismo-konformen Nutzerprofils wird für Oracle anhand nachfolgendem SQL-Script ermöglicht:

```
CREATE USER GISMOUSER
  DEFAULT TABLESPACE DATA
  PROFILE DEFAULT
  ACCOUNT UNLOCK;
  -- 1 Role for GISMOUSER
  GRANT CONNECT TO GISMOUSER;
  ALTER USER GISMOUSER DEFAULT ROLE ALL;
```

```
-- 9 System Privileges for GISMOUSER
GRANT CREATE TRIGGER TO GISMOUSER;
GRANT CREATE TYPE TO GISMOUSER;
GRANT DEBUG ANY PROCEDURE TO GISMOUSER;
GRANT CREATE PROCEDURE TO GISMOUSER;
GRANT CREATE SYNONYM TO GISMOUSER;
GRANT CREATE TABLE TO GISMOUSER;
GRANT CREATE SEQUENCE TO GISMOUSER;
GRANT CREATE INDEXTYPE TO GISMOUSER;
GRANT CREATE VIEW TO GISMOUSER;
```

4.2.2. Weitere Konfigurationseinstellungen

Weitere vorbereitende Konfigurationsschritte (z.B. das externe Anlagen einer Tabellenstruktur) sind für de Nutzung der Datenbankschnittstelle nicht mehr erforderlich. Die Erzeugung der internen Organisationsstruktur für die Daten durch Anlegen der notwendigen Tabellen innerhalb der Datenbank(en) wird automatisiert von der Datenbankschnittstelle übernommen.

4.3. Aufbau einer Verbindung zum Datenbankserver in Gismo

Nach Einrichtung der Nutzerdaten für den Master-Datenbank-User, kann über die grafische Benutzeroberfläche von Gismo der Verbindungsaufbau zu einem oder mehreren Datenbankservern aufgebaut werden. Die Einzelverbindungen werden dabei in benannten Serververbindungslisten verwaltet, so dass für den erstmaligen Verbindungsaufbau die Arbeitsschritte:

- Erzeugung einer Serververbindungsliste und deren Umbenennung (Abbildung 17)
- Hinzufügen einzelner Datenbankverbindungen (Abbildung 18)

erforderlich ist.

Die erstellte Serververbindungsliste wird im Gismo-Homeverzeichnis des Anwenders verschlüsselt abgelegt und steht dem Anwender in der nächsten Bearbeitungssession mit Gismo automatisch wieder zur Verfügung. Eine erneute Eingabe der Verbindungsdaten zu den Datenbanken ist bei Nutzung einer vorhandenen Verbindungsliste dann nicht mehr erforderlich. Innerhalb einer Serververbindungsliste können zudem beliebig viele Einzelverbindungen verwaltet werden.

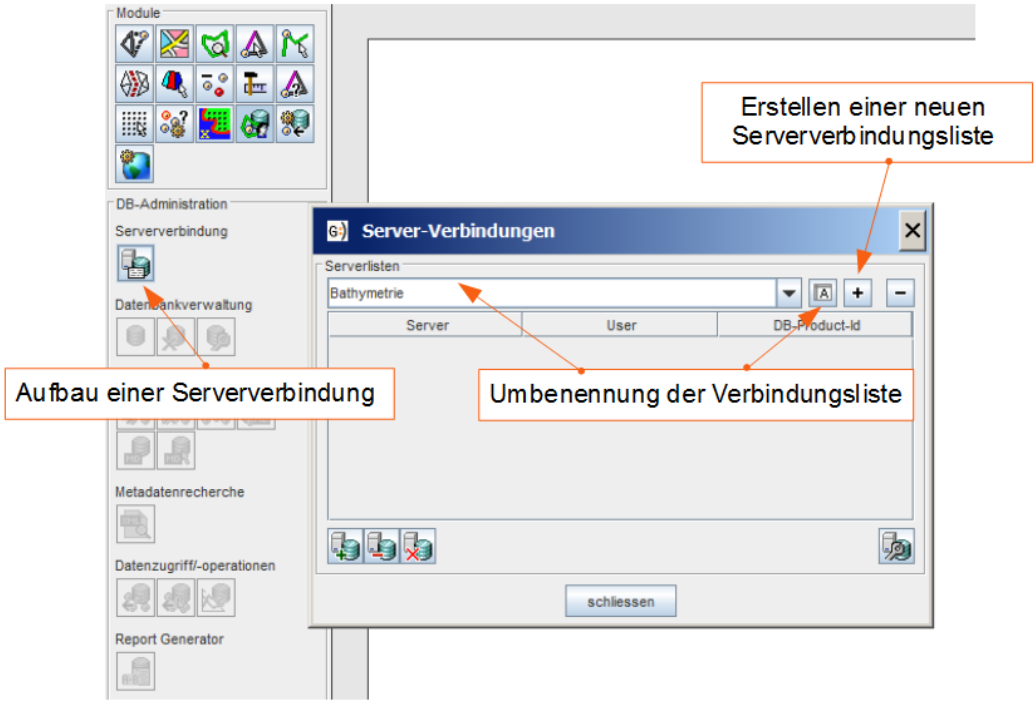


Abbildung 17: Erstellen einer neuen Serververbindngsliste in Gismo

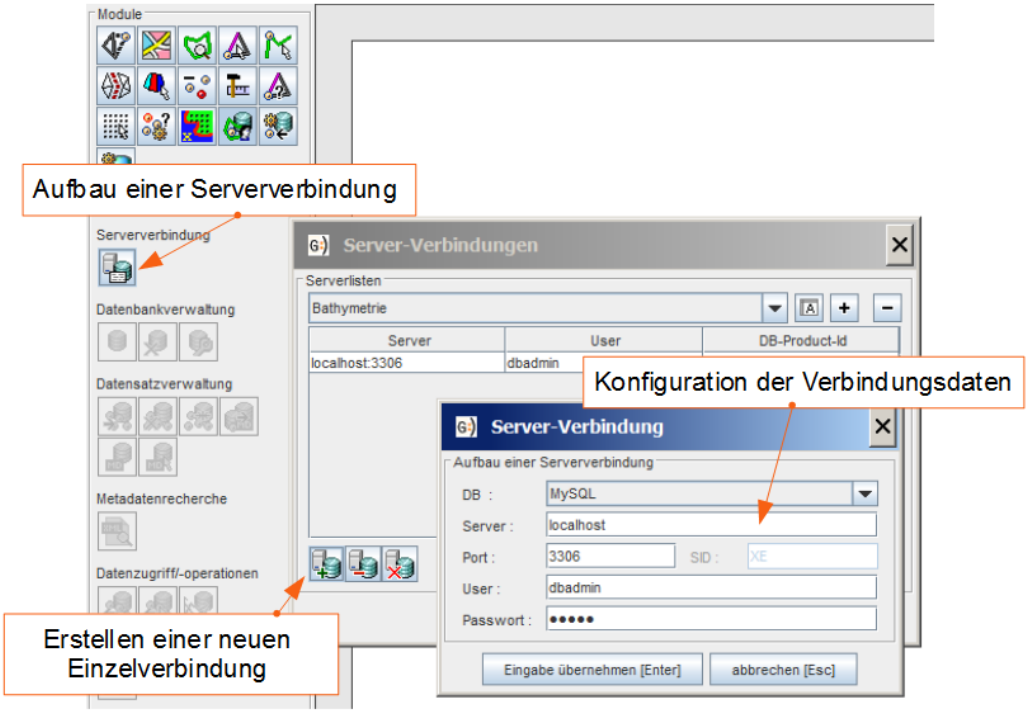


Abbildung 18: Hinzufügen einer Datenbankverbindung in Gismo

4.4. Konfiguration von Gismo zum automatisierten Datenbankverbindungsaufbau beim Programmstart

Die Datenbankschnittstelle von Gismo kann für einen automatischen Verbindugsaufbau zu einem oder mehreren Datenbankservern konfiguriert werden. Hierzu erwartet das Programmsystem eine Serververbindungsliste mit dem Dateinamen „auto_connection.props“ entweder im Installationverzeichnis von Gismo unter:

`INSTALLATION_DIR/smilesoftware/config/auto_connection.props`

oder im Homeverzeichnis des Anwenders unter

`USER_HOME/.gismo/config/auto_connection.props`

Beim Programmstart von Gismo wird zunächst im Homeverzeichnis und dann im Installationsverzeichnis nach einer Default-Datenbankverbindung gesucht. Der Verbindung im Homeverzeichnis wird somit eine höhere Priorität eingeräumt und erlaubt dem Anwender die Installationskonfiguration von Gismo individuell zu übersteuern.

Die Einrichtung erfolgt über das Dialogfenster zur Verwaltung der Server-Verbindungslisten im Administrator-Datenbankmodul (Abbildung 19). Basierend auf der aktuell gewählten Server-Verbindungsliste stehen über die Administrationsfunktionalität nachfolgende Optionen zur Verfügung:

- Default-Datenbankverbindung im Installationsverzeichnis löschen
- Auswahl als Default-Datenbankverbindung in das Installationsverzeichnis kopieren
- Default-Datenbankverbindung in den Benutzereinstellungen löschen
- Auswahl als Default-Datenbankverbindung in die Benutzereinstellungen kopieren

Für die Modifikation der Installationskonfiguration von Gismo müssen Schreibberechtigungen im Installationsverzeichnis bestehen.

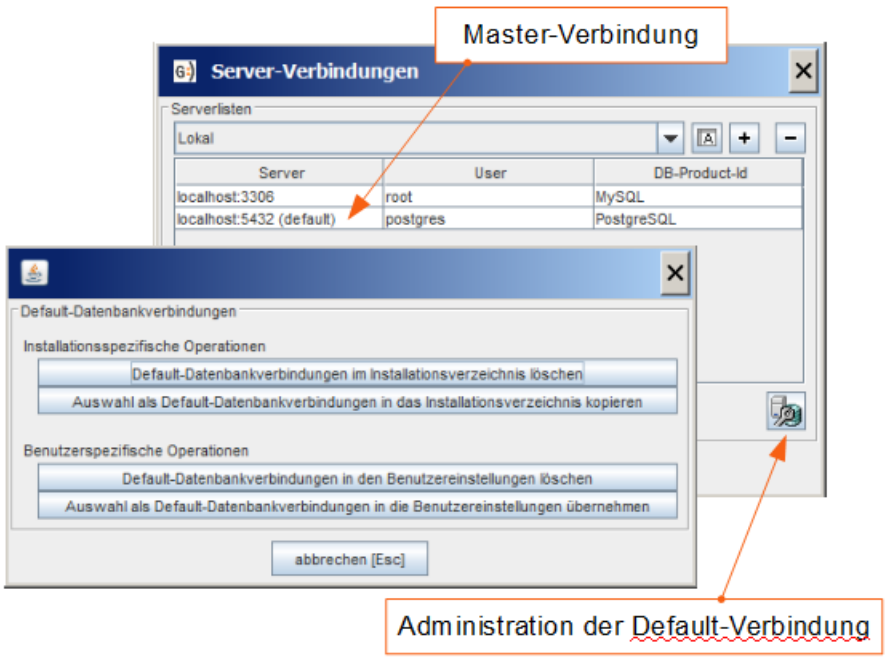


Abbildung 19: Dialogfenster zur Einstellung der Default-Datenbankverbindung und Festlegung der Master-Verbindung

Enthält die Verbindungsliste mehr als einen Datenbankserver, so kann innerhalb der Liste ein Server als Master-Verbindung ausgewählt werden. Die Auswahl erfolgt in der tabellarischen Ansicht der Serverbindungslisten-Konfiguration durch Betätigung der rechten Maustaste auf einem Tabelleneintrag. Die gewählte Master-Verbindung wird mit dem Zusatz `(default)` gekennzeichnet.

4.5. Einrichtung einer Nutzerverwaltung

Voraussetzung für die Einrichtung einer applikationsspezifischen Nutzerverwaltung ist einerseits die Verwendung einer speziell konfigurierten Gismo-Version, die bei jedem Programmstart gültige Nutzerdaten abfragt, sowie die Verfügbarkeit einer Default-Datenbankverbindung (vgl. Kapitel 4.4). Sind diese Voraussetzungen erfüllt, kann die Administration der Nutzerverwaltung im Datenbank-Modul unter der Rubrik *Datenbankverwaltung* in der Administrator-Tool-Funktion durch einen Anwender mit Administratorrechten erfolgen. Beim erstmaligen Aufruf der Option *Nutzerverwaltung* der Toolbox-Funktionen werden alle erforderlichen Tabellenstrukturen in der Datenbank der Default-Verbindung automatisch angelegt und daraufhin eine grafische Eingabeoberfläche zum Anlegen, Editieren und Löschen von Nutzern geöffnet (Abbildung 20). Über die grafische Benutzeroberfläche werden die Nutzer mit den zugeordneten Rollen (vgl. Kapitel 3.4) angelegt und abschließend in der Nutzerdatenbank gespeichert.

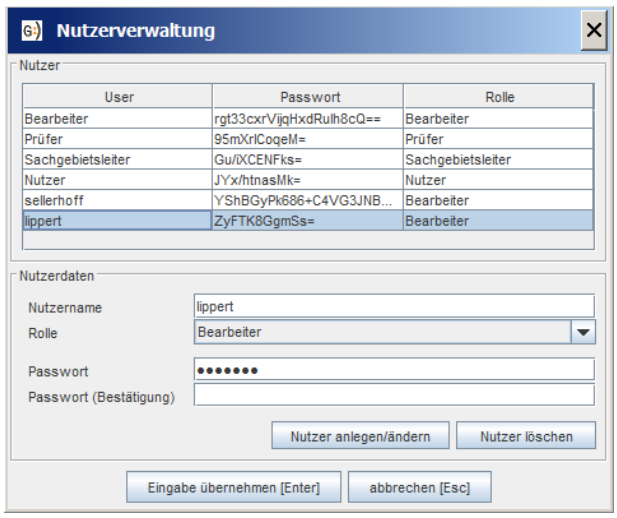


Abbildung 20: Eingabemaske der Benutzerverwaltung

5. Grundlagen der Organisation und Pflege des Datenbestandes

5.1. Datenorganisation

Für die Datenorganisation, d.h. die Strukturierung der Datenbankserver und der Datenbanken, sind in Gismo generell keine Vorgaben bzw. Einschränkungen aktiv. Die Organisation des Datenbestandes kann nach aufgabenspezifischen Erfordernissen frei ausgestaltet werden.

Aus dem praktischen Anwendungsbetrieb haben sich jedoch geeignete Organisationsformen für große Datenbestände herauskristallisiert. Hierbei kann folgende Grundaufteilung für eine zentral verwaltete Datenhaltung als besonders geeignet angesehen werden:

- **Quelldatenserver**: in den Quelldatenserver werden die bathymetrischen Daten mit allen verfügbaren Attributen in ihrer originären Struktur (die Daten enthalten u. U. auch noch Wasserflächen) eingepflegt
- **Modellierungsdatenserver**: aus dem Quelldatenserver werden die Daten in den Modellierungsdatenserver übernommen. Hierbei werden nur die essentiell wichtigen Attribute mitgeführt. Innerhalb des Modellierungsservers werden die Daten nach aufgabenspezifischen Gesichtspunkten modelliert. Zu den Modellierungs- / Modifikationsschritten könnten gehören:
  - Triangulation unstrukturierter Datensätze
  - evtl. Zusammenfügen von Datensätzen (sofern zeitlich „dicht“ genug beisammen)
  - Ergänzung des Datenbestandes durch Hinzufügen von „Ersatzsystemen“ (z.B. Modelle zum Schließen von Datenlücken)
  - etc.
- **Produktdatenserver**: im Produktdatenserver werden aus dem Modellierungsserver abgeleitete Produkte verwaltet. Gängige Produkte sind beispielsweise Rasterkacheln eines konsistenten Digitalen Geländemodells, die auf dem Datenbestand für einen Zeitpunkt durch Interpolation generiert wurden.

Aufgabenspezifisch kann die Zusammenlegung von Quell- und Modellierungsdatenserver sinnvoll sein, wenn eine strikte Trennung hier nicht notwendig oder zielführend ist (z.B. wegen des signifikant höheren Datenvolumens).

Innerhalb der einzelnen Server ist eine weitere Strukturierung durch Anlegen von Datenbanken sinnvoll. Als mögliche Strukturierung bieten sich Aufteilungen nach Datenerheber, sowie Datenbanken für begrenzte Zeiträume an. Die Unterstrukturierung auf Datenbankebene kann mit der Funktionalität von Gismo vorgenommen werden (mit Ausnahme einer Verwendung von Oracle).

Auf Basis der unterschiedlichen, zentralen Datenserver besteht die Möglichkeit, lokale Projektdatenserver aufzubauen. Hierzu stehen Funktionen zur Selektion von Datenbanken bzw. Datensätzen mit der entsprechenden Kopierfunktionalität in Gismo zur Verfügung.

Von der Grundkonzeption der Datenbankschnittstelle lassen sich Server unterschiedlicher Datenbanksysteme problemlos kombinieren, so dass beispielsweise für den Produktserver abweichend von den weiteren Servern die PostgreSQL/PostGIS-Lösung zur effizienten Speicherung von Rasterkacheln genutzt werden könnte.

5.2. Datenimport in die Datenbank

Das Befüllen der Datenbanken erfolgt über die Funktionalität von Gismo, entweder

- durch den Import einzelner aus dem Dateisystem geladener Datensätze in die Datenbank
- oder durch einen Batch-Import von Dateien aus einer Verzeichnisstruktur (optional mit Unterverzeichnissen) unter Angabe des Dateiformates und von Filterkriterien

Beide Ansätze stehen gleichberechtigt nebeneinander, wobei jedoch der erste Ansatz die sukzessive Interaktion des Anwenders auf der grafischen Benutzeroberfläche von Gismo erfordert. Für den Import einer Vielzahl von Datensätzen aus dem Dateisystem ist dieser Ansatz nur bedingt geeignet, so dass in diesem Szenario der Batch-Import die zu bevorzugende Methodik darstellt.

Voraussetzungen für die Anwendung des Batch-Importes sind

- die zu importierenden Dateien haben eine identische Struktur und können mit demselben Dateifilter verarbeitet werden
- die Datensätze sollten einen vergleichbaren Inhalt (z.B. das gleiche Lagesystem) besitzen
- die Ziel-Datenbank ist für alle Datensätze gleich

Im Zuge des Batch-Importes können weitreichende Optionen für die automatisierte Prozessierung der Daten eingestellt werden (Konfigurationsmaske in Abbildung 20):

- automatisierte Rastererkennung von Daten, die in einem Dateiformat ohne Rasterstruktur-Information vorliegen (i.d.R. Tripeldaten im Ascii-Format)
- automatisierte Koordinaten-Reduktion (z.B. Reduktion 8-stelliger UTM-Koordinaten auf 6-stellige Darstellung)
- automatisierte Koordinaten-Transformation (Wahl der Transformationsbibliothek. Geotools oder NTv2-konforme Transformation über die Proj4-Bibliothek)
- automatisierte Triangulation und Bestimmung einer nicht-konvexen Berandung
- Methoden zur Datenvalidierung
- Wahl der Methodik der Hüllpolygon-Generierung (für grafische Präsentation)

Die besondere Effizienz des Batch-Imports im Vergleich zum Einzeldatensatz-Import ist also nicht allein durch die multiplen Datei-Import gegeben, sondern vor allem auch durch die Zusammenfassung unterschiedlichster vorgeschalteter Bearbeitungsoptionen.

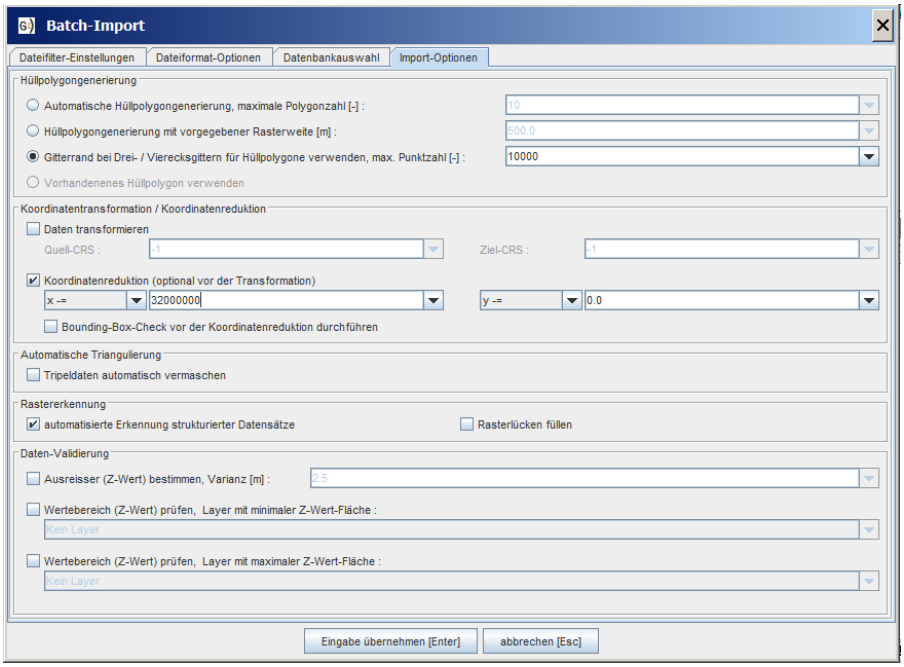


Abbildung 21: Optionen des automatisierten Prozessings der Daten beim Batch-Import

Beim Daten-Import werden dateiformatabhängig alle verfügbaren Metadaten übernommen. Weiterhin erfolgt die automatische Generierung elementarer Meta-Informationen (vgl. Kapitel 3.8.3).

5.3. Bearbeiten von Datensätzen

Für die Bearbeitung von Datensätzen einer Datenbank können grundsätzlich zwei Vorgehensweisen unterschieden werden:

- einzelne Datensätze werden interaktiv aus der Datenbank im *Bearbeitungs-Modus* (vgl. unterschiedliche Extraktionsmodi, Kapitel 3.8.4) geladen, mit den Funktionen des Werkzeugs Gismo bearbeitet und abschließend durch den Anwender in die Datenbank zurückgeschrieben
- über einen Metadaten-Layer wird eine Bearbeitungsfunktionalität (z.B. Datenpunkte mit Maskierung löschen) auf die darin zusammengestellten Datensätze ausgeführt. Im Rahmen dieser Bearbeitungs-Operation werden die Daten automatisch extrahiert, modifiziert und zurückgeschrieben, so dass hier keine weitere Anwenderinteraktion erforderlich ist

Bezüglich der Bearbeitung von Einzeldatensätzen existieren unterschiedliche Wege, um Datensätze aus der Datenbank für die Bearbeitung zu extrahieren:

Auswahl eines Datensatzes aus einer grafischen Selektionskomponente: innerhalb der Selektionskomponente werden die Datensätze in einer Baumstruktur angeordnet (Abbildung 22), wobei der inhaltliche Aufbau der Baumstruktur durch den Anwender nach aufgabenspezifischen Gesichtspunkten modifiziert werden kann. Die Icons der Datensätze in der grafischen Darstellung geben darüber hinaus Informationen zur Struktur der Daten (Punktmenge, TIN, Raster, etc.).

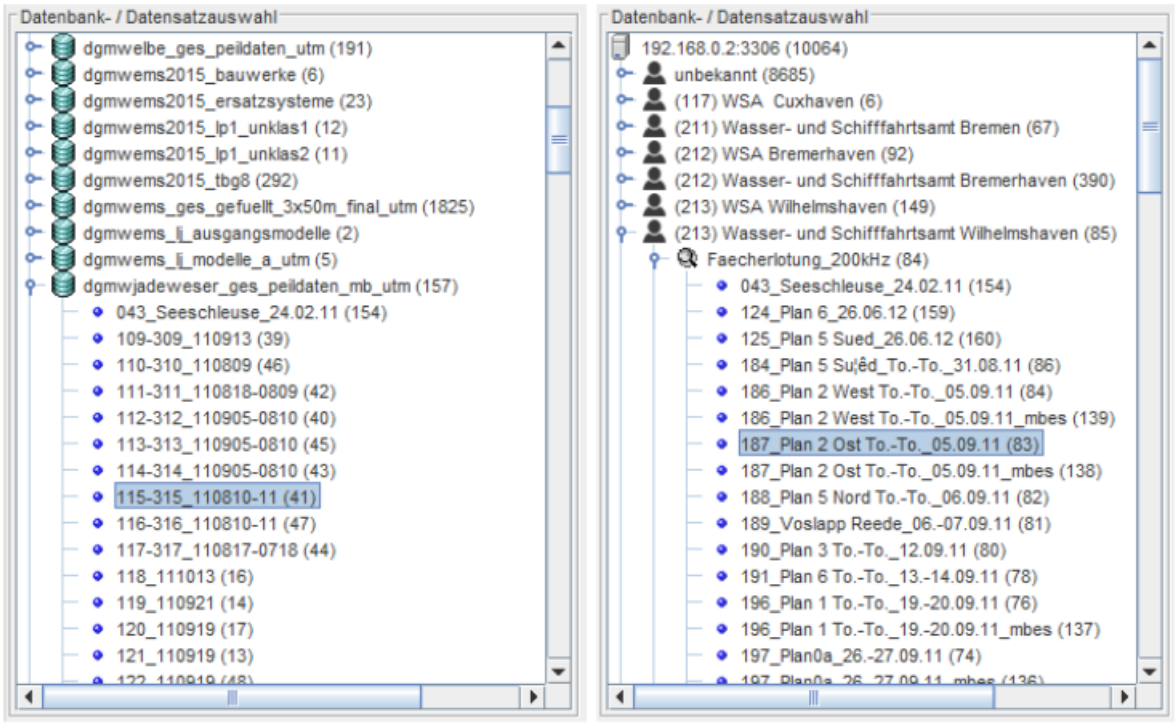


Abbildung 22: Selektionskomponente für Datensätze eines Datenbankservers in unterschiedlicher Organisationssichten, links: Datenbankserver-Datenbank-Datensatz, rechts: Datenbankserver-Datenerheber-Datenart

Mausgesteuerte Selektion eines Datensatzes aus einem Metadaten-Layer: die Selektion eines Datensatzes für die Bearbeitung kann interaktiv auf der Zeichenfläche des Werkzeugs Gismo in der räumlichen Darstellung der Datensätze eines Metadaten-Layers vorgenommen werden.

Extraktion von Datensätze aus speziellen Sichten/Darstellungen: der Datendurchgriff ist in weitere Visualisierungskomponenten integriert. Beispielsweise werden die Meta-Informationen beim Generieren einer bathymetrischen Zeitreihe mitgeführt, so dass aus dieser speziellen Darstellungsform, das Laden eines Datensatz ausgeführt werden kann (nachfolgende Abbildung).

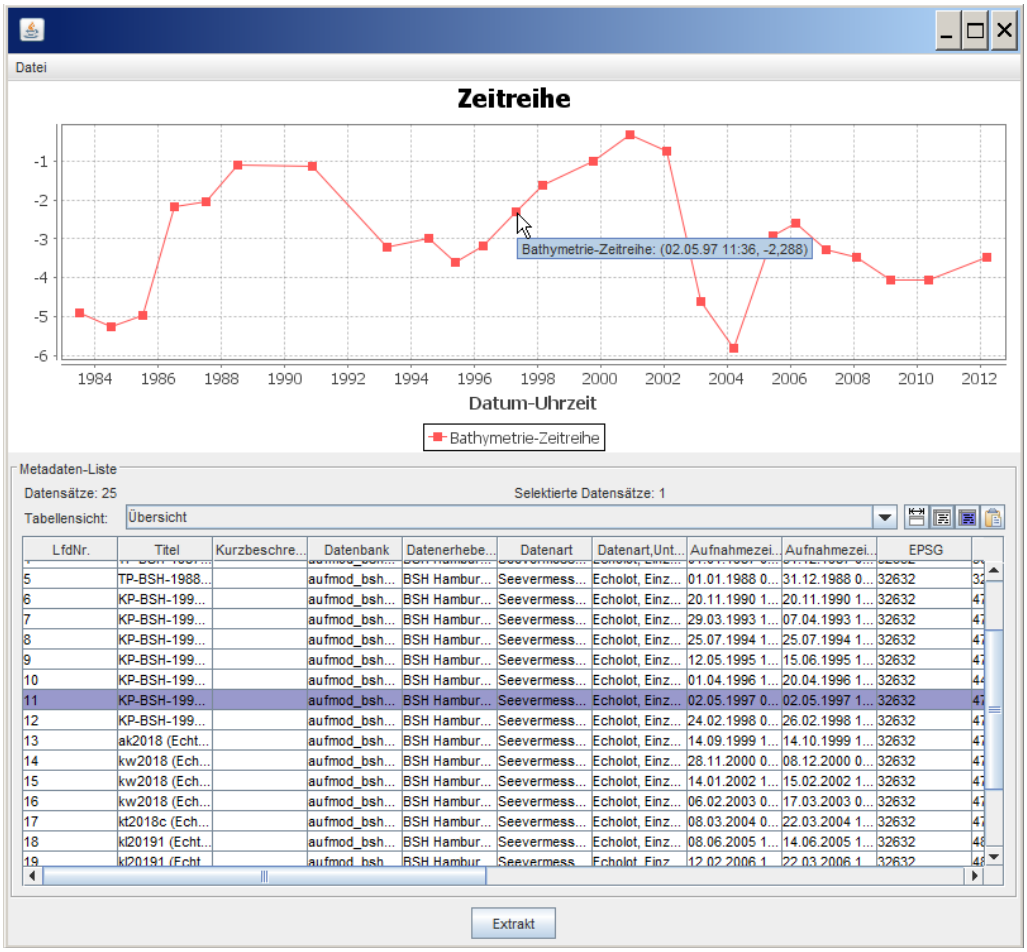


Abbildung 23: bathymetrische Zeitreihe mit Möglichkeit der direkten Extraktion selektierter Datensätze

Die Unterstützung unterschiedlicher Wege der Extraktion von Daten aus der Datenbank ist eine grundlegende Konzeption der Datenbankschnittstelle von Gismo und soll eine effiziente Navigation über umfangreiche Datenbestände zum Zwecke der Bearbeitung und Analyse erlauben.

Wurde ein Datensatz im Einzelbearbeitungsmodus aus der Datenbank extrahiert und modifiziert, so werden die Änderungen nicht automatisch in die Datenbank zurückgeschrieben. Die Aktualisierung in der Datenbank muss explizit durch den Anwender erfolgen. Der Status eines Datensatzes kann dem Tooltip der Layerliste entnommen werden (Abbildung 24), hierbei gibt auch das Layer-Icon Auskunft über durchgeführte Modifikation (Highlighting in magenta zur Anzeige von Modifikationen).

Für das Zurückschreiben in die Datenbank wird die gleiche Funktionalität der grafischen Benutzeroberfläche von Gismo wie beim Einzel-Import eines Datensatzes ausgeführt. Anhand der mitgeführten Verbindungsdaten des Datensatzes (Datenbankserver, Datenbank, DB-ID) wird per default die Option *Aktualisierung des Datensatzes in der Datenbank* als Voreinstellung gewählt.

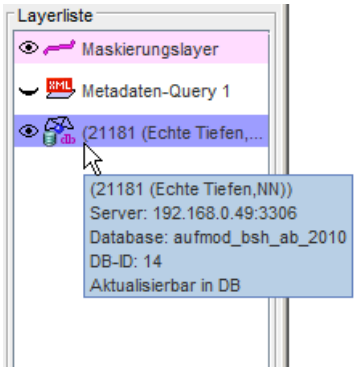


Abbildung 24: Status-Information eines Datensatzes

5.4. Editieren von Metadaten

Das Editieren von Metadaten wird mit dem Gismo-eigenen Metadaten-Editor als grafische Eingabekomponente ermöglicht. Der Metadaten-Editor erlaubt

- die Bearbeitung der Metadaten eines einzelnen Datensatzes
- die Bearbeitung der Metadaten eines Metadaten-Layers im Bulk-Editing-Modus

Der Aufbau des Metadaten-Editors (Abbildung 25) folgt diesem flexiblen Ansatz unterschiedlicher Bearbeitungsszenarien, so dass folgende Arbeitsbereiche im Editor unterschieden werden können:

Metadaten-Liste: tabellarische Präsentation der Metadaten im oberen Fensterabschnitt. Die Liste enthält im Fall der Einzelbearbeitung eines Datensatzes genau einen Eintrag, im Bearbeitungsszenario auf Basis eines Metadaten-Layers, die Anzahl der dort zusammengestellten Datensätze. Die Spaltenkonfiguration der tabellarische Ansicht lässt sich nach nutzerspezifischen Anforderungen konfigurieren und über eine einfache Kopierfunktionalität ins betriebssystem-eigene Clipboard zur Weiterverarbeitung in andere Programmsystemen übertragen.

Innerhalb der Tabellenansicht kann eine Einfach- oder Mehrfach-Selektion der Tabelleneinträge durchgeführt werden. Bei Mehrfach-Selektion wird der Bulk-Editing-Modus des Editors aktiv.

Metadaten-Editor-Bereich: im unteren Fensterabschnitt befinden sich die Eingabefelder für die einzelnen Metadaten-Elemente. Diese sind nach inhaltlichen Gesichtspunkten in unterschiedliche Reiter strukturiert.

Die einzelnen Eingabefelder der Metadaten-Elemente werden auf Basis der selektierten Metadaten der Tabellenansicht mit Werten befüllt. Im Fall des Bulk-Editing-Modus werden Felder mit unterschiedlichen Inhaltswerten mit dem Eintrag ** *Verschiedene Werte* ** versehen. Beim Editieren werden die Nutzereingaben in sämtliche, selektierte Metadaten übernommen. Modifizierte Metadaten werden fortan in der Tabellensicht farblich gekennzeichnet (grün hinterlegte Zeile).

Metadaten-Aktions-Bereich: aus dem Metadaten-Editor können unterschiedliche Bearbeitungsaktionen über die Schaltflächen im seitlichen Bereich der Tabellensicht ausgeführt werden (z.B. Löschen selektierter Datensätze).

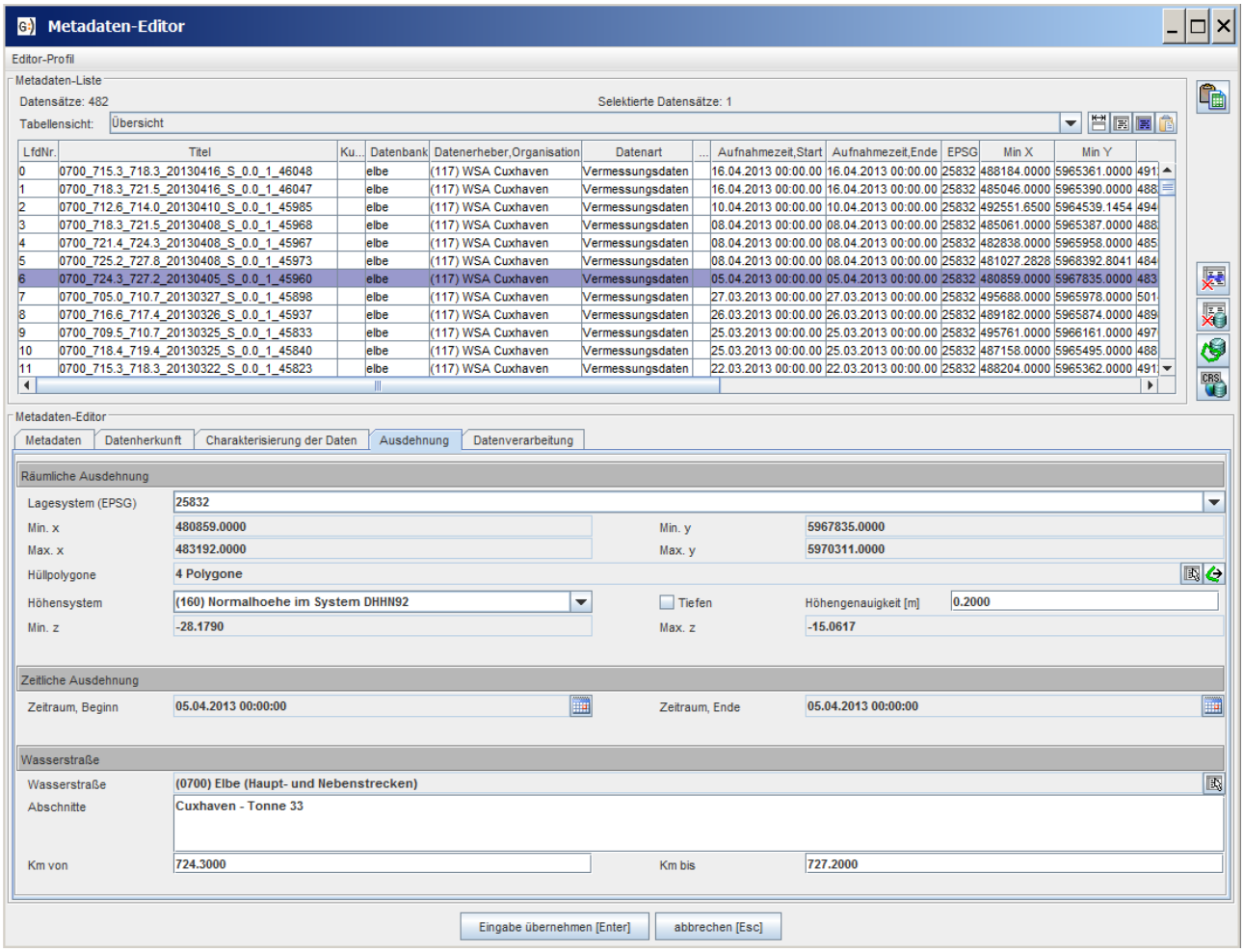


Abbildung 25: Metadaten-Editor

Nach Abschluss des Editieren eines oder mehrerer Datensätze wird der Anwender aufgefordert die geänderten Daten in die Datenbank zurückzuschreiben. Im Zuge der Schreiboperation werden stets nur die Metadaten in der Datenbank aktualisiert, eventuell vorhandene Modifikationen der Geometriedaten bei einer Einzeldatensatzbearbeitung werden nicht gespeichert.

In der Konzeption und Realisierung des Metadaten-Editors wurden weiterhin folgende Teilaspekte verfolgt:

Editor-Profile: der flexiblen und dynamischen Struktur der Metadaten folgend, erlaubt der Metadaten-Editor unterschiedliche Profilansichten, welche die Eingabemasken der Metadaten-Elemente nach unterschiedlichen Anforderungen ausgestalten. Derzeit enthält der Metadaten-Editor die Profile *Basis-Editor* und *SeevermessungsDB-Editor*, welche unterschiedliche Sichten auf den identische Metadateninhalt erlauben und dynamisch zur Laufzeit umgeschaltet werden können. Der Aufbau des Metadaten-Editors ist „baukastenartig“ realisiert, so dass leicht weitere Profilansichten ergänzt werden könnten.

Listenauswahl in den Eingabefeldern: die Listen der Eingabefelder für Metadateneinträge werden zum Teil mit vordefinierten Listeneinträgen (z.B. Dienstellenverzeichnis der WSV) befüllt. Diese sind derzeit statisch verdrahtet. Die Listen werden jedoch um bereits in den Datenbanken verwendete Werten ergänzt, so dass die Logik beim Aufbau der Listen nach folgendem Schema erfolgt:

- Übernahme der vordefinierten Listenwerte
- Übernahme von verwendeten Werten, die mit einer Selektions-Operation aus den Metadaten aller verbundenen Datenbanken ermittelt werden konnten (Distinct-Operation)

Die Wiederverwendung/Synchronisierung von Metadateneinträgen wird mit diesem Mechanismus unterstützt.

5.5. Suche auf dem Datenbestand

Die Suche auf dem Datenbestand wird mit der Recherche-Funktionalität der grafischen Benutzeroberfläche (Abbildung 26) durchgeführt. In der Recherche-Komponente erfolgt

- die Auswahl der Datenbankserver / Datenbanken / Datensätze für die Suchanfrage
- die Festlegung der Filterkriterien zur
  - räumlichen Filterung (Zoombereich, Maskierungsbereich)
  - zeitlichen Filterung durch Vorgabe eines Zeitfensters
  - Freitext-Filterung (entweder auf allen Metadaten-Elementen oder auf einer Auswahl)
  - Filterung nach Freigabestaus (wenn verwendet)
- die Ausführung der Suchanfrage auf Basis der Filtereinstellungen
- die Übernahme des Suchergebnisses

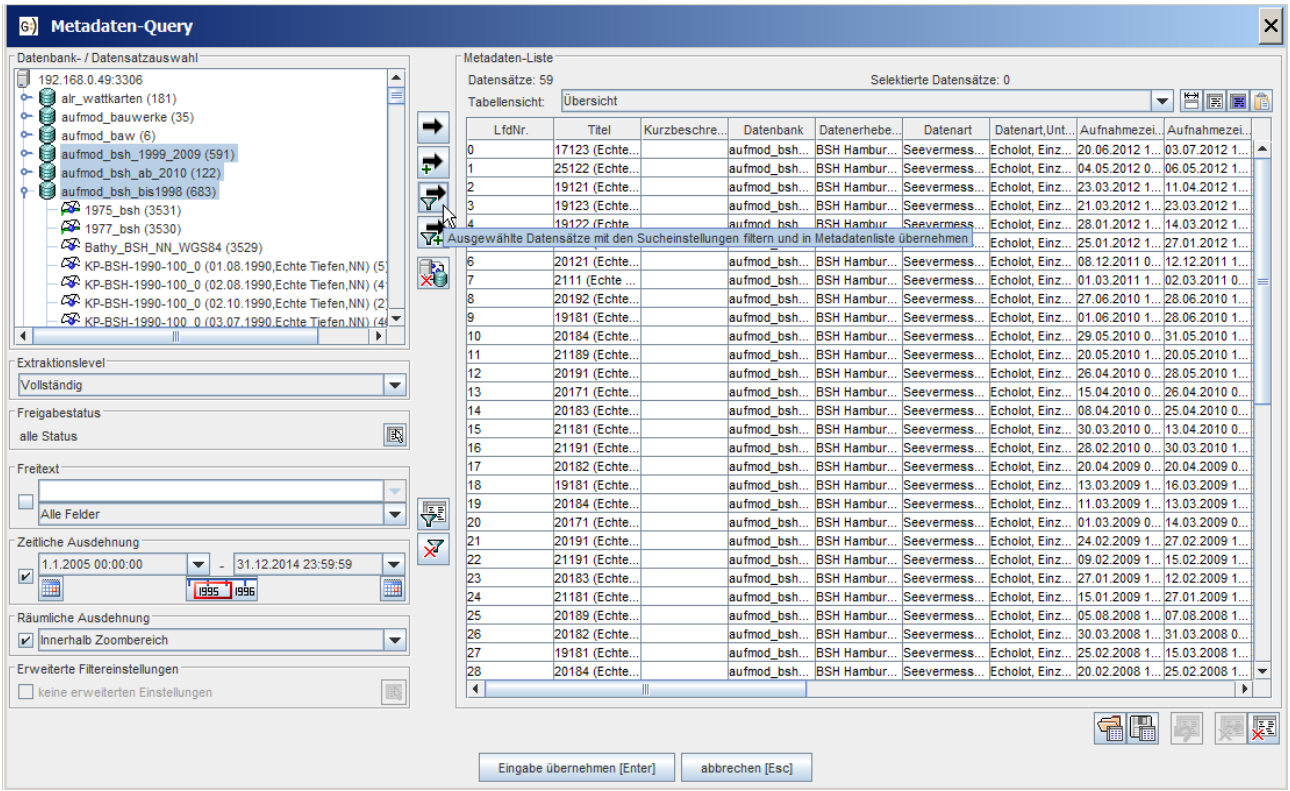


Abbildung 26: Recherche-Komponente zur Durchführung einer Suchanfrage

Ergebnis der Recherche-Anfrage ist ein Metadaten-Layer. Auf Grundlage des Metadaten-Layers wird die Suche auf dem Datenbestand durch zahlreiche Visualisierungsmöglichkeiten (Einfärben nach Aufnahmezeit, Datenerheber, Datenart, etc.), sowie durch Selektionsmechanismen (interaktive Auswahl einzelner Metadaten der Such-Anfrage) weitergehend unterstützt.

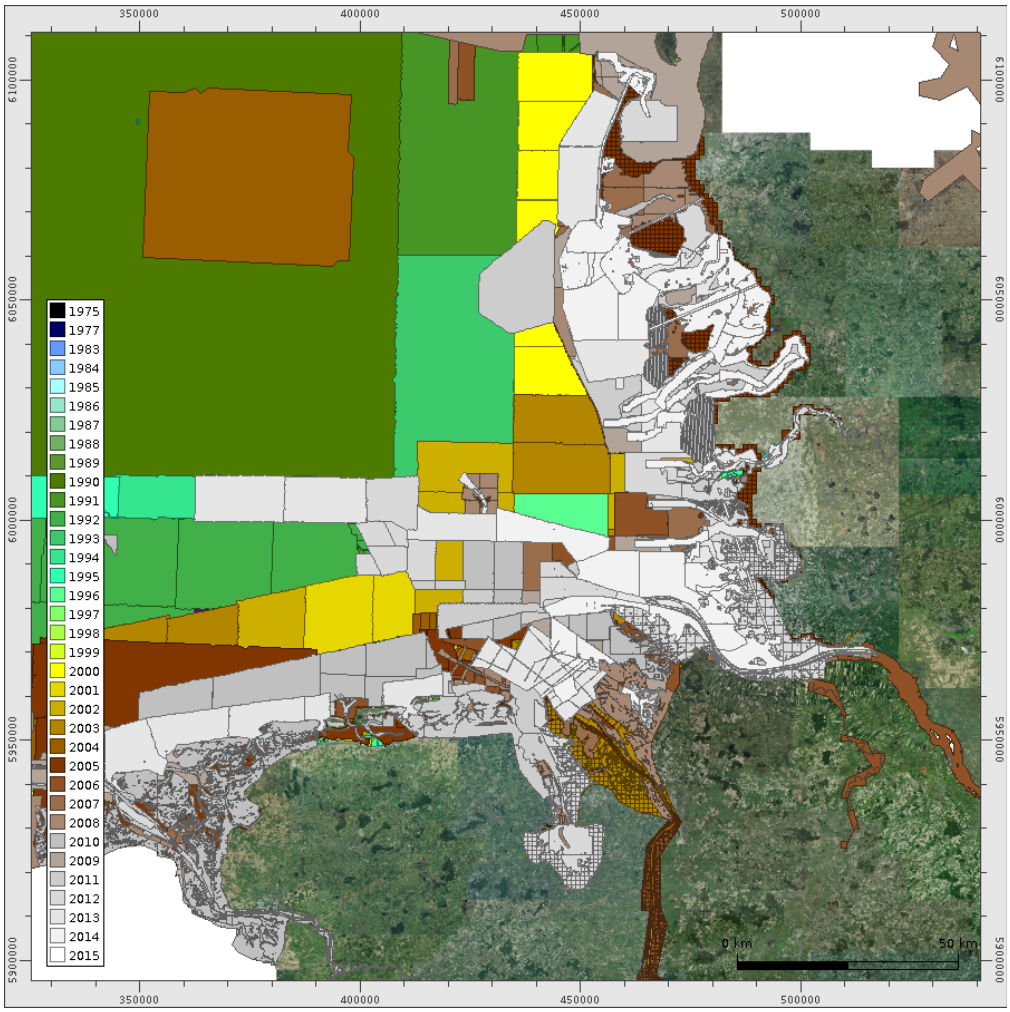


Abbildung 27: grafische Präsentation der Datensätze einer Suchanfrage nach Aufnahmejahr

5.6. Datensicherung

Für die Datensicherung des Datenbestandes stehen die datenbankseitigen Sicherungsfunktionen und -mechanismen zur Verfügung. Diesbezüglich sind zu nennen:

- SQL-Dumps mit den in den Datenbanksystemen integrierten Werkzeugen `mysqldump`, `pg_dump`, etc. (auch mit automatischer Kompression)
- Sicherungsmethoden auf Dateisystem-Ebene durch Archivierung der Datenbankfiles. Bei dieser Sicherungsoption sind jedoch zwingend produkt-spezifische Eigenschaften und Anforderungen zu beachten (Server shut down als Voraussetzung, etc.)
- inkrementelle Sicherungsmechanismen

Gemeinsam ist diesen Methoden zur Datensicherung, dass mit ihnen Datenbanken nur als komplette Einheiten gesichert und wiederhergestellt werden können. Eine Teilsicherung bzw. auch

die Wiederherstellung einzelner Datensätze innerhalb einer Datenbank ist über die genannten Mechanismen nicht möglich.

Alternativ kann eine Datensicherung über den Datenexport in ein Dateiformat erfolgen, welches die Informationen der Datensätze einer Datenbank ohne Informationsverlust speichert. Diese Anforderung ist in Gismo durch das Format

- UGRID-NetCDF

gegeben. Über den Batch-Import (Kapitel 5.2) kann dann eine komplette Datenbank aus dem Dateisystem reimportiert werden. Darüber hinaus besteht beim Dateiexport die Möglichkeit, einzelne Files aus der Dateisicherung wiederum in den Datenbestand über einen Einzel-Import zu übernehmen.

5.7. Datenexport

Über den Datenexport wird die Ausgabe des Datenbestandes bzw. von Teilen des Datenbestandes in Dateiformate/Metadatenformate ermöglicht. Hierbei werden folgende Ausgabestrategien verfolgt:

- Export in vordefinierte Ausgabeformate
  - WSV3D-Datenarchiv-Export
  - UGRID-NetCDF (informationsverlustfreier Export)
  - etc.
- konfigurierbarer Export durch nutzerdefinierte Auswahl
  - der Dateiformate für unterschiedliche Datenarten (TIN, Raster, etc.)
  - des Metadatenformats
  - der Organisationsform (Ausgabeverzeichnisstruktur, Dateibenennung, etc.)

Für die vordefinierten Formate wird ein möglicher Re-Import durch die Batch-Import-Funktionalität sichergestellt. Die konfigurierbare Ausgabe ist demgegenüber als reiner Export zu sehen, für den nicht zwangsläufig eine Importmöglichkeit über die Batch-Methodik besteht.

5.8. Kopieren von Daten

5.8.1. Kopieren einer Zusammenstellung von Datensätzen

Das Kopieren einer Zusammenstellung von Datensätzen erlaubt die Übertragung von Datensätzen

- zwischen unterschiedlichen Datenbankservern
- zwischen unterschiedlichen Datenbanken

Hierbei sind bei der Zusammenstellung der Datensätze sowohl in der Anzahl als auch in ihrer Herkunft keine Limitierungen gesetzt. Das Kopierziel bei dieser Operation ist jedoch stets eine vom Anwender definierte, gemeinsame Datenbank. Als einzige Einschränkung prüft die Methodik, dass Datensätze nicht auf die gleiche Datenbank kopiert werden (Quelle und Ziel sind für einzelne Datensätze identisch).

Praktisch wird das Kopieren von Datensätzen über die Erstellung eines Metadaten-Layers organisiert. In diesem Metadaten-Layer können die Daten mit Hilfe einer Recherche-Anfrage an die verbundenen Datenbankserver nach unterschiedlichsten Gesichtspunkten zusammengestellt werden. Das Kopieren selbst wird durch vollständige Extraktion der jeweiligen Datensätze aus der Quell-Datenbank und Re-Import dieser Datensätze auf der Ziel-Datenbank durchgeführt. Über diesen konzeptionellen Ansatz der Kopiermethodik ist die Übertragung von Datensätzen zwischen unterschiedlichen Datenbankprodukten (z.B. von MySQL nach PostgreSQL) problemlos möglich.

In diesem Zusammenhang ist nochmals darauf hinzuweisen, dass im Zuge der Kopier-Operation der Datensatz-Identifikator in Form einer UUID gemäß den Regeln aus Kapitel 3.8.3 für jeden Datensatz unverändert übernommen wird.

Anwendung findet diese Kopierfunktionalität beispielsweise für eine einfache Re-Organisation der Daten, wenn die zugrunde liegende Datenbankstruktur modifiziert werden soll, oder auch für die Erstellung von lokalen Projektdatenbanken.

5.8.2. Spiegeln von Datenbanken auf entfernte Datenbankserver

Die komplette Spiegelung von Datenbanken auf entfernte Datenbankserver ermöglicht das einfache Kopieren von Datensätzen in ihrer vorliegenden Organisationsstruktur. Eine freie Zusammenstellung der zu kopierenden Daten, wie in der Kopierfunktionalität des vorangegangenen Abschnittes, ist in dieser Option nicht vorgesehen.

Die Spiegelung erfolgt nach folgenden Regeln und Arbeitsschritten:

- die zu kopierende Datenbankstruktur darf auf dem Ziel-Server noch nicht vorhanden sein
- der Ziel-Server muss das Anlegen von Datenbanken über die Datenbankschnittstelle unterstützen
- vor dem Kopieren wird die Datenbankstruktur auf dem Ziel-Server erzeugt
- Kopieren aller Datensätze durch vollständige Extraktion der Datensätze aus der Quell-Datenbank und Re-Import in die Ziel-Datenbank

In diesem Zusammenhang ist nochmals darauf hinzuweisen, dass über die Spiegelung der Datensatz-Identifikator in Form einer UUID gemäß den Regeln aus Kapitel 3.8.3 für jeden Datensatz unverändert übernommen wird. Besonders bei der kompletten Spiegelung von Datenbeständen auf einen entfernten Server kann diese Eigenschaft für eine spätere Synchronisationsmethodik sinnvoll verwendet werden.

5.9. Synchronisation von Datenbeständen

Mit der Synchronisation von Datenbeständen wird die Zielstellung verfolgt, Bearbeitungszustände von Datenbeständen auf unterschiedlichen Datenbankservern abzugleichen. Grundlage der Synchronisation ist eine detaillierte Analyse zweier Datenbestände durch Vergleich der Metadaten der einzelnen Datensätze. Für die Vergleichsoperation zweier Datensätze sollen unterschiedliche Meta-Informationen in den Operator einfließen, z.B.

- Datensatz-Identifikator (UUID)
- Titel
- Anzahl der Geometrien

- räumliche und zeitliche Ausdehnung
- etc.

Auf Basis dieser Analyse können daraufhin Synchronisationsmechanismen entworfen werden, welche eine automatisierten Abgleich der Datenbestände zur Folge haben.