M. Widmer, Februar 1997

Ergänzung zum Metadata-Management

(Analyse)

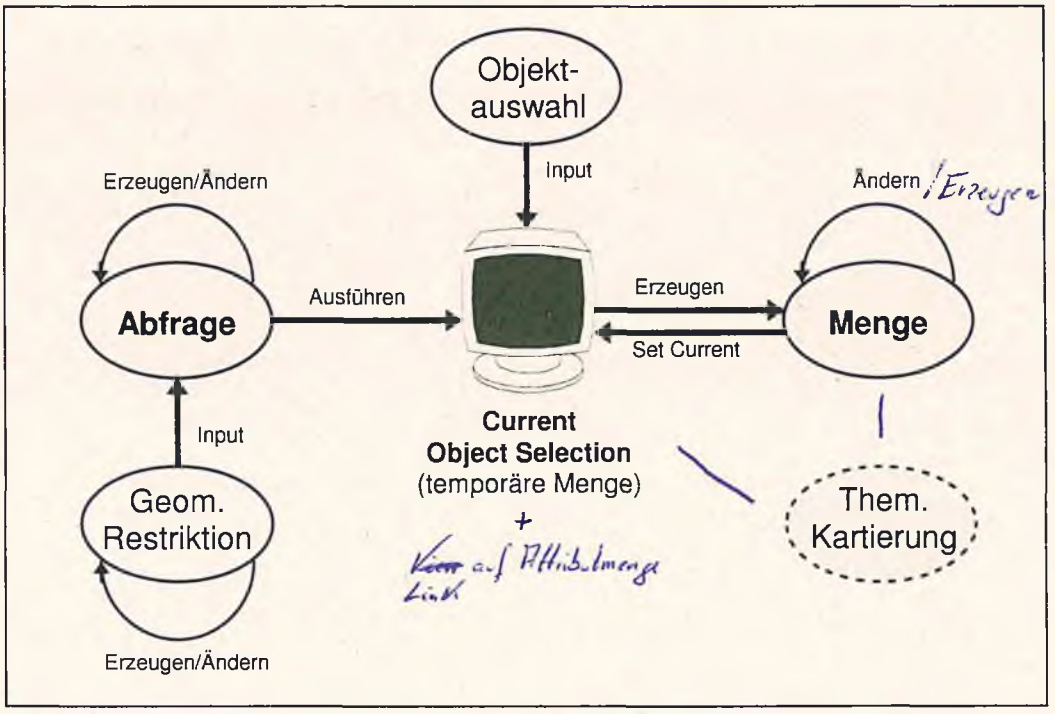
Version 0.1

Inhaltsverzeichnis

1 Ergänzungen zum Metadata-Management

1.1 Beschreibung

Die Metadaten werden um die Verwaltungskomponente von Abfragen und Mengen erweitert. Somit wird die Grundlage geschaffen, um ein durchgehendes Benutzer-Interaktionskonzept für die Analyse abzubilden.



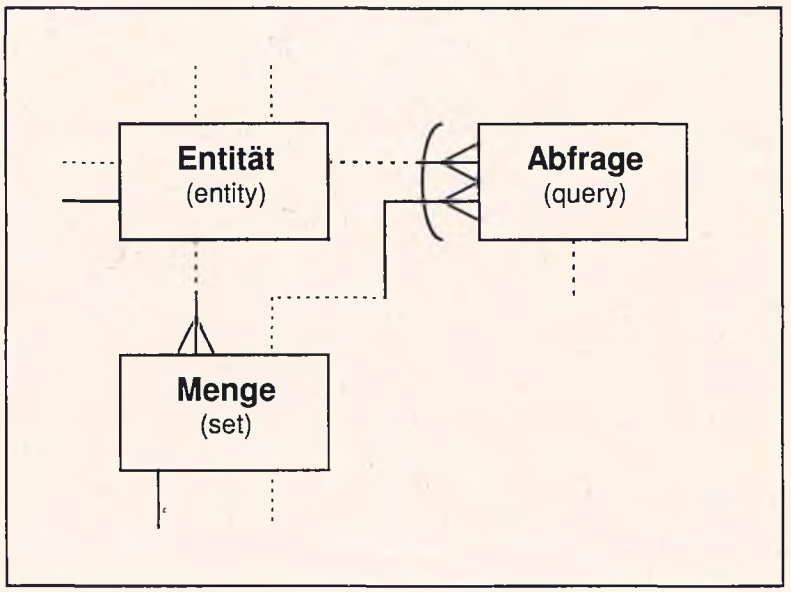
Die wichtigste Benutzerinteraktion im Bereich der Analyse ist primär das Formulieren von Abfragen und die graphische Visualisierung und Auswertung der so erhaltenen Daten. Sekundär werden Mengen zur Speicherung eines bestimmten Datenbestandes verwendet. Aufbauend auf diesen Basiskomponenten ergeben sich zusammen mit Komponente für die geometrische Restriktion und Objektauswahl vier bzw. fünf Interaktionsphasen, wobei die Thematische Kartierung noch nicht genauer spezifiziert ist.

- Spezifikation einer geometrischen Restriktion
- Spezifikation der Datenbank-Abfrage
- Geometrische Objektauswahl
- Spezifikation einer Menge
- (Spezifikation der Thematischen Kartierung)

Diese Interaktionsphasen werden typischerweise in der obigen Reihenfolge durchlaufen, wobei jede Phase auch mehrfach ausgeführt werden kann. Somit kann der Anwender jeden Teilschritt verifizieren und wenn notwendig korrigieren, sofern jeder Teilschritt ein entsprechendes Feedback produziert.
Durch eine geeignete Kombination und Iteration dieser Funktionskomponenten sollte es dem Anwender möglich sein, eine Vielzahl von unterschiedlichsten Aufgaben lösen zu können unter Anwendung derselben Grundfunktionen.

1.2 Das Datenmodell

Eine Menge bzw. eine Abfrage basiert immer auf genau einer Entität. Dabei kann eine Abfrage auf einer Menge basieren und somit explizit auf genau einer Entität.

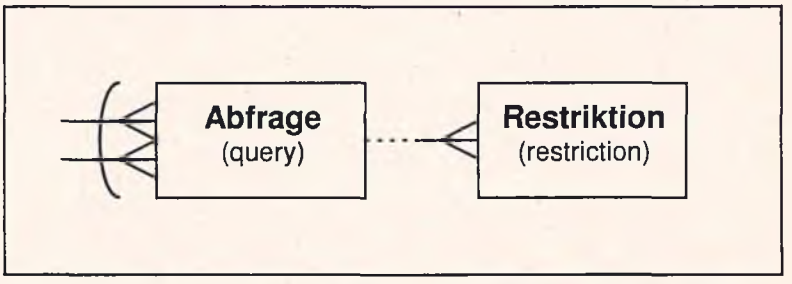


2 Verwaltung von Abfragen

2.1 Beschreibung

Abfragen sind Methoden die in Form von Parametern gespeichert werden. Dabei wird die sachliche Restriktion (Entität, Attribut und Beziehung) von der geometrischen Restriktion getrennt.

2.2 Das Datenmodell



Eine Abfrage basiert auf den Metadaten. Als Quelle kann jeder GeoData-Set, der in den Metadata beschrieben ist, angegeben werden. Ebenso kann eine Menge die Quelle für eine Abfrage sein.

2.2.1 Abfrage

query : Tabelle

Feldname	Felddatentyp	Beschreibung
q_id	Zahl	ID der Abfrage
name	Text	Name der Abfrage
owner	Text	Besitzer der Abfrage
scope	Zahl	Besitzstand (1=Public, 2=Private)
type	Zahl	Type
source_entity	Zahl	ID der zugrundeliegenden Entität
source_dtype	Zahl	Zugrundeliegende Quelle (1=Entität, 2=Menge)
source_did	Zahl	ID der zugrundeliegender Quelle
target_entity	Zahl	ID der Ziel-Entität (wenn Abfrage über logische Beziehung)
relation_id	Zahl	ID der Relation (wenn Abfrage über logische Beziehung)
nr_restriction	Zahl	Anzahl der sachlichen Restriktionen
date	Datum/Zeit	Datum der letzten Änderung
desc	Text	Beschreibung

2.2.2 Restriktion

restriction : Tabelle

Feldname	Felddatentyp	Beschreibung
restriction_id	Zahl	ID der Restriktion
q_id	Zahl	ID der Abfrage
log_operation	Zahl	Logische Operation (1= '=', 2= '!=', 3= '<=', 4= '>=', 5= '<', 6= '>', ...)
arith_operation	Zahl	Arithmetische Operation (1= '+', 2= '-', 3= '*', 4= '/', ...)
a_id	Zahl	ID des zugrundeliegenden Attributs
val_type	Zahl	Werttype (1=Inhalt des Attributs, 2=Attribut selbst)
attr_val	Text	Attributwert

3 Verwaltung von Mengen

3.1 Beschreibung

Mengen sind Analyse-Resultate, die permanent gespeichert werden. Die logische Richtigkeit ist genau beim Zeitpunkt der Erstellung gewährleistet.
Jede Menge entspricht einer Datenbank-Tabelle, wobei für die Verwaltung gemeinsame Tabellen zur Verfügung stehen. Es werden nur homogene Mengen unterstützt. D.h. alle Objekte müssen von derselben Entität sein.

Es gibt vier verschiedene Typen von Mengen:

- Objektmenge
- Sachdatenmenge
- Beziehungsmenge
- Infopaket

3.1.1 Objektmenge

Eine Objektmenge ist eine einspaltige Tabelle mit der ID der Objekte.

z.B. Tabelle *set_12* (entspricht Menge 12)

Objekt-ID
#12345678
#12345679

3.1.2 Sachdatenmenge

Eine Sachdatenmenge entspricht einer Objektmenge mit zusätzlichen Wertattributen.

z.B. Tabelle *set_13* (entspricht Menge 13)

Objekt-ID	***W_COL_1***	***W_COL_2***	***...***
#12345678	123.35	Peter Muster	
#12345679	11.30	Hans Meier	

3.1.3 Beziehungsmenge

Eine Beziehungsmenge entspricht einer Menge, in welcher die Beziehung zwischen zwei Objekten abgespeichert wird. Es werden nur 1:n-Beziehungen unterstützt.
Die Beziehungsinformation kann entweder aus dem Datenmodell oder aus einer Verschneidung aufgebaut werden.

Weiter muss man sich überlegen, ob es sinnvoll währe Beziehungsattribute zu unterstützen (z.B. Pendlerströme, etc.).

z.B. Tabelle *set_14* (entspricht Menge 14)

Objekt-ID	*Objekt-ID-2*	*ev. Attribut*
#12345678	#12345677	123
#12345679	#12345677	34
(unique)	(not unique)	

3.1.4 Infopaket

Ein Infopaket entspricht einer multidimensionalen Tabelle (siehe Pivot-Tabelle in Excel). Es wird zwischen Wertattribut und Gliederungsmerkmal unterschieden wird.
Diese Funktionalität wird im EU-Projekt EUROSCOPE gefordert.

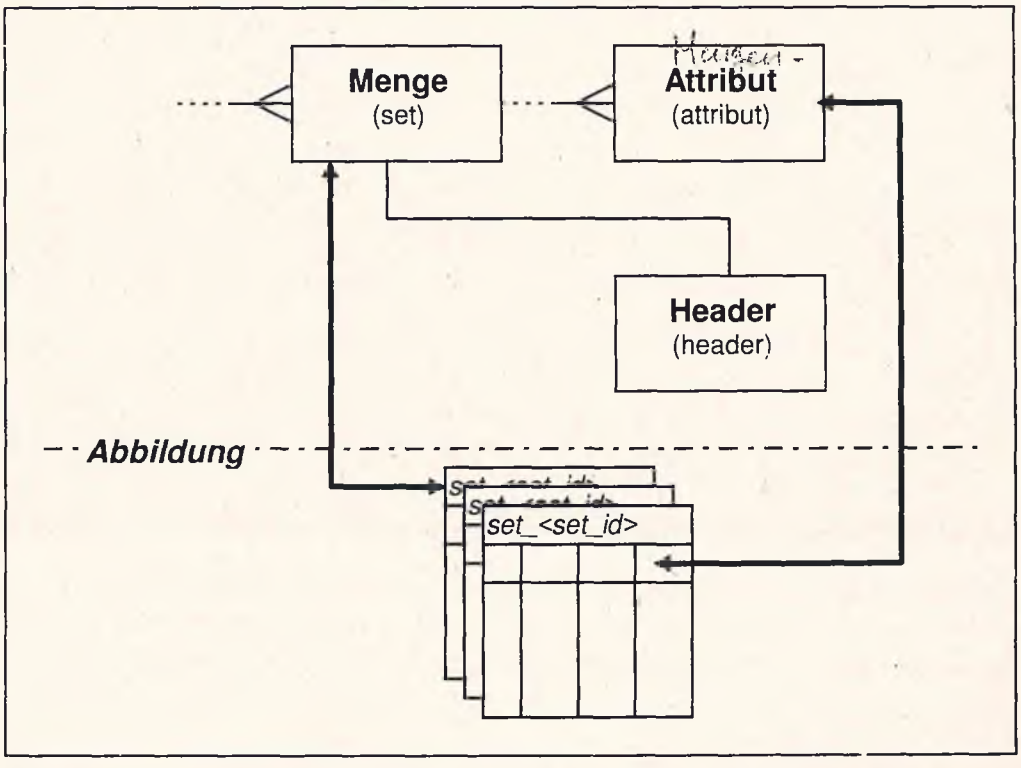
3.2 Das Datenmodell

Die Mengen werden in drei verschieden Tabellen verwaltet:

- *Menge*
  Verwaltungstabelle für die Mengen und da[illegible] die zugehörigen Tabellen.
  Diese Tabellen heissen jeweils set_<s_id>.

- *Attribut*
  Verwaltungstabelle für die Sachdaten einer Menge

- *Header*
  Verwaltungstabelle für die Köpfe der Listenausgabe einer Menge (?!).



3.2.1 Menge

set : Tabelle

Feldname	Felddatentyp	Beschreibung
s_id	Zahl	ID der Menge (Die Daten stehen in der set_<s_id>-Tabelle
name	Text	Name der Menge
owner	Text	Besitzer der Menge
scope	Zahl	Besitzstand (1=Public, 2=Private)
type	Zahl	Type (1=Objektmenge, 2=Sachdatenmenge, 3=Beziehungsmenge, 4=Infopaket)
entity	Zahl	ID der zugrundeliegenden Entität
count	Zahl	Anzahl Einträge in der set_<s_id>-Tabelle
status	Zahl	Status (1=Intitialisiert, 2=...)
date	Datum/Zeit	Datum der letzten Änderung
desc	Text	Beschreibung
entity_2	Zahl	Wenn Type=3, dann ID der zeiten Entität. Ansonsten -1.
bez_id	Zahl	Wenn Type=3 und Beziehung ist über logische Bez. aufgebaut, dann ID der Bez.

3.2.2 Attribut

attribut : Tabelle

Feldname	Felddatentyp	Beschreibung
attr_id	Zahl	ID des Attributs (Der Spaltenname ist W_COL_<attr_id - 1>)
s_id	Zahl	ID der Menge (Die Daten stehen in der set_<s_id>-Tabelle
type	Zahl	Type (1=Vorspalte, 2=Wertattribut, 3=Gliederungsmerkmal)
ref_name	Text	Ursprungsname des Attributs (SDE-Spaltenname, ...)
format	Text	Ausgabeformat (z.B. N14, A12, ...)
name	Text	Name des Attributs
data_type	Text	Datentyp (C=char, D=date, F=float, I=Integer)
data_length	Zahl	Max. Länge des Attributs (wenn Datentyp = C)
data_precision	Zahl	Anzahl Dezimalstellen (wenn Datatyp = F oder I)
data_scale	Zahl	Anzahl Nachkommastellen (wenn Datatyp = F)

3.2.3 Header

header : Tabelle

Feldname	Felddatentyp	Beschreibung
s_id	Zahl	ID der Menge (Die Daten stehen in der set_<s_id>-Tabelle
title	Text	Titel der Liste
header	Zahl	Anzahl der Sachdatenattribite inkl. Vorspalte

(Wird nur für den Mengentransfer benötigt und sollte anders modelliert werden)

3.3 Mengentransfer GIS <-> SIS

Alle Mengentypen können vom GIS-System nach SIS exportiert bzw. importiert werden. Für die Sprachumsetztung der beiden Systeme werden die Key-Tabellen benötigt (siehe Dokument *Fortschreibungsschnittstelle*).

Die Schnittstelle zu SIS verhält sich wie eine offene File-Schnittstelle mit einfachen Lese- und Schreibzugriffen.

(siehe Dokument von der Software AG: *Schnittstelle GRADIS-SIS*)

4 API für Mengenfunktionalität

Als Beispiel für den Funktionsumfang im Bereich der Mengen ist nachfolgend das bestehende API aufgelistet.

```
Interface functions for accessing external (DB) sets:
-----------------------------------------------------

If not stated otherwise, all functions return 0 for success or -1 in case of
error. If a function returns an error, sis_err_put() has been called with an
apropriate error number and text.


• sis_set_current_user()
Has to be called before using any other sis_set_xxx() functions. Tell the set-
module the name of the (login-)user as well as a DBA-flag. This information will
be used for checking access rights and setting owner of new sets.
(If the DBA-flag is set, the current user may access, delete etc. any set.)

• sis_set_get_curr_usr()
(used by sis_sss_if.c)

• sis_set_init_sss()
used by sis_sss_if.c)

• sis_set_check_access()
Returns access rights of current user for given set.
Return values are
   SIS_WRITE_ACCES,
   SIS_READ_ACCESS,
   SIS_NO_ACCESS
   or -1 in case of error.

• sis_set_create_new()
Initialize creation of a new set.
Has to be followed by sis_set_insert() /sis_set_copy()/ sis_set_merge() and
sis_set_commit_new() or sis_set_cancel().

• sis_set_create_temp()
create a temporary set.
see description of sis_set_create_new() also.

• sis_set_lock()
Lock a set before editing.
Has to be followed by sis_set_update() or sis_set_cancel().

• sis_set_cancel()
Cancel current operation.
May be called after calling sis_set_create_new() or sis_set_lock() instead of
sis_set_commit_new(), sis_set_update(), sis_set_copy(), or sis_set_merge().

• sis_set_reset()
Cancel all current current operations.
Also clears the local 'cache' containing the last few sets that have been
processed.

• sis_set_commit_new()
Completes the creation of a new set after having called sis_set_create_new() and
sis_set_create_new(), sis_set_copy() or sis_set_merge().
Before committing a new set, at least the following components have to be set:
   SET_I_PROJECT,
   SET_I_ENTITY.
All other components will have their default values. If operation fails,
sis_set_cancel() has to be called.
```

- **sis_set_insert()**

Insert Objects into a set. Set must be created with sis_set_create_new(). Function has to submit a call-back function for obtaining the object IDs. If operation fails, sis_set_cancel() has to be called.

- **sis_set_update()**

Update a set after having edited with sis_set_put_xxx(). Set must have been locked with sis_set_lock().

- **sis_set_delete()**

Delete a set.

- **sis_set_copy()**

Copy a set. New set has to be reserved with sis_set_create_new(). After copying, New set has to be committed with sis_set_commit_new(). If operation fails, sis_set_cancel(new) has to be called.

- **sis_set_get_elt()**

Function must(!) be called until it returns Null ptr. Get first/next element (Object-ID) of a set. Flag-argument != 0 means first, else next.
Null ptr is returned in case of error or end of table. In case on an error, Flag-argument is set to -1, else 0.
Where-clause:
  NULL-ptr.
  or string of the format "where ..." (will be applied to set-table).

- **sis_set_get_es()**

similar to sis_set_get_elt() but returns a pointer to a structure of type Es_data.

- **sis_set_get_id()**

Get set-ID of a set by name and owner.

- **sis_set_get_int()**

Get integer component of a set (by ID).
Allowed components are:
  SET_I_ID,
  SET_I_SCOPE,
  SET_I_TYPE,
  SET_I_PROJECT,
  SET_I_ENTITY,
  SET_I_COUNT,
  SET_I_STATUS,
  SET_I_NR_ATTRS.

- **sis_set_get_str()**

Get string component of a set (by ID).
Allowed components are:
  SET_C_NAME,
  SET_C_OWNER,
  SET_C_DATE,
  SET_C_DESCRIPTION.

- **sis_set_put_int()**

Put integer component of a set (by ID). Set has to be locked with sis_set_lock().
Allowed components are:
  SET_I_SCOPE,
  SET_I_PROJECT,
  SET_I_ENTITY.

- **sis_set_put_str()**

Put string component of a set (by ID). Set has to be locked with sis_set_lock().
Allowed components are:
  SET_C_NAME,
  SET_C_DESCRIPTION.

- **sis_set_get_str_by_key()**

Auxiliary function to obtain a string for a given key value (eg. set-type).
Allowed components are:
  currently only SET_I_TYPE.

- **sis_set_get_key_by_str()**

Auxiliary function to obtain a key for a given string value (eg. set-type).
Allowed components are:
  currently none

- **sis_set_get_list()**

Get a list of all sets. Returns all sets if current user is dba, else only sets where owner is current user or where scope is PUBLIC.
Filter(s) may be set with sis_set_set_xxx_filter(). Currently filters have no effect !

- **sis_set_set_int_filter()**

Set filter for integer component.
Allowed components are:
  currently none

- **sis_set_set_str_filter()**

Set filter for string component.
Allowed components are:
  SET_C_NAME,
  SET_C_OWNER,
  SET_C_DATE.
  (for SET_C_NAME, simple *-wild-card-stings are allowed)

- **sis_set_reset_filter()**

Reset single or all filter(s).
Allowed components are:
  SET_ALL,
  SET_C_NAME,
  SET_C_OWNER,
  SET_C_DATE.

- **sis_set_type_bez(set_id)**

change type of set to SET_TYP_BEZ and add bez attributes to ES table.

- **sis_set_type_sdt(set_id)**

change type of set to SET_TYP_SDT (used from sis_sss_import)

- **sis_set_status_full()**

- **sis_set_status_ids()**

- **sis_set_status_names()**

(used from sis_sss_import)

- **sis_set_ins_single()**

insert a single row into an ES table.
table may be of type SET_TYP_ENT, SET_TYP_SDT or SET_TYP_BEZ.

- **sis_set_ins_multiple()**

insert multiple rows int an ES table by using a given subquery.

- **sis_set_merge()**

merge two sets depending on the given operator and the given attribute flag. put result into the new and given set. sis_set_merge() returns the number of rows which were inserted into the new set or a negative value which indicates that there was any error. All possible error messages will be written by the underlying db function or by the get set ptr function.

- **sis_set_get_attrs()**

read all attribute names of a set into a single linked list.

- **sis_set_delete_nulls()**

- **sis_set_rename_sdt_attr()**

- **sis_set_atr_get_str()**

- **sis_set_sdt_get_int()**
read integer component from a single sdt attribute.

- **sis_set_sdt_col_min_max_f()**

- **sis_set_sdt_col_min_max_i()**

- **sis_set_sdt_col_min_max_c()**

- **sis_set_sdt_col_min_max_alist()**
read min/max of sdt attributes of type float, int or character.

- **sis_set_if_sdt_unique()**
read unique values of a single sdt attribute.

- **sis_set_aggregate_new()**
aggregate a set of type SET_TYP_BEZ identified by its set_id to a new one of type SET_TYP_ENT identified by its new name. the return value may be negative, which indicates an error. otherwise the positive set id will be returned. in case of a positive return value user must set the scope, the description and commit the new set. in case of a negative return value no further action is necessary.

- **sis_set_is_subset()**
function checks if set 1 is a subset of set 2.
the sets are identified by its set id's. function returns 1 if set 1 is a subset, otherwise 0 or ERR_GENERAL. legal values for parameter mode are SIS_ID_ID, SIS_RID_ID, SIS_ID_RID and SIS_RID_RID.