Technisches Handbuch

MengenManager
- Analyse mit ArcView -

ESRI Gesellschaft für
Systemforschung und
Umweltplanung AG

Autor: Markus Widmer
ESRI AG
Beckenhofstrasse 72
CH-8006 Zürich

Datum: 19.09.16 15:27

Datei: Techn. Handbuch V1.0.doc

Version: 1.0.1

Inhaltsverzeichnis

1 Einleitung

Der MengenManager ist ein Analyse-Tool, um beliebige Sachdaten mit Geo-Objekten zu verknüpfen und diese gemeinsam zu nutzen. Dabei stehen nicht die Geo-Objekte im Mittelpunkt, sondern sach- und raumbezogenen Sichten auf die Geo-Daten in Form von Mengen. In Mengen können Geo-Objekte, Sachdaten und Beziehungen abgelegt werden.

Der MengenManager verfügt über eine metadatengesteuerte graphische Benutzeroberfläche und ermöglicht eine einheitliche Dokumentation von Mengen in einer relationalen Datenbank (RDBMS). Entscheidend dabei ist, dass die Flut von Sachdaten übersichtlich zur Verfügung gestellt wird und der Anwender flexibel mit den Mengen arbeiten kann, damit Informationen effizient genutzt und neu erzeugt werden können. Eine Form der Informationsgewinnung ist die beliebige raumbezogene Verdichtung von Daten auf ein höheres räumliches Niveau. Dies kann über feste oder über ad-hoc aufbaubare Beziehungen zwischen verschiedenen Geo-Objekten geschehen.

Die GIS-Komponente des MengenManagers ist ***ArcView***. Mit Standardfunktionen von ***ArcView*** werden Mengen erzeugt, visualisiert und weiterverarbeitet.

2 Das Semantic Data Dictionary

2.1 Datenmodellierung

Beim Design eines Informationssystems nimmt die Datenmodellierung eine Zentrale Stellung ein. Im Verlauf der Datenmodellierung wird im wesentlichen festgelegt, welche Aspekte der Realwelt in welcher Art und Weise im System abgebildet werden.

In der Regel wird folgendermassen vorgegangen:

- *Erstellen eines konzeptionellen, vom Zielsystem unabhängigen Datenmodells*
  Dabei wird in enger Zusammenarbeit mit den Fachanwendern definiert, welche Objekte für das System relevant sind, mit Hilfe welcher Attribute sie beschrieben werden können und welche Beziehungen zwischen den Objekten existieren. Das dabei entstehende Modell wird in einer leicht verständlichen Weise, meist in Form eines Entity-Relationship-Model (ER-Modell), beschrieben.

- *Erstellen eines logischen, systemspezifischen Datenmodells*
  In dieser Phase wird das Modell des vorhergehenden Schritts auf die Konzepte des Zielsystems abgebildet. Es wird z.B. festgelegt, *wie* eine Beziehung zwischen zwei Entitäten im Zielsystem abgebildet wird. Dabei gibt es meistens verschiedene Möglichkeiten diese Abbildung zu realisieren und bedingt sehr gute Kenntnisse des Zielsystems.

- *Erstellen des physischen Datenmodells*
  In dieser Phase wird das Modell physisch implementiert.

2.2 Beschreibung aller Phasen

Das Semantic Data Dictionary (SDD) ist eigens dafür konzipiert worden, um eine detaillierte Beschreibung des konzeptionellen, des logischen und des physischen Datenmodells - inklusive Informationen über die Abbildung der verschiedenen Modelle - abzulegen.

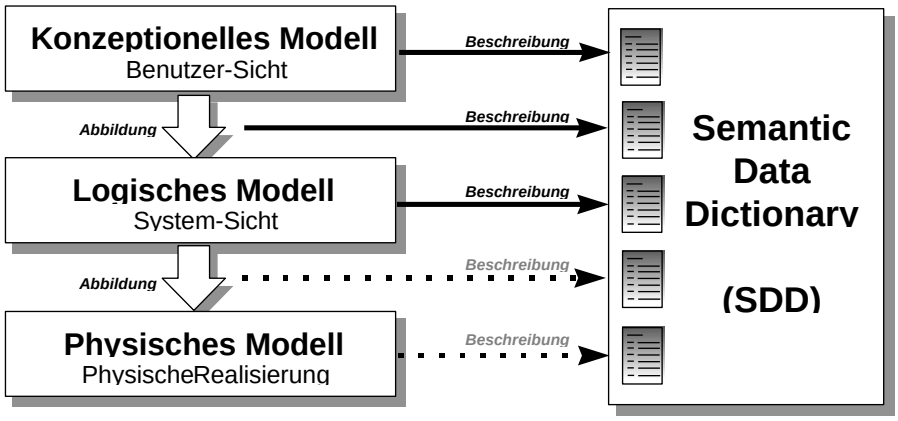


Abb. 1: Abbildung der Modelle im Semantic Data Dictionary

Damit wird es möglich, mit kleinem Aufwand sehr flexible, benutzerorientierte Anwendungen zu entwickeln. Sobald alle Funktionen konsequent auf dem SDD basieren, muss bei einer Änderung des Datenmodells lediglich das SDD nachgeführt werden. Eine Modifikation der Funktionalität ist nicht nötig. Die Funktionalität kann somit

in unterschiedlichen thematischen Zusammenhängen ohne weiteren Programmieraufwand verwendet werden. Zudem kann die Interaktion mit dem Benutzer in der ihm vertrauten Begrifflichkeit gestaltet werden.
Mit dem SDD wird eine wesentliche Voraussetzung für einen künftigen Datenaustausch mit einem Fremdsystem geschaffen.

2.3 Das Datenmodell des Semantic Data Dictionary

Das SDD enthält eine Beschreibung aller im System abgebildeten Klassen von Realweltobjekten sowie alle systemrelevanten Eigenschaften. Im wesentlichen sind das Entitäten, die zugehörenden Attribute und die Beziehungen. Ausserdem wird die Art und Weise der Abbildung dieser Objekte durch das System beschrieben.
Weiter werden die Analyseresultate (Mengen), deren Sachdaten und die arithmetische Berechnungsformel für die Bildung neuer Mengenattribute im Dictionary gespeichert.

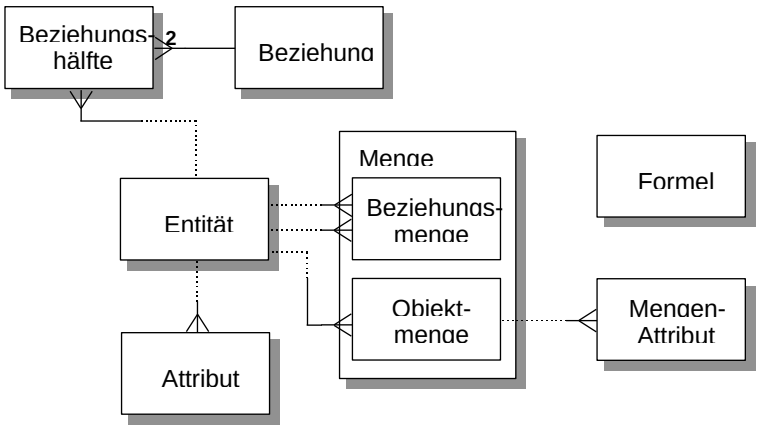


Abb. 2: Das Semantic Data Dictionary - Datenmodell

2.3.1 Entität

Eine Entität ist ein Objekt der realen oder der gedachten Welt. Sie wird definiert durch einen Satz von identischen Attributen und Beziehungen zu anderen Entitäten.

2.3.2 Beziehung

Jede Beziehungen gilt zwischen zwei Entitäten. Dabei können auch mehrere solcher Beziehungen definiert sein. (z.B. zwischen Kante und Knoten können die beiden Beziehungen *'beginnt an'* und *'endet an'* existieren.)

Dabei werden nachfolgende Beziehungstypen unterschieden:

- *Objektreferenz*
  Attribut des referenzierenden Objekts. Enthält den Objektschlüssel (ID) des referenzierten Objekts.

- *Namensreferenz*
  Sprechender Identifikator des referenzierten Objekts. Kann eindeutig aus einem Attribut des referenzierenden Objekts abgeleitet werden.

- *Hierarchischer Schlüssel*
  Sprechender Identifikator des referenzierten Objekts. Kann eindeutig aus einem Attribut des referenzierenden Objekts mittels einem Substring abgeleitet werden.

Für die vorgängig aufgelisteten Beziehung müssen Regeln festgelegt werden, wie der sprechende Indentifikator des referenzierten Objekts eindeutig abgeleitet werden kann.

2.3.3 Beziehungshälfte

Eine Beziehung besteht aus genau zwei Beziehungshälften. Die Beziehung wird einmal aus Sicht der referenzierenden (1. Beziehungshälfte) und aus Sicht der referenzierten Entität (2. Beziehungshälfte) beschrieben.
Dieser Abstraktionsgrad ist eingeführt worden, um später ev. Mehrstufige-Beziehungen unterstützen zu können.

Es werden nachfolgende Kardinalitäten von Beziehungen unterstützt:

- *1:c* (1:1, bedingt)
- *1:1* (1:1, unbedingt)
- *1:cn* (1:n, bedingt)
- *1:n* (1:n, unbedingt)

2.3.4 Attribut

Jede Attributbeschreibung ist genau einer Entität zugeordnet.

Es werden nachfolgende Attributtypen unterstützt:

- *Zahl (Integer-Wert)*
- *Gleitkommazahl (Float-Wert)*
- *Text (String)*

2.3.5 Unterstützung mehrerer Sprachen

Die Benutzernamen von Entitäten, Attributen und Beziehungsworte können in mehreren Sprachen definiert werden.

2.3.6 Menge

Mengen sind Analyseresultate, die permanent im System gespeichert werden. Die logische Richtigkeit ist genau beim Zeitpunkt der Erstellung gewährleistet. Es werden nur homogene Mengen unterstützt, d.h. alle Objekte in der Menge müssen derselben Entität angehören.
Die Instanz einer Menge entspricht einer Datenbank-Tabelle, wobei die Verwaltung vom SDD übernommen wird.

Es gibt zwei Typen von Mengen:

- *Objektmenge*
- *Beziehungsmenge*

2.3.6.1 Objektmenge

Eine Objektmenge ist bildlich eine Tabelle mit beliebig vielen Zeilen und eins bis ca. 250 Spalten, wobei die erste Spalte den Objektschlüssel beinhaltet. Sind mehrere Spalten vorhanden, so sind zu dieser Objektmenge noch zusätzliche Mengenattribute vorhanden.

2.3.6.2 Beziehungsmenge

In diesem Mengentyp werden Beziehungen zwischen zwei Objekten gespeichert. Es werden nur 1:n - Beziehungen unterstützt. Die Beziehungsinformation kann entweder aus dem SDD oder ad-hoc mittels der GIS-Funktionalität ‘Verschneidung’ aufgebaut werden.

2.3.7 Mengenattribut

Mengenattribute sind immer Wertattribute, die genau einer Menge zugeordnet sind.

Es werden nachfolgende Mengenattributtypen unterstützt:

- *Zahl (Integer-Wert)*
- *Gleitkommazahl (Float-Wert)*
- *Text (String)*

2.3.8 Formel

Für die arithmetische Berechnung eines neuen Mengenattributes kann eine beliebige Formel mit Variablen definiert werden. Dabei werden die Variablen mit Mengenattributen einer Menge ersetzt.

3 MengenManager

3.1 Das Benutzer-Interaktionsmodell

Das Benutzer-Interaktionsmodell für die Analyse mit ***ArcView*** beschränkt die Benutzer-Interaktion auf diejenigen Komponenten, die für den Anwender relevant sind, ohne dass auf interne Mechanismen Rücksicht genommen werden muss. Im Bereich der Analyse ist es primär die Objektauswahl in Kombination mit der Sachdatenauswahl und die graphische Visualisierung und Auswertung der so erhaltenen Daten. Ausserdem werden Mengen zur Speicherung eines bestimmten Datenbestandes verwendet. Aufbauend auf diesen Basiskomponenten ergeben sich folgende Interaktionsphasen:

- Objektauswahl
- Sachdatenauswahl
- Spezifikation einer Menge.

Diese Interaktionsphasen werden typischerweise in der obigen Reihenfolge durchlaufen, wobei jede einzelne Phase auch mehrfach ausgeführt werden kann. Wichtig dabei ist, dass alle Teilschritte eine entsprechende Rückmeldung liefern und somit für den Benutzer verifizierbar und wenn notwendig korrigierbar sind.

Durch eine geeignete Kombination und Iteration dieser Komponenten ist es dem Anwender möglich, eine Vielzahl von unterschiedlichsten Aufgaben durch Gebrauch der immer gleichen Grundfunktionen lösen zu können.

Generell orientiert sich das GUI-Konzept am *Object-Action-Model*, bei welchem zuerst eine Objektselektion erfolgt und anschliessend die auf dieses Objekt anzuwendende Aktion ausgewählt wird. Deshalb wird bei der Analyse mit ***ArcView*** die erste Interaktionsphase dazu verwendet, eine ‘aktuelle Objektselektion’ zu erstellen. In der anschliessenden Interaktionsphase wird die Aktion ausgewählt, die auf diese ‘aktuelle Objektselektion’ angewendet werden kann.

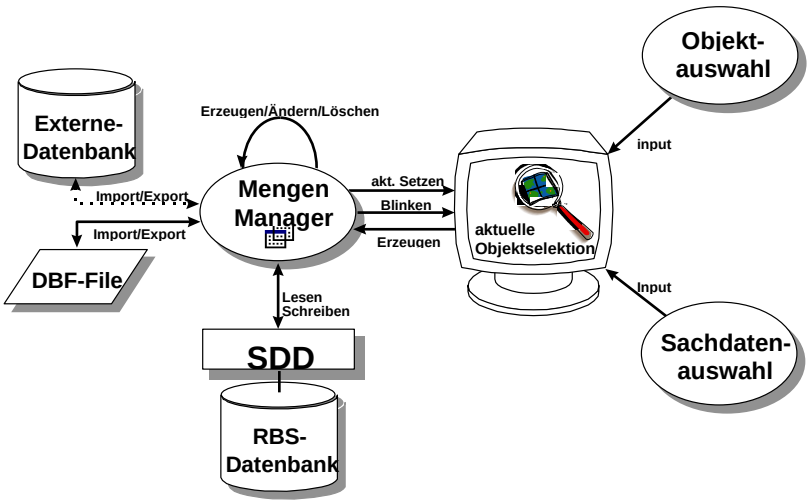


Abb. 3: Das Interaktions-Konzept

3.2 Die graphische Oberfläche

Die Gruppierung der Aktionen für die Spezifikation einer Menge erfolgt in einem modeless ‘Action-Panel’. Die wesentlichen Funktionen betreffen das Erstellen und Speichern einer Menge und Funktionen zum Bearbeiten der Sachdaten einer Menge.

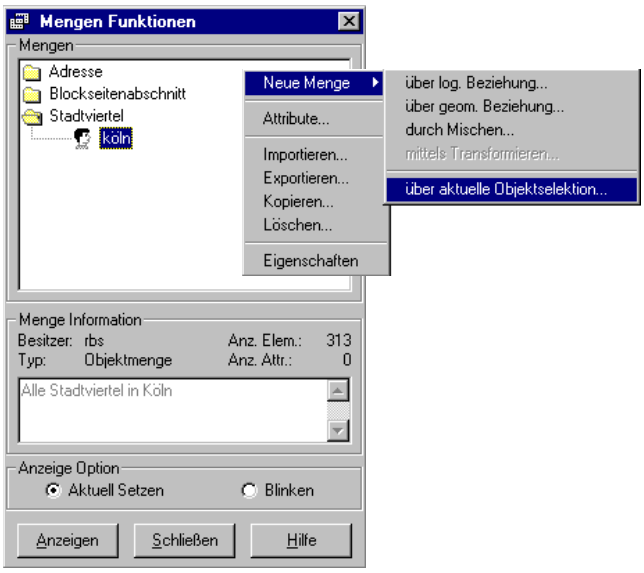


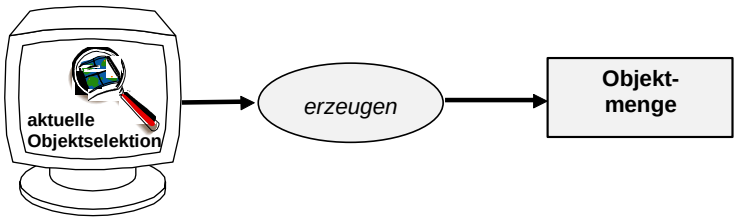
Abb. 4: Der MengenManager

Wie Eingangs erläutert, werden im SDD alle Daten - Geoinformationen sowie Sachdaten - organisiert und beschrieben. Diese Meta-Datenbeschreibung sichert den Überblick über die Daten und ermöglicht komfortable Suchmöglichkeiten. Damit wird das semantische Datenmodell automatisch in die Benutzeroberfläche eingebaut.

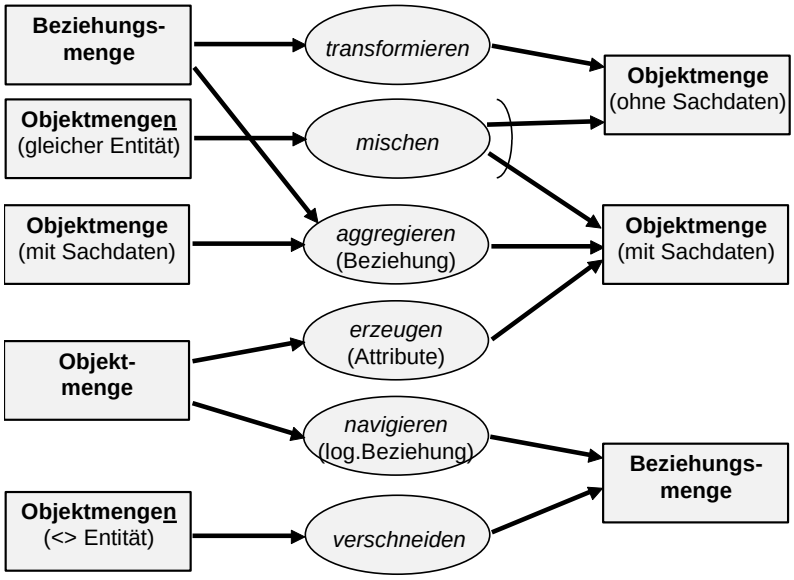
3.3 Aktionen mit Mengen

3.3.1 Menge erzeugen

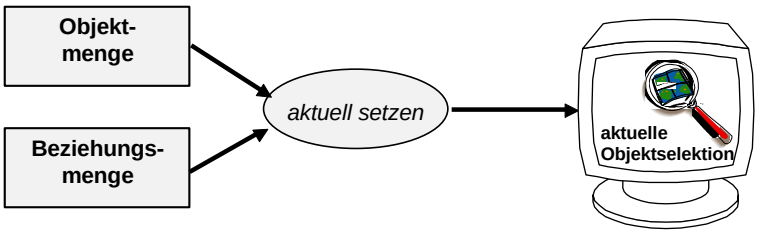
- Die aktuelle Objektselektion in ***ArcView*** kann mittels der Operation ‘*Neue Menge aus aktueller Objektselektion*’ in eine persistente Menge überführt werden.



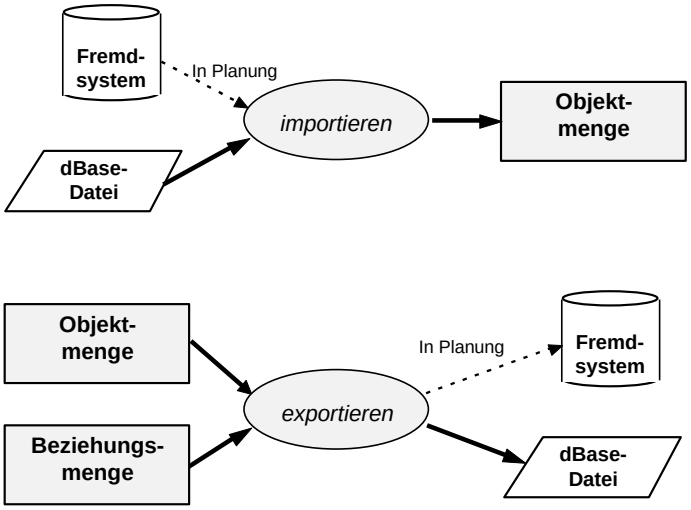
- Sind bereits Mengen im System vorhanden, so können verschiedenste Operation angewendet werden, um neue Menge zu erzeugen.



- Die Menge kann über die Operationen '*aktuell setzen*' in eine aktuelle Objektselektion in ***ArcView*** überführt werden.



- Eine Menge kann von einem RBS-Fremdsystem '*importiert*' oder in dieses '*exportiert*' werden.



3.3.2 Menge kopieren

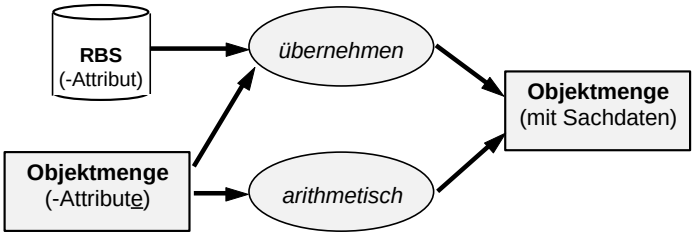
Alle Mengen, auf denen ein lesender Zugriff besteht, können kopiert werden, d.h. der Typ der Inputmenge entspricht dem Typ der Outputmenge. Dabei ändert sich nur der Besitzer und eventuell der Besitzstand der Menge.

3.3.3 Menge löschen

Es können alle Mengen, die dem aktuellen User gehören, gelöscht werden.

3.3.4 Attribut erzeugen

Attribute können von bestehenden Objektmengen mit Sachdaten oder direkt vom RBS, an eine Objektmenge kopiert werden. Weiter kann ein neues Attribut, über die bereits bestehenden Sachdaten einer Menge, mittels einer arithmetischen Operation erzeugt werden.



3.3.5 Attribut löschen

Attribute einer Objektmenge können, sofern die Objektmenge dem aktuellen User gehört, gelöscht werden.

3.4 Architektur

Beim Design und der Codierung des MengenManagers wurde versucht die neuesten Software-Standards zu verwenden, um eine moderne offene Architektur zu erhalten. Dabei wurde auf nachfolgende Technologien gesetzt:

- *Object Linking and Embedding OLE-Automatisierungs-Server/Client*
  Für die Kommunikation zwischen der OCX-Komponente (Server) und der graphischen Oberfläche (Client).
- *Open DataBase Connectivity ODBC*
  Datenbankschnittstelle zwischen dem RDBMS (ORACLE) und der OCX-Komponente.
- *Dynamic Data Exchange DDE*
  Für die Kommunikation zwischen ***ArcView*** (Server) und der graphischen Oberfläche (Client).

Die verwendeten Programmiersprachen sind Visual C++, Visual Basic und Avenue.

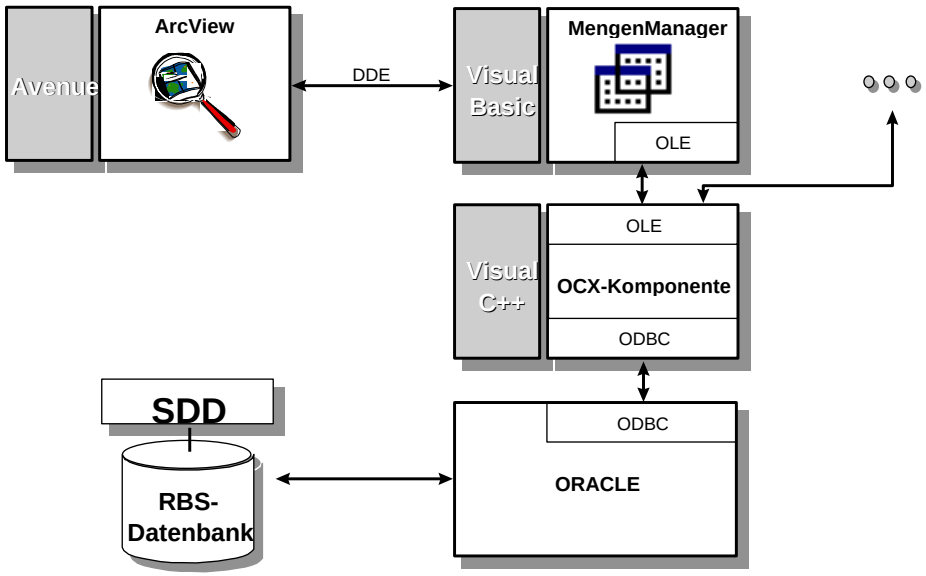


Abb. 5: Architektur

Durch die Kapselung des Funktionsmoduls, in Form eines OLE-Automatisierungs-Server mit Laufzeit-Objekten (OCX-Komponente) und der graphischen Oberfläche (MengenManager) ist die Wiederverwendbarkeit gewährleistet. So kann, basierend auf der OCX-Komponente, eine neue Benutzeroberfläche nach applikationsspezifischen Bedürfnissen mit einer beliebigen Programmiersprache entwickelt werden.

3.5 OCX-Komponente

Ein OLE-Automatisierungs-Server ist eine Anwendung, die anderen Anwendungen, Automatisierungs-Clients genannt, eine Schnittstelle zur Manipulation von Laufzeit-Objekten bietet. Diese vom Automatisierungs-Client beeinflussbaren Laufzeit-Objekte werden von diesem automatisiert, da auf sie direkt zugegriffen werden kann.

Die OCX-Komponente (Server) bildet die Schnittstelle zwischen der Datenbank (RDBMS) und der graphischen Oberfläche (Client).

4 Klassen

Klassen werden normalerweise für die Abstraktion von Objekten der realen oder der gedachten Welt verwendet.

Eine Klasse besteht aus:

- einer Menge von *Datenelemente* (Attribute), mit denen die Daten der Klasse dargestellt werden.
- einer Zusammenstellung von *Elementfunktionen* (Methoden). Diese stellen die möglichen Operationen dar, die mit der Klasse durchgeführt werden können. Sie werden als Schnittstelle (Interface) der Klasse bezeichnet.

4.1 Klassenmodell

Das Klassenmodell sieht wie folgt aus:

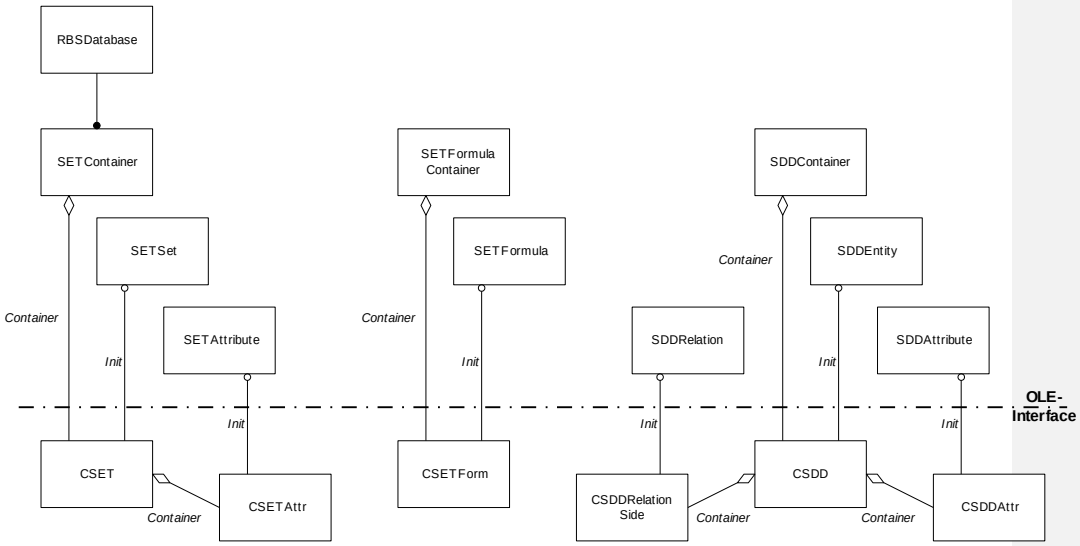


Abb. 6: Klassenmodell der OCX-Komponente

5 Methoden

Die Elementfunktionen (Methoden) sind eine Menge von Operationen, die pro Klasse zur Verfügung stehen. Sie lassen sich - je nach Verwendungszweck - in Verwaltungs-, Implementierungs-, Hilfs- und Zugriffsfunktionen unterteilen.

Der Name der Methode ist gegliedert in:

- *Aktion* (Get, Set, Delete, Create, ...)
- *Subjekt* (FirstSet, NextSet, Count, ...)

5.1 Klasse RBSDatabase

Die Klasse *RBSDatabase* ist für die Verbindung zur Datenbank zuständig. Dabei ist erforderlich, dass der ODBC-Treiber den Sourcenamen ‘RBS’ hat.

5.1.1 Connect()

Interface	`long Connect()`
Parameters	-
Description	Diese Methode muss vor jeder anderen Methode aufgerufen werden, um die Verbindung zur Datenbank herzustellen.
Returns	(-1) Bei einem Fehler. (0) Bei keinem Fehler.

5.1.2 Disconnect()

Interface	`long Disconnect()`
Parameters	-
Description	Diese Methode muss am Ende aufgefrufen werden, um die Verbindung zur Datenbank abzubrechen.
Returns	(-4) Bei einem Fehler. (0) Bei keinem Fehler.

5.2 Klasse SETContainer

Die Klasse *SETContainer* ist ein Container, in welchen die Mengen-Objekte gespeichert werden. Es werden Methoden zur Verfügung gestellt, um diesen Container zu füllen, zu leeren oder abzufragen.

5.2.1 CreateList()

Interface	`long CreateList()`
Parameters	–
Description	Mit dieser Methode werden aus der Datenbank die Mengen-Objekte gelesen und im Memory abgelegt.
Returns	(-9) Keine Verbindung zur Datenbank. (-2) Internes SQL-Statement ist falsch. (-3) Internes SQL-Statement ist falsch. (0) Bei keinem Fehler.
Notes	Mit den Methoden `GetFirst()` und `GetNext()` kann der Name der Menge, in der aktuellen Liste, gelesen werden.

5.2.2 FreeList()

Interface	`void FreeList()`
Parameters	–
Description	Diese Methode gibt den Speicher frei, der in der Funktion `CreateList()` alloziert worden ist.
Returns	-
Notes	Vorgängig muss die Funktion `CreateList()` aufgerufen worden sein.

5.2.3 GetCount()

Interface	`long GetCount(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Listeneinträge
Description	Diese Funktion liefert die Anzahl der Mengen zurück, die im Container gespeichert ist.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Mengen kann mittels einem Loop mit der Funktion `GetFirst()` und `GetNext()` genutzt werden.

5.2.4 GetFirstSet()

Interface
`long GetFirstSet(BSTR FAR* name)`

Parameters
`name` – Platzhalter, in welchen der erste Mengenname in der Liste geschrieben wird.

Description
Mit dieser Methode wird der erste Mengenname aus der Liste zurückgegeben.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(-3)	Kein Objekt gefunden.
(0)	Bei keinem Fehler.

Notes
Die Methode `CreateList()` muss vorgängig aufgefrufen worden sein.

5.2.5 GetNextSet()

Interface
`long GetNextSet(BSTR FAR* name)`

Parameters
`name` – Platzhalter, in welchen der Mengenname in der Liste geschrieben wird.

Description
Mit dieser Methode wird jeweils der nächste Mengenname aus der Liste zurückgegeben.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(1)	Listenende erreicht..
(0)	Bei keinem Fehler.

Notes
Die Methoden `CreateList()` und `GetFirst()` müssen vorgängig aufgefrufen worden sein.

5.2.6 GetSetId()

Interface
`long GetSetId(long pos, long id)`

Parameters
`pos` – Position in der Liste.
`id` – Platzhalter, in welchen die ID der Menge geschrieben wird.

Description
Mit dieser Methode kann die ID einer Menge an einer bestimmten Position in der Liste ermittelt werden.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(-2)	Die Liste ist leer.
(-3)	Das Mengenobjekt konnte nicht gefunden werden.
(-4)	Die angegebene Position ist falsch.
(0)	Bei keinem Fehler.

Notes
Die Methode `CreateList()` muss vorgängig aufgefrufen worden sein.

5.2.7 GetSetnameOnPosition()

Interface

```
long GetSetnameOnPosition(long      pos,
                          BSTR FAR* name)
```

Parameters

`pos` – Position in der Liste.
`name` – Platzhalter, in welchen der Mengenname geschrieben wird.

Description

Mit dieser Methode kann der Mengenname an einer bestimmten Position in der Liste ermittelt werden.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(-2)	Die Liste ist leer.
(-3)	Das Mengenobjekt konnte nicht gefunden werden.
(-4)	Die angegebene Position ist falsch.
(0)	Bei keinem Fehler.

Notes

Die Methode `CreateList()` muss vorgängig aufgefrufen worden sein.

5.3 Klasse SETSet

Die Klasse *SETSet* umfasst die Mengen-Operationen und alle Funktionalität, um Mengen zu erstellen, zu laden, abzuspeichern und anzuwenden.

5.3.1 ChangeTypeToRelationSet()

Interface

```
long  ChangeTypeToRelationSet(long id,
                              long rel_id)
```

Parameters

`id` – ID der Menge, die in eine Beziehungsmenge umgewandelt werden soll.
`rel_id` – ID der Beziehung.

Description

Diese Methode wandelt eine Menge, die vorher mit `InitNew()` erzeugt worden ist, anhand einer bestehenden Menge und der Beziehungsinformation in eine Beziehungsmenge um.

Returns

(-2)	Die Attribut- oder die Beziehungsliste ist leer.
(-3)	Zu der angegebenen ID konnte kein Objekt gefunden werden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Die Objektmenge ist gesperrt.
(-19)	Die Menge konnte nicht mehr freigegeben werden.
(-23)	Die zugrunde liegende Entität konnte nicht gefunden werden.
(-24)	Die zugrunde liegende Beziehung konnte nicht gefunden werden.
(-25)	Der Referenztyp der Beziehung ist falsch. Es werden nur Beziehungsmengen von der Kardinalität 1:n unterstützt.
(-26)	Der gefundene Beziehungstyp wird nicht unterstützt.
(0)	Bei keinem Fehler.

Notes

Die Menge muss vorher mit `InitNew()` erzeugt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.2 Commit()

Interface

`long Commit()`

Parameters

–

Description

Diese Methode beendigt die Erzeugung einer neuen Menge, wenn zuvor `InitNew()` in Kombination mit `SetNameRef(), Transform(), Copy()` oder `Merge...()` aufgerufen worden ist oder die Methode beendigt die Manipulation der Zustandsänderung der Menge (`SetName(), SetEntity(), ...`), sofern die Menge vorgängig mit `Lock()` gesperrt worden ist.

Ebenfalls wird diese Methode bei der Manipulation von Attributen (siehe SETAttribute) benötigt. Die Methode beendigt die Erzeugung eines neuen Attributes, wenn zuvor `InitNewRbs(), InitNewArithmetic()` oder `InitNew...()` (Aggregatfunktionen) aufgerufen worden ist oder die Methode ändert ein Attribut in der Datenbank, wenn dieses zuvor mit `SetName()` oder `SetDescription()` editiert worden ist.

Returns

(-2)	Die Entitäts- oder die Attributliste ist leer.
(-3)	Zugrundeliegendes Objekt (Menge oder Attribut) nicht gefunden.
(-11)	Interner Puffer überlaufen.
(-14)	Allgemeiner Fehler.
(-16)	Vorgängig erhaltene Werte sind falsch.
(-18)	Attribut kann nicht an eine Beziehungsmenge angefügt werden.
(-21)	Der Datentyp ‘Datum’ wird noch nicht unterstützt.
(-22)	Entitätsliste ist leer.
(-23)	Entität nicht gefunden.
(-27)	Attribut nicht gefunden.
(-28)	Nicht genügend Speicherplatz vorhanden.
(< -100)	SQL-Error.
(0)	Bei keinem Fehler.

Notes

Bei der Verwendung dieser Methode von Mengen-manipulationen müssen, bevor diese Methode aufgerufen werden kann, mindestens die Entitätsgehörigkeit mit `SetEntity()` gesetzt werden. Die anderen Komponenten werden automatisch oder mit Default-Werten gefüllt.

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden.

5.3.3 Copy ()

Interface	`long Copy(long id)`
Parameters	`id` – ID der Quell-Menge, die kopiert werden soll.
Description	Diese Methode kopiert eine Menge.
Returns	(-2) SQL-Error. (-3) SQL-Error. (-14) Die Quell-Menge konnte nicht gesperrt werden. (-15) Die begonnene Aktion muss zuerst abgeschlossen werden. (-17) Die Quell-Menge ist gesperrt. (-19) Die Quell-Menge konnte nicht freigegeben werden. (< -100) SQL-Error. (0) Bei keinem Fehler.
Notes	Die Ziel-Menge muss vorher mit `InitNew()` erzeugt worden sein. Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.4 CreateAttributeList()

Interface	`long CreateAttributeList()`
Parameters	–
Description	Mit dieser Methode werden die Attribute einer Menge aus der Datenbank gelesen und im Memory abgelegt. Mit den Methoden `GetFirstAttribute()` und `GetNextAttribute()` kann aus der Attributliste der Name des Attributes abgefragt werden.
Returns	(-2) Internes SQL-Statement ist falsch. (-3) Internes SQL-Statement ist falsch. (-9) Keine Verbindung zur Datenbank. (-14) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.3.5 Delete ()

Interface
```
long Delete()
```

Parameters
–

Description
Mit dieser Methode kann die Menge gelöscht werden.

Returns

(-2)	Internes SQL-Statement ist falsch.
(-3)	Das Mengen-Objekt ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Die Menge ist gesperrt.
(-18)	Keine Berechtigung die Menge zu löschen.
(-19)	Menge konnte nicht freigegeben werden.
(0)	Bei keinem Fehler.

Notes
Das SETSet-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.3.6 FlashSet()

Interface
```
long FlashSet(LPCTSTR path,   LPCTSTR tablename,
              long    attrId, LPCTSTR op,
              LPCTSTR value)
```

Parameters
`path` – Pfad in welchen die dBase-Datei geschrieben werden soll.
`tablename` – Name der dBase-Datei.
`attrId` – 0 (wird nicht verwendet).
`op` – “” (wird nicht verwendet).
`value` – “” (wird nicht verwendet).

Description
Diese Methode schreibt gemäss den Parametern eine dBase-Datei (`tablename`.dbf) mit den Identifikatoren der Geo-Objekte einer Menge.

Returns

(-1)	Die Enitätsliste ist nicht initialisiert worden.
(-2)	Internes SQL-Statement ist falsch.
(-5)	dBase-Datenbank konnte nicht geöffnet werden.
(-8)	Fehler beim Schreiben der dBase-Datei.
(-9)	Keine Verbindung zur Datenbank.
(-12)	dBase-Datei existiert bereits.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(0)	Bei keinem Fehler.

Notes
Die Menge muss vorher mit `Init()` initialisiert worden sein.

5.3.7 FreeAttributeList()

Interface	`void FreeAttributeList()`
Parameters	–
Description	Diese Methode gibt den Speicher frei, der durch der Methode `CreateAttributeList()` alloziert worden ist.
Returns	-
Notes	Vorgängig muss die Funktion `CreateAttributeList()` aufgerufen worden sein.

5.3.8 GetCount()

Interface	`long GetCount(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Anzahl der Elemente der Menge geschrieben wird.
Description	Mit dieser Methode kann die Anzahl der Elemente der Menge abgefragt werden.
Returns	(-3) Die Menge ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.3.9 GetCurrentObjectSelection()

Interface

```
long GetCurrentObjectSelection(
                              LPCTSTR path,
                              LPCTSTR tablename,
                              long    entity,
                              long    relationEntity)
```

Parameters

`path` – Pfad in wo sich die dBase-Datei befindet.
`tablename` – Name der dBase-Datei.
`entity` – ID der (Quell-) Entität.
`relationEntity` – Handelt es sich um eine Beziehungs-menge, so muss die ID der Zielentität angegeben werden, ansonsten 0.

Description

Mit dieser Methode kann eine dBase-Datei in eine Menge überführt werden. Weist die Datei mehr als eine Spalte auf, so werden die restlichen Spalten automatisch als Attribute erkannt und abgespeichert. Die Namenskonvention der ersten Spalte ist:
<nickname>_ID, wobei <nickname> der Name der Business-Tabelle der zugrundeliegenden Entität ist.

Returns

(-2)	Internes SQL-Statement ist falsch.
(-3)	Internes SQL-Statement ist falsch.
(-5)	dBase-Datenbank konnte nicht geöffnet werden.
(-8)	Fehler in der dBase-Datei.
(-9)	Keine Verbindung zur Datenbank oder Recordset in dBase-Datei konnte nicht gelesen werden.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-16)	Im Angegebenen Pfad ist keine dBase-Datei vorhanden.
(-20)	Die erste Spalte der dBase-Datei ist kein numerischer Wert oder Attributtyp wird nicht unterstützt.
(-28)	Kein Speicherplatz vorhanden.
(0)	Bei keinem Fehler.

Notes

Das SETSet-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.3.10 GetDate()

Interface

```
long GetDate(BSTR FAR* value)
```

Parameters

`value` – Platzhalter, in welchen das Erstellungsdatum geschrieben wird.

Description

Mit dieser Methode kann das Erstellungsdatum der Menge abgefragt werden.

Returns

(-3)	Das Mengen-Objekt ist nicht initialisiert worden.
(0)	Bei keinem Fehler.

Notes

Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.11 GetDescription()

Interface	`long GetDescription(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Beschreibung geschrieben wird.
Description	Mit dieser Methode kann die Beschreibung der Menge abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.12 GetEntity()

Interface	`long GetEntity(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Entitäts-ID geschrieben wird.
Description	Mit dieser Methode kann Entitäts-ID der Menge abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.13 GetFirstAttribute()

Interface	`long GetFirstAttribute(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der erste Attributname der Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Attributname der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `CreateAttributeList()` muss vorgängig aufgefrufen worden sein.

5.3.14 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID der Menge geschrieben wird.
Description	Diese Methode liefert die ID der Menge zurück.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.15 GetName()

Interface	`long GetName(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Name geschrieben wird.
Description	Mit dieser Methode kann der Name der Menge abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.16 GetNextAttribute()

Interface	`long GetNextAttribute(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der Attributname der Liste geschrieben wird.
Description	Mit dieser Methode wird jeweils der nächste Attributname der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht. (0) Bei keinem Fehler.
Notes	Die Methoden `CreateAttributeList()` und `GetFirstAttribute()` müssen vorgängig aufgefrufen worden sein.

5.3.17 GetNoOfAttributes()

Interface	`long GetNoOfAttributes(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Attribute der Menge
Description	Diese Funktion liefert die Anzahl der Attribute zurück.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Attributen kann mittels einem Loop mit der Funktion `GetFirstAttribute()` und `GetNextAttribute()` genutzt werden.

5.3.18 GetOwner()

Interface	`long GetOwner(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Besitzer geschrieben wird.
Description	Mit dieser Methode kann der Besitzer der Menge abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.19 GetProject()

Interface	`long GetProject(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Projekt-ID geschrieben wird.
Description	Mit dieser Methode kann Projekt-ID der Menge (ID der langen Transaktion) abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.20 GetRelationEntity()

Interface	`long GetRelationEntity(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Zielentitäts-ID geschrieben wird.
Description	Mit dieser Methode kann Zielentitäts-ID der Beziehungsmenge abgefragt werden.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.21 GetScope()

Interface	`long GetScope(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Besitzstand geschrieben wird.
Description	Mit dieser Methode kann der Besitzstand der Menge abgefragt werden. 1 = Allgemein 2 = Private (3 = Infoassistent)
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.22 GetStatus()

Interface	`long GetStatus(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Status der Menge geschrieben wird.
Description	Mit dieser Methode kann der Status der Menge abgefragt werden. 1 = Initialisiert 2 = Referenziert durch Element-ID 3 = Referenziert durch den Elementnamen 4 = Referenziert durch Element-ID und -Namen.
Returns	(-3) Das Mengen-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.23 GetType ()

Interface `long GetType(long FAR* value)`

Parameters `value` – Platzhalter, in welchen der Typ geschrieben wird.

Description Mit dieser Methode kann der Type der Menge abgefragt werden.
1 = Objektmenge ohne Sachdaten
2 = Objektmenge mit Sachdaten
3 = Beziehungsmenge

Returns

(-3)	Das Mengen-Objekt ist nicht initialisiert worden.
(0)	Bei keinem Fehler.

Notes Das SETSet-Objekt muss vorher mit `init()` initialisiert worden sein.

5.3.24 Init()

Interface `long Init(long pos)`

Parameters `pos` – Positionsindex der Menge in der Liste

Description Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(-2)	Die Liste ist leer.
(-3)	Kein Objekt gefunden.
(-4)	Angegebene Position ist falsch.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(0)	Bei keinem Fehler.

Notes Die Liste muss vorher mit `CreateList()` erzeugt worden sein.

5.3.25 InitNew()

Interface `long InitNew(BSTR FAR* name)`

Parameters `name` – Ein Zeiger auf den Namen der neuen Menge.

Description Diese Methode erzeugt eine Menge mit dem angegebenen Namen. Der Status der Menge ist auf initialisiert gesetzt.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Internes SQL-Statement ist fehlgeschlagen.
(-5)	Bei einem Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(0)	Bei keinem Fehler.

Notes Nach diesem Aufruf muss `GetCurrentObjectSelection()`, `Transform()`, `Copy()` oder `Merge...()` folgen und am Schluss `Commit()` oder `Rollback()`.

5.3.26 Lock ()

Interface
```
long Lock()
```

Parameters
–

Description
Diese Methode sperrt die Menge für Änderungszwecke.

Ebenfalls wird diese Methode verwendet, um die Menge zu sperren, wenn Manipulationen an Attributen erfolgt.

Returns

(-3)	Das Mengen-Objekt ist nicht initialisiert worden.
(-14)	Die Menge konnte nicht gesperrt werden.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(0)	Bei keinem Fehler.

Notes
Nach diesem Funktionsaufruf muss `Commit()` oder `Rollback()` folgen.

5.3.27 MergeAnd()

Interface
```
long MergeAnd(long setId1,
              long setId2,
              long FAR* newRows)
```

Parameters
`setId1` – ID der ersten Quell-Menge.
`setId2` – ID der zweiten Quell-Menge.
`newRows` – Platzhalter, in welchen die Anzahl der Zeilen, die in die Menge eingetragen worden sind, geschrieben wird.

Description
Diese Methode bildet die Schnittmenge von zwei Mengen der selben Entität.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Quell-Menge nicht gefunden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Die Quell-Menge ist gesperrt.
(-19)	Quell-Menge konnte nicht freigegeben werden.
(0)	Bei keinem Fehler.

Notes
Die Ziel-Menge muss vorher mit `InitNew()` erzeugt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.28 MergeMinus()

Interface

```
long MergeMinus(long setId1,
                long setId2,
                long FAR* newRows)
```

Parameters

`setId1` – ID der ersten Quell-Menge.
`setId2` – ID der zweiten Quell-Menge.
`newRows` – Platzhalter, in welchen die Anzahl der Zeilen, die in die Menge eingetragen worden sind, geschrieben wird.

Description

Diese Methode bildet die Differenzmenge von zwei Mengen der selben Entität.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Quell-Menge nicht gefunden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Die Quell-Menge ist gesperrt.
(-19)	Quell-Menge konnte nicht freigegeben werden.
(0)	Bei keinem Fehler.

Notes

Die Ziel-Menge muss vorher mit `InitNew()` erzeugt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.29 MergeOr()

Interface

```
long MergeOr(long setId1,
             long setId2,
             long FAR* newRows)
```

Parameters

`setId1` – ID der ersten Quell-Menge.
`setId2` – ID der zweiten Quell-Menge.
`newRows` – Platzhalter, in welchen die Anzahl der Zeilen, die in die Menge eingetragen worden sind, geschrieben wird.

Description

Diese Methode bildet die Vereingungsmenge von zwei Mengen der selben Entität.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Quell-Menge nicht gefunden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Die Quell-Menge ist gesperrt.
(-19)	Quell-Menge konnte nicht freigegeben werden.
(0)	Bei keinem Fehler.

Notes

Die Ziel-Menge muss vorher mit `InitNew()` erzeugt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.30 Rollback ()

Interface

```
long  Rollback()
```

Parameters

–

Description

Diese Methode beendigt die vorgängige(n) Operation(en). Sie darf nach `InitNew()`, `Lock()`, `Transform()`, `SetNameRef()`, `GetCurrenObjectSelection()`, `Copy()` oder `Merge...()` aufgerufen werden.

Ebenfalls wird diese Methode bei der Manipulation von Attributen (siehe SETAttribute) benötigt. Die Methode die Methode darf nach `Init-NewRbs()`, `InitNewArithmetic()` oder `InitNew...()` (Aggregatfunktionen) aufgerufen werden.

Returns

(-3)	Menge nicht gefunden.
(0)	Bei keinem Fehler.

5.3.31 SetCurrentObjectSelection()

Interface

```
long SetCurrentObjectSelection(
                            LPCTSTR path,
                            LPCTSTR tablename)
```

Parameters

`path` – Pfad wo die dBase-Datei hingeschrieben werden soll.
`tablename` – Name der dBase-Datei.

Description

Mit dieser Methode kann eine Menge in eine dBase-Datei überführt werden. Die Namenskonvention der ersten Spalte ist:
<nickname>_ID, wobei <nickname> der Name der Business-Tabelle der zugrundeliegenden Entität ist.

Returns

(-1)	Entitäts- oder Attributliste ist nicht erzeugt worden.
(-2)	Internes SQL-Statement ist falsch.
(-3)	Die Menge ist nicht initialisiert worden.
(-5)	dBase-Datenbank konnte nicht geöffnet werden.
(-8)	Fehler in der dBase-Datei.
(-13)	Es existieren Attributfelder, die identische Namen haben.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-16)	Datentyp wird nicht untersützt.
(0)	Bei keinem Fehler.

Notes

Das SETSet-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.3.32 SetDescription()

Interface	`long SetDescription(BSTR FAR* value)`
Parameters	`value` – Wert, mit dem die Beschreibung der Menge überschrieben werden soll.
Description	Mit dieser Methode kann die Beschreibung der Menge geändert oder erfasst werden.
Returns	(-14) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu angelegt werden.

5.3.33 SetEntity()

Interface	`long SetEntity(long FAR* value)`
Parameters	`value` – Wert, mit dem die Entitätszugehörigkeit überschrieben werden soll.
Description	Mit dieser Funktion kann die Entitätszugehörigkeit der Menge geändert oder erfasst werden (ID der Entität).
Returns	(-14) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu angelegt werden.

5.3.34 SetName()

Interface	`long SetName(BSTR FAR* value)`
Parameters	`value` – Wert, mit dem der Name der Menge überschrieben werden soll.
Description	Mit dieser Methode kann der Name der Menge geändert oder erfasst werden.
Returns	(-14) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu angelegt werden.

5.3.35 SetNameRef()

Interface
`long SetNameRef()`

Parameters
–

Description
Diese Methode ändert den Status der Menge (4: Referenziert durch ID und Elementname) und kopiert automatisch den sprechenden RBS-Identifikator - OBJEKTNAME an die bestehende Menge, sofern diese den Status 'Referenziert durch ID' hat.

Returns

(-2)	Entitätsliste ist nicht erzeugt worden.
(-3)	SQL-Statement ist fehlgeschlagen.
(-14)	Bei einem Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden. Die Menge hat den falschen Status oder Typ.
(-16)	Die Menge ist gesperrt.
(-18)	Die Menge konnte nicht freigegeben werden.
(-20)	Entität nicht gefunden
(-23)	Bei keinem Fehler.
(0)	

Notes
Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu erzeugt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.3.36 SetProject()

Interface
`long SetProject(long FAR* value)`

Parameters
`value` – Wert, mit dem die Projektzugehörigkeit überschrieben werden soll.

Description
Mit dieser Funktion kann die Projektzugehörigkeit der Menge geändert oder erfasst werden (ID der langen Transaktion).

Returns

(-14)	Bei einem Fehler.
(0)	Bei keinem Fehler.

Notes
Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu angelegt werden.

5.3.37 SetScope()

Interface	`long SetScope(long FAR* value)`
Parameters	`value` – Wert, mit dem der Besitzstand überschrieben werden soll.
Description	Mit dieser Funktion kann der Besitzstand der Menge geändert oder erfasst werden (Assistent = 3, Privat =2, Public =1).
Returns	(-14) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Die Menge muss vorher mit `Lock()` gesperrt oder mit `InitNew()` neu angelegt werden.

5.3.38 Transform ()

Interface	`long Transform(long relationSetId)`
Parameters	`relationSetId` – ID der Beziehungsmenge, die in eine Objektmenge transformiert wird.
Description	Mit dieser Methode wird die 'nach'-Spalte einer Beziehungsmenge in eine Menge überführt.
Returns	(-3) Beziehungsmenge nicht gefunden. (-14) Allgemeiner Fehler. (-15) Die begonnene Aktion muss zuerst abgeschlossen werden. (-16) Angegebene Menge ist keine Beziehungsmenge. (-17) Beziehungsmenge ist gesperrt. (-19) Beziehungsmenge konnte nicht freigegeben werden. (0) Bei keinem Fehler.
Notes	Die Menge muss vorher mit `InitNew()` erzeugt worden sein. Wenn die Funktion erfolgreich war, muss mittels den Funktionen `SetScope()` und `SetDescription()` der Besitzstand bzw. die Beschreibung gesetzt werden. Danach muss mit `Commit()` die Transaktion beendet oder mit `Rollback()` zurückgesetzt werden.

5.4 Klasse SETAttribute

Die Klasse *SETAttribute* umfasst die Attribut-Operationen und alle Funktionalität, um Sachdaten einer Menge zu erstellen, abzuspeichern, zu löschen und anzuwenden.

5.4.1 CreateUniqueList()

Interface	`long CreateUniqueList()`
Parameters	–
Description	Diese Methode erstellt eine eindeutige Liste aller Werte eines Attributes.
Returns	(-2) Internes SQL-Statement ist fehlgeschlagen. (-3) Internes SQL-Statement ist fehlgeschlagen. (-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-9) Keine Datenbankverbingung. (0) Bei keinem Fehler.
Notes	Mit den Methoden `GetFirst()` und `GetNext()` kann der Attributwert aus der aktuellen Liste gelesen werden

5.4.2 Delete()

Interface	`long Delete()`
Parameters	–
Description	Diese Methode löscht das Attribut.
Returns	(-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-14) Allgemeiner Fehler. (-16) Menge hat keine Attribute. (-17) Menge ist gesperrt. (0) Bei keinem Fehler.
Notes	-

5.4.3 FlashAttributeValue()

Interface

```
long FlashAttributeValue(LPCTSTR path,
                         LPCTSTR tablename,
                         long    attrId,
                         LPCTSTR op,
                         LPCTSTR value)
```

Parameters

`path` – Pfad in welchen die dBase-Datei geschrieben wird.
`tablename` – Name der dBase-Datei.
`attrId` – ID des Attributes
`op` – Operator für den Vergleich (Mögliche Operatoren sind: =, <>, >=, <=, >, <, like, is null, is not null)
`value` – Wert mit welchem verglichen werden soll.

Description

Diese Methode schreibt gemäss den Parametern eine dBase-Datei (`tablename`.dbf) mit den Identifikatoren der Geo-Objekte einer Menge.

Returns

(-1)	Die Enitätsliste ist nicht initialisiert worden.
(-2)	Internes SQL-Statement ist falsch.
(-5)	dBase-Datenbank konnte nicht geöffnet werden.
(-8)	Fehler beim Schreiben der dBase-Datei.
(-9)	Keine Verbindung zur Datenbank.
(-12)	dBase-Datei existiert bereits.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(0)	Bei keinem Fehler.

Notes

Die Menge und das Attribut muss vorher mit `Init()` initialisiert worden sein.

5.4.4 FreeUniqueList()

Interface

```
void FreeAttributeList()
```

Parameters

–

Description

Diese Methode gibt den Speicher frei, der durch der Methode `CreateUniqueList()` alloziert worden ist.

Returns

-

Notes

Vorgängig muss die Funktion `CreateUniqueList()` aufgerufen worden sein.

5.4.5 GetAverage()

Interface
`long GetAverage(double FAR* average)`

Parameters
`average` – Platzhalter, in welchen der Durchschnittswert eines Mengen-Attributes geschrieben wird.

Description
Mit dieser Methode kann der Durchschnittswert eines Mengen-Attributes abgefragt werden.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Internes SQL-Statement ist fehlgeschlagen.
(-6)	Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden.
(-9)	Keine Datenbankverbindung.
(0)	Bei keinem Fehler.

Notes
Das Mengen- und das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.6 GetCount()

Interface
`long GetCount(long FAR* value)`

Parameters
`value` – Platzhalter, in welchen die Anzahl der Elemente der eindeutigen Liste geschrieben wird.

Description
Mit dieser Methode kann die Anzahl der Elemente der eindeutigen Liste abgefragt werden.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(-2)	Die List ist Leer.
(0)	Bei keinem Fehler.

Notes
Das Mengen- und Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein. Ebenfalls muss die Liste vorgängig mit `CreateUnique-List()` erzeugt worden sein.

5.4.7 GetDescription()

Interface	`long GetDescription(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen die Beschreibung des Attributes geschrieben wird.
Description	Mit dieser Methode kann die Beschreibung des Attributes abgefragt werden.
Returns	(-3) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.8 GetFirst ()

Interface	`long GetFirst(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der erste Wert von der eindeutigen Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Wert von der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `CreateUniqueList()` muss vorgängig aufgefrufen worden sein.

5.4.9 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID des Attributes geschrieben wird.
Description	Mit dieser Methode kann die ID des Attributes abgefragt werden.
Returns	(-3) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.10 GetLength()

Interface	`long GetLength(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Länge des Attributfeldes geschrieben wird.
Description	Mit dieser Methode kann die Länge des Attributfeldes abgefragt werden.
Returns	(-3) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.11 GetMinMaxFloat()

Interface	`long GetMinMaxFloat(double FAR* min,` `double FAR* max)`
Parameters	`min` – Platzhalter, in welchen der kleinste Wert des Attributs geschrieben wird. `max` – Platzhalter, in welchen der grösste Wert des Attributs geschrieben wird.
Description	Mit dieser Methode kann der kleinste und der grösste Wert des Attributs abgefragt werden.
Returns	(-2) Internes SQL-Statement ist fehlgeschlagen. (-3) Internes SQL-Statement ist fehlgeschlagen. (-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-9) Keine Datenbankverbindung. (0) Bei keinem Fehler.
Notes	Das Mengen und Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.12 GetMinMaxInteger()

Interface

```
long GetMinMaxInteger(long FAR* min,
                      long FAR* max)
```

Parameters

`min` – Platzhalter, für den kleinsten Wert des Attributs.
`max` – Platzhalter, für den grössten Wert des Attributs.

Description

Mit dieser Methode kann der kleinste und der grösste Wert des Attributs abgefragt werden.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Internes SQL-Statement ist fehlgeschlagen.
(-6)	Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden.
(-9)	Keine Datenbankverbindung.
(0)	Bei keinem Fehler.

Notes

Das Mengen und Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.13 GetName()

Interface

```
long GetName(BSTR FAR* value)
```

Parameters

`value` – Platzhalter, in welchen der Name des Attributes geschrieben wird.

Description

Mit dieser Methode kann der Name des Attributes abgefragt werden.

Returns

(-3)	Bei einem Fehler.
(0)	Bei keinem Fehler.

Notes

Das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.14 GetNext ()

Interface

```
long GetNext(BSTR FAR* value)
```

Parameters

`value` – Platzhalter, in welchen der Wert von der eindeutigen Liste geschrieben wird.

Description

Mit dieser Methode wird jeweils der nächste Wert aus der Liste zurückgegeben.

Returns

(-1)	Die Liste ist vorgängig nicht erzeugt worden.
(1)	Listenende erreicht..
(0)	Bei keinem Fehler.

Notes

Die Methoden `CreateUniqueList()` und `GetFirst()` müssen vorgängig aufgefrufen worden sein.

5.4.15 GetNo()

Interface

```
long GetNo(long FAR* num)
```

Parameters

`num` – Platzhalter, in welchen die Anzahl Werte des Attributs geschrieben wird.

Description

Mit dieser Methode kann die Anzahl der Werte, welche nicht 'NULL' sind, des Attributs abgefragt werden.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Internes SQL-Statement ist fehlgeschlagen.
(-6)	Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden.
(-9)	Keine Datenbankverbindung.
(0)	Bei keinem Fehler.

Notes

Das Mengen- und Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.16 GetNoNull()

Interface

```
long GetNoNull(long FAR* num)
```

Parameters

`num` – Platzhalter, in welchen die Anzahl Werte des Attributs geschrieben wird.

Description

Mit dieser Methode kann die Anzahl der Werte, welche 'NULL' sind, des Attributs abgefragt werden.

Returns

(-2)	Internes SQL-Statement ist fehlgeschlagen.
(-3)	Internes SQL-Statement ist fehlgeschlagen.
(-6)	Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden.
(-9)	Keine Datenbankverbindung.
(0)	Bei keinem Fehler.

Notes

Das Mengen- und Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.17 GetStddev()

Interface	`long GetStddev(double FAR* stddev)`
Parameters	`stddev` – Platzhalter, in welchen die Standardabweichung eines Mengen-Attributes geschrieben wird.
Description	Mit dieser Methode kann die Standardabweichung eines Mengen-Attributes abgefragt werden.
Returns	(-2) Internes SQL-Statement ist fehlgeschlagen. (-3) Internes SQL-Statement ist fehlgeschlagen. (-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-9) Keine Datenbankverbindung. (0) Bei keinem Fehler.
Notes	Das Mengen- und das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.18 GetSum()

Interface	`long GetSum(double FAR* sum)`
Parameters	`sum` – Platzhalter, in welchen die Summe eines Mengen-Attributes geschrieben wird.
Description	Mit dieser Methode kann die Summe eines Mengen-Attributes abgefragt werden.
Returns	(-2) Internes SQL-Statement ist fehlgeschlagen. (-3) Internes SQL-Statement ist fehlgeschlagen. (-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-9) Keine Datenbankverbindung. (0) Bei keinem Fehler.
Notes	Das Mengen- und das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.19 GetType()

Interface	`long GetType(long FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Typ geschrieben wird.
Description	Mit dieser Methode kann der Typ des Attributes abgefragt werden (1=Text, 2=Zahl, 3=Gleitkommazahl, 4=Datum).
Returns	(-3) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.20 GetVariance()

Interface	`long GetVariance(double FAR* var)`
Parameters	`var` – Platzhalter, in welchen die Varianz eines Mengen-Attributes geschrieben wird.
Description	Mit dieser Methode kann die Varianz eines Mengen-Attributes abgefragt werden.
Returns	(-2) Internes SQL-Statement ist fehlgeschlagen. (-3) Internes SQL-Statement ist fehlgeschlagen. (-6) Das Mengen- oder Attribut-Objekt ist nicht initialisiert worden. (-9) Keine Datenbankverbindung. (0) Bei keinem Fehler.
Notes	Das Mengen- und das Attribute-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.4.21 Init()

Interface

```
long Init(long pos)
```

Parameters

`pos` – Positionsindex des Attributs in der Liste.

Description

Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.

Returns

(-2)	Die Attributliste ist leer.
(-3)	Das Mengen-Objekt ist nicht initialisiert worden.
(-4)	Die angegebene Position ist falsch.
(0)	Bei keinem Fehler.

Notes

Die Liste der Attribute muss vorher mit `SET-Set.createAttributeList()` erzeugt und das Mengen-Objekt mit `Init()` initialisiert worden sein.

5.4.22 InitNewArithmetic()

Interface

```
long InitNewArithmetic(long    formulaId,
                       LPCTSTR attributeArray,
                       long    num,
                       LPCTSTR name)
```

Parameters

`formulaId` – ID der arithmetischen Formel mit welcher das neue Attribut berechnet und erzeugt wird.
`attributeArray` – Liste von Attribut-Positionsindizes, die in die Formel eingesetzt werden (Format: “n1;n2;n3“ z.B. “3;5;1“).
`num` – Anzahl der Attribute die benötigt werden.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Methode kann mittels einer arithmetische Formel, die im SDD gespeichert ist und über die numerischen Attribute der Menge ein neues Attribut erzeugt werden.

Returns

(-6)	Das Mengen-Objekt ist nicht initialisiert worden.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-16)	Die aktuelle Menge hat keine Attribute oder Attribut-ID ist falsch.
(-17)	Die Menge ist gesperrt.
(-28)	Kein Speicherplatz verhanden.
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.23 InitNewAverage()

Interface

```
long InitNewAverage(long    flag,
                    long    attributeSetId,
                    long    relationSetId,
                    long    attrId,
                    LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Durchschnitt' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.24 InitNewCount()

Interface

```
long InitNewCount(long    flag,
                  long    attributeSetId,
                  long    relationSetId,
                  long    attrId,
                  LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Anzahl' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.25 InitNewMaximum()

Interface

```
long InitNewMaximum(long    flag,
                    long    attributeSetId,
                    long    relationSetId,
                    long    attrId,
                    LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Maximum' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.26 InitNewMinimum()

Interface

```
long InitNewMinimum(long    flag,
                    long    attributeSetId,
                    long    relationSetId,
                    long    attrId,
                    LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Minimum' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.27 initNewRbs()

Interface

```
long initNewRbs(long attrId, LPCTSTR name)
```

Parameters

`attrId` – ID eines Entitäts-Attributes, welches im SDD gespeichert ist.
`name` – Name des Attributes (kann unterschiedlich sein, zum SDD-Attributnamen).

Description

Mit dieser Methode kann ein Entitäts-Attribut vom RBS (Raumbezugssystem) an eine Menge kopiert werden.

Returns

(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Menge ist gesperrt.
(-23)	Attribut nicht gefunden.
(-27)	Attribut nicht gefunden.
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.28 initNewSet()

Interface

```
long initNewSet(long    setId, long attrId,
                LPCTSTR name)
```

Parameters

`setId` – ID der Menge von welcher ein Attribut kopiert werden soll.
`attrId` – ID eines Attributes, von der Menge mit der ID 'setID'.
`name` – Name des Attributes (kann unterschiedlich sein, zum Mengen-Attributnamen).

Description

Mit dieser Methode kann ein Attribut einer Menge an die Menge kopiert werden.

Returns

(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-15)	Die begonnene Aktion muss zuerst abgeschlossen werden.
(-17)	Menge ist gesperrt.
(-23)	Attribut nicht gefunden.
(-27)	Attribut nicht gefunden.
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.29 InitNewStddev()

Interface

```
long InitNewStddev(long    flag,
                   long    attributeSetId,
                   long    relationSetId,
                   long    attrId,
                   LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Standardabweichung' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.30 InitNewSum()

Interface

```
long InitNewSum(long    flag,
                long    attributeSetId,
                long    relationSetId,
                long    attrId,
                LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Summe' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.31 InitNewVariance()

Interface

```
long InitNewVariance(long    flag,
                     long    attributeSetId,
                     long    relationSetId,
                     long    attrId,
                     LPCTSTR name)
```

Parameters

`flag` – Herkunft des Attributes.
1 = von der Menge; 'attributeSetId' = Mengen-ID
'attrId' = Attribut-ID einer Menge
2 = von der Entität; 'attributeSetId' = Entitäts-ID
'attrId' = Attribut-ID einer Entität
`attributeSetId` – ID der Attributmenge oder der Entität.
`relationSetId` – ID der Beziehungsmenge.
`attrId` – ID des Attributes der Attributmenge oder der Entität.
`name` – Name des neuen Attributes welches automatisch erstellt wird.

Description

Mit dieser Funktion kann ein Attribut einer Attributmenge oder Entität über eine Beziehungsmenge aggregiert werden. Die Aggregation wird mittels der Funktion 'Varianz' berechnet.

Returns

(-1)	Mengen-Liste nicht erstellt.
(-6)	Die aktuelle Menge ist nicht initialisiert worden.
(-14)	Allgemeiner Fehler.
(-17)	Menge ist gesperrt.
(-27)	Keine Attribute gefunden. (ev. Liste nicht erstellt)
(0)	Bei keinem Fehler.

Notes

Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden, ansonsten `Commit()`.

5.4.32 SetDescription()

Interface

```
long SetDescription(LPCTSTR name)
```

Parameters

`name` – Neue Beschreibung des Attributes

Description

Mit dieser Methode kann die Beschreibung des Attributes geändert werden.

Returns

(-14)	Bei einem Fehler.
(0)	Bei keinem Fehler.

Notes

Das Attribut muss vorher mit `Init()` initialisiert und die Menge mit `Lock()` gesperrt worden sein.
Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden.

Bei Beendigung aller `Set...()`–Transaktion(en) muss `Rollback()` oder `Commit()` aufgerufen werden.

5.4.33 SetName()

Interface	`long SetName(LPCTSTR name)`
Parameters	`name` – Neuer Attributname.
Description	Mit dieser Methode kann der Attributname geändert werden.
Returns	(-14) Bei einem Fehler. (0) Bei keinem Fehler.
Notes	Das Attribut muss vorher mit `Init()` initialisiert und die Menge mit `Lock()` gesperrt worden sein. Wenn diese Operation fehl schlägt muss `Rollback()` aufgerufen werden ansonsten `Commit()`.

5.5 Klasse SETFormulaContainer

Die Klasse *SETFormulaContainer* ist ein Container, in welchen die Formeln-Objekte gespeichert werden. Es werden Methoden zur Verfügung gestellt, um diesen Container zu füllen, zu leeren oder abzufragen.

5.5.1 Count()

Interface	`long Count(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Listeneinträge
Description	Diese Funktion liefert die Anzahl der Formeln zurück, die im Container gespeichert ist.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Mengen kann mittels einem Loop mit der Funktion `GetFirst()` und `GetNext()` genutzt werden.

5.5.2 CreateList()

Interface	`long CreateList()`
Parameters	–
Description	Mit dieser Methode werden aus der Datenbank die Formeln-Objekte gelesen und im Memory abgelegt.
Returns	(-9) Keine Verbindung zur Datenbank. (-2), (-3) Internes SQL-Statement ist falsch. (0) Bei keinem Fehler.
Notes	Mit den Methoden `GetFirst()` und `GetNext()` kann der Name der Formel aus der aktuellen Liste, gelesen werden.

5.5.3 FreeList()

Interface	`void FreeList()`
Parameters	–
Description	Diese Methode gibt den Speicher frei, der in der Funktion `CreateList()` alloziert worden ist.
Returns	-
Notes	Vorgängig muss die Funktion `CreateList()` aufgerufen worden sein.

5.5.4 GetFirst ()

Interface	`long GetFirst(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der erste Formelname in der Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Formelname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `CreateList()` muss vorgängig aufgefrufen worden sein.

5.5.5 GetNext ()

Interface	`long GetNext(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der nächste Name aus der Liste geschrieben wird.
Description	Mit dieser Methode wird jeweils der nächste Formelname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht.. (0) Bei keinem Fehler.
Notes	Die Methoden `CreateList()` und `GetFirst()` müssen vorgängig aufgefrufen worden sein.

5.6 Klasse SETFormula

Die Klasse *SETFormula* umfasst die Funktionalität, um Formeln abzufragen.

5.6.1 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID der Formel geschrieben wird.
Description	Diese Methode liefert die ID der Formel zurück.
Returns	(-3) Das Formel-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Formel-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.6.2 GetName()

Interface	`long GetName(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Name geschrieben wird.
Description	Mit dieser Methode kann der Name der Formel abgefragt werden.
Returns	(-3) Das Formel-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Formel-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.6.3 GetVariableNumber()

Interface	`long GetVariableNumber(long FAR* num)`
Parameters	`num` – Platzhalter, in welchen die Anzahl benötigten Variablen der Formel geschrieben wird.
Description	Mit dieser Methode kann die Anzahl benötigten Variablen der Formel ermittelt werden.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht. (0) Bei keinem Fehler.
Notes	Das Formel-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.6.4 Init()

Interface	`long Init(long pos)`
Parameters	`pos` – Positionsindex der Formel in der Liste
Description	Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (-3) Kein Objekt gefunden. (-4) Angegebene Position ist falsch. (-15) Die begonnene Aktion muss zuerst abgeschlossen werden. (0) Bei keinem Fehler.
Notes	Die Liste muss vorher mit `CreateList()` erzeugt worden sein.

5.7 Klasse SDDContainer

Die Klasse *SDDContainer* ist ein Container, in welchen die Entitäts-Objekte gespeichert werden. Es werden Methoden zur Verfügung gestellt, um diesen Container zu füllen, zu leeren oder abzufragen. Bei der Erzeugung der Liste von Entitäts-Objekten werden zusätzlich alle Attribut- und Beziehungs-Objekte im Memory abgelegt.

5.7.1 CreateList()

Interface	`long CreateList()`
Parameters	–
Description	Mit dieser Methode werden aus der Datenbank die Entitäts-, Attribute- und Beziehungs-Objekte gelesen und im Memory abgelegt.
Returns	(-9) Keine Verbindung zur Datenbank. (-2) Internes SQL-Statement ist falsch. (-3) Internes SQL-Statement ist falsch. (0) Bei keinem Fehler.
Notes	Mit den Methoden `GetFirst()` und `GetNext()` kann der Name der Entität aus der aktuellen Liste gelesen werden.

5.7.2 FreeList()

Interface	`void FreeList()`
Parameters	–
Description	Diese Methode gibt den Speicher frei, der in der Funktion `CreateList()` alloziert worden ist.
Returns	-
Notes	Vorgängig muss die Funktion `CreateList()` aufgerufen worden sein.

5.7.3 GetCount()

Interface	`long GetCount(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Listeneinträge
Description	Diese Funktion liefert die Anzahl der Entitäten zurück, die im Container gespeichert ist.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Entitäten kann mittels einem Loop mit der Funktion `GetFirst()` und `GetNext()` genutzt werden.

5.7.4 GetFirst()

Interface	`long GetFirst(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der erste Entitätsname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Entitätsname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `CreateList()` muss vorgängig aufgefrufen worden sein.

5.7.5 GetNext()

Interface	`long GetNext(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der Entitätsname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird jeweils der nächste Entitätsname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht.. (0) Bei keinem Fehler.
Notes	Die Methoden `CreateList()` und `GetFirst()` müssen vorgängig aufgefrufen worden sein.

5.8 Klasse SDDEntity

Die Klasse *SDDEntity* umfasst die Funktionalität, um Entitäten abzufragen.
Jedes Entitäts-Objekt ist selber ein Container in welchem die Attributs- und Beziehungs-Objekte verwaltet werden. Es werden zusätzlich Methoden zur Verfügung gestellt, um diese Container abzufragen.

5.8.1 GetFirstAttribute()

Interface	`long GetFirstAttribute(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der erste Attributsname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Attributsname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `SDDContainer.CreateList()` muss vorgängig aufgefrufen worden sein.

5.8.2 GetFirstRelation()

Interface	`long GetFirstRelation(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der erste Beziehungsname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird der erste Beziehungsname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-3) Kein Objekt gefunden. (0) Bei keinem Fehler.
Notes	Die Methode `SDDContainer.CreateList()` muss vorgängig aufgefrufen worden sein.

5.8.3 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID der Entität geschrieben wird.
Description	Diese Methode liefert die ID der Entität zurück.
Returns	(-3) Das Entitäts-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Entitäts-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.8.4 GetName()

Interface	`long GetName(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Name geschrieben wird.
Description	Mit dieser Methode kann der Name der Entität abgefragt werden.
Returns	(-3) Das Entitäts-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Entitäts-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.8.5 GetNextAttribute()

Interface	`long GetNextAttribute(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der Attributname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird jeweils der nächste Attributname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht.. (0) Bei keinem Fehler.
Notes	Die Methoden `SDDContainer.CreateList()` und `GetFirstAttribute()` müssen vorgängig aufgefrufen worden sein.

5.8.6 GetNextRelation()

Interface	`long GetNextRelation(BSTR FAR* name)`
Parameters	`name` – Platzhalter, in welchen der Beziehungsname aus der Liste geschrieben wird.
Description	Mit dieser Methode wird jeweils der nächste Beziehungsname aus der Liste zurückgegeben.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht.. (0) Bei keinem Fehler.
Notes	Die Methoden `SDDContainer.CreateList()` und `GetFirstRelation()` müssen vorgängig aufgefrufen worden sein.

5.8.7 GetNickname()

Interface	`long GetNickname(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Kurzname geschrieben wird.
Description	Mit dieser Methode kann der Kurzname der Entität abgefragt werden. Dieser Name wird bei der Kommunikation mit ***ArcView*** verwendet. Ebenso wird davon ausgegangen, dass die Datenbank-Business-Tabelle dieser Entität (logische Ebene) denselben Namen hat.
Returns	(-3) Das Entitäts-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Entitäts-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.8.8 GetNoOfAttributes()

Interface	`long GetNoOfAttributes(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Listeneinträge
Description	Diese Funktion liefert die Anzahl der Attribute pro Entität zurück, die im Container gespeichert ist.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Attributen kann mittels einem Loop mit der Funktion `GetFirstAttribute()` und `GetNextAttribute()` genutzt werden.

5.8.9 GetNoOfRelations()

Interface	`long GetNoOfRelations(long FAR* num)`
Parameters	`num` – Platzhalter, für die Anzahl der Listeneinträge
Description	Diese Funktion liefert die Anzahl der Beziehungen pro Entität zurück, die im Container gespeichert ist.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (0) Bei keinem Fehler.
Notes	Die Information der Anzahl von Beziehungen kann mittels einem Loop mit der Funktion `GetFirstRelation()` und `GetNextRelation()` genutzt werden.

5.8.10 Init()

Interface	`long Init(long pos)`
Parameters	`pos` – Positionsindex der Entität in der Liste
Description	Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (-3) Kein Objekt gefunden. (-4) Angegebene Position ist falsch. (0) Bei keinem Fehler.
Notes	Die Liste muss vorher mit `SDDContainer.CreateList()` erzeugt worden sein.

5.9 Klasse SDDAttribute

Die Klasse *SDDAttribute* umfasst die Funktionalität, um Entitätsattribute abzufragen.

5.9.1 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID des Attributs geschrieben wird.
Description	Diese Methode liefert die ID des Attributes zurück.
Returns	(-3) Das Attribut-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Attribut-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.9.2 GetName()

Interface	`long GetName(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Name geschrieben wird.
Description	Mit dieser Methode kann der Name des Attributs abgefragt werden.
Returns	(-3) Das Attribut-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Attribut-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.9.3 GetType()

Interface	`long GetType(BSTR FAR* num)`
Parameters	`num` – Platzhalter, in welchen der Attributtyp geschrieben wird.
Description	Mit dieser Methode kann der Attributtyp abgefragt werden. ('I' = Zahl, 'F' = Gleitkommazahl, 'C' = Text, 'D' = Datum)
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (1) Listenende erreicht. (0) Bei keinem Fehler.
Notes	Das Attribut-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.9.4 Init()

Interface	`long Init(long pos)`
Parameters	`pos` – Positionsindex des Attributs in der Liste
Description	Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (-3) Kein Objekt gefunden. (-4) Angegebene Position ist falsch. (0) Bei keinem Fehler.
Notes	Die Liste muss vorher mit `SDDContainer.CreateList()` erzeugt worden sein.

5.10 Klasse SDDRelation

Die Klasse *SDDRelation* umfasst die Funktionalität, um Beziehungen zwischen Entitäten abzufragen.

5.10.1 GetCardinality()

Interface	`long GetCardinality(long FAR* card)`
Parameters	`card` – Platzhalter, in welchen die Kardinalität geschrieben wird.
Description	Diese Methode liefert die Kardinalität der Beziehung, betrachtet von der Quell-Entität, zurück.
Returns	(-3) Das Beziehungs-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Beziehungs-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.10.2 GetEntityId()

Interface	`long GetEntityId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID der Entität geschrieben wird.
Description	Diese Methode liefert die ID der Ziel-Entität, der Beziehung zurück.
Returns	(-3) Das Beziehungs-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Beziehungs-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.10.3 GetId()

Interface	`long GetId(long FAR* id)`
Parameters	`id` – Platzhalter, in welchen die ID der Beziehung geschrieben wird.
Description	Diese Methode liefert die ID der Beziehung zurück.
Returns	(-3) Das Beziehungs-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Beziehungs-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.10.4 GetName()

Interface	`long GetName(BSTR FAR* value)`
Parameters	`value` – Platzhalter, in welchen der Name geschrieben wird.
Description	Mit dieser Methode kann der Name der Beziehung abgefragt werden.
Returns	(-3) Das Beziehungs-Objekt ist nicht initialisiert worden. (0) Bei keinem Fehler.
Notes	Das Beziehungs-Objekt muss vorher mit `Init()` initialisiert worden sein.

5.10.5 Init()

Interface	`long Init(long pos)`
Parameters	`pos` – Positionsindex der Beziehung in der Liste
Description	Diese Methode initialisiert, gemäss dem Positionsindex in der Liste, das Objekt.
Returns	(-1) Die Liste ist vorgängig nicht erzeugt worden. (-2) Die Liste ist leer. (-3) Kein Objekt gefunden. (-4) Angegebene Position ist falsch. (0) Bei keinem Fehler.
Notes	Die Liste muss vorher mit `SDDContainer.CreateList()` erzeugt worden sein.

6 Kommunikation zwischen ArcView und dem MengenManager

6.1 Allgemeines

ArcView ist DDE-Server (source) und der MengenManager ist DDE-Client (destination). Jede Aktion über DDE wird von dem MengenManager gestartet, weil dort die mengenbezogene Benutzerinteraktion ist. Der MengenManager steuert ***ArcView***, indem sie Avenue-Kommandos via DDE LinkRequest absetzt. Jeder Aufruf sollte eine komplette "*Transaktion*" enthalten, da sonst Inkonsistenzen auftreten können, wenn der Benutzer zwischen zwei "inhaltlich zusammenhängenden" DDE Aufrufen an den relevanten ***ArcView***-Zuständen ändert (z.B. am selection set).
Die einzige Aktion, bei der ***ArcView*** den MengenManager anstösst, ist das Öffnen bzw. das Aktivieren des MengenManagers (DDEClient.Execute in RBS.OpenSetPanel).
Als Entitätsnamen werden die Kurznamen verwendet, die auch im Datenmodell (Datenbank Business-Tabellen) vergeben wurden.
Der Transfer von Mengen findet in Form von dBase (*.dbf) Dateien statt. Die Übermittlung aller Mengeninhalte via DDE hat sich als zu langsam erwiesen. Für das Löschen der dBase-Dateien nach Gebrauch, ist grundsätzlich ***ArcView*** zuständig.

6.2 ArcView-seitige Implementierung

Die Zusammenfassung der Avenue-Scripts geschieht in einer Extension. Die beschriebenen Zustandsdaten werden beim Laden der Extension initialisiert und beim Entfernen der Extension gelöscht. Die vom MengenManager übergebenen dBase-Dateien werden mit der Feature-Tabelle verknüpft. Bei der wiederholten Übergabe wird die Verknüpfung aufgelöst, die alte dBase-Datei gelöscht und mit der neuen dBase-Datei wieder verknüpft
Es gibt einen fixen View (den "RBS-View"), auf den sich alle Operationen mit dem MengenManager beziehen.
Jeder Entität wird ein Thema, des RBS-Views, fix zugewiesen.
Es wird ein Dictionary mit Entitätsnamen und zugehörigen Themen angelegt. Zu jedem Thema gibt es eine extra VTab (die "Attribut-VTab") mit zugeladenen Attributen; sie ist nach Initialisierung undefiniert (`nil`). Die VTabs werden ebenfalls in einem Dictionary verzeichnet.

6.3 Funktionen

6.3.1 Menge erzeugen aus aktueller Objektselektion

Das Avenue Script *RBS.SaveSelection* schreibt die ID-Spalte und ggf. zuvor mit "Menge anzeigen" geladene Attributspalten der aktuellen Entitäts-FTab-Selection, in eine dBase-Datei. Falls die Datei aus einer früheren RBS.*-Operation noch vorhanden ist, wird sie überschrieben. Beim Verlassen der RBS Extension wird sie gelöscht.

Interface	`ret = av.Run( "RBS.SaveSelection", { } )`
Parameters	-
Description	siehe oben.
Returns	Eine komma-separierte Liste mit vier Elementen.

Kein Fehler:

```
>0,entity,path,file   # Elemente,
                      Kurzname der Entität,
                      Pfad,
                      Dateiname ohne Erweiterung.
```

Fehler:

```
0,entity,,            Keine Elemente selektiert.
-1,,,                 Kein Thema aktiv.
-2,,,                 Datei nicht geschrieben.
```

Notes

```
Globals:
    _RBS_View     (in)
    _RBS_FTabs    (in)
    _RBS_FileName (in/out)
```

6.3.2 Menge aktuell setzen

Das Avenue Script *RBS.SetSelection* liest die genannte dBase-Datei in die, der angegebenen Entität zugeordnete Attribut-VTab ein und stellt über die xx-ID eine Verknüpfung von der Entitäts-FTab zur geladenen Attribut-VTab her. In der Entitäts-FTab werden alle Einträge mit Attribut-VTab-Entsprechung markiert.

Interface

```
ret = av.Run( "RBS.SetSelection",
                          {"entity", "file" } )
```

Parameters

`entity` - Kurzname der Entität.
`file` - Eine dBase-Datei mit der vollständigen Angabe des Pfads inklusive der Datei-Erweiterung.

Description

siehe oben.

Returns

Kein Fehler:

```
>=0                    # Elemente markiert.
```

Fehler:

```
-1                     Flasche Entität erhalten.
-2                     Datei nicht lesbar.
-3                     Spalte xx-ID nicht gefunden.
```

Notes

```
Globals:
    _RBS_TTabs    (in)
    _RBS_VTabs    (in)
```

6.3.3 Menge blinken lassen

Das Avenue Script *RBS.FlashSet* lässt Geo-Objekte, anhand der ID-Spalte in der dBase-Datei, aufblinken und macht den "Blitz"-Button in ***ArcView*** verfügbar. Mit diesem Button kann das Blinken wiederholt werden. RBS.FlashSet löscht die übergebene dBase-Datei. Der Aufbau der blinkenden Geometrie in Avenue ist für sehr viele Objekte zu langwierig. Daher wird ab einem bestimmten Limit (abhängig von der Dimension) nur noch die bounding box aller Objekte angezeigt.

Interface	`ret = av.Run( "RBS.FlashSet",{"entity", "file" } )`
Parameters	`entity` - Kurzname der Entität. `file` - Eine dBase-Datei mit der vollständigen Angabe des Pfads inklusive der Datei-Erweiterung.
Description	siehe oben.

Returns	Kein Fehler:
	`>=0 # Elemente markiert.`
	
	Fehler:
	`-1 Flasche Entität erhalten.`
	`-2 Datei nicht lesbar.`
	`-3 Spalte xx-ID nicht gefunden.`

Notes	`Globals:`
	`_RBS_TTabs (in)`
	`_RBS_VTabs (in)`
	`_RBS_FlashShape (out)`

6.3.4 Geometrische Beziehung

Das Avenue Script *RBS.Overlay* ermittelt durch geometrische Überlagerung eine Beziehung jedes Elements aus der Quellmenge zu (0 oder) einem Element aus der Zielmenge. Die Ergebnismenge ist eine Beziehungsmenge, d.h. sie hat zwei ID-Spalten.
Die Zielmenge muss vom Typ "Polygon" sein. Bei Linie-in-Polygon und Polygon-in-Polygon Verschneidung können mehrdeutige (n:m) Beziehungen entstehen. In diesem Fall wird nur die Beziehung mit der grössten Überlagerung (Länge bzw. Fläche) übernommen.
RBS.Overlay löscht bei erfolgreichem Durchlauf die beiden übergebenen dBase-Dateien. Die zurückgegebene Resultatdatei wird beim nächsten RBS.*-Aufruf, der eine dBase-Datei erzeugt, überschrieben und beim Beenden der RBS Extension gelöscht.

Interface

```
ret = av.Run( "RBS.Overlay",
                {"src_entity", "src_file",
                 "dst_entity", "dst_file" } )
```

Parameters

`src_entity` - Kurzname der Quell-Entität.
`src_file` - Die Von-dBase-Datei mit der vollständigen Angabe des Pfads inklusive der Datei-Erweiterung.
`dst_entity` - Kurzname der Ziel-Entität.
`dst_file` - Die Nach-dBase-Datei mit der vollständigen Angabe des Pfads inklusive der Datei-Erweiterung.

Description

siehe oben.

Returns

Eine komma-separierte Liste mit drei Elementen.

Kein Fehler:

```
>0,path,file          # Elemente,
                      Pfad der Datei,
                      Dateiname ohne Erweiterung.
```

Fehler:

```
 0,,                  Keine Verschneidung möglich.
-1,,                  Quell-Entität falsch.
-2,,                  Quell-Datei nicht lesbar.
-3,,                  Quell-xx-ID nicht gefunden.
-4,,                  Ziel-Entität falsch.
-5,,                  Ziel-Datei nicht lesbar.
-6,,                  Ziel-xx-ID nicht gefunden.
-7,,                  Ziel-Entität kein Polygon.
-8,,                  Datei nicht gschrieben.
-9,,                  Fataler Fehler oder
                      Abbruch durch Anwender.
```

Notes

```
Globals:
    _RBS_TTabs      (in)
    _RBS_Filename   (in/out)
```

7 Installation

7.1 Voraussetzungen

Der MengenManager ist eine 32-Bit Anwendung und setzt das Betriebssystem Windows NT oder Windows 95 voraus. Entwickelt und getestet wurde die Anwendung auf dem RDBMS von ORACLE Version 7 mit dem Oracle7 32-Bit ODBC Treiber.
Weiter wird die ***ArcView*** - Version 3.0a in Deutsch oder Englisch benötigt.

7.2 Directory-Struktur

Der Directory-Baum ist wie folgt gegliedert:

```
./rbs_nnn
      +---  bin
      |       +---   ...
      |
      +---  data
      |       +---   shape
      |       |         +---      ...
      |       |         +---      ...
      |       +---   cover
      |                 +---      ...
      |                 +---      ...
      +---  install
      |       +---   SETContainer
      |       |         +---      ...
      |       |         +---      setup.exe
      |       |         +---      ...
      |       +---   database
      |       |         +---      dictionary
      |       |         |            +---   rbs_dm_v100.sql
      |       |         |            +---   rbs_set_v100.sql
      |       |         +---      insert
      |       |                      +---   DB-MetaDaten.mdb
      |       |                      +---   insert_attributes.sql
      |       |                      +---   insert_entity.sql
      |       |                      +---   insert_formula.sql
      |       |                      +---   insert_nls_attrs.sql
      |       |                      +---   insert_nls_entity.sql
      |       |                      +---   insert_nls_relation.sql
      |       |                      +---   insert_relation.sql
      |       |                      +---   insert_relation_side.sql
      |       +---   extension
      |                 +---      rbs.avx
      |
      +---  lib
              +---   ...
```

7.3 Datenbank

Die einheitliche Dokumentation des Semantic Data Dictionary (SDD) geschieht in einer relationalen Datenbank.

7.3.1 Tablespace

Der Dictionary wird in einem Tablespace abgelegt und sollte je nach Bedarf entsprechend dimensioniert werden.

Beispiel:

```
SQLPLUS> create tablespace SET_RBS
             datafile '<path>\<filename>.dbf' size 40M
             online;
```

Der Tablespace ist ein reservierter Speicherplatz in der Datenbank und sollte von Zeit zu Zeit überprüft werden, ob noch genügend Platz vorhanden ist.

7.3.2 Dictionary

Das Dictionary für das SDD wird mit SQL-Plus eingespielt.

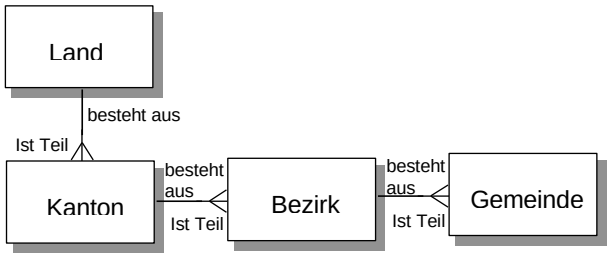
```
SQLPLUS> start rbs_dm_v100.sql
SQLPLUS> start rbs_dm_v100.sql
```

7.3.3 Inhalt

Als Eingabehilfe steht eine kleine MS-Access-Anwendung (DB-MetaDaten.mdb) zur Verfügung. Dabei können die nachfolgenden SQL-Dateien generiert werden.

Die SQL-Dateien im Directory 'Insert' sind als Beispiele zu verstehen. Dabei muss das semantische Datenmodell der Anwendung in Form dieser SQL-Dateien beschrieben werden.

Beispiel:



7.3.3.1 Entität

Datei *insert_nls_entity.sql:*

```
insert into rbs_nls_entities (E_ID,NLS_ID,NLS_NAME,TEXT)
            values (1,1,'Land','Beschreibung');
insert into rbs_nls_entities (E_ID,NLS_ID,NLS_NAME,TEXT)
            values (2,1,'Kanton','Beschreibung');
insert into rbs_nls_entities (E_ID,NLS_ID,NLS_NAME,TEXT)
            values (3,1,'Bezirk','Beschreibung');
insert into rbs_nls_entities (E_ID,NLS_ID,NLS_NAME,TEXT)
            values (4,1,'Gemeinde','Beschreibung');
```

Datei *insert_entity.sql:*

```
REM Land
insert into rbs_entities (E_ID,E_TYPE,O_TYPE,NICKNAME,NAME_RULE,RANGE_RULE,
N_MANDATORY,N_DEF_DRAW,N_DEF_DRKY,R_MANDATORY,R_DEF_DRAW,R_DEF_DRKY)
            values (1,'E','O','CH',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
REM Kanton
insert into rbs_entities (E_ID,E_TYPE,O_TYPE,NICKNAME,NAME_RULE,RANGE_RULE,
N_MANDATORY,N_DEF_DRAW,N_DEF_DRKY,R_MANDATORY,R_DEF_DRAW,R_DEF_DRKY)
            values (2,'E','O','KT', NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
REM Bezirk
insert into rbs_entities (E_ID,E_TYPE,O_TYPE,NICKNAME,NAME_RULE,RANGE_RULE,
N_MANDATORY,N_DEF_DRAW,N_DEF_DRKY,R_MANDATORY,R_DEF_DRAW,R_DEF_DRKY)
            values (3,'E','O','BZ', NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
REM Gemeinde
insert into rbs_entities (E_ID,E_TYPE,O_TYPE,NICKNAME,NAME_RULE,RANGE_RULE,
N_MANDATORY,N_DEF_DRAW,N_DEF_DRKY,R_MANDATORY,R_DEF_DRAW,R_DEF_DRKY)
            values (4,'E','O','GD', NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
```

7.3.3.2 Attribut

Datei *insert_nls_attrs.sql:*

```
REM  Land
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (1,1,'Objektname','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (2,1,'Fläche','Beschreibung');
REM  Kanton
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (3,1,'Objektname','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (4,1,'Flaeche','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (5,1,'Land','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (6,1,'Bezeichnung','Beschreibung');
REM  Bezirk
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (7,1,'Objektname','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (8,1,'Fläche','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (9,1,'Kanton','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (10,1,'Bezeichnung','Beschreibung');
REM  Gemeinde
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (11,1,'Objektname','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (12,1,'Fläche','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (13,1,'Bezirk','Beschreibung');
insert into rbs_nls_attrs (A_ID,NLS_ID,NLS_NAME,TEXT)
            values (14,1,'Bezeichnung','Beschreibung');
```

Datei *insert_ attributes.sql:*

```
REM Land
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (1,1,'G','C',1,0,'','T','F',NULL,'OBJEKTNAME');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (1,2,'G','F',8,3,'','T','F',NULL,'FLAECHE');
REM Kanton
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (2,3,'G','C',2,0,'','T','F',NULL,'OBJEKTNAME');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (2,4,'G','F',8,3,'','T','F',NULL,'FLAECHE');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (2,5,'G','I',1,0,'','T','F',NULL,'CH_ID');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (2,6,'G','C',25,0,'','T','F',NULL,'NAME');
REM Bezirk
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (3,7,'G','C',4,0,'','T','F',NULL,'OBJEKTNAME');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (3,8,'G','F',8,3,'','T','F',NULL,'FLAECHE');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (3,9,'G','I',2,0,'','T','F',NULL,'KT_ID');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (3,10,'G','C',50,0,'','T','F',NULL,'NAME');
REM Gemeinde
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (4,11,'G','C',4,0,'','T','F',NULL,'OBJEKTNAME');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (4,12,'G','F',8,3,'','T','F',NULL,'FLAECHE');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (4,13,'G','I',4,0,'','T','F',NULL,'BZ_ID');
insert into rbs_attributes (E_ID,A_ID,ATTR_TYPE,DATA_TYPE,PRECISION,SCALE,
            RULE,MANDATORY,DEF_DRAW,DEF_DRKY,ARC_ATTR)
            values (4,14,'G','C',50,0,'','T','F',NULL,'NAME');
```

7.3.3.3 Beziehung

Datei *insert_nls_relation.sql:*

```
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (1,1,'besteht aus','Beschreibung');
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (2,1,'ist Teil','Beschreibung');
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (3,1,'besteht aus','Beschreibung');
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (4,1,'ist Teil','Beschreibung');
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (5,1,'besteht aus','Beschreibung');
insert into rbs_nls_relation (REL_SIDE_ID,NLS_ID,NLS_NAME,TEXT)
            values (6,1,'ist Teil','Beschreibung');
```

Datei *insert_relation_side.sql:*

```
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (1,1,2,'I',999,'T',1);
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (2,2,1,'R',1,'T',1);
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (3,2,3,'I',999,'T',2);
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (4,3,2,'R',1,'T',2);
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (5,3,4,'I',999,'T',3);
insert into rbs_relation_side (REL_SIDE_ID,RS_ENTITY_1,RS_ENTITY_2,
            REFERENCE_TYPE,CARDINALITY,MANDATORY,R_ID)
            values (6,4,3,'R',1,'T',3);
```

Datei *insert_relation.sql:*

```
insert into rbs_relation (R_ID,REL_TYPE,ATTRIBUTE,NAME_RULE,
            RANGE_RULE,REL_CLASS) values (1,'O',5,'','','');
insert into rbs_relation (R_ID,REL_TYPE,ATTRIBUTE,NAME_RULE,
            RANGE_RULE,REL_CLASS) values (2,'O',9,'','','');
insert into rbs_relation (R_ID,REL_TYPE,ATTRIBUTE,NAME_RULE,
            RANGE_RULE,REL_CLASS) values (3,'O',13,'','','');
```

7.3.3.4 Formel

Die Formeln können beliebig in der nachfolgenden Syntax definiert werden.

Datei *insert_formula.sql:*

```
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(1,'Addition','a1 + a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(2,'Addition','a1 + a2 + a3','N',NULL,3,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(3,'Addition','a1 + a2 + a3 + a4','N',NULL,4,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(4,'Addition','a1 + a2 + a3 + a4 + a5','N',NULL,5,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(5,'Addition','a1 + a2 + a3 + a4 + a5 + a6','N',NULL,6,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(6,'Addition',
            'a1 + a2 + a3 + a4 + a5 + a6 + a7','N',NULL,7,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(7,'Addition',
            'a1 + a2 + a3 + a4 + a5 + a6 + a7 + a8','N',NULL,8,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(8,'Addition',
            'a1 + a2 + a3 + a4 + a5 + a6 + a7 + a8 + a9','N',NULL,9,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(9,'Addition',
            'a01 + a02 + a03 + a04 + a05 + a06 + a07 + a08 + a09 + a10',
            'N',NULL,10,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(10,'Subtraktion','a1 - a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(11,'Multiplikation','a1 * a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(12,'Division','a1 / a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(13,'Prozent','a1 * 100 / a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(14,'Dichte (pro m2)','a1 / a2','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(15,'Dichte (pro ha)','a1 / ( a2 / 10000 )','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(16,'Dichte (pro km2)',
            'a1 / ( a2 / 1000000 )','N',NULL,2,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(17,'Multiplikation','a1 * a2 * a3','N',NULL,3,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(18,'Division','a1 / a2 / a3','N',NULL,3,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(19,'Wurzel','SQRT(a1)','N',NULL,1,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(20,'Subtraktion','a1 - a2 - a3','N',NULL,3,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(21,'Subtraktion','a1 - a2 - a3 - a4','N',NULL,4,0);
insert into rbs_formula (F_ID,F_NAME,F_FORMULA,F_TYPE,F_DESC,F_VARS,F_CONSTS)
            values(22,'Subtraktion','a1 - a2 - a3 - a4 - a5','N',NULL,5,0);
```

7.4 Programm

Die Anwendung besteht aus einer EXE- und einer OCX-Datei. Wobei beide Teile installiert bzw. registriert werden müssen.

7.4.1 Setup

Das Anwendung *SETContainer* wird standard mässig mit setup installiert.

```
...\rbs_nn\install\SETContainer\setup.exe
```

7.4.2 Registrieren

Das Registrieren der OCX-Datei geschieht mit dem Programm regsvr32:

```
regsvr32 d:\winnt\system32\setcontainer.ocx
```

7.5 Umgebungsvariablen

Für jeden Benutzer müssen nachfolgende Umgebungsvariabeln definiert werden:

```
RBSHOME D:\rbs_nnn
RBSIMPORT D:\Temp
Path %Path%;%RBSHOME%\bin
```

7.6 Extension

Wird das Projekt rbs.apr verändert, so muss wie unter 7.6.1 beschrieben vorgegenagen werden. Für eine reine Installation genügt das Kopieren der Extension .

7.6.1 Erzeugen der ArcView Extension rbs.avx

1. Überflüssige Table Docs, Graphics etc. entfernen.
2. Kontrollieren, ob alle Scripts kompiliert sind.
3. Script MakeRBS starten (darin steht hart codiert der Pfad des erzeugten rbs.avx).
4. In rbs.avx mittels ASCII-Editor alle "c:/rbs_nn" ersetzen durch "$RBSHOME".

Weitere Hinweise für den Entwickler:

- Die Selection Color wird in InstallRBS eingestellt (um Unterscheidbarkeit zu gelben Flächenthemen zu sichern).
- Die Extension enthält das Script RBS.Test, welches zum Testen modifiziert werden kann.
- Die duch die Extension hinzugefügten Buttons werden durch RBS...Enable Scripts in ihrer Verfügbarkeit kontrolliert.
- Das Script RBS.About liefert eine about-box (ist im Hilfe-Menu des View-Doc installiert).
- Die Klasse DDEClient liefert Fehlermeldungen nur in Textform (GetErrorMsg). Dadurch ist die Fehlerbehandlung vom installierten ArcView NLS supplement abhängig. Workaround: Abfragen aller möglichen Meldungen mit "...or..." (siehe RBS.OpenSetPanel)

7.6.2 Installation der ArcView Extension rbs.avx

Die Extension in das ArcView - Extension - Verzeichnis kopieren.

```
copy rbs.avx ...\Arcview\Ext32\
```

Anhang A: Tabellenstruktur des Semantic Data Dictionary

I. Entität

Tabelle RBS_ENTITIES:

Spalte	Datentyp	Null?	Beschreibung
e_id	number	not null	ID der Entität.
e_type	varchar2(1)	not null	Typ der Entität: 'G' für Gruppe 'M' für Member einer Gruppe 'E' für Entität ohne Gruppe
o_type	varchar2(1)		(wird nicht verwendet)
nickname	varchar2(2)	not null	Kurzname der Entität.
name_rule	varchar2(32)		(wird nicht verwendet)
range_rule	varchar2(32)		(wird nicht verwendet)
n_mandatory	varchar2(1)		(wird nicht verwendet)
n_def_draw	varchar2(1)		(wird nicht verwendet)
n_def_drky	number		(wird nicht verwendet)
r_mandatory	varchar2(1)		(wird nicht verwendet)
r_def_draw	varchar2(1)		(wird nicht verwendet)
r_def_drky	number		(wird nicht verwendet)

Tabelle RBS_NLS_ENTITIES:

Spalte	Datentyp	Null?	Beschreibung
e_id	number	not null	ID der Entität.
nls_id	number	not null	ID der Sprache.
nls_name	varchar2(32)	not null	Name der Entität in der jeweiligen Landessprache.
text	varchar2(80)	not null	Erläuternder Text zu der Entität in der jeweiligen Landesprache.

II. Beziehung

Tabelle RBS_RELATION:

Spalte	Datentyp	Null?	Beschreibung
r_id	number	not null	ID der Relation.
rel_type	varchar2(1)	not null	Relationstyp: 'H' für hierarchische Beziehung 'N' für Beziehung über Objektnamen 'O' für Beziehung über Objekt-ID
attribute	number		ID des Attributes.
name_rule	varchar2(32)		(wird nicht verwendet)
range_rule	varchar2(32)		(wird nicht verwendet)
rel_class	varchar2(3)	not null	

III. Beziehungshälfte

Tabelle RBS_RELATION_SIDE:

Spalte	Datentyp	Null?	Beschreibung
rel_side_id	number	not null	ID der Beziehungshälfte.
rs_entity_1	number	not null	ID der einen Entität.
rs_entity_2	number	not null	ID der anderen Entität.
reference_type	varchar2(1)	not null	Typ der Beziehung zwischen beiden Entitäten: 'R' = Entität 1 referenziert Entität 2 'I' = Entität 1 wird von Entität 2 referenziert
cardinality	number	not null	Kardinalität der Beziehung: 1 = z.B. Eine Kante kann nur an einem Knoten beginnen 999 = unbegrenzt z.B. An einem Knoten können beliebig viele Kanten beginnen
mandatory	varchar2(1)	not null	(wird nicht verwendet)
r_id	number		ID der Relation. Zwei Beziehungshälften haben jeweils die gleiche r_id.

Tabelle RBS_NLS_RELATION:

Spalte	Datentyp	Null?	Beschreibung
rel_side_id	number	not null	ID der Beziehungshälfte.
nls_id	number	not null	ID der Sprache.
nls_name	varchar2(32)	not null	Name des Beziehungsworts in der jeweiligen Landessprache.
text	varchar2(80)	not null	Erläuternder Text zu der Entität in der jeweiligen Landesprache.

IV. Attribut

Tabelle RBS_ATTRIBUTES:

Spalte	Datentyp	Null?	Beschreibung
e_id	number	not null	ID der Entität.
a_id	number	not null	ID des Attributes (Laufnummmer).
attr_type	varchar2(1)	not null	(wird nicht verwendet)
data_type	varchar2(1)	not null	Datentyp des Attributes: 'C' für Text-Attribute 'D' für Datum-Attribute 'F' für Gleitkommazahl-Attribute 'I' für Zahl-Attribute
precision	number		Anzahl aller Dezimalstellen bzw. Anzahl der Buchstaben.
scale	number		Anzahl Nachkommastellen.
rule	varchar2(32)		(wird nicht verwendet)
mandatory	varchar2(1)	not null	(wird nicht verwendet)
def_draw	varchar2(1)	not null	(wird nicht verwendet)
def_drky	number		(wird nicht verwendet)
arc_attr	varchar2(64)	not null	Kolonnenname des Attributes auf der log. Ebene.

Tabelle RBS_NLS_ATTRS:

Spalte	Datentyp	Null?	Beschreibung
a_id	number	not null	ID des Attributs
nls_id	number	not null	ID der Sprache
nls_name	varchar2(32)	not null	Name des Attributs in der jeweiligen Landessprache
text	varchar2(80)	not null	Erläuternder Text zu dem Attribut in der jeweiligen Landesprache

V. Sprache

Tabelle RBS_NLS_LANGS:

Spalte	Datentyp	Null?	Beschreibung
nls_id	number	not null	ID der Sprache: 1 = 'GERMAN' 2 = 'DANISH' 3 = 'DUTCH' 4 = 'FINNISH' 5 = 'FRENCH' 6 = 'AMERICAN' 7 = 'HUNGARIAN' 8 = 'NORWEGIAN' 9 = 'RUSSIAN' 10 = 'SWEDISH'
language	varchar2(32)	not null	Name der Sprache.

VI. Menge

Tabelle RBS_SETS:

Spalte	Datentyp	Null?	Beschreibung
s_id	number	not null	ID der Menge.
s_name	varchar2(16)	not null	Name der Menge.
s_owner	varchar2(16)	not null	Besitzer der Menge.
s_scope	number	not null	Besitzstand: 1 = Allgemein 2 = Privat
s_type	number	not null	Typ der Menge 1 = Objektmenge ohne Sachdaten 2 = Objektmenge mit Sachdaten 3 = Beziehungsmenge
s_project	number	not null	(wird nicht verwendet)
s_entity	number	not null	ID der Entität.
s_count	number	not null	Antahl Elemente in der Menge.
s_status	number	not null	Status der Menge: 1 = Initialisiert 2 = Referenziert durch Ids 3 = Referenziert durch Objektname 4 = Ref. durch ID und Objektname
s_date	date	not null	Erstellungsdatum
s_desc	varchar2(200)		Beschreibung
s_entity_2	number		Im Falle einer Beziehungsmenge: ID der Entität, zu der die zu der Menge in Beziehung stehenden Objekte gehören. Sonst: -1
s_bez_id	number		(wird nicht verwendet)
s_in_use	number		Flag, welches anzeigt, ob die Menge in Bearbeitung ist.

VII. Mengenattribut

Tabelle RBS_SDT_DEFS:

Spalte	Datentyp	Null?	Beschreibung
s_id	number		ID der Menge.
k_def_nr	number		Die Attribute einer Menge werden jeweils mit 1 beginnend durchnummeriert. k_def_nr enthält die laufende Nummer. Das Attribut mit der Nr. 1 hat für die eigentliche Verwaltung keine Bedeutung (-> Datenschnittstelle). Die Attribute der SET-Tabelle beginnen mit der Nr. 2, wobei die Spaltennamen folgendermassen gebildet werden: k_def_nr = 2 -> W_COL_1 k_def_nr = 3 -> W_COL_2 usw.
k_col_type	varchar2(1)		k_col_type hat für die Verwaltung keine Bedeutung (-> Datenschnittstelle). Das Attribut mit k_def_nr = 1 enthält ein 'V' für 'Vorspalte', alle anderen enthalten ein 'W' für 'Wertespalte'.
k_col_name	varchar2(47)		(wird nicht verwendet)
k_col_id	varchar2(12)		unbenutzt (-> Datenschnittstelle).
k_col_format	varchar2(4)		Ausgabeformat (-> Datenschnittstelle)
name	varchar2(32)		Benuzerdefinierter Attributame.
data_type	varchar2(1)		Datentyp des Attributes: 'C' für Text-Attribute 'D' für Datum-Attribute 'F' für Gleitkommazahl-Attribute 'I' für Zahl-Attribute
data_length	number		Max. Länge des Attributes ('C').
data_precision	number		Anzahl Dezimalstellen ('F', 'I').
data_scale	number		Anzahl Nachkommastellen ('I').
k_desc	varchar2(200)		Beschreibung.

Kommentar [.1]:

Tabelle RBS_SDT_HDRS:

Spalte	Datentyp	Null?	Beschreibung
s_id	number		ID der Menge.
k_sdt_title	varchar2(72)		Titel der Listen.
k_headers	number		Anzahl Attribute inkl. 'Vorspalte'.
k_entries	number		Anzahl Elemente.

VIII. Formel

Tabelle RBS_FORMULA:

Spalte	Datentyp	Null?	Beschreibung
f_id	number	not null	ID der Formel.
f_name	varchar2(32)	not null	Name der Formel.
f_formula	varchar2(80)	not null	Formel (z.B. a1 * a2)
f_type	varchar2(1)	not null	= N (Number)
f_desc	varchar2(320)		Beschreibung.
f_vars	number		Anzahl Variabeln.
f_consts	number		Anzahl Konstanten.

Anhang B: Anwendungsbeispiel des Steuerelements *SETContainer*

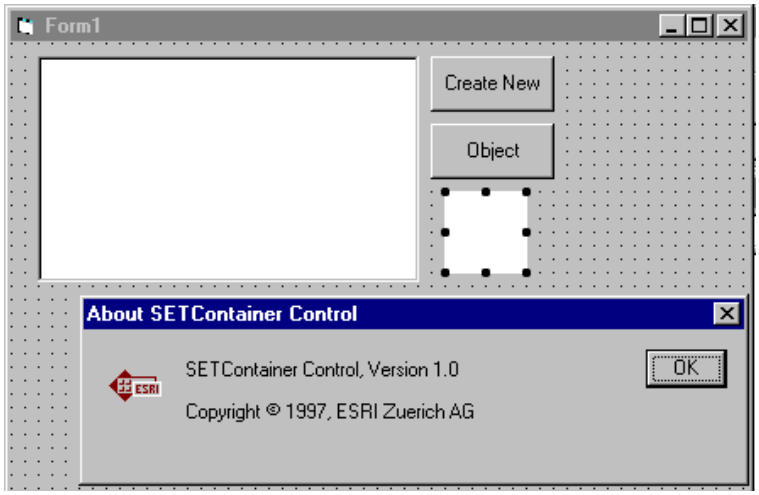
Dieses Anwendungsbeispiel basiert auf Visual Basic Version 4.0.

• OCX-Einbinden

Steuerelement
für die Objekt-
mengenverwaltung

- Das OCX muss in der Registry von Windows-NT registriert sein.
- In Visual Basic unter dem Menüpunkt
  Tools ->
  Custom Controls ...
  das Steuerelement ***SETContainer*** einbinden.
- Aktivieren des Steuerelementes  in der Tool-Box und in ein Formular einfügen. Somit ist ein Objekt der Klasse SETContainer instanziert worden. Darauf können nun alle Methoden, wie in der Dokumentation OLE-Automation beschrieben, angewendet werden.

• Die Oberfläche erstellen



- Button mit Click-Event
  cmdCreateNew_click()
- Button mit Click-Event
  cmdObject_click()
- Unsichtbares Steuerelement
  SETContainer
- About-Box des Steuer-
  elementes SETContainer

• Die Click-Events

1. Eine Liste erzeugen

In der nachfolgenden Funktion werden alle Mengen, die in der Datenbank eingetragen sind, in einer ListBox dargestellt.

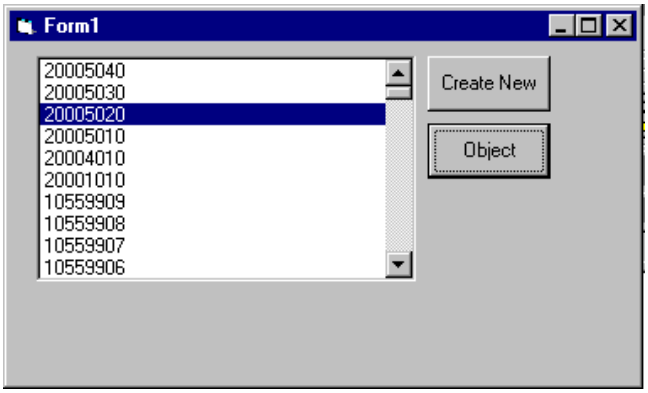
```
Private Sub cmdCreateNew_Click()
    Dim ret As Long
    Dim num As Long
    Dim name As String
    Dim mlg As String

    ' Liste erzeugen
        ret = SETContainer1.CreateSetList()

    ' Test auf den Returnwert
    if ret <> 0 Then
        MsgBox "FEHLER: Liste nicht erzeugt!"
        Exit Sub
    End If

    ' Einträge in der ListBox löschen
    List1.Clear

    ' Anzahl der Mengen in der Datenbank ermitteln
    ret = SETContainer1.GetCount(num)

    ' Test auf den Returnwert
    if ret <> 0 Then
        MsgBox "FEHLER: Liste ist leer!"
        Exit Sub
    End If

    ' Ersten Mengenname in der Liste abfragen
    ret = SETContainer1.GetFirst(name)

    ' Test auf den Returnwert
    If ret = 0 Then
        ' Den Mengenname in der ListBox eintragen
        List1.AddItem name
        For i = 0 To num
            ' Den nächsten Mengenname in der Liste ermitteln
            ret = SETContainer1.GetNext(name)
            ' Test auf den Returnwert
            If ret <> 0 Then
                If ret <> 1 Then
                    MsgBox "FEHLER: In GetNext-Funktion!"
                End If
                ' End of list
                Exit For
            End If
            ' Den Mengenname in der ListBox eintragen
            List1.AddItem name
        Next i
    Else
        mlg = "ERROR in GetFirst-Funktion " & "(" & ret & ")"
        MsgBox mlg
    End If
End Sub
```

2. Ein Objekt in der Liste abfragen

In dieser Funktion wird ein SETSet-Objekt anhand der aktuellen Position in der Liste initialisiert und abgefragt.

```
Private Sub cmdObject_Click()
    Dim obj As New SETSet
    Dim ret As Long
    Dim num As Long
    Dim id As Long
    Dim name As String
    Dim mlg As String

    ' Positionindex in der ListBox ermitteln
    num = List1.ListIndex

    ' Das Interface-Object anhand der aktuellen Position in der Liste
    ' initialisieren
    ret = obj.Init(num)
    If ret = 0 Then
        ' Informationen vom SET-Objekt holen
        ret = obj.GetName(name)
            ret = obj.GetId(id)
        mlg = "ID = " & id & ", Mengenname = " & name
        MsgBox mlg
    Else
        MsgBox "Init failed !"
    End If
End Sub
```

• Die Oberfläche



Anhang C: Test von DDE als Verbindung Avenue-Visual Basic

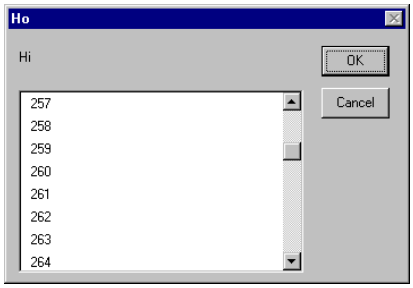
• Visual Basic ➔ Avenue

Starte Visual Basic
Erzeuge TextBox 'Text1' und folgenden click-event-handler für das Form:

```
Private Sub Form_Click()
    Dim Cmd, Z, I   ' Declare variables.
    Q = Chr(34)     ' Define quotation marks.

    If Text1.LinkMode = vbNone Then
       'Z = Shell("c:\ESRI\AV_GIS30\ARCVIEW\BIN32\ARCVIEW.EXE", 4)
        Text1.LinkTimeout = 6000
        Text1.LinkTopic = "ARCVIEW|System"    ' Set link topic.
        Text1.LinkMode = vbLinkManual         ' Set link mode.
    End If

    ' Create a global Avenue variable _list and init it to <empty list>
    Cmd = "_list = List.Make"
    Text1.LinkItem = Cmd
    Text1.LinkRequest

    ' Add elements to _list
    For I = 1 To 10000
        Cmd = "_list.Add(" & I & ")"
        Text1.LinkItem = Cmd
        Text1.LinkRequest
    Next I

    ' Display _list in a MsgBox
    Cmd = "MsgBox.ListAsString(_list," & Q & "Hi" & Q & "," & Q & "Ho" & Q & ")"
    Text1.LinkItem = Cmd
    Text1.LinkRequest

    MsgBox "LinkExecute DDE demo with ESRI ArcView finished.", 64
   End
End Sub
```

Starte ArcView
Starte das VB-Programm und clicke in das Form => es werden 10000 Zahlen nach Avenue geschickt und dort in einer List gesammelt. Diese wird in einer MsgBox angezeigt.
Dauer: ca. **30sec**
Zum Vergleich das äquivalente Avenue-Script (läuft ca. **3sec**):

```
_list = List.Make

for each i in 1..10000
  _list.Add(i)
end

MsgBox.ListAsString(_list,"Hi","Ho")
```



- **Avenue ➔ VisualBasic**

Zum obigen Visual Basic Projekt füge folgenden Handler hinzu:

```
Private Sub Form_LinkExecute(CmdStr As String, Cancel As Integer)
    Cancel = False
    If Val(CmdStr) = 1 Then Print "begin"
    If Val(CmdStr) = 10000 Then Print "end"
End Sub
```

Setze in den Properties - Form1 den Eintrag LinkMode auf „1 - Source“

Mache aus dem Visual Basic Projekt ein EXE namens Project1.exe

Starte Project1.exe

Führe in ArcView folgendes Avenue-Script aus:

```
_ddec = DDEClient.Make("project1","form1")

if( _ddec.hasError ) then
      MsgBox.Info(_ddec.GetErrorMsg, "DDEClient.Make error:")
end
for each i in 1..10000
      _ddec.Execute(i.AsString)
end

_ddec.Close
```

Es wird eine DDE-Verbindung zum Form1 der Applikation Project1 aufgebeut und 10000 Zahlen geschickt. Im Visual Basic Form erscheint „begin“ und „end“.
Dauer: ca. **17sec**
Zum Vergleich ein äquivalentes VisualBasic Programm (läuft ca. **2sec**):

```
Private Sub Command1_Click()
    Dim C As Integer
    For I = 1 To 10000
        Form_LinkExecute Str(I), C
    Next I
End Sub
```