Köln RBS - Konzept Metadaten
(Diskussionsgrundlage)

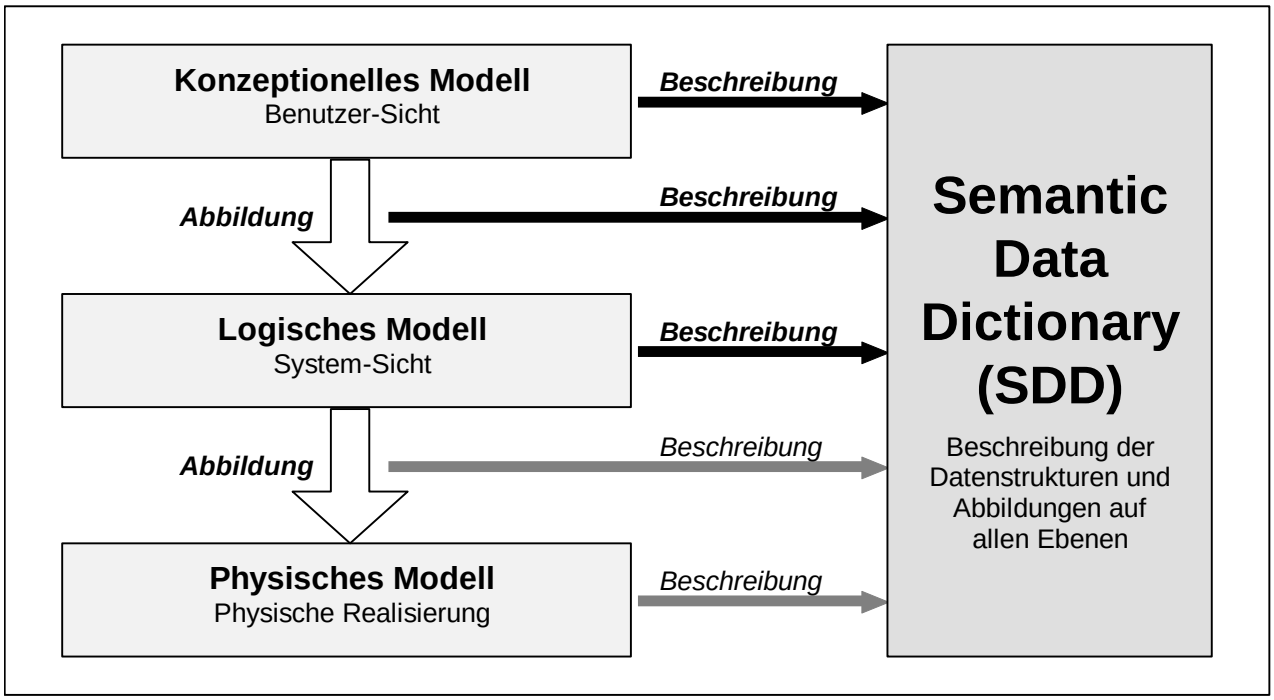
1 Übersicht und Begriffsbildung

Inhalt des vorliegenden Papiers ist ein Konzept zur Speicherung und Verwendung von Daten, welche Struktur und Inhalte einer Geodatenbank beschreiben. Zweck ist die Verwendung dieser beschreibenden Daten in der Datenverwaltung und in allgemein einsetzbaren Anwendungsprogrammen.

Daten, welche die *Inhalte* einer (Geo-)Datenbank beschreiben, bezeichnet man traditionell als *Metadaten*. Daten, welche die *Struktur* einer (Geo-)Datenbank beschreiben, bezeichnet man weithin als *Data Dictionary*. Die *anwendungsbezogene* Strukturierung einer Datenbankanwendung bezeichnet man als *semantisches Modellieren*.

Wir versuchen, beide Ansätze miteinander zu verbinden sowie die üblicherweise auf der logischen Ebene verhafteten Strukturdaten auf die konzeptionelle (=Anwendungs-) Ebene zu heben und um Anwendungskonzepte (Darstellung, Verfahren etc.) zu erweitern. Bisher wurde dafür in Köln auf der RBS-Seite der Begriff *Metadata Dictionary* verwendet, der strenggenommen nicht ganz exakt ist (es handelt sich ja nicht um eine strukturelle Beschreibung der Metadaten, sondern um eine strukturelle *und konzeptionelle* Beschreibung der *Geo*daten). Auf SIS-Seite steht begrifflich die Beschreibung der *Informationsobjekte* gegenüber, die gelegentlich einfach als Metadaten bezeichnet wird.

Vorschlag: Man könnte den neuen Begriff **Semantic Data Dictionary (SDD)** verwenden als Kombination von Data Dictionary auf der konzeptioneller Ebene, Metadaten und Erweiterungen um Anwendungsaspekte.



Das Semantic Data Dictionary integriert folgende Bereiche in einer durchgängigen Struktur:

- **E/R-Modellierung der Anwendungsdaten:** Die Datenmodellierung auf der konzeptionellen Ebene ist der Kern der strukturellen Beschreibung einer Anwendung. Die Ergebnisse der Modellierung auf der konzeptionellen sowie auf der logischen Ebene werden im SDD dokumentiert.
- **Zugriffskontrolle:** Es wird ein Benutzer-/Gruppen-Konzept eingeführt, so dass ein kontrollierter Zugriff auf die Geo-Daten und deren Bestandteile erfolgt. Dabei werden unterschiedliche Schutzmechanismen unterstützt (RDBMS, file system, etc.).

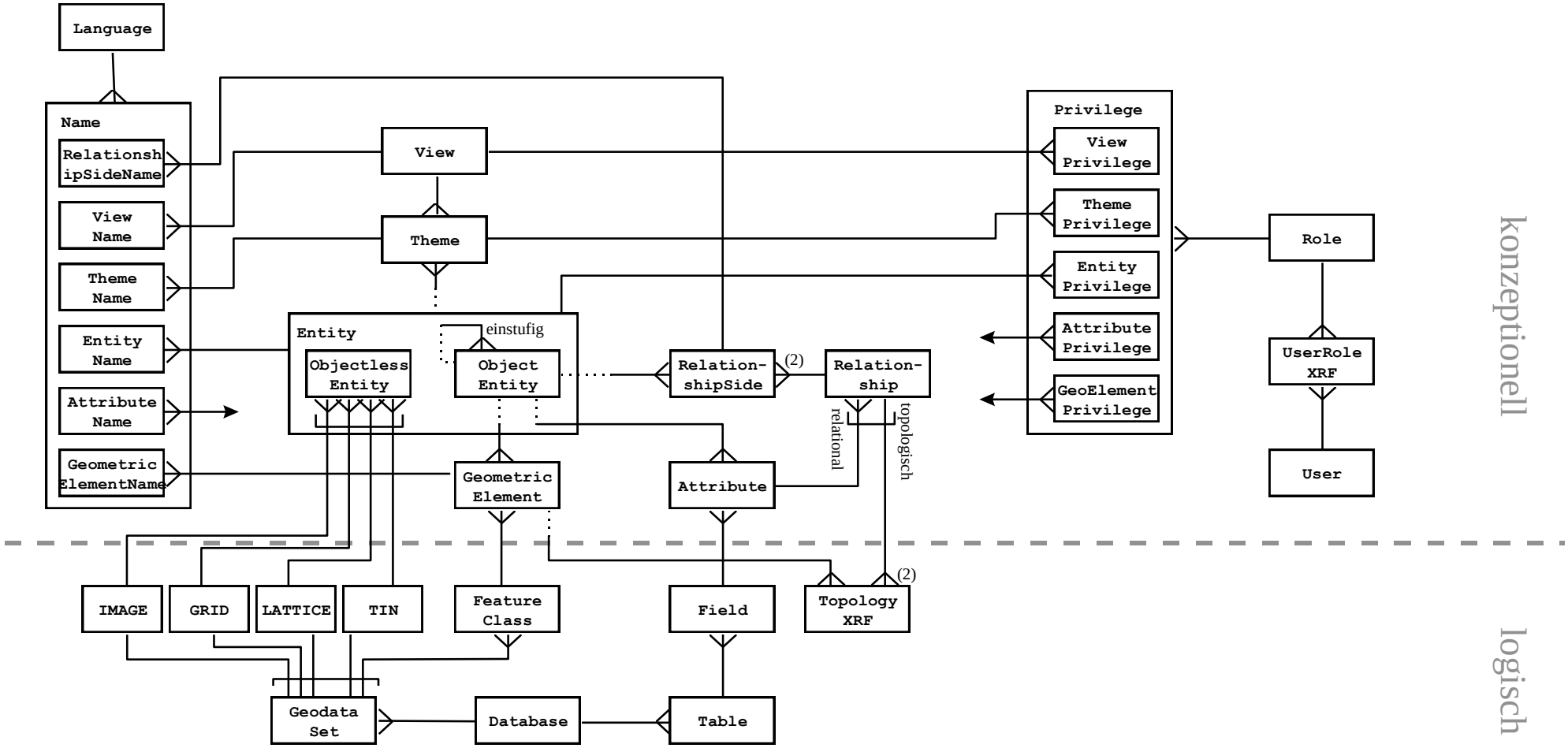
- **National Language Support NLS:** Die interne Sprachregelung bleibt dem Benutzer verborgen. Denn die Kommunikation sollte in der Begriffswelt des Anwender erfolgen. Dabei werden mehrerer nationale Sprachen unterstützt.
- **Darstellung:** Es können verschiedenen Darstellungsschichten gebildet und zusammengefasst werden. Somit kann jeder Anwender seine Darstellung unabhängig vom Datenmodell definieren (analog ArcView).
- **Analyse:** Zukünftig wird das SDD um die Verwaltungskomponente von Abfragen und Mengen erweitert. Benutzerdefinierte Abfragen sind Methoden die in Form von Parametern gespeichert werden. Dabei wird die sachliche Restriktion von der geometrischen Restriktion getrennt. Mengen sind Analyse-Resultate, die permanent gespeichert werden. Dabei wird zwischen Objekt-, Sachdaten- und Beziehungsmenge unterschieden.

2 Semantic Data Dictionary

Es folgen "Meta-Meta-Daten", d.h. wir spezifizieren die Struktur derjenigen Daten, welche wiederum Struktur und Inhalt der Anwendungsdatenbank beschreiben.

2.1 E/R-Diagramm

Das folgende E/R-Diagramm beschreibt das SDD auf konzeptioneller Ebene und enthält Ansätze einer Abbildung in die logische, ESRI-GIS-übergreifende Ebene. Die Abbildung dieses "ESRI-Esperanto" auf die einzelnen Softwaresysteme ARC/INFO, SDE, etc. leitet sich weitgehend aus den entsprechenden Abstraktionen in ADF, ArcProjekt, Avenue und MapObjects ab.


Language
Name
RelationshipSideName
View Name
Theme Name
Entity Name
Attribute Name
Geometric ElementName
View
Theme
Entity
einstufig
Objectless Entity
Object Entity
Relation-shipSide
(2)
Relation-ship
relational
topologisch
Geometric Element
Attribute
Privilege
View Privilege
Theme Privilege
Entity Privilege
Attribute Privilege
GeoElement Privilege
Role
UserRole XRF
User
konzeptionell
IMAGE
GRID
LATTICE
TIN
Feature Class
Field
(2)
Topology XRF
Geodata Set
Database
Table
logisch

2.2 Modellierung des Anwendungs-E/R-Diagramms

2.2.1 Entity

Beschreibung

Direkte Entsprechung zur Entity des *Anwendungs*-E/R-Diagramms. Eine **Entity**-Instanz entspricht einer Objekt*klasse* der realen oder gedachten Welt, z.B. Garagen, Silos, Waldflächen etc. (siehe **ObjectEntity**) oder einer objektlosen Datenschicht, z.B. gescannte Karte, Geländemodell (siehe **ObjectlessEntity**)

Für die Darstellung ist eine **Entity** die Basis für ein **Theme**.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	ENT_ID	integer	Identifikator
F	C_USR_ID	integer	Erzeugt durch (User)
	C_TIME	timestamp	Zeitpunkt der Erzeugung
F	M_USR_ID	integer	Letzte Veränderung durch (User)
	M_TIME	timestamp	Zeitpunkt der letzten Veränderung

Erläuterungen

Diskussionspunkte

2.2.1.1 ObjectEntity

Beschreibung

Eine **ObjectEntity**-Instanz entspricht einer Objekt*klasse* der realen oder gedachten Welt, z.B. Garagen, Silos, Waldflächen etc.

Alle Instanzen einer Anwendungs-Entity haben

- einen eindeutigen Identifikator
- dieselbe Attributmenge
- dieselben Beziehungen zu anderen Entities
- dieselbe Repräsentation durch GeometricElements
- ggf. dieselbe Zugehörigkeit zu einer Entity group

Relationale Implementierung (zusätzlich zu Entity)

Key?	Name	Typ	Inhalt
	TAXO	character	taxonomischer Typ (Simple, Group, Member)
F	GROUP	integer	= E_ID der group-Entity (nur für TAXO='M')

Erläuterungen

In GRADIS entspricht die **Entity** einer Art(-Gruppe). Arten mit identischer Attributstruktur lassen sich zu Artgruppen zusammenfassen. Beispiel: Artgruppe "Gebäude" aus Arten "Wohnhaus" und "Garage". Diese verschiedenen Sichten auf dieselben Daten werden in der Anwendung benötigt. Der GRADIS-Ansatz besitzt die Einschränkungen "Einstufigkeit" (keine Verschachtelung möglich) und "Attributredundanz" (hat Garage ein Attribut "Anzahl Kfz", dann alle Gebäude).

Ein allgemeiner, redundanzfreier Ansatz wäre ein echtes Klassensystem mit vollständiger Hierarchie von Super- und Subklassen, Vererbung und Spezialisierung. Die Implementierung wäre aber unüberschaubar aufwendig und vermutlich mit erheblichen Performance-Problemen behaftet.

Daher bilden wir das einstufige Typisierungssystem von GRADIS nach, indem eine group-**Entity** letztlich durch eine feature class repräsentiert wird, die alle **Attributes** aller beteiligten member-**Entities** und ein Typ-Attribut zur Unterscheidung der member-Zugehörigkeit hat. Die member-**Entities** werden gebildet durch eine Objektselektion über das Typ-**Attribute** und die Zuordnung derjenigen Teilmenge der **Attributes**, welche für das jeweilige Member Bedeutung haben. Die Attribut-Redundanz kann wenn nötig dadurch vermieden werden, daß pro member-**Entity** eine extra Tabelle angelegt wird, die nur die member-spezifischen **Attributes** enthält. Allerdings ist dann mehr Aufwand zum Erhalt der referentiellen Integrität notwendig.

Beispiel: Group-**Entity** "Gebäude" hat Attribute "Anzahl Kfz" und "Anzahl Wohneinheiten" sowie ein Typ-**Attribute** mit dem Wertebereich {Wohnhaus,Garage}. Member-**Entity** "Garage" wird gebildet durch Selektion aller "Gebäude" mit Typ='Garage' und Zuordnung des **Attributes** "Anzahl Kfz"; "Anzahl Wohneinheiten" ist kein **Attribute** von "Garage".

Verschachtelte group-member-Beziehungen lassen sich in der Praxis vermeiden und werden daher (zunächst) nicht unterstützt.

Die in GRADIS vorhandenen "name rule" und "name mandatory" werden nun wie jedes andere Attribut behandelt, da die ESRI-GIS kein spezielles Objektname-Attribut kennen.

Diskussionspunkte

2.2.1.2 ObjectlessEntity

Beschreibung

Datenschicht ohne Objektbildung, z.B. gescannte TK, Geländemodell etc.

Relationale Implementierung (siehe Entity)

Key?	Name	Typ	Inhalt

Erläuterungen

Jede ObjectlessEntity bezieht sich eindeutig auf eine objektlose Datenschicht (IMAGE, GRID, TIN, LATTICE)

Diskussionspunkte

2.2.2 Relationship

Beschreibung

Direkte Entsprechung zur Relationship des *Anwendungs*-E/R-Diagramms., z.B. Blockseite ist Teil von Blockseitenabschnitt, Straßenabschnitt beginnt an / endet an Straßenknoten

Relationale Implementierung

Key?	Name	Typ	Inhalt

(implementiert durch View auf **RelationshipSide**)

Erläuterungen

Der Split in Beziehungs*hälften* ist eine Normalisierung, die (später) eine Zusammensetzung von mehrstufigen Beziehungen ermöglicht.

Diskussionspunkte

2.2.3 RelationshipSide

Beschreibung

Relationship-Hälfte, dient zur Normalisierung; wird auf Anwenderebene nicht einzeln präsentiert

Folgende *Beziehungstypen* werden unterschieden:

- Object Reference: Bezug über den eindeutigen Objektidentifikator. Dieser ist permanent, d.h. er kann nach Erzeugung des Objekts nicht mehr geändert werden. Dadurch wird ein spezielles Problem referentieller Integrität vermieden (Umbenennung eines referenzierten Objekts).
- Name Reference: Beziehung wird über Objektnamen hergestellt. Bei Kardinalität = 1 ist Eindeutigkeit der Namen sicherzustellen. Referentielle Integrität ist beim Umbenennen eines referenzierten Objekts gefährdet.
- Hierarchy: Verwendung eines hierarchischen Schlüsselsystems. Die Beziehung wird über substring-Vergleiche hergestellt.
- Topology: gespeicherte topologische Beziehung im engeren geometrischen Sinne, wird je nach Topologietyp (z.B. Knoten-Kanten-Netz) unterschiedlich in ARC/INFO repräsentiert.
- Geometry: geometrisch-topologische Beziehung, die bei Bedarf "on-the-fly" errechnet wird (typisch für SDE!)

Kardinalität von Beziehungen:

Es wird für das Anwendungs-E/R-Diagramm die Dritte Normalform vorausgesetzt, so daß man mit den Kardinalitäten 1:c, 1:1, 1:cn, 1:n auskommt (n:m wird durch Kreuzreferenz-Entities modelliert). Dabei bedeutet "c" (*can*), daß auch 0 erlaubt ist, d.h. die Beziehung ist optional (kann fehlen).

Referentielle Integrität von Beziehungen

Es muß festgelegt werden, wie sich das System beim Löschen/Umbenennen referenzierter oder referenzierender Objekte verhält. (z.B. was passiert beim Löschen eines Blocks mit den enthaltenen Blockseiten)

1. Wird ein Objekt gelöscht, das von anderen referenziert wird, dann
    - werden alle referenzierenden Objekte ebenfalls gelöscht
    - werden alle Referenzen auf NULL gesetzt
    - wird dies vom System nicht zugelassen
2. Wird das einzige Objekt gelöscht, welches ein bestimmtes anderes Objekt referenziert, dann
    - wird das referenzierte Objekt ebenfalls gelöscht
    - wird dies ohne weiteres zugelassen
    - wird dies vom System nicht zugelassen
3. Wird ein per Name Reference referenziertes Objekt umbenannt, dann
    - werden alle referenzierenden Objekte umbenannt
    - wird dies vom System nicht zugelassen

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	RSI_ID	integer	Identifikator
F	ENT_ID	integer	von (Entity)
F	RSI_ID2	integer	gegenüberliegende (RelationshipSide)
	TYPE	character	Beziehungstyp: Object Reference, Name Reference, HIerarchy, TOpology, GEometry
	KEY_RULE	character	für TYPE='H' : Ausschnitt aus Schlüssel, z.B. '1:5'
F	ATT_ID	integer	referenzierendes (Attribute) auf der "von"-Seite"
	CARD	integer	Kardinalität an der "von"-Seite (0=unbegrenzt)
	MAND	bool	"can"-Beziehung, muß nicht aufgebaut werden
	INT_RULE	character	Regeln für referentielle Integrität

Erläuterungen

Angelehnt an GRADIS Konzept (siehe Workshop Meta-Datenmodell S. 4-11 ff.)

Das beschreibende Relationswort (z.B. "besteht aus") steht in der NLS-Tabelle (RelationshipSideName).

Die Attribut-Referenz wurde von Relationship nach RelationshipSide übernommen, um auf beiden Seiten jedes Relationships die Wahlfreiheit des Attributs zu bewahren. (In GRADIS war durch den Relationship-Typ bereits das Attribut eindeutig festgelegt, über welches referenziert wird.)

Konsistenzbedingungen der obigen Tabelle:

1. Durch die RSI_ID2 Verweise entstehen Paare.
2. Beide Seiten jedes Paars haben denselben TYPE
3. Mindestens eine Seite jedes Paars hat CARD=1

Beziehungen höherer Ordnung, z.B. die ternäre Beziehung "Abbiegevorschrift" (von Abschnitt - über Knoten - nach Abschnitt) werden durch Normalisierung (3NF) und einfügen einer Kreuzreferenz-Entity in binäre Relationships abgebildet.

Der Typ "Topology" wird in **TopologyXRF** weiter aufgeschlüsselt in die von ARC/INFO unterstützten Topologietypen.

Diskussionspunkte

- Sind die Integritätsregeln mit vertretbarem Auwand zu realisieren? Sind sie vollständig? ==> Regelbasierte Fortschreibung

2.2.4 Attribute

Beschreibung

Direkte Entsprechung zu den Entity-Attributen des Anwendungs-E/R-Diagramms. Eine **Attribute**-Instanz entspricht einer Eigenschaft einer Objektklasse der realen Welt, z.B. Garagenfläche, Silohöhe, Baumanteil unter 10 Jahre etc.

Jedes **Attribute** ist genau einer **Entity** zugeordnet. Eine Mehrfachverwendung physikalischer Tabellenspalten ist aber über **Field** möglich. Dort geschieht auch die Abstraktion von physischen Tabellenvarianten wie RDBMS, INFO.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	ATT_ID	integer	Identifikator
F	ENT_ID	integer	Zugeordnet zu (Entity)
F	FLD_ID	integer	Daten in (Field)
	TYPE	character	Character, Date, Floating point, Integer
	WIDTH	integer	max. Darstellungsbreite des Werts (für Oberfläche)
	VAL_RULE	text	Regel zur Bildung von Attributwerten
	VAL_MAND	bool	Attributwert zwingend?
	DOM_CLASS	character	Discrete, Range, Unrestricted
	DOM_LOW	character	kleinster zul. Wert für Ranges
	DOM_HIGH	character	größter zul. Wert für Ranges
	DOM_INT	character	ggf. Schrittweite für Intervallwerte (beg. bei DOM_LOW)
	UNIT	character	Bezeichnung der Einheit
F	C_USR_ID	integer	Erzeugt durch (User)
	C_TIME	timestamp	Zeitpunkt der Erzeugung
F	M_USR_ID	integer	Letzte Veränderung durch (User)

	M_TIME	timestamp	Zeitpunkt der letzten Veränderung

Erläuterungen

Die (cn:1) Beziehung zu **Entity** ist zu hinterfragen: eine (cn:m) Beziehung würde es erlauben, dasselbe **Attribute** mehreren **Entities** zuzuordnen. Dann müßte man aber bei Änderungen immer auf Konsistenz bzgl. meherer **Entities** achten. Der Benutzer/Anwendungsverwalter dürfte aber die beschreibenden Daten zu einem **Attribute** meist mit einer bestimmten **Entity** im Hinterkopf erfassen. Die mögliche Redundanz durch Wiederholung von **Attribute**-Definitionen für verwandte **Entities** hält sich in Grenzen.

Diskussionspunkte

- NLS für UNIT nötig?

2.2.5 GeometricElement

Beschreibung

Geometrischer Bestandteil bzw. Alternative der Repräsentation einer Objektlasse der realen Welt, z.B. Silofläche / Siloreferenzpunkt, Schemaplan / Lageplan, Flächenfüllung / Grenzlinie

Das GeometricElement ergänzt die klassische Datenmodellierung (Entity mit Attributen und Beziehungen) um geometrische Objekte.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	GEL_ID	integer	Identifikator
F	ENT_ID	integer	Zugeordnet zu (Entity)
F	FCL_ID	integer	Daten in (Feature Class)
	EXPR	text	SELECT Ausdruck zur Teilmengenbildung aus der Feature Class
	TYPE	character	Geometrietyp: Polygon, Line, X=point, Text, Raster
F	ATT_ID	integer	Attribut, das den Text enthält (bei TYPE='T')
	GEO_MAND	bool	Geometrie zwingend?
	MIN_SCALE	integer	Untergrenze (exklusiv) des Maßstabsbereichs, in dem das GeometricElement sinnvollerweise dargestellt wird
	MAX_SCALE	integer	Obergrenze (inklusiv) des Maßstabsbereichs
F	C_USR_ID	integer	Erzeugt durch (User)
	C_TIME	timestamp	Zeitpunkt der Erzeugung
F	M_USR_ID	integer	Letzte Veränderung durch (User)
	M_TIME	timestamp	Zeitpunkt der letzten Veränderung

Erläuterungen

Ein Geo-Objekt kann aus mehreren Geometrieteilen unterschiedlichen Typs bestehen. Dieses Konzept ist bei ESRI-GIS bisher nur sehr eingeschränkt (z.B. POLY mit label) zu finden. Es ist jedoch in komplexen Datenbeständen üblich (z.B. GDF, ATKIS) und damit für Anwendungen sinnvoll.

Diskussionspunkte

2.3 Darstellungsmodell

2.3.1 View

Beschreibung

Kartographische Darstellung in Form mehrerer Schichten (**Themes**). Die **View** wird im Kartenfenster der Anwendungsoberfläche angezeigt oder kann als Darstellungselement in einem Kartenlayout (Plot) verwendet werden.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	VIW_ID	integer	Identifikator
F	C_USR_ID	integer	Erzeugt durch (User)
	C_TIME	timestamp	Zeitpunkt der Erzeugung
F	M_USR_ID	integer	Letzte Veränderung durch (User)
	M_TIME	timestamp	Zeitpunkt der letzten Veränderung

Erläuterungen

Starke Analogie zum View in ArcView.

Diskussionspunkte

2.3.2 Theme

Beschreibung

Darstellungsschicht eines **Views**. Definiert auf Basis einer **Entity** die Symbolisierung derselben. Für die **Themes** wird eine Zeichenreihenfolge festgelegt.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	THM_ID	integer	Identifikator
F	VIW_ID	integer	gehört zu (View)
	ORDER	integer	Zeichenfolge (1=wird als erstes gezeichnet)
	SYM_SET	character	Name eines ARC/INFO symbolset, ArcView legend file ...
F	SYM_ATT	integer	zur Symbolisierung verwendetes (Attribute)
F	SYM_LUT	integer	zur Symbolisierung verwendete lookup (Table)
F	C_USR_ID	integer	Erzeugt durch (User)
	C_TIME	timestamp	Zeitpunkt der Erzeugung
F	M_USR_ID	integer	Letzte Veränderung durch (User)
	M_TIME	timestamp	Zeitpunkt der letzten Veränderung

Erläuterungen

Starke Analogie zum Thema in ArcView.

Die SYS_LUT Table hat eine Spalte VALUE, welche den Attribute-Wert enthält. Die Spalte SYMBOL enthält eine Symbolnummer innerhalb des SYM_SET.

Diskussionspunkte

- Es sollte eine Auswahl von Objekten (zusätzlich zu der durch GeometricElement) zugelassen werden, am besten durch Beziehung zu einer Abfrage oder Menge.

2.4 National Language Support

Das stark normalisierte NLS-Konzept erlaubt einen einfachen Wechsel zwischen verschiedenen Sprachen. Die Auswirkungen auf die Performance dürften wegen des geringen Volumens der beteiligten Tabellen unwichtig sein (sie werden alle im RAM des RDBMS-Servers gehalten).

NLS-Namen gibt es für die Dictionary-Entities View, Theme, Entity, Attribute, RelationshipSide und GeometricElement. Pro Sprache und Dictionary-Objekt darf es nur einen NLS-Namen geben (z.B. jedes **Attribute** hat nur einen deutschen Namen)

Für alle Dictionary-Entities außer RelationshipSide müssen die NLS-Namen innerhalb der Dictionary Entity eindeutig sein (z.B. jedes **Attribute** hat einen anderen NLS-Namen)

2.4.1 Language

Beschreibung

Natürliche Sprache, die auf konzeptioneller Ebene (Anwendungsoberfläche) verwendet wird

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	LAN_ID	integer	Identifikator
	DESC	text	Bezeichnung der Sprache

Erläuterungen

Diskussionspunkte

2.4.2 Name

Beschreibung

Name in natürlicher Sprache, der auf konzeptioneller Ebene verwendet wird, incl. beschreibender Text

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	NAM_ID	integer	Identifikator
F	LAN_ID	integer	in (Language)
F	xxx_ID	integer	ist Name für (xxx) (s.u.)
	NAME	text	Name in der natürlichen Sprache
	DESC	text	ausführliche Becshreibung

Erläuterungen

Für jedes xxx aus {RelationshipSide, View, Theme, Entity, Attribute, Geometric Element} gibt es eine eigene Namentabelle.

Diskussionspunkte

2.5 Zugriffsrechte

Zu unterscheiden ist der Zugriff auf Metadaten vom Zugriff auf die Geodaten/Attributwerte. Bei **Entity** und **Attribute** wird der Zugriff auf Die Geodaten/Attributwerte geregelt, diesbzgl. Metadaten ändert nur der außerhalb des Anwendungssystems operierende Verwalter. Dagegen werden Entitäten des Darstellungsmodells auch vom Anwender verändert, d.h. für **View** und **Theme** wird der Zugriff auf die *Meta*daten geregelt.

2.5.1 User

Beschreibung

Benutzerkennung des Anwendungssystems

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	USR_ID	integer	Identifikator

	DESC	text	Beschreibung

Erläuterungen

Diskussionspunkte

- Wo wird das password gespeichert? Abstützen auf RDBMS USER/ROLE (OPS$xxx user)?

2.5.2 UserRoleXRF

Beschreibung

Benutzer-Rollen-Zuordnung

Relationale Implementierung

Key?	Name	Typ	Inhalt
F	USR_ID	integer	Benutzer hat ...
F	ROL_ID	integer	... Rolle

Erläuterungen

Diskussionspunkte

2.5.3 Role

Beschreibung

Rolle einer Anwendergruppe im System in Form einer Menge von Zugriffsrechten, z.B. "RBS Fortführung"

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	ROL_ID	integer	Identifikator
	DESC	text	Beschreibung

Erläuterungen

Diskussionspunkte

2.5.4 Privilege

Beschreibung

Zugriffsrecht auf eine Entität

Relationale Implementierung

Key?	Name	Typ	Inhalt
F	ROL_ID	integer	ist Teil von (Rolle)
F	xxx_ID	integer	Zugriff auf (xxx) (s.u.)
	ACCESS	character	Zugriffsrechte: (Read,New,Alter,Delete und Komb.)

Erläuterungen

Für jedes xxx aus {View, Theme, Entity, Attribute, GeometricElement} gibt es eine eigene Privilegtabelle.

Diskussionspunkte

2.6 Logische Sicht

2.6.1 FeatureClass

Beschreibung

Abstraktionsebene für Datenebenen in ESRI-GIS: ARC/INFO feature class/subclass, SDE layer (eine Geometrieart), etc.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	FCL_ID	integer	Identifikator
F	GDS_ID	integer	ist in (GeodataSet)
	FCL_TYPE	character	Typ: XC, LC, PC, ND, RT, SC, RG, AN, XE, LE, SL
	SUBCLASS	character	subclass (bei RT, SC, RG, AN)
F	TBL_ID	integer	bei FCL_TYPE in {XE,LE}: event (Table)
	LAYER	integer	bei FCL_TYPE='SL': SDE layer Nummer

Erläuterungen

Diese Abstraktion wird bereits vielfach praktiziert: ADF, ArcProjekt, ArcView, MapObjects kennen alle eine Art "Layer", welche von der physischen Datendarstellung abstrahiert.

XC=POINT cover, LC=LINE cover, PC=POLY cover, ND=NODE, RT=ROUTE, SC=SECTION, RG=REGION, AN=ANNO, XE=POINT EVENT, LE=LINE EVENT, SL=SDE layer

Sinnvolle Annahme: SDE layer sind homogen bzgl. (group) Entity! SDE läßt zwar das Speichern beliebiger features in einem layer zu, jedoch ist ein split in layer nach Entities sinnvoll.

Diskussionspunkte

- Evtl. auch shapefiles aufnehmen

2.6.2 GeodataSet

Beschreibung

Abstraktionsebene für Datenbestände in ESRI-GIS: ARC/INFO cover, SDE layer, etc.

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	GDS_ID	integer	Identifikator
F	DBS_ID	integer	gehört zu (Database)
	TYPE	character	COVER, SDE, MLLAYER, ASLAYER, GRID, TIN, LATTICE, IMAGE
	SOURCE	character	cover / dataset / layer / ... Name

Erläuterungen

Analog zu ADF GEODATA_SET

Diskussionspunkte

2.6.3 Field

Beschreibung

Abstrahiert von unterschiedlichen physischen Tabellenspalten: INFO item, RDBMS column...

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	FLD_ID	integer	Identifikator
F	TBL_ID	integer	Spalte in (Table)
	NAME	character	physischer Feldname

Erläuterungen

Analog zu ADF FIELD. Die Entspechungen der Felder NULL, DOM_... finden sich jedoch bei **Attribute**

Diskussionspunkte

2.6.4 Table

Beschreibung

Abstrahiert von unterschiedlichen physischen Tabellen: INFO DF, RDBMS table...

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	TBL_ID	integer	Identifikator
F	DBS_ID	integer	gehört zu (Database)
	NAME	character	physischer Tabellenname
	TYPE	character	INFO, RDBMS

Erläuterungen

Analog zu ADF TABLE.

Diskussionspunkte

2.6.5 TopologyXRF

Beschreibung

Abstrahiert von den ARC/INFO-unterstützten Arten topologischer Beziehungen und verweist auf die beteiligten Geometrien.

Relationale Implementierung

Key?	Name	Typ	Inhalt
F	REL_ID	integer	(Relationship)
F	GEL_ID	integer	(GeometricElement)
	TYPE	character	ARC/INFO Topologietyp: REGION, ROUTE, ARC/NODE, ARC/POLY, XE=POINT EVENT, LINE EVENT

Erläuterungen

Diskussionspunkte

2.6.6 Database

Beschreibung

Physische Datenbank (workspace, RDBMS Zugang, SDE dataset, ...)

Relationale Implementierung

Key?	Name	Typ	Inhalt
P	DBS_ID	integer	Identifikator
	DBS_TYPE	character	WORKSPACE, LIBRARY, ARCSTORM, INFO, RDBMS, DATASET
	DBS_NAME	character	workspace / library Name, database (dbs), dataset Name
	DBS_PATH	character	kpl. Pfad zu workspace / INFO dir / map library connect info für RDBMS / SDE

Erläuterungen

Analog zu ADF DATABASE.

IMAGE Dateien werden in WORKSPACEs organisiert.

Diskussionspunkte

3 Bezug zu anderen Ansätzen im Bereich Metadaten

3.1 Application Development Framework ADF

ADF kann für die vorliegende Anforderung (dynamische Nutzung des Dictionary zur Laufzeit der Anwendung, "Interpreter") nicht verwendet werden, da es technisch gesehen ein Programmgenerator ("Compiler") ist.

Die Struktur des Data Dictionary (DD) aus ADF wurde an einigen Punkten übernommen, insbesondere im Bereich Darstellungsmodell und logische Sicht.

3.2 ArcProjekt

ArcProjekt realisiert derzeit nur Ausschnitte des skizzierten Konzepts und speichert die SDD-Daten in proprietären Dateien, welche nur einem Anwender lokal zur Verfügung stehen.

Die SDD-Konzepte sollen in die Weiterentwicklung (ArcProjekt 3) einfließen.

3.3 CEN/TC 287 Geographic Information

Auf Basis der CEN/TC 287 wird derzeit in Bozen ein Metadatensystem entwickelt. Strukturelle Daten (Dictionary) sind nicht Inhalt, dafür werden Metadaten sehr detailliert aufgeschlüsselt. Es steht der Aspekt Meta-Informationssystem im Vordergrund.

3.4 GRADIS Metadata Dictionary

Das in Köln entwickelte GRADIS Metadata Dictionary bildet neben ADF die zweite wesentliche Ausgangsbasis für das vorliegende Konzept.

3.5 Strategisches Informationssystem SIS

Die SIS-Metadatenstruktur ist bisher nicht eingeflossen. Hier ist noch direkte Diskussion mit der SAG nötig.

4 Begriffliche Entsprechungen

SDD Entity	GRADIS / MDD	ARC/INFO-ArcProjekt	ArcView	SDE	MapObjects	ADF
FeatureClass	Art	feature class / subclass		entity type		THEME
Field		item	Field	column		FIELD
GeoDataset		coverage, ML layer, AS layer, grid, tin, lattice, image		layer		GEODATA_SET
Table	table	INFO DF, RDBMS table	Table	table		TABLE
Database	Operat	workspace, ML library, AS library		dataset		DATABASE
GeometricElement	geom. Element	feature class / subclass				-
Attribute	Benutzerattribut	item	Field	column		THEME_FIELD
Entity	Artgruppe/Art					
Relationship	Beziehung	relate, topology		join		
Theme		Thema	Thema	-		THEME_SET_XRF
View		View	View	-		-