Enterprise Business Intelligence[1] – agile modellorientierte Entwicklung

White Paper der solvistas GmbH

Thomas Neuböck
Rico Pommerenke
Jürgen Raab
Melissa Schmidt
Andreas Weißenböck

Linz, März 2017

1 Einleitung

Business Intelligence (BI) und Data Warehouse (DWH) Projekte weisen in vielerlei Hinsicht Besonderheiten auf. Insbesondere ist eine BI-/DWH-Entwicklung ein kontinuierlicher Prozess, der, verglichen mit anderen IT-Projekten, keinen so klar definierten Endzeitpunkt aufweist. Wurden Anforderungen umgesetzt, entstehen durch laufende Datenanalysearbeiten der Endanwender neue Wünsche, die möglichst unverzüglich im BI-/DWH-System umgesetzt werden sollen. Bezüglich einer eingesetzten Vorgehensweise für die Entwicklung in diesem Bereich sind daher folgende Fragen zu stellen:

- Wie zufrieden sind unsere Kundinnen und Kunden?
- Setzen wir das um und verstehen wir, was die Kundinnen und Kunden wollen?
- Wie bewältigen wir fachliche und technische Komplexität?
- Wie sieht es mit der Transparenz und Verfügbarkeit von fachlichem und technischem Wissen aus?

[1] Die in (Neuböck T., Raab J., Weißenböck A., März 2014), (Hiebl J., Hörak K., Linner K., Neuböck T., Raab J., Weißenböck A., Juli 2014) und (Neuböck T., Raab J., Weißenböck A., Dezember 2014) eingeführte agile modellorientiert Vorgehensweise der solvistas GmbH wurde nach der Erprobung im deutschen öffentliche Krankenversicherungsbereich auch in Österreich angewendet. Die vorliegende Darstellung bezieht sich auf ein Projekt in einem österreichischen öffentlichen Bereich, in dem ein Enterprise Business Intelligence (EBI) eingeführt wurde. Der Ansatz in dieser vorliegenden Darstellung wurde ausgebaut. Insbesondere erfolgte eine Erweiterung auf ein gesamtes BI-System, d.h. dass in diesem White Paper auch die Präsentationschicht (auf Basis von IBM Cognos) eingebunden ist.

- Haben wir eine Architektur (und eine Umsetzung), die einfach gewartet und erweitert werden kann?
- Haben wir ausreichende Richtlinien und Standards zur Qualitätssicherung und werden diese auch gelebt?
- Wie gehen wir bei geänderten Anforderungen oder aufgetretenen Hindernissen um?
- Wie hoch sind die Eigenständigkeit, das Verantwortungsbewusstsein und der Motivationsgrad im Entwicklungsteam?

Aus der Beantwortung dieser Fragen ergeben sich folgende Forderungen an ein effektives und effizientes Vorgehensmodell bei der Entwicklung von BI-/DWH-Systemen:

- Kundenorientiert Umsetzung von Fachthemen
- Verringerung der Kommunikations- und Verständnislücke zwischen Fachlichkeit und Technik
- Verbesserter Umgang mit der fachlichen und technischen Komplexität durch Anforderungszerlegung und kurze Umsetzungszyklen
- Breite Wissensstreuung in der Abteilung
- Einführung und Umsetzung einer leicht wartbaren und erweiterbaren Architektur
- Qualitätsgesicherte Lösungen
- Flexibilität bei sich ändernden Anforderungen oder auftretenden Hindernissen
- Motivationsförderung in den Projektteams

solvistas setzt in BI- und DWH-Projekten ein an die Charakteristik solcher Projekte angepasstes Vorgehensmodell ein: *solvistas modellorientierte agile Vorgehensweise bei BI-/DWH-Projekten.* Dabei werden insbesondere die in Abbildung 1 dargestellten Sichtweisen kontinuierlich im Fokus gehalten: Managementsicht (inkl. der Vorgehensweise selbst), fachliche Sicht, Architektursicht und Modellsicht.

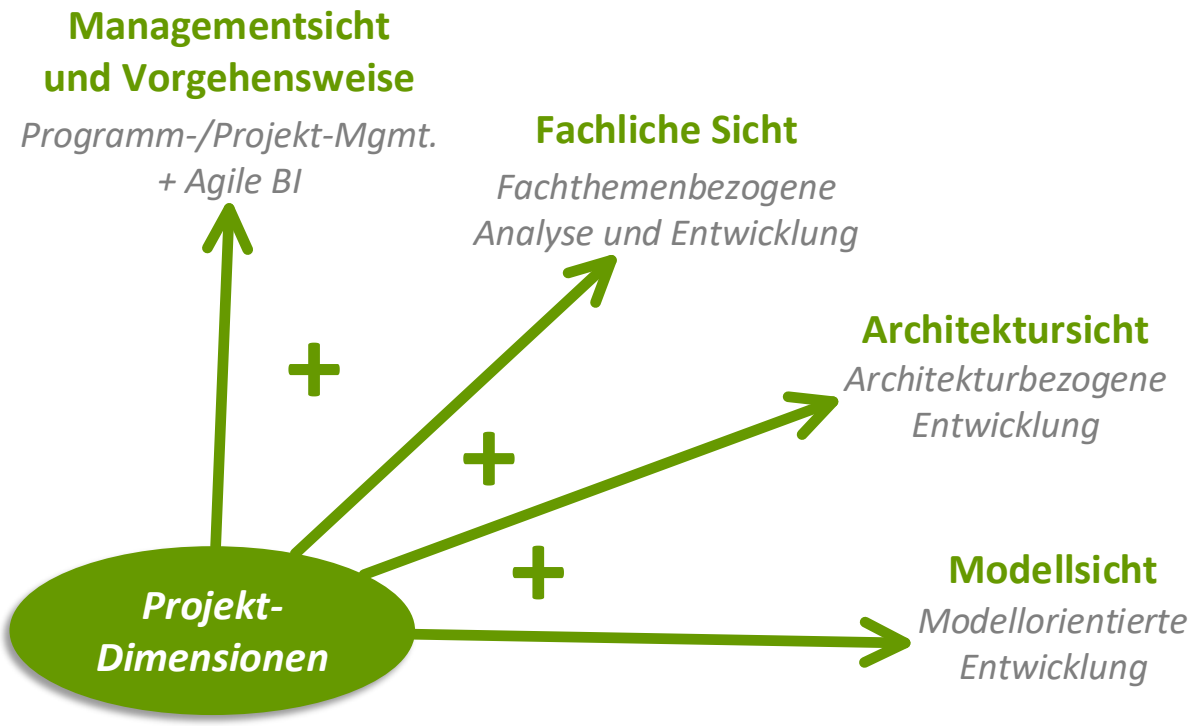


Abbildung 1: Dimensionen eines BI-Projektes

Die Managementsicht und Projektvorgehensweise verlangt ein übergeordnetes Programm- und Projektmanagement. Darin eingebettet werden Projekte und Programme in einer agilen Weise durchgeführt (agile BI). solvistas ist Unterzeichner des *Memorandum für Agile Business Intelligence* der BI- und DWH-Community *TDWI* (http://www.tdwi.eu/wissen/agile-bi/unterzeichner/).

Das BI-System wird mit dem Grundsatz „Think Big, Act Small“ schrittweise erweitert. Bei komplexen Fachthemen sind Evaluierungsphasen oder prototypische Umsetzungen im Sinne einer evolutionären Vorgehensweise (oder gemäß „Spiral-Modell“) denkbar.

Die Tiefe von Fachthemen erfordert eine Zerlegungsstrategie, um Komplexität zu reduzieren. Verschiedene Themen verlangen auch unterschiedliche fachliche Sichtweisen. Mit dem Vorgehensmodell wird eine fachthemenbezogene BI-/DWH-Entwicklung erleichtert.

Die BI-/DWH-Architektur muss wichtige Anforderungen an ein zukunftsorientiertes BI-/DWH-System erfüllen, bspw. eine flexible Erweiterbarkeit, die Bereitstellung historisierter Informationen, Datenintegration und Definition eines „Single Point of Truth“ oder die Bereitstellung verschiedener fachlicher Sichtweisen.

Die Vorgehensweise in BI-/DWH-Projekten fordert eine modellorientierte Entwicklung. Modelle erlauben die Einbringung verschiedener Sichtweisen und ergeben eine formale Dokumentation. Drei Modellebenen werden unterschieden:

- Fachliche Modellebene (konzeptuelle Modelle)
- Logische Modellebene
- Physische Modellebene

In der fachlichen Ebene werden Anforderungen und Sichtweisen des Fachbereichs beschrieben und modelliert. In der logischen Modellebene werden die fachlichen Anforderungen und Sichtweisen in der BI-/DWH-Architektur plattformunabhängig abgebildet. Die physische Modellebene definiert die Umsetzung der logischen Modellebene auf eine konkreten BI-/DWH-Plattform.

Der folgenden Darstellung liegt ein spezifisches Projekt in einem öffentlichen österreichischen Unternehmen zu Grunde.

2 Architektursicht

Die Architektur eines BI-/DWH-Systems muss ein festes und stabiles technisches Fundament für die Umsetzung fachlicher Anforderungen bilden. Die Änderbarkeit und Ausbaufähigkeit ist hinsichtlich sich ändernder und neuer Anforderungen zu gewährleisten. DWH-Schichten müssen um fachliche Themen einfach erweitert werden können.

In Abbildung 2 ist ein generisches Architekturbild dargestellt. Das gesamte BI-System besteht aus einer Präsentationsschicht (PRS), dem Data Warehouse (DWH) und einem Teil zur Verwaltung der Metadaten.

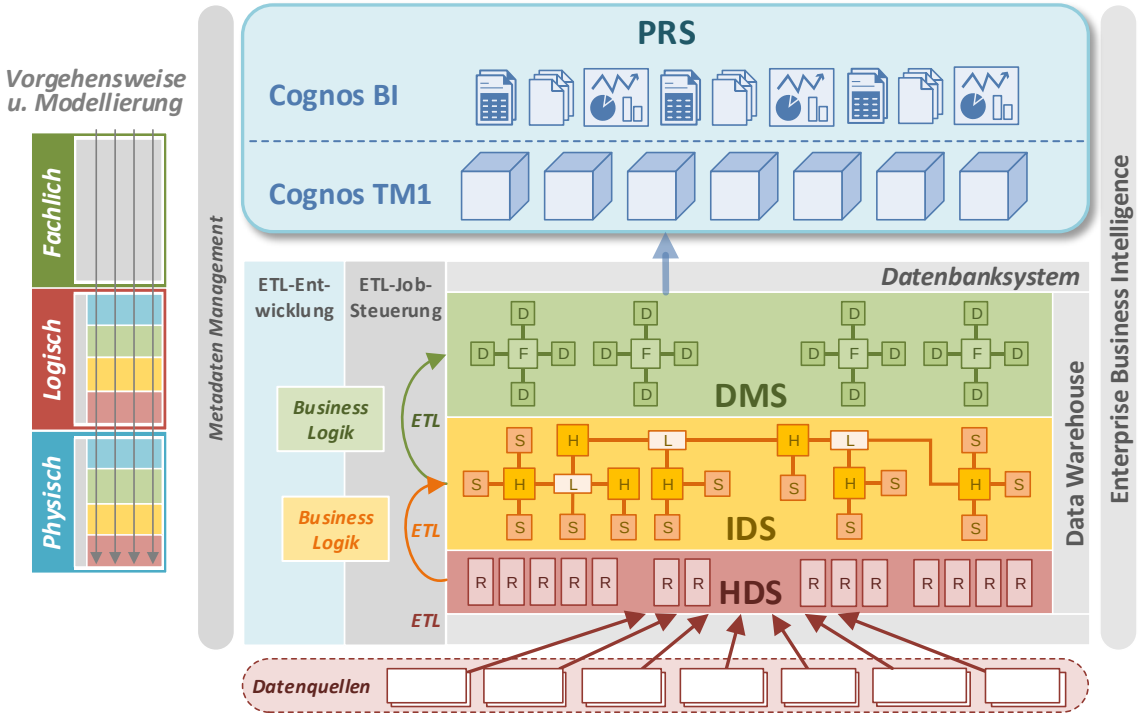


Abbildung 2: BI-/DWH-Architektur

Die Präsentationsschicht bezieht Daten aus dem DWH und stellt diese dem Anwender geeignet zur Verfügung (im EBI-Projekt über Cognos BI und Cognos TM1). Im Falle von multidimensionalen OLAP (MOLAP) werden Würfeldaten als Teil der Präsentationsschicht betrachtet.

Das Data Warehouse gliedert sich in drei Datenschichten: historisierte Datenschicht (HDS), integrierte Datenschicht (IDS) und Data Mart Schicht (DMS). Diese Datenschichten werden in einer Oracle-Datenbank verwaltet.

In der HDS werden Daten aus den operativen Systemen strukturell möglichst unverändert übernommen („System of Records“), wobei die Datenhistorie nicht verloren gehen darf. Dies wird im Rahmen der Iteration 2 eine umfangreiche Darstellung der verschiedensten Quelldatensysteme ermöglichen.

Die IDS beinhaltet alle Daten in einer konsolidierten und trotzdem leicht erweiterbaren Form. Sie sind dort historisiert und in der feinsten Granularität verfügbar. Die IDS stellt somit eine gesamtheitliche und historisierte Unternehmenssicht auf die DWH-Daten dar. Die Datenstrukturen werden gemäß Data Vault Modellierung umgesetzt. Diese trennt fachliche Schlüssel der Geschäftsobjekte (**H**ub-Tabellen) von deren fachlichen Kontextinformationen (**S**atellit-Tabellen) und deren Beziehungen (**L**ink-Tabellen). Die IDS wird aus der HDS heraus befüllt. Zusätzlich soll die IDS auch wichtige abgeleitete Daten beinhalten, die später in der DMS benötigt werden (gemäß definierter Geschäftsregeln). Die bereits bestehenden Daten aus der Iteration 1 zum fachlichen Thema Finanzen werden im Rahmen der zweiten Iteration durch neue Anforderungen ergänzt. Die Informationen der verschiedenen Quellsysteme bzgl. LI-Berichte werden in der IDS mit bestehenden Daten integriert, um einen Gesamtbestand mit Verschneidungen der Informationen zu generieren.

In der DMS werden alle Fachthemen in sogenannten Star-Schemen abgebildet. Diese Schicht bildet die Basis für die PRS. Die DMS wird aus der IDS aufgebaut und stellt somit neben den Informationen Finanzen, zukünftig auch die Daten für die LI-Berichte bereit. Neben den Data Mart (einem anfrageorientierten Teilbestand der IDS) zu Finanzen, der bereits in der ersten Iteration erstellt wurde, ist der Aufbau eines entsprechenden Data Mart für die LI-Berichte angedacht.

In der ETL-Entwicklung werden für alle Schichten ETL-Jobs erstellt, welche die Daten in die jeweilige Datenschicht transferieren und ggf. nach einer spezifizierten Business Logik transformieren. ETL-Jobs müssen auf Entwicklungs-, Test- und Produktionsumgebung verteilt und in die ETL-Jobsteuerung integriert werden.

Die gesamte BI-/DWH-Entwicklung erfolgt dabei iterativ und modellorientiert. Aus fachlichen Modellen werden logische Modelle als Vorgabe für die Datenschichten erstellt. Diese werden physisch in das Datenbanksystem übertragen (Datenbankschema und ETL-Jobs).

Die beschriebene Architektur wird Iteration zwei beibehalten und bedarfsorientiert erweitert. Dazu gehört die Erweiterung der HDS, um alle Datenbestände der Vorsysteme bzgl. LI-Berichte (WIPIS, SAP BW, VIPER, usw.). Die Integration der neuen Bestände in die IDS, dazu gehört die Erweiterung bestehender Hub-Satelliten und Link-Satelliten Konstrukte und die Einarbeitung neuer Teilaspekte, wie Personal und IKT. Die spezifischen Datenbestände der IDS zu den LI-Berichten werden anschließend in einem Star-Schema der DMS (Data Mart) für die Verarbeitung in einem TM1 Cube aufbereitet, welcher die multidimensionalen Daten wiederum dem Reporting bereitstellt.

3 Modellsicht

Modelle dienen zum einen dazu, ausgehend von den fachlichen Anforderungen schrittweise die physische Umsetzung zu erarbeiten, und zum anderen die Beziehung zwischen Fachlichkeit und technischer Umsetzung zu dokumentieren und dabei die Nachvollziehbarkeit zu gewährleisten. Drei Modellebenen werden unterschieden (Abbildung 3): Fachliche Modellebene (konzeptuelle Modelle), logische Modellebene und physische Modellebene.

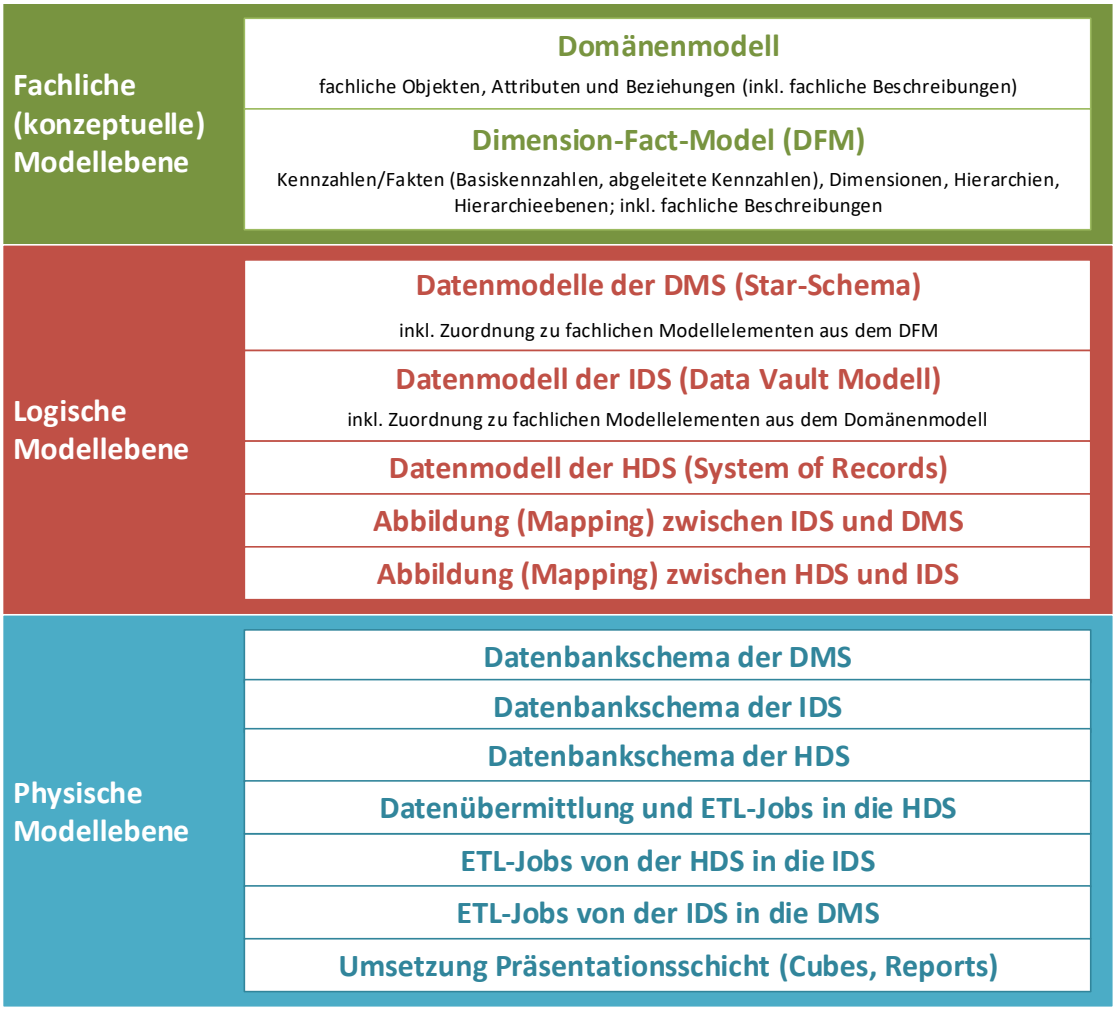


Abbildung 3: Modellebenen

Konzeptuelle Modelle sind Modelle, die der Realität (dem fachlichen Anwendungsgebiet) am nächsten sind. Ausgehend von den fachlichen Anforderungen wird im Domänenmodell eine unternehmensweite Gesamtsicht fachlich relevanter Objekte (Geschäftsobjekte), deren Eigenschaften und Beziehungen dargestellt. Das dimensionale Faktenmodell (DFM; in Anlehnung an Matteo Golfarelli) enthält auswertungsrelevante Fakten (z.B. Geschäftsereignisse), Kennzahlen (sowohl Basiskennzahlen als auch abgeleitete Kennzahlen) und Dimensionen. Letztere können wiederum hierarchisch in Ebenen gegliedert sein, um Auswertungshierarchien zu beschreiben. Eine Dimensionsebene kann zusätzlich beschreibende Attribute aufweisen. Sowohl im Domänenmodell als auch im dimensionalen Faktenmodell sind einzelne Modellelemente auch fachlich zu beschreiben. Analyseanforderungen und Analyseprozesse können zusätzlich in fachlichen Geschäftsanwendungsfällen dokumentiert werden.

In der logischen Modellebene wird die Vorgabe zur technischen Umsetzung plattformunabhängig dokumentiert. Datenmodelle in der DMS werden im Star-Schema (nach Ralph Kimball) modelliert. Als Vorgabe dient das fachliche DFM. Die Beziehung zwischen DFM und DMS ist im Modell zu dokumentieren. Das Datenmodell der IDS wird als Data Vault Modell (nach Dan Linstedt) geführt. Es wird auf Basis des Domänenmodells erstellt. Die Beziehung zwischen Domänen- und IDS-Modell ist festzuhalten. Das Datenmodell der HDS ist einfach gehalten und kann als „System of Records“ betrachtet werden. Es spiegelt

die Datenstrukturen der operativen Systeme möglichst 1:1 wider. Letztendlich sind in der logischen Modellebene die Abbildungen zwischen HDS und IDS auf der einen und zwischen IDS und DMS auf der anderen Seite festzulegen. Analyseanforderungen und Analyseprozesse der Präsentationsschicht werden zusätzlich beschrieben (z.B. Inhalte von Reports) oder/und als Systemanwendungsfälle modelliert.

Die physische Modellebene stellt die technische Umsetzung der Vorgaben aus der logischen Modellebene dar. Das Datenbankschema der HDS, IDS und DMS wird erstellt bzw. erweitert. Daten werden aus den operativen Systemen extrahiert und bereitgestellt. ETL-Jobs zur Beladung der HDS, der IDS (aus der HDS) und der DMS (aus der IDS) sind zu entwickeln und die PRS zu erweitern (z.B. Cubes und Reports).

Abbildung 4 beschreibt den generischen Prozess dieser modellorientierten Vorgehensweise und die daraus entstehenden groben Lieferobjekte. Dieser Prozess ist der Einfachheit halber als rein sequentielle Abfolge von Schritten dargestellt. Leichte Abweichungen von dieser Reihenfolge, Parallelisierungen und Rückkoppelungen sind jedoch möglich und auch sinnvoll (insbesondere auch abhängig von den eingesetzten Werkzeugen und der generellen Vorgehensweise).

Aufgenommene fachliche Anforderungen werden analysiert und genauer spezifiziert. Daraus entstehen allgemeine Fachkonzepte, ein Domänenmodell und ein Dimensional Fact Model (DFM). Für die Präsentationsschicht werden die Zugriffsmöglichkeiten über Cubes und Berichte fachlich festgelegt.

Um die Lieferobjekte der logischen Modellebene zu generieren, werden die Datenschichten analysiert und ihre Spezifikation erweitert. Als Lieferobjekte entstehen logische Datenmodelle der DMS, IDS und HDS. Zusätzlich sind die Abbildungen zwischen diesen Schichten festzuhalten.

Die durch die logische Modellebene vorgegebenen Definitionen werden auf der konkreten Plattform umgesetzt. Es werden das Datenbankschema bzw. die Datenobjekte der HDS, der IDS und der DMS erstellt. Die Datenübermittlung und die ETL-Jobs zur Beladung der HDS, der IDS und der DMS sind der angegebenen Reihe nach zu entwickeln. Schließlich sind die Teile der Präsentationssicht (Cubes und Reports) zu realisieren.


Fachliche Anforderung
Fachliche Anforderung analysieren und spezifizieren
Fachkonzept Fachliche Beschreibung
DFM
Domänenmodell
Präsentationsschicht (Cubes, Reports) analysieren und spezifizieren
Spezifikation in der PRS (Cubes, Reports)
Data Mart Schicht (DMS) analysieren und spezifizieren
DMS-Modell
Integrierte Datenschicht (IDS) analysieren und spezifizieren
IDS-Modell
Historisierte Datenschicht (HDS) analysieren und spezifizieren
HDS-Modell
Abbildung von HDS nach IDS analysieren und spezifizieren
Abbildung von HDS nach IDS
Abbildung von IDS nach DMS analysieren und spezifizieren
Abbildung von IDS nach DMS
Datenbankschema der HDS bereitstellen
Datenbankschema der HDS
Datenübermittlung und ETL-Jobs in die HDS umsetzen
Datenübermittlung und ETL-Jobs in die HDS
Datenbankschema der IDS bereitstellen
Datenbankschema der IDS
ETL-Jobs von HDS in IDS umsetzen
ETL-Jobs von HDS in IDS
Datenbankschema der DMS bereitstellen
Datenbankschema der DMS
ETL-Jobs von IDS in DMS umsetzen
ETL-Jobs von IDS in DMS
Präsentationsschicht (Cubes, Reports) umsetzen
Präsentationsschicht (Cubes, Reports)

Abbildung 4: Modellorientiertes Vorgehen

4 Werkzeugeinsatz und Infrastruktur

Im betrachteten Projekt werden die Datenschichten in einer Oracle-Datenbank verwaltet. Ansonsten kommen generell die Werkzeuge von IBM InfoSphere Information Server und IBM Cognos zum Einsatz. ETL-Jobs werden mit DataStage umgesetzt. Die Datenqualitätsbehebung und -kontrolle wird mit QualityStage durchgeführt. Die Modellierung erfolgt mit Hilfe des Data Architect und FastTrack. Alle Metadaten werden im zentralen Metadaten Repository gepflegt. Die Beschreibung zur fachlichen Modellebene wird zusätzlich im Information Governance Catalog verwaltet. In der Präsentationsschicht kommen IBM Cognos TM1 und IBM Cognos BI zum Einsatz. Es wird empfohlen folgende technische Umgebungen bereitzustellen: Entwicklungsumgebung, Testumgebungen – idealerweise zwei Umgebungen für Integrationstests und Fach- bzw. Abnahmetests, Produktionsumgebung.

4.1 Werkzeugabhängiger Entwicklungsprozess

In den Abbildungen Abbildung 5, Abbildung 6 und Abbildung 7 ist der werkzeugspezifische Modellierungs- und Entwicklungsprozess der vorgeschlagenen Vorgehensweise anhand des InfoSphere Information Servers und IBM Cognos dargestellt.

Dem modell- und metadatengetriebenen Ansatz folgend sind die bereits beschriebenen Modellarten als zentrales Konstrukt der Entwicklung anzusehen. Um diese Modelle fachanwenderorientiert entwickeln zu können, sind die fachlichen Begrifflichkeiten (Metadaten) von den Anwenderinnen und Anwendern zu definieren. Dies geschieht im Information Governance Catalog (A). Dort werden die Begrifflichkeiten wie Terme, Kategorien und Governanceregeln beschrieben und gewartet (1).

Aus den genannten Termen werden fachliche und logische Modelle im Information Data Architect (B) entwickelt (2). Das Ergebnis dieses Schrittes sind die logischen und fachlichen Modelle selbst. Diese werden wiederum eindeutig mit den fachlichen Begriffen verknüpft (3). Diese Verknüpfung wird mit dem Information Data Architect Plugin Information Governance Catalog for Eclipse (C) durchgeführt. Die kollaborative Arbeit wird durch SVN unterstützt.

Da es sich bei Information Data Architect um ein Produkt handelt, welches nicht direkt in das Metadaten Repository des Information Servers schreibt, müssen die fachlichen und logischen Modelle über den Infosphere Metadata Asset Manager (D) in das Repository importiert (4). Anschließend stehen die Modelle im zentralen Metadaten Repository mit entsprechender Verknüpfung zu den Termen zur Verfügung.

Aufbauend auf den logischen Datenmodellen werden im Information Data Architect (B) die physischen Datenmodelle automatisch generiert (5). Dies geschieht datenbankspezifisch, somit sind nur noch kleinere Anpassungen an dem resultierenden Modell durchzuführen. Nach der Erstellung sind die Modelle wieder durch den Infosphere Metadata Asset Manager (D) in das Metadaten Repository zu importieren (6). Neben den, nun zentral im Repository bereitgestellten physischen Datenmodellen, werden über den Information Data Architect auch die erforderlichen DDL-Statements erstellt, um die benötigten Strukturen auf den Datenbanken anzulegen.

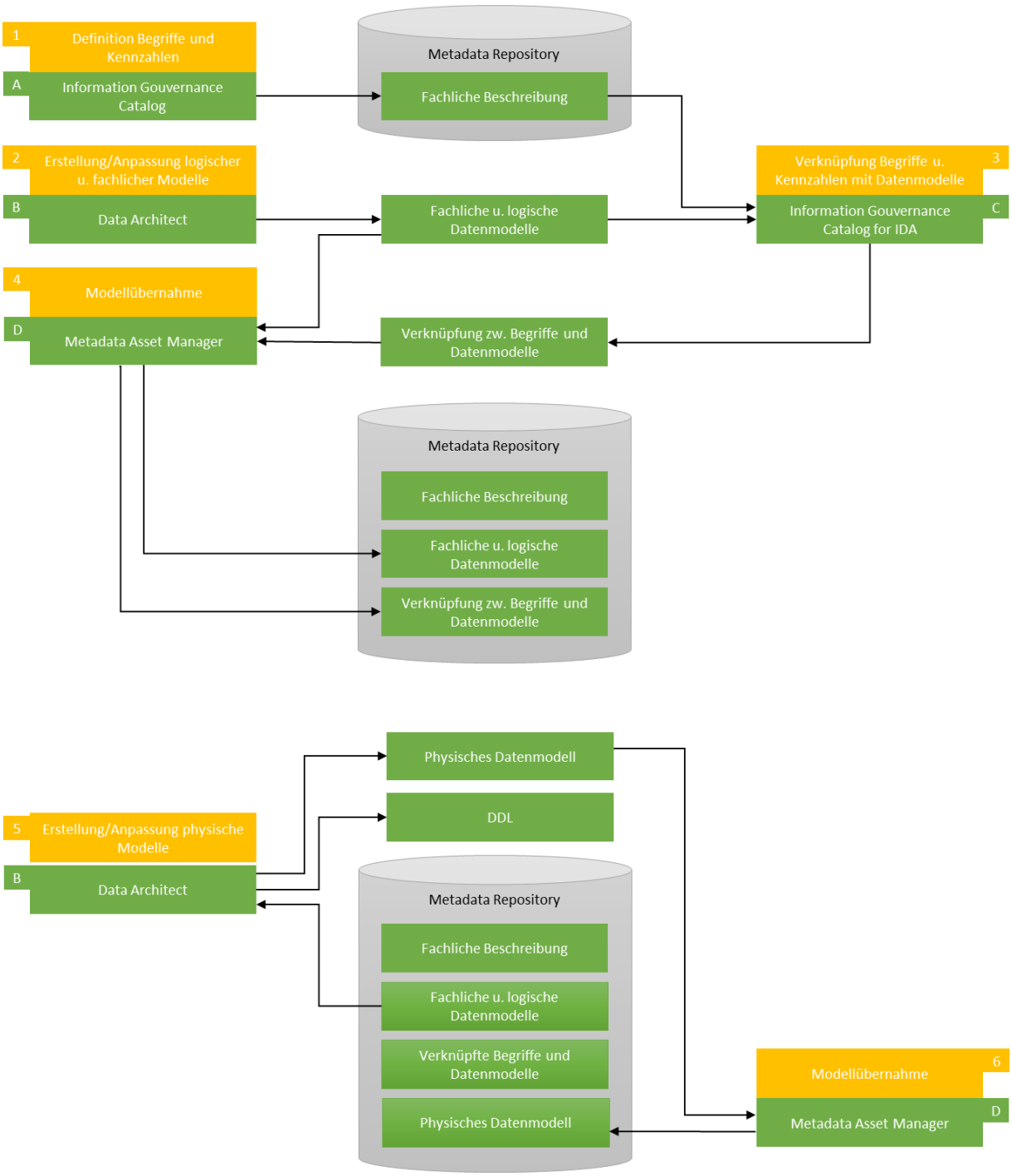


Abbildung 5: Plattformabhängige Modellierungs- und Entwicklungsprozesse (Teil 1)

Aufbauend auf den physischen Datenmodellen der drei DWH-Schichten werden im nächsten Schritt die Mapping-Vorschriften definiert (7). Dies erfolgt im FastTrack (E), welches wiederum direkt auf die Datenmodelle im Repository zugreift. Anmerkung: Das Mapping geschieht in FastTrack bereits auf Basis des physischen Datenmodells.

Aus den definierten Mapping-Vorschriften und physischen Datenmodellen werden im DataStage (F) die ETL-Prozesse entwickelt (8). Die Metadaten der Prozesse sind anschließend voll in das zentrale Repository integriert. Neben der Definition der ETL-

Prozesse werden auch Datenqualitätsregeln durch, die in DataStage integrierte Komponente QualityStage, behandelt und angewendet.

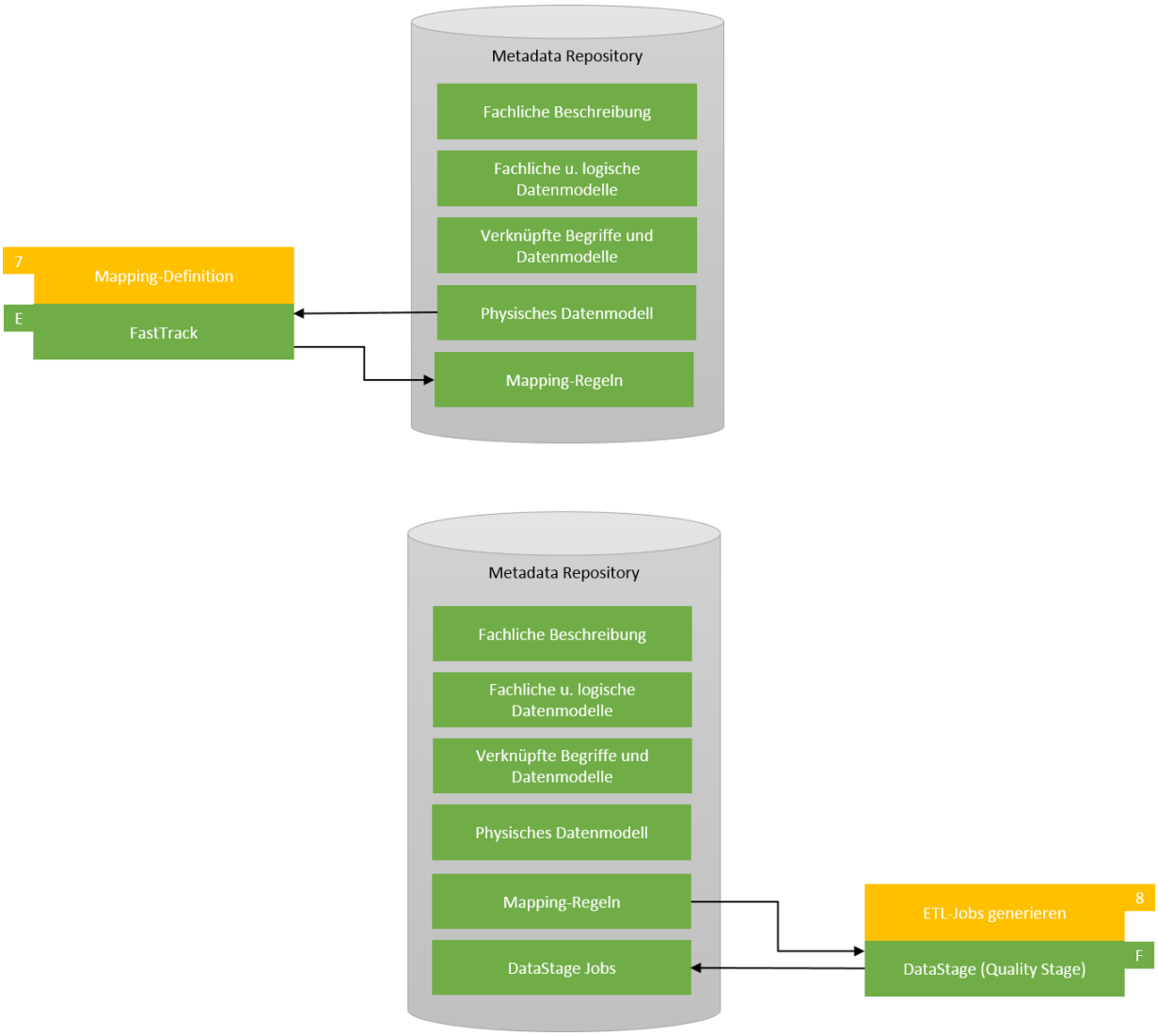


Abbildung 6: Plattformabhängige Modellierungs- und Entwicklungsprozesse (Teil 2)

Nachdem die Daten aus den ETL-Jobs bereitstehen, werden diese in der Präsentationsschicht durch IBM Cognos TM1 (G) und Cognos BI (H) weiterverarbeitet. Durch Cognos TM1 erfolgt eine multidimensionale Aufbereitung und In-Memory Bereitstellung der Daten, um diese anschließend in Cognos BI als Reports bereitstellen zu können. Die daraus resultierenden Metadaten des TM1 Cubes (Dimensionen & Fakten) (9) werden wiederrum durch den Infosphere Metadata Asset Manager (D) in das zentrale Metadaten Repository importiert (10).

Mit dem Import der im Prozess als letztes generierten Metadaten des Tools Cognos BI (H) stehen auch die BI-Objekte bzgl. Reporting im zentralen Metadaten Repository zur Verfügung (12). Diese vielfältige Generierung und Verarbeitung von Metadaten führt dazu, dass eine eindeutige und durchgängige (von der Quelle bis zum Report) Data- bzw. Business Lineage erstellt werden kann. In Abbildung 7 werden die Zusammenhänge der IBM Cognos Tools zum zentralen Metadaten Repository des InfoSphere Information Servers gezeigt.

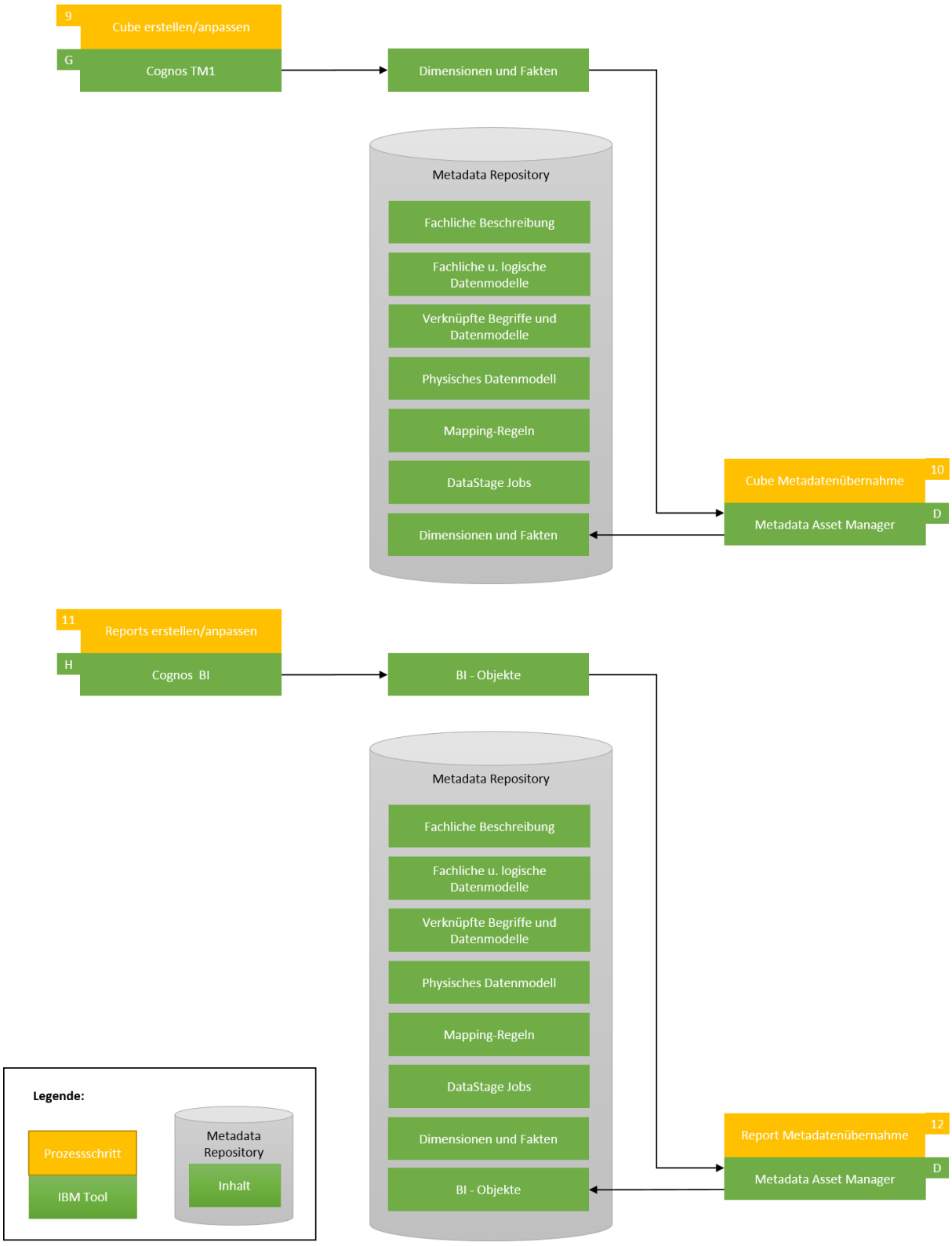


Abbildung 7: Plattformabhängige Modellierungs- und Entwicklungsprozesse (Teil 3)

4.2 Werkzeugabhängige Prozessartefakte

Aus dem beschriebenen Entwicklungsprozess entstehen die in Tabelle 1 aufgelisteten Ergebnisse (Artefakte).

Tabelle 1: Werkzeugabhängige Artefakte des Entwicklungsprozesses

Kennung	Prozessschritt	Produkt	Metadaten - Artefakt	Ablage
1A	Definition Begriffe und Kennzahlen	Information Governance Catalog	Fachliche Beschreibung	MDR
2B	Erstellung/Anpassung logischer u. fachlicher Modelle	Data Architect	Fachliche u. logische Datenmodelle	SVN
3C	Verknüpfung Begriffe u. Kennzahlen mit Datenmodellen	Information Governance Catalog for IDA	Verknüpfung zw. Begriffe und Datenmodelle	MDR
4D	Modellübernahme	Metadata Asset Manager	Fachliche u. logische Datenmodelle Verknüpfung zw. Begriffe und Datenmodelle	MDR
5B	Erstellung/Anpassung physische Modelle	Data Architect	Physisches Datenmodell DDL	SVN
6D	Modellübernahme	Metadata Asset Manager	Physisches Datenmodell	MDR
7E	Mapping-Definition	FastTrack	Mapping-Regeln	MDR
8F	ETL-Jobs generieren	DataStage (Quality Stage)	DataStage Jobs	MDR
9G	Cube erstellen / anpassen	Cognos TM1	Dimensionen und Fakten	TM1
10D	Cube Metadaten Übernahme	Metadata Asset Manager	Dimensionen und Fakten	MDR
11H	Reports erstellen / anpassen	Cognos BI	BI – Objekte	BI
12D	Report Metadaten Übernahme	Metadata Asset Manager	BI – Objekte	MDR

MDR = Metadaten Repository; SVN = Subversion; TM1 = Cognos TM1; BI = Cognos BI

5 Vorgaben zur Modellierung von Modellen

5.1 Modellschichten

Fachliche Modelle	**Architekturunabhängige Darstellung der Fachlichkeit:** • Domänenmodell • Dimension Fact Model (DFM) • Kennzahlenbeschreibung
Technische Modelle	**Plattformspezifische Eigenschaften der Modelle:** • Datentypenmodell
Logische Modelle	**Plattformunabhängige Darstellung der Architektur:** • Logisches Datenmodell der DMS • Logisches Datenmodell der IDS (als Data Vault) • Logisches Datenmodell der HDS

Physische Modelle	**Plattformspezifische Umsetzung der logischen Modelle:** • Erstellung des physischen Datenbankschemas für die DMS • Erstellung des physischen Datenbankschemas für die IDS • Erstellung des physischen Datenbankschemas für die HDS

5.2 Grundsätzliche Richtlinien für die Modellierung

1. Es ist darauf zu achten, dass Abkürzungen eindeutig sind und immer in einem gleichen Kontext verwendet werden.
2. Leerzeichen in Bezeichnungen von Attributen, Entitäten, Objekten und Beziehungen sind nicht zulässig und generell durch Unterstriche „_“ zu ersetzen.

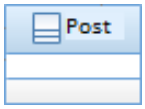
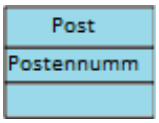
6 Fachliche Modelle

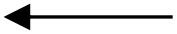
Das Domänenmodell und Dimensional Fact Model (DFM) stellen fachliche (konzeptionelle) Modelle dar.

6.1 Modellierung des Domänenmodells

Im Domänenmodell werden relevante Geschäftsobjekte und deren Beziehungen innerhalb der Businessdomäne dargestellt, um den fachlichen Kontext besser verstehen zu können. Damit wird die Grundlage für eine gemeinsame Begriffsbildung gelegt.

Die Erstellung erfolgt mit einfachen UML-Klassendiagrammen im IBM InfoSphere Data Architect. Dazu werden die folgenden Modellelemente verwendet:

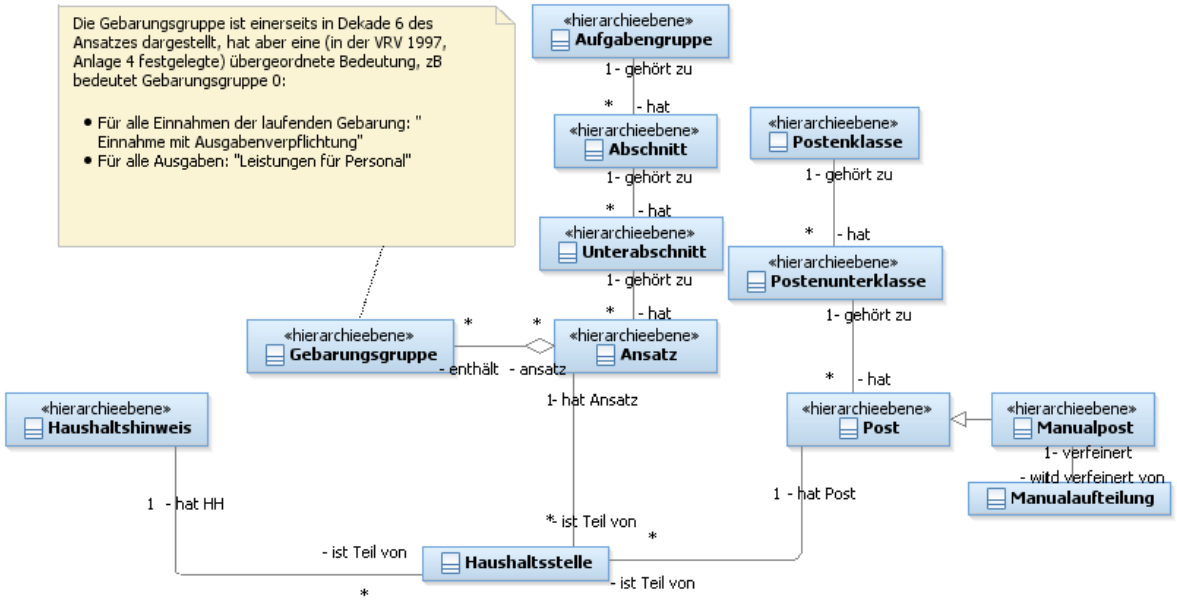
Modellelement	Symbol	Erläuterung
Fachliches Objekt (Klasse)		Eine Klasse beschreibt ein fachliches Objekt der Domäne. Beispiele: Posten, Ansatz, Haushaltshinweis
Eigenschaft eines fachlichen Objekts (Attribut)		Wichtige Eigenschaften fachlicher Objekte können mit Attributen spezifiziert werden. Dabei ist zu beachten, dass ein Domänenmodell nur eine grobe Darstellung liefern soll. Eine detaillierte Ausspezifizierung auf Attributs Ebene soll daher nicht erfolgen. Beispiel: Die Postennummer ist ein wichtiges Attribut des fachlichen Objektes Posten.
Beziehung zwischen fachlichen Objekten	1_____* 0..1_____*	Eine Assoziation stellt eine fachliche Beziehung zwischen fachlichen Objekten dar (soll auch fachlich beschriftet werden). Dabei wird auch die Kardinalität zwischen den Objekten berücksichtigt. Beispiel: Die

(Assoziation)	* ——— *	Postenunterklasse hat mehrere Posten.
IS-A-Beziehung zwischen fachlichen Objekten (Generalisierung)		Die Generalisierung bzw. Spezialisierung stellt einen besonderen Beziehungstyp (IS-A-Beziehung) zwischen zwei fachlichen Objekten dar. Beispiel: Die Manualpost ist eine Post.

6.1.1 Richtlinien für das Domänenmodell

1. Das gesamte Domänenmodell wird in fachliche Teilmodelle gegliedert (z.B. Finanzen, Personal, …). Jedes Modellelement ist genau einem Teilmodell zugeordnet, kann aber in anderen Teilmodellen als Verknüpfung verwendet werden. In einem separaten Modell können alle Teilmodelle als Verknüpfungen zusammengeführt werden, um einen Gesamtüberblick zu erhalten. Der Name jedes Domänenmodells setzt sich aus einem Präfix und einem fachlichen Namen wie folgt zusammen: DOM_*fachlicherName* (z.B. DOM_Finanzen).
2. Ein Objekt (Klasse) im Domänenmodell (z.B. Post, Ansatz) muss einem Business Term im Business Glossary entsprechen und muss sich dort auch wiederfinden. Gleiches gilt für die fachlichen Attribute (z.B. Postennummer).
3. Als Attribute sollen maximal wichtige fachliche Schlüssel in den Klassen eingetragen werden. Ein detailliertes Ausspezifizieren von Klassen bis auf Attributsebene ist zu unterlassen. Das Domänenmodell soll einen fachlichen Überblick geben und erhebt nicht den Anspruch auf Detailtreue.
4. Für Objekte, Beziehungen und Attribute wird die Groß- und Kleinschreibung verwendet.
5. Objekte, Attribute und Beziehungen sind fachlich sprechend und möglichst ohne Abkürzungen zu bezeichnen (z.B. Haushaltshinweis und nicht HH).
   Ausnahme: Allgemein gültige Abkürzungen
6. Jede Beziehung zwischen Objekten (Klassen) ist fachlich zu benennen. Leserichtung und Kardinalitäten sind anzugeben.
7. Gibt es zu Objekten (Klassen) oder Beziehungen weitere Anmerkungen (z.B. Bedingungen) können diese verbal formuliert und als UML-Notiz dem Modellelement angefügt werden.

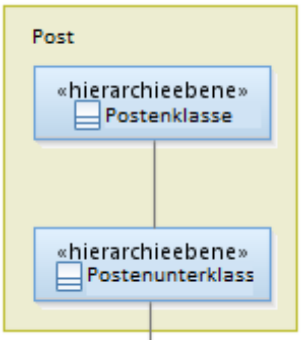
6.1.2 Beispiel eines Domänenmodells



6.2 Modellierung des Dimension Fact Model (DFM)

Das Dimension Fact Model (DFM) stellt wie das Domänenmodell ein konzeptuelles Modell dar und wird mit IBM InfoSphere Data Architect als UML-Klassendiagramm erstellt. Darin werden Kennzahlen, Dimensionen und deren Auswertungshierarchien abgebildet. Folgende Modellelemente werden dazu herangezogen:

Modellelement	Symbol	Erläuterung
Hierarchieebene (Klasse)	«hierarchieebene» Fachgruppe	Eine Klasse beschreibt eine Hierarchieebene einer Dimension. Beispiele: Postenklasse und Postenunterklasse stellen Hierarchieebenen der Hierarchie Post dar
Rollup-Beziehung (Assoziation)	«hierarchieebene» Postenklasse «hierarchieebene» Postenunterklas	Die Assoziation wird im DFM verwendet, um hierarchische Rollup-Beziehungen darzustellen. Beispiele: Eine Postenunterklasse gehört zu einer Postenklasse.

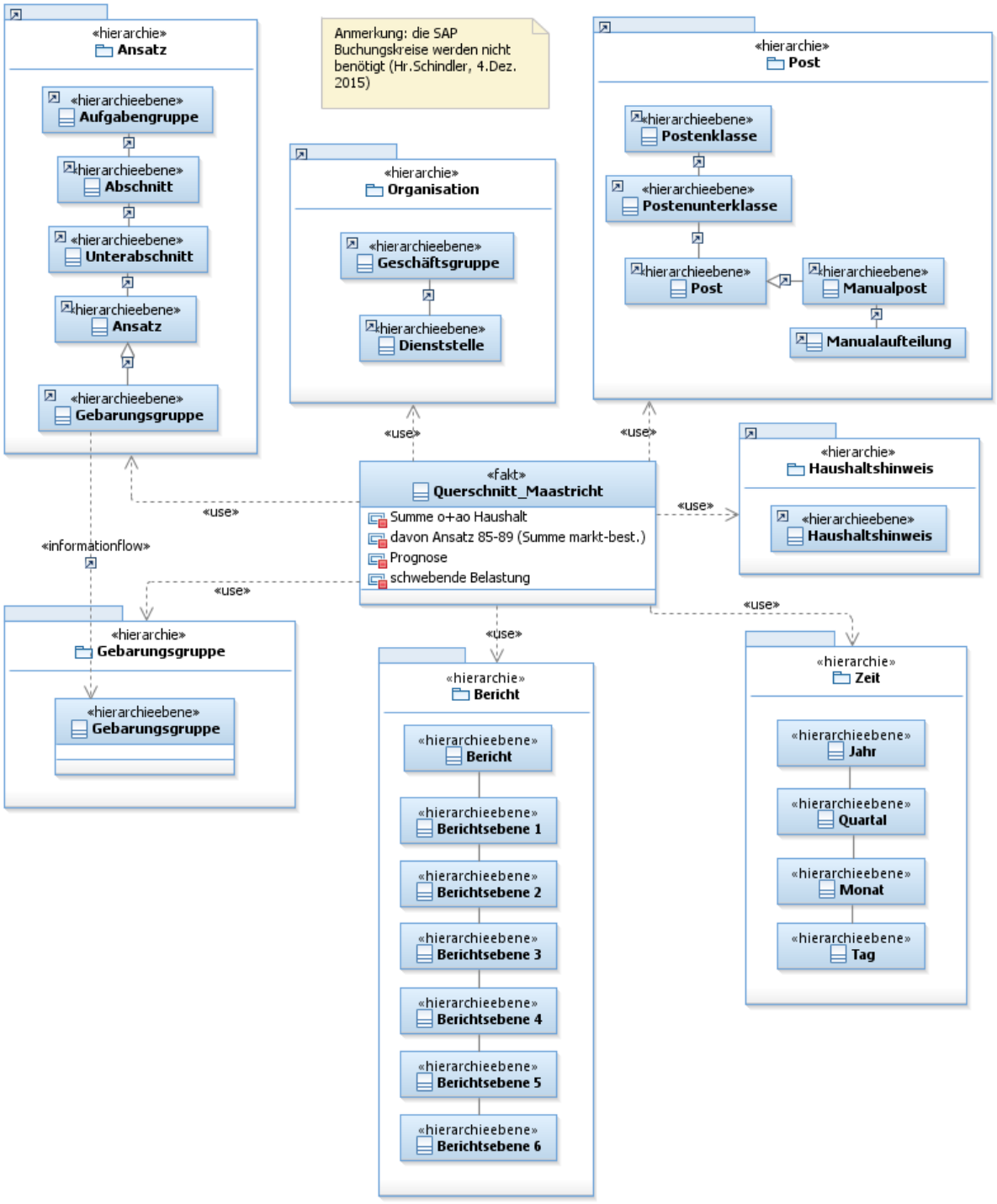
Dimension (Rechteck)		Für Dimensionen werden keine speziellen Modellelemente verwendet. Hierarchieebenen werden lediglich grafisch mit Rechtecken zusammengefasst. Beispiel: Die Dimension Post beinhaltet die Hierarchieebenen Postenklasse und Postenunterklasse.
Kennzahlen, Fakten (Klasse)		Fakten werden mit Klassen beschrieben. Beispiel: Der Fakt Verordnung enthält die Kennzahlen Nettobetrag und Rabattbetrag.

6.2.1 Richtlinien für das Dimension Fact Modell (DFM)

1. Das Dimension Fact Model (DFM) wird in fachliche Teilmodelle untergliedert. Der Name jedes DFM setzt sich aus einem Präfix und einem fachlichen Namen wie folgt zusammen: DFM_*fachlicherName* (z.B. DFM_QS_MAASTRICHT).
2. Pro Fakt ist ein DFM-Modell mit den zugehörigen Dimensionen zu bilden.
3. Um in die Modellelemente Hierarchieebenen und Fakt unterscheiden zu können, werden zwei Stereotypen eingeführt: hierarchieebene, fakt.
4. Die Beziehungen werden ohne Kardinalität und Beschriftung angegeben, da die Bedeutung der Beziehungen im DFM unmissverständlich ist.
5. Falls alternative Hierarchien dargestellt werden müssen, kann das UML-Element XOR herangezogen werden.
6. Im Modellelement Hierarchieebenen werden keine Attribute angegeben.
7. Für den fachlichen Bezeichner und Attribute werden die Groß- und Kleinschreibung verwendet.
8. Bezeichner und Attribute sind fachlich sprechend und möglichst ohne Abkürzungen zu bezeichnen (z.B. Haushaltshinweis und nicht HH).

   Ausnahme: Allgemein gültige Abkürzungen

6.2.2 Beispiel für das Dimension Fact Modell (DFM)



6.3 Modellierung von Kennzahlen

Kennzahlen werden als Attribute in den verschiedenen Schichten verwendet und im Information Governance Catalog fachlich beschrieben.

6.3.1 Richtlinien für die Kennzahlen

1. Kennzahlen werden nur abgebildet und nicht modelliert. Die Beschreibung und die Berechnungsvorschriften werden im Information Governance Catalog abgelegt.

2. Es ist darauf zu achten, dass Kennzahlen immer unternehmensweit einheitlich definiert werden (keine zwei Begriffe für die gleiche Kennzahl).
3. Kennzahlen werden auf Basis der HDS/IDS-Schicht ermittelt und in der IDS-Schicht gebildet und abgelegt.
4. Kennzahlen sind fachlich sprechend und möglichst ohne Abkürzungen zu bezeichnen.
   Ausnahme: Allgemein gültige Abkürzungen

6.4 Zusammenfassung Modellierung fachliche Modelle

Artefakt	**Bezeichnung**	**Werkzeug**
Domänenmodell	DOM_{fachlicher Name}	Data Architect
Dimension Fact Modell	DFM_{fachlicher Name}	Data Architect
Kennzahl im DFM-Modell	Fachlicher Name z. B. Nettobetrag_Euro oder Diagnose_Anzahl	Information Governance Catalog Die Kennzahlen sind in den Metadaten zu erfassen, zu beschreiben und zu verlinken.

7 Technische Modelle

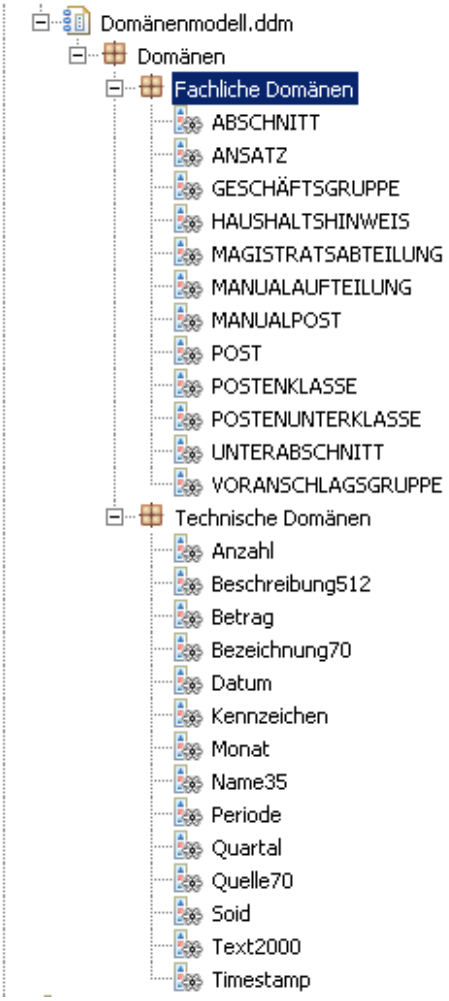
7.1 Datentypenmodell

Das Modell zur Datentypendefinition wird im Data Architect als Domänenmodell umgesetzt (dies ist nicht mit dem fachlichen Domänenmodell zu verwechseln). In diesem Modell werden die Datentypen für die physische Umsetzung der logischen Datenmodelle definiert. Da als Grundsatz gilt, dass logische Datenmodelle plattformunabhängig sein müssen, werden die datenbankspezifischen Informationen in einem gesonderten Modell (dem Datentypenmodell) allgemein und fachlich bezogen definiert und bei der logischen zur physischen Umsetzung automatisch zugrunde gelegt.

7.1.1 Richtlinien für das Datentypenmodell

1. Allgemein
   - Es gibt nur ein Datentypenmodell für das gesamte Enterprise Business Intelligence
   - Das Datentypenmodell wird in fachliche und technische Domänen unterteilt
2. Fachliche Domäne
   - Es werden alle verwendete Business Keys und deren Datentypen beschrieben
3. Technische Domäne
   - Es werden die technischen Datentypen beschrieben.
   - Der Name der definierten Datentypen setzt sich aus einem Bezeichner und der Stringlänge bei Strings zusammen. (z.b. Name512)

7.1.2 Beispiel für ein Datentypenmodell



8 Logische Modelle

	Modell	Beschreibung
Logische Modelle	Logisches Datenmodell der DMS	• Information Governance Catalog (Metadata Repository) • DataArchitect (Übernahme ins Metadata Repository: Metadata Asset Manager)
	Logisches Datenmodell der IDS (als Data Vault)	• Information Governance Catalog (Metadata Repository) • DataArchitect (Übernahme ins Metadata Repository: Metadata Asset Manager)
	Logisches Datenmodell der HDS	• Information Governance Catalog (Metadata Repository) • DataArchitect (Übernahme ins Metadata Repository: Metadata Asset Manager)

8.1 Architekturschichten

Die logischen Datenmodelle der Architekturschichten werden 1:1 in physische Datenmodelle abgebildet. Drei Architekturschichten sind dabei zu berücksichtigen:

Architekturschicht	Name des Schemas	Anmerkung
Data Mart Schicht	DMS	Aus dem logischen DMS-Datenmodell wird über den IBM InfoSphere Data Architect das physische Datenbankschema generiert
Integrierte Datenschicht	IDS	Aus dem logischen IDS-Datenmodell wird über den IBM InfoSphere Data Architect das physische Datenbankschema generiert.
Historisierte Datenschicht	HDS	Aus dem logischen HDS-Datenmodell wird über den IBM InfoSphere Data Architect das physische Datenbankschema generiert.

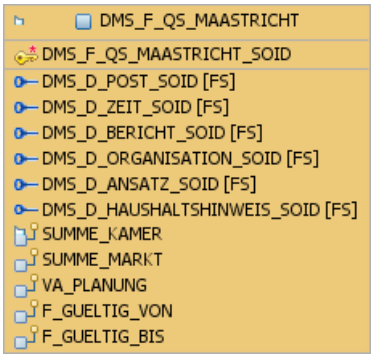
8.2 Allgemeine Festlegungen zu den logischen Datenmodelle

1. Business Logik wird generell in der IDS abgebildet (Business Satellit) und nicht in der DMS
2. Abkürzungen sind generell zu vermeiden, da diese je nach Domäne eine andere Bedeutung haben können
3. Ausnahmen sind allgemein gültige Abkürzungen
4. Alle Konstrukte innerhalb der logischen Modelle werden in Großbuchstaben geschrieben
5. Alle Attributdatentypen sind durch einen Eintrag des Datentypenmodelles abzubilden. Statische Datentypdefinition ist nicht erlaubt
6. Alle fachlichen Attribute und Kennzahlen müssen durch einen verbundenen Term im IGC dokumentiert werden
7. In den Entitätsbezeichnungen sind keine Umlaute erlaubt

8.3 Modellierung der Data Mart Schicht (DMS)

In der Data Mart Schicht (DMS) werden Fakten- und Dimensionstabellen als logische Datenmodelle (Entity-Relationship) dargestellt. Aus diesem Modell können die physischen Datenstrukturen erstellt werden. Folgende Modellelemente werden verwendet:

Modellelement	Symbol	Erläuterung

Dimension (Entität)		Eine Dimension wird als Entität (Tabelle) dargestellt (grün) Konventionen: • Der Name einer Dimension beginnt mit dem Präfix DMS_D_.
Fakten (Entität)		Fakten werden als Entität (Tabelle) dargestellt, die Kennzahlen enthält und Dimensionen referenziert. (orange) Konventionen: • Der Name eines Fakts beginnt mit dem Präfix DMS_F_.
Beziehung zwischen Dimension und Fakt (Relationship)		Beziehungen zwischen Dimensionen und Fakten werden als Beziehungen dargestellt und ergeben ein klassisches Star-Schema.

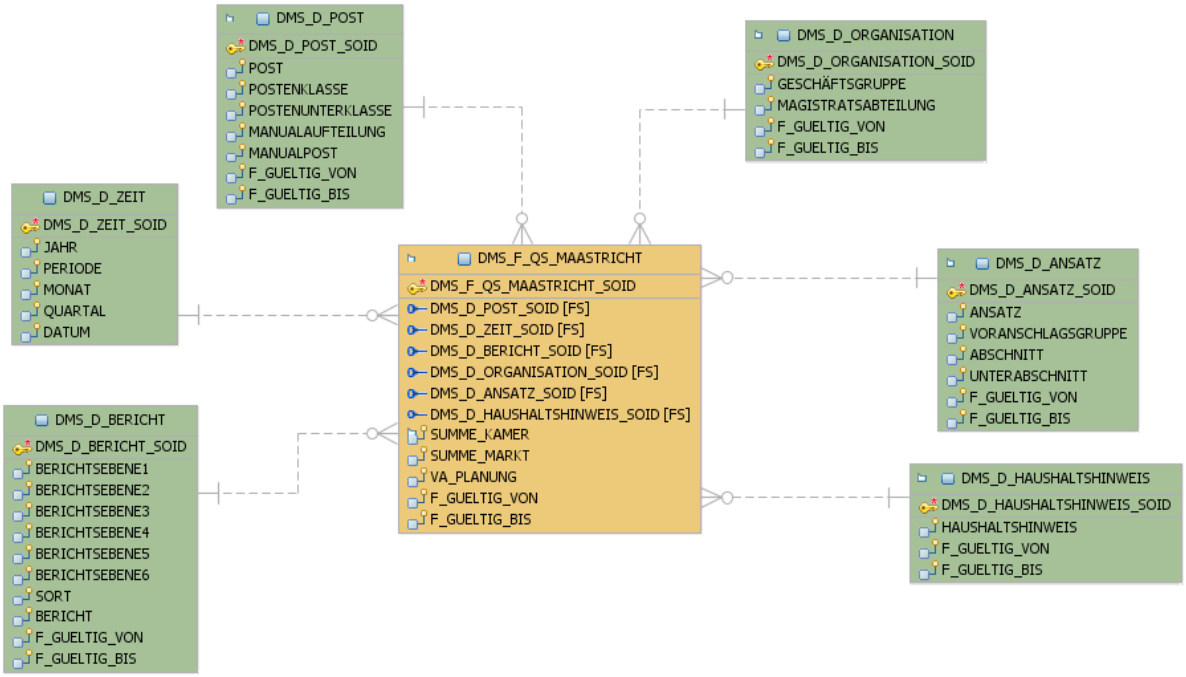
8.3.1 Richtlinien für das DMS-Modell

1. Allgemeines
   Da aus dem DMS-Modell ein physisches Datenbankschema abgeleitet wird, sind die Regeln der Datenbank bei der Namensvergabe einzuhalten:
   a. keine Umlaute
   b. als Sonderzeichen ist nur „_“ erlaubt
   c. Berücksichtigung der max. Länge von Bezeichnern.
2. Modellname
   Das DMS-Modell wird in Teilmodelle untergliedert. Der Name jedes Teilmodells setzt sich aus einem Präfix und einem fachlichen Namen wie folgt zusammen: DMS_{*fachlicher Name}* (z.B. DMS_ Finanzen). Ein Teilmodell der DMS entspricht einem oder mehreren DFM-Teilmodell(en).
3. Tabellenname
   - Wörter werden dabei mit „_“ getrennt.
   - Bezeichner für Dimensionen und Fakten werden aus zwei Präfixen (DMS_für die Bezeichnung der Schicht und D_ für Dimensionen oder F_ für Fakten) und einem sprechenden Namen gebildet (z.B. DMS_D_POSTEN, DMS_F_QS_MAASTRICHT).
4. Primärschlüssel
   Für Dimensions- und Faktentabellen werden jeweils eigene Primärschlüssel vergeben.
5. Beziehungen

Beziehungen werden nicht beschriftet. Die Kardinalität wird in „Krähenfußnotation“ angegeben.

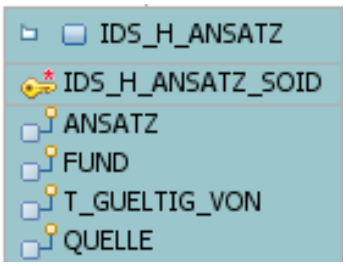
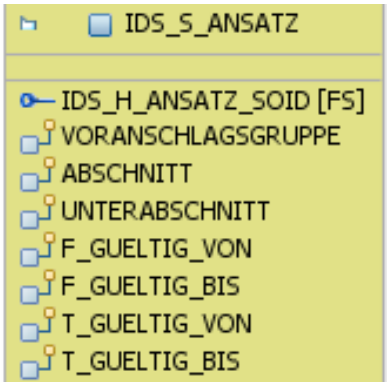
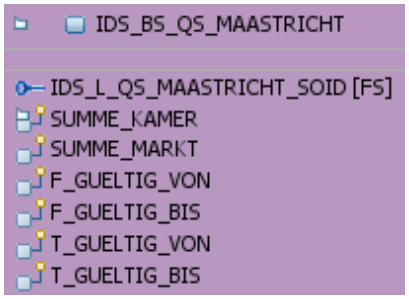
6. Attribute
   Für jedes Attribut gilt folgende Festlegung:
   - Wörter werden dabei mit „_“ getrennt.
   - Jedes Attribut erhält ein Suffix, sofern dieses nicht schon vorliegt.
   - Die zulässigen Suffixe für die Attribute sind im Kapitel Zulässige Suffixe der Attribute in den logischen und physischen Datenmodellen festgelegt und beschrieben.
   - Schlüsselattribute sollten bei der ersten Anlage des Modells immer am Anfang der Entität stehen.
   - Die technischen Attribute sind im Modell an das Ende der Entität zu stellen.
7. Datenquelle
   Die Datenquelle der DMS ist ausschließlich die IDS
8. Farbgebung
   Die Entitäten sind im IDA mit folgenden Farben zu kennzeichnen:
   - Dimension: grün
   - Fakten: orange

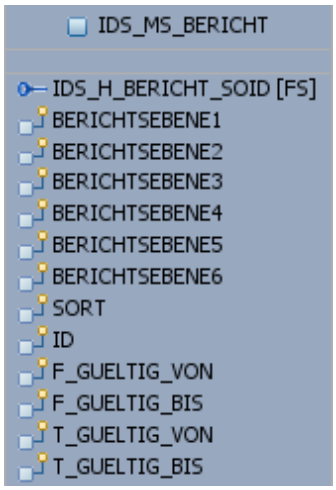
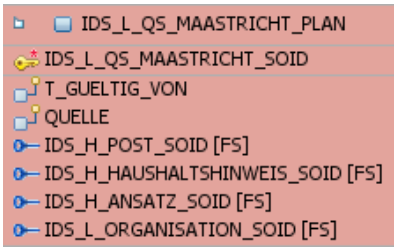
8.3.2 Beispiel für ein DMS Modell



8.4 Modellierung der Integrierten Datenschicht (IDS)

In der Integrierten Datenschicht (IDS) werden alle Daten des unternehmensweiten Data Warehouse gesammelt und historisiert abgelegt. Es werden auch alle Kennzahlen, die in der Data Mart Schicht erforderlich sind, bereits in der IDS errechnet. Die IDS ist ein logisches Datenmodell in ER-Notation, welches nach Data Vault geführt wird. Folgende Modellelemente werden verwendet:

Modellelement	Symbol	Erläuterung
Hub (Entität)		Ein Hub ist eine Tabelle, die ein Geschäftsobjekt repräsentiert. Sie enthält nur Business-Keys, die sich nicht verändern. In einem Hub wird ein künstlicher Primärschlüssel definiert. Zusätzlich wird der Hub durch administrative Attribute, die das Ladedatum und die Datenquelle angeben, ergänzt. (Zyanblau)
Raw Satellit (Entität)		Im Satellit werden die eigentlichen Informationen eines Geschäftsobjekts (Hub) in historisierter Form verwaltet. Pro Hub können mehrere Satelliten definiert werden, die verschiedene Kontexte darstellen. Ein Satellit wird genau von einem Hub referenziert und enthält Attribute zu einem Geschäftsobjekt in historisierter Form. Dabei kann es pro SOID immer nur einen zeitlich gültigen Satz geben. (PK = SOID + F_GUELTIG_VON + T_GUELTIG_VON) Mit einem Ende Datum als zusätzliches Attribut wird die Historisierung gesteuert. Dieses gibt an, ob der Datensatz noch gültig ist oder bereits „beendet“ wurde. In diesem Satellit wird keinerlei Business Logik abgebildet, sondern lediglich eine Vorselektion für die, auf diesen Satelliten basierenden, Business Satelliten durchgeführt. Zusätzlich wird der Satellit durch administrative Attribute, die das technische Ladedatum angeben, erweitert. (gelb)
Business Satellit (Entität)		Im Business Satelliten werden die Informationen des basierenden Raw Satelliten mit Business Logik angereichert. Ein Business Satellit kann mehrere Raw Satelliten als Basis haben, allerdings nur mit einem Hub verbunden sein. Zudem enthält der BS fachliche Datumsangaben, um das Geschäftsobjekt historisiert abzulegen. Zusätzlich wird der Satellit durch administrative Attribute, die das

		technische Ladedatum angeben, erweitert. (Magenta)
Multiactive Satellit (Entität)		Der Multiactive Satellit ist ein normaler Satellit in dem allerdings zu einem Zeitpunkt pro SOID mehrere Einträge gültig sein können. (Beispiel: Zu einem Kunden können zu einem Zeitpunkt mehrere Telefonnummern gültig sein) (blau)
Link (Entität)		Beziehungen zwischen Geschäftsobjekten werden als eigene Objekte dargestellt. Ein Link wird von mehreren Hubs oder anderen Links referenziert. Zusätzlich wird der Link noch um das Ladedatum als administratives Attribut erweitert. (rot)
Beziehungen (Relationship)		Beziehungen sind logisch zu sehen. Fachliche Beziehungen werden über Links abgebildet. Logische Beziehungen existieren • zwischen Hubs und (Raw+ Business) Satelliten, • zwischen Links und (Raw+ Business) Satelliten, • zwischen Hubs und Links und • zwischen Links und Links.

8.4.1 Richtlinien für das IDS-Modell

1. Allgemeines
   Da aus dem IDS-Modell ein physisches Datenbankschema abgeleitet wird, sind die Regeln der Datenbank bei der Namensvergabe einzuhalten:
   d. keine Umlaute
   e. als Sonderzeichen ist nur „_“ erlaubt
   f. Berücksichtigung der max. Länge von Bezeichnern.

2. Modellname
   Das IDS-Modell wird in Teilmodelle gegliedert. Der Name jedes Teilmodells setzt sich aus einem Präfix und einem fachlichen Namen wie folgt zusammen: IDS_*fachlicher Name* (z.B. IDS_FINANZEN). Ein Teilmodell der IDS entspricht einem DOM-Teilmodell (z.B. zum Modell IDS_ FINANZEN gibt es auch ein Modell DOM_ FINANZEN).

3. Modellierungsmethode
   Die Modellierung der IDS erfolgt als ER-Modell nach Data Vault. Wobei Abweichungen der Modellierungsmethode je nach Anforderungen möglich sind. Allerdings entsprechend dokumentiert werden müssen.
4. Tabellennamen
   Wörter werden mit „_“ getrennt.
   - **Hubs** besitzen den **Präfix IDS_H_** gefolgt von einem sprechenden Namen (z.B. IDS_H_POSTEN).
   - **Satelliten** besitzen den **Präfix IDS_S_** gefolgt von einem sprechenden Namen (z.B. IDS_S_POSTEN, S_ANSATZ).
   - **Business Satelliten** sind spezielle Satelliten, die Kennzahlen beinhalten. Sie beginnen mit dem **Präfix** IDS_**BS**_ (steht für „Business Satellit“) gefolgt von einem sprechenden Namen (z.B. IDS_BS_QS_MAASTRICHT)
   - **Links** besitzen den **Präfix IDS_L_** gefolgt von einem sprechenden Namen, der die Bedeutung der Beziehung widerspiegelt (z.B. IDS_L_QS_MAASTRICHT).
5. Datumsangaben/Zeitreihenbildung
   In der IDS ist grundsätzliche eine Verarbeitung von Timestamps zu verwenden. Zeiträume sind immer als links-offene Intervalle zu definieren. Sollte das Quellsystem (HDS) eine andere Zeitreihenlogik bzw. Datumsformate verwenden, werden diese auf IDS-Methodik umgesetzt.
6. Attribute
   Für jedes Attribut gilt folgende Festlegung:
   g. Wörter werden mit „_“ getrennt.
   h. Jedes Attribut erhält ein Suffix, sofern dieser nicht schon vorliegt.
   i. Die zulässigen Suffixe sind im Kapitel Zulässige Suffixe der Attribute in den logischen und physischen Datenmodellen festgelegt und beschrieben.
   j. Schlüsselattribute sollten bei der ersten Anlage des Modells immer am Anfang der Entität stehen.
   k. Die technischen Attribute sind im Modell an das Ende der Entität zu stellen.
7. Standardattribute
   Folgende Standardattribute sind zu führen bei:
   - Hubs
     - Primärschlüssel: IDS_H_*Hubname*_SOID (z.B. IDS_H_POSTEN_SOID)
     - Business Key: POSTENNUMMER
     - Ladedatum: T_GUELTIG_VON (timestamp)
     - Datenquelle (Name der Quelltabelle): QUELLE
   - Satelliten
     Achtung: Satelliten besitzen keinen eigenen Primärschlüssel
     - Foreign Key: Fremdschlüssel auf Hub oder Link
     - 1…n Attribut(e)
     - Fachliches Ladedatum: F_GUELTIG_VON
     - Fachliches Enddatum: F_GUELTIG_BIS
     - Technisches Ladedatum: T_GUELTIG_VON
     - Technisches Enddatum: T_GUELTIG_BIS, wenn der Datensatz gültig ist wird mit „31.12.2999 00:00:00.0“ gefüllt, ansonsten mit dem Datum an dem dieser Satz beendet wurde. (Das Enddatum wird im ETL-Job gesetzt, nicht in der Datenbank oder an andere Stelle)
   - Links
     - Primärschlüssel: IDS_L_*Linkname*_SOID (z.B. IDS_L_QS_MAASTRICHT_SOID)
     - Foreign Key1: Fremdschlüssel auf Hub 1
     - Foreign Key2: Fremdschlüssel auf Hub 2

- Ladedatum: T_GUELTIG_VON

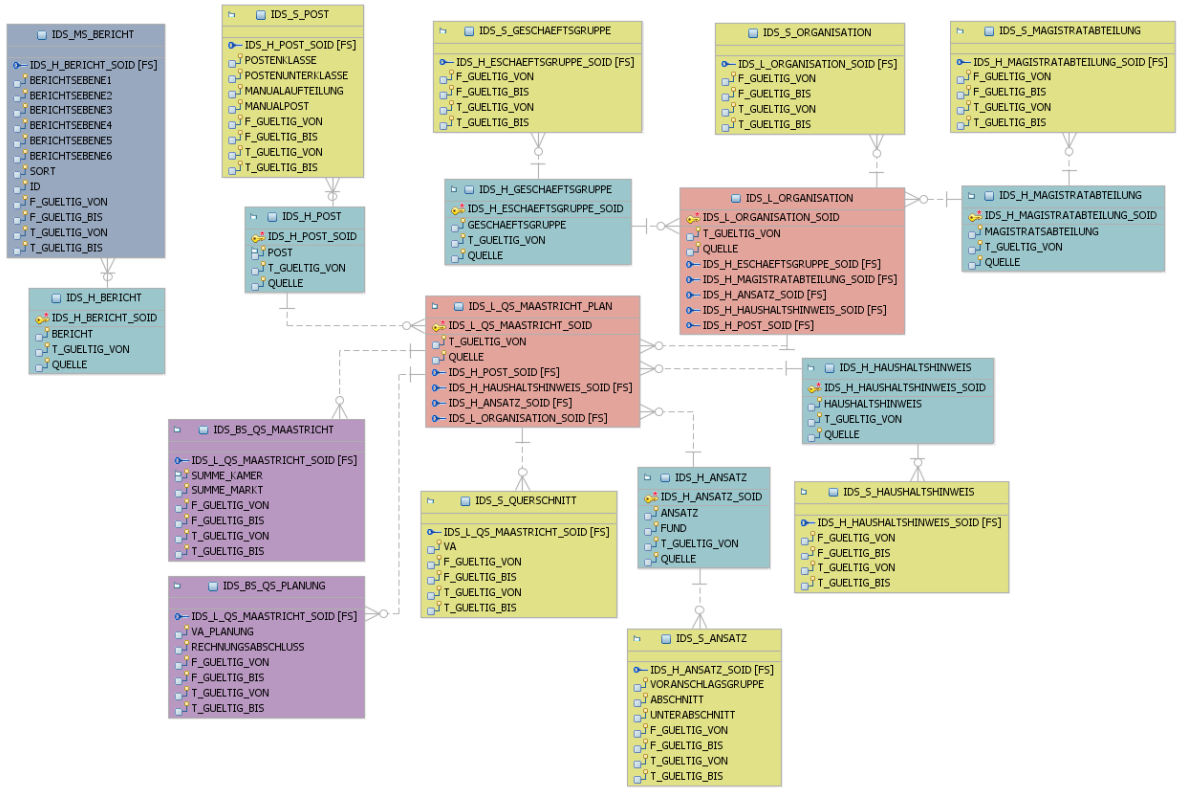
8. Datenquelle
   Für die Datenquelle der IDS gelten folgende Festlegungen:
   - Die Datenquelle für die IDS, ist **ausschließlich** die HDS. Die Struktur der IDS ist dabei jedoch unabhängig von der Quellstruktur der HDS.
   - Alle in der IDS benötigten Attribute, ggf. auch aus verschiedenen operativen Systemen, werden über Tabellen der HDS zur Verfügung gestellt.

9. Datenübernahme
   Bei der Datenübernahme sind folgende Vorgaben zu berücksichtigen:
   - Die aus der HDS zu übernehmende Attributnamen **müssen** in sprechende Attributnamen umgesetzt werden.
   - Festlegung:
     - Wahr = 1 und Falsch = 0.
     - J = 1 und N = 0

10. Datenbestand
    In der IDS gilt der **Grundsatz der Unveränderlichkeit**, d. h. Daten, die einmal in der IDS gespeichert wurden, werden nicht mehr verändert. Eine Ausnahme bildet das bis-Datum des Gültigkeitsstempels (T_GUELTIG_VON und F_GUELTIG_VON). Bei Korrekturen werden nur neue Sätze hinzugefügt (Insert / Append).

11. Farbgebung
    Die Entitäten sind im IDA mit folgenden Farben zu kennzeichnen:
    l. Hubs (H): zyanblau
    m. Satellit (S): gelb
    n. Business Satellit (BS): magenta
    o. Multiactive Satellit (MS): blau
    p. Link (L): rot

8.4.2 Beispiel für ein IDS-Modell



8.5 Modellierung der Historischen Datenschicht (HDS)

1. Allgemeines
   Da aus dem HDS-Modell ein physisches Datenbankschema abgeleitet wird, sind die Regeln der Datenbank bei der Namensvergabe einzuhalten:
   q. keine Umlaute
   r. als Sonderzeichen ist nur „_“ erlaubt
   s. Berücksichtigung der max. Länge von Bezeichnern.
2. Modellname
   Das HDS-Modell wird in Teilmodelle gegliedert. Der Name jedes Teilmodells setzt sich aus einem Präfix und einem fachlichen Namen wie folgt zusammen: HDS_*fachlicher Name* (z.B. HDS_FINANZEN).
3. Datenquelle
   Als Quelle der HDS dient je nach angebundener Datenbank ein anderes Quellsystem. Die HDS ist dabei ein 1:1 Abzug des Quellsystems. Ausnahmen sind im Folgenden beschrieben.
4. Modellierungsmethode
   Die Modellierung der HDS erfolgt als ER-Modell, dabei wird die Modellierungsmethode aus dem Quellsystem übernommen.
5. Tabellennamen
   Da die Quellsysteme zum Teil sehr kryptische Namensgebungen der Tabellen haben können, werden die Namen fachlich sinnvoll in Bezug auf den Inhalt gewählt.
6. Attribute
   Da die Quellsysteme zum Teil sehr kryptische Namensgebungen der Attribute haben können, werden die Namen fachlich sinnvoll in Bezug auf den Inhalt gewählt.
7. Standardattribute
   Folgende Standardattribute sind zu führen:
   - Ladedatum: HDS_T_GUELTIG_VON (Timestamp)
   - Ladekennzeichen: HDS_LADE_KZ (Je nach Vorhandensein des Datensatzes (I)nsert, (U)pdate oder (D)elete)
8. Datenübernahme
   Bei der Datenübernahme sind lediglich Name und Datentyp abzuändern und fachlich sprechend zu benennen. Bei der Übernahme darf keine fachliche Logik angewendet werden.
9. Datenbestand
   In der HDS gilt der **Grundsatz der Unveränderlichkeit**, d. h. Daten, die einmal in der HDS gespeichert wurden, werden nicht mehr verändert. Im Gegensatz zu den anderen Schichten gibt es keine Ausnahmen. Bei Korrekturen werden nur neue Sätze hinzugefügt (Insert / Append) und mit einem entsprechenden Ladekennzeichen versehen.

9 Physische Modelle

9.1 Allgemeine Festlegungen zu den physischen Datenmodelle

1. Alle fachlichen Informationen, die datenbankunabhängig sind, werden ausschließlich aus dem logischen Datenmodell generiert. Ausgenommen sind folgende Konstrukte:

- Indizes
- Datenbankbenutzer
- Tabelspaces
- Datenbankpartitionierung

2. Namenskonventionen und Bezeichnungen werden aus dem logischen Datenmodell übernommen und behalten ihre Gültigkeit.
3. Abkürzungen sollten generell vermieden werden. Ausnahmen sind allgemein gültige Abkürzungen und Abkürzungen, die technisch nötig sind (Oracle: max. 30 Zeichen)

4. Alle Konstrukte innerhalb der physischen Modelle werden in Großbuchstaben geschrieben
5. Alle Attributdatentypen sind durch einen Eintrag des Datentypenmodelles umgesetzt und im logischen Datenmodell definiert. Eine Datentypdefinition im physischen Modell ist nicht erlaubt.
6. Generell sind alle Konstrukte im Nutzerbereich <Schicht>_ADMIN einzurichten
7. Benutzerbereiche:
   Es gibt einen default Benutzerbereich pro Architekturebene, der wie folgt benannt ist: <Architekturschicht>_ADMIN (Beispiel: EBIDMS_ADMIN). Zudem gibt es mind. einen weiteren Benutzer dem alle Berechtigungen für Tabellen, Sequenzen usw. zugeordnet sind. Dabei sind folgende Rechte als default zu setzen:
   - Tabellen: D/I/S/U
   - Sequence: S

   Es ist zu beachten, dass Tabellen über den Standarduser <Schemaname>_ADMIN angelegt werden, weitere Arbeiten, allerdings über den zusätzlich angelegten <Schemaname>_USER erfolgen.
8. Für jedes Schema gibt es zwei Tablespaces <Schemaname>_D und <Schemaname>_I (Beispiel: EBIDMS_D oder EBIDMS_I)
   - Im Tablespaces <Schemaname>_D werden alle Daten abgelegt (Tabellen, …)
   - Im Tablespaces <Schemaname>_I werden alle Indizes abgelegt

9.2 Erstellung des physischen Datenbankschemas für die DMS

Aus dem logischen DMS-Datenmodell wird mit dem IBM InfoSphere Data Architect das physische Datenbankschema generiert.

9.2.1 Richtlinien für das DMS-Modell

1. **Datenbankname**:
   EBI
2. **Default Benutzer**:
   EBIDMS_ADMIN
3. **Weitere Benutzer**:
   EBIDMS_USER: Tabellen: D/I/S/U
4. **Indizes**
   Die jeweiligen Tabellen erhalten folgende Indizes:
   Dimensionen
   SOID (<Tabellenname>_SOID_IDX) bestehend aus der SOID
   PK (<Tabellenname>_PK_IDX) bestehend aus der SOID und F_GUELTIG_VON
   Fakt

   SOID (<Tabellenname>_SOID_IDX)
   PK (<Tabellenname>_PK_IDX) bestehend aus der SOID und F_GUELTIG_VON
5. **Tablespaces**
   - EBIDMS_D für alle Tabellen
   - EBIDMS_I für alle Indizes

9.3 Erstellung des physischen Datenbankschemas für die IDS

Aus dem logischen IDS-Datenmodell wird mit dem IBM InfoSphere Data Architect das physische Datenbankschema generiert.

9.3.1 Richtlinien für das IDS-Modell

1. **Datenbankname**:
   EBI
2. **Default Benutzer**:
   EBIIDS_ADMIN
3. **Weitere Benutzer**:
   EBIIDS_USER: Tabellen: D/I/S/U, Sequenzen: S
4. **Indizes**
   Die jeweiligen Tabellen erhalten folgende Indizes:
   - Satelliten:
     - PK (UNIQUE) (<Tabellenname>_PK_IDX) bestehend aus der SOID, F_GUELTIG_VON und T_GUELTIG_VON
   - Hubs
     - SOID (UNIQUE) (<Tabellenname>_SOID_IDX) bestehend aus der SOID
     - BK (UNIQUE) (<Tabellenname>_BK_IDX) bestehend aus allen Business Keys
   - Links
     - SOID (UNIQUE) (<Tabellenname>_SOID_IDX) bestehend aus der SOID
     - BK (UNIQUE) (<Tabellenname>_BK_IDX) bestehend aus allen Business Keys
5. **Tablespaces**
   - EBIIDS_D für alle Tabellen
   - EBIIDS_I für alle Indizes

9.4 Erstellung des physischen Datenbankschemas für die HDS

Aus dem logischen HDS-Datenmodell wird mit dem IBM InfoSphere Data Architect das physische Datenbankschema generiert.

9.4.1 Richtlinien für das HDS-Modell

6. **Datenbankname**:
   EBI
7. **Default Benutzer**:
   EBIHDS_ADMIN
8. **Weitere Benutzer**:
   EBIHDS_USER: Tabellen: D/I/S/U
9. **Indizes**
   Die jeweiligen Tabellen erhalten folgende Indizes:
   - Tabellen:
     - HDS_T_GUELTIG (<Tabellenname>_T_GUELTIG_IDX) bestehend aus HDS_T_GUELTIG_VON
     - Primary Key (UNIQUE) (<Tabellenname>_PK_IDX) bestehend aus dem Primary Key
10. **Tablespaces**
    - EBIHDS_D für alle Tabellen
    - EBIHDS_I für alle Indizes

10 Literatur

Hiebl J., Hörak K., Linner K., Neuböck T., Raab J., Weißenböck A. (2014). *Agile modellorientierte DWH-Entwicklung mit IBM InfoSphere.* Linz: solvistas GmbH.

Neuböck T., Raab J., Weißenböck A. (2014). *Agile modellorientierte DWH-Entwicklung – Modellebenen und Architektur.* Linz: solvistas GmbH.

Neuböck T., Raab J., Weißenböck A. (2014). *Eine modellgetriebene Vorgehensweise in der Data Warehouse Entwicklung.* Linz: solvistas GmbH.