

# Zusammenfassung Data Warehousing FS2018

Alex Neher

June 14, 2018

## Inhalt

<b>1 Die Notwendigkeit von Data Warehouses</b>	<b>6</b>
1.1 Entscheidungsunterstützung (Skript S15) . . . . .	6
1.2 Ungenügen der "gängigen" Datenhaltung (Skript S17) . . . . .	7
1.3 Ungenügen der operativen Datenbanken für Entscheide (Skript S18) . . . . .	7
1.4 SQL-Abfragen für Management-Zwecke (Skript S18ff) . . . . .	7
1.5 OLAP vs OLTP (Skript S24) . . . . .	8
<b>2 Daten vs. Informationen vs. Wissen vs. Weisheit (Skript S26)</b>	<b>9</b>
<b>3 Das Data Warehouse</b>	<b>11</b>
3.1 Definition Data-Warehouse (Skript S 37) . . . . .	12
3.2 Bestandteile eines Data-Warehouses (Skript S43) . . . . .	12
3.3 Welche Datenbank für welche Tasks? (Buch S40ff) . . . . .	13
<b>4 Referenzarchitekturen</b>	<b>14</b>
4.1 Bestandteile des Referenzmodells (Skript S45ff) . . . . .	14
4.1.1 Datenquelle (Skript S46) . . . . .	14
4.1.2 Arbeitsbereich / Staging Area (Skript S46 / Buch S55) . . . . .	15
4.1.3 Basis-DB / Operational Data Store (ODS) (Skript S47 / Buch S58) . . . . .	15
4.1.4 Ableitungsdatenbank (=Enterprise-)Data Warehouse) (Skript S47/Buch S64) . . . . .	16
4.1.5 Auswertungsdatenbank (Data-Mart) (Skript S50/Buch S67) . . . . .	16
4.1.6 Integrationsbereich (Skript S54) . . . . .	18
4.1.7 Auswertungsbereich (Buch S72) . . . . .	18
4.1.8 Verwaltungsbereich (Skript S272) . . . . .	19
4.2 Referenzmodelle in die Praxis umgesetzt (Skript S63) . . . . .	20
4.2.1 "Hub and Spoke"-Architektur . . . . .	20
4.2.2 Star-Architektur . . . . .	20
4.2.3 Snowflake-Architektur . . . . .	21
4.2.4 Galaxy-Architektur . . . . .	21
4.2.5 Beispiel zur Erstellung eines Datenbankschemas . . . . .	22
<b>5 ETL</b>	<b>25</b>
5.1 Extraktion (Skript S 55/Buch S56) . . . . .	25
5.2 Transformation (Skript S56/Buch S57) . . . . .	25
5.3 Load (Skript S58/Buch S58) . . . . .	26
5.4 Monitor (Skript S60/Buch S54) . . . . .	27

<b>6 Unstrukturierte Daten (Skript S72)</b>	<b>28</b>
6.1 Nutzen . . . . .	28
6.2 Herausforderungen . . . . .	28
6.3 Analyse (Skript S74ff/Buch S131ff) . . . . .	28
6.3.1 NLP - Natural Language Processing . . . . .	28
6.3.2 Data Mining . . . . .	28
6.3.3 Text Mining . . . . .	28
6.3.4 Klassifikation . . . . .	29
6.3.5 Regression . . . . .	29
6.3.6 Abhängigkeitsentdeckung / Assoziationsanalyse . . . . .	30
6.3.7 Clustering . . . . .	31
6.3.8 Visualisierungstechnik . . . . .	32
6.3.9 Fallbasierte Systeme . . . . .	32
6.3.10 Entscheidungsbaumverfahren (Buch S136) . . . . .	33
6.4 Architekturen (Skript S84/Buch S163ff) . . . . .	34
6.4.1 Keine physische Integration . . . . .	34
6.4.2 Pysische aber nicht logische Integration . . . . .	34
6.4.3 Physische und logische Integration . . . . .	35
6.4.4 Beispiel . . . . .	36
<b>7 Informations- und Datenqualität (Skript S95)</b>	<b>38</b>
7.1 Definition . . . . .	38
7.2 Ursachen und Orte von Qualitätsmängel . . . . .	38
7.3 Messbarkeit von Datenqualität . . . . .	39
7.4 Verbesserung der Datenqualität . . . . .	41
7.4.1 Daten verstehen . . . . .	41
7.4.2 Daten verbessern . . . . .	43
7.4.3 Daten steuern . . . . .	45
<b>8 Historisierung (Skript S144/Buch S195)</b>	<b>46</b>
8.1 SCD Typ 0 - Keine Historisierung, keine Aktualisierung . . . . .	46
8.2 SCD Typ I - Keine Historisierung, Aktualisierung . . . . .	46
8.3 SCD Typ II - Vollhistorisierung . . . . .	46
8.4 SCD Typ III - Historisierung mit neuem Attribut (Spalte) . . . . .	47
8.5 Tupelversionierung . . . . .	47
<b>9 Multidimensionale Datenmodellierung (Script S153 / Buch S201)</b>	<b>48</b>
9.1 Nutzen von MDDB . . . . .	48
9.2 Kennzahlen, Fakten und Dimensionen . . . . .	50
9.3 OLAP Würfel-Paradigma . . . . .	51
9.3.1 Die 10 OLAP Würfel-Paradigmen . . . . .	51
9.3.2 Grundoperationen am Würfel . . . . .	52
9.4 Dimensionstruktur, -klassifizierung und -hierarchie . . . . .	53
9.4.1 Die 9 Regeln der Klassifikationshierarchie . . . . .	53
9.5 Dünnbesetztheit . . . . .	54
9.6 Modelle in multidimensionalen Datenbanken . . . . .	55
9.6.1 semantisches Modell . . . . .	55
9.6.2 logisches Modell . . . . .	55
9.6.3 physisches Modell . . . . .	55
9.7 OLAP-Modelle in multidimensionalen Datenbanken . . . . .	55
9.8 CUBE, ROLLUP und GROUPING SETS (Skript S177) . . . . .	57

<b>10 MDX (Multidimensional Expressions) (Skript S196)</b>	<b>58</b>
10.1 MDX SELECT . . . . .	58
10.2 DAX (Data Analysis Expression) . . . . .	59
<b>11 Planung der Systemtechnischen Architektur (Skript S205)</b>	<b>60</b>
11.1 Realtime Data-Warehouse Systeme (Skript S206) . . . . .	60
11.2 In-Memory Computing (Skript S207/Buch S174) . . . . .	60
11.2.1 RAM-Management . . . . .	60
11.2.2 Zeilenorientiertes vs. Spaltenorientiertes Speichern . . . . .	61
11.2.3 Chancen und Risiken des In-Memory Computing . . . . .	63
11.3 Cloud-basiertes Data-Warehouse (Skript S218) . . . . .	63
11.3.1 Dictionary Daten-Komprimierung . . . . .	64
<b>12 Hadoop (Skript S229)</b>	<b>65</b>
12.1 Anwendungen . . . . .	65
12.2 Motivation . . . . .	66
12.3 Möglichkeiten . . . . .	66
12.4 Aufbau . . . . .	66
12.5 Architektur . . . . .	66
12.6 MapReduce . . . . .	67
12.7 Speichern mit HDFS . . . . .	69
12.8 Anwendungen . . . . .	69
<b>13 Speicherarten (Skript S256)</b>	<b>70</b>
13.1 Objektspeicher (Skript S258) . . . . .	71
13.1.1 Definiton . . . . .	71
13.1.2 Motivation . . . . .	71
13.1.3 Arhitektur . . . . .	72
<b>14 XMLA (Skript S268)</b>	<b>73</b>
14.1 Execute . . . . .	73
14.2 Discover . . . . .	74
<b>15 Metadaten (Skript S271/Buch S339)</b>	<b>75</b>
15.1 Definition . . . . .	75
15.2 Meta-Metamodell . . . . .	75
15.3 Data Warehouse Manager . . . . .	76
15.4 Metadaten Manager . . . . .	76
15.5 Unterscheidung von Metadaten . . . . .	76
15.5.1 Technische Metadaten . . . . .	76
15.5.2 Geschäftliche Metadaten . . . . .	77
15.5.3 Prozessuale Metadaten . . . . .	77
15.6 Metadaten-Strategie . . . . .	77
15.6.1 Wie . . . . .	77
15.6.2 Wer . . . . .	78
15.7 Normierung der Metadaten . . . . .	78
<b>16 Data Warehouse Strategie (Skript S286)</b>	<b>80</b>
16.1 Reifegradmodelle . . . . .	80

## Abbildungsverzeichnis

1.1	Beispiel eines Expertensystems . . . . .	6
2.1	DIKW-Pyramid . . . . .	9
2.2	Die Beziehung zwischen Daten - Informationen - Wissen - Weisheit . . . . .	10
3.1	Aufbau eines Datawarehouses . . . . .	11
3.2	Bestandteile eines Data-Warehouses . . . . .	12
3.3	Welche Bestandteile eines Data-Warehouses werden für was benutzt? . . . . .	13
4.1	Referenzmodell eines Data Warehouses nach Bauer & Günzel . . . . .	14
4.2	Eingliederung des ODS in die Data Warehouse Architektur . . . . .	16
4.3	Bereiche eines Data-Warehouses nach Bauer & Günzel . . . . .	18
4.4	Hub and Spoke / Nabe und Speiche Architektur . . . . .	20
4.5	Starschema mit einer Fakttabelle in der Mitte und fünf Dimensionstabellen . . . . .	20
4.6	Beispiel einer Schneeflocken-Architektur . . . . .	21
4.7	Galaxy-Architektur . . . . .	21
4.8	Leeres Datenbank-Schema mit Dimensions- und Faktentabellen. . . . .	22
4.9	Datenbank-Schema mit Dimensions- und Faktentabellen. . . . .	23
4.10	Datenbank-Schema mit Dimensions- und Faktentabellen, die miteinander in Beziehung stehen. . . . .	23
4.11	Fertiges, ausnormalisiertes Sternschema der Kursverwaltung. . . . .	23
4.12	Dimensionstabellen für die Zeit, den Kurs und das Kurszentrum . . . . .	24
4.13	Faktentabelle der Einschreibungen . . . . .	24
6.1	Klassifikation (links) vs. Regression (rechts) . . . . .	29
6.2	Partitionierendes vs. Hierarchisches Clustering . . . . .	31
6.3	Beispiel von agglomerativem (oben) und diversiven (unten) clustering . . . . .	31
6.4	Daten visuell aufbereitet mittels eines Streudiagrammes . . . . .	32
6.5	Funktionsweise eines Fallbasierten Systemes . . . . .	32
6.6	Beispiel eines Entscheidungsbaumes . . . . .	33
6.7	Beispiel von BLOBs und strukturierten Daten in der selben DB . . . . .	34
6.8	Die drei verschiedenen Methoden zur Integration von unstrukturierten Daten. . . . .	35
6.9	E-Mail-Analyse mit dem Text-Mining Tool UIMA . . . . .	37
7.1	Beispiel von schlechter Datenqualität . . . . .	38
7.2	”Messung“ der Datenqualität . . . . .	39
7.3	Kosten-Qualitäts Gegenüberstellung . . . . .	39
7.4	Vorgehensmodell zur Datenqualitätsverbesserung . . . . .	41
7.5	Prozess der Dublettenbeseitigung als iterativer Prozess . . . . .	45
8.1	Beispiel einer Aktualisierung. . . . .	46
8.2	Beispiel einer Vollhistorisierung. . . . .	46
8.3	Beispiel einer Historisierung mit neuer Spalte. . . . .	47
8.4	Tupelversionierung . . . . .	47
9.1	Ein solches ERD ist nicht mehr übersichtlich und somit nutzlos . . . . .	48
9.2	Vergleich verschiedener Datenbank-Dimensionen . . . . .	48
9.3	Beispiel einer Drill-Across Operation . . . . .	52
9.4	Klassifikationshierarchie . . . . .	53
9.5	Vor- und Nachteile der verschiedenen OLAP-Systemen . . . . .	56
9.6	Vergleich der verschiedenen OLAP-Systeme . . . . .	56
9.7	Abwägung der Performance vs. Funktionalität für die verschiedenen OLAP-Systeme . . . . .	56
9.8	Beispiel der Anwendung von ROLLUP . . . . .	57
9.9	Beispiel der Anwendung von CUBE . . . . .	57
9.10	Beispiel der Anwendung von GROUPING SETS . . . . .	58

10.1	Visualisierung des Codebeispiels von Listing 5 . . . . .	59
11.1	Vergleich der NUMA und UMA-Architektur . . . . .	60
11.2	Speicherverwaltung von SAP-HANA . . . . .	61
11.3	Klassisch- vs. Zeilen- vs. Spaltenbasierte Datenspeicherung . . . . .	61
11.4	Beispiel einer spaltenorientierten Abfrage . . . . .	62
11.5	Beispiel einer zeilenorientierten Abfrage . . . . .	62
11.6	Beispiel der Dictionary Compression . . . . .	64
12.1	Vergleich einer klassischen RDBMS-Architektur im Vergleich zu Hadoop . . . . .	66
12.2	MapReduce verwandelt ungeordnete Daten in Key-Value Paare . . . . .	68
12.3	Ablauf eines Map-Reduce Aufrufes . . . . .	68
12.4	Speichervorgang unter HDFS . . . . .	69
13.1	Vergleich zwischen klassischer (links) und partitionierter (rechts) Datenhaltung .	70
13.2	Beispiel von RAID 6. Hier können 2 Festplatten defekt sein ohne dass ein Datenverlust auftritt . . . . .	70
13.3	Vergleich Dateisystem (links) und Objektspeicherung (rechts) . . . . .	72
15.1	MOFs 4-Ebenen Architektur am Beispiel von Casablanca . . . . .	75
15.2	Abstrahierte Version von 15.1 . . . . .	75
15.3	Unterscheidung zwischen technischen und geschäftlichen Metadaten . . . . .	76
15.4	CWM-Schichtenmodell . . . . .	78
15.5	CWM-Metamodell . . . . .	78
15.6	Vereinfachung des Umherschiebens von Metadaten durch das CWM . . . . .	79
16.1	Data Warehouse Strategie . . . . .	80
16.2	Beispiel eines Reifegradmodells (biMM) . . . . .	80

# 1 Die Notwendigkeit von Data Warehouses

## 1.1 Entscheidungsunterstützung (Skript S15)

Data Warehouses sind keine neuen Erfindungen. Bereits in den 1960er Jahren wurden sogenannte **Managementsinformationssysteme** entwickelt. Diese MIS dienten dazu, Entscheidungsträgern alle benötigten Informationen zeitnah, fehlerfrei, flexibel, ergonomisch, effizient, effektiv und inspirativ zur Verfügung zu stellen. Diese Systeme treffen also nicht selbst Entscheidungen, sie **unterstützen** die Entscheidungsträger lediglich bei ihrer Entscheidung.

Es gibt vier Arten der Entscheidungsunterstützung:

**Modellbasiert:** z.B. Lineare Optimierung - Ein Mathematischer Ansatz basierend auf einem Modell  $\Rightarrow$  Abbildung der Realität

**Wissensbasiert:** z.B. Expertensysteme - Ansätze von Künstlicher Intelligenz

**Datenbasiert:** Basierend auf grossen Datenmengen  $\Rightarrow$  Data-Warehouse, OLAP oder Data-Mining

**KI:** Basierend auf Vorschlägen von Systemen, die Entscheidungen auf Basis von Daten und/oder gelernten Inhalten ( $\rightarrow$  Machine Learning)

**Ein Expertensystem** (XPS oder ES) ist ein Computerprogramm, das Menschen bei der Lösung von komplexen Problemen wie ein menschlicher Experte unterstützen kann, indem es Handlungsempfehlungen aus einer Wissensbasis ableitet.

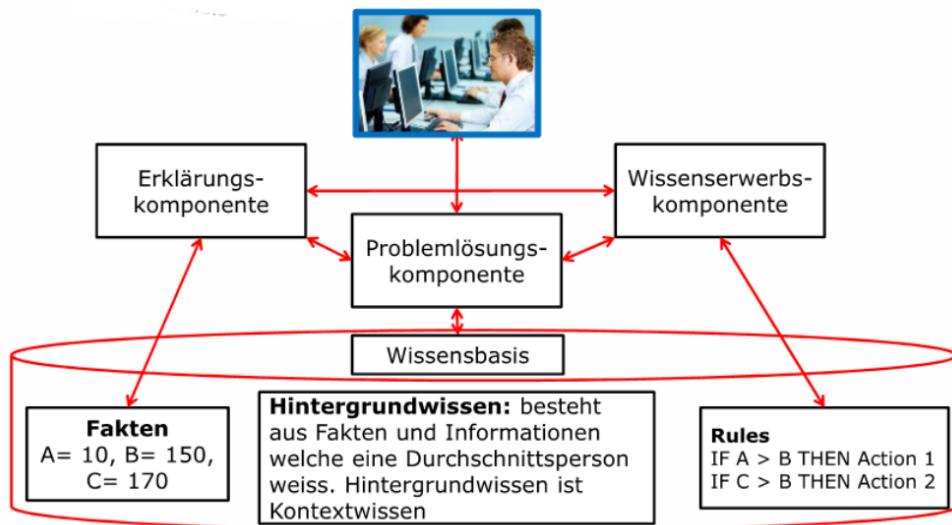


Abb. 1.1: Beispiel eines Expertensystems

## 1.2 Ungenügen der "gängigen" Datenhaltung (Skript S17)

- Verschiedene Datenformate
- Verschiedene Werkzeuge
- Heterogenität der Daten
  - Technisch (Mainframe / DBMS / Flatfile etc)
  - Logisch (Schemata / Formate / Darstellungen etc.)
  - Syntaktisch (Datum / Codierung / Währung)
  - Qualitativ (Fehlende / Falsche / Doppelte Werte)
  - Verfügbarkeit (Permanent / Periodisch / Temporär)
  - Rechtlich (Datenschutz / Zugriffsverwaltung / Archivierung)

→ Neuer Ansatz einer Datenaufbereitung muss her: **Homogenisierung**

## 1.3 Ungenügen der operativen Datenbanken für Entscheide (Skript S18)

"Reguläre" Datenbanken im Geschäftsumfeld sind zu fest mit geschäftsrelevanten Lese- und Schreiboperationen beschäftigt. Bei solchen Datenbanken spricht man von OLTP-System (Online Transactional Processing). Diese Datenbanken sind also ziemlich schlecht geeignet für eine analytische, vorausschauende Bewirtschaftung, da solche Auswertung viel Zeit und vor allem Rechen-Performance benötigen.

Ausserdem liegen Daten in OLTP-Datenbanken meist in der 3. Normalform vor. Während dies eine sehr vernetzte und effiziente Art der Datenspeicherung ist, ist die 3. Normalform ein schlechtes Abbild des intuitiven Denkens eines Managers.

→ Neuer Ansatz einer Datenbank muss her: **analytische Datenbanken**

## 1.4 SQL-Abfragen für Management-Zwecke (Skript S18ff)

Zusätzlich zu den vorhin genannten Gründen, sind Manager des SQL meist nicht mächtig. Sie wollen lieben "Drag and Drop" Interfaces, um sich ihre Daten "zusammenzuklicken" wie z.B. Microsoft Access.

Zudem sind Datenbank-Abfragen stets **zweidimensional** in Tabellen dargestellt. Wenn man nun aber Daten in drei Dimensionen auswerten will (z.B. Zeit, Ort und Anzahl), so ist dies zwar möglich mittels Tabellen, aber nicht sonderlich leserfreundlich.

→ Neuer Ansatz der Datenabfrage muss her: **OLAP**

## 1.5 OLAP vs OLTP (Skript S24)

Merkmale	OLTP System	OLAP System
Ausrichtung auf	Programm, BWL Prozess	Mensch, Analyse
Zeitliche Reichweite	Taktisch	Strategisch
Entscheidungsstufe	Tief	Hoch
Zweck	Rationalisierung & Automatisierung	Planning & Entscheidung
Anwenderzahl	Hoch	Tief
Entscheidung	Deduktiv	Induktiv / Explorativ
Bewirtschaftung I	Andernd	Befragend
Bewirtschaftung II	Auf Datensatzebene	Auf Aggregatsebene
Anwendungsmuster	Voraussehbar	Variierend
Befragungsmuster	Einfach	Komplex
Bearbeitung	Repetitiv	Ad hoc / unstrukturiert
Betriebliches Wissen	Verarbeitend	Generierend
Verteilungsgrad	Dezentral	Zentral
Performance-Bedarf	Durchgehend hoch	Variierend
Mehrbenutzersynchronisation	Hoch	Tief bis keine
Optimierung	Schneller Insert & Delete	Schnelles Lesen
Transaktionsdurchsatz	Hoch	Tief
Transaktionsdauer	Kurze Mutationen weniger Tupel	Lange Abfragen vieler Tupel
Abfragen	Häufige, einfache Abfragen	Weniger häufige, komplexe Anfragen
Antwortzeiten	(Mili)sekunden	Sekunden, Minuten, Stunden
Endbenutzerwerkzeug-Hersteller	DB-Hersteller	Markt
Zeitbezug	Aktuell	Historisch
Zeitdimension	Zeitpunkt	Zeitraum
Beständigkeit	Dynamisch	Statistisch
Granularität	Fein	Grob
Datenbestand	Vollständig	Lückenhaft
Redundanz	Normalisiert	Denormalisiert
Datenqualität / Aussagekraft	Tief	Hoch
Aufbereitung	Anwendungsneutral	Analyseorientiert
Aktualisierung	Laufend	Periodisch
Verarbeitungseinheit	Keon	Gross
Verteilungsgrad	Dezentral	Zentral
Datenquelle	Aktuelle Unternehmensdaten	Interne & externe Daten

## 2 Daten vs. Informationen vs. Wissen vs. Weisheit (Skript S26)

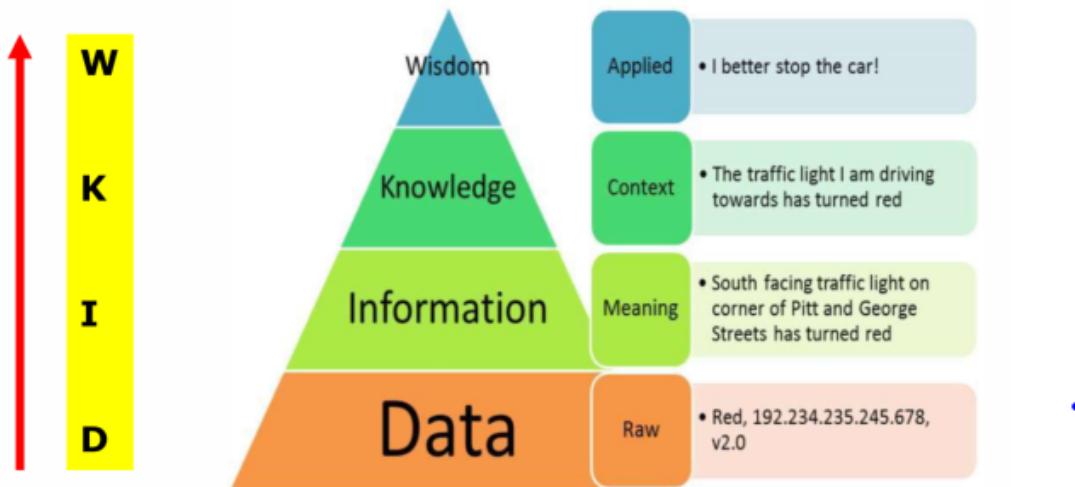


Abb. 2.1: DIKW-Pyramid

Bei Entscheidungsfindungen muss unterschieden werden zwischen

- Daten
- Informationen
- Wissen
- Weisheit

**Daten** Daten sind das, was in Datenbanken oder Excel-Tabellen gespeichert wird. **Unstrukturierte Fakten** wie z.B die Zahlenreihenfolge

**Rot, 192.234.235.245.678.v2.0**

**Informationen** Aus Daten alleine werden wir nicht schlau. Diese Daten müssen zuerst in einen **Zusammenhang** gebracht werden:

**Das südliche Rotlicht Pitt/George St. ist soeben rot geworden**

Nun können wir aus dieser, vorher völlig nutzlosen Zahlenreihe eine **Information** extrahieren. Nämlich dass sie Koordinaten sind und sich das "Rot" auf ein Lichtsignal bezieht..

Die Daten stellen zwar den eigentlichen Wert der Information dar (die Koordinaten und Rotlicht-Licht), sind aber ohne Zusammenhang völlig wertlos.

**Wissen** Aus Informationen **Wissen** zu machen ist nun, zumindest maschinell gesehen wesentlich schwerer. Wir Menschen generieren Wissen, indem wir Informationen geistig verarbeiten. Soll heißen, wir **interpretieren und ordnen** die gegebene Information.

Das heisst in diesem Fall, wir checken wo wir sind und in welche Richtung wir uns bewegen. Je nach dem ist das Wissen dann:

**Ich fahre in eine völlig andere Richtung, diese Information interessiert mich nicht.**

oder aber

**Das Rotlicht, auf welches ich zufahre, ist gerade rot geworden**

**Weisheit** Weisheit wird definiert als **Anwendung von Wissen auf eine Problemlösung**. Das heisst, das erworbene Wissen wird mit einer Portion Erfahrung und gesundem Menschenverstand gemixt. In unserem Fall, angenommen wir fahren auf das Rotlicht zu, sagt uns die Erfahrung, dass man bei einem roten Rotlicht stoppen soll und der gesunde Menschenverstand wirft noch ein, dass es entweder einen Unfall geben wird oder aber sicherlich eine Busse, sollte man erwischt werden. Das Ganze resultiert in:

**Ich sollte vermutlich nächstens einmal anhalten**

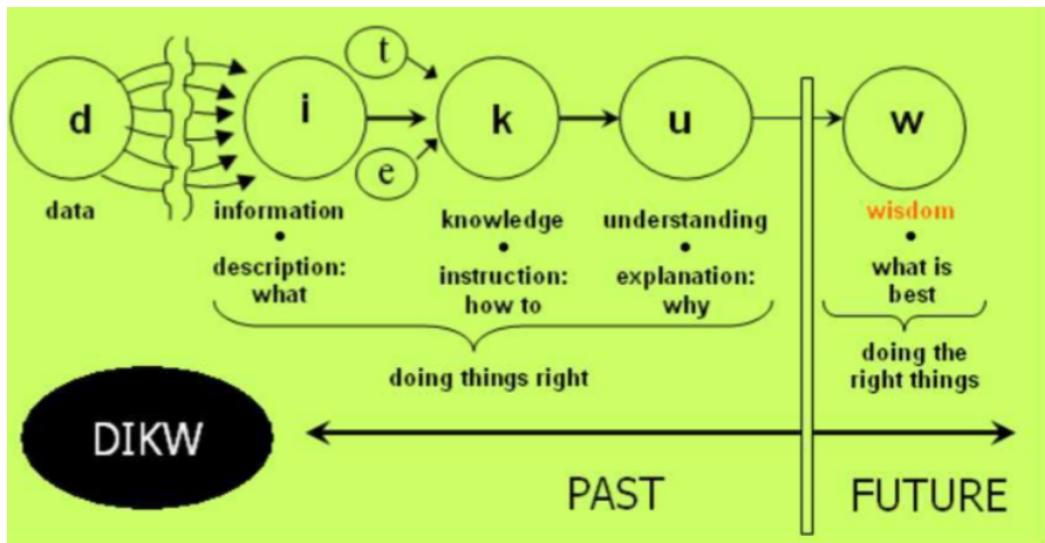


Abb. 2.2: Die Beziehung zwischen Daten - Informationen - Wissen - Weisheit

Zusammengefasst kann man also sagen, Informationen sind das Verständnis von Zusammenhängen in Daten, oder auch **was die Daten bedeuten**. Wissen ist das Verständnis dieser Information (**was heisst das für mich?**) und Weisheit bestimmt, **was nun zu tun sei**.

### 3 Das Data Warehouse

“In einer optimalen Welt würden Daten ”perfekt“ abgelegt werden, leicht zugänglich, platzsparend, sicher und für verschiedene Zwecke nützlich. Da wir aber leider nicht in einer optimalen Welt leben, ist dies nicht der Fall.“ (Buch, S32)

Daten sind in der Praxis meist nicht optimal abgelegt. Daten existieren meist

- in unterschiedlichen Formaten (Excel, Access, DB etc)
- in unterschiedlichen DB-Strukturen
- in unterschiedlichen IT-Architekturen und -Systemen. Meist auch uralt Legacy-Systeme (Wie z.B. Cobol)
- zeit-aktuell und dynamisch
- zu detailliert und feingranular für wirksame Management-Abfragen
- in einem Format, das für Änderungstransaktionen optimiert wurde (z.B. 3. Normalform)
- mit begrenzten Zugriffsrechten (z.B. aus Security-Gründen)
- in einem schlecht verfügbaren Zustand (Legacy-System, proprietäres Format, Security-Gründe)
- in einem Format, welches komplexe SQL-Queries verlangt, um an Informationen oder Wissen zu gelangen.

→ Lösung: **Data-Warehouse**

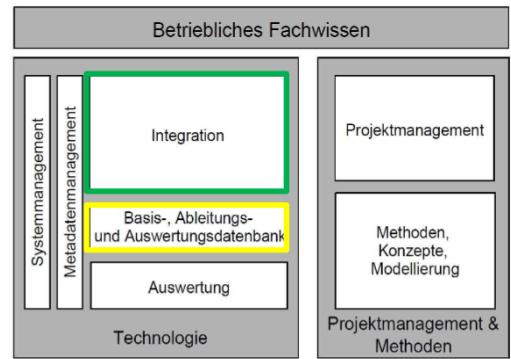


Abb. 3.1: Aufbau eines Datawarehouses

### 3.1 Definition Data-Warehouse (Skript S 37)

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historic data derived from transaction data, but can include data from other sources. Data warehouses separate analysis workload from transaction workload and enable an organisation to consolidate data from several sources. (Oracle corp: Data warehousing Guide 11g (2007))

“A data warehouse is an organization’s data with a corporate wide scope for use in decision support and informational applications.” (IBM Corp: Enterprise Data Warehousing with DB2.9 - Redbook (2008))

Zusammengefasst kann man also sagen, ein Data Warehouse ist eine Datenbank, welche nicht (ausschließlich) zur Speicherung von Informationen genutzt wird, sondern hauptsächlich als Hilfsmittel bei Entscheidungen eingesetzt wird (→ Experten-Systeme)

### 3.2 Bestandteile eines Data-Warehouses (Skript S43)

**SSRS:** SQL Server Reporting Services

**SSAS:** SQL Server Analysis Services

**SSIS:** SQL Server Integration Services

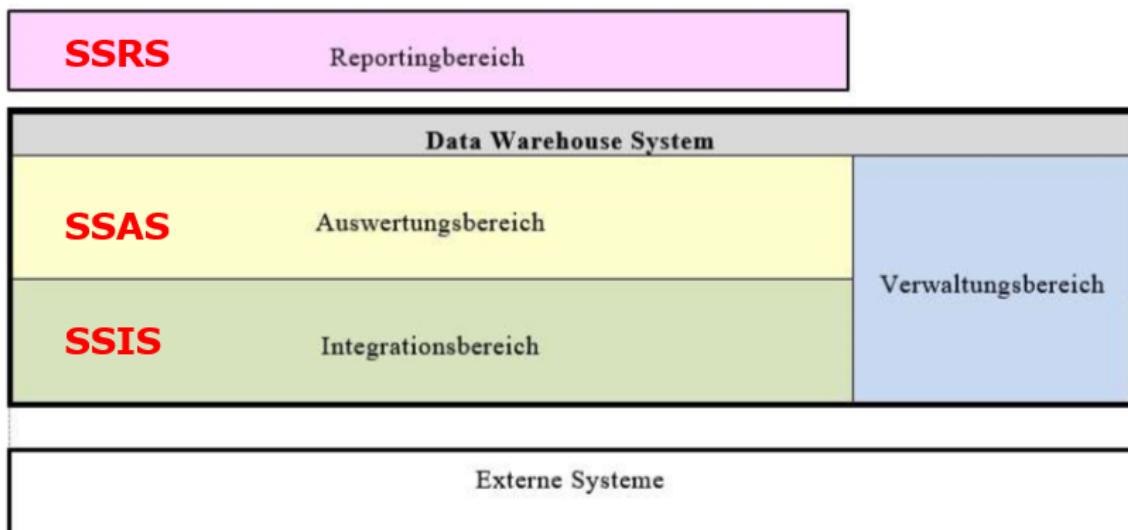


Abb. 3.2: Bestandteile eines Data-Warehouses

### 3.3 Welche Datenbank für welche Tasks? (Buch S40ff)

Name (Praxis)	Name Buch	Wofür	
Online DB's ERP, CRM, ...	<b>Datenquelle</b> (oft Extern)	Daily Business rasche Reaktion rasches lesen/schreiben	
Staging Area	<b>Arbeitsbereich</b>	periodische Transformation Homogenisierung	temporäre Resultate S. 55 (Buch)
Operational Data Store (ODS)	<b>Basis-DB</b> (z.T. Data Warehouse)	"Datendrehscheibe"	aus finanziellen Gründen verzichtet man oft darauf S. 58 (Buch)
(Central) Data Warehouse	<b>Ableitungsdaten bank</b>	möglichst homogenisierter Datenbestand	Strukturierung der Daten orientiert sich an Auswertungs- bedürfnissen S. 64 (Buch)
DataMart	<b>Auswertungs- Datenbank</b>	Teilkopien von DW Optimierte Unterstützung von z.B. Abteilungen.	Grundlage für Mehrdimens. Auswertungs-Tools S. 67 (Buch)
Cube	<b>Würfel</b>	für analytische mehrdimensionale rasche Abfragen	aus DataMart oder aus DataWarehouse erstellt

Abb. 3.3: Welche Bestandteile eines Data-Warehouses werden für was benutzt?

Verschiedene Datenbanken können (und sollten) für verschiedene Tasks verwendet werden.

## 4 Referenzarchitekturen

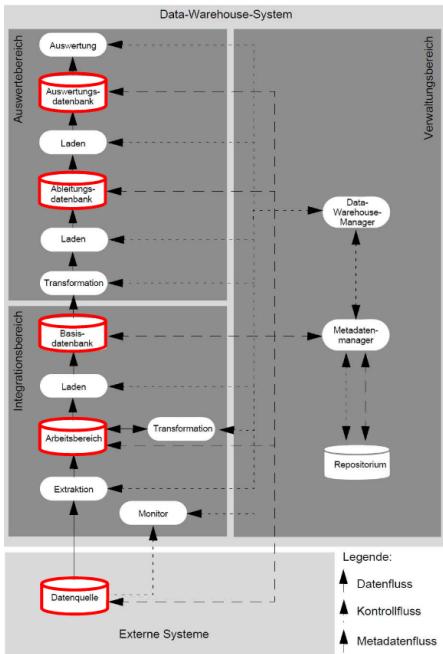


Abb. 4.1: Referenzmodell eines Data Warehouses nach Bauer & Günzel

Eine **Referenzarchitektur** ist ein Referenzmodell für eine Überklasse von Architekturen, in unserem Falle von Data Warehouses.

Ein **Referenzmodell** ist ein allgemeines Modell für eine Klasse von Dingen. Ein Referenzmodell sollte folgende zwei Eigenschaften haben:

- Es können bestimmte Sachverhalte oder Modelle auf dessen Basis erstellt werden.
- Das allgemeine Modell kann als Vergleichsobjekt dienen.

In diesem Modul wird ausschliesslich mit dem hier dargestellten Referenzmodell von Bauer & Günzel gearbeitet.

### 4.1 Bestandteile des Referenzmodells (Skript S45ff)

#### 4.1.1 Datenquelle (Skript S46)

Als **Datenquelle** eines Data-Warehouses werden oft beliebige *Bezugs-Datenbestände* benutzt (auch *effektive Daten* oder *Primärdaten* genannt)

Das können zum Beispiel:

- Daten aus **Legacy-Systemen**
- Daten aus **Anwendungsprogrammen**
- Daten aus **zentralen/dezentralen Arbeitsplatz-DBs**
- **Textdateien** (z.B. im ASCII oder UTF-8 Format)
- Tabellen aus z.B. Excel

sein.

Es gibt weiterhin einige Probleme mit Datenquellen aus verschiedenen Systemen, wie zum Beispiel:

- Unterschiedliche Zeichencodierung (ASCII, ANSI, UTF-8)
- Unterschiedliche Trennzeichen zwischen Datenfeldern (Komma, Semikolon)
- Unterschiedliche Zeilenwechsel (CR/LF, LF)
- Unterschiedlichen Sortierungen (alphanumerisch, numerisch)

Deshalb wird immer häufiger auf einheitliche XML-Inputs zurückgegriffen.

#### 4.1.2 Arbeitsbereich / Staging Area (Skript S46 / Buch S55)

Der Arbeitsbereich integriert die Daten aus den verschiedenen vorhin genannten Datenquellen. Diese Integration passiert nach der Extraktion aus diesen Datenquellen.

Die Daten werden mit Hilfe des Metadaten-Repository anhand ihrer Metadaten zusammengefügt und abgelegt.

Das **Metadaten-Repository** besteht normalerweise aus verschiedenen Datenbank-Tabellen zur Verwaltung von Metadaten. Diese Metadaten stammen aus sehr unterschiedlichen Systemen und enthalten alle notwendigen Beschreibungen zu ihrem System und der Umwelt. Somit können die heterogenen Daten fast ohne Programmieraufwand zu einer homogenen Masse zusammengefügt werden.

Die Staging Area ist ein **flüchtiges Zwischendepot** für die Daten. Hier werden die notwendigen Transformationen durchgeführt werden, welche in weiteren Schritten notwendig sind.

#### 4.1.3 Basis-DB / Operational Data Store (ODS) (Skript S47 / Buch S58)

Ein ODS ist eine Datenbank, welche aktuelle/operative Daten hält, meist in kleinen Teilmengen unterteilt. Diese Datenbank ist eine **temporäre** Zwischenstation zwischen der Staging Area und dem Data Warehouse.

Das Datenmodell des OBS entspricht meist demjenigen der Datenquelle. Dies ist vor allem dann der Fall, wenn der OBS aus Leistungsgründen vom Quellsystem getrennt wurde und benutzt wird um gelegentliche Einzelfall-Analysen durchzuführen.

Das Datenmodell des OBS kann andererseits auch demjenigen des Data Warehouses entsprechen. Dies ist dann der FALL, wenn der ODS als temporärer Zwischenspeicher zwischen den Quelldaten und dem Data Warehouse genutzt wird, oder aber wenn der ODS vom Data Warehouse Daten erhält.

Eine Basis-DB/ein ODS muss nach Gauer & Günzel folgende Eigenschaften haben:

- Die Daten sind **integriert** von den jeweiligen Datenquellen. Das heisst die verschiedenen Datenformate und Schematas wurden vereinheitlicht.
- Sie enthält nebst aktuellen Daten auch **historische Daten**, jedoch in geringerer Feingranularität wie das Data Warehouse.
- Sie ist **Anwendungsneutral**, d.h. sie ist nicht für eine spezielle Anwendung optimiert bzw. fokussiert.
- Nach einer definierten Zeitspanne werden die Daten in die **Ableitungsdatenbank** übertragen, wo sie je nach Auswertungsbedarf in einem anderen Detaillierungsgrad abgelegt werden.

- Die **Aktualisierung** der Daten erfolgt zu **beliebigen Zeitpunkten**. Dieser Zeitpunkt wird durch den Aktualisierungsbedarf gesteuert.
- Die Daten wurden bereits in der Staging Area bereinigt.

Der ODS wird entweder laufend oder periodisch mit Daten aus der Staging Area gefüttert. Vor allem in kleineren Data Warehouse Systemen fällt der ODS sogar manchmal ganz weg und die Staging Area übernimmt auch diese Arbeiten. Im professionellen Umfeld sind jedoch sowohl der ODS wie auch die Staging-Area ein integraler Bestandteil der Data Warehouse Architektur.

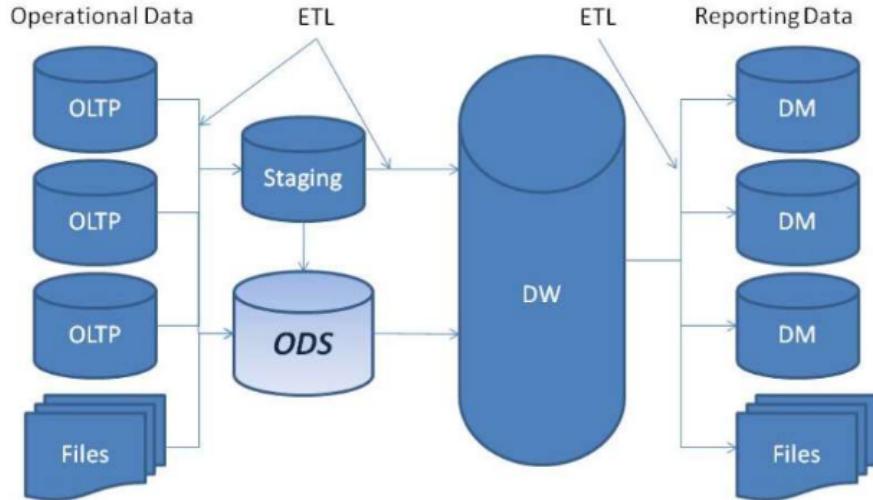


Abb. 4.2: Eingliederung des ODS in die Data Warehouse Architektur

#### 4.1.4 Ableitungsdatenbank (=Enterprise-)Data Warehouse (Skript S47/Buch S64)

Wenn von einem Data Warehouse gesprochen wird, ist in der Regel ein *Enterprise-Data-Warehouse* gemeint (= ein Data-Warehouse mit unternehmensweitem Datenmodell und unternehmensweiten, universellen Datenbestand).

Die Datenbestände eines Data Warehouses werden zwar periodisch ergänzt, jedoch werden praktisch nie Daten gelöscht oder verändert, wenn sie mal im Data Warehouse sind. Aufgrund dessen beinhalten solche Systeme eine enorme Menge von Daten (auch VLDBs für Very Large DataBases).

Es ist jedoch selten nötig auf den gesamten Datenbestand dieser riesigen Datenbanken zuzugreifen. Aufgrund dessen werden Daten, die sehr selten benutzt werden in grossen Unternehmen meist in Teilkopien des Data Warehouses, sogenannten *Data Marts* gespeichert.

#### 4.1.5 Auswertungsdatenbank (Data-Mart) (Skript S50/Buch S67)

Ein Data-Mart ist eine **Teilkopie** eines Data-Warehouses, die aber auf demselben Datenmodell basieren. Solche Data-Marts werden meist für Abteilungen einer Unternehmung wie z.B. das Marketing erstellt. Der Vorteil dieser Data-Marts ist, dass diese unabhängig sind vom zentralen DW.

Vorteile eines Data-Marts sind z.B.

- Bessere Leistung, da nicht alle Analysen/Auswertungen auf dem zentralen DW gemacht werden müssen

- Entlastung des DW
- Im Falle von lokalen Data-Marts weniger Netzwerkbela

Die Pflege und (Weiter)Entwicklung des Data-Marts liegt jeweils in der Verantwortung der einzelnen Abteilungen.

Dass ein Data-Mart existieren kann, **muss** ein zentrales Data-Warehouse existieren. Deshalb werden solche Data-Marts meist als **abhängige Data-Marts** bezeichnet.

Diese abhängigen Data-Marts stehen im Gegensatz zu den (historischen) **unabhängigen Data-Marts**. Diese Data Mars erhalten ihre Daten direkt von verschiedenen Quellsystemen (gehen also nicht über ein zentrales Data-Warehouse). Jedoch widerspricht das dem Gedanken von einem zentralen "Datenhort" des Data-Warehousing und *sollten* heute nicht mehr verwendet werden (oder man sollte sie zumindest nicht mehr als Data-Marts bezeichnen...)

Ein "Problem" von Data-Marts ist, dass sie unabhängig voneinander sind. Somit werden meist dieselben Aggregationen mehrmals auf verschiedenen Data-Marts durchgeführt, was auf längere Zeit nicht sehr kosteneffizient ist. Stattdessen sollten diese Aggregationen eher auf dem zentralen DW durchgeführt werden (was aber nicht immer geht, da die verschiedenen Abteilungen nicht wissen, dass diese Aggregation bereits in einem anderen Data-Mart durchgeführt wurde)

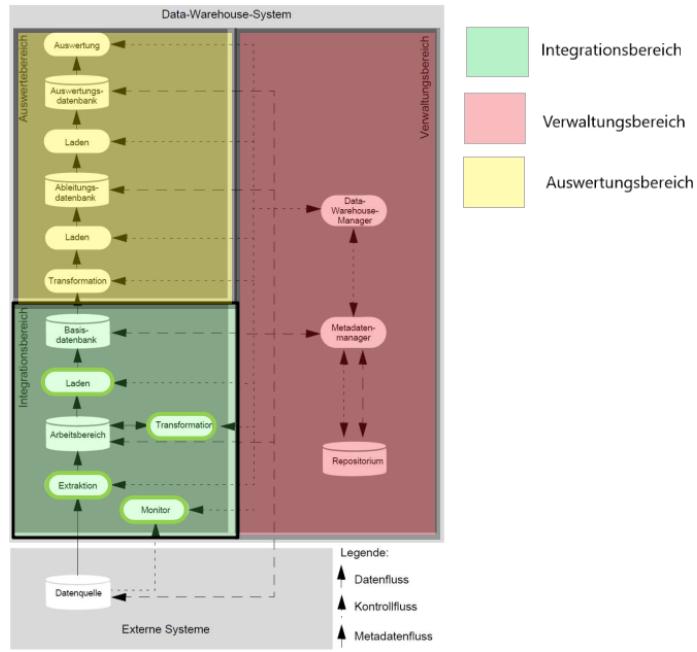


Abb. 4.3: Bereiche eines Data-Warehouses nach Bauer & Günzel

#### 4.1.6 Integrationsbereich (Skript S54)

Der Integrationsbereich bereitet die Daten, die aus verschiedenen Datenquellen kommen via staging area und ODS so auf, dass sie ins zentrale DW überführt werden können.

Im Zusammenhang mit dem Integrationsbereich wird oft von **ETL** gesprochen. ETL steht für *Extract - Transfer - Load* und entspricht den wesentlichen Arbeitsschritten, die durchgeführt werden müssen, bis die Daten vom Quellsystem im DW landen.

#### 4.1.7 Auswertungsbereich (Buch S72)

Der Auswertungsbereich des Data Warehouses umfasst die **Ableitungsdatenbank/Data Warehouse** und die **Auswertungsdatenbank**. Genaueres über diese beiden Bestandteile des Auswertungsbereiches können in Kapitel 4.1.4 - Ableitungsdatenbank und 4.1.5 - Auswertungsdatenbank nachgelesen werden.

Wie der Name schon vermuten lässt, werden die Daten, die von den Datenquellen über ETL-Prozesse in die Basisdatenbank geladen wurden, hier ausgewertet.

Auswerten kann vieles sein. So können einfache Aggregationsfunktionen zur Auswertung benutzt werden, aber auch komplexe statistische Methoden, die z.B. beim Data Mining zum Einsatz kommen.

Zusammengefasst kann man sagen: Eine **Auswertung** ist die *Anwendung von Auswertefunktionen auf ausgewählte Daten zur Generierung von neuer Information*.

Zudem werden die ausgewerteten Daten so aufbereitet und bereitgestellt, dass sie auch in anderen Systemen weiterverarbeitet werden können und/oder an andere Personen/Unternehmen weitergegeben werden.

Last but not least werden die Ergebnisse der Auswertung wieder in die Basisdatenbank zurückgegspeichert, so dass sie die Qualität der Datenbasis erhöht und zukünftige Auswertungen somit verbessert.

#### 4.1.8 Verwaltungsbereich (Skript S272)

Der Verwaltungsbereich eines Data Warehouses enthält den **Data Warehouse Manager**, den **Metadaten Manager** und das **Repositorium** (Abb. 4.3)

##### **Data Warehouse Manager (Buch S43)**

Der Data-Warehouse Manager (eine Person) initiiert, steuert und überwacht die einzelnen Data Warehouse Prozesse. Wie in Abb. 4.3 zu sehen ist, übernimmt er unter anderem die zentrale Steuerung von:

- Monitor
- Extraktion
- Transformation
- Laden
- Auswertung

Zudem kann er, wenn nötig auch den Datenbeschaffungsprozess triggern. Das kann manuell geschehen oder anhand des Monitors (Kap. 5.4)

##### **Metadaten Manager (Buch S. 82)**

Ähnlich wie beim Data Warehouse Manager kann beim Metadaten Manager recht einfach auf dessen Tätigkeit geschlossen werden: Es ist eine **Software**, die die Metadaten des Data Warehouses überwacht, steuert und initialisiert.

Genauer gesagt heisst das, der Metadaten Manager verwaltet das **Metadaten Reppositorium**, in welchem alle Metadaten (also Operative, Struktur-, Prozess- und Begriffsmetadaten) gespeichert werden.

Es können ausserdem Metadaten aus Entwurfs- und Modellierungswerkzeugen (wie z.B. SSMS oder MySQL Workbench) extrahiert werden

Der Metadaten Manager übergibt ausserdem die Metadaten, die er aus dem Reppositorium zieht dem Data Warehouse Manager, der diese wiederum an die metadatengesteuerten Werkzeuge, die diese zur Laufzeit lesen, interpretieren und ausführen. Diese Werkzeuge erzeugen selbst wieder Metadaten (z.B. Log- oder Reportdateien), die schlussendlich wieder im Reppositorium landen und später wieder diesen Tools gefüttert werden.

Das Ziel ist es, eine vollständig automatisierte Aktualisierung der Metadaten zu erreichen.

##### **Repositorium (Buch S79)**

Das Reppositorium ist der zentrale Storage für die Metadaten im Data Warehouse.

## 4.2 Referenzmodelle in die Praxis umgesetzt (Skript S63)

### 4.2.1 "Hub and Spoke"-Architektur

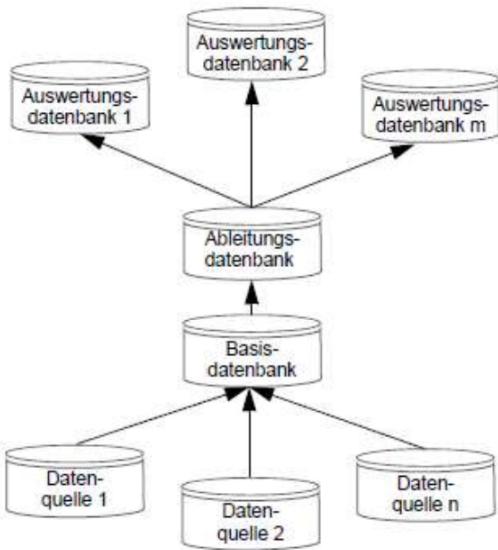


Abb. 4.4: Hub and Spoke / Nabe und Speiche Architektur

Bei der Hub and Spoke (auch Nabe und Speiche Architektur) werden aus mehreren Quellsystemen (Speichen/Spokes) Daten in eine Basisdatenbank geladen, dort ETL-ed und anschliessend in eine Ableitungsdatenbank überführt (Hub/Nabe). Von dort werden sie anschliessend wieder in verschiedene Auswertung-Datenbanken verteilt (Speiche/Spokes).

Das Core-Data-Warehouse ist also eine der zentralen Naben oder Hubs und ist verantwortlich für die Integration, QA und die Verteilung von Daten an die verschiedenen Data-Marts (Auswertungs-Datenbanken).

### 4.2.2 Star-Architektur

Die Star- oder Stern-Architektur ist eine der drei am weitesten verbreiteten Datenmodelle. Sie besteht aus einer sog. **Faktentabelle** in der Mitte, die von mehreren **Dimensionstabellen** sternförmig umgeben ist.

Durch eine solche Anordnung sind die Datenbanken normalerweise *denormalisiert*, können also redundante Daten enthalten. Jedoch wird dieser eventuelle höhere Speicherbedarf in Kauf genommen, da ein solches Sternschema eine wesentliche höhere Performance bietet im ETL-Bereich wie eine normalisierte Datenbank.

Eine weitere Verbesserung der Performance könnte man mit dem, aus der Stern-Architektur abgeleiteten **Schneeflockenarchitektur** erzielen.

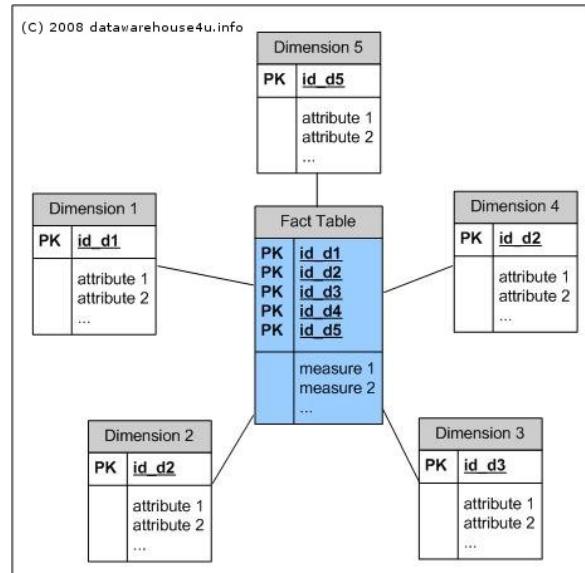


Abb. 4.5: Starschema mit einer Fakttabelle in der Mitte und fünf Dimensionstabellen

### 4.2.3 Snowflake-Architektur

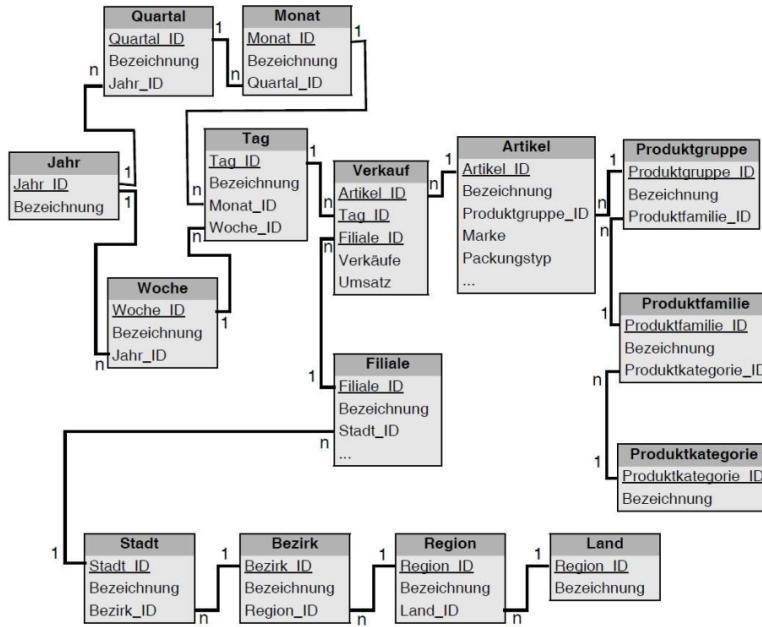


Abb. 4.6: Beispiel einer Schneeflocken-Architektur

Die Snowflake- oder Schneeflocken-Architektur ist eng mit der Stern-Architektur verbunden. Der einzige Unterschied zwischen der Stern- und der Schneeflocken-Architektur ist, dass die Dimensionstabellen gleichzeitig auch Faktentabellen sind, die wiederum von anderen Tabellen umgeben sind. Diese Dimensionstabellen werden über Joins verknüpft.

### 4.2.4 Galaxy-Architektur

Die Galaxy- oder Galaxie-Architektur ist ebenfalls stark mit dem Stern-Schema verwandt. Allerdings können sich hier mehrere Faktentabelle eine Dimensionstabelle teilen.

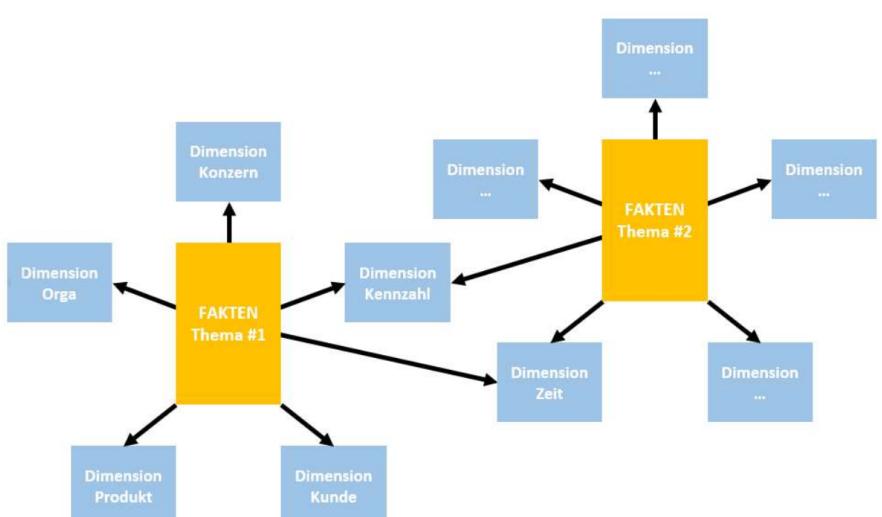


Abb. 4.7: Galaxy-Architektur

Bei den drei oben genannten Architekturen werden Daten als **Fakten** bezeichnet und somit in der **Faktentabelle** gespeichert. Fakten werden, je nach Literatur auch **Metriken** oder **Kennzahlen** genannt. Solche Faktentabellen können über die Zeit sehr gross werden und enthalten Kennzahlen, die sich aus dem laufenden Geschäft ableiten lassen, wie z.B. Profitabilität, Kosten oder Ausgaben.

Wie in Abbildung 4.5 zu sehen ist, enthalten Faktentabellen Fremdschlüsse zu den Einträgen in den Dimensionstabellen. Das impliziert, dass es jeden Eintrag zu einer Kombination von Dimensionen nur einmal geben kann.

Im Gegensatz zu den riesigen und volatilen Faktentabellen sind Dimensionstabellen recht statisch und vergleichsweise klein.

#### 4.2.5 Beispiel zur Erstellung eines Datenbankschemas

Es soll das DB-Design für eine Kursverwaltung erstellt werden. Die DB soll in einem Sternschema aufgebaut sein (Sternschema: Siehe Kap. 4.2.2)

#### Ablauf

1. Identifikation der Fakten
2. Identifikation der Dimensionen
3. Festlegen der Dimensionshierarchie
4. Herstellung der Beziehungen

#### Identifizierung der Fakten

Welches sind die numerischen Werte, die die Entwicklung eines Geschäfts am Besten ausdrücken?

In unserem Fall wäre das sicher einmal die *Anzahl der Kursbesucher*.

#### Identifikation der Dimensionen

Was sind Dinge, die das wann, wo, was wer, etc. der Fakten beschreiben?

In unserem Fall wären das z.B. der Standort (wo?), die Zeit(wann?) und der Kurs (was?)

Momentan sieht unser Schema also so aus:



Abb. 4.8: Leeres Datenbank-Schema mit Dimensions- und Faktentabellen.

## Festlegen der Dimensionshierarchie

Nun wird geschaut, welche Elemente gehören zu den Dimension bzw. Fakten?

Das leere Datenbankschema wird nun aufgefüllt:

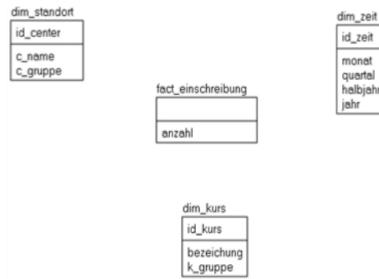


Abb. 4.9: Datenbank-Schema mit Dimensions- und Faktentabellen.

## Herstellen der Beziehungen

Ein DB-Schema, in welchem die Tabellen keine Beziehungen zueinander haben ist recht nutzlos. Nun muss geschaut werden, wie diese Tabellen oder Entitäten miteinander in Beziehung stehen. So hat z.B. ein Standort n Kurse. Diese Kurse werden zu n Zeiten durchgeführt.

Grafisch dargestellt sieht unser DB-Schema mit Beziehungen nun so aus:

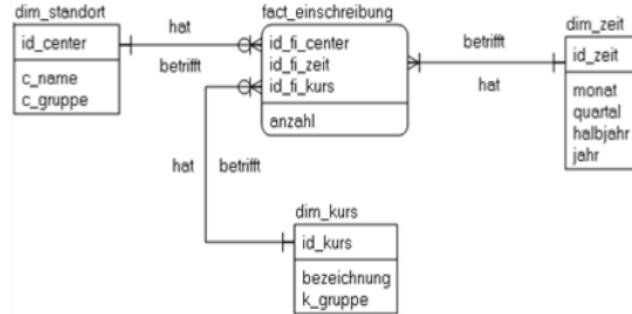


Abb. 4.10: Datenbank-Schema mit Dimensions- und Faktentabellen, die miteinander in Beziehung stehen.

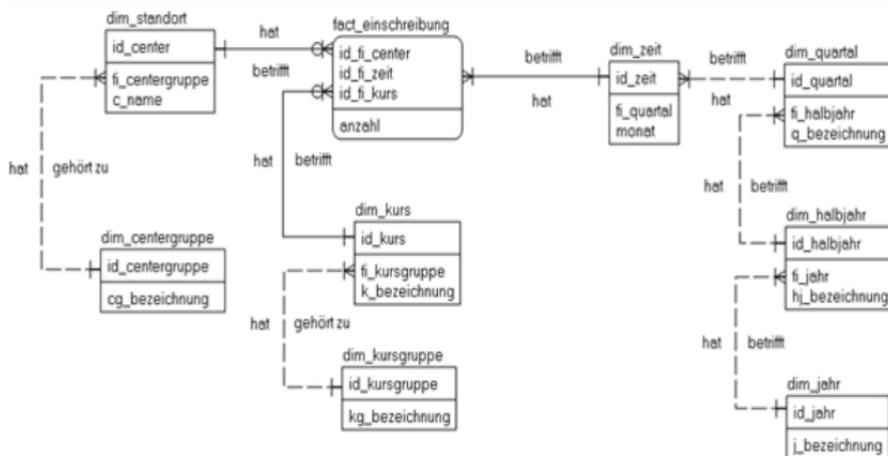


Abb. 4.11: Fertiges, ausnormalisiertes Sternschema der Kursverwaltung.

Werfen wir nun einen genaueren Blick auf die Tabellen, die hier dargestellt sind.

dim_zeit				
id_zeit	monat	quartal	halbjahr	jahr
1101	Januar	q1	h1	2011
1102	Februar	q1	h1	2011
1103	März	q1	h1	2011
1104	April	q2	h1	2011
1105	Mai	q2	h1	2011
1106	Juni	q2	h1	2011
1107	Juli	q3	h2	2011
1002	...	...	...	...

dim_kurs		
id_kurs	Bezeichnung	k_gruppe
k1	ReBalancing	Sport und Freizeit
k2	Sonsteinding	Sport und Freizeit
k3	Linux Einführung	Informatik
k4	Body Toning	Sport und Freizeit
k5	Datenmodellierung	Informatik

dim_center		
id_center	c_name	c_gruppe
c1	Rappi	D
c2	Hottu	D
c3	Bellenz	I
c4	Genf	F
c5	Basu	D

Abb. 4.12: Dimensionstabellen für die Zeit, den Kurs und das Kurszentrum

In Abb. 4.13 sieht man zudem noch die korrespondierende Faktentabelle. Es fällt auf, dass Dimensionstabellen weniger Einträge haben wie Faktentabellen und auch eher statisch sind. Nachdem man die gewünschten Attribute der Dimension erfasst hat, wird die Dimensionstabelle normalerweise nicht mehr gross geändert. Die Faktentabelle hingegen wächst stetig und kann in grösseren Data Warehouses problemlos mehrer 10 Millionen Einträge umfassen.

Ebenfalls ist auffällig, dass die Dimensionstabelle erstaunlich viele Redundanzen hat. Je doch ist dies gewollt. Man nimmt Redundanzen in Kauf, um dafür eine bessere Performance zu erhalten (→ Denormalisierung).

Faktentabellen, im Gegensatz dazu, besitzen keine Redundanzen. Diese doppelten Einträge, die zu sehen sind werden *notwendige Redundanzen* genannt. Denn wenn man sie löschen würde, entnahme das dem System Informationen.

Eine weiter auffällige Tatsache ist, dass die Primärschlüssel der Dimensionstabelle offenbar eine Semantik besitzen (1101 für den 01. Monat des Jahres 2011). Es ist zu empfehlen, jegliche Semantik wegzulassen bei Primärschlüsseln. Deshalb wird im produktiven Umfeld meist auf das IDENTITY oder AUTO\_INCREMENT zurückgegriffen.

fact_einschreibung			
id_fi_center	id_fi_zeit	id_fi_kurs	anzahl
c1	1101	k1	204
c1	1101	k2	68
c1	1101	k3	121
c1	1101	k4	286
c1	1101	k5	49
c2	1101	k1	87
c2	1101	k2	29
c2	1101	k3	52
c2	1101	k4	123
c2	1101	k5	21
c3	1101	k1	228
...	...	...	...

Abb. 4.13: Faktentabelle der Einschreibungen

## 5 ETL

### 5.1 Extraktion (Skript S 55/Buch S56)

Unter Extraktion versteht man die **Übertragung der Daten vom Quellsystem in die Staging Area**

Je nach dem, welche Extraktions-Strategie gewählt wurde, werden die Daten zu unterschiedlichen Zeitpunkten extrahiert:

**Sofort** Der Extraktionsprozess wird bei jeder Änderung sofort gestartet

**Trigger-/Ereignisbasiert** z.B. Beim Erreichen einer festgelegten Anzahl von Änderungen

**Periodisch** Die Daten haben eine geforderte Mindestaktualität. Nach Ablauf dieser festgelegten Periode werden die Änderungen in eine sog. Snapshot-Datei überführt, die anschliessend in die Datenbank importiert wird.

**Anfrage** Eine Anfrage auf die Datenbank erfordert aktualisierte Daten.

Es gibt zwei grundsätzlich zwei **Extraktionstechniken**

**Replikationsbasiert** Geänderte Tupel werden in spezielle Tabellen geschrieben

**Logbasiert** Jede Änderung wird in einer Logdatei protokolliert. Durch die Auswertung dieser Logdatei wird entschieden, welche Daten geändert und somit extrahiert werden müssen

Die Extraktion geschieht meist über SQL und es werden meist Schnittstellen zu den Datenbanken verwendet wie z.B. ODBC, OLE DB oder JDBC.

**Open Database Connectivity (ODBC)** ist eine standardisierte Datenbankschnittstelle, mit welcher über SQL auf die Datenbank zugegriffen werden kann. Der Vorteil von ODBC ist, dass es eine API gibt, die es einem erlaubt, eine DB-Anwendung zu schreiben, die unabhängig vom Datenbankmanagement-System (DBMS) (mySQL, postgreSQL, Oracle etc) funktioniert, solange ODBC einen Treiber dafür hat.

**Java Database Connectivity (JDBC)** ist eine universelle Java-basierte Datenbankschnittstelle für Datenbanken diverser Hersteller. JDBC ist vor allem auf relationale Datenbanken ausgelegt.

### 5.2 Transformation (Skript S56/Buch S57)

Nach der Extraktion müssen diese rohen Daten transformiert werden. Das heisst, sie müssen so aufbereitet werden, dass sie auch tatsächlich Sinn ergeben. Die Transformation lässt sich auf folgende Arbeitsschritte herunterbrechen:

**Data migration / Data wrangling** Daten werden **vereinheitlicht** also dass sie z.B. dieselben Masseinheiten oder Datenformate verwenden.

**Data clean(s)ing** Daten werden **bereinigt**, also doppelte Daten gelöscht, falsche Daten korrigiert oder veraltete Daten aktualisiert.

Dabei nimmt das Data Wrangling mit Abstand am meisten Zeit in Anspruch (bis zu 80%)

### 5.3 Load (Skript S58/Buch S58)

Die extrahierten und bereinigten Daten müssen nun noch in die **Basis- oder Ableitungsdatenbank übertragen** werden. Dafür werden meist Ladewerkzeuge des Datenbankherstellers verwendet (SSIS bei SQL oder SQL\*Loader bei Oracle)

Weitere Möglichkeiten sind z.B. die SSIS Integration von Visual Studio, SSIS direkt oder DataWriter von IBM

#### SQL-Codebeispiele für den Load-Prozess

##### SQL-INSERT

---

```
INSERT INTO table_name
  VALUES (
    value1,
    value2,
    value3,
    ...
  )
;
```

---

##### SQL BULK INSERT

---

```
//FileType=1 (TxtFile1.txt)

"Kelly","Reynold","kelly@reynold.com"
"John","Smith","bill@smith.com"
"Sara","Parker", "sara@parker.com"

//FileType=2 (TxtFile2.txt)

Kelly,Reynold,kelly@reynold.com
John,Smith,bill@smith.com
Sara,Parker,sara@parker.com

BULK INSERT TmpStList FROM 'c:\TxtFile1.txt' WITH (FIELDTERMINATOR = ',',',')
BULK INSERT tmpStList FROM 'c:\TxtFile2.txt' WITH (FIELDTERMINATOR = ',',',')
```

---

Aufgrund dessen, dass während des Ladens die betroffenen Datenbanken gar nicht oder nur eingeschränkt nutzbar sind, ist die Effizienz des Ladevorgang essentiell. Aus diesem Grund werden Ladevorgänge meist in der Nacht, am frühen Morgen oder übers Wochenende durchgeführt.

Es wird dabei zwischen **Online- und Offline-Ladevorgängen** unterschieden. Beim Online-Laden kann die Datenbank weiterhin verwendet werden. Beim Offline-Ladevorgang ist die Datenbank während des Vorgangs offline, kann also nicht verwendet werden. Ob eine Datenbank einen Online-Ladevorgang unterstützt, hängt davon ab, ob die verwendete Anwendung oder Datenbank dies zur Verfügung stellt. So könnten z.B. während des Ladevorgang die Datenänderungen in ein Logfile geschrieben werden und dieses Logfile wird nach erfolgreichem Ladevorgang in die aktualisierte Datenbank übertragen.

Ebenfalls unterscheidet man zwischen dem **Initialisierungsladen** (alle Daten, DB wird zum ersten Mal gefüllt) und dem **Aktualisierungsladen** (nur geänderte Daten werden in die DB geladen)

Daten werden entweder in eine Basis- Ableitungs- oder Auswertungsdatenbank geladen. Aufgrund der grossen Datenmengen können nicht die normalen Datenmanipulationswerkzeuge eingesetzt werden. Stattdessen werden meist sog. **Bulk-Loader**, die auf genau diese Tasks

spezialisiert sind, verwendet. Diese Bulk-Loader verwenden aus Effizienzgründen keine Schnittstelle zur DB, sondern greifen direkt darauf zu. Deshalb können bestimmte Bulk-Loader auch nur für bestimmte Datenbanken verwendet werden.

#### 5.4 Monitor (Skript S60/Buch S54)

Der Monitor spielt je nach Extraktionsstrategie eine wichtige Rolle. Er überwacht die Datenquellen und bemerkt somit Datenänderungen sofort. Je nach Extraktionsstrategie wird somit ein ETL-Prozess getriggert, ein Logeintrag oder eine Snapshot-Datei gemacht o.ä.

Monitore haben vor allem in Data Warehouses eine sehr grosse Bedeutung. Dank ihnen müssen weniger DB-Zugriffe gemacht werden und gleichzeitig werden Redundanzen vermieden.

Es gibt einige Sachen, die zu bedenken sind beim Einführen einer Monitor-Lösung:

- Wird die gesamte Änderung festgehalten oder nur das neue/geänderte Tupel? (→ Je nach dem Platzproblem)
- Benachricht das Quellsystem den Monitor sobald es eine Änderung gegeben hat (→ Trigerring) oder fragt der Monitor regelmässig beim Quellsystem nach (→ Polling)? (→ je nach dem, Überbelastung von Quell- oder Zielsystem)
- Überwacht das Quellsystem selbst, welche Änderungen erfolgen, oder wird dies von einem externen Dienst übernommen? (→ Überbelastung Quellsystem vs. Kosten eines externen Dienstes)
- Sollen Änderungen sofort zum Data Warehouse gepusht werden oder nur zu bestimmten Zeiten (z.B. über Nacht)? (→ Belastung des Data Warehouses und Quellsystems vs. Synchrone Daten.)

## 6 Unstrukturierte Daten (Skript S72)

Ein Grossteil der Daten, der uns heute vorliegt, ist **unstrukturiert**. Das heisst, es ist mit viel Aufwand verbunden, aus grossen Datenmengen Informationen zu extrahieren (man denke z.B. daran, wie viel Aufwand es benötigen würde, um 10'000 E-Mails nach Thema zu sortieren vs. wie viel Aufwand es benötigt, 10'000 Datensätze nach ID zu sortieren).

Mithilfe von **strukturierten Daten** kann man *automatisiert* Informationen aus Datenquellen extrahieren und aufbereiten.

### 6.1 Nutzen

Wenn man heute unstrukturierte Daten auswerten würde, könnte man z.B. automatisch Kundenbewertungen und -beschwerden aus Foren und Blogs auslesen, Twitter-Posts analysieren u.v.m.

Zusammengefasst kann man sagen: Unstrukturierte Daten ermöglichen **Analysen, die zu schnellen Reaktionen und Entscheidungen führen**.<sup>5</sup>

### 6.2 Herausforderungen

- Kein vorgegebenes/einheitliches Schema
- Kein vorgegebenes/einheitliches Datenformat
- Semantikabhängig (z.B. JSON vs. XML)
- Grosse Mengen → können nicht manuell ausgewertet werden.
- Priorität kann nicht automatisiert gemessen werden.
- Datenqualität kann nicht automatisiert gemessen werden

### 6.3 Analyse (Skript S74ff/Buch S131ff)

#### 6.3.1 NLP - Natural Language Processing

NLP beschreibt die Techniken und Methoden zur maschinellen Verarbeitung natürlicher Sprache. Ziel ist eine direkte Kommunikation zwische nMensch und computer auf Basis der natürlichen (menschlichen) Sprache.

NLP ist eine sehr AI-lastige Technologie und mittels welcher Text in strukturierte Informationen gebündelt werden kann. NLP setzt sich mittels der Verarbeitung der natürlichen Sprache, dem Verstehen und der Semantik von Wörtern und Sätzen der Klassifizierung von Texten auseinander.

#### 6.3.2 Data Mining

Data Mining ist die systematische Anwendung statistischer Methoden auf grosse Datenbestände mit dem Ziel, neue Querverbindungen und Trends zu erkennen.

#### 6.3.3 Text Mining

Text Mining ist eine Untermethode von Data Mining und konzentriert sich vor allem auf Data Mining von grossen Texten wie z.B. die Plagiaterkennung, die gesamte Doktorarbeiten durchliest und analysiert.

### 6.3.4 Klassifikation

Klassifikation ist eine Unterart des Data-Minings und versucht, den Datenbestand in vorgegebene Klassen zuzuordnen.

### 6.3.5 Regression

Die Regression ist eine Unterart des Data-Minings und zielt darauf ab, einen Ursache-Wirkung-Zusammenhang zwischen einzelnen Merkmalen der zugrunde liegenden Datenbasis zu ermitteln.

Am Beispiel von Abbildung 6.1 kann Klassifikation vs. Regression so unterschieden werden: Mittels **Klassifikation** kann ermittelt werden, dass ca. die Hälfte der untersuchten Gene krank sind, während mit der **Regression** gezeigt werden kann, dass, je höher die Konzentration von Gen 1 ist, desto länger leben die Patienten (Trend)

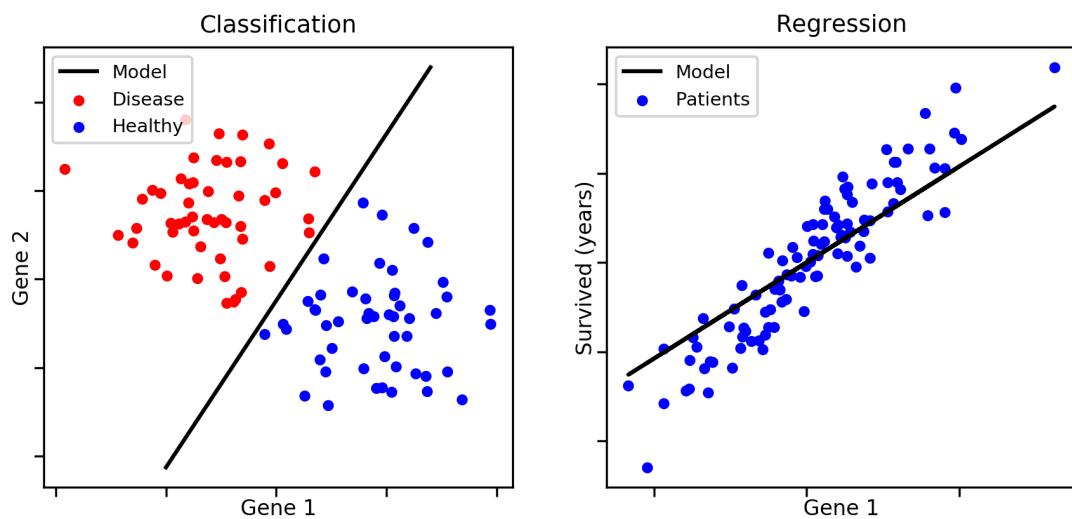


Abb. 6.1: Klassifikation (links) vs. Regression (rechts)

### 6.3.6 Abhängigkeitsentdeckung / Assoziationsanalyse

Die Abhängigkeitsentdeckung oder auch Assoziationsanalyse wird verwendet, um Zusammenhänge von Merkmalen und Muster und/oder Abhängigkeiten in grossen Datenbeständen zu finden.

Ein Beispiel davon ist die **Warenkorbanalyse**. Es soll gezeigt werden, dass Kunden, die Zahnbürsten kaufen, mit einer grossen Wahrscheinlichkeit auch Zahnpasta kaufen.

Die Analyse geschieht auf Basis von zwei Werten. Dem **Support** und der **Confidence**. Der Support ist die Wahrscheinlichkeit, dass der Kunde sowohl X, wie auch Y kauft. Die Confidence ist die Wahrscheinlichkeit, mit welcher der Kunde noch Y kauft, nachdem er bereits X im Warenkorb hat.

Mathematisch kann das so ausgedrückt werden:

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)} \quad (1)$$

Ein ebenfalls wichtiger Begriff ist der **Lift**. Er beschreibt die Abhängigkeit zwischen der Zahnbürste und der Zahnpasta im Warenkorb.

**Lift = 1:** X und Y werden unabhängig voneinander gekauft.

**Lift > 1:** Es besteht eine positive Abhängigkeit. Wenn jemand X kauft, kauft er wahrscheinlich auch Y.

**Lift < 1:** Es besteht eine negative Abhängigkeit. Wenn jemand X kauft, ist es unwahrscheinlich, dass er auch Y kauft.

**Beispiel** Es soll die Abhängigkeit zwischen dem Verkauf von **Zahnbürsten** und **Zahnpasta** überprüft werden.

- 20% aller Kunden kaufen sowohl eine Zahnbürste, wie auch Zahnpasta.
- 10% aller Kunden kaufen nur eine Zahnbürste.
- 40% aller Kunden kaufen nur Zahnpasta

$$\text{lift}(\text{Zahnbuerste} \rightarrow \text{Zahnpasta}) = \frac{\text{support}(\text{Zahnbuerste} \rightarrow \text{Zahnpasta})}{\text{support}(\text{Zahnbuerste}) * \text{support}(\text{Zahnpasta})} \quad (2)$$

Mit den oben genannten Zahlen eingesetzt ergibt dies:

$$\text{lift}(\text{Zahnbuerste} \rightarrow \text{Zahnpasta}) = \frac{0.2}{0.1 * 0.4} = 5 \quad (3)$$

Der Lift ist also wesentlich höher wie 1, woraus geschlossen werden kann, dass eine **starke Abhängigkeit** besteht zwischen dem Kauf von Zahnbürsten und Zahnpasta.

### 6.3.7 Clustering

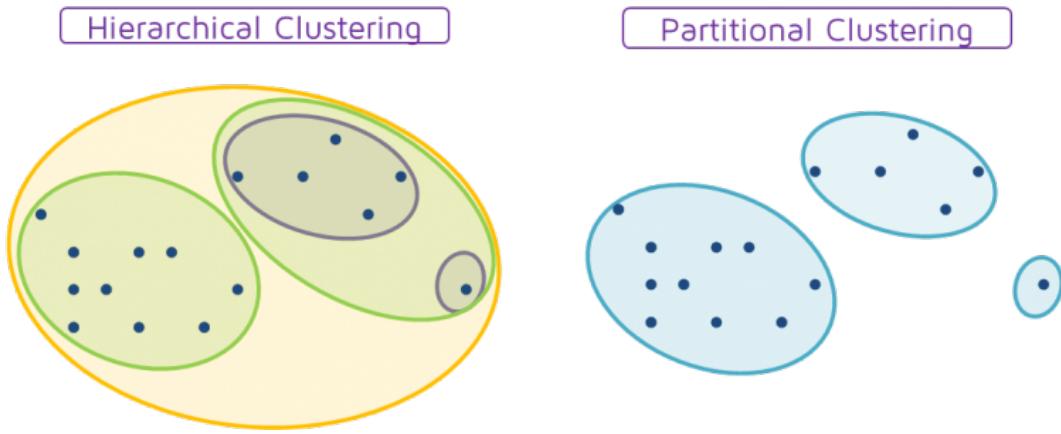


Abb. 6.2: Partitionierendes vs. Hierarchisches Clustering

Clustering ist ähnlich wie die Klassifikation, allerdings werden beim Clustering-Verfahren automatisch passende Klassen erstellt. Das Clustering-Verfahren zielt darauf ab, Gruppen oder Klassen zu erstellen, innerhalb von welchen sich die Objekte sehr ähnlich sind. Gleichzeitig sollten sich die erstellten Gruppen aber möglichst stark voneinander unterscheiden.

Clustering wird normalerweise nur zu Beginn ausgeführt, um Klassen zu erstellen, mit welchen anschliessend eine Klassifikation durchgeführt werden kann.

Es wird generell zwischen dem **partitionierenden** Clustering und dem **hierarchischen** Clustering unterschieden (Abbildung 6.2):

**Partitionierendes Clustering** unterteilt die gegebenen Daten in eine **vorgegebene Anzahl** von Klassen. Dabei werden zuerst die vorgegebene Anzahl Clusterzentren/Klassen aus den Datensätzen ausgewählt. Anschliessend werden die restlichen Datensätze diesen Clusterzentren zugeordnet.

Nach dem ersten Durchgang werden erneut Clusterzentren/Klassen gewählt und die Daten werden wiederum diesen, neuen, Clusterzentren zugeordnet. Dieses ganze Spiel wird solange wiederholt, bis die Daten 'optimal' geclustert sind (möglichst ähnlich innerhalb der Klasse/des Clusters, möglichst verschieden zwischen den Klassen/Cluster).

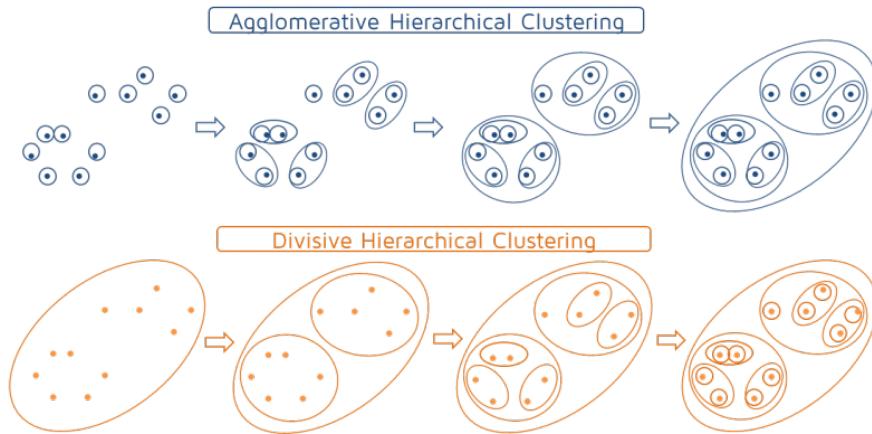


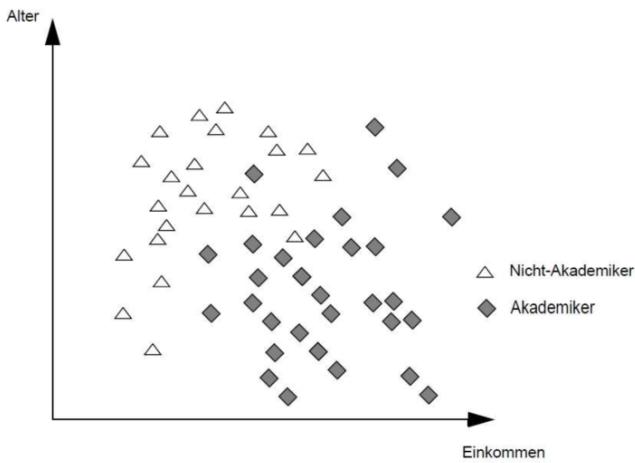
Abb. 6.3: Beispiel von agglomerativem (oben) und diversiven (unten) clustering

**Hierarchisches Clustering** kann wiederum in *agglomeratives* und *diversives* Clustering unterteilt werden. (Siehe Abbildung 6.3)

Das *agglomerative* Clustering funktioniert so, dass jeder Datensatz eine eigene Klasse/ein eigenes Cluster ist. Anschliessend werden die **zwei ähnlichsten Klassen zu einer zusammengefasst**. Dieses Zusammenfassen zweier ähnlicher Klassen wird solange durchgeführt wird, bis alle Daten in einem einzigen Cluster zusammengeführt sind.

Beim *diversiven* Clustering geht es in die andere Richtung. Die Datensätze werden bei jeder Iteration so halbiert, dass die Elemente innerhalb eines Clusters sich so ähnlich wie möglich sind, die Cluster untereinander aber möglichst unterschiedlich zueinander sind ('optimale' Cluster also). Diese Unterteilung wird solange fortgesetzt, bis jeder Datensatz seinen eigenen Cluster hat. Dieses Verfahren wird in der Praxis aber eher weniger angewandt.

### 6.3.8 Visualisierungstechnik



Bei der Visualisierungstechnik überlässt man die tatsächliche Auswertung dem Menschen. Die Daten werden mittels **zwei bis drei Merkmalen** im **zwei- oder dreidimensionalen** Raum dargestellt (z.B. Streudiagramm, siehe Abbildung 6.4). Es ist nun dem Menschen überlassen, aus dieser Grafik etwas nützliches herauszulesen.

Abb. 6.4: Daten visuell aufbereitet mittels eines Streudiagrammes

### 6.3.9 Fallbasierte Systeme

Solche Systeme greifen auf Daten von ähnlichen, bereits gelösten Problemen zurück. Es gibt eine **Falldatenbank** gefüllt mit bereits gelösten, getaggeden oder anderswie codierten Problemen, auf welche zugegriffen werden kann. (Siehe Abbildung 6.5)

Dieses System beruht auf der Annahme, dass geringe Änderungen an einer Problemstellung auch nur geringe Änderungen am Lösungsweg zur Konsequenz haben (also **lineare Probleme**). Somit sind fallbasierte Systeme *nicht* geeignet, um komplexe Probleme zu lösen.

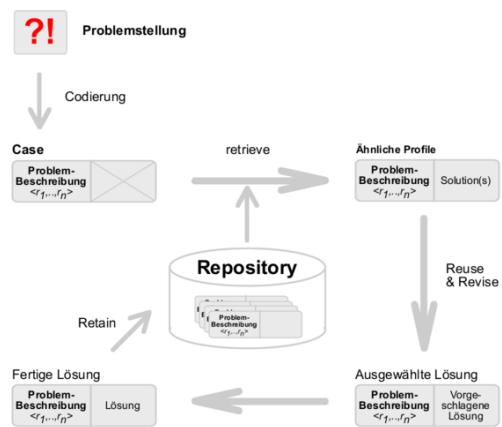


Abb. 6.5: Funktionsweise eines Fallbasierten Systems

### 6.3.10 Entscheidungsbaumverfahren (Buch S136)

Ein Entscheidungsbaum ist die **grafische Darstellung einer Datenteilung** (z.B. mittels Clustering). Dabei stellen die nicht-Blatt Knoten des Baumes die Merkmale der Daten dar, nach denen der Datensatz geteilt wird. Über die Kanten werden diese Merkmale verbunden. Die Teildatenbestände (z.B. Cluster oder Klassen) werden schlussendlich in den Blättern dargestellt.

Ein solcher Baum wird mittels des **Entscheidungsbaumverfahrens**. Zuerst wird ein Merkmal bestimmt, nach welchem sortiert werden soll (in Abbildung 6.6 wäre dies z.B. ob  $Schallengewicht \leq 0.168$  und  $MSE = 10.273$  ist) und es wird danach sortiert. Dieser Vorgang wird nun wiederholt (dieses mal ist z.B. das Merkmal links  $Schallengewicht \leq 0.059$  und  $MSE = 4.635$  ist).

Der Vorgang wird nun solange wiederholt, bis alle Datensätze zu einem Cluster gehören, oder sich kein Merkmal mehr finden lässt, durch welches geteilt werden kann.

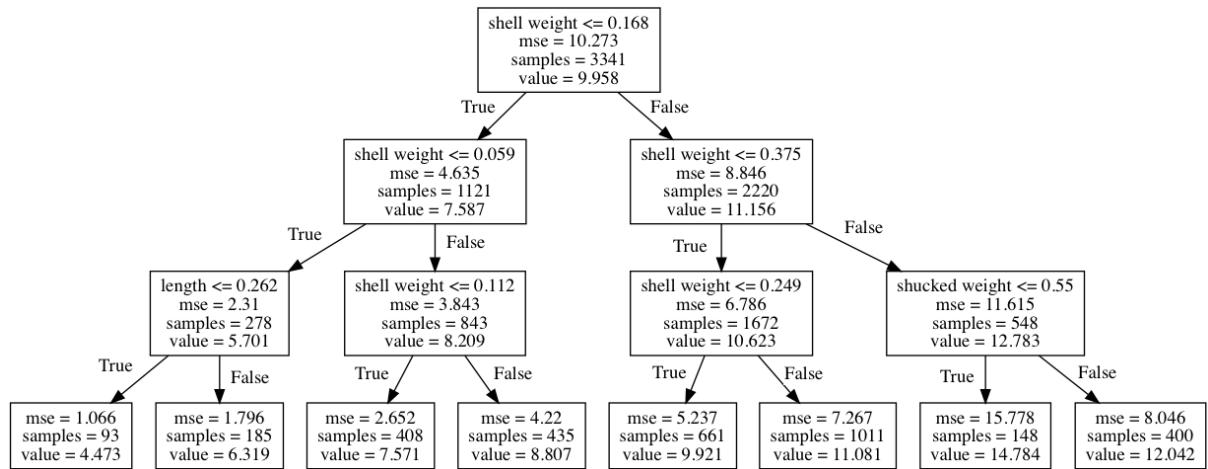


Abb. 6.6: Beispiel eines Entscheidungsbaumes

Um Bäume zu verbessern, die eine hohe Fehlerquote aufweisen, kann **Pruning** verwendet werden. Dabei werden einzelne Knoten und Kanten des Baumes entfernt, um den Baum zu verkleinern und somit die Teilgenauigkeit des Baumes insgesamt zu erhöhen.

Entscheidungsbäume werden zum Beispiel zur **Klassifizierung** von Daten hinzugezogen werden, um die Klassen einfacher und übersichtlicher darstellen zu können.

## 6.4 Architekturen (Skript S84/Buch S163ff)

### 6.4.1 Keine physische Integration

Strukturierte und unstrukturierte Daten existieren **parallel** in verschiedenen Speicherungsmöglichkeiten. Diese Daten werden nur verknüpft, falls beide für eine Auswertung benötigt werden.

### 6.4.2 Physische aber nicht logische Integration

Strukturierte und unstrukturierte Daten werden zwar in der gleichen Speicherungsmöglichkeit abgelegt, sind jedoch nicht verknüpft.

Unstrukturierte Daten werden oft als BLOB (Binary Large Object) gespeichert. BLOBS sind prinzipiell einfach riesige Binär-Dateien, in welche alle unstrukturierten Daten (Blogposts, Bewertungen, YouTube Videos etc.) reingeschmissen werden.

In relationalen Datenbanken werden BLOBS oft extern gespeichert. Das heisst, die Datenbank legt nur eine Referenz ab, wo das BLOB gefunden werden kann, sollte es gebraucht werden.

Je nach dem welcher DB-Engine verwendet wird, muss eine BLOB-Spalte in einer DB-Tabelle als *BLOB*, *LONGBLOB*, *LONGRAW* oder auch *BYTERA* definiert werden.

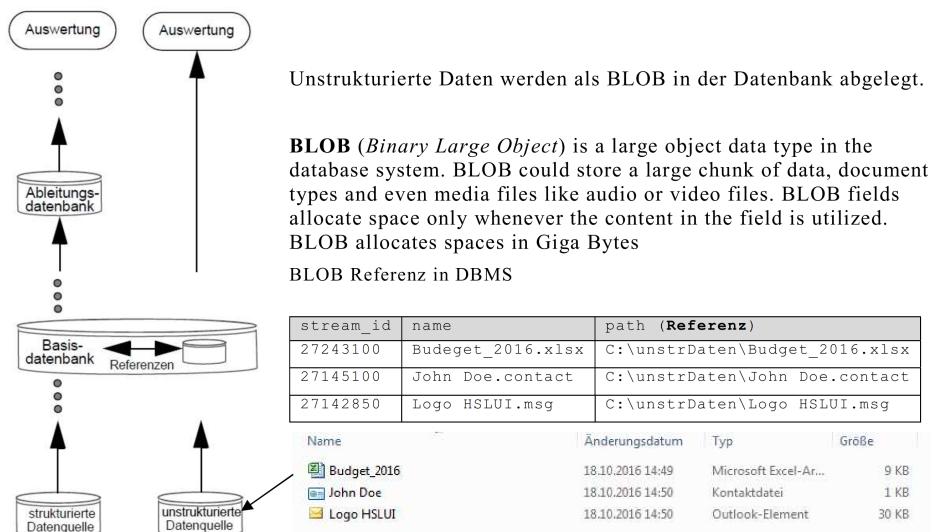


Abb. 6.7: Beispiel von BLOBs und strukturierten Daten in der selben DB

### 6.4.3 Physische und logische Integration

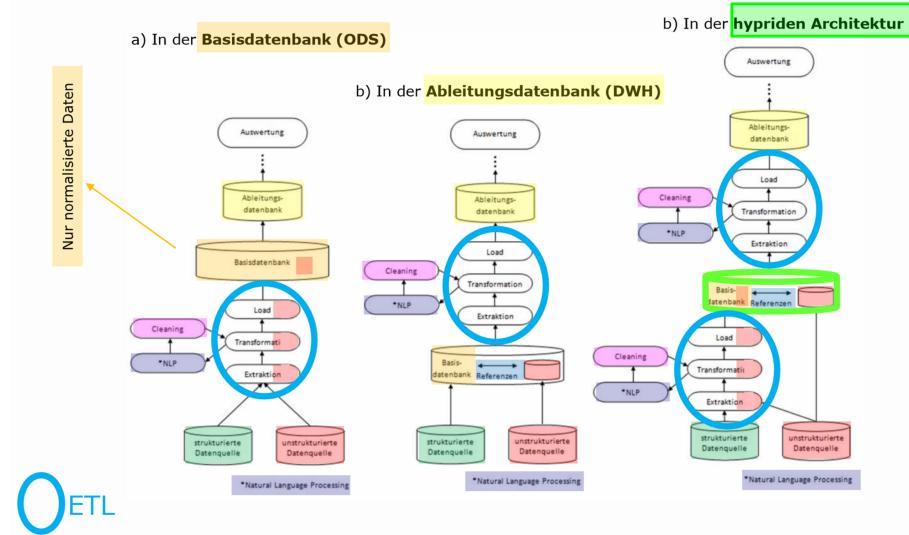


Abb. 6.8: Die drei verschiedenen Methoden zur Integration von unstrukturierten Daten.

**Extraktion von unstrukturierten Daten** geschieht meist mittels NPL und UIMA (Unstructured Information Management Architecture). Dabei muss darauf geachtet werden, dass zwar alle unnötigen Formatierungen entfernt werden, so dass die Daten gesäubert und integriert werden können, jedoch die wichtigen Formatierungen beibehalten werden (wie z.B. HTML-tags für title o.ä.)

**Transformation von unstrukturierten Daten** hat fast noch eine grössere Bedeutung als die Bereinigung von strukturierten Daten. Der Prozess besteht hauptsächlich daraus, Synonyme, Homonyme und Ähnlichkeit zu erkennen, irrelevante Daten wie Füllwörter oder Formatierungszeichen zu entfernen und/oder Daten in andere Sprachen zu übersetzen.

Für diesen Task werden Werkzeuge wie z.b. der DataWrangler, Open Refine oder Data-Cleaner verwendet, oder aber auch NLP-Werkzeuge wie GATE, UIMA oder Stanford's Core NLP Suite.

#### Laden von unstrukturierten Daten

- a) / **Basisdatenbank** Die Daten werden bereits im Datenbeschaffungsprozess analysiert, aufbereitet und strukturiert bevor sie in der DB abgelegt werden.  
Eine Verwendung der unstrukturierten Daten ist nach Ablegung nicht mehr möglich.
- b) / **Data Warehouse** Die Daten werden möglichst ohne Informationsverlust in der Basis-DB/ODS abgelegt und erst anschliessend im Data-Warehouse strukturiert.
- c) / **Hybrid-Architektur:** Die strukturierten und unstrukturierten Daten werden parallel in der Basis-DB abgelegt.

#### 6.4.4 Beispiel

**Aufgabenstellung:** Folgende E-Mail soll in ein Data-Warehouse integriert werden.

---

Subject: Beschwerde  
Von: peer.haber@bluewin.ch  
Gesendet: Di. 18.10.2016 15:27  
An: Customer Service

---

-----  
Geehrtes Support Team  
leider bin ich mit dem Vertrag vom 21.10.2016 nicht zufrieden. sie haben mir  
am Telefon erklaert, dass im Vertrag 5GB Daten inbegriffen sind.  
Leider ist dem nicht so. Bitte kontaktieren Sie mich unter folgender Nummer:

+41 79 678 23 63

mit freundlichen Gruessen  
Peer Haber  
peer.haber@bluewin.ch

---

Die Mail soll in die Architektur 3a) integriert werden. Dies geschieht in vier Schritten:

1. Extraktion der Daten aus den jeweiligen Ursprungsformaten → Extraktion
2. Analyse der Daten → Transformation
3. Datenbereinigung → Transformation
4. Laden der Daten in ein Datenbanksystem des Wata-Warehosues → Laden

#### Schritt 1 - Extraktion

Der Text des E-Mails wird aus dem Ursprungsformat extrahiert:

---

Geehrtes Support Team  
leider bin ich mit dem Vertrag vom 21.10.2016 nicht zufrieden. sie haben mir  
am Telefon erklaert, dass im Vertrag 5GB Daten inbegriffen sind.  
Leider ist dem nicht so. Bitte kontaktieren Sie mich unter folgender Nummer:

+41 79 678 23 63

mit freundlichen Gruessen  
Peer Haber  
peer.haber@bluewin.ch

---

## Schritt 2 - Analyse der Daten

Der extrahierte Text wird mittels Data Mining oder NLP analysiert:

The screenshot shows a text editor with an email message. The message content is:

Geehrtes Support Team,  
leider bin ich mit dem Vertrag vom 21.10.2016 nicht zufrieden. Sie haben mir am Telefon erklärt, dass im Vertrag 5 GB Data inbegriffen sind. Leider ist dem nicht so. Bitte kontaktieren Sie mich unter folgender Nummer:  
+41 79 678 23 63  
mit freundlichen Grüßen  
Peer Haber  
peer.haber@bluewin.ch

Below the text is a 'Annotation Types' panel with the following categories:

- DocumentAnnotation (selected)
- EmailAddress
- Name
- Sentence
- Token

Abb. 6.9: E-Mail-Analyse mit dem Text-Mining Tool UIMA

Die *semantische Auswertung* der oben gezeigten E-Mail zeigt folgendes:

**Kategorisierung:** Kategorie: Beschwerde

**Auswertung Freitexteingabe:** Unzufriedener Kunde.

**Auswertung Dokument:** Analyse des Vertrages, ob Kunde recht hat.

## Schritt 3 - Datenbereinigung

- Erkennen von Synonymen und Homonymen
- Beseitigung von irrelevanten Daten (z.B. Füllwörtern)
- Evtl. Übersetzung in andere Sprachen.

Geehrtes Support Team,

leider bin ich mit dem Vertrag vom 21.10.2016 nicht zufrieden. Sie haben mir am Telefon erklärt, dass im Vertrag 5 GB Data inbegriffen sind. Leider ist dem nicht so. Bitte kontaktieren Sie mich unter folgender Nummer:

+41 79 678 23 63

mit freundlichen Grüßen

Peer Haber

peer.haber@bluewin.ch

## Schritt 4 - laden der Daten in ein Datenbanksystem des Data-Warehouses

	kunden_id	name	vorname	mail	vertrag	vertrag_datum	zufrieden
▶	678	Haber	Peer	peer.haber@bluewin.ch	Ja	21.10.2016 00:00:00	Nein

## 7 Informations- und Datenqualität (Skript S95)

### 7.1 Definition

“Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung festgelegter oder vorausgesetzter Erfordernisse beziehen.” (DIN 55350/1995)

Eine weitere Definition nach Cordts 2009 lautet: *Die Datenqualität ist der Grad der Eignung von Daten zur Erfüllung eines bestimmten Verwendungszweck.*

Solche Erfordernisse sind zum Beispiel:

- Korrektheit
- Vollständigkeit
- Relevanz
- Konsistenz
- Aktualität

### 7.2 Ursachen und Orte von Qualitätsmängel

Schätzungsweise 10-20% aller Daten in einem operativen System sind fehlerhaft. Diese Daten zu verbessern nimmt ca. 80% des Aufwands im ETL-Prozess in Anspruch.

Schlechte Datenqualität kann folgende Ursachen haben:

**Schlechte Datenerfassung:** Durch menschliches Versagen in der Dateneingabe/Datenerfassung; Falsch (oder gar nicht) geschulter Mitarbeiter.

**Schlechte Prozesse:** Durch mangelhaft definierte Abläufe und Zuständigkeiten

**Schlechte Architektur:** Durch unsaubere/überhastete Systemwechsel/Migrationen oder infolge von Sparmassnahmen am falschen Ort.

**Schlechte Definitionen:** Durch mangelhafte/fehlende Planung oder Sorgfalt. Durch mangelhafte/fehlende Doku.

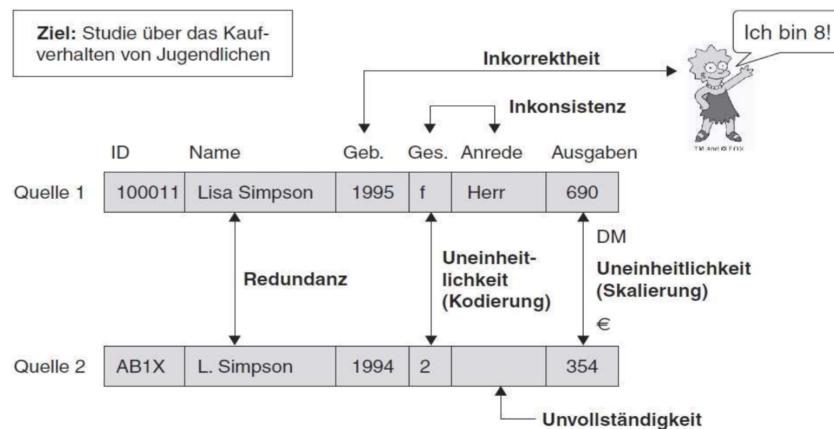


Abb. 7.1: Beispiel von schlechter Datenqualität

### 7.3 Messbarkeit von Datenqualität

Momentan gibt es noch keine akzeptierte Metrik für die Beurteilung der Datenqualität. Allerdings gibt es Grafiken zur Visualisierung des "sweet spot" zwischen überhöhten Kosten durch schlechte Datenqualität und überhöhten Kosten durch zu hohe Investitionen in die Verbesserung der Datenqualität (Abb. 7.3).

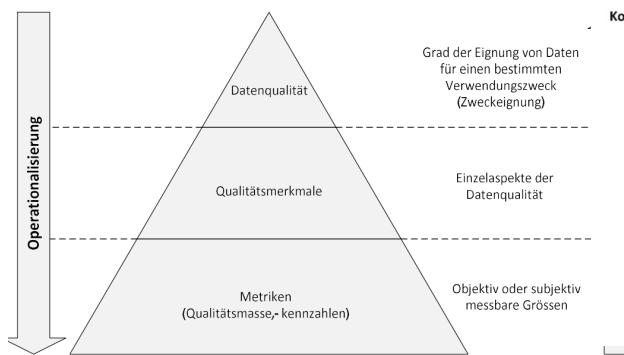


Abb. 7.2: "Messung" der Datenqualität

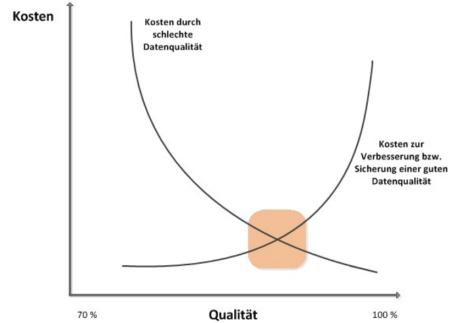


Abb. 7.3: Kosten-Qualitäts Gegenüberstellung

Zudem hat die DGIQ (Deutsche Gesellschaft für Informations- und Datenqualität) 15 Dimensionen der Datenqualität definiert. Diese sind in vier Kategorien eingeteilt. Diese Einteilung basiert auf einer Studie des MITs.

**Zugänglichkeit:** Informationen sind zugänglich, wenn sie anhand einfacher Verfahren auf direktem Weg für den Anwender abrufbar sind.

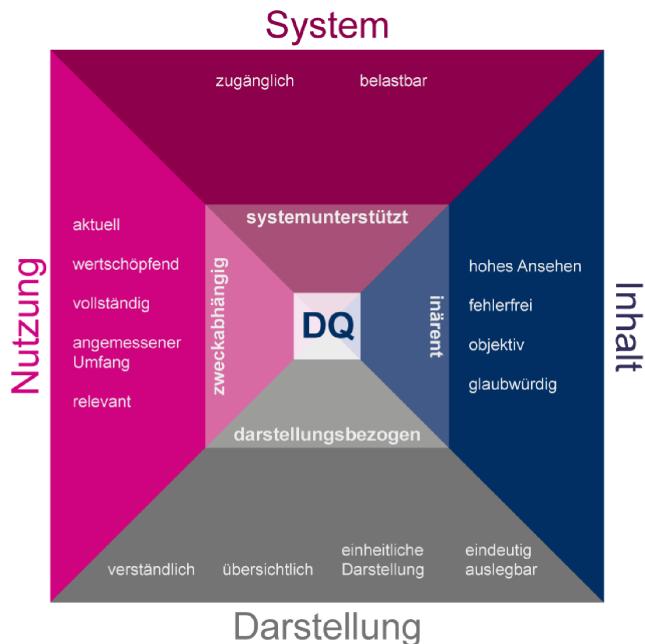
**Bearbeitbar:** Informationen sind leicht bearbeitbar, wenn sie leicht zu ändern/für unterschiedliche Zwecke zu verwenden sind.

**Hohes Ansehen:** Informationen sind hoch angesehen, wenn die Informationsquelle, das Transportmedium und das verarbeitenden System im Ruf einer hohen Vertrauenswürdigkeit und Kompetenz stehen.

**Fehlerfrei:** Informationen sind fehlerfrei, wenn sie mit der Realität übereinstimmen

**Objektiv:** Informationen sind objektiv, wenn sie streng sachlich und wertfrei sind.

**Glaubwürdig:** Informationen sind glaubwürdig, wenn Zertifikate einen hohen Qualitätsstandard ausweisen oder die Informationsgewinnung und -verbreitung mit hohem Aufwand betrieben werden.



**Eindeutig auslegbar:** Informationen sind eindeutig auslegbar, wenn sie in gleicher, fachlich korrekter Art und Weise begriffen werden.

**Einheitlich dargestellt:** Informationen sind einheitlich dargestellt, wenn die Informationen fortlaufend auf dieselbe Art und Weise abgebildet werden.

**Übersichtlich** Informationen sind übersichtlich, wenn genau die benötigten Informationen in einem passenden und leicht fassbaren Format dargestellt sind.

**Verständlich** Informationen sind verständlich, wenn sie unmittelbar von den Anwendern verstanden und für deren Zwecke eingesetzt werden können

**Relevant:** Informationen sind relevant, wenn sie für den Anwender notwendige Informationen liefern.

**Angemessener Umfang:** Informationen sind von angemessenem Umfang, wenn die Menge der verfügbaren Information den gestellten Anforderungen genügt.

**Vollständig:** Informationen sind vollständig, wenn sie nicht fehlen und zu den festgelegten Zeitpunkten in den jeweiligen Prozess-Schritten zur Verfügung stehen.

**Wertschöpfend:** Informationen sind wertschöpfend, wenn ihre Nutzung zu einer quantifizierbaren Steigerung einer monetären Zielfunktion führen kann.

**Aktuell:** Informationen sind aktuell, wenn sie die tatsächliche Eigenschaft des beschriebenen Objektes zeitnah abbilden

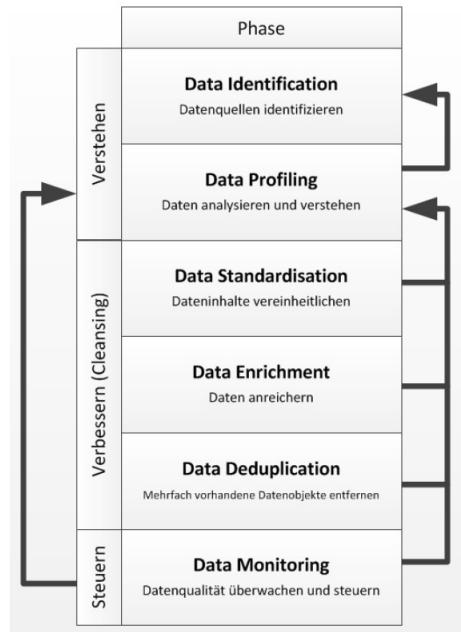
## 7.4 Verbesserung der Datenqualität

Die Verbesserung der Datenqualität kann in drei Phasen aufgeteilt werden, die wiederum in mehrere Tasks unterteilt sind.

1. Daten verstehen
  - Data Identification
  - Data Profiling
2. Daten verbessern
  - Data Standardisation
  - Data Enrichment
  - Data Deduplication
3. Daten Steuern
  - Data Monitoring

### 7.4.1 Daten verstehen

Was für Daten liegen vor?, Welche Daten kann man benutzen? Wo liegen diese Daten? Wer hat die Rechte auf diese Daten zuzu-griffen? Wer kann die Daten für die weitere Verwendung "freigeben"? Was bedeuten diese Daten? In welchem Kontext stehen diese Daten?



### Data Identification

*Woher kommen unsere Daten?* Durch die Auswertung der Metadaten kann einiges ermittelt werden, wie zum Beispiel die Art des Quellsystems.

- Wie und von wem wurden die Daten erstellt?
- Welche Benutzer greifen am häufigsten zu?
- Für welche Zwecke werden die Daten vor allem eingesetzt?
- In welchem Zustand befinden sich die Daten?

Bei der Data Identification werden die benötigten Daten identifiziert (wie das der Name schon sagt), ihr Ist-Zustand beschrieben und spezifiziert.

Das Ziel der Data Identification ist es, schlussendlich eine Metadatensammlung vorliegen zu haben und anhand dessen erste Angaben zur Datenqualität machen zu können.

## Data Profiling

Nachdem im vorherigen Schritt der Ist-Zustand der Daten beschrieben wurde, wird im Data-Profiling der Soll-Zustand beschrieben und spezifiziert. Das heisst, es werden Kriterien definiert wie zum Beispiel

- Benennungen
- Datentypen
- Domänen
- Minima und Maxima
- Kardinalitäten (Beziehungen zwischen Tabellen)
- Muster
- Fehlertoleranz
- Geschäftsregeln

Ausserdem werden Beziehungen von Daten untereinander festgehalten.

Jede Abweichung des Ist-Zustands vom definierten Soll-Zustand ist ein Qualitätsmangel. Somit kann genauer eingeschätzt werden, in welchem Gesamtzustand sich die Daten befinden. Dies wiederum ermöglicht es, Fehler zu ermitteln, priorisieren und anschliessend die Ursache zu beheben.

Es gibt grundsätzlich drei verschiedene Arten des Data-Profiling:

**deskriptiv/beschreibend:** Analyse von Häufigkeiten, Abhängigkeitsanalysen, Ausreisser-Tests etc.

**kognitiv/lernend:** Regelinduktionen, Segmentierungen oder Klassifizierungen

**deduktiv/ableitend:** (Geschäfts-)Regelanalyse

Eine oder mehrere dieser Data-Profiling Methoden wird auf sämtliche Datenobjekte des Systems sowie deren Beziehungen untereinander angewandt. Man untersucht die Daten einer Spalte auf die oben genannten Kriterien und kann somit Rückschlüsse auf die Datenqualität ziehen (→ Column Profiling)

Ebenfalls untersucht man die Abhängigkeiten von mehreren Spalten in einer Tabelle und kann somit z.B. überprüfen, in welcher Normalform sie sich befinden oder wie viele Fremdschlüsse in der Tabelle vorhanden sind (→ Multi Column Profiling)

Das Ziel des Data-Profiling ist es, ein sogenanntes Soll-Metadatenrepository vorliegen zu haben. Das heisst soviel wie, es soll eine Spezifikation vorliegen, wie die Metadaten der Daten aussehen müssen, bevor sie ins Zielsystem integriert werden können. Ausserdem liegt nach dem Data Profiling normalerweise ein detaillierteres Verständnis der vorliegenden Daten vor, da man sich eingehend mit ihnen beschäftigt hat.

## 7.4.2 Daten verbessern

Je gründlicher das Data Profiling durchgeführt wird, desto zuverlässiger und besser ist die Bereinigung. Je besser die Bereinigung, desto höher die Datenqualität.

### Data Standardisation

Daten müssen vereinheitlicht werden. Vereinheitlichung heißt meist auch Homogenisierung. Daten müssen in ein einheitliches Format und eine einheitliche Semantik gebracht werden.

Data Standardisation beinhaltet vor allem folgende Themen:

**Vervollständigen:** z.B. fehlende Vornamen nachtragen

**Synonyme entfernen:** Mehrere Schreibweisen eines Wortes mit der gleichen Bedeutung

Fa. vs. Firma

unverheiratet vs. ledig

Herr vs. Hr. vs. Mr.

etc.

**Homonyme entfernen:** Eine Schreibweise eines Wortes mit mehreren Bedeutungen

Peter (Vor- oder Nachname?)

Kim (männlich oder weiblich?)

Lieferung (Von hier oder zu hier?)

etc...

**Felder zusammennehmen:** Strasse und Hausnummer gehören zusammen

**Felder aufteilen:** Vor- und Nachname erhalten je ein eigenes Feld

**Rauschdaten eliminieren:** Entweder Jahrgang **oder** Alter. Beides ist überflüssig

**Rechtschreibung:** Straße / Strasse oder Brun-Brücke / Brunbrücke

**Einheitliche Darstellung:** +41 79 vs. 0079 / 03.02.2018 vs. 2018-02-03

**Eliminatin von Stoppwörtern:** z.B. der, die, das, und, oder, doch

**Einheitliche Abwandlungsformen:** Konjugation und Deklination

**Konjugation:** z.B. Welche Personenform wird genommen? (Er hat am 03.02 Geburtstag vs. Ich habe am 03.02 Geburtstag)

**Deklination:** Welche Fälle werden verwendet? (die schönen Häuser vs. der schönen Häuser)

**Einheitliche Einheiten:** Nur metrisch oder nur imperial

1

Data Standardisation wird häufig mittels linguistischen und algorithmischen Verfahren durchgeführt. Dabei werden Referenzdaten gesammelt, die anschliessend gepflegt werden müssen, dass sie später wieder angewendet werden können.

Das Ziel der Data Standardisation ist es, Daten in einer strukturell und inhaltlich direkt vergleichbaren Einheit vorliegen zu haben.

---

<sup>1</sup>Stoppwörter nennt man Wörter, die bei einer Volltextindizierung nicht beachtet werden, da sie sehr häufig auftreten und keine Relevanz für die Erfassung der Daten besitzen.

## Data Enrichment

Nachdem in der Data Standardisation unnötige Informationen entfernt worden sind, werden im Schritt des Data Enrichment zusätzliche, nützliche Informationen hinzugefügt, die zukünftige Analysen vereinfachen sollen. Dies können z.B. sein:

**Demographische Daten:** Nationalität, Herkunft, Schulbildung, Einkommen etc.

**Geographische Daten:** Land, Provinz, Bevölkerungsdichte etc. (Geodaten helfen oft bei der Deduplizierung von Daten)

**Produktcode:** EAN / EPC / ISBN etc. von Produkten

**Kundeninformationen:** Bestell- und Zahlungsverhalten, Interessen etc.

**Tags:** z.B. "Spätzahler", "Tierliebhaber", "Technologiebegeistert"

**Umsatz:** Umsatz, der der Kunde generiert.

**Sumindex:** Summe aller Bestellungen, die der Kunde im Laufe einer Zeitspanne aufgegeben hat.

Das Ziel des Data Enrichments ist es, Daten in so einem Format vorliegen zu haben, die die Suche nach Dubletten vereinfacht. Ebenfalls können mit diesen angereicherten Daten bessere Analysen durchgeführt werden. Data Enrichment hat überhaupt keinen Einfluss auf die Datenqualität, jedoch kann die Datenmenge erheblich ansteigen.

## Data Deduplication

Dubletten oder auch Duplikate sind Datensätze, die zwar syntaktisch unterschiedlich sind, semantisch jedoch dasselbe meinen. Es wird zwischen zwei Arten von Dubletten unterschieden: Den *vollständig identischen* und den *sich bis zu einem gewissen Grad ähnlichen* Dubletten.

Dubletten sind redundant und verstossen somit gegen die Information Quality Dimensionen. Ursachen für Dubletten können z.B. ein Fehler bei der Data Standardisation sein (Grösse sowohl in inch wie auch in cm angegeben), oder auch nicht aktuelle Daten (Kunde zweimal erfasst nachdem er umgezogen ist, einmal mit der neuen und einmal mit der alten Adresse).

Es gibt jedoch auch Fälle, da sind Dubletten gewollt. Das kann z.B. sein bei einer Mehrfachablage für eine Verlosung, oder aber auch bei Gratis Trials von Online-Diensten (ich bin einmal Peter Müller und nach dem Ablauf des Gratis Trials bin ich plötzlich Sandro Hüslar, aber eigentlich immer noch die selbe Person)

Dubletten können Anomalien und Fehlern in Analysen und Prognosen auslösen oder können in grosser Anzahl auch zu einem Ressourcenproblem führen. Deshalb sollten sie so gut wie möglich beseitigt werden. Diese Beseitigung findet in zwei Teilschritten statt:

1. Dublettenerkennung Dies wird meist mittels Clustering durchgeführt. Die gefundenen möglichen Dubletten werden aufgelistet und auf Redundanz untersucht.
2. Eliminierung und/oder Merging von Dubletten (Siehe Abb. 7.5)

Mehrere Arten der Dublettenerkennung und -entfernung finden sich im Skript auf S120ff.

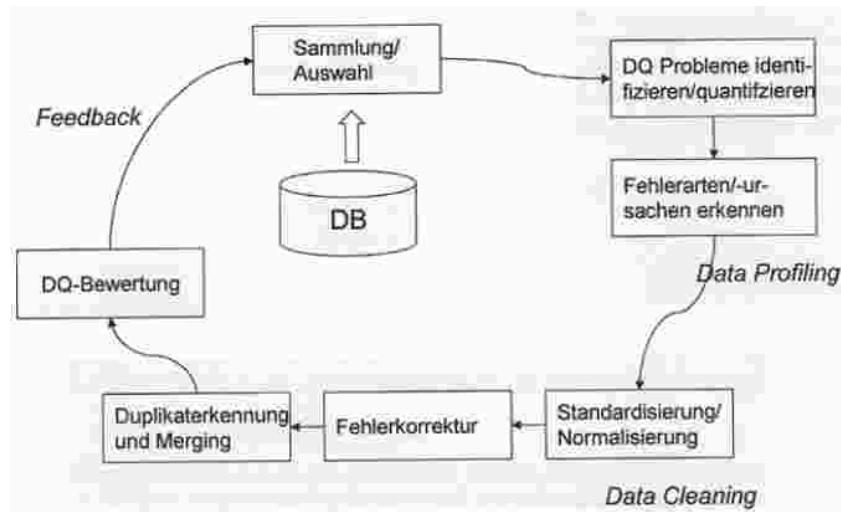


Abb. 7.5: Prozess der Dublettenbeseitigung als iterativer Prozess

### 7.4.3 Daten steuern

Da sich Daten immer wieder ändern, können Sie sich auch ungewollt verschlechtern. Ein System- und/oder Datenqualitätsowner muss da dauernd ein Auge darauf haben.

## Data Monitoring

Wie bei den letzten paar Themen klar geworden sein sollte, ist die Datenqualitätspflege ein iterativer Prozess. Das Monitoring begleitet diesen Prozess und dokumentiert und misst diesen.

Aufgaben des Data Monitorings sind z.B.

- Messen und Steuern der Datenqualitätskennzahlen (→ Data Auditing)
  - Alarmierung bei Regelverletzung
  - Erkennen von (Daten-)Trends

Das Ziel des Data Monitorings ist es, ein Höchstmaß von Wissen und Bewusstsein für den momentanen Stand der Datenqualität zu haben.

## 8 Historisierung (Skript S144/Buch S195)

Unter der Historisierung versteht man, dass man nicht nur aktuelle Datenbestände verwaltet, sondern auch veraltete Daten. Wenn also z.B. ein Mitarbeiter eine Lohnerhöhung erhält, so wird der alte Lohn nicht überschrieben, sondern beide Einträge existieren parallel.

Es existieren verschiedene Methoden zur Historisierung. In diesem Modul wird hauptsächlich die SCD-Versionierung behandelt. SCD steht für Slowly Changing Dimensions. Es gibt die SCD-Typen 0, 1, 2 und 3. Wir werden uns ebenfalls kurz mit der Tupelversionierung auseinandersetzen.

### 8.1 SCD Typ 0 - Keine Historisierung, keine Aktualisierung

Es wird **keine Historisierung** durchgeführt. Es wird überprüft, ob unter diesem Primärschlüssel bereits ein Datensatz existiert. Falls dem so ist, **wird nichts gemacht**. Ansonsten wird der neue Datensatz in die Datenbank eingefügt.

Es werden also nur neue Daten hinzugefügt, bereits existierende Datensätze werden jedoch **nicht** verändert.

### 8.2 SCD Typ I - Keine Historisierung, Aktualisierung

Es wird ebenfalls **keine Historisierung** durchgeführt. Im Gegensatz zu SCD Typ 0 sind Aktualisierungen jedoch möglich. Erhält ein Mitarbeiter also eine Lohnerhöhung von CHF 5'000 und sein Lohn steigt von CHF 10'000 auf CHF 15'000, dann wird der alte Lohn in der Datenbank mit dem neuen Lohn überschrieben. Es gibt kein Zeichen mehr davon, dass der Mitarbeiter jemals nicht CHF 15'000 verdient hat.



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA
123	ABC	Acme Supply Co	IL

Abb. 8.1: Beispiel einer Aktualisierung.

Es muss jedoch darauf geachtet werden, dass z.B. persistente Datenaggregate (z.B. Views), die mit dem alten Datensatz zusammenhängen nun evtl. nicht mehr korrekt sind und somit neu berechnet werden müssen.

### 8.3 SCD Typ II - Vollhistorisierung

Diese Methode findet am meisten Anwendung. Es wird **nichts überschrieben**. Stattdessen werden zwei Extra-Spalten **dat-von** und **dat-bis** unterhalten, in denen festgehalten wird, in welchem Zeitraum dieser Datensatz gültig ist.

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date
123	ABC	Acme Supply Co	CA	01-Jan-2000	21-Dec-2004
124	ABC	Acme Supply Co	IL	22-Dec-2004	

Abb. 8.2: Beispiel einer Vollhistorisierung.

Im Gegensatz zu SCD Typ 1 sind persistente Datenaggregate immer noch korrekt, da der alte Datensatz ja noch existiert. Sofern sie mit den korrekten Versionen verbunden sind zumindest.

## 8.4 SCD Typ III - Historisierung mit neuem Attribut (Spalte)

Bei jeder Aktualisierung eines Datensatzes wird der gesamten Tabelle eine neue Spalte angehängt, in welche der alte Wert geschrieben wird.

Supplier_Key	Supplier_Code	Supplier_Name	Original_Supplier_State	Effective_Date	Current_Supplier_State
123	ABC	Acme Supply Co	CA	22-Dec-2004	IL

Abb. 8.3: Beispiel einer Historisierung mit neuer Spalte.

Bei dieser Art der Historisierung wird jedoch meistens nur der letzte alte Wert gespeichert. Sollte die Acme Supply Co nun also wieder in einen neuen Staat ziehen, so wird CA mit IL überschrieben und IL wird mit dem aktuellen Staat überschrieben. Somit ist diese Art von Historisierung eigentlich nur bei solchen Daten zu empfehlen, bei welchen eine einmalige Änderung zu erwarten ist.

## 8.5 Tupelversionierung

Bei der Tupelversionierung werden die Tupel nicht mit einem Zeitstempel, wie in SCD Typ II, sondern mit einer Versionierungsnummer versehen. Es werden keine bestehenden Tupel aktualisiert, sondern nur neue Tupel erzeugt. Das Tupel mit der höchsten Versionierungsnummer ist jeweils das aktuelle gültige.

Produkt			
ANR_VNR	Artikel	Produktgruppe	Produktfamilie
1235-001	Quickphone 150	Singleband	Mobiltelefon
1235-002	Quickphone 150	Dualband	Mobiltelefon
1237-001	Quickphone 100	Singleband	Mobiltelefon
1239-001	Quickphone 200	Dualband	Mobiltelefon
⋮	⋮	⋮	⋮

Faktentabelle					
ANR_VNR	Filial_ID	Zeit_ID	Verkauf	Einkauf	Preis
1235-001	50013	02.03.2011	60	200	299,00
1237-001	50013	02.03.2011	31	150	599,00
1235-002	50013	05.03.2011	50	300	199,00
1237-001	50013	05.03.2011	20	120	99,00
1235-002	50013	06.03.2011	53	140	199,00
1239-002	50013	06.03.2011	35	134	99,00
124623-003	50013	07.03.2011	40	123	2499,00
⋮	⋮	⋮	⋮	⋮	⋮

Abb. 8.4: Tupelversionierung

## 9 Multidimensionale Datenmodellierung (Script S153 / Buch S201)

### 9.1 Nutzen von MDDB

Ab einer gewissen Komplexität sind Entity-Relationship Diagramme schlicht und einfach nicht mehr brauchbar (z.B. in Abb. 9.1).

Ebenfalls benötigt man ab einer gewissen Komplexität eine unschön grosse Anzahl von Joins, um anständige Datenaggregate zu kriegen, was bald auch zu Performanceproblemen führt.

Die Lösung der beiden oben genannten Probleme heißt **multidimensionale Datenbanken**. Anstelle von einer komplizierten zweidimensionalen Darstellung der Daten nehmen wir einfach eine Dimension hinzu .

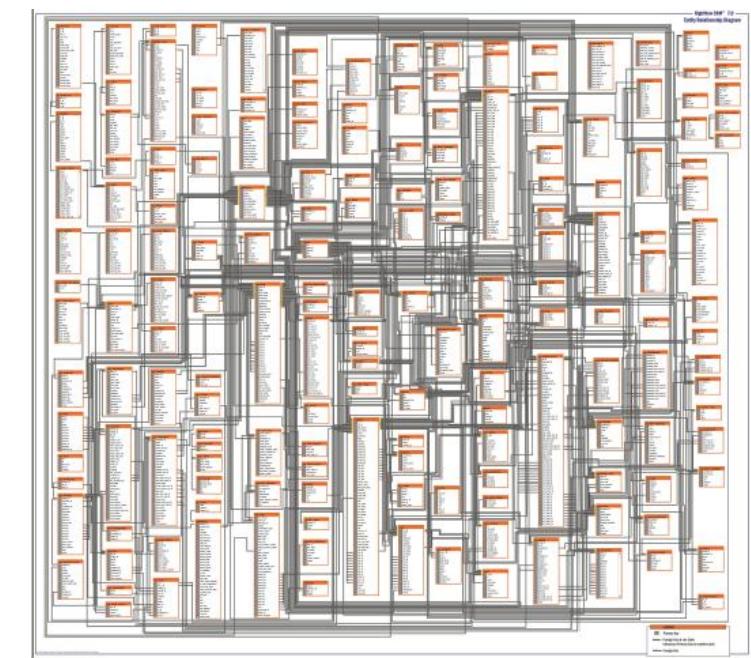


Abb. 9.1: Ein solches ERD ist nicht mehr übersichtlich und somit nutzlos

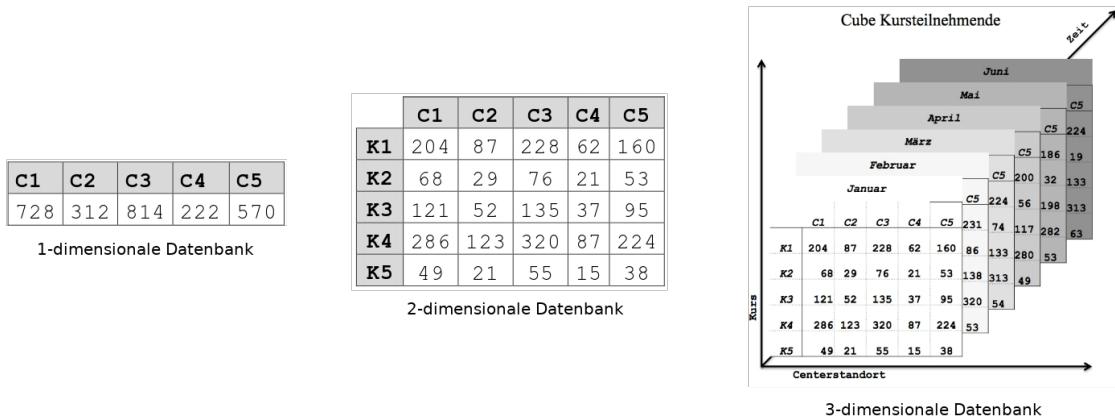


Abb. 9.2: Vergleich verschiedener Datenbank-Dimensionen

In Abbildung 9.2 kann eine Gegenüberstellung der verschiedenen Datenbank-Dimensionen gesehen werden. Jede Dimension fügt ein zusätzlicher Wert hinzu. So wird in der 1-dimensionalen Datenbank nur das Kurszentrum gespeichert, während in der 2-dimensionalen Datenbank zusätzlich noch die Anzahl Kursteilnehmer pro Kurs hinterlegt wird. Die 3-dimensionale Datenbank schichtet nun mehrere 2-dimensionale Datenbank-”scheiben” aufeinander in eine Würfelform. Jede dieser ”Scheiben” symbolisiert ein Monat.

Anstelle, dass man nun ein ellenlanges Join-Statement von allen möglichen Tabellen machen

muss, wenn man herausfinden will, wie viele Teilnehmer der Kurs K2 im Center C4 im Februar hatte, muss man nur die korrekten Koordinaten wissen (3/3/1).

Eine MDDB (Multi-Dimensional Data Base) hat viele Vorteile. Einige davon sind:

- Intuitive Benutzeroberfläche
- Kurze und stabile Antwortzeiten
- Hohe Zugriffs-, Analyse- und Daten Präsentationsflexibilität
- Interaktives Navigieren in den Datenbeständen
- Schnelle Erstellung von geeigneten Ad-hoc-Berichten und Grafiken
- Kein technisches Datenbank-Wissen vorausgesetzt
- Vereinfachte Datenmustererkennung
- Vereinfachte Erkennung von schlecht bereinigten Daten aufgrund von unplausiblen Analysedaten

## 9.2 Kennzahlen, Fakten und Dimensionen

**OLAP-Systeme** werden verwendet, um anhand von Kennzahlen Entscheidungen zu treffen (Siehe Kap. 1.3-1.5). Das heisst, in den Datenbanken werden **Kennzahlen** gespeichert.

Eine solche **Kennzahl** symbolisiert eine **Eigenschaft**. Das mag z.B. ein Umsatz, ein Preis oder auch ein monatlicher Stromverbrauch sein.

Ein anderer Begriff für Kennzahlen sind **Fakten**. Mehrere Fakten können zu einem neuen Fakt aggregiert werden (z.B. Kennzahl 330 und Kennzahl 54 können aggregiert werden zum Fakt, dass das Produkt mit der PID 54 CHF330.- kostet.)

Zusätzlich wird der **Geschäftsaspekt**, auch **Dimension** genannt hinzugenommen. Er beschreibt den **Beobachtungspunkt oder -raum** der Kennzahl/des Fakts. So ein Dimension kann z.B. eine Filiale, eine Abteilung oder auch ein Quartal sein. Dimensionen sind sogenannte **deskriptiven Werte**, das heisst sie enthalten **keine messbaren Werte**, sondern geben den vorhandenen Kennzahlen lediglich einen **Kontext**.

Dabei liefern gröbere Dimensionen eine gröbere Granularität des Faktes (Dimension HSLU vs. Dimension Student → Fakt Verwaltungskosten Universität vs. Verwaltungskosten Student).

Es ist nicht immer einfach, Fakten und Dimensionen auseinanderzuhalten. So kann ein Preis sowohl Fakt, wie auch Dimension sein. Wenn man die Entwicklung des Preises im Verlaufe der Zeit beobachtet, dann ist der Preis ein Fakt. Wenn man nun aber die Anzahl von verkauften Produkten in verschiedenen Preissegmenten beobachtet, dann ist der Preis die Dimension.

Grundsätzlich kann man sagen *Eine Analyse basiert primär auf der Interpretation der Ausprägungen eines Fakts in mehreren Dimensionen.*

### 9.3 OLAP Würfel-Paradigma

Je mehr Dimensionen man hat, desto feiner wird die Granularität der Analyse.

Um zurückzukommen auf Abb. 9.2: In der 1-dimensionalen Datenbank kann eine 1-dimensionale Analyse durchgeführt werden:

*Wie viele Kursteilnehmer hat das Kurszentrum C3.*

**Dimension:** Kurszentren

**Fakten:** Kursteilnehmen

Nehmen wir eine Dimension hinzu, so kann in der 2-dimensionalen Datenbank bereits eine 2-dimensionale Analyse durchgeführt werden. Zusätzlich kommt nun noch die Kurs-Dimension hinzu:

*Wie viele Kursteilnehmer hat der Kurs K2 im Kurszentrum C3*

**Dimension:** Kurszentren, Kurs

**Fakten:** Kursteilnehmen

Nehmen wir nun zusätzlich noch die dritte Dimension hinzu, so kann nun eine 3-dimensionale Analyse durchgeführt werden, die wiederum feingranularer ist wie die 2-dimensionale Analyse:

*Wie viele Kursteilnehmer hat der Februar-Kurs K2 im Kurszentrum C3*

**Dimension:** Kurszentren, Kurs, Monat

**Fakten:** Kursteilnehmen

Man kann nun noch beliebig viele Dimensionen hinzufügen, bis man die gewünschte Granularität erreicht hat (Wie viele weibliche Kursteilnehmer in der Altersgruppe 35-40, die im Bereich Gesundheitspflege tätig sind, hatte der Februar-Kurs K2 2016 im Kurszentrum C3?). Man ist nicht an den 'physischen' 3-Dimensionalen Raum gebunden, sondern kann so viele Dimensionen erstellen, wie man möchte, denn **der Würfel ist nur ein Modell und nirgends so materialisiert**. Das einzige Problem, das auftauchen könnte ist, dass es ein bisschen schwierig ist, ein 10-dimensionaler Würfel auf Papier darzustellen.

#### 9.3.1 Die 10 OLAP Würfel-Paradigmen

1. Sechs bis max. acht Dimensionen bleiben analytisch überschaubar und praktikabel
2. Die verwendeten Dimensionen müssen unabhängig voneinander sein
3. Die verwendeten Dimensionen können durchaus einer parallelen Dimensionshierarchie entnommen sein wie z.B. Print oder Broadcast
4. Es muss nicht an jeder Dimensionskante gleich viele Elemente haben. In Abb. 9.2 hat es z.B. 5 Kurse und 6 Monate.
5. Die Elemente einer Dimension müssen immer die gleiche Granularität besitzen. Ihre Fakten sind also unmittelbar vergleichbar. Im Beispiel von Abb. 9.2 sind es Monate als Zeitangabe und nicht z.B. Monate und Quartale gemischt.
6. Unter Kombination mehrerer Dimensionen lassen sich mit dem gleichen Zahlenmaterial unterschiedliche Würfel herstellen. Jeder Würfel ist auf ein Analysemuster hin optimiert. Jede Analyse braucht ihren eigenen Würfel.

7. Das Produkt (Projektion) aller Elemente aller Dimensionen ergibt die Anzahl der Fakten (oder NULL falls keine Daten vorhanden sind)
8. Eine Zeitdimension hat es fast immer. Eine solche lässt zeitbezogene Entwicklungsanalysen zu.
9. Die Fakten eines Würfels dokumentieren immer je nur eine relevante Kenngröße wie z.B. den Umsatz oder den Gewinn
10. Eine Dimension bringt Fakten immer in die gleiche Klasse von Beobachtungsräumen: Pro Monat, pro Quartal, pro Jahr, pro Abteilung etc.

### 9.3.2 Grundoperationen am Würfel

**Slicing:** Ausschneiden von Scheiben aus dem Würfel. Ändert nichts an der Granularität, macht aus einer 3-D Analyse eine 2-D Analyse

**Dicing:** Durch Einschränken von einer (oder mehreren) Dimensionen wird aus dem Würfel ein kleinerer erzeugt. (Z.B. aus dem Würfel in Abb. 9.2 den Würfel der Kursteilnehmer von Kurs K2/K3 im Zentrum C2/C3 während Jan/Feb ausschneiden.)

**Pivoting/Rotation:** Drehen des Würfels, so dass min. eine andere Dimension sichtbar ist

**Drill-Down:** Detaillierung eines Datenfeldes auf einzelne Werte (z.B. von Monaten auf einzelne Tage)

**Drill-Up / Roll-Up:** Gegenteil von Drill-Down; Verallgemeinerung eines Datenfeldes (z.B. von Monaten auf Quartalssicht)

**Drill-Across:** Verschiebung des Fokus auf derselben Dimensionsstufe (z.B. andere Region oder anderes Produkt)

**Drill-Through:** Ähnlich wie Drill-Across; Auswertung der gleichen Ansicht mit anderen Würfeln

**Drill-In / Merge:** Verringerung der Granularität durch Entfernen von einzelnen Dimensionen

**Split:** Gegenteil von Merge; Erhöhung der Granularität eines einzelnen Wertes durch Hinzufügen zusätzlicher Dimensionen (z.B. den Umsatz dieses Produktes in dieser exakten Filiale berechnen)

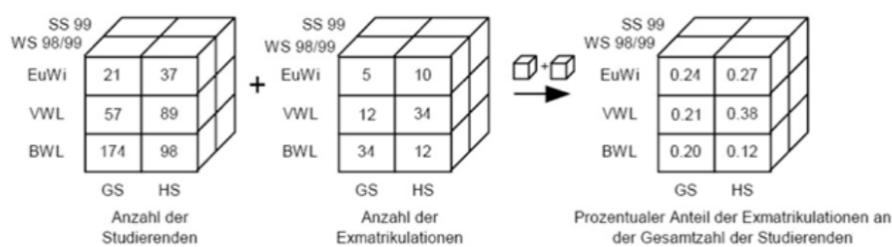


Abb. 9.3: Beispiel einer Drill-Across Operation

## 9.4 Dimensionstruktur, -klassifizierung und -hierarchie

Jede Dimension sollte eine Zusammenfassung mehrerer Elemente zu einem neuen, übergeordneten Dimensionswert ermöglichen (→ roll-up). Das Gegenteil von roll-up ist drill-down. Diese beiden Funktionen ergeben einen hierarchischen 1:n Klassifikationspfad, wie zu sehen in Abb. 9.4.

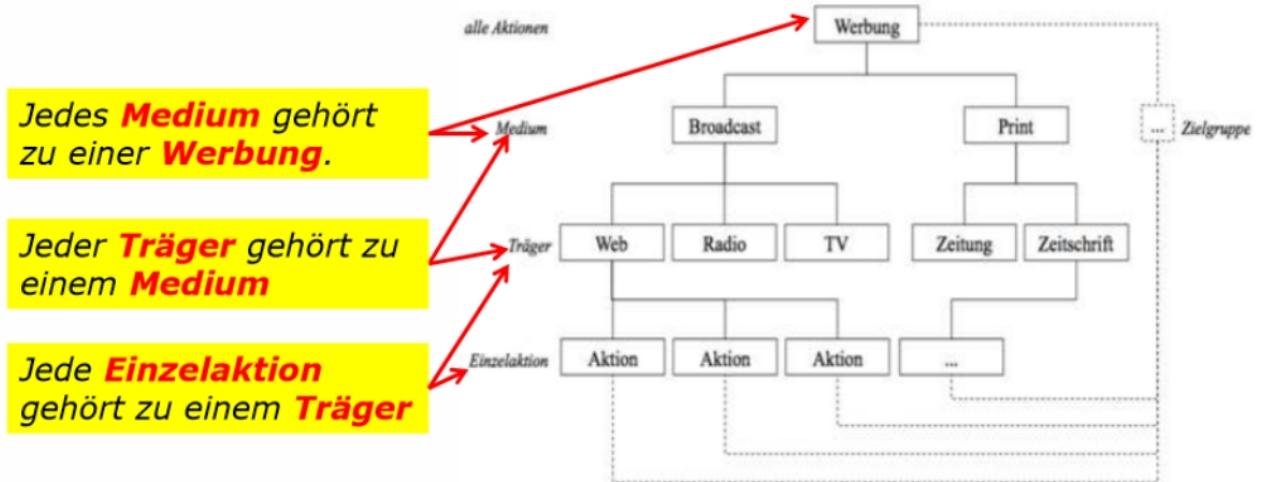


Abb. 9.4: Klassifikationshierarchie

### 9.4.1 Die 9 Regeln der Klassifikationshierarchie

1. Die Klassifikationsstruktur ist strukturiert
2. Der Klassifikationsbaum muss vollständig sein. (Gehen wir davon aus, dass der Baum nach Radio und TV noch weitergeht)
3. Die Bestimmung der Elemente und Zuordnung der Unterelemente ist schwer aber unerlässlich für künftige Analysen (Gehören Plakat-Aktionen zu Print oder Broadcast?)
4. Die Bestimmung der Hierarchiestufen ist schwer.
5. Höchstens 7 Hierarchiestufen mit je max. 15 - 20 Elementen um die Übersicht zu bewahren
6. Es ist durchaus möglich, dass die gleichen Strukturelemente bis hin zur Wurzel parallel nach einer weiteren Sachlogik zusammengefasst werden. Im abgebildeten Beispiel analysieren wir nach dem Medium. Es könnte aber auch nach Zielgruppe zerlegt werden. So entsteht eine neue, parallele Klassifikationshierarchie.
7. Dimensionen aus parallelen Klassifikationshierarchien gelten als unabhängig und können deshalb im Würfel eigene Kanten bilden.
8. Die Hierarchie stellt uns oft vor die Frage wonach man detaillieren soll. Straftaten nach Alter oder nach Geschlecht oder nach Nationalität? Dies wird durch die geplanten Analysen bestimmt. Die Managementfragen sind zentral, bevor man mit dem Design der Dimensionsstruktur beginnt. Es kann dann halt sein, dass in mehreren Ästen eines Baumes gleiche Elemente vorkommen.
9. Eine zusammengefasste Dimension ist eine neue Dimension. Aber die beiden Dimensionen sind nicht unabhängig, können also keine eigenen Würfel-Kanten sein. Parallel Dimensionen sind unabhängig.

## 9.5 Dünnbesetztheit

In der Praxis sind längst nicht alle Würfel 'voll besetzt'. Diese fehlenden Daten werden auch als 'missing Data' bezeichnet. Es wird prinzipiell zwischen drei Arten von missing Data unterschieden:

**unmöglich** Es ist unmöglich, solche Daten zu erheben

**unbekannt:** Die Daten sind einfach nicht bekannt.

**(noch) nicht eingetreten:** Das Event für diese Daten ist (noch) nicht eingetreten.

Zusätzlich wird zwischen der natürlichen und der logischen Dünnbesetztheit unterschieden.

**Natürliche Dünnbesetztheit** beschreibt die Dünnbesetztheit aufgrund von nicht eingetretenen Daten (non-event) (Im Juli werden keine Skis verkauft). Jedoch hat man nicht NULL Skis verkauft, sondern 0. Mathematisch gesehen ist der Würfel also voll besetzt und es kann problemlos gerechnet werden.

**Logische Dünnbesetztheit** beschreibt ebenfalls die Dünnbesetztheit von nicht eingetretenen Daten, jedoch nicht non-event nicht-Daten sondern event-not-applicable nicht-Daten. Nehmen wir an, ein Produkt ist im Feb. 2018 auf den Markt gekommen. Es ist recht schwer, Verkaufsdaten für dieses Produkt im Nov. 16 zu finden. Diese Verkaufszahlen wären undefined oder NULL. Im Gegensatz zur natürlichen Dünnbesetztheit kann mit der logischen Dünnbesetztheit nicht korrekt gerechnet werden, da gewisse Anfragen aufgrund einer Unmöglichkeit zurückgewiesen werden und somit zu fehlenden oder verfälschten Analysen führen können.

Deshalb müssen undefinierte Felder in einem Würfel dementsprechend gekennzeichnet werden (entweder mit NULL oder n/a (not applicable)). Welcher Wert genau in undefinierte Datenfelder geschrieben werden soll, müsste in den Metadaten hinterlegt werden.

Bei einem logisch dünnbesetzten Würfel kann die Anzahl der undefinierten und diejenige der definierten (auch 0) Zellen verglichen werden. Daraus erhält man den Füllgrad des Würfels, d.h. wie viel Prozent des Würfels enthält tatsächlich Daten. Bei einem sehr hohen Füllgrad spricht von einem dicht besetzt (engl. dense) Würfel im Gegensatz zu einem dünn besetzten (sparse) Würfel.

## 9.6 Modelle in multidimensionalen Datenbanken

In den bisher bekannten Datenbanken benötigt man zwei Modelle bzw. Schemata: Das **logische Modell** und das **physische Modell**. Bei multidimensionalen Datenbanken kommt nun zusätzlich noch das **semantische Modell** dazu.

Die folgenden Modelle sind nach absteigendem Abstraktionslevel geordnet.

### 9.6.1 semantisches Modell

Der Datenwürfel mit seinen Dimensionen, Hierarchien und Fakten ist ein gutes semantisches Modell und hilft uns bei der Vorstellung von mehrdimensionalen Analysen. Der Datenwürfel ist eine möglichst wahrheitsgetreue Abbildung der Realität.

### 9.6.2 logisches Modell

Das logische Modell wird benötigt, um die Business-Logik (Requirements, interne Organisation, Prozesse etc.) darzustellen. Das Endprodukt eines logischen Modells ist normalerweise das **ERD** (Entity Relation Diagram). Das ERD zeigt die Beziehungen zwischen verschiedenen Kategorien von Daten und ist essenziell für das Datenbank-Design.

### 9.6.3 physisches Modell

Das physische Modell baut anschliessend auf dem vorher erstellten logischen Modell auf und zeigt das tatsächliche Layout der Datenbank.

Nun werden Tabellen und Spalten mit Primär- und Fremdschlüssel erstellt nach den Vorgaben des logischen Modells. Im Gegensatz zum logischen Modell ist das physische Modell abhängig von der verwendeten Datenbank-Software.

## 9.7 OLAP-Modelle in multidimensionalen Datenbanken

Wie bereits in Kap. 1.5 besprochen wurde, wird OLAP (Online Analytical Processing) für die Analyse von Daten verwendet. Es wird zwischen vier Varianten des OLAP unterschieden:

**MOLAP:** Datenanalysen in multidimensionalen Systemen. MOLAP speichert Zahlen in Form von **Datenpunkten** und hat deshalb einen Performance-Vorteil gegenüber anderen OLAP-Systemen, die Daten als Datensätze speichern.

**ROLAP:** Datenanalysen in relationalen Systemen. Das System zieht Daten direkt aus bestehenden Datenbanken und benötigt so wesentlich weniger Speicherplatz.

**HOLAP:** Datenanalysen in hybriden Systemen. Hybrid für sowohl relational wie auch multidimensional.

**DOLAP:** Datenanalysen in Desktop Systemen. Bei diesen Systemen nimmt nicht die tatsächliche Analyse die meiste Zeit in Anspruch, sondern das Erstellen und Aufrfrischen der dargestellten Daten-Würfel auf dem Desktop.

	<b>Advantages</b>	<b>Disadvantages</b>
<b>MOLAP</b>	<ul style="list-style-type: none"> <li>Excellent performance: MOLAP cubes are built for fast data retrieval, and are optimal for slicing and dicing operations.</li> <li>Can perform complex calculations: All calculations have been pre-generated when the cube is created. Hence, complex calculations are not only doable, but they return quickly.</li> </ul>	<ul style="list-style-type: none"> <li>Limited in the amount of data it can handle: Because all calculations are performed when the cube is built, it is not possible to include a large amount of data in the cube itself. This is not to say that the data in the cube cannot be derived from a large amount of data. Indeed, this is possible. But in this case, only summary-level information will be included in the cube itself.</li> <li>Requires additional investment: Cube technology are often proprietary and do not already exist in the organization. Therefore, to adopt MOLAP technology, chances are additional investments in human and capital resources are needed.</li> </ul>
<b>ROLAP</b>	<ul style="list-style-type: none"> <li>Can handle large amounts of data: The data size limitation of ROLAP technology is the limitation on data size of the underlying relational database. In other words, ROLAP itself places no limitation on data amount.</li> <li>Can leverage functionalities inherent in the relational database: Often, relational database already comes with a host of functionalities. ROLAP technologies, since they sit on top of the relational database, can therefore leverage these functionalities.</li> </ul>	<ul style="list-style-type: none"> <li>Performance can be slow: Because each ROLAP report is essentially a SQL query (or multiple SQL queries) in the relational database, the query time can be long if the underlying data size is large.</li> <li>Limited by SQL functionalities: Because ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs (for example, it is difficult to perform complex calculations using SQL), ROLAP technologies are therefore traditionally limited by what SQL can do. ROLAP vendors have mitigated this risk by building into the tool out-of-the-box complex functions as well as the ability to allow users to define their own functions.</li> </ul>
<b>HOLAP</b>	HOLAP technologies attempt to combine the advantages of MOLAP and ROLAP. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.	

Abb. 9.5: Vor- und Nachteile der verschiedenen OLAP-Systemen

	<b>MOLAP</b>	<b>HOLAP</b>	<b>ROLAP</b>
<b>Technologie</b>	<b>Multidimensional</b>	<b>Gemischt</b>	<b>Relational</b>
<b>Datenmenge</b>	<b>Für kleinere bis mittlere Datenmengen</b>	<b>Für große Datenmengen</b>	<b>Für große Datenmengen</b>
<b>Datenzugriff</b>	<b>Nur OLAP-Server</b>	<b>OLAP-Server und Datenbankserver</b>	<b>Daten vom Datenbankserver</b>
<b>Speicherbedarf</b>	<b>Hoch</b>	<b>Normal</b>	<b>Normal</b>
<b>Antwortzeit</b>	<b>Schnelle</b>	<b>Variert</b>	<b>Etwas langsamer bei Aggregationen</b>
<b>Aufbereitung</b>	<b>Langsam</b>	<b>Nur Aggregationen</b>	<b>Nur Aggregationen</b>

Abb. 9.6: Vergleich der verschiedenen OLAP-Systeme

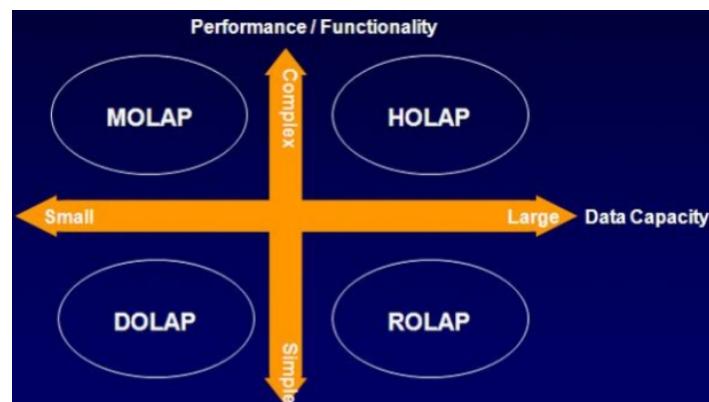


Abb. 9.7: Abwägung der Performance vs. Funktionalität für die verschiedenen OLAP-Systeme

## 9.8 CUBE, ROLLUP und GROUPING SETS (Skript S177)

GROUP BY [...] WITH CUBE oder auch direkt GROUP BY CUBE liefern dasselbe Ergebnis wie UNION ALL, ist aber vor allem wenn UNION ALL in Kombination mit GROUP BY oder ORDER BY verwendet wird einfacher zu verstehen.

### ROLLUP

ROLLUP ist eine Erweiterung der GROUP BY 'Methode'. Es ermöglicht es, zusätzliche Zeilen einzufügen, die das Subtotal einer Query anzeigen.

---

```
SELECT warehouse, product, SUM(quantity)
FROM Inventory
GROUP BY ROLLUP(warehouse, product)
```

---

Listing 1: Beispiel der Nutzung von ROLLUP

	warehouse	product	model	quantity
▶	San Fransisco	iPhone	6s	50
	San Fransisco	iPhone	7	10
	San Fransisco	iPhone	X	200
	San Fransisco	Samsung	Galaxy S	200
	San Fransisco	Samsung	Note 8	100
	San Jose	iPhone	6s	100
	San Jose	iPhone	7	50
	San Jose	iPhone	X	150
	San Jose	Samsung	Galaxy S	200
	San Jose	Samsung	Note 8	150

(a) Die Tabelle 'warehouse'

	warehouse	product	SUM(quantity)
▶	San Fransisco	iPhone	260
	San Fransisco	Samsung	300
	San Fransisco	(null)	560
	San Jose	iPhone	300
	San Jose	Samsung	350
	San Jose	(null)	650
	(null)	(null)	1210

(b) Die Summe der einzelnen Produkte und in allen Warehouses

Abb. 9.8: Beispiel der Anwendung von ROLLUP

### CUBE

Ähnlich wie ROLLUP berechnet auch CUBE Subtotale von Queries. Allerdings generiert CUBE ein Subtotal von allen möglichen Kombinationen der spezifizierten GROUP BY Spalten

---

```
SELECT warehouse, product, SUM(quantity)
FROM Inventory
GROUP BY CUBE(warehouse, product)
ORDER BY warehouse, product
```

---

Listing 2: Beispiel der Nutzung von CUBE

WAREHOUSE	PRODUCT	SUM(QUANTITY)
San Fransisco	Samsung	300
San Fransisco	iPhone	260
San Fransisco	(null)	560
San Jose	Samsung	350
San Jose	iPhone	300
San Jose	(null)	650
(null)	Samsung	650
(null)	iPhone	560
(null)	(null)	1210

Summe aller Produkte im Warehouse San Fransisco

Summe aller Produkte im Warehouse San Jose

Summe der jeweiligen Produkte über alle Warehouses verteilt

Abb. 9.9: Beispiel der Anwendung von CUBE

## GROUPING SETS

GROUPING SETS erlauben es einem, Daten gleichzeitig nach verschiedenen Gruppen zu gruppieren. Im untenstehenden Beispiel werden die gleichen Daten sowohl nach `Color`, wie auch nach `Item` gruppiert.

---

```
SELECT Item, Color, SUM(Quantity) AS QtySum
FROM Inventory
GROUP BY GROUPING SETS(Item, Color)
```

---

Listing 3: Beispiel der Nutzung von GROUPING SETS

Item	Color	Quantity
NULL	Blau	225
NULL	Rot	443
Stuhl	NULL	311
Tisch	NULL	347

Abb. 9.10: Beispiel der Anwendung von GROUPING SETS

## 10 MDX (Multidimensional Expressions) (Skript S196)

MDX steht für multidimensional expressions. Es ist eine von Microsoft entwickelter Standard und ist eine Query-Sprache, die auf SQL aufbaut und äusserst komplex aber auch extrem mächtig ist.

Das Ziel von MDX ist es folgende Dinge zu definieren oder weiterzuentwickeln:

- Eine standardisierte Query-Sprache mit eindeutiger Semantik
- Standardisierte Zugriffsprotokolle
- Abfrageoptimierungen
- Zugriffsstrukturen für die Datenverwaltung
- Standardisierte Datenaustauschformate

In MDX werden sehr viele Klammern gebraucht. Deren Bedeutung:

**Geschweifte Klammern :** Eine Ansammlung von Tupel derselben Dimension als solche zu kennzeichnen

**Runde Klammern ():** Definition der Tupel in der WHERE Bedingung

**Eckige Klammern [ : ]** Definiton der Namen von Dimensionen, Leveln und deren Membern, sowie auch Namen von Cubes.

### 10.1 MDX SELECT

Der MDX SELECT Befehl ist grundsätzlich dasselbe wie in SQL: Er ermöglicht das Lesen und Verbinden von Daten aus Daten-Würfeln.

Der SELECT Befehl benötigt mindestens die Anzahl der Achsen, die das Resultat enthalten soll (**AXIS**). Es können maximal 128 Achsen angegeben werden, wobei jedoch maximal zwei Achsen angezeigt werden können. Dabei handelt es sich bei Achse 0 um Spalten und bei Achse 1 um Zeilen. Ebenfalls muss die Anzahl Elemente auf jeder Achse angegeben werden (**MEMBERS**). Schlussendlich muss noch der Datenwürfel mittels CUBE angegeben werden.

---

```
SELECT Ausdruck1 ON COLUMNS, Ausdruck2 ON ROWS, Ausdruck3 ON AXIS(2), Ausdruck4 ON
AXIS(3)
FROMN CubeName
```

---

Listing 4: Allgemeiner Syntax von MDX' SELECT

---

```
SELECT {[Mitarbeiter].[Mitarbeiter].MEMBERS} ON AXIS(0) --kann alternativ auch als
'0' oder 'COLUMNS' geschrieben werden.
{[Produkt].[Artikel].MEMBERS ON AXIS(1) --kann alternativ auch als '1' oder
'ROWS' geschrieben werden.
FROM [DW1fach]
```

---

Listing 5: Beispiel MDX' SELECT

Das Beispiel von Listing 5 beschreibt eine Abfrage, bei der alle Mitarbeiter in den Spalten stehen und alle Artikel in den Zeilen.

	All	Bohlmann, Alfred	Hahn, Gerhard	Hasköy, Nuket	Jarowski, Eva	Karls, Carla	Ladegast, Jörg	Meier-Kunze, Ingrid	Selig, Ida	Sincero, Maria
A.O.C. Côtes du Rhône	51317	7837	7836	3036	3527	5913	4654	2661	9798	6055
Aarhäuser Klosterberg Kabinett	623	95	103	40	(NULL)	(NULL)	99	57	130	99
Aarhäuser Klosterberg Rülander	1083	221	213	20	58	174	110	(NULL)	127	160
After Eight Liquid	799	149	105	40	(NULL)	148	197	(NULL)	135	25
Alnberger Spätzleberg	500	59	173	(NULL)	15	42	40	44	23	104
Alrkönig	981	230	185	40	55	207	34	73	87	70
Alsaace Pinot Noir	1155	218	175	28	129	188	36	110	181	90
Alsaace Pinot Noir	1057	303	160	76	81	68	85	58	202	24
Altänder Apfelbrand	293	14	50	(NULL)	35	104	(NULL)	(NULL)	90	(NULL)
Amselfelder Weiß	1016	72	120	41	87	118	152	43	313	70
Apfel mit Schuss	297	16	34	21	50	20	(NULL)	36	65	55
Aquavat Linie	508	167	24	130	(NULL)	(NULL)	15	40	76	56
Armagnac Exquiseé	612	86	74	70	(NULL)	100	30	95	92	65
Armagnac Supérieur	805	201	171	20	45	41	35	67	105	120
Aua gutem Schrot und Korn 32%	372	58	(NULL)	(NULL)	60	10	70	104	70	
Aua gutem Schrot und Korn 38%	742	265	184	26	52	51	16	18	54	76
Australie Top 1 Redwine	580	160	163	(NULL)	(NULL)	66	45	(NULL)	60	86
Bourbon Whisky 8 Years	791	134	17	50	63	94	62	53	245	73
California Old Oak Creek	184	(NULL)	15	(NULL)	(NULL)	20	27	(NULL)	52	70
California Val Verde	1577	285	364	174	25	96	72	44	331	186
Chartreuse verte	793	127	109	108	151	85	3	10	120	80
Chile Cabernet Sauvignon	755	44	149	74	62	30	38	15	231	112
Chile Rotwein konkloer Weingenuss	297	53	44	(NULL)	(NULL)	65	76	4	(NULL)	55
Cognac Exquiseé	539	87	75	(NULL)	(NULL)	58	8	52	151	108

Abb. 10.1: Visualisierung des Codebeispiels von Listing 5

## 10.2 DAX (Data Analysis Expression)

DAX ist die Sprache hinter PowerPivot, Power BI Desktop und SQL Server Analysis Services. Sie verwendet einige der typischen Excel-Funktionen, besitzt jedoch noch einige weitere Funktionen und Features, die mit relationalen Daten arbeiten und dynamische Aggregationen durchführen können. Es wird als Evolution der MDX mit einer Prise von Excel-Funktionen angeschaut, die einfach zu lernen ist und gleichzeitig die Flexibilität und Macht von PowerPivot bietet.

# 11 Planung der Systemtechnischen Architektur (Skript S205)

## 11.1 Realtime Data-Warehouse Systeme (Skript S206)

Realtime Data-Warehouse Systeme sind Data-Warehouses, die Daten in Realtime zur Verfügung stellen können. Jedoch muss man sich jeweils genau überlegen, ob eine solche "Data Freshness" überhaupt Sinn macht für die zu unterstützenden Geschäftsprozesse.

Ein klassisches Data-Warehouse durchläuft in regelmässigen Zyklen von einem Tag bis zu einem Monat den Datenbeschaffungsprozess (Kap. 4.1.8). Es kann jedoch durchaus Prozesse geben, die täglich oder sogar im Minutentakt aktualisierte Daten benötigen.

Der Datenbeschaffungsprozess ist meist sehr rechenintensiv und wird deshalb in Off-Peak Zeiten (z.B. über Nacht) durchgeführt.

## 11.2 In-Memory Computing (Skript S207/Buch S174)

Realtime Data-Warehouses benötigen In-Memory Computing für die schnellere Verarbeitung von Daten. In-Memory Computing bedeutet, dass die Daten direkt im RAM gespeichert und verarbeitet werden. Aufgrund der wesentlich tieferen Zugriffszeiten und schnelleren Lese/Schreibe-Geschwindigkeit im Gegensatz zu normalen Festplatten kann die Datenverwaltung und -analyse so erheblich verschnellert werden.

RAM ist per Definition volatil. Da man logischerweise nicht will, dass die gesamten Daten des Data-Warehouses nach einem allfälligen Neustart des Servers plötzlich weg sind, werden normale Festplatten trotzdem noch verwendet, jedoch nur zu Backup-Zwecken (Snapshots, Mirroring oder auch Transaction-Logs).

### 11.2.1 RAM-Management

In In-Memory Data Warehouses wird die sogenannte NUMA-Architektur eingesetzt (Abb. 11.1).

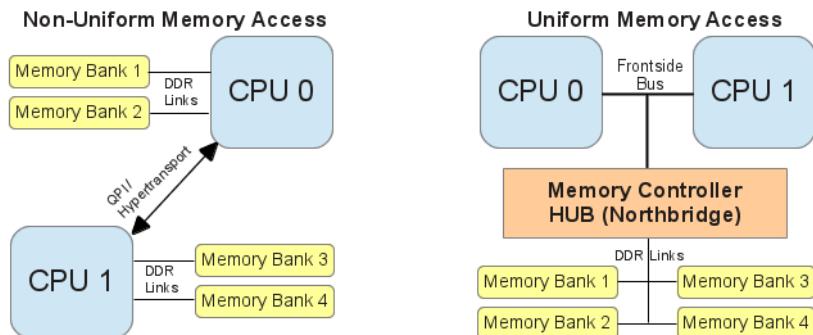


Abb. 11.1: Vergleich der NUMA und UMA-Architektur

Im Gegensatz zu UMA (Uniform Memory Access) können hier nicht alle CPUs auf alle RAM-Bänke zugreifen, sondern jede CPU hat "ihre" RAM. Jedoch wissen alle CPUs auf welchen Bänken, bzw. bei welcher CPU sich die benötigte Information befindet. So findet eine rege Kommunikation zwischen den einzelnen CPUs statt. Die Verwaltung, welche Informationen bei welcher CPU abgespeichert werden, wird vom Betriebssystem übernommen.

Da das Betriebssystem aber meist keine Ahnung des Datenbankschemas und der verschiedenen Abhängigkeiten unter den jeweiligen Tabellen hat, kann es die Informationen gar nicht am effizientesten verteilen. Dafür hat die SAP eigens ein Tool entwickelt: SAP HANA. Beim Starten der Datenbank legt HANA seinen eigenen Hauptspeicher an (Abb. 11.2). Dort wird der Programmcode für die Datenbanktabellen, die temporären Berechnungen und der Speicher für interne Prozesse gehandhabt. Wird in diesem Hauptspeicher wieder Platz frei, gibt HANA diesen Speicher nicht mehr ans Memory zurück, sondern behält es für allfällige spätere Nutzen noch bei sich. So muss es das Betriebssystem nicht mehr nach mehr Memory fragen, sondern kann diesen freien Speicher direkt wieder einem der in Abb. 11.2 dargestellten Bereiche zuordnen.

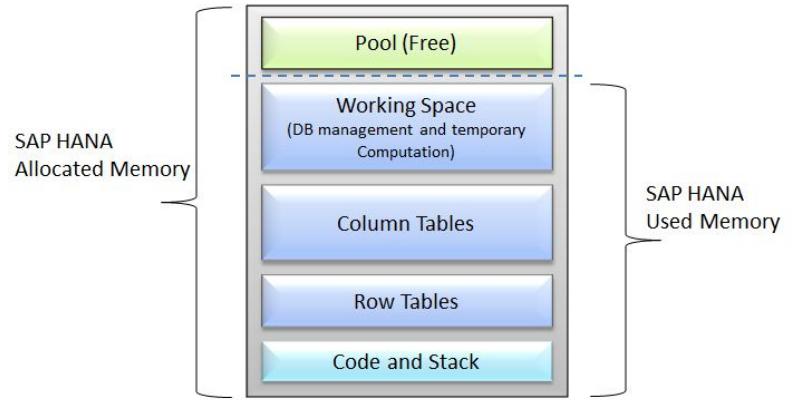


Abb. 11.2: Speicherverwaltung von SAP-HANA

### 11.2.2 Zeilenorientiertes vs. Spaltenorientiertes Speichern

Lager	Produkt	Umsatz
Deutschland	A	1.000
Deutschland	B	1.500
Frankreich	A	1.100

Zeilenorientiert:		Spaltenorientiert:	
Zeile 1	Deutschland	Spalte 1	Deutschland
	A		Deutschland
	1.000		Frankreich
Zeile 2	Deutschland	Spalte 2	A
	B		B
	1.500		A
Zeile 3	Frankreich	Spalte 3	1.000
	A		1.500
	1.100		1.100

Abb. 11.3: Klassisch- vs. Zeilen- vs. Spaltenbasierte Datenspeicherung

so viele Daten schreiben, von da her kann die vergleichsweise langsame Schreibgeschwindigkeit vernachlässigt werden.

Im Gegensatz zu klassischen, relationalen Datenbanken (Abb. 11.3 oben) abspeichern, werden Daten in In-Memory Datenbanken zeilen- oder spaltenbasiert (Abb. 11.3 links, bzw. rechts) abgespeichert. Die Speicherungsart der Daten hängt davon ab, ob die Datenbank für OLAP oder OLTP verwendet wird.

Bei OLTP-Systemen werden die Daten meist zeilenbasiert abgespeichert, da dies schnellen Schreibzugriff ermöglicht. Der Nachteil dabei ist der vergleichsweise langsame Lesezugriff, was bei OLTP-Systemen jedoch zu verkraften ist, da solche Systeme vor allem schreibheavy Operationen durchführen.

Bei OLAP-Systemen ist es genau umgekehrt. Da man auf schnellen Lesezugriff angewiesen ist, werden die Daten meist spaltenorientiert abgespeichert. OLAP-Systeme müssen nicht

## Vergleich Zeilen- und Spaltenorientierte Abfragen

Welche Speicherart effizienter ist, hängt von der Art der Abfrage ab.

Nehmen wir ein Beispiel, bei dem die spaltenorientierte Abspeicherung im Vorteil ist:

```
SELECT SUM(SALES)
FROM SALES
WHERE DATA > 2012-01-01
```

Wie in Abbildung 11.4 zu sehen ist, muss hier nur auf zwei Spalten zugegriffen werden: Man checkt in der Date Spalte, ob das Datum  $\geq$  2012-01-01 ist und wenn ja, dann addiert man die korrespondierende Summe zum Resultat.

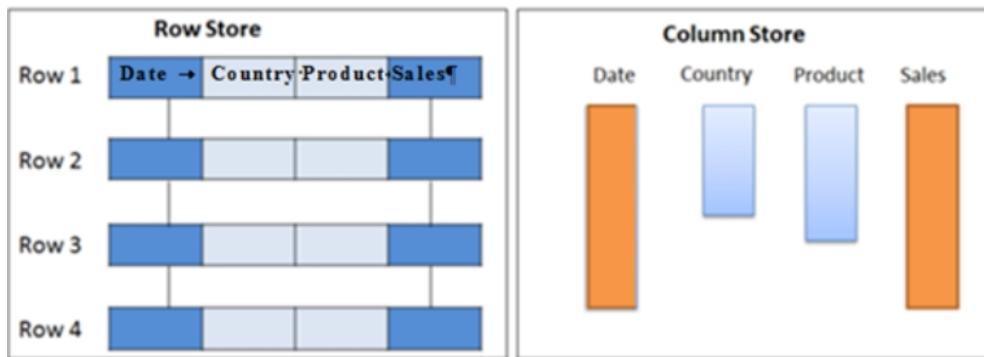


Abb. 11.4: Beispiel einer spaltenorientierten Abfrage

Nehmen wir nun ein anderes Beispiel wie folgendes:

```
SELECT *
FROM SALES
WHERE COUNTRY = 'INDIA'
```

so ist die zeilenbasierte Speicherart eindeutig effizienter, wie Abbildung 11.5 zeigt. Während eine spaltenbasierte Datenbank hier bei jeder Spalte überprüfen müsste, ob die korrespondierende Country-Spalte 'India' enthält, muss die zeilenbasierte Datenbank nur in einer Spalte 'India' suchen und dann dieser Zeile 'entlangfahren'



Abb. 11.5: Beispiel einer zeilenorientierten Abfrage

Zusammengefasst kann man sagen: Wenn die Abfrage wenige Spalten aber viele Zeilen beinhaltet, ist die spaltenbasierte Speicherart effizienter; bei vielen Spalten und wenigen Zeilen hat die zeilenbasierte Speicherung die Oberhand.

### 11.2.3 Chancen und Risiken des In-Memory Computing

Sicht	Chancen	Risiken
<b>Business</b>	<ul style="list-style-type: none"> <li>Analysen in Echtzeit</li> <li>Neue Analyse Möglichkeiten</li> <li>Beschleunigen von Geschäftsprozessen</li> <li>Wettbewerbsvorteil</li> <li>Bessere Entscheidungsmöglichkeiten</li> </ul>	<ul style="list-style-type: none"> <li>Erhöhte Ausfallzeit vor allem zu Beginn</li> <li>Erhöhtes Risiko von Datenverlust</li> <li>Einführung bringt nicht die erhoffte Performance Verbesserung</li> <li>Vorhandene Geschäftsprozesse müssen angepasst werden</li> </ul>
<b>IT</b>	<ul style="list-style-type: none"> <li>Bereitstellung von neuen IT-Lösungen für das Business</li> <li>Hohe Zufriedenheit der Fachabteilung mit der IT</li> <li>Optimale CPU Ausnutzung</li> <li>Schlankere IT-Umgebung</li> <li>Reduzierte IT-Kosten (variable)</li> </ul>	<ul style="list-style-type: none"> <li>Erwartungen werden nicht erfüllt</li> <li>Fehlendes Know-how</li> <li>Hohe Einführungskosten durch neue HW &amp; Lizizenzen</li> <li>Applikationen müssen angepasst werden</li> <li>Kein umfangreiches Monitoring</li> <li>Versuchsobjekt von DB-Anbietern</li> </ul>

### 11.3 Cloud-basiertes Data-Warehouse (Skript S218)

Heutzutage ist alles mögliche und unmögliche in der Cloud. Wieso nicht auch das Data-Warehouse? Wieso mühselig und kostenintensiv ein eigenes Data-Warehouse aufbauen, wenn Grossanbieter wie Oracle und SAP dies bereits als SAAS anbieten? Es hat einige Vorteile, wie

- Keine Wartungs- und Upgrade-Kosten
- Kein Zeit- und Ressourcenaufwand für die Verwaltung der riesigen Datenmengen
- Keine Kosten für den internen Betrieb
- Dynamisch skalierbar

### 11.3.1 Dictionary Daten-Komprimierung

Wie am Beispiel der Abbildung 11.6 zu sehen ist, funktioniert die Dictionary-Compression so, dass es ein Dictionary gibt, in welchem alle Daten abgespeichert sind. Die Spalten halten schlussendlich keine Daten mehr, sondern nur noch Pointer zum Ort der Daten im Dictionary. So können Redundanzen (wie z.B. das Doppelte Speichern des Namens "John") vermieden werden.

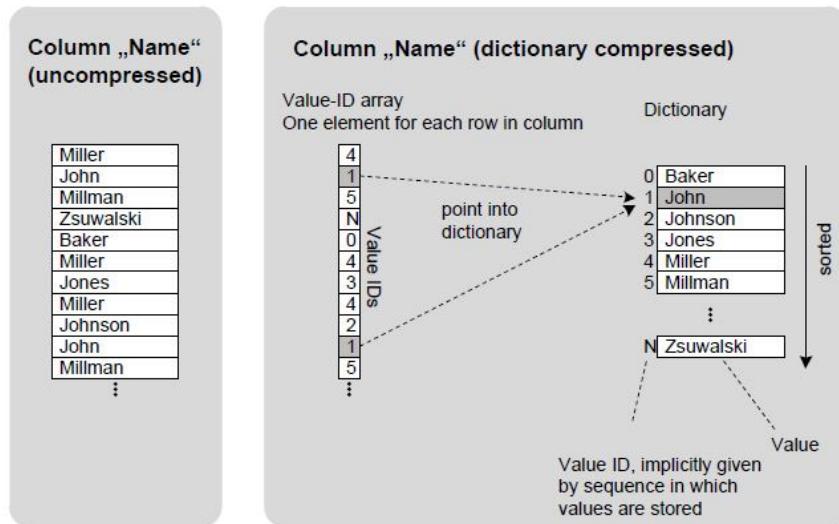


Abb. 11.6: Beispiel der Dictionary Compression

## 12 Hadoop (Skript S229)

Apache Hadoop ist ein in Java geschriebenes Framework für skalierbare, verteilte arbeitende Software. Es basiert auf dem MapReduce-Algorithmus von Google sowie auf Vorschlägen des Google-Dateisystems und ermöglicht es, intensive Rechenoperationen mit grossen Datenmengen im Petabyte-Bereich auf Computerclustern durchzuführen.

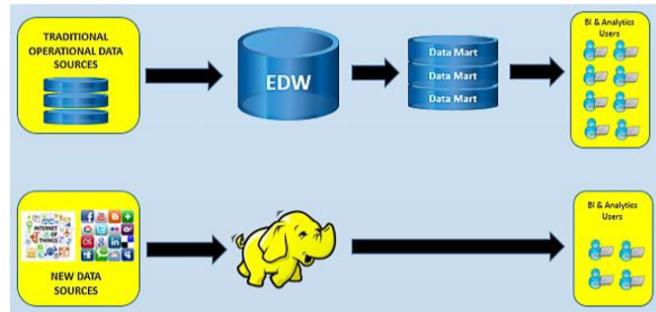
Hadoop wurde von Apache entwickelt und ist open source.

Die bekanntesten Anwender von Hadoop sind wohl Facebook IBM, AOL und Yahoo

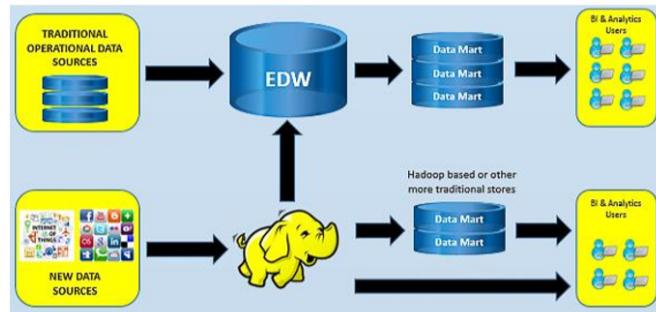
### 12.1 Anwendungen

Hadoop kann auf drei Arten verwendet werden.

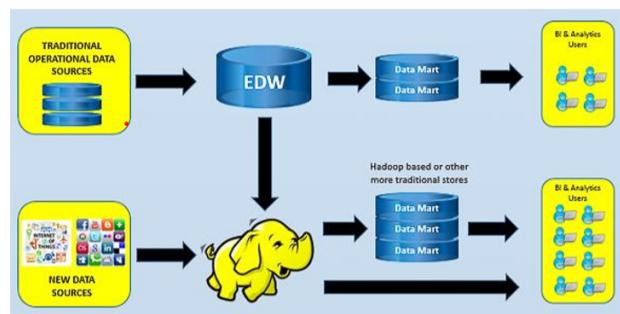
1. Als neuer Datenspeicher



2. Als Datenplattform und zusätzlicher Input für das Enterprise Data Warehouse



3. Als Daten Plattform und Basis für BI und Analytics



## 12.2 Motivation

- Automatische Generierung von Daten führt zu riesigen Datenmengen
- Kosten pro GB sind in den letzten Jahren extrem gesunken
- Währenddessen haben sich die Transferzeiten nicht gross verändert
  - Die Zeit, um eine komplette Festplatte auszulesen wird somit immer länger
- Mehrere Server mit eigenen Festplatten werden zu Cluster verbunden
- Durch die verteilte Speicherung ist parallele Verarbeitung möglich.

## 12.3 Möglichkeiten

Hadoop kann intensive Rechenprozesse parallel in Clustern durchführen. Somit können grosse Datenmengen auf mehrere tausend Server verteilt werden.

Für die parallele Verarbeitung mehrerer Tasks wird Nebenläufigkeit verwendet. Man verwendet bei schwächerer Hardware scheinbare Nebenläufigkeit handelt (präemptives Multitasking → Scheduler weist jedem Task gewisse CPU-Zeit zu) oder wenn mehrere Cores oder CPUs zur Verfügung stehen, kann echte Nebenläufigkeit durchgeführt werden.

Hadoop wird als Schnittstelle zwischen Real Time Data und Data Warehouses verwendet.

## 12.4 Aufbau

Hadoop verwendet das HDFS (Hadoop Distributed File System) um Daten zu speichern und den MapReduce Algorithmus und diese zu verarbeiten. Weitere Bausteine, auf welche hier nicht weiter eingegangen wird, können im Skript auf S232 nachgesehen werden.

## 12.5 Architektur

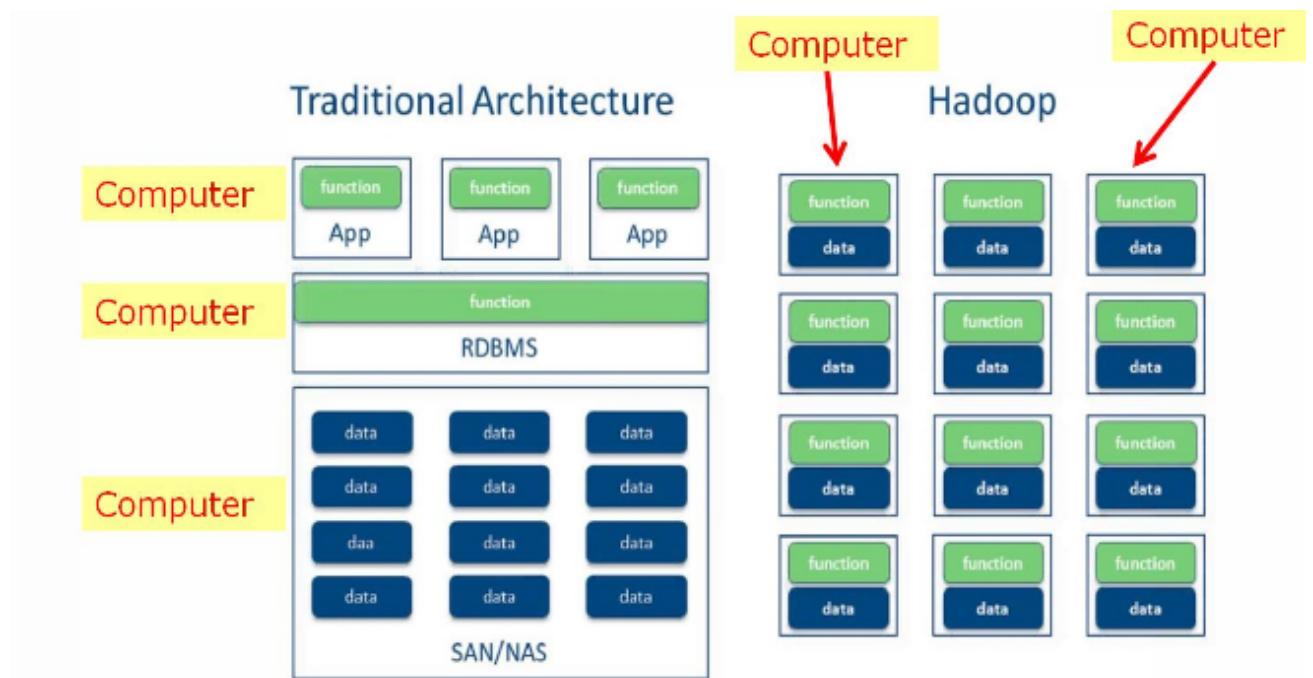


Abb. 12.1: Vergleich einer klassischen RDBMS-Architektur im Vergleich zu Hadoop

**Master:** Verwaltet die Anfragen vom Client. Der Master enthält zusätzlich zu den Slaves eine NameNode sowie einen JobTracker

**Client:** Der Client schickt Anfragen zum Speichern und Verarbeiten von Daten. Der Client kann ein Software-UI oder auch ein CMD-Tool sein

**Cluster/Rechnerverband:**

Ansammlung mehrerer Computer/Server (Nodes). Durch Cluster wird erstens die Rechenpower und zweitens die Redundanz und somit Verfügbarkeit erhöht.

**Node:** Rechner in einem Cluster, jede Node enthält ein Task Tracker und eine Data Node

**HDFS:** Hadoop Distributed File System. Hadoops Dateisystem

**Name Node:** Managt das Dateisystem, die Indexierung und die Metadaten. Es gibt eine Active und eine Standby Name-Node, die im Falle des Ausfalls der Active NN einspringen könnte

**Data Node.** Verwaltet die Daten und liefert sie den Clients. Die Name Node kommuniziert mit den Data Nodes

**Job Tracker Node:** Erhält Anfragen vom Client und plant und überwacht das MapReducing. Nur 1 Job Tracker pro Cluster

**Task Tracker:** Führt alle MapReduce Transaktionen aus, Mehere Task Tracker pro Cluster

## 12.6 MapReduce

Wie bereits erwähnt, verwendet Hadoop den MapReduce Algorithmus zur Verarbeitung von grossen Datenmengen. Das Ziel dieses Algorithmus ist es, ungeordnete Datenmengen in Key-Value Paare umzuwandeln:

**Map:** Generiert für jeden Datensatz ein Key-Value Paar

[Store, SalesAmount]

[New York, \$450]

**Reduce:** Summiert dieselben Key-Value Paare

[New York, \$450]  
 [New York, \$500]  
 [San Jose, \$444]  
 [San Jose, \$600]

[New York, \$950]  
 [San Jose, \$1044]

Der MapReduce Algorithmus läuft folgendermassen ab:

1. Verteile die Inputdaten (wenn möglich fair) auf die verfügbaren Nodes auf
2. Jede Node führt nun ein Map-Durchgang mit den ihr zugewiesenen Daten durch
3. Die entstandenen Key-Value Paare werden so gemischt, dass alle gleiche Keys bei derselben Node zu liegen kommen
4. Jede Node führt ein Reduce auf die ihr zugewiesenen Daten und Keys aus.

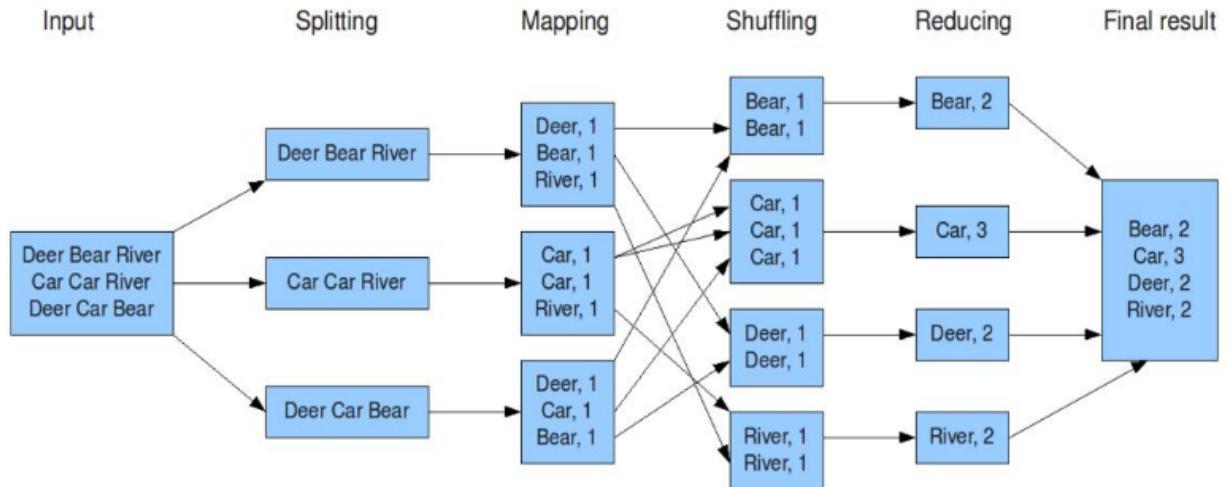


Abb. 12.2: MapReduce verwandelt ungeordnete Daten in Key-Value Paare

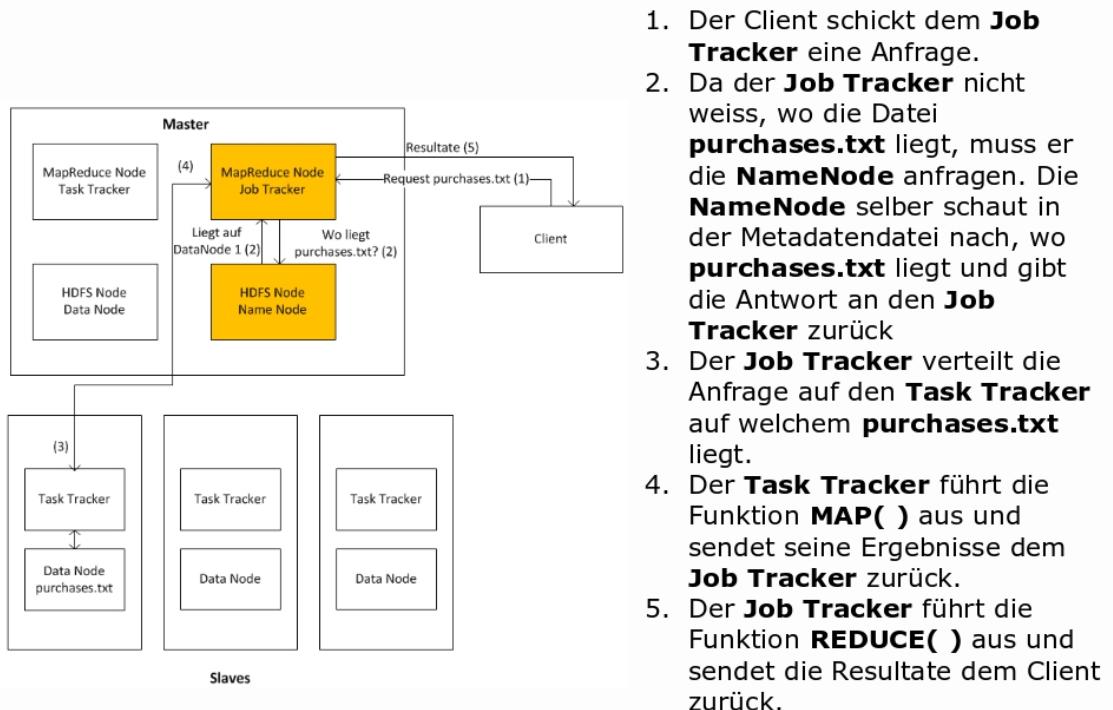
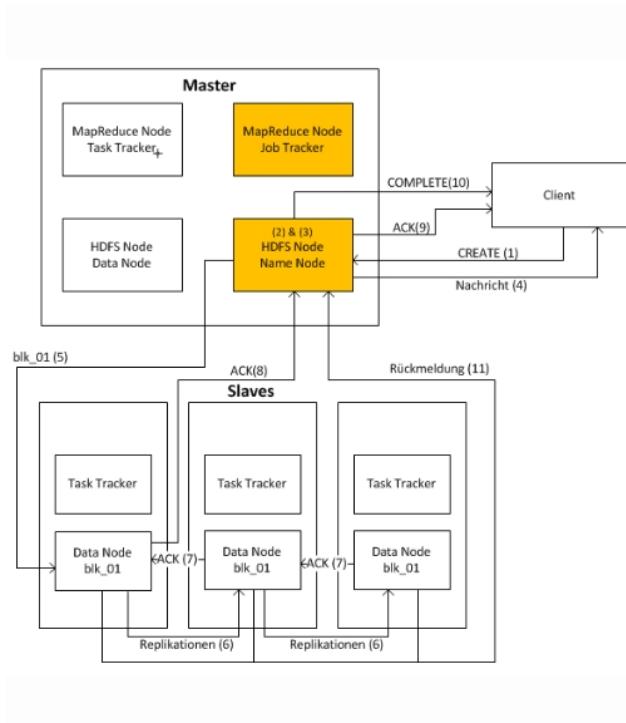


Abb. 12.3: Ablauf eines Map-Reduce Aufrufes

## 12.7 Speichern mit HDFS



- Der **HDFS Client** sendet eine **Create** Anfrage an die **NameNode**.
- Der NameNode prüft ob die Datei vorhanden ist.
- Die **NameNode** erstellt eine **Metadaten Datei** von **purchases.txt** (Name, Grösse).
- Die **NameNode** sendet dem **HDFS Client** eine Nachricht. Diese enthält die Nummern der Slaves, auf welche die Blöcke gespeichert werden.
- Der erste Block (blk\_1) wird in eine **DataNode** eines Slaves geschrieben.
- Der Block (blk\_1) wird zwei Mal in eine andere **DataNode** im Cluster dupliziert (Default Wert von Hadoop = 2 Replikationen).
- Damit die **NameNode** weiss, dass die Blöcke dupliziert wurden, schickt jede Node ein Acknowledged Paket (ACK) zur vorherigen Node.
- Die erste Node sendet ein **ACK** an die **NameNode**.
- Die NameNode sendet dem Client ein **ACK** Paket. Der Client weiss nun, dass er die restlichen Blöcke verarbeiten kann.
- Sobald alle Blöcke in der **DataNode** gespeichert wurden, meldet der **NameNode** dem Client den Status **Complete**.
- Die verschiedenen **DataNodes** senden nun der **NameNode** einen **Block Report**. Dieser enthält die **Blocknamen** und die **Speicherorte**, welche in die **Metadatendatei** geschrieben werden.

Abb. 12.4: Speichervorgang unter HDFS

## 12.8 Anwendungen

Mehr Anwendungen von Hadoop können auf Seite 255 im Skript nachgelesen werden.

## 13 Speicherarten (Skript S256)

**Klassische Datenhaltung:** Räumlich konzentriert halten und zentral/dezentral zur Verfügung stellen

**Daten-Partitionierung:** Räumlich verteilt halten und zentral/dezentral zur Verfügung stellen

**RAID** steht für *Redundant Array of Independent Disks* und erlaubt die redundante Absicherung von Daten in einem Festplatten-Verbund, so dass keine Daten verloren gehen im Falle eines Festplatten-Defekt

**Dateidienste** verwalten die Datenbestände mit einem eigenen Dienst-Betriebssystem, welches auch für Zugriffsicherung und ggf. Verschlüsselung zuständig ist.

**NAS-Systeme:** *Network Attached Storage* sind über Netzwerk zugreifbar und sind auf Firmware basierte Dateidienste, die vorwiegend in kleinen Umgebungen zum Einsatz kommen

**SAN-Systeme:** *Storage Area Network* verteilen die Datenbestände über ein eigenes Storage-Netzwerk

**Spechervirtualisierung:** Ist eine Speicherart, bei der direkt adressierbare Daten-Volumen auf mehrere Festplatten oder SAN-Nodes verteilt werden (z.B. LVM bei Linux). Spechervirtualisierung wird von NAS und SAN unterstützt.

In Data Warehouses benötigt man extreme Mengen von temporärem Speicherplatz zum Zwischenspeichern von Datensätzen beim Sortieren, Joinen oder sonstiges Aggregieren und es werden viele Daten ausgelagert. Deshalb wird in solchen Umgebungen meist mit Spechervirtualisierung mittels SAN gearbeitet.

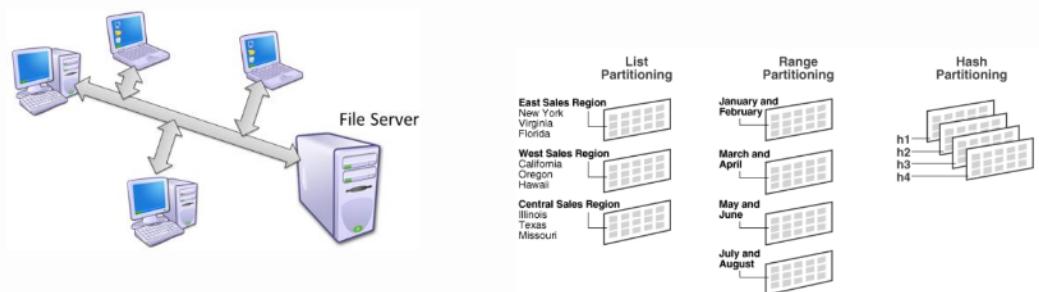


Abb. 13.1: Vergleich zwischen klassischer (links) und partitionierter (rechts) Datenhaltung

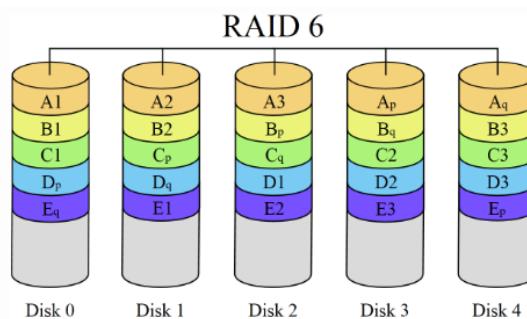


Abb. 13.2: Beispiel von RAID 6. Hier können 2 Festplatten defekt sein ohne dass ein Datenverlust auftritt

## 13.1 Objektspeicher (Skript S258)

### 13.1.1 Definiton

“Objektspeicherung ist ein allgemeiner Begriff für einen Ansatz, wie man eigenständige Einheiten an Storage manipuliert und adressiert, die man Objekte nennt.”

Objekte enthalten Daten. Jedoch werden Objekte nicht hierarchisch gespeichert (wie z.B. Dateien, die normalerweise in einem Baum abgespeichert werden (/home/user/Documents/-foo.txt)). Objekte sind flach organisiert. Jedes neue Objekt wird einfach in den Storage-Pool geschmissen. Ebenfalls sind Objekte atomar, es können also keine Objekte innerhalb von anderen Objekten abgelegt werden.

Ebenso wie herkömmliche Dateien haben Objekte Metadaten. Das Objekt wird jedoch mithilfe dieser Metadaten charakterisiert. Jedes Objekt erhält eine unique ID, die in den erweiterten Metadaten abgelegt wird.

Objektspeicherung wird meist mit dem Valet-Service in Restaurants verglichen: Der Besucher gibt sein Auto ab und erhält dafür einen Beleg. Der Valet parkt das Auto irgendwo, wo genau ist dem Besucher prinzipiell egal. Wenn der Besucher sein Auto wieder will, so kann er nur den Beleg abgeben und erhält so sein Auto wieder.

Bei Objektspeicher ist das sehr ähnlich. Der Besucher ist der Server oder der Enduser und das Auto sind Daten. Der Valet ist der Objektspeicher und der Beleg ist eine eindeutige ID des gespeicherten Objekts. Der Server übergibt dem Objektspeicher ein Objekt, das er speichern will. Er erhält dafür nun eine eindeutige ID zurück. Wenn er das Objekt wieder lesen will, muss er dem Server nur diese eindeutige ID mitgeben und erhält das abgespeicherte Objekt zurück.

### 13.1.2 Motivation

Es gibt mehrere Gründe, wieso ein Objektspeicher einem klassischen Dateisystem vorzuziehen ist.

- Objekte sind nicht hierarchisch organisiert. Der Storage Pool erlaubt die relativ einfache Speicherung von enormen Mengen von unstrukturierten Daten.
- Ein hierarchisches Dateisystem ist ungeeignet wenn eine Datei aus Milliarden von Dateien gelesen werden soll.
- Objektspeicher bestehen aus vergleichsweise günstigen Clustern
- Storage Pools sind praktisch grenzenlos skalierbar
- Storage Pools erlauben einen schnellen Lese- und Schreibzugriff, auch bei enormen Datensätzen
- Objektspeicherung erlaubt ein schnelles Indexieren, eine schnelle Suche und Langzeit-Archivierung
- Storage Pools sind Cloud-fähig

Aus diesen und anderen Gründen verwenden viele grosse Player wie Spotify, Facebook und Google Objektspeicher.

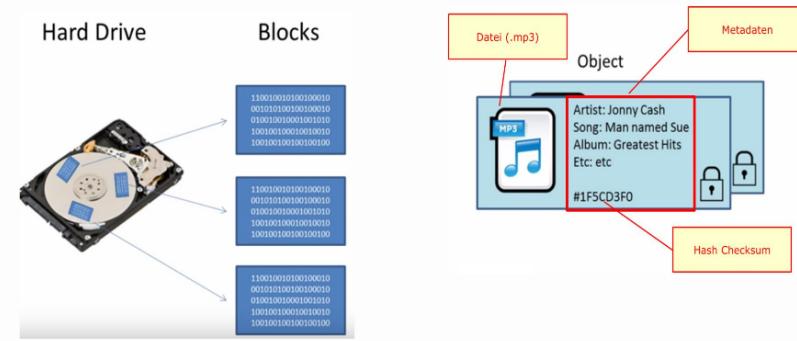


Abb. 13.3: Vergleich Dateisystem (links) und Objektspeicherung (rechts)

### 13.1.3 Arhitektur

Objektspeicherungs-Software besteht aus vielen lose gekoppelten Services und ist Hardwareunabhängig. Ausserdem wird solche Software in drei Layer aufgeteilt:

**Storage-Layer:** Schnittstelle zu den Data-Nodes

**Metadatenmanagement-Layer:** Bestimmt wo die Objekte gespeichert werden, wie sie auf die Nodes verteilt werden und wie sie geschützt werden.

**Präsentation-Layer:** Verwaltet die Schnittstelle zu den Clients sowohl über HTTP- wie auch über Dateisystem-Protokolle

Da Objektspeicher-Software meist über HTTP mit dem Client interagiert, wird auch hier die normale HTTP-API verwendet:

GET Objekt holen

PUT Objekt erstellen

COPY Objekt kopieren

DELETE Objekt löschen

HEAD Metadaten eines Objekt abrufen

POST Objekt-Metadaten erstellen oder bearbeiten

## 14 XMLA (Skript S268)

XMLA steht für *XML for Analysis* und ist eine API, die es Programmen erlaubt, mit OLAP- oder MDDB-Systemen zu kommunizieren. Die Kommunikation läuft über HTTP, SOAP und XML.

HTTP und XML sollten bereits bekannt sein. SOAP (Simple Object Access Protocol) ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können.

XMLA verwendet MDX (Siehe Kap. 10) als Abfragesprache. Die MDX-Statements werden in `<Statement>` Tags gehüllt. Zur Übertragung verwendet XMLA die SOAP-Methoden `Execute` und `Discover`

### 14.1 Execute

Das Rückgabewert einer `Execute` Methode ist ein Satz Tupel (entweder OLAP oder multidimensional). Die Methode verwendet das `<Command>` Tag, in welchem ein MDX, DMX oder auch SQL-Statement eingebettet werden kann, und das `<Properties>` Tag, in welchem gewissen Eigenschaften wie Timeout, Catalog name etc. mitgegeben werden können.

---

```
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
  <Command>
    <Statement>
      SELECT [Measures].MEMBERS ON COLUMNS FROM [Adventure Works]
    </Statement>
  </Command>
  <Properties>
    <PropertyList>
      <DataSourceInfo>
        Provider=MSOLAP;Data Source=local;
      </DataSourceInfo>
      <Catalog>
        Adventure Works DW Multidimensional 2012
      </Catalog>
      <Format>
        Multidimensional
      </Format>
      <AxisFormat>
        ClusterFormat
      </AxisFormat>
    </PropertyList>
  </Properties>
</Execute>
```

---

Listing 6: MDX-SELECT Statement mit XMLA übermittelt

## 14.2 Discover

Der Rückgabewert der `Discover` Methode ist ein Rowset voller Metadaten. Diese Methode macht Metadatenabfragen.

```
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis:rowset">
  <RequestType>MDSHEMA_CUBES</RequestType>
  <Restrictions>
    <RestrictionList>
      <CATALOG_NAME>
        Adventure Works DW Multidimensional 2012
      </CATALOG_NAME>
    </RestrictionList>
  </Restrictions>
  <Properties>
    <PropertyList>
      <DataSourceInfo>
        Provider=MSOLAP;
        Data Source=local;
      </DataSourceInfo>
      <Catalog>
        Adventure Works DW Multidimensional 2012
      </Catalog>
      <Format>
        Tabular
      </Format>
    </PropertyList>
  </Properties>
</Discover>
```

Listing 7: Anfrage für Cube-Listen aus der einem bestimmten Catalog mit XMLA übermittelt

Das Ganze wird nun über SOAP übermittelt, was schlussendlich so aussieht:

```
<soap:Envelope>
  <soap:Body>
    <Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
      <Command>
        <Statement>
          SELECT Measures.MEMBERS ON COLUMNS FROM Sales
        </Statement>
      </Command>
      <Properties>
        <PropertyList>
          <DataSourceInfo/>
          <Catalog>
            FoodMart
          </Catalog>
          <Format>
            Multidimensional
          </Format>
          <AxisFormat>
            TupleFormat
          </AxisFormat>
        </PropertyList>
      </Properties>
    </Execute>
  </soap:Body>
</soap:Envelope>
```

Listing 8: SELECT-Anfrage in XMLA verpackt, über SOAP versendet

# 15 Metadaten (Skript S271/Buch S339)

## 15.1 Definition

Als Metadaten bezeichnen wir jede Art von Informationen, die für den Entwurf, die Konstruktion und die Benutzung eines Informationssystems benötigt wird  
(Bauer/Günzel, 2009)

Auf Data-Warehousing angewendet heisst das soviel wie

“We think about metadata as all the information that defines and describes the structures, operations and contents of the DW/BI system” (Kimball, 2008)

Man kann also sagen, Metadaten beschreiben Daten/Datenstrukturen sowie manuelle und maschinelle Prozesse

## 15.2 Meta-Metamodell

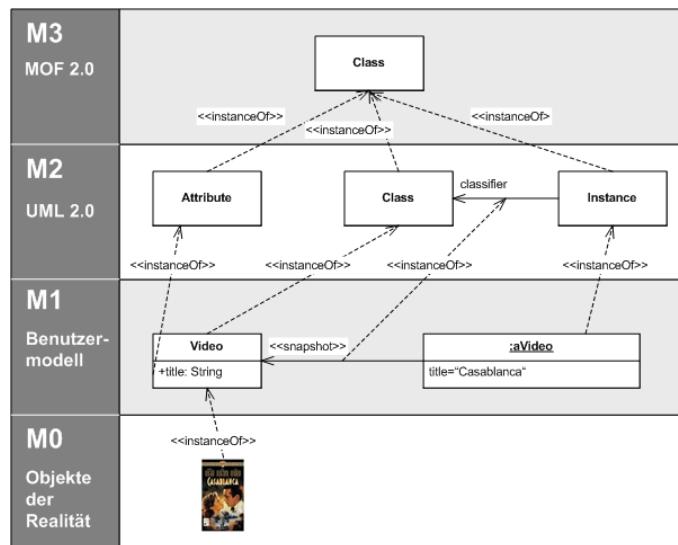


Abb. 15.1: MOFs 4-Ebenen Architektur am Beispiel von Casablanca

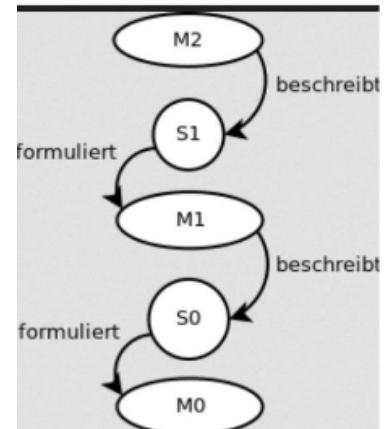


Abb. 15.2: Abstrahierte Version von 15.1

Die MOF (Meta Object Family) beschreibt eine 4-Schichten Metadaten-Architektur.

**M0 - Objekt** Konkrete, Real-Life Daten  
"Casablanca"

**M1 - Modell** Modelle, wie UML- oder Objektmodelle, die die Objekte aus M0 beschreiben  
"Casablanca ist ein Video mit dem Titel 'Casablanca'"

**M2 - Meta-Modell** Meta-Modelle definieren, wie die Modelle von M1 aufgebaut und strukturiert sind  
"Ein Video ist eine Klasse mit definierten Attributen"

**M3 - Meta-Meta-Modell** Das Meta-Meta-Modell definiert das Meta-Modell aus M2 und gleichzeitig sich selbst (M3). M3 stellt somit die oberste Ebene dar, die sich selbst beschreibt und somit ein unendlicher Meta-Loop generiert.  
"Eine Klasse ist eine Klasse ist eine Klasse ist eine..."

### 15.3 Data Warehouse Manager

Siehe Kap. 4.1.8 - Verwaltungsbereich

### 15.4 Metadaten Manager

Siehe Kap. 14.4

### 15.5 Unterscheidung von Metadaten

Optimal hätte ein Data Warehouse ein zentrales Metadaten-Repository. In Realität ist es jedoch meist so, dass zwischen technischen und geschäftlichen Metadaten unterschieden wird.

Je nach dem, wen man fragt, kommen noch prozessuale Metadaten hinzu.



Abb. 15.3: Unterscheidung zwischen technischen und geschäftlichen Metadaten

#### 15.5.1 Technische Metadaten

Diese Metadaten beschreiben und definieren das Datenbank-System und werden meist automatisch angelegt. Sie sind Daten über das Data Warehouse Modell und sind meist im DBMS abgelegt

Solche Metadaten umfassen zum Beispiel:

- Server und Instanzen (Ort und Name)
- Datenbankname
- Datenbankobjekte (Tabellen, Indizes, Sichten, etc.)
- Datenmodelle der Quellsysteme
- Datenextraktionsregeln
- Datentransformationsregeln
- Daten zur Versionierung
- Data Cleansing Regeln
- Daten Aggregations Regeln
- Namen von Tabellenviews
- Daten über die Steuerung des Lade- prozesses
- Daten über den Datenupdatezyklus
- Primär-/Fremdschlüssel Beziehungen
- Datenstrukturen
- Datenverteilung (Partitionierung)
- Quell- und Zielsystem Beschreibung von ETL-Prozessen
- etc.
- Transformationsvorschriften, -richtlinien (Policies) -
- 
- Datenqualitätskriterien
- Ausnahme- und Fehlerbehandlungen
- Datentypen, Wertebereiche (Domänen)
- Auskunft über Bezeichner (Namen)
- Auskunft über offene Transaktionen
- Rechtevergabe

- Statistiken über Mengeninformationen und Datendichte
- Serverstatistiken (Uptime, Anzahl Verbindungen, ...)
- Zugriffsstatistiken (Häufigkeiten von richtigen und falschen Anmeldungen, ...)
- System-, Versions- und Konfigurationsdaten

### 15.5.2 Geschäftliche Metadaten

Diese Metadaten beschreiben die Datenherkunft, ihre geschäftliche Bedeutung, Rollen und Verwendung. Im Gegensatz zu den technischen Metadaten unterstützen sie die geschäftlichen Analysen.

- Datenlieferanten
- Datennutzer
- Datenflüsse
- Bedeutung und Definition der Geschäftsdaten
- Geschäftsregeln der Datentransformation (Minimum, Maximum, NULLBehandlung usw.)
- Gruppierungen
- Aggregationen
- Dimensionshierarchien
- Vordefinierte Berichte
- Vordefinierte Queries
- Daten für die Query und Reporting Tools
- Report Verteilinformation
- Sicherheits- und Zugriffsprivilegien

### 15.5.3 Prozessuale Metadaten

Diese Metadaten beschreiben die Prozesse im Data Warehouse

- Regeln zur Datenextraktion, zur Transformation und zum Ladevorgang
- Auslösezeiten, Endzeiten von Vorgängen
- Benutzte Ressourcen (CPU, Primär-, Sekundärspeicher usw.)
- Betroffene Objekte
- Mengen, z.B. Tupelmengen (Row Counts)
- Vollständigkeits- und Qualitätsstatistiken für die Auditierung

Zudem wird unterschieden zwischen beschreibenden und vorgangsbezogenen Metadaten.

#### Beschreibende Metadaten

Darunter fallen Daten, die das System *beschreiben* (wie der Name das schon vermuten lässt), also z.B. über alle Schnittstellen, Komponenten und Daten

#### Vorgangsbezogene Metadaten

Darunter fallen Daten, die wichtig sind in bestimmten Prozessen wie z.B. den ETL-Prozessen, also Daten über die Quell- und Zielsysteme, über die Quell- und Zieldaten etc.

## 15.6 Metadaten-Strategie

Die Strategie befasst sich mit den Fragen *Wie und wer pflegt und integriert die Metadaten?*

### 15.6.1 Wie

Die *Wie*-Frage hängt extrem von den verwendeten Produkten ab, aber das Ziel wäre eine totale Zentralisierung in einem einzigen Repository.

Prinzipiell können drei Ansätze unterschieden werden:

**Do-it Yourself:** Die Firma kümmert sich selbst um die Metadaten

**Kernprodukt:** Die Firma verwendet ein Produkt, das das Repository und dessen Struktur vorgibt

**Gesamtlösung:** Die Firma hat einen Lieferanten, der bereits die gesamte Infrastruktur liefert hat und somit auch eine zentrale Repository-Lösung bieten muss.

### 15.6.2 Wer

Die Person, die die Metadaten pflegt sollte optimalerweise alles wissen, sowohl im Geschäftlichen wie auch im Informatik-Aspekt.

Ein bisschen genauer heisst das:

- Sehr gute Kenntnisse von DB/DBMS
- Sehr gute Kenntnisse von SQL
- Sehr gute Kenntnisse von XML
- Sehr gute Kenntnisse von den verwendeten Diensten und Produkten
- Gesunden Menschenverstand

### 15.7 Normierung der Metadaten

Das Ziel ist nicht nur ein zentrales Repository, sondern auch eine Normierung aller Metadaten. Das heisst

- Einheitlichkeit
- Herstellerneutralität
- Maschinenlesbarkeit
- Einfach bearbeitbar mit Tools.

Ein recht verbreiteter Standard ist das *Common Warehouse Metamodel* oder CWM. Es umfasst die Modellierung, Beschreibung, Zugriff und Austausch von Metadaten und die Beschreibung anderer Objekte und Komponenten.

Das CWM wurde wie das MOF (Kap. 15.2) von der OMG entwickelt und greift bei verschiedenen Metadaten auch auf das MOF zurück, um eine gemeinsame Basis zu finden, mag das auf M2 oder auch erst M3 sein.

CWM ermöglicht die Beschreibung der Datenmodelle der Quell- und Zielsysteme sowie die Datenverschiebungen zwischen diesen beiden Systemen. Zudem können via CWM Abbildungsvorschriften zwischen dem physischen und dem logischen Modell des Data Warehouse definiert werden.

Management	Warehouse Process			Warehouse Operation	
Analysis	Transformation	OLAP	Data Mining	Information Visualization	Business Nomenclature
Resource	Relational	Record	Multidimensional	XML	
Foundation	Business Information	Data Types	Expressions	Keys and Indexes	Software Deployment
Object Model	Core		Behavioral	Relationship	Instance

Abb. 15.4: CWM-Schichtenmodell

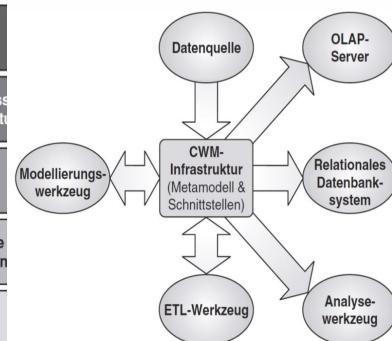


Abb. 15.5: CWM-Metamodell

Ein Ziel des CWM ist die Vereinfachung des Metadatenaustauschs. Es fungiert als gemeinsamer Nenner. Alle Metadaten laufen zuerst über das CWM bevor sie vom Quellsystem ins Zielsystem überführt werden.

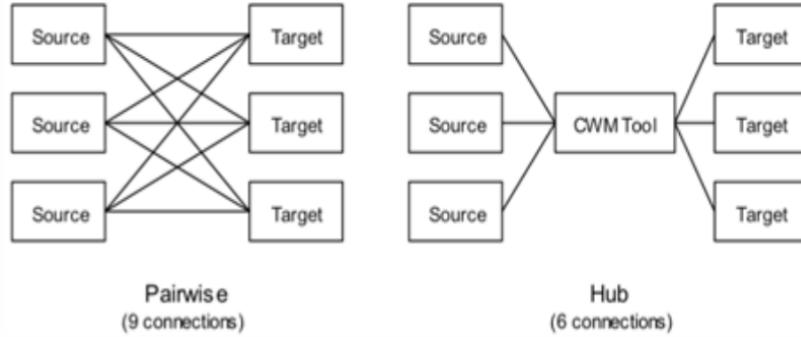


Abb. 15.6: Vereinfachung des Umherschiebens von Metadaten durch das CWM

Zusammengefasst soll das CWM Dinge vereinfachen wie z.B.

- ETL-Applikationsentwicklung
- Tabellenerstellung in Data Warehouse und Data Mart
- Erstellens des Database Persistence Layer
- OLAP Persistenzschicht
- Datenmigration vom Data Warehouse zum Data Mart
- Aufsetzung des Reporting Tools

CWM-Definitionen werden in XMI verfasst. XMI steht für *XML Metadata Interchange* und ist ein Standard der OMG für den Austausch von Metadaten

---

```

<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.1" timestamp="Jun 20 2005 09:35:52" xmlns:CWM="org.omg.CWM1.0"
  xmlns:CWMRDB="org.omg.CWM1.0/ Relational" xmlns:CWMOLAP="Olap"
  xmlns:CWMTFM="Transformation">
<XMI.header>
  <XMI.documentation>
    <XMI.exporter>Meta Integration Model Bridge</ XMI.
    <XMI.exporterVersion>4.1.0 - Aug 16 2004 12:21:11</XMI.
  </XMI.documentation> <XMI.metamodel xmi.name="CWM" xmi.version="1.0"/>
</XMI.header>
<XMI.content>
  <CWMRDB:Catalog xmi.id="_4" name="Model" visibility="public">
    <CWM:Namespace.ownedElement>
      <CWMOLAP:Schema xmi.id="_5" name="Logical" visibility="public"
        namespace="_4">
        <CWMOLAP:Schema.dimension>
          <CWMOLAP:Dimension xmi.id="_6" name="Assessment Summary"
            visibility="public" schema="_2">
            <CWM:Namespace.ownedElement>
              <CWMTFM:TransformationMap xmi.id="7" visibility="public"
                namespace="_6">
                <CWM:Namespace.ownedElement><CWMTFM:ClassifierMap
                  xmi.id="_8" name="unnamed_8" visibility="public"
                  namespace="7" transformationMap="7">

```

---

Listing 9: Ausschnitt einer Definition einer relationalen 'Administrations'-Tabelle

## 16 Data Warehouse Strategie (Skript S286)

Die Data Warehouse Strategie ist ein Teil der IT Strategie und sollte keinesfalls ein isolierter Prozess sein. Die IT-Strategie wiederum ist ein Teil der Unternehmensstrategie.

Bei der Strategiefindung spielen neben den Daten auch der Zweck des Warehouses, dessen Organisation, die Technik, das Budget und auch der Mensch eine entscheidende Rolle.

Wenn eine Strategie und Architektur des Data Warehouse gefunden ist, ist das Thema nicht abgeschlossen. Beide Dinge sollten ein permanenter Prozess sein.

### 16.1 Reifegradmodelle

Reifegradmodelle stufen den Zustand eines Systems oder eines Teilsystems innerhalb einer vordefinierten Abstufung ein. Diese Abstufungen zeigen den Ausprägungs- Entwicklungs und Leistungsstand des Systems.

Ein Reifegradmodell dient zur Standortbestimmung und dient unter anderem dazu, festzustellen, wie gross der Aufwand ist, um zur nächsthöheren Stufe zu gelangen.

Eine tiefere Stufe heisst nicht zwingend, dass das System eine niedrigere Qualität ht.

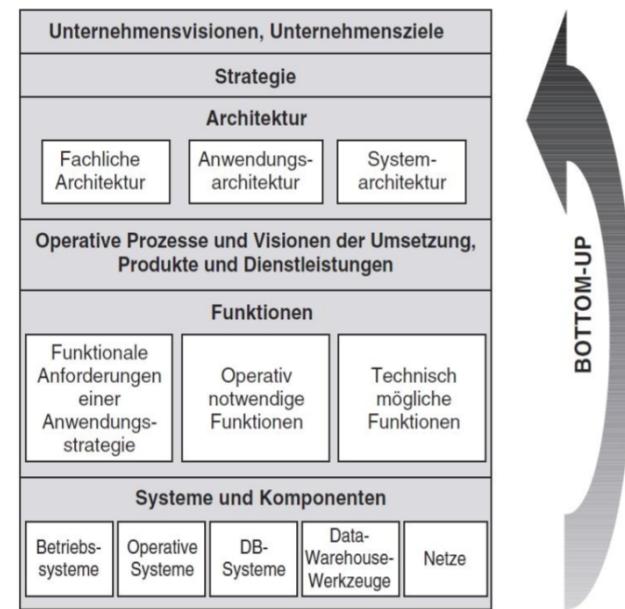


Abb. 16.1: Data Warehouse Strategie

Vorteile		Nachteile	
<ul style="list-style-type: none"> <li>• Standortbestimmung = Selbstreflexion</li> <li>• Erzwingt Systembetrachtung von anderem Winkel <ul style="list-style-type: none"> <li>• Aufdeckung von Über- bzw. Untertwicklung</li> <li>• Man kann gleich noch aufräumen</li> <li>• Bei gutem Reifegrad kann der in Kommunikation mit Partner verwendet werden</li> </ul> </li> </ul>		<ul style="list-style-type: none"> <li>• Wildwuchs an Reifegradmodellen</li> <li>• Überaufwand für Erreichen bestimmter Stufen</li> <li>• Reifegrad basiert nur auf qualitativ und nicht quantitativ</li> <li>• Meist nur auf ein Teilsystem fokussiert. Keine Gesamtübersicht</li> </ul>	

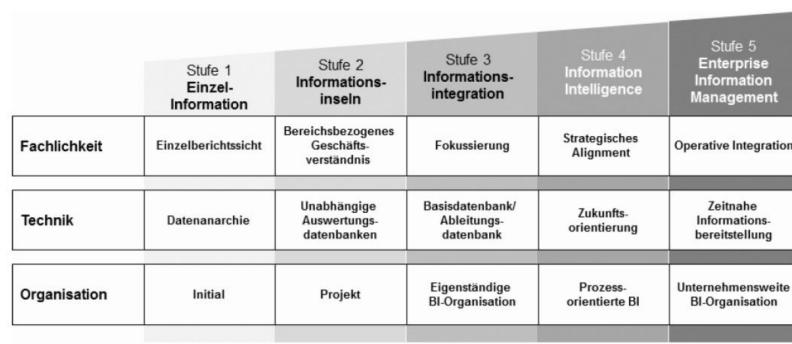


Abb. 16.2: Beispiel eines Reifegradmodells (biMM)