

*Die Wirklichkeit ist nur ein Teil des Möglichen.
Friedrich Dürrenmatt*

In diesem Kapitel betrachten wir eine Seite von Business Intelligence: die Datenbereitstellung. Die Verwendungsseite, der Teil von BI also, der die Managementaufgaben unterstützt, wird in den folgenden Kap. 3 bis 5 behandelt.

Unter **Data Warehousing** versteht man den Geschäftsprozess, der die *Datenbeschaffung* aus internen und extern zugänglichen Quellen, die *Datentransformation und -aufbereitung* gemäß der Quell- und Zieldatenbankschemata, die *Datenqualitätssicherung* und die *Speicherung* im (zentralen) Data Warehouse bzw. in (dezentralen) *Data Marts* (Benutzersichten) und die auf OLAP basierende *Datenanalyse* umfasst.

Nutzer des Data Warehousing sind Middle- bis Topmanager eines Unternehmens, für deren Planungs-, Entscheidungs- und Controllingaufgaben die Daten in geeigneter personalisierter und strukturierter Form als Data Marts präsentiert, vorgehalten und zweckmäßig gepflegt werden.

Nach einer Einführung (Abschn. 2.1) stellen wir die **Architektur** eines Data Warehouses im Abschn. 2.2 dar. Aus Endbenutzersicht spielen dabei *Data Marts* eine entscheidende Rolle, da sie die benötigten Daten anwenderspezifisch in effizienter und personalisierter Form als *Datenwürfel* bereitstellen.

Mit der Architektur allein ist es allerdings noch nicht getan. Vielmehr müssen die Geschäftsdaten, die den diversen Geschäftsprozessen wie Einkauf, Management, Herstellung oder Vertrieb zugeordnet sind, sowie externe Daten wie Web-, Markt- und Branchenda-

ten in das **Data Warehouse** überführt werden. Dazu dient der Teilprozess **Extraktion, Transformation und Laden (ETL)**, speziell die sog. **Datenintegration** (siehe Abschn. 2.3), die es ermöglicht, heterogene Daten aus verteilten, autonomen Datenquellen zu übernehmen [224].

Neben semantischen und strukturellen Konflikten bestehen besonders aufwändige Integrationsprobleme dann, wenn zwei Datenbestände, die zusammengeführt werden sollen, keinen gemeinsamen eindeutig identifizierenden Schlüssel (*Primärschlüssel* oder *Schlüsselkandidat*) haben. Ein Teilproblem dabei ist das Erkennen und Bereinigen von *Duplikaten*, auch *Dubletten* genannt. Ein Beispiel sind Adressdateien von verschiedenen Adressanbietern ohne gemeinsamen identifizierenden Schlüssel, die speziell in der *Kundenbetreuung* (Customer Relationship Management) oder im *Marketing* auftauchen.

Ein weiteres Beispiel ist die Volkszählung von 2011 in Deutschland (Zensus 2011), die ganz überwiegend als Abgleich staatlicher Register – u. a. Dateien der Einwohnermeldeämter, der Bundesagentur für Arbeit und der Haus- und Grundstückseigner – durchgeführt wurde. Im Englischen wird solch eine Volkszählung als *Administrative Record Census* (ARC) bezeichnet. Auch hier existiert kein globaler Schlüssel, der in allen (amtlichen) Registern mitgeführt wird. Dies gilt insbesondere für die Steuernummer, die zwar jedem deutschen Steuerpflichtigen von Geburt an zugewiesen wird, die aber auf den Kreis der Finanzämter beschränkt ist. So ist sie beispielsweise nicht im Datensatz amtlich gemeldeter Personen bei den deutschen Einwohnermeldeämtern gespeichert.

Ein bis ins Ende der neunziger Jahre im Unternehmensbereich unterschätztes Problem stellt die **Qualität der Daten** dar. Im Abschn. 2.4 wird auf die verschiedenen Aspekte der Datenqualität, den Datenqualitätssicherungsprozess und auf die Methoden und Werkzeuge zur Qualitätssicherung wie *Data Profiling* und *Data Cleansing* eingegangen.

Abschnitt 2.5 beschreibt das Konzept des sog. **Online Analytic Processing (OLAP)**. Des weiteren diskutieren wir geeignete *logisch-konzeptionelle Datenstrukturen* von Datenwürfeln. Hier geht es darum, ob Datenwürfel mit Hilfe eines relationalen DBMS (relationales OLAP bzw. ROLAP), mit spezifischen multi-dimensionalen Datenstrukturen (MOLAP) oder in Mischform als hybrid OLAP (HOLAP) gespeichert werden sollen. Das Kapitel endet mit dem Abschn. 2.6, der die Frage behandelt, wie **multi-dimensionale Daten** modelliert werden können.

2.1 Einführung

Man kann Business Intelligence (BI) aus drei Blickwinkeln betrachten. Aus der Sicht des strategischen Managements ist der Begriff *Business Intelligence* sehr attraktiv, da er „intelligente“ Software verheißt, was immer dabei „Intelligenz“ bedeuten mag. Damit tut man BI aber Unrecht, da der Begriff ausschließlich wie „Intelligence“ in *Central Intelligence Service* zu interpretieren ist, d. h. als Erfassung von Daten und deren Auswertung. Auf der dispositiven, planerischen Ebene sowie bei der Revision spricht man deshalb zu Recht auch

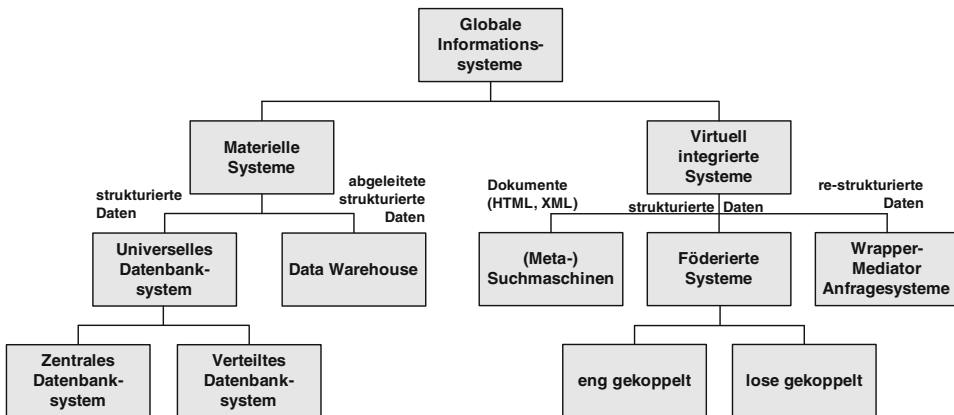


Abb. 2.1 Klassifikation betrieblicher Informationssysteme (in Anlehnung an [81])

von *Online Analytical Processing (OLAP)*, d. h. der Analyse relevanter Daten. OLAP-Daten stehen im Gegensatz zu den operierenden Daten, die als *Online Transaction Processing (OLTP)* bezeichnet werden.

Eine für analytische Daten konzipierte Datenbank wird als **Data Warehouse (DW)** bezeichnet. Diese integriert in einheitlicher Art und Weise – IT-technisch, statistisch und semantisch – die aus den internen Geschäftsprozessen und externen Quellen stammenden Daten. Ein Data Warehouse stellt somit ein auf materialisierten oder virtuellen Datenwürfeln beruhendes betriebliches Informationssystem dar, das die **Integration** von Daten aus autonomen, heterogenen Teilinformationssystemen ermöglicht. Die duale Funktion eines Data Warehouses dient damit zugleich als **Datenauswertungs-** und **Integrationsplattform**. Neben den Realdaten spielen *Metadaten*, die im **Repository** hinterlegt werden, bei Aufbau und Nutzung eines Data Warehouses eine große Rolle, denn sie beschreiben das *DW-Schema* einschließlich aller *Restriktionen* (engl. constraints). Varianten alternativer Integrationsmöglichkeiten von Daten sind in Abb. 2.1 dargestellt. Einige dieser Varianten wie *Wrapper-Mediator-Anfragesysteme* werden ausführlich in [224] behandelt. Wie man in Abb. 2.1 erkennt, gehört ein Data Warehouse zu den materiellen Systemen. „Konkurrenten“ dieser Architektur sind unter den virtuell integrierten Systemen die *Föderierten Systeme* und die oben bereits erwähnten *Wrapper-Mediator-Anfragesysteme*. Die Erfahrung aus den letzten fast zwanzig Jahren zeigt, dass die Unternehmen weltweit der DW-Variante den Vorzug geben [12].

Die Analyse der Daten erfolgt nicht in den ERP-Systemen selbst. Sie wird aus Gründen der Performanz und der Integration im Data Warehouse ausgeführt, da das Datenvolumen bei Großunternehmen in der Größenordnung von Terabytes ($\sim 10^{12}$ Bytes) und höher liegt. Der Begriff *Data Warehouse* wurde von *Inmon* Mitte der neunziger Jahre geprägt [132].

Ein **Data Warehouse** ist ein

- subjektorientierter,
- integrierter,
- zeitraumbezogener und
- nicht-volatiler

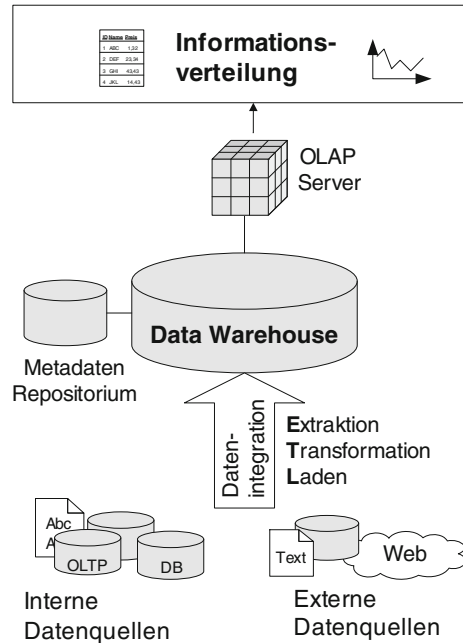
Datenbestand, um die Entscheidungs-, Planungs- und Controllingprozesse des Managements zu unterstützen [132, S. 33].

Wegen der Art und des Umfangs der aggregierten Daten, des überwiegend lesenden Zugriffs mit Punkt- und Bereichsabfragen, sind spezielle physische Datenstrukturen notwendig. Einen Vergleich der Datenstrukturen von *Bitmaps* und *baumbasierten Indexstrukturen* bietet [137]. Eine weitere sehr effiziente Datenstruktur für Datenwürfel sind *UB-Bäume*, die den Vorteil schneller Aufdatierungen und Datenabfragen bieten [189].

Datenwürfel als generische Datenstrukturen weisen eine Menge von zugehörigen Operationen auf. Dazu gehören Zugriffoperatoren wie Marginalisieren (Dicing), Traversieren von Klassifikationsbäumen (Drill-down, Roll-up) und Konditionieren (Slicing). Hinzu kommen die SQL-Aggregatfunktionen wie *min/max*, *avg*, *sum* und *count*. Die Nicht-Standard OLAP-Operatoren *top-ten*, *median* oder der Gruppierungsoperator *group-by* bzw. die Sortieroption *order by* führen zu sog. „langen Transaktionen“, da sie sehr rechenintensiv sind. Sie würden im ERP-System zu umfangreichen Sperrungen (locks) von Datenobjekten in der Datenbank führen und die regulären Geschäftsprozesse damit zu stark belasten. Sie werden stattdessen ins *Data Warehouse* ausgelagert (siehe Abb. 2.2). Auch die Zugriffsoptimierung im Data Warehouse unterscheidet sich von den klassischen OLTP-Datenbanksystemen, da Leseoperationen dominieren, und das Löschen von Daten wegen des historischen Bezugs der Analysen nahezu nicht vorkommt.

Das Data Warehouse ähnelt den **statistischen Informationssystemen** [171, 277, 172]. Der einzige Unterschied liegt in den Datenquellen und den Geschäftsprozessen. Beim Data Warehouse liefern die betriebseigenen Geschäftsprozesse und Fremdquellen die Daten. Dagegen sind die *statistischen Informationssysteme* „Sammelstellen“ von Daten, die auf Nachweisen wie bei der amtlichen Ein- und Ausfuhr, auf gesetzlichen Erhebungen wie beim Mikrozensus, der periodischen Berichterstattung der Unternehmen, der Lohn- und Einkommensteuererhebung bei den in Deutschland Steuerpflichtigen oder, wie bei der Volkszählung (Zensus2011), auf der Zusammenfassung von amtlichen Registern (sog. *Registerabgleich*) beruhen. Datenlieferanten sind dabei Privatpersonen (oder Haushalte), Unternehmen, nicht gewerbliche Organisationen sowie der „Staat“ mit seinen Gebietskörperschaften, Bundesländern und der Bundesrepublik.

Abb. 2.2 Das Data Warehouse Konzept



Die Tab. 2.1 verdeutlicht den Zusammenhang zwischen OLTP- und OLAP-Systemen. Im Folgenden wird die unterschiedliche Sichtweise der Nutzer auf OLTP- und OLAP-Daten verdeutlicht.

Beispiel Betrachten wir den Unternehmensbereich *Auftragsannahme und Fakturierung*. Das Fachkonzept auf Datenebene ist als *Entity-Relationship-Modell (ERM)* in Abb. 2.3 dargestellt. Eine typische OLTP-Abfrage, eine sog. „Punktabfrage“, auf operierende Daten (*Mikrodaten*) durch einen Vertriebsachbearbeiter könnte lauten:

```
select Datum, Betrag
from Rechnung
where Rechnungsnr = 100
```

Im Gegensatz dazu interessiert sich ein Manager im Vertrieb weniger für einen Einzelfall, sondern für eine Fallmenge und deren statistische Eigenschaften, also mehr für *OLAP*- oder *Makrodaten*, d. h. gruppierte und aggregierte Daten. Diese haben grundsätzlich mehrere Dimensionen, sind damit **mehrdimensional**. Eine *Dimension* ist ein Attribut, das nicht von anderen im Datenwürfel enthaltenen Attributen funktional abhängig ist [21]. Interessanterweise lassen sich alle Dimensionen eines Datenwürfels gemäß des sog. **3D-Prinzips** auf drei Grunddimensionen zurückführen: *Zeit*- und *Raumdimension* sowie *Sachdimensionen*. Nur letztere kann in weitere, voneinander unabhängige, fachliche Dimensio-

Tab. 2.1 OLTP und OLAP im Vergleich [21, S. 8ff]

Merkmal	OLTP	OLAP
Anwendungsbereich	Operative Systeme	Analytische Systeme
Nutzer	Sachbearbeiter	Entscheidungs- und Führungskräfte
Datenstruktur	zweidimensional, nicht verdichtet	multidimensional, subjektbezogen
Dateninhalt	detaillierte, nicht verdichtete Einzeldaten	verdichtete und abgeleitete Daten
Datenverwaltungsziele	transaktionale Konsistenzerhaltung	zeitbasierte Versionierung
Datenaktualität	aktuelle Geschäftsdaten	historische Verlaufsdaten
Datenaktualisierung	durch laufende Geschäftsvorfälle	periodische Datenaktualisierung (Snapshots)
Zugriffsform	lesen, schreiben, löschen	lesen, anfügen, verdichten
Zugriffsmuster	vorhersehbar, repetitiv	ad hoc, heuristisch
Zugriffshäufigkeit	hoch	mittel bis niedrig
Antwortzeit	kurz (Sekundenbruchteile)	mittel bis lang (Sekunden bis Minuten)
Transaktionsart und Dauer	kurze Lese und Schreiboperationen	lange Lesetransaktionen

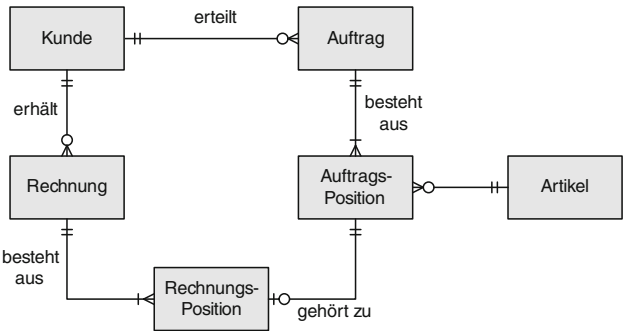


Abb. 2.3 ER-Modell für die Funktionsbereiche Auftragsannahme und Vertrieb

nen zerlegt werden [171]. Beispiele einer Zerlegung der Sachdimension sind Artikel, Qualitätsstufe, Verpackung oder Vertreter. Typischerweise haben Datenwürfel mehr als drei Dimensionen.

Ein aus obigen operierenden Daten abgeleiteter und für die Vertriebsabteilung nützlicher (einfacher) Datenwürfel *Absatzstatistik* könnte wie Abb. 2.4 aussehen.

Um Prognosen zu erstellen, kann die Vertriebsabteilung sich dann die benötigten Zeitreihen einfach mittels der folgenden Bereichsabfrage – sicherlich innerhalb einer

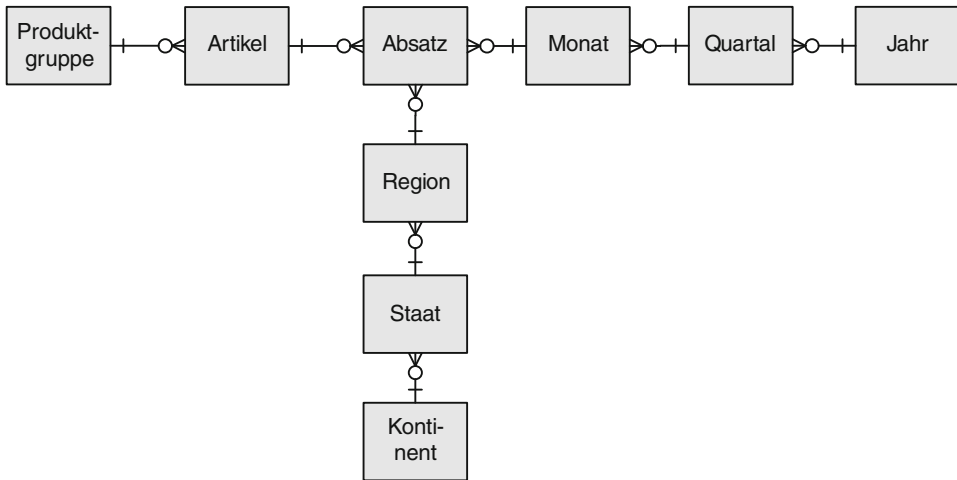


Abb. 2.4 ER-Modell für einen Datenwürfel des Vertriebs

benutzerfreundlichen GUI – generieren. Dabei wird hier *Multidimensional Expressions (MDX)*, eine an SQL angelehnte Abfragesprache für multidimensionale Daten, verwendet:

```

SELECT Produktgruppe.MEMBERS ON COLUMNS
Jahr.CHILDREN on ROWS
FROM Sales
WHERE (Measures.Umsatz, Region.Deutschland)
  
```

Die Tatsache, dass Data Warehousing einen zentralen Bestandteil betrieblicher Informationssysteme ausmacht, hat mit gewisser „Symmetrie“ betrieblicher Datenverarbeitung zu tun. Während mit den herkömmlichen ERP-Systemen, wie sie beispielsweise SAP und Oracle anbieten, die Unternehmen die **Logistik operierender Daten** ab Mitte der siebziger Jahre in den Griff bekommen haben, ist Vergleichbares auf Makrodatenebene erst gut fünfzehn Jahre später mit dem Einzug von *Data Warehousing* und der *Business Intelligence* geschehen.

Weiterführende Literatur Dem Leser, der über Data Warehousing insgesamt mehr erfahren will, empfehlen wir das didaktisch gut geschriebene, umfassende Sammelwerk von Bauer und Günzel (2009), „Data Warehouse Systeme“ [21]. Es sind hier vornehmlich die Kap. 1–4 von Interesse. Ebenso lesenswert als Einführung ist das Buch „Building the Data Warehouse“, verfasst vom Vater der Data-Warehousing-Idee, Inmon (2009) [132]. Schließlich kann die Einführung „Business Intelligence“ von Kemper, Mehanna und Unger (2006) herangezogen werden [143].

2.2 Data Warehouse Architektur

Als **Data Warehouse Architektur** wird der planvolle, fachkonzeptionelle Struktur-entwurf des *Data Warehouse Systems* und dessen Einbettung in sein reales Umfeld bezeichnet.

Eine solche Architektur veranschaulicht Abb. 2.5. Im Folgenden erläutern wir die Architekturkomponenten eines Data Warehouses und anschließend verschiedene Architekturvarianten.

2.2.1 Architekturkomponenten

Die Architekturkomponenten können entlang des Datenflusses von den Vorsystemen hin zu Werkzeugen der Endanwender aufgezählt werden.

Notwendige Voraussetzung des Data-Warehouse-Prozesses ist die Verfügbarkeit von **Datenquellen**. Die Datenquellen bestehen aus den ERP- und anderen operativen Systemen oder externen Quellen, die die benötigten Daten in Form von Tabellen, Dokumenten oder sonstigen *Medien* (Bilder, Audio, Video) bereitstellen. Externe Quellen sind im wesentlichen Daten aus dem Web, von Wirtschaftsforschungsinstituten, Banken, Zulieferern und Kunden, sowie periodisch anfallende Branchen- und Wirtschaftsreports.

Diese Daten werden extrahiert (selektiert bzw. repliziert) (siehe Abschn. 2.3) und in einem Zwischenspeicher (**Staging Area**, auch Arbeitsbereich genannt) für notwendige Transformationen zur Erreichung der gewünschten Homogenität, Integrität und Qualität der Daten gespeichert. Zur Einhaltung der angestrebten Aktualität werden hierbei **Monitore** eingesetzt.

Die Daten werden je nach vorgefundener **Datenqualität** und existierenden Konflikten mittels sog. Transformationen qualitativ überarbeitet (siehe Abschn. 2.4). Anschließend werden sie permanent im (**Core**) **Data Warehouse** gespeichert.

Inmon [132] propagiert die zusätzliche Verwendung eines **Operational Data Store (ODS)** (auch Basisdatenbank). Der wesentliche Unterschied zum Data Warehouse ist der, dass ein ODS ein bereinigtes, integriertes und normalisiertes Schema aller operativen Systeme hat und die Granularität auf Transaktionsebene ist. Die Daten sind somit nicht für analytische Abfragen optimiert. Das ODS versorgt das Data Warehouse mit Daten. Wegen des Aufwands zum Aufbau und zur Pflege wird in der Praxis oft auf einen Operational Data Store verzichtet [21, S. 54].

Die Speicherung kann prinzipiell zentral im (**Core**) **Data Warehouse** erfolgen, als praktischer hat sich jedoch erwiesen, die Speicherung entlang der betrieblichen Prozessketten als **Data Marts** (zusätzlich) vorzuhalten. Diese repräsentieren die von der jeweiligen Fachabteilung benötigten **Datenwürfel (Data Cubes)** (siehe Abschn. 2.5).

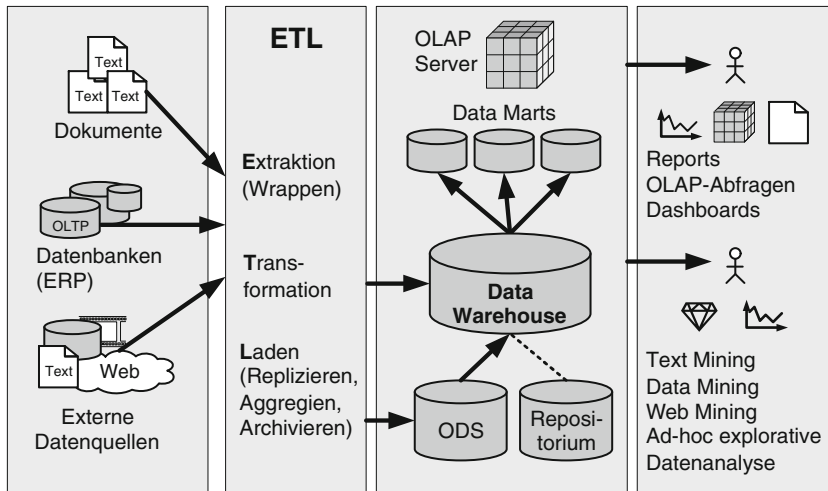


Abb. 2.5 Architektur eines Data Warehouses

Das Data Warehouse beinhaltet den *Zwischenspeicher* (Staging Area), das *Core Data Warehouse* oder kurz Data Warehouse, *Data Marts*, falls sie „materialisiert“ sind und nicht nur bei Bedarf („on the fly“) generiert werden, das *Archiv*, das *Repositorium*, das Metadaten vorhält, und das *Log* zur Steuerung und Konsistenzhaltung von Transaktionen. Das Archiv dient dabei der Replikation benötigter operativer und externer Daten sowie der Datensicherung der aktuell nicht mehr im direkten Zugriff benötigten aggregierten Daten. Dazu rechnen auch die mit Zeitstempel versehenen Versionen von Metadaten wie beispielsweise obsoletere Produktklassifikationen, Stellenpläne usw., die weiterhin noch historisch von Interesse sind und zeitliche Versionen repräsentieren.

Die auf die Informationsbedürfnisse einzelner Endanwender zugeschnittenen Data Marts können materialisiert oder virtuell sein. Sie werden materialisiert, d. h. permanent gespeichert, wenn hohe Zugriffsgeschwindigkeit wichtiger ist als zusätzlicher Speicheraufwand. Andernfalls werden sie *ad-hoc* generiert und *on-the-fly* bereitgestellt.

In einem **Archiv** werden replizierte bzw. langfristig gesicherte Datenbestände gespeichert. Hinzu tritt das **Repositorium**, das die benötigten **Metadaten** bereitstellt. Es spielt nicht nur einmalig beim Aufbau und der laufenden Pflege eines Data Warehouse eine zentrale Rolle, sondern hat große Bedeutung für das *Monitoring*, die *Abfrageoptimierung*, die *statistische Analyse* selektierter Daten und für die korrekte *Interpretation* von Analyseergebnissen. Dies beruht wesentlich auf der statistischen Natur gruppierter, aggregierter Daten, wie Häufigkeitsverteilungen, Zeitreihen- oder Querschnittsdaten, sowie von gepoolten Längs- und Querschnittsdaten.

Die Objekte, auf die sich die anwendungsorientierten **Metadaten** im Repositorium beziehen, haben unterschiedliche Granularität: Data Warehouse, Data Mart, Attribut, Wert und Fußnote. Die Komplexität wird deutlich, wenn die Metadaten betrachtet werden, die

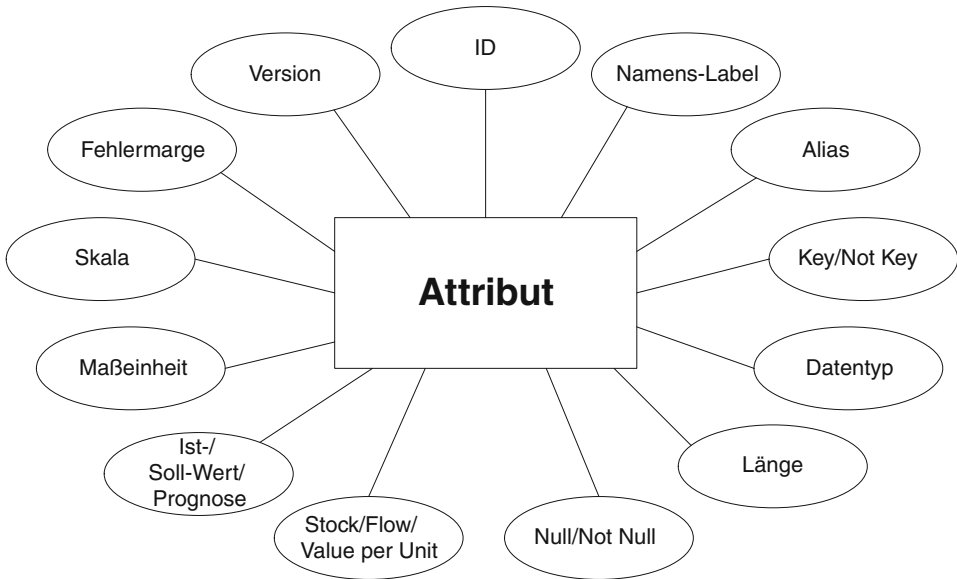


Abb. 2.6 Metadaten von *Attribut* im Repository

Attribute strukturiert beschreiben (siehe Abb. 2.6). Zum konzeptionellen Entwurf eines Repositoriums aus Anwendersicht siehe [171].

Aus funktioneller Sicht besitzt das Data Warehouse analog zu einem Datenbanksystem einen DW-Manager, einen DW-Optimierer und ein Monitor. Der **DW-Manager** ist die zentrale Funktionskomponente, die für die komplette Ablaufsteuerung der Datenflüsse zuständig ist. Dazu gehören in Kooperation mit dem **Monitor** die (zeit- oder fallgesteuerte) Initiierung, Steuerung und Überwachung aller Datenflüsse wie Laden, Bereinigung, Aggregation und Integration in korrekter Reihenfolge. Die Performanz der Abfragen wird dabei wesentlich vom **DW-Optimierer** beeinflusst. Schließlich obliegt dem DW-Manager die Handhabung von Fehlersituationen.

Wie Abb. 2.7 deutlich macht, ist der Datenfluss im Data Warehouse linear strukturiert. Er führt von den operativen Systemen bzw. externen Datenquellen über den Zwischenspeicher (*Staging Area*), auf die die ETL-Werkzeuge zugreifen, zu den Speicherkomponenten des Data Warehouse.

2.2.2 Architekturvarianten

Im Folgenden stellen wir die häufigsten BI-Architekturen vor [13]. Dabei werden zwei Architekturvarianten am meisten verwendet: Data Marts mit gemeinsamen Dimensionen und abhängige Data Marts (Nabe-und-Speiche-Architektur) [12].

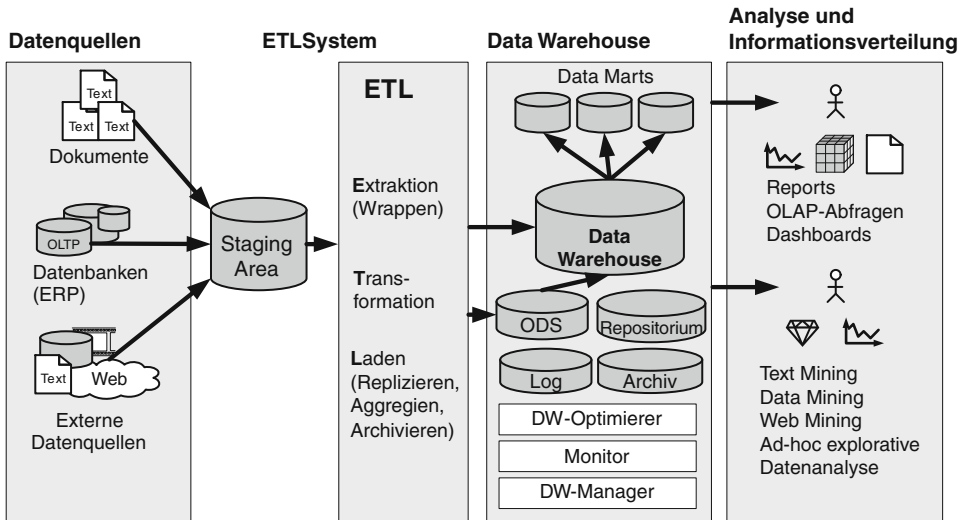


Abb. 2.7 Datenflüsse und Architektur eines Data Warehouses (ohne Kontrollflüsse)

2.2.2.1 Unabhängige Data Marts

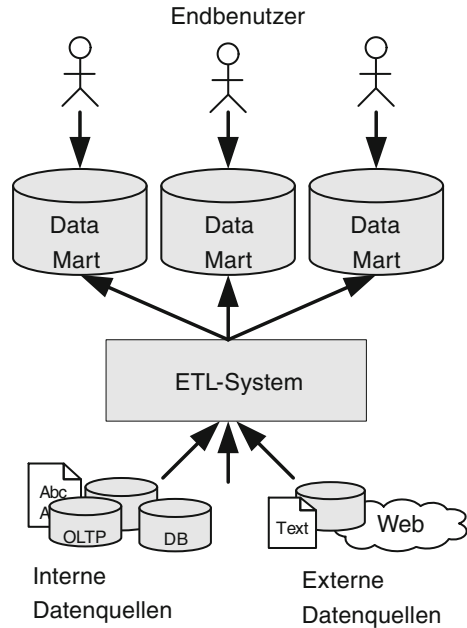
Unabhängige Data Marts sind „kleine Data Warehouses“, die fachbezogen generiert werden (siehe Abb. 2.8). Der Aufbau unabhängiger Data Marts entspricht dem „Bottom-up-Prinzip“.

Wenn BI in einem Unternehmen implementiert wird, wird oftmals mit *einer* Abteilung angefangen und für diese ein unabhängiges Data Mart entwickelt. Wenn nun weitere Abteilungen mit Data Marts versorgt werden sollen, entsteht das Risiko, dass die einzelnen Data Marts voneinander unabhängig entwickelt werden. Dies kann dazu führen, dass Dimensionen und Kennzahlen in den unabhängigen Data Marts unterschiedlich definiert werden und dadurch keine Vergleichbarkeit zwischen den Abteilungen möglich ist. Es gibt also im Unternehmen keine gemeinsame Datenwahrheit („single point of truth“). Ein weiteres Problem besteht darin, dass für jeden Data Mart ein eigener Datentransformations-Prozess (ETL-Prozess) erstellt werden muss [13].

2.2.2.2 Data Marts mit gemeinsamen Dimensionen

Um die Inkonsistenzen bei unabhängigen Data Marts zu verhindern, plädiert Kimball [144] für die Definition von gemeinsamen Dimensionen für verschiedene Data Marts. Beispielsweise ist die Ortsdimension mit der Filialstruktur sowohl für den Absatz-Würfel als auch für den Personaleinsatz-Würfel identisch. Dadurch können Kennzahlen bei identischer Dimensionsstruktur verglichen werden. Außerdem kann von einem Würfel zum anderen gesprungen werden (Drill-Across).

Abb. 2.8 Unabhängige Data Marts



2.2.2.3 Abhängige Data Marts: Nabe und Speiche

Dagegen zeichnet abhängige Data Marts aus, dass der ODS sie im Rahmen eines unternehmensweiten normalisierten Schemas mit Daten versorgt. Dies entspricht dem „*Top-down-Ansatz*“ und als Architektur dem *Hub and Spoke* (Nabe und Speiche) Konzept. Inmon [132] ist ein Verfechter dieser Architektur und in der Praxis zählt diese, mit der alternativen Architektur von Kimball (Data Marts mit gemeinsamen Dimensionen), zu den beiden am meisten eingesetzten.

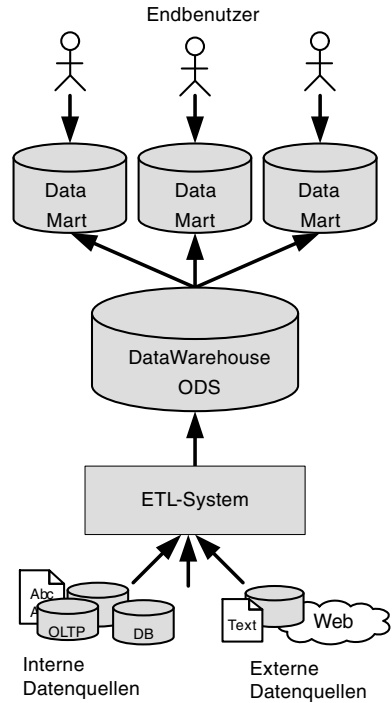
Die (partielle) Hub-Spoke Architektur wird deutlich, wenn die Beziehungen zwischen den Data Marts und dem Data Store des Data Warehouses betrachtet werden (siehe Abb. 2.9).

2.2.2.4 Föderierte Architektur

Föderierte Architekturen integrieren die Quellsysteme nur virtuell, d. h. es wird nicht eine Kopie durch einen ETL-Prozess erstellt. Stattdessen wird bei einer Anfrage direkt auf die einzelnen Quellsysteme zugegriffen und diese on-the-fly zu einem Ergebnis integriert. Dies kann u. a. dann notwendig sein, wenn die Quellsysteme nicht unter der eigenen Kontrolle stehen und sich schnell ändern (etwa Wetterdaten).

Auch wenn mehrere Data Warehouses vorhanden sind, was z. B. nach einer Firmenübernahme der Fall ist, erlaubt eine föderierte Architektur eine einheitliche Sicht, als ob es ein gemeinsames Data Warehouse geben würde. Eine föderierte Architektur besteht typischerweise aus zwei Komponenten: Wrapper (Umhüller) und Mediator (siehe Abb. 2.10). Wrapper abstrahieren die speziellen Zugriffs- und Schema-Eigenschaften der Quelle und

Abb. 2.9 Hub-and-Spoke Architektur

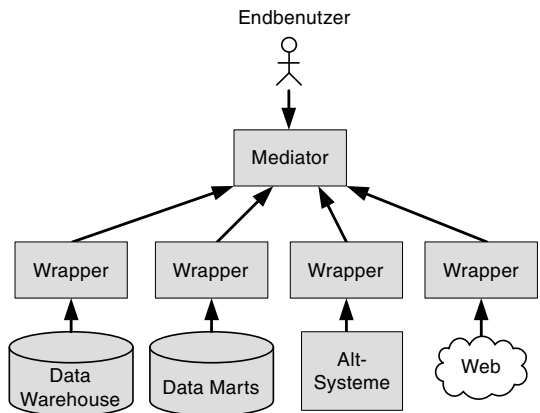


kümmern sich um die Vereinheitlichung der Daten. Der Mediator verteilt Anfragen auf die einzelnen Quellen und integriert die Resultate zu einem einheitlichen Ergebnis [224, 319].

2.2.2.5 Massiv-Parallele Verarbeitung

Firmen wie Google, Facebook, Yahoo! oder LinkedIn verarbeiten Datenmassen, die mit den vorher vorgestellten Architekturen nur schwer zu bewältigen sind (Stichwort Big Data).

Abb. 2.10 Föderation mittels der Wrapper-Mediator Architektur



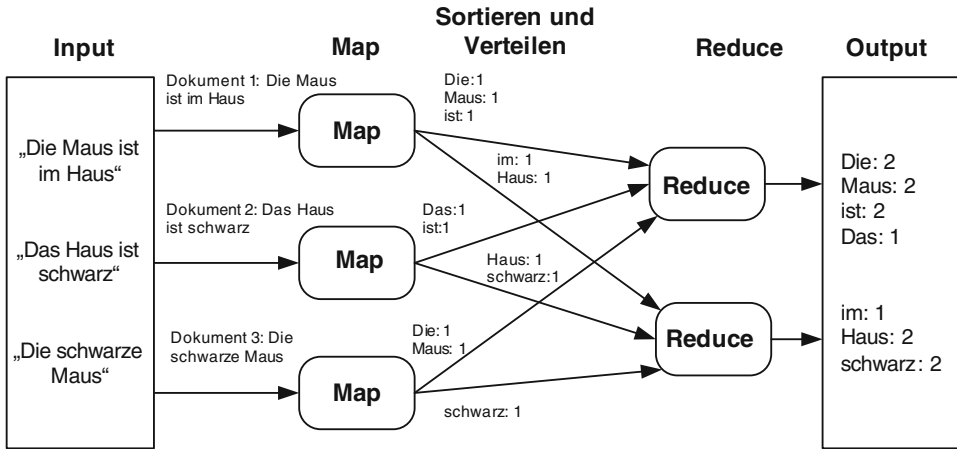


Abb. 2.11 Beispiel: MapReduce Verarbeitung

Darauf reagierend haben Firmen wie Google Technologien entwickelt, die Daten im Bereich von Petabytes parallel auf hunderten oder sogar tausenden von Servern verarbeiten können.

Ein oft genutztes Muster um große Aufgaben in kleinere Teilaufgaben aufzuteilen, ist das von Google entwickelte MapReduce [73]. MapReduce ist für die verteilte nebenläufige Batch-Verarbeitung von sehr großen Datenmengen konzipiert. Es besteht aus zwei Hauptschritten: Dem Map-Schritt und dem Reduce-Schritt. Dazwischen werden die Ergebnisse sortiert und auf verschiedene Reduce-Prozesse verteilt.

Der Input für den Map-Schritt sind Schlüssel-Wert-Paare (Key-Value Pairs). Schlüssel und Wert können einen beliebigen Datentyp haben. Beispielsweise könnten die Schlüssel aus URL-Adressen und die Werte aus HTML-Inhalten der Adressen bestehen [164].

Beispiel Es soll die Anzahl aller in verschiedenen Dokumenten vorkommenden Wörter gezählt werden (Häufigkeitsanalyse). Der erste Schritt besteht in der Transformation der einzelnen Dokumente (Text-Dateien) in Schlüssel-Wert-Paare, damit diese dann vom Map-Schritt bearbeitet werden können (siehe Abb. 2.11).

Map Im Map-Schritt werden die (verteilten) Input-Daten zu neuen Schlüssel-Wert-Paaren transformiert. Die Form der Umrechnung wird in der Map-Funktion von einem Programmierer spezifiziert.

Beispiel (Fortsetzung) Der Input für den Map-Schritt besteht aus den Schlüssel-Wert-Paaren (Dokument-Nr, Text) (siehe Abb. 2.11). Dafür muss nun eine Map-Funktion definiert werden, die die Anzahl der verschiedenen Wörter im Dokument zählt. Der Output sind wiederum Schlüssel-Wert-Paare, diesmal mit dem Wort als Schlüssel und der Anzahl als Wert.

Sortieren und Verteilen Anschließend werden alle Schlüssel-Wert-Paare von einem Master-Controller zusammengefasst und nach dem Schlüssel geordnet. Schlüssel-Wert-Paare mit dem gleichen Schlüssel werden auf den gleichen Reducer verteilt.

Beispiel (Fortsetzung) Der Output des Map-Schrittes wird so auf Reducers verteilt, dass alle Schlüssel-Wert-Paare mit dem gleichen Schlüssel, d. h. mit dem gleichen Wort, zum gleichen Reducer verteilt werden (siehe Abb. 2.11).

Reduce Im Reducer-Schritt werden jeweils alle Schlüssel-Wert-Paare mit dem selben Schlüssel bearbeitet und die Werte kombiniert. Die Reduce-Funktion gibt an, wie die Werte verarbeitet werden sollen.

Beispiel (Fortsetzung) Alle Schlüssel-Wert-Paare mit dem gleichen Wort werden vom gleichen Reducer bearbeitet (siehe Abb. 2.11). Die Reduce-Funktion berechnet die Summe aller Werte mit dem gleichen Schlüssel. Das Ergebnis ist eine Liste von Schlüssel-Wert-Paaren mit den in allen Dokumenten vorkommenden Wörtern und der Anzahl (Häufigkeit) des jeweiligen Wortes in allen Dokumenten.

Hadoop ist das bekannteste Open-Source-Framework, das auf der *MapReduce*-Architektur aufbaut [164]. Wesentliche Bestandteile sind neben dem MapReduce ein verteiltes Dateisystem (Hadoop Distributed File System (HDFS)). Aufbauend auf diesen Kern-Bestandteilen von Hadoop haben Firmen verschiedene Erweiterungen entwickelt, wie z. B. *HBase* und *Hive*. Das Open-Source Projekt *HBase* ist eine massiv-verteilte nicht-relationale Datenbank [164]. Es basiert auf den Ideen des Google-Projekts *BigTable* [56]. *Hive* erweitert Hadoop um Data Warehouse Funktionalität [295]. *HiveQL* ist eine Abfragesprache für *Hive*, die an SQL angelehnt ist.

Eine massiv-parallele Verarbeitung ist nur dann erforderlich, wenn eine Erweiterung eines Servers (Stichwort Scale-Up) nicht mehr möglich ist, da die Datenmengen in die Tera- oder gar Petabyte gehen. Jedoch liegen die Datenmengen für viele Analysen nicht in diesen Größenbereichen. So verarbeiten selbst im Hadoop-Cluster von Facebook 90 % aller Analyseaufgaben weniger als 100 GB an Daten [256]. Bei Microsoft ergab die Auswertung eines großen Clusters, dass der Median der Datenmenge für eine Analyse sogar nur 14 GB betrug [256]. Für viele solcher Analyseaufgaben ist ein hochgerüsteter Server z. B. mit bis zu 50 CPU-Kernen und einigen Hundert GB Hauptspeicher die bessere Alternative. Map-Reduce ist im übrigen als Batch-Verfahren nicht für die sehr schnelle Beantwortung von Abfragen geeignet.

2.2.2.6 Hauptspeicherdatenbanken

In den letzten Jahrzehnten ist nicht nur die Geschwindigkeit von Hauptprozessoren (CPUs) exponentiell gestiegen und deren Preis exponentiell gesunken, sondern auch die Hauptspeicherpreise sind exponentiell gefallen. So ist der Preis von 1 GB Hauptspeicher (RAM) von 1000 \$ Ende der 90er Jahre auf 100 \$ im Jahr 2005 gesunken und liegt im Jahr 2010

unter 10 \$ [242, S. 15]. Damit sind Server mit mehreren Hundert GB oder gar einigen TB für den Unternehmenseinsatz ökonomisch attraktiv geworden.

Festplatten sind um den Faktor 10.000 langsamer als Hauptspeicher [242, S. 14]. Hauptspeicherdatenbanken (In-Memory Databases) speichern nun die kompletten Daten im Hauptspeicher [242]. Der Hauptspeicher ist bekanntlich flüchtig. Daher bieten sich Hauptspeicherdatenbanken eher für analytische als für operative Informationssysteme an, da dort die Anforderungen an die Persistenz nicht so hoch sind. Andererseits können durch Replikation zwischen Servern sowie der zusätzlichen Speicherung auf Festplatte bzw. Flash-Speichern (SSD) auch Hauptspeicherdatenbanken persistent gehalten werden. Damit kommen sie zunehmend auch für operative Systeme in Frage.

Hauptspeicherdatenbanken nutzen weitere Technologien, um die Datenverarbeitung zu verbessern und mehr Daten im begrenzten RAM speichern zu können. Herkömmliche Datenbanken speichern Einträge „zeilenweise“, was für OLTP-Anwendungen und das blockweise Auslesen der Daten, die auf Festplatten gespeichert sind, auch recht effizient ist. Analytische Systeme aggregieren typischerweise bestimmte Spalten (wie z. B. Umsatz) für eine Untermenge von Zeilen. Die *spaltenorientierte Speicherung* ist darauf optimiert, dass die Daten nicht mehr zeilen-, sondern spaltenweise organisiert werden. In analytischen Systemen sind innerhalb einer Spalte oft viele identische Werte sowie NULL-Werte vorhanden. Hier ermöglicht eine effiziente *Komprimierung* der Spalten, dass der „knappe“ Hauptspeicher für größere Datenmengen genutzt werden kann.

Weiterführende Literatur Dem Leser, der sich über die Architektur von Data Warehouses informieren will, empfehlen wir auch hier das Sammelwerk „Data Warehouse Systeme“, von Bauer und Günzel (2006) [21], speziell die Kap. 2, 4 und 8. Ebenso lesenswert als Einführung ist das Buch „Building the Data Warehouse“, verfasst vom Vater der Data-Warehousing-Idee, Inmon (1996) [132]. Köppen, Saake und Sattler (2012) beleuchten „Data Warehouse Technologien“ [156].

2.3 Datenintegration

Datenintegration in einem Data Warehouse ist ein Prozess mit den drei Schritten Extraktion, Transformation und Laden (ETL), der operative bzw. externe Daten aus heterogenen Datenquellen in einem Data Warehouse schemakonform semantisch korrekt vereinigt.

Operative Daten sind die in einer Unternehmung geschäftsbedingt anfallenden Daten. Diese liegen überwiegend in strukturierter Form (meist in relationalen Datenbanken) vor. **Externe Daten** dagegen sind Daten Dritter, die als Reports oder über das Internet oft in semi-strukturierter (als XML-Dokumente) oder unstrukturierter Form als Dokumente im

HTML-, Word- oder PDF-Format beschafft werden. Nur in Ausnahmefällen sind sie festformatiert.

Externe Daten im Telekommunikationsbereich von Privatpersonen haben im Datenvolumen (Text, Audio und Video) rasant zugenommen, da immer mehr Menschen in sozialen Netzwerken, wie Facebook oder Twitter, mittels Smartphones, Tablets oder Handys und mithilfe von z. B. SMS und neuerer Sensortechnik wie z. B. GPS oder Beschleunigungssensoren nebenläufig über gemeinsam erlebte Ereignisse berichten.

Diese Rohdaten werden durch die **4 Vs** – *Volume* (Datenvolumen), *Velocity* (Veränderungsgeschwindigkeit), *Variety* (Datenvielfalt) und *Veracity* (Wahrheitswert) – charakterisiert. Man denke hierbei beispielsweise an Fußballspiele der europäischen Champions League, die vor rund 100.000 Zuschauern im Stadion und Millionen vor den Fernsehschirmen ausgespielt werden. Tausende von Twitter-Nachrichten werden schlagartig initiiert, wenn beispielsweise Strafstöße verhängt werden oder Tore fallen.

Diesen Datenströmen ist gemeinsam, dass sie *heterogen* sind, *verteilt* anfallen und möglicherweise *widersprüchlich* und *unsicher* sind. Auch der *Wahrheitswert* (truth) und der *Glaubwürdigkeitsgrad* (trust) der Botschaft sind kritisch zu sehen. Man denke hier nur an das Ereignis „Wembley-Tor“ beim Länderspiel Deutschland – England im Wembley Stadion, London, 1966: „Tor oder nicht Tor?“. Das Problem im Forschungsbereich hinsichtlich solcher großen Datenmengen liegt weniger bei den ersten drei „Vs“, sondern bei der Messung des *Wahrheitswertes* (Veracity) der Mitteilung und der Auflösung möglicher Widersprüche.

ETL steht für die **Extraktion** und Bereinigung (engl. data cleansing) der als geschäftszielrelevant erachteten Daten aus verschiedenen, heterogenen Quellen, die **Transformation** dieser Daten in das Zielschema durch zweckmäßiges Umformen, Gruppieren, Aggregieren oder einfach nur das Replizieren und schließlich das **Laden** der Daten in das *Data Warehouse* zur Datenbereitstellung als *Data Mart*.

Beispiel Man denke etwa an operative Daten wie Artikel-, Lagerbestands- und Vertriebsdaten aus diversen Geschäftsvorfällen, die zu *Summen monatlicher absoluter Absatzzahlen* verdichtet, nach *Absatzregionen* untergliedert und nach Jahren gruppiert werden müssen, um den gewünschten Informationsbedarf des Leiters der Vertriebsabteilung abzudecken.

Transformationen von Daten werden benötigt, um u. a. (relative oder prozentuale) Marktanteile, Wachstumsfaktoren und -raten oder Indizes zu berechnen. Demgegenüber stehen Fremddaten, die beispielsweise einschlägige Verbände oder staatliche Stellen als Branchendaten in periodischen Reports zur Verfügung stellen. Oder es handelt sich um Konjunkturdaten, die als HTML- oder Word-Dokumente – bestehend aus Fließtext, Abbildungen und Formeln –, von Wirtschaftsforschungsinstituten oder Wirtschaftsverbänden im World Wide Web (WWW) veröffentlicht sind und erst aus dem Internet zielorientiert selektiert und dann dem Zielschema entsprechend aufbereitet werden. In der Automobilindustrie sind dies die Monatsdaten der Neuzulassungen von Kraftfahrzeugen, die das Kraftfahrzeug-Bundesamt in Flensburg periodisch veröffentlicht bzw. dem Verband der deutschen Automobilindustrie (VdA) online überstellt.

Auf einen Punkt ist hinzuweisen, wenn ein Data Warehouse entworfen, implementiert oder angepasst wird: Getreu dem Motto „Auf einen alten Kopf passt nicht immer ein neuer Hut“ muss beachtet werden, dass vorab die **Aufbauorganisation**, d. h. die Weisungs- und Disziplinarstruktur des Unternehmens neu konzipiert oder zumindest angepasst wird, bevor Geschäftsprozesse, Datenstrukturen – hier Datensichten, Datenwürfel und das Data Warehouse – und die betriebswirtschaftlichen Funktionen als Algorithmen mittels gängiger Software-Entwicklungs-Methoden entworfen werden. Hier liegt ein in der Praxis „schwer zu beackerndes betriebswirtschaftliches Feld“.

Beispiel Zahlreiche, ineinander verzahnte aber autonome Sichten auf analytische Daten sind auch in Universitäten erkennbar. Es war bis Mitte 2000 ein Fakt, dass diese staatlichen Institutionen hinsichtlich Einsatz und Nutzung administrativer IT-Technik weit hinter der Wirtschaft zurückfielen. Eine Ausnahme bildete damals die Freie Universität Berlin, die Mitte 2000 mit dem *SAP Campus System Projekt* Neuland betrat; denn Universitäten weichen mit ihrer relativ flachen, aber extrem breiten Organisationsstruktur fundamental von den gängigen Stereotypen von Informationsstrukturen im Bank-, Industrie- und Handelsbereich ab. Dies liegt an der akademischen Mitbestimmung sowie der Stellung der Fachbereiche, denen in der Wirtschaft Abteilungen entsprechen, und der besonderen Forschungsinstitute. Die traditionell begründete Kultur und der wissenschaftliche Rang der einzelnen Fachbereiche führt zu ihrer Autonomie und Heterogenität in der Universitätslandschaft.

Diese Erfahrung machte auch das IT-Team *KCoIT* der FU Berlin, das für den Aufbau eines universitären Data Warehouses zuständig war. Die technische und fachliche („semantische“) Heterogenität der rund fünfzehn Datenquellen – von einer Vielzahl Dateien, über MS Access bis zu *HIS*-Datenbanken und *SAP RS/3* – wird aus der folgenden Liste der FU-Datenquellen deutlich [287]:

- Studierende und Absolventen: *HIS-SOS*
- Prüfungen: *SAP-Campus*
- Personal und Stellen: *SAP-HR* und *ORG Management*
- Organisationsdaten: *SAP-ORG-Management*
- Haushalt (inklusive Drittmittel): *SAP FI/PSM*
- Forschungsprojekte: Spezial-Applikationen in *SAP*
- Flächen: Raum-Datenbank (*MS Access*)
- Publikationen: Uni-Bibliographie (*Aleph*)
- diverse kleine Datenbanken

Die Verschiedenheit der Organisationsstruktur und der Finanzwirtschaft in deutschen Universitäten sind sehr spezifisch und unterscheiden sich erheblich von denen der Wirtschaft (siehe Abb. 2.12 und 2.13). Das Organisationsdiagramm der *FU Berlin* aus dem Jahr 2011 zeigt die ganze Vielfalt von Fachbereichen, wissenschaftlichen, Service- und internationalen Einrichtungen. Die Komplexität des Entwurfs liegt offenkundig an (deutschen)

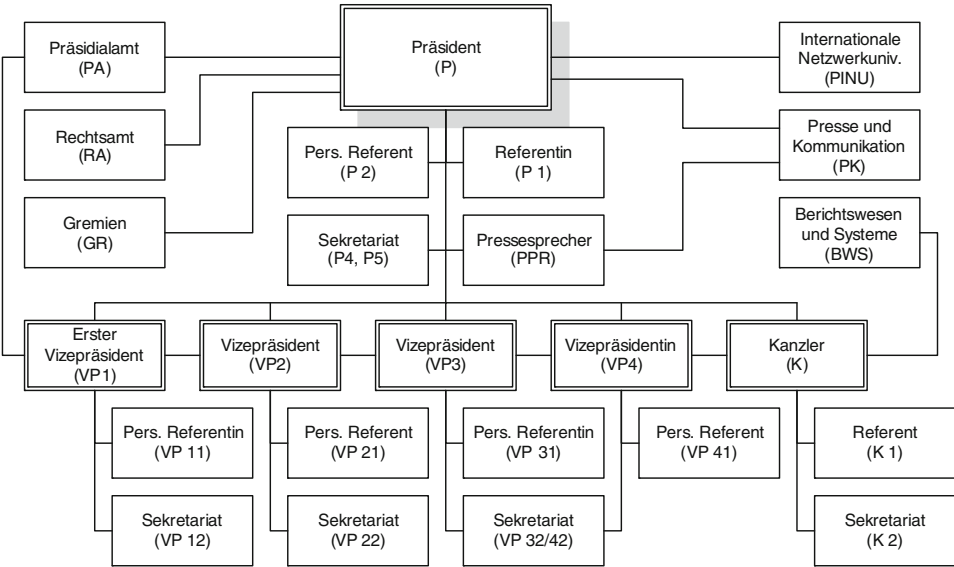


Abb. 2.12 Organigramm des Präsidialamts der Freien Universität Berlin, Stand 2013, <http://www.fu-berlin.de>

Abb. 2.13 Finanzkreise und Aufbauorganisation der FU Berlin, Stand 2005; Quelle: [287]

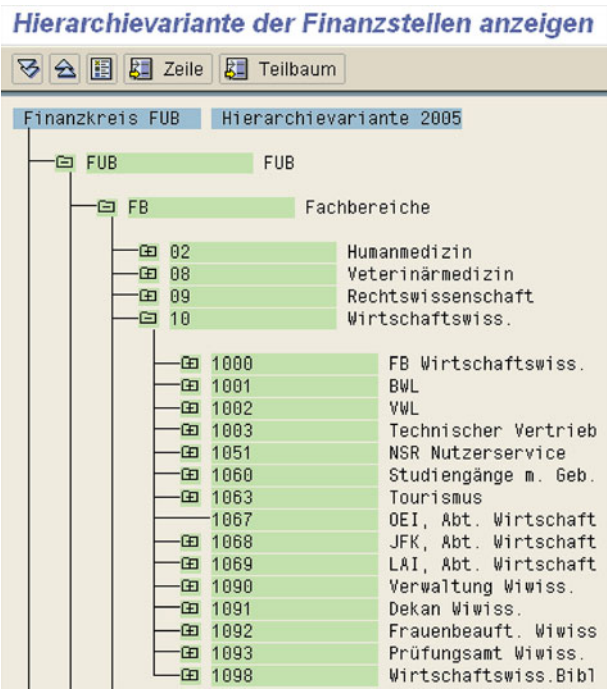


Abb. 2.14 Heterogenität in der Aufbauorganisation des FB Wirtschaftswissenschaft der FU Berlin; Quelle: [287]

Organisation und Besetzung anzeigen

Bezeichnung	Kürzel
Präsidium	P
Medizinische Fakultät Charite	020000
FB Veterinärmedizin	080000
FB Rechtswissenschaft	090000
FB Wirtschaftswissenschaft	100000
Wiss.MitarbeiterIn(Qualifst.)	D10910033-0
UniversitätsprofessorIn	Fz 10-0235
Emeritus	Ez 10-001
Emeritus	Ez 10-002
HonorarprofessorIn	Kz 10-001
HonorarprofessorIn	Kz 10-002
WE01 Betriebswirtschaftslehre	100100
WE02 Volkswirtschaftslehre	100200
Executive Master of Business Marketing	100300
Servicebereich Datenverarbeitung	105100
Willy-Scharnow-Institut für Tourismus	106300
Verwaltung FB Wirtschaftswissenschaft	109000
Frauenbeauftragte FB WiWi	109200
Prüfungsamt FB Wirtschaftswissenschaft	109300
Wirtschaftswissenschaftliche Bibliothek	109800

Universitäten in der *Unbalanciertheit*, *Asymmetrie* und der *Dynamik der Änderung der Aufbauorganisation*, die vom Staat, dem Universitätspräsidenten und dem Akademischen Senat beeinflusst wird.

Mangelnde Strukturkonstanz wird auch deutlich, wenn man einen einzigen Fachbereich, hier den *Fachbereich Wirtschaftswissenschaft*, mit dem Schlüssel 100000, herausgreift. Neben wissenschaftlichen Einrichtungen tauchen in der Binnenstruktur Serviceeinrichtungen, spezielle Institute wie Tourismus und Emeriti auf. Dies führt auch zu einem aufwändigen Schlüsselsystem (siehe Spalte „Kürzel“ in Abb. 2.14).

Fünfundfünfzig Jahre lang haben sich die einzelnen Abrechnungs- und Verwaltungssysteme seit der Universitätsgründung 1948 als sog. **Insellösungen**, die „mächtigen“ Abteilungsleitern unterstellt waren, auseinander entwickelt und zu der unterschiedlichen Strukturierungstiefe gleichartiger Organisationseinheiten und Terminologie mit Homonymen und Synonymen geführt [287]. Dies lag an bürokratischen Hemmnissen wie fehlender globaler Planung und Steuerung der IT und mangelnder Abstimmung zwischen Präsident und Kanzler, also wissenschaftlicher und kaufmännischer Leitung. Hinzu kommen im Universitätsbereich die partiellen, aber hochschulpolitisch immer wieder gewollten, permanenten Änderungen an der Aufbauorganisation und den Studien- und Prüfungsordnungen. Für die IT-Gruppe *Arbeitsbereich KCoIT* ergab sich als Lösung, die Organisation an das neue Datenbankkonzept anzupassen und eine sog. „Statistische Organisationseinheit (SOE)“ als die kleinste gemeinsame Verdichtungsebene von Kostenstellen, Drittmittelprojekten, Personal und Studiengängen zu schaffen [287]. Einen begrenzten Einblick auf SOEs gewährt Abb. 2.15.

10 FB Wirtschaftswissenschaft						
100000						
FB WiWi Zentral		Biblio- theken	BWL	VWL		Techni- scher Vertrieb
109000		107000	100100	100200		100300
FB-Verwaltung	Sonstige WiWi	FB Bibliothek WiWi	Betriebswirtschaftslehre	Volkswirtschaftslehre	VWL ZI	Technischer Vertrieb
109001	109099	107001	100101	100201	100202	100301

Abb. 2.15 Auflösung der institutionellen Heterogenität in der Aufbauorganisation durch SOEs;
Quelle: [287]

Da im Projekt nicht der eingangs erwähnte Fehler „neuer Hut auf alten Kopf“ gemacht wurde, stand den nicht unbeträchtlichen kurzfristig anfallenden Projektkosten ein erheblicher, sich teilweise erst langfristig auswirkender Nutzen in der Übersichtlichkeit der Aufbauorganisation, der Benutzerfreundlichkeit und der Vereinfachung der Arbeitsprozesse gegenüber. Dazu sind zu rechnen [287]:

1. die Reduzierung der Gliederungstiefe, z. B. Finanzpositionen (Kostenarten): vorher ca. 2900 Finanzpositionen, nachher 43 Ausgabe- und Einnahmearten,
2. die Schaffung mehrerer Verdichtungsebenen von der Grobansicht zum Detail, und
3. die Benennung der Ausprägungen der Verdichtungsebenen durch „Hausjargon“ sowie auf der Detailebene durch Fachtermini.

2.3.1 ETL-Prozess

Der ETL-Prozess muss sowohl *performant* sein, um Sperrzeiten beim Zugriff auf die Datenquellen zu minimieren, als auch *effektiv*, um die Qualität der Daten hinsichtlich der vorgegebenen Qualitätsziele zu sichern. Dies erfordert, dass die Daten u. a. vollständig, korrekt, aktuell und widerspruchsfrei im Data-Warehouse bereitgestellt werden. Neuere Einsatzgebiete von Data Warehouses erfordern das beschleunigte Hinzufügen von Daten, teilweise sogar in **Echtzeit** (*Real-Time Data Warehousing*) [217]. ETL richtet sich daher zunehmend auf die Minimierung der **Latenzzeit** aus, das heißt der Zeitspanne, die benötigt wird, bis die operierenden bzw. Fremddaten aus den Quellsystemen zur Nutzung bereitstehen. Hierzu ist eine häufige Durchführung und/oder Parallelisierung des ETL-Prozesses unabdingbar.

ETL setzt ein Repositorium (Metadatenbank) voraus, das insbesondere die notwendigen Datenbereinigungs- und Transformationsregeln sowie die Schemadaten als Metadaten aufnimmt und langfristig pflegt.

Dabei sind **Metadaten** Daten über Daten oder Funktionen. Metadaten beschreiben nicht nur syntaktische Aspekte der Daten, wie Bezeichner, Datentyp oder Länge, und technische Aspekte wie projekt- oder programmbezogene Angaben oder Kenngrößen zur Speichernutzung, sondern darüber hinaus die statistischen sowie semantischen Eigenschaften. Diese sind für eine fachlich korrekte Nutzung der Data-Warehouse-Daten unabdingbar. So lässt sich z. B. mit dem Meta-Attribut *Skala* verhindern, dass die Aggregatfunktionen *avg* und *sum* auf nominal oder ordinal skalierte Daten angewendet oder Personalbestandsgrößen (stock) zeitlich mittels *sum* aggregiert werden [179] (siehe Abschn. 2.4).

Die meisten ETL-Systeme enthalten das Werkzeug **Data Profiling**. Dieses ermöglicht, einfache beschreibende Statistiken zu den Tabellen einer Datenbank aus verschiedenen Sichtweisen zu ermitteln, um eine Art „Bestandsaufnahme“ der Qualitätslage vorzunehmen. Dies ist wichtig, um die Datenqualität zu verbessern, wenn Daten aus Altsystemen extrahiert werden, da deren Qualität i. Allg. nicht bekannt sein dürfte und hinsichtlich der gesteckten Datenqualitätsziele des Data Warehouses nicht ausreicht. Beispielsweise lassen sich in einer Tabelle *Kundenbestellung* der Prozentanteil Nullwerte je Attribut berechnen oder Datensätze entdecken, deren Datentyp nicht konsistent zum DW-Schema ist. Transformationsregeln müssen darauf genau abgestimmt werden, um später korrekte Operationen im Data Warehouse zu garantieren.

Zum Füllen eines Data Warehouses werden zur Homogenisierung von heterogenen Daten Transformationen verwendet, die unter dem Begriff *data migration* bzw. *data mapping* zusammengefasst werden. Eine Auswahl [21]:

1. Transformation in (de-)normalisierte Datenstrukturen
2. Erkennen und Eliminierung von Duplikaten (Objektidentifizierung, Deduplizierung)
3. Schlüsselbehandlung oder Erzeugung künstlicher Schlüssel z. B. bei uneinheitlichen internationalen Postleitzahlen
4. Anpassung von Datentypen wie bei Hausnummern kodiert als *integer* oder *string*
5. Vereinheitlichung von Zeichenketten, Datumsangaben usw. wie ‚1.2.2011‘ statt ‚2.1.2011‘
6. Umrechnung von Maßeinheiten und Skalierung, beispielsweise von l/100 km in g/mi
7. Kombination oder Separierung von Attributwerten wie Trennung von Titeln und Familiennamen
8. Anreicherung von Attributen durch Hintergrundwissen wie Zuordnung der Regionalstruktur zu PLZ
9. Berechnung abgeleiteter Aggregate mittels Variablentransformation, SQL-Aggregatfunktionen und OLAP-Operatoren

Die Benutzeroberfläche eines zum Data Profiling geeigneten Tools, hier des Data Profilers im Oracle Warehouse Builder, ist in Abb. 2.16 dargestellt. Die Werkzeugleiste erlaubt es, gewünschte Transformationen durch Drag-and-Drop zu aktivieren, beispielsweise zwei Tabellen mittels *Joiner* zu verknüpfen oder die Selektion von Datensätzen durch *Filterbedingungen setzen* zu aktivieren. Mit „Deduplicator“ in der rechten Leiste wird ein Werkzeug bereitgestellt, das es ermöglicht Duplikate beispielsweise in Adressdateien zu erkennen.

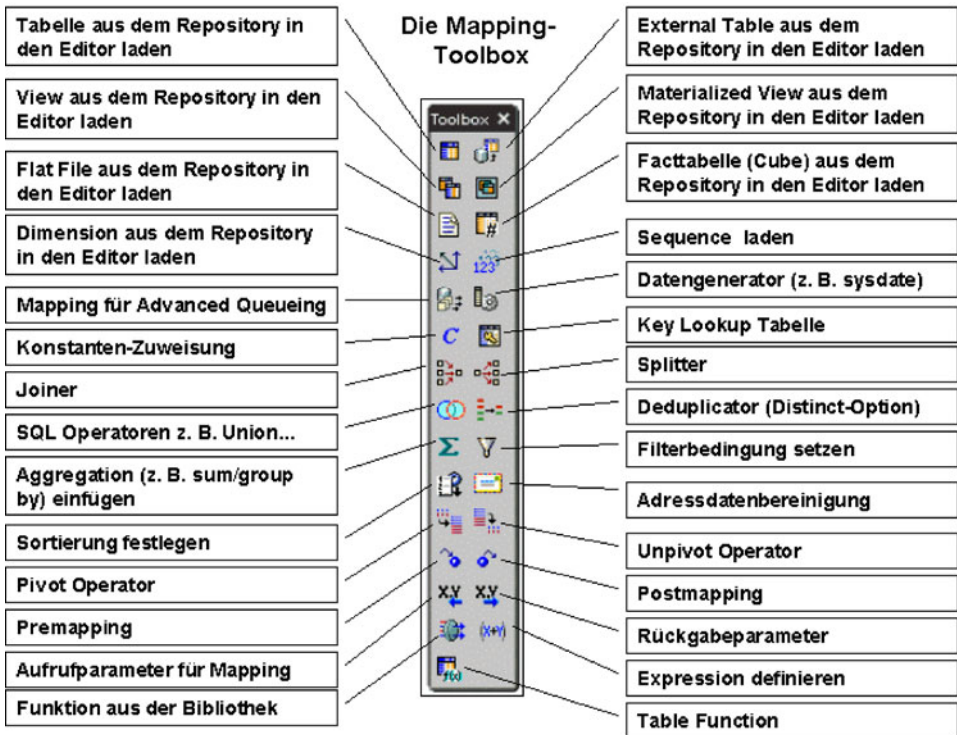


Abb. 2.16 Mapping Toolbox des Oracle Warehouse Builder (OWB)

Im Folgenden klassifizieren wir mögliche Integrationskonflikte. Diese treten beim Aufbau eines Data Warehouses i. Allg. kombiniert auf. In der Literatur unterscheidet man zwischen Schema- und Datenkonflikten [172, 222, 156].

2.3.2 Schemakonflikte

Konzeptionelle Schemakonflikte treten in vielfältiger Form auf. Sie werden durch unterschiedliche Datenmodellierung von autonomen, heterogenen Datenbanken verursacht. Grundannahme für die Auflösung ist, dass überhaupt, ein gemeinsames Schema eines Data Warehouses als Zieldatenbank existiert. In der Praxis ist dies wegen der verfügbaren Vorinformation und der Erfahrung der beteiligten internen und externen Experten gegeben. Allerdings ist die Konfliktauflösung äußerst mühsam.

1. **Namenskonflikte:** Bei Namenskonflikten liegt die Ursache bei den sich semantisch unterscheidenden Abbildungen realer Sachverhalte. Es treten sowohl Synonyme als auch Homonyme auf. Synonyme sind darauf zurückzuführen, dass semantisch äquivalente

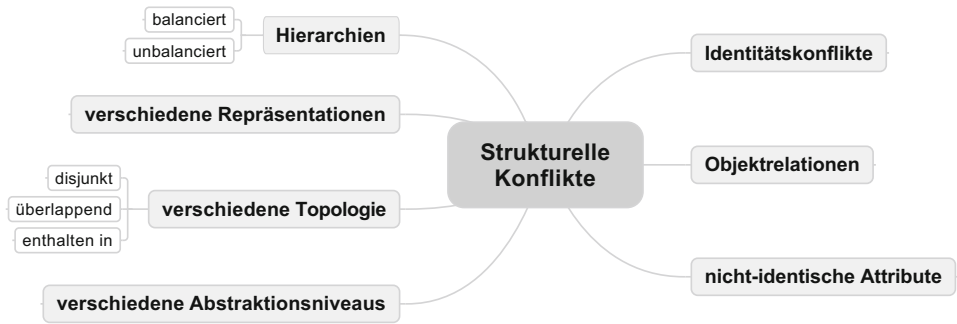


Abb. 2.17 Strukturelle Konflikte

Objekte unterschiedlich bezeichnet werden, z. B. „Mandant“ und „Klient“ oder „Artikel“, „Produkt“, „Teil“ usw. Bei Homonymen sind unterschiedliche Objekte gleich benannt, z. B. *Bank* im Sinne von „Parkbank“ oder „Postbank“.

2. **Strukturelle Konflikte:** Strukturelle Konflikte haben ihre Ursache in unterschiedlicher Modellierung ähnlicher Sachverhalte in den zu integrierenden Schemata. Man kann zwischen verschiedenen strukturellen Konflikten unterscheiden (siehe Abb. 2.17). Die Identitätskonflikte sind auf unterschiedliche Primärschlüssel zurückzuführen. Konflikte entstehen durch unterschiedliche oder fehlende Objektbeziehungen. Konflikte durch unterschiedliche Attribute gehen meist auf nicht zueinander passende Attributmengen oder fehlende Attribute zurück, während Konflikte auf unterschiedlichen Abstraktionsebenen aus Generalisierung und Aggregation innerhalb des jeweiligen Schemas resultieren.

Konflikte durch verschiedene Topologien (unterschiedliche Realitätsausschnitte) ergeben sich unter anderem bei disjunkten oder überlappenden Abbildungen von realen Sachverhalten. So sind im Zeitablauf sich langsam ändernde, nicht isomorphe Hierarchien ebenfalls Auslöser für Konflikte (siehe auch Historisierung im Abschn. 2.6.6). Mit der Verwendung von Attributen als eigenständigen Entitäten und Relationen treten weitere Konflikte durch unterschiedliche Repräsentationen auf. Man denke hier beispielsweise an das Attribut *Kundentyp* des Objekttyps *Kunde* bzw. an die Objekttypen *Kleinkunde* und *Großkunde* als Spezialisierungen vom Objekttyp *Kunde*.

3. **Konflikte durch Datenrepräsentation:** Es sind unterschiedliche Repräsentationskonflikte möglich (siehe Abb. 2.18). Wenn Daten in den Quellsystemen in unterschiedlicher Art gespeichert sind, dann kommen sie oft mit verschiedenen Datentypen und Wertebereichen vor, bilden jedoch semantisch äquivalente Attribute ab. Weiterhin treten Probleme durch Null- und Default-Werte auf, die im Quellsystem zwar zulässig sind, aber im Zielsystem erst zugelassen werden müssen. Skalierungskonflikte entstehen durch die Verwendung unterschiedlicher Maßeinheiten, Maßstabsfaktoren und Detailgrade. Bei Konflikten durch die unterschiedliche Definition von Attributen (sog. *Harmonisierungskonflikte*) liegt die Ursache in unterschiedlicher inhaltlicher Bestimmung durch

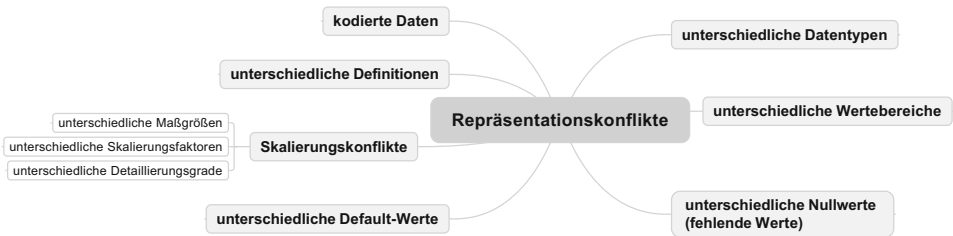


Abb. 2.18 Repräsentationskonflikte

die zuständigen Experten. So kann man durchaus unterschiedlich Beginn und Ende von Projekten definieren und damit die *Projektdauer*.

Durch den Anwender können anstelle eines vollständigen Attributnamens Abkürzungen oder Kodierungen (sog. *Aliasse*) zugewiesen werden; selbst kodierte Daten sind meist nicht einheitlich über alle Quellsysteme hinweg definiert. Man denke hierbei an die Kodierung von *Geschlecht* durch $\{0, 1\}$ bzw. $\{1, 0\}$.

2.3.3 Datenkonflikte

Bei den Datenkonflikten liegt die Ursache in den Daten selbst, die in den Quellsystemen gespeichert sind. Zur Lösung dieser Konflikte ist in der Regel zeitaufwändiges Eingreifen nötig. Es eignet sich hier der Einsatz regelbasierter Verfahren, denn durch die Verwendung von Hintergrundwissen erreicht man die angestrebte Lösung oft aufwandsgünstiger. Man kann zwischen verschiedenen Datenkonflikten unterscheiden (siehe Abb. 2.19).

1. **Veraltete oder ungültige Daten:** Diese Datenkonflikte resultieren aus nicht stattfindender Aktualisierung, wodurch Daten veralten oder sogar unbrauchbar werden. Bei der ersten Erdölkrise 1973 lag in der Bundesrepublik Deutschland nur eine fünf Jahre alte, damit technologisch veraltete Input-Output-Tabelle vor, um die Folgewirkung der Krise volkswirtschaftlich abzuschätzen.
2. **Mehrdeutigkeit von Feldern:** Steht in einem Feld ein Datum, das für mehrere Sachverhalte genutzt wird oder mehrere Bedeutungen besitzt, ergeben sich Konflikte wegen Mehrdeutigkeit. *Alter* eines Mitarbeiters kann entweder sein *Lebensalter* oder die *Dauer seiner Betriebszugehörigkeit* sein. Umsatz oder Absatz können mengen- oder wertmäßig ambivalent benutzt werden und bei Netto-Kaltniete kann Unklarheit herrschen, ob die Betriebskosten einbezogen sind oder nicht.
3. **Fehlerhafte Daten:** Diese entstehen aufgrund nicht korrekter Erfassung oder Eintragung von Daten (*Eingabefehler*), der Verwendung von sogenannten *Platzhaltern* (dummy values, defaults) oder dem gänzlichen Fehlen von Daten (*Nullwerte*). Dies kann strukturell bedingt sein, falls niemals Werte existieren können oder fallweise eintreten, sodass Werte temporär fehlen. Oftmals werden die Daten auch in verschiedener

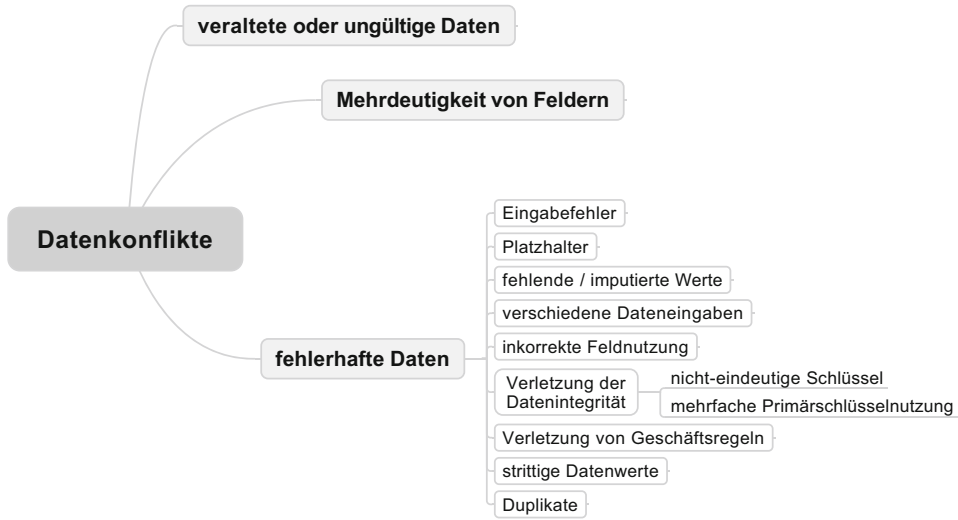


Abb. 2.19 Datenkonflikte

Form erfasst (*Formfehler*) oder es werden Felder falsch verwendet (*Anwendungsfehler*). Inhaltliche Widersprüche (*semantische Inkonsistenz*), *Verletzungen der* (Objekt- bzw. referentiellen) *Integrität* oder von *Geschäftsregeln*, sowie nicht erkannte *Duplikate* führen ebenso zu Datenfehlern. Besondere Aufmerksamkeit erfordern unplausible oder *zweifelhafte Daten*. Sie liegen beispielsweise im einfachsten Fall vor, wenn ein minderjähriger Privatkunde einen sehr teuren Gebrauchtwagen der Luxusklasse erworben haben soll.

Ein Beispiel für einfache Konfliktauflösung durch Transformation geht aus der Abb. 2.20 hervor. Diese Art der Konversion eines Datentyps in einen anderen kommen häufig vor, beispielsweise wie hier, wenn eine Zeichenkette (String) in eine Zahl umgewandelt werden muss.

Da die Datenintegration stets viel fachliches Hintergrundwissen über die im Data Warehouse zu speichernden Daten und über die verfügbaren Datenquellen verlangt, ist überwiegend eine manuelle Spezifizierung der Transformationen unabdingbar. Folglich ist der ETL-Prozess sehr zeitaufwändig und damit teuer. Es ist daher nicht verwunderlich, dass die Anbieter von marktgängigen Datenbank-Managementsystemen (DBMS) mittels intensiver Forschung und Entwicklung diesen Arbeitsprozess in Teilen zu automatisieren versuchen.

Ein erster Versuch in Richtung *Halbautomatisierung* ist das in [222] konzipierte ETL-Tool SAETL, (siehe Abb. 2.21). Die linke Seite des Diagramms zeigt die üblichen Komponenten des ETL-Werkzeugkastens wie *Server/Laufzeitsystem*, *ETL-Entwurfskomponente*, *ETL-Werkzeugkasten* und *Repositorium* zusammen mit den überwiegend systemtechnisch benötigten Metadaten. Das rechte Teildiagramm ergänzt diese Komponenten um ein *Analysewerkzeug* für interne und externe Datenquellen, ein Entwurfswerkzeug für Benutzer-

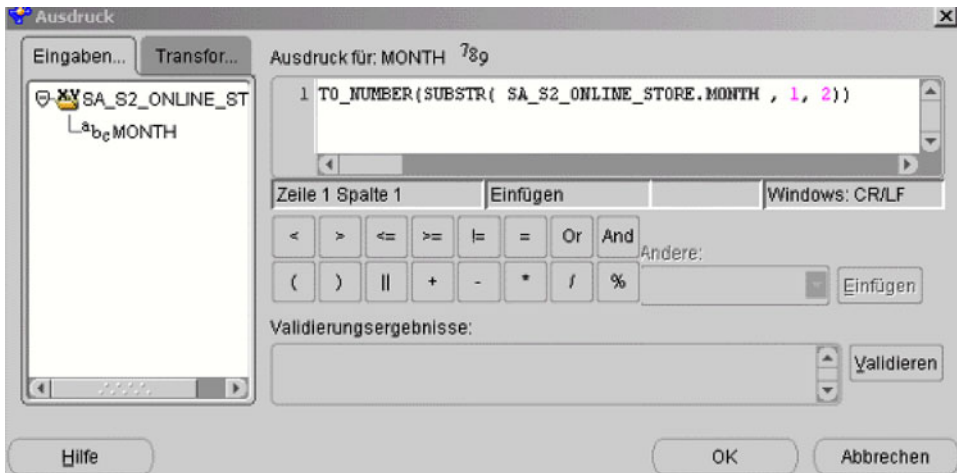


Abb. 2.20 Bedieneroberfläche zur Konfliktauflösung mittels Transformationen von Datentypen im OWB

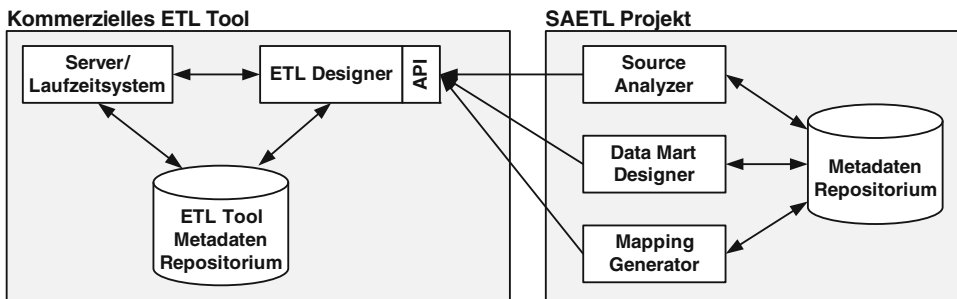


Abb. 2.21 SAETL-Architektur: Integration mit kommerziellem ETL Designer [222]

sichten und ein Transformations-Werkzeug, das massiv von technischen, semantischen und statistischen Metadaten eines (erweiterten) *Repositoriums* Gebrauch macht.

Der Query-Prozessor einer solchen Architektur würde beispielsweise prüfen und verhindern, dass auf *Wachstumsraten* von Absatzdaten (x_1, x_2, \dots, x_n) die SQL-Aggregatfunktion *avg*, d.h. $\bar{x} = \sum_{v=1}^n x_v$, angewendet werden kann. Da das Transformationswerkzeug für das neu generierte (verhältnisskalierte) Attribut *Wachstumsrate_Absatz* im Repositorium als *statistischen Datentyp* = ‚Wachstumsrate‘ festhält und für *avg* nur der statistische Datentyp = ‚absolut/kardinal skaliert‘ zulässig ist, wird verhindert, statistischen Unsinn zu generieren. Zulässig wäre dagegen in diesem Fall, das geometrische Mittel $\bar{x} = \sqrt[n]{x_1 x_2 \dots x_n}$ anzuwenden, da ihm der Datentyp ‚Wachstumsrate/Verhältnisskala‘ im Repositorium bei den Aggregatfunktionen zugeordnet ist.

Weiterführende Literatur An erster Stelle legen wir dem interessierten Leser das Buch von Leser und Naumann (2007) ans Herz [182]. Es behandelt in hinlänglicher Tiefe und didaktisch geschickt die Themen *Autonomie*, *Heterogenität* und *Verteilung*, *Schema-* und *Metadatenmanagement* sowie die *Datenintegration*. Mehr auf den Telekommunikationsaspekt von Verteilung und Integration gehen die Autoren Abeck et al. (2003) in „Verteilte Informationssysteme: Integration von Datenübertragungstechnik und Datenbanktechnik“ ein [3].

2.4 Datenqualität

Unter **Qualität** versteht man nach der DIN-Norm zum Qualitätsmanagement den „Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt“ [80]. **Datenqualität** kann man in Anlehnung an [291] auch kurz mit „Eignung der Daten zur Nutzung bei gesteckten Verwendungszielen“ umschreiben.

Die Qualität von Daten wird demzufolge bestimmt durch die Gesamtheit der ihnen zugeordneten Qualitätseigenschaften bzw. -merkmale. Die vorgegeben Qualitätsziele leiten sich aus dem Anwendungskontext ab. Dieser ergibt sich aus dem gewünschten Informations- und Wissensstand. Dazu ist es sinnvoll, sich den Zusammenhang zwischen Daten, Information und Wissen in Erinnerung zu rufen (siehe Abb. 2.22) [143, 1].

Wissen wird hier aus pragmatischer Sicht interpretiert und reichert Planungs-, Entscheidungs- und Controllingsituationen mit den dafür notwendigen fachlichen Informationen an. *Information* selbst setzt wiederum auf vorhandenen, korrekten und fachlich interpretierbaren Daten auf. Dabei können bzw. müssen die Qualitätsanforderungen aus den Analysezielen der Endanwender abgeleitet werden.

Diese Vorgaben in einzelne Qualitätsmerkmale herunterzubrechen ist angesichts der Vielzahl qualitativer Aspekte nicht einfach. Deren Anzahl liegt bei etwa 75 Kriterien [122]. Eine Vielzahl von Klassifikationsschemata sind dabei hilfreich. Im Weiteren folgen wir [127]. Um zu einer systematischen, „intelligenten“, *nicht* „totalen“ Datenqualitätssicherung zu gelangen, sind vorweg formal-abstrakte Definitionen, Messvorschriften und Schwellenwerte für Annahme-/Ablehnungsentscheidungen von Daten unabdingbar.

2.4.1 Kenngrößen der Qualitätsmessung

Wie aus Abb. 2.23 ersichtlich ist, lässt sich die Datenqualität in vier Hauptkriterien unterteilen [122, 127]:

1. *Glaubwürdigkeit*,
2. *Nützlichkeit*,

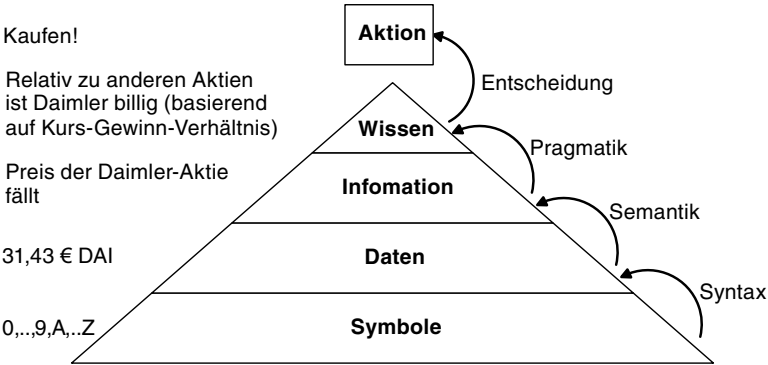


Abb. 2.22 Daten, Information und Wissen [1]

- 3. Interpretierbarkeit und
- 4. Schlüsselintegrität.

Die recht heterogene Hauptkategorie **Glaubwürdigkeit** repräsentiert Qualitätseigenschaften der Daten, die Realitätsnähe, Fehlerfreiheit und Widerspruchsfreiheit zu den Metadaten sowie die Vertrauenswürdigkeit der Datenquellen gewährleisten.

In der zweiten, recht gut messbaren Hauptkategorie **Nützlichkeit** werden Qualitätskriterien zusammengefasst, die insgesamt die Verwendbarkeit und Zweckdienlichkeit sichern. In diesem Sinn können Daten *nutzlos* sein, wenn sie unvollständig („lückenhaft“), noch

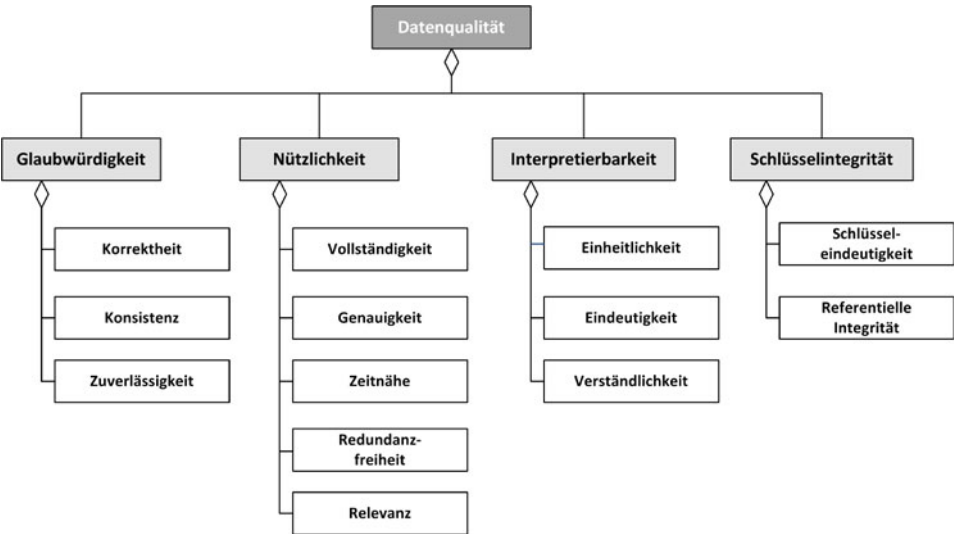


Abb. 2.23 Datenqualitätskriterien nach Hinrichs [127]

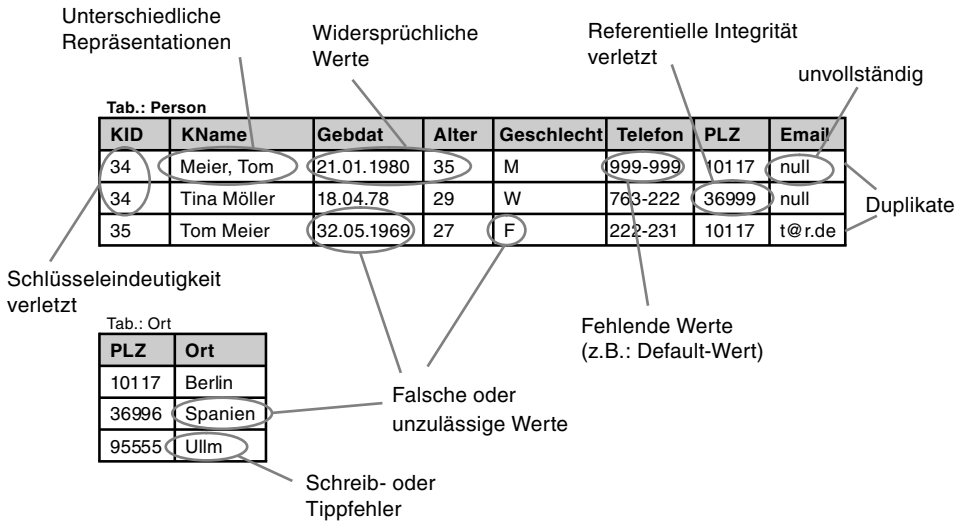


Abb. 2.24 Beispiele von Datenfehlern

nicht revidiert trotz Revisionsnotwendigkeit, nicht ausreichend detailliert (beispielsweise Quartals- statt Monatsdaten), nicht mehr aktuell, also veraltet, und irrelevant sind.

In der dritten, überwiegend gut messbaren Hauptkategorie **Interpretierbarkeit** werden Qualitätskriterien zusammengefasst, die syntaktische und semantische Eigenschaften der Daten betreffen, sodass die Daten semantisch einheitlich („harmonisiert“), eindeutig und verständlich sind.

Die letzte Kategorie **Schlüsselintegrität** sichert die eindeutige Identifizierung aller Entitäten als *Schlüsseleindeutigkeit* der Kandidatenschlüssel und die Existenzabhängigkeit solcher Objekte bei wechselseitigen Beziehungen als *referentielle Integrität* ab. In der Datenbanktheorie wird diese Integritätsart als Teil umfassender Integritätsbedingungen (constraints) angesehen und vom Datenbankmanagementsystem (DBMS) mit verwaltet und bei der Datenpflege verwendet. Beispiel für die Schlüsselintegrität ist die Forderung, dass eine Kunden-Nr. nur einmal vergeben wird. Die referentielle Integrität ist verletzt, wenn noch ein Kundenauftrag existiert, der zugehörige Artikel aber schon längst aus dem Sortiment gestrichen ist.

Wenn eine Datenbank eines dieser Qualitätsmerkmale nicht einhält, kann dies verheerende Auswirkungen für Datenanalysen im Rahmen von OLAP- oder Data-Mining-Aktivitäten haben, die auf diesen Daten aufsetzen (Schlagwort „Garbage in – Garbage out“).

Um dies zu verdeutlichen, sind in Abb. 2.24 repräsentative Fehler entsprechend der obigen Klassifikation eingebaut. Die Abbildung erweckt vordergründig den Eindruck, dass die Datenqualität einzig eine Frage von Fehlern in den Daten selbst ist. Dem ist jedoch nicht so, da Datenqualitätsmängel von Einzel- und Mehrquellenproblemen herrühren und durch Fehler auf Instanz- und Datenbankschemaebene verursacht sein können. Diesen Zu-

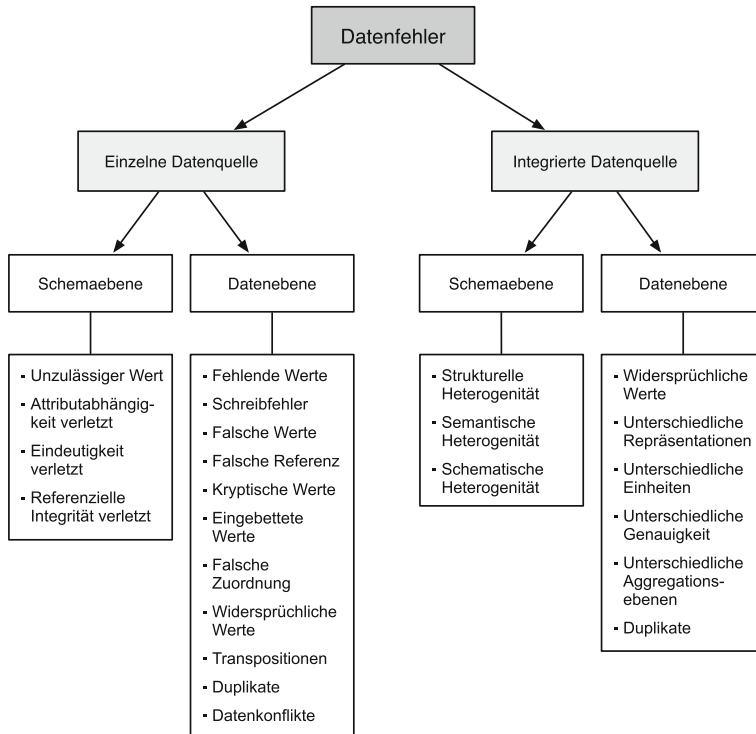


Abb. 2.25 Datenqualitätsmängel abhängig von Ebene und Quelle [182]

sammenhang macht die Klassifikation in Abb. 2.25 deutlich [182, 250]. Auf der Ebene der Blattknoten in diesem Baumdiagramm wird erneut die Komplexität der Datenqualitäts-sicherung sichtbar.

Getreu dem Motto „You can’t control what you can’t measure“ [75] ist es nötig, geeignete Messgrößen (Qualitätskenngrößen) bereitzustellen, um Soll-Ist-Qualitätsabweichungen aufzudecken (Lokalisationsproblem) und ggf. zu korrigieren („zu reparieren“).

Im Folgenden folgen wir den Ausführungen in [181]. Bei den Sollwerten handelt es sich um die gewünschten, vorab zu definierenden Qualitätsniveaus. Weiterhin nutzt die Praxis der Datenqualitätssicherung sog. *Schwellen-* oder *kritische Werte*, die eine Datenbereinigungsaktion auslösen, wenn die betreffende Kenngröße den Schwellenwert unter- bzw. überschreitet.

Die Verfahren lassen sich im Wesentlichen zurückführen auf:

1. *semantisch-technische Prüfung* der realen Daten gegen die zugehörigen Metadaten,
2. *statistische Prüfung* durch explorative Datenanalyse anhand einfacher, deskriptiver und statistischer Maßzahlen und
3. *Konsistenzprüfung* realer Daten hinsichtlich des Einhaltens der Geschäftsregeln und Validierungsregeln.

Geschäftsregeln (engl. Business Rules) erfassen dabei *Tatbestände*, Geschäftsprozesse und betriebswirtschaftliche Beziehungen aller Art.

Beispiel Es muss stets gelten: $\text{sum}(\text{Monatgehälter aller Mitarbeiter je Abteilung}) \leq \text{monatliches Abteilungsbudget}$. Absatz (d_t), Produktionsrate (p_t) und Lagerstände (I_t, I_{t-1}) müssen die Lagerbestandsgleichung, $I_t = I_{t-1} + p_t - d_t$, die sich auf die Periode $(t-1, t]$ bezieht, erfüllen.

Wenden wir uns zuerst der Gruppe *Glaubwürdigkeit* von Qualitätsindikatoren zu. Eine erste Definition von *Korrektheit* sollte jede Abweichung zwischen dem Beobachtungswert x_i und dem zugehörigen unbekannten, aber tatsächlichen Wert μ_i für ein gegebenes Objekt O_i messen. Beispielsweise passt der Name „Peter Schmidt“ nicht zu der Anrede „Frau“ in einem deutschen Werbebrief, d. h. die betreffende Geschlecht-Vorname-Regel ist verletzt: $\text{Vorname}(\text{Kunde}, \text{„Peter“}) \Rightarrow \text{Anrede}(\text{Kunde}, \text{„Herr“})$. *Korrektheit* der Daten würde dagegen vorliegen, wenn „Peter“ und „Herr“ gemeinsam in der Anschrift zusammen mit dem Nachnamen *Schmidt* auftreten würden.

Eine Alternative zu obiger Vorgehensweise ist die Berechnung von (prozentualen) Häufigkeiten, die in der Praxis der **Datenprofilerstellung** (engl. Data Profiling) ganz überwiegend verwendet werden. In der Datenbanktheorie werden die zu einem Kollektiv gehörenden Häufigkeiten im Rahmen der Abfrageoptimierung entgegen ihrer Bedeutung in der Statistik als *Histogramm* bezeichnet. Deren Vorteil liegt darin, dass *Häufigkeiten* kardinal skaliert und unabhängig von der Skala des zugrundeliegenden Daten- oder Messraums sind. Ist der Datenraum metrisch (kardinal) skaliert, so steht ein ganzes Bündel unterschiedlich geeigneter Ähnlichkeitsmaße für die Qualitätsmessung zur Verfügung [175].

Ausgangspunkt für die *Ermittlung von Datenprofilen* ist die Festlegung bzw. Kenntnis des Wertebereichs eines jeden Attributs. Diese Angaben sind üblicherweise im Repositorium hinterlegt. Das Attribut A habe den Wertebereich $\text{range}(A) = \{a_1, a_2, \dots, a_{|\text{range}(A)|}\}$. Hierbei bezeichnet $|\text{range}(A)|$ die Kardinalität (Anzahl der Elemente) des Attributs A . Sei $\chi : \text{range}(A) \rightarrow \{0, 1\}$ eine charakteristische Funktion mit $\chi_A(a) = 1$, falls $\text{cond}_A(a)$ wahr ist und 0 sonst, wobei $a \in \text{range}(A)$ ist.

Falls beispielsweise $\text{range}(\text{Geschlecht}) = \{m, w\}$ ist, ist für das Attribut $A \equiv \text{Geschlecht}$, $\text{cond}_A(0) = \text{falsch}$, da $a = 0$ offensichtlich kein Element von A ist. Dann wird die Kenngröße *prozentuale Häufigkeit korrekter Werte*, $q(A)$, von Attribut A in Tabelle T mit Anzahl Tupeln $n = |T|$ durch folgende Formel gegeben:

$$q(A) = \frac{\sum_{i=1}^n \chi_A(a_i)}{n} \quad (2.1)$$

Die Kenngröße **Konsistenz** hat mehrere Facetten. Zunächst will man damit messen, ob die Daten zur Realität widersprüchlich sind. Hinrichs [127] schränkt den Begriff auf den Vergleich mit technischen Metadaten ein. Dazu zählen Datentyp-, Feldlängen- und Wertebereichskonsistenz.

Beispiel In kommerziellen Finanzbuchhaltungssystemen wäre der Datentyp *float* für Umsatzwerte fehlerhaft, da wegen buchhalterischer Normen der Typ *decimal* vorgeschrieben ist, der genau zwei Stellen bei allen Zwischenrechnungen hinter dem Komma berücksichtigt.

Wir greifen die *Wertebereichskonsistenz* eines Attributs A heraus. Beispielsweise könnte im Repository das Attribut *Geschlecht* mit $\text{range}(A) = \{m, w\}$ enthalten sein. Falls ein Datensatz gefunden wird, bei dem statt „w“ das Zeichen „f“ oder der String „Mann“ oder „weiblich“ usw. auftritt, zeigt die zugehörige charakteristische Function $\chi(a)$ den Wert 0 an. Folglich setzen wir

$$q_{\text{range}}(A) = \frac{\sum_{i=1}^n \chi_{\text{range}(A)}(a_i)}{n} \quad (2.2)$$

wobei

$$\chi_{\text{range}(A)}(a) = \begin{cases} 0 & \text{falls } a \notin \text{range}(A) \\ 1 & \text{falls } a \in \text{range}(A) \end{cases} \quad (2.3)$$

Zum anderen kann der *Konsistenzbegriff* auf das Beseitigen von semantischen oder statistischen Widersprüchen sowie das Einhalten von in Datenbanken eingebetteten Restriktionen (*constraints*) erweitert werden [176]. Operative und auch aggregierte Daten müssen **Geschäftsregeln** und **Bilanzgleichungen** erfüllen, die als Spezialfälle von sog. **Edits** (Validierungsregeln) aufgefasst werden können. So gilt beispielsweise in der Debitorenbuchhaltung die

- Geschäftsregel 1: „Falls eine Kundenforderung innerhalb von 14 Tagen bezahlt wird, dann ist 3 % Skonto zulässig“ und die
- Geschäftsregel 2: „Summe aller Sollbuchungen = Summe aller Habenbuchungen“.

Edits sind Validierungsregeln für Daten [38]. Sie können vom Regeltyp

1. *einfach*
2. *logisch*
3. *probabilistisch*
4. *arithmetisch* oder
5. *statistisch/fuzzy*

sein [170].

Der Regeltyp *einfach* liegt beispielsweise bei *Datentyp(Umsatz)* = ‚numerisch‘ vor. Die obige Skontoregel stellt ein *logisches* Edit dar, wie auch die Regel *Falls Person.Beruf* = ‚Bus-Fahrer‘, dann *Person.Lebensalter* > 21. Die Regel *Middle-Manager.Lebensalter* ≥ *Einschulungsalter* (6) + *Grundschuljahre* (4) + *Gymnasiumsahre* (8) repräsentiert ein *arithmetisches* Edit. Derselbe Edittyp gilt für die bekannte, oben bereits erwähnte *Lagerhaltungsgleichung*, die erzwingt, dass Daten aus den Abteilungen *Einkauf*, *Produktion*, *Lager* und

Verkauf gleichungskonsistent sind, d. h. die Gl. 2.4 erfüllen:

$$I_t = I_{t-1} + p_t - d_t \quad (2.4)$$

Man beachte, dass die in dieser „Bilanzgleichung“ enthaltenen Größen nicht unbedingt ex post die Lagerhaltungsgleichung erfüllen müssen. Dies kann der Fall sein, wenn *Erfassungsfehler*, *Messfehlern* oder *Schätzspielräume* beispielsweise bei der Bilanzpolitik, bei Bestandsbewertungen und Inventuren von Wichtigkeit sind.

Weiterhin spielen **partielle Informationen** eine wesentliche Rolle. So kann die Produktion p_t über Bestandsdifferenzen und Endnachfrage aus den Daten der Abteilungen *Einkauf/Lagerhaltung* und *Vertrieb* gemäß $\hat{p}_t = \Delta I_t = I_t - I_{t-1} + d_t$ abgeschätzt werden. Andererseits lässt sich die Produktion p_t durch *Stücklistenauflösung* anhand der Daten der Fertigungsabteilung und des Vertriebs berechnen. Dies erfolgt durch Multiplikation des betreffenden Endnachfragevektors d_t mit der Matrix $(I - V)^{-1}$, wobei die zugehörige, zeitinvariante Verflechtungs- oder Leontiev-Matrix (V) und die hinsichtlich der Zeilen- und Spaltenanzahl isomorphe Einheitsmatrix (I) gegeben sein müssen, d. h. $\hat{p}_t = (I - V)^{-1} d_t$. Die Komponenten des Variablenvektors p_t repräsentieren dabei die Bruttoproduktionsmengen der Endprodukte und Halbfabrikate. Die spannende Frage ist nun allerdings, wie mit möglichen Abweichungen $\Delta p_t = \hat{p}_t - \tilde{p}_t$ umgegangen werden soll. Welchen der beiden Werte nimmt man oder „mittelt“ man einfach mit oder ohne Gewichtung durch Glaubwürdigkeitsgrade, Streuungsmaßzahlen usw.? Oder publiziert man beide Werte und fügt eine Fußnote an, wie dies in der amtlichen Statistik üblich ist? Da Letzteres im Unternehmensbereich nicht statthaft ist, käme ein Vorgehen wie im Abschn. 4.6 infrage, wo sich Differenzen, die auf Schätz- und Messfehler in den Variablen beruhen, durch einen verallgemeinerten *Kleinst-Quadrate-Ansatz* (engl. General Least Squares (GLS)) ausgleichen lassen. Abschließend sei darauf hingewiesen, dass die obige, „klassische Stücklistenauflösung“ angesichts der großen Anzahl von möglichen, aber nicht immer technologisch zulässigen Varianten (beim Golf etwa 10^{60} Varianten!) und der stark fluktuierenden, d. h. stochastisch zu modellierenden, Endnachfrage in der Automobilbranche, durch die Anwendung Bayesscher Netzwerke unter Einsatz massiver Rechenkapazitäten sehr erfolgreich ersetzt werden kann [99].

Die Untergruppe **Nützlichkeit** in der Datenqualitätstaxonomie enthält als erste der fünf Qualitätskriterien *Vollständigkeit*, *Genauigkeit* und *Redundanzfreiheit*. Von den beiden anderen Kriterien *Zeitnähe* – interpretiert als Aktualität der Daten – und *Relevanz* – im Sinn von Abdeckung eines wohl definierten Informationsbedürfnisses – ist letzteres wesentlich schwieriger zu messen, da Bedürfnisse subjektiv und fallbezogen sind [127].

Vollständigkeit ermittelt den Füllgrad, in dem ausgezählt wird, wieviele Nicht-Null-Werte je Attribut A in allen n Datensätzen einer gegebenen Tabelle auftreten. Dabei wird als Qualitätsmaßzahl wiederum die relative Häufigkeit gewählt (vgl. Gl. 2.5).

$$\chi_A(a) = \begin{cases} 0 & \text{falls } a \text{ fehlt (null)} \\ 1 & \text{falls } a \text{ fehlt nicht} \end{cases} \quad (2.5)$$

Der jeweilige Wert der Maßzahl *Attributsvollständigkeit* wird durch einfaches Auszählen gewonnen. Auf der Ebene des SQL-Codes sieht dies wie folgt aus:

```
(SELECT COUNT(*) FROM table  
WHERE column is not null) /  
(SELECT COUNT(*) FROM table);
```

Es muss angemerkt werden, dass neben der *Attributsvollständigkeit* in empirischen Untersuchungen, wie sie beispielsweise das Marketing kennt, die *Objektvollständigkeit* tritt. Hier ist zu analysieren, wie viele *Entitäten* (Datensätze) wie Artikel, Kunden, Mitarbeiter etc. in den jeweiligen Tabellen fehlen.

Genauigkeit wird in verschiedenen Kontexten unterschiedlich definiert. So wird *Genauigkeit* im technischen Umfeld, beispielsweise in der Messtechnik, unterschieden in *Richtigkeit* (bias), die durch *Kalibrierung* veränderbar ist, und *Präzision*, die umgekehrt proportional zur *Varianz* σ^2 ist. Genauigkeit wird im Folgenden mit Blick auf den wirtschaftlichen Kontext reduziert auf die Prüfung, ob die Daten im gewünschten *Detaillierungsgrad* und der benötigten *Granularität* vorliegen.

Dazu gehören die *Anzahl Stellen* und die *Anzahl Nachkommastellen* numerischer Attribute vom Typ *Gleitkommazahl* (float) sowie der Hierarchieebenen-Bezug (Spezifizität). So erfordern konsolidierte Konzernbilanzen weltweit agierender Unternehmen Planungsrechnungen mit *großen* Gleitkommazahlen (double), ebenso wie bei Aktienkursen und Wechselkursen die Angabe von nur zwei Dezimalstellen nicht ausreicht. In Aufträgen reichen zwei Dezimalstellen beim ausgewiesenen Gesamtbetrag in Euro aus. Hinsichtlich der Spezifität ist unstrittig, dass beispielsweise in Produktkatalogen „Sommerbekleidung“ weniger spezifisch ist als „Strandhemd“.

Ein schwieriges Problemfeld stellt die **Redundanz** dar. Dabei geht es neben der Normalisierung von Tabellen im relationalen Datenbankmodell [83] vornehmlich um das Erkennen von *Dubletten* in einer oder mehreren Datenquellen. Darunter ist das Auftreten von mehreren Datensätzen zu verstehen, die sich auf dieselbe reale Entität bzw. Person wie „Tom Meyer“ beziehen. Man vergleiche in Abb. 2.24 den ersten und dritten Datensatz. Die algorithmische Dublettenerkennung verursacht einen quadratischen Aufwand $O(n^2)$, wenn in zwei Quellen jeweils der Länge n nach Datenpaaren mittels *vollständiger Enumeration* (kompletter Paarvergleich) gesucht wird. Es gibt jedoch effizientere Methoden [226, 227]. Die Güte der „*Deduplizierung*“ – gemessen durch die Fehlerraten der Dublettenerkennung – verschlechtert sich erheblich, wenn der Anteil Ausreißer, fehlender Werte und spezieller Datenfehler wie Übertragungs- und Tippfehler, Zahlendreher, Verwechslungen, kryptischer Werte etc. hoch ist (siehe die Datenfehlertypen unter der Rubrik *Datenebene* in Abb. 2.25).

Die **Dublettenerkennung** kennt zwei verschiedene Verfahrensgruppen. Falls ein Kandidatenschlüssel, z. B. der Primärschlüssel einer Tabelle, existiert, kann dieser benutzt werden, da er eine Entität identifiziert. Ein Beispiel ist in den USA die Sozialversicherungsnummer. Allerdings ist diese tatsächlich nicht frei von Datenfehlern, sodass Dubletten in dem Sinn existieren, dass zwei Datensätze zweier verschiedener US-Bürger eine identische Sozialversicherungsnummer haben können.

Das echte Problem der *Dublettenerkennung* entsteht, wenn globale Schlüssel nicht vorhanden sind, was üblicherweise bei der Integration von Daten aus unterschiedlichen (autonomen und heterogenen) Datenquellen der Fall ist [226, 224]. Anstelle eines identifizierenden, möglicherweise konkatenierten Schlüssels, sind „trennscharfe“, beiden Datensätzen gemeinsame Ersatzattribute zu verwenden. Das sog. *Pre-Processing* (die Datenbereinigung), das gewählte *Ähnlichkeitsmaß* zwischen Datensatzpaaren $(a, b) \in A \times B$ und das *Klassifikationsverfahren*, mit dem entschieden wird, ob ein Datenpaar (a, b) eine Dublette ist oder nicht, sind entscheidende Güteeinflussgrößen bei der Redundanzerkennung. Eine dafür brauchbare charakteristische Funktion $\chi_{\text{dup}} : A \times B \rightarrow \{0, 1\}$ zeigt eine 0 an, wenn ein Duplikat existiert, sonst eine 1:

$$\chi_{\text{dup}}(t) = \begin{cases} 0 & \text{falls Duplikat existiert für Tupel } t \\ 1 & \text{sonst} \end{cases} \quad (2.6)$$

Sei N^* die Anzahl eindeutiger Datensätze in einer gegebenen Tabelle T . Multiple Duplikate werden dabei nur einmal gezählt. Wir definieren dann als *redundanzfreie Kenngröße*:

$$q_{\text{dup}}(T) = \frac{\sum_{t \in T} \chi_{\text{dup}}(t)}{n} = \frac{N^*}{n} \quad (2.7)$$

Sei $T_0 = \{\text{Hans, Dave, Bernd, Claus, Hans, Hans, Bernd, Mike}\}$. Folglich gilt $N^* = 5$ und $q_{\text{dup}}(T_0) \approx 63\%$.

Einheitlichkeit und *Eindeutigkeit* können in ähnlicher Weise definiert werden [127]. Daten gelten als *einheitlich* („harmonisiert“), wenn die vordefinierten Formate eingehalten, und die Werte somit syntaktisch standardisiert repräsentiert werden. Klassisches Beispiel sind Datumsangaben wie TT.MM.JJJJ.

Für die *Eindeutigkeit* ist es notwendig, dass im Repository geeignete (semantische) Metadaten zur Definition und Erfassung der jeweiligen Attribute vorliegen. Man nehme nur Mehrdeutigkeiten, die entstehen können, wenn „Betriebszugehörigkeit“ oder „Pensionshöhe“ nicht präzise definiert und berechenbar festgelegt werden.

Ähnliches gilt für *Verständlichkeit*. Hier spielen für die Interpretation von Daten und den Wissenserwerb die semantischen Metadaten eine entscheidende Rolle. Dazu rechnen u. a. Definition, Version, Berechnungsart, Variablentyp, Quellenverweis, Erhebungsmethode, Maßeinheit und Skala.

Abschließend kommen wir zur **Schlüsselintegrität**. Die Schlüsseleindeutigkeit bedeutet, dass ein Kandidatenschlüssel *id* als Attribut oder (minimale) Attributkombination jeden Datensatz einer Tabelle eindeutig „identifiziert“. Die zugehörige Indikatorfunktion χ_{dist} signalisiert 0, falls mehr als ein Wert $p \in \text{range}(id)$ in T auftritt, sonst 1. So darf z. B. die Artikelnummer in den Artikelstammdaten nicht mehrfach vergeben sein. Die prozentuale Häufigkeit eindeutiger Schlüsselwerte wird definiert als

$$q_{\text{dist}}(T) = \frac{\sum_{t \in T} \chi_{\text{dist}}(t)}{n}. \quad (2.8)$$

Referentielle Integrität garantiert, dass beispielsweise bei Aufdatierungstransaktionen, die sich auf Aufträge an Lieferanten beziehen, der Fall nicht eintritt, dass ein Auftrag exis-

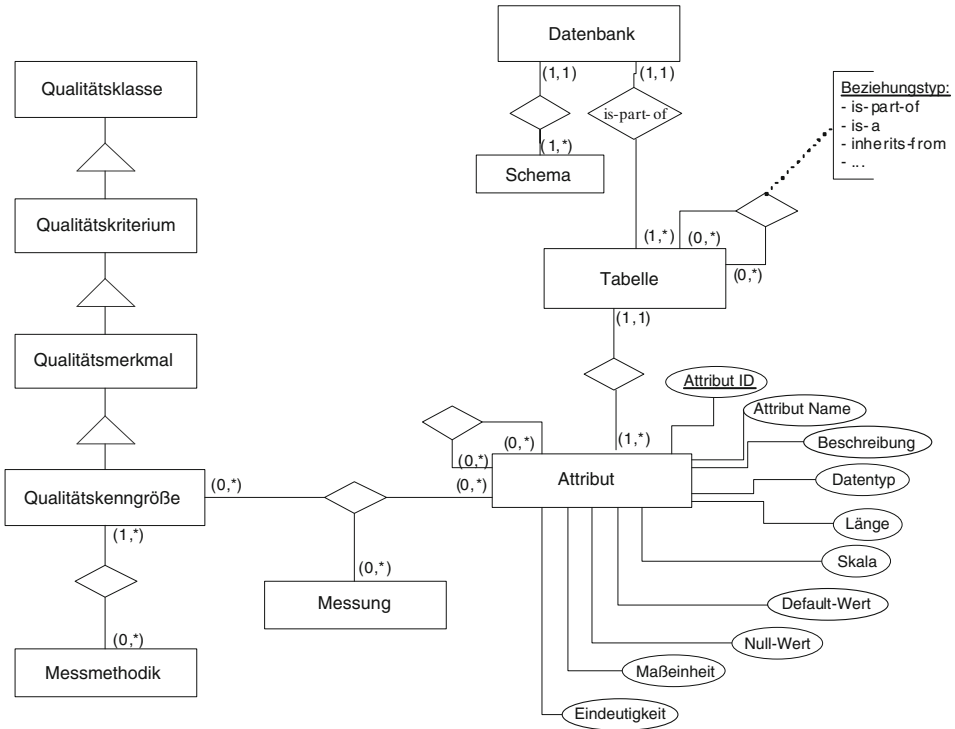


Abb. 2.26 ERM für datenqualitätsbezogene Metadaten [36]

tiert, der betreffende Lieferantenstammsatz aber bereits gelöscht ist. Diese Integrität stellt sicher, dass für jeden Wert $f \in \text{range}(F)$ eines Fremdschlüssels F einer Tabelle T nur solche Werte zulässig sind, die auch in der Menge der aktuell verwendeten und gültigen Primärschlüsselwerte id in T vorkommen. Folglich wählt man als Indikatorfunktion für korrekte Referenzen $\chi_{\text{refer}} : \text{range}(id) \cup \{null\} \rightarrow \{0,1\}$:

$$\chi_{\text{refer}}(f) = \begin{cases} 1 & \text{falls } f \notin \{\text{range}(id) \cup \{null\}\} \\ 0 & \text{sonst} \end{cases} \quad (2.9)$$

Die Maßzahl für referentielle Integrität lautet dann

$$q_{\text{refer}}(F) = \frac{\sum_{f \in \text{range}(F)} \chi_{\text{refer}}(f)}{|\text{range}(F)|}. \quad (2.10)$$

Wir schließen diesen Abschnitt mit Abb. 2.26, das den Entwurf eines Entity-Relationship-Modells für datenqualitätsbezogene Metadaten im Repository eines Data Warehouses zeigt [36].

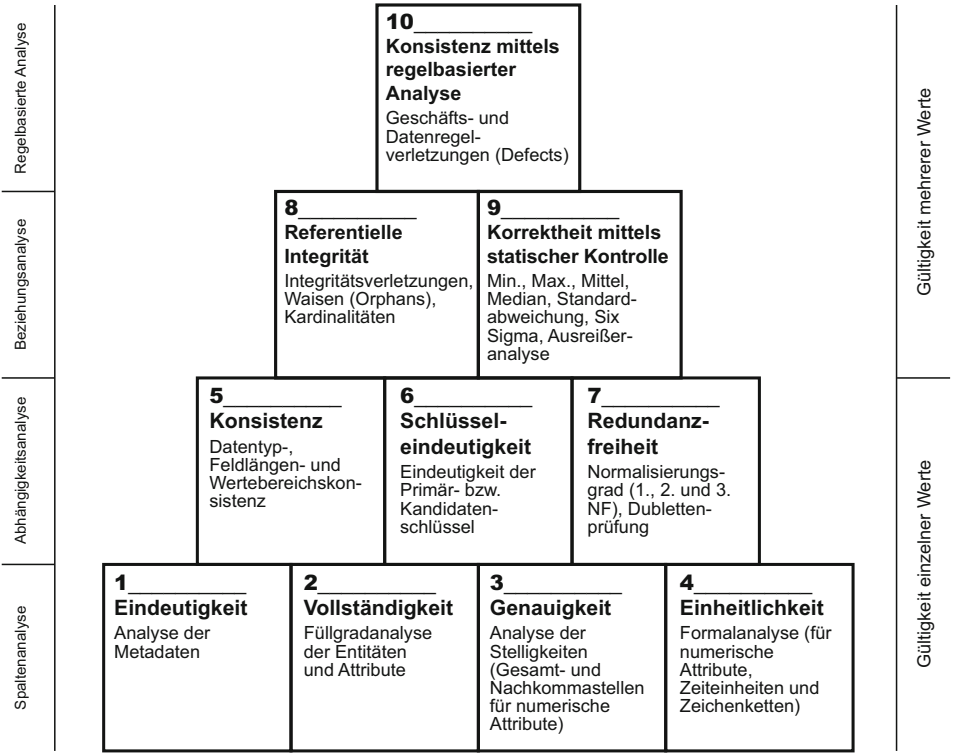


Abb. 2.27 Fachkonzeptionelle Hierarchie der Arbeitsschritte zur Datenqualitätssicherung [37]

2.4.2 Qualitätssicherungsprozess

Mit der definitorischen Festlegung von Qualitätskriterien und der Klärung der Frage nach deren Messung im betrieblichen Umfeld ist das Problem der Qualitätssicherung noch nicht gelöst. Vielmehr sind die einzelnen Schritte zur Berechnung der ausgewählten Maßzahlen in eine zweckmäßige Abarbeitungsreihenfolge (*Data-Quality-Workflow*) zu bringen. Dazu existiert eine Reihe von Vorschlägen, siehe etwa [36]. Die Grundlage für einen Prozess zur Sicherung der Datenqualität bildet eine fachkonzeptionell hierarchisch strukturierter Ablauf der in Frage kommenden Qualitätskriterien [37]. Dies ist im Diagramm in Abb. 2.27 dargestellt. Die Nummerierung in den einzelnen Kästen, beginnend mit Schritt 1 und mit Schritt 10 endend, gibt die lineare Ordnung der Arbeitsschritte wider. Die in [36] empfohlene Abarbeitung der Qualitätsprüfung erfolgt entsprechend ebenenweise von „unten nach oben“ und auf derselben Ebene von „links nach rechts“. Die Ebenen werden in der Reihenfolge *Spaltenanalyse*, *Abhängigkeitsanalyse*, *Beziehungsanalyse* und zuletzt *regelbasierte Analyse* abgearbeitet.

Es ist einsichtig, dass Analyseschritte auf derselben Hierarchiestufe je nach den betrieblichen Gegebenheiten nebenläufig ausgeführt werden können wie z. B. Genauigkeits- und

Einheitlichkeitsprüfungen. Allerdings wird auch deutlich, dass die einzelnen Ebenen *seriell* abgearbeitet werden müssen, weil der Vorgangsgraph wie folgt linear strukturiert, d. h. total geordnet ist: $\{1, 2, 3, 4\} \rightarrow \{5, 6, 7\} \rightarrow \{8, 9\} \rightarrow \{10\}$. So sollte z. B. eine regelbasierte Prüfung (10) erst dann gestartet werden, wenn u.a. Integritäts- (8) und Korrektheitsprüfungen (9) abgeschlossen sind.

Die Ursache von Datenqualitätsproblemen ist überwiegend in den operativen Systemen zu suchen. Insbesondere sind die manuelle Dateneingabe durch Mitarbeiter und Medienwechsel eine Hauptursache fehlerhafter und unvollständiger Daten. Der Slogan eines Qualitätsmanagers bringt das Entscheidende auf den Punkt: „The measurement of quality is the price of non-conformance. Do it right the first time“. Der Einsatz von Anreizsystemen für die Verbesserung der betrieblichen Datenqualität ist ein neuerer Forschungsansatz [215]. Das deutsche Management ist sich dieser Problematik erst nach der Verbreitung des Data Warehousing auf breiter Front bewusst geworden. **Data Quality Improvement** und **Data Quality Assurance** – *Assurance* ist durch *Improvement* zu unterstützen – sind jedoch nicht neu im volkswirtschaftlichen Umfeld der Statistischen Landesämter und des Statistischen Bundesamts sowie im Marktforschungsbereich der großen Unternehmen.

2.4.3 Datenqualitätsberichte

Mit Hilfe des **Data Profiling**, wie es u. a. IBM, ORACLE, SAP, SAS und Oracle als Add-On zu ihren Data Warehouse Systemen anbieten, können die einzelnen Tabellen eines Data Marts bzw. eines Data Warehouses hinsichtlich ihrer Qualität überprüft werden. Dies setzt eine benutzerfreundliche Bedieneroberfläche und einen übersichtlich gestalteten Qualitätsbericht als Ergebnis des Profiling voraus [36].

Wir stellen als Beispiel eines solchen Reports einen Screenshot von *DART* vor, der sich auf eine Ebene 2-Prüfung der *Konsistenz* von *Datentyp*, *Feldlänge* und *Wertebereich* von Kundenstammdaten bezieht (siehe Abb. 2.28). Es wird das Attribut *Geschlecht* der Tabelle „KUNDEN_STAMM“ herausgegriffen. Die *Analyse der Datentypkonsistenz* zeigt, dass der Datentyp VARCHAR2 vereinbart wurde, aber zu rund 97 % der Datentyp NUMBER dominiert. Entsprechend weist das Kontrollfeld „Qualitätsmaßnahme“ auf „Effektive Daten prüfen“ hin. Im zweiten Segment der Level-2-Analyse wird diese Angabe um das Ergebnis der *Feldlängenanalyse* ergänzt. Statt wie vereinbart Feldlänge = 10, gilt in 97 % der Fälle die Länge = 1. Auch hier wird eine Überprüfung empfohlen. Schließlich ergibt die Analyse der *Wertbereichskonsistenz*, dass die kodierten Werte 0 und 1 nur in 50 % bzw. 47 % der Tabelle auftreten. Dies liegt bei diesem Beispiel am häufigen Auftreten der Zeichenketten „Frau“ und „Herr“. Die Analyse der *Datentypkonsistenz* zeigt, dass in erheblichem Umfang unzulässige, vertippte oder schlicht falsche Werte in den Kundenstammdaten existieren und daher erheblicher *Bereinigungsbedarf* im Rahmen eines **Data Cleansing** besteht.

Weiterführende Literatur Battini und Scannapieco (2006) „Data Quality: Concepts, Methods and Techniques“ geben eine gute Einführung und erklären die wesentlichen Konzep-

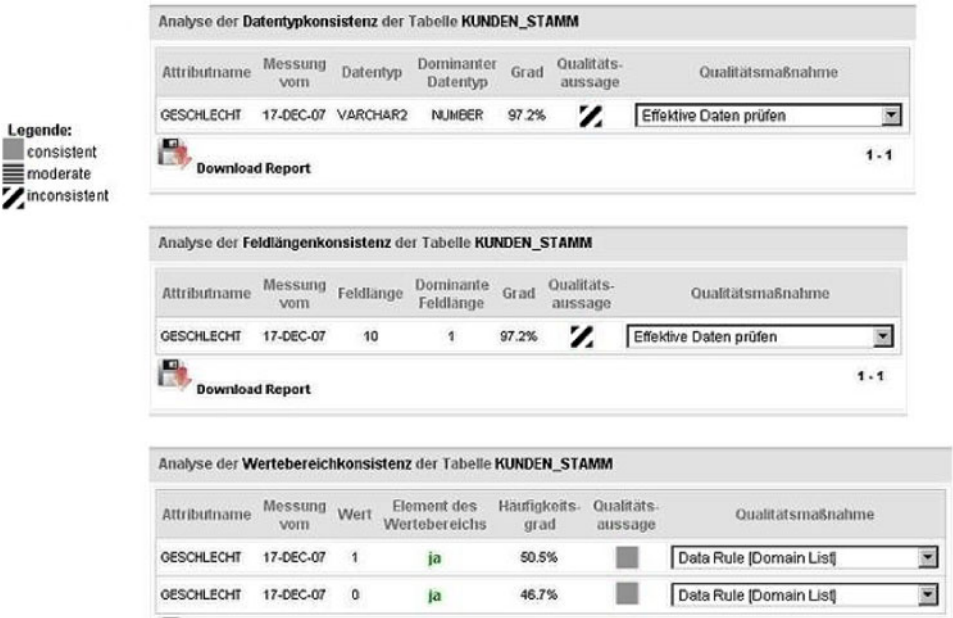


Abb. 2.28 Konsistenzprüfung von Kundendaten bzgl. Attribut *Geschlecht* [222]

te und Methoden [20]. Mehr praxisorientiert ist Apel et. al (2009) „Datenqualität erfolgreich steuern“ [11]. Als letztes empfehlen wir noch eine Dissertation von Naumann [223], die das Problem „Datenqualität“ in vorbildlicher Breite und Tiefe behandelt. Sie wurde übrigens im Jahr 2003 mit dem Preis der besten Dissertation auf dem Gebiet der Informatik von der deutschen *Gesellschaft für Informatik* (GI) ausgezeichnet.

2.5 Online Analytical Processing (OLAP)

Online Analytical Processing (OLAP) ist eine Vorgehensweise, die schnelle und flexible Ad-Hoc-Analysen von multidimensionalen Daten für einen breiten Nutzerkreis ermöglicht [64].

2.5.1 Anforderungen an OLAP Systeme

Pendse und Creeth [238] stellten 1995 fünf Anforderungen vor, die ein OLAP System erfüllen soll. Diese können mit dem Akronym **FASMI** (engl. Fast Analysis of Shared Multi-dimensional Information) zusammengefasst werden:

1. *Fast*. Die Antwortzeit erlaubt die interaktive Benutzung des OLAP-Systems. Dies ist der Fall bei Antwortzeiten von unter einer Sekunde für einfache Abfragen und maximal 20 Sekunden für selten benutzte komplexe Abfragen.
2. *Analysis*. Das OLAP-Tool bietet dem Nutzer intuitive Werkzeuge an, um sämtliche Analysen selbständig und ohne individuelle Programmierung durchzuführen.
3. *Shared*. Ein Mehrbenutzer-Betrieb auf die gemeinsamen Daten ist möglich.
4. *Multidimensional*. Die Daten sind multidimensional strukturiert.
5. *Information*. Sämtlich Daten, unabhängig von der Datenherkunft, werden dem Nutzer transparent bereitgestellt.

2.5.2 Fakten und Dimensionen

Typische Fragestellungen in einem operativen System sind z. B.:

Wie viele Stück (welche *Anzahl*) vom *Artikel XYZ* sind noch im *Lager* vorhanden? Welche *Adresse* hat *Kunde* mit *Kunden-Nr. 123*?

Diese (operativen) Abfragen (*queries*) sind meist **Punktabfragen**, die eine begrenzte Anzahl von unmittelbar transaktionsbezogenen Werten zurückgeben, beispielsweise *Lagerbestand* = 35 Stück.

Für *strategische* und *taktische* Analysen hingegen sind **Punkt-** oder **Bereichsabfragen**, die ganz überwiegend eine analytische Zielrichtung haben, wie die folgende typisch:

Wie viel *Umsatz* wurde *per Standort* mit dem *Artikel XYZ* in den *letzten drei Jahren* erwirtschaftet?

Für diese Art von Abfragen sind weder einzelne Datensätze noch Werte einzelner Attribute von Interesse, sondern es interessieren den Manager eher gut präsentierte Mengen von *aggregierten*, *gruppierten* und ggf. *sortierten Daten*. Bei den dabei entstehenden multidimensionalen Datenräumen – repräsentiert durch sog. „Datenwürfel“ – kann man zwischen Mengen von Fakten ($\mathcal{F} \subseteq \mathcal{A}$) und Dimensionen ($\mathcal{D} \subseteq \mathcal{A}$) unterscheiden, wobei für die gesamte Attributmenge $\mathcal{A} = \mathcal{F} \cup \mathcal{D}$ gilt. Eine Dimension $D \in \mathcal{D}$ ist ein spezielles Attribut, das paarweise von allen anderen Dimensionen $D' \in \mathcal{D}$ funktional unabhängig ist, d. h. $D \perp D'$ für alle $D, D' \in \mathcal{D}$ [171, 166]. Dimensionen spannen den Datenwürfel als **Kreuzprodukt** $D_1 \times D_2 \times \dots \times D_d$ auf.

Ein *hierarchisches Attribut* $H \in \mathcal{H}$ ist eine spezielle Dimension insofern, als auf ihrem Wertebereich eine partielle Ordnung definiert ist. Wir kommen weiter unten darauf zurück. Anschaulich kann man sich eine Hierarchie H als Wurzelbaum vorstellen, dessen Wurzelknoten mit *ALL* markiert wird und horizontal gruppierte Knoten eine Hierarchieebene (Verdichtungsebene) mit homogener Semantik bilden (siehe Abb. 2.29). So sind Berlin, München und Hamburg **Instanzen** der Hierarchieebene *Stadt*.

Abb. 2.29 Einfache Hierarchie der Ortsdimension [162, S. 440]

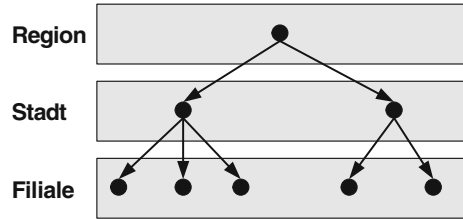
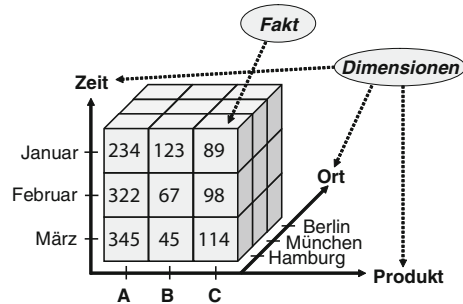


Abb. 2.30 Multidimensionaler Datenwürfel



Fakten, $F \in \mathcal{F}$, auch *summarische Attribute* genannt, sind Elemente der Faktenmenge $(\mathcal{F}, \text{Agg}_{\text{SQL}})$. Sie sind Kennzahlen oder Maßzahlen und stellen in Verbindung mit den Aggregatfunktionen, Agg_{SQL} , die SQL bereitstellt, die für analytische Fragestellungen relevanten *quantitativen* Werte oder Daten dar. Sie sind die *Zelleninhalte* eines Datenwürfels. In Abb. 2.30 könnte der Zelleninhalt den summarischen Umsatz darstellen, gruppiert nach Zeit, Ort und Produkt. Die Aggregatfunktion wäre dann SUM und Umsatz das zugehörige *summarische Attribut*.

Die zugehörigen Attribute F haben den Datentyp „numerisch“ und repräsentieren zusammen mit den gewählten Standard-SQL-Aggregatfunktionen $\text{Agg}_{\text{SQL}} = \{\min, \max, \text{count}, \text{sum}, \text{avg}\}$ die vom Management benötigten betriebswirtschaftlichen Kenngrößen (*Key Performance Indicators (KPIs)*) (siehe Abschn. 4.6). Weitere Beispiele sind $\text{sum}(\text{Umsatz})$, $\text{count}(\text{Personalnummer})$ oder $\min(\text{Deckungsbeitrag})$.

Dimensionen bilden die Teilmenge $\mathcal{D} \subseteq \mathcal{A}$. Eine Dimension $D \in \mathcal{D}$ ist deskriptiver Natur – nicht messender oder zählender Art wie Fakten – und hat einen symbolischen Datentyp. Dies heißt, sie sind *ordinal* oder *nominal skaliert* und „indizieren“ Fakten in ihrer Eigenschaft als *zusammengesetzte Schlüsselattribute*. Sie ermöglichen folglich unterschiedliche, *qualifizierende* Sichten. Sie lassen sich als die *Seiten* eines Datenwürfels ansehen. Als solche sind sie paarweise „orthogonal“ (\perp), d. h. *funktional unabhängig* voneinander [166]. Typische Dimensionen sind z. B. *Mitarbeiter*, *Produkt*, *Zeit* oder *Land*, nicht aber *Stadt* und *Filiale*, da $\text{Filiale} \rightarrow \text{Stadt}$ gilt. Die beiden ortsbezogenen Attribute, Land und Filiale sind nicht „orthogonal“ im obigen Sinn. Sie gehören zur Gruppe *hierarchische Attributmenge* \mathcal{H} . Dimensionen qualifizieren also Fakten und beantworten z. B. die Frage „Was wurde wann, wo, von wem verkauft?“.

Die **Hierarchiemenge** \mathcal{H} repräsentiert die Menge *hierarchischer Attribute*. Eine *Hierarchie* $H \in \mathcal{H}$ kann man sich konzeptionell als einen Wurzelbaum vorstellen, in dem jeder Elternknoten in semantischer Sicht eine Generalisierung der mittels Ordnungsrelation $<$ zugeordneten Kinds-knoten darstellt. Die Orts-Dimension kann beispielsweise auf Region-Stadt- oder Filial-Granularität betrachtet werden (siehe Abb. 2.29). Ein hierarchisches Attribut weist verschiedene Verdichtungsebenen auf. In mathematischer Hinsicht stellt H eine *irreflexive Halbordnung* $(H, <)$ dar, die irreflexiv, transitiv und asymmetrisch ist. Dies lässt sich in Abb. 2.29 veranschaulichen, in dem die Vorgänger-/Nachfolgerrelationen zwischen den verschiedenen Knoten (Ebenen oder *Kategorien*) $h \in H$ als Kanten (Knotenpaare) wiedergegeben werden.

2.5.3 OLAP Grundoperationen

Die *Basisdatenstruktur* für OLAP im Data Warehouse ist der *p-dimensionale Datenwürfel*, der auch als *multi-dimensionale Tabelle* bezeichnet wird, $T = (\text{Aggregatfunktion}, \text{Fakten}, D_1 \times D_2 \times \dots \times D_d)$. Er wird mittels *Kreuzprodukt* (crossing), $T = \times_{i=1}^d \text{range}(D_i)$, des Wertebereichs $\text{range}()$ seiner Menge von Dimensionen, \mathcal{D} , definiert.

Bildet speziell eine Dimension D eine (balancierte) Hierarchie H ab, also $D \equiv H$, so kann dies in algebraischer Notation mittels *Schachtelungsoperator* (engl. nesting) „/“ bei q Hierarchieebenen als $D = A_1/A_2/\dots/A_q$ notiert werden. Beispielsweise kann die Zeitdimension in einem **Betriebskalender** eines industriellen Unternehmens wie folgt aufgebaut sein: $D = \text{Schicht}/\text{Tag}/\text{Woche}/\text{Monat}/\text{Quartal}/\text{Halbjahr}/\text{Jahr}$. Man beachte, dass in einem Unternehmen wegen überregionaler Unterschiede bei Feier- und Ferientagen verschiedene Betriebskalender nebeneinander existieren können. Den Datenwürfel legt weiterhin eine Menge von Operationen und eine Menge von *Restriktionen* fest [175]. Die Menge der Operatoren (*Ops*) setzt sich aus drei Teilmengen zusammen:

1. **Transformationen** (\mathcal{T}) wie \log , \ln , \exp , sqrt , Quotienten sowie Wachstumsraten,
2. **SQL-Aggregatfunktionen** (Agg_{SQL}) wie \min , \max , sum , count , avg
3. **OLAP-Operatoren** ($\mathcal{Ops}_{\text{OLAP}}$) wie Slicing (σ_T), Dicing (π_T), Roll-up (ρ_T), Drill-Down (δ_T)

Diese Operatoren erlauben eine flexible Navigation hinsichtlich von *Aggregation* (Roll-up) und *Disaggregation* (Drill-down), Projektion und Selektion innerhalb des multi-dimensionalen Datenraums. Es kann zwischen folgenden grundlegenden **OLAP-Operatoren** unterschieden werden, die zwar mathematisch gesehen abgeschlossen, aber keinesfalls aus Anwendersicht *vollständig*, geschweige denn *minimal* sind [179]:

- *Slicing*. Der Slicing- oder Selektions-Operator $\sigma_T(\text{cond})$ – in der Statistik *Konditionierung* genannt – selektiert Tupel aus dem Würfel T gemäß der Selektionsbedingung *cond*.

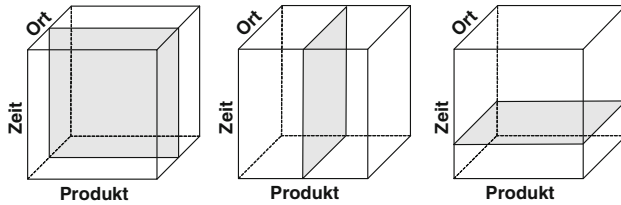
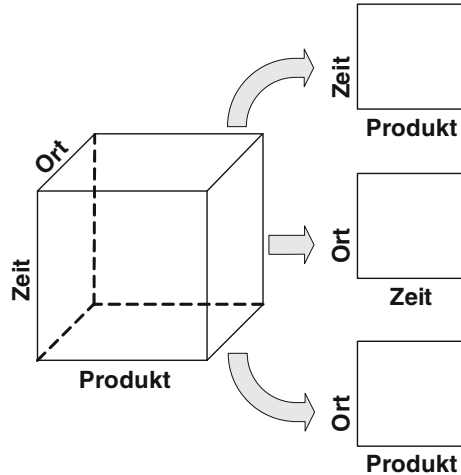


Abb. 2.31 Konditionieren (Slicing)

Abb. 2.32 Marginalisieren
(Dicing)



Der Operator $\sigma_T(\text{cond})$ legt für mindestens eine Dimension einen oder mehrere Werte fest und schneidet damit ein „Scheibe“ aus dem Datenwürfel (siehe Abb. 2.31).

Beispiel: Es sollen aus dem Datenwürfel T nur die Zellen aus dem Jahr 2010 angezeigt werden, also $T' = \text{sum}(\text{umsatz}) \text{ group by Produktgruppe} \times (\text{Jahr} = 2010) \times \text{Stadt}$ (siehe Abb. 2.31, rechtes Diagramm).

- *Dicing.* Der Dicing- oder Projektions-Operator $\pi_T(D')$ legt mehrere Dimensionen $D' \subseteq D$ des Würfels T fest, auf die projiziert werden soll und schneidet damit einen kleineren „Unter-Würfel“ (engl. sub cube) aus T (siehe Abb. 2.32). Dies wird in der multivariaten Statistik als *Marginalisierung* bezeichnet.

Beispiel: Es sollen die Verkaufsdaten per Produkt und Jahr angezeigt werden, d. h. der Teilwürfel $T' = \text{sum}(\text{umsatz}) \text{ group by Produkt} \times \text{Jahr}$.

- *Roll-up.* Der Roll-up-Operator ρ_T entspricht der *Aggregation* und geht in der Dimensionshierarchie vom Speziellen hoch zum Generellen (siehe Abb. 2.33). Unter bestimmten semantischen und statistischen Bedingungen [175], lässt sich der zugehörige Wert eines „höher gelegenen“ Knotens (Attributs), h , aus den Werten der Menge seiner „Nachfolgeknoten“, $\text{successor}(h)$ berechnen, z. B. $\text{sum}(\text{Umsatz}) \text{ group by Stadt}$ aus $\text{sum}(\text{Umsatz}) \text{ group by (Stadt/Filiale)}$.

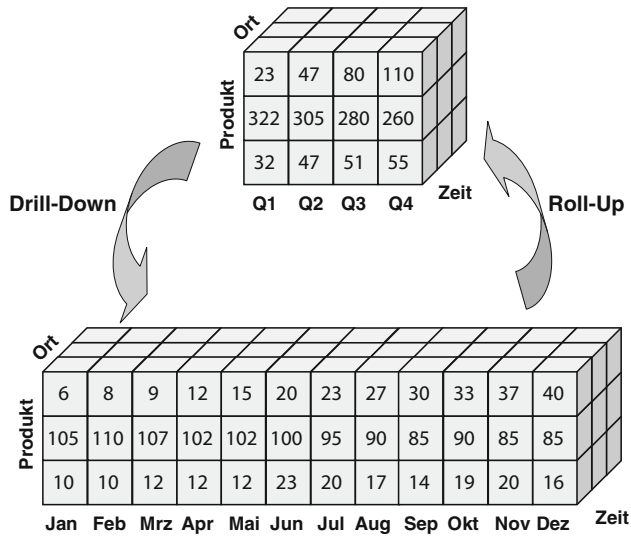


Abb. 2.33 Roll-Up und Drill-Down [162, S. 443]

Beispiel: Statt *monatlichen* Umsätzen sollen *Quartalsumsätze* ermittelt werden (siehe Abb. 2.33).

- *Drill-down.* Umgekehrt geht Drill-Down δ_T als *Disaggregation* vom Generellen hinunter zum Spezifischen in der Hierarchie (siehe Abb. 2.33). Der Operator δ_T entspricht damit dem inversen Operator zur Aggregation, wodurch die Disaggregation algorithmisch exakt berechnet werden kann, $\delta_T = \rho_T^{-1}$.

Beispiel: Erträge sollen nicht per *Produktgruppe*, sondern disaggregiert per *Produkt* ausgewiesen werden.

- *Drill-across.* Der Drill-across-Operator springt basierend auf einer Dimension D in einem Datenwürfel zu einem andere Datenwürfel. Voraussetzung ist, dass beide Datenwürfel mindestens eine Dimension gemeinsam haben.

Beispiel: Von einem Verkaufs-Cube für die Dimension *Produkte* kann man auf den Lager-Cube springen.

- *Drill-through.* Der Drill-Through-Operator erlaubt das Herausspringen aus dem Datenwürfel und die Anzeige der darunter liegenden *operativen Daten*.

Beispiel: Für eine Teilsicht eines Absatz-Cubes werden die einzelnen zugrunde liegenden Verkaufs-Transaktionen angezeigt.

- *Pivotierung.* Bei der Pivotierung kann der Nutzer die für die Analyse relevanten Dimensionen rotieren (permutieren), wobei beim Listen oder Drucken eines Würfels mit mehr als zwei Dimensionen eine Schachtelung (nesting) notwendig wird.

Beispiel: Produkt- und Orts-Dimension eines Würfels werden zum Drucken *ausgetauscht* (engl. rotiert) und die Zeitdimension (z. B. Jahr) wird innerhalb der Attributwerte von *Ort* *geschachtelt*, kurz: *Produkt* \times (*Ort*/*Jahr*).

2.5.4 Summierbarkeit

Eine notwendige Bedingung dafür, dass die kombinierte Anwendung von SQL-Aggregatfunktionen und OLAP-Operatoren bei Datenwürfeln semantisch und statistisch korrekte Ergebnisse bei einer Abfrage liefern, ist die **Summierbarkeit** bzw. allgemein die **Aggregierbarkeit** [169, 179].

Die Summierbarkeitsbedingung setzt sich aus mehreren einzelnen Teilbedingungen zusammen. Grundsätzlich muss **Disjunktheit** oder **Überlappungsfreiheit** der Klassenzuordnung von Objekten im Datenwürfel gewährleistet sein. So kann ein Kunde nicht gleichzeitig in der Klasse „Großkunde“ und „Laufkundschaft“ sein. Weiterhin muss für die Aufgliederung einer Objektmenge die **Vollständigkeit** der Zerlegung gelten. Beide Bedingungen stellen sicher, dass die Objektmenge eine **Partition** bildet. *Mehrfachzählungen* und *Unterrepräsentation* sind damit ausgeschlossen. Beispielsweise definiert die Telefonnummer eines Privatkunden keine vollständige Grund- und Auswahlgesamtheit für Marketinganalysen, da es durchaus Kunden *ohne* Festnetz- bzw. Mobilfunkanschluss gibt. Für solche Fälle sieht SQL eine Erweiterung des Wertebereichs des Attributs *Telefonanschluss* um einen „Nullwert“ vor, der in seiner Pragmatik nicht ungefährlich ist. Im wirtschaftlichen Bereich und bei Behörden wird die Merkmalsausprägung „sonstige“ benutzt.

Weiterhin muss mittels Repositorium, das die notwendigen Metadaten zu den Attributen und SQL-Aggregatfunktionen enthält, die **Typverträglichkeit** von Fakten und Aggregatfunktionen geprüft werden. Einen Mittelwert (AVG) auf *KundenNr* anzuwenden ist wegen der *Typunverträglichkeit* von Attribut und SQL-Aggregatfunktion statistisch gesehen Unsinn; denn die Skalen passen nicht zueinander wie die zugehörigen Metadaten: $Skala(KundenNr) = \text{„nominal“}$ und $Skala(AVG) = \text{„kardinal“}$ zeigen. Gleiches gilt für die Mittelwertsberechnung von (ordinal skalierten) Qualitätsstufen z. B. bei der Erstellung von Qualitätsreports. Die marktgängigen Data Warehouse Managementsysteme aller namhaften Hersteller berücksichtigen unseres Wissens derartige Restriktionen bislang nicht.

Eine wichtige Rolle für die korrekte Aggregation spielt der *semantische Datentyp* der Fakten $F \in \mathcal{F}$. So sollte im **Repositorium** für jedes Attribut ($A \in \mathcal{A}$) dessen semantischer und nicht nur technischer Datentyp als Metadaten gespeichert sein, ist es aber bei allen Herstellern von Datenbanken nicht. Diese Größen können einmal sog. *Stromgrößen* (engl. flows) sein, die *ausnahmslos* *summier-* oder *aggregierbar* mittels Agg_{SQL} sind (siehe Tab. 2.2). Dies gilt z. B. für den Neukundenzugang pro Quartal, Einnahmen, Ausgaben, Aufwand, Ertrag, Kosten (jeweils in Euro/Jahr gemessen) und Leistung (Material Einheiten/Jahr). Wir haben bei der Aufzählung dieser Attribute vom Typ „Stromgröße“ bewusst die **Maßeinheit** angeführt. Sie verdeutlicht, dass die Attribute *zeitraumbezogen* sind.

Nur *teilaggregierbar* sind dagegen *Bestandsgrößen* (engl. stocks), die *zeitpunktbezogen* sind. Beispiele sind die Attribute *Anzahl der Mitarbeiter*, *Bilanzpositionen* oder *-summen*. Diese Größen dürfen nicht entlang der Zeit-Dimension aggregiert, wohl aber können sie zur Zeitreihenanalyse herangezogen werden. Über andere Dimensionen wie Orts- oder Sachdimension, selbst zum selben Zeitpunkt, können dagegen Bestandsgrößen sehr wohl

Tab. 2.2 Typverträglichkeit für Temporale und Nicht-Temporale Aggregation [169]

	Temporal			Nicht-Temporal		
	Stock	Flow	Value-per-Unit	Stock	Flow	Value-per-Unit
Sum	Nicht ok	Ok	Nicht ok	Ok	Ok	Nicht ok
Min	Ok	Ok	Ok	Ok	Ok	Ok
Max	Ok	Ok	Ok	Ok	Ok	Ok
Count	Ok	Ok	Ok	Ok	Ok	Ok

summiert werden und liefern sachlich korrekte Ergebnisse. Die Aggregierung von entlang der Ortsdimension summierten Attributen ist Teil einer Querschnittsanalyse.

Die SQL-Aggregatfunktionen *SUM* und *AVG* sind nicht semantisch sinnvoll anwendbar, wenn der semantische Datentyp des zugrundeliegenden (summarischen) Attributs gleich „Einheitswert“ (engl. value per unit (vpu)) ist. Dies trifft u. a. auf Stückkosten, Benzinpreis je Liter oder Verbrauchseinheitsgrößen gemessen in Liter je 100 km zu. Tabelle 2.2 zeigt die temporale und räumliche Aggregierbarkeit für verschiedene Aggregationsfunktionen und semantischen Datentypen.

Weitere Gründe, die dazu führen, dass einige Aggregat-Funktionen nicht anwendbar sind, hängen von der **Skala** der jeweiligen Attribute ab. Vorsicht ist bei Nominalskalen – nur Kleiner-, Gleich- und Größer-Vergleiche sind erlaubt – geboten. Beispiele sind *Schlüssel* wie Mitarbeiter-Nummer, Produkt-Nummer, aber auch andere Attribute wie Postleitzahlen (PLZ). Bei diesem Datentyp fehlt aus semantischen Gründen die lineare Ordnung, und damit sind noch nicht einmal die SQL-Aggregatfunktionen *MIN* und *MAX* anwendbar.

Oftmals will man rekursiv höher aggregierte Fakten anhand von weniger aggregierten Fakten entlang einer Dimension rekursiv berechnen. Das Summierbarkeitsproblem spielt auch hier eine Rolle hinsichtlich der korrekten Anwendung der verwendeten Aggregatfunktionen (Agg_{SQL}) [179]. Wir unterscheiden nach [110] zwischen folgenden drei Typen von Aggregatfunktionen:

1. *Distributive* Aggregatfunktionen: Summe (*SUM*), Anzahl (*COUNT*), Maximum (*MAX*), Minimum (*MIN*). Diese können auf Partitionen angewendet werden und es existieren folglich rekursive Berechnungsformeln.
2. *Algebraische* Aggregatfunktionen: Mittelwert (*AVG*), gestutzter Mittelwert (*trimmed AVG*), Standardabweichung (*STDV*), etc. Hier kann man ohne weitere Hilfsfunktionen eine der genannten Funktionen nicht allein aus partitionierten Mengen berechnen. Beispielsweise kann man das arithmetische Mittel (*AVG*) nur über die *Hilfs-* bzw. *Reparaturfunktionen* (*SUM*, *COUNT*) berechnen [179]. Man beachte allerdings dabei, zu welchen unterschiedlichen Ergebnissen SQL beim Auftreten von Nullwerten (engl. missing values) führen kann.
3. *Holistische* Funktionen: Median, Rang, Percentile, Modalwert usw. Hierzu rechnen alle Aggregatfunktionen, die weder distributiv noch algebraisch sind. Bei diesen muss auf die Grundgesamtheit aller Fakten zurückgegriffen werden. So kann beispielsweise der

Median (siehe Halbwertszeit, 50 %-Punkt) nicht aus disjunkten und insgesamt vollständigen Teilmengen berechnet werden.

Abschließend werfen wir noch einen Blick auf hierarchische Attribute $H \in \mathcal{H}$ (siehe Abb. 2.29). Bildet H einen Wurzelbaum, ist jeder Knoten durch einen eindeutigen Pfad mit dem Wurzelknoten verbunden, so ergeben sich eindeutige (lineare) *Aggregationspfade*, wie beispielsweise in der örtlichen Relation: $H = \text{Block} \rightarrow \text{Stadtteil} \rightarrow \text{Bezirk} \rightarrow \text{Stadt} \rightarrow \text{Kreis} \rightarrow \text{Bundesland}$.

Es gibt jedoch manchmal Parallelhierarchien, wie bei den Zeitrelationen $H_1 = \text{Tag} \rightarrow \text{Woche} \rightarrow \text{Jahr}$ und $H_2 = \text{Schicht} \rightarrow \text{Tag} \rightarrow \text{Monat} \rightarrow \text{Quartal} \rightarrow \text{Jahr}$, die unterschiedliche Granularität aufweisen und deren Mengen funktionaler Abhängigkeiten verschieden sind [166]. So gilt nicht $\text{Woche} \rightarrow \text{Monat}$ oder $\text{Woche} \rightarrow \text{Jahr}$, da eine Woche am Monatsanfang oder -ende zwei Monaten zugeordnet werden und die letzte Woche eines Jahres sich auf zwei Jahre erstrecken kann. Ein solcher *Aggregationspfad* ist *unzulässig*, die Alternative dagegen *zulässig*. Korrekte Aggregation (Summierbarkeit von hierarchischen Attributen $H \in \mathcal{H}$) von Endknoten auf Zwischen- oder Wurzelknoten setzt daher auch voraus, eine vollständige, detaillierte Analyse der funktionellen und schwach funktionellen Abhängigkeiten der beteiligten Attribute (Kategorien) von H durchzuführen, um widersprüchliche Aussagen – insbesondere bei alternativen *Aggregationspfaden* – auszuschließen [166].

2.5.5 Speicherarten

Man kann zwischen den drei Speicherarten *relationales OLAP (ROLAP)*, *multidimensionales OLAP (MOLAP)* und *hybrides OLAP (HOLAP)* unterscheiden.

- **ROLAP** benutzt relationale Datenbanken, um die Zellen eines OLAP-Würfels zu speichern. Die Haupteigenschaften sind: Geringe Performanz, keine Redundanzen, skalierbar.
- Beim **MOLAP** wird der OLAP-Würfel in einem speziellen proprietären Format gespeichert. In diesem werden Aggregate einzelner Dimensionen und Teil-Würfel vorberechnet gespeichert. Dies ermöglicht kurze Antwort-Zeiten. Nachteile sind die redundante Datenhaltung (Einzelzellen und Aggregate von Zellen), hoher Speicherbedarf und die Investition in spezielle proprietäre Technologien.
- **HOLAP** stellt einen Kompromiss zwischen ROLAP und MOLAP dar. Nur die aggregierten Daten werden in MOLAP gespeichert. Die einzelnen Fakten werden in relationalen Tabellen vorgehalten.

Weiterführende Literatur Einen guten Einstieg in OLAP bietet erneut das Buch von Bauer und Günzel [21]. Es sind hier die Kapitel 3 „Phasen des Data Warehousing“, speziell der Absatz zu „Analysephase“ zu erwähnen. Vom historischen Standpunkt aus gesehen ist aus heutiger Sicht der Beitrag von Pendse und Creeth [238] über „The OLAP Report: Succeeding with On-line Analytic Processing“ interessant.

2.6 Multidimensionale Datenmodellierung

Multidimensionale Datenmodellierung ist der Entwurf eines *logisch-konzeptionellen* (sog. semantischen) Datenmodells für einen oder mehrere Datenwürfel. Dazu stehen verschiedene Modellierungssprachen wie ERM, UML, OOMD, ME/R oder ADAPT zur Verfügung.

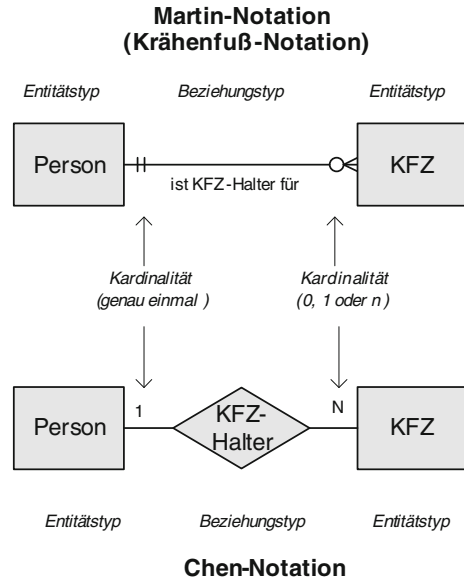
2.6.1 Multidimensionale Modellierungssprachen

Für die semantische Datenmodellierung wurden im Laufe der Zeit diverse visuelle Modellierungssprachen entwickelt. Die mit Abstand wichtigsten sind hierbei das **Entity-Relationship-Modell (ER-Modell)** [61] und das **UML-Klassendiagramm** [233]. Ein ER-Diagramm stellt ein konzeptionelles Datenmodell dar und abstrahiert von der konkreten technischen Implementierung. Es fasst gleichartige Entitäten wie z. B. die einzelnen Personen *Hans Maier* und *Robert Müller* zu dem *Entitätstypen* „Person“ zusammen. *Entitätstypen* können mit *Beziehungstypen* verbunden sein. So können Personen KFZ-Halter von Autos sein. Beziehungstypen haben eine Kardinalitätsangabe, die festlegt, wie viele Entitäten mindestens und höchstens an einer Beziehung beteiligt sein können (z. B. kann eine Person Halter von keinem, einem oder mehreren KFZs sein. Ein KFZ hat genau einen Halter). Für die visuelle Darstellung von ER-Diagrammen gibt es verschiedene Notationsformen. Abbildung 2.34 zeigt zwei häufig verwendete: die Chen-Notation und die Martin-Notation (Krähenfuß-Notation).

Neben den allgemeinen Sprachen wie ER-Modellierung oder UML, wurden auch diverse spezielle Modellierungssprachen für die *multidimensionale* Datenmodellierung entwickelt. **Multidimensionale Entity-Relationship-Modellierung (ME/R-Modellierung)** [260] erweitert das ER-Diagramm um drei neue Elemente: Faktenrelation, Dimensionsebene und hierarchische Beziehung. **Object Oriented Multidimensional Model (OOMD)** [303] ist eine multi-dimensionale Erweiterung von UML. **Application Design for Analytical Processing Technologies (ADAPT)** [47] basiert auf einer nicht-konventionellen Modellierungssprache und bietet eine große Anzahl von verschiedenen Modellierungselementen. So gibt es Symbole für Würfel, Dimension, Formel, Datenquelle, verschiedene Dimensionstypen (wie aggregierende, partitionierende, Kennzahlen- oder sequentielle Dimension), Dimensionselemente (wie Hierarchie) und Beziehungstypen. *ADAPT* verbindet die semantische, logische und physikalische Ebene der Datenmodellierung in einer Sprache. Die große Anzahl von Symbolen und die Verquickung von semantischer, logischer und physikalischer Datenmodellierung erfordert jedoch einen hohen Lernaufwand und führt zu einer großen Komplexität der so erstellten Diagramme [299].

Einerseits bieten multidimensionale Modellierungssprachen spezielle Elemente für die multi-dimensionale Modellierung an und sind dadurch ausdrucksstärker als klassische Modellierungssprachen. Andererseits verursacht die Einführung einer neuen Modellie-

Abb. 2.34 Entity-Relationship Diagramm in der Martin-Notation (*oben*) und Chen-Notation (*unten*)



rungssprache hohen Lernaufwand und etabliert eine Hürde für Personen, die in der traditioneller Datenmodellierung, d. h. ER-Modellierung, geschult sind [210]. Moody und Kortink [210] argumentieren, dass eine neue multi-dimensionale Modellierungssprache unnötig ist und multi-dimensionale Modellierung sehr gut mit ER-Modellen funktioniert.

Wir stimmen diesem Urteil zu und modellieren im Folgenden mit ER-Diagrammen. Wir nutzen die Martin-Notation (Krähenfuß-Notation) weil diese zu kompakteren Diagrammen führt als die Chen-Notation. Abweichend von der klassischen ER-Modellierung visualisieren wir in einigen Diagrammen Entitätstypen, die Faktentabellen sind, als Würfel, um den Unterschied von Fakten und Dimensionen für den Leser besser zu verdeutlichen.

2.6.2 Star-Schema

Das **Star-** oder **Stern-Schema** besitzt eine zentrale Faktentabelle und für jede Dimension genau eine Dimensionstabelle (siehe Abb. 2.35). Der Name „Star“ oder „Stern“-Schema rührt daher, dass das Schema wie ein Stern aussieht. Die Kardinalität zwischen der Faktentabelle und einer Dimensionstabelle ist $n : 1$, d. h. ein Fakt (z. B. Umsatz: 200.000 Euro) wird jeweils durch eine Dimensions-Ausprägung für jede Dimension beschrieben (z. B. Zeit: 10.10.2011, Filiale: Schönebergerstr. 12, Berlin, Produkt: Bücherstützen). Eine Dimensions-Ausprägung (z. B. Produkt: Bücherstütze) ist mit 0, 1 oder n Fakten verbunden.

Abbildung 2.36 in Verbindung mit Abb. 2.37 zeigt ein Beispiel eines Star-Schemas für die Absatzanalyse. Die Faktentabelle repräsentiert Kennzahlen zum Absatz wie *Umsatz* und *Menge*. Die vier Dimensionstabellen beschreiben die beiden Absatzkenngrößen bezüglich

Abb. 2.35 Star-Schema mit einer Fakten-Tabelle und vier Dimensions-Tabellen [209]

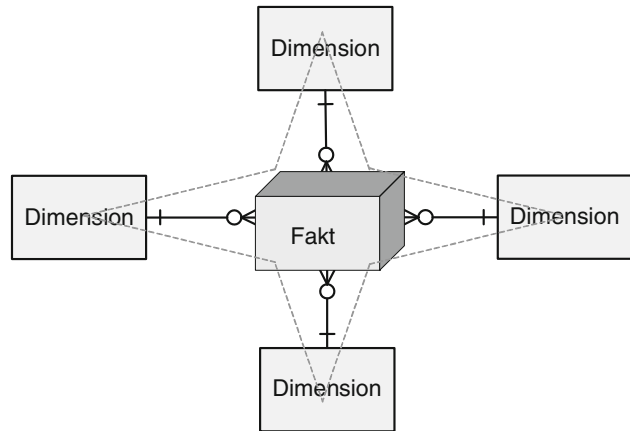
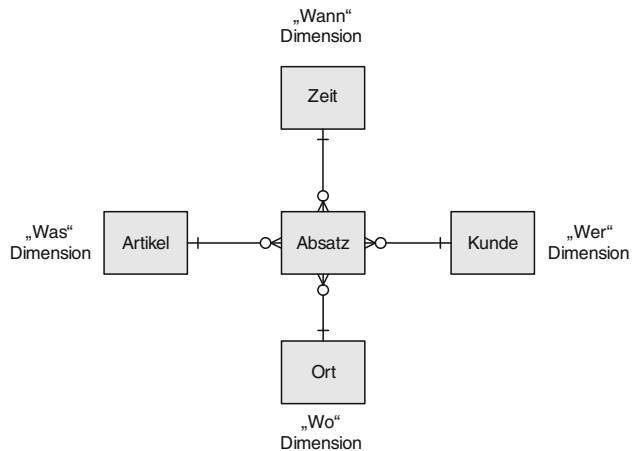


Abb. 2.36 Star-Schema für Absatzanalysen [209]



der Orts- und Zeitdimension sowie der zwei Sachdimensionen. Sie ermöglichen es, Fragen zum „wo“ (Ortsdimension), „wann“ (Zeitdimension), „was“ (Artikeldimension) und „wer“ (Kundendimension) zu beantworten.

Ist als Speicherstrategie *ROLAP* vorgesehen (siehe Abschn. 2.5.5), ist ein relationaler Tabellenentwurf erforderlich. Abb. 2.37 zeigt den Tabellenentwurf für das Beispiel von Abb. 2.36. Das Star-Schema wird dabei wie folgt in Tabellen abgebildet:

- Die **Fakten-Tabelle** enthält einen *zusammengesetzten* (konkatenierten) *Primärschlüssel* bestehend aus den einzelnen Primärschlüsseln aller vier Dimensionstabellen. Weitere Attribute der Fakten-Tabelle sind ein oder mehrere Kennzahlen (z. B. Umsatz oder Absatzmenge), die von dem konkatenierten Primärschlüssel voll funktional abhängig sind.
- Jede Dimension wird genau auf eine **Dimensions-Tabelle** abgebildet. Die Dimensionstabellen enthalten einen Primärschlüssel (PK) für die niedrigste gewünschte Aggrega-

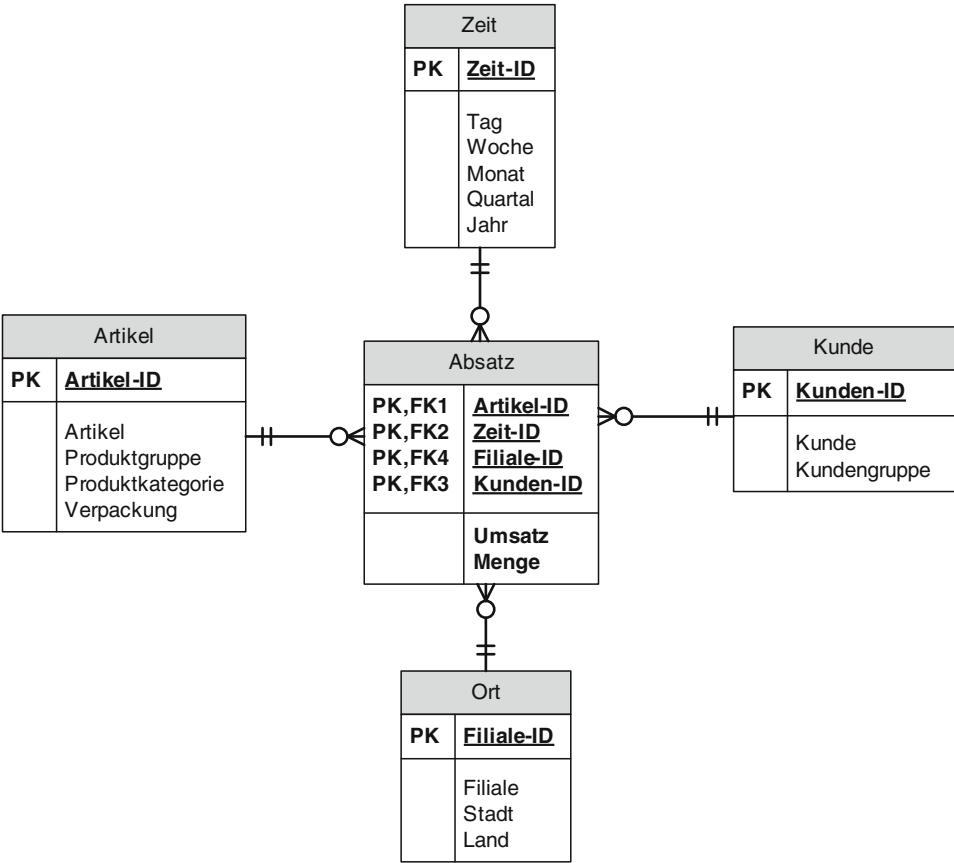


Abb. 2.37 Tabellenentwurf des Star-Schemas

tionsstufe (feinste Granularität). In Abb. 2.37 sind das die Primärschlüssel *Filiale-ID* für die Ortsdimension bzw. *Artikel-ID* für die Artikeldimension. Hierarchische Attribute werden innerhalb einer Dimensionstabelle gespeichert [21, S. 217]. Für die Ortsdimension *Land/Stadt/Filiale* sind dies die Kategorien *Filiale*, *Stadt* und *Land*.

Dimensionstabellen sind nicht normalisiert und weisen darum Redundanzen auf [21, S. 218]. So ist z. B. in der Tabelle *Artikel* das Attribut *Produktgruppe* funktional abhängig von *Artikel*, d. h. $Artikel \mapsto Produktgruppe$. Diese Redundanz wird im Star-Schema aus Performanzgründen bewusst in Kauf genommen [137], um die Anzahl aufwändiger Tabellenverknüpfungen bei *Abfragen* zu reduzieren. Da sich die Zugriffsform ganz überwiegend bei OLAP-Daten auf „Lesen, Anfügen und Verdichten“ beschränkt, bereiten Update-Anomalien keine großen Probleme, falls alle notwendigen *Restriktionen* zur Konsistenzerhaltung im Data Warehouse mittels *Trigger* integriert sind.

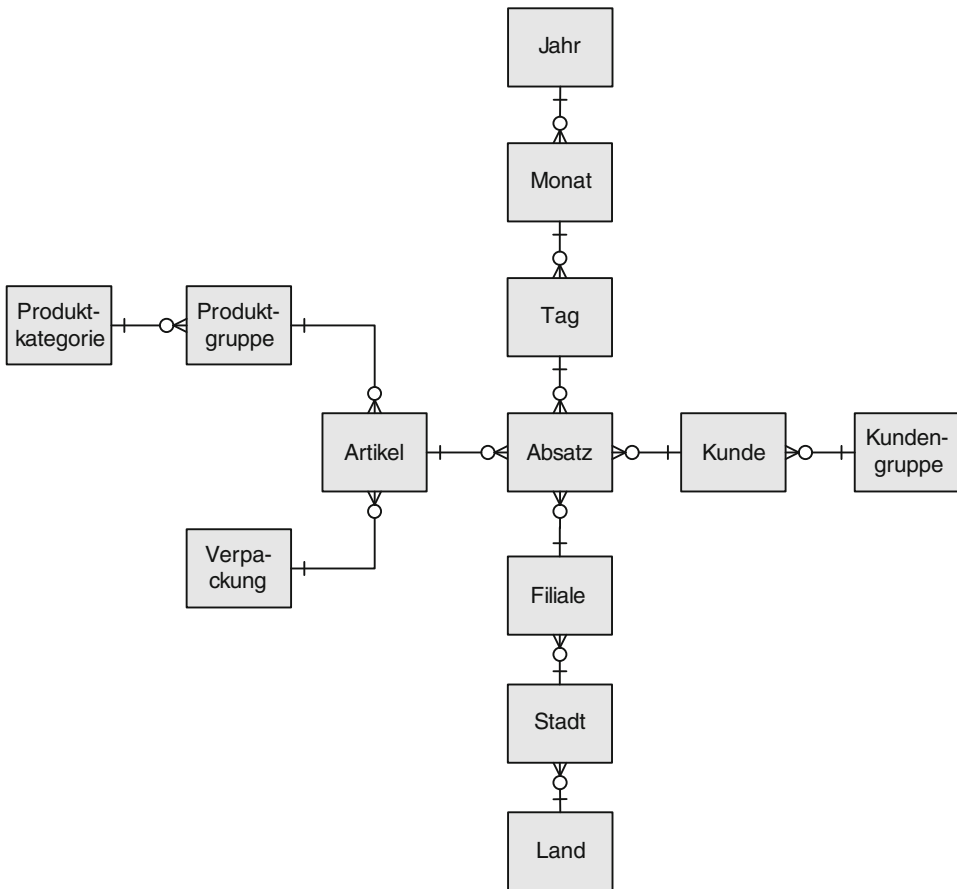


Abb. 2.38 Snowflake-Schema

Vor- und Nachteile Die Vorteile des Star-Schemas sind die Einfachheit des Schemas, die geringe Anzahl von Dimensionstabellen und die geringe Anzahl von Join-Operationen, da keine Dimensionshierarchien verbunden werden müssen [143, S. 64]. Die Nachteile liegen in der Größe und Redundanz der Dimensionstabellen und der damit eventuell verursachten langen Antwortzeit bei sehr großen Dimensionstabellen mit vielen Hierarchiestufen [143, S. 64].

2.6.3 Snowflake-Schema

Der Name „Snowflake“ oder „Schneeflocken“-Schema resultiert aus der Form des Diagramms (siehe Abb. 2.38), welches an eine Schneeflocke erinnert. Im Snowflake-Schema werden die im Star-Schema vorhandenen Redundanzen in den Dimensionstabellen aufgelöst und die Hierarchie-Ebenen mit jeweils eigenen Tabellen modelliert.

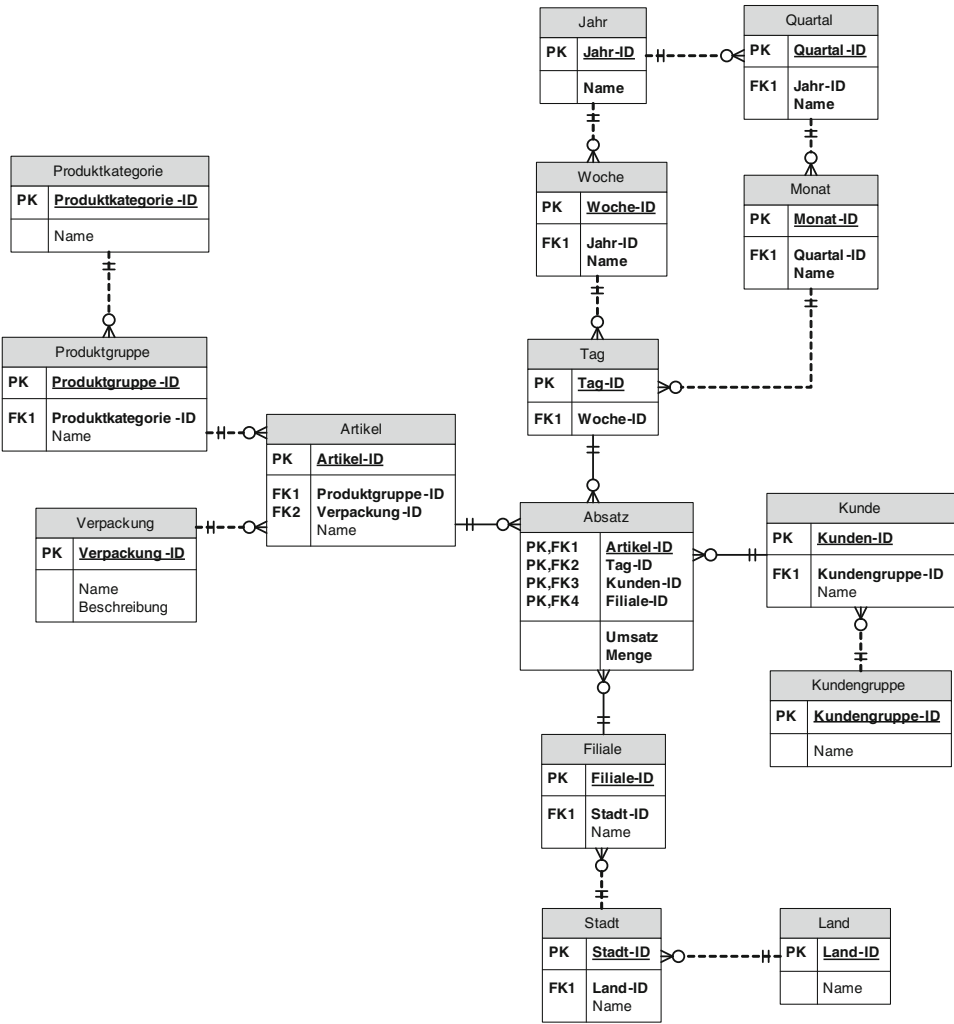
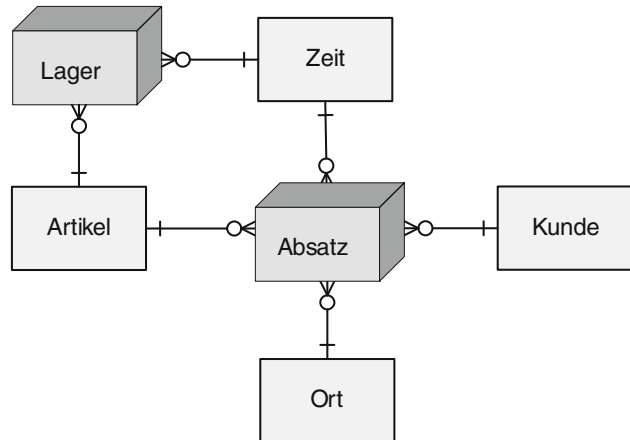


Abb. 2.39 Snowflake Schema Tabellenentwurf [21, S. 216]

In unserem vorigen Beispiel wurden im Star-Schema die Hierarchie-Ebenen der Ortsdimension in einer Tabelle gespeichert. Im Snowflake-Schema sind diese in separate Tabellen aufgeteilt (siehe Abb. 2.39).

Für die Tabelle *Artikel* werden die vorhandenen Klassifikationsebenen *Artikel*, *Produktgruppe*, *Produktkategorie* und *Verpackung* in extra Tabellen aufgeteilt. Dabei sind zwei parallele Hierarchie-Ebenen vorhanden: *Artikel* \mapsto *Produktgruppe* \mapsto *Produktkategorie* und *Artikel* \mapsto *Verpackung*. Die obere Hierarchie-Ebene (stärkere Verdichtung) ist mit der unteren Hierarchie-Ebene (weniger starke Verdichtung) mit einer 1-zu-n-Relation verbunden.

Abb. 2.40 Galaxie-Schema mit zwei Fakten-Tabellen *Absatz* und *Lager*



Vor- und Nachteile Ein Vorteil des Snowflake-Schemas ist die Beseitigung der Redundanzen in den Dimensionstabellen und damit verbundenen Problemen wie z. B. Update-Anomalien. Ein Nachteil ist die größere Anzahl der Join-Operatoren, die die abhängigen Dimensionstabellen verbinden müssen. Die Dimensionstabellen werden jedoch oft wegen Performance-Gründen nicht vollständig normalisiert. Man spricht dann im englischen von „slightly normalized dimensions“ [143, S. 65].

2.6.4 Galaxie-Schema

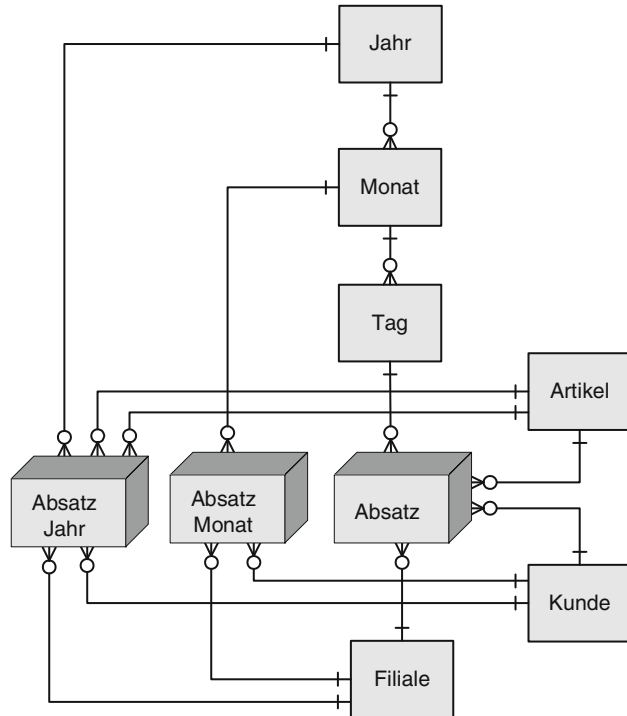
Das Galaxie-Schema enthält mehr als eine Faktentabelle (siehe Abb. 2.40) [145] [21, S. 220]. Die Faktentabellen haben einige jedoch nicht alle Dimensionen gemeinsam. Das Galaxie-Schema repräsentiert somit mehr als einen Datenwürfel (Multi-Cubes). Die Datenwürfel sind jedoch durch gemeinsame Dimensionen verbunden und ermöglichen dadurch den Drill-Across (siehe Abschn. 2.5.3) von einem Würfel zum anderen.

Beispiel In Abb. 2.40 nehmen wir an, dass ein Zentrallager für die Artikel vorhanden ist. Im Zentrallager werden Artikel nicht für bestimmte Filialen oder Kunden reserviert. Folglich ist das *Lager* unabhängig von *Kunde* und *Filiale* und hat als Dimensionen nur *Artikel* und *Zeit*. Als Kennzahl ist für das *Lager* nur die Menge der Artikel zu einem Zeitpunkt relevant, nicht aber der Umsatz.

2.6.5 Fact-Constellation-Schema

In den Fakten-Tabellen werden die Fakten (z. B. Umsatz) in einer geringen Verdichtung vorgehalten (z. B. pro Tag, Filiale und Artikel). Sollen aggregierte Auswertungen erstellt

Abb. 2.41 Fact Constellation
Schema mit drei Fakten-Tabel-
len: *Absatz*, *Absatz-Monat* und
Absatz-Jahr



werden (z. B. Umsatz pro Quartal), so werden die Fakten *on-the-fly* verdichtet. Im Fact-Constellation-Schema werden aus Performanz-Gründen stattdessen zusätzliche Summationstabellen erstellt, die dadurch eine schnellere Auswertung erlauben (siehe Abb. 2.41). Ein Nachteil ist jedoch, dass durch die zusätzlichen Summationstabellen das Schema und der ETL-Prozess unübersichtlicher werden [143, S. 64].

2.6.6 Historisierung

In einem Data Warehouse werden hauptsächlich Daten in die Faktentabellen hinzugefügt (z. B. neue Absatzzahlen). Weniger häufig werden auch neue Daten in den Dimensionstabellen angefügt (z. B. neue Artikel oder Filialen). Eine weiterer Fall ist, dass Dimensionsausprägungen irrelevant geworden sind, da z. B. bestimmte Artikel nach einem Saisonwechsel nicht mehr hergestellt werden. Dann dürfen die Dimensionsausprägungen in den Dimensionstabellen *nicht* gelöscht werden, da noch historische Daten in der Faktentabelle existieren und mit diesen verbunden sind.

Alle gerade beschriebenen Fälle sind unproblematisch für die Auswertung, da sie nicht *struktureller Art* sind. Wenn hingegen Dimensionsausprägungen verändert werden, kann es zu falschen Auswertungen von historischen Daten kommen.

Abb. 2.42 Update-Verfahren überschreibt vergangene Dimensionsangaben [143, S. 66]

Verkäufer-ID	Name	Abteilung
123	Müller, Robert	Turbinen
124	Maier, Oliver	Motoren
125	Schmidt, Peter	Turbinen
123	Müller, Robert	Motoren

Beispiel: Ein Mitarbeiter im Verkauf wechselt die Abteilung. Falls die Abteilungszuordnung einfach überschrieben wird, werden auch alle vergangenen Umsätze nicht mehr der ursprünglichen, sondern der neuen Abteilung zugeordnet (siehe Abb. 2.42). Will man die Abteilungsumsätze vergangener Jahre miteinander vergleichen, so würde dieses Vorgehen ein falsches Ergebnis liefern.

Diesem Effekt der langsam verändernden Dimensionen (engl. *Slowly Changing Dimensions*) kann durch eine geeignete Historisierungsstrategie begegnet werden. *Historisierung* ist von der *Archivierung* und dem *Backup* abzugrenzen [143, S. 66]. **Archivierung** speichert alte nicht mehr relevante Daten an einem anderen Ort und löscht diese in der primären Datenbank. **Backup** ist die redundante Speicherung von Daten, um bei einem Systemausfall diese wiederherstellen zu können. Archivierung und Backups sind für alle Systeme, also auch für Data Warehouses, erforderlich. *Historisierung* ist kein Ersatz für Backups und Archivierung.

Es kann zwischen verschiedenen Historisierungsstrategien unterschieden werden [143, S. 66] [146, S. 95]:

- **Update-Verfahren.** Das Update-Verfahren überschreibt einfach den aktuellen Dimensionseintrag mit den aktuellen Daten, d. h. auf eine Historisierung wird verzichtet (siehe Abb. 2.42). Die beschriebenen Probleme bei der Analyse der historischen Daten werden in Kauf genommen [143, S. 68].
- **Delta-Historisierung.** Ein Methode die Historie die Dimensionsausprägungen zu konservieren, ist das Einfügen von Gültigkeitsintervallen [143, S. 70]. Eine Abfrage kann mit Hilfe der Gültigkeitsintervalle für jeden beliebigen Zeitpunkt die relevanten Dimensionsausprägungen ermitteln (siehe Abb. 2.43).
- **Delta-Historisierung mit Current-Flag.** Für die meisten Auswertungen ist nur die aktuelle Dimensionsausprägung relevant. Um die Abfragen zu vereinfachen wird das Dimensionsschema darum oft mit einem Gültigkeitsfeld (Current-Flag) erweitert, das anzeigt ob die Zeile aktuell ist (siehe Abb. 2.44) [143, S. 71].

Verkäufer-ID	Name	Abteilung	Gültig-von	Gültig-bis
123	Müller, Robert	Turbinen	01.01.2005	31.12.2009
124	Maier, Oliver	Motoren	03.04.1997	
125	Schmidt, Peter	Turbinen	12.04.2008	
123	Müller, Robert	Motoren	01.01.2010	

Abb. 2.43 Delta-Historisierung mit Gültigkeitsintervallen [143, S. 70]

Verkäufer-ID	Name	Abteilung	Gültig-von	Gültig-bis	Current
123	Müller, Robert	Turbinen	01.01.2005	31.12.2009	0
124	Maier, Oliver	Motoren	03.04.1997		1
125	Schmidt, Peter	Turbinen	12.04.2008		1
123	Müller, Robert	Motoren	01.01.2010		1

Abb. 2.44 Current-Flag zur Markierung aktueller Datensätze [143, S. 69]

Dimensions-Tabelle

Verkäufer-ID	Name	Abteilung	Gültig-von	Gültig-bis	Current
123.01	Müller, Robert	Turbinen	01.01.2005	31.12.2009	0
124.01	Maier, Oliver	Motoren	03.04.1997		1
125.01	Schmidt, Peter	Turbinen	12.04.2008		1
123.02	Müller, Robert	Motoren	01.01.2010		1

Fakten-Tabelle

Verkäufer-ID	Kunden-ID	Jahr	Umsatz
123.01	2	2009	1.020,00 €
124.01	4	2009	5.523,00 €
125.01	2	2009	7.550,00 €
123.02	3	2010	3.023,00 €
124.01	2	2010	1.020,00 €
125.01	4	2010	5.523,00 €

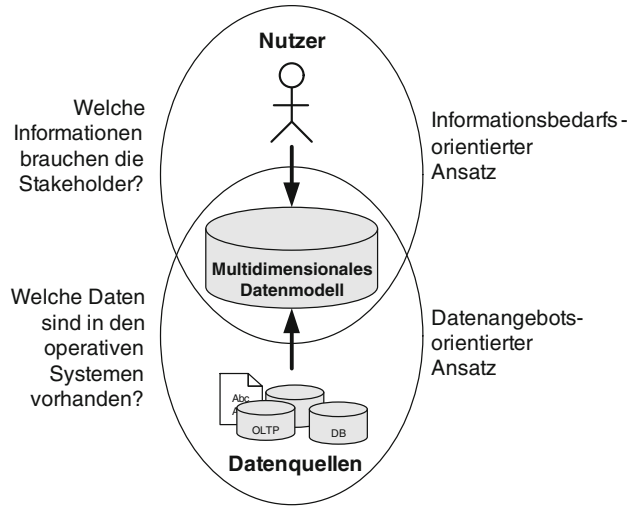
Abb. 2.45 Delta-Historisierung mit künstlicher Schlüsselersweiterung [143, S. 69]

- **Delta-Historisierung mit künstlicher Schlüsselersweiterung.** Bei dieser Historisierungsvariante wird der Primärschlüssel der Dimensionen aus zwei Teilen zusammengesetzt: Der erste Teil (z. B. Verkäufer-ID 123) wird um einen künstlichen Zähler erweitert, der die Version angibt (z. B. 123.02 für die zweite Version des Verkäufers 123). Sobald eine neue Aktualisierung die Dimension ändert, wird eine neue Version mit inkrementiertem Zähler zur Dimensionstabelle hinzugefügt. Ab diesem Zeitpunkt werden neue Fakten mit dem korrespondierenden aktuellen Dimensionsschlüssel hinzugefügt [143, S. 70]. Dadurch haben zeitlich zusammengehörige Fakten und Dimensionen dieselbe Schlüsselersweiterung (siehe Abb. 2.45).

2.6.7 Vorgehensweisen für die multidimensionale Modellierung

Man kann zwischen zwei Vorgehensweisen für die multidimensionalen Modellierung unterscheiden. Es sind dies ein Datenangebots-orientierter und ein Informationsbedarfs-orientierter (Datennachfrage-orientierter) Ansatz (siehe Abb. 2.46). In der Praxis werden aber auch oft Mischformen eingesetzt bzw. beide Verfahren iterativ kombiniert.

Abb. 2.46 Datenangebots-orientierter und Informationsbedarfs-orientierter Ansatz



2.6.7.1 Datenangebotsorientierter Ansatz

Beim **angebotsorientierten Ansatz** startet man mit den Daten, die in den operativen Systemen vorhanden sind. Die Transformation des bestehenden operativen Datenmodells in ein multidimensionales Modell erfolgt dabei in vier Schritten [209, 210]:

1. Klassifikation der Entitäten
2. Konzeptioneller (High Level) Star-Schema-Entwurf
3. Detailliertes Design der Faktentabellen
4. Detailliertes Design der Dimensionstabellen

1. Klassifikation der Entitäten

Die Entitäten des bestehenden operativen Unternehmens-Datenschemas werden in einer von drei Kategorien klassifiziert [209, 210, 208]:

- **Transaktions-Entitäten.** Diese Entitäten (Bewegungsdaten) speichern detaillierte Datensätze von Geschäftsereignissen wie z. B. Aufträge, Lieferungen, Zahlungen, Hotelbuchungen, Versicherungsansprüche oder Flugreservierungen. Dies sind die zentralen Geschäftsvorfälle für das Unternehmen.
- **Komponenten-Entitäten.** Diese Entitäten (Stammdaten) sind direkt mit einer Transaktions-Entität durch eine 1:n Relation verbunden. Sie beschreiben das Geschäftsereignis und beantworten Fragen über das „wo“, „wann“, „wer“, „was“, „wie“ und „warum“ des Ereignisses.
- **Klassifikations-Entitäten.** Dies sind Entitäten die mit einer Komponenten-Entität durch eine Kette von 1:n Relationen verbunden sind. Diese definieren die Hierarchien im Datenmodell.

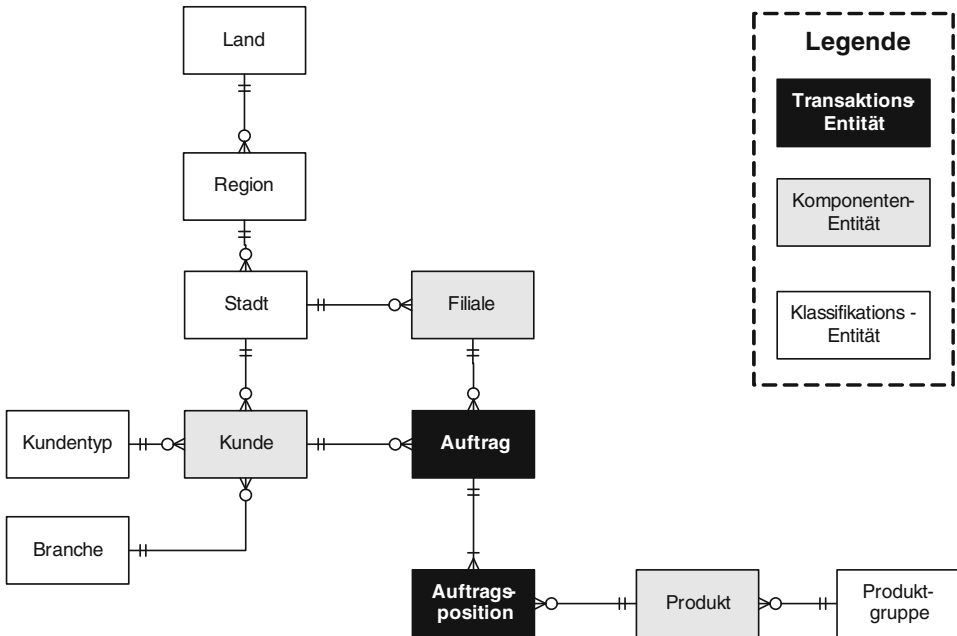


Abb. 2.47 Beispiel: Klassifikation der Entitäten

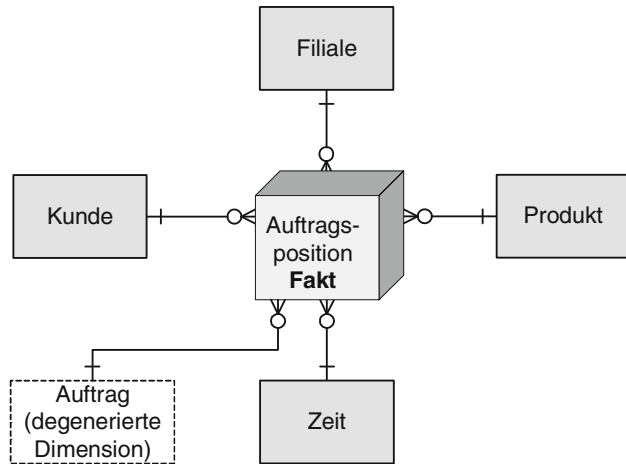
Abbildung 2.47 zeigt ein Beispiel für die Klassifikation der Entitäten in einem operativen System.

2. Konzeptioneller Star-Schema-Entwurf

In diesem Schritt werden ein oder mehrere grobe Star-Schemata entworfen.

- **Auswahl der Transaktions-Entitäten.** Jede Transaktions-Entität kann die Faktentabelle eines potenziellen Star-Schemas bilden. Jedoch werden nur Transaktions-Entitäten, die entscheidungsrelevant sind, für ein Star-Schema herangezogen. Außerdem können in einem Galaxie-Schema auch mehrere Star-Schemata verbunden werden (siehe Abschn. 2.6.4) [209].
- **Bestimmung des Aggregationsniveaus.** Anschließend wird das Aggregationsniveau (die Granularität) des Star-Schemas festgelegt. Je höher das Aggregationslevel ist, desto niedriger sind Speicheranforderungen und desto performanter sind Abfragen. Andererseits gehen durch die Aggregation Informationen verloren und dadurch werden bestimmte Abfragen unmöglich [209].
- **Identifizierung der Dimensionen.** Komponenten-Entitäten, die mit der Transaktions-Entität verbunden sind, kommen als Dimensionen für diesen Cube in Frage. Jedoch sind nicht alle Dimensionen für die gewählte Granularität bzw. vorgesehenen Analysen relevant. Dimensionen, die keine zusätzlichen beschreibenden Attribute oder Hierarchien haben, werden als *degenerierte Dimensionen* bezeichnet (wie z. B. Zeit) [209, 210].

Abb. 2.48 Beispiel: Konzeptioneller Star-Schema-Entwurf



Degenerierte Dimensionen benötigen nicht unbedingt eigenständige Dimensions-Tabellen. Man kann sie stattdessen als Attribut direkt in der Fakten-Tabelle aufnehmen (siehe Abb. 2.48).

Beispiel (Fortsetzung) Basierend auf der Klassifikation aus Abb. 2.47 wird eine Faktentabelle ausgewählt. Die beiden Transaktions-Entitäten *Auftrag* und *Auftragsposition* sind durch eine Master-Detail-Beziehung verbunden und können in eine Faktentabelle kombiniert werden (siehe Abb. 2.48). Dabei wählt man die detailliertere Transaktions-Entität (*Auftragsposition*) als Faktentabelle und kollabiert die Master-Transaktions-Entität (*Auftrag*) in die Faktentabelle. Die Dimensionen des Star-Schemas bestehen aus den Komponenten-Entitäten, die mit den beiden Transaktions-Entitäten verbunden sind. Die *Auftrags*-Entität beinhaltet ein Attribut *Datum*, das im Datenwürfel zur *Zeit*-Dimension wird. *Auftrag* wird nicht als extra Dimension modelliert, sondern ist eine degenerierte Dimension, die nur mit einem Attribut in der Faktentabelle vermerkt wird. Es wird die niedrigste Granularität gewählt, d. h. Transaktionen werden nicht aggregiert.

3. Detaillierter Entwurf der Faktentabellen

- **Bestimmung der Primärschlüssel.** Der Primärschlüssel setzt sich aus den Primärschlüsseln aller Dimensionstabellen und gegebenenfalls aus den degenerierten Dimensionen zusammen [209].
- **Bestimmung der Kennzahlen.** Die Fakten (Kennzahlen) sind numerische Attribute in der Faktentabelle [209]. Dabei ist auf die *Summierbarkeit* der Kennzahlen zu achten (siehe Abschn. 2.5.4) [169, 209]. In der Faktentabelle sollen nach Möglichkeit absolute Werte von Kennzahlen gespeichert werden, wie z. B. Verkaufsmenge und -preis, und daraus prozentuale Kennzahlen on-the-fly berechnet werden, beispielsweise prozentuale Abschläge auf den Verkaufspreis mithilfe der jeweiligen Rabattklasse.

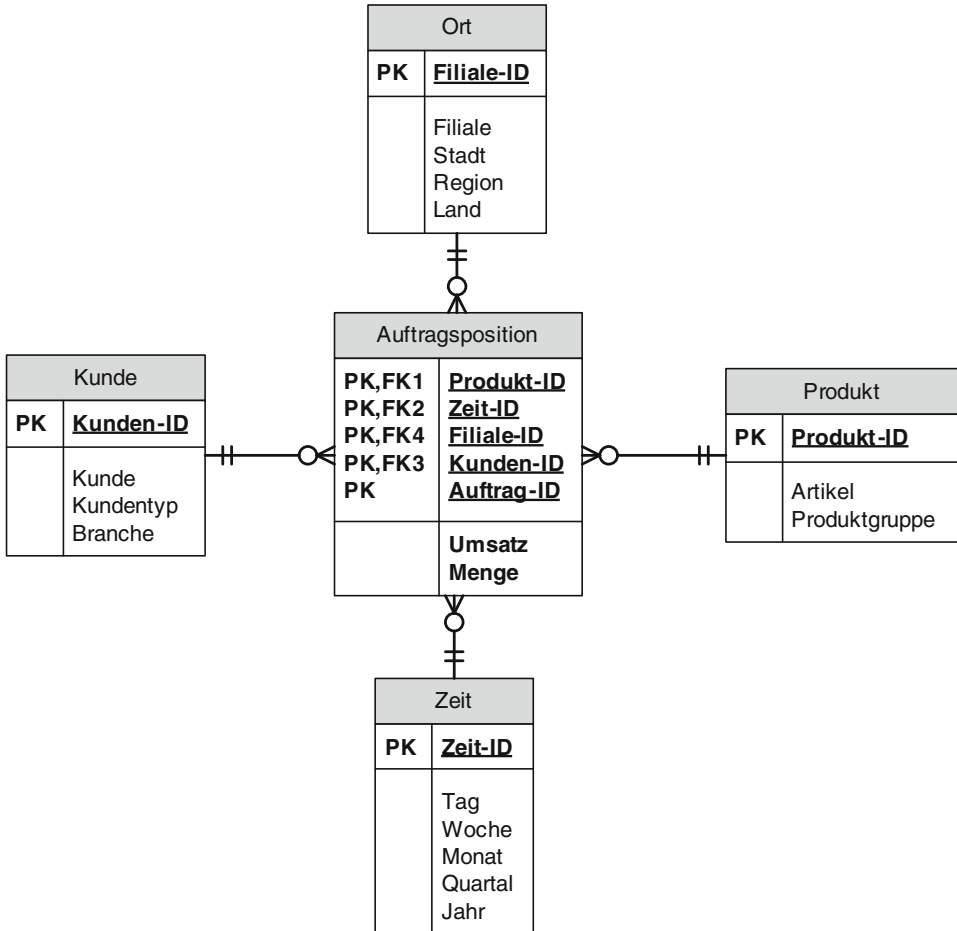


Abb. 2.49 Beispiel: Detaillierter Star-Schema-Entwurf

Beispiel (Fortsetzung) Der zusammengesetzte Primärschlüssel der Faktentabelle besteht aus den Primärschlüsseln aller Dimensionen, d. h. Filiale-ID, Produkt-ID, Kunde-ID, Zeit-ID und Auftrag-ID (siehe Abb. 2.49). Als Kennzahlen werden Umsatz und Menge gewählt.

4. Detaillierter Entwurf der Dimensionstabellen

- **Bestimmung der Primärschlüssel.** Eine Dimensionstabelle besitzt typischerweise einen nicht-zusammengesetzten Primärschlüssel. Bei einem Snowflake Schema verweisen die Fremdschlüssel in den untergeordneten Dimensionstabellen auf den Primärschlüssel der übergeordneten Dimensionstabelle.
- **Bestimmung der Historisierungsstrategie.** Siehe Abschn. 2.6.6.

- **Sprechende Bezeichner.** In den operativen Systemen können Abkürzungen oder Identifikationsnummern in den Komponenten- bzw. Klassifikations-Entitäten vorhanden sein. In den Dimensions-Tabellen sollten statt dessen sprechende Bezeichner benutzt werden, deren Beschreibung aus dem Repository abrufbar sind.

Beispiel (Fortsetzung) Es wird ein Star-Schema gewählt (siehe Abb. 2.49). Die Primärschlüssel der Dimensionstabellen sind nicht zusammengesetzt (Filiale-ID, Produkt-ID, Kunde-ID und Zeit-ID).

2.6.7.2 Informationsbedarfsorientierter Ansatz

Im nachfrageorientierten Ansatz startet man von den Informationsbedürfnissen der Nutzer [282]. Anschließend analysiert man, welche Daten innerhalb und außerhalb des Unternehmens notwendig sind, um diese Informationsbedürfnisse zu erfüllen. Die **Informationsbedarfsanalyse** ist somit eine Untersuchung der Anforderungen (engl. Requirements) an die bereitzustellenden Unternehmensinformationen aus Nutzersicht.

Stroh, Winter und Wortmann [282] schlagen als Prozess für die Informationsbedarfsanalyse fünf Schritte vor:

1. *Gewinnung:* Hierbei wird der Informationsbedarf durch verschiedene Methoden wie Interviews oder Dokumentanalyse erhoben. Man kann zwischen induktiven und deduktiven Methoden zur Gewinnung des Informationsbedarfs unterscheiden [129, S. 367]. Induktive Verfahren ermitteln subjektive, personenbezogene Informationsbedürfnisse. Deduktive Verfahren leiten den Informationsbedarf logisch aus der Analyse von Aufgaben bzw. Zielen ab. Induktive und deduktive Methoden können auch kombiniert angewendet werden.

Als Methoden der induktiven Informationsbedarfsgewinnung werden in der Literatur [282, 281] [105, S. 265] [159, S. 61] erwähnt:

- *offene Interviews:* Der Befragte benennt seinen Informationsbedarf. Dies wird begleitet durch unterstützende und klärende offene Fragen des Interviewers und veranschaulicht durch Situationen aus seinem Arbeitsalltag [159, S. 61].
- *Survey:* Anhand einer Liste von Informationsprodukten wählt und priorisiert der Mitarbeiter seinen Informationsbedarf.
- *Beobachtung am Arbeitsplatz:* Der Analyst begleitet einen Mitarbeiter und beobachtet den Arbeitsalltag und die darin vorkommenden Aufgaben und deren Informationsbedarf [105, S. 267].
- *Dokumentenanalyse:* Bestehende elektronische und papiergebundene Berichte und Dokumente werden nach den enthaltenen und für die Erstellung notwendigen Informationen analysiert [105, S. 265].

Methoden der deduktiven Informationsbedarfsgewinnung sind [159, S. 62ff]:

- *Strategieanalyse:* Anhand der Unternehmensstrategie werden die für die Umsetzung und das Monitoring notwendigen Informationen abgeleitet.

- *Entscheidungs- und Aufgabenanalyse*: Analytische Modelle zur Entscheidungsunterstützung werden nach den erforderlichen Input-Daten untersucht.
 - *Prozessanalyse*: Die in einem Geschäftsprozess notwendigen Informationen werden analysiert.
2. *Dokumentation*: Die erfassten Informationsbedürfnisse müssen verständlich dokumentiert und zugänglich gemacht werden.
 3. *Übereinstimmung*: Die von verschiedenen Nutzergruppen mit unterschiedlichen Methoden gewonnenen Informationsbedürfnisse werden auf Widersprüche untersucht, die Terminologie wird vereinheitlicht und der Informationsbedarf wird nach dessen Dringlichkeit gereiht.
 4. *Validierung*: Der abgeleitete und vereinheitlichte Bedarf wird mithilfe von Befragungen und Prototypen validiert.
 5. *Management*: Die Informationsbedarfsanalyse ist kein einmaliges Ereignis, sondern ein kontinuierlicher Prozess, der auf geänderte interne und externe Gegebenheiten reagieren sollte. Der Informationsbedarf wird festgehalten, seine Veränderungen erfasst und versioniert.

Weiterführende Literatur Einen guten Einstieg in OLAP bietet wiederum das Buch von Bauer und Günzel (2009) [21]. Es ist hier das Kapitel 5 „Das multidimensionale Datenmodell“ empfehlenswert. Eine weitere, fast schon klassische Referenz ist Kimball [146] „The Data Warehouse Toolkit“. Das Buch enthält viele brauchbare Hinweise und Praktikertipps getreu dem Motto „So wird es gemacht!“ mit dem Verweis auf *Good Practice*.

Business Intelligence

Müller, R.M.; Lenz, H.-J.

2013, XXII, 306 S. 198 Abb., Softcover

ISBN: 978-3-642-35559-2