# 10. Data Warehouse

## Übersicht

- Klärung wichtiger Begriffe
- Architektur eines Data Warehouse
- Back-End Komponenten
- Front-End Komponenten
- Implementierung eines Data Warehouse durch relationale Datenbanksysteme

## Literatur


- W. H. Immon: Building the Datawarehouse, 2nd Edition, J. Wiley, 1996.
- L. Volk: Das Data-Warehouse-Konzept und seine Umsetzung im Projekt “Business Information Warehouse” bei der SAP AG, Diplomarbeit im Studiengang Informatik der Universität Marburg
- Unterlagen des Seminars “Data-Warehousing”, abgehalten im Wintersemester 96/97 am Studiengang Informatik der Universität Marburg.






# Online-Transaction Processing (OLTP)

- bestimmte während der letzten 30 Jahren im wesentlichen die Entwicklung bei der Datenbanktechnologie.
- Charakterisierung von OLTP-Anwendungen
  - kurze Transaktionen
  - wenig Datenzugriffe pro Transaktion
  - viele Änderungen
  - keine historische Daten, d. h. beim Ändern werden Daten überschrieben
- Beispielsanwendungen
  - Buchungssysteme: Buchung eines Flugs
  - Bestellung von Waren
  - …
- Optimierung dieser Datenbanken führte zu der Theorie der Normalformen.
  - Normalisierte Relationen sind gut für Datenbanken mit häufigen Änderungsoperationen.





# Online-Analytical Processing (OLAP)

- seit kurzem stehen Anwendungen im Vordergrund, die sich mit der **Analyse** der Daten aus verschiedenen **heterogenen** Datenbanken beschäftigen.
- Charakterisierung von OLAP-Anwendungen
  - lange Transaktionen
  - hohe Anzahl an Datenzugriffen pro Transaktion
  - lesender Zugriff auf Daten, “keine” Änderungen
  - Zugriff auf historische Daten
- Beispiele
  - Auslastung der Atlantikflüge über den Zeitraum der letzten 2 Jahren
  - Verkaufszahlen in einem Gebiet, in einer Produktgruppe in einem Zeitintervall
- Optimierung solcher Datenbanken führt zu der Einführung von Redundanz, so daß (lesende) Anfragen effizient unterstütz werden.
- Redundante Daten werden in einer separaten Datenbank, dem **Data Warehouse**, gehalten.





# 10.1 Data Warehouse - eine Definition

- ist eine themenorientierte, historische und autonome Datenbank eines Unternehmens, in der Daten aus verschiedenen unabhängigen heterogenen Quellsystemen integriert und verwaltet werden. Ziel ist einem Unternehmen durch zeitbezogene Abfragen und Analysen entscheidungsunterstützende Ergebnisse zu liefern.

## Themenorientierung

- entscheidungsrelevante Sachthemen eines Unternehmens stehen im Vordergrund
  - umsatzstärkste Kunden
  - kostenintensivste Produkte

  Man spricht dann auch von einer multidimensionalen Datenbank

## Integration

- Verknüpfung der Daten aus verschiedenen heterogenen Datenbanken
- passiver Ansatz (lazy, on demand)
  - Extraktion der Daten aus den Quellsystemen bei der Anfrageausführung
- aktiver Ansatz (eager, in advance)
  - kein Zugriff auf Quellsysteme bei der Anfrageausführung
  - Extraktion im voraus

## Gründe für den aktiven Ansatz im DW

- Vermeiden der langsamen Zugriffe auf die Quellsysteme
- Verfügbarkeit von Quellsystemen nicht erforderlich
- unwesentliche Beeinträchtigung der Performance der Quellsysteme
- Analysen und Anfragen erfordern i.a. nicht die neuesten Daten
- Anreicherung der Daten bei der Extraktion
- historische Daten werden im DW aufbewahrt (kein Löschen der Daten)





## Zeitbezug

- Daten liegen im DW in mehreren Versionen (bzgl. Zeit) vor.
- Anfragen können sich nun auf einen Zeitpunkt bzw. Zeitraum beziehen

## Autonomie

- DW ist eine eigenständige und voll funktionsfähige Datenbank, die bezgl. der spezifischen Anforderungen optimiert ist.





# 10.2 Das mehrdimensionale Datenmodell

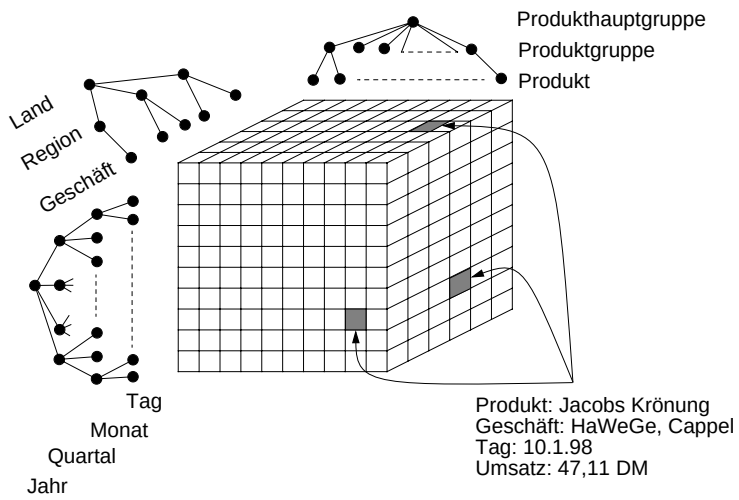
- Entscheidungsunterstützung basiert auf einem mehrdimensionalen Datenmodell (konzeptionelle Ebene)

## Bestandteile des Datenmodells

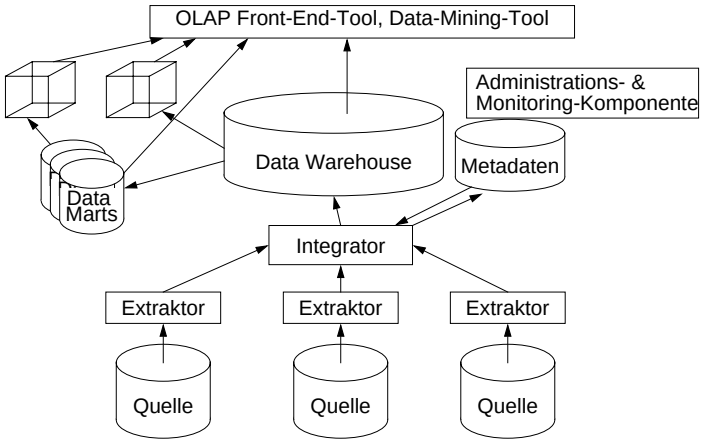
- **Meßgrößen** (**Fakten**)
  - numerische Wertebereiche
  - Beispiele: Umsatz, verkaufte Einheiten, Kosten, Deckunsbeitrag
- **Dimensionen**
  - Meßgrößen liegen in einem mehrdimensionalen Kontext
  - Beispiel: Umsatz hängt z. B. ab von den Dimensionen Produkt, Ort und Zeit (“natürliche Parameter von Umsatz”)
  - Dimension setzt sich zusammen aus mehreren Attributen z. B.: Ort besitzt als Attribute Region, Land und Geschäft
  - ein Attribut einer Dimension ist eindeutig (innerhalb der Dimension)
  - Attribute einer Dimension sind oft hierarchisch strukturiert Tag, Monat, Jahr (d h. es bestehen FDs zwischen den Attributen)







# 10.3 Architektur eines DW-Systems








# 10.3.1 Back-End Komponenten

## Voraussetzung für eine effektive Extraktion und Integration

- Zustand der Quellsysteme
- Konsistenz zwischen den Quellsystemen (wenn möglich)

## Integration

1. Phase: Schrubben der Daten (data scrubbing)
   - Eliminierung von Duplikaten
   - Einfügen von Default-Werten
   - Korrektur von Fehlereingaben
   - Aufspüren von Integritätsverletzungen und “ungewöhnlichen” Daten
2. Phase: Transformation
   - Vereinheitlichung von Maßeinheiten, Attributsnamen, Attributswerten
   - Ableitung von betriebswirtschaftlichen Kennzahlen





## Besonderheiten bei der Integration

- Datenanreicherung bei der Integration
  - Zusammenführung von Daten aus verschiedenen Quellen (z. B. Kunde mit Bonitätsprofil)
- Berücksichtigung der Detailstufe (bei der Zeit)
- Metadaten unterstützen den Abbildungsprozeß zwischen den Modellen der Quellsysteme und dem Modell des Data Warehouse





# Laden von Daten ins DW

- Datenimport kann manuell, periodisch oder ereignisgesteuert ausgelöst werden.
- komplette Neuerzeugung des DW ist zu aufwendig
- Stattdessen findet ein inkrementelles Laden der Daten ins Data Warehouse statt.

## Problem:

- Wie können Änderungen im Quellsystem erkannt werden?
- Data Shipping
  Anlegen einer speziellen Log-Datei, in die Änderungen (mittels Trigger) geschrieben werden
- Transaction Shipping
  Log-Datei des TA-Managers wird durchsucht.

## Aufbau spezieller Datenstrukturen im DW

- Anpassen von “materialisierten” Sichten
- Einfügen in Indizes

# Verwaltung der Metadaten

## Administrative Metadaten

- konzeptionelle (mehrdimensionale) Schema
- logisches Implementierungsschema (z. B. im relationalen Modell)
- vordefinierte Berichte
- Beschreibung der Quellsysteme
- Regeln zur Transformation und Archivierung

## Applikationsspezifische Metadaten

- Verzeichnis über die Anwendungsterminologie

## Operationale Metadaten

- Statistiken des laufenden Betriebs
  - Antwort- und Ladezeiten
  - Nutzung von Indizes und materialisierten Sichten





# 10.3.2 Front-End Komponenten

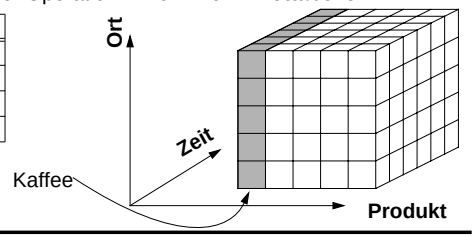
## Anforderung

- Anwender (z. B. Manager eines Unternehmens) soll selbständig und interaktiv das Data Warehouse durchsuchen können.
  - Kennzahlenvergleich
  - Ursachenanalyse

- Um diesen Anforderungen gerecht zu werden, stellt die OLAP-Engine einige Basisoperationen auf dem Hyperwürfel zur Verfügung
- Präsentation der Ergebnisse einer Operation in Form von **Pivottabellen.**

| Produkt | Ort | Zeit | Umsatz |
|---|---|---|---|
| Kaffee | Cappel | Januar | 47,11 |
| Kaffee | Wehrda | Februar | 333,33 |
| Kaffee | Cappel | Januar | 99,01 |
| Obst | … | … | … |

flache Tabelle








## Beispiel einer Pivottabelle:

| | Januar | Februar | März |
|---|---|---|---|
| Cappel | 574,11 | 550,20 | 1200,30 |
| Wehrda | 808,80 | 770,77 | 500,34 |
| Kernstadt | 478,30 | 450,28 | 289,89 |

- Die Dimension Produkt wurde auf die Produktgruppe Kaffee “eingefroren”. Man spricht dann auch vom Ausblenden einer Dimension
- Kaffeeumsatz wird jetzt in Abhängigkeit der **angezeigten** Dimension Ort und Zeit veranschaulicht.

## Mehrdimensionale Pivottabellen

- Durch Verschachtelung zweier Dimensionen können mehr als zwei Dimensionen in einer Pivottabelle veranschaulicht werden.

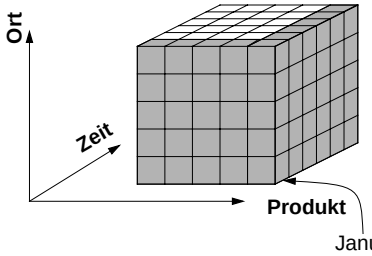
| | | Januar | Februar | März |
|---|---|---|---|---|
| Obst | Cappel | 574,11 | 550,20 | 1200,30 |
| | Wehrda | 808,80 | 770,77 | 500,34 |
| Spielzeug | Cappel | 89,89 | 91,91 | 1000,01 |
| | Wehrda | 890,89 | 78,34 | 888,90 |



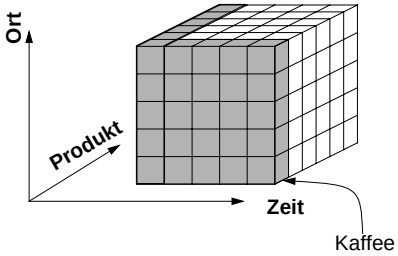


# Rotation

- Vertauschen von angezeigten und ausgeblendeten Dimensionen.
- Im Fall eines dreidimensionalen Würfels entspricht dies einer Rotation um 90 Grad




angezeigte Dimensionen: Produkt, Ort
ausgeblendete Dimension: Zeit
eingferorener Zeitpunkt: Januar




angezeigte Dimensionen: Zeit, Ort
ausgeblendete Dimension: Produkt
eingefrorenes Produkt: Kaffee

# Roll-Up/Drill-Down

- Roll-Up
  - Erhöhung der Aggregationsstufe
  - Verdichtung der Meßgrößen im Würfel
- Drill-Down
  - inverse Operation zu Roll-Up
  - Reduzieren der Aggregationsstufe
  - Erhöhung des Detailgrads der Daten
- Bei diesen Operationen werden sehr oft bereits die Attributshierarchien genutzt

## Weitere Operationen

- Ranging
- Sortieren
- Gruppieren / Aggregate
- Ausnahmebehandlung





# Data Mining

## Ziel

- Unterstützung der Ursachenanalyse
- automatische bzw. halbautomatische Mustersuche im Data Warehouse
  - von besonderem Interesse: Datenausreiser

## Techniken

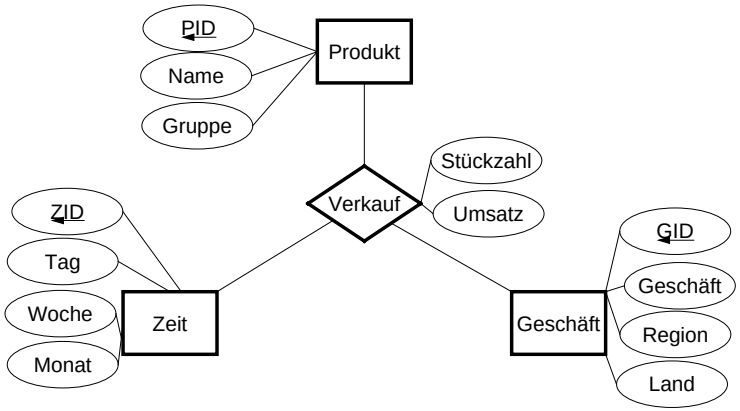
- Erstellung einer Klassifikation der Daten (z. B. Autoversicherungsklassen)
- Auffinden von Assoziationsregeln der Form $X \Rightarrow Y$
  - X und Y sind typischerweise Produkte
  - Konfidenz:
    rel. Häufigkeit, bei wieviel Zugriffen auf X auch auf Y zugegriffen wird
  - Support:
    rel. Häufigkeit der Transaktionen, bei denen sowohl auf X als auch auf Y zugegriffen wird.
- Visual Data Mining





# 10.4 Datenmodellierung

- Transformation des mehrdimensionalen Datenmodells in ein ER-Diagramm








# Star-Schema

## Definition

Ein Star-Schema besteht aus einer Menge von Relationen $D_1,\ldots,D_n$, F mit folgenden Eigenschaften:

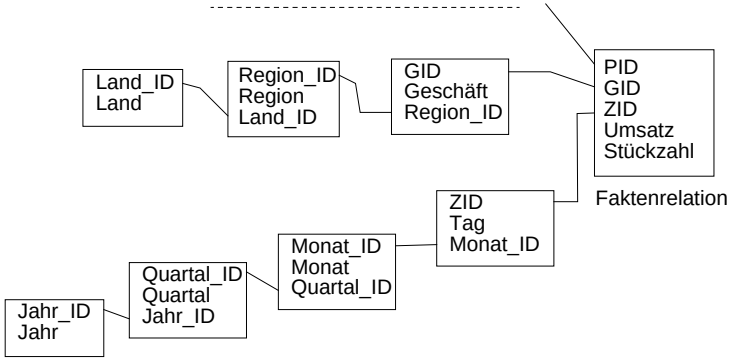
1. Jede Relation $D_i$ modelliert eine Dimension des Datenwürfels. Zusätzlich zu den Attributen der Dimension bekommt $D_i$ einen künstlichen Primärschlüssel $d_i$ zugeordnet. $D_i$ wird dann auch als **Dimensionsrelation** bezeichnet.
2. Die Relation F verbindet die Dimensionen miteinander, indem die Fremdschlüssel der Dimensionsrelation (d. h. $d_1,\ldots,d_n$) und die entsprechenden Meßgrößen als Attribute umgesetzt werden. Diese Relation wird dann auch als **Faktenrelation** bezeichnet.

## Bemerkungen:

- Das Star-Schema (genauer die Dimensionsrelationen) liegen nicht in dritter Normalform vor.

# Snowflake-Schema

- Werden die Dimensionsrelationen des Star-Schema in Normallform gebracht, so entsteht das Snowflake-Schema








## Beobachtung

- Die Faktenrelation ist im Vergleich zu den Dimensionsrelationen sehr groß.

## Leistungsbeurteilung

- keine Redundanz beim Snowflake-Schema
  - Wird der Name einer Produktgruppe geändert, so ist im Snowflake-Schema nur ein Datensatz, im Star-Schema i. a. mehrere Datensätze, von der Änderung betroffen.
  - Das Snowflake-Schema benötigt i. a. weniger Speicherplatz (zusätzlich müssen aber die künstlichen Schlüssel gespeichert werden).
  - Die Redundanz besitzt aber auf Grund der obigen Beobachtung kaum eine praktische Relevanz.
- Durch die Denormalisierung werden bei Anfragen im Star-Schema weniger Joins berechnet. Zudem kann der Join zwischen der Faktenrelation und seinen Dimensionsrelationen sehr effizient berechnet werden.





# Anfragen auf dem Star-Schema

## Anfoderungen

- Ausführung von Aggregationen auf Teilmengen im Würfel
- Detailierungsgrad des Würfels soll beliebig einstellbar sein

## Allgemeine Form einer Anfrage

| | | |
|---|---|---|
| **select** | $g_1,\ldots,g_k$, $agg(f_1), \ldots, agg(f_m)$ | ← aggregierte Meßgrößen |
| **from** | $D_1,\ldots,D_n$, F | ← Relationen des Star-Schemas |
| **where** | <Selektionsbedingung auf $D_1$> | **and** |
| | … | **and** |
| | <Selektionsbedingung auf $D_n$> | **and** |
| | $D.d_1 = F.d_1$ | **and** |
| | … | ← Bedingung des Star-Joins |
| | $D.d_n = F.d_n$ | |
| **group by** | $g_1, \ldots, g_k$; | |





## Beispiel:

Finde die Quartalsumsätze aller Geschäfte differenziert nach Ländern im Jahr 1996 für die Produktgruppe Kaffee.

| | | |
|---|---|---|
| **select** | PName, Land, Quartal, **sum**(Umsatz) | |
| **from** | Produkt P, Geschäft G, Zeit Z, FaktenRel F | |
| **where** | Jahr = 1996 | **and** |
| | Produktgruppe = “Kaffee” | **and** |
| | Z.ZID = F.ZID | **and** |
| | P.PID = F.PID | **and** |
| | G.GID = F:GID | |
| **group by** | PName, Land, Quartal; | |

## Bemerkungen

- Die spezielle Form des Joins (Anfrage) wird auch als **Star-Join** (Star-Anfrage) bezeichnet.
- Voraussetzung dabei ist, daß die Meßgrößen uneingeschränkt additiv sind, da die Meßgrößen der Faktentabelle bereits aggregiert sind (z. B. auf einen Tag). Es ist z. B. nicht möglich, die Anzahl der Kunden am Tag x zu bestimmen, die Kaffee kauften (da ein Kunde mehrere Kaffeeprodukte gleichzeitig kaufen kann).

# 10.5 SQL-Erweiterungen

- Defizite bei SQL im Hinblick auf Anfragen im Data Warehouse
  - einfache Gruppierungsanfragen müssen in mehreren SQL Anweisungen aufgespalten werden.

## Beispiel

- Relation: Anrufe(VVWahl, VNummer, ZVWahl, ZNummer, Datum, Länge)

  Ein Tupel dieser Relation protokolliert ein Telefongespräch und registriert dabei die Telefonnummern des Anrufers (VVWahl, VNummer), die Nummer des Angerufenen (ZVWahl, ZNummer), das Datum und die Länge des Gesprächs.
- Anfrage: Finde für jeden Kunden, das von ihm geführte längste Telefongespräch und die entsprechende Vorwahl.
- Lösung in SQL in zwei Schritten

  **create view** Tmp **as**

  **select** VVWahl, VNummer, MaxL = **max**(Länge)
  **from** Anrufe
  **group by** VVWahl, VNummer;





**select** t.VVWahl, t.VNummer, ZVWahl, Länge
**from** Anrufe a, Tmp t
**where** Länge = MaxL **and**
a.VVWahl = t.VVWahl **and**
a.VNummer = t.VNummer;

Die Berechnung besteht aus einem Subselect (Erzeugung von Tmp) und einem Join zwischen Tmp und Anrufe.

## Erweiterungsvorschlag von SQL: **Gruppierungsvariablen**

- Gruppierungsvariablen werden in der group-by Klausel definiert.
- In einer zusätzlichen Klausel (**suchthat**-Klausel) kann der Bereich der Gruppierungsvariablen noch eingeschränkt werden.
- In der select-Klausel können nun die Aggregatsoperationen auf die Gruppierungsvariablen angewendet werden.
- Beispiel:

**select** VVWahl, VNummer, R.ZVWahl, R.Länge
**from** Anrufe
**group by** VVWahl, VNummer: R
**suchthat** R.Länge = **max**(Länge);





- In der Terminologie des mehrdimensionalen Datenmodells entspricht die suchthat-Klausel einem Herausschneiden einer Scheibe aus einem WÜrfel.

## Noch eine Anfrage:

- Gib für jeden Kunden, die durchschnittliche Länge der Telefongespräche mit der Vorwahl 069 und 089 (in einem Datensatz).
- In Standard SQL müssen zwei Sub-Anfragen (in Form von Views) und ein Join zwischen den Views berechnet werden (Übung)
- Durch die Verwendung von Gruppenvariablen vereinfacht sich die Formulierung der Anfrage erheblich:

**select** VVWahl, VNummer, **avg**(R.Länge), **avg**(S.Länge)
**from** Anrufe
**group by** VVWahl, VNummer: R, S
**suchthat** R.ZVWahl = “069” **and**
S:ZVWahl = “089”;





# 10.6 Materialisieren von Aggregaten

## Motivation

- Viele Anfragen: Gruppierung der Faktenrelation F

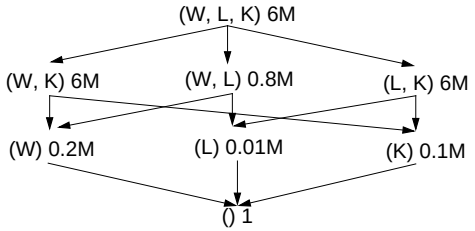
  **select** $g_1, \ldots, g_k$, agg(f)
  **from** F
  **group by** $g_1, \ldots, g_k$;

  Ist die Aggregationsoperation vorgegeben kann solch eine Anfrage allein über die Liste der Gruppierungsattribute beschrieben werden.

  Notation: $(g_1, \ldots, g_k)$
- Steigerung der Effizienz von Anfragen Q durch Materialisieren von Sichten V, die den oben definierten Anfragen entsprechen.
  - Für eine Anfrage Q und eine Sicht V definieren wir
    $Q << V \quad \Leftrightarrow$ Anfrage Q kann mit der Sicht V beantwortet werden.
  - Für zwei Sichten V und W definieren wir
    $V \leq W \quad \Leftrightarrow$ Attributmenge von W enthält die Attributmenge von V

## Beispiel:

- Betrachten wir die Faktenrelation des TPC-D Benchmarks, die aus 3 Dimensionsattributen **W**are, **L**ieferant, **K**unde besteht.
- Es ergeben sich nun $2^3$ Gruppierungsmöglichkeiten.
  - Diese Gruppierungsmöglichkeiten lassen sich als Knoten in einem Graphen veranschaulichen, wobei die Kanten durch die ≤-Relation gegeben sind.
  - Zusätzlich notieren wir die Anzahl der Tupel in den Sichten.








## Fragestellung:

- Nehmen wir vereinfachend an, daß genau n Sichten materialisiert werden können und daß die Anfragen identisch zu den Sichten sind. Welche Sichten der Faktenrelation sollen materialisiert werden?
- Die Kosten einer Anfrage werden dabei durch die Anzahl der Tupel von der Sicht v ausgedrückt (**Cost(v)**), die zur Anfragebearbeitung herangezogen wird.
- Diese Fragestellung kann dahingehend verallgemeinert werden, daß statt der Anzahl der Sichten der verfügbare Speicherplatz vorgegeben wird.
- Bereits diese stark vereinfachte Problemstellung ist NP-vollständig. Deshalb ist es gerechtfertigt, Heuristiken zur Lösung heranzuziehen.

## Folgerungen

- Die oberste Sicht muß stets materialisiert sein, da sonst die entsprechende Anfrage nicht beantwortet werden kann.





# Greedy-Algorithmus

- Im Folgenden betrachten wir einen iterativen Algorithmus zur näherungsweisen Lösung des obengenannten Problems.
  - Menge S enthält die bisher ausgewählten Sichten. S wird anfangs mit der obersten Sicht initialisiert.
  - Pro Schritt wird die Sicht mit dem höchsten Gewinn hinzugefügt.

## Algorithmus Gewinn(S, V)

Eingabe: Menge S der ausgewählten Sichten, V eine noch nicht ausgewählte Sicht
Ausgabe: Der Gewinn der Sicht V.

1. **FOREACH** Sicht W mit W ≤ V **DO**
   minCost := $\min_{X \in S} Cost(X)$;
   **IF** Cost(V) < minCost **THEN**
   $B_W$ := minCost - Cost(V);
   **ELSE**
   $B_W$ := 0;
2. **RETURN** $\sum_{W \leq V} B_W$;





# Zusammenfassung

- Datawarehouse ist derzeit das Thema im Bereich Datenbanken und zwar sowohl in der Forschung als auch in der Industrie.
- In diesem Kapitel konnte nur kurz die Problemstellung angerissen werden.
- Interessante Fragestellungen im Bereich des Data Warehouse sind unter anderem:
  - Wartung und Aufrechterhaltung materialisierter Sichten
  - Datenintegration
  - effiziente Verfahren zum inkrementellen Laden eines Warehouse
  - schnelle Berechnung von Star-Joins
  - Indexstrukturen für Data Warehouse (Zeit-Indexstrukturen)
  - Data Mining und Visualisierung
- Einige Implementierungsaspekte werden im nächsten Semester ausführlich behandelt!