Schlagwörter

- Data Warehouse
- On-Line Analytical Processing
- Multi-dimensional Analysis
- Data Cube
- Data Mining
- Decision Support System
- Management Information System

Motivation

Mache Daten zu Information um

- bessere Übersicht über Geschäftsprozesse
- Analyse Kundenverhalten
- Analyse Produktverkäufe
- ...

um schneller auf Veränderungen reagieren zu können.

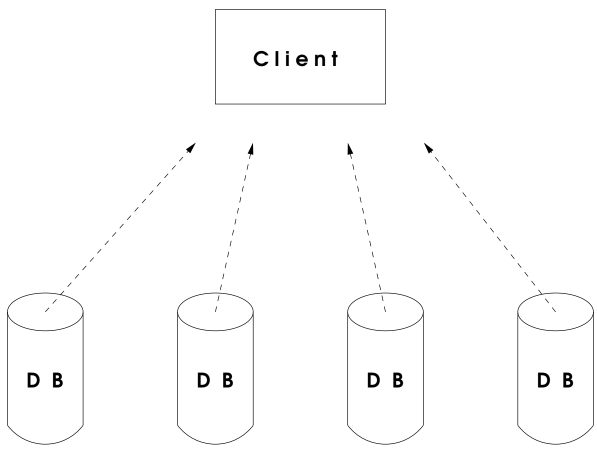
Entscheidungen fällen

Definition

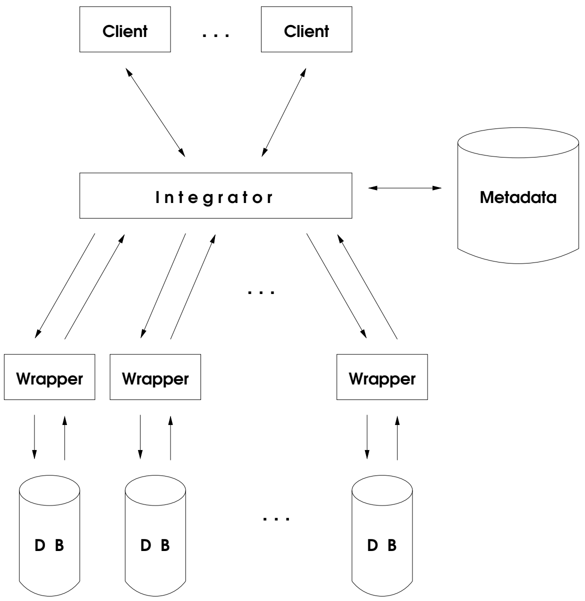
Data Warehouse is a repository of integrated information, available for querying and analysis.

Grundproblem

Integration verschiedener (heterogener) Datenquellen.



Multi-Database System Ansatz

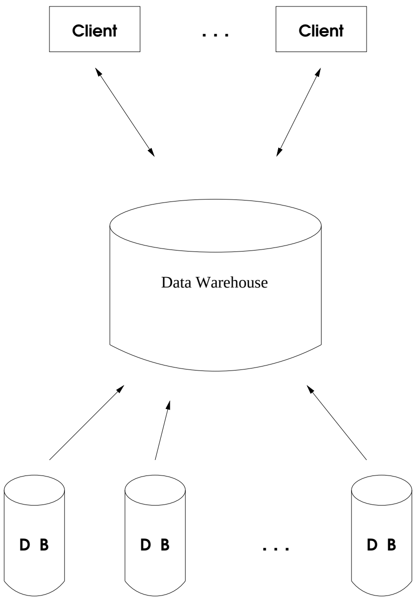


Nachteile des anfragegetriebenen Ansatzes

- Verzögerungen bei der Anfragebearbeitung
  - langsame oder nicht verfügbare Datenquellen
  - komplexe Filter- und Konvertierungsfunktionen
- ineffizient und potentiell teuer für häufig vorkommende Anfragen
- Wettbewerb mit lokaler Bearbeitung

$\Longrightarrow$ von der Industrie nicht angenommen

Data Warehouse Ansatz



Vorteile des Data Warehouse Ansatzes

- hohe Performanz
  aber nicht unbedingt aktuelle Information
- keine Interferenzen mit lokaler Bearbeitung
  - komplexe Anfragen im Data Warehouse
  - OLTP in den lokalen Datenquellen
- Daten im Data Warehouse kopiert
  - möglich: Änderungen, Anmerkungen, Zusammenfassungen, Restrukturierungen
  - Speicherung historischer Daten

$\Longrightarrow$ Industrie springt auf diesen Zug

Wo Licht ist, da ist auch Schatten

- Anfragegetriebener Ansatz besser für
  - Daten mit hoher Änderungsrate
  - variierende Datenquellen
  - wirklich große Datenmengen
  - Klienten mit unvorhersehbaren Anforderungen

Bsp.: Ad-hoc Anfragen an das Web

Zwei verschiedene Aspekte des DW:

1. Wie bekommt man die Daten ins Data Warehouse?
   Data Warehousing
2. Was macht man mit den Daten im Data Warehouse?
   Warehouse DBMS

Eigenschaften der Daten im Data Warehouse

- Unmengen von oft historischen Daten
- wenige Änderungen oder Änderungen off-line
- oft auch nur Zufügen und kein Ändern der Daten

Beispiele:

- alle Transaktionen von K-Mart
- vollständige Kundenhistorie bei einer Versicherung
- vollständige Liste aller Börsentransaktionen

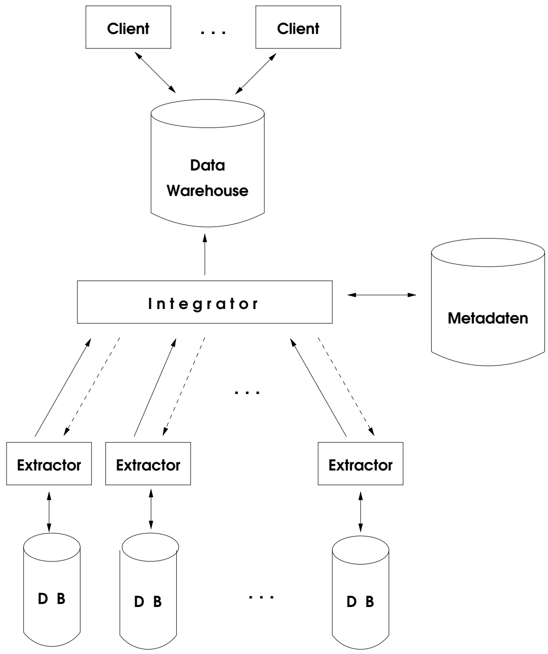
Warehouse DBMS Schlagwörter

- hauptsächlich benutzt für Decision Support (DSS)
  - Synonym: On-Line Analytical Processing (OLAP)
  - komplexe Anfragen, substantielle Aggregation
  - TPC-D Benchmark
- kann mehrdimensionale Datenbanken/Analyse unterstützen
  - Synomym: Data Cube
  - Sicht auf relationale Daten: alle möglichen Gruppierungen und Aggregierungen
  - Sichten materialisiert, virtuell, oder gemischter Ansatz
  - häufigste Prozesse:
    * roll-up: Aggregierung entlang einer Dimension
    * drill-down: Deaggregierung entlang einer Dimension

Fortsetzung

- auch für Data Mining benutzt
  - Entdeckung bestimmter Muster in den Daten
- Herausforderungen
  - Datenankentwurf
  - Anfragebearbeitung und -optimierung
  - Sichtenverwaltung

Grundarchitektur



Stand der Technik

- Warehouse ist Standard- oder spezialisiertes DBMS
- alles ist relational
- Extraktion und Integration im Batchbetrieb, meistens off-line, oft "von Hand"
- Integrator fordert niemals Daten von Quellen an

Data Cube (Bsp)

Szenario:

- Teile von Lieferanten werden an Kunden zu einem bestimmten Preis geliefert

drei Dimensionen:

- part
- supplier
- customer

Interessenfokus:

- total sales

berechnet aus den einzelnen Verkaufspreisen (*sp*).

Data Cube

Tripel:

- p, s, c

auch Zelle genannt.

Innerhalb einer Zelle werden die Daten des Interessenfokus aggregiert. Hier also die Verkaufspreise (sp).

In diesem Beispiel: Jede Dimension nur ein Attribut. Meistens aber jede Dimension mehrere Attribute.

Mögliche Aggregierungen

select	part, customer, sum(sp) AS TotalSales
from	BaseRelation
group by	part, customer

Mögliche Aggregierungen

1. $(-, -, -)$
2. $(ALL, -, -)$
3. $(-, ALL, -)$
4. $(-, -, ALL)$
5. $(ALL, ALL, -)$
6. $(ALL, -, ALL)$
7. $(-, ALL, ALL)$
8. (ALL, ALL, ALL)

$-$ entsprechen Attributen in der **group by**-Klausel

Hierarchisch Organisierte Dimensionen

Dimension Zeit:

- Jahr
- Monat
- Tag
- Woche
- Quartal

Jetzt leicht zu verstehen: drill-down und roll-up.

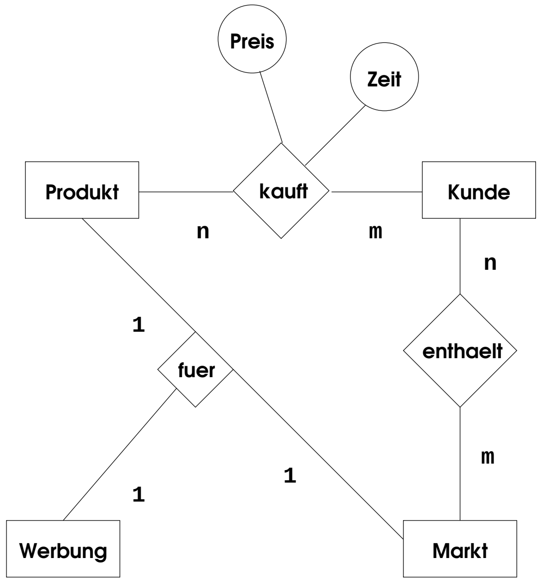
Datenbankentwurf für Data Warehouse

- Denormalisierung
- Star-Design
- Dimensional Model

Motivation:

1. Performanz
2. einfache Anfragen

Bsp ER-Diagramm



Dimensional Model

Beliebige Fakten

normalisiertes Design:

> Die SQL-Anfrage von 2 Seiten Umfang mit 14 Bedingungen kann auf einer 2 Millionen Dollar Maschine 60 mal pro Tag ausgeführt werden.

denormalisiertes Design:

> Der 5-Zeiler kann auf einer einfachen Workstation 1000 mal pro Tag ausgeführt werden

(Kimball, Strehlo: Why Decision Support Fails and how to fix it. DATAMATION June 1994, SIGMOD Record 24(3) Sept. 1995)

Unzulänglichkeiten von SQL

Beispiel für Anfrage, die schlecht auszudrücken ist:

> Wie hat sich der Umfang der Verkäufe in den letzten beiden Jahren entwickelt?

Ob absolut/relativ/Differenz spielt dabei keine Rolle.

"Schwierige Anfragen"

- ranking
- top 10, bottom 25%
- cumulative sum
- labeling (e.g. high, medium, low)
- break rows (falls der Wert, auf dem sortiert wurde sich ändert, wird eine Zusammenfassung eingeführt)

Prinzipieller:

1. vergleichende Antworten
2. sequentielle Verarbeitung

Vergleichen

- Produkt-Dimension
  - Verkaufsverhältnis von Antons zu Bertas Produkten
  - Verkaufsverhältnis von Antons Produkten zu allen Produkten
  - Verkaufsverhältnis von Produkten für die Werbung und für die keine Werbung gemacht wurde
- Kunden-Dimension
  - Verhältnis der Waren, die zu Ketten, im Vergleich zu Einzelhandelsgeschäften geliefert wurden.
  - Verkäufe von meinen Produkten an neue zu alten Kunden
- Zeit-Dimension
  - Weihnachtsverkäufe dieses/letztes Jahr

Vergleichen

durch: ALTERNATE a WHERE cond

Beispiel:

```
select  pname,
        sum(salesdollar)
        / sum(alternate salesdollars
              where time.year = 1993)
from    sales, product, market, time
where   ...
```

Sequentielle Verarbeitung

add functions:

- number()
- numberof()
- totalof(expr)
- rank(expr)
- tertile(expr)
- quartile(expr)
- movingavg(expr,num)
- movingsum(expr,num)
- cumulative(expr)

Break-Rows

spezielle **break by**-Klausel