

# Data Warehousing und Data Mining

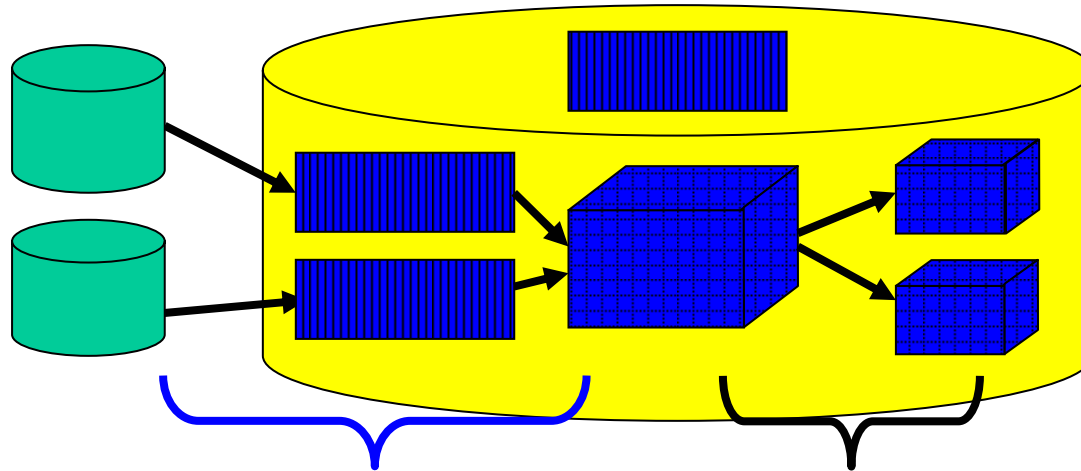
Einführung in Data Mining

Ulf Leser

Wissensmanagement in der  
Bioinformatik

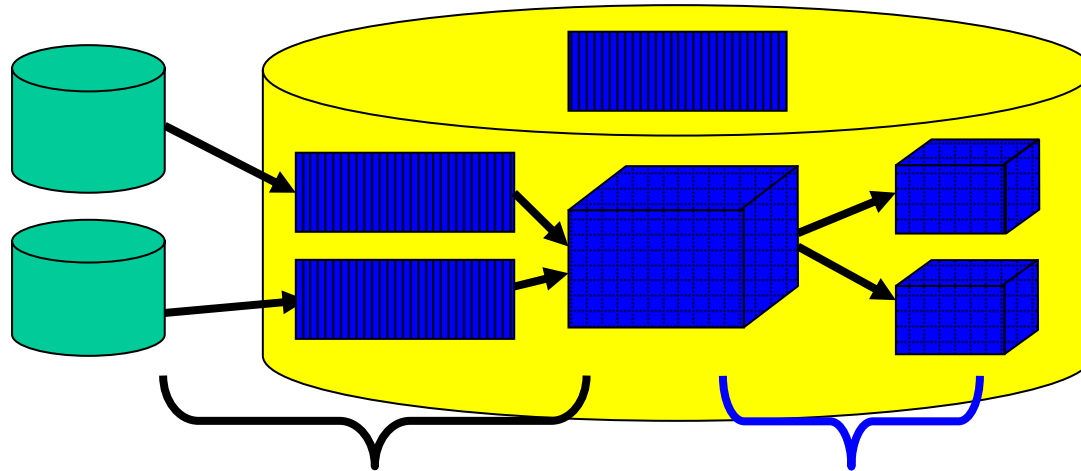


# Ermittlung von Änderungen



- Von Quellen zum Cube
  - Änderungen kommen aus Quellen
  - **Differential Snapshot Problem**
  - Staging Area speichert nur Änderungen
  - Zusätzliche Spalte „change“
    - +: Tupel muss eingefügt werden
    - -: Tupel muss gelöscht werden
    - Updates als delete + insert
  - [Man kann den Cube auch als MV über den Quellen betrachten ...]

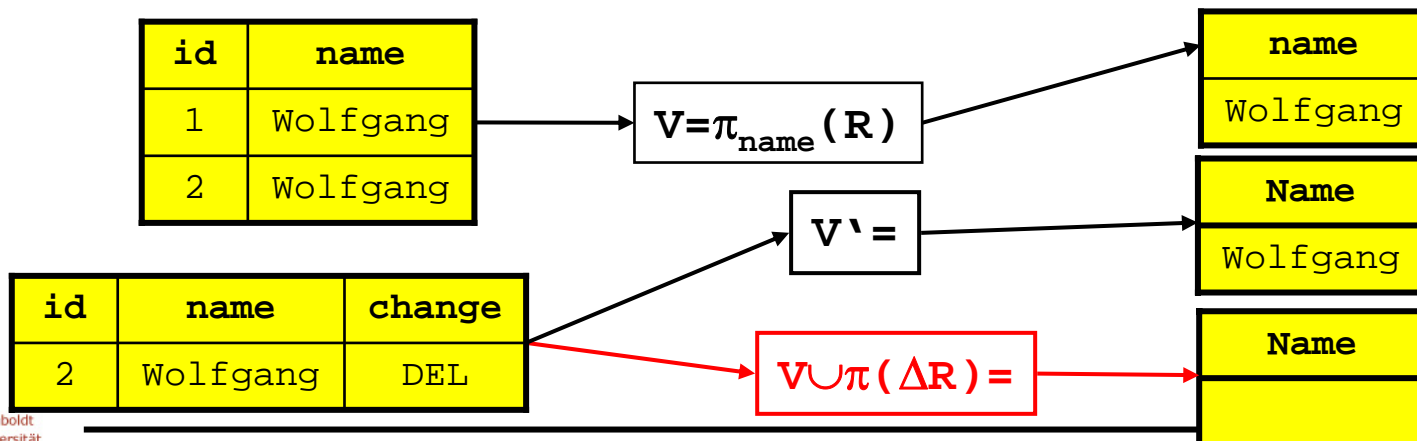
# Ermittlung von Änderungen



- Von Basisrelationen zu MVs
  - Änderungen werden durch SQL erzeugt
  - Kein Differential Snapshot Problem
    - Da man die **alten Versionen der Tabellen** nicht aufheben kann / will
  - Besser: **Logging aller Änderungen** über Trigger
  - Für jede Basistabelle anlegen einer „Schattentabelle“ mit zusätzlicher Spalte `type_of_change` (`DEL`, `INS`)
    - MV Logs

# MV: Selektion und Projektion

- MV ohne Join aber mit Selektion und/oder Projektion
- Selektion
  - Ausnutzung der Distributivität der Selektion
  - Sei also  $V = \sigma(R)$
  - $V' = \sigma(R') = ?$
  - $V' = \sigma(R') = \sigma(R \cup \Delta R) = \sigma(R) \cup \sigma(\Delta R) = V \cup \sigma(\Delta R)$
  - Sehr schön: Wir müssen **nur das Delta** betrachten
- Projektion
  - Gilt  $V' = \pi(R') = \pi(R \cup \Delta R) = \pi(R) \cup \pi(\Delta R) = V \cup \pi(\Delta R) ?$



# MV mit Joins

---

- Wir beginnen mit nur einem Join
- $$\begin{aligned}V' &= R' \bowtie S' = (R \cup \Delta R) \bowtie S' \\ &= (R \bowtie S') \cup (\Delta R \bowtie S') \\ &= (R \bowtie (S \cup \Delta S)) \cup (\Delta R \bowtie S') \\ &= (R \bowtie S) \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie S') \\ &= V \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie S') \\ &= V \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie (S \cup \Delta S)) \\ &= V \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie S) \cup (\Delta R \bowtie \Delta S)\end{aligned}$$
- Problem: **Stimmt nicht**

# MV mit Joins

---

- Also: Joins von Deltas vermeiden
- $V' = R' \bowtie S' = (R \cup \Delta R) \bowtie S'$   
...  
 $= V \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie S')$
- Unschön: Wir benötigen den **alten Zustand R** (bzw. S) zur Berechnung von  $\Delta V$ 
  - Das ist teuer – alle Relationen müssen doppelt gehalten werden
  - (Wunsch: R/S immer aktualisieren und nur die Logs zusätzlich halten)

# Vorhalten zweier Zustände?

---

- Möglichkeit 1
  - R wird bei Aktualisierung von V „zurückgerechnet“:  $R = R' - \Delta R$
- Alternative : Wir **verzichten auf Löschooperationen**
  - Das ist in DWH durchaus realistisch
  - Dann gilt wieder das Distributivgesetz
  - Damit
    - $V' = R' \bowtie S' = (R \cup \Delta R) \bowtie S'$   
 $= V \cup (R \bowtie \Delta S) \cup (\Delta R \bowtie S) \cup (\Delta R \bowtie \Delta S)$
  - Noch kein Gewinn: Wir haben den **neuen Zustand und die Deltas**
  - Was tun?
    - $V = R \bowtie S = (R' - \Delta R) \bowtie (S' - \Delta S)$   
 $= (R' \bowtie S') - (R' \bowtie \Delta S) - (\Delta R \bowtie S') \cup (\Delta R \bowtie \Delta S)$
    - $V' = R' \bowtie S'$
    - $\Delta V = V' - V =$   
 $= (R' \bowtie \Delta S) \cup (\Delta R \bowtie S') - (\Delta R \bowtie \Delta S)$

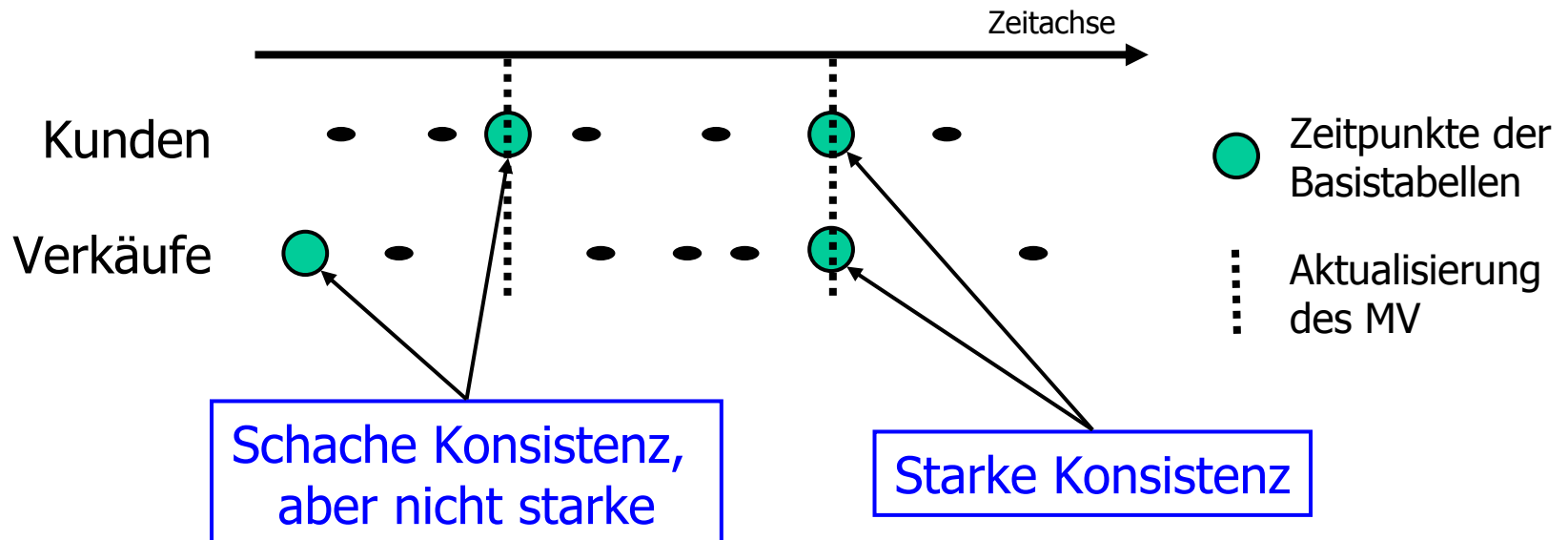
# Konsistenzgrade

- **Schwache Konsistenz**

- Alle Views sind bzgl. jeder Basisrelation konsistent. Damit ist nicht sichergestellt, dass ein global gültiger Zustand erreicht ist

- **Starke Konsistenz**

- Alle Views entsprechen einem global gültigen Zustand
- Erfordert synchronisierte Exports / Logfiles



# Kompression Log-Files

```
SELECT ..., sum(amount*price)
FROM sales S, ...
WHERE ...
GROUP BY T.day, P.name, R.regio
```

- Idee: Gruppierung schon im Log-File vornehmen
  - Übernahme der Gruppierungsattribute aus der Anfrage

day	name	region	customer	amount	change
12	100	3	1	100	INS
12	100	3	3	20	DEL
12	100	3	1	25	INS
11	100	2	1	5	INS
11	100	2	3	20	INS
...	...	...	...	...	

- Unterscheidung zwischen INS/DEL in Gruppierung codieren
- Erster Schritt
  - `SELECT day, name, regio, decode('INS',+1,-1), sum(amount)`  
`FROM log`  
`GROUP BY day, name, regio, change`

## 2. Schritt

```
SELECT ..., sum(amount*price)
FROM sales S, ...
WHERE ...
GROUP BY T.day, P.name, R.regio
```

- Ergibt

day	product	region	SUM	change
12	100	3	125	+1
12	100	3	20	-1
11	100	2	25	+1

- 2. Schritt: Auflösung von INS / DEL

- `SELECT day, name, regio, sum(amount*change)`  
`FROM log`  
`GROUP BY day, name, regio`

day	product	region	SUM
12	100	3	105
11	100	2	25

- Enthält alle relevanten Änderungen in **komprimierter Form**

# Definition von MVs

---

```
CREATE MATERIALIZED VIEW product_sales_mv
BUILD IMMEDIATE
REFRESH FAST
ENABLE QUERY REWRITE
AS
SELECT P.prod_name, SUM(S.amount), COUNT(*)
FROM   sales S, products P
WHERE  S.product_id = P.id
GROUP BY P.prod_name;
```

- Materialisierte Sicht **wird als Tabelle** angelegt
  - Indexierung, Partitionierung, STORAGE Klauseln, etc.
- „**Query Rewrite**“ muss explizit erlaubt werden

# Aktualisierungsstrategien

---

- Wann wird aktualisiert

- ON DEMAND: Nur bei Aufruf von speziellen Funktionen im Package DBMS\_MVIEW
  - `dbms_mview.refresh (mv), refresh_all,`  
`refresh_dependent (tab)`
- ON COMMIT: Beim Abschluss jeder Transaktion, die mindestens eine Basistabelle verändern
  - Bestmögliche Aktualität
  - Schwieriger Konsistenzbegriff

- Wie wird aktualisiert

- COMPLETE: Neuberechnung des Views bei Änderungen an mindestens einer Basistabelle
- FAST: **Inkrementelles Nachführen** von Änderungen
- FORCE: Ausführung von FAST, wenn möglich; sonst COMPLETE

# Inkrementelle Aktualisierung (fast)

---

- Erfordert das **explizite Anlegen eines MV Logs**

```
CREATE MATERIALIZED VIEW LOG ON sales
WITH SEQUENCE, ROWID
(prod_id, cust_id, time_id, channel_id,
 promo_id, quantity_sold, amount_sold)
INCLUDING NEW VALUES;
```

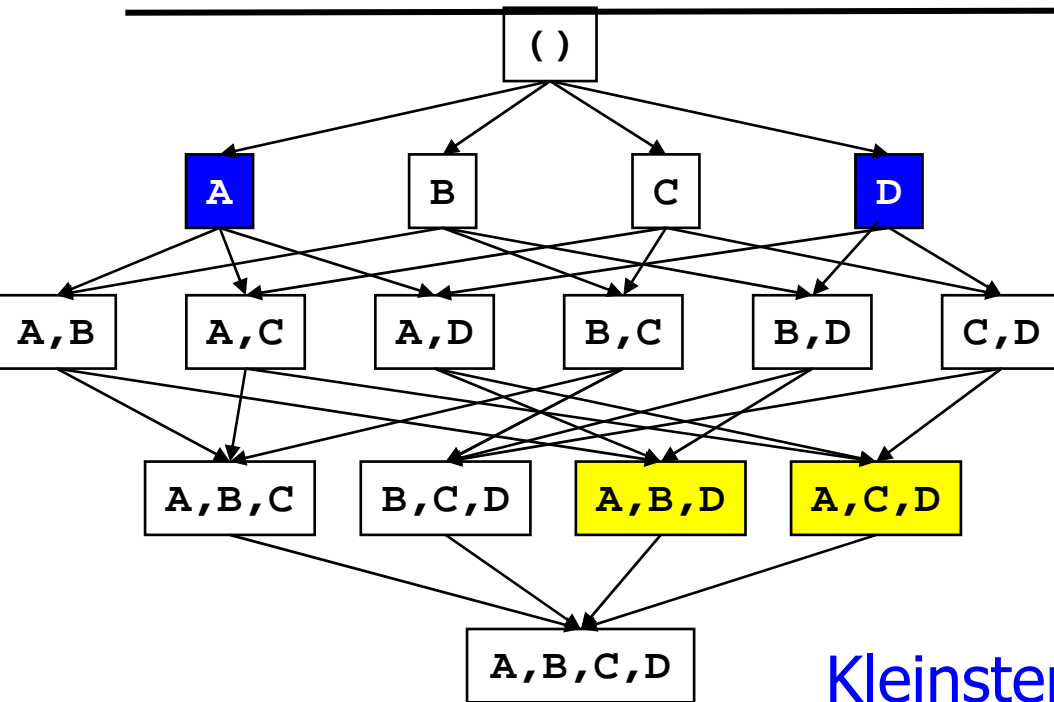
- Legt **Trigger** auf Mastertabellen an
  - MV-Log speichert **Deltas** von Änderungen
  - FAST Refresh spielt diese Deltas in MV ein
- Diverse **Einschränkungen**
  - Benötigt immer ein count(\*) bei Gruppierung – **Zählvariable**
  - Rowids müssen immer dabei sein

# Auswahl materialisierter Sichten

---

- Wir betrachten nur MV **mit Aggregation**
  - Berechnung von Aggregaten sind tendenziell teuer
  - Wiederholtes Scannen sehr großer Tabellen
  - Vorberechnung ist daher attraktiv
- Problem: Der vollständige CUBE benötigt viel zu viel Platz
  - Beispiel: 5 Dimensionen a 50 Knoten
    - Gleichverteilung der Werte auf alle Knoten
  - Die höchste Granularität hat  $\sim 50^5 = 3 \cdot 10^8$  Tupel
  - Dazu:  $\sim 5 \cdot 50^4 + 20 \cdot 50^3 + 20 \cdot 50^2 + 5 \cdot 50 = 3.3 \cdot 10^7$  Tupel mit weniger Gruppierungsattributen
  - Tatsächliche Zahlen abhängig vom **Füllgrad des Cube** ab
    - Wie schätzen?

# Kleinsten Vorfahre



## Kleinsten gemeinsamer Vorfahr

- Mehrere zur Auswahl
- Verlangt MV mit hoher Auflösung – braucht Platz
- Zur Ableitung sind zwei Aggregationen pro Query notwendig

# Problem

---

- Definition

*Gegeben eine Workload  $Q$ ,  $|Q|=n$ , mit Anfragefrequenzen  $h$ , ein Aggregationsgitter  $A$ , eine Kostenmatrix  $c$  und eine Konstante  $m$ . Das **MV-Selektion Problem** sucht die Menge  $M \subseteq A$ , die den folgenden Ausdruck minimiert*

$$c(Q, M) = \sum_{i=1 \dots n} h_i * \min_{a_j \in M} (c(q_i, a_j))$$

*unter Beachtung der **Nebenbedingung***

$$\sum_{a_i \in M} |a_i| \leq m$$

# Greedy-Heuristik [HRU98]

---

- Grundidee: **Iteratives Greedy-Verfahren**
  - MV werden nacheinander ausgewählt
  - Auswahl erfolgt nach dem „Nutzen pro Platz“ des neuen MV
- Definition

*Gegeben eine Anfragemenge  $Q$  und eine Menge  $M$  von MV. Der Nutzen eines MV  $a \notin M$  ist*

$$B(a, Q, M) = c(Q, M) - c(Q, M \cup a)$$

*Der Benefit-per-Space (BS) von  $a$  ist*

$$BS(a, Q, M) = \frac{B(a, Q, M)}{|a|}$$

# Eigenschaften

---

- Nutzt den Platz nicht perfekt
  - Terminiert, wenn der nächste beste MV nicht mehr passt
  - Verbesserung: Abarbeitung einer nach Platz und BS sortierten Liste von MV
- Theorem

*Der vorherige Algorithmus liefert eine Menge  $M$  von MV, deren **Nutzen mindestens  $(63-x)\%$  des optimalen Nutzen beträgt***

  - $X$ : Größe des größten MV in Prozent der Gesamtgröße aller MV
- Beweis: Literatur
- Beispiel:
  - Wenn kein View mehr als 10% der Gesamtgröße aller MV benötigt, garantiert der Algorithmus eine Güte von 53%

# Wo sind wir?

---

- Einleitung & Motivation
- Architektur
- Modellierung von Daten im DWH
- Umsetzung des multidimensionalen Datenmodells
- Extraction, Transformation & Load (ETL)
- Indexstrukturen für DWH
- Logische Optimierung
- Materialisierte Sichten
- Data Mining

# Inhalt dieser Vorlesung

---

- Was ist Data Mining?
- Data Mining Prozess
- Typische Problemstellungen & Anwendungen
- Datenaufbereitung
- Deskriptive Datenanalyse
- Oracle Data Mining

# Beispiel

- Wann werden Baseballspiele durchgeführt?
- Beobachtungen des Vortags

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...	...	...	...	...

- Ziel: Vorhersage der Spieldurchführung aufgrund früherer Beobachtungen

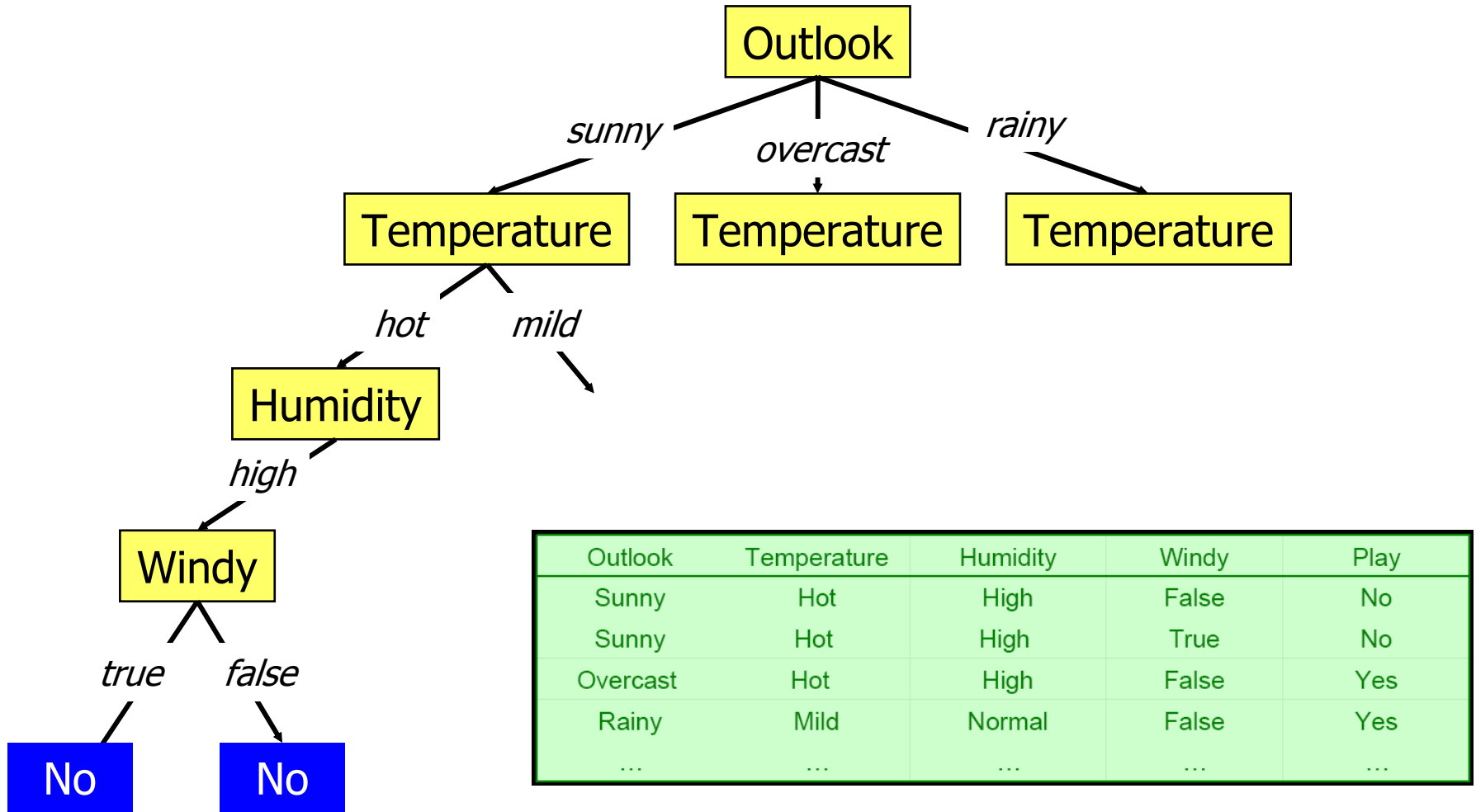
# Vorhersage

---

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Hot	Normal	False	?
...	...	...	...	?

- Einfacher Versuch: Regeln ableiten

# Entscheidungsbäume

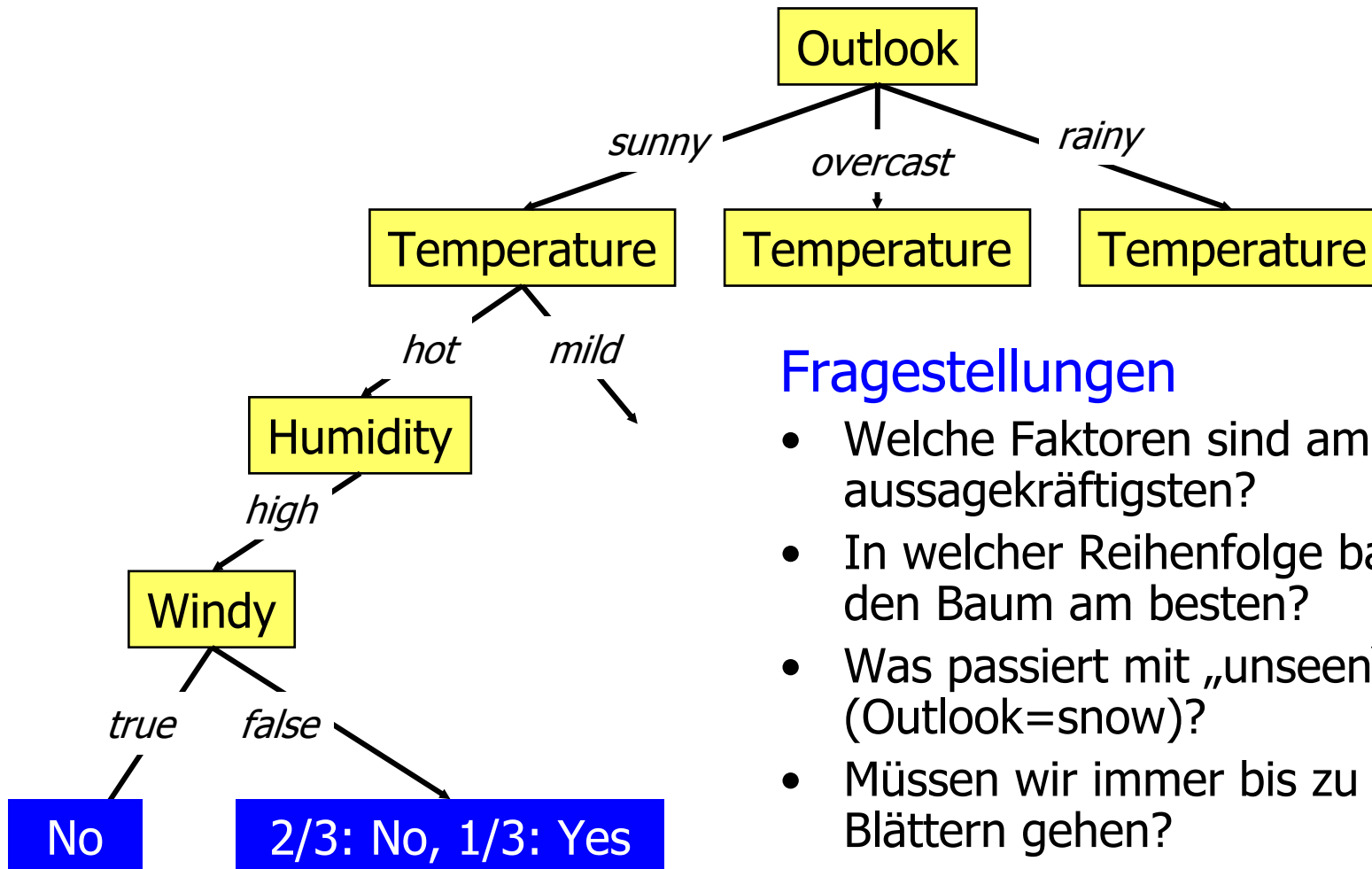


Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...	...	...	...	...

# Mehr Beobachtungen

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...	...	...	...	...
Sunny	Hot	High	False	Yes
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
...	...	...	..	...

# Entscheidungsbäume



## Fragestellungen

- Welche Faktoren sind am aussagekräftigsten?
- In welcher Reihenfolge bauen wir den Baum am besten?
- Was passiert mit „unseen“ Werten (Outlook=snow)?
- Müssen wir immer bis zu den Blättern gehen?

# Entscheidungsregeln

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

- Im wesentlichen eine andere Darstellung des Baumes
- Regeln = **Modell der Wirklichkeit**
  - Gemessen in einer (nicht per-se vorgegebenen) Menge von Attributen
- Extreme Vereinfachung
  - Regeln stimmen nur selten immer
  - **Support**: Auf wie viele der Trainingsdaten passt die Regel?
  - **Confidence**: Für wie viele der passenden Trainingsdaten sagt die Regel das richtige Ergebnis voraus?

# Gestern: Statistik

---

- Früher: Manuelle **statistische Analyse** übersichtlicher Datensätze
  - Eher wenige Datensätze, eher wenig Attribute
- Statistik: Formulieren von Hypothesen und Testen
  - „Hypothesis-driven“
  - „Wie hoch ist die statistische Evidenz, dass das Wetter am Vortag mit der Austragung eines Baseballspiels korreliert?“
  - Hypothesen werden idealerweise vor der Datenanalyse formuliert
    - **Kein Bias**, keine Beeinflussung
    - Beispiel: Doppel-Blind Tests in der Medizin

# Heute: Data Mining

---

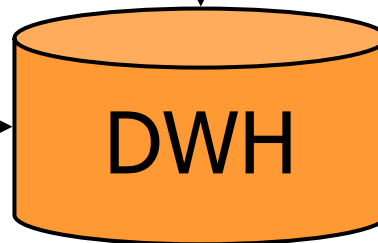
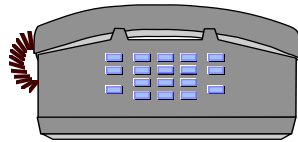
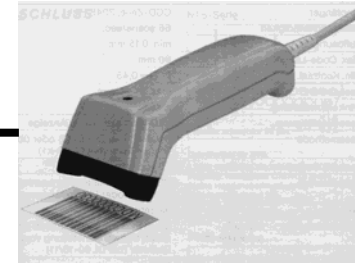
- Riesige **Datenberge**
  - Business: Weblogs, e-commerce, Telekoms, Transaktionen, Börsendaten, Kundendaten, ...
  - Forschung: Astronomie, Teilchenphysik, Bioinformatik, ...
  - Jeder: Nachrichten, Blogs, Webseiten, Fotos, ...
- Manuelle Analyse ausgeschlossen
  - Millionen oder **Milliarden von Datensätzen**
  - Hochdimensionale Daten mit Hunderten von Attributen
  - „**Data-Driven**“: Data Mining generiert die Hypothese und soll sie auch gleich testen
    - Vorsicht: Irgendwas findet man immer
- „We are drowning in data and starving for knowledge“
  - „Was machen Kunden eigentlich auf meiner Webseite?“

# Beispiele

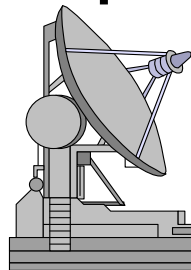
Welche Kunden erreiche ich mit welcher Werbung am Besten?



Welche Assoziationen bestehen zwischen den in einem Supermarkt gekauften Waren?



Bei welchen Telefonkunden besteht der Verdacht eines Betrugs?



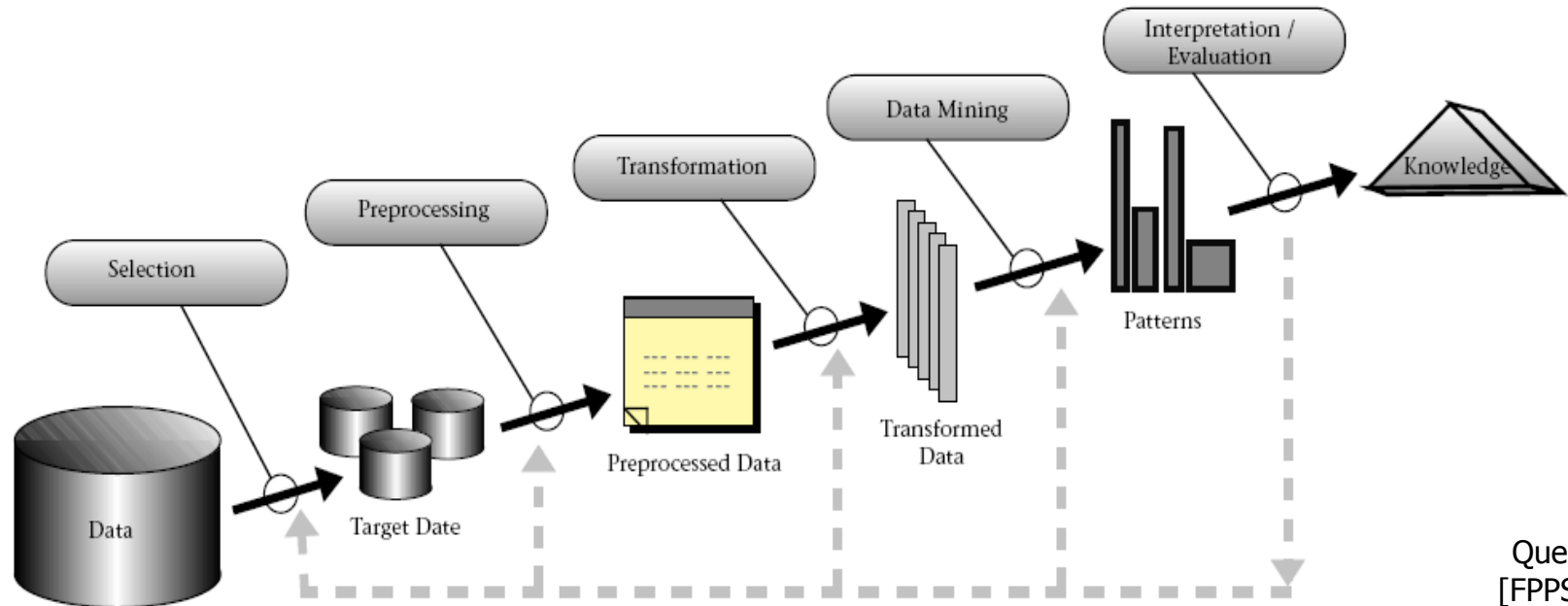
Zu welcher Klasse gehört dieser Stern?

# Knowledge Discovery in Databases [FPSS96]

---

- “KDD is the non-trivial process of identifying **valid, novel, useful and ultimately understandable patterns** in data”
  - Valid: Muster sind im statistischen Sinne valide (signifikant)
  - Novel: Bisher unbekannt
  - Useful: keine hunderten undurchschaubarer Assoziationen
    - Hängt natürlich vom Zweck ab
  - Understandable: Einsicht in die Daten
- „An Art or a Science“?
  - In allen Daten steckt irgendwas
    - Multiples Testing Problem; Overfitting; ...
  - Ableitung „**interessanter**“ **Muster** ist nicht leicht
  - Die Anwendung eines DM Tools ist dagegen sehr leicht

# KDD als Prozess



Quelle:  
[FPPS96]

- Anwendung verstehen
- Daten auswählen
- Data cleaning und preprozessieren
- Datenreduktion (Dimensionen, Gruppierung, ...)
- Explorative Datenanalyse
- Data Mining
- Interpretation und Anwendung der Ergebnisse

# Inhalt dieser Vorlesung

---

- Was ist Data Mining?
- Data Mining Prozess
- **Typische Problemstellungen**
  - Klassifikation
  - Clustering
  - Assoziationsregeln
- Datenaufbereitung
- Deskriptive Datenanalyse
- Oracle Data Mining

# Eingabe

---

- Im folgenden betrachten wir meistens
  - Eine Menge  $O = \{o_1, o_2, \dots, o_n\}$  von **Objekten**
  - Jedes Objekt  $o_i$  wird beschrieben durch die gleiche Menge von Attributen  $A = \{a_1, a_2, \dots, a_m\}$ 
    - Die **Attribute** heißen auch **Dimensionen oder Feature**
  - Die Attributwerte  $o_{ij}$  können kategoriell, diskret, oder kontinuierlich sein
    - Geordnet, halbgeordnet, ungeordnet
  - Eine Klassifikationsfunktion  $f$  ist eine Funktion  $f: O \rightarrow C$ , die Objekte aus  $O$  in eine **Menge von Klassen**  $C = \{C_1, C_2, \dots, C_l\}$  abbildet
    - Man muss das Ergebnis von  $f$  nicht für alle  $o_i$  kennen
    - $l=2$ : Binäres Klassifikationsproblem

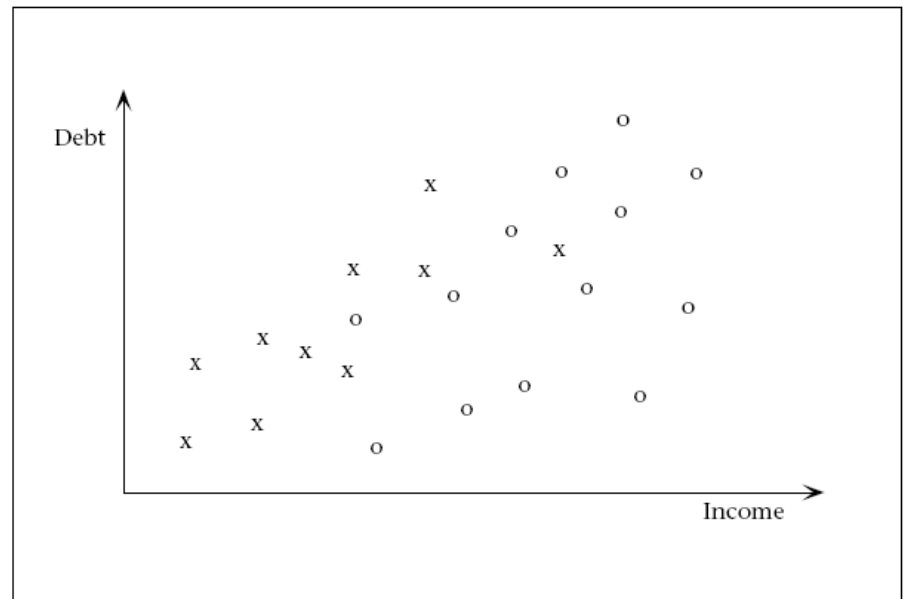
# Die wichtigsten drei Fragestellungen

---

- Klassifikation
  - Gegeben eine Menge von Objekten mit Klassenzugehörigkeit (Trainingsdaten) und eine Menge von Objekten ohne (Testdaten)
  - Frage: **Welcher Klasse** gehören die unklassifizierten Objekte an?
  - Beispiel: Fraud-Detection bei Kreditkarten
- Clustering
  - Gegeben eine Menge von Objekten
  - Frage: Wie kann man diese Objekte in möglichst **homogene Cluster** einteilen?
  - Beispiel: Segmentierung von Kunden
- Assoziationsregeln
  - Gegeben eine Menge von jeweils gemeinsam durchgeführten Aktionen
  - Frage: Welche Aktionen kommen **besonders häufig zusammen** vor?
  - Beispiel: Welche Produkte werden besonders häufig gemeinsam gekauft?

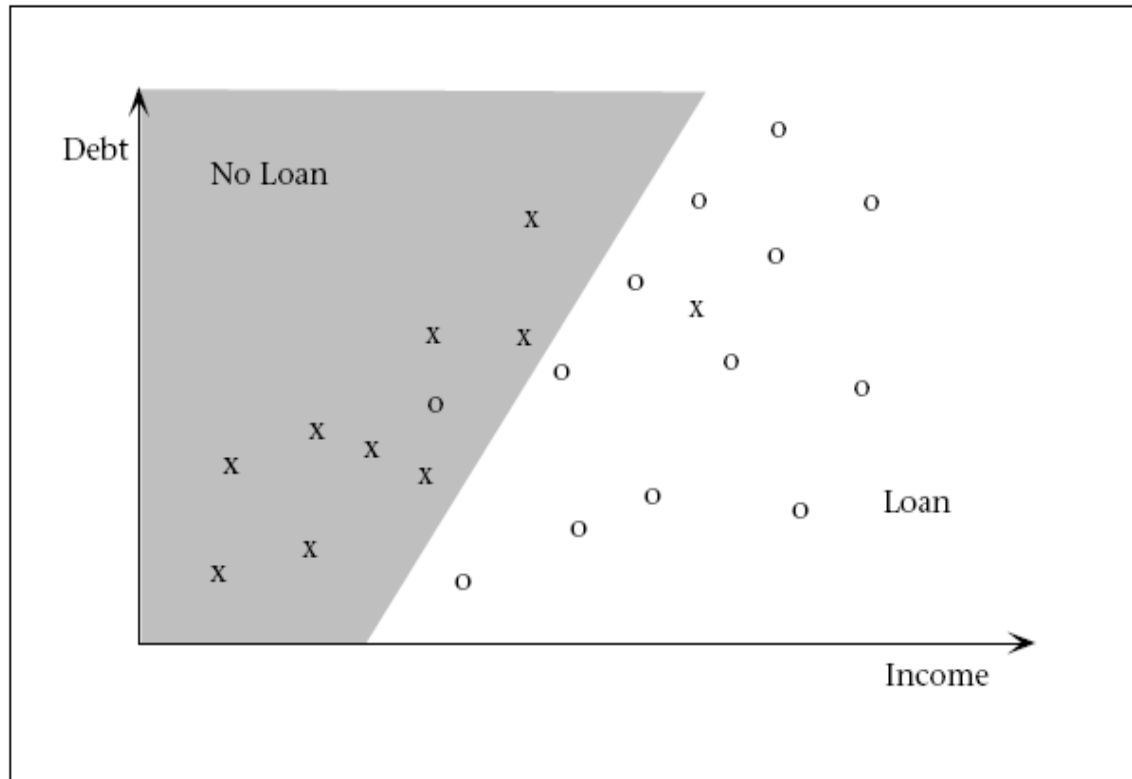
# Klassifikation

- Attribute `debt`, `income`
- Jeder Kunde als Punkt im **zweidimensionalen Raum**
- Klassen:
  - „loan was fine“ (o)
  - „loan was lost“ (x)



- Historische Objekte haben Klassenzugehörigkeit
- Finde die Funktion, die für beliebige Werte (`debt`, `income`) die Klasse berechnet
  - Für neue Kunden also **vorhersagt**

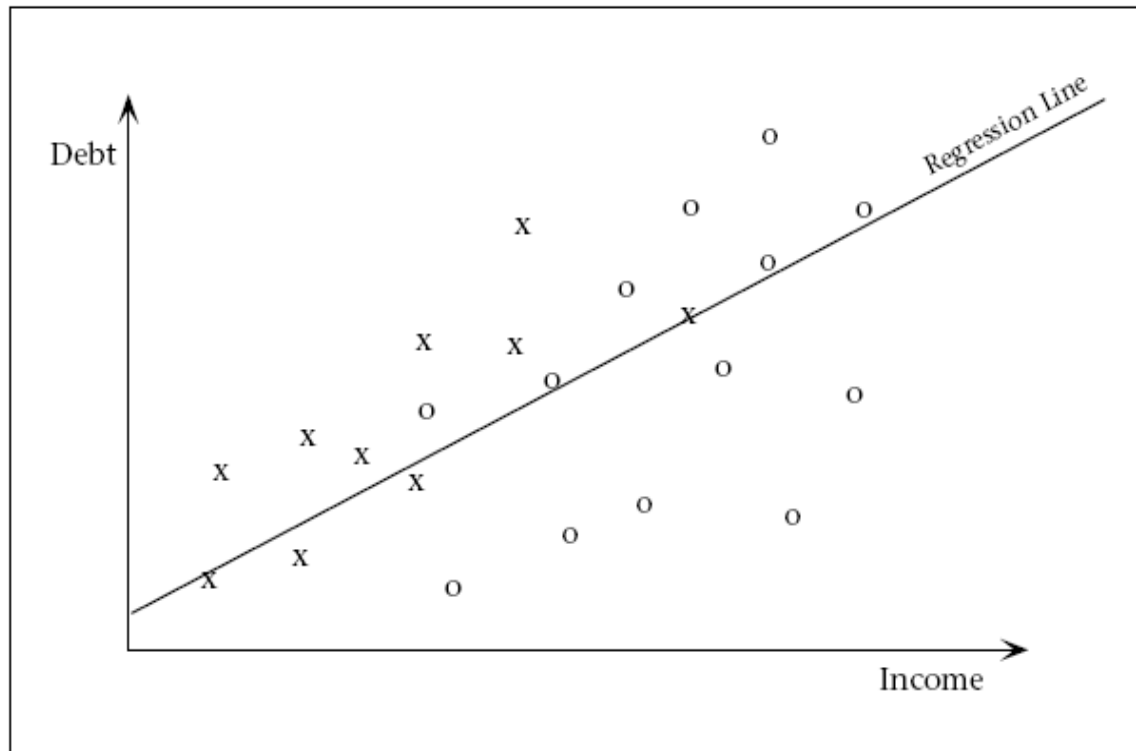
# Lineare Trennung



Quelle:  
[FPPS96]

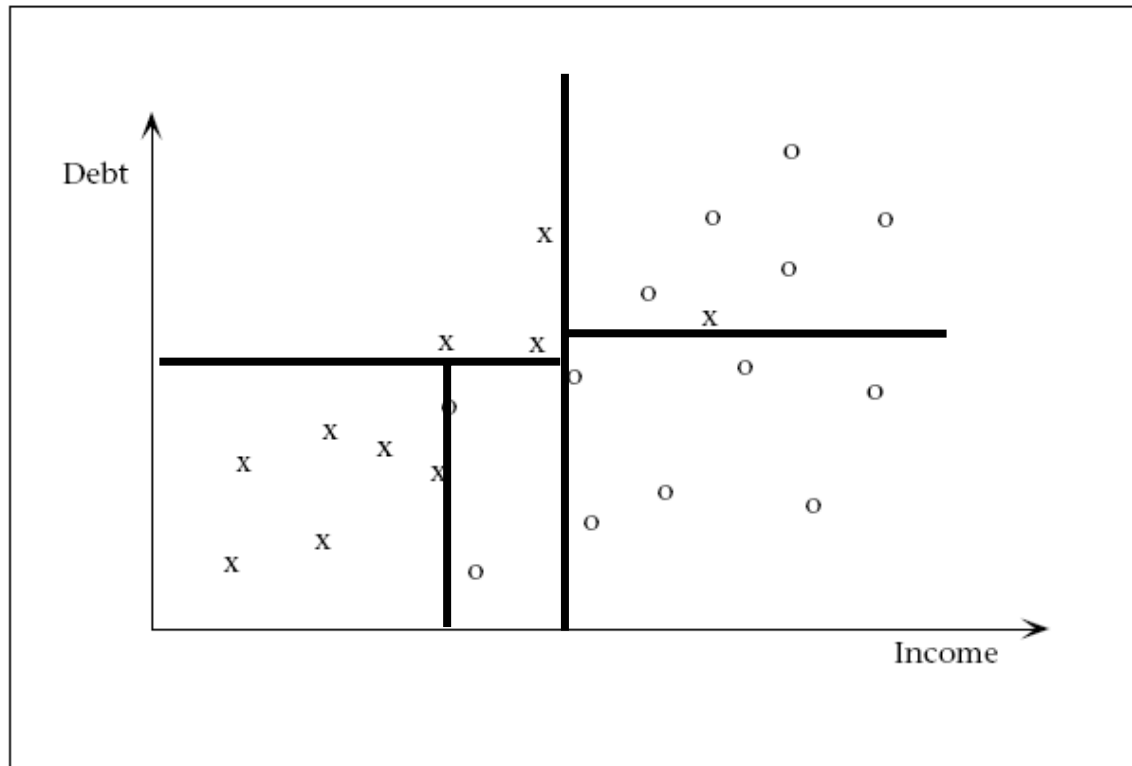
- Inkaufnahme eines gewissen Fehlers (Ausreißer?)

# Lineare Regression



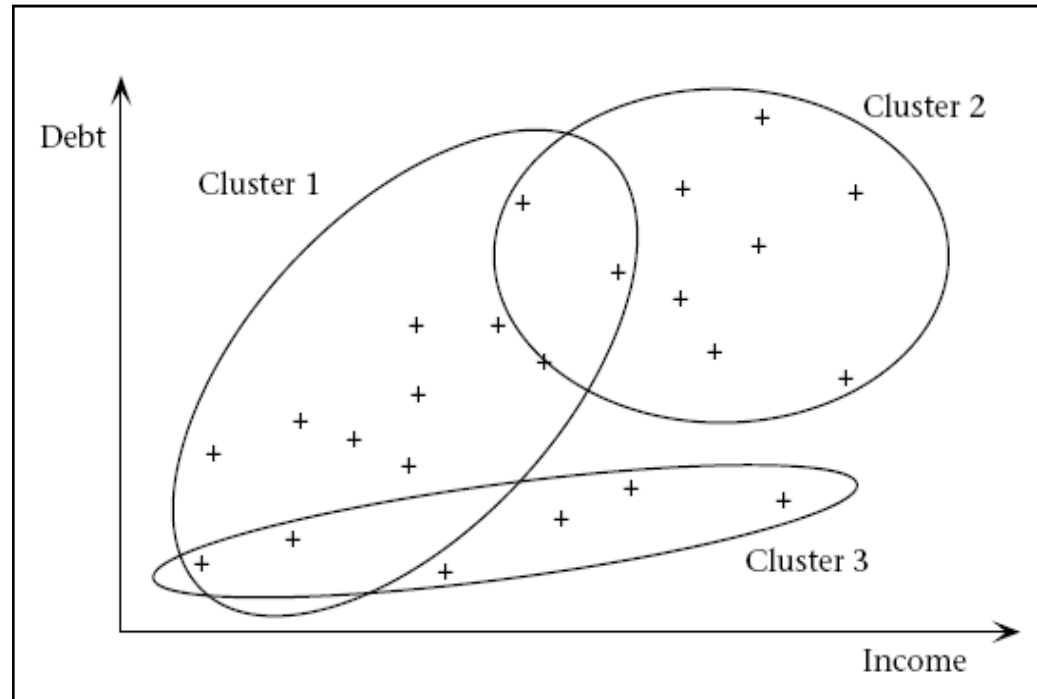
- **Berechnung der Trennfunktion**, die den Fehler minimiert
  - Komplexere Funktionen als lineare sind natürlich möglich
- Geht nur bei numerischen Attributen

# Hierarchische Aufteilung



- **Verwendung lokaler Trennfunktionen**
  - Lokal für einen Subraum der Daten
- Ähnlichkeit zu mehrdimensionalen Indexen ist kein Zufall

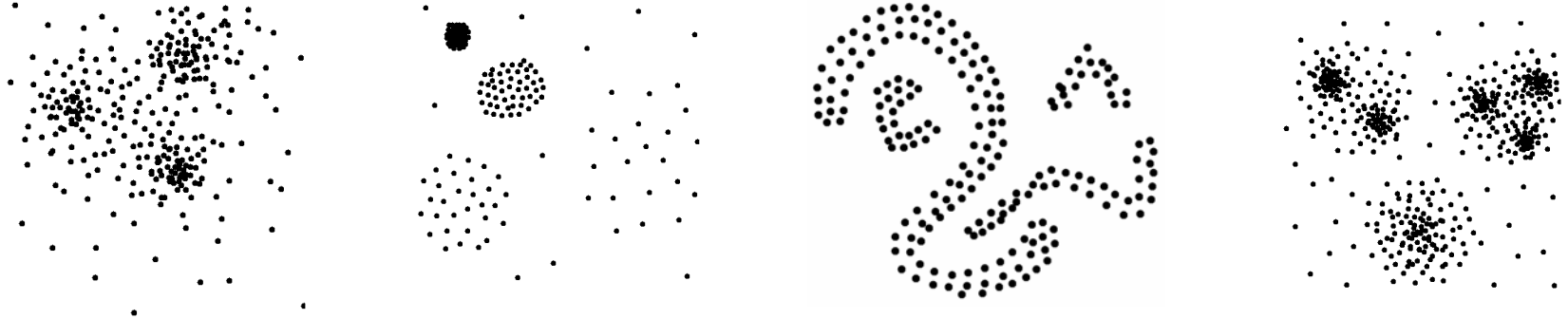
# Clustering



- Finde Gruppen zusammengehörender Objekte
- Basiert auf einem **Abstandsmaß** für Objekte
  - Zusammengehörend = „nahe“ im Raum

# Nicht immer so einfach

---



- Problem ist schillernder und deutlich weniger gut definiert als Klassifikation
  - Wie groß sollen die Cluster ein?
  - Welche Form dürfen die Cluster haben?
  - Wie viele Cluster erwartet man?
  - Müssen alle Punkte geclustert werden?
  - Dürfen sich Cluster überlappen?

Quelle:  
[ES00]

# Association Rule Mining

- Welche Ereignisse (Items) geschehen häufig zusammen?

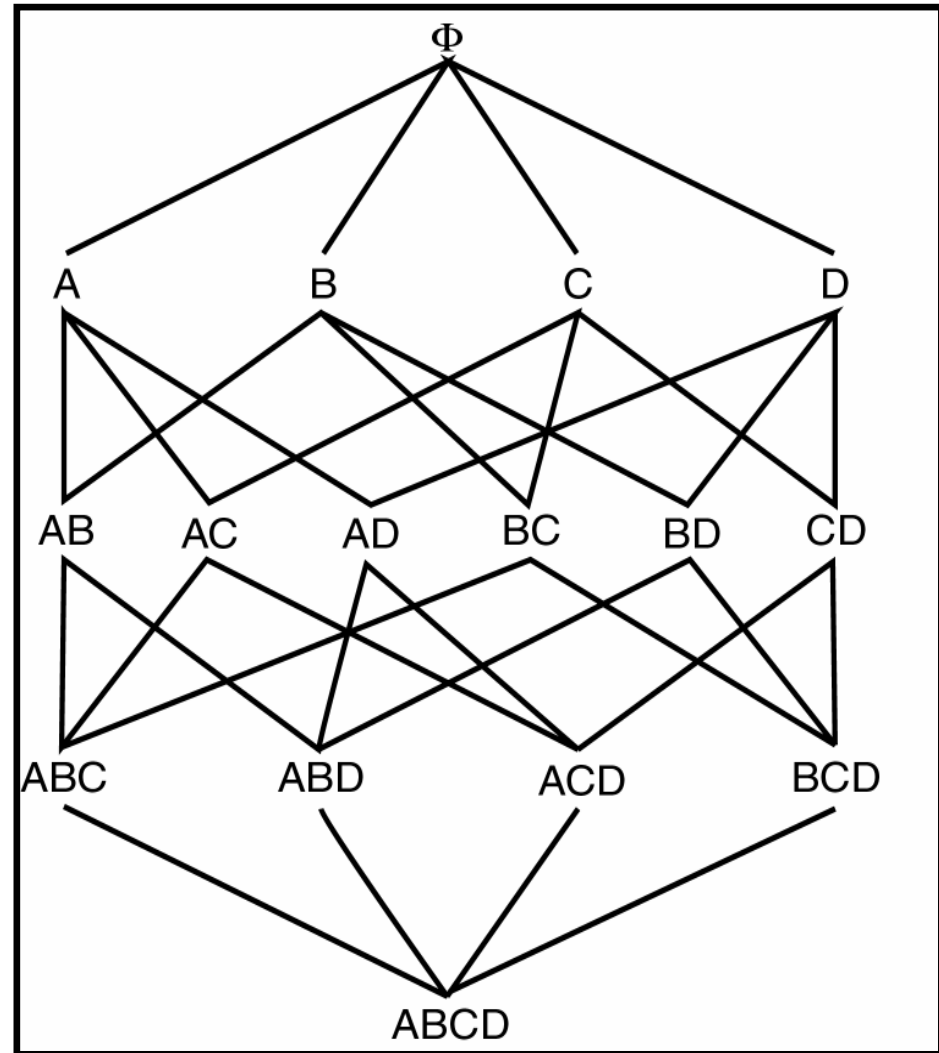
Transaction	Items
$t_1$	Bread, Jelly, Peanut Butter
$t_2$	Bread, Peanut Butter
$t_3$	Bread, Milk, Peanut Butter
$t_4$	Beer, Bread
$t_5$	Beer, Milk

Quelle:  
[Dun02]

- Problem: Es gibt so viele mögliche Itemsets!
  - Wie viele?

# Grundprinzip: „Large Itemset property“

- Jede Subgruppe eines häufigen Itemsets muss häufig sein
- ... oder ...
- Häufige große Itemsets müssen aus häufigen kleinen Itemsets bestehen



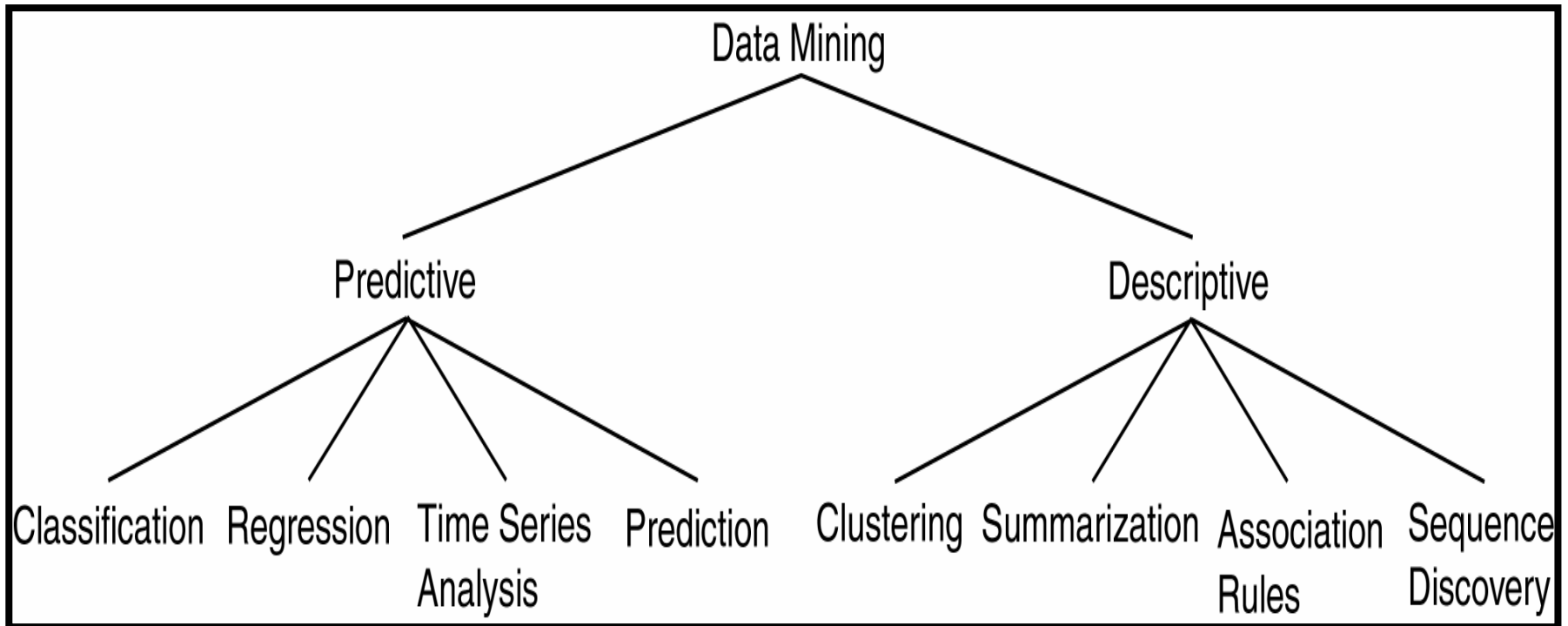
# Weitere Themen

---

- **Text-Mining**: Clustering und Klassifikation von Texten
  - Wie weit sind zwei Texte voneinander entfernt?
  - Featureräume mit 100.000 Dimensionen
- **Web-Mining**
  - Web Usage: Welche Webseiten werden häufig in einer bestimmten Reihenfolge besucht
  - Web Structure: Einbeziehung der Linkstruktur von Webseiten in die Aufgaben (wie z.B. Klassifikation oder Ähnlichkeit)
- **Spatial Mining**
  - Objekte mit geographischem Bezug
- **Graph-Mining**
  - Darstellung von Objekten als vernetzte Graphen (z.B. social networks)
  - Ähnliche Objekte = ähnliche Subgraphen
- ...

# Klassifikation [Dun02]

---



# Beispiele aus der Praxis [AN00]

---

- Bonitätsprüfung im Versandhandel
  - Klassifikation von Kunden nach Bonität
  - Gelernt aus ~5000 Beispielen mit ~100 Features
    - Abbuchungen, Zahlungsverhalten, ...
  - 1-2% Verbesserung in der Vorhersage bringt >Millionen Euro
- Verbundkäufe in Warenkorbdaten
- Kundensegmentierung aus Bons
- Kundensegmentierung in der Telekommunikation aus Nutzungsverhalten
  - Welche Tarife anbieten? Welche Tarife schaffen?
- Tarifeinstufung von Kunden bei Versicherungen
  - Zitat: „80% der Zeit geht für Datenauswahl (welche Feature?) und Datenaufbereitung drauf.“

# Inhalt dieser Vorlesung

---

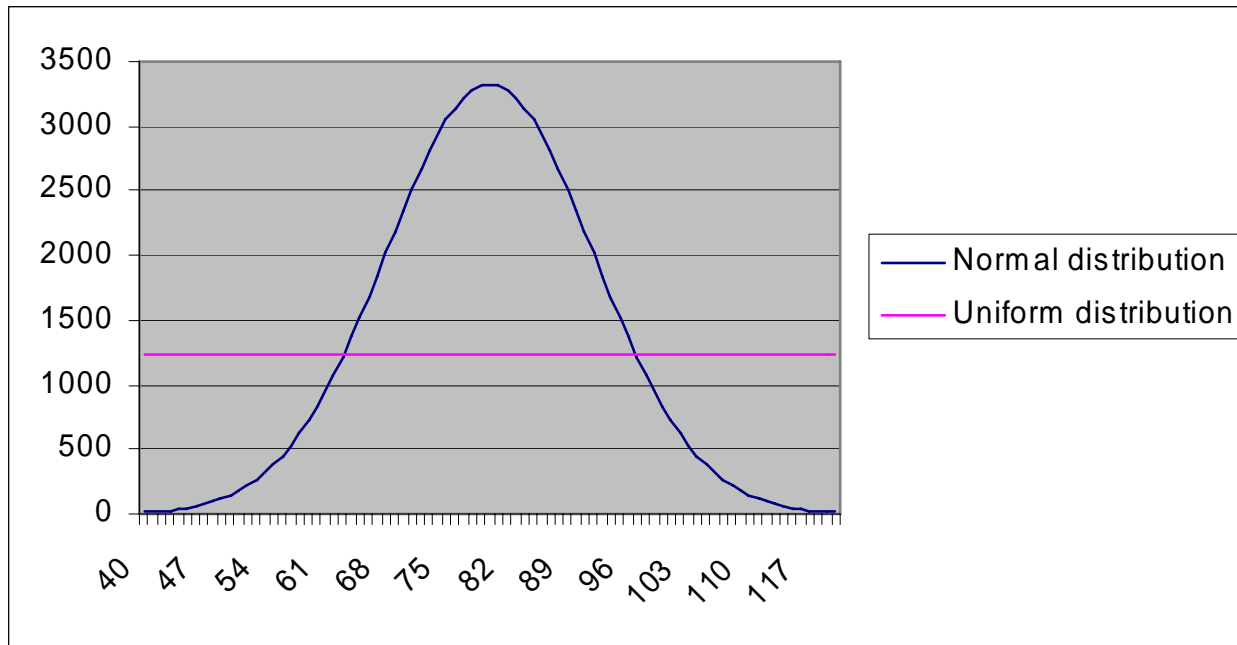
- Was ist Data Mining?
- Data Mining Prozess
- Typische Problemstellungen
- Datenaufbereitung
- Deskriptive Datenanalyse
- Oracle Data Mining

# Datenaufbereitung

---

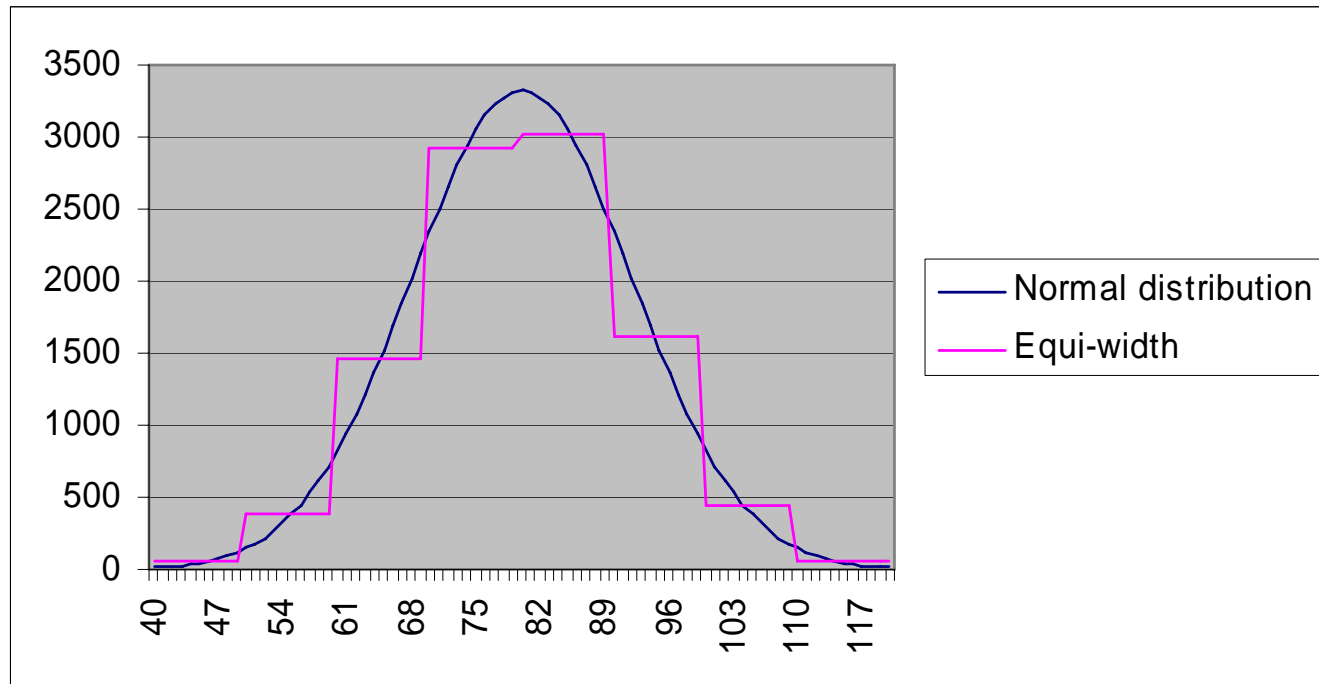
- Viele DM Verfahren reagieren empfindlich auf **Ausreißer, fehlende Werte, Datenfehler** etc.
- Preprocessing: Herstellung einer homogenen und bereinigten Datenbasis
  - Das ist unser ETL Prozess: **Erhöhung der Datenqualität**
- Beispiele
  - Ersetzung von fehlenden Werten durch Schätzen, Extrapolation
    - Aber Datenverteilung muss erhalten bleiben
  - Diskretisierung von Werten (Binning)
    - Z.B. Einteilung des Einkommens von Kunden in 5 Bereiche
    - Glättet Ausreißer, reduziert die Zahl verschiedener Werte
  - Ranking von Werten
    - Statt absoluten Einkommen wird der Rang benutzt
    - Glättet Ausreißern
    - Lässt aber auch ev. wichtige Unterschiede verschwinden

# Binning



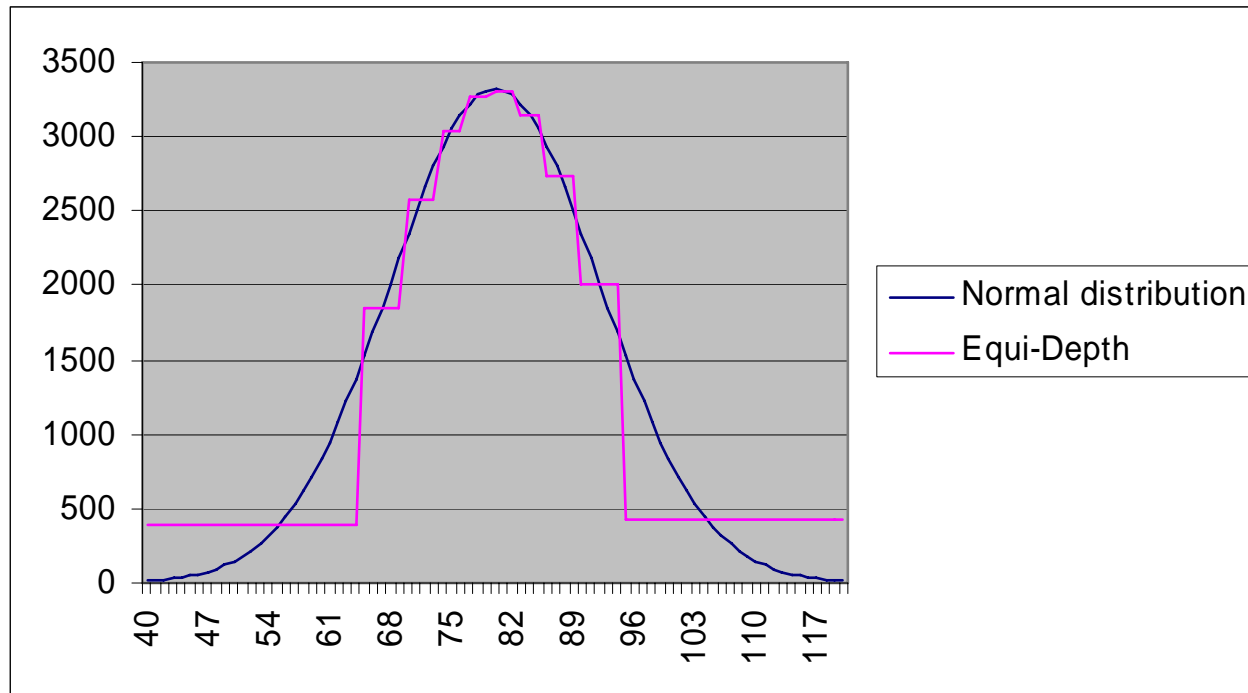
- Extremfall: Ersetzen durch einen Wert
- Schlechte Idee

# Equi-Width Histograms



- Zahl der Bins festlegen und **Raum äquidistant aufteilen**
- Bins enthalten unterschiedlich viele Werte und decken den ganzen Raum gleichmäßig ab
- Berechnung durch Sortierung und einen Scan

# Equi-Depth



- Zahl der Bins festlegen, dann Raum so aufteilen, dass alle **Bins gleich viele Tupel** enthalten
- Führt zu gleichgroßen Bins mit unterschiedlicher Breite

# Inhalt dieser Vorlesung

---

- Was ist Data Mining?
- Data Mining Prozess
- Typische Problemstellungen
- Datenaufbereitung
- **Deskriptive Datenanalyse**
- Oracle Data Mining

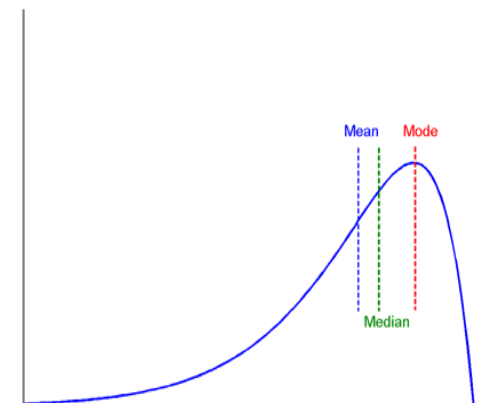
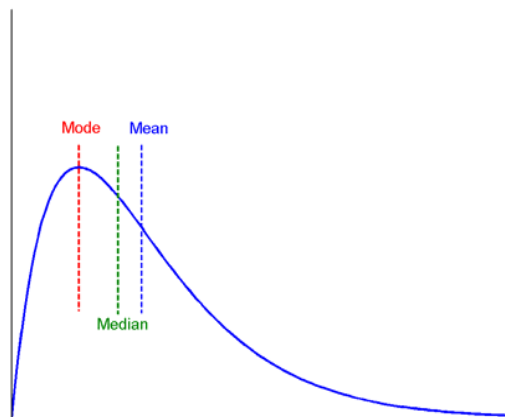
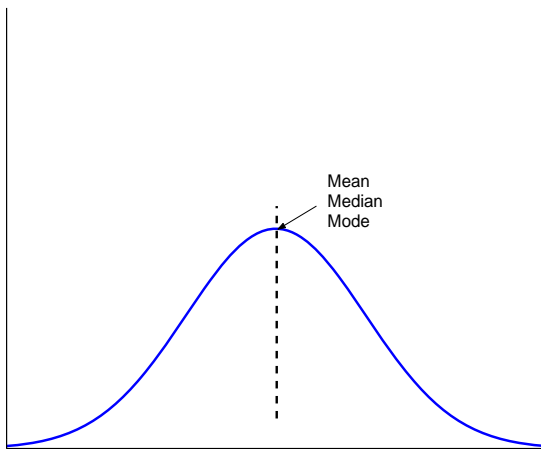
# Descriptive Datenanalyse

---

- Ziel: Ein „Gefühl“ für die Daten bekommen
  - „Welche Werte sind wie häufig?“
  - „Unterliegen die Werte einer bestimmten Häufigkeitsverteilung?“
  - „Sind zwei (oder mehr) Attribute stark korreliert?“
  - „Welche Attribute trennen meine Klassen besonders gut?“
- Das ist bei 2.000.000.000 Tupeln gar nicht so einfach
- Wichtige Vorbereitung zur **Auswahl des Data Mining Verfahrens**
- Hier: Nur ganz einfache statistische Kennwerte
  - Und deren Berechnung im DWH

# Univariate Beschreibung

- Beschreibung der Verteilung eines einzelnen Attributs
- Häufigste Werte
  - Mittelwert: Der Mittelwert
  - Median: Der mittlere Wert
  - Mode: Der häufigste Wert
- „Skewed data“ (Abweichung von Normalverteilung)

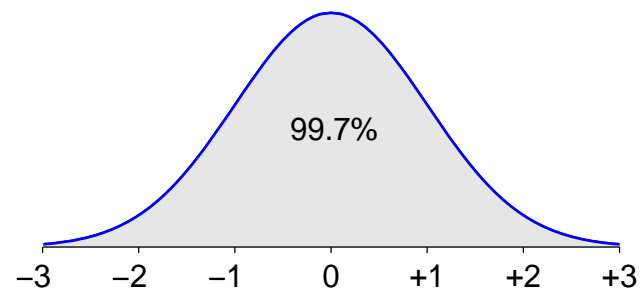
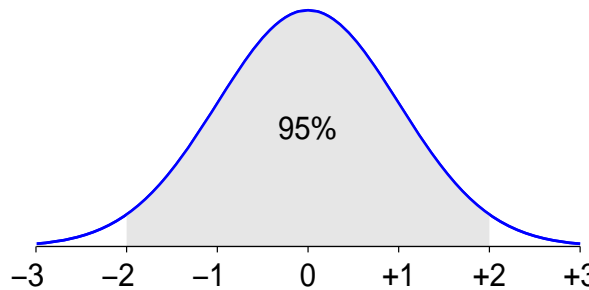
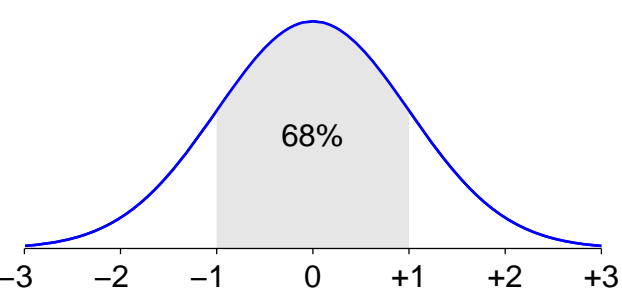


Quelle: [HK05]

# Normalverteilte Daten

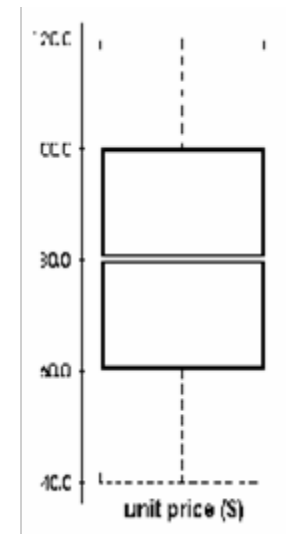
---

- Weitere Werte: **Standardabweichung** und Varianz
  - $[\mu - \sigma, \mu + \sigma]$ : Ca. 68% der Datenpunkte
  - $[\mu - 2\sigma, \mu + 2\sigma]$ : Ca. 95% der Datenpunkte
  - $[\mu - 3\sigma, \mu + 3\sigma]$ : >99% der Datenpunkte

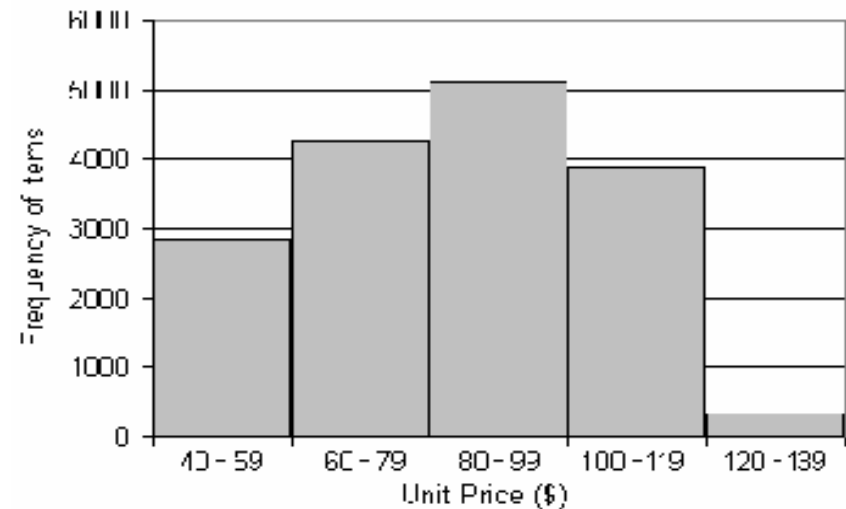


# Visualisierung

- Boxplots – auf einen Blick
  - Min und max
  - Erstes und drittes Quartiel
  - Mittelwert und (meist) Median



- Histogramme



# SQL

---

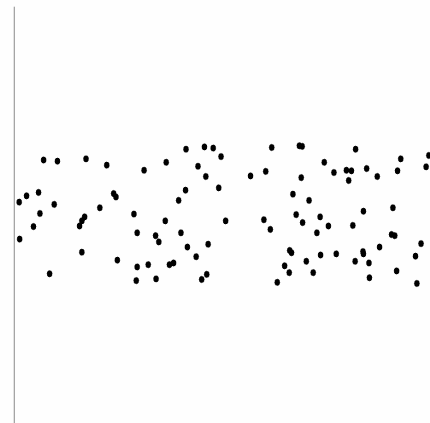
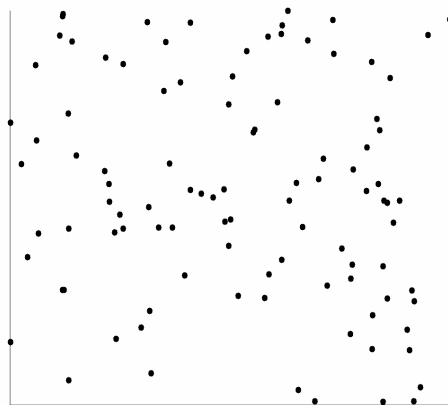
- avg, stddev, median, quartile, sind alles Standard SQL Funktionen
- Aber wie findet man den mode des Attributs t.a?
  - mode = häufigster Wert

```
SELECT a, cnt
FROM (SELECT a, count(a) cnt
      FROM t
      GROUP BY a
      ORDER BY count(a))
WHERE ROWNUM=1;
```

# Multivariate Beschreibung

---

- Gleichzeitige Betrachtung der Verteilungen zweier (oder mehr) Attribute
- Im einfachsten Fall sind die beiden Attribute **unabhängig** voneinander verteilt
  - Dann lohnt sich die gemeinsame Beschreibung nicht
  - Keine Korrelation
  - Visualisierung im Scatter-Plot



# Kontingenztafeln

- Oft sind Attribute nicht unabhängig voneinander
  - Wir werden das trotzdem oft einfach annehmen um Dinge einfach und schnell zu halten
- Kontingenztafel für kategorielle Attribute

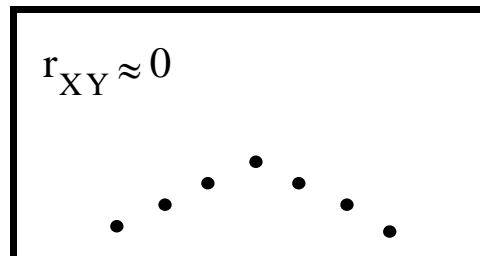
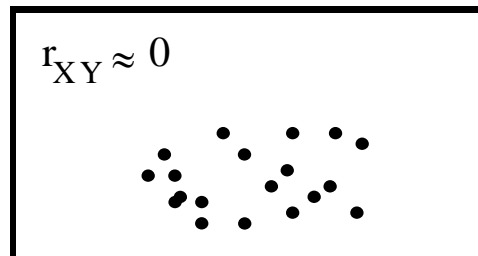
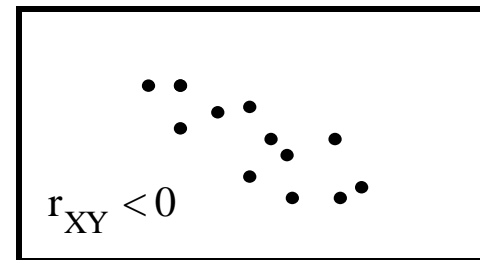
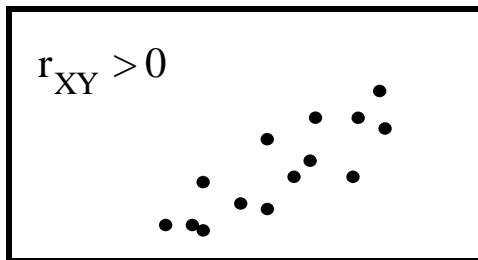
	Mittelfristig Arbeitslos	Langfristig Arbeitslos	Summen
Ohne Ausbildung	19	18	37
Mit abgeschlossener Ausbildung	43	20	63
Summe	62	38	100

- Was erwartet man für unabhängige Attribute?
- Korrelationskoeffizient bemisst die Abweichung

# Korrelationskoeffizient

- Misst die (lineare) Korrelation zweier Attribute X und Y

$$r_{XY} = \frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$



# SQL

$$r_{XY} = \frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- Berechnung Kontingenztabelle für Attribute t.a und t.b?

```
SELECT  a,b,count(*)
FROM    t
GROUP BY cube(a,b);
```

- Berechnung des Korrelationskoeffizienten für t.a und t.b?

```
SELECT up/sqrt(down)
FROM (SELECT sum((a-ma)*(b-mb)) up
      FROM t, (SELECT avg(a) ma, avg(b) mb
               FROM t) tm),
      (SELECT sum(sqr(a-ma))*sum(sqr(b-mb)) down
      FROM t, (SELECT avg(a) ma, avg(b) mb
               FROM t) tm);
```

# Inhalt dieser Vorlesung

---

- Was ist Data Mining?
- Data Mining Prozess
- Typische Problemstellungen
- Datenaufbereitung
- Deskriptive Datenanalyse
- Oracle Data Mining

# Oracle Data Mining (ODM)

---

- Menge von typischen Data Mining Algorithmen
- Alle Daten müssen **in einer Tabelle** vorliegen
  - Kein Multi-Relational Mining
- Zugriff über **zwei APIs**
  - PL/SQL Packages DBMS\_DATA\_MINING and DBMS\_DATA\_MINING\_TRANSFORM
  - Java API
  - Gleich Funktionalität
- ODM Models
  - Die meisten Verfahren berechnen Modelle
  - **Modelle werden in der DB gespeichert** und können ausgetauscht, angewandt, exportiert und importiert werden

# ODM Algorithmen

---

- Klassifikationsalgorithmen
  - Decision Tree
  - Naive Bayes und Adaptive Bayes Networks
  - SVM mit linearen Kernel
- Regression
- Clustering
  - K-Means
    - Euklidischer- oder Cosinus-Abstand
    - Cluster werden durch Centroide repräsentiert
  - Orthogonal Partitioning Clustering
    - Hierarchisches Clustering mit achsenparallelen Schnitten
- Association Rule Mining
  - A-Priori Algorithmus mit Min-Support/Confidence Filtering
- Text Mining
  - Clustering mit k-Means
  - Klassifikation mit SVM

# Literatur

---

- Han, J. and Kamber, M. (2006). "Data Mining. Concepts and Techniques", Morgan Kaufmann.
- Alpar, P. and Niedereichholz, J., Eds. (2000). "Data Mining im praktischen Einsatz". Braunschweig/Wiesbaden, Vieweg Verlagsgesellschaft.
- Dunham, A. M. H. (2002). "Data Mining". New Jersey, Pearson Education Inc.
- Ester, M. and Sander, J. (2000). "Knowledge Discovery in Databases". Berlin, Springer.
- Fayyad, U. M., Piatetsky-Shapiro, G. and Smyth, P. (1996). "From Data Mining to Knowledge Discovery in Databases." AI Magazine 17(3): 37-54.
- Ganti, V., Gehrke, J. and Ramakrishnan, R. (1999). "Mining Very Large Databases." IEEE Computer: 38-45.