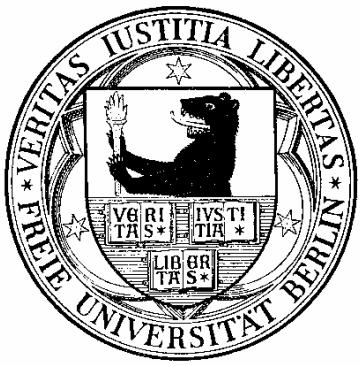


Datenmodellierung

und

Datenbanksysteme

Foliensatz zur Vorlesung
WS 2005 / 2006

Version 8.0

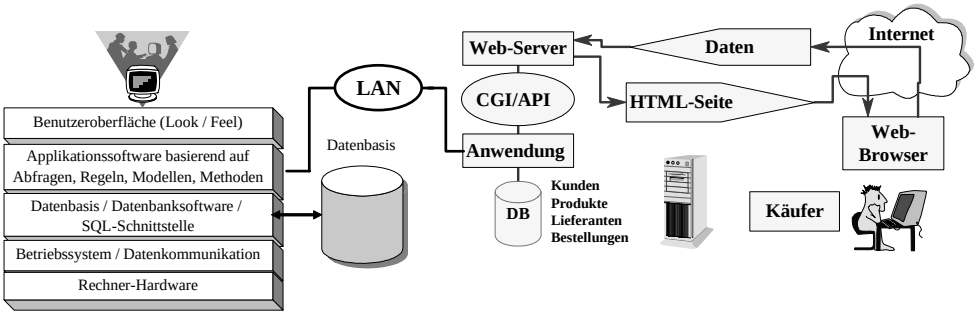
Uwe H. Suhl
Freie Universität Berlin
Lehrstuhl für Wirtschaftsinformatik
suhl@wiwiss.fu-berlin.de
http://www.wiwiss.fu-berlin.de/suhl/

Inhaltsverzeichnis

1	**Einführung** Einleitung, Informationsverarbeitung und Datenbanksysteme im Wandel, Bedeutung der Datenbasis und Entwicklungstrend, Historische Entwicklung der IV, Systementwicklung und Datenbanken, Datenintegrität	**1**
2	**Datenbankkonzepte** Grundlegende Datenbankkonzepte, Abstraktionsebenen und Schemata, Merkmale und Systemfunktionen von Datenbanksystemen, Datenbanken und Web, Datenkataloge, Kostenaspekte, Datenbankadministration	**7**
3	**Konzeptionelle Datenmodellierung** Datenmodelle, Begriffe der ER-Modellierung, Kardinalitäten von Beziehungen, UML zur Datenmodellierung, Grundregeln der Datenmodellierung, Beispiele, Phasenschema zur Datenmodellierung	**14**
4	**Logische Datenmodelle in Datenbanksystemen** Relationenmodell, Objektorientiertes Datenmodell, Objektrelationale Datenabnken, Vertiefung des Relationenmodells, relationale Operatoren	**27**
5	**Relationales Datenbankdesign (1)** Problembereiche beim Entwurf der Relationstypen, Normalisierung von Relationen und Normalformen, Grundlegende Definitionen, Normalformen auf der Basis funktionaler Abhängigkeiten (1NF – BCNF)	**34**
6	**Structured Query Language (SQL)** Einführung, SQL-Normen, Open SQL, SQL-Formen, Anlegen einer Beispieldatenbank, SQL-Sprachelemente, Implementation der relationalen Operatoren in SQL, Select-Beispiele, Unterabfragen, Spaltenfunktionen, Views, Tabellendaten übertragen, Indexe, Datenkontrolle, SQL-3	**40**
7	**Programmierung von Datenbank-Anwendungen** Einbettung von SQL in eine Programmiersprache, prinzipielle Möglichkeiten des SQL Zugriffs auf DB, ODBC, MS-Access, Datenbankzugriffe mit embedded SQL, SQL Cursor Management, JDBC, SQLJ, Stored Procedures und Triggers, Transaktionsverarbeitung	**53**
8	**Relationales Datenbankdesign (2)** Verlustfreie und abhängigkeitstreue Zerlegungen, mehrwertige Abhängigkeiten, 4NF, PJ-Abhängigkeiten, 5NF, Datenintegrität, Datenbankschema	**64**
9	**Relationale Modellierungsprobleme** Modellierung von: überlappenden Objektklassen, Preisstaffeln, Belegungsproblemen, Strukturstücklisten, symmetrischen Straßennetzen, Historisierung von Daten, Maschinenbelegungsplan, Artikel und Artikelmengen	**72**
10	**Grundlagen physischer Datenspeicherung auf Festplatten** Leistungsaspekte, Speichersysteme, relative und indizierte Dateien, Indexe, B-Bäume	**78**
11	**Data Mining Grundlagen** Data Mining Problemstellung, Data Warehouse, Grundprozesse des Data Mining, OLTP, OLAP und Data Mining, wichtige Methoden des Data Mining, Clustering, Assoziationsanalyse, Klassifikationsprobleme, Anbieter	**86**
12	**Literaturverzeichnis**	**101**

1. Einführung

- **Betriebliche und Web-basierte Anwendungssysteme operieren heute überwiegend auf relationalen Datenbanken.**
- **Das Datenmodell für die geplanten Anwendungsfunktionen, Auswahl und Einsatz von DBMS, die Planung der Anwendungsarchitektur sind für ein Unternehmen von strategischer Bedeutung.**
- **Das größte Wachstum verzeichnen Datenbanksysteme für Web-Anwendungen; die Datenbank liegt dabei i.d.R. auf einem Web-Server**
- **E-Business-Systeme sind i.d.R. lose mit den „Legacy Systemen“ gekoppelt**

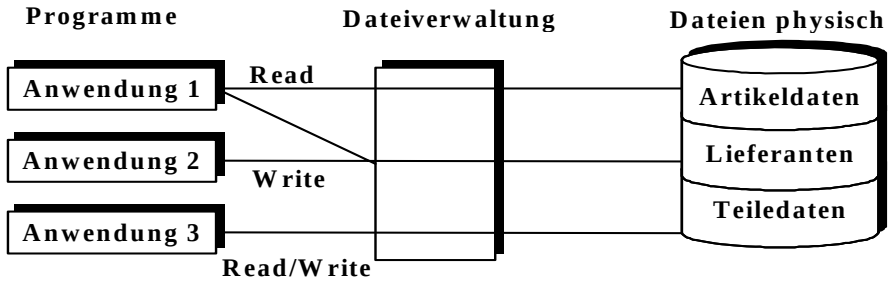


Datenbankentwurf

- **Aufgabe des Datenbankentwurfs ist die Planung der manuell oder maschinell zu verarbeitenden Daten für die Anwendungssysteme.**
- **Eine sorgfältig konzipierte und dokumentierte Datenbasis ist Voraussetzung für die Entwicklung integrierter Anwendersysteme.**
- **Wesentlicher Bestandteil des Datenentwurfs ist die *Datenmodellierung*, die als Ergebnis ein Datenmodell des betrachteten Anwendungssystems liefert.**
- **Im Datenmodell werden durch Abstraktionsprozesse die relevanten *Objekt-* und *Beziehungstypen* sowie deren *Attribute* und *Schlüssel* festgelegt.**
- **Datenmanagement spielt im Rahmen des Informationsmanagements eine wichtige Rolle und umfaßt folgende Aufgabenbereiche:**
  - ***Datenmodellierung* mit Aufbau unternehmensweiter Datenmodelle**
  - ***Datenstandardisierung* sämtlicher Datenelemente im Unternehmen**
  - ***Datennutzungsadministration*: Datenbereitstellung für Anwendungssysteme**
  - ***Einsatz u. Betrieb* von Datenbanksoftware (→ Datenbanksysteme)**
  - ***Logisches und physisches Datenbankdesign*, Normalisierung, Datenverteilung, Zugriffspfade (Indexe)**
  - **Eng verbunden sind Aspekte der *physischen Speicherung* von Massendaten: Network Attached Storage (NAS) bezeichnet plattformunabhängige Speichersysteme (z.B. EMC²)**

Konventionelle Dateiverarbeitung (Historie)

- **Dateiorganisation, Zugriffsmethode und Satzstrukturen der Dateien sind in die Logik der Anwenderprogramme eingebunden.**
- **Bereits Satzänderungen führen zu Änderungen von Programmen oder Include-Dateien mit Satzbeschreibungen.**
- **Makrobibliotheken müssen neu generiert und Programme neu kompiliert und gelinkt werden.**
- **Vorteile ergeben sich für die konventionelle Dateiverarbeitung, wenn es auf die kürztest mögliche Zugriffszeit ankommt.**
- **Bei zeitkritischen Anwendungen kann man auch DB-Tabellen in Hauptspeicher-Datenstrukturen verarbeiten.**



Beispiel aus einem COBOL Programm

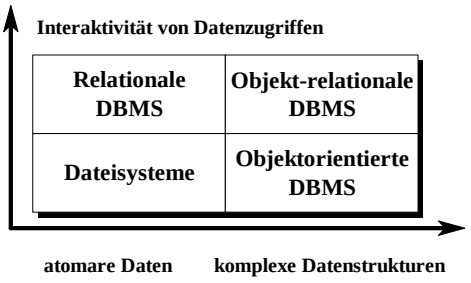
```
file-control.
select optional sportdatei assign to "sport",
        organization is indexed,
        access mode is dynamic record key is nr,
        alternate key is zname with duplicates.
data division.
file section.
fd sportdatei.
01 satz.
         05 nr            pic 999.
         05 name.
                  10 vname    pic x(15).
                  10 zname    pic x(18).
        05 adresse.
                  10 strasse    pic x(15).
                  10 plz         pic 9(4).
                  10 ort         pic x(20).
                    .....................
read sportdatei invalid key .............,
        not invalid key ..................
```

Wandel der IV und der DBMS

- **Heutige RDBMS sind für Transaktionsverarbeitung konzipiert / optimiert.**
- **Internet-, Multimedia- u. Data-Warehouse-Anwendungen werden wichtiger:**

OLTP **Transaktionsverarbeitung** **OLIP** **Internet Processing**	**OLCP** **Complex Processing z.B. in Data-Warehouse-Anwendungen** **OLAP** **Datenanalye / Data Mining**
Workflow **Vorgangsbearbeitung am PC mit automatischer Weiterleitung an andere Bearbeiter**	**Visualisierung** **verstärkter Einsatz von Bildbearbeitung und Mustervergleichen**

- **Diese Anforderungen führen graduell zu neuen RDBMS, die objekt-relationale Erweiterungen bieten, u.a. LOBs (Large Objects) mit verschiedenen Datentypen und Funktionen.**



Transaktionsorientierte und analyseorientierte IV

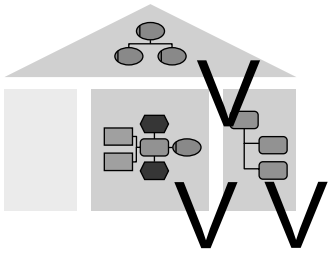
- **Operative IV (OLTP) unterstützt das Tagesgeschäft, z.B. Erfassung von Rechnungseingängen, Lagerbestandsfortschreibung etc.**
- **Die Datenbasis wird i.d.R. in relationalen Datenbanken gespeichert.**
- **Eine typische Frage, die mit SQL beantwortet werden kann, ist:**
  Wie hoch ist der momentane Lagerbestand von Artikel 4711 im Lager x ?
- **Zunehmend wichtiger werden Fragestellungen, die nach Mustern fragen:**
  - **Wie häufig hat ein Kunde, der Produkt x gekauft hat, auch Produkt y gekauft?**
  - **Welche Attribute sind für einen Kunden kennzeichnend, der überproportional häufig in Autounfälle verwickelt ist**
  - **Welche Merkmale sind bei Risikokrediten typisch?**
- **Für diese Fragestellungen sind RDBMS nur begrenzt einsatzfähig.**
- **Wichtige Unterschiede zwischen transaktionsorientierter und analyseorientierter IV:**

Einsatzbereich	transaktionsorientiert	analyseorientiert
Datenmanipulation	tupelorientiert	sichtspezifisch
Datenmenge	i.d.R. klein	sehr groß
Betrachtungsebene	detailliert	aggregiert
Zeithorizont	detailliert	historisch, gegenwärtig

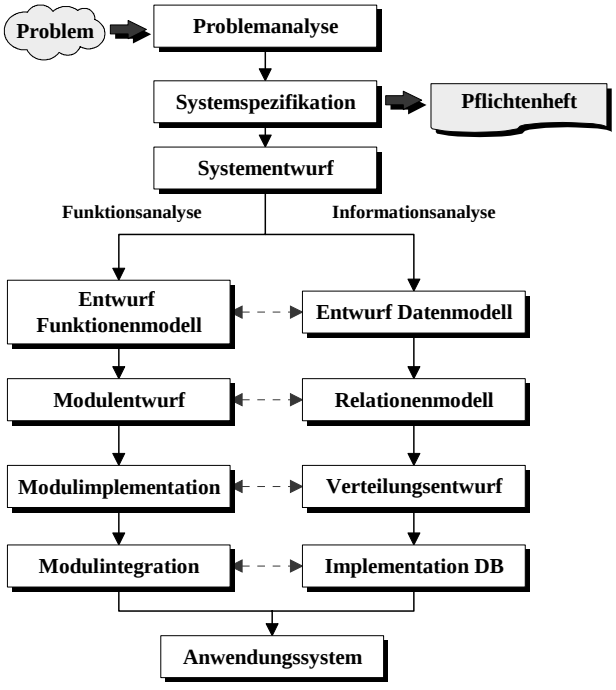
Systementwicklung und Datenbankdesign

- ***Funktionenmodell*: Systemaufgaben werden funktionsorientiert beschrieben und aus den Anforderungen des *Pflichtenhefts* abgeleitet.**
- ***Benutzeroberfläche*: zeigt das System aus Anwendersicht (Masken, Reports).**
- ***Datenmodell*: konzeptionelle (logische) Beschreibung aller Informations-objekte der Anwendung; basiert auf einem Modellierungsprozess.**
- ***Organisationskonzept*: Beschreibung der Abläufe und Funktionen im Umfeld des Unternehmens und seiner Aufbauorganisation (EPK).**
- ***Systemkonfiguration*: Systemplattform mit weiterer Basissoftware.**
- ***Datenbankdesign*: Verfeinerung des konzeptionellen Datenmodells in Form von Relationstypen; Basis für den Einsatz relationaler Datenbanksysteme.**
- ***Systemarchitektur*: im Softwareentwurf wird das *Funktionenmodell* verfeinert und u.a. in Modulspezifikationen des Anwendungssystems überführt; Module werden in der Implementation *programmiert*.**
- ***Kommunikationsmodell*: vernetzte Anwendungssysteme erfordern ggf. eine *Verteilung von Daten und Funktionen*; relevante Aspekte sind die Art der Verteilung, technische Schnittstellen (Hardware), Protokolle, Software**

Datenbankentwurf in einer Systementwicklung



- **Funktionenmodell und Datenmodell müssen in den einzelnen Phasen abgeglichen werden.**
- **Im Datenmodell müssen zukünftige Funktionen bereits berücksichtigt werden.**
- **Konzeptionelle Änderungen des Datenmodells sind später mit sehr hohem Aufwand verbunden.**

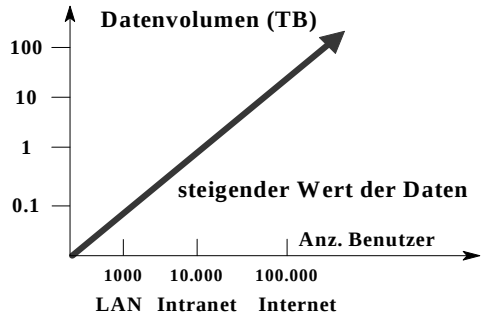


Phasenschema zum Datenbankentwurf

- **Informationsanalyse der geplanten Systemfunktionen**
  - **Analyse der Anforderungsdefinition, Feststellung benötigter Daten**
  - **Output: Beschreibung der logisch benötigten Datenbestände**
- **Konzeptioneller Entwurf der Datenbasis und der Anwendersysteme**
  - **ER-Modellierung, grobe Definition der Relationstypen**
  - **Aufbau und Fortschreibung eines Datenkatalogs**
  - **Output: Relationstypen, Datenkatalog**
- **Relationale Datenmodellierung (logischer Entwurf)**
  - **Integritätsregeln, ggf. Verteilungs- und Integrationskonzept für RT**
  - **Analyse von Abhängigkeiten, Normalisierung**
  - **Definition von Datensichten, Zugriffsrechten, Indexen**
  - **Output: endgültiges Relationenmodell**
- **Verteilungsentwurf (bei verteilter Datenhaltung)**
  - **Aufteilung der Datenbasis auf verschiedene Rechner mit ggf. heterogenen DBMS**
- **Implementation der Datenbank**
  - **Umsetzung Relationen und Sichten (Views); physisches DB-Design, Aufbau der DB**
  - **Output: operationale Datenbank mit Dokumentation**
- **Implementation der Anwendersysteme (Softwareentwicklung)**

Datenwachstum ohne Ende

- **Aufteilung der IT-Kosten: Hardware 10%, Software 30%, Datenbasis 60%**
- **Operationelle Daten für ein mittelgroßes Unternehmen umfassen > 100 GB.**
- **Bei einem *Data Warehouse* ist das Ziel, aus Daten und Metadaten durch *Data Mining* Software neue Erkenntnisse und Zusammenhänge zu gewinnen.**
- **Daten kombiniert mit intelligenter Software können dann als eigenständige Wissensbasis dienen (100-1000 TB).**
- **Jedes Jahr verdoppelt sich der Umfang der gespeicherten Daten.**
- **Auswertung der Daten (z.B. Bon-Daten) wird kritisch und komplexer.**
- **Insbesondere EC (B2C) impliziert hohes Datenwachstum.**
- **EMC² – Weltmarktführer in Speichersystemen sieht Speicher heute als wichtigste Komponente von IT-Systemen.**
- ***Moore's Gesetz*: Leistung von Prozessoren und Speicherdichte verdoppelt sich im ∅ etwa alle 18 Monate.**



Beispiel Amazon USA

- **Individueller Seitenaufbau**
  - **Kunde sieht Angebote von Dingen, die ähnlich sind zu den *zuvor gekauften***
  - **ähnlich zu denen *auf der Karte* sind**
  - ***die andere wie er* gekauft haben**
  - **die auf der *Wunschliste* stehen**
  - **ähnlich zu den in den letzten Minuten *angeschauten***
  - ***Kauf von Dingen, von denen der Kunde gar nicht wusste, dass er sie wollte bzw. brauchte!***
- **US Vertrieb mit 26 Datenbanken**
- **1500 Server (meist unter Linux)**
  - **Größter ~20TB, 5TB Rohdaten**
  - **Größter Transaktionsspeicher ~1TB + DB-Files**
  - **99.99% Verfügbarkeit**
- **Life-Cycle durchschnittlich drei Monate**
- **Kunden zunehmend kritisch bzgl. Antwort-Zeiten und Verfügbarkeit**
- **Steigendes Volumen an Kunden-Features**
- **Internet-Latenzen zwingen bewegliche Daten näher an den Kunden**
- **Erhöhte Verfügbarkeit fordert Einsatz von redundanten DBs an untersch. Lokalitäten**

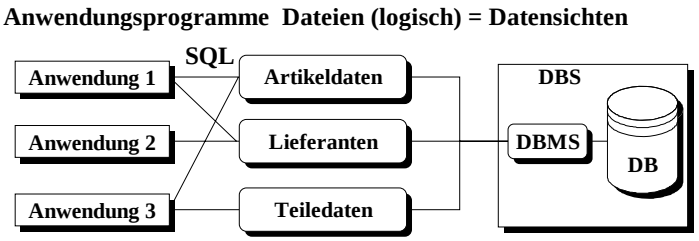
Datenintegrität

Hierunter versteht man die Korrektheit und Vollständigkeit gespeicherter Daten im Zeitablauf. Man kann drei grundlegende Bereiche unterscheiden:

- ***Datenkonsistenz*: logische Richtigkeit und Abbildungstreue der Daten; man unterscheidet:**
  - **physische Datenintegrität, d.h. korrekt gespeicherte Daten**
  - **semantische Datenintegrität, d.h. korrekte Umsetzung in ein Datenmodell**
  - **strukturelle Datenintegrität, d.h. die korrekte Repräsentation im Relationenmodell (Normalisierung, Abhängigkeitstreue usw.)**
  - **operationale Datenintegrität, d.h. die Einhaltung von anwendungsbezogenen Integritätsregeln im Rahmen der Transaktionsverarbeitung**
- ***Datensicherheit*: Bewahrung der Daten vor (partiellem) Verlust und unerlaubtem Zugriff (auch Modifikation) durch Maßnahmen technischer, organisatorischer und softwaretechnischer Art**
- ***Datenschutz*: Vermeidung mißbräuchlicher Benutzung von Daten (Datenschutzgesetze auf Bundes- und Länderebene)**
- **Datenintegrität in RDBMS wird in Kapitel 8.7 detaillierter behandelt**

2 Datenbankkonzepte - Grundlagen

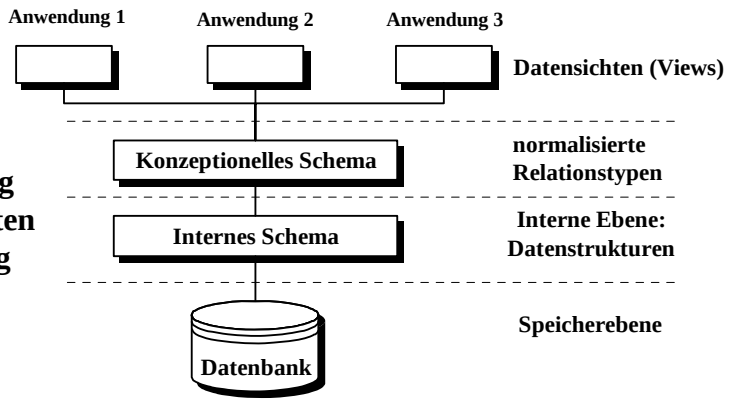
- **Datenbanksystem (DBS) besteht aus Datenbank DB und Datenbank-verwaltungssystem (DBMS Data Base Management System)**



- **Datenbank: weitgehend redundanzfrei gespeicherte Unternehmensdaten**
- **DB entsteht durch einen mehrstufigen Modellierungsprozess**
- **Datenspeicherung ist unabhängig von zugreifenden Anwenderprogrammen**
- **kommerzielle IV basiert heute i.d.R. auf relationalen DBS (RDBMS)**
- **Datenzugriff in RDBMS erfolgt mit Hilfe der Sprache SQL**
- **DBMS: Softwaresystem für folgende Kernfunktionen:**
  - **Zugriff (lesen, speichern, ändern, löschen) auf die gespeicherten Daten**
  - **offeriert lediglich Datensichten, d.h. logische Ausschnitte**
  - **trennt Programme von der internen Datenspeicherung**
  - **unterstützt Datenschutz, Datensicherung und Datenkonsistenz**

Abstraktionsebenen und Schemata

- **die Trennung von logischen und physischen Datenstrukturen sowie von Datensichten erfordert eine getrennte Beschreibung dieser Bereiche**
- **das American National Standard Institute (ANSI) sieht seit 1975 ein Konzept zur Datenbankarchitektur mit drei Abstraktionsebenen vor**
- **die Daten jeder Ebene werden durch ein Schema beschrieben: externes Subschema, konzeptionelles Schema und internes Schema**
- **Zweck dieser Trennung ist die Isolation von Änderungen auf den einzelnen Ebenen, ohne daß andere Ebenen, z.B. Nutzersichten, betroffen sind**
- **Die Idee ist gleichartig zur Idee des Abstrakten Datentyps: Kapselung von Daten und Zugriffsmethoden**



Merkmale von Datenbanksystemen

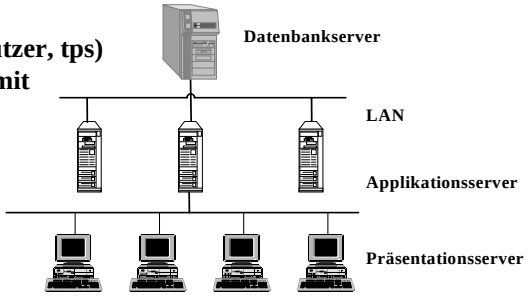
- ***Zentrale Datenbasis*: Daten eines größeren Anwendungsbereiches, häufig die des ganzen Unternehmens, werden gesamtheitlich gespeichert**
- ***spezifische Datensichten*: die einzelnen Anwendungsprogramme (Benutzer) "sehen" nur den für sie relevanten Datenausschnitt**
- ***kontrollierte Redundanz*: Daten werden nur dann mehrfach gespeichert, wenn dies aufgrund des schnellen Zugriffs unbedingt erforderlich ist**
- ***logische Datenunabhängigkeit*: individuelle, anwendungsorientierte Sichten sind weitgehend unabhängig von logischer Gesamtstruktur der Datenbank**
- ***physische Datenunabhängigkeit*: interne Datenorganisation und Speicherung sind für die Anwenderprogramme weitgehend unsichtbar; Datenzugriff erfolgt nicht direkt, sondern nur über das DBMS**
- ***Datenschutz*: durch kontrollierten Zugriff und Vergabe von Zugriffsrechten ist die mißbräuchliche Benutzung von Daten erschwert**
- ***Datensicherheit*: durch eine Reihe von Maßnahmen (Logging von Transaktionen, Backups, automatischer Wiederanlauf bei Systemausfällen) werden eine Reihe von Problemen, die bei individueller Datensicherung auftreten, vermieden (dafür gibt es andere Probleme !)**

Systemfunktionen von Datenbanksystemen

- ***Dateimanagement*: Speicherallokation, Zugriffe auf Festplatte**
- ***Puffermanagement*: Datenblocktransfer zwischen Platte und Puffern im HS**
- ***Abfrageinterpreter* interpretiert bzw. übersetzt SQL-Anweisungen in Programmaufrufe entsprechender "low level"-Routinen des RDBMS**
- ***Abfrageoptimierer*: sucht für eine gegebene Abfrage an Hand von Statistik- und Indexdateien einen "guten" Zugriffspfad auf die gewünschten Daten**
- ***Autorisierungs- und Integritätskontrolle*: überwacht den Zugriff auf die Daten, d.h. überprüft die Autorisierung und Datenintegrität**
- ***Datensicherung*: Maßnahmen um Folgen von Systemabstürzen oder anderen Fehlersituationen klein zu halten**
- ***Kontrolle von Mehrfachzugriffen*: synchronisiert Mehrfachzugriffe und sperrt temporär Datenzugriffe, die zu Konflikten führen würden**
- **Folgende Datenbestände müssen verwaltet werden:**
  - ***Datenbank* (Benutzerdaten)**
  - ***Metadaten* über: Tabellen, Attribute, Nutzer und Autorisierung**
  - ***Index-Dateien* zum schnellen Wiederauffinden von Daten**
  - **Statistikdaten zur Auswertung durch den Abfrageoptimierer**

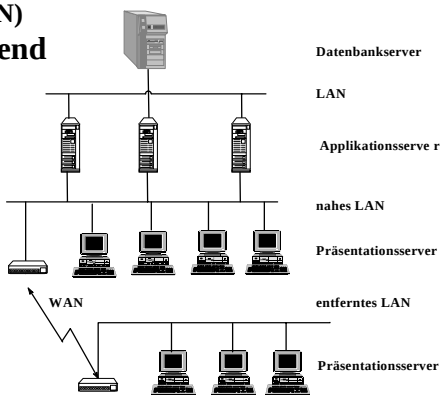
Grundlegende Architekturmerkmale von DBS

- **Herkömmliche DBS**
  - **eine Datenbank und *ein* DBMS auf den Festplatten *eines* Rechners (PC-Mainframe)**
  - **Datenzugriff über SQL-Anweisungen**
- **PC DBMS**
  - **Fokus auf Einzelplatzanwendungen, erlauben kein echtes OLTP**
  - **Stärken: integrierte, komfortable Entwicklungsumgebung**
- **Client-Server-Datenbank-Architekturen**
  - **beinhalten Clients, LAN, Protokolle und Datenbankserver mit DBMS**
  - **Datenbankserver ist heute überwiegend ein SQL-Server mit RDBMS**
  - **Clients senden SQL-Anweisungen u. erhalten die Ergebnismenge; daher können Server u. Clients unterschiedliche Systemplattformen aufweisen**
  - **Serverplattform: 32-64 Bit Prozessoren mit UNIX o. Windows Server BS (32-64 Bits)**
  - **höhere Leistungsanforderungen (Anz. Nutzer, tps) erfordern 3-stufige Verteilungskonzepte mit Applikationsservern ⇒ R/3**
- **Verteilte DBS**
  - **Semantisch zusammengehörige Datenbanken sind in einem Computer-Netzwerk verteilt**



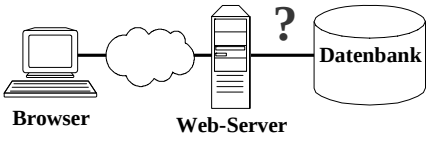
Dreistufige Client-Server-Architektur von R/3

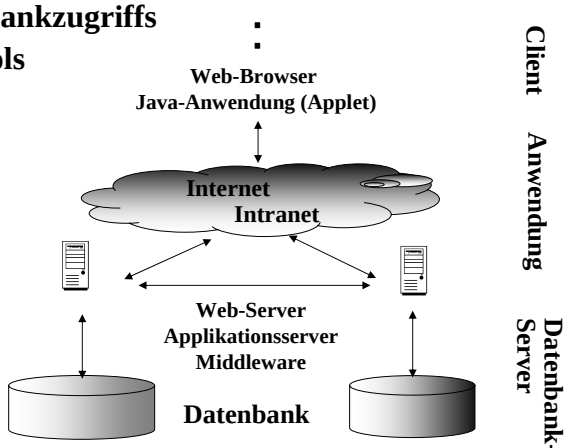
- **gedacht für größere Unternehmen, mit vielen aktiven Benutzern; auch zweistufige Architekturen sind möglich**
- **ein Datenbankserver mit einem RDBMS**
  - **verwaltet alle Anwendungsdaten, Metadaten (Data Dictionary), Software**
  - **RDBMS: Oracle, MS SQL-Server, SAP-DB, DB2**
- **mehrere Applikationsserver**
  - **führen alle R/3-Anwendungs- und Systemprogramme aus**
  - **verarbeiten und puffern Eingabedaten, führen SQL-Zugriffe zum Datenbankserver aus**
  - **hoher Datendurchsatz zwischen Datenbank- und Applkationsserver erfordert eine sehr schnelle Netzverbindung (i.d.R. ein schnelles LAN)**
- **viele Präsentationsserver mit PCs als Frontend**
  - **realisieren die grafische Benutzerschnittstelle**
  - **Dialogführung, Datenerfassung, -ausgabe**
  - **Anbindung erfordert nur geringe Datenübertragungsraten**
  - **Verbindungsmöglichkeiten zum Applikationsserver über:**
    - LAN
    - Standleitung
    - Internet (TCP/IP)
    - Modem



Datenbanken in Web-Applikationen

- **Art der Veröffentlichung: dynamische Web-Seiten (HTML) oder Web-Services (XML)**
- **Netzwerk-Umgebung (LAN oder WAN)**
- **Client- oder Server-Datenbank**
- **Implementiertes formales Datenmodell: RDBMS, XML, ORDBMS, …**
- **Technologievariante des Datenbankzugriffs**
- **Entwicklungstechniken und -tools**





Basistechnologien für den Web-Zugriff auf DB

- **Common Gateway Interface (CGI)**
  - **War die erste technologische Lösung (oftmals in Perl programmiert)**
  - **Web-Server reagiert auf Client-Anforderungen durch Aufruf eines CGI-Programms, das dann ein ausführbares Programm zum Datenzugriff startet (ODBC)**
  - **Resultate werden vom Web-Server zurück an den Web-Client gesendet**
  - **Bindet viele Systemressourcen (Prozesse) und ist nicht mehr konkurrenzfähig**
- **Application Programming Interface (API)**
  - **WWW-Server realisiert eine direkte Programmierschnittstelle zum DBMS**
  - **verbreitet im Freeware-Bereich, weniger geeignet für komplexe Geschäftsprozesse**
  - **Beispiele: ASP (Avtive Server Pages) und ASP.NET, PHP, Java-Servlets**
- **Application-Server**
  - **Realisiert die Logik einer Applikation auf einem eigenen Server**
  - **Web-Server baut Verbindung zum Applikations-Server auf**
  - **Geeignet für komplexe Anwendungen mit hohen Performance-Anforderungen**
- **DBMS mit Web-Server-Funktionalität**
  - **Datenbankserver erfüllen Web-Server-Aufgaben**
  - **Einbindung von logischen Komponenten in die DBMS-Funktionalität**
  - **Geeignet insbesondere für Multimedia- und technische Anwendungen auf ODBMS**
- **Web-Server**
  - **Spracherweiterungen ermöglichen einen direkten Zugriff auf Datenbanken**
  - **Web-Services mit XML (Extensible Markup Language)**

Datenkataloge (Data Dictionary)

- **Datenkataloge (Data Dictionary): integraler Bestandteil moderner DBMS**
- **verantwortlich ist die Instanz Datenbankadministration**
- **redundanzfreies Verzeichnis aller Datenelemente einer Datenbank: Abgleich ob Attribut mit gleicher Bedeutung aber anderem Namen vorhanden ist, bzw. ob gleicher Name für Attribut mit anderer Bedeutung vergeben wurde**
- **eindeutige Festlegung der Bedeutung eines Datenfeldes**
- **Datenbeschreibung: Name, Typ, Größe, Dimension, Grenzwerte, ggf. Verschlüsselung und externe Darstellung**
- **Muß- oder Kannfeld beim Anlegen**
- **Kennzeichnung: Schlüsselfeld, Schlüsseltyp, Attributgruppen**
- **Zuständigkeit für die Verwaltung und Vertraulichkeit der Attribute**
- **bei modernen DBMS ist Software zur Erstellung und Verwaltung eines Datenkataloges integrierter Systembestandteil**
- **neben Datenbeschreibungen können auch die auf den Daten definierten Funktionen (Methoden), die auf den Daten ausgeführt werden und vom Eintritt vorher definierter Ereignisse abhängig sind, spezifiziert werden**

Kosten beim Einsatz und Betrieb eines DBMS

- **Lizenzkosten (untergeordnete Rolle in der Praxis)**
  - **Preispolitik der Hersteller unterschiedlich, nur bei IBM plattformunabhängig (Kernel)**
- **Support- und Wartungskosten**
  - **einfache Installation ca. 10 %, aufwendige Installation ca. 15 % der Gesamtkosten**
  - **zwischen Plattformen kein einheitlicher Trend**
  - **bei Unix-Betriebssystemen i.d.R. höher als bei Windows NT Server**
- **Entwicklungskosten**
  - **nur bei größeren Unix-Installationen**
  - **Entwicklungsaufwand-Faktoren**
- **Schulungskosten**
  - **Ausbildungsangebote: 32 Tage für Oracle-Systeme, 6 Tage für SAP DB**
  - **für Administrationsaufgaben bis zu 6 Wochen**
- **Beratungskosten**
  - **gesonderte Preisvereinbarung, hohe Tagessätze für Spezialisten, z.B. Tuning**
  - **Erfahrungshintergrund und Know-how der Berater sind entscheidend**
- **Administrationskosten (größter Kostenfaktor - personalabhängig)**
  - **Gesamtkostenanteil bei kleinen Installationen bis zu 80%, typisch 50%**
  - **geeignete Werkzeuge und Beeinflußbarkeit des Systems sind kostenentscheidend**

Wichtige Aspekte bei der Auswahl von RDBMS

- **die strategische Bedeutung eines DBMS erfordert vom Anbieter langfristige Garantien für Support und Weiterentwicklung !**
- **Unterstützte SQL-Dialekte und SQL-Erweiterungen**
  - **ISO SQL-92, SQL-2, SQL-3, ORACLE, DB2 SQL, ODBC, JDBC, SQLJ, CGI, ISA**
  - **Updateable Join Views, Outer Joins**
  - **Stored Procedures und Triggers**
  - **Einzelsatzzugriff, Scrollable Cursors**
  - **Domains, Long Datatype (BLOBs)**
- **Precompiler und Call-Schnittstellen**
  - **Java, PHPx, C/C++, COBOL**
  - **Call-Schnittstelle (SQL/CLI)**
  - **ODBC / JDBC-Treiber**
- **Skalierbare Performance**
  - **Multi-Threaded / Multi-Server-Architektur, SMP-Unterstützung**
  - **Optimierung von Zugriffspfaden**
  - **Speicherung und Abruf optimierter Zugriffe bei embedded SQL**
  - **Asynchrones Logging, Group Commits**
  - **Zeitverhalten und erzielbare Transaktionsraten**
  - **Performance: Statistiken**

Wichtige Aspekte bei der Auswahl von RDBMS (2)

- **Verfügbare Systemplattformen (Client / Server)**
- **Verfügbarkeit und Ausfallsicherheit**
  - **Reorganisationsfreiheit (dynamische Plattenspeicherverwaltung**
  - **Online-Backup und -Recovery, dual Logging, Spiegelplattenbetrieb**
- **Integrität und Zugriffsschutz**
  - **Row-Level Locking, Isolation Levels**
  - **Domain-Integrität, deklarative referentielle Integrität**
  - **Mehrstufiges Benutzerkonzept**
- **physisches DB-Design: Flexibilität / Einfachheit der Anordnung von Tabellen, Freiplätzen, Tuningmöglichkeiten, Zugriffsoptimierung, Restriktionen**
- **Verteilte Datenbanken**
  - **Heterogene Plattformen**
  - **Verteilte Joins und Updates, referentielle Integritätsregeln**
  - **Replizierte Tabellen (zeitverzögerter Update), Two Phase Commit, Lokale Autonomie**
- **Datenbankutilities zum Export und Import von Daten, Datenformate**
- **Administrationskosten - wichtigster Kostenblock**
- **Kompatibilität: SAP R/3 zertifiziert ?**
- **Neuer Trend: MySQL, PostgreSQL, open Source, weltweite Entwickler-Community, sehr kompakt und effizient für EC-Anwendungen**

Datenbankadministration

- **umfaßt logischen und physischen Datenbankentwurf und Datenpflege**
- **Definition der Zugriffspfade, partitionieren und replizieren von Daten**
- ***Replikationsplanung*: welche Daten wo repliziert werden, ob Replikate gleich sein müssen (synchrone Replikation) oder zeitversetzter Update möglich ist**
- **Installieren neuer Versionen der Datenbanksoftware (u.a. DBMS)**
- **Anlegen und Verwalten von Speicherbereichen**
- **Anlegen und Löschen von Datenbanken**
- **Verwaltung des / der Datenbank-Schemas bzw. Schemata**
- **Nutzeradministration: Einrichten / Löschen von Nutzerkennungen, Vergabe von Zugriffsrechten (Tabelle, Attribut, Tupel (wertabhängig))**
- **Erstellen und Verwalten von Indizes**
- **Erstellen und Verwalten von Trigger-Definitionen, Stored Procedures**
- **Laden und Entladen von Daten**
- **Backup- und Restore-Aktivitäten**
- **Tuning und Monitoring des Datenbanksystems**
- **datenbankbasiertes Archivieren, d.h. langfristige Speicherung aller wichtigen Daten aus Sicherheitsgründen und durch den Gesetzgeber**

Datenbankbasiertes Archivieren

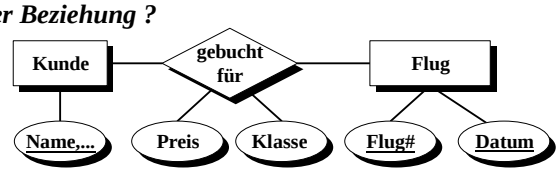
- **Verschieben von Daten aus der Datenbank auf Tertiärspeicher (Band, CD)**
- **ist aus systembedingten und gesetzlichen Gründe notwendig:**
  - **Wachstum der Bewegungsdaten verursacht Speicherplatz- und Laufzeitprobleme**
  - **eine verkleinerte Datenbank vereinfacht Backup und Restore-Verfahren**
  - **Stammdaten können leichter auf einem aktuellen Stand gehalten werden**
  - **aus gesetzlichen Gründen müssen Daten „aufbewahrt“ werden**
  - **Wiederverwendung von Daten, z. B. Konstruktionsdaten für Neuentwicklungen**
- **Anforderungen an archivierte Daten (in Dateien)**
  - **die Daten müssen für einen langen Zeitraum lesbar und interpretierbar sein**
  - **technische Anforderung: mit neuer Hardware müssen die alten Archivdaten lesbar sein; diverse Problemfelder (u.a. Integer-Daten, Dezimaldaten, EBCDIC, ASCII-Zeichen)**
  - **Änderungen der Software und der Relationstypen führen häufig dazu, daß archivierte Daten eine andere Bedeutung bekommen als die, die sie hatten; dies erfordert hohe Anforderungen an Anwendungen, um solche Änderungen zu erkennen und abzulehnen**
- **Zielkonflikte beim Archivieren**
  - **IRS verlangt, daß Archive mit *eigener IT* gelesen und verarbeitet werden können**
  - **Problemfaktoren sind unterschiedliche Hardware von Anwendern und Finanzbehörden**
  - **dagegen verlangen Wirtschaftsprüfer, daß Archive durch Verschlüsselung vor Manipulationen geschützt werden müssen**
  - **da viele Softwareanbieter ihre Produkte weltweit vermarkten, besteht hier das Problem, daß Anforderungen der Finanzbehörden einzelner Länder sich ausschließen**

3 Konzeptionelle Datenmodellierung

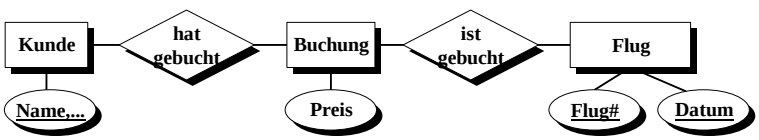
- **Modellierung bedeutet**
  - **Vereinfachte Abbildung eines Realitätsausschnittes durch Abstraktion (Verallgemeinerung)**
  - **Resultiert in einer Vereinfachung und dient der Bewältigung der Komplexität**
  - **Erfordert Erfahrung und ist durch komplexe Einflussfaktoren nicht automatisierbar**
- ***Datenmodell*: (halb-) formale Beschreibung der Datenbasis**
- **Man unterscheidet zwei Arten von Datenmodellen:**
  - ***konzeptionelle Datenmodelle* werden nicht direkt von Datenbanksystemen unterstützt; zwei wichtige Vertreter sind das *(Extended) Entity Relationship Modell* und *UML* (Unified Modeling Language), eine objektorientierte Designsprache zur Systementwicklung**
  - ***Logische Datenmodelle* werden direkt von Datenbanksystemen unterstützt; typischer Vertreter ist das *relationale Datenmodell***
  - **Logische Datenmodelle für DBS werden im Kapitel 4 betrachtet**
- ***Entity Relationship Modell (ER-M)***
  - **Grundelemente sind Objekt- und Beziehungstypen mit ihren Attributen**
  - **Ein ER-M basiert auf einer grafischen Notation und dient primär zum Design und Validierung eines Datenmodells vor der Implementierung**
  - **Ist gut geeignet ein Gesamtdatenmodell für ein Unternehmen darzustellen**
  - **Im ER-M fehlen Elemente für *Datenmanipulation und Datenintegritätsrestriktionen***
  - **muss in ein logisches Datenmodell z.B. für ein RDBS transformiert werden**
  - **Alternativ kann eine Anwendung auch direkt in einem logischen Datenmodell abgebildet werden – z.B. in Form von Relationstypen, was manchmal einfacher ist**

Entity Relationship Modellierung

- **basiert auf Objekttypen und ihren Beziehungen zueinander**
  - **Objekttypen besitzen Attribute**
  - **Attribute besitzen keine eigenen Attribute**
  - **Zusammengesetzte Attribute sind eine Kombination mehrerer Attribute**
  - **Ein mehrwertiges Attribut kann gleichzeitig mehrere Werte annehmen**
- **bei der Modellierung bereitet manchmal die Unterscheidung zwischen Objekttypen, Beziehungstypen und Attributen Schwierigkeiten:**
  - ***Objekt oder Attribut ?* bzw. *Objekt oder Beziehung ?***
- **Beispiel**
  - **Ein Kunde bucht einen Flug**
  - **Objekttypen: Kunde, Flug**
  - **Beziehungstyp: Buchung mit eigenen Attributen Preis und Klasse**



  - **Möglich ist auch die Modellierung in der Buchung als Objektklasse definiert wird**



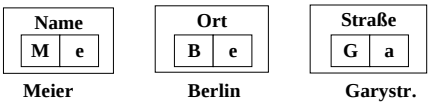
  - ***In den folgenden Beispielen werten die Attribute nicht mehr aufgeführt***

Definitionen und Begriffe der ER-Modellierung

- ***Objekte*: reale oder abstrakte Dinge, Personen, Begriffe, gedankliche Abstraktionen, die in dem betrachteten System relevant sind**
- ***Attribute*: definieren *Eigenschaften* von Objekten**
- ***Wertebereich eines Attributes (Domain):* definiert die möglichen Ausprägungen des Attributes: Zeichenketten, Datum, boolesche oder numerische Daten**
- ***Attributwert*: Ausprägung eines Attributes**
- ***Nullwerte*: repräsentieren unbekannte oder ggf. nicht existente Attributwerte**
- ***Objekttyp*: beschreibt gleichartige Objekte, d.h. mit gleichen Attributen**
- ***Objektklasse*: Menge aller Objekte eines Objekttyps; Objektklassen können disjunkt, sich überlappen oder Teilmengen voneinander sein**
- ***Beziehungen zwischen Objekten*: logische Verknüpfungen zwischen Objekten von Objektklassen; Beziehungen können eigene Attribute besitzen, die erst durch die Verknüpfung der Objektklassen relevant werden**
- ***Beziehungstyp*: beschreibt gleichartige Beziehungen zwischen Objekttypen mit ggf. gleichen zusätzlichen Attributen für den Beziehungstyp**
- ***Beziehungsklasse*: Menge aller Beziehungen eines Beziehungstyps**
- ***Konzeptionelles Schema*: Definitionen der Objekt- und Beziehungstypen**

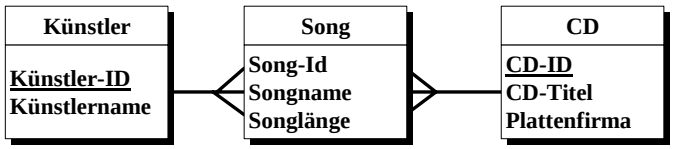
Schlüssel

- **Attribut o . Attributgruppe zur Identifikation, Klassifikation oder Verknüpfung von Objekten**
- **Kandidatenschlüssel identifiziert Objekte einer Objektklasse zeitinvariant eindeutig und ist *minimal***
- **Primärschlüssel ist ein ausgewählter Kandidatenschlüssel; Kennzeichnung durch Unterstreichen in einer Attributgruppe**
- **zusammengesetzter Schlüssel ist eine Attributgruppe, z.B. Preisliste (<u>Teile-Nr</u>, <u>Lieferanten-Nr</u>, Preis)**
- **Klassifikationsschlüssel (sprechender Schlüssel)**
- **Verbundschlüssel besteht aus klassifizierendem und Zählteil**
- **Fremdschlüssel ist Primärschlüssel in einem anderen Objekttyp**
- **Suchschlüssel ist ein Attribut oder Attributgruppe über deren Werte nach Objekten einer Objektklasse gesucht werden kann**
- **Sekundärschlüssel: nicht identifizierende(s) Attribut(gruppe); wird i.d.R. als Suchschlüssel (über einen Index) benutzt**
- **Matchcode: mehrere Suchbegriffe, die ein Attribut unvollständig beschreiben:**



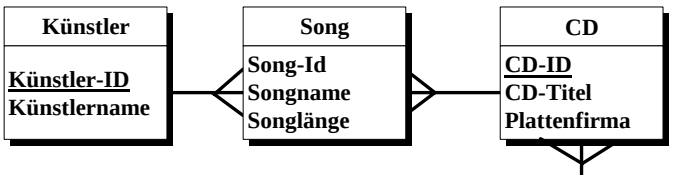
Beispiel 1: CD-Datenmodell (1)

- **Grafische Notation (*Kardinalitäten nach Chen Notation*) als ER-Diagramm**
- ***Kardinalitäten werden später in Schlageter-Stucky-Notation definiert***
- **es sollen CDs, deren Titel, die Plattenfirma und die Songs auf der CD modelliert werden**
  - **Eine CD wird als Objekttyp modelliert, mit den Attributen CD-ID (Primärschlüssel), CD-Titel und Plattenfirma**
  - **Ein Song wird ebenfalls als Objekttyp modelliert**
  - **Zwischen CD und Song besteht eine 1:m Beziehung mit der Bedeutung *CD enthält* (links nach rechts) bzw. *Song ist enthalten in***
  - **Die Krähenfüsse symbolisieren, dass eine CD i.d.R. mehrere Songs enthalten kann; jedoch ist ein Song nur auf genau einer CD enthalten (Annahme)**
  - **Ein Song kann jedoch auf mehreren CDs enthalten sein**
  - **Künstler sollen ebenfalls einbezogen werden**
  - **Jeder Song wird von genau einem Künstler gespielt (Annahme)**
  - **Ein Künstler kann mehrere Songs spielen**

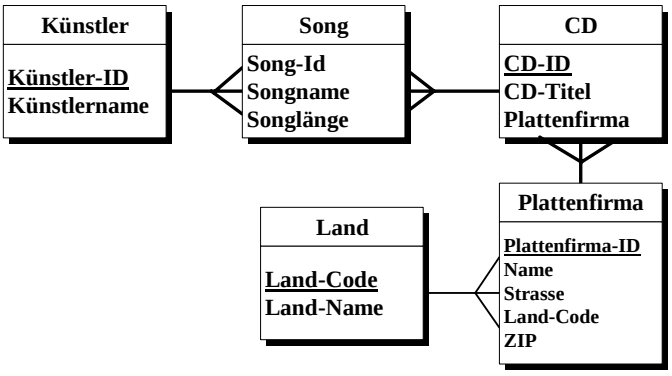


  - **Problem: eine Plattenfirma wird i.d.R. mehrere CDs produzieren**
  - **Plattenfirma muss also als ein Objekt modelliert werden**

Beispiel 1: CD-Datenmodell (2)

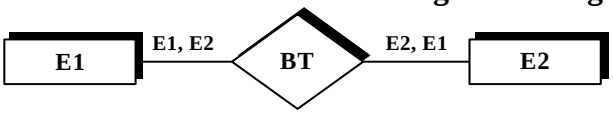


- **Problem: es gibt Datenredundanzen**
- **Zur Vermeidung dieser Redundanzen muss ein Bundesland als Objekt modelliert werden**

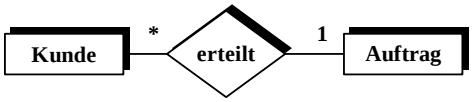


Kardinalitäten von Beziehungstypen

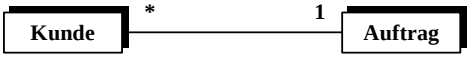
- ***die ab hier vorgestellte Notation geht auf Schlageter-Stucky zurück***
- **jedem Beziehungstyp BT sind für alle beteiligten Objekttypen E1, E2, ... Kardinalitäten zugeordnet, die Integritätsbedingungen darstellen**
- **card (BT, E) = [m, n] mit m, n ≥ 0, ganzzahlig und m ≤ n bedeutet, mit wieviel Objekten *ein* Objekttyp E minimal und maximal in Beziehung stehen kann; vereinfachte Schreibweise: m, n**
- **zur weiteren Vereinfachung häufiger Fälle wird folgende Notation benutzt:**
  - **k: genau k-mal, k > 0, ganzzahlig, *: ≥ 0, d.h. beliebig viele, +: ≥ 1, c: 0,1**
- **Beziehungstypen BT mit Kardinalitäten werden grafisch folgendermaßen repräsentiert:**



- **Beispiel: jeder Auftrag ist genau einem Kunden zugeordnet, ein Kunde muß jedoch keinen Auftrag erteilt haben:**



- **weisen BT keine eigenen Attribute auf, dann kann die Raute weggelassen werden**

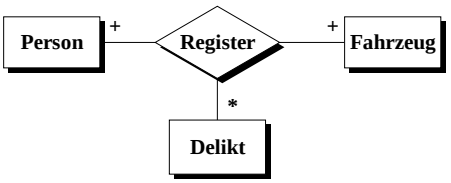


Beispiele für Kardinalitäten

Art	Beziehung	Beispiel
1-1	Projektleitung	Projekte — 1 1 — Projektleiter
n,m-*	Reservierung	Flug — 0,#Sitze * — Passagier
*-1	Familienzugehörigkeit	Ehepaare — * 1 — Kinder
c-c	Heirat	Frauen — c c — Männer
+-c	Religionszugehörigkeit	Religionen — + c — Personen
*-+	Staatsangehörigkeit	Personen — * + — Staaten
-	Teileverwendung	Teile — * * — Produkte
n,m-+	Lagerorte	Teile — 1,3 + — Lagerorte

Beispiel 2: Verkehrszentralregister (VZR)

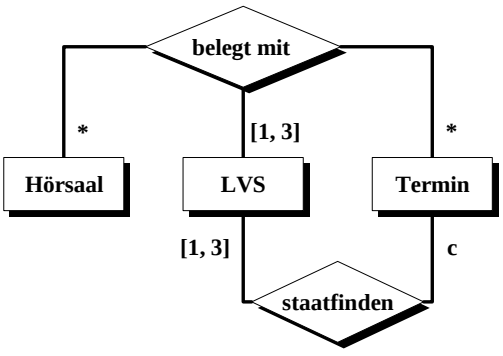
- **Im VZR werden u.a. Entscheidungen registriert von**
  - **Fahrerlaubnisbehörden (Entzug oder Neuerteilung von Fahrerlaubnissen)**
  - **Bußgeldbehörden, die eine Verkehrsordnungswidrigkeit mit einer Geldbuße von mindestens 80,00 DM oder einem Fahrverbot [in Monaten] ahnden**
  - **Es werden nur rechtskräftige bzw. bestandskräftige Entscheidungen registriert**
  - **eingetragene Entscheidungen werden mit Punkten gewichtet**
  - **Es sollen nur Delikte berücksichtigt werden, die mit *einem* Kfz verursacht wurden**
  - **Mit einem Delikt soll auch Fahrer und benutztes Kfz gespeichert werden**
  - **Das Datenmodell in Form eines ERD sieht folgendermaßen aus:**



  - **In diesem Fall existiert *ein Beziehungstyp* zwischen *drei Objektklassen***
  - **Die Kardinalitäten müssen bei n > 2 Objektklassen anders interpretiert werden; dies ist inhaltlich kompatibel mit der oben gegebenen Definition**
  - **Im obigen ERD gibt die Kardinalität an einer Objektklasse an, wie häufig *ein* Objekt aus der O-Klasse *in der Beziehung* mindestens und maximal vorkommen kann**

Beispiel 3: Zeit- und Raumplan für Lehrveranstaltungen

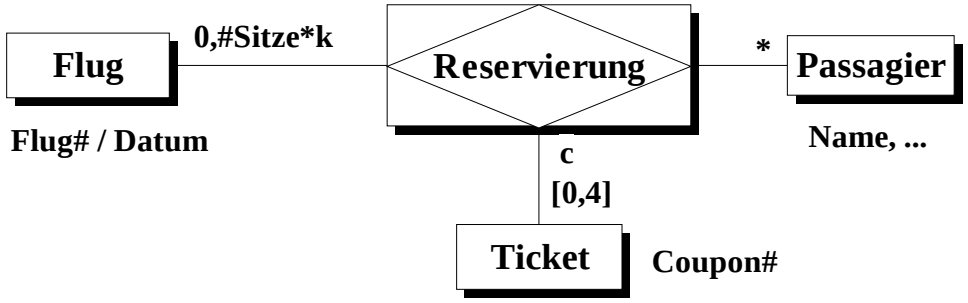
- **Annahmen über die Zeit- und Raumplanung von Lehrveranstaltungen in einer Vorlesungswoche:**
  - **Eine LVS besteht aus Vorlesung und (optional) aus ein bis zwei Übungen**
  - **Alle Veranstaltungen einer LVS werden als der LVS zugehörig betrachtet**
  - **Eine LVS wird in bestimmten Räumen zu bestimmten Terminen abgehalten**
  - **alle LVS finden *zeitlich überschneidungsfrei* statt**
  - **Jede betrachtete LVS muss zeitlich und räumlich zugeordnet werden**
  - **Das ERD sieht folgendermaßen aus:**



  - **Einige Integritätsregeln können nicht im ERD abgebildet werden!**

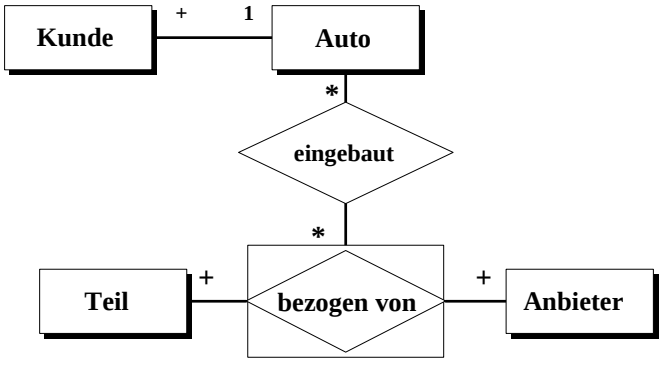
Beispiel 4: Flugreservierung (Beziehungsobjekttypen)

- repräsentieren Beziehungstypen, die sich in einem anderen Beziehungstyp wie ein Objekttyp verhalten
- Beispiel: Flugreservierungen und Ticket(heft)
  - ein Passagier kann beliebig viele Flüge reservieren
  - Anzahl Reservierungen pro Flug werden durch die Anzahl Sitzplätze begrenzt multipliziert mit einem Überbuchungsfaktor k
  - *Erst nach* einer Reservierung *kann ein* Ticket(heft) ausgestellt werden
  - jedes Ticket(heft) enthält maximal vier Coupons für die einzelnen Flüge (Flugstrecken)
  - für jeden Coupon kann eine Reservierung existieren



Beispiel 5: Autoreparatur (Beziehungsobjekttypen)

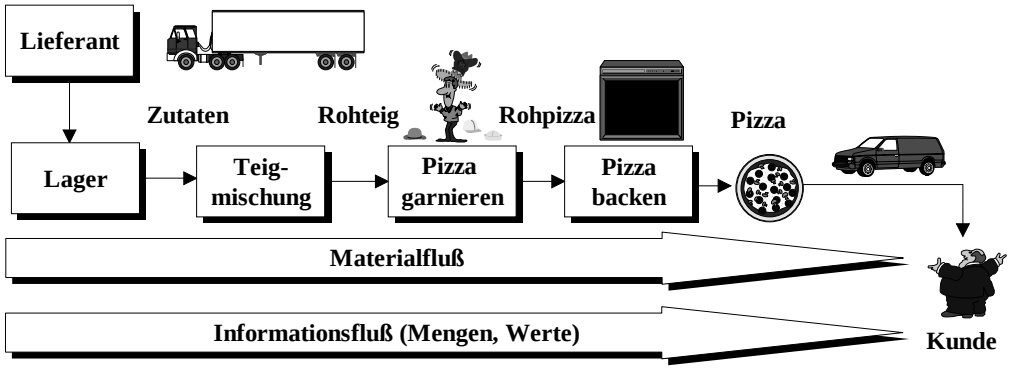
- Ein Kunde bringt (s)ein Auto zur Reparatur in eine Autowerkstatt
- Es können Teile ausgewechselt bzw. zusätzlich eingebaut werden
- Es sind aber manchmal nur Einstellungsarbeiten vorzunehmen
- einige Teile kann die Werkstatt von unterschiedlichen Anbietern mit unterschiedlichen Preisen beziehen
- Bei der Datenmodellierung sollen die eingebauten Teile mit deren Verkaufspreisen (die über den Bezugspreisen liegen) und der Anbieter festgehalten werden



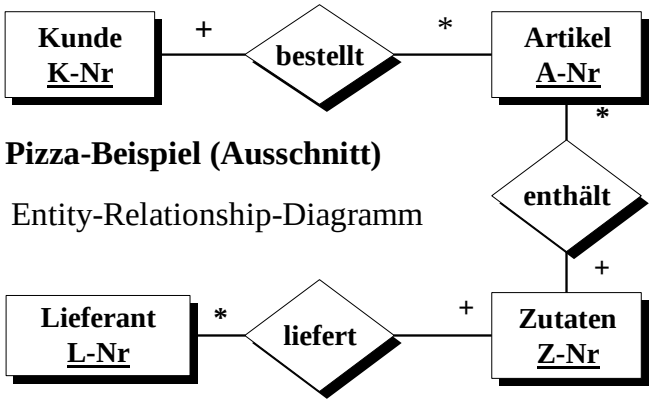
Beispiel 6: Pizza-Bäckerei

- **Verkauf überwiegend an Stammkunden (telef. Bestellung) durch Lieferung, jedoch auch an Laufkundschaft, mit angeschlossenem Restaurant**
- **Produktion und Beschaffung: Einkauf der Zutaten bei Lieferanten, Lagerung im Materiallager, Pizzateig herstellen (Mischer), Rohteig herstellen, Pizzas aus Rohteig formen, Pizzas garnieren, Pizzas backen**

Beschaffung / Produktion



Beispiel 6: ERD für Pizza-Bäckerei



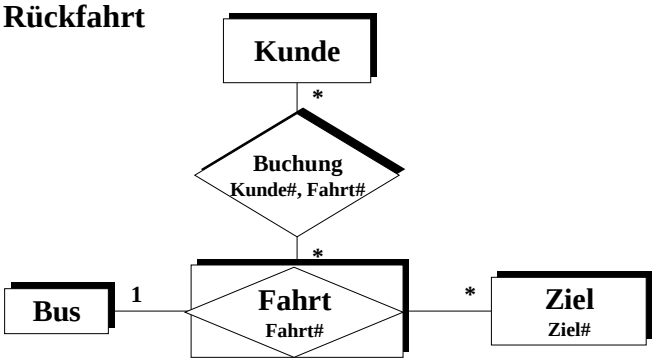
Pizza-Beispiel (Ausschnitt)

Entity-Relationship-Diagramm

- **Im obigen ERD ist der Auftrag noch nicht gut modelliert**
- **Die Daten eines Auftrags bestehen aus dem „Auftragskopf“ und den einzelnen Auftragspositionen**

Beispiel 7: Busreiseunternehmen

- Tiefflieger AG führt regelmäßige, eintägige Städtefahrten von Berlin durch
- die Busse werden durch die Bus# und diverse andere Attribute beschrieben
- bei Buchung wird Name, Anschrift, Tel.-Nr. der buchenden Person notiert, die Anzahl der benötigten Plätze welcher Fahrt, den Gesamtpreis, Buchungsdatum, Zahlungsart und Zahlungsdatum
- die Preise sind Änderungen unterworfen; ein Kunde, der eine Fahrt bucht, hat den zu diesem Zeitpunkt geltenden Preis zu bezahlen
- Objektklassen: Kunde, Fahrt, Bus, Ziel; Beziehungstyp: Buchung
- Es gibt alternative Möglichkeiten für die Modellierung
- Preise gelten für Hin- und Rückfahrt



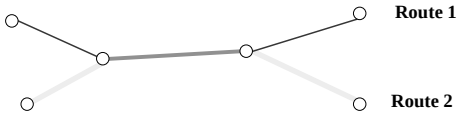
Beispiel 8: Linienfluggesellschaft

- eine LFG hat einen eindeutigen Airline-Code (z.B. LH) und Bezeichnung
- ein Flug definiert die zeitbezogene Verbindung zwischen zwei Flughäfen und wird durch eine Flug-Nr und Abflugdatum bestimmt (⇒ Abflugzeit)
- die Flug-Nr besteht aus einem Verbundschlüssel Airline-Code, Zahl
- eine Flugroute besteht aus ein oder mehreren Flügen und wird durch deren Flug-Nummern (⇒ Abflugzeiten) definiert
- eine LFG bedient eine genau definierte Anzahl von Flugrouten
- ein Flugzeug hat eine Registrier-Nr, Typ, Beschreibung, #Sitzplätze
- ein Flugzeug führt pro Tag i.d.R. mehrere Flüge durch, die durch Abflugzeit, Flug-Nr and Abflughafen definiert werden (Rotationen)
- wird ein Flugticket für einen Kunden ausgestellt, so besteht es aus ein oder mehreren Coupons und enthält u.a.. Name, Austellungsdatum, Preis, Status
- ein Sitzplatz beschreibt für ein Flugzeug u.a. die Sitzplatz-Nr, Smoking / Nonsmoking, Window /Aisle
- Reservierungen eines Kunden (Passagier) beziehen sich auf bestimmte Flüge; eine Reservierung kann ohne Ticket vorgenommen werden;
- vor Flugantritt muß einem Passagier ein Sitzplatz zugeteilt werden und eine Bordkarte ausgestellt werden

Beispiel 8: Flugrouten

- **Flugrouten bestehen aus einzelnen Streckenabschnitten (Flügen) zwischen jeweils zwei Flughäfen**

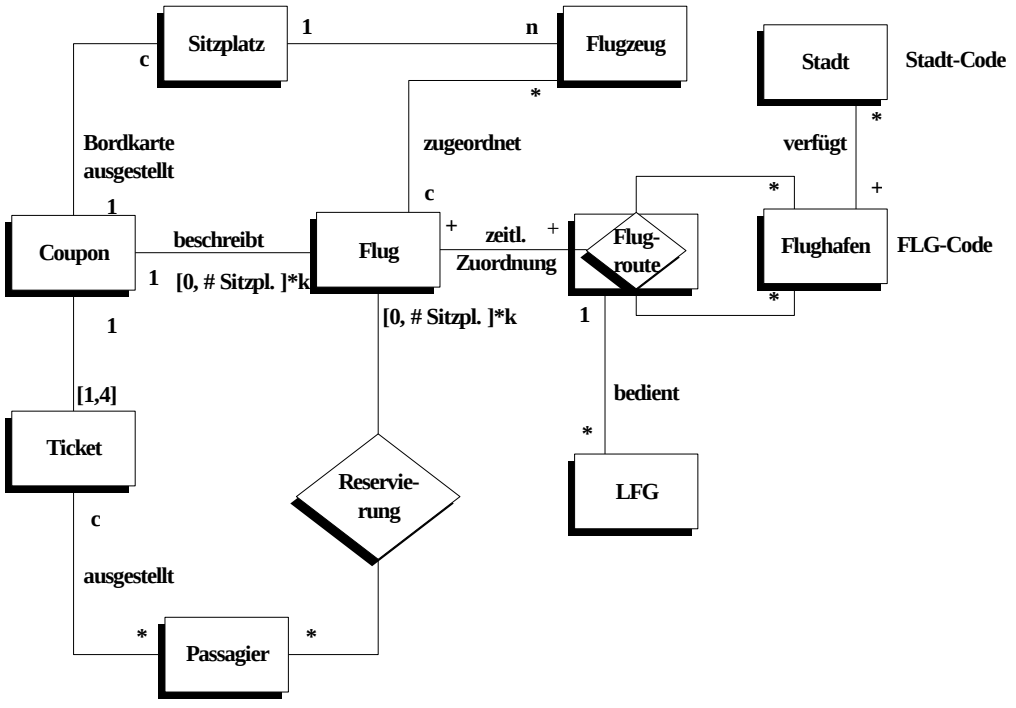
  Streckenabschnitte unterschiedlicher Flugrouten können identisch sein



- **Route (R#, von, nach, Abflugzeit, ...)**
- **Flug (Flug-Nr, Datum, von, nach, Abflug-Zeit, ...)**
- **Flug-Nr: besteht aus Airline-Code und einer für die Airline eindeutigen Nr.**
- **für die einzelnen Flüge werden unterschiedliche Flug-Nr unterstellt**
- **ein Flughafen wird durch einen drei Zeichen umfassenden Code identifiziert**
- **eine Stadt kann mehrere Flughäfen haben und ein Flughafen kann zu mehreren Städten zugeordnet sein.**

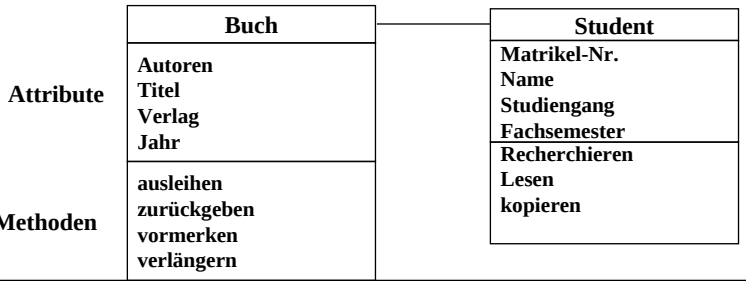
- **Orte (Ort-Nr, ...), Strecke (S-Nr, von, nach, ...), Routenabfolge (Pos-Nr, S-Nr, R-Nr), Routen (R-Nr, ...)**

Beispiel 8: ERD für Linienfluggesellschaft



UML Objekte und Beziehungen

- **Ein *Objekt* in UML ist Elementbaustein einer Anwendung; es gilt:**
  - **Ein Objekt hat einen *Zustand*, ein *Verhalten* und eine *Identität***
  - **Der Zustand wird durch Attribute beschrieben; jedes Attribut hat einen *Attributwert***
  - **es hat klar definierte Eigenschaften und ein Verhalten, das von seinem Zustand abhängt**
  - **von außen sichtbaren Funktionen (Methoden), die das extern sichtbare Objektverhalten bestimmen; die Menge der Methoden bildet die *Schnittstelle* eines Objekts**
  - **von außen *nicht sichtbare* Zustände und Funktionen**
  - **Ein Objekt hat eine unveränderliche *Objektidentität* (OID)**
- **Eine *Klasse* ist eine Zusammenfassung gleichartiger Objekte; jedes Objekt gehört zu (ist Instanz von) genau einer Klasse**
- **Assoziationen sind Beziehungen analog dem ER-M**
- **Beispiel UML Klassendiagramme, Assoziation der Klassen Buch-Student**

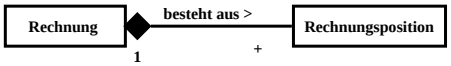


Assoziationen in UML

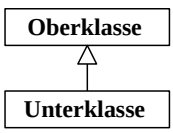
- **Analog der Beziehung zwischen Objektklassen in ER-M**
- **Beziehung zwischen Objekten einer oder mehrerer Klassen**
- **Weitere Konzepte in UML:**
  - **Aggregation (hat-Beziehung): Objekte sind Bestandteile eines anderen Objektes**

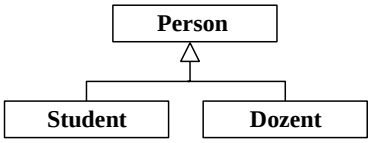
Normale Aggregation

Komposition: Klasse ist existentiell abhängig



  - **Generalisierung (ist-Beziehung) entsteht meistens durch Vererbung**

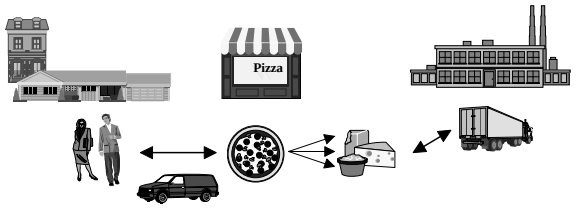




- **Kardinalitäten in UML sind analog zur Chen-Notation, d.h. die Kardinalitäten werden an den Kanten vertauscht angebracht (Sichtweise)**
- **Man beachte die Unterschiede zur Schlageter-Stucky-Notation**

Umwandlung eines ER-Modells in Relationstypen

- **jeder Objekt- und Beziehungstyp wird in einem RT abgebildet**
- **manchmal ergeben sich dann unnötig viele RT**
- **nur m:n-Beziehungen müssen als eigener RT abgebildet werden**



- **Relationstypen für Beispiel 6**
  - **Kunde (K-Nr, Name, Anschrift, Tel.-Nr., ...)**
  - **Artikel (A-Nr, A-Bezeichnung, Verkaufspreis, ...)**
  - **Zutat (Z-Nr, Z-Bez, Bestand, V-Preis, s, S, Bestell-Ind,...)**
  - **Rezeptur (A-Nr, Z-Nr, Menge,...)**
  - **Lieferant (L-Nr, Name, Anschrift, Tel-Nr, Fax-Nr, ...)**
  - **Auftragskopf (Auf-Nr, K-Nr, Datum, ...)**
  - **Auftragspos (Auf-Nr, Pos-Nr, A-Nr, Menge, ...)**
  - **Tour (Tour-Nr, Auf-Nr, L-Datum, L-Uhrzeit, ...)**

Umwandlung eines ER-Modells in Relationstypen

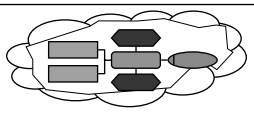
- **Relationstypen für Beispiel 2**
  - **Person (P#, Name, Vorname, PLZ, Ort, Geburtsdatum, …)**
  - **Fahrzeug (F#, Hersteller, Typ, ...)**
  - **Delikt (D#, Grund, Punkte, Geldbuße, Fahrverbot, ...)**
  - **Register (R#, P#, F#, D#, Datum, …)**
- **Relationstypen für Beispiel 5**
  - **Auto (PKW#, ...)**
  - **Kunde (K#, Name, ...)**
  - **Teil (T#, Bezeichnung, ...)**
  - **Bezug (T#, Anbieter, Preis, ...)**
  - **Eingebaut (PKW#, T#, Anbieter, Anzahl, ...)**
- **Relationstypen für Beispiel 7**
  - **Kunde (Kunde#, Name, Anschrift, Telefon#)**
  - **Fahrt (Fahrt#, Ziel#, Datum, Bus#, Personal#, ...)**
  - **Bus (Bus#, Kennzeichen, Anzahl_Sitze, ...)**
  - **Ziel (Ziel#, Ort, Entfernung, Einzelpreis, ...)**
  - **Fahrer (Personal#, Name, ...)**
  - **Buchung (Kunde#, Fahrt#, Personen, Betrag, Gesamtpreis, ...)**

Einige Grundregeln der Datenmodellierung

1. **alle Attribute eines Objekttyps sollten vom gesamten K-Schlüssel abhängig sein; ggf. sind Attribute in einem neuen Objekttyp aufzunehmen**
2. **ein Fremdschlüssel steht stets im Objekttyp mit der Kardinalität c oder 1; in einer 1:1 Beziehung steht er in einem der beiden Objekttypen:**
   - **Vertreter (V-Nr, Name, ..., Provision, Fixum)**
   - **Kunde (K-Nr, Name,..., *V-nr*), V-Nr ist Fremdschlüssel für Vertreter**
   - **Person (P-Nr, Name, Sozialvers-Nr), Sozialvers-Nr ist Fremdschlüssel für BfA-RT**
3. **eine Attributgruppe mit eigenständiger Struktur in einem RT, wird über Zerlegung bzw. einen neuen RT entfernt:**
   - **Auftrag (Auf-Nr, K-Nr, Besteller, Datum, .. , Pos-Nr, A-Nr, Menge)**
   - **Auftragskopf (Auf-Nr, K-Nr, Besteller, Datum, Rabattsatz, ..)**
   - **Auftragsposition (Auf-Nr, Pos-Nr, A-Nr, Menge, Liefertermin)**
4. **überlappende Objektklassen mit gleichen Attributen werden in einer generalisierten Objektklasse zusammengefasst; es verbleiben die spezialisierten Objektklassen:**
   - **z.B. Geschäftspartner (generalisiert)**
   - **Kunden (spezialisiert)**
   - **Lieferanten (spezialisiert)**

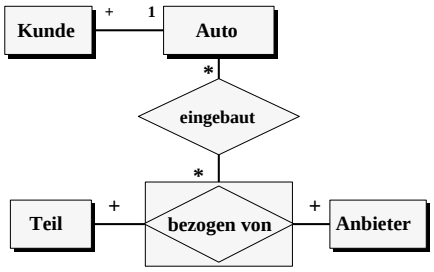
Phasenschema zur Datenmodellierung

1. Abstecken des Problemrahmens



2. Festlegen der Objekttypen

3. Festlegen der Beziehungstypen



4. Festlegen der Attribute, Schlüssel

5. Überführen in Relationstypen

Auto (PKW#, ...)

Kunde (K#, Name, ...)

Teil (T#, Bezeichnung, ...)

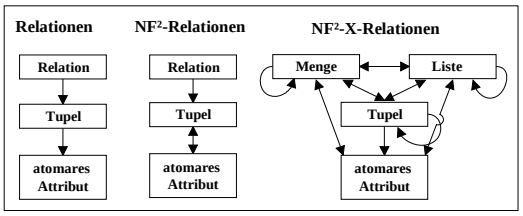
Bezug (T#, Anbieter, Preis, ...)

Eingebaut (PKW#, T#, Anbieter, Anzahl, ...)

6. Normalisierung, Integritätsregeln

4 Logische Datenmodelle in Datenbanksystemen

- **Kriterien zur Beurteilung von Datenmodellen:**
  - **Unterstützung durch DBMS-Anbieter (Industrietrends, de facto Standards)**
  - **einfache Modellierungsmöglichkeiten für möglichst umfassende Anwendungen**
  - **Eignung für bestimmte Anwendungsklassen**
  - **Effizienz bei Datenmanipulation (Abfragen), Transaktionsverarbeitung**
- **Kommerzielle DBMS unterstützen folgende Datenmodelle:**
  1. **hierarchische (netzwerkartige) Datenmodelle: haben nur noch historische Bedeutung**
  2. **relationales Datenmodell (*partiell* NF^2-Modell, bzw. erweiterte NF^2-Modelle)**
  3. **Objektorientierte Datenmodelle (OODBMS)**
  3. **objekt-relationale Datenmodelle (ORDMS) mit SQL-3 (Oracle 8, DB2)**
- **der Industriestandard ist immer noch das relationale Datenmodell**
- **Trend geht in Richtung ORDBMS**
  - **noch sind nicht alle Konzepte und Tools implementiert**
  - **Prognose: in 5 Jahren werden klassische RDBMS nur für Altanwendungen eingesetzt**
  - **Bald gibt es wohl nur noch wenige Anbieter: Oracle, MS (SQL-Server), IBM (DB2), MySQL**



Objektorientiertes Datenmodell

- **das objektorientierte Datenmodell wird nur durch wenige DBMS unterstützt**
- **Merkmale objektorientierter Datenmodellierung**
  - **Objekt-Identität: Objekte behalten ihre Identität, solange sie existieren, unabhängig von ihren Werten; ihre Identitätskennung wird niemals für ein anderes Objekt verwendet**
  - **komplexe Objekte können aus anderen Objekten definiert werden**
  - **Kapselung: der interne Objektszustand ist nur durch die für seine Klasse definierten Methoden veränderbar; die Implementationsdetails sind nicht nach außen sichtbar**
  - **Klassen: Objekte werden durch gemeinsame Eigenschaften klassifiziert**
  - **Vererbung: Klassen können als gerichteter, azyklischer Graph angeordnet werden; Teilklassen erben die Eigenschaften und Methoden ihrer Oberklassen**
  - **Erweiterbarkeit: der Benutzer kann in der Datenbank noch nicht enthaltene Typen und Methoden nach seinen Vorstellungen definieren**
- **Vorteile OODBMS gegenüber RDBMS:**
  - **bessere Modellierbarkeit durch größere Typenvielfalt**
  - **Wiederverwendbarkeit von Klassen für neue Applikationen**
  - **Zugriffsvorteile bei komplexen Abfragen, da Methodenzugriff einer Klasse möglich ist**
- **schwacher, inkonsistenter Industriestandard, kaum Anbieter**
- **Kommerzielle OODBMS: Poet (http://www.x-solutions.poet.com/de), ObjectStore (http://www.objectstore.net)**

Objektrelationale Datenbanken (ORDBMS)

- **Basieren auf der Erweiterung RDBMS mit objektorientierten Konzepten**
- **Ausgangspunkt waren die Schwächen des Relationenmodells**
  - **Unzureichende Vielfalt an Datentypen, fehlende Funktionen**
  - **Keine Unterstützung von Objekt-Hierarchien (Vererbung)**
- **Objektorientierte Konzepte**
  - **Neue Datentypen, komplexe Objekte**
  - **Vererbung, Polymorphismus, Speichern von Operationen**
  - **Benutzerdefinierbare Zugriffsmethoden**
- **Spracherweiterungen von SQL-2 auf SQL-3 (SQL2003) ⇒**
  - **Löst viele Schwächen von RDBMS, alle Sprachelemente von SQL-2 werden unterstützt.**
  - **Neue Basisdatentypen: BOOLEAN, BLOB, CLOB, BIGINT**
  - **Neue Typkonstruktoren: ROW, ARRAY, REF, MULTISET**
  - **Benutzerdefinierte Datentypen (Distinct-Typ und strukturierte Typen)**
  - **Typhierarchien (Subtypen), typisierte Sichten und Sichthierarchien (Subsichten)**
  - **Bisherige Expertise und Investitionen bleiben erhalten**
- **Nachteile**
  - **Höhere Komplexität und Kosten (für eine kleine Klasse von Anwendungen ?)**
  - **SQL wird extrem komplex – noch ist der neue Standard nicht voll implementiert**

Relationenmodell

- **eine relationale Datenbank wird durch eine endliche Menge von Relationen bestimmter Relationstypen repräsentiert**
- **Relationstypen (auch relationale Schemata) werden durch einen Namen und ihre Attribute A_i mit deren Wertebereiche definiert: RT = $\{A_1,A_2,...A_k\}$**
- ***Relationstypen sind also ungeordnete Mengen*; hierfür wird häufig verkürzt RT ($A_1,A_2,...A_k$) geschrieben**
- **bei dieser Definition ist die Reihenfolge der Attribute irrelevant; sie entspricht der Implementation in RDBMS, bei denen Attribute über ihren Attributnamen angesprochen werden**
- **die übliche mathematische Definition einer Relation R als Teilmenge des kartesischen Produkts unterstellt eine willkürliche Numerierung der Attribute; nach dieser Definition sind A_1 x A_2 und A_2 x A_1 zu unterscheiden**
- **die Menge der Werte, die ein Attribut A_i annehmen kann, wird mit $W(A_i)$ bezeichnet; $W(A_i)$ wird auch als Domain von A_i bezeichnet**
- **typische Datentypen für Attribute sind: numerisch (Integer, Dezimal(p,q), Currency, Float), Zeichenketten (Char(n) bzw. Varchar(n)) und Date**
- **jeder RT besitzt mindestens einen Primärschlüssel (Unterstreichung)**
- **eine Relation ist eine Ausprägung eines Relationstyps; sie ist eine Menge**

Relationstypen und Relationen

- **ein Relationstyp beschreibt nur die statischen Eigenschaften von Objekt- und Beziehungstypen; alle Relationen eines RT haben die gleichen Eigenschaften**
- **(endliche) Relationen werden durch zweidimensionale Tabellen *repräsentiert,* deren Spalten aus dem Namen der Attribute bestehen; die Zeilen der Tabelle enthalten die Tupel der Relation, wobei *eine Reihenfolge nicht definiert ist***
- **das Ändern, Löschen, Hinzufügen von Tupeln einer Relation R erzeugt eine neue Relation R' des gleichen Relationstyps**
- **Objekttypen und Beziehungstypen werden in Relationstypen überführt**
- **in einer Relation sind alle Elemente voneinander verschieden; ein Element x aus R heißt auch Tupel, d.h. ein Tupel repräsentiert die Attributwerte eines bestimmten Objektes oder einer Beziehung**
- **Null-Werte repräsentieren nicht existente oder unbekannte Attributwerte; sie sind nicht für Primärschlüssel erlaubt**
- **eine Relation ist vergleichbar mit einer herkömmlichen Datei: jedem Datensatz entspricht ein Tupel der Relation, wobei jedoch weder die physische Dateistruktur noch die Speicherhierarchie sichtbar ist**
- **die Speicherstrukturen von Relationen sind transparent**

Relationstypen und Relationen (2)

- **als Sprache zur Datendefinition (DDL), Datenmanipulation (DML) und Datenkontrolle hat sich die Sprache SQL durchgesetzt, für die es seit 1989 Standards gibt**
- **Datendefinition, Datenzugriff, Datendarstellung, Datenspeicherung und Zugriffsberechtigungen beziehen sich grundsätzlich immer auf Relationen**
- **Relationen sind eigenständig, d.h. haben keine Verbindung untereinander; eine Verknüpfung von Relationen erfolgt nur über gemeinsame Attribute**
- **die konzeptionelle Einfachheit des Relationenmodells besteht also darin, daß es *aus Benutzersicht* (!) nur einen Datentyp, nämlich die Tabelle, gibt**
- **das Relationenmodell geht maßgeblich auf die Arbeiten von E.F. Codd am damaligen IBM Forschungslabor San Jose (1965-77) zurück; erster Prototyp 1978: System-R (IBM)**
- **Relationale Integritätsregeln**
  - **Objektintegrität: der definierte Primärschlüssel enthält zu keinem Zeitpunkt Nullwerte**
  - **referentielle Integrität bedeutet, daß für jeden Fremdschlüssel einer Relation R1, der Primärschlüssel in einer Relation R2 ist, gilt: jeder Wert des F-Schlüssels in R1 muß (mindestens) einem Wert des P-Schlüssels in R2 entsprechen oder ein Null-Wert sein**
  - **komplexere Integritätsregeln (z.B. transitionale Bedingungen) sind nicht standardisiert**

Relationale Operatoren

- **operieren auf Relationen und bilden die mathematische Basis der Datenmanipulation mit SQL**
- **alle Operationen können auf folgende Grundoperatoren zurückgeführt werden:**
  1. **Projektion einer Relation**
  2. **Auswahl aus einer Relation**
  3. **Kartesisches Produkt von Relationen (natürlicher Verbund)**
  4. **Vereinigung, Differenz und Durchschnitt kompatibler Relationen**
  5. **Umbenennung von Attributnamen**
- **viele SQL-Implementationen bieten weitere Operatoren an, die als Erweiterung der Relationenalgebra zu betrachten sind, z.B.:**
  - **Outer Join**
  - **Behandlung von Nullwerten mit dreiwertiger Logik**

A	B
a	b
a	?

B	C
b	c
?	c

Verbund über B →

A	B	C
a	b	c

? = b1 →

A	B	C
a	b	c
a	b1	c

Projektion von Relationen

- **gegeben ist eine Relation R mit den Attributen A_1, A_2, ..., A_n**
- **wir wählen r dieser Attribute aus (r < n) und fassen sie in der Auswahlliste T zusammen**
- **die Projektion P über T erzeugt aus R eine r-stellige Relation B, indem aus allen Tupeln von R die Komponenten gestrichen werden, die nicht in T sind**
- **Schreibweise: B = P(R,T)**
- **Beispiel: R sei die Relation Auftrag und T = (Auf-Nr); dann ist B = P(R,T):**

Auftrag

Auf-Nr	K-Nr	A-Nr	Bestellung
9012	4711	233	234
9012	4711	123	12
9034	3256	567	123

B

Auf-Nr
9012
9034

- **die Projektion wählt also aus einer Relation eine Teilmenge von Spalten aus, wobei doppelte Tupel aus der entstehenden Relation entfernt werden**

Auswahl und kartesisches Produkt von Relationen

Auswahl aus einer Relation

- **R erzeugt eine neue Relation B gleichen Grades, bei der alle Tupel x übernommen werden, die eine logische Auswahlbedingung L(x) erfüllen**
- **Schreibweise: B = S(R,L)**
- **Beispiel: B = S(Auftrag,L), wobei L definiert ist als: Bestellung > 100**

B

Auf-Nr	K-Nr	A-Nr	Bestellung
9012	4711	233	234
9034	3256	567	123

- **die Auswahl wählt also aus einer Relation eine Teilmenge der Zeilen aus**

Kartesisches Produkt von Relationen

- **sind A und B Relationen, so ist das kartesische Produkt C = A x B definiert als die Menge C = { (x,y) | x ∈ A, y ∈ B }**

Kompatible, Vereinigung und Differenz von Relationen

Definition von kompatiblen Relationen

- **gegeben seinen zwei Relationen A und B mit A = (A_1,A_2,..,A_n) und B = (B_1,B_2,..,B_k). A und B heißen kompatibel, wenn gilt:**
  - **sie haben den gleichen Grad, d.h. n = k**
  - **alle Wertebereiche der entsprechenden Attribute stimmen überein, d.h. $W_i(A_i) = W_i(B_i)$, i = 1, ..., k**

Vereinigung von Relationen

- **die Vereinigung C von zwei kompatiblen Relationen A und B ist die Vereinigungsmenge C = A ∪ B**

Differenz von Relationen

- **die Differenz C von zwei kompatiblen Relationen A und B ist die Differenzmenge C = A - B**

Innerer Theta-Verbund von Relationen

- **A und B seien Relationen und ein Attribute A_1 von A und B_1 von B mit kompatiblen Wertebereichen**
- **sei θ ein Vergleichsoperator aus { =, ¬ =, ≤, ≥, <, > }**
- **die Relation C des inneren Theta-Verbundes bezüglich A, B, A_1, B_1 und θ ist definiert als die Menge aller Tupel A x B für die A_1 θ B_1 wahr ist;**
- **Beispiel: A = (A_1,A_2,A_3), B = (B_1,B_2), der Theta-Verbund C mit A_2 < B_1 lautet:**

A

A1	A2	A3
1	2	3
4	5	6
7	8	9

B

B1	B2
3	1
6	2

C

A1	A2	A3	B1	B2
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

- **ein Theta-Verbund mit dem Vergleichsoperator = wird Equi-Join genannt**
- **Beispiel: A = (A_1,A_2), B = (B_1,B_2), der Equi-Join C mit A_2 = B_2 lautet:**

A

A1	A2
a	w
b	x
c	y

B

B1	B2
d	y
d	z

C

A1	A2	B1	B2
c	y	d	y

Natürlicher Verbund von Relationen (natural join)

- **erzeugt aus zwei Relationen A und B eine neue Relation C, wobei folgende Regeln zu beachten sind:**
  - **es seinen T_A und T_B Auswahllisten von Attributen aus A und B: T_A = (A_1,A_2,...A_k) und T_B = (B_1,B_2,...B_k), mit identischen Wertebereichen der Attributpaare (A_1,B_1), ..., (A_k,B_k)**
  - **in das kartesische Produkt C von A und B werden nur solche Tupel berücksichtigt, deren Werte in allen Attributen, die durch T_A und T_B definiert werden, übereinstimmen**
- **Schreibweise: C = A x B (T_A = T_B)**
- **Beispiel: sei T_A = (A_1,A_3) und T_B = (B_1,B_2), wobei die Wertebereiche von A_1 und B_1 bzw. von A_3 und B_2 identisch sein mögen; dann ist C = A x B (T_A = T_B):**

A

A1	A2	A3
a	b	c
u	v	w
x	y	z

B

B1	B2	B3	B4
a	c	1	2
x	z	6	9
r	s	5	7
x	z	7	3

- **der natürliche Verbund C = A x B (T_A = T_B) entspricht dem kartesischen Produkt A x B mit anschließender Auswahl $A.A_1$ = $B.B_1$, $A.A_2$ = $B.B_2$, ... $A.A_n$ = $B.B_n$ und anschließender Projektion, wobei entweder die Attribute aus T_A oder T_B entfernt werden**

C

A1	A2	A3	B3	B4
a	b	c	1	2
x	y	z	6	9
x	y	z	7	3

Äußerer Verbund (outer join)

- **im äußeren Verbund werden auch Tupel mit Null-Werten in entsprechenden Spalten, in die Ergebnisrelation übernommen, die keine Entsprechung haben**
- **beim Equi-Join und äußeren Verbund sind drei Fälle zu unterscheiden:**
  - **beim linken äußeren Verbund werden Tupel aus der linken Relation übernommen und ggf. rechts mit Nullwerten ergänzt; Schreibweise A *= B für die Equi-Join-Attribute**
  - **beim rechten äußeren Verbund werden Tupel aus der rechten Relation übernommen und ggf. links mit Nullwerten ergänzt; Schreibweise A =* B für die Equi-Join-Attribute**
  - **der zweiseitige äußere Verbund, Schreibweise A *=* B, ist definiert als Vereinigungs-menge der anderen beiden outer joins**
- **Beispiel 1: A = (A_1,A_2), B = (B_1,B_2) seien vom Equi-Join-Beispiel; es sollen linker äußerer Verbund C mit A_2 *= B_2 und rechter äußerer Verbund D mit A_2 =* B_2 bestimmt werden:**
- **Beispiel 2: Kunde (*K-Nr*, ...), Auftrag (Auf-Nr, K-Nr, ...) Beim linken äußere Verbund mit Kunde.K-Nr *= Auftrag.K-Nr werden auch Kunden ohne Aufträge in die Ergebnisrelation übernommen**

C (linker äußerer Verb.)

A1	A2	B1	B2
a	w	?	?
b	x	?	?
c	y	d	y

D (rechter äußerer Verb.)

A1	A2	B1	B2
c	y	d	y
?	?	d	z

Modellierungsprobleme und Abfrageprobleme im Relationenmodell

- **Modellierungsprobleme im Relationenmodell**
  - **es gibt eine m:n-Beziehung *Studiert* zwischen Dozenten und Studenten**
  - **Dozent (D-Nr, D-Name, D-Daten), Student (M-Nr, S-Name, S-Daten),**
  - **Studiert (D-Nr, M-Nr, Daten)**
  - **wer (S-Name, S-Daten) studiert bei der Dozentin mit dem Namen (D-Name) Smart? Es ergeben sich drei Suchprozesse, die entsprechend ineffizient sind**
  - **einfache Fragestellungen können also bereits zu komplexen Abfragen führen**
- **Nichtvollständigkeit der relationalen Operatoren (SQL-1, SQL-2)**
  - **gegeben ist der Relationstyp Angestellte (P-Nr, Name, Attribute, P-Nr-Manager)**
  - **es werden folgende Annahmen getroffen:**
  - **der RT Angestellte repräsentiert alle Angestellten eines Unternehmens**
  - **P-Nr: eindeutige Personal-Nr einer Angestellten**
  - **Name: Name der Angestellten**
  - **Attribute: weitere Attribute, bzw. Attributgruppen**
  - **P-Nr-Manager: Personal-Nr der / des Vorgesetzten im gleichen Relationstyp; es wird unterstellt, dass dieses Attribut keine Null-Werte erlaubt**
  - **für eine Angestellte, die keine(n) Vorgesetzte(n) hat, d.h. in der Hierarchie an der „Spitze“ des Unternehmens steht, ist ihre Personal-Nr identisch zu P-Nr-Manager**
  - **es wird folgende Fragestellung betrachtet: zu einem Angestellten mit gegebener P-Nr sollen alle direkten und indirekten Vorgesetzten ermittelt werden**
  - **diese Fragestellung kann nicht mit den relationalen Operatoren gelöst werden**

5 Relationales Datenbankdesign 1

- **Relationstypen können eine Reihe von Strukturdefiziten aufweisen, die zu folgenden Problemkreisen gehören:**
  - ***Datenredundanz*, d.h. ein und derselbe Attributwert wird mehrfach gespeichert; daraus ergeben sich u.a. potentielle Inkonsistenzen bei Änderungen dieses Attributwertes**
  - ***Löschanomalien*, d.h. durch das Löschen eines Tupels gehen Informationen unwiderruflich verloren**
  - ***Repräsentanzprobleme*, d.h. bestimmte Informationen lassen sich nicht mit den gegebenen Relationstypen darstellen; man bezeichnet dies auch als Einfügeanomalie**
- **einige Strukturdefizite sind durch *Zerlegung* der Relationstypen vermeidbar, ein Vorgang, den man als *Normalisierung* bezeichnet**
- **Beispiel: Der Relationstyp Bücher beschreibt eine Datenbank des Lehrstuhls: Bücher (Sig-Nr, Autor, Schlagwort, Titel, Sach-Nr, Sachgebiet)**

Sig-Nr	Autor	Schlagwort	Titel	Sach-Nr	Sachgebiet
X1	Korth	RDBMS	Database Systems	13	Datenbanken
X1	Silber	RDBMS	Database Systems	13	Datenbanken
X2	Denert	Systementwickl.	Software Engineering	10	Systementw.
X2	Denert	Datenmodellier.	Software Engineering	10	Systementw.
X3	Balzert	Systementwickl.	Systementwicklung	10	Systementw.

- **Bücher weist alle oben genannten Strukturdefizite auf, z.B.:**
  - **ein Schlagwort kann erst eingetragen werden, wenn ein Buch dafür existiert**
  - **mit dem Löschen des letzten Buches zu einem Sachgebiet geht dieses selbst verloren**

Normalisierung von Relationen und Normalformen

- ***Normalformen* (NF) kennzeichnen den Zustand, in dem sich Relationstypen befinden; man unterscheidet erste (1NF), zweite (2NF), dritte (3NF), Boyce-Codd NF (BCNF), vierte (4NF) und Project-Join-Normalform (PJNF, 5NF)**
- **NF basieren auf den Konzept *funktionaler Abhängigkeit* zwischen Attributen eines Relationstyps und *mehrwertigen Abhängigkeiten***
- **je höher eine NF ist, desto restriktiver sind die Anforderungen an den Relationstypen; eine NF der Stufe k erfüllt auch die Kriterien der Stufe i wenn k > i gilt; die BCNF liegt zwischen 3NF und 4NF**
- **unter *Normalisierung* versteht man die Zerlegung eines Relationstypen in solche höherer Normalformen**
- **die Zerlegung sollte so beschaffen sein, daß sie durch Natural-Join-Operationen rückgängig gemacht werden kann (*verlustfreie Zerlegung*)**
- **weiterhin sollen alle funktionalen Abhängigkeiten eines Relationstypen auch durch die normalisierten Typen erfüllt werden (*Abhängigkeitstreue*)**
- **unter *Denormalisierung* versteht man den umgekehrten Vorgang. Eine kontrollierte Denormalisierung kann aus Performance-Gründen erforderlich werden, um höhere Datenlokalität zu erreichen**

Grundlegende Definitionen

- **gegeben sei ein Relationstyp RT (A_1,A_2,...A_k), wobei die A_i Attribute sind**
- **bei Aussagen, die einen bestimmten Relationstyp RT betreffen, sprechen wir auch kurz von einer Relation R des Typs RT**
- **grundlegend ist die Unterscheidung zwischen Relationstyp und Relation; eine Relation ist eine spezielle Ausprägung eines Relationstyps**
- **alle Aussagen über funktionale u. mehrwertige Abhängigkeiten, NF, verlustfreie u. abhängigkeitstreue Zerlegungen gelten grundsätzlich für RTs**
- **es werden im folgenden Attribute X, Y, Z des Relationstyps RT verwandt, mit X, Y, Z $\subseteq$ {A_1,A_2,...A_k}, d.h. X, Y, Z können auch Attributgruppen sein**
- **ein Relationstyp RT befindet sich in 1NF, wenn die Wertebereiche der Attribute A_i elementar aus der logischen Datensicht sind**
- **ein Kandidatenschlüssel (Schlüsselkandidat) eines Relationstypen RT ist eine minimale Attributmenge, die jedes Attribut von RT eindeutig bestimmt**
- **ein Relationstyp kann mehrere Schlüsselkandidaten aufweisen**
- ***Schlüsselattribut*: ein Attribut, das Bestandteil eines Schlüsselkandidaten ist**
- ***Nichtschlüsselattribut*: ein Attribut, das in keinem Kandidatenschlüssel der betrachteten Relationstypen vorkommt**

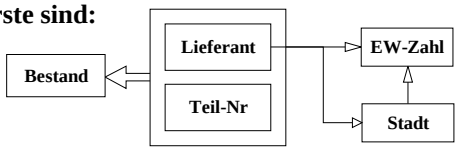
Normalformen auf Basis funktionaler Abhängigkeiten

- ***Funktionale Abhängigkeit von Attributen*: In einer Relation des Typs RT mit den Attributen X und Y heißt Y funktional abhängig von X, wenn es zu jedem Wert des Attributes X genau einen Wert des Attributes Y gibt.**
- **Schreibweise: R.X $\rightarrow$ R.Y**
- **ist Y nicht von X funktional abhängig, so schreibt man auch R.X /$\rightarrow$ R.Y**
- **ist der Relationstyp eindeutig, so schreibt man nur X $\rightarrow$ Y**
- **Bemerkungen:**
  - **f. A. der Attribute eines Relationstyps ergeben sich aus den Sachzusammenhängen**
  - **f. A. können an Hand einer Relation nur falsifiziert werden**
  - **ist X Schlüsselkandidat, dann hängen per Definition alle anderen Attribute des Relationstyps R funktional von X ab**
- ***Volle funktionale Abhängigkeit*: Y heißt voll funktional abhängig von X, wenn:**
  - **Y von X funktional abhängig ist**
  - **es gibt keine echte Teilmenge der Attribute von X, von denen Y funktional abhängig ist**
  - **Schreibweise R.X $\Rightarrow$ R.Y**
  - **Bemerkung: eine funktionale Abhängigkeit X $\rightarrow$ Y kann nur dann keine volle funktionale Abhängigkeit sein, wenn X ein zusammengesetztes Attribut ist**

Beispiele

- **Der Relationstyp Erste beschreibt Lieferanten, deren eindeutigen Standort und Teilebestände. EW-Zahl ist die Einwohnerzahl der Stadt:**
  - **Erste (<u>Lieferant</u>, <u>Teil-Nr</u>, Stadt, EW-Zahl, Bestand)**
  - **Die funktionalen Abhängigkeiten der Relation Erste sind:**

(Lieferant, Teil-Nr) ⇒ Bestand
Lieferant → Stadt
Lieferant → EW-Zahl
Stadt → EW-Zahl



- **Die f. A. des Banken-Beispiels sind (wobei erst hier Annahmen präzisiert werden):**

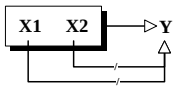
Kunde: K-Name → Stadt, K-Name → Strasse
Zweigstelle: Z-Name → Stadt, Z-Name → Spar-Einl
Girokonto: Konto-Nr → Kontostand, Konto-Nr → Z-Name
Kredit: Kredit-Nr → Kreditbetrag, Kredit-Nr → Z-Name

- **Die f. A. von Bücher sind:**

(Sig-Nr, Autor, Schlagwort) → Titel, Sach-Nr, Sachgebiet
(Sig-Nr, Autor) → Titel, Sach-Nr, Sachgebiet
Sig-Nr → Titel, Sach-Nr, Sachgebiet
Sach-Nr → Sachgebiet

Die zweite Normalform

- **Ein Relationstyp R befindet sich genau dann in der 2NF, wenn:**
  - **er sich in 1 NF befindet**
  - **jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängt**
  - **ein RT R kann also nur dann die 2NF verletzen, wenn R einen zusammengesetzten Kandidatenschlüssel und mindestens ein Nichtschlüsselattribut enthält**
  - **ist R ein RT mit (X1,X2) als Schlüsselkandidat und einem anderen Attribut Y, so läßt sich die 2NF über die funktionalen Abhängigkeiten folgendermaßen veranschaulichen:**



- **Erste ist nicht in 2NF, da EW-Zahl und Stadt f.a. von Lieferant sind**
- **wir zerlegen Erste in zwei RTs Zweite und Bestände:**
  - **Zweite (<u>Lieferant</u>, Stadt, EW-Zahl)**
  - **Bestände (<u>Lieferant</u>, <u>Teil-Nr</u>, Bestand)**
  - **Beide RTs sind in 2NF**
- **Die RTs Kredit, Girokonto sind nicht in 2NF. Die zerlegten RTs sind in 2NF:**
  - **Kredit-Info (<u>Kredit-Nr</u>, Z-Name, Kreditbetrag)**
  - **Kunde-Kredit (<u>K-Name</u>, <u>Kredit-Nr</u>)**
  - **Giro-Info (<u>Konto-Nr</u>, Z-Name, Kontostand)**
  - **Kunde-Giro (<u>K-Name</u>, <u>Konto-Nr</u>) ist in 2NF (sogar in BCNF)**

Beispiele 2NF - Fortsetzung

- **Im Relationstyp Bücher (Sig-Nr, Autor, Schlagwort, Titel, Sach-Nr, Sachgebiet) ist (Sig-Nr, Autor, Schlagwort) (einziger) Schlüsselkandidat wegen der Sig-Nr → (Titel, Sach-Nr, Sachgebiet) ist Bücher nicht in 2NF**
- **zur weiteren Normalisierung bietet sich folgende Zerlegung an:**
  Bücher-Sachgebiet (Sig-Nr, Titel, Sach-Nr, Sachgebiet)
  Autor-Schlagwort (Sig-Nr, Autor, Schlagwort)
- **beide Relationen sind in 2NF**
- **Bücher-Sachgebiet enthält nach dieser Zerlegung immer noch Redundanzen**

Bücher-Sachgebiet

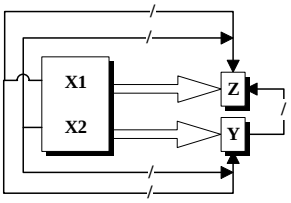
Sig-Nr	Titel	Sach-Nr	Sachgeb
X1	Database Systems	13	Datenbanken
X2	Softw-Engineering	10	Systementw
X3	Systementwicklung	10	Systementw

Autor-Schlagwort

Sig-Nr	Autor	Schlagwort
X1	Korth	RDBMS
X1	Silber	RDBMS
X2	Denert	Sysen
X2	Denert	Damod
X3	Balzert	Sysen

Transitive Abhängigkeit und 3 NF

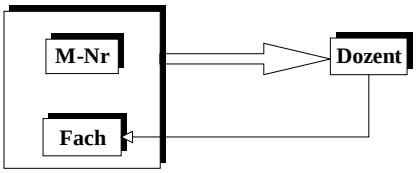
- **In einem Relationstyp RT mit RT = (X, Y, Z) ist das Attribut Z transitiv vom Schlüsselkandidaten X abhängig, wenn gilt:**
  - **Y ist kein Schlüsselkandidat**
  - **Z ist funktional abhängig von Y**
  - **Schematisch: X → Y, Y → Z, aber Y /→ X**
  - **R kann also nur dann transitive Abhängigkeiten aufweisen, wenn mindestens zwei Nichtschlüsselattribute existieren, von denen eines von dem anderen funktional abhängt.**
- **ein Relationstyp befindet sich genau dann in der 3NF, wenn:**
  - **er sich in der 2NF befindet**
  - **kein Nichtschlüsselattribut transitiv von einem Kandidatenschlüssel abhängt**



- **ein Relationstyp RT kann also nur dann nicht in 3NF sein, wenn neben einfachen oder zusammengesetzten Kandidatenschlüsseln mindestens zwei Nichtschlüsselattribute existieren**
- **Beispiel: Person (P-Nr, Name, Sozialvers -Nr) ist in 3NF, da Sozialvers-Nr Kandidatenschlüssel ist**

Beispiele zur 3NF

1. **Zweite befindet sich nicht in 3NF, da EW-Zahl transitiv über Stadt von Lieferant abhängt. Zerlegt man Zweite in die Relationstypen Standort und Städte, wobei:**
   Standort (Lieferant, Stadt) und Städte (Stadt, EW-Zahl),
   so befinden sich alle R-Typen Bestände, Standort und Städte in der 3NF
2. **Der Relationstyp SFD (M-Nr, Fach, Dozent) beschreibt, welches Fach ein Student bei welchem Dozenten in einem Semester hört; Annahmen:**
   - **Dozent ist eindeutig**
   - **ein Student hört ein Fach bei nur einem Dozenten**
   - **ein Dozent liest genau ein Fach**
   - **ein Fach wird i.d.R. von mehreren Dozenten gelesen**
   - **(M-Nr, Dozent) ist also auch Schlüsselkandidat**
   - **man hat folgende funktionale Abhängigkeiten:**



   - **SFD ist in 3NF, da keine Nichtschlüsselattribute existieren**
   - **SFD weist trotzdem sowohl Repräsentanz- als auch Löschanomalien auf:**
     ist (4711, Psychologie, Schmidt) das einzige Tupel in SFD mit Dozentin Schmidt und wird dieses Tupel gelöscht, dann geht die Information verloren, daß Schmidt Psychologie liest

Beispiele zur 3NF (2)

3. **Der Relationstyp Bücher-Sachgebiet (Sig-Nr, Titel, Sach-Nr, Sachgebiet) weist folgende funktionale Abhängigkeiten auf:**
   Sig-Nr --> (Titel, Sach-Nr, Sachgebiet)
   Sach-Nr --> Sachgebiet
   Demnach existiert eine transitive Abhängigkeit
   Sig-Nr --> Sach-Nr --> Sachgebiet und Sach-Nr ist kein Schlüsselkandidat.
   Bücher-Sachgebiet ist also nicht in 3NF; wir zerlegen diesen Relationstyp in:
   Buch (Sig-Nr, Titel, Sach-Nr)
   Sachgebiet (Sach-Nr, Sachgebiet)
   Sach-Nr in Buch ist ein Fremdschlüssel für den Relationstypen Sachgebiet

Buch

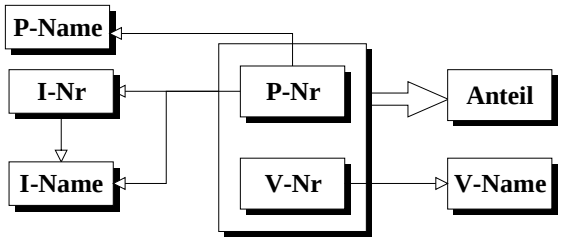
Sig-Nr	Titel	Sach-Nr
X1	Database Systems	13
X2	Softw-Engineering	10
X3	Systementwicklung	10

Sachgebiet

Sach-Nr	Sachgebiet
10	Systementw
13	Datenabanken

Beispiel zur 3NF (3)

- **Dozenten gehören jeweils einem Universitätsinstitut an und werden durch ihre Personalnummer identifiziert; sie halten i.d.R. mehrere Vorlesungen**
- **in einem Institut sind i.d.R. mehrere Mitarbeiter tätig**
- **eine Vorlesung wird durch Namen und Vorlesungs-Nr. definiert**
- **an einer Vorlesung können mehrere Mitarbeiter mit einem prozentualen Zeitaufwand beteiligt sein**
- **Universität (P-Nr, P-Name, I-Nr, I-Name, V-Nr, V-Name, Anteil) mit**
  - **P-Nr: Personal-Nr eines Dozenten P-Name: Name eines Dozenten**
  - **I-Nr: Instituts-Nr I-Name: Institutsname**
  - **V-Nr: Vorlesungs-Nr V-Name: Vorlesungsname**
  - **Anteil: Semesteranteil (%) eines Mitarbeiters an einer Vorlesung**
- **die funktionalen Abhängigkeiten der Attribute von Universität sind:**



- **Universität wird in vier RT in 3NF zerlegt:**
  - **Vorlesungen (V-Nr,V-Name)**
  - **Mitarbeiter (P-Nr,P-Name,I-Nr)**
  - **Institute (I-Nr,I-Name)**
  - **Lehrangebot (P-Nr,V-Nr,Anteil)**

Boyce-Codd Normalform

- **2NF und 3NF erfassen nur Abhängigkeiten der Nichtschlüsselattribute von Kandidatenschlüssel, nicht jedoch Abhängigkeiten zwischen diesen**
- **Probleme kann es geben, wenn RT folgende Merkmale aufweist:**
  - **mehrere Kandidatenschlüssel**
  - **zusammengesetzte Kandidatenschlüssel**
  - **überlappende Kandidatenschlüssel, d.h. die zusammengesetzten Schlüssel weisen mindestens ein gemeinsames Attribut auf**
- **Determinante**
  - **ein Attribut X eines Relationstyps RT heißt Determinante, wenn es mindestens ein anderes Attribut von RT gibt, das von X *voll funktional abhängt***
  - **Beispiel: Der Relationstyp Erste hat folgende Determinanten: (Lieferant,Teil-Nr), Lieferant, Stadt**
- **Boyce-Codd Normalform**
  - **ein Relationstyp R ist genau dann in Boyce-Codd Normalform (BCNF), wenn jede Determinante von R ein Schlüsselkandidat ist**
  - **die BCNF ist die höchste Normalform auf der Basis funktioneller Abhängigkeiten**
  - **eine Relation in BCNF ist auch in 3NF**
  - **die BCNF wird für das Datenbankdesign der Basisrelationstypen angestrebt**
- **SFD ist in 3NF, jedoch nicht in BCNF, da Dozent Det. jedoch kein SK ist**

6 Structured Query Language (SQL)

- **Vorläufer Sequel wurde 1974 im System-R-Projekt bei IBM entwickelt**
- **Datenmanipulationen basieren auf relationalen Operatoren**
- **ANSI-Standard seit 1986, erweitert 1989; der Sprachumfang von 1989 wird auch als SQL-1 bezeichnet, neuere Standards: SQL-2 bzw. SQL-3**
- **trotz Standardisierung ist keine Portabilität gewährleistet, da alle Anbieter Erweiterungen zulassen und *embedded SQL* zu unpräzise definiert ist**
- **SQL-1 weist Sprachelemente für folgende Funktionen auf:**
  - ***Datendefinition* (data definition language, DDL) Create (table, index, view), alter table (kein ANSI) drop (table, index, view)**
  - ***Abfragebefehle* (query language, QL) Select (where, expressions, order by, group by, join, subqueries)**
  - ***Datenmanipulation* (data manipulation language, DML) insert, update, delete**
  - ***Transaktionsverarbeitung*: Transaktionen können durch commit abgeschlossen und durch rollback rückgängig gemacht werden; einige SQL-Implementationen erlauben auch das explizite Sperren von Daten auf Tupel oder Relationenebene**
  - ***Datenkontrolle* (data control language, DCL) Anweisungen für Authorisierungen, Zugriffsrechte zu Relationen und Sichten: authority (connect, resource, DBA), priviliges (grant, revoke)**

SQL-Sprachnormen

- **dienen der Portabilität, dem Investitionsschutz, Unabhängigkeit von einem Anbieter; dies gilt besonders für SQL als Schnittstelle zu einem RDBMS**
- **wichtige Normungsinstitutionen für SQL**
  - **ANSI: American National Standards Institute, legt alle US-amerikanischen Standards fest, vergleichbar zur deutschen DIN-Behörde**
  - **ISO: International Organisation for Standardization, Zusammenschluß nationaler Standardisierungs-Organisationen mit dem Ziel, weltweit bindende Standards zu schaffen**
- **1986 erste SQL-Norm von ANSI, genannt SQL-1**
- **1989 Erweiterung von SQL-1, ANSI/ISO-Standard (< 100 Seiten) Wesentliche Neuerungen waren:**
  - **Check und Default-Options für Werteinschränkungen von Attributen im Create-Befehl**
  - **Objekt- und referentielle Integrität**
- **weiterer ANSI-Standard von 1989 regelt embedded SQL für die Trägersprachen Ada, C, COBOL, FORTRAN, PASCAL und PL/1;**
- **1992 ISO-Standard für netzwerkweiten Datenbankzugriff im Rahmen von Client-Server-RDBMS-Anwendungen**
- **1992 ISO Standard für SQL-2 (SQL/92); Spezifik. umfasst > 600 Seiten!**
- **1999 ANSI/ISO-Standard für SQL-3 (SQL/99) mit objektrelationalen Erweiterungen, komplexen Datentypen u. benutzerdef. Prozed. (> 1100 S.)**

SQL-Formen und Schnittstellen

- **SQL/CLI (Call Level Interface) für SQL-Calls über Objectcode-Bibliotheken**
  - **Gegenstück zu embedded SQL, ca. 35 Firmen bilden die SQL-Access Group, die an Standardisierung von SQL/CLI arbeiten**
  - **API-Technik d.h. Anwenderprogramme, die mit kompatiblen Compilern kompiliert wurden, können auf eine RDB zugreifen**
  - **Nachteil ist geringerer Datenschutz, da Routinen mit CLI-Aufrufen ggf. modifiziert werden können; ein embedded SQL-Programm ist zugangssicherer**
- **Open Database Connectivity (ODBC) ist ein SQL/CLI-Standard, der sich ursprünglich an Windows ausrichtete und von Microsoft initiiert wurde**
  - **Ziel: Daten in heterogenen RDBMS Windows und NT-Clients zugänglich zu machen**
  - **scrollende Cursor erlauben in einer Tabelle vorwärts oder rückwärts zu scrollen, ohne Netzlast durch weitere Anfragen zu erzeugen**
- **man unterscheidet folgende SQL-Formen:**
  - **interaktives SQL: SQL-Anweisungen werden interaktiv direkt eingegeben und unmittelbar ausgeführt; Ergebnistabellen werden auf dem Bildschirm angezeigt**
  - **eingebettetes SQL (embedded): SQL-Anweisungen werden in eine Trägersprache, z.B. COBOL oder C eingebettet; das Programm muß meistens präkompiliert werden**
    - SQL-Anweisungen werden durch Calls auf RDBMS-Moduln ersetzt
    - das modifizierte Quellenprogramm muß dann compiliert und gelinkt werden
  - **Dynamisches SQL: die genaue Form einer SQL-Anweisung wird erst zur Laufzeit definiert, z.B. eine logische Bedingung für eine where-Klausel**

Anlegen einer Beispieldatenbank - fiktives Beispiel

- **ein Bankhaus hat Zweigstellen in verschiedenen Städten**
- **es werden nur Girokonten und Kredite betrachtet**
- **ein Kunde kann mehrere Konten oder Kredite haben**
- **ein Konto oder Kredit kann zu mehreren Kunden gehören**
- **der Einfachheit wird Kunden-Name als Primärschlüssel definiert**
- **diese Anwendung wird durch folgende Relationstypen definiert:**
  - **Kunde (<u>K-Name</u>, Straße, Stadt)**
  - **Zweigstelle (<u>Z-Name</u>, Stadt, Spar-Einl)**
  - **Girokonto (<u>K-Name</u>, <u>Konto-Nr</u>, Z-Name, Kontostand)**
  - **Kredit (<u>K-Name</u>, <u>Kredit-Nr</u>, Z-Name, Kreditbetrag)**
- **die Relationstypen werden interaktiv erstellt (z.B-MS Access) oder mit der SQL-Anweisung create table definiert:**

```
create table  Kunde
              (K_Name          char(20) not null,
               Strasse         char(30),
               Stadt           char(30),
               primary key     (K_Name))
```

Beispieldatenbank (2)

```
create table   Zweigstelle
               (Z_Name          char(15) not null,
                Stadt           char(30),
                Spar_Einl       decimal (15,2),
               primary key      (Z_Name))
create table   Girokonto
               (K_Name          char(20) not null,
               Konto_Nr,        integer not null,
                Z_Name          char(15),
                Kontostand      decimal (15,2),
                primary key     (Konto_Nr, K_Name)
                foreign key     (Z_Name) references Zweigstelle,
                foreign key     (K_Name) references Kunde)
create table   Kredit
               (K_Name          char(20) not null,
                Kredit_Nr,      integer not null,
                Z_Name          char(15),
                Kreditbetrag    decimal (15,2),
                primary key     (Kredit_Nr, K_Name)
                foreign key     (Z_Name) references Zweigstelle,
                foreign key     (K_Name) references Kunde)
```

SQL-Sprachelemente

- **Datentypen in ANSI-SQL**
  - **smallint** — 16-Bit-Dualzahl mit Vorzeichenbit
  - **integer** — 32-Bit-Dualzahl mit Vorzeichenbit
  - **decimal(m,n)** — m-stellige Zahlen ($m \leq 15$) mit n Nachkommastellen, $0 \leq n \leq m$
  - **float(p,q)** — Gleitkommazahlen (hardwareabhängig)
  - **char(n)** — Zeichenketten der Länge n, $1 \leq n \leq 254$
  - **varchar(n)** — Zeichenketten variabler Länge bis 254 Zeichen
  - **long varchar** — Zeichenketten var. Länge bis 32.767 Zeichen
  - **date** — Datumswerte ('tt.mm.jjjj')
  - **time** — Zeitwerte ('hh.mm[.ss]')
  - **timestamp** — Datum/ Zeit ('jjjj-mm-tt-hh.mm.ss.zzzzzz') z : microseconds
- **einige ANSI-SQL-Datentypen müssen in Access anders deklariert werden:**
  - Einige ANSI-SQL-Datentypen werden nicht unterstützt
  - **integer** — 16-Bit-Dualzahl mit Vorzeichenbit
  - **long** — 32-Bit-Dualzahl mit Vorzeichenbit
  - **single** — 32-Bit Gleitkommazahl (IEEE)
  - **double** — 64-Bit Gleitkommazahl (IEEE)
  - **string** — Zeichenkette (0,..65.535 Bytes)
  - **currency** — 64-Bit Dualzahl mit Vorzeichenbit, auf 10000 skaliert, daraus ergeben sich Dezimalzahlen mit 15 Vorkomma und vier Nachkommastellen
- **Wichtig sind auch BLOBs und CLOBs: binary / character large objects**

Zeichenketten und SQL

- **Überlegungen: wie groß ist die minimale, maximale Länge einer Zeichenkette, welche durchschnittliche Länge besitzt sie und wie groß ist die Varianz ?**
- **Bei char-Feldern besitzt das Feld immer die gleiche feste Länge, eine kürzere Zeichenkette wird mit Leerzeichen ergänzt**
- **Der ANSI-Standard definiert dies auch bei varchar-Feldern**
- **Bei einigen RDBMS, z.B. mySQL wird nur die exakte Länge plus einem (1, 2 oder 4 Byte) Längenfeld gespeichert**
- **Besitzt eine Zeichenkette nur eine geringe Varianz, z.B. eine Telefon-Nr. (char(13) oder z.B. der ISO-genormte Ländercode (char(3), z,B. US, FR, D,.., so ist eine char-Variable optimal**
- **Nimmt ein Feld z.B. eine URL auf, dann kann es z.B. by MySQL sinnvoll sein, dafür ein varchar-Feld zu verwenden, denn es gibt URLs, die weit über 100 Zeichen lang sind**
- **Bei einigen RDBMS können varchar-Felder nicht indiziert werden; damit können Suchoperationen über solche Felder i.a. nicht so schnell ausgeführt werden**

Logische Bedingungen in SQL

- **Konstanten-Vergleich: Attribut mit einer Konstanten**
- **einfache arithmetische Vergleiche, z.B. where preis <= 1000**
- **mit and, or oder not zusammengesetzte logische Bedingungen**
- **between z.B. where gebjahr between 1960 and 1970**
- **in, z.B. where stadt in ('Berlin', 'München', 'Hamburg')**
- **like, z.B. where zuname like "_M%"**
- **quantifizierte Bedingungen: all, any, some und exists**
- **besteht die Ergebnistabelle einer Unterabfrage aus mehr als einem Tupel, dann muß eines der Prädikate all, some, in, exists zwingend verwendet werden**
- **all bzw some, z.B. where gebjahr <= all (select-Komponente)**
- **exists, z.B. where exists (select-Komponente)**
- **Vergleich zweier Attribute mit kompatiblen Wertebereichen**
- **Verbundbedingung: relation1.attribut = relation2.attribut**
- **Null-Selektion: attribut *is null***

Tabelle löschen, Tupel eingeben, ändern, löschen

- **löschen einer Tabelle mit allen Indizes, Views und Befugnissen aus der Datenbank: drop table tabname**
- **Tupel erfassen: eine Möglichkeit ist die Insert-Anweisung:**
  Insert into [creator.]{tabname | viewname} [(attribute)]
  values (attributwerte | select Anweisung)
  Beispiel: insert into Girokonto values ("Mops", 4711, "Kiel", 1400)
- **Tupel ändern: update [creator.]{tabname | viewname}**
  set attribut1 = wert1 [, attribut2 = wert2] ... [where Bedingung]
  - **fehlt die where-Klausel, so werden alle Tupel verändert !**
  - **Beispiel 1: update Girokonto set Kontostand = Kontostand * 1.01**
  - **Beispiel 2: Konten über DM 10.000,- sollen 2% Guthabenzinsen bekommen, solche darunter 1% :**
    update Girokonto set Kontostand = Kontostand * 1.02 where Kontostand > 10000
    update Girokonto set Kontostand = Kontostand * 1.01 where Kontostand <= 10000 and Kontostand >= 0
- **Tupel löschen: delete from [creator.]{tabname | viewname} [where bedingung]**
  - **fehlt die where Klausel , so werden alle Tupel der Relation gelöscht**

Select-Anweisung

- **dient zur Abfrage von Tabellendaten, wobei das Ergebnis eine Tabelle ist**
  select [all | distinct] { [x.]attribut [, ...] | * }
  from [owner.]{ tabname | viewname } x [, ...]
  [where logische Bedingung]
  [group by [x.]attribut [, ...] [having bedingung]]
  [order by { [x.]attribut | nummer } [asc | desc] [, ...]]
- **nur die select und die from-Komponente müssen verwendet werden**
- **die Reihenfolge der Komponenten ist einzuhalten**
- **having darf nur nach group by verwendet werden**
- **x: Kürzel für Tab- o. Viewnamen, wird in der from Komponente spezifiziert**
- **select wählt die Attribute der Ergebnistabelle; Distinct: keine doppelte Zeilen**
- **from bestimmt, aus welchen Tabellen die Daten abgefragt werden**
- **where erlaubt die Spezifikation von logischen Bedingungen**
- **group by gruppiert die Zeilen auf Basis einer oder mehrerer Spalten**
- **having selektiert für group by bestimmte Tupel durch eine log. Bedingung**
- **order by sortiert das Ergebnis nach Attributwerten; desc absteigend**

Verarbeitung von SQL-Befehlen

1. **FROM: Definiert die Ausgangstabellen**
2. **WHERE: Selektiert die Tupel, die der Bedingung genügen**
3. **GROUP BY: Gruppiert die Tupel auf Basis gleicher Werte**
4. **HAVING: Selektiert Gruppen, die der Bedingung genügen**
5. **SELECT: Selektiert Attribute**
6. **ORDER BY: Sortiert die Tupel**

```
SELECT SpielerNr, SUM(Betrag) AS Summe
FROM Strafen WHERE YEAR(Datum)>1995
GROUP BY SpielerNr
HAVING COUNT(Betrag)>=2
ORDER BY SUM(Betrag) ASC;
```

1.

SpielerNr	Datum	Betrag
2	1.1.2000	100
6	15.3.1997	50
2	1.11.1999	100
1	32.5.1994	20
4	30.6.1998	200
6	4.5.2000	50

2.

SpielerNr	Datum	Betrag
2	1.1.2000	100
6	15.3.1997	50
2	1.11.1999	100
4	30.6.1998	200
6	4.5.2000	50

3.

SpielerNr	Datum	Betrag
2	1.1.2000, 1.1.99	100, 100
6	15.3.1997, 30.6.1998	50, 50
4	30.6.1998	200

4.

SpielerNr	Datum	Betrag
2	1.1.2000, 1.1.99	100, 100
6	15.3.1997, 30.6.1998	50, 50

5.

SpielerNr	Summe
2	200
6	100

6.

SpielerNr	Summe
6	100
2	200

Implementierung der relationalen Operatoren in SQL

- **Projektion von Relationen: sei R (A_1,A_2,...,A_n) und T = (A_1,A_2,...A_k) eine Auswahlliste; dann lautet die Projektion B = P(R,T) in SQL:**
  Select distinct A_1,A_2,...A_k from R
- **der Auswahloperator B = S(R,L) lautet in SQL: Select * from R where L(x)**
  - **die logische Bedingung L(x) kann auch eine Kombination einzelner Bedingungen sein**
  - **je zwei Attributpaare (A_i, B_i) können durch einen Vergleichsoperator ◊ aus { =, ¬=, ≤, ≥, <, > } verknüpft werden und diese Ausdrücke durch logische Operatoren lop aus { and, or, not } kombiniert werden: A_1 ◊ B_1 lop A_2 ◊ B_2 ... lop A_n ◊ B_n**
  - **Auswertungsreihenfolge von links nach rechts; Klammern erleichtern Festlegung**
- **Kartesisches Produkt C = A x B zweier Relationen A, B lautet in SQL:**
  Select * from A, B
- **Innerer Theta-Verbund lautet in SQL: Select * from A, B where L(x)**
- **Zwei Relationen R1(A_1,A_2,...,A_n) und R2(B_1,B_2,...,B_m) sind kompatibel, wenn n = m und dom(A_i) =dom (B_i) for $1 \le i \le n$**
- **Vereinigung C von zwei kompatiblen Relationen A und B lautet in SQL:**
  - **Select * from A union Select * from B**
  - **A ∪ B enthält alle Tupel die sowohl in A als auch in B zu finden sind (keine Duplikate)**

Einfache Select-Beispiele

- **finde alle Kunden, die ein Girokonto in einer Zweigstelle in Kiel haben:**
  select distinct K_Name from Girokonto where Z_Name = "Kiel"
- **finde alle Kunden, die ein Girokonto, Kredit oder beides in einer Zweigstelle in Kiel haben:(select K_Name from Girokonto where Z_Name = "Kiel")**
  union (select K_Name from Kredit where Z_Name = "Kiel")
- **finde Namen und Städte aller Kunden, die in irgendeiner Zweigstelle einen Kredit aufgenommen haben:**
  select distinct Kunde.K_Name, Stadt from Kredit, Kunde
  where Kredit.K_Name = Kunde.K_Name
- **finde Namen und Städte aller Kunden, die in einer Zweigstelle in Kiel einen Kredit aufgenommen haben:**
  select distinct Kunde.K_Name, Stadt from Kredit, Kunde
  where Kredit.K_Name = Kunde.K_Name and Z_Name = "Kiel"
- **finde alle Kontonummern, deren Kontostand zwischen 5000 und 10000 DM liegt: select distinct Konto_Nr from Girokonto where**
  Kontostand between 5000.00 and 10000.00
  dies ist äquivalent zu: Kontostand >= 5000.00 and Kontostand <= 10000.00

Einfache Select-Beispiele - Fortsetzung

- **SQL erlaubt mit dem Prädikat like auch den Vergleich und die Suche von Zeichenketten mit Ähnlichkeitsoperatoren:**
  - **% ist ein Platzhalter für eine beliebige Zeichenkette**
  - **_ ist ein Platzhalter für ein einzelnes, beliebiges Zeichen**
  - **der Vergleich berücksichtigt Groß-Kleinschreibung**
  - **z.B. "Ha%" ist wahr beim Vergleich mit "Hannover" und "Hameln"**
  - **"Ha____" ist wahr beim Vergleich mit "Hameln"**
- **finde alle Kunden, deren Straßennamen mit B anfängt und genau fünf Zeichen umfaßt: select K_Name from Kunde where Strasse like "B____"**
- **finde alle Kunden wo der String "weg_" in der Straße nicht vorkommt: select K_Name from Kunde where Strasse not like "%weg_%" escape "\"**
- **Korrelationsvariablen sind Kurzbezeichnungen für Tabellen, die im from-Teil einer Select-Anweisung definiert werden; es gelten ähnliche Scoping-Regeln wie in anderen höheren Programmiersprachen**
  - **finde Namen und Städte aller Kunden, die in irgendeiner Zweigstelle einen Kredit aufgenommen haben: select distinct KU.K_Name, Stadt from Kredit KR, Kunde KU where KR.K_Name = KU.K_Name**
  - **finde die Namen aller Kunden, die ein Konto in einer Zweigstelle haben, an der Kunde Schmidt ein Konto hat: select distinct KU.K_Name from Girokonto KO, Girokonto KU where KO.K_Name = "Schmidt" and KO.Z_Name = KU.Z_Name**

SQL-Unterabfragen (subquery)

- **sind ein wichtiges SQL-Konstrukt; das Ergebnis kann von der übergeordneten Select-Abfrage weiterverarbeitet werden**
- **Unterabfragen können nach in, all, some und exists durchgeführt werden**
- **finde die Namen der Zweigstellen, deren Spar_Einl größer sind als die mindestens einer Zweigstelle in Kiel:**

```
select Z_Name from Zweigstelle where Spar_Einl > some
        (select Spar_Einl from Zweigstelle where Stadt = "Kiel")
```

- **finde die Namen der Zweigstellen, deren Spar_Einl größer sind als die aller Zweigstellen in Kiel:**

```
select Z_Name from Zweigstelle where Spar_Einl > all
        (select Spar_Einl from Zweigstelle  where Stadt = "Kiel")
```

- **finde die Namen der Kunden, die ein Girokonto, jedoch keinen Kredit in einer Zweigstelle in Kiel haben:**

```
select K_Name from Kunde where exists
(select * from Girokonto where Girokonto.K_Name = Kunde.K_Name
        and Z_Name = "Kiel")
and not exists
        (select * from Kredit where Kredit.K_Name = Kunde.K_Name
        and Z_Name = "Kiel")
```

Spaltenfunktionen

- **in der Select-Anweisung dürfen folgende Funktionen auf Mengen von Tupel, die durch group by gruppiert wurden, angewandt werden:**
  - **avg** Mittelwert einer Spalte
  - **count** Wertezahl einer Spalte oder aller Tupel einer Tabelle
  - **max** Größter Wert einer Spalte
  - **min** Kleinster Wert einer Spalte
  - **sum** Summe der Werte in einer Spalte
- **außer bei count werden Nullwerte ignoriert; Funktionen dürfen nicht geschachtelt werden, d.h. max (avg ( )) ist nicht zulässig!**
- **Beispiele ohne group by, d.h. eine Gruppe**
  - **Anzahl Kunden des Bankhauses: select count (*) from kunde**
  - **höchster Kontostand aller Girokonten: select max (Kontostand) from Girokonto**
  - **mittlerer Kontostand aller Girokonten: select avg (Kontostand) from Girokonto**
- **mittlerer Kontostand jeder Zweigstelle: select Z_Name, avg (Kontostand) from Girokonto group by Z_Name; Duplikate dürfen nicht entfernt werden!**
- **bestimme die Anzahl der unterschiedlichen Girokonto-Kunden einer Zweigstelle: select Z_Name, count (distinct K_Name) from Girokonto group by Z_Name; Duplikate müssen eliminiert werden!**

Beispiel mit group by und having

- **wir betrachten eine Ausprägung des Relationstyps Girokonto:**

K_Name	Konto_Nr	Z_Name	Kontostand
Susi	3456	Berlin	8000
Mops	4711	Kiel	3000
Klotz	4521	Berlin	6000
Klops	1234	Kiel	1100
Bolle	5634	Aachen	7000

- **es sollen alle Zweigstellen bestimmt werden, bei denen der mittlere Kontostand der Konten mit mindestens DM 1200,- mehr als DM 4000,- beträgt; die Ausgabe soll absteigend sortiert nach Z_Name erfolgen:**

select Z_Name, avg (Kontostand) from Girokonto where Kontostand > 1200 group by Z_Name having avg (Kontostand) > 4000 order by Z_Name desc

K_Name	Konto_Nr	Z_Name	Kontostand
Mops	4711	Kiel	3000
Susi	3456	Berlin	8000
Klotz	4521	Berlin	6000
Bolle	5634	Aachen	7000

- **group by gruppiert Tupel in dieser Tabelle auf Basis gleicher Z_Namen**
- **having wählt Gruppenzeilen, die der Bedingung avg (Kontostand) > 4000 genügen, d.h. die Gruppen 2 und 3 der obigen Tabelle; die Tupel der Spalte Z_Name werden absteigend sortiert ausgegeben:**

Z_Name	avg(Kontostand)
Berlin	**7000**
Aachen	**7000**

Beispiele mit group by und having (2)

- **finde die Zweigstelle(n) mit dem höchsten mittleren Kontostand:**

```
select Z_Name from Girokonto group by Z_Name
having avg (Kontostand) >= all
(select avg (Kontostand)
from Girokonto
group by Z_Name)
```

- **finde den mittleren Kontostand aller Girokonto-Kunden, die in Berlin leben und mindestens drei Girokonten haben:**

```
select avg (Kontostand) from Girokonto, Kunde
where Girokonto.K_Name = Kunde.K_Name and Stadt = "Berlin"
group by Girokonto.K_Name
having count (distinct Konto_Nr) >= 3
```

Anmerkung: distinct wird in diesem Fall nicht benötigt

Nullwerte in SQL 2

- **Nullwert (null) ist ein Extra-Symbol und gehört zu keinem Wertebereich**
- **sind Nullwerte in den Relationen enthalten, gelten folgende Regeln:**
  - **skalare Ausdrücke ergeben *null*, sobald ein Nullwert in die Berechnung eingeht**
  - **fast alle Vergleiche mit dem Nullwert ergeben den Wahrheitswert *unknown* statt *t* oder *f***
  - **Ausnahme: is null bzw. is not null ergeben bei Anwendung auf Nullwerte true bzw. false**
  - **boolesche Ausdrücke basieren auf dreiwertiger Logik mit den Wahrheitswerten *true*, *false* und *unknown* mit folgenden Wahrheitstabellen:**

and	true	false	unknown
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

not	
true	false
false	true
unknown	unknown

or	true	false	unknown
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

- **Beispiel: 3 Integer Attribute A,B,C haben die Werte: A=10, B=20, C=null**
  - **A < B or B < C ist true; A > B and B > C ist false**
  - **A > B or B > C ist unknown; not (B = C) ist unknown**

View

- **ein View definiert eine spezielle Datensicht auf eine Tabelle**
- **erst beim Zugriff auf die virtuelle View-Tabelle wird die View-Definition ausgeführt**
- **das Select im View unterliegt Einschränkungen**
- **create view viewname [(attributliste)] as select-statement**
  drop view viewname
- **Beispiel: Es soll ein View auf Girokunden definiert werden, deren Kontostand mehr als 100.000,- DM beträgt. Es sollen die Kunden der Zweigstellen in Kiel bestimmt werden:**

```
create view bigshots as
select Z_Name, K_Name, Konto_Nr from Girokonto
where Kontostand > 100000

select K_Name, Konto_Nr from Bigshots
where Z_Name = "Kiel"
```

Tabellendaten übertragen und Indexe

- **der insert-Befehl erlaubt auch die Übertragung von Daten über eine select Anweisung: insert into tabname [attributliste] select-block**
- **Beispiel: Allen Kreditkunden einer Zweigstelle in Kiel, deren Kreditbetrag höher als DM 50.000,- ist, sollen ein Sparkonto mit einem Guthaben von DM 100,- erhalten.**
- **es wird eine Relation mit dem Relationstyp Sparkonto definiert:**
  Sparkonto (K-Name, Konto-Nr, Z-Name, Kontostand)
- **insert into Sparkonto select K_Name, Kredit_Nr, Z_Name, 100**
  from Kredit where Z_Name = "Kiel" and Kreditbetrag > 50000
- **Indexe**
- **die Indizierung von Attributen dient der Zeitreduktion von select-Abfragen bzw. um Eindeutigkeit bestimmter Attribute sicherzustellen**
- **der Zusatz unique erlaubt keine doppelten Attributwerte**
- **create [unique] index indexname on tabname (attribut [asc | desc] [, attribut [asc | desc] ...];**
- **Index löschen: drop index indexname**

Datenkontrolle

- **die Systemadministration vergibt für die einzelnen Nutzer folgende Rechte:**
  - **Connect-Rechte**
  - **Resource-Rechte, d.h. Tabellen erstellen und löschen**
  - **DBA-Rechte, d.h. es dürfen auch nicht selbst angelegte Tabellen gelöscht werden**
- **System Authorities**
  - **grant { connect | resource | dba } to authorization-name [, ...] [identified by password]**
  - **revoke { connect | resource | dba } from authorization-name [, ...]**
  - **Beispiel: trouble_maker soll nicht mehr mit der Datenbank arbeiten: Revoke connect from trouble_maker**
- **um den Zugriff auf nicht selbst angelegte Tabelle zu ermöglichen, müssen Rechte definiert werden für: Select, Insert, Delete, Update und Index**
- **sie können vom DBA oder dem Nutzer, der die Tabellen ursprünglich angelegt hat, vergeben oder entzogen werden**
- **grant {privilegienliste | all} on [creator.]{tabname | viewname} to {namensliste | public } [with grant option]**
  Beispiel: Grant select, update (Name) on Kunde to trouble_maker
- **Zugriffsberechtigung entziehen: revoke {privilegienliste | all } on [creator.]{tabname | viewname} from {namensliste | public }**

Neuerungen von SQL-3 - Ausschnitt

- Objektrelationale Erweiterungen
  - **Neue Datentypen: BLOB, CLOB, Set, Listen, Arrays**
  - **UDT: User Defined Types**
  - **Typkonstruktoren: ROW, ARRAY, REF**
  - **Benutzerdefinierte Ordnungen für benutzerdefinierte Typen**
  - **Typisierte Tabellen mit Subtabellenbildung**
  - **Typisierte Sichten**
- **User-defined Routines (UDRs)**
  - **Können Teil eines UDT sein oder separat definiert werden**
  - **Kann in SQL oder extern in einer (unterstützten) Programmiersprache definiert werden**
  - **Wird mit einem SQL CALL Befehl aufgerufen**
  - **Kann 0+ Parameter aufweisen, wobei jeder Parametervom Typ IN, OUT or INOUT sein kann**
  - **an SQL-invoked function returns a value.**
  - **Specified params are input params with 1 designated as result param.**
  - **External routine defined by specifying an external clause that identifies 'compiled code' in operating system's file storage.**
  - **ORDBMS bietet Methoden wie Object-Codes dynamisch in das DBMS gelinkt werden können**
- **Zum jetzigen Zeitpunkt sind nur Teile des Standards implementiert!**

SQL-3 Datentypen

- **Basis-Datentypen in SQL-2: Number, Character, Date, ...**
- **Basis Datentypen in SQL-3 können per Definition erweitert werden: Video, Image, Audio, Text, Web-Pages, ...**
- **SQL-3 Row Types:**
  - **create table filiale (filial_nr varchar(3),**
    adresse row (strasse varchar(25), stadt varchar(20), plz varchar(5)):
  - **Die Spaltenattribute müssen also nicht mehr atomar sein!**
- **SQL-3 UDT**

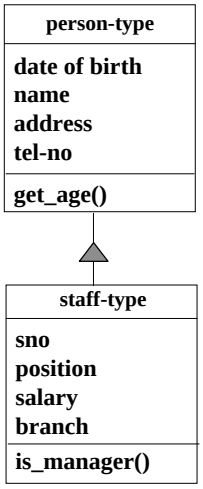
```
create type person_type as (
private
date_of_birth date,
public
name            varchar(15) not null,
address         varchar(50) not null,
tel_no          varchar(13) not null,
function get_age (p person_type) returns integer
/* code to calculate age from date_of_birth */
return
end)
not final;
```

Beispiel für eine "User-Defined Routine (UDR)"

SQL-3 Subtypes und Supertypes

- **Mit Sub-Types und Super-Types kann Vererbung realisiert werden**
- **CREATE TABLE** staff **OF STAFF_TYPE (PRIMARY KEY sno);**
- **CREATE TYPE staff_type UNDER person_type AS (**

```
sno   varchar(5) not null unique,
position varchar(10) not null,
Salary number(7,2),
bno varchar(3) not null,

create function is_manager
(s staff_type) returns  boolean
begin
if s.position = 'manager' then
    return true;
else
    return false;
end if
end)
not final;
```



SQL-3: LOBs

- **Large Object (LOB)**
  - **BLOB (Binary Large Object): Bilder / Audio, variable Länge bis 4GB**
  - **CLOB (Character Large Object): alphanum. Zeichenkette bis 4 GB**
  - **BFILE (Binary File): Loaktor auf externe Datei, Dateigröße bis zu 4GB**
- **Beispiel**

```
create directory mitarbeiter_bilder as 'c:\bilder\' ;

insert into mitarbeitertupeltabelle
values ( 'tom',
            empty_blob(),
            bfilename ('mitarbeiter_bilder', tom.jpg'),
            empty_clob(),
            bfilename ('mitarbeiter_bewerbung','tom.rtf')
);
```

- **Weitere wichtige Datenstrukturen**
  - **Listen: geordnete Menge mit Duplikaten**
  - **Eindimensionale Arrays mit fester Dimension**
  - **Set: ungeordnete Menge ohne Duplikate**
  - **Multiset: ungeordnete Menge mit Duplikaten**

7. Programmierung von Datenbank-Anwendungen

- **bei den meisten Anwendungssystemen ist es erforderlich, SQL-Anweisungen *aus* einer Programmiersprache auszuführen**
- **Eine Prozedur enthält also eine Mischung aus den Sprachelementen der Trägersprache (host language) und SQL**
- **Ziele sind u.a. Performance und weitgehende Datenbank-Unabhängigkeit**
- **die wichtigsten Möglicheiten sind: proprietäre APIs, ESQL, CLI**
- **proprietäre APIs**
  - **PC-RDBMS z.B. MS-Access erlauben die direkte Einbettung von SQL in Visual Basic / C#: GUI-Design, Applikationsteil, Datenhaltung werden integriert**
- **ESQL (Embedded SQL (statisch))**
  - **SQL-Anweisungen werden in eine unterstützte Programmiersprache eingebettet wobei seit 1992 eine standardisierte Syntax für den SQL-Teil existiert (SQL-92)**
  - **Ein Precompiler übersetzt die SQL-Konstrukte in die Syntax der Trägersprache (CLI)**
- **CLI (Call level interface)**
  - **Anwendungsprogramme sind vollständig in der Syntax der Trägersprache geschrieben**
  - **SQL-Anweisungen werden in *Zeichenketten* übergeben (kein Pre-Compiler erforderlich)**
  - **Ein DBMS spezifischer ODBC-Treiber verarbeitet die SQL-Anweisungen und übergibt sie zur Laufzeit an das RDBMS**
  - **Typische Vertreter sind ODBC (→) und JDBC (→)**

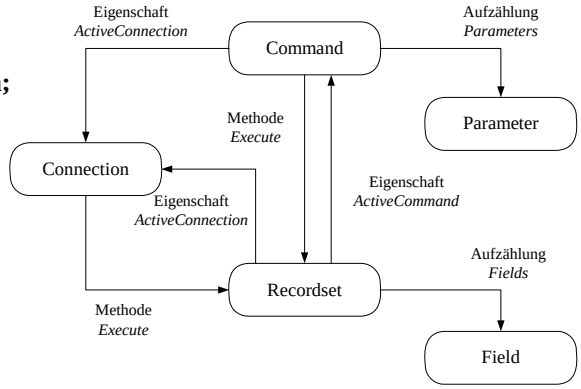
ODBC (Open DataBase Connectivity)

- **ist eine standardisierte Schnittstelle, die einen einheitlichen Zugriff auf verschiedene Datenbanksysteme ermöglicht**
- **Ein DBMS-spezifischer ODBC-Treiber verarbeitet SQL-Befehle, leitet sie an das DBMS weiter und liefert Daten sowie Fehlermeldungen zurück**

- **Client und DBMS können auf unterschiedlichen Systemplattformen laufen**
- **Unter Windows ist der ODBC-Treiber eine DLL, die für das DBMS unter Systemsteuerung → Verwaltung → Datenquellen ausgewählt werden kann**
- **ODBC ist gut geeignet wenn eine Anwendung für unterschiedliche DBMS konzipiert werden muss und höchste DB-Performance nicht kritisch ist**
- **bei Multi-Tier- ODBC-Treibern werden**
  - **die SQL-Anfragen nicht durch den Treiber bearbeitet, sondern vom DBMS**
  - **die Ergebnisdaten und ggf. Fehlermeldungen vom Server-DBMS zurückgeliefert**
  - **Kann der Austausch eines ODBC-Treibers durch einen des DBMS-Anbieters Performance-Gewinne bringen**

Microsoft Access ADO

- **MS Access**
  - **Benutzeroberfläche zum Zugriff auf Datenbank**
  - **Oberfläche ist getrennt von elementaren Datenbankfunktionen**
  - **Zugriff auf Datenbank erfolgt über Programmbibliothek: Jet-Engine**
  - **DAO (Data Access Objects): VB 3.0, für Jet-Datenbanken optimiert, über ODBC können auch fremde Datenbanksysteme angesprochen werden.**
  - **RDO (Remote Data Objects): VB 4.0 Enterprise. Ermöglicht effizienten Zugang zu externen Datenbank-Servern (Oracle, MS SQL-Server, ..).**
  - **DAO Erweiterung mit ODBCDirect: VB 5.0**
  - **ADO: Seit VB 6.0; einheitliches Objektmodell zum Zugriff auf jegliche Art von Datenbanksystemen; Löst DAO und ODBCDirect ab; basiert auf OLE DB, einer neuen Interface-Generation aufbauend auf COM (Component Object Model)**
  - **ADO.Net : jetzige MS Datenbank-zugriffstechnologie im Rahmen des .Net Frameworks**



Verwendung von ADO in Visual Basic

```
Dim con as New ADODB.Connection
Dim rec As ADODB.Recordset

con.Open "Provider=Microsoft.Jet.OLEDB.3.51;" & _
         "Data Source=" & App.Path & "\spieler.mdb"

Set rec = con.Execute("Select name, vorname
   from Spieler")
While Not rec.EOF
    List1.AddItem rec!vorname & " " & rec!Name
    rec.MoveNext
Wend
set rec = Nothing
set con = Nothing
```

Transaktionsverarbeitung mit ADO in VB

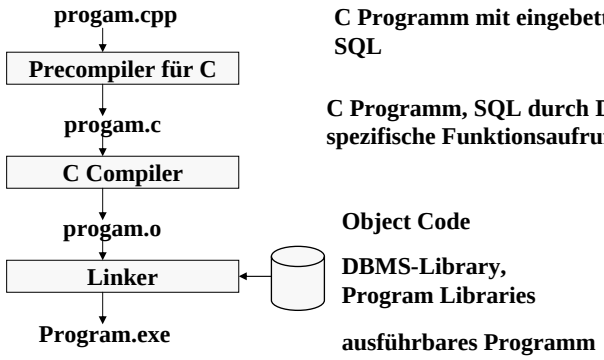
```
dim con as new ADODB.Connection
dim sql as string
On Error Goto Fehlerbehandlung
con.BeginTrans 'Transaktion einleiten
for i = 0 to anz
   sql = "insert into Verzeichnis (name, vorname) values " _
         & ('" & Trim$(TxtName(i).Text) & "', '" _
         & Trim$(TxtVorname(i).Text) & "')"
   con.Execute sql
next
con.CommitTrans 'Transaktion abschließen
exit sub

Fehlerbehandlung:
con.RollBack
```

Verwendung von ADO in ASP

```
<% option explicit
       dim con, rec, sql %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
   <title>ASP-Beispiel</title>
   <link rel=stylesheet type "text/css" href="../css/formate.css"
   <!--#include file="adovbs.inc"-->
</head>
<body>
<%
   set con = createobject("adodb.connection")
   set rec = createobject("adodb.recordset")
   sql = "select vorname, name from spieler;"
   con.open "spieler-dsn"
   rec.open sql, con %>
<select name="" size=10>
<%
   do while not rec.eof
      response.write "<option> "
      response.write rec.fields("vorname") & " " & rec.fields("name")
      response.write "</option>"
      rec.movenext
   loop %> </select>
</body></html>
```

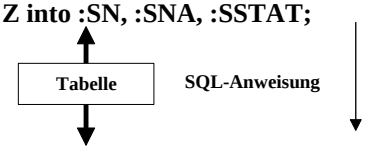
Embedded SQL

- **Precompiler übersetzt Programm mit ESQL in Trägersprachenprogramm**
- **durch komplexe Systemschnittstellen zwischen SQL und Trägersprache müssen Compiler, Linker und DBMS zusammenpassen**
- **Unterstützte Trägersprachen hängen vom RDBMS Anbieter ab**
- **ESQL enthält Konstrukte für**
  - **die Unterscheidung der SQL-Anweisungen von denen der Trägersprache**
  - **den Zugriff auf die Resultat-Tabelle eines SQL-Befehls (Cursor-Konzept)**
  - **die Fehlerbehandlung / Kommunikation mit dem RDBMS (SQLCA)**
- **Prinzip**

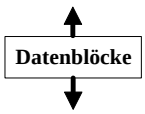


Programm mit SQL-Anweisungen - Prinzip

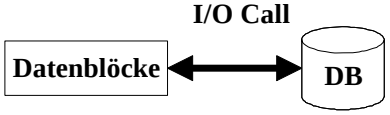
DCL normale Variablen (lokal, global);
Exec SQL DCL SQL-Tabellen;
Exec SQL Declare Z cursor for Select S#, SNAME, STATUS FROM S;
.....
Exec SQL Include SQLCA;
.....
Exec SQL fetch Z into :SN, :SNA, :SSTAT;
.....



- **DBMS**
  - **query processing (SQL)**
  - **buffer management, path optimization**
  - **compilation / interpretation (SQL)**



- **Betriebssystem / Datenmanagement**
  - **index, buffer management**
  - **space / device management**
  - **locking, recovery and logging**



Preprocessing für embedded SQL (ANSI-Standard)

- **die Ausführungen betreffen überwiegend Workstations und Mainframes**
- **eingebettete SQL-Anweisungen müssen mit *exec sql* beginnen; Beispiel: *exec sql delete from* Kunde *where* Name = "Mops“**
- **In Java (SQLJ) werden eingebettete SQL-Anweisungen wie folgt definiert: *#sql { SQL-Anweisung}* (SQLJ-Specs im ANSI-Standard)**
- **die Regeln der Trägersprache bestimmen die Stellen, an denen ausführbare SQL-Anweisungen erscheinen dürfen**
- **Variablen der Trägersprache die von SQL-Anweisungen benutzt werden (*host variables*, HV), müssen in einer Sektion definiert werden; Beginn und Ende lauten *exec sql begin declare* , *exec sql end declare***
- **Tabellen bzw. Views werden durch *exec sql declare* table-name table (...) definiert, wobei (...) mit den Angaben übereinstimmen muß, die bei der Einrichtung der Tabelle mit create table table-name (...) gemacht wurden**
- **die Ausführung einer SQL-Anweisungen setzt einen sqlcode, z.B : 0 ok, <0: Fehlermeldung, >0: Select liefert keine Daten (mehr)**
- **eine SQL-Abfrage liefert i.d.R. eine Tabelle, die i.d.R. von Trägersprachen als primitiver Datentyp nicht unterstützt wird; Ergebnisse können nur über einen Cursor sequentiell abgerufen und in HV gespeichert werden**

Host- und Indikatorvariablen

- **erhalten in den SQL-Anweisungen als Prefix einen Doppelpunkt, um sie von den SQL-Variablen zu unterscheiden**
- **Namen von Host- und SQL-Variablen können, müssen aber nicht gleich sein**
- **HV müssen in Vergleichen und Zuweisungen den gleichen Datentyp wie SQL-Variablen aufweisen**
- **Beispiel: *exec sql select * from* Kunde *where* K_Name = :Kunden_Name**
- **Schema der Deklaration von Hostvariablen in der Trägersprache:**

  exec sql begin declare section

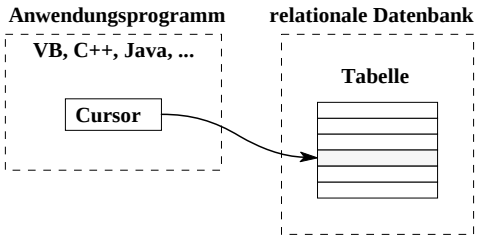
  (deklaration der Hostvariablen)

  exec sql end declare section
- **bei Mainframe RDBMS existiert das Konzept der Indikatorvariablen (IV)**
- **eine IV korrespondiert mit genau einer HV (Input- oder Output-IV)**
- **IV werden zum Lesen und ´Scheiben von Nullwerten und bei Fehlern gesetzt**
- **ist z.B. die Übertragung in die HV unvollständig, da die HV nicht groß genug ist, so stellt der Wert der IV die Anzahl der übertragenen Zeichen dar**
- **Beispiel: Eintragung eines neuen Kunden ohne den Stadtnamen: ind = -1**

  insert into Kunde values :K_Name, :strasse, :stadt:ind

SQL Cursormanagement

- **bei der Ausführung einer Select-Abfrage kann als Ergebnis kein Tupel, genau ein Tupel oder mehrere Tupel geliefert werden**
- **ist sicher, daß es nur höchstens ein Ergebnistupel gibt, dann können die Werte dieses Tupels direkt in HV gespeichert werden; Beispiel:**
  ***exec sql select* Stadt *into* :stadt:ind *from* Kunde *where* K_Name = "K"**
- **können mehrere Ergebnistupel entstehen, muß das Cursorkonzept angewendet werden**
- **ein SQL Cursor zeigt auf die jeweils aktuelle Tabellenzeile und ermöglicht einen zeilenweisen, sequentiellen Zugriff**
- **die Deklaration *declare name cursor for* assoziiert eine Abfrage mit einem Cursor; der Cursorname muss im gesamten Modul eindeutig sein**
- **eine open-Anweisung positioniert den Cursor vor die erste Zeile der Tabelle**
- **mit jedem fetch wird der Zeiger aktualisiert und überträgt die aktuelle Zeile in die HV; das Ende wird durch sqlcode > 0 signalisiert**



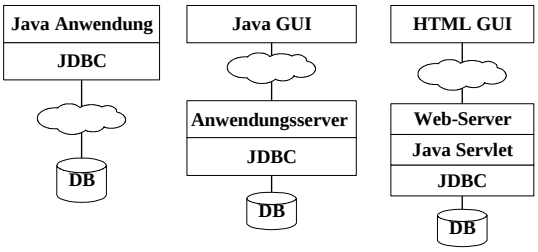
Cursor Beispiel

- **im folgenden Beispiel wird PL/I als Trägersprache verwendet:**

```
exec sql whenever sqlerror goto error
exec sql declare c1 cursor for
        select pnr, name, brutto
        from person
        where kost = :kostnr
        order by name;
exec sql open c1
exec sql fetch c1 into :pnr, :name, :brutto;
Do while (sqlcode = 0);
        display (pnr, name, brutto);
        exec sql fetch c1 into :pnr, :name, :brutto;
end do;
display ('end of list');
exec sql close c1;
return
error:
display 'fetch Fehler des Cursors c1! sqlcode: ', sqlcode
end
```

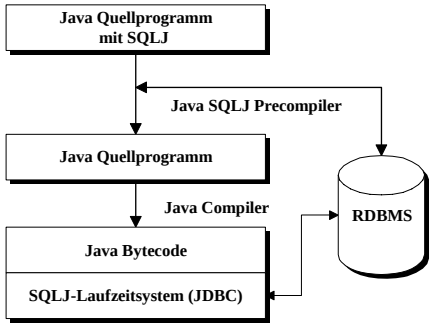
JDBC / SQLJ

- **steht für Java Database Connectivity - ein Java API für DBMS**
- **Mit JDBC hat Sun Microsystems einen Standard definiert, um aus Java-Programmen auf relationale Datenbanksysteme zugreifen zu können**
- **JDBC ist eine analoge Schnittstelle wie ODBC, die jedoch speziell für Java konzipiert wurde, um deren Plattformunabhängigkeit zu gewährleisten**
- **Merkmale von JDBC**
  - **JDBC bietet Objekte, Klassen und Methoden – eine semantisch viel höhere Ebene als ODBC**
  - **Ein DBMS spezifischer JDBC-Treiber ist in Java geschrieben und daher portabel**
  - **daneben gibt es JDBC-ODBC Middleware, die auf einem ODBC-Treiber aufbauen**
  - **im Gegensatz zu ESQL wird eine SQL-Ergebnistabelle als Instanz der Klasse ResultSet zur Verfügung gestellt; diese Klasse besitzt Methoden zum Durchlaufen der Tabelle und zum Zugriff auf die Ergebnisse**
  - **der Datenbankzugriff kann über Java-Applets, Java-Serveranwendung oder über Java-Servlets erfolgen; letztere liefern nur HTML-Code an den Browser zurück; Web-Server muss das Servlet-API unterstützen**
  - **SQL-Anweisungen werden grundsätzlich als Zeichenketten übergeben und werden vom Java-Compiler nicht syntaktisch überprüft (Unterschied zu SQLJ)**



SQLJ

- **Ist ein ESQL speziell für für Java und ein ANSI-Standard**
- **das SQLJ-Laufzeitsystem benutzt i.d.R. die JDBC-Schnittstelle (gleiche Treiber)**
- **SQLJ Merkmale**
  - **Precompiler erzeugt ein reines Java-Programm mit Calls zum SQLJ-Laufzeitsystem**
  - **Precompiler überprüft die SQL-Syntax und die Typkompatibilität zwische Java-Hostvariablen und SQL-Variablen**
  - **optionale Online-Prüfung der SQL-Anweisungen mit den Datenbanktabellen**
  - **daher besseres Fehlerhandling als bei JDBC**
  - **SQLJ unterstützt nur statisches SQL**
  - **Bessere Zugriffsoptimierung möglich**
- **Die Vorteile von Java – Plattformun-abhängigkeit, Objektorientierung bleiben auch hier bestehen**



Stored Procedures und Triggers (SPT)

- sind datenbezogene Programmroutinen, die vom DBMS auf dem Server durch Triggers transaktionsorientiert angestoßen werden
- auch Einbettung in prozeduralem Code (embedded stored procedures)
- können vorkompilert sein, d.h. die SQL-Anweisungen sind analysiert und Zugriffsoptimierungen bereits erfolgt
- Anwendungen: Datendefinitionen, Prüfungen, Integritätsregeln, Sequenz von zusammenhängenden DML-Anweisungen (Reduktion der Netzlast)
- SQL-Zugriffe über ein LAN sind relativ langsam; greifen 100 PCs im LAN nur über SQL-Befehle auf einen DB-Server zu, gibt es Engpässe

Vor- und Nachteile

\+ Performance-Verbesserungen (tps), typisch um den Faktor zwei im LAN, wenn SPT statt SQL-Kommandos auf dem Server ausgeführt werden

\+ zentrale Serverspeicherung ergibt Vorteile bei neuen Anwendungen und Softwarepflege, da SPT nur zentral entwickelt bzw. geändert werden müssen

\+ verbesserte Endbenutzerfähigkeiten, da z.B. mit Tabellenkalkulation Daten über SPT in die Datenbank geprüft übernommen werden können

\- nicht standardisiert, d.h. Einsatz von SPT führen zur völligen Abhängigkeit

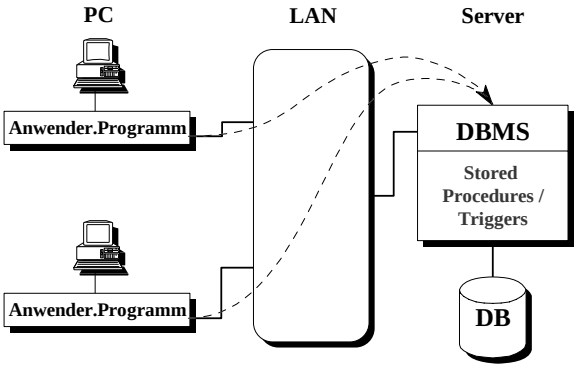
- Ausweg: Java, C, C++ mit ESQL, Aufruf über RPCs

Verlagerung von Anwendungslogik in das DBMS

- Beispiel (syntaktische Grundgerüst):

```
Create Trigger Bestell_Zaehler
on insertion on Buch_Bestellung best:
update Kunde
set Anz_Best = Anz_Best + 1
where K_Nr = New best.K_Nr
```

- Der Zeitpunkt der Trigger-Aktivierung kann genau spezifiziert werden
- Immediate / deferred
- For each row
- Before / after (Oracle) steuert ob der Trigger vor oder nach einer Änderung aktiviert wird



Transaktionssysteme

- **Auskunfts- und Buchungssysteme, integrierten Informationssysteme**
- **Dialogverarbeitung (Online Transaction Processing (OLTP)) mit kurzen Antwortzeiten und permanente Systemverfügbarkeit sind extrem wichtig, da die Systeme in die betrieblichen Abläufe integriert sind**
- **vier kennzeichnende Eigenschaften beschreiben die Ausführung einer Transaktion (TA), die unter dem Begriff (ACID) bekannt sind:**
  1. ***Atomarität*, d.h. eine TA muß entweder vollständig oder gar nicht ausgeführt werden**
  2. ***Konsistenz*, d.h. vor und nach Ausführung einer Transaktion befindet sich die Datenbank in einem konsistenten Systemzustand, der alle Integritätsbedingungen erfüllt**
  3. ***Isolation*: trotz Mehrbenutzerbetrieb wird jede TA vollständig eigenständig abgearbeitet (logischer Einbenutzerbetrieb)**
  4. ***Dauerhaftigkeit*, d.h. erfolgreich beendete TAs sind permanent gespeichert und „überleben" erwartete Fehler (Abstürze, Rechnerausfälle) durch Logging- und Recovery-Maßnahmen**
- **Hauptkomponenten eines TA-Systems: DBS, Transaktionsmonitor (TP-Monitor), Netzwerk, Protokolle und Transaktionsprogramme**
- **der TP-Monitor steuert die Ausführung der TA-Programme und die Kommunikation mit Terminals und DBMS; z.B CICS (IBM, OS/390)**

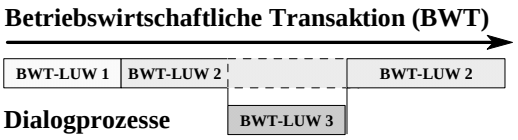
Transaktionsmanagement und Datensicherheit

- **unter Transaktionsmanagement versteht man alle Maßnahmen zur korrekten Ausführung bzw. Rückgängigmachung von Transaktionen**
- **eine Datenbanktransaktion ist eine logische Arbeitseinheit (logical unit of work, LUW), die entweder vollständig oder überhaupt nicht ausgeführt werden soll**
- **sämtliche Veränderungen der Datenbank sind in einer LUW zunächst nur temporär**
- **SQL bietet durch die Operationen *Commit work* und *Rollback work* Einfluß auf das Transaktionsmanagement**
- **Commit work signalisiert dem System die erfolgreiche Beendigung einer Transaktion und die Änderungen werden in die Datenbank übernommen**
- **sollen temporäre Änderungen nicht übernommen werden, dann wird Rollback work ausgeführt**

Commit work	Zustand n - 1
Insert ... Delete ... Update ...	DB-Transaktion (LUW)
Commit work oder Rollback work	⇒ Zustand n ⇒ Zustand n - 1

Unterschiedliche Transaktionsbegriffe

- **Betriebswirtschaftliche Transaktionen (BWT) sind funktional zusammengehörige Verarbeitungseinheiten, die konsistente, betriebswirtschaftlich zulässige Datenbankänderungen durchführen, z.B. Soll- und Haben-Buchung**
- **BWT betreffen i.d.R. mehrere Relationen; diese müssen bis zum Abschluß der BWT - mit entsprechenden Sperren - verwaltet werden**
  - **z.B. müssen beim Anlegen eines Fertigungsauftrages Stücklisten mit Materialpreisen, Material-Nummern, Arbeitspläne mit Kosten kopiert und temporär gesperrt werden**
- **eine BWT wird als Folge logisch zusammengehöriger Dialogschritte implementiert; ein Dialogschritt umfaßt die Dateneingabe, das Eingabeformular und das Abschicken des Datenpaketes und der resultierenden Antwort**
- **der Begriff BWT Logical Unit of Work (BWT-LUW) ist die Gesamtheit aller Dialogschritte einer BWT einschließlich aller Datenbankänderungen**
- **Dialogschritte können asynchron in eigenen Prozessen ausgeführt werden**
- **auch eine BWT-LUW muß vollständig oder gar nicht ausgeführt werden**
- **RDBMS kennen keine *prozeßübergreifenden Transaktionsabläufe***
- **aus Sicht des RDBMS gehört zu jedem Dialogschritt eine eigene DB-LUW in der Datenbankänderungen erfolgen (SQL-Commit -Rollback)**



Sperrprotokolle für DB-Transaktionen

- **Problem: verzahnte Abarbeitung von Transaktionen (lost update)**

Transaktion t1	Transaktion t2	Kontostand #4711
		1000,-
Read #4711		
	Read #4711	
kto#4711 +300		
	kto#4711 -100	
Write #4711		1300,-
	Write #4711	900,-

- **Sperren reservieren Zugriff auf ein Datenelement für eine TA und sperren Zugriffe anderer Transaktionen aus**
- **Granularität der Sperren: Tabellen, Zeilen, Felder, Bedingungen (where)**
- **serieller Schedule := TAs laufen streng nacheinander ab**
- **serialisierbarer Schedule := Schedule ist äquivalent zu seriellem Schedule**
- **Ergebnisäquivalenz := beide Schedules hinterlassen die Datenbank im selben Datenzustand: ist praktisch im voraus nicht feststellbar**
- **Deadlock: gegenseitiges Warten auf Entsperrung von TAs**
  - **T1 fordert exclusive Sperre für x an und bekommt sie,**
  - **T2 fordert exclusive Sperre für y an und bekommt sie**
  - **T1 fordert exclusive Sperre für y an und wartet, T2 fordert exclusive Sperre für x an und wartet**
- **Auflösung: Erstellung der Abhängigkeiten und Rollback einer beteiligten TA**

8 Relationales Datenbankdesign 2

- *Verlustfreie Zerlegungen*
  - Fertigung (<u>A-Nr</u>, <u>Farbe</u>, <u>M-Nr</u>) beschreibt Artikel, die in bestimmten Farben auf bestimmten Maschinen produziert werden
  - eine Ausprägung von Fertigung sei:

A-Nr	Farbe	M-Nr
112	weiß	3
111	weiß	5
111	rot	3

  - eine Zerlegung von Fertigung durch Projektionen sei:
    R1 (<u>A-Nr</u>, <u>M-Nr</u>) und
    R2 (<u>A-Nr</u>, <u>Farbe</u>)

A-Nr	M-Nr
112	3
111	5
111	3

A-Nr	Farbe
112	weiß
111	weiß
111	rot

  - auf welchen Maschinen kann Artikel 111 in weiß gefertigt werden ? Der natürliche Verbund von R1 und R2 über A-Nr liefert Fertigung-Falsch

A-Nr	Farbe	M-Nr
112	weiß	3
111	weiß	5
111	weiß	3
111	rot	5
111	rot	3

  - die entstehende Relation enthält also zusätzliche Tupel
- allgemein gilt: wird eine Relation R über Projektionen zerlegt und anschließend eine Relation RZ durch natürlichen Verbund der entsprechenden Attribute erzeugt, so gilt nur $R \subseteq RZ$
- Eine verlustfreie Zerlegung ist sehr wichtig, da Daten verloren gehen können
- Frage: unter welchen Bedingungen ist eine Zerlegung verlustfrei?

Abhängigkeitstreue Zerlegungen

- wir betrachten SFD (<u>M-Nr</u>, <u>Fach</u>, Dozent)
- die funktionalen Abhängigkeiten von SFD sind:
  1. (M-Nr, Fach) $\Rightarrow$ Dozent
  2. Dozent $\rightarrow$ Fach
- Schlüsselkandidaten sind (M-Nr, Fach) und (M-Nr, Dozent)
- SFD ist nicht in BCNF, da Dozent kein Schlüsselkandidat ist.
- die Zerlegung in SD (<u>M-Nr</u>, <u>Dozent</u>) und DF (<u>Dozent</u>, Fach) ergibt zwar Relationstypen in BCNF, diese erfüllen jedoch nicht die funktionale Abhängigkeit Nr. 1.!
- Welche Konsequenzen ergeben sich? Wir betrachten dazu ein Beispiel:

SFD

M-Nr	Fach	Dozent
4711	BWL	Kruschwitz
4711	Wirtschaftsinformatik	Lenz
4700	Wirtschaftsinformatik	Suhl

SD

M-Nr	Dozent
4711	Kruschwitz
4711	Lenz
4700	Suhl

DF

Dozent	Fach
Kruschwitz	BWL
Lenz	Wirtschaftsinformatik
Suhl	Wirtschaftsinformatik

- die Zerlegungen sind offensichtlich verlustfrei; das Tupel (4711, Suhl) könnte in SD ohne Probleme eingetragen werden, ist jedoch nicht zulässig!
- keine andere Zerlegung von SFD ist abhängigkeitstreu

Vergleich Boyce-Codd Normalform und 3NF

für das Datenbankdesign der Basisrelationen werden folgende Ziele gefordert :

- **alle Relationstypen sind in der BCNF**
- **die Zerlegungen sind verlustfrei, d.h. können über natürliche Verbund-operationen rückgängig gemacht werden**
- **die Zerlegungen erfüllen alle bestehenden funktionalen Abhängigkeiten der Ausgangstypen**
- **man kann mathematisch beweisen (Ullman, 188ff):**
  - **jeder RT läßt sich verlustfrei in BCNF zerlegen**
  - **jeder RT läßt sich verlustfrei in 3NF zerlegen und die Zerlegungen erfüllen alle bestehenden funktionellen Abhängigkeiten der Ausgangstypen**
  - **es gibt RT für die es keine BCNF-Zerlegung gibt, die abhängigkeitstreu sind**
- **gibt es keine Zerlegung in BCNF mit diesen Eigenschaften, dann muß man RT in 3NF akzeptieren**
- **bei Zielkonflikten zwischen BCNF und Erfüllung aller bestehenden funktionalen Abhängigkeiten, muß im Einzelfall abwägen, ob 3NF oder BCNF bevorzugt wird**
- **ggf. müssen als Kompromiß Datenredundanzen bzw. Nullwerte für Attribute in Kauf genommen werden; (geht bei Schlüsselattributen aber nicht!)**

Mehrwertige Abhängigkeiten von Attributen (1)

- **bei der funktionalen Abhängigkeit eines Attributes Y von einem Attribut X eines Relationstyps RT bestimmt ein Wert von X genau einen Wert von Y**
- **es gibt Fälle, wo ein Wert von X mehrere Werte von Y definiert und diese Tupel unabhängig von anderen Attributen sind**
- **Bsp.: der RT Kinder (<u>P#</u>, <u>K#</u>) gibt an, welche Kinder eine Person hat; es gilt:**
  - **Die Anzahl der Kinder einer bestimmten Person ist eine natürliche Zahl (mit 0)**
  - **Hat Person kein Kind, so gibt es kein Tupel in Kinder mit der P# dieser Person**
  - **Hat eine Person n Kinder (n > 0) so treten in Kinder n Tupel dieser Person auf**
- **Definition: Es sei R (X, Y, ...) ein RT mit mindestens zwei Attributen; Attribut Y ist *mehrwertig abhängig* (mwa) vom Attribut X, wenn jeder Attributwert x von X eine nur von x abhängige Menge von Attributwerten von Y impliziert; Schreibweise: R.X ↠ R.Y**
- **Eine mehrwertige Abhängigkeit (mwA) heißt *trivial*, wenn eine mwA existiert und RT entweder nur zwei Attributen X, Y hat oder wenn Y Teil von X ist**
- **Liegt ein Relationstyp mit trivialen mwA vor, so lässt sich der RT nicht weiter zerlegen; Kinder ist also maximal normalisiert**
- **Hätte Kinder ein weiteres nicht mwa Attribut Z, z.B. Name, so wäre Kinder nur in 1NF, sollte also zerlegt werden und es liegt eine trivial mwA vor**
- **Fertigung (<u>A-Nr</u>, <u>Farbe</u>, <u>M-Nr</u>) hat keine mwA; warum ?**

Mehrwertige Abhängigkeiten von Attributen (2)

- **Probleme mit mwA gibt es dann, wenn *mehrere unabhängige mehrwertige Abhängigkeiten* in einem Relationstyp auftreten**
- **Beispiel: der RT Lehrangebot (LVS, Dozent, Buch) beschreibt:**
  - **Lehrveranstaltungen (LVS), Dozenten und Literaturquellen**
  - **eine LVS kann von verschiedenen Dozenten durchgeführt werden**
  - **für eine LVS können mehrere Bücher benutzt werden**
  - **ein Dozent kann mehrere LVS lehren, ein Buch kann für mehrere LVS benutzt werden**
  - **Lehrangebot ist in BCNF; jede LVS impliziert die möglichen Dozenten und Bücher; es gibt keine nichttrivialen funktionalen Abhängigkeiten**
  - **Dennoch bestehen offensichtliche Redundanzen**
- **Es sei R (X, Y, Z) ein RT mit mindestens drei Attributen und mindestens zwei unabhängigen mwA X ↠ Y und X ↠ Z; dann gilt:**
  - **die Menge der Werte des Attributes Y, die zu einem Wertepaar von (X,Z) gehört, ist nur vom Wert des Attributes X, jedoch nicht vom Wert des Attributes Z abhängig**
  - **die Menge der Werte des Attributes Z, die zu einem Wertepaar von (X,Y) gehört, ist nur vom Wert des Attributes X, jedoch nicht vom Wert des Attributes Y abhängig**
  - **Formal: zu zwei Tupeln (x,y,z) und (x,y',z') der Relation vom Typ R müssen auch die Tupel (x, y', z) und (x, y, z') Bestandteil der Relation sein**
- **Zerlegt man R in R1 (X, Y) und R2 (X, Z), so entfallen die Probleme, da die mwA dann trivial sind**

Definition der vierten Normalform

Ein Relationstyp R befindet sich in der 4NF, wenn:

- **er sich in BCNF befindet**
- **außer funktionalen nur *triviale mehrwertigen Abhängigkeiten* existieren**
- **Beispiel: Autor-Schlagwort wird zerlegt in: Autor (Sig-Nr, Autor) und Schlagwort (Sig-Nr, Schlagwort)**

Autor

Sig-Nr	Autor
X1	Korth
X1	Silber
X2	Denert
X3	Balzert

Schlagwort

Sig-Nr	Schlagw
X1	RDBMS
X2	Systementwick.
X2	Datenmodellier.
X3	Systementwick.

- **Bücher (Sig-Nr, Autor, Schlagwort, Titel, Sach-Nr, Sachgebiet) wurde also in folgende Relationstypen in 4NF überführt:**
  - **Buch (Sig-Nr, Titel, Sach-Nr), Sachgebiet (Sach-Nr, Sachgebiet)**
  - **Autor (Sig-Nr, Autor), Schlagwort (Sig-Nr, Schlagwort)**
- **Lehrangebot (LVS, Dozent, Buch) ist in BCNF, jedoch nicht in 4NF**
- **es gibt folgende mwA: LVS ↠ Dozent, LVS ↠ Buch**
- **die Zerlegung Lehrer (LVS, Dozent), VST (LVS, Buch) führt zu RT in 4NF**
- **Fertigung (A-Nr, Farbe, M-Nr) ist also in 4 NF, da keine mwA existieren**

Vierte Normalform – anderes Beispiel

- **Wir betrachten den RT MPQ1 (<u>M-Nr</u>, <u>P-Nr</u>, <u>Q</u>)**
  - **der beschreibt welche Qualifikation(en) (Q) ein Mitarbeiter (M-Nr) *haben muss*, um in einem bestimmten Projekt (P-Nr) mitarbeiten zu können**
  - **Z.B. werden für Projekt 3 (P-Nr = 3) Qualifikationen A und B gefordert**
  - **MPQ1 besitzt keine nichttrivialen funktionalen Abhängigkeiten**
  - **MPQ1 besitzt aber nichttriviale mwA P-Nr↠Q und P-Nr↠M-Nr und ist *nur in BCNF***
  - **MPQ1 lässt sich verlustfrei zerlegen in R1=(<u>M-Nr</u>, <u>P-Nr</u>) und R2=(<u>P-Nr</u>, <u>Q</u>)**
  - **MPQ1 lässt sich auch verlustfrei zerlegen in R1, R2 und R3=(<u>M-Nr</u>, <u>Q</u>)**
  - **Die RT R1-R3 sind alle in 4NF**

MPQ1		
M-Nr	**P-Nr**	**Q**
101	3	A
101	3	B
101	4	A
101	4	C
102	3	A
102	3	B
103	5	D

R1	
M-Nr	**P-Nr**
101	3
101	4
102	3
103	5

R2	
P-Nr	**Q**
3	A
3	B
4	A
4	C
5	D

R3	
M-Nr	**Q**
101	A
101	B
101	C
102	A
102	B
103	D

  - **Die Zerlegung von MPQ1 in R1 und R3 ist jedoch nicht verlustfrei wie man unmittelbar sieht, wenn man deren natürlichen Verbund über M-Nr bildet**

Project-Join-Abhängigkeiten

- **Es gibt Relationstypen in 4NF mit Redundanz- und damit Update-Anomalien**
- **Zu Beginn dieses Kapitels wurde die Frage gestellt, wann *eine* Zerlegung verlustfrei ist**
- **Nun wird die Frage betrachtet wann ein RT R(<u>A</u>,<u>B</u>,<u>C</u>) *prinzipiell* in *drei* RT R1(<u>A</u>,<u>B</u>), R2(<u>B</u>,<u>C</u>) und R3(<u>C</u>,<u>A</u>) verlustfrei zerlegt werden kann**
- **das obige Theorem hilft dabei nicht, denn dort werden nur zwei RT betrachtet**
- **damit solche Zerlegungen zeitinvariant möglich sind, müssen offenbar folgende Bedingungen gelten:**
  - **ist (a1,b1) in R1 ∧ (b1,c1) in R2 ∧ (c1,a1) in R3 ⇒ (a1,b1,c1) ist in R**
  - **äquivalent ist: aus (a1,b1,c2), (a2,b1,c1), (a1,b2,c1) in R ⇒ das Triplet (a1,b1,c1) ist in R**
  - **Dies gilt weil (a1,b1) dann und nur dann in R1 auftritt, wenn es ein c2 aus C gibt, sodass (a1,b1,c2) in RT ist; analoges gilt für die zwei anderen Tupel**
  - ***Diese Bedingung muss zeitinvariant gelten und heißt Join-Abhängigkeit***
  - ***Sie folgt nicht aus funktionalen oder mehrwertigen Abhängigkeiten des RT!***
  - **Wenn ein RT aber zeitinvariant verlustfrei zerlegt werden kann und damit die obige Join-Abhängigkeit gilt, so weist der RT prinzipiell Redundanzen auf**

Project-Join-Abhängigkeiten (2)

- **Welche inhaltlichen Konsequenzen stecken hinter Join-Abhängigkeiten?**
- **Wir betrachten folgenden RT Liefern1 (Lieferant, Teil, Projekt)**
- **Er beschreibt welche Teile ein Lieferant für ein bestimmtes Projekt liefert**
- **Annahme: Liefern1 ist in 4NF, d.h. es gibt keine mwA, d.h. ein Lieferant liefert nicht alle Teile und alle Projekte**
- **eine Ausprägung von Liefern1 sei**

Liefern1		
Lieferant	**Teil**	**Projekt**
Schmidt	Bolzen	X
Schmidt	Schrauben	Y
Winkler	Schrauben	Z
Adam	Bolzen	Y
Adam	Nägel	X

- **Die Join-Bedingung (a1,b1,c2), (a2,b1,c1), (a1,b2,c1) in R ⇒ das Triplet (a1,b1,c1) ist in R gilt offenbar nicht, denn man hat:**
  - **da (Schmidt,Bolzen,X) ∧ (Adam,Bolzen,Y) ∧ (Schmidt,Schrauben,Y) in Liefern1 sind, müsste (Schmidt,Bolzen,Y) in Liefern sein und das ist nicht der Fall**
  - **Würde die Join-Bedingung zeitinvariant gelten, so müssten die kursiv markierten Tupel in Liefern 2 vorhanden sein**
  - **Also kann Liefern1 nicht verlustfrei in drei RT zerlegt werden, da es mindestens eine Instanz gibt, wo dies nicht möglich ist**

Liefern2		
Lieferant	**Teil**	**Projekt**
Schmidt	Bolzen	X
Schmidt	Schrauben	Y
Winkler	Schrauben	Z
Adam	Bolzen	Y
Adam	Nägel	X
Adam	*Bolzen*	*X*
Schmidt	*Bolzen*	*Y*

Fünfte Normalform (Project-Join-Normalform)

- **Ein RT ist in 5NF, wenn er in 4NF ist und es *keine* verlustfreie Zerlegung in kleinere RT mit unterschiedlichen Kandidatenschlüsseln gibt**
- **Während die 4NF auf mehrwertigen Abhängigkeiten basiert, ist also bei der 5NF das *Fehlen* von *verlustfreien Zerlegungen* Kern der Definition**
  - **äquivalent ist, dass der RT nur triviale Join-Abhängigkeiten aufweist**
  - **Beispiel: Der RT MPQ2 beschreibt die *gemeinsamen* Qualifikationen von Mitarbeitern die sie haben, mit denen, die in einem Projekt gefordert werden um dort zu arbeiten**
  - **MPQ2 ist weniger restriktiv als MPQ1, weil ein MA in einem Projekt arbeiten kann ohne alle im Projekt geforderten Qualifikationen zu erfüllen**
  - **MPQ2 ist in 4NF; MPQ2 kann verlustfrei in die 3 RT R1-R3 zerlegt werden; MPQ2 weist also Redundanzen auf und ist daher nicht in 5NF**

R1	
M-Nr	**P-Nr**
101	3
101	4
102	3
103	3
103	4
103	5

R2	
M-Nr	**Q**
101	A
101	B
102	A
102	B
103	A
103	C

R3	
P-Nr	**Q**
3	A
3	B
4	A
4	B
5	A
5	C

MPQ2		
M-Nr	**P-Nr**	**Q**
101	3	A
101	3	B
101	4	A
101	4	B
102	3	A
102	3	B
103	3	A
103	4	A
103	5	A
103	5	C

Fünfte Normalform (2)

- **Wir betrachten eine *inhaltliche Variante MPQ3* des RT MPQ2:**
  - **MPQ3 erfasst die Mitarbeiter in Projekten mit den Qualifikationen die sie für die betrachteten Projekte benötigen**
  - **MPQ3 hat keine mwA und keine join-Abhängigkeiten; wir betrachten die RT R1-R3 die aus den Projektionen entstehen und die nat. Joins R4 = R1 x R2 und R5 = R4 x R3**

MPQ3		
M-Nr	P-Nr	Q
101	3	A
101	3	B
101	4	A
101	4	C
102	3	A
102	3	B
102	4	A
102	4	B

R2	
P-Nr	Q
3	A
3	B
4	A
4	B
4	C

R1	
M-Nr	P-Nr
101	3
101	4
102	3
102	4

R3	
M-Nr	Q
101	A
101	B
101	C
102	A
102	B

R4		
M-Nr	P-Nr	Q
101	3	A
101	3	B
101	4	A
101	4	B
101	4	C
102	3	A
102	3	B
102	4	A
102	4	B
102	4	C

R5		
M-Nr	P-Nr	Q
101	3	A
101	3	B
101	4	A
101	4	B
101	4	C
102	3	A
102	3	B
102	4	A
102	4	B

  - **Das Tupel (101,4,B) ist aber nicht in MPQ3!**
  - **Wie man sich an Hand des Beispiels klar machen kann, gibt es auch keine andere verlustfreie Zerlegung; daher ist MPQ3 in 5NF**
  - **Die vermeintlich sehr ähnlichen RT MPQ1-MPQ3 weisen also einen unterschiedlichen Normalisierungsgrad auf der z.T. auf PJ-Abhängigkeiten basiert**

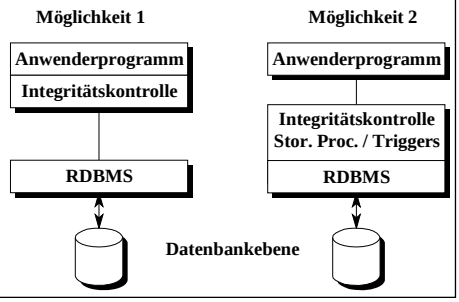
RT	NF	PJ (2-fach)	PJ (3-fach)	mwA
MPQ1	BCNF	j	j	2
MPQ2	4NF	n	j	0
MPQ3	5NF	n	n	0

Fünfte Normalform (3)

- ***5NF-Normalisierung (ohne Beweis)*: ist ein RT nicht in 5NF, so gibt es eine Zerlegung in kleinere RT in 5NF deren Verbund wieder den RT ergibt**
- **Man kann beweisen, dass ein RT in 5NF auch in 4NF ist**
- **RT in 5NF sind also bzgl. Redundanzen minimal**
- **Gegeben sei der RT Person (<u>P#</u>, Name, Strasse, PLZ)**
  - **Er beschreibt eine Person, die genau einen Wohnort hat; Person ist in 4 NF;**
  - **Person lässt sich verlustfrei zerlegen in: R1 (<u>P#</u>, Name), R2 (<u>P#</u>, Strasse), R3 (<u>P#</u>, PLZ)**
  - **Es existiert keine weitere verlustfreie Zerlegung. Person ist in 5 NF, weil alle Kandidatenschlüssel gleich sind**
- **Fertigung (<u>A-Nr</u>, <u>Farbe</u>, <u>M-Nr</u>) ist in 4NF und ließ sich nicht verlustfrei zerlegen in R1 und R2; der Verbund R1 x R2 mit R3 (<u>M-Nr</u>, <u>Farbe</u>) über (Farbe, M-Nr) ist aber verlustfrei; daher ist Fertigung *nicht in 5NF***
- **Gegeben ist ein RT Plan (<u>A-Nr</u>, <u>M-Nr</u>, Dauer), der angibt wie viel ZE die Herstellung eines Artikels auf einer bestimmten Maschine dauert; es gilt:**
  - **Eine Maschine kann nicht jeden Artikel fertigen und ein Artikel wird nicht auf allen Maschinen gefertigt**
  - **Dauer hängt voll funktional vom einzigen Kandidatenschlüssel in Plan ab**
  - **Plan lässt sich nicht verlustfrei in kleinere Relationstypen zerlegen**
  - **Plan ist also in 5NF**

Datenintegrität

- **physische Datenintegrität, d.h. korrekt gespeicherte Daten**
- **semantische Datenintegrität, d.h. korrekte Umsetzung in ein ERM**
- **strukturelle Datenintegrität, d.h. die korrekte Repräsentation im Relationenmodell (Normalisierung, Abhängigkeitstreue usw.)**
- **operationale Datenintegrität, d.h. die Einhaltung von Integritätsregeln im Rahmen der Transaktionsverarbeitung**
- **komplexe Integritätsregeln können über Trigger und Stored Procedures implementiert werden**
- **jede SQL-Operation wird dann auf ihre Auswirkung bzgl. der Integritätsregeln überprüft mit der Folge, dass die Operation ggf. nicht ausgeführt wird oder weitere Aktionen vom DBMS veranlasst werden**
- **für Trigger u. Stored Proc. spricht, dass sie im Datenbankkatalog gespeichert sind und daher für jede Anwendung bindend sind**



Datenintegritätsarten

1. **Objektintegrität: jedes Datenobjekt wird eindeutig identifiziert**
2. **Datentypintegrität: Einschränkung des Wertebereichs eines Attributes**
3. **referentielle Integrität**
4. **allgemeine statische und transitionale Integritätsbedingungen**

- **Erläuterung an einem Beispiel:**
  - **Mitarbeiter (<u>Persnr</u>, Name, Urlaubsanspruch, Abteilung)**
  - **Abteilung (<u>Bezeichnung</u>, AnzMitarbeiter); Abteilung ist Fremdschlüssel in Mitarbeiter**
- **Datentypintegrität**
  - **Der Urlaubsanspruch eines Mitarbeiters liegt zwischen 20 und 30 Tagen: $\forall u \in$ Urlaubsanspruch $\Rightarrow (u \geq 20 \wedge u \leq 30)$**
  - **Oracle 8: create table mitarbeiter (...constraint check (urlaubsanspruch >= 20 and urlaubsanspruch <= 30) ...)**
  - **SQL-2: create domain uanspruch integer check value >= 20 and value <= 30 create table mitarbeiter (... Urlaubsanspruch uanspruch ...)**
- **Referentielle Integrität: für jedes Tupel der Relation Mitarbeiter muss ein entsprechendes Tupel in der Relation Abteilung existieren:**
  - $m \in$ Mitarbeiter $\exists a \in$ Abteilung mit m.Abteilung=a.Bezeichnung
  - create table Abteilung (... primary key (Bezeichnung) ...)
  - create table Mitarbeiter (... foreign key (Abteilung) references Abteilung ...)

Integritätsbedingungen

- **Statische Integritätsbedingungen**
  - **haben die Form: $t \in R \Rightarrow B(t_1, t_2, ...t_n)$, t_i sind Tupelvariablen**
  - **z.B. der Wert von AnzMitarbeiter jedes Tupels von Abteilung soll die Summe der entsprechenden Tupelwerte von Mitarbeiter sein**
  - **Oracle 8:**

```
create trigger statib after insert or update on abteilung as
if exists (select * from abteilung a where a.anzahlmitarbeiter !=
   (select count(*) from mitarbeiter where abteilung = a.bezeichnung))
   then raise error
endif
```

- **Transitionale Integritätsbedingungen**
  - **Beispiel: $\forall m \in$ Mitarbeiter: m.Urlaubsanspruch.old $\leq$ m.Urlaubsanspruch.new**
  - **Oracle 8:**

```
create trigger transib after update on mitarbeiter for each row as
if :old.urlaubsanspruch > :new.urlaubsanspruch then raise error
endif
```

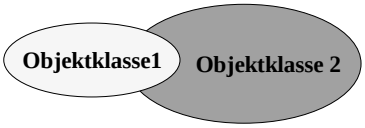
- **SQL-2 unterstützt weder Trigger noch ein Old/New-Konzept**
- **daher bestehen keine direkten Möglichkeiten transitionale Integritätsbedingungen zu formulieren**

Datenbankschema

- **Abschluß der Phase Implementation -Entwurf ist ein Datenbankschema**
- **dieses Schema ist die Basis für die Einrichtung der Datenbank mit SQL im Ziel-Datenbanksystem**
- **im Datenbankschema wird spezifiziert:**
- **Aufstellung aller Relationstypen in hinreichend normalisierter Form (Basisrelationen); dies beinhaltet:**
  - **Namen der Relationstypen**
  - **Namen der Attribute mit Definition der primitiven Datentypen**
  - **Festlegung der Primärschlüssel, Aufzählung der Kandidatenschlüssel**
- **Feststellung aller funktionalen und mehrwertigen Abhängigkeiten**
- **Feststellung interrelationaler Abhängigkeiten (Fremdschlüssel, referentielle Integrität)**
- **Feststellung weiterer Integritätsbedingungen, die in Form von logischen Bedingungen oder algorithmisch spezifiziert werden, z.B. transitionale Integritätsbedingungen**

9. Relationale Modellierungsprobleme

- **Modellierung überlappender Objektklassen**
  - **Objekte, die zu mehr als einer Objektklasse gehören, erfordern besondere Beachtung, um Inkonsistenzen zu vermeiden**
  - **Beispiel: In einem Unternehmen sind Mitarbeiter als Instruktoren oder Entwickler tätig. Jeder Instruktor lehrt in genau einem Fachgebiet. Die Entwickler sind für genau ein Projekt verantwortlich. Einige Entwickler sind auch als Instruktoren tätig.**
  - **eine Modellierungsmöglichkeit wäre:**
    Entwickler (Pers-Nr, Name, Projekt-Nr)
    Instruktoren (Pers-Nr, Name, Fach-Code)



  - **obwohl beide Relationen in BCNF sind, gibt es Redundanzen**
- **die Lösung besteht in der Einführung einer übergeordneten Objektklasse Mitarbeiter (Pers-Nr, Name), die alle globalen Daten enthält und zwei Relationstypen für die lokalen Daten:**
  - **Entwickler (Pers-Nr, Projekt-Nr)**
  - **Instruktor (Pers-Nr, Fach-Code)**

Modellierung von Preisstaffeln

- **häufig findet man mengenabhängige Preisstaffeln, die sowohl vom Lieferanten, als auch vom Artikel abhängen**
- **zur Modellierung dieser Problemstellung modelliert man:**
  Lieferant-Artikel (L-Nr, A-Nr, Ab-Menge, Preis)
- **für eine Bestellmenge x des Artikels a vom Lieferanten l wird der zugehörige Preis wie folgt bestimmt:**

```
select *
from Lieferant-Artikel
where L-nr = l and A-Nr = a
and Ab-Menge =
                (select max (Ab-Menge)
                 from Lieferant-Artikel
                 where x >= Ab-Menge
                 and L-Nr = l
                 and A-Nr = a)
```

Modellierung von Belegungsproblemen

- **die Belegung eines Objektes (z.B. Räume, Maschinen, Werbetafeln usw.) in einzelnen Zeiteinheiten (z.B. Stunden, Tagen, Kalenderwochen) kann wie folgt modelliert werden:**
- **Objekt (O-Nr, ...), Belegung (O-Nr, Z-Nr)**
- **sollen alle Objekte, die im Intervall von, bis nicht belegt sind bestimmt werden, so könnte man folgende select-Abfrage formulieren:**
  select * from Objekt
  where O-Nr not in (select distinct O-Nr from Belegung where Z-Nr >= von and Z-Nr <= bis)
- **diese Lösung erfordert bei langen Belegungsintervallen viel Speicherplatz; eine Alternativlösung sieht folgendermaßen aus:**

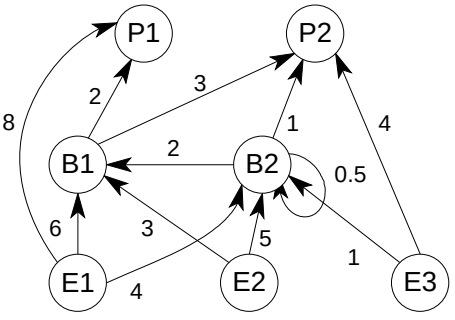
  Belegung (O-Nr, von-Z-Nr, bis-Z-Nr)

  select * from Objekt where O-Nr not in
  (select O-Nr from Belegung where bis-Z-Nr >= von and von-Z-Nr <= bis)

- **statt der diskreten Zeit-Nummern können auch Datumsangaben verwendet werden**

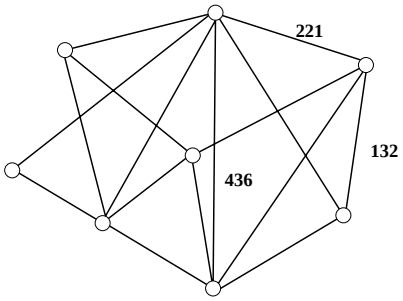
Modellierung von Strukturstücklisten

- **Strukturstückliste: Zusammensetzung von Produkten aus ihren einzelnen Komponenten (Baugruppen, Einzelteilen usw.)**
- **Stücklistenrepräsentation: Graph, Matrixnotation, relational**
- **Speicherung als Listenstrukturen im Hauptspeicher oder relational auf Festplatte**
- **redundanzfreie Speicherung im Gozintographen G = (V, E, B)**
  - **V: Knotenmenge, alle Komponenten,**
  - **E: Kantenmenge, geordnete Paare (a,b) mit a,b ∈ V, wenn Teil a direkt in Teil b eingeht;**
  - **B((a,b)) gibt an, wieviel Einheiten von a für ein Teil von b benötigt werden**
- **Modellierung mit zwei Relationstypen:**
  Teile (Teile-Nr, Bezeichnung, ...)
  Struktur (Oteil-Nr, Uteil-Nr, Anzahl, ...)
- **allgemein können gerichtete Graphen ohne Mehrfachkanten in dieser Form repräsentiert werden**



Modellierung von symmetrischen Straßennetzen

- **Asymmetrische Straßennetze können wie Stücklisten modelliert werden (gerichteter Graph)**
- **Die Entfernung zwischen zwei Knoten wird durch Kantenbewertung definiert**
- **Symmetrische Straßennetze werden durch ungerichtete Graphen repräsentiert**



- **Es gibt hier folgende Möglichkeiten:**
  - **als unsymmetrischer Graph, wobei der Entfernungswert der Tupel (von, nach) und (nach, von) identisch ist; durch Datenredundanz kann es hier zu Konsistenzproblemen kommen**
  - **es werden nur solche Paare (von, nach) gespeichert, für die *von* lexikographisch kleiner als *nach* ist**
  - **soll für ein Paar (von, nach) die Entfernung bestimmt werden, so wird vor der Suche überprüft, ob von lexikographisch kleiner als nach ist. Ist dies nicht der Fall, so wird nach dem Paar (nach, von) gesucht**

Maschinenbelegungsplan

- **ein Maschinenbelegungsplan (MBP#) beinhaltet die zeitliche Sequenz (S#), in der Teile (T#) auf einer bestimmten Maschine (M#) an einem bestimmten Fertigungsdatum gefertigt werden sollen.**
- **ein Teil kann in einem MBP mehrfach gefertigt werden**
- **an einem Tag können mehrere MBP für eine Maschine existieren**
- **Die Fertigungsdauer eines Teils ist konstant und hängt auch von der Maschine ab auf der es bearbeitet wird; Beispiel:**

Maschine				Maschinenbelegungsplan			
M#	**MBP#**	**F.datum**		**MBP#**	**T#**	**S#**	**F.Dauer**
1	21340	1.4.01		21340	303	1	10
1	21341	1.4.01		21340	404	2	2
2	21342	1.4.01		21340	303	3	10
				21341	505	1	15
				21342	303	1	11

- **Aufgaben:**
  - **Welche Relationstypen werden zur Repräsentation benötigt ?**
  - **Welche Kandidatenschlüssel existieren ?**
  - **Welche funktionalen Abhängigkeiten existieren ?**
  - **In welcher höchsten Normalform befinden sich die RT ?**

Lösung

- **Maschine (<u>MBP#</u>, M#, Fertigungsdatum)**
  - **MBP# → M# (ein Maschinenbelegungsplan wird für eine bestimmte Maschine erstellt)**
  - **MBP# → Fertigungsdatum**
- **Maschinenbelegungsplan (<u>MBP#</u>, <u>S#</u>, T#, Fertigungsdauer)**
  - **Jedes Teil hat eine eindeutig definierte Fertigungsdauer, die auch von der Maschine abhängt: (M#, T#) ⇒ Fertigungsdauer; wegen (MBP# → M# und (M#, T#) ⇒ Fertigungsdauer) gilt: (MBP#, T#) ⇒ Fertigungsdauer.**
  - **Ein Maschinenbelegungsplan (MBP#) beinhaltet die zeitliche Sequenz (S#), in der Teile (T#) …: (MBP#, S#) ⇒ T# ; wegen (MBP#, T#) ⇒ Fertigungsdauer und (MBP#, S#) ⇒ T#) gilt: (MBP#, S#) ⇒ Fertigungsdauer**
  - **Einziger Kandidatenschlüssel: (MBP#, S#)**
  - **Schlüsselattribute: MBP#, S#**
  - **Nichtschlüsselattribute: T#, Fertigungsdauer**
  - **Determinanten: (MBP#, S#), (MBP#, T#)**
- **2. NF ist gegeben, da jedes NSA voll funktional vom Kandidatenschlüssel abhängt.**
- **3. NF ist gegeben, da keine transitiven Abhängigkeiten existieren (Fertigungsdauer wird nicht alleine durch T# bestimmt).**
- **BCNF ist nicht gegeben, da die Determinante (MBP#, T#) kein Kandidatenschlüssel ist**

Historisierung von Daten im Relationenmodell

- **Speichern zeitlicher Veränderungen von Attributwerten mit dem entsprechenden Änderungsdatum in einem Datenbestand**
- **z.B. die Gehaltsentwicklung einer MitarbeiterIn in einer Personaldatenbank**
- **erste Lösungsvariante - zwei Relationen**
  - **die aktuellen Daten werden in einem RT Mitarbeiter gehalten**
  - **die zeitlichen Veränderungen werden in einer RT Mitarbeiter-Hist gehalten:**
  - **Mitarbeiter (<u>P-Nr</u>, AB-Datum, Name, J-Gehalt,...)**
  - **Mitarbeiter-Hist (<u>P-Nr</u>, <u>AB-Datum</u>, J-Gehalt,..)**
- **Ermittlung der zum Datum x gültigen Daten der MitarbeiterIn P1:**

Select * from Mitarbeiter-Hist where P-Nr = P1 and AB-Datum = (Select Max (AB-Datum) from Mitarbeiter-Hist where X >= AB-Datum and P-Nr = P1)

- **Vor- und Nachteile dieser Lösung:**
  - **Trennung aktueller von historischen Daten**
  - **schnelle Abfragen auf aktuelle Daten**
  - **Redundanzen sind schwer vermeidbar**
  - **Komplexe Abfragen auf historische Daten**

Zweite Lösungsvariante - eine Relation

- **bei dieser Lösung werden aktuelle und historische Daten in einem RT gehalten: Mitarbeiter (P-Nr, AB-Datum, Name, J-Gehalt,...)**
- **der Zugriff auf aktuelle Daten wird über einen View erleichtert:**

  Create View Mitarbeiter-Aktuell

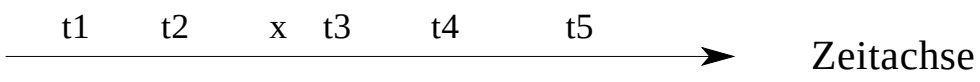
  as select P-Nr, Name, Max (AB-Datum)

  from Mitarbeiter group by P-Nr, Name
- **auf aktuelle Daten kann wie folgt zugegriffen werden:**

  Select * from Mitarbeiter-Aktuell where P-Nr = P1
- **Vor- und Nachteile dieser Lösung**
  - **nur eine Relation, keine Verschiebung von Tupeln erforderlich**
  - **der Relationstyp ist nicht in 2NF**
  - **langsamerer Zugriff auf aktuelle Daten**
  - **Primärschlüssel muß mit dem AB-Datum versehen werden**
  - **Problem, wenn der Primärschlüssel auch Fremdschlüssel ist**

Modifizierte erste Lösungsvariante - zwei Relationen

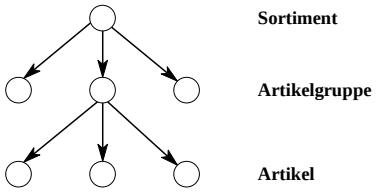
- **eine Vereinfachung von Abfragen auf historische Daten ergibt sich durch Aufnahme des BIS-Datums in Mitarbeiter-Hist:**
  Mitarbeiter-Hist (P-Nr, AB-Datum, BIS-Datum, Name, J-Gehalt,..)



- **Ermittlung der bis Datum X gültigen Daten der MitarbeiterIn P1:**
  Select * from Mitarbeiter-Hist where P-Nr = P1
  and X between AB-Datum and BIS-Datum
- **in einer Relation werden die aktuellen Daten gehalten**
- **die zweite Relation enthält pro Entität null oder mehrere Tupel mit vergangenen oder zukünftigen Attributwerten**
- **es muß sichergestellt werden, daß die Tupel entsprechend ihrem Aktualitätsgrad in der richtigen Relation gehalten werden, d.h. eine Tupelverschiebung kann notwendig werden**

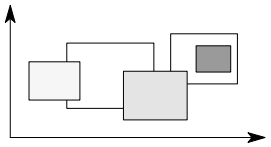
Modellierung von Artikel und Artikelmengen

- **Vielfach findet man ein Artikelsortiment von mehreren tausend Artikel, die in einer Indexmenge I = {1,…,n} zusammengefasst werden**
- **Die Artikel-Nr ist typisch eine alphanumerische Zeichenkette fester Länge**
- **Artikel (A-Index, Artikel-Nr, A-Bezeichnung,…)**
- **Artikelgruppen sind fest definierte Teilmengen aus dem Artikelsortiment, die sich auch überlappen können**
- **Jede Artikelgruppe wird durch eine Indexmenge I_g beschrieben, $I_g \subseteq I$, $g \in G$**
- **Artikelgruppe (AG-Index, AG-Nr, AG-Bezeichnung,…)**
- **Mehrere Artikelgruppen können noch zu Sortimenten S_j zusammengefasst werden**
- **Wie werden Artikelgruppen und Sortimente relational modelliert ?**
- **AG (AG-Index, A-Index)**
- **Sortiment (S-Index, S-Nr, S-Bezeichnung,…)**
- **SG (S-Index, AG-Index)**



Geodatenproblem Rechteck

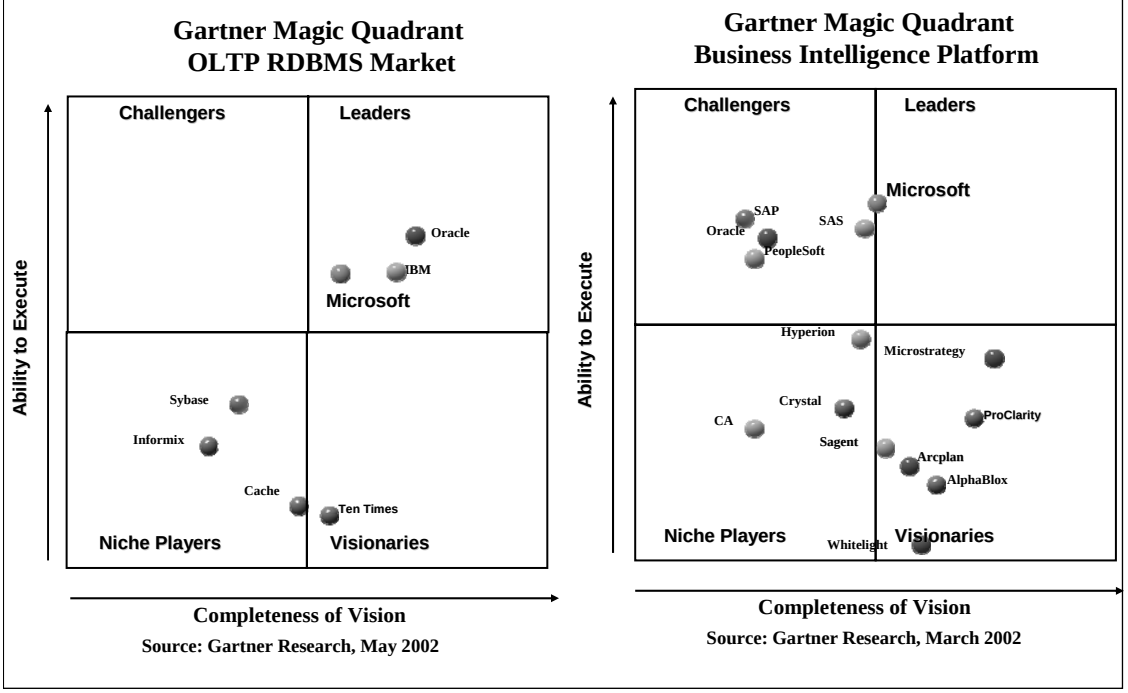
- **Ein RT Rechteck (x1, y1, x2, y2) beschreibt Rechtecke in der Ebene R^2**
- **Jedes Rechteck wird durch zwei Koordinatenpaare (x1,y1) für die linke untere Ecke und (x2,y2) für die rechte obere Ecke definiert**
- **Ein Rechteck ist nichtleer, d.h. es gilt $x1 \leq x2$ und $y1 \leq y2$**
- **Gegeben ist ein Rechteck R = (x1r, y1r, x2r, y2r); formulieren Sie eine SQL-Abfrage, die alle Rechtecke aus dem Relationstyp Rechteck bestimmt, die sich mit R auch teilweise überlappen.**
- **Lösung: ?**



10 Grundlagen physischer Datenspeicherung

- **basiert auf Speicherungsform u. Zugriffsmethode von Dateien**
- **Speicherungsform:**
  - **Art der physischen Dateispeicherung: sequentielle, indexsequentielle, relative Dateien**
  - **Speicherungsform einer Datei wird bei ihrer Erstellung festgelegt und ist unveränderbar**
- **Zugriffsmethode: *wie* auf die Datensätze zugegriffen wird:**
  - **sequentieller bzw. logisch fortlaufender Zugriff auf Datensätze**
  - **wahlfreier (direkter) Zugriff auf einen Datensatz**
- **typisch sind folgende Dateioperationen:**
  - **Einfügen und Löschen von Datensätzen**
  - **Modifikation von Feldinhalten eines Datensatzes**
  - **Suche nach Datensätzen mit bestimmten Feldinhalten**
- **Merkmale von Dateiverarbeitungen**
  - **ein Datensatz muß bei heutigen Rechnerarchitekturen grundsätzlich erst in den Hauptspeicher gelesen werden, um seine Daten zu bearbeiten oder auszugeben**
  - **I/O ist extrem langsam im Vergleich zu Operationen im Hauptspeicher; zeitkritische Programme sollten daher so entworfen werden, daß minimales I/O erforderlich ist**
  - **zur Durchsatzsteigerung bei Transaktionsverarbeitung werden große Halbleitercaches mit intelligentem Datenmanagement eingesetzt (Zugriffszeit < 3 ms)**
  - **eine Read/Write-Anweisung eines Programms kann je nach Situation null oder mehrere physische I/O-Operationen benötigen**

Business Intelligence und OLTP RDBMS Plattformen



Leistungsaspekte bei Datenbankankwendungen

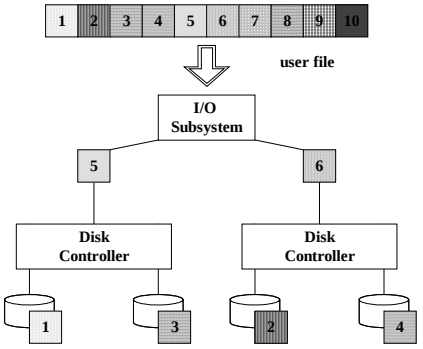
- **Betreffen folgende prinzipielle Bereiche**
  - **Auswahl des geeigneten Datenbanksystems (selten entscheidungsrelevant)**
  - **Anwendungsprogramme mit SQL**
    - **Schlecht strukturierte Relationstypen**
    - **Ineffiziente SQL-Anweisungen bei alternativen Konstrukten**
    - **Datenbanktransaktionen die zu erhöhtem „Locking“ und damit „Warten“ führen**
  - **Speichersubsysteme**
    - **sind als zentrale Plattform von großer Bedeutung**
    - **I/O ist extrem langsam (DBMS I/O: 25ms), daher sollten möglichst viele Datenzugriffe aus dem Hauptspeicher erfolgen (Faktor > 200) → Hauptspeicheradressierung**
  - **Systemplattform**
    - **Prozessoren: 32 Bit / 64 Bit, SMP kann bei modernen DBMS zu großen Leistungssteigerungen führen, TPS-Benchmarks**
    - **Betriebssystem: Engpass ist die Hauptspeicheradressierung z.B. bei XP (2 GB)**
    - **Viele Betriebssystem bieten bereits 64 Bit-Adressierung (z.B. MS SQL Server 2003), dadurch können wesentlich mehr Daten im HS gehalten werden (Daten Cache)**
  - **Netzwerk**
    - ***Effektive* Datenübertragungsrate im LAN / WAN**
    - **Einfluss der Netzwerkkarten mit denen der Datenbankserver am Netz verbunden ist**

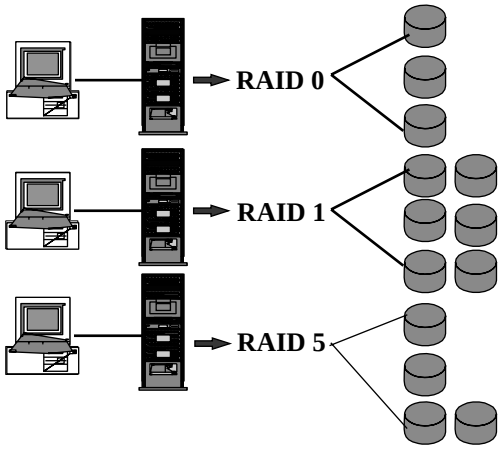
Zeitkomponenten einer DML-Anweisung

- **I/O-Zeit**
  - **Calls der DBMS-Software zum Auffinden der Daten (Indexseiten und Datenseiten)**
  - **Zurückschreiben der veränderten Daten (Update von Indexseiten),**
  - **Logging, Speichermanagement**
- **CPU-Zeit**
  - **Parsen von DML-Anweisungen**
  - **DML-Interpretation / Kompilation / Zugriffsoptimierung**
  - **Vergleiche, Sortieren, Join-Operationen, Duplikateliminierung**
  - **Übertragen von Daten in logische Tabellen der Anwenderprogramme**
- **Datenübertragungszeit im LAN (Mbit /sec)**
- **Einflußfaktoren**
  - **logisches und physisches DB-Design, Systemplattformen, Netzbandbreite**
  - **Art der DML-Anweisungen und Zugriffsoptimierung des DBS**
  - **Größe der Datenbank, Treffermenge (Anz. Datensätze) einer DML-Anweisung:**
  - **Fähigkeit des DBMS zur Wiederverwendung bereits optimierter DML-Anweisungen**
- **Zielsetzungen**
  - **Minimierung der Anzahl physischer I/O-Operationen (Hauptfaktor)**
  - **Minimierung der CPU-intensiven Prozesse (z.B. Join-Operationen, Sortierung, Elimination von Duplikaten)**

Raid Speichersysteme

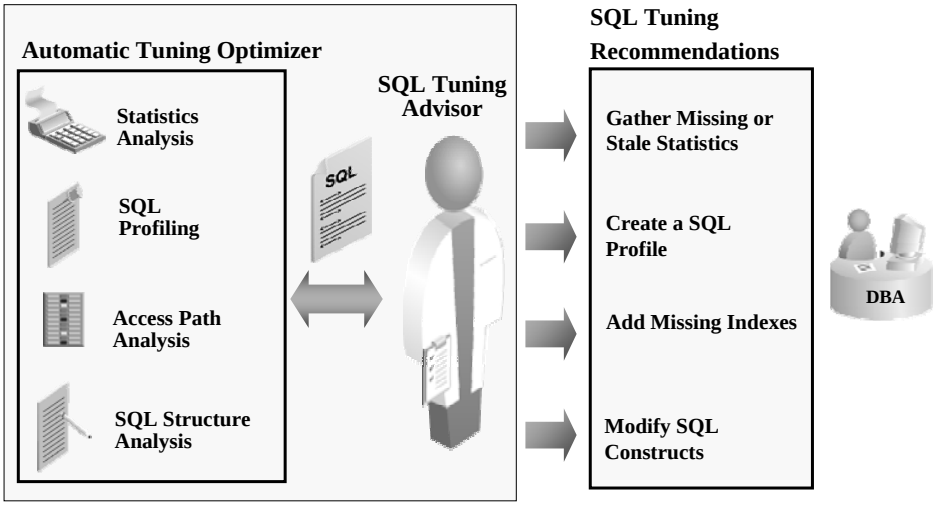
- **SCSI-Festplatten mit Standardformaten bilden eine logische Einheit;**
- **Raid-Merkmale**
  - **RAID 0: nur Data Striping, keine erhöhte Datensicherheit, höchste I/O-Leistung**
  - **RAID 1: nur Datenspiegelung, höchste Datensicherheit, langsameres schreiben als bei RAID 0, schnelleres schreiben als bei RAID 5, höchster Platzbedarf**
  - **RAID 5: Data Striping mit XOR-Wert, gute Datensicherheit, weniger Platzbedarf als bei RAID 1, mehr als bei RAID 0, langsamstes Schreiben**





Oracle Automatic Tuning Analyzer

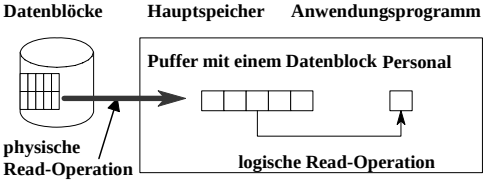
- Ist ein Query Optimizer der im Tuning Modus läuft
  - **Benutzt die gleichen Algorithmen mit zusätzlichen Analysen, die viel Zeit kosten**
- Experimentiert automatisch mit alternativen Zugriffsmethoden
  - **Testet alternative Indexe zur Zugriffsoptimierung**
  - **Analysiert SQL-Befehle die zu zeitintensiven Zugriffsverfahren führen**



Elementare Speicherungsformen

Sequentielle Speicherung und Zugriff

- **Datensätze werden in Datenblöcken nacheinander sortiert oder unsortiert gespeichert**
- **der Zugriff kann nur in der gespeicherten Reihenfolge erfolgen, jedoch nicht direkt**
- **Zu- oder Abgänge von Datensätzen erfordern im Normalfall ein Neuschreiben der Datei**
- **Veränderungen des Inhalts der Datensätze (Updates) können direkt (Rewrite) vorgenommen werden, wenn sich die Datei auf Festplatte befindet**
- **durch die gegebene Verarbeitungsreihenfolge ist das Blocken von Datensätzen sinnvoll: dabei werden eine feste Anzahl von Datensätzen in einem Datenblock zusammen gefaßt**
- **die Verarbeitungszeit der gesamten Datei wird vorwiegend durch die *Anzahl der physischen* I/O-Operationen bestimmt, die durch einen höheren Blockungsfaktor reduziert wird**
- **Beispiel:**



COBOL-Programmebene

```
01 Personal.
02 Name pic x(60).
02 Bruttogehalt pic 9(12)V99.
```

Read Master into Personal. (Logische Read-Operation)

VM/CMS-Ebene

Filedef Master Disk Person Daten C (lrecl 80 block 400

Speicherungsformen für direkten Zugriff

Relative Dateien

- **jeder Datensatz wird durch eine relative Satz-Nr. *rsn* eindeutig identifiziert, die die Position des Datensatzes relativ zum Dateianfang festlegt**
- **die *rsn* erlaubt mit *einer* physischen I/O-Operation Zugriff auf den Datensatz**
- **ist nur dann sinnvoll anwendbar, wenn eine *natürliche Zahl als Primärschlüssel* benutzt werden kann und es keine großen Lücken im Zahlenbereich des Primärschlüssels gibt**

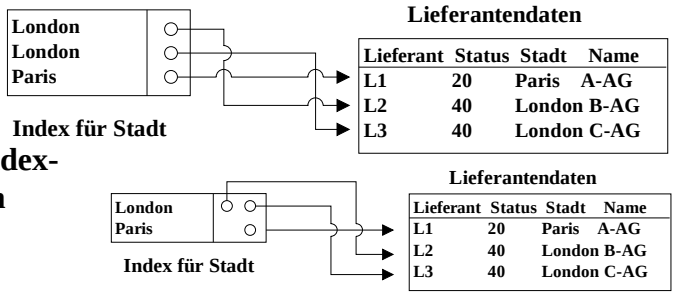
Indizierte Dateien

- **erlauben sowohl den direkten als auch *logisch fortlaufenden* Datenzugriff**
- **im Unterschied zu relativen Dateien können Schlüssel eine beliebige Struktur aufweisen**
- **im folgenden Beispiel werden die Attribute Mitglieds-Nr (nr) und Name indiziert**
- **Mitglieds-Nr ist eindeutig und wird als Primärschlüssel verwandt**
- **Name ist nicht eindeutig (Option *duplicates)***

```
file-control.
    select optional sport assign to "c:\sport.dat", organization is indexed,
    access mode is dynamic, record key is nr, alternate key is name with duplicates.
    data division.
    file section.
    fd sport.
    01 satz.
       05 nr       pic 999.
       05 name     pic x(40).
       05 adresse.
    start sportdatei key = name, invalid key perform nicht-da,
    end-start.
```

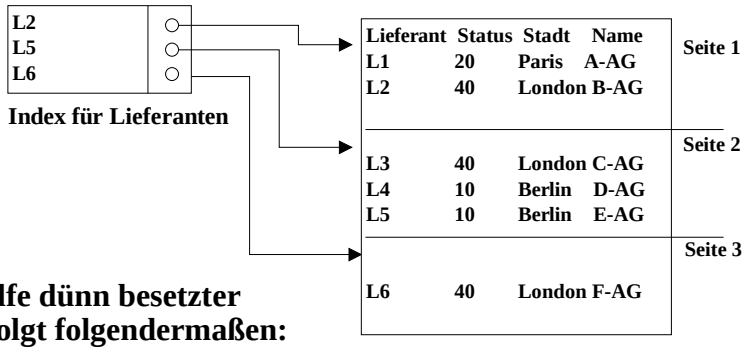
Dichter Index

- die Indexdatei umfaßt alle Attributwerte des indizierten Attributes mit den Satzadressen der zugehörigen Datensätze
- die Datensätze der Indexdatei werden i.d.R. sortiert nach Attributwerten gespeichert; Beispiel:
- die Indexdatei kann auch in Form einer invertierten Datei realisiert werden:
- sortierte Speicherung der Index-datei erlaubt binäres Suchen
- Abfragen, die sich nur auf Attributwerte eines Index beziehen, können schneller ausgeführt werden, als ohne Index
- Abfragen existenzieller Natur können manchmal ohne Zugriff auf den Datenfile beantwortet werden: z.B. gibt es einen Lieferanten in Berlin?
- Teile der Indexdatei können im Hauptspeicher gehalten werden
- Update, Einfügen und Löschen neuer Datensätze werden verlangsamt
- Zielkonflikt bei der Definition eines Index für ein Attribut: haben Abfragen oder Updates Priorität



Dünn besetzter Index

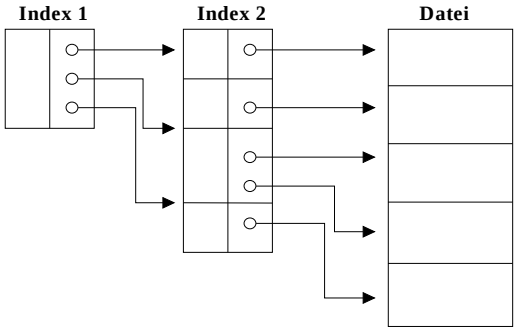
- nur ein Teil der Attributwerte u. Satzadressen sind in der Indexdatei
- die Datensätze werden in Datenblöcken (Seiten, Clustern) gespeichert
- in die Indexdatei wird nur der "größte Wert" des indizierten Attributes einer Seite übernommen:



- die Suche mit Hilfe dünn besetzter Indexdateien erfolgt folgendermaßen:
  - die Indexdatei wird nach einem Wert durchsucht, der größer gleich dem vorgegebenen Wert ist. Dieser Eintrag möge auf Seite p verweisen
  - Seite p wird in den Hauptspeicher gelesen und dort nach dem gewünschten Datensatz (sequentiell oder binär) durchsucht
  - soll der Datensatz eingefügt werden und ist nicht genügend Platz vorhanden, dann muß ein neuer Datenblock erzeugt werden, der den Datensatz aufnimmt
  - der Schlüssel dieses Datensatzes muß in der Indexdatei nachgetragen werden

Merkmale dünn besetzter Indexe

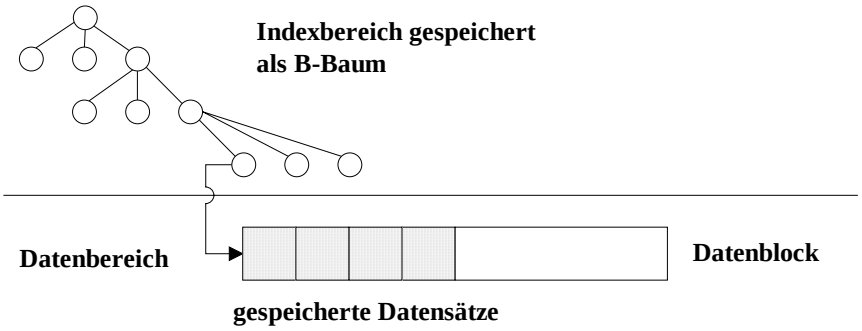
- **belegen weniger Speicherplatz, da weniger Einträge in der Indexdatei**
- **können i.d.R. schneller durchsucht werden, da die Indexdatei kleiner ist**
- **Update-Operationen erfordern weniger Aufwand, da die Indexdatei nicht in allen Fällen geändert werden muß**
- **mehrere dichte Indexe sind möglich, z.B. für jedes Attribut einen Index**
- **es kann nur höchstens ein dünn besetzter Index existieren, da ein dünn besetzter Index die Datei durch die Sortierreihenfolge in Cluster zerlegt**
- **dünn besetzte Indexe können auch hierarchisch organisiert werden, indem man eine Hierarchie von Indexen aufbaut**
- **die Satzadressen auf Ebene k der Indexdatei verweisen dann auf Indexdatensätze der Ebene k+1; lediglich auf der untersten Ebene verweisen die Adressen auf die eigentlichen Datensätze:**



B-Bäume

- **sind eine spezielle Form hierarchisch organisierter Indexe, die auf Grund ihrer Struktur den wahlfreien Zugriff mit garantierter Effizienz erlauben**
- **fast alle Datenbank- und Dateisysteme benutzen B-Bäume als Speicherform, da sie sowohl logisch fortlaufende, als auch wahlfreie Verarbeitung bieten**
- **der Indexbereich ist als B-Baum der Ordnung m organisiert, (m ist eine festgewählte natürliche Zahl ≥ 2)**
- **Ein B-Baum der Ordnung m kann folgendermaßen charakterisiert werden:**
  - **m-Wege Suchbaum, d.h. aus jedem Knoten führen maximal m Kanten; ein Knoten weist p Schlüssel und p+1 Satzadressen auf, wobei p < m ist**
  - **entweder ist der Baum leer oder aus seiner Wurzel führen mindestens zwei Kanten**
  - **jeder Knoten außer der Wurzel hat mindestens m/2 Schlüssel, alle Endknoten des B-Baumes haben die gleiche Tiefe**
  - **Satzadressen in den Endknoten zeigen auf Datenblöcke, in denen Datensätze logisch fortlaufend gespeichert werden**
  - **Lösch- bzw. Einfügeoperationen können ggf. die Tiefe des B-Baumes erniedrigen bzw. erhöhen**
  - **Theorem: Suche oder ein Einfügen in einen B-Baum der Ordnung m mit n Schlüsseln erfordert höchstens log n I/O-Operationen, wobei der Logarithmus die Basis m/2 hat**
  - **hält man bei einem B-Baum mit m = 256 den Wurzelknoten und eine weitere Indexebene im Hauptspeicher, so benötigt man für n = 100000 höchstens zwei I/O-Operationen**

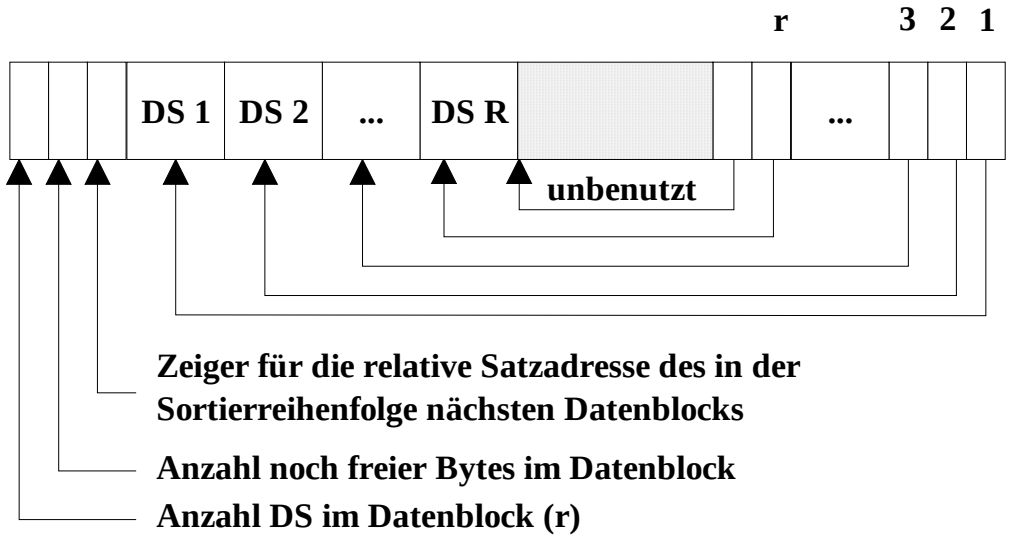
Struktur von Dateien mit B-Baum Index



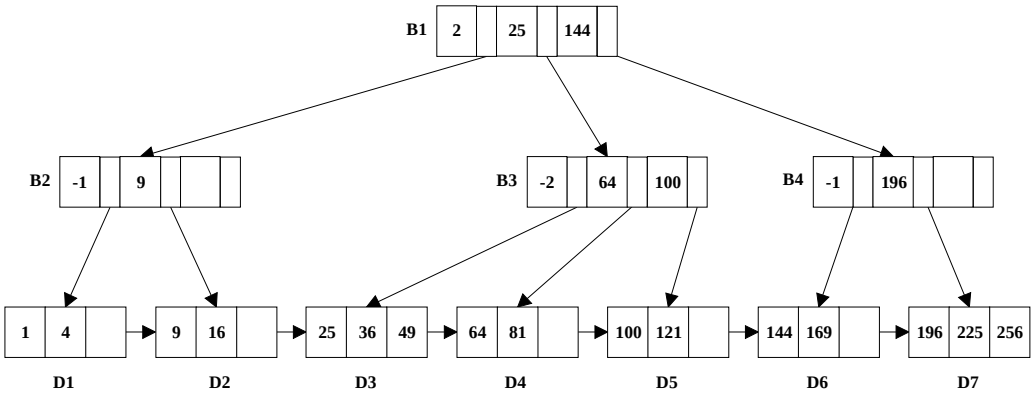
p	a_0	K_1	a_1	K_2	....	K_p	a_p

- Aufbau eines Indexblocks im Indexbereich
- p bezeichnet die Anzahl der gespeicherten Schlüssel, die aufsteigend sortiert sind, d.h. $K_1 < K_2 < .... < K_p$
- die a_i, i = 0,...,p sind Satzadressen, die auf Teilbäume verweisen
- im Teilbaum mit der Adresse a_0 sind alle Schlüssel K mit $K < K_1$; im Teilbaum mit der Adresse a_p, solche, die $\geq K_p$ sind
- im Teilbaum mit Adresse a_i $(1 \leq i < p)$ sind alle Schlüssel K mit $K_i \leq K < K_{i+1}$
- in den Endknoten des B-Baumes verweisen die Adressen auf die Datenblöcke

Aufbau eines Datenblockes im Datenbereich



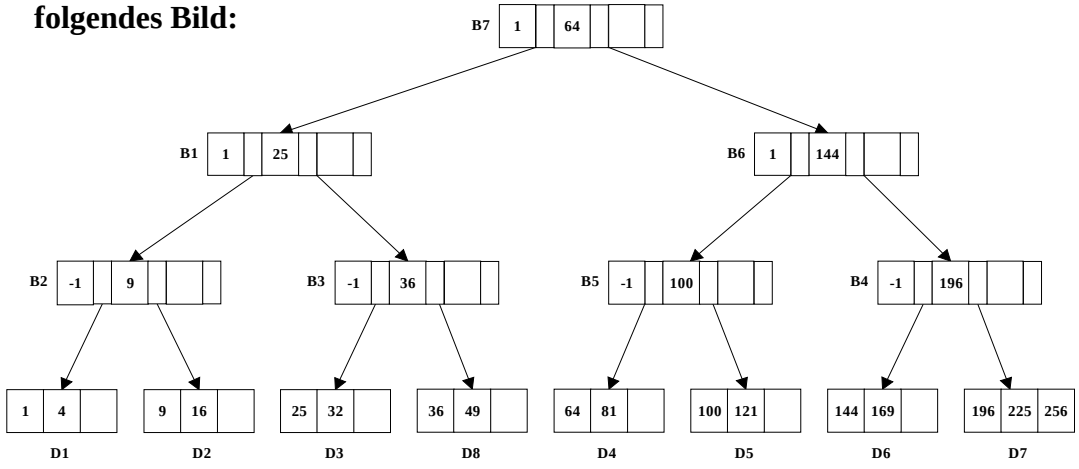
Beispiel eines B-Baumes der Ordnung 3



- **es soll ein DS mit dem Schlüsselwert 32 eingefügt werden. Dieser DS gehört in den Datenblock (DB) D3**
- **da aber dort kein Platz mehr ist, wird ein neuer DB D8 angelegt, der die DS "36" und "49" aufnimmt, während D3 "25" und "32" erhält**
- **der Schüssel 36 mit dem Zeiger zu D8 muß nun in B3 eingefügt werden, der jedoch bereits die maximale Anzahl von Schlüsseln enthält**

Beispiel (2)

- **es wird daher ein neuer B-Block B5 angelegt, in den die Zeiger zu D4, D5 und der Schlüssel 100 gespeichert werden. In B3 wird der Schlüssel 36 und die Zeiger zu D2 und D8 gespeichert**
- **der Schlüssel 64 mit dem Zeiger zu B5 wird auf die nächste Ebene weitergereicht, wo weitere Split-Operationen ausgeführt werden, so daß der B-Baum eine weitere Hierarchie-Ebene dazugewinnt. Das Endresultat zeigt folgendes Bild:**

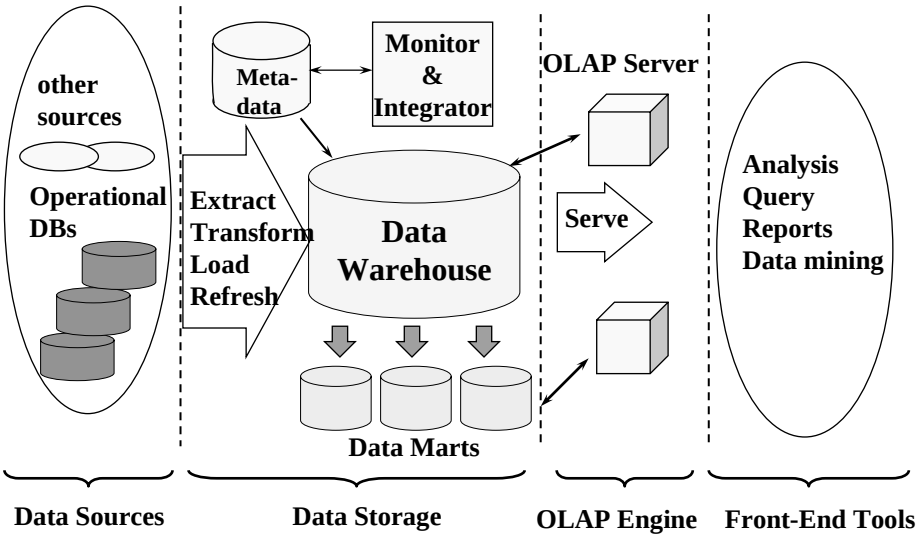


Data Warehouse

- ***ist eine unternehmensweit vereinheitlichte und integrierte Datenbasis mit möglichst vielen entscheidungsrelevanten Daten für schnellen und flexiblen Zugriff von Managementunterstützungssystemen (MUS)***
- **MUS: IS zur Unterstützung von Entscheidungsprozessen auf operativer, taktischer und strategischer Ebene (daten-, modellorientiert, wissensbasiert)**
  - **Managementinformationssysteme (MIS), Führungsinformationssysteme (EIS)**
  - **entscheidungsunterstützende Systeme (DSS), Expertensysteme (ES, XPS)**
- ***(multimediale) Datenquellen für DW*: Transaktionsdatensysteme mit deren RDBMS, Markdaten, Verbandsdaten, Daten von On-Line-Diensten**
- **trotz des hohen Hardwareaufwandes empfiehlt sich eine (redundante) Speicherung der entscheidungsrelevanten Daten aus folgenden Gründen:**
  - **zur Vereinheitlichung und Integration der heterogen Datenquellen mit Unterschieden in Begriffs-, Maß-, Zeitdefinitionen , Verschlüsselung und Codierung**
  - **Entscheidungsanalysen sind themenorientiert, z.B. bezogen auf Kunden, Produkte, Markgebiete, Verträge usw.; hingegen werden die Daten in Transaktionssystemen prozessorientiert gespeichert und sind daher nur schwer auszuwerten**
  - **für die Managementunterstützung ist die Analyse von historischen Daten in Form von Zeitreihen (z.B. die Absatzentwicklung eines Produktes über 12 Monate) wichtig; hingegen sind in Transaktionssystemen i.d.R. nur aktuelle Daten relevant**
  - **operative Systeme werden durch aufwendige Auswertungen nicht beeinträchtigt; Datenextraktion kann nachts erfolgen**

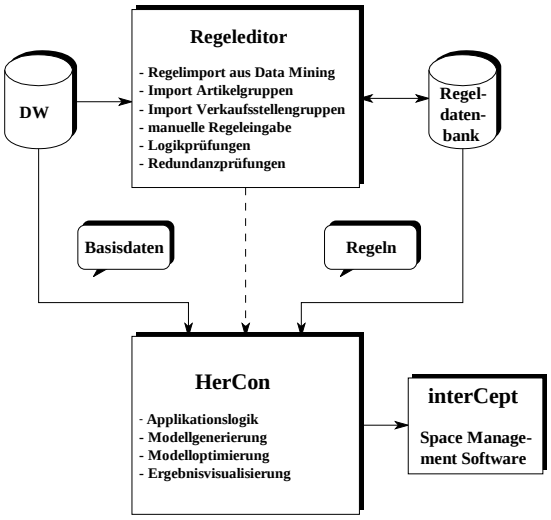
Grundlegende Data Warehouse Architekturen

- ***Definition: A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process***
- ***Data warehousing: The process of constructing and using data warehouses***
- ***Data Mart: a subset of corporate-wide data that is of value to a specific groups of users.***



Beispiel 1: HerCon, Data Mining, RDBS und Optimierung

1. **ein Data Warehouse (DW), in dem Stamm-, Leistungs- und Marktforschungsdaten abgelegt sind; diese Daten werden bereits in der manuellen Planung benutzt,**
2. **eine relationale Regeldatenbank mit einem integrierten Regeleditor, der zur komfortablen Eingabe und Verwaltung von Regeln dient,**
3. **ein 0-1 Optimierungsmodell, mit dem für jede Filiale unter Einbeziehung aller Restriktionen, Regeln und Basisdaten ein rohertragsmaximales Sortiment bestimmt wird,**
4. **eine Optimierungskomponente zur Lösung des Optimierungsmodells,**
5. **einer Visualisierungskomponente, mit der eine optimale Modellösung nach unterschiedlichsten Kriterien aufbereitet und dargestellt werden kann.**



Beispiel 2

Sortimentsanalyse im Supermarkt

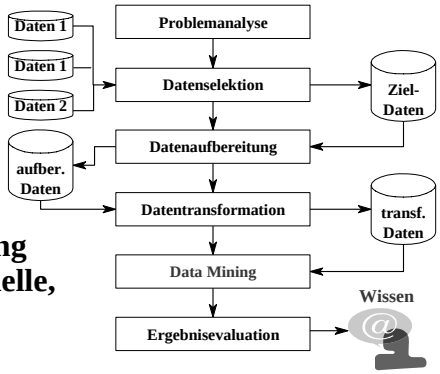
- **In Warenwirtschaftssystemen fallen pro Kunde u.a. Datum, Artikelnummer, verkaufte Menge und ggf. eine Kunden-Nr. an**
- **betrachtet man ein bestimmtes Zeitintervall und die hier pro Kunde verkauften Artikel (in diesem Beispiel ohne Mengen) so erhält man:**

Kunde	gekaufte Artikel
1	4711, 2021, 4387, 8996, 3577
2	2021, 2047, 4387, 4331
3	2021, 4331

- **Welche Artikel werden gemeinsam gekauft ?**
- **Welche Artikel sollen in den Regalen „nahe zusammen“ platziert werden?**
- **Einfluss auf die Preispolitik: wenn Artikel Y häufig zusammen mit X gekauft wird, dann kann u.U. X knapp und Y großzügig kalkuliert werden**
- **Welche Artikel verkaufen sich schlecht und ggf. warum?**
- **Welche gleichartigen Artikel verkaufen sich besser als andere gleichartige Artikel ?**
- **Wie kann das Marketing besser an die Käuferbedürfnisse angepasst werden?**

Grundprozesse des Data Mining

- **Ausgangspunkt ist ein Geschäftsproblem; in der Problemanalyse sollen Analysefokus und Erfolgskriterien definiert werden**
- **Nach Spezifikation der Analyseziele erfolgt eine Datenselektion / Daten-extraktion**
  - **aus ggf. unterschiedlichen Datenquellen kann ein *Data Warehouse* erstellt werden**
  - **Mit einem Web Crawler können Daten aus dem Web extrahiert werden, usw.**
- **Die wesentlichen Ziele der Datenaufbereitung sind:**
  - **Korrektur von Ausreißern, Ergänzung fehlender Werte, Datenreduktion (Löschen)**
  - ***Datenaufbereitung ist i.d.R. der zeitaufwendigste Prozess des Data Mining (>> 50% !)***
- **Bei der Datentransformation werden Attribute verknüpft, Daten skaliert und normiert**
- **Im Data Mining kommen vielfältige Konzepte / Methoden zum Einsatz:**
  - **Modellbildung**
  - **Klassifizieren / Segmentieren / Clustering**
  - **Mustererkennung**
  - **Bestimmung von Abhängigkeiten**
  - **Statistische Verfahren**
- **Ziel der Ergebnisevaluation ist die Generierung von Wissen (Erkenntnissen) wobei Ziele, Modelle, Verfahren, Prozesse einer kritischen Analyse unterzogen werden (müssen)**

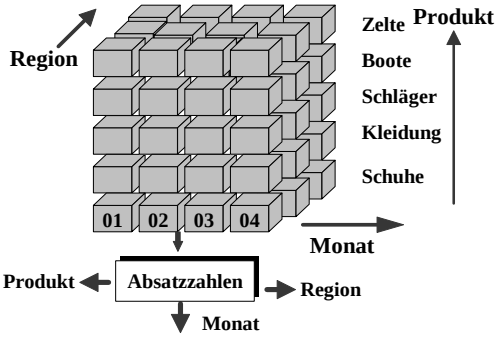


OLTP, OLAP und Data Mining

- ***OnLine Transaction Processing (OLTP):* klassische Transaktions-Verarbeitung in (relationalen) Datenbanken mit SQL**
- ***Online Analytic Processing (OLAP):* bezeichnet komplexere Abfragen auf historisierten Daten, bei denen die Daten aus logischer Sicht nach bestimmten Dimensionen aufbereitet wurden (dimensional data, cube view)**
  - **Der Fokus liegt beim OLAP liegt primär auf einer multidmensionalen Auswertung der Daten (eines Data Warehouse)**
  - **Ziel: detaillierter Einblick was in Zeitperioden passiert ist**
  - **Dazu gibt es eine Reihe spezieller Operationen (slice, dice, drill down, rotate …) , die in Softwaretools implementiert wurden, um Daten effizient aufzubereiten und auszuwerten**
  - **Visualisierung der Ergebnisse**
  - **OLAP kann in der Regel nicht vorhersagen was in der Zukunft passiert oder warum bestimmte Ereignisse stattgefunden haben**
  - **Techniken des OLAP können aber beim DM eine wichtige Rolle spielen, um die Problemstellung zu verstehen und ein DM-Projekt zu präzisieren**
- ***Data Mining***
  - **Fokus liegt auf der Wissensgerierung aus versteckten Zusammenhängen in den Daten**
  - **Konzepte und Methoden basieren u.a. auf Modellbildung, Mustererkennung, Ableitung probabilistischer Regeln, Klassifikationen von Objekten mit einem Prognoseziel, Präsentation der Ergebnisse in grafischer Form**

OLAP und Data Mining (2)

- **Logische Datensicht ist multidimensional**
- **Trennung zwischen Dimensionen und Daten**
- **Dimensionen werden hierarchisch angeordenet z.B.**
  - **Zeit:Tag → Woche → Monat → Jahr**
  - **Produkt: Produkt → P-Gruppe → Hersteller**
  - **Dimensionen haben Attribute, die in Tabellen abgelegt werden**
- **ROLAP: Dimensional Modeling Using Relational DBMS**
- **MOLAP: Dimensional Modeling Using the Multi Dimensional Model**



Fragestellung	Data Mining	OLAP
Kundenwert	Welche 10 Kunden bieten uns zukünftig das beste Potential und welche Attribute sind maßgebend?	Wer waren letztes Jahr unsere 10 besten Kunden?
Storno-Kunden	Welche Kunden werden vielleicht im nächsten Jahr zu einem Mitbewerber wechseln?	Welche Kunden haben im letzten Jahr ihren Vertrag gekündigt?
Cross-Selling	Welche bestehenden Kunden werden unser neues Produkt kaufen?	Wie viel haben wir von unserem neuen Produkt an bestehende Kunden verkauft?
Kredit-Prüfung	Welche Faktoren bestimmen das Kreditrisiko von Kunden?	Welche Kunden haben in den letzten 2 Jahren ihre Kreditzahlungen nicht erfüllt?

OLTP versus OLAP

- **Daten für OLAP werden i.d.R redundant gespeichert:**
  - **komplexe Abfragen bei OLAP betreffen große Datenmengen, die auch für leistungsfähige CPUs und I/O-Subsysteme die Performance für OLTP reduzieren (u.a. Locking)**
  - **Daten für OLAP werden für hohe Performance denormalisiert gespeichert**
  - **Die Historisierung der Daten für OLAP erfordert spezielle Speichertechniken**
  - **Eine Möglichkeit ist, ein Data Warehouse zu benutzen**
  - **Eine zweite Möglichkeit ist, die Daten aus der RDB transparent zu kopieren (Hauptspeicher, Festplatte) und die OLAP Abfragen auf der Kopie auszuführen**
  - **Die folgende Tabelle zeigt schematisch die Unterschiede zwischen OLTP und OLAP auf**

	OLTP	OLAP
Nutzer	Anwender, IT-Professional	IT-Professional
Fokus	Operative Anwendungen	EUS, Wissensgenerierung
DB Design	Anwendungsorientiert	Problemspezifisch, Projektorientiert
Daten	Zeitaktuell, detailliert, normalisierte Tabellen, konsistent	Historische Daten, aggregiert, mehrdimensional, integriert, konsolidiert
Nutzung	Permanent, Updates	ad-hoc, projektbezogen, Updates selten
Zugriff	Read / Wrrite über Primärschlüssel	Durchsuchen der Daten methodenabhängig
Abfragen	SQL	Komplexe Abfragen
# Tupel Zugriffe	Typisch unter 100	Millionen
# Nutzer	Ggf. Tausende	Ggf. Hunderte
DB Größe	100 MB-GB	100GB-TB
Performance	Transaktionen / ZE	Abfragen / ZE, Bearbeitungszeit, Algorithmen

OLAP Server Architectures

- **Kern von OLAP ist eine multidimensionale Sicht der Daten und zwar sowohl des Daten-models als auch der Operationen auf den Daten**
- **Daten Modelle**
  - **Relationen**
  - **Star und Snowflake Schemas**
  - **Daten-Würfel**
- **Wichtige Operatoren**
  - **slice & dice**
  - **roll-up, drill down**
  - **pivoting**
- **Relational OLAP (ROLAP)**
  - **Data Warehouse wird mit einem relationalen oder objekt-relationale DBMS implementiert; fehlende Komponenten werden mit OLAP Middleware implementiert**
  - **Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services**
  - **Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services**
  - **greater scalability**
- **Multidimensionales OLAP (MOLAP)**
  - **Daten werden in Arrays sparse gespeichert (sparse matrix techniques: Attributwert, Index, Zeiger auf Start einer Zeile / Spalte)**
  - **Indexstrukturen erlauben den Zugriff auf (pre-computed) summierte Daten**
  - **Extraktion aus RDB**
- **Hybrid OLAP (HOLAP): relationale Speicherung, aggregierte Daten in sparse Arrays**

Typische OLAP Daten-Würfel-Operationen

- **Roll up (drill-up): Daten werden aufsummiert bzw. der Detallierungsgrad verkleinert, wobei die Dimensionentabelle verwendet wird**
- **Drill down (roll down): die umgekehrte Operation, d.h. von summierten Daten wird die Detaillierungsebene vergrößert, bis hin zu einzelnen Tupeln**
- **Slice and dice: Project und Select**
- **Pivot (rotate): Rotation des Würfels, Visualizierung von 3D zu 2D Ebenen**
- **Beim Aufbau des Daten-Würfels werden diverse Algorithmen (Sorting, Hashing, Grouping) auf die dimensionalen Attribute angewandt, um betroffene Tupel in neue interne Speicherstrukturen zu überführen**
- **Große Arrays werden in kleinere Blöcke zerlegt, die im Hauptspeicher plaziert werden**
- **Speichertechniken für dünn besetzte Matrizen sind besonders wichtig, weil im mehrdimensionalen Würfel nur relativ wenige Punkte belegt sind →**
- **Bei einer effizienten Datenaggregation müssen die Speicherstrukturen so durchlaufen werden, dass auf möglicht wenig Speicherzellen zugegriffen wird und möglichst viele Zugriffe im Hauptspeicher erfolgen**

Beispiele für ROLAP Operationen (SUM)

- **Summierung pro Produkt und Periode : SELECT sum(amt) FROM SALE WHERE date = 1**

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

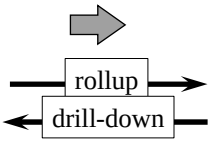
- **Summierung nach Datum: SELECT date, sum(amt) FROM SALE GROUP BY date**

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

ans	date	sum
	1	81
	2	48

- **Summierung pro Produkt und Periode**
  SELECT date, sum(amt) FROM SALE GROUP BY date, prodId

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4



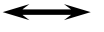
sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

MOLAP Würfel

- **Dimension 2**

sale	prodId	storeId	amt
	p1	s1	12
	p2	s1	11
	p1	s3	50
	p2	s2	8

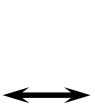
Multi-dimensional cube

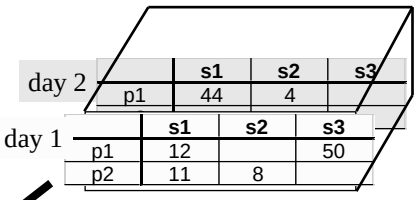


	s1	s2	s3
p1	12		50
p2	11	8	

- **Dimension 3**

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

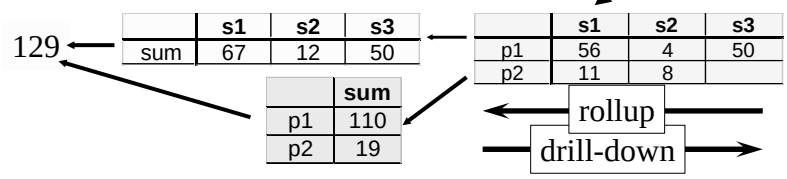




	s1	s2	s3
p1	44	4	

	s1	s2	s3
p1	12		50
p2	11	8	

- **Summation über Produkte / Stores**



	s1	s2	s3
sum	67	12	50

	s1	s2	s3
p1	56	4	50
p2	11	8	

	sum
p1	110
p2	19

- **Summation über Hierarchien**

store s1 in Region A;
stores s2, s3 in Region B

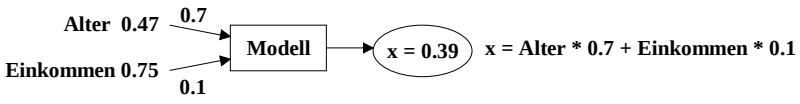
	region A	region B
p1	56	54
p2	11	8

Wichtige Methoden des Data Mining - Übersicht

- **Clusteranalyse**
  - zerlegt eine Menge von Objekten so in Teilmengen (Cluster), dass die Ähnlichkeit der Objekte innerhalb eines Clusters groß und die Ähnlichkeit der Objekte verschiedener Cluster klein sind
  - Partitionierende Verfahren: Anzahl Cluster und Anfangspartitionierung werden vorgegeben
  - Hierarchische Verfahren: hierbei werden die einzelnen Objekte iterativ fusioniert bis eine Fehlersumme einen Abbruch signalisiert
- **Assoziationsregeln**
  - Eine Regel besteht aus einem Paar logischer Aussagen,. d.h. Aussagen, die wahr oder falsch sein können
  - Die erste Komponente des Paars wird linke Seite (*Bedingung*) genannt, die zweite Komponente rechte Seite (*Konsequenz*) der Regel;
  - Eine aussagenlogische Regel besagt: wenn die linke Seite wahr ist, dann ist auch die rechte Seite wahr; z.B. wenn rote Grütze gekauft wird, dann auch Vanillesoße
  - In einer probabilistischen Regel ist die rechte Seite nur mit einer Wahrscheinlichkeit *p* wahr, wenn die linke Seite wahr ist (*bedingte Wahrscheinlichkeit)*
  - Aus einem Datenbestand wird an Hand von Häufigkeitsanalysen versucht, interessante Regeln abzuleiten, die mit hinreichend hoher Wahrscheinlichkeit wahr sind
- **Entscheidungsbäume**
  - an Hand einer Menge von Beispielobjekten und ihrer diskreten Attributwerte wird ein Baum so aufgebaut, dass er für andere Objekte zur Entscheidungsfindung dienen kann (Klassifikator)
  - Die Endknoten des Baumes bilden homogene Beispielmengen. Der Pfad von der Wurzel zu einem Endknoten beschreibt, die angewandten Bedingungen (ausgewählte Attributwerte)

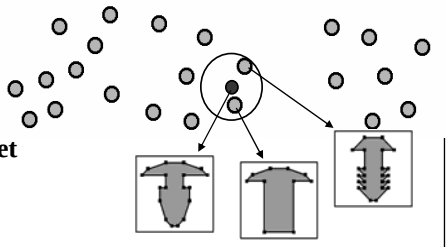
Wichtige Methoden des Data Mining – Übersicht (2)

- **Statistische Methoden**
  - **Korrelationsanalysen:** man vergleicht die Werte bestimmter Attribute in den Datensätzen. Lässt sich bei einem signifikant großen Teil der Werte ein funktionaler Zusammenhang wie f(x) = y, feststellen, dann gelten die Daten als korreliert
  - **Regressionsanalyse:** aus Zeitreihendaten und z.B. einem Funktionstyp f(x) = a+bx (lineare RGA) werden die Parameter a, b so bestimmt, dass ein Fehlermaß bezogen auf die Zeitreihe klein ist
  - Die Funktion f kann dann zur Prognose eingesetzt werden
  - Bei multipler Regressionsanalyse ist die Funktion $f(x_1,x_2,..\int,x_n)$ von mehreren Parametern abhängig
  - Es gibt eine Reihe von nichtlinearen Funktionen z.B. ax^b, ae^{bx}, a + b ln(x), a + b / x die in lineare Funktionen transformiert werden können und die dann mit Methoden der linearen Regressionsanalyse untersucht werden können.
- **Neuronale Netze**
  - sind gerichtete Graphen, dessen Knoten selbstständig rechnende Einheiten sind
  - Die Kanten verbinden die Knoten in verschiedenen typischen Formen
  - An Hand von Beispieldaten wird das Netzt trainiert, um für neue Daten Werte zu bestimmen. Für jedes Beispiel ist bekannt, was die gewünschten Ausgabe-Attributwerte sein sollen.
  - Weichen tatsächliche und gewünschte Ausgabewerte voneinander ab, dann müssen die Gewichte im Netz so verändert werden, dass sich der Fehler bei der Ausgabe verringert. Dieser Prozess erfolgt im Idealfall so lange, bis alle Beispiele richtig berechnet werden



Grundlagen (1)

- **Ähnlichkeitsanalyse komplexer Objekte**
  - **Es müssen geeignete Merkmale definiert bzw. ausgewählt werden, die für die Unterscheidung (Klassifikation, Ähnlichkeit) der Objekte relevant sind**
  - **Die Merkmale eines Objektes werden in einem Merkmals-Vektor zusammengefaßt**
  - **Zu einem gegebenen Anfrage-Objekt sollen in der Datenbasis ähnliche Objekte gesucht werden; dazu werden die entsprecheden Merkmals-Vektoren verglichen**
  - **Ein Ähnlichkeitkeitsmaß von Vektoren im R^n wird über Metriken definiert: seien p, $q \in R^n$ die Merkmalsvektoren von zwei Objekten dann ist $d(p,q) \in R_+$ deren "Abstand"**
- **Skalen-Niveaus von Merkmalen**
  - ***Binär*: Merkmalswerte sind binär**
  - ***Nominal*: Merkmalswerte können eine endliche Anzahl diskreter Werte annehmen**
  - ***Ordinal*: es existiert eine (vollständige) Ordnungs-relation zwischen den Merkmalsvektoren (besser / schlechter) aber kein meßbarer Abstand**
  - ***Metrisch*: die Ähnlichkeit von Objekten kann über eine Metrik (Distanzfunktion) definiert werden; dabei wird nicht das Objekt selbst sondern ein zugehöriger Merkmalsvektor betrachtet**

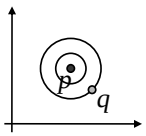


Grundlagen (2)

- **Eine Metrik d ist eine Funktion mit folg. Eigenschaften**
  - **$d: R^n x R^n \rightarrow R_+$, wobei $d(p,q) = 0 \Leftrightarrow p=q$ für $p,q \in R^n$**
  - **$d(p,q) = d(q,p)$ und $d(p,q) \leq d(p,r)+d(r,q)$ (Dreiecksungleichung)**
  - **Bei binären Attributwerten von p,q wird definiert: $d(p,q)=\delta(p_1-q_1)+\delta(p_2-q_2)+...$ wobei $\delta(p_i-q_i)=1$, wenn $p_i=q_i$ und 0 sonst**
  - **Bei kategorischen Attributwerten von p, q wird definiert: d(p,q) = (n – g) / n, wobei g die Anzahl der gleichen Attributwerte in p und q ist**
  - **Beispiele**

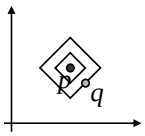
Euklidische Metrik

$d_1 = ((p_1-q_1)^2+(p_2-q_2)^2+...)^{1/2}$



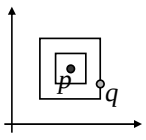
Manhattan-Metrik (L_2)

$d_2 = |p_1-q_1|+|p_2-q_2|+...$



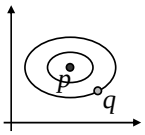
Maximums-Metrik

$d_\infty = \max\{|p_1-q_1|, |p_2-q_2|,...\}$



Gewichtete Euklidische Metrik

$d = (w_1(p_1-q_1)^2 + w_2(p_2-q_2)^2+...)^{1/2}$



  - **Gewichtete Metriken kommen dann zum Einsatz, wenn die Wertebereiche der Attribute sehr unterschiedlich sind, z.B. $A_1=\{0.1,0.02,0.15\}$ und $A_2=\{9.1,10.2,8.15\}$**

Clustering

- **Beim Clustering wird eine Menge von Objekten so in Teilmengen (Cluster) zerlegt, dass die Ähnlichkeit der Objekte innerhalb eines Clusters maximiert und die Ähnlichkeit der Objekte verschiedener Cluster minimiert wird**
- **Typische Anwendungen**
  - **Kunden- / Artikel- / Fillialsegmentierung: Ermittlung homogener Gruppen**
  - **Bestimmung von Benutzergruppen im Web durch Clustering der Web-Logs**
  - **Strukturierung großer Mengen von Textdokumenten durch hierarchisches Clustering**
- **Vielzahl Methoden; wir betrachten hier nur partitionierende Verfahren**
  - **gegeben ist eine *Anfangspartition* in k Cluster und eine Metrik d**
  - **die Partition in *k* Cluster soll mit Hilfe exakter oder heuristischer Verfahren so redefiniert werden, dass eine Gütefunktion minimiert wird**
- **Grundbegriffe beim Clustering:**

Eine Menge M **wird disjunkt in k Cluster C_i zerlegt :** $M = \bigcup_{i=1}^{k} C_i, \quad C_i \subset R^n$

Zentroid μ_C : Mittelwert aller Punkte eines Clusters C : $\mu_C = 1/|J| \sum_{j \in J} p^j$

wobei $C \subset R^n, C = \{p^1, p^2, .., p^{|J|}\}$, **d.h.** $|C| = |J|$

Maß für die Kompaktheit eines Clusters C : $TD^2(C) = \sum_{p \in C} d(p, \mu_C)^2$

Maß für die Kompaktheit eines Clustering : $TD^2 = \sum_{i=1}^{k} TD^2(C_i)$

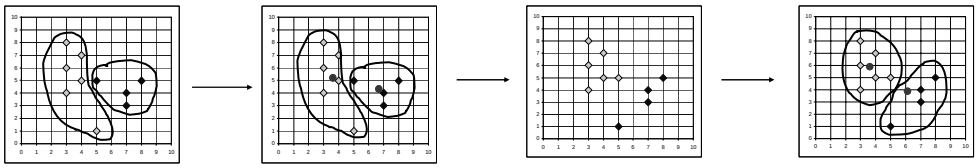
Ein Basis-Algorithmus

- **Gegeben ist die zu untersuchende endliche Menge von Objekten mit deren Attributmenge $M \subset R^n$; M sei partitioniert in k Cluster C_i, i=1,..,k**

1. **Initialisiere $M = C_1 \cup C_2 \cup ... C_k$**
2. **Berechne für jedes C_i den Zentroid-Punkt $z_i \in R^n$, i=1,..,k**
3. **Für jeden Punkt $p \in M$ wird $d(p,z_i)$ berechnet; bilde die neuen Cluster C_i durch Zuordnung jedes Punktes p zum nächsten Zentroid-Punkt, bei Gleichheit wird das C_i mit kleinerem i gewählt**
4. **Wenn die Partitionierung gleich bleibt, so terminiere anderenfalls gehe nach 2.**

- **Beispiel: M = {6, 9, 37, 38, 43, 47}, k=2, C1={6,38,43}, C2={9,37,47}**

1. **$M = C1 \cup C2$**
2. **z1 = 29, z2 = 31**
3. **Die neuen Cluster sind C1={6,9} und C2 = {37,38,43,47} und z1=7.5 und z2= 41.25; dies ist die endgültige Klassenaufteilung und der Algorithmus terminiert**



Ein anderer Basis-Algorithmus

- **Gegeben ist die zu untersuchende endliche Menge von Objekten mit deren Attributmenge $M \subset R^n$; M sei partitioniert in k Cluster C_i, i=1,..,k**

1. **Initialisiere $M = C_1 \cup C_2 \cup \ldots C_k$**
2. **Berechne für jedes C_i den Zentroid-Punkt $z_i \in R^n$, $TD^2(C_i)$, i=1,..,k und TD^2**
3. **Berechne für jedes $p \in C_i$ die Veränderung von TD^2, die sich ergibt, wenn p in einen anderen Cluster verschoben wird; p wird verschoben, wenn sich TD^2 reduziert; bei einer Verschiebung müssen die Zentroide neu berechnet werden.**
4. **Wenn Punkte ihre Klasse wechselten, dann gehe nach 2. anderenfalls terminiere.**

- **Beispiel: M = {6, 9, 37, 38, 43, 47}, k=2, C1={6,38,43}, C2={9,37,47}**

1. **$M = C1 \cup C2$**
2. **z1 = 29, z2 = 31, $TD^2(C)=1582$**
3. **Wir betrachten p=43; wenn p nach C2 verschoben wird, so ist C1=={6,38} und C2={9,37,47,43}; dann ist z1= 22 und z2 = 34 und $TD^2(C) = 1396$;**
   Beim Wechsel des Elementes 9 nach C1 und 38 nach C2 verringert sich $TD^2(C)$
4. **Mit C1={6,9} und C2 = {37,38,43,47} erhalten wir die endgültige Klassenaufteilung**

- **Es gibt viele Varianten des Basisalgorithmen**
  - **Statt Zentroid wird ein repräsentatives Element vorgegeben (Metedoid)**
  - **Parameter k und Anfangspartitionierung können ein Problem darstellen**
  - **Elimination sehr kleiner Cluster, Aufspalten von großen Clustern**

Hierarchisches Clustering - Anwendung

- **Im Einzelhandel werden regelmäßig Aktionsdisplays angeboten, bei denen häufig gekaufte Artikel in fester Mengenzusammensetzung kombiniert werden**
- **Die Aktionsdisplay orientieren sich thematisch: z.B. Weihnachten, Ostern, Pfingsten ...**
- **Aktionsdisplays werden häufig auf Grund von Erfahrung zusammengestellt**
- **Zunächst werden die Verkaufsfilialen einem Clustering unterzogen**
- **Bei dieser Anwendung [14] ist kein Clustering vorgeben; mit dem Verfahren von Ward wird ein Clustering bestimmt; Kernpunkt sind:**
  - **Das Verfahren startet mit den einelementigen Clustermengen der Filialen**
  - **Danach werden sukzessive diejenigen Objekte fusioniert, die zur kleinsten Zunahme der Fehlerquadratmengen ΔF_{ij} führen, wobei**

n : Anzahl der Merkmale, n_i : Anzahl Objekte im Cluster i, n_j : Anzahl Objekte im Cluster j
x_{ik} : Wert des Merkmals k im Cluster i, x_{jk} : Wert des Merkmals k im Cluster j
k : Sortimentsindex, $\Delta F_{ij} = (n_i * n_j) / (n_i + n_j) * \sum_{k=1}^{n} (x_{ik} - x_{jk})^2$
$x_{ij,k}$: Wert des neuen Merkmals k im neugebildeten Cluster aus i und j, $x_{ij,k} = (x_{ik} + x_{jk}) / 2$

  - **ΔF_{ij} steigt mit jeder Fusionierung und der Erhöhung der Anzahl Cluster**
  - **Kritisch ist das Abbruchkriterium, d.h. wann man die Fusionierung abbricht**
  - **Nachdem die Filialen Clustern zugeordnet wurden, wird mit mathematischer Optimierung Anzahl und Zusammensetzung von Aktionsdisplays bestimmt**

Assoziationsanalyse

- **Assoziationsregeln: Ableitung von Regeln zwischen Objekten aufgrund hinreichend vieler Beobachtungen, z.B. für Warenkorbanalysen**
- **Kennzeichnend für Warenkorbanalysen sind:**
  - **an Scannerkassen werden alle Kaufvorgänge protokolliert (Bondaten)**
  - **Bei jedem Kaufvorgang werden Filiale, Artikel, Mengen, Datum aufgezeichnet**
  - **Es fallen pro Filiale in kurzer Zeit große Transaktionsdatenmengen an**
  - **Ziel ist festzustellen welche Waren typisch gemeinsam gekauft werden**
- **Assoziationsregeln haben die Struktur von wenn-dann Bedingungen und sind anders als Regeln der Logik wahrscheinlichkeitstheoretischer Natur**
- **Regeln werden aus Häufigkeitsanalysen von Tupeln der Datenbank erzeugt**
- **Wir betrachten der Einfachheit halber binäre Attribute**
  - **Eine typische Regel lautet dann: wenn $A=1 \wedge B=1 \Rightarrow C=1$ mit Wahrscheinlichkeit p d.h. $p= p(C=1 \mid A=1, B=1)$ (bedingte Wahrscheinlichkeit)**
  - **Der Support ist der Anteil der Tupel bei denen A und B und C auftreten an allen Tupeln**
  - **Die Konfidenz ist der Quotient aus der Anzahl Tupel die C enthalten an den Tupeln, die A und B enthalten**
  - **Beispiel: wenn unter 100000 Tupeln 2000 Tupel A und B enthalten, von denen 800 C enthalten, so ist der Support dieser Regel 800/100000=0.8 und die Konfidenz 800/2000=0.4; die Konfidenz ist also ein Schätzwert für p.**
  - **Support: Maß für *Wichtigkeit* einer Regel; Konfidenz: Maß für *Richtigkeit* einer Regel**

Assoziationsanalyse - Grundlegende Definitionen

- **Menge von Items (Objekten): $I=\{I_1,I_2,\ldots,I_m\}$**
- **Transaktionsmenge D: Menge von Transaktionen (TA) $D=\{t_1,t_2,\ldots,t_n\}$, $t_i \subseteq I$**
- **Itemset C: eine beliebige Teilmenge von I, d.h. $C \subseteq I$; *die Items werden in C lexikographisch sortiert, d.h. C wird als geordnete Menge betrachtet***
- **Support eines Itemsets: Prozentsatz der TA aus D, die diesen Itemset enthalten**
- **Häufige Itemsets: haben einen minmalen Support in D, z.B. 25% oder 0.25**
- **Assoziations-Regel (AR): eine wahrscheinlichkeitsorientierte Regel der Form $X \Rightarrow Y$, wobei X, $Y \subseteq I$ und $X \cap Y = \varnothing$**
- **Support s einer AR $X \Rightarrow Y$: Prozentsatz der TA aus D, die $X \cup Y$ enthalten**
- **Konfidenz k einer AR $X \Rightarrow Y$: Quotient aus der Anzahl der TA die $X \cup Y$ enthalten, zu der Anzahl von TA die X enthalten**
- **Assoziationsproblem: Aus gegebenem I und D sollen alle Regeln vom Typ $X \Rightarrow Y$ bestimmt werden, die eine(n) Mindestsupport und Mindestkonfidenz ausfweisen; diese Aufgabe besteht aus zwei Teilen:**
  - **Schritt 1: Finde alle Itemsets, die bestimmte Attributwerte mit minimalem Support enthalten: sogenannte häufige Mengen (eigenständige Forschungsarbeiten!)**
  - **Ableitung von Regeln aus häufigen Mengen mit gegebener Mindestkonfidenz**

Bestimmung der häufig auftretenden Itemsets

- **C_k: Kandidaten-Itemsets der Länge k, L_k: Menge der häufigen Itemsets der Länge k**
- **Die Operation *Subset(C_k,t)* bestimmt alle Kandidaten aus C_k, die in der Transaktion t enthalten sind**
- **Die Operation *A-Gen(L_k)* erzeugt mit einer Join-Operation aus Itemsets p,q$\in L_k$ ein Element r $\in C_k$, wenn sie in der ersten k-1 Items übereinstimmen (lexikogr. Ordnung!)**
- **Nach der Join-Operation folgt in A-Gen(L_k) eine Prune-Operation, um alle Kandidaten-Itemsets zu entfernen, die nicht in L_{k-1} sind**
- **Beispiel: L_3 = {(1 2 3), (1 2 4), (1 3 4), (1 3 5), (2 3 4)}; A-Gen(L_3) erzeugt {(1 2 3 4), (1 3 4 5)}; die Prune-Operation löscht (1 3 4 5) und man erhält C_4= A-Gen(L_3) = {(1 2 3 4)}**

```
Apriori (I, D, minsup)
L1 := {häufige 1-Itemsets aus I};
k := 2;
while Lk-1 ≠ ∅ do
   Ck := A-Gen(Lk - 1);
   for each t ∈ D do
       CT := Subset(Ck, t); for each Kandidat c ∈ CT do c.count++;
       Lk := {c ∈ Ck | (c.count / |D|) ≥ minsup};
       k++;
return ∪k Lk;
```

Generierung von Regeln aus häufigen Itemsets

- **Nach der Bestimmung der häufigen Mengen können Regeln mit einer Mindestkonfidenz generiert werden**
  - **X sei die Menge der häufig vorkommenden Itemsets aus Schritt 1**
  - **für jede Teilmenge A von X die Regel A ⇒ (X – A) bilden**
  - **Regeln streichen, die nicht die minimale Konfidenz haben**
  - **Berechnung der Konfidenz einer Regel A ⇒ (X – A) = sup (X) / sup (A)**
- **Algorithmus zur Generierung der Regeln mit einer Mindestkonfidenz k**
  - **Input D, I, L, s, k**
  - **R Menge der Assoziationsregeln die einen Mindestsupport s und eine Mindestkonfidenz k aufweisen**

```
Gen-AR (D, I, L, s, k)
R = ∅
For each l ∈ L do
     For each x ⊂ l mit x ≠ ∅ und x ≠ L do
          if  support(l) / support(x) ≥ k then
              R = R ∪ { x ⇒ (l – x)}
          endif
     endfor
endfor
```

Beispiel

- I = { Bier, Brot, Butter, Marmeldade, Milch}
- Support von {Brot, Butter} ist 60%

Transaktion	Items
t_1	Brot, Marmelade, Butter
t_2	Brot, Butter
t_3	Brot, Milch, Butter
t_4	Bier, Brot
t_3	Bier, Milch

- Anwendung des Algorithmus Apriori

Bestimmung der häufigen Itemsets, wobei s = 0.3 und k = 0.5

Pass	Kandidaten	Häufige Itemsets
1	{Bier}, {Brot}, {Butter}, {Marmelade}, {Milch}	{Bier}, {Brot}, {Butter}, {Milch}
2	{Bier, Brot}, {Bier, Milch}, {Bier, Butter}, {Brot, Milch}, {Brot, Butter}, {Butter, Milch}	{Brot, Butter}

- Einfache Regeln

X ⇒ Y	s	k
Brot ⇒ Butter	0.6	0.75
Butter ⇒ Brot	0.6	1.00
Bier ⇒ Brot	0.2	0.50
Butter ⇒ Marmelade	0.2	0.33

Beispiel (1)

- Dieses Beispiel ist aus [13]
- Voraussetzung: Itemsets mit einem Support ≥ 0.25, d.h. mind. 2 von 8

tid	Itemset
1	Brot, Honig, Käse
2	Käse
3	Brot, Butter, Käse, Marmelade
4	Brot, Wurst
5	Brot, Butter
6	Brot, Wurst
7	Brot, Käse
8	Brot, Butter, Käse

Zählung 1 - Itemmengen

C_1	Support
Brot,	7
Butter	3
Honig	1
Käse	5
Marmelade	1
Wurst	2

L_1
Brot,
Butter
~~Honig~~
Käse
~~Marmelade~~
Wurst

- Generierung der Kandidaten mit 2 Items aus L_1 und mit 3 Items aus L_2; zu beachten ist, dass in C_3 (temp) Tupel entfernt werden müssen, die in C_2 nicht auftreten

Zählung 2-Itemmengen

C_2	Support
Brot, Butter	3
Brot, Käse	4
Brot, Wurst	2
Butter, Käse	2
Butter, Wurst	0
Käse, Wurst	0

L_2
Brot, Butter
Brot, Käse
Brot, Wurst
Butter, Käse
~~Butter, Wurst~~
~~Käse, Wurst~~

Generierung Kandidaten mit 3-Items

C_3 (temp)	C_3
Brot, Butter, Käse	Brot, Butter, Käse
Brot, Butter, Wurst	~~Brot, Butter, Wurst~~
Brot, Käse, Wurst	~~Brot, Käse, Wurst~~

- Zählung der Kandidaten mit 3 Items
- Das Verfahren bricht ab, weil keine 4 Itemsets möglich sind

Zählung 3-Itemmengen

C_3	Support
Brot, Butter, Käse	2

Beispiel (2)

- **Häufige Mengen**

L_1	
Brot	7
Butter	3
Käse	5
Wurst	2

L_2	
Brot, Butter	3
Brot, Käse	4
Brot, Wurst	2
Butter, Käse	2

L_3	
Brot, Butter, Käse	2

- **Generierung von Regeln aus den häufigen Mengen L_2 und L_3**

Regeln	Konfidenz	>k = 0.6 ?
Butter ⇒ Brot	3/3	j
Brot ⇒ Butter	3/7	n
Käse ⇒ Brot	4/5	j
Brot ⇒ Käse	4/7	n
Wurst ⇒ Brot	2/2	j
Brot ⇒ Wurst	2/7	n
Käse ⇒ Butter	2/5	n
Butter ⇒ Käse	2/3	j

2-Item Regeln	Konfidenz
Butter ⇒ Brot	1.00
Käse ⇒ Brot	0.80
Wurst ⇒ Brot	1.00
Butter ⇒ Käse	0.67

Regeln	Konfidenz	>k = 0.6 ?
(Butter, Käse) ⇒ Brot	2/2	j
(Brot, Käse) ⇒ Butter	2/4	n
(Brot, Butter) ⇒ Käse	2/3	j

3-Item-Regeln	Konfidenz
(Butter, Käse) ⇒ Brot	1.00
(Brot, Butter) ⇒ Käse	0.66

- **Resultate**

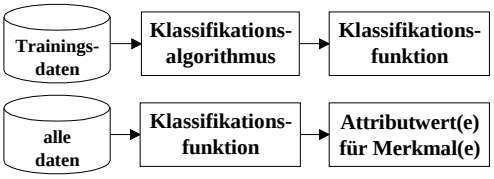
2-Item Regeln	k	s
Butter ⇒ Brot	1.00	0.38
Käse ⇒ Brot	0.80	0.50
Wurst ⇒ Brot	1.00	0.25
Butter ⇒ Käse	0.66	0.25

3-Item-Regeln	k	s
(Butter, Käse) ⇒ Brot	1.00	0.25
(Brot, Butter) ⇒ Käse	0.66	0.25
Butter ⇒ (Brot, Käse)	0.66	0.25

Klassifikationsprobleme

- **Gegeben: eine Menge O von Objekten des Formats $(o_1,.., o_d)$ mit Attributen A_i, i=1,..,d und Klassenzugehörigkeit c_i, $c_i \in C = \{c_1, \ldots, c_k\}$; gesucht wird:**
  - **die Klassenzugehörigkeit für Objekte aus D \ O**
  - **eine Klassifikationsfunktion K : D → C**
  - **Beispiel: D ist in der Tabelle dargestellt**
  - **Eine triviale K-Funktion ist:**
    Risiko='mittel'
    if Alter > 45 and PS<135 ⇒ Risiko='niedrig'
    if PS>275 ⇒ Risiko='hoch'
    if Alter < 25 ⇒ Risiko='hoch'

id	Alter	Autotyp	PS	Risiko
1	35	Boxter S	275	hoch
2	19	Polo	55	hoch
3	49	Avensis 1,8	129	niedrig
4	45	Passat TDI	140	mittel
5	49	E 55 AMG	375	hoch
6	55	E 200 CDI	122	niedrig



- **Abgrenzung zum Clustering**
  - **Klassifikation: Klassen apriori bekannt**
  - **Clustering: Klassen werden erst gesucht**
- **Verwandtes Problem: Prognoseverfahren**
  - **Gegeben ist eine endliche Menge von numerischen Werten (x_i,y_i), i=1,.,n, wobei unterstellt wird, dass das Merkmal y von x in unbekannter Form abhängt**
  - **Ist x gegeben, wobei $x \notin \{x_1,\ldots,x_n\}$, so wird der entsprechende Wert für y gesucht**
  - **Als Methode kommt z.B. die Regressionsanalyse in Betracht, wenn bestimmte statistische Bedingungen gelten**
  - **Z.B. Hypothese: Einkommen = c + log (Alter) + Gehalt (Berufsgruppe)?**

Systemanbieter von Data Mining Tools (Auswahl)

- **http://www.spss.com/de/**
- **http://sas.com/**
- **http://www.statsoft.de/**
- **http://www-306.ibm.com/software/data/iminer/**
- **http://www.cognos.com/**
- **http://www.prudsys.com/Software/Softwarepakete/**
- **http://www.insightful.com/products/iminer/**
- **http://www.data-miner.com/**
- **http://www.thearling.com/**
- **http://www.angoss.com/**
- **http://www.oracle.com/technology/software/index.html#dw**
- **http://www.microsoft.com/uk/windowsserversystem/bi/platform.mspx**

12 Literaturverzeichnis

1. **Date, C.J., An Introduction to Database Systems, Addison-Wesley, 2004**
2. **Saake, G. und K.-U. Sattler, Datenbanken & Java (JDBC, SQLJ und ODMG), dpunkt,2000**
3. **Codd, E.F., The Relational Model for Database Management: Version 2, Addison-Wesley,**
4. **Yarger, R.J. G. Reese & T. King, MySQL & mSQL, O'Reilly Verlag, 2000**
5. **Vetter, M., Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung, Teubner, 1989**
6. **Heuer, A. und G. Saake, Datenbanken - Konzepte und Sprachen, Thomson Publishing, 1995**
7. **Database Modeling & Design, Toby J. Teorey, Morgan Kaufmann Publishers, Inc. San Francisco, Calaifornia**
8. **Knowledge Discovery in Databases: Techniken und Anwendungen, Ester M. und Sander J., Springer, 2000**
9. **Data Mining: Concepts and Techniques, Han J. and Kamber M., Morgan Kaufmann Pub., 2000**
10. **Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Pub., 2000**
11. **Diplomarbeit C. Weiner und N. Askar zum Themenkomplex Aktionsmanagement basierend auf Mathematischer Optimierung**
12. **Data Mining im Marketing, Vorlesung, pdf-Dokument, K.D. Wilde, Katholische Universität Eichstätt-Ingolstadt (2005)**
13. **Data Mining im Marketing, Übung, pdf-Dokument, A. Englbrecht, Katholische Universität Eichstätt-Ingolstadt (2005)**