universität wien

Kapitel 1

Einführung und Motivation

Inhalt

1.1 Was ist ein DBMS?

Ein **Datenbankmanagementsystem** (**DBMS**) besteht aus einer Sammlung korrelierter Daten (i.e. Datenbank) und Programmen, die darauf zugreifen

Ein DBMS speichert Informationen über einen spezifischen Ausschnitt der Realität

„Modellierung der Welt“

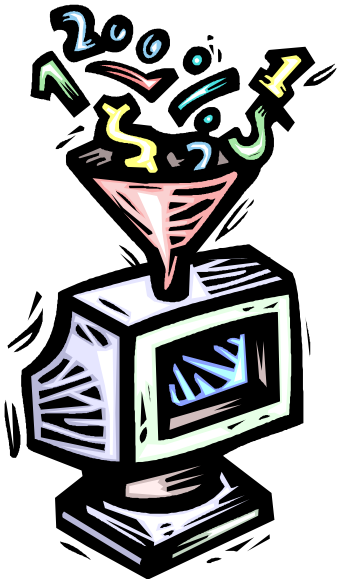
Datenbanksystem ist Teil eines Informationssystems

Effizienz

Zugriffszeit, Speicherplatz, …

Komfort

Zugriffsarten, Sicherheit, Persistenz, Korrektheit, Wiederherstellbarkeit, …



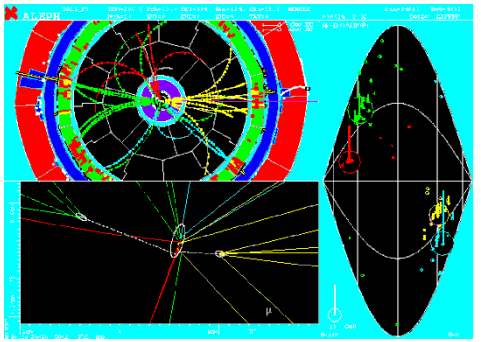
Anwendungsbereiche

Allgemeine Anwendungen

- Banken, Fluglinien, Universitäten, Telekommunikation, …
- Sind in allen Bereichen unseres Lebens zu finden
  - Heute vor allem im Internet und Web

Spezielle Anwendungen

- Im Vordergrund Verwaltung großer Datenmengen, z.B.
  - Magellansonde (3 Terabyte, NASA 1989)
  - Wettervorhersage (50 Terabyte, Climate Corporation aktuell)
  - CERN Experiment (1-100 Petabyte, Datagrid heute und zukünftig)
  - …



Schlagworte: Big Data, Data Science

1.2 DBMS versus Dateisysteme

In einem **Dateisystem** wird die Information persistent in Dateien gespeichert, die durch das Betriebssystem verwaltet werden

Nachteile von Dateisystemen (Dateien)

- Daten Redundanz und Inkonsistenz
  - Fehlende einheitliche Gesamtschau (Modellierungsschwäche)
- Verschiedene Dateiformate
  - doppelt gespeicherte Information
- Schwierigkeit/Komplexität beim Zugriff auf Daten
  - Für jede Aufgabe eigenes, neu zu schreibendes, Programm
- Fehlende Datenintegrität
  - Keine logische Bedingungen auf den Daten
- Inkonsistente Zustände nach einem Absturz
  - Fehlende atomare Operationen
- Beeinflussende konkurrenzierende Zugriffe
  - Bei Dateisystem nur rudimentär kontrolliert
- Sicherheitsproblematik
  - Bei Dateisystemen üblicherweise sehr simpel

Datenbanksysteme bieten Lösungen für all diese Probleme!

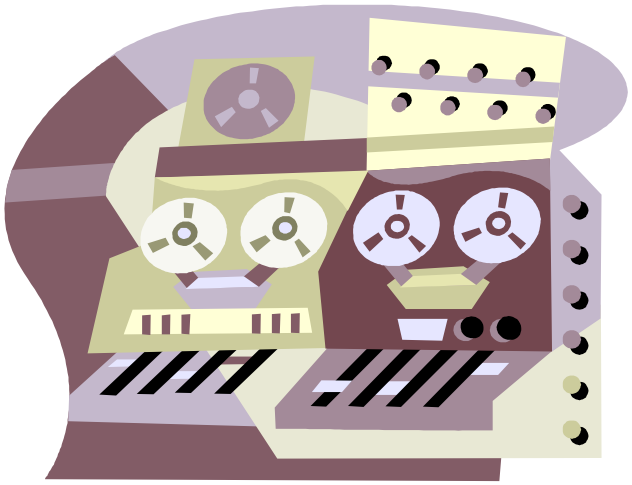
Erkenntnis

Daten werden integriert in einem Datenbanksystem gehalten und dem Benutzer zur Verfügung gestellt

- Die Datenbank ist das Herz jedes Unternehmens

Zitat

"Im Jahre 2000 werden 95 Prozent aller Daten in Datenbanken gespeichert, d.h. es gibt keine eigenständigen Programme mehr, sondern nur mehr Applikationsmodule in Datenbanksystemen!"



1.3 Eigenschaften von DBMS

- Persistenz
- Verwaltung von Sekundärspeichern
- Mehrbenutzerfähigkeit
- Zuverlässigkeit
- Datensicherheit
- Ad-hoc Abfragesprachen
- …

Persistenz

Daten bleiben in der Datenbank über die Ausführungszeit von Anwendungsprogrammen hinaus erhalten

- d.h. Daten „überleben“ das Ende von Sitzungen, das Ende von Transaktionen
- Der Gegensatz dazu sind „flüchtige“ (volatile oder transiente) Daten
  - z.B. die Werte in Programmvariablen im Hauptspeicher, die beim Beenden des Programms verschwinden

Daten sind (zum Teil) sehr langlebig, müssen daher gesichert auf Hintergrundspeichern (Sekundär-/Tertiärspeichern) gehalten werden

Daten können „in situ“ aktualisiert werden

Verwaltung von Sekundärspeichern

Verwaltung (sehr) großer Datenmengen

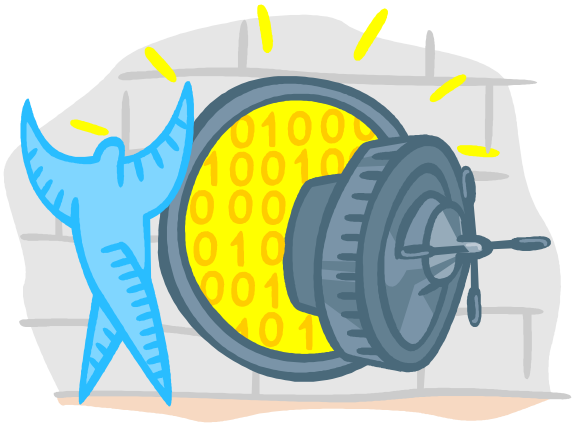
- üblicherweise auf Platten

Datenbanken sind Ein-/Ausgabe-intensiv

Spezifische Techniken zur Erhöhung der Performanz

- Pufferung (DB Puffer im Hauptspeicher)
- Indexierung, Clusterbildung
- Optimierung der Abfragen der Benutzer

Üblicherweise transparent für den Benutzer



Mehrbenutzerfähigkeit

- Mehrere Benutzer können gleichzeitig auf den Daten arbeiten
- DBMS sorgt dafür, dass keine unerwünschten Wechselwirkungen durch gleichzeitige Manipulation derselben Daten eintreten
- Erhaltung der **Integrität**

Bei der nachfolgend angegebenen verzahnten (pseudo-parallelen) Ausführung der beiden Anwendungen geht die Änderung der Anwendung 2 verloren.

Anwendung 1	Anwendung 2
`read(x)`	
	`read(x)`
	`x = x-20`
	`write(x)`
`x = x+10`	
`write(x)`	

Zuverlässigkeit der Daten

Daten sind teuer und strategisch (überlebens-)wichtig - müssen daher zuverlässig sein

DBMS verwendet dazu **Transaktionen**

Eine Transaktion ist eine Folge von Aktionen, wie Lese- und Schreibzugriffe auf Daten der Datenbank, die eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt

DBMS bestätigt jede Änderung

bei Systemfehler: DB-Zustand wiederherstellen, der genau alle bestätigten Änderungen enthält.

roll-backward:

Eliminieren der Auswirkungen aller unbestätigten Transaktionen

roll-forward:

Nachziehen der Auswirkungen aller bestätigten Transaktionen auf Sicherungskopie

Datensicherheit

Schutz vor unberechtigtem Zugriff

- Lesender oder schreibender (verändernder) Zugriff

Berechtigungssystem definieren

- Sicherheitssubjekte (Benutzer, Rollen, etc.)
- Sicherheitsobjekte (Daten)
- Rechte (Lesen, Schreiben - i.e. verändern)
- Weitergabe von Rechten

Zugriff durch Nichtberechtigte verhindern

- bei jedem Zugriff Berechtigungen überprüfen

Ad-hoc Abfragesprachen

- Abfrage von Daten ohne neues, eigenes Programm schreiben zu müssen
  - Interaktiver Betrieb
- Abfragesprachen müssen nicht prozedural sein, sondern erlauben auch deklarativen Zugriff
  - Man beschreibt nur, was man will, nicht wie man es aus dem DBMS „herauszieht“
  - Intuitive Schnittstelle, oft an natürlichen Sprachen orientiert (meist Englisch)
- DBMS unterstützen üblicherweise mehrere Sprachen, wie z.B. SQL, QBE, etc.

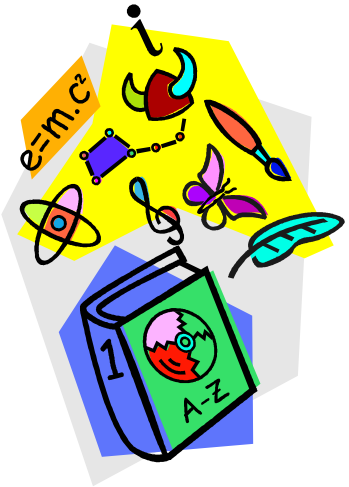
„Wie hoch ist das Durchschnittsgehalt der Manager in den einzelnen Städten in denen mindestens 5 Manager beschäftigt sind?“

```
select city, avg(salaray)
from emp, dept
where emp.deptno = dept.deptno
  and emp.job = "manager"
  group by city
  having count(*) >= 5
```

1.4 Historie und Typen von Datenbanksystemen

1950er und Anfang 1960er

- Sequentielle/r Datenspeicherung und -zugriff waren Standard
- Hauptspeicher waren weitaus kleiner als Datenmengen
- Magnetische Bänder waren die Basis
- Lochkarten wurden auch eingesetzt



Ende 1960er und 1970er

- Einsatz von Festplatten
  - Ende der Sequentialität, Schneller und direkter Zugriff
- **Netzwerk und Hierarchische Datenmodelle**
- Schlüssel-Paper von Codd definiert das **relationale Modell**
  - E.F. Codd, „A Relational Model for Large Shared Data Banks“, CACM, Vol. 13, No. 6, 1970

Historie und Typen von Datenbanksystemen (2)

1980er
- Am Beginn kaum praktische Anwendung des relationalen Modells
  - Annahme großer Leistungseinbrüche
- Bahnbrechendes Projekt **System R von IBM** (Sequel)
- Weitere Projekte
  - IBM DB2, Oracle, Ingres, DEC Rdb
- Wichtige Arbeiten über **parallele und verteilte DBMS**

Anfang 1990er
- Decision Support Anwendungen
- **Objekt-relationale DBMS**
  - Postgres

Ende 1990er
- DBMS mit hohen Transaktionsraten und hoher Verlässlichkeit (24x7)
- Einsatz von DBMS am Web und Grid
- **Objekt DBMS**

2000 and beyond
- Neue Datentypen und -modelle und neue Größenordnung der Datenmengen
- **XML DBMS**, **NoSQL DBMS**, Cloud DBMS

Hierarchisches Datenbankmodell

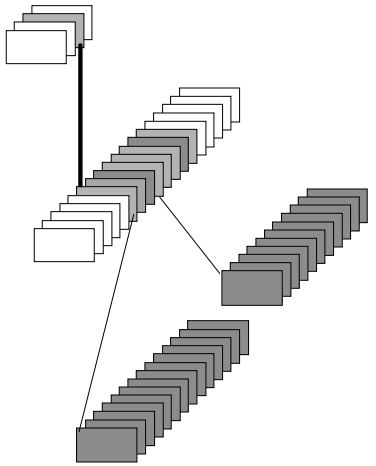
Daten sind in **baumartiger** Struktur organisiert

Einzelne Records sind durch **Links** verbunden

Jeder Record ist ein Sammlung von Feldern, die jeweils einen Wert enthalten

- Ähnlichkeit zu Dateisystemen von vielen Betriebssystemen

Jede DB-Anfrage liefert **maximal einen Record** als Ergebnis zurück



Hierarchisches Datenbankmodell (IMS)

Älteste DBMS-Kategorie: Seit 1968 auf dem Markt

- Populärster Vertreter: IMS/VS - „Information Management System / Virtual Storage“ (IBM)

Veraltet, ist weitgehend von anderen Datenbankmodellen abgelöst worden

Renaissance durch **XML**

Pfad-Ausdrücke

GU Kurs (KursNr = ‘I09’)
Angebot (Ort = ‘München’)
Teilnehmer (TnNr > 100)

Netzwerkdatenbankmodell

Ziel **Objekte** und ihre **Beziehungen** abzubilden

- Verallgemeinerung des hierarchischen Datenbankmodells

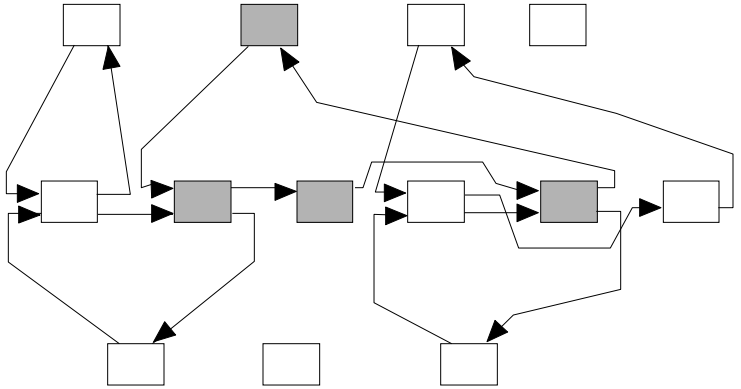
Records und Sets (= Beziehungen)

- **keine strenge Hierarchie**
  - ein Datensatz kann mehrere Vorgänger haben
  - kann dadurch m:n-Beziehungen abbilden

Jede DB-Anfrage liefert **maximal einen Record** als Ergebnis zurück

Geht auf die Normierungsbemühungen von CODASYL (Conference on Data Systems Languages) zurück

- eine Reihe von DBMS-Implementierungen:
  - DMS/100 (UNIVAC)
  - IDMS (Cullinet Software)
  - DBMS (DEC)
  - UDS (Siemens)



Netzwerk-Datenmodell
(CODASYL/DBTG)

navigierende Operationen

FIND ANY	Kurs **RECORD** **USING** KursNr;
FIND FIRST	Angebot **RECORD** **WITHIN** wird_angeb;
GET Angebot **RECORD**	

Semantische Datenmodellierung

Ziel ist die DBMS-unabhängige Modellierung der im IS zu verwaltenden Informationsobjekte und deren Beziehungen

Semantik ist die **Bedeutungslehre von Zeichen**

Semantisch „reichere“ Repräsentation der Information

Dient als Vorgabe für die Umsetzung ins DBMS-Datenmodell

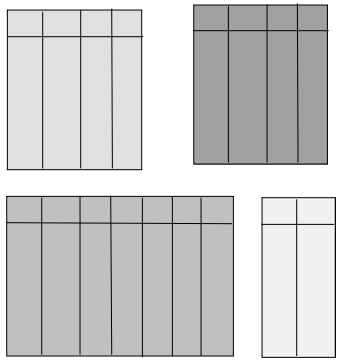
- DBMS-seitig evtl. ergänzende Implementierungen zur Überwachung von spezifizierten Integritätsbedingungen erforderlich

Ur-Vater der semantischen Datenmodelle: **Entity-Relationship-Modell (ERM)**

- Die ERM-Grundkonzepte finden sich praktisch in allen semantischen DM wieder, lediglich die verwendeten Symbole sind z.T. unterschiedlich

Relationales Datenbankmodell

- Daten werden als **Tupel** repräsentiert, die in **Relationen** (Tabellen) gruppiert sind
  - Formale Basis Prädikatenkalkül erster Ordnung
  - 1970 erste Veröffentlichung von E.F. Codd zum „Relationalen Datenmodell“
  - Ähnlichkeit zu **Spreadsheet** Modell
- DB-Anfrage liefert eine **Menge** (Relation) von Tupeln als Ergebnis zurück
- Grundlagenforschung (IBM) in den 1960er um DB-Anwendungen „einfacher“ zu machen
  - Verschiedene Prototyp-Entwicklungen
    - System R, IBM San Jose
    - Ingres, University of California, at Berkeley
- Heute **wichtigste kommerziell verfügbare DB-Technologie**
- Viele relationale DBMS auf dem Markt:
  - DB2, Informix (IBM), Oracle (Oracle),
  - Ingres (CA), SQL-Server (Microsoft),
  - MySQL (OpenSource, Oracle), …



Relationales Datenmodell

Mengen-Operationen

```
SELECT  k.KursNr, k.Titel
FROM    Kurse AS k
        JOIN Angebot AS a
        ON
        k.KursNr = a.KursNr
WHERE   a.Ort = 'Ulm'
```

Aktuelle Anforderungen an Datenbankmanagementsysteme

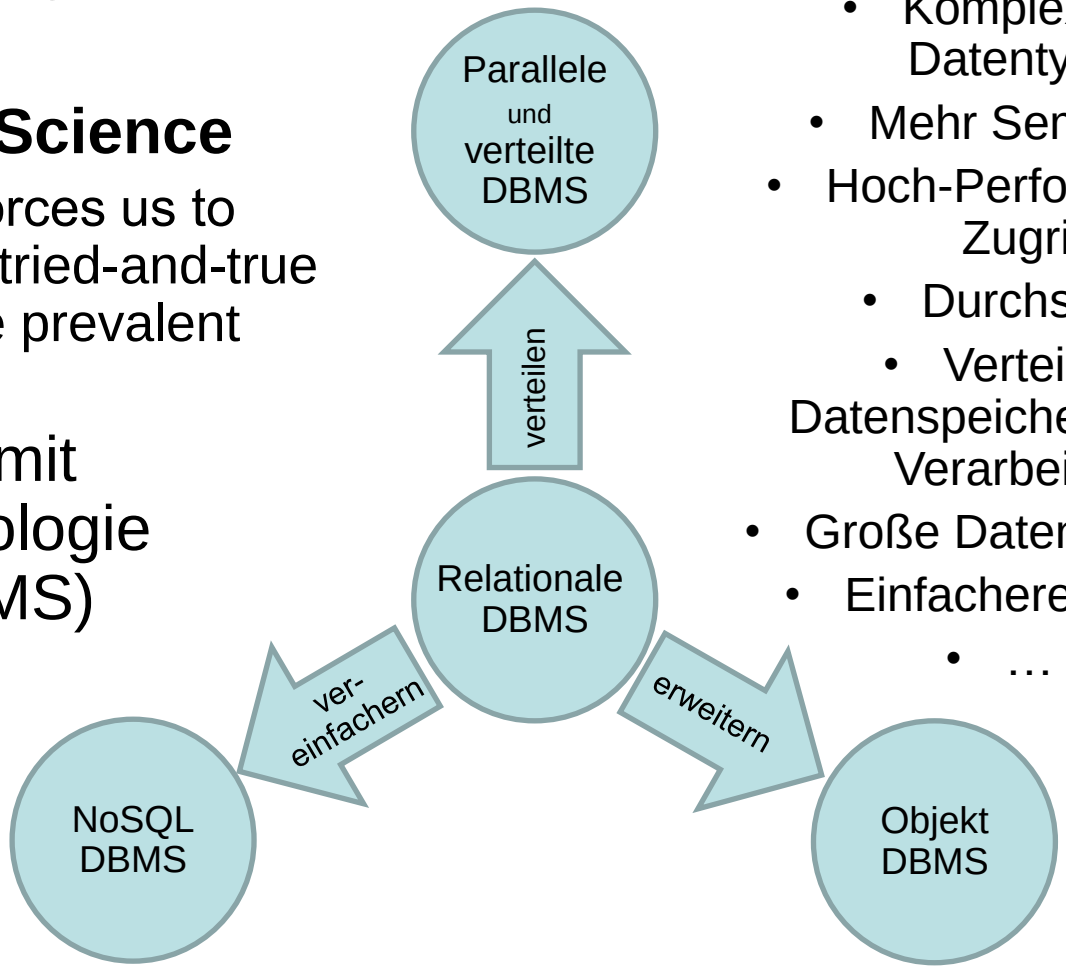
Neue Anwendungsbereiche stellen neue Anforderungen an Datenbanksysteme

Schlagworte

Big Data / Data Science

„data whose size forces us to look beyond the tried-and-true methods that are prevalent at that time“

Datenmengen, die mit Standard-Technologie (relationales DBMS) nicht effizient verarbeitet werden können



- Komplexere Datentypen
- Mehr Semantik
- Hoch-Performanter Zugriff
- Durchsatz
- Verteilte Datenspeicherung und Verarbeitung
- Große Datenmengen
- Einfachere Daten
- …

Paralleles Datenbanksystem

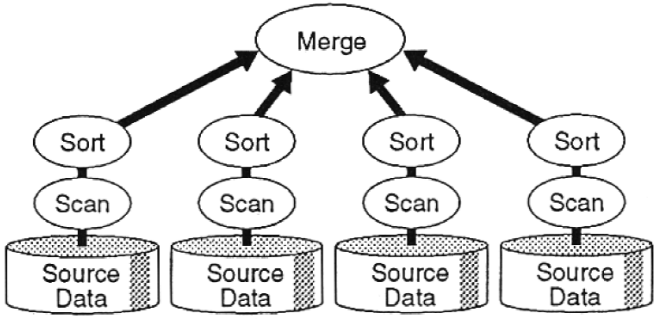
Eng gekoppelte Ausführungseinheiten

Ziel Geschwindigkeitsverbesserung durch parallele Ausführung der Operationen

Eigenschaften

Geografisch zentralisiert, Zentralisierte Administration, Schnelles Interconnect (LAN, high-speed network), Lokale Transaktion

Fokus: **Performanz**



Aus DeWitt, Gray, Parallel Database Systems, CACM 1992

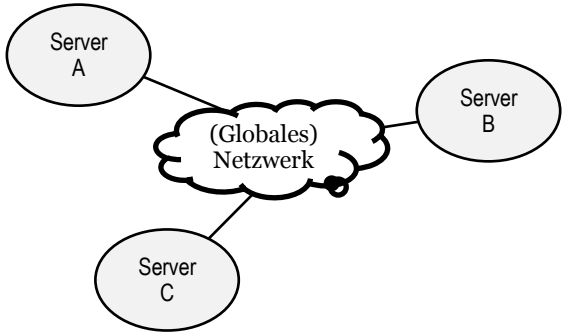
Verteiltes Datenbanksystem

Lose gekoppelte Ausführungseinheiten

Ziel einfacherer Datenzugriff und Effizienz-verbesserung (Administration, Sicherheit, …)

Eigenschaften

Geografisch verteilt, Separierte Administration, Langsames Interconnect (Internet), Zerlegung in lokale und globale Transaktion



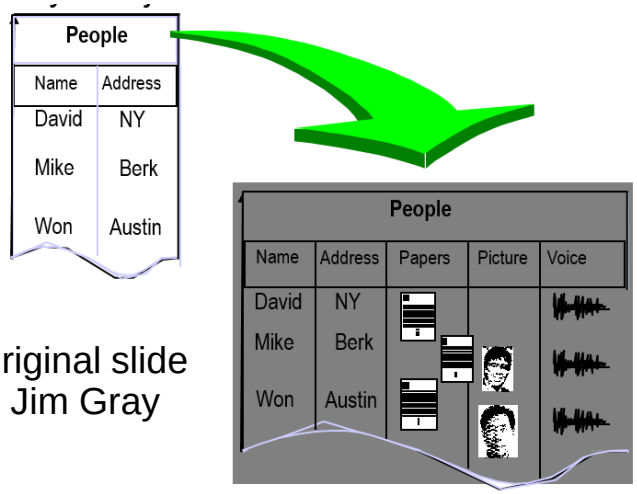
Fokus: **Gemeinsame Nutzung der Daten, Autonomie, Verfügbarkeit**

Objekt DBMS

The Old World

Millionen von Objekten

100-Byte Objekte



Original slide
Jim Gray

The New World

Billionen von Objekten

"Big objects" (>1MB)

Objekte zeigen komplexe Struktur und Verhalten

Führt zum Einsatz von OO Technologien

Objekt DBMS verwalten Daten als Objekte im Sinne der Objektorientierung

Eigenschaften

- Ein Objekt enthält **Daten** und **Methoden** und kann Teil einer **Hierarchie** sein (Generalisierung und Spezialisierung)
- Ein Objekt besitzt eine von seinem Zustand (Daten) unabhängige **Identität**
- Ein Objekt schützt seine Daten durch die **Methoden** (das Verhalten), wie Daten verändert werden können
- Eine objektorientierte Anwendung kann als Netzwerk **interagierender** (über Methoden-Call) **Objekte** gesehen werden

NoSQL Datenbanksysteme

NoSQL = Not only SQL

Alternatives Datenmodell, typische weitere Eigenschaften:

- **Skalierbarkeit** durch verteilte Datenspeicherung
- Speicherung sehr **großer Datenmengen** auf „Commodity Hardware“
- **Schwache/keine Konsistenz**-Bedingungen

Definition http://nosql-database.org/

- Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**.
- Core NoSQL Systems:
  - Key-Value Stores, Wide Column Stores, Document Stores, Graphdatenbanken, …
- Soft NoSQL Systems:
  - Objekt Databanken, Grid&Cloud Databanken, XML Databanken, …

Basis für heutige verteilte Datenspeicherung, z.B. Google, Amazon, …

- Viele Systeme verfügbar
  - Google BigTable, Amazon Dynamo, CouchDB, MongoDB, BaseX, Neo4J, …
  - Im weiteren Sinne auch RDBMS mit XML-Unterstützung: Oracle, DB2, …

Auswahl von gängigen DBMS Produkten

Bezeichnung	*Organisation*	*Erscheinungsjahr*	*Letzte Version*	*Lizenz*
Oracle	Oracle Corporation	Nov. 1979	12c Release 1 (Juni 2013)	Proprietär
DB2	IBM	1982	10.5 (April 2013)	Proprietär
MS SQL-Server	Microsoft	1989	2014 (April 2014)	Proprietär
Informix	IBM	1985	12.10.xC5 (März 2015)	Proprietär
Ingres	Ingres Corp.	1974	Ingres Database 10 (Oktober 2010)	GPL und proprietär
PostgreSQL	PostgreSQL Global Development Group	Juni 1989	9.4.4 (Juni 2015)	PostgreSQL licence (Free and Open Source)
MySQL	Sun Microsystems (Oracle Corporation)	Nov. 1996	5.7.6 (März 2015)	GPL und proprietär
MariaDB	MariaDB Community	Feb. 2010	10.1.7 (September 2015)	GPL v2
MonetDB/SQL	MonetDB Developer Team	2004	11.21.5 (Juli 2015)	MonetDB Public License v1.1

Siehe: http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems