UNIVERSITÄT DES SAARLANDES

Saarbrücken, 21.04.2015
Information Systems Group

Vorlesung „Informationssysteme“

Vertiefung zu Kapitel 1: Motivation

Erik Buchmann (buchmann@cs.uni-saarland.de)

Foto: M. Strauch

Aus den Videos wissen Sie...

- ...das DBMS viele angenehme Eigenschaften mitbringen
  - logische Datenunabhängigkeit, Sichten, Programmierschnittstellen, Anfrageoptimierung, etc.
- ...diese Eigenschaften automatisch zur Verfügung stellt
  - WENN Sie ein paar Dinge beachten, z.B. eine saubere Modellierung

- Vertiefung heute:
  - Welche Eigenschaften genau sind das alles?
  - Warum müssen Sie ein paar Dinge selbst beachten?
  - Brauchen Sie diese Eigenschaften in Zeiten von NoSQL, Multicore-Systemen etc. tatsächlich noch alle?
  - Was gibt es außer Relationalen DBMS noch alles?

Eigenschaften von DBMS

Foto: M. Strauch

Motivation

- Beispiel: Sie möchten einen großen Webshop eröffnen, der Katzenfutter mit Mausgeschmack anbietet
  - mehrere Sorten
    - Trockenfutter, Naßfutter, Rennmaus
  - mehrere Varianten
    - Standard, Bio, Senior
  - tausende Kunden
  - mehrere Mitarbeiter
    - Beratung
    - Versand
    - Einkauf

Bild: Stupipedia, User Radieschen

Redundanzfreie Datenintegration

- **Daten** sind nur einmal gespeichert
- **Funktionen** sind nur einmal implementiert

- Änderungen müssen nur einmal vorgenommen werden, keine Inkonsistenzen durch „vergessen“ von Daten

- Gegenbeispiel:

Bestellübersicht

Kunde	Datum	Bestellung
Erik Buchmann	23.4.14	1x Trocken Standard, 2x Bio Naß
Eric Buchmann	23.04.14	2x Bio Naß, 1x Standard Trocken

Datenbankoperationen

- Erstellen des Schemas
- Einfügen, Ändern, Löschen, Abfragen von Daten

- Datenbank kann das richtig gut (Stichwort: Optimierer)

- Beispiel für eine komplexe Verbund-Anfrage:
  „Finde den Kunden, der in diesem Monat am meisten Umsatz hatte“

Bestellung

Kunde	Bestellung	Datum
Erik	001	15.03.
Jens	002	03.04.
Jens	003	06.04.
Jens	004	10.04.
Erik	005	20.04.

Warenkorb

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Artikel

Artikel	Preis
Trocken	5,76
Naß	6,23
Rennmaus	16,99

Anfrageoperationen in relationalen DBMS

- Selektion: Alle Kunden die mit „E“ anfangen
- Projektion: Nur die Spalte „Kunde“ aus der Relation „Bestellung“
- Verbund: Welcher Artikel gehören zu welcher Bestellung?
- Gruppierung: Welche Bestellungen wurden pro Monat getätigt?
- Aggregate (Min, Max, Avg, Count): Wieviele Artikel enthält die DB?
- Sortierung: Sortiere Bestellungen nach Menge

Bestellung

Kunde	Bestellung	Datum
Erik	001	15.03.
Jens	002	03.04.
Jens	003	06.04.
Jens	004	10.04.
Erik	005	20.04.

Warenkorb

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Artikel

Artikel	Preis
Trocken	5,76
Naß	6,23
Rennmaus	16,99

Datenbank-Katalog

- Physische Datenunabhängigkeit

- Datenbankoperationen sind **deklarativ**, d.h., Anwender formuliert Informationsbedürfnis unabhängig von Ort und Art der Datenspeicherung
- Speicherstrukturen dürfen sich ändern, ohne dass man die Anwendungen anpassen müsste
- DBMS entscheidet über Anfrageausführung

- Beispiel: Finde die drei Bestellungen mit der größten Bestellmenge
  - Daten als Array oder Liste gespeichert?
  - Existiert ein Index auf die Daten?
  - Mit welchem Algorithmus findet man das Maximum?

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Benutzersichten

- Logische Datenunabhängigkeit

- Benutzer und Anwendungen sehen nur, was sie sehen sollen
- Änderungen am Schema, ohne die Anwendungen anzupassen

- Beispiel

Artikel (1)

Artikel	Preis pro 100g	Packung
Trocken	2,88	200g
Naß	2,06	300g
Rennmaus	16,99	100g

Artikel (2)

Artikel	Preis pro Dose	Packungs-inhalt
Trocken	5,76	1
Naß	3,09	2
Rennmaus	16,99	1

Sicht

Artikel	Preis
Trocken	5,76
Naß	6,18
Rennmaus	16,99

Integritätssicherung (1/2)

- DBMS soll sicherstellen, dass die Daten in sich schlüssig sind
  - Fehlermeldung möglichst schon beim Eintragen falscher Daten
- Relationenschema + lokale Integritätsbedingungen
  - „Bestellung“ in *Bestellung* und „Artikel“ in *Artikel* taucht nur einmal auf
  - „Menge“ in *Warenkorb* und „Preis“ in *Artikel* ist >0
  - „Datum“ in *Bestellung* ist > 01.01.2015

Bestellung

Kunde	Bestellung	Datum
Erik	001	15.03.
Jens	002	03.04.
Jens	003	06.04.
Jens	004	10.04.
Erik	005	20.04.

Warenkorb

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Artikel

Artikel	Preis
Trocken	5,76
Naß	6,23
Rennmaus	16,99

Integritätssicherung (2/2)

- DBMS soll sicherstellen, dass die Daten in sich schlüssig sind
  - Fehlermeldung möglichst schon beim Eintragen falscher Daten
- Datenbankschema + globale Integritätsbedingungen
  - zu jeder „Bestellung“ in *Bestellung* existieren ein oder mehrere Einträge in *Warenkorb*
  - zu jeder „Bestellung“ in *Warenkorb* existiert exakt ein Eintrag in *Bestellung*

Bestellung

Kunde	Bestellung	Datum
Erik	001	15.03.
Jens	002	03.04.
Jens	003	06.04.
Jens	004	10.04.
Erik	005	20.04.

Warenkorb

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Artikel

Artikel	Preis
Trocken	5,76
Naß	6,23
Rennmaus	16,99

Zugriffsschutz

- sollte selbsterklärend sein...

Transaktionen

- Zusammenfassung von Operationen zu einer Einheit, die logisch zusammengehören
  - Ausführung entweder ganz oder gar nicht,
    ggf. zurücksetzen von unvollständigen Transaktionen

- Beispiel: Bei Transaktion „Bob“ ist Konto nicht gedeckt

Op.	Alice	Bob
1	subtrahiere 10x Nassfutter von „Lager“	subtrahiere 5x Nassfutter von „Lager“
2	buche 100 EUR vom Konto des Kunden ab	buche 20 EUR vom Konto des Kunden ab
3	addiere 100 EUR zu „Verbindlichkeiten“	
4	setze Lieferstatus auf „erfüllt“	

Rücksetzen der Transaktion

Synchronisation

- Parallele Transaktionen dürfen sich nicht gegenseitig beeinflussen
- Beispiel:
  zwei parallele Transaktionen „Alice“ und „Bob“ auf der gleichen DB

Dirty Reads

Alice	Bob
X := 10	
schreibe X	
	lese X X := X * 5
Abbruch der Transaktion	
	schreibe X

Phantoms

Alice	Bob
schreibe X	
schreibe Y	
schreibe Z	zähle Daten-objekte
lösche Y	

Lost Updates

Alice	Bob
	lese X
lese X	
X := X+1 **schreibe X**	**schreibe X**

Warum Transaktionen nicht seriell ausführen?

Datensicherung

- Als abgeschlossen („Commit“) an den Nutzer gemeldete Transaktionen sind **persistent** in der DB gespeichert

- Das ist *weitaus* weniger leicht als es sich anhört ;-)
  - Schreiboperationen landen üblicherweise erst im Datenbank-Cache, dann im Betriebssystem-Cache, dann im Cache der Festplatte
  - Rechner kann während einer Schreiboperation abstürzen
  - Aus Performanzgründen arbeiten DBMS mit größeren Speicherseiten, d.h., Lesen und Schreiben erfolgt Seiten- und nicht Datensatz-weise
  - Transaktionskonzept erzwingt das Führen von Logs (Query-Log, Back-Log)
  - Anpassen von Indexstrukturen

Auf einen Blick zusammengefasst

- **Integration,** redundanzfreie Verwaltung des Datenbestands über Anwendungsgrenzen hinweg
- **DB-Operationen** wie Speichern, Suchen, Ändern
- **Katalog**, physische Datenunabhängigkeit
- **Benutzersichten,** logische Datenunabhängigkeit
- **Integritätssicherung**, Sicherstellen der Korrektheit der DB
- **Zugriffsschutz** vor unautorisierten Zugriffen
- **Transaktionen,** mehrere DB-Operationen als Einheit
- **Synchronisation**, parallele Transaktionen koordinieren
- **Datensicherung,** Wiederherstellung von Daten nach Systemfehlern

Andere Ansätze

Foto: M. Strauch

Im Fokus dieser Vorlesung

- Relationale DBMS
  - Daten in Tabellenform, genauer als Sets von Tupeln, die jeweils eine Attributmenge enthalten

Bestellung

Kunde	Bestellung	Datum
Erik	001	15.03.
Jens	002	03.04.
Jens	003	06.04.
Jens	004	10.04.
Erik	005	20.04.

Warenkorb

Bestellung	Artikel	Menge
001	Trocken	2
001	Naß	2
002	Naß	10
003	Trocken	5
003	Rennmaus	3
004	Naß	1
005	Naß	100

Artikel

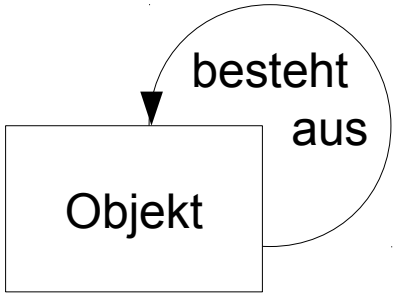
Artikel	Preis
Trocken	5,76
Naß	6,23
Rennmaus	16,99

- Aber:
  - nicht alle Daten lassen sich gut als Relationen abbilden
  - nicht alle Daten lassen sich sinnvoll in Relationen verarbeiten

Wofür sind relationale DBMS weniger geeignet?

- Hierarchien und Graphen
  - komplexe Beziehungen zwischen Objekten
  - verschachtelte Objekte mit einer Vererbungshierarchie
- datenintensive Anwendungen mit einfachem Datenschema
  - Schlüssel-Wert-Paare, viele Big Data-Probleme
- semistrukturierte Daten
  - XML-Dokumente und „schemafreie“ Daten
- umfangreiche Binärdaten
  - im Sinne von Daten, die das DBMS nur speichern, aber nicht interpretieren kann, z.B. verschlüsselte Daten, Videos oder MS-Word-Dokumente
- Datenströme

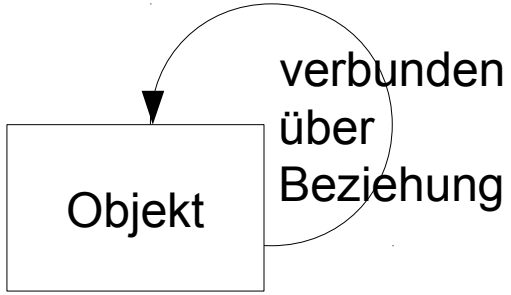
Hierarchische Daten



- Baumstruktur mit Vater-Kind-Beziehungen, keine vorab bekannten Instanzen
  - objektorientierte Programmiersprachen
  - Klassifikationshierarchien, Taxonomien
  - ...
- Ineffizient in relationalen DBMS
  - Finde alle Inhaltsstoffe in der Bestellung mit ID = 1
- Alternative Systeme
  - XML-Datenbanken, Graph-Datenbanken Breiten- und Tiefensuche ohne zahllose Iterationen über die selbe Tabelle
  - Objektorientierte Erweiterungen relationaler DBMS

ID	Parent	Objekt
1	NULL	Bestellung
2	1	Karton
3	1	Karton
4	1	Karton
5	2	Trockenfutter
6	5	Mausaroma
7	5	Karotten
8	5	Flugasche
9	2	Beipackzettel
10	1	Lieferschein
...	...	...

Graph-Daten



- Graphstruktur bildet beliebige Beziehungen zwischen Objekten ab
  - soziale Netzwerke
  - Wissensmanagement
  - Ontologien
  - ...

Objekt	Beziehung	nach
Datenbanken	ist Lehrstoff von	Informationssysteme
Informationssysteme	wird gelehrt an	UdS
Uds	beschäftigt	Erik B.
Erik B.	lehrt	Informationssysteme
Erik B.	hat gearbeitet in	KIT
Datenbanken	ist Lehrstoff von	Database Systems
Database Systems	wird gelehrt an	KIT

- Ineffizient in relationalen DBMS
  - Welche Beziehungen bestehen zwischen UdS und KIT?
- Alternative Systeme
  - XML-Datenbanken, Graph-Datenbanken kürzester Pfad, Breiten- Tiefensuche

Schlüssel-Wert-Paare

- Einfache Struktur, große Datenmengen
  - Human Genome Project
  - Teilchenbeschleuniger-Daten vom CERN
  - …
- Idee: in den Schlüssel einkodierte Zugriffsmuster (vgl. Hashtable)
  - Beispiel: Datum im Schlüssel einkodiert, Anfrage: Alle Bestellungen im April
- Ineffizient in relationalen DBMS
  - die meisten der schönen Eigenschaften werden nicht gebraucht
- Alternative Systeme
  - Key-Value-Stores (Hashtable-Interface)

Schlüssel	Wert
00001	Preis Naßfutter: 6.23
00002	Preis Trockenfutter: 5.76
150315	Bestellung Erik 2x Trockenfutter
150403	Bestellung Jens 10x Naßfutter
150506	Bestellung Jens 5x Trockenfutter
150510	Bestellung Jens 3x Rennmaus
150520	Bestellung Erik 100x Naßfutter
...	...

Semistrukturierte Daten

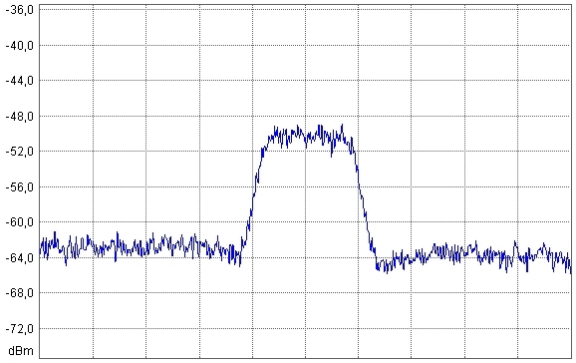
- keine formale Struktur, Strukturinformationen in den Daten eingebettet
  - Typisch für XML und HTML
- Ineffizient in relationalen DBMS
  - Typische Anfragen hier:
    Finde einen Vaterknoten, dessen Kindknoten bestimmte Merkmale aufweisen
    z.B. Welche Bestellung enthält sowohl Naßfutter als auch Trockenfutter
- Alternative Systeme
  - XML-Datenbanken,
    Anfragesprache XQuery

```
<XML>
 <Bestellung>
  <Karton>
   <Trockenfutter>
    <Inhaltsstoffe>
     </Mausaroma>
     </Karotten>
     </Flugasche>
    </Inhaltsstoffe>
    <Menge>1</Menge>
   </Trockenfutter>
   <Naßfutter>
    <Menge>5</Menge>
   </Naßfutter>
   </Beipackzettel>
  </Karton>
  </Lieferschein>
 ...
 </Bestellung>
</XML>
```

Umfangreiche Binärdaten

- nichts, was ein relationales DBMS interpretieren könnte
  - Multimedia-Daten wie Video, Audio
  - proprietäre Dokumentformate, Textverarbeitung etc.
  - verschlüsselte Daten
- Ineffizient in relationalen DBMS
  - passen nicht gut zur Speicherstruktur (→ Speicherseiten) von DBMS
  - einzige Operationen hier sind schreiben, lesen, löschen
- Alternative Systeme
  - Multimedia-Datenbanken und ähnliche Spezialisten

Datenströme



- Daten, die kontinuierlich produziert werden
  - Videosignale einer Kamera
  - Meßwerte in einem Sensornetzwerk
  - Daten aus einem digitalen Stromzähler
- Ineffizient in relationalen DBMS
  - Datenströme haben potentiell unendliche Länge
  - passen nicht gut zur Speicherstruktur
    Datenströme: einfügen durch anhängen ans Ende
- Alternative Systeme
  - Data Stream Management Systems
    - sequentieller Zugriff, kontinuierliche Abfragen, Einfügen durch Anhängen ans Ende, „Veralten“ von Daten

Zeit	Wert
10:00	34
10:01	35
10:02	31
10:03	30
10:04	33
10:05	33
20:06	33
...	...

Zum Abschluss

Foto: M. Strauch

Wie geht es weiter?

- nächste Video-Einheit
  - Einführung in das Entity-Relationship-Modell
- bis Montag, 27.04., 12 Uhr
  - Bearbeitung der Quizzes
    - Thema 1: Einführung
    - Thema 2: Entity-Relationship-Modell
- Dienstag, 28.04., GHH 12-14 Uhr
  - Tutorium: Organisatorisches
  - Vorstellung der Themen, Zusammenstellung der Gruppen
- Donnerstag, 30.04., GHH 10-12 Uhr
  - Erweiterungen des Entity-Relationship-Modells