Datenbanken
Einführung

Prof. Dr. Ludger Martin

Gliederung

- Einführung
- Erste Definitionen
- Datenbank-Generationen
- Relationale Datenbanken
- Objektrelationale Datenbanken
- NoSQL Datenbanken
- Aufbau von relationalen Datenbanken und DB-Systemen

Einführung

Datenbanken sind:

- Entstanden in 60er Jahren
- Hilfsmittel zur effizienten, rechnergestützten Organisation, Speicherung, Manipulation, Integration und Verwaltung großer Datensammlungen
- Vom Anwendungsprogramm unabhängiges Betriebsmittel

Einführung

Funktionen von Datenbanken:

- Speicherung und Verwaltung großer Datenbestände
- Dauerhaft
- Frei von Redundanzen
- Plausibilitäts- und Konsistenzbedingungen
- Zugriffsschutz
- Anfragen an Daten
- Aktualisierungen von Daten

Einführung

Forderungen an Datenbanken:

- Effizienter und schneller Zugriff
- Gleichzeitiger Zugriff vieler Benutzer
- Netzwerkzugriff
- Zugriff auf mehrere Datenbanken (Schemen)

Erste Definitionen

Ziel: Abhängigkeit zwischen Daten und darauf operierenden Programmen aufzuheben bzw. einzuschränken

- Ein *Datenbanksystem* (DBS) besteht aus einer Software, dem *Datenbankmanagementsystem* (DBMS) und einer Anzahl von *Datenbanken* (DB)
- DBMS wird (zumindest teilweise) im Hauptspeicher gehalten und steht unter Kontrolle des Betriebssystems
- Daten in der Datenbank werden (aufgrund der Größe) im Sekundärspeicher abgelegt
- DBMS ist Schnittstelle zwischen Benutzer und Daten; komfortabler, effizienter Zugriff, zentralistische Kontrolle

Datenbank-Generationen

- **Filesysteme auf Band**
  - 50er Jahre
  - Programmgesteuerte Verarbeitung von Daten
  - Lochkarten oder Magnetbänder als Speicher
  - Ausschließlich sequentieller Zugriff

Datenbank-Generationen

- **Filesysteme auf Platte**
  - 60er Jahre
  - Batch und Dialogverarbeitung
  - Wahlfreier- und Mehrfachzugriff auf Dateisystem
  - Dateisystem als Vorstufe von DB-System
  - Ein Programm benötigt Struktur von Daten, um sie lesen oder schreiben zu können
    - Hohe Redundanz, mehrfache Kopien bei Mehrfachzugriff
    - Inkonsistenz von Änderungen eines Programms
    - Unflexible/aufwendige Programmwartung, bei Änderungen müssen alle Programme angepasst werden
    - Datenschutzprobleme, es kann nicht kontrolliert werden, wer auf Daten zugreift

Datenbank-Generationen

- **Netzwerk- und hierarchische Datenbanken**
  - 70er Jahre
  - Erste echten Datenbanken
  - Unterscheidung von logischen und physischen Daten
  - Verwaltung von Daten (DBMS)
  - Datenmodelle aus Bäumen oder aus Graphen
    - Knoten – Record-Typen
    - Kanten – Beziehungen

Relationale Datenbanken

- 80er Jahre
- Klarere Unterscheidung zu physischen und logischen Datenmodell
- In Grundzügen bis heute erhalten (de-facto-Standard)
- Einfaches konzeptionelles Modell (Relationen bzw. Tabellen)
- Physische Speicherung nach außen unsichtbar, Benutzer muss sich nicht darum kümmern
- Abfrage durch Structured Query Language

Relationale Datenbanken

- Daten werden in Tabellen gespeichert

buch

invnr	**autor**	**titel**	**verlag**	**jahr**
027-2408	Jones	Algorithms	PH	2003
188-2887	Jameson	Web Design	JO	2006

entleihen

invnr	**lesernr**	**datum**
027-2408	428456	2006-09-30

leser

lesernr	**name**	**telefon**
428456	Andrews	07-8446524

  - Beziehungen zwischen Tabellen
  - Operationen zum Zugriff auf Tabellen

Relationale Datenbanken

- **Relationale Datenbanken** (Teil 3)
  - Vorteile:
    - Redundanzfreie Einmalspeicherung
    - Mehrbenutzerzugriff
    - Einfach strukturierte Datenobjekte (record-orientiert)
    - Einfache Datentypen (identisch zu höheren Programmiersprachen)
    - Kurze, wiederkehrende Auftragsarten (Anfragen oder Updates)
    - Hohe Transaktionsraten

→ In dieser Vorlesung betrachtet

Objektrelationale Datenbanken

- 90er Jahre
- Um objektorientierte Konzepte erweiterte relationale Datenbanksysteme
- Objekte bestehen aus Struktur und Verhalten
- Jedes Objekt hat ein künstliches identifizierendes Merkmal (Objektidentifikator)
- Speicherung von Objekten und nicht nur von Records
- Abfrage durch Object Query Language

Objektrelationale Datenbanken

- Komplexere Datentypen (picture, audio, video, ...)
- Neue Datentypen können erzeugt oder eingefügt werden
- Lange andauernde Transaktionen
  → andere Mehrbenutzermechanismen
- Haben sich nicht durchgesetzt, da relationale DB viel mächtiger waren.
- Persistenz-Frameworks ermöglichen die automatische Abbildung von Objekten in relationalen Datenbanken

NoSQL Datenbanken

- Ab1998
- Haben keine SQL-Schnittstelle
- Sind mit dem Internet gewachsen.
- Für sehr große Datenmengen (z.B. mehrere Petabyte)
- SQL-Datenbanken legen Wert auf Integrität und Transaktionen und haben dadurch Schwierigkeiten mit sehr großen Datenmengen.

NoSQL Datenbanken

- Eigenschaften von NoSQL Datenbanken
  - Datenbankmodell ist nicht relational.
  - Der Fokus liegt auf verteilte und horizontale Skalierbarkeit.
  - Haben schwache oder keine Schemarestriktionen.
  - Datenreplikation ist einfach.
  - Einfacher Zugriff über ein bereitgestelltes API.

NoSQL Datenbanken

- Key-Value Stores
  - Weisen einen Wert einem Schlüssel zu.
  - Ähnlich zu einem assoziativen Array.
  - Sind schemalos und erlauben keine Struktur, Verschachtelung oder Referenzen.
  - Einfache API (z.B. `sessionStorage` bzw. `localStorage` im Browser)
    - `getItem(string key)` – Wert holen
    - `setItem(string key, string value)` – Wert schreiben

NoSQL Datenbanken

- Key-Value Stores (Fortsetzung)
  - Sind performant und leicht zu partitionieren.
  - Sind flexibel in den Typen von Werten.
  - Das fehlen eines Schemas verursacht oft ein Durcheinander im Datenmanangement.
  - Eine DB ist ein Key-Value Store, wenn:
    - Werte durch Schlüssel identifiziert werden.
    - Für jeden Schlüssel existiert genau ein Wert.
    - Bei Angabe eines Schlüssel kann der dazugehörige Wert abgefragt werden.

NoSQL Datenbanken

- Document Stores
  - Sind schemalos unterstützen keine referentielle Integrität.
  - Aber erlauben Strukturierung von Daten.
  - Sind ähnlich zu Key-Value Stores, zu jeder Document-ID (`_id`) kann ein Datensatz (Document) gespeichert werden.
  - Jedes Dokument kann eine eigene interne Struktur (z.B. JSON) haben.
  - Jedes Dokument hat eine Version (`_rev`).
  - Dokumente sind nicht miteinander verknüpft.

NoSQL Datenbanken

- Document Stores (Fortsetzung)
  - Geeignet für horizontale Skalierung
    - Jede Anfrage bekommt das Dokument mit der größten Versionsnummer.
    - Eventual Consistency
    - Konsistenz erst nach einer gewissen Zeit erreicht.
  - Implementierungen: MongoDB, CouchDB, ...

NoSQL Datenbanken

- XML-Datenbanken
  - XML Version 1.0 und 1.1
  - XML-Daten sind selbstbeschreibend (Daten mit Attributnamen).
    - Document Type Definition
    - XML Schema
  - XML-Dokumente sind hierarchische Daten (Baum).
  - XML-Daten können vom System interpretiert werden.
  - XML-Dokumente sind beliebig tief strukturiert.

NoSQL Datenbanken

- XML-Datenbanken (Fortsetzung)
  - Eigene Abfragesprachen
    - XPath – lokalisieren von Teilen in einem Quelldokument ausgehend von einem *current node*
      Beispiel: `descendant::vorname/text()`
    - XQuery – Erweiterung von XPath
  - Implementierungen: BaseX, Oracle Berkeley DB XML, ...

Aufbau von Datenbanken und Datenbanksystemen

- Daten einmal und dauerhaft gespeichert (frei von Redundanzen, über Programmende hinaus).
- Verwaltung der Datenbank obliegt einer zentralen Kontrollinstanz (Veränderungen und Aktualisierungen ohne Inkonsistenzen).
- Alle Benutzer/Programme können zeitgleich auf gleiche Daten zugreifen (mit Zugriffskontrolle und Sicherungskopien).
- Unterschiedliche logische Sichten für einzelne Benutzer auf Daten → Datenunabhängigkeit

Aufbau von Datenbanken und Datenbanksystemen

- **Datenmodell**
  - Konzepte für Struktur einer Datenbank
  - Datentypen, Beziehungen und Bedingungen
  - Menge von Operatoren (für Anfragen und Updates)
- *High-level* bzw. *konzeptionelle* Modelle: Abstraktionskonzepte nahe am Benutzer
- *Low-level* bzw. *physikalische* Modelle: Beschreibung wie Daten im Rechner gespeichert werden

Aufbau von Datenbanken und Datenbanksystemen

- Beschreibung einer Datenbank wird *Datenbankschema* genannt.
- Ein Schema wird während der Lebenszeit einer Datenbank nur selten geändert.
- Ein Schema beschreibt Objekte bzw. *Entitäten* einer Anwendung über die Daten gespeichert werden sollen
  - Mit Eigenschaften
  - Mit Bedingungen
  - Mit Beziehungen

Aufbau von Datenbanken und Datenbanksystemen

- DBMS kommuniziert mit Außenwelt mit Schnittstellen bzw. Sprachschnittstellen.
- Die Sprachen sind:
  - *Data Definition Language* (DDL)
  - *Data Manipulation Language* (DML)
  - *Data Administration Language* (DAL)
  - Meist vereint in SQL
    - Anfragen (Queries)
    - Veränderungen (Einfügen, Ändern, Löschen – Updates)
- Für DML gibt es meist eine Schnittstelle zu einer höheren Programmiersprache.

Aufbau von Datenbanken und Datenbanksystemen

- DML Sprachen sind meist einfacher und haben i.d.R. keine Kontrollstrukturen (Schleifen usw.).
- Beispiele:
  - Zeige die Titel aller Bücher von Autor Jones.
    ```
    SELECT titel FROM buch WHERE autor='Jones';
    ```
  - Zeige eine Liste aller Verlage, von denen Bücher vorhanden sind. `SELECT verlag FROM buch;`
- Mit Benutzerrechten kann geregelt werden, welche Sprachteile (DDL, DML oder DAL) und Tabellen genutzt werden dürfen.

Aufbau von Datenbanken und Datenbanksystemen

- Schritte bei einer SQL-Ausführung eines DBMS

```
SELECT titel FROM buch WHERE autor='Jones'
```

1. Überprüfung der Syntax.
2. Feststellen, ob Tabelle in DB definiert ist und ob Benutzer Informationen lesen darf.
3. Welche Operationen müssen intern zur Beantwortung ausgeführt werden.
4. Erstellung eines (effizienten) Programms zur Berechnung der Antwort.
5. Holen der Tabelle aus der Datenbank.
6. Aufbereiten der Tabelle zwecks Ausgabe.
7. Sicherstellen, dass Tabelle nicht zeitgleich durch anderes Programm verändert werden kann.

Literatur

- Vossen, Gottfried: Datenmodelle, Datenbank-sprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- Kudraß, Thomas: Taschenbuch Datenbanken, Hanser, 2007
- Meier, Andreas und Michael Kaufmann: SQL & NoSQL Databases, Springer 2019
- M. Heiderich, C. Mattheis, J. Dahse und fukami: Sichere Webanwendungen, Galileo Press, 2009
- Oracle: MySQL 8.0 Reference, https://dev.mysql.com/doc/refman/8.0/en/