TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS13/14

Henrik Mühe (muehe@in.tum.de)
http://www-db.in.tum.de/teaching/ws1314/dbsys/exercises/

Blatt Nr. 1

Gruppenaufgabe 1 (nicht Zuhause vorbereiten)

Sie designen ein Tool zur Univerwaltung, welches Sie aus gegebenen Anlass *TUMoffline* nennen wollen. Früh entschließen Sie sich zum Einsatz eines Datenbanksystems als Backend für Ihre Daten. Ihr Kollege ist skeptisch und würde die Datenverwaltung lieber selbst implementieren. Überzeugen Sie ihn von Ihrem Entschluss. Finden Sie stichhaltige Antworten auf die folgenden von Ihrem Kollegen in den Raum gestellten Äußerungen:

a) Die Installation und Wartung eines Datenbanksystems ist aufwendig, die Erstellung eines eigenen Datenformats ist straight-forward und flexibler.

b) Mehrbenutzersynchronisation wird in diesem Fall nicht benötigt.

c) Es ist unsinnig, das jeder Entwickler zunächst eine eigene Anfragesprache (SQL) lernen muss, nur um Daten aus der Datenbank zu extrahieren.

d) Redundanz ist hilfreich, wieso sollte man auf sie verzichten?

a) Eine Datenformate sind inhärent unflexibel, da es keinen standartisierten Pfad zur Erweiterung, Verteilung, Recovery etc.pp. gibt. Man bedenke beispielsweise, dass allein viele Dateisysteme keine Dateien über einer fixen Größe, bei FAT32 beispielsweise traditionell 2GB erlauben. Hinzu kommt, dass durch die manuelle Erstellung von Dateiformaten keine standardisierten Datentypen verwendet werden und das gesamte "Schema" der Datenspeicherung leicht uneinheitlich wird. Im Vergleich dazu ist der Aufwand zu Erstinstallation einer Datenbank vernachlässigbar, insbesondere, da heutzutage nicht zwangsläufig ein komplexes Produkt wie IBM DB2[a] o.Ä. verwendet werden muss sondern man für kleine Eigenentwicklungen auch durchaus eine eingebettete Datenbank wie etwa sqlite[b] verwendet werden kann.

b) Mehrbenutzersynchronisation ist inhärent notwendig, wenn sie ein System entwickeln, auf das mehrere Personen zugreifen. Insbesondere ist diese eine der Eigenschaften, die nicht einfach "nachgepatched" werden kann, sondern sehr tief in eine Datenverwaltungsschicht integriert werden muss. Datenbanksysteme erlauben es, ohne über Nebenläufigkeit nachdenken zu müssen auf Daten zuzugreifen und das "erwartete" Ergebnis zu erhalten. Siehe dazu das ACID Paradigma[c], was i.A. von DBMS erfüllt wird.

c) Ein eigenes Datenformat und dessen API (wenn es denn zumindest eine API für den Zugriff gibt) muss auch gelernt werden, dafür ist SQL standardisiert und kann auch beim Wechsel des DBMS weiterverwendet werden.

d) Redundanz sorgt auch für Anomalien, etwa beim Updaten von Daten.

[a] `http://www-01.ibm.com/software/data/db2/`
[b] `http://www.sqlite.org/`
[c] `http://en.wikipedia.org/wiki/ACID`

Hausaufgabe 1

Schätzen Sie die Größe der Datenbank von Facebook ab. Ermitteln Sie hierzu möglichst aktuelle statistische Werte über die Zahl der Nutzer, Fotos, Videos etc.pp auf Facebook und schätzen Sie fehlende Informationen geeignet ab. Nutzen Sie dann ihre Kenntnis über den Speicherverbrauch von Daten in einem Computersystem, um die "Datenbankgröße" von Facebook zu ermitteln.

a) Geben Sie die abgeschätzte Größe sowie Ihren Rechenweg an.

b) Welche Faktoren beeinflussen diese Größe. Denken Sie an Metadaten, zusätzlich generierte Informationen, Backups, Redundanz etc.pp.

a) Beispiel aus Gruppe 5:

```
Fotos: 100 Mrd. * 90 kB = 9 PB
Videos (hochgeladen): 960 Mio. * 20MB = 19,2 PB
Gruppen (1000 Zeichen / Gruppe): 900 Mio. * 4 KB = 3,6 TB
User: 800 Mio * 4KB (Beschreibung) = 3,2 TB
Freundesreferenzen (ca. 130 Freunde): 800 Mio * 130 * 4 byte = 416 GB
Postings (ca. 80/User): 800 Mio * 80 *  400 byte = 25 TB
Insgesamt: 28.232216 PB
```

b)
- Beispielsweise speichert man komplexe Anfragen und Zwischenergebnisse materialisiert, so etwa die Recommendations für Freunde.
- Wiederum verteilt Facebook die Daten auf sehr viele Server und hält mehr als eine Kopie, um bei einem Systemausfall direkt noch alle Daten zur Verfügung zu haben. Traditionell hat man dies nicht gemacht (früher, bei anderen Firmen), sondern einen ausgefallenen Server aus Logs rekonstruiert. Dies ist heute allerdings sinnlos. Warum?
- Facebook speichert nicht nur komplexe Aggregate sondern auch einfachere kompakte Resultate. Hierzu wird ein verteilter Hauptspeichercache verwendet. Alle dort gespeicherten Daten sind wiederum mehrfach vorhanden und werden primär für die Anfragebeantwortung verwendet.
- Mehrere Versionen von Bildern für verschiedene Größen und in verschiedenen Formaten für unterschiedliche Endgeräte etc.pp.

Hausaufgabe 2

Erläutern Sie den Unterschied zwischen dem Relationalen Modell und dem Graphstrukturieren Modell.

a) Nennen Sie ein typisches Einsatzgebiet für das jeweilige Modell.

b) Im Datenbankbereich unterscheidet man zwischen Modellen, welche ein festes Schema voraussetzen und anderen, die kein Schema benötigen. Ein Schema ist hierbei eine Vorgabe, wie Daten repräsentiert werden, beispielsweise, das jede Vorlesung genau eine eindeutige Nummer hat, einen Namen von weniger als 200 Zeichen und eine Semesterwochenstundenzahl.

   1) Was ist der Vorteil/Nachteil einer solchen Vorgabe für den Anwender?

2) Was ist der Vorteil/Nachteil einer solchen Vorgabe für den Entwickler des Datenbanksystems?

a) Relationales Modell: Ein CMS. Graphstrukturiertes Modell: Datenspeicherung in der Bioinformatik/Medizin, Speicherung von inhärent graphstrukturierten Daten wie etwa U-Bahn Netze ;-)

b) 1) Schema sorgt für sauber und einheitlich abgelegte Daten und bietet Garantien über die Vollständigkeit und das Format der vorhanden Daten. Es geht Flexibilität verloren, da neue Daten, die abgelegt werden müssen, oft eine Schemaänderung notwendig machen.

2) Ein Schema erlaubt i.A. mehr Optimierungsmöglichkeiten da mehr Informationen und dadurch mehr "Constraints" über die Daten bekannt sind. Das Datenbanksystem kann so Anfragen schneller bearbeiten.

Gruppenaufgabe 2 (nicht Zuhause vorbereiten)

Finden Sie ein Beispiel für ein Problem (bzw. eine Inkonsistenz), die auftreten kann, wenn unkontrolliert parallel auf Daten zugegriffen wird. Ein traditionelles Beispiel hierfür ist eine gegenseitige Bank-Überweisung zwischen zwei Konten A und B. Wenn A einen Betrag x zu B überweist und B einen Betrag x' zu A, sollte immer gelten $Kontostand(A) + Kontostand(B)$ ist konstant, da sonst Geld verschwunden ist. Konstruieren Sie einen Ablauf zweier gegenseitiger Überweisungen, bei dem die Eigenschaft, dass die Kontostandssumme konstant sein soll nach dem Abschluss der zwei Überweisungen verletzt ist.

- A liest eigenen Kontostand in Variable a ein.
- A dekrementiert a um x.
- B liest eigenen Kontostand in Variable b ein.
- B dekrementiert b um x'
- B liest As Kontostand in Variable a' ein.
- B inkrementiert a' um x'.
- B schreibt a' in As Kontostand zurück. ...