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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS16/17

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1617/grundlagen/

Blatt Nr. 01

Gruppenaufgabe 1 (nicht Zuhause vorbereiten)

Sie designen eine Webanwendung zur Univerwaltung. 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?

Lösung:

a) Eine Datenformate sind inhärent unflexibel, da es keinen standardisierten Pfad zur Erweiterung, Verteilung, Recovery etc. 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[1] o.Ä. verwendet werden muss sondern man für kleine Eigenentwicklungen auch durchaus eine eingebettete Datenbank wie etwa sqlite[2] 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[3], 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.

[1] `http://www-01.ibm.com/software/data/db2/`
[2] `http://www.sqlite.org/`
[3] `http://en.wikipedia.org/wiki/ACID`

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

Hausaufgabe 1

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, dass 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?

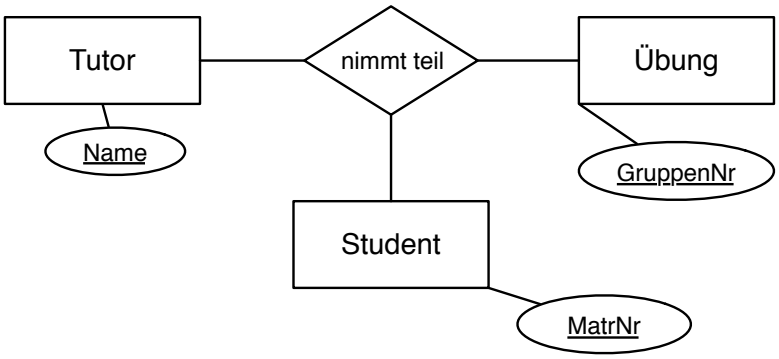
Lösung:

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.

Hausaufgabe 2



Angenommen, das hier modellierte Übungssystem entspricht dem Übungssystem in Grundlagen: Datenbanken. Bestimmen Sie die MinMax Angaben so, dass folgende Einschränkungen modelliert werden:

- Ein Tutor hält mindestens eine Übung.
- Eine Übung wird von mindestens einem Studenten besucht.

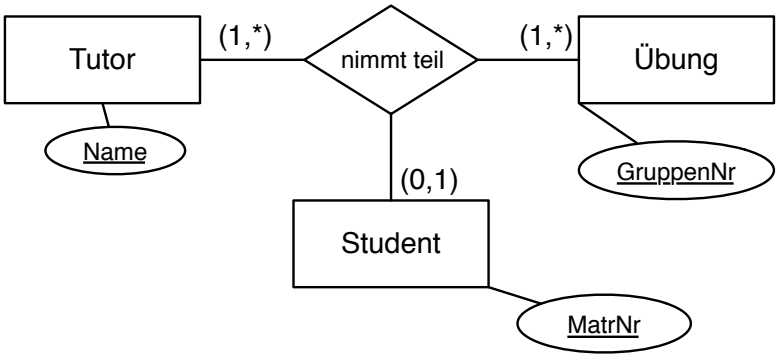
- Ein Student kann höchstens eine Übung besuchen.

Betrachten Sie nun die folgende Ausprägung, die die Beziehung modellieren soll:

Name	GruppenNr	MatrNr
⋮	⋮	⋮
Lang	G12	23
Passing	G27	42
Passing	G27	43
⋮	⋮	⋮
Passing	G28	97
Passing	G28	98
Passing	G28	99
⋮	⋮	⋮

Welche Beziehung besteht zwischen der MinMax Notation und einer solchen Ausprägung?

Lösung:



Im Bezug auf die tabellarische Repräsentation der Beziehung sagen die MinMax Angaben gerade aus, wie oft ein konkreter Wert, etwa der Name des Tutors *Lang*, minimal und maximal vorkommen darf. Die MinMax-Angaben lassen sich also leicht herleiten, indem man sich die tabellarische Repräsentation einer Beziehung vorstellt und überlegt, welche Einschränkungen für die Wiederholung eines konkreten Wertes in dieser Repräsentation gelten sollen. Dadurch, dass im Beispiel keine Matrikelnummer mehrfach vorkommt wird etwa die Einschränkung modelliert, dass ein Student nicht mehrere Übungen besuchen darf.

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.

Lösung:

- 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. ...