DBS: Exercise 1 28/4/17

Tutor: Felicia Burtscher (Thur 12pm)
John Nguyen, Gavithishan Ravechandran

---

1. Aufgabe: Begriffe & Definitionen

(4 P) Was ist ein Datenbanksystem? Wie stehen die Begriffe Datenbank, Datenbanksystem und Datenbankmanagementsystem zueinander im Verhältnis?

A: Eine Datenbank (DB) ist eine Sammlung von verwandten Daten und ein Database-Management-System (DBMS) ist ein allgemeines Software-System, das die Herstellung und Verwaltung einer Datenbank verwendet wird. Die Zusammenfassung einer DB und eines DBMSs wird ein Datenbanksystem (DBS) genannt.

(6 P) Aus welchen Teilen setzt sich ein Datenmodell zusammen? Welche Funktion erfüllen die Teile

A: Das Datenmodell beschreibt die darunterliegende Struktur einer Datenbank. Es ist eine Sprache, die aus dem Data-Definition-Language (DDL) und dem Data-Manipulation-Language (DML) besteht. Der DDL bietet Methoden sowohl zur Beschreibung des Datenbankschemas (z.B. welche Datentypen und Verhältnisse die Daten besitzen) als auch zusätzliche Attribute wie Beschränkungen. Der DML ermöglicht den Zugang und Manipulierung der Daten, indem Methoden zum Einfügen, Abfragen, Updaten und Löschen definiert werden.

(5 P) Was versteht man unter "physischer Datenunabhängigkeit"?

A: Physische Datenunabhängigkeit ist eine Art von Datenabstraktion, wobei komplexe Implementierungen auf der physischen Ebene (d.h. *wie* die Daten gespeichert werden) vom Benutzer der logischen Ebene verborgen werden, der hauptsächlich nur mit den konzeptionellen Datenstrukturen befasst. Ein Anwendungsprogramm besitzt physische Datenunabhängigkeit, falls sie immer noch valide bleibt, obwohl die Speicherstruktur geändert hat.

(5 P) Was versteht man unter "logischer Datenunabhängigkeit"?

A: Obwohl die Daten auf der logischen Ebene wesentlich einfacher dargestellt werden, bleibt immer noch einige Komplexitäten. Dadurch ist logische Datenunabhängigkeit die höchste Abstraktionsebene, in der die Benutzer einen möglichst einfacher Ausblick (d.h. die semantische Bedeutung) der Daten bekommen. Damit bleiben Anwendungsprogramme immer noch valide, auch wenn das Schema (auf der logischen Ebene) sich verändert hat.

(4 P) Nennen Sie zwei funktionale und zwei nicht-funktionale Anforderungen an ein Datenbanksystem.

A: Zwei funktionale Anforderungen einer DB sind: 1) sie muss sicher sein, d.h. einige Daten kann nur von berechtigten Personen greifbar sein), und 2) sie muss konsistent sein, d.h. verschiedene Exemplare von Daten müssen übereinstimmen.

Zwei nicht-funktionale Anforderungen sind: 3) Die DB soll möglichst benutzerfreundlich aufgebaut werden, und 4) sie muss möglichst effizient laufen, d.h. schnelle Antwort- und Updatezeit.

(1 P) Was sind Meta-Daten?

A: Metadaten sind Daten, die sowohl die Struktur der DB als auch den Typ und Format jedes Datenobjektes beschreiben. Außerdem kann die Metadaten Beschränkungen der Daten bestimmen. Die Metadaten werden (gentrennt von der DB-Daten) im DB-Catalog gespeichtert.

Quellen *Fundamental of Database Systems*, 7ED, Elmasri, Navathe, 2016.
Database System Concepts, 6ED, Silberschatz, Korth, Sudarshan, 2006.

2. Aufgabe: Arten von Datenbanksysteme

(5 P) Recherechieren Sie, welche Arten von Datenbanksystemen existieren und wie sich diese gruppieren lassen.

A: Es gibt drei grundlegende Arten von DBSs, welche sich von ihrer technischen Umsetzung unterscheiden.

1. Hierarchie Modell, Netzwerk Modell. Das Hierarchie Modell wird nach einer nach unten gerichteten Baumstruktur aufgebaut. Die Beziehung zwischen den Daten wird mit Pfeilen dargestellt. Das Netzwerk Modell entspricht zum größten den Hierarchischen Modell, aber bei diesem Modell können die Daten untereinander verbunden werden. Somit entspricht es nicht immer der nach unten gerichteten Baumstruktur.

2. Relationales Modell, Deduktives Modell. Das relationales Modell wird nicht wie die beiden oberen Modellen in einer Baumstruktur gespeichert, sondern in einer zweidimensionalen Tabelle. Die Bearbeitung der Daten erfolgt auf mathematischer Relationstheorien. Das Deduktives Modell entspricht dem relationalen Modell, nur dassdie Daten auf Prädikats Berechnungen bearbeitet werden.

3. Entity-Relationship Modell, Objekt-orientiertes Modell. Das Entity-Relationship Modell stellt die Daten (Entities) und ihre Beziehungen (Relationships) zueinander als Objekte dar. Das objekt-orientiertes Modell erweitert dieses Konzept, indem Operationen auf den Daten ermöglicht werden. Damit entsprechen Daten-Objekte den Klassen vom Objekt-Orientierte-Paradigma.

Quellen:

http://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/

https://www.analyticsvidhya.com/blog/2014/11/types-databases-evolution/

http://www.brighthub.com/internet/web-development/articles/110654.aspx

(5 P) Was sind NoSQL-Datenbanksysteme? Welche Arten von Datenbanksystemen könnten NoSQL-Datenbanksysteme sein. Begründen Sie Ihre Antwort.

A: NoSQL ist eine Abkürzung für *Not Only SQL*. Der Vorteil von NoSQL-Datenbanksystemen sind, dass sie große Mengen an Daten speichern können und abrufen können und damit sind sie geeignet für Big-Data/Data-Mining Anwendungen. Dies wird ermöglicht, da sie nicht der rela-

tionalen Ansatz verfolgen oder keine festen Tabellenschemas verfolgen. Oft sind die Schemas dynamisch, sodass die DB mit sich ständig verändernd Daten umgehen kann.

Da NoSQL-Datenbanksysteme nicht den relationalen Ansichten verfolgen, gehören alle andern Arten der Datenbanken zu der NoSQL-Datenbanksystemen. Es wird immer nur die erste Instanz betrachtet, so können zum Beispiel nur Daten direkt angelegt werden(nicht relational) aber die Daten selbst intern entsprechen der relationalen Modellen.

Quellen: https://www.mongodb.com/nosql-explained

(10 P) Was ist RDF? Erklären Sie kurz den Aufbau, bzw. die Funktionsweise von RDF. Geben Sie einen einfachen Beispieldatensatz in RDF an.

A: Der **Resource Description Framework** (RDF) ist eine Sammlung von Spezifikationen des World-Wide-Web-Consortiums (W3C) zur Verarbeitung der Metadaten von Webressourcen. Eine Ressource ist alles, was mit einem Uniform-Resource-Identifier (URI) identifiziert werden kann, wie z.B ein Dokument oder datei. Der RDF begründet ein allgemeines System zur Beschreibung solche Ressourcen, ohne Details des Anwendungsbereich anzunehmen.

Das RDF-Datenmodell wird durch Aussagen aufgebaut, die aus drei Komponenten bestehen. Die erste Komponente ist das Subjekt (eine Ressource), zweitens kommt der Property-Type (ein Attribute des Subjekts), und zuletzt ist der Wert (eine andere Ressource oder ein atomarer Wert, z.B String oder Zahl). Der Property-Typ verknüpft die äußeren Kompoonenten. Zum Beispiel, die Aussage

$$\text{(resource R)} \text{ — property type P} \rightarrow \text{(value V)}$$

bedeutet, dass die Ressource R den Wert V für das Attribut P hat. In diesem Modell kann die Ressourcen und Werte als Knoten und die Property-Types als Kanten eines gerichteten Graphen betrachtet werden. Wir zusammenfassen die Aussage als 3-Tupel.

Als ein konkretes Beispiel betrachten wir den Datensatz von Studierenden:

student	**name**	**number**	**major**
student1	bob	12345	Comp Sci
student2	jane	54321	Philosophy
...	...	...	...

Im RDF-Format sieht die Daten wie folgt aus:

{	student1,	name,	bob }
{	student1,	number,	12345 }
{	student1,	major,	Comp Sci }
{	student2,	name,	jane }
{	student2,	number,	54321 }
{	student2,	major,	Philosophy }
{	...	...	... }

Quellen: https://www.w3.org/TR/WD-rdf-syntax-971002/

(5 P) Wie könnte eine einfache SPARQL-Abfrage an Ihren Beispieldatensatz aussehen? Erklären Sie, wie ihre Abfrage funktioniert.

A: Unsere einfache SPARQL Abfrage ersucht die Studentnummern aller Studierende, deren Name gleich 'bob' ist und außerdem hat das Hauptfach 'Comp Sci':

```
PREFIX info: <http://www.example.com/attributeNames/>

SELECT ?student  ?number

WHERE {
    ?student  info:name "bob"
    ?student  info:major "Comp Sci"
    ?student  info:number ?number
}
```

Zuerst schauen wir die SELECT-Klausel an. Hier deklarieren wir drei Variablen, in denen die Ergebnisse gespeichert werden. Zunächst betrachten wir die WHERE-Klausel. Wir listen die Abfragebedingung als 3-Tupel auf. Jede Bedingung wird auf die Ergebnisse der vorgegangenen Bedingung getestet. Das erste 3-Tupel fragt alle Studierende an, bei denen das *name*-Attribut den Wert 'bob' hat. Von denen ersucht das nächste 3-Tupel die Studierende, die 'Comp Sci' als Hauptfach studieren. Schließlich bittet das letzte 3-Tupel um die Studentnummern von diesen Ergebnissen.

Zuletzt merken wir, dass in SPARQL, die Attributnamen mittels eines URI bezeichnet werden, wie z.B: http://www.example.com/attributeNames/major. Da jedes Attribut ein gemeinsames (in unserem Beispiel) Präfix haben, können wir dieses in einer Variable speichern, die *info* heißt. Dies wird in der ersten Zeile der Abfrage deklariert.

3. Aufgabe: Typische Abfragen

(2 P) Wie könnte umgangssprachlich eine typische Abfrage an ein Datenbanksystem eines Kursverwaltungssystems wie dem KVV aussehen?

A: Return the all the course information for computer science modules available this semester.

(2 P) Wie könnte umgangssprachlich eine typische Abfrage an ein Datenbanksystem einer Online-Videothek/Online-Mediathek aussehen?

A: Return all videos that have the tag 'computer science' in order most views to least views

(2 P) Wie könnte umgangssprachlich eine typische Abfrage an ein Datenbanksystem einer Photo-Sharing-Plattform wie Instagram aussehen?

A: Return all photos with the hashtag *#cat* orderd from newest to oldest.

(2 P) Wie könnte umgangssprachlich eine typische Abfrage an ein Datenbanksystem eines Mikrobloggingdienstes wie Twitter aussehen?

A: Return all tweets posted by users who are followed by the user 'snoopy'.

(2 P) Wie könnte umgangssprachlich eine typische Abfrage an ein Datenbanksystem eines sozialen Netzwerks wie Facebook aussehen?

A: Return usernames of users who have a mutual friend with the user 'leonard943'.

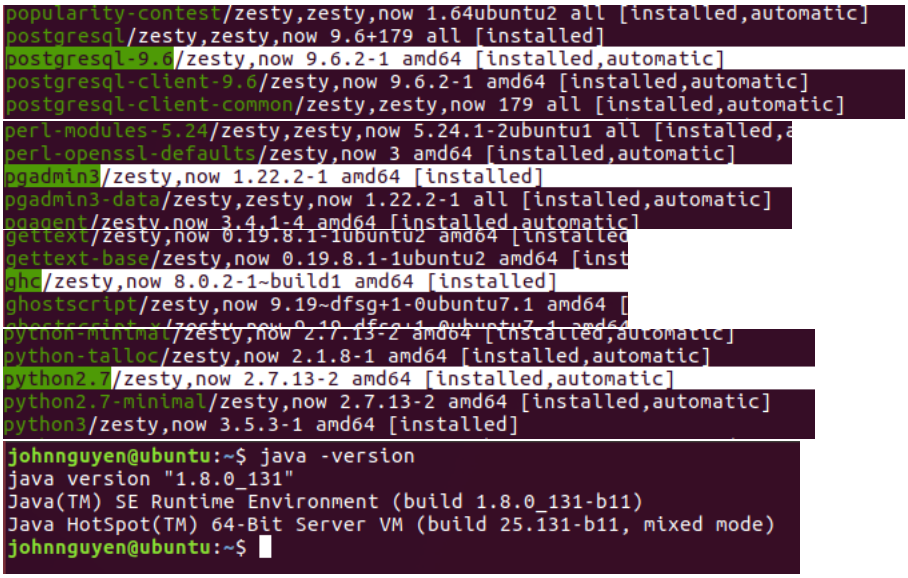
4. Aufgabe: PostgreSQL & Programmiersprachen

All code can be found in the following repository: https://github.com/jxnguyen/DBS

The following screenshots show the results of the following unix command:

```
apt list --installed
```

INSTALLATION



HASKELL

```
main = do
  -- ask for input
  putStrLn "Hey what's your name?"
  -- read input
  name <- getLine
  -- print output
  putStrLn ("Hello " ++ name ++ "!")
```

This program is executed with the command:

```
ghc hello.hs && ./hello
```

PYTHON

```
# prompt for name
name = raw_input("Hey what's your name? ")
# print message
print "Hello " + name + "!"
```

This program is executed with the command:

```
python hello.py
```

JAVA

```
import java.util.Scanner;

public class Hello {

    public static void main(String[] args) {
        // scanner to read input
        Scanner scanner = new Scanner(System.in);
        // prompt user for name
        System.out.print("Hey what's your name? ");
        // get input
        String name = scanner.next();
        // print message
        System.out.println("Hello " + name + "!");
    }
}
```

This program is executed with the command:

```
javac Hello.java && java Hello
```