Prüfungsprotokoll Informatik Praxis

Datenmanagement & Datenexploration (Vertiefung)

Prüfer: Professor Seidl
Beisitzer: Christoph Brochaus

- Einführung in Datenbanken
- Data Mining Algorithmen
- Indexstrukturen für Datenbanken

Note: 2,0
Dauer: 60 Minuten

Info

Die drei Prüfungsgebiete passen grundsätzlich prima zusammen. Es gibt zahlreiche Überschneidungen und Referenzen (z.B. Indexstrukturen im Data Mining). Zu Data Mining und Indexstrukturen gibt es gute Skripte aus den Vorlesungsfolien von Herrn Seidl selbst. Einführung in Datenbanken kann man gut aus dem Kemper-Buch lernen, das auch zur Vorbereitung empfohlen wird. Die klassische Einführungsvorlesung von Herrn Jarke genügt jedoch auch völlig.

Die Prüfung verlief insgesamt recht locker. Ich hatte vorher bereits die mündliche Scheinprüfung in Indexstrukturen bei Herrn Seidl gemacht (was ich jedem nur empfehlen kann, man bekommt einen guten ‚Drift' vom Professor und dem Fragenkatalog). Ich habe versucht, mehr über die eigentlichen Fragen hinaus zu sagen. Das ist mir vermutlich etwas zu gut gelungen, denn Herr Seidl gab mir nach einiger Zeit zu verstehen, dass er auch gerne mal wieder eine Frage stellen würde.

Dies war auch ein Grund dafür, dass die Prüfung recht lange gedauert hat. Der andere war, dass ich die praktischen Aufgaben nur mit etwas Mühe und teilweise erst nach Hilfestellung von Herrn Seidl lösen konnte. Die Note finde ich insgesamt sehr fair.

Viel Erfolg bei Euren Prüfungen. Bitte macht ein Prüfungsprotokoll.

Einführung in Datenbanken (35 Minuten)

Wie geht man vor, wenn man eine Datenbank entwerfen möchte?

(Klassischer Einsteiger) Anforderungsanalyse, Kundenkontakt, Pflichtenheft formalisieren, Sichten konsolidieren, konzeptueller Entwurf, relationaler Entwurf, Implementierung.

Wie führt man die konzeptuelle Modellierung durch?

ER,UML. Letzteres besser, weil mächtiger und OO-geeignet (Notation für Operationen). Noch etwas zu Funktionalität, (min, max) und Multiplizität gesagt. Kurze Erwähnung von Generalisierung, die im relationalen Modell nicht ohne weiteres möglich ist.

Malen Sie mal ein Beispiel in ER und dann in UML auf.
Das klassische Beispiel mit Vorlesungen und Studenten. Dabei noch gewitzelt, dass Absolventen später vermutlich unfähig sein werden, mal ein anderes Schema zu entwerfen.

Welche Relationen können Sie aus dem Diagramm übernehmen?
Zwei Relationen übernommen. Die dritte habe ich geschlabbert. Die musste ich dann später im Synthesealgorithmus nachholen.

Gut, nun nehmen wir mal an, Ihrem Kunden ist Ihr Entwurf zu teuer. Er schlägt stattdessen folgenden Entwurf vor: {[VorlNr, Titel, MatrNr, Name]}. Ist das eine gute Idee?
Die Frage habe ich auch schon mal irgendwo gelesen. Natürlich schlechte Idee, da zwei Konzepte in einer Relation vereint sind. Im besten Fall (bei sorgfältiger Implementierung) also schlechtes Laufzeit- und Speicherplatzverhalten. Im schlechteren Fall Anomalien (Einfüge-, Lösch-, Updateanomalien; am Beispiel erläutert).

Wie kann man denn ein gutes Schema sicherstellen?
Durch Normalformen (Prinzip erklärt, dabei mehrfach darauf hingewiesen, dass ein ordentlicher Entwurf diese Prüfung überflüssig macht).

Ist denn das obige Schema nicht schon in 1NF?
Doch, klar. Alle Schemata des relationalen Modells sind ja per Definition 1NF (Attribute atomar). Bei mengenwertigen Domänen Tabellen ‚flachklopfen'.

Habe dann 2NF/3NF erklärt. Dabei festgestellt, dass man ohne die Definition von funktionaler Abhängigkeit, voller funktionaler Abhängigkeit, Superschlüssel, Kandidatenschlüssel nicht weiterkommt – also auch noch erläutert. Schliesslich Synthesealgorithmus erwähnt.

Wie funktioniert denn der Synthesealgorithmus?
Kanonische Überdeckung (Linksreduktion, Rechtsreduktion, Verweise auf leere Mengen entfernen, FDs mit gleichen linken Seiten zusammenfassen). Aus $A \rightarrow B$ die $R=A \cup B$ erzeugen, Teilmengen akkumulieren, ggf. Pseudo-FD für Kandidatenschlüssel anlegen, aufräumen.

Dann wenden Sie den mal auf das obige Schema an.
Versucht, dabei stümperhafte Fehler gemacht. Letztendlich (nach sanften Schubsern) noch hinbekommen.

Wie sieht denn eine klassische Abfrage in SQL aus?
SELECT .. FROM .. WHERE .. Dabei schon mal die Zusammenhänge von RA und SQL erläutert (SELECT=Projektion, FROM=Kartesisches Produkt, WHERE=Selektion mit Prädikat).

Machen Sie mal eine Beispielanfrage in SQL auf die Relation: Alle Studenten ausgeben, die die Vorlesung ‚EDB' hören.
Hingeschrieben, dabei Fehler gemacht, aber auf Nachfrage korrigiert.

Was ist denn nochmal ein JOIN und wo braucht man den hier?
Verbindung zweier Tabellen an ihren ‚Verkettungen', ihrem Joinprädikat. Wird in SQL im WHERE-Teil mit AND realisiert.

Welche Anfragesprachen kennen sie sonst so?
Prozedurale RA, deklarativer Relationenkalkül, letzterer als Tupelkalkül und Domänenkalkül (Bindung von freien Variablen an Tupel oder Wertebereiche). Alle drei gleich mächtig, eingeschränkt auf sichere Anfragen. SQL auch gleich mächtig, dabei eher wie Tupelkalkül.

Ausser dem JOIN kennt RA ja sechs Basisoperationen. Welche?
*Projektion, Selektion, Attributumbenennung, Mengenvereinigung, Mengendifferenz, kartesisches Produkt. Letzteres hat |R| * |S| Elemente.*

Wie funktioniert denn die Mengenvereinigung in SQL?
SELECT … UNION … SELECT

Data Mining Algorithmen (10 Minuten)

Ab hier hat Herr Seidl spürbar das Tempo gestrafft, wohl da der EDB-Teil bereits sehr umfangreich war.

Was ist das Grundproblem der Klassifikation?
Train & Test: Label für Trainigsset D-O und Classifier für Testset finden.

Erklären Sie den Bayes-Classifier.
Probabilistisches Verfahren, Hypothesen bewerten, Optimal Bayes, Naive Bayes, Independence Hypothesis, Bayes-Network.

Erläutern Sie den k-NN Classifier.
Instanz-basiertes Verfahren, Lazy Evaluation, im Gegensatz zu Bayes nur Mittelwert schätzen und nicht Kovarianzmatrix; Beispiel aufgemalt, dabei k und ε verwechselt; Decision Set und Decision Rules, standard und weighted.

Was ist ein Decision Tree?
Intuitives Verfahren, Baumstruktur, Knoten sind Tests auf Attribute, Kanten sind Testergebnisse, Blätter sind Labels=Entscheidungen, Aufbau, Split, Attributauswahl durch Entropy/Gain und GINI, Overfitting, Prepruning, Postpruning (Error reducing und Minimal Cost Complexity).

Indexstrukturen für Datenbanken (15 Minuten)

Was ist ein R-Baum?
Struktur beschrieben. Dabei auf die Dimensionalität hingewiesen ($2 \leq d \leq 8$). Probleme bei höheren Dimensionen beschrieben.

Wie sieht so ein R-Baum aus?
Da wusste ich nicht viel zu. In der Vorlesung wurden R-Bäume immer als MURs repräsentiert. Ich wusste aber noch die formalen Anforderungen, das hat mich etwas gerettet.

Wie ist das Laufzeitverhalten für Abfragen im R-Baum?
Keine Ahnung. Etwas mit B-Bäumen herumgedoktort, schliesslich aufgegeben.

Welche Indexstrukturen gibt es für Daten ohne räumliche Informationen?
Metrische Daten. Einzige bekannte Information: Distanzen, die jedoch umständlich zu berechnen sind. Vier Grundeigenschaften aufgezählt (Nichtnegativität, Definitheit, Symetrie, Dreiecksungleichung).

Wie funktioniert das Abschätzen von Entfernungen in metrischen Daten?
Über Dreiecksungleichung: Umweg über dritten Punkt kann nie kürzer sein, als der direkte Weg zwischen zwei Punkten. Pythagoras erwähnt. Noch etwas vom allgemeinen Prinzip der Obermenge und Antwortmenge erzählt, was nicht richtig falsch, aber hier auf jeden fall auch nicht ganz richtig war.

Was ist ein M-Tree?
Aufbau und Prinzip erklärt: Baum speichert neben den üblichen Informationen auch Routing-Objekt und Distanz zum Vorgängerknoten. Beide Informationen kann man verwenden, um Distanzberechnungen zu sparen.