Studienarbeit — Vorlage
VL Informationssysteme, Uni Saarland

Jens Dittrich

1. Juli 2014

[**Fristen:**

Wann	**Was**
13. Juli	Projektbeschreibung (Kapitel 1) genehmigt vom Tutor
20. Juli	E/R-Modell (Kapitel 1, 2)
10. August	Relationales Modell und SQL-Datenbankschema inklusive Beispieldaten und Skripte (Kapitel 1–4.2)
1. September	Finale Abgabe (Kapitel 1–9)
bis 24. September	Projektvorstellung

Alle Abgaben nur über das Moodle als pdf bzw. als SQL-Skript. Die Abgaben vom 20. Juli und 10. August erlauben es den Tutoren, Ihnen früh Feedback zu geben und damit Folgefehler zu verhindern. Diese Abgaben können Sie in späteren Abgaben noch verbessern. Die Qualität der ersten Abgaben fliesst aber mit in die Gesamtwertung ein. Benutzen Sie möglichst LaTeX zur Erstellung des Dokuments. Abgaben anderer Dokumenttypen werden aber auch akzeptiert, sofern sie in pdf abgegeben werden.

Um die Lesbarkeit von SQL-Befehlen zu erhöhen, ist folgende Formatierung erwünscht:

```
SELECT          a.attribut2, count(*)
FROM            RelationA a
                JOIN RelationB b ON a.ID=b.a_ID
WHERE           a.attribut='SQL' AND b.attribut>42
GROUP BY        a.attribut2
HAVING          count(*) > 2;
```

]

1 Projektbeschreibung

[1–2 Seiten Beschreibung abgestimmt mit dem Tutor. Genehmigt vom Tutor bis spätestens 13. Juli. Danach noch Änderungen möglich in Abstimmung mit Tutor. In diesem Fall die nachträglichen Änderungen hier klar dokumentieren. Falls vorgegebenes Thema, 1:1-Kopie der Themenbeschreibung; Achtung: die Projektbeschreibung ist kein vollständiges Pflichtenheft. D.h. Sie haben gewisse Freiheiten Funktionalitäten zu

implementieren oder auch nicht. Versuchen Sie das Projekt so zu entwickeln, dass es möglichst vollständig die Anforderungen, die sich aus der Projektbeschreibung ergeben, erfüllt. Die minimalen Anforderungen ergeben sich aus den folgenden Abschnitten. Im Zweifel fragen Sie Ihren Tutor.]

2 E/R-Modell des Szenarios

[Inklusive Chen und min/max-Notation; alle Dinge, die sich nicht klar aus dem Modell ergeben, kurz erläutern — also: warum wurde ein Zusammenhang auf diese Art modelliert, welche Vor-/Nachteile hat das?]

3 Relationales Modell

[Umsetzung des E/R-Modells]

3.1 Liste der Relationen

[gerne bereits vereinfacht]

3.2 Zusätzliche Einschränkungen und Integritätsbedingungen

[Liste zusätzlicher Einschränkungen, die nicht aus Abschnitt 3.1 vorgehen, die aber mit in die Datenbank übernommen werden sollten. D.h. alle zusätzlichen Integritätsbedingungen.]

4 Datenbankschema

4.1 SQL

[in möglichst kurzer und übersichtlicher SQL- Notation; bitte die Statements formatieren so wie oben beschrieben; bei Teams von zwei Leuten mindestens 20 sinnvolle Tabellen, bei Teams von drei Leuten mindestens 30 sinnvolle Tabellen, inklusive Integritätsbedingungen; zusätzlich elektronische Abgabe eines schema.sql-Skriptes, das dieses Schema in Postgres erzeugt]

4.2 Beispieldaten

[sinnvolle Beispieldaten im Umfang ähnlich wie im Fotoagenturschema der Vorlesung, d.h. max. 10 Tupel pro Tabelle; auf 1-2 Seiten übersichtlich dargestellt; zusätzlich elektronische Abgabe eines daten.sql-Skriptes, das dieses Schema in Postgres erzeugt; gerne zusätzliche auch eine Datenbank mit grösseren Daten, aber kein Muss]

4.3 Konsistenztrigger

[Definition von mindestens zwei sinnvollen Konsistenztriggern. Vollständige Implementierung und kurze Erläuterung der Funktionsweise mit Beispieldaten; Implementierung von min/max wo immer sinnvoll;

zusätzlich elektronische Abgabe eines konsistenz.sql-Skriptes]

5 Normalisierung

[Für mindestens 5 Tabellen Erläuterung, ob sie in 3NF oder BCNF sind.]

6 Logische Datenunabhängigkeit

[Vollständig implementiert für Lesen und Schreiben; Auflistung der erstellten Sichten und Regeln in SQL]

7 Beispielanfragen

[Alle implementiert gegen die Sichten, SQL-Statements hier aufgelistet.]

7.1 Änderungsoperationen

[je zwei INSERTs, UPDATEs und DELETEs, vier sinnvolle Transaktionen]

7.2 Leseoperationen

[zehn einzelne Statements, die für die Applikation sinnvolle Daten anfragen und einen vollständigen Überblick über die benötigen Daten geben; ohne zu gruppieren und aggregieren, davon vier mit nicht-skalaren Unteranfragen]

7.3 Analyseoperationen

[fünf einzelne Statements, die für die Applikation sinnvolle Daten anfragen und einen vollständigen Überblick über die benötigen Daten geben mit Gruppierung und Aggregation, zwei davon mit HAVING]

7.4 MapReduce

[eine Anfrage aus Abschnit 7.3 mit HAVING als map()/reduce()-Programm geschrieben]

8 Indexe

[auch wenn Indexe für so kleine Datenmengen keinen Performance-Effekt haben, legen Sie geeignete Indexe an, damit Ihre Datenbank auch für grosse Datenmengen ihre Beispielanfragen aus Abschnitt 7 effizient verarbeiten kann.]

9 Zusammenfassung

[max. 2 Seiten: was von der Projektbeschreibung wurde letztendlich implementiert und was nicht? Wo geht die Implementierung über die Projektbeschreibung hinaus und wenn ja wie?]