Prüfer: Balke, Eckstein
Beginn: 9:30
Ende: 9:50
Note: 4.0 (Ergänzungsprüfung, sonst wäre es eine 3 gewesen)
Allgemeines: Die Prüfung war sehr angenehm. Die Prüfer haben immer geholfen und ich fühlte mich trotz der ernsten Situation wenig unter Druck gesetzt.

Balke: Womit wollen wir anfangen?

Ich: Relationale Algebra.

Balke: Okay, in der Relationalen Algebra haben wir Grundoperatoren. Welche sind das?

Ich: Alle aufgeschrieben: Selektion, Projektion, Rename, Vereinigung und Differenz, sowie das Kartesiche Produkt.

Balke: Gut. Warum sind dies Grundoperationen? Wir haben ja noch ein paar mehr kennengelernt.

Ich: Die anderen Operationen können wir mit diesen Grundoperationen darstellen. Join ist Selektion auf ein Kartesisches Produkt.

Balke: Ja, und in welchem Zusammenhang steht das mit anderen Abfragesprachen?

Ich: Wir nennen eine Abfragesprache relational vollständig, wenn wir jede Operation durch mindestens eine Operation der Abfragesprache darstellen können.
SQL ist relational vollständig und sogar mächtiger als RA. Eine SQL Query wird intern in RA umgewandelt und optimiert.

Balke: Ja, und wie funktioniert diese Umwandlung?

Ich: ... [Ich wusste wirklich nicht, was er von mir hören wollte. Hab irgendwas gestammelt.]

Balke: Sie haben ja gesagt, dass jede Operation durch einen Ausdruck dargestellt werden kann. Schreiben Sie doch mal ein grundsätzliches Query auf!

Ich:
SELECT ...
FROM ... AS ....
JOIN ... ON ...
WHERE (Condition)
GROUP BY ...
HAVING ...
ORDER BY ...

Balke: Okay. Und wie brechen wir das jetzt in RA. runter?

Ich: Ja... [musste wieder überlegen]. Also das Select ist die Projektion. AS ist rename. Where ist die Selektion. Join ist Kartesisches Produkt mit Selektion.

Balke: Richtig. Und was ist mit Group By, Having, und Order By?

Ich: Dort ist SQL mächtiger als RA.

Balke: Ja, aber es wird intern trotzdem in RA. dargestellt.

Ich: Hm...

Balke: In welcher Reihenfolge wird Ihre Query denn ausgewertet?

Ich: Ja... Order By zuletzt. Wahrscheinlich erst einmal die Join oder die Selektion, je nachdem, was effizienter ist. Dann werden die Gruppen gebildet und Having ausgewertet. Dann die Projektion und das Order By.
[war wohl einigermaßen richtig]

Balke: Und innerhalb der Gruppen können sie Having als Selektion darstellen. Gut. Nochmal zurück zu den Joins. Welche Joins gibt es denn?

Ich: Theta Join, Equi Join, Natural Join, Semi Join. Alle kurz erklärt, was sie tun, wie man sie mit Grundoperationen darstellen kann und wofür man sie braucht.

Balke: Mir fehlen da aber noch ein paar.

Ich: Die outer Joins. Nullwerte wo kein Join Partner.

Balke: Und wozu eignen die sich besonders? Wann nehmen wir die?

Ich: Hmm... [Keine Ahnung gehabt]

Balke: Bei Aggregationsfunktionen. Okay - wir haben ja auch über aktive Datenbanken gesprochen. Können Sie mir darüber etwas sagen?

Ich: Wir nennen eine Datenbank aktiv, wenn sie vordefinierte Ereignisse/Zustände selbstständig erkennen kann und angemessene Aktionen eigenständig durchführen kann. Dabei haben wir uns hauptsächlich auf Trigger konzentriert.

Balke: Ja, und was sind Trigger?

Ich: Wir legen ein Ereignis fest, und wenn dieses eintritt, feuert der Trigger und führt vordefinierte Aktionen aus.

Balke: Schreiben Sie mal einen auf. So grundsätzlich.

Ich: [musste ich viel hin und her basteln]
CREATE TRIGGER name
AFTER/BEFORE UPDATE/INSERT/DELETE OF attribut ON tabelle
REFERENCING NEW/OLD AS foo
WHEN foo.attrib = ...
BEGIN
SQL
END

[Er hat mir bei Referencing geholfen]

Balke: Ja, wir haben ja jetzt nur die Operationen UPDATE/INSERT/DELETE. Es wären ja noch

viel mehr Ereignisse denkbar. Könnten wir die auch mit Triggern abfangen?

Ich: ...

Balke: Wir wollen ja Integritätsbedingungen absichern. Kann man da noch erweiterte [hab ich vergessen] absichern?

Ich: Ja, müsste gehen, [Voll in die Falle.]

Balke: Nein, geht nicht. Sonst turingvollständig... Stored procedures...
Balke: Springen wir ein bisschen zurück. Wie kommen wir denn von einem ER Diagramm zu einem Relationalen Schema?

Ich: Jeder Entitätstyp wird eine Relation.

Balke: Ja, nehmen wir zum Beispiel mal sowas hier...
[Er hat einen Entitätstyp A mit den Attributen A1 und A2 gezeichnet, dann einen Entitätstyp B mit dem Attribut B1 und eine Relation zwischen den beiden mit Black Diamond Notation.]

Ich: Uh, Black Diamond ist nicht so mein Fall, aber das ist One-To-Many, oder?

Balke: Richtig.

Ich: Okay, also wir haben viele A, die mit jeweils genau einem B in Verbindung stehen. Also bekommen alle A einen Fremdschlüssel von B....
[Da wurde es richtig häßlich, weil ich die Relationen total vermurkst habe. Ich wollte immer ein Attribut von A als Fremdschlüssel von B definieren. Balke hat aber sehr geholfen und gesagt, dass ich mich nicht verrückt machen lassen soll. Irgendwann bin ich drauf gekommen, dass ich für den Relationstyp eine eigene Relation brauche, die jedem Schlüssel A eine Entität B zuweist. Viel Zeit vergeudet.]

Balke: Na schön, die Zeit ist fast um. Haben Sie noch eine Frage, Frau Eckstein?

Eckstein: Ja, wir haben ja über das Drei Schichten Modell gesprochen. Können Sie uns darüber etwas sagen?

Ich: Ja, wir wollen die Benutzer/Anwendungen und die Speicherung der Daten entkoppeln. [Diagramm gezeichnet] Externe Schicht - hier sind alle Benutzer. Konzeptionelle Schicht - hier sind unsere Tabellen. Interne Schicht - hier sind unsere Tabellen gespeichert. Wenn wir die Tabellen von einer Festplatte auf eine andere verschieben, ändert sich für den Benutzer nichts (bekommt er gar nicht mit). Und wenn wir ein Atrribut nicht mehr Speichern, sondern berechnen, ändert sich für ihn auch nichts.

Eckstein: Ja, und wie nennen wir das?

Ich: Physikalische Unabhängigkeit und Logische Unabhängigkeit.

Eckstein: Wie wird das Modell an sich genannt?

Ich: ANSI / Sparc. Oder 3 Layer Architecture.

Eckstein: Jawohl. Dann gehen Sie bitte kurz raus.