11. Übung zu Einführung in Datenbanksysteme

Abgabe bis Dienstag, den 11. Juli

1. Aufgabe *7 Punkte*

Gegeben ist eine relationale Datenbank mit den folgenden beiden Relationen:
veranstaltung (VName, vnr, Raum, Dozent)
nimmtTeil (vnr, Note, Student)
mit n bzw. m Datensätzen. Die physische Datenstruktur für die Relationen und ihre Attribute ist wie folgt: Veranstaltung: char(20), int = 4 Bytes, int, char(20), nimmtTeil: int,int,char(20).

Es ist anzunehmen, dass die Relation Veranstaltung vollständig in den Hauptspeicher passt. Weiterhin gelten die folgenden Werte:
mittlere Zugriffszeit: 10ms
Block = 4 KB
Datenrate: 2 MB/s

Betrachten Sie die folgende Anfrage:

```
Select v.VName, v.Dozent, n.Student
from veranstaltung v, nimmt_teil n
where v.Vname = n.VName
and v.VName = "DBS"
```

(a) Schätzen Sie die Zeit ab, die benötigt wird, um die gegebene Anfrage unter Verwendung des *nested loop join* auszuwerten.

(b) Schätzen Sie die Zeit ab, die benötigt wird, um die gegebene Anfrage unter Verwendung eines Index auf dem Attribut vnr der Relation nimmtTeil auszuwerten. Machen Sie geeignete Annahmen über die Selektivität von vnr in der Relation *nimmtTeil*.

(c) Kann man mit der Clustertechnik (2. Art) die Zeit für die Anfrage verbessern? Zeigen Sie die neue physische Dateistruktur (anhand einiger Beispieleinträge) und berechnen Sie die Zeit für die von Ihnen gewählte physische Organisation der Daten.

(d) Geben Sie zu allen drei Verfahren jeweilig den Ausführungsplan (Executionplan) an.

2. Aufgabe *6 Punkte*

Geben Sie für die folgenden Schedules an, ob sie serialisierbar, konflikt-serialisierbar oder view-serialisierbar sind.

(a) T1:R(X), T2:R(X), T1:W(X), T2:W(X), T2:Commit, T1:Commit

(b) T1:W(X), T2:R(Y), T1:R(Y), T2:R(X), T2:Commit, T1:Commit

(c) T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Abort

(d) T2:R(X), T3:W(X), T3:Commit, T1:W(Y), T1:Commit, T2:R(Y), T2:W(Z), T2:Commit

3. Aufgabe *3 Punkte*

Gegeben sind die folgenden Transaktionen T1 und T2.

T1:		T2:	
	read-item(x)		read-item(x)
	read-item(y)		x:=x+1
	y=y+x		write-item(x)
	write-item(y)		read-item(z)
	read-item(x)		z:=z+1
	read-item(z)		write-item(z)
	z:=z+x		commit
	write-item(z)		
	abort		

(a) Geben Sie ein Schedule für T1 und T2 an, das zu einem *dirty read* führt.

(b) Geben Sie ein Schedule für T1 und T2 an, das zu einem *lost update* führt.

4. Aufgabe *7 Punkte*

Betrachten Sie folgenden Schedules. Sind die Schedules vollständig? Sind sie konfliktserialisierbar? Erstellen Sie für jeden Schedule einen Präzedenzgraphen. Geben Sie für jeden konfliktserialisierbaren Schedule eine äquivalente serielle Ausführungsreihenfolge an.

(a) r_1(X), r_2(X), w_1(X), w_2(X), $commit_1$, $commit_2$

(b) w_1(X), r_2(Y), r_1(Y), r_2(X), $commit_1$, $commit_2$

(c) w_1(X), r_2(X), w_1(X), $commit_2$, $commit_1$

5. Aufgabe *4 Punkte*

Angenommen, ein DBMS erkennt *increment* und *decrement* als Aktionen, zusätzlich zu *read* und *write*. Die beiden Aktionen seien Versionen des blinden Schreibens (der Wert der Variable ist der Aktion nicht bekannt). Zusätzlich zu shared und exclusive Locks seien zwei Spezialsperren implementiert: ein Objekt muss in *I-Modus* gesperrt werden bevor inkrementiert , und in *D-Modus* bevor dekrementiert. I-Locks sind kompatibel zu I- und D-Locks auf dem gleichen Objekt, nicht jedoch zu S- und X-Locks.

Erläutern Sie, wie der Gebrauch von I- und D-Locks Nebenläufigkeit erhöht. Zeigen Sie dazu ein Schedule, das bei striktem 2PL nur S- und X-Locks verwendet. Erläutern Sie, wie der Gebrauch von I- und D-Locks mehr überlappende Interaktionen zuläßt, wobei das strikte 2PL fortgeführt wird.

6. Aufgabe *6 Punkte*

Wie kann man das 2PL abschwächen, wenn man niedrigere Isolationslevel (siehe Übung 8) als serialisierbar verwendet.