FernUniversität in Hagen

Lösungsvorschläge zur Klausur „1666 Datenbanken in Rechnernetzen"

09.03.2013

Aufgabe 1 (Multiple Choice) (10 Punkte)

Welche der folgenden Aussagen sind korrekt? Kreuzen Sie die Wahr (W) oder die Falsch (F) Spalte an.

W	F	Das Zwei-Phasen-Commit-Protokoll ...
☒	☐	... garantiert globale Serialisierbarkeit von verteilten Transaktionen.
☐	☒	... garantiert eine serielle Ausführung von verteilten Transaktionen.
		Garantiert eine serialisierbare Ausführung ...
☒	☐	... verhindert ein fortgepflanztes Rollback.
☐	☒	... garantiert den gleichzeitigen Start der Subtransaktionen.
		Garantiert das quasi gleichzeitige Commit der Subtransaktionen.

W	F	Was macht eine Teiltransaktion, wenn sie den ready-to-commit Zustand erreicht?
☒	☐	Sie hält alle Sperren.
☐	☒	Sie gibt ihre Sperren frei.
☐	☒	Sie wartet auf das prepare-to-commit von der globalen Transaktion.
		Sie wartet auf das commit von der globalen Transaktion.
☐	☒	Sie sendet ein prepare-to-commit an die globale Transaktion.
		Prepare-to-commit hat bereits vorher stattgefunden. Außerdem wird das prepare-to-commit von der globalen Transaktion aus geschickt.

W	F	Welche der folgenden Aussagen zu den Replikationsverfahren sind wahr, welche sind falsch?
☐	☒	Durch die redundante Speicherung von Daten wird auf jeden Fall die Verfügbarkeit erhöht, unabhängig vom Replikationsverfahren.
		ROWA behindert die Verfügbarkeit, da Updates bei Ausfall eines Knotens nicht mehr durchgeführt werden können.
☒	☐	Wenn die Primary Copy ausfällt, kann ein anderer Knoten zur Primary Copy ernannt werden.
☐	☒	Diese neue Primary Copy synchronisiert sich zunächst mit der bisherigen Primary Copy, damit sie auch tatsächlich auf dem aktuellen Stand ist.
		Da die alte Primary Copy ausgefallen ist, müsste sie dann warten, bis die alte Primary Copy wieder da ist. Dann bräuchte sie erst gar nicht anzufangen, da dann auch die alte Primary Copy weitermachen könnte.
☒	☐	Das ROWA-Verfahren garantiert den Lese-Transaktionen, dass sie immer einen konsistenten Datenbankzustand sehen
☐	☒	Beim Primary Copy Verfahren kann durch die Einbeziehung von „Schnappschuss-Versionen" erreicht werden, dass die Lese-Transaktionen auf den verschiedenen Knoten den aktuellen Datenbankzustand sehen.
		Das zwar nicht, aber zumindest, dass bei veralteten Daten der sichtbare Datenbank-Zustand konsistent ist.

W	F	Fragen zur globalen Serialisierbarkeit:
☒	☐	Wenn in einer globalen Schedule eine der lokalen Schedules nicht serialisierbar ist, dann ist die globale Schedule auch nicht serialisierbar.
☐	☒	Wenn alle lokalen Schedules serialisierbar sind, dann ist auch globale Serialisierbarkeit gegeben. *Hinzukommen muss, dass es eine serielle Reihenfolge der Transaktionen gibt, die bei allen lokalen Schedules gilt.*

W	F	Fragen zur Partitionierung und Allokation:
☒	☐	Eine Relation kann horizontal partitioniert werden durch eine Selektion.
☐	☒	Um sicherzustellen, dass die horizontalen Partitionen durch einen Join wieder zur Ausgangsrelation zusammengeführt werden können, muss in jeder Partition der Primärschlüssel vorkommen. *Horizontale Partitionen werden durch eine Vereinigung zusammengeführt, nicht durch einen Join.*
☒	☐	Bei einer abgeleiteten horizontalen Partitionierung braucht für die Zusammenführung der Partitionen zur Ausgangsrelation kein Join mit der externen Relation durchgeführt werden, die in der Partitionierungsbedingung vorkommt.
☐	☒	Um beim Ausfall eines Knotens sicherzustellen, dass die Daten trotzdem zugreifbar bleiben, werden üblicherweise mehrere Partitionen gemeinsam allokiert. *Es ist umgekehrt: einzelne Partitionen werden mehrfach an unterschiedliche Knoten allokiert.*
☒	☐	Um eine vertikal partitionierte Relation wieder zusammenzufügen, wird in der Regel ein Join über den Primärschlüssel der Ursprungsrelation durchgeführt. Für die Durchführung dieses Joins ist der Semi-Join schlechter geeignet als eine Übertragung aller Teilrelationen an den Zielknoten. *Das ist richtig. Da die Joinbedingung lautet: Primärschlüssel = Primärschlüssel, müssen sowieso alle Tupel übertragen werden. Damit sind in diesem Falle die zusätzlichen Schritte des Semi-Join-Verfahrens nur Overhead, sie bringen aber keinen Nutzen.*

Hinweise zur Bewertung:

Da durch ein zufälliges Ankreuzen leicht ein paar Punkte erworben werden können, werden folgende Regeln bei der Bewertung angewendet:

1. Jede richtige Lösung gibt einen halben Punkt.
2. Für eine falsche Lösung wird ein halber Punkt abgezogen.
3. Falls nichts angekreuzt ist (weder W noch F), so gibt es 0 Punkte (d.h., es wird auch nichts abgezogen)
4. Pro Teilaufgabe (eigenständige Box zum Ankreuzen) ist die minimale Punktzahl 0 (es gibt also keine negativen Punkte und falsche Antworten werden nicht mit richtigen Antworten aus anderen Teilaufgaben verrechnet).
5. Die Gesamtpunktzahl wird auf die nächste ganze Zahl aufgerundet.

Aufgabe 2: Serialisierbarkeit von Schedules (12 Punkte)

Bewertung: 4 Punkte je Teilaufgabe

Globale Serialisierbarkeit setzt zunächst einmal lokale Serialisierbarkeit voraus. Darüber hinaus muss dann eine serielle Reihenfolge gefunden werden, die mit jeweils mindestens einer der lokalen äquivalenten, seriellen Ausführungsreihenfolgen „kompatibel" ist.

a) S_A: $< r_1[a]\ r_2[a]\ w_2[a]\ r_3[b]\ w_3[b]\ w_1[a] >$

$T_1 < T_2$ $T_2 < T_1$

S_A ist nicht serialisierbar, damit existiert auch keine entspr. globale serielle Schedule.

S_B: $< r_1[c]\ w_1[c]\ r_2[d]\ w_2[d]\ r_3[e]\ w_3[e] >$ (unerheblich in diesem Fall)

b) S_A: $< r_1[a]\ r_2[a]\ w_1[a]\ w_2[b]\ r_3[a]\ r_3[b]\ w_3[b] >$

$T_2 < T_3$

$T_2 < T_1$ $T_1 < T_3$

S_A ist serialisierbar: $S_{A,\ seriell}$: $< T_2, T_1, T_3 >$

S_B: $< r_2[c]\ w_2[c]\ r_3[d]\ w_3[d]\ r_1[e]\ w_1[e] >$

T_1, T_2 und T_3 haben paarweise disjunkte Objektmengen, damit ist jede serielle Ausführung der drei Transaktionen eine äquivalente serielle Ausführungsreihenfolge.

Damit ist $< T_2, T_1, T_3 >$ eine globale, äquivalente serielle Ausführungsreihenfolge dieser Transaktionen.

c) S_A: $< r_1[a]\ r_3[b]\ w_1[a]\ w_3[b]\ r_2[a]\ w_2[a] >$

$T_1 < T_2$ (T_3 kann beliebig eingeordnet werden)

S_B: $< r_3[c]\ w_3[c]\ r_2[d]\ w_2[d]\ r_1[d]\ w_1[d] >$

$T_2 < T_1$

Es gibt keine äquivalente globale serielle Schedule für S_A und S_B, welche alle lokalen Abhängigkeiten berücksichtigen kann.

Aufgabe 3: Verteilter Join (24 Punkte)

Punktevergabe:

Aufgabe a): Je Fall 3 Punkte
Aufgabe b): Je Fall 3 Punkte

(a)

Fall 1:
Zunächst findet eine Übertragung der gesamten Tabelle Projekte nach Knoten K_M statt, anschließend wird die Ergebnismenge des Joins zurück an K_P geschickt.

Fall 2:
Zunächst findet eine Übertragung der gesamten Tabelle Mitarbeiter nach Knoten K_P statt, anschließend wird der lokale Join durchgeführt, und die Ergebnismenge des Joins steht damit bereits an K_P zur Verfügung.

Fall 3:
Zunächst wird auf K_P die Projektion auf P-Leiter durchgeführt und das Ergebnis nach K_M geschickt. Dort findet der Semi-Join statt, und das Ergebnis wird wiederum nach K_P geschickt. Dort findet der endgültige Join statt.

Fall 4:
Zunächst wird auf K_P der Bitvektor berechnet. Dieser wird nach K_M geschickt. Dort wird die Hashfunktion auf MNR angewendet, die sich qualifizierenden Tupel werden wiederum nach K_P geschickt, wo der endgültige Join stattfindet.

(b)

Fall 1:

Übertragung 1 von K_P nach K_M:
4.000 (Projekte) * 10.000 (Byte / Projekt)
➔ Ca. 40 MB

Übertragung 2 von K_M nach K_P:
1.000 (Mitarbeiter) * 5.000 (Byte / Mitarbeiter)
➔ Ca. 5 MB

Gesamte Übertragung also: 45 MB.

Fall 2:

Übertragung von K_M nach K_P:
50.000 (Mitarbeiter) * 5.000 (Byte / Mitarbeiter)
➔ Ca. 250 MB

Gesamte Übertragung also: 250 MB.

Fall 3:

Übertragung 1 von K_P nach K_M:
1.000 (P_Leiter-Nummern) * 6 (Byte / P_Leiter-Nummer)
➔ Ca. 6 kB

Übertragung 2 von K_M nach K_P:
1.000 (Mitarbeiter) * 5.000 (Byte / Mitarbeiter)
➔ Ca. 5 MB

Gesamte Übertragung also: ca. 5 MB.

Fall 4:

Zunächst wird auf K_P der Bitvektor berechnet. Er ist etwa zur Hälfte mit 1 und zur Hälfte mit 0 belegt.

Übertragung 1 von K_P nach K_M:
Länge des Bitvektors: ca. 1000 Bits, also < 1KB.

Auf K_M wird die Hashfunktion auf MNR angewendet. Wegen der Gleichverteilung der Schlüssel bei der Abbildung auf den Hashvektor qualifizieren sich etwa die Hälfte der Tupel.

Übertragung 2 von K_M nach K_P:
50.000 / 2 (Mitarbeiter) * 5.000 (Byte / Mitarbeiter)
➔ Ca. 125 MB

Gesamte Übertragung also: ca. 125 MB.

Aufgabe 4: Partitionierung und Allokation (22 Punkte)

a) (16 Punkte)

Die Relation Lieferant wird sinnvollerweise nach dem Attribut *LiefPLZ* horizontal partitioniert, da die Zugriffe der Dispositionsabteilung nach Postleitzahl erfolgen.

$$Lieferant_1 := SL_{LiefPLZ \in [01000, 39999]}\ Lieferant$$

$$Lieferant_2 := SL_{LiefPLZ \in [40000, 69999]}\ Lieferant$$

$$Lieferant_3 := SL_{LiefPLZ \in [70000, 99999]}\ Lieferant$$

Da die Dispositionsabteilungen nur die Bestellungen der Lieferanten ihres PLZ-Bereichs überwachen, sollte die Bestellrelation abhängig von der Postleitzahl des zugehörigen Lieferanten zerlegt werden (abgeleitete horizontale Partitionierung).

$$Bestellung_1 := Bestellung\ NSJ\ (SL_{LiefPLZ \in [01000, 39999]}\ Lieferant)$$

$$Bestellung_2 := Bestellung\ NSJ\ (SL_{LiefPLZ \in [40000, 69999]}\ Lieferant)$$

$$Bestellung_3 := Bestellung\ NSJ\ (SL_{LiefPLZ \in [70000, 99999]}\ Lieferant)$$

Die Teil-Relation sollte zuerst vertikal partitioniert werden, da die Dispositionsabteilungen über den Schlüssel TeilNr nur auf den Bestand zugreifen und die Lager im Gegensatz dazu nur die restlichen Attribute verwenden.

$$Teil_{Bestand} := PJ_{\{TeilNr,\ Bestand\}}\ Teil$$

$$Teil_{Lager} := PJ_{\{TeilNr,\ TeilName,\ LagerNr,\ RegalNr\}}\ Teil$$

Die $Teile_{Lager}$ Partition kann nun noch horizontal nach LagerNr partitioniert werden (gemischte Partitionierung).

$$Teil_{Lager1} := SL_{LagerNr = 1}\ Teil_{Lager}$$

$$Teil_{Lager2} := SL_{LagerNr = 2}\ Teil_{Lager}$$

$$Teil_{Lager3} := SL_{LagerNr = 3}\ Teil_{Lager}$$

b) (6 Punkte)

Die drei Lieferant-Partitionen werden in jedem Fall bei den zugehörigen Dispositionsabteilungen gespeichert. Redundante Speicherung macht keinen Sinn, da nur die jeweiligen Abteilungen auf diese Partitionen zugreifen. Dasselbe gilt auch für die drei Bestell-Partitionen.

Für die Partitionen der Teil-Relation müssen verschiedene Fälle unterschieden werden:

(1) Nicht-redundante Speicherung:

Die drei $Teil_{Lagerx}$-Partitionen werden fast ausschließlich von den entsprechenden Lagern verwendet; daher wird jede dieser Partitionen beim zugehörigen Lager gespeichert. Die Partition $Teil_{Bestand}$ sollte an einem der Knoten der Dispositionsabteilungen gespeichert werden, da diese häufiger als die Lager darauf zugreifen.

(2) Redundante Speicherung:

Die drei $Teil_{Lagerx}$-Partitionen werden wie oben beim zugehörigen Lager gespeichert. Die Partition $Teil_{Bestand}$ kann nun an jedem Lagerknoten und an jedem Knoten der Dispositionsabteilungen redundant gespeichert werden, da sie von allen Knoten benötigt wird.

Aufgabe 5: Tree-Quorum (18 Punkte)

a) (6 Punkte)

Das Schreibquorum ist durch 2 Parameter bestimmt:

- die Anzahl der erforderlichen Ebenen (e) und
- die Anzahl (a) von erforderlichen Zustimmungen je Teilebene (Teilbaum).

e und a sind abhängig von der Höhe (h) und dem Verzweigungsgrad (v) des Abstimmungsbaumes.

Die Parameter des Schreibquorums $Q_W(e_W,a_W)$ müssen den folgenden Nebenbedingungen genügen:

Q_W: $2 * e_W > h$, $2 * a_W > v$
(Behandlung Schreib-Schreib-Konflikte)

Daraus ergibt sich:

$e_W > h / 2$ ➔ $e_W > 2$ ➔ $e_W = 3$

$a_W > v / 2$ ➔ $a_W > 1,5$ ➔ $a_W = 2$

b1) (6 Punkte)

Das erforderliche Quorum kann erreicht werden, ein Update ist daher möglich.

Begründung:

Es wird ein „Level-Quorum" von 3 und ein „Verzweigungs-Quorum" von 2 verlangt.

Ebene 1:Kann durch die Zustimmung der Wurzel erfüllt werden.

Ebene 2: Kann durch die Zustimmung der Knoten (2, 3), (2, 4) oder (3, 4) erreicht werden (jeweils 2 Stimmen).

Ebene 3: Kann in keinem der Teilbäume unter den Knoten 2, 3, oder 4 direkt erreicht werden.

Im Teilbaum unter 4:

13 ist erreichbar und kann Zustimmung geben

12 kann keine Zustimmung geben, da nicht erreichbar, dies kann aber durch

die Zustimmung von 35 und 37 ersetzt werden.

Im Teilbaum unter 2:

5, 6, und 7 können keine Zustimmung geben

Durch Zustimmung von 15 und 16 kann jedoch die Zustimmung von 5 ersetzt werden, und durch die Zustimmung von 17 und 18 wird die Zustimmung von 6 ersetzt.

B2) (6 Punkte)

Das erforderliche Quorum kann nicht erreicht werden, ein Update ist daher nicht möglich.

Begründung:

Es wird ein „Level-Quorum" von 3 und ein „Verzweigungs-Quorum" von 2 verlangt.

Ebene 1:Kann durch die Zustimmung der Wurzel erfüllt werden.

Ebene 2:Hier ist nur Knoten 4 verfügbar (1 Stimme).

Alternative 1: Die Zustimmung von Knoten 3 wird durch die Zustimmung der nächsten Teilbaumebene (2 Stimmen erforderlich) ersetzt:

Knoten 8 und 10 sind erreichbar, damit ist die Zustimmung dieser Ebene gesichert.

Ebene 3:Für Knoten 4, 8 und 10 werden 2 weitere Stimmen auf der nächsten Ebene benötigt:

Knoten 4: 11 und 13 können Zustimmung geben

Knoten 10:29 und 31 können Zustimmung geben

Knoten 8: nur ein Knoten (24) verfügbar, dies reicht nicht aus.

Alternative 2: Die Zustimmung von Knoten 2 wird durch die Zustimmung der nächsten Teilbaumebene (2 Stimmen erforderlich) ersetzt:

Knoten 5 und 6 sind erreichbar, damit ist die Zustimmung der 2. Teilbaumebene gesichert.

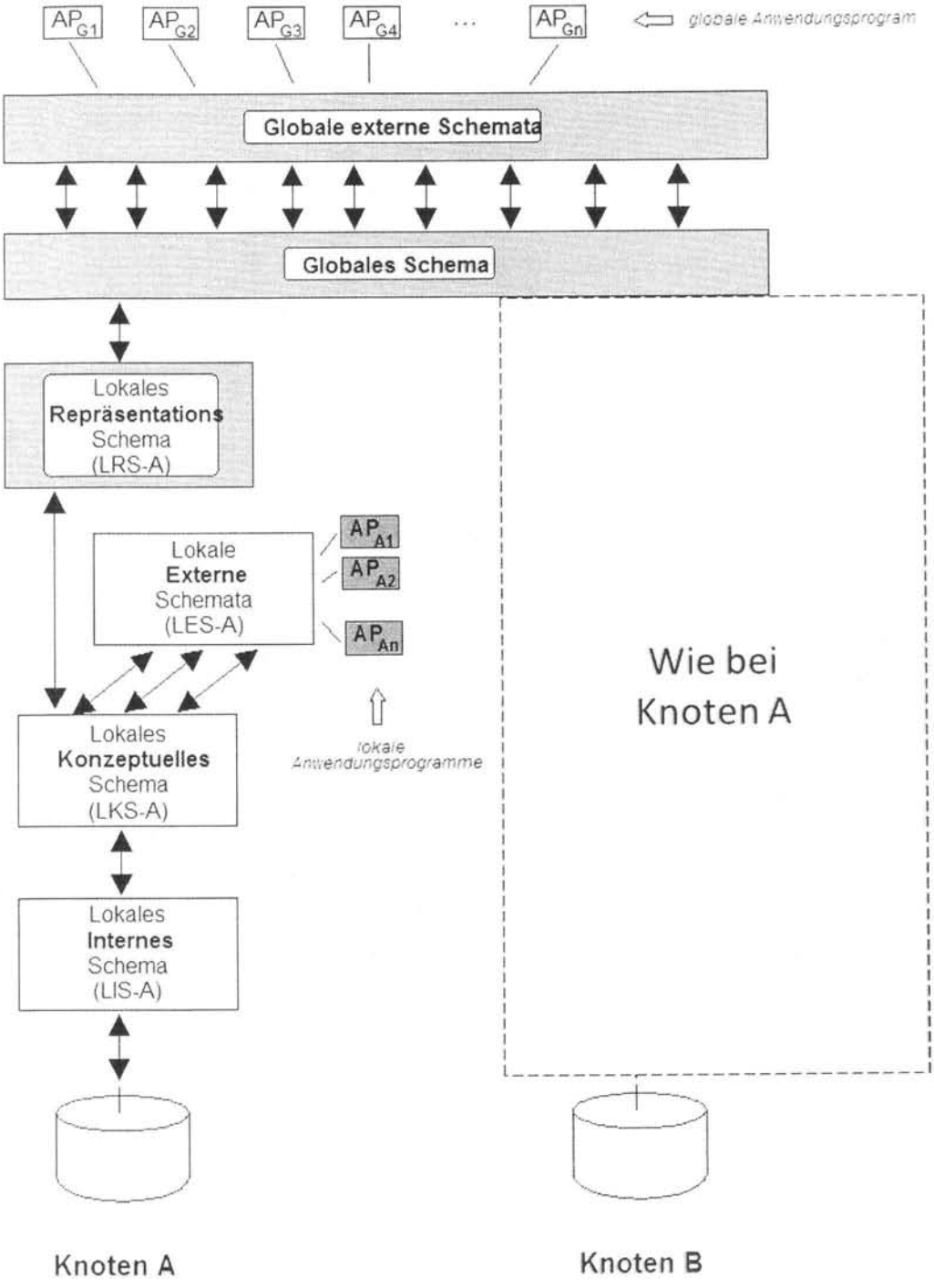
Da aber unter Knoten 6 nur ein weiterer Knoten erreichbar ist, kann die 3. Ebene nicht

gewonnen werden und damit ist das Quorum auch in diesem Fall nicht erreicht.

Damit kann auch mit dieser (letzten) Alternative das erforderliche Quorum nicht erreicht werden.

Aufgabe 6: Globale Schema-Architektur (14 Punkte)

a) (6 Punkte)



b) (4 Punkte)

An jedem Knoten wird zusätzlich zu den existierenden Schemata ein lokales Repräsentationsschema (LRS) eingeführt, das diejenigen lokalen Relationen, die global zur Verfügung stehen sollen, an allen Knoten in einheitlich struktureller Form und mit derselben Semantik „nach außen" repräsentiert. Man bezeichnet das LRS deshalb manchmal auch als Export-Schema.

Das lokale Repräsentationsschema ist als (spezielle) Sicht auf dem existierenden lokalen konzeptuellen Schema definiert. Das jeweils existierende lokale konzeptuelle Schema wird nicht verändert, ebensowenig wie die existierenden lokalen externen Schemata und deren Abbildungen auf das lokale konzeptuelle Schema. Die lokalen Repräsentationsschemata führen also eine Homogenisierung der (Darstellung der) lokalen Schemata durch, sofern diese global verfügbar gemacht werden sollen.

c) (4 Punkte)

Das globale Schema wird intern in drei Subschemata aufgeteilt:

- Das globale konzeptuelle Schema (GKS) realisiert die konzeptionelle Gesamtsicht des verteilten Datenbanksystems.
- Das globale Partitionierungsschema (GPS) beschreibt die Partitionierung der Relationen des GKS.
- Das globale Allokationsschema (GAS) beschreibt die physische Platzierung der Partitionen (die Allokation).