ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. G. Alonso *C. Hagen, B. Kemme*

AR 1: Transaktionsverwaltung

Praktische Übung 2 Lösungsvorschlag

Aufgabe 1: Konsistenz in Oracle

a) Oracle vermeidet Dirty Reads durch die Schreibsperren, die automatisch durch das System gesetzt werden, wenn Update-, Delete- oder Insert-Statements ausgeführt werden. Lost Update kann dadurch nicht vermieden werden, und auch Repeatable Read wird normalerweise nicht garantiert. Oracle spricht von "Statement-Level Read Consistency" und meint, daß ein einzelnes SQL-Statement eine konsistente Sicht der Datenbank hat. Aufeinanderfolgende Statements innerhalb der gleichen Transaktion können jedoch unterschiedliche Zustände sehen. Repeatable Read kann für Nur-Lese-Transaktionen erzwungen werden, indem die Transaktion als read-only deklariert wird (SET TRANSACTION READ ONLY). Für Transaktionen, die auch Änderungen vornehmen, kann vollständige Konsistenz nur durch explizites Setzen von Sperren erreicht werden.

b) Oracle's Konsistenzbegriff ist auf größmögliche Effizienz durch maximale Parallelität von Transaktionen ausgerichtet. Deswegen wird so wenig wie möglich gesperrt. Transaktionen in Oracle sind jedoch nicht Konflikt-Serialisierbar. Beispiele für inkorrekte Schedules finden sich in einigen Beispielen des Aufgabenblattes. Dem Anwendungprogrammierer steht die Möglichkeit offen, mittels SET TRANSACTION READ ONLY bzw durch das explizite Setzen von Sperren die Konsistenz zu garantieren.

c) Das folgende Beispiel zeigt eine Folge von Eingaben in den drei Fenstern (f1, f2, f3). Alle drei Queries am Ende liefern unterschiedliche Ergebnisse:

```
f1: set transaction read only;
f2: update Auto set Farbe = 'rot';
f2: commit work;
f2: set transaction read only;
f3: update auto set Farbe = 'blau';
f3: commit work;
f3: set transaction read only;
f1: select Farbe from Auto;
f2: select Farbe from Auto;
f3: select Farbe from Auto;
```

Wurde die Option SET TRANSACTION READ ONLY gesetzt, speichert ORACLE den Zeitpunkt des Beginns der Transaktion. Leseoperationen der Transaktion sehen den Zustand der Datenbank, wie er diesem Zeitpunkt existierte. Dies wird erreicht, indem der alte Wert aus dem Log gelesen wird, falls sich der Wert eines Tupels seit Transaktionsbeginn geändert hat.

Das Verhalten von read-only-Transaktionen entspricht dem Mehrversionen-Sperrverfahren. (Siehe z.B. [BHG87], Kapitel 5.4) Es läßt sich für diese Art von Transaktionen relativ leicht umsetzen. Mehrversionen-Sperverfahren, die auch Schreiber berücksichtigen, sind wesentlich komplexer, da beim Commit einer Transaktion berücksichtigt werden muß, welche Versionen sie gelesen hat und welche Änderungen andere Transaktionen vorgenommen haben.

d) Bei Nur-Lese-Transaktionen greift Oracle auf die UNDO-Information im Log zu. Da jedoch der Speicherplatz für das Log begrenzt ist, wird er wiederverwendet, indem sehr alte Informationen überschrieben werden. Dabei kann es vorkommen, daß Werte überschrieben werden, die noch von Lesetransaktionen benötigt werden. Man kann dies vermeiden, indem man die Logs groß genug wählt (Systemadministrator) oder explizite Lesesperren auf die betroffenen Tabellen setzt (Programmierer).

Aufgabe 2: Sperrverwaltung in Oracle

a) Oracle verwendet Multigranularity-Sperren, ähnlich wie sie in der Vorlesung vorgestellt wurden. Dabei werden Tabellen und Tupel gesperrt. Im Unterschied zum allgemeinen Konzept der Multigranularity-Sperren gibt es jedoch keine Lesesperren für Tupel verwaltet. Die folgende Tabelle zeigt den Zusammenhang zwischen den Tabellen-Sperren von Oracle und den in der Vorlesung vorgestellten Sperrtypen:

Vorlesung	*Oracle*
IR	RS
IW	RX
R	S
W	X
IRW	SRX

Tabelle 1: Sperrmodi

b) Das System erwirbt vor Leseoperationen keine Lesesperren. Deshalb kann es vorkommen, daß eine Transaktion inkonsistente Ergebnisse wie im obigen Beispiel sieht, wo zwischen zwei Leseoperationen von TA 2 Änderungen von TA 1 vorgenommen wurden. Man kann dies durch die SELECT-FOR-UPDATE-Option ändern. Sie sorgt dafür, daß Schreibsperren für die gelesenen Objekte gesetzt werden und garantiert damit Konsistenz.

c) Wenn man in Transaktion 2 statt des Updates ein Delete ausführt, dann sind die Ergebnisse nicht mehr korrekt. Dies liegt daran, daß Oracle nicht erkennen

kann, daß ein gelöschter Tupel eigentlich zur Lesemenge einer Query-Operation gehört.

d) SELECT FOR UPDATE sorgt dafür, daß für die Tupel einer Leseoperation Schreibsperren gesetzt werden. Dies blockiert andere Transaktionen mit konfligierenden Operationen. Im obigen Beispiel wird auf diese Weise ein Deadlock verhindert, da Transaktion 2 blockiert wird, bis Transaktion 1 beendet ist. Vertauscht man im obigen Beispiel die Operatioen 2 und 3, so tritt jedoch wieder ein Deadlock auf. Dies liegt an der "ungünstigen" Reihenfolge der Operationen.

   Generell lassen sich Deadlocks nur vollständig vermeiden, indem zirkuläres Warten verhindert wird. Dies kann entweder durch "Preclaiming" erreicht werden (alle Transaktionen erwerben alle Sperren, die sie benötigen werden, sofort zu Beginn der Transaktion in *einer* atomaren Sperranforderung), oder durch eine vordefinierte Reihenfolge, in der Sperren erworben werden müssen. Legt man im obigen Beispiel zum Beispiel fest, daß jede Transaktion zuerst ihre Sperren für die "Auto"-Tabelle setzen muß, bevor sie Sperren für die "Halter"- Tabelle setzen kann, so werden Deadlocks vermieden.