Und ewig "locked" die Datenbank - Locking in Oracle R11 und IBM DB2

PROMATIS software GmbH
Ettlingen (TechnologieRegion Karlsruhe)

Schlüsselworte

R11, DB2, Locking, Partitionierung, LOB, Materialized Views, PL/SQL

Einleitung

In vielen Unternehmen befinden sich Datenbanksysteme unterschiedlicher Hersteller im Einsatz. Immer wieder müssen Anwendungssysteme von Produkten eines Herstellers zu denen eines anderen Herstellers migriert werden. Diese Migrationsprojekte benötigen tiefgehende Kenntnisse der realisierten Konzepte in den unterschiedlichen Datenbanksystemen. Oftmals wird dabei in Migrationsprojekten dem unterschiedlichen Verhalten beim Sperren von Datensätzen (Locking) wenig Aufmerksamkeit geschenkt. Dies kann durchaus zu Problemen führen. Der Beitrag führt zunächst in die Systeme der Datenbankhersteller IBM und Oracle ein. Das Locking-Verhalten von Oracle R11 und von IBM DB2 wird genauer betrachtet. Welche Möglichkeiten existieren, wie können Anwendungen möglichst einfach umgestellt werden? Wie können Probleme und Fallstricke umgangen werden? Anhand konkreter Szenarien von parallelen SQL-Befehlen (Insert, Update, Delete, Select) und bestimmten Systemeinstellungen werden die Locking-Konzepte der beiden Datenbanksysteme dargestellt und miteinander verglichen. Abschließend bewertet der Autor, in welchen Fällen welche Art von Maßnahme geeignet ist.

1 Ausgangspunkt

Eigentlich sollte die Migration einer Anwendung von Oracle nach DB2 gar kein Problem darstellen. Schließlich handelt es sich bei den Produkten beider Hersteller um ausgereifte und praxiserprobte relationale Datenbanksysteme, die nur geringfügig voneinander abweichen. Diese Vermutung ist in der Tat vielfach verbreitet und so mancher Softwareentwickler ist der Meinung, dass es vollkommen egal ist, ob die Datenbankschicht blau oder rot ist, in der seine Daten persistiert werden. Dieser Annahme muss leider eine deutliche Absage erteilt werden. Wie diverse Migrationsprojekte zeigen, kommt es insbesondere bei der Migration von Oracle nach DB2 immer wieder zu den gleichen Problemen. Hier liegt die Ursache meistens in den bereits genannten kleinen aber feinen Unterschieden der Systeme. Die Auswirkungen dieser Abweichungen sind meistens jedoch so groß, dass dadurch ein Migrationsprojekt an den Rand des Scheiterns gerät.

2 Locking-Verhalten

2.1 Locking-Verhalten unter Oracle 11g

Das Locking-Verhalten unter Oracle gestaltet sich relativ einfach. Die folgenden Umstände, die jedem Oracle Datenbankadministrator als selbstverständlich erscheinen, bereiten bei einer achtlosen Migration von Oracle nach DB2 in der Regel größere Probleme:

- Führt eine Anwendung gegen eine Oracle DB einen einfachen „Select“ aus, dann führt das zu keinen Locks, es sei denn die Anwendung verwendet einen SELECT FOR UPDATE.
- Die Anwendung bekommt bei einem „Select“ grundsätzlich immer den letzten persistierten (COMMIT) Stand der Daten als Ergebnis geliefert.
- Führt eine Anwendung einen schreibenden Zugriff (UPDATE) durch, dann werden exakt die benötigten Sätze gelockt.

Somit ist klar, dass beim Auftreten von Timeouts oder Deadlocks in der Regel die Ursache nicht in der Datenbank, sondern in der Anwendung zu suchen ist, da parallele Zugriffe auf dieselben Datensätze erfolgen. Tritt dieser Fall ein, so ist die Anwendungsentwicklung aufgefordert zu analysieren, welche Session den "Blocker" und welche andere Session den "Waiter" darstellt. Gegebenenfalls ist zu überlegen, ob die Zugriffe serialisiert werden müssen bzw. welche anderen Konzepte zum Vermeiden von Deadlocks zum Einsatz kommen sollen.

2.2 Locking-Verhalten unter DB2

Das Verhalten von DB2 rund um das Thema Locking weicht hier wesentlich vom Verhalten einer Oracle Datenbank ab. Wobei diesbezüglich noch zu unterscheiden ist, auf welcher Plattfom DB2 zum Einsatz kommen soll. IBM stellt das Produkt DB2 auf z/OS (Großrechnerumfeld) und LUW (Linux Unix Windows) zur Verfügung. In der Regel werden bereits hier die ersten Fehler bei einer Migration begangen. Ein Vorteil der Oracle Datenbank besteht darin, dass sie sich in der Regel auf unterschiedlichen Plattformen identisch verhält. Bei DB2 ist dies nicht der Fall. Die Varianten für z/OS und LUW weichen in zentralen Funktionalitäten signifikant voneinander ab. Dies geht sogar soweit, dass in einigen Fällen wichtige Funktionen nur auf einer der beiden Plattformen verfügbar sind. Im weiteren Verlauf der Betrachtung soll z/OS als DB-Referenz verwendet werden, da hier erfahrungsgemäß die größten Unterschiede zu Oracle zu verzeichnen sind. Bei einem Umstieg auf z/OS sind zunächst die folgenden Feinheiten zu beachten:

- DB2 unterscheidet beim Locking zwischen einem Exclusive Lock und einem Shared Lock. Greift eine Anwendung in DB2 per SELECT auf eine Tabelle zu, dann werden Shared Locks gesetzt. Das bedeutet, dass andere Sessions zwar immer noch lesend – mit weiteren Shared Locks – auf die Datensätze zugreifen können, aber ein Exclusive Lock kann nicht mehr gesetzt werden. D.h. auch bei einem SELECT-Zugriff kann es in der Folge zu Deadlocks kommen, wenn durch eine zweite Session ein schreibender Zugriff versucht wird.
- DB2 beherrscht Row-Level-Locking (RLL) und Page-Level-Locking (PLL). Eine Page in DB2 entspricht in etwa einem Datenblock unter Oracle. Welches der beiden Verfahren zum Einsatz kommt, hängt von der Konfiguration der DB ab. In vielen Fällen sind DB2-Systeme aber auf PLL konfiguriert, da dadurch die Menge an parallel geschriebenen Locks verkleinert werden kann. Dieses Verfahren hat jedoch den Nachteil, dass es bei zwei schreibenden Zugriffen zu Deadlocks kommen kann, wenn die betroffenen Datensätze ungünstiger weise auf der gleichen Datenpage liegen.
- Während bei Oracle das Konzept des Isolation Levels nicht bekannt ist (bei einem SELECT werden immer die zuletzt persistierten Datensätze gelesen), verfügt DB2 über 4 Isolation Levels: Repeatable Read (RR), Read Stability (RS), Cursor Stability (CS) und Uncommitted Read (UR). Will man den Isolation Level unter DB2 explizit steuern, dann ist das im jeweiligen SELECT-Zugriff ausdrücklich anzugeben. RR ist das höchste Level und bedeutet, dass beim Öffnen eines Cursors immer genau die Datensätze geliefert werden, die beim Öffnen des Cursors gültig waren. Auch bei mehrmaligem „Abarbeiten“ des Cursors ist die Menge immer die gleiche, unabhängig davon, was sich in der Zwischenzeit auf der DB verändert hat. Bei RS bleiben die vorhandenen Sätze unverändert, jedoch können beim mehrmaligen „Abarbeiten“ des Cursors neue Rows hinzukommen, sog. Phantom Rows – je nachdem was sich in der Tabelle seit dem ersten Öffnen des Cursors verändert hat. Bei CS wird nur noch garantiert, dass sich die aktuelle Zeile, an der sich die Cursor-Verarbeitung befindet, nicht mehr ändern kann. Alle anderen Zeilen des Cursors können durch andere Sessions geändert werden. RS schließlich, das schwächste Isolation Level, welches auch als Dirty Read bezeichnet wird, liest beim Öffnen des Cursors alle Datensätze in seinem aktuellen Zustand – auch dann, wenn diese von anderen Sessions verändert wurden und aktuell noch kein Commit erfolgt ist. Diese Zugriffsvariante wird deshalb als Dirty Read bezeichnet, da bei einem Rollback (einer anderen Session) die Daten, die im Cursor dargestellt wurden, mittlerweile ungültig sind und die Session, die sich des Cursors bedient, somit mit inkonsistenten Daten arbeitet.

Unter den beschriebenen Isolation Levels entspricht Read Stability am ehesten dem Verfahren, welches von Oracle implementiert wird.

3 Behandlung von Deadlocks nach der Migration

Betrachtet man die Ausführungen zum Thema Locking-Verhalten in DB2 und Oracle, so ist es kaum verwunderlich, dass eine von Oracle nach DB2 migrierte Anwendung, die bislang nie mit Deadlocks zu tun hatte, plötzlich massiv von diesem Problem betroffen ist. In der Regel wird dieses Thema auf die Datenbankadministratoren abgewälzt, da sich vermeintlich an der Anwendung nichts geändert hat. Als Grund wird sehr häufig der Umstand ausgemacht, dass Oracle auf Row-Ebene lockt und DB2 eben im Standard mit Page-Level-Locking arbeitet. In vielen Migrationsprojekten wird basierend auf dieser Annahme der falsche Schluss gezogen, dass ein Umstellen der DB2-Instanz auf Row-Level-Locking das Problem nachhaltig lösen könnte. Diesem Lösungsansatz muss jedoch aus folgenden Gründen eine Absage erteilt werden:

- Kommt es in einer Anwendung zu Deadlocks, dann ist es grundsätzlich die Pflicht der Anwendungsentwicklung genau zu prüfen, wie die Anwendung auf die Daten zugreift, und ob das auf DB2-Seite verwendete Isolation Level korrekt gewählt ist. Durch diese Maßnahme lassen sich in der Regel schon sehr viele Deadlocks in den Griff bekommen.
- Ferner ist zu überlegen, ob die von Oracle 1:1 übernommene Indizierung in der Regel auf die unter DB2 implementierten Zugriffe passt. Darauf zu beharren, dass die Zugriffe unter DB2 genau die gleichen sind, bringt hier nichts. Schließlich unterscheiden sich die Optimizer der beiden Hersteller ebenfalls signifikant und es ist erstaunlich, wie sich Zugriffspläne bei einer Migration verändern können. Somit müssen alle DB2-Zugriffe über einen DB-Trace ausgewertet und ggf. die Indizierung angepasst werden.
- Ein weiteres Verfahren zur Vermeidung von Deadlocks besteht in der Verwendung von Optimistic Locking auf Anwendungsseite. Um dies zu implementieren, sind jedoch weitreichende Anpassungen am Datenmodell der Anwendung erforderlich. So muss in die betroffenen Tabellen ein Kennzeichen aufgenommen werden. In der Regel handelt es sich dabei um einen Timestamp, über den die Anwendung erkennen kann, ob seit dem letzten Zugriff auf den Datensatz andere Sessions schreibend zugegriffen haben. Über dieses Verfahren können Deadlocks in der Regel weiter dramatisch reduziert werden.
- Sind diese Verfahren ausgeschöpft, kann in der Tat als letzte Lösung Row-Level-Locking zum Einsatz kommen, wobei dann die folgenden Aspekte beachtet werden müssen.

Bei der Umstellung auf Row-Level-Locking werden insbesondere von hochtransaktionalen Anwendungen deutlich mehr Locks geschrieben, die von DB2 verwaltet werden müssen. Wie viele Locks auf einer Tabelle bzw. Partition unter DB2 maximal gehalten werden dürfen, regelt ein Konfigurationsparameter der DB2. Dieser Parameter muss je nach Anwendung sorgfältig gewählt werden, da DB2 – falls der Wert für die maximale Anzahl an Locks überschritten wird – automatisch eine Lock-Escalation durchführt und somit wieder statt der betroffenen Datensätze die gesamte Partition bzw. Tabelle gelockt wird. Kommt es vermehrt zu Lock-Escalations, so hat man nichts gewonnen und die Anwendung verhält sich genauso gut oder schlecht wie mit der Einstellung Page-Level-Locking. Ferner ist zu bedenken, dass ein DB2-Administrator den Parameter für die maximale Anzahl an Locks pro Tabelle nur sehr ungern wesentlich erhöht, da jedes Lock Hauptspeicher beansprucht. Ebenso sollte man mit der Umstellung auf Row-Level-Locking bei DB2 vorsichtig sein, wenn man es mit einer sog. Coupling-Facility (CF) zu tun hat. Eine CF entspricht in der Oracle Welt einer RAC Architektur und durch das Umstellen auf Row-Level-Locking kann es in einer solchen Umgebung im Nachgang zu einer stark erhöhten Locking-Kommunikation zwischen den einzelnen Knoten kommen, die bei hochtransaktionalen Anwendungen durchaus zu einer ernsten Belastung für den Betrieb werden können.

4 Weitere Probleme bei einer Migration von Oracle nach DB2

Im Folgenden sollen noch einige weitere Problempunkte bei der Migration einer Anwendung von Oracle nach DB2 beleuchtet werden, die ein ähnliches Konfliktpotenzial bergen, wie beispielsweise das unterschiedliche Locking-Verhalten der beiden Systeme:

4.1 Verwendung von LOBs

Sowohl Oracle als auch DB2 können große Daten über LOB-Objekte in der Datenbank abspeichern. Einer Verwendung von LOB-Objekten unter DB2 steht somit nichts im Weg. Allerdings sollte man bei einer Migration beachten, dass die Verwendung von LOB-Objekten unter DB2 diverse Unterschiede zu Oracle aufweist: So werden z.B. LOB-Objekte unter DB2 unterschiedlich behandelt, insbesondere wenn an bestehende LOBs per Append weitere Teile angehängt werden sollen. Oracle führt einfach eine Append-Funktion durch, während DB2 das komplette LOB neu schreibt – erweitert um den zusätzlichen Teil. Dies kann bei vielen Append-Zugriffen auf LOBs zu einer deutlich verschlechterten Performance führen. Unter DB2 kann es daher erforderlich sein, große LOB-Objekte auf mehrere Teilobjekte aufzuteilen und jeden Teil in einer separaten Row zu speichern. Beim Auslesen des gesamten Objekts müssen daher die unterschiedlichen Fragmente zusammengesetzt werden. Ferner hat DB2 bis zu Version 9 einschließlich Probleme mit der Reorganisation einer LOB-Tabelle. Auch dieser betriebliche Aspekt sollte bei der Planung der Migration beachtet werden.

4.2 Verwendung von Materialized Views

Kommen in der fraglichen Anwendung Materialized Views zum Einsatz, dann sind auch diese nur bedingt nach DB2 übertragbar. Zwar kennt DB2 das Konzept der Materialized View (unter DB2 spricht man von einer Materialized Query Table – MQT), aber Konzepte wie beispielsweise das automatische Refresh einer View unter Verwendung von Materialized View Logs steht unter DB2 nicht zur Verfügung.

4.3 Verwendung von PL/SQL-Code

Viele Oracle-basierte Anwendungen verwenden in der DB den abgespeicherten PL/SQL-Code als integralen Bestandteil der Anwendung, insbesondere für reine Datenmanipulationsfunktionen bei denen keine Interaktion mit einem Client erforderlich ist. Zwar bietet DB2 seit Version 10 auch PL/SQL Stored Procedures an, jedoch hierbei von einer einfachen Migration auszugehen wäre fatal. In vielen Fällen bleibt nur der Ausweg, die betroffenen PL/SQL-Anteile in Java oder Cobol nachzuimplementieren.

4.4 Verwendung von Partitionierung

Ein weiterer großer Unterschied der beiden Plattformen besteht im Bereich der Partitionierung von Tabellen. Unter Oracle stehen hier List-, Hash- und Range Partitionierung zur Verfügung, und es besteht sogar die Möglichkeit verschiedene Partitionierungstechniken hierarchisch anzuwenden. Dabei ist zu beachten, dass DB2 in der Version 10 aktuell lediglich Range Partitionierung und Partitionierung nach Wachstum (Partition by Growth) zur Verfügung stellt. Insbesondere das Fehlen der List Partitionierung führt bei einer Migration zu großen Problemen, da hier auf DB2-Seite in der Regel ein vollkommen neues Partitionierungskonzept erarbeitet werden muss.

4.5 Zeichensätze

Aktuell verwenden viele Oracle Applikationen datenbankseitig einen UTF-Zeichensatz, welcher zunächst relativ einfach auf UNICODE auf DB2-Seite abgebildet werden kann. Wie so oft liegt aber auch hier der Teufel im Detail. So ist hier z.B. zu beachten, dass beim Anlegen einer Tabelle unter DB2 die Größe einer Spalte nur in der Einheit BYTES definiert werden kann. Unter Oracle wird hier bei UTF-Zeichensätzen aber oft die Einheit CHARACTER verwendet. In diesem Fall muss bei der Migration ermittelt werden, um welchen Faktor die betroffenen Tabellen zu erweitern sind.

4.6 Datentypen

Bei den Datentypen ist es in der Regel möglich, eine einfache Abbildung von Oracle nach DB2 vorzunehmen. Wie jedoch die Beispiele VARCHAR und LOB schon gezeigt haben, gestaltet sich diese Abbildung nicht immer ohne Probleme. Vorschläge zur Abbildung von Oracle auf DB2-Datentypen können der einschlägigen Literatur entnommen werden. Exemplarisch seien an dieser Stelle nur zwei Beispiele genannt:

- DATE: Der Datentyp DATE unterscheidet sich auf beiden Plattformen signifikant. Unter Oracle werden in einer Variablen vom Typ DATE zusätzlich Informationen zur Uhrzeit abgelegt. Unter DB2 handelt es sich dabei tatsächlich nur um ein Datum.
- TIMESTAMP: Unter DB2 gibt es seit der Version 10 die Möglichkeit bei der Anlage einer Spalte vom Typ TIMESTAMP eine Präzision mitzugeben. Auch dieser Umstand ist bei der Migration von TIMESTAMP Oracle nach TIMESTAMP DB2 zu beachten.

5 Zusammenfassung

Das Locking-Verhalten in DB2 und Oracle unterscheidet sich an verschiedenen Stellen signifikant voneinander. Vernachlässigt man diesen Tatbestand bei einer Migration der Datenhaltung einer Anwendung von der Oracle nach DB2, dann sind größere Probleme vorprogrammiert, deren Lösung nicht alleine durch den Datenbankadministrator vorgenommen werden kann. In der Regel besteht bei derartigen Projekten sogar die Notwendigkeit, Anwendungscodes zu modifizieren. Abseits der Locking-Thematik gibt es noch diverse weitere Plattformunterschiede, die bei einer Migration zu beachten sind. Bevor mit der Migration einer Anwendung begonnen wird, ist eine Voranalyse der zu erwartenden Problemfälle empfehlenswert.

Kontakt

PROMATIS software GmbH
Pforzheimer Str. 160
76275 Ettlingen, Deutschland

Tel.: +49 7243 2179 0
Fax: +49 7243 2179 99

mailto:info@promatis.de
www.promatis.de
www.horus.biz

Stand der Dokumentation: April 2014

