1000 mal schneller
Praxisgeschichten aus der Oracle-Datenbankwelt

Matthias Schulz
Schulz IT Services GmbH
Nürnberg

Schlüsselworte
Oracle Datenbank; ETL; ELT; IT-Probleme; Lösungen; Praxisbeispiele; Wartezeiten; Lastverteilung; Skalierbarkeit; Performance; Arbeitsteilung; Parallelisierung; Direct Path Inserts (INSERT APPEND;

Einleitung
Unterhaltsame und lehrreiche Geschichten aus der Welt der Oracle Datenbank:

1. *Ein fauler Apfel reicht*

   Ein einzelnes kleines Programm bremst die gesamte Datenbank aus und das ohne selbst aufzufallen. [OLTP]

2. *Einer für alle?*

   Wie die Performance zum Glücksspiel wird, wenn eine einzelne Query alle Aufgaben lösen soll. [OLTP]

3. *Ich hatte eine Lösung, aber sie passte nicht zum Problem*

   Wenn der falsche Zeitpunkt der Erstellung von Tabellenstatistiken die Query-Laufzeit von Minuten zu Tagen verlängert. [ETL/ELT; DWH]

4. *Egoisten unter sich*

   Wie sich die Dateiladeprozesse (ETL/ELT) durch mehrfache gleichzeitige Insert-Append in eine Tabelle drastisch beschleunigen lassen, obwohl das eigentlich nicht geht. [ETL/ELT; DWH]

5. *1000 mal schneller*

   Wenn Abfragen viel Zeit damit verbringen Datensätze zu erzeugen die am Ende niemand sieht. [OLTP; DWH]

Ein fauler Apfel reicht

Kann ein einzelnes kleines Programm, dass alle 10 Minuten läuft und dabei selbst nie auffällig wird, alle anderen Datenbankprozesse ausbremsen?

Das Datenvolumen wächst unaufhörlich, genauso wie die Ansprüche an die Verarbeitungsgeschwindigkeit. Dies stellt immense Herausforderungen an die moderne IT und vor allem an die Datenbanken.

Und dennoch wird gerade bei den Datenbanken oft übersehen, das ein fauler Apfel reicht!

Der faule Apfel bei Datenbanken ist meist schlechtes SQL. Baufehler in SQL-Abfragen (Queries) oder Ladeprozessen (ETL/ELT) führen zu langen Laufzeiten und Wait-Events und werden leicht bemerkt.

Aber es gibt einen Performancekiller, der selbst kaum CPU, I/O oder sonstige Ressourcen verbraucht und dessen SQL-Abfragen daher auch nicht im geringsten auffallen. Perfekt getarnt bremst er allen laufenden Prozesse aus. Alles läuft zäh, wie mit angezogener Handbremse.

Der Name des Performancekiller lautet „massenhafte Hard Parses“.

Was macht Hard Parses so problematisch? Und wie findet man sie Verursacher?

Einer für alle?

Einer für alle, warum nicht auch bei SQL? Weil dabei die Performance schnell zum reinen Glücksspiel wird.

Die neue Version einer Unternehmensanwendung wird ausführlich getestet und läuft über Monate hinweg völlig problemlos. Doch eines Tages laufen Abfragen die gerade noch unter einer Sekunden liefen plötzlich über 20 Minuten! Prozess um Prozess kommt hinzu und innerhalb weniger Minuten geht nichts mehr. Ein Hochlast-OLTP-System bricht zusammen.

Was ist passiert? Und warum? Und wie kann man Katastrophen dieser Art verhindern?

Ich hatte eine Lösung, aber sie passte nicht zum Problem

Wenn der falsche Zeitpunkt der Erstellung von Tabellenstatistiken die Query-Laufzeit von Minuten zu Tagen verlängert.

Ein Ladeprozess läuft plötzlich statt Minuten mehrere Tage lang. Nach Abbruch und Neustart ist alles wieder gewohnt schnell, doch das Problem kommt immer mal wieder. Nach genauer Analyse zeigt sich, dass der Oracle Query Optimizer hier „je nach Laune“ zwei völlig verschiedene Ausführungspläne erstellt hat.

Aber warum? Gerade bei Ladeprozessen ist der Zeitpunkt der Erstellung von Statistiken entscheidend. Oft werden alle Tabellenstatistiken von einem zentralen Job erstellt. Wenn nun zum Zeitpunkt der Statistikerstellung die Tabelle gerade leer ist, so wird für die Anzahl der Zeilen der Wert 1 eingetragen. Im Ladeprozess wird die Tabelle nun zwar mit Millionen Datensätzen gefüllt, nur leider weiß der Oracle Query Optimizer nichts davon - in der Statistik steht ja „1“!

Wie erhält man stabile Laufzeiten? Und warum sind korrekte Statistiken dabei so wichtig?

Egoisten unter sich

Wie sich die Dateiladeprozesse durch mehrfache gleichzeitige Insert-Append in eine Tabelle drastisch beschleunigen lassen, obwohl das eigentlich nicht geht.

1,5 Terrabyte in tausenden Dateien müssen so schnell wie möglich in die Datenbank geladen werden. Die Dateien werden von 20 parallelen Prozessen geladen, aber die erreichte Performance reicht nicht aus.

Der Einsatz von „direct path inserst“ (INSERT APPEND) erhöht den Durchsatz eines Prozesses pro Tabelle zwar deutlich, aber nun kann nur noch eine Datei in eine Tabelle zur gleichen Zeit geladen werden. Das Ziel wird auch so nicht erreicht.

Ideal wäre, wenn alle Prozesse gleichzeitig direct path Inserts in die gleiche Zieltabelle ausführen könnten, aber das geht doch eigentlich gar nicht? - Geht nicht gibt’s nicht!

1000 mal schneller

Wenn Abfragen viel Zeit damit verbringen Datensätze zu erzeugen die am Ende niemand sieht.

Langezeit lebte sie im verborgenen, eine kleine teure SQL-Abfrage, die niemanden je aufgefallen wäre, wenn sich nicht eines Tages die Datenmengen geändert hätten. Ein verwendeter View kombinierte die Datensätze seiner zu Grunde liegenden Tabellen nun zu mehreren Millionen Zeilen, nur um diese dann durch eine Gruppierung (GROUP BY) wieder bis auf wenige Zeilen zusammenzufassen. Millionen Zeilen der Kind-Tabelle, die gelesen, in den Temp-Tablespace geschrieben, sortiert und gruppiert wurden, obwohl sie eigentlich nicht benötigt wurden.

Der Grund war die Verwendung eines Views mit Vater-Kind-Beziehung jedoch ohne Fremdschlüssel (Foreign Keys). Foreign Keys helfen dem Query-Optimizer effiziente Abfragen zu erstellen und nicht benötigte Kind-Tabellen aus Queries zu eliminieren. Sie sind jedoch teuer, da sie bei allen DML-Operationen geprüft werden und die hierfür benötigten Indexes belegen zusätzlichen Speicher und bremsen DML-Operationen noch weiter aus. Aus diesen Gründen wird in vielen Data Warehouse Systemen auf Foreign Keys verzichtet.

Lässt sich das Dilemma aus DML-Performance und Optimizer-Information lösen?

Kontaktadresse:
Matthias Schulz
Schulz IT Services GmbH
Tauberstraße 28
D-90449 Nürnberg

Telefon: +49 (0) 911-384 9090
E-Mail info@schulz-it-services.de
Internet: www.schulz-it-services.de