

Bevor wir richtig zur Sache kommen, ist es sinnvoll zu klären, was wir unter SQL-Tuning verstehen. Das tun wir in diesem Kapitel.

2.1 SQL-Tuning: Definition und Ziele

Peter: „Das ist doch klar, was SQL-Tuning ist.“

Autor: „Könntest du bitte deine Definition geben.“

P.: „SQL-Tuning ist ein Prozess, der die Ausführungspläne bestmöglich optimiert.“

A.: „Was verstehst du denn unter der Optimierung?“

P.: „Natürlich Reduzierung der Laufzeit der jeweiligen SQL-Anweisung.“

A.: „So selbstverständlich ist das nicht. Wenn mehrere Prozesse gleichzeitig eine SQL-Anweisung ausführen und um gewisse Datenbankressourcen konkurrieren, kann man sich in einigen Situationen für einen nicht optimalen Ausführungsplan entscheiden. Somit verschlechtert sich die Laufzeit einer Ausführung, aber die Wartezeit wegen der Konkurrenz sinkt, sodass sich die Gesamtlaufzeit der konkurrierenden Prozesse verbessert.“

P.: „Wenn das die einzige Bemerkung ist ...“

A.: „Wesentlich weniger gefällt mir deine bestmögliche Optimierung.“

P.: „Warum?“

A.: „Aus mehreren Gründen. Erstens ist dieses Kriterium nicht notwendig für das SQL-Tuning. Es reicht vollkommen eine akzeptable Performanz aus, welche mit SQL-Tuning erreicht wird. Deine Definition lockt dich in eine Falle: Häufig weiß man nicht, was die bestmögliche Optimierung ist. Dementsprechend weiß man nicht, ob der Tuning-Prozess beendet ist oder nicht. Außerdem: Das, was auf einem System optimal

ist, kann auf einem anderen nicht optimal sein. Die Definition muss sich also auf ein bestimmtes System beziehen.“

P.: „Du hast meine Definition komplett ruiniert.“

A.: „Warte, Peter. Ich bin noch nicht fertig. Wir müssen noch sagen, dass wir in diesem Buch unter SQL-Tuning das Tuning der einzelnen SQL-Anweisungen verstehen.“

P.: „Ist das nicht selbstverständlich?“

A.: „Nein. Stelle dir vor, dass dein System nach einer Oracle-Migration inperformant geworden ist. In diesem Fall hilft das Tuning der einzelnen SQL-Anweisungen nicht, weil das normalerweise zu viele sind. Man muss also nach einem Grund (oft nach einem Optimizer-Feature) suchen, der das Problem verursacht. Das ist ein anderer Prozess, obwohl man ein ähnliches Prinzip wie bei dem formalen SQL-Tuning bei dieser Analyse gebrauchen kann (s. das Beispiel im ► Anhang).“

P.: „Könntest du bitte definieren, was du unter SQL-Tuning verstehst. Die Kritik muss doch konstruktiv sein.“

A.: „Bevor ich das tue, möchte ich noch einen Aspekt des SQL-Tunings erwähnen. Die Verbesserungsmaßnahmen für einen Ausführungsplan dürfen keine anderen verschlechtern. Dies kann bei akuten Performanz-Problemen besonders kritisch sein, bei denen man keine Zeit für das Testen der Verbesserungsmaßnahmen hat. Jetzt kann ich formulieren: Unter SQL-Tuning verstehen wir einen Prozess, bei dem eine akzeptable Performanz einer SQL-Anweisung auf einem System erreicht wird. Verbesserungsmaßnahmen für eine SQL-Anweisung dürfen dabei keine anderen beeinträchtigen. Wir fangen direkt mit einer inperformanten SQL-Anweisung an. Wie man an die problematischen SQL-Anweisungen kommt, wird ausführlich in [1] erklärt.“

2.2 SQL-Tuner

Es gibt drei Parteien, die SQL-Tuning durchführen: Oracle, Entwickler und Datenbank-administrator. In diesem Abschnitt besprechen wir kurz, wie jeder von diesen SQL-Tunern das tut.

2.2.1 Oracle

Peter: „Macht Oracle SQL-Tuning? Das ist neu für mich.“

Autor: „Der Optimizer von Oracle erstellt performante Ausführungspläne und erleichtert somit enorm das Leben der anderen zwei SQL-Tuner.“

P.: „In meinen Augen ist das kein SQL-Tuning. Wir sind uns doch einig, dass das SQL-Tuning für eine Verbesserung der inperformanten Ausführungspläne zuständig ist.“

A.: „Der Optimizer wird immer mehr mit Elementen des automatischen Tunings bereichert.“

P.: „Was meinst du damit genau?“

A.: „Zum Beispiel das Feature Statistics Feedback (Cardinality Feedback in Oracle 10 und 11). Das ist ein Teil der automatischen Reoptimierung (automatic reoptimization) in 12c. Wenn Oracle merkt, dass die reale Kardinalität bei der ersten Ausführung weit von seiner Schätzung liegt, berechnet der Optimizer einen neuen Ausführungsplan, in dem die bei der ersten Ausführung gewonnene Kardinalität berücksichtigt wird, und wendet diesen Plan bei der nächsten Ausführung an. Ziehen wir noch ein Feature in Betracht: Adaptive Plans. Zur Ausführungszeit entscheidet Oracle, welche Join-Methode (Nested Loop oder Hash) am besten anzuwenden ist.“

P.: „Das sind lediglich einige Elemente von SQL-Tuning.“

A.: „Bei Oracle gibt es noch SQL-Tuning-Advisor. Dieser Advisor kann in zwei Modi funktionieren: Im automatischen und im manuellen. Beim Tuning macht Oracle einige Verbesserungsvorschläge (z. B. zu Zugriffspfaden), ermittelt Statistiken, die spezifisch für die jeweiligen SQL-Anweisungen sind (sprich: für die jeweiligen Prädikate), und speichert diese Statistiken in Form von Optimizer-Hints in einem SQL-Profile ab. Diese Statistiken stellen Anpassungen oder Korrekturen der Optimizer-Schätzungen im jeweiligen Ausführungsplan dar. SQL-Profiles haben wir bereits ziemlich ausführlich in [1] besprochen. Ich vermute, du hast das vergessen.“

P.: „Ich habe das tatsächlich vergessen. Ich nehme meine Einwände zurück: Oracle macht SQL-Tuning. Kannst du etwas zur Qualität dieses Tunings sagen?“

A.: „Die Qualität des automatischen SQL-Tunings ist in Oracle 11 und 12 merkbar angestiegen. Das grundlegende Prinzip der Ermittlung von spezifischen Statistiken ist absolut richtig und effektiv. Es ist kein Wunder, dass dasselbe Prinzip in einigen anderen Methoden vom SQL-Tuning benutzt wird (z. B. in [4]).“

P.: „Wofür sind solche Methoden gut? Oracle hat das ja bereits implementiert.“

A.: „Diese Methoden sind dem automatischen SQL-Tuning von Oracle ähnlich, aber nicht identisch damit. Man darf auch nicht vergessen, dass das automatische SQL-Tuning von Oracle lizenzpflichtig ist. Die jeweiligen Methoden kann man beispielsweise einsetzen, wenn man keine Tuning-Pack-Lizenz hat.“

P.: „Wenn solche Methoden so gut sind, wofür braucht man noch das formale SQL-Tuning?“

A.: „Ich kann ein paar Gründe nennen. Erstens hat jede Methode ihre Nachteile und muss nicht immer helfen. Zweitens kann die Ermittlung der spezifischen Optimizer-Statistiken unakzeptabel lange dauern (z. B. im Fall eines Join von vielen großen Tabellen). Unserer Meinung nach kann man solche Ermittlungen vermeiden, falls die Laufzeitstatistiken im Ausführungsplan bereits zur Verfügung stehen. Dies erspart viel Zeit. Es gibt noch einen Grund. Das formale SQL-Tuning ist sehr einfach. Die Analyse der Laufzeitstatistiken im Ausführungsplan basiert auf einigen einfachen Regeln und kann von jedem Entwickler oder Datenbankadministrator durchgeführt werden. Aus diesen Gründen bahnen wir unseren eigenen Weg in das SQL-Tuning (Abb. 2.1).“

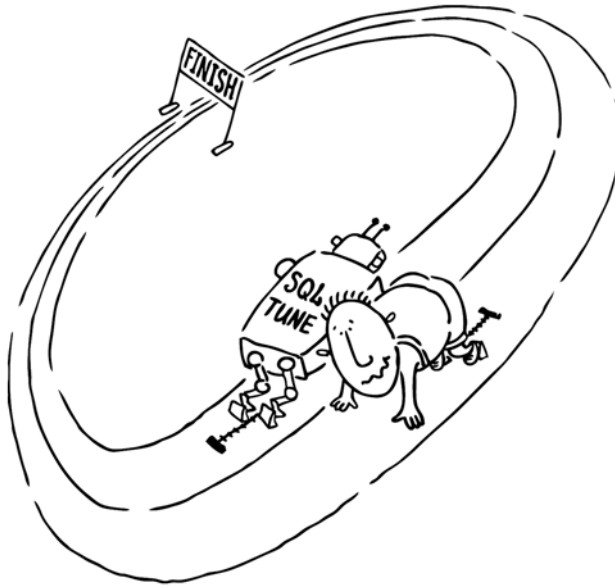


Abb. 2.1 Verschiedene Wege – dasselbe Ziel

2.2.2 Entwickler

Autor: „Ein Entwickler hat sehr gute Voraussetzungen für erfolgreiches SQL-Tuning. Er kennt sich mit dem jeweiligen Datenmodell aus und weiß häufig sofort, wie ein performanter Ausführungsplan aussehen soll. Wenn das Tuning eine Umformulierung der SQL-Anweisung verlangt, ist das für ihn auch kein großes Problem, weil er normalerweise über umfangreiche SQL-Kenntnisse verfügt.“

Peter: „Hier bin ich absolut deiner Meinung. Deswegen wende ich mich oft an die Entwickler, wenn eine SQL-Anweisung inperformant läuft.“

A.: „Bedeutet das, dass du gar nicht versuchst, selbst das Problem zu lösen?“

P.: „Meine Kenntnisse im SQL-Tuning reichen dafür meistens nicht aus. Außerdem bin ich der Meinung, dass SQL-Tuning eigentlich zu den Aufgaben von Entwicklern gehört.“

A.: „Du bist also sicher, dass ein Entwickler wesentlich besser als ein Datenbankadministrator das SQL-Tuning anwenden kann.“

P.: „So ist meine Meinung.“

A.: „Dann versuche ich, deine Meinung zu ändern. In diesem Abschnitt zeige ich einige Nachteile vom Entwickler als SQL-Tuner. Im nächsten erkläre ich, warum ein Datenbankadministrator auch erfolgreich im SQL-Tuning sein kann.“

P.: „Welche Nachteile meinst du? Du hast doch selber gesagt, dass ein Entwickler sehr gute Karten beim SQL-Tuning hat.“

A.: „Wenn kein Entwickler in Greifweite ist, nützt dir das nichts. Was machst du in so einer Situation?“

P.: „Ich stecke dann in Schwierigkeiten. Das ist aber kein Nachteil für den Entwickler.“

A.: „Je nachdem, wie man das betrachtet. Ein Entwickler muss dir ständig für das SQL-Tuning bereit stehen. Das ist nicht immer der Fall.“

P.: „Das ist mir klar. Ich hoffe aber, dass du noch ein paar Nachteile nennen kannst.“

A.: „Stelle dir vor, dass eine Software, welche für relativ kleine Firmen entwickelt wurde, bei einer großen Firma eingesetzt wird. In diesem Fall unterscheiden sich das Datenvolumen und die Datenverteilung vom Standard, an den ein Entwickler gewöhnt ist. Dann sind seine Vorteile beim SQL-Tuning zumindest zum Teil weg. Ich habe mehrmals erlebt, dass ein Entwickler mit solch einer Situation Probleme hatte. Es wurde dann auf angebliche Engpässe der Hardware oder eine falsche Parametrisierung der Datenbank verwiesen. Als ein Datenbankadministrator hinzugezogen und die Sache genauer analysiert wurde, stellten sich trotzdem Probleme im SQL-Bereich heraus.“

P.: „Das habe ich auch erlebt.“

A.: „Ein Entwickler neigt nicht selten zu unnötigen Änderungen am SQL-Text beim SQL-Tuning.“

P.: „Ist das schlecht?“

A.: „Wenn ein Performanz-Problem akut ist, muss es möglichst schnell behoben werden. Für eine Programmänderung benötigt man aus organisatorischen Gründen in der Regel einige Tage. Diese Tage musst du als Verantwortlicher für die jeweilige Datenbank irgendwie überleben.“

P.: „Wie soll denn das SQL-Tuning praktiziert werden?“

A.: „Das erkläre ich gleich im nächsten Abschnitt.“

2.2.3 Datenbankadministrator

Autor: „Ich höre immer wieder, dass kein SQL-Tuning ohne Datenmodellkenntnis möglich ist. Leider ist diese Meinung sehr verbreitet. Was denkst du dazu, Peter?“

Peter: „Na ja, einiges über das Datenmodell muss man schon wissen.“

A.: „Ich bin hingegen sicher, dass das nicht notwendig ist. Das formale Verfahren, das wir in diesem Buch beschreiben, setzt keine Datenmodellkenntnisse voraus.“

P.: „Das kann ich mir kaum vorstellen.“

A.: „In der Praxis sehe ich oft, dass die Datenbankadministratoren sehr gern das SQL-Tuning den Entwicklern überlassen, weil sie fest überzeugt sind, dass sie selbst keine Chance beim SQL-Tuning haben. Die Entwickler haben ihrerseits nichts dagegen, wenn die Datenbankadministratoren diese Aufgabe übernehmen (Abb. 2.2).“

P.: „Wer soll denn das SQL-Tuning durchführen?“

A.: „Der Datenbankadministrator ist verantwortlich für die Datenbank. Es liegt also in erster Linie in seinem Interesse, dass die Datenbank performant läuft.“

P.: „Du denkst also, dass der Datenbankadministrator das SQL-Tuning übernehmen soll.“

A.: „Der Datenbankadministrator soll mit dem SQL-Tuning anfangen. Das ist besonders wichtig, wenn keine Entwickler zu erreichen sind oder das jeweilige Performanz-Pro-

Abb. 2.2 Erst nach Ihnen

blem akut ist. Das formale SQL-Tuning ermöglicht ihm diese Aufgabe. In den meisten Fällen kann er erfolgreich allein das SQL-Tuning zu Ende bringen. Nur sehr selten geht es ohne Entwickler nicht.“

P.: „Wann denn?“

A.: „Falls man zwecks Tuning die SQL-Anweisung ändern muss. Es ist auch möglich, dass das Datenmodell für die jeweilige SQL-Anweisung so ungünstig ist, dass sie kaum zu tunen ist. In diesem Fall muss man entweder das Datenmodell ändern oder die jeweilige SQL-Anweisung komplett umschreiben. Dafür ist ein Entwickler notwendig.“

P.: „Ich muss mich mit diesem Gedanken abfinden, was mir noch schwer fällt. Der Datenbankadministrator ist doch im Vergleich zum Entwickler sehr benachteiligt, was das SQL-Tuning angeht.“

A.: „Welche Nachteile außer fehlenden Datenmodellkenntnissen, welche gar nicht notwendig für das SQL-Tuning sind, kannst du noch nennen?“

P.: „Ein durchschnittlicher Datenbankadministrator ist nicht sehr gewandt in SQL. Er ist imstande, lediglich einfache SQL-Anweisungen zu programmieren.“

A.: „Oft ist das vollkommen ausreichend, weil man in den meisten Fällen ohne Änderungen an der SQL-Anweisung tunen kann.“

P.: „Wie ist das möglich?“

A.: „Das besprechen wir im Abschnitt „Die Richtlinie: Tuning möglichst ohne Änderungen der SQL-Anweisung“. Ich glaube, wir haben genug über den Begriff „SQL-Tuning“ diskutiert. Fangen wir jetzt mit dem formalen SQL-Tuning an. Zunächst muss ich aber einiges über Ausführungspläne erzählen.“

Formales SQL-Tuning für Oracle-Datenbanken

Praktische Effizienz - effiziente Praxis

Nossov, L.; Ernst, H.; Chupis, V.

2016, XIV, 111 S. 110 Abb. in Farbe., Hardcover

ISBN: 978-3-662-45291-2