Faktenbasierte Indexierung - Ein Erfahrungsbericht

Faktenbasierte Indexierung zu erstellen, gehört wohl zu den Königsdisziplinen der IT. Dass sie sich in verschiedener Hinsicht auszahlt, zeigt der nachfolgende Bericht eindrücklich auf.

Von Lothar Flatz, Diso AG

1. Einleitung

Während meiner zehnjährigen Tätigkeit für Oracle Consulting war ich mit vielen spannenden Aufgaben betraut. Da gab es zum Beispiel die Mitarbeit an der Software zur Auswertung der Messdaten beim CERN. Oder eben die **komplette Neuindexierung einer grossen Onlinedatenbank**. Als die Anfrage kam, überlegte ich zuerst, immerhin wollte keiner meiner sehr erfahrenen Kollegen bei Oracle Consulting diesen Auftrag übernehmen. Doch wer nicht wagt, der nicht gewinnt. Das Resultat: Die Ziele meines Auftraggebers wurden erreicht, ja, es wurde ein sensationeller Erfolg. Nach meiner ehrenvollen Berufung in die Real Performance Group der Oracle, als einer von fünf Europäern, bekam ich andere Aufgaben, sodass ich leider keine Möglichkeit hatte, weitere Gesamtindexierungen zu machen.

Ausgangslage

Der nachfolgende Text bezieht sich auf die Indexierung einer Onlinedatenbank. Wenn also von Index die Rede ist, dann ist damit der konventionelle B* Index gemeint. Für ein Data Warehouse gelten andere Indexierungsregeln, diese würden den Rahmen dieses Artikels sprengen.

Kaffeepause mit Folgen

Wir sassen gemütlich bei einem Kaffee bei einem Stammkunden bei dem ich als Oracle Berater einen Tuning Auftrag hatte. Der erfahrene DBA-Spezialist, völlig entspannt, kam wieder einmal auf sein Lieblingsthema. „*Ich wette, mindestens 30 % unserer Indexe sind überflüssig. Die Folge: Unsere Ausführungspläne sind instabil. Und das braucht Platz. Am liebsten würde ich die Indexierung komplett überprüfen lassen“.*

Trifft das zu? Ja. Tatsächlich werden bei eingekauften Applikationen die Indexe in der Regel von den Entwicklern erstellt. Die Entwickler erleben das System selten unter Volllast. Zudem ist die Indexierung **nie an die Bedürfnisse eines bestimmten Kunden** angepasst. Das bedeutet: Wir müssen nachindexieren. Aber: Die Grundlage, die Entwickler geschaffen haben, wird nicht mehr infrage gestellt. Es kommen lediglich neue Indexe dazu. Das Resultat: Ein **unübersichtlicher, dichter Dschungel aus Indexen.**

Was wäre wenn?
Wie wäre es, einmal alles auf Start zu setzen und von Beginn weg richtig zu machen? Die Fakten dazu haben wir. Das Werkzeug auch, ist die Oracle Datenbank ja eine unerschöpfliche Quelle von Informationen. Über die Informationen, die im Shared Pool und in AWRs gespeichert sind, müsste eine geeignete Indexierung erstellt werden können.

Der Gedanke liess mich nicht mehr in Ruhe. Einige Monate später kam die Chance, auf die ich gewartet hatte. Einer meiner Kunden hatte genug von halbherzigen Lösungen. Er wollte Ordnung in der Indexstruktur seiner wichtigsten Applikationen. Zugegeben: Am Anfang brauchte ich meine ganze Überredungskraft, um den Kunden davon zu überzeugen, dass die Indexierung von Grund auf neu gemacht werden muss. Denn ich war überzeugt, dass wir uns bei Verbesserungen auf Basis der bestehenden Indexierung nur verzetteln würden. Welcher Index sollte gelöscht werden, welcher nicht? Solche Diskussionen würden den Prozess unnötig lähmen und ineffizient gestalten. Das wusste ich aus Erfahrung.

Natürlich hätten wir nun einfach einen grossen SQL Tuning Set erstellen können und diesen durch den SQL Tuning Adviser abarbeiten lassen. Aber, dann hätten wir uns blind auf die Entscheide SQL Tuning Adviser verlassen müssen. Auch wir hätten das Wissen und die Erfahrungen der Entwickler und der DBAs nicht verwenden können.

Dazu die Zusammenfassung der Argumente aus der Fachliteratur:
First, while the automated tools reduce the complexity of the physical design process, it is still nontrivial to identify a representative workload that can be used to drive the physical design in its entirety. Second, automated tools do not consider all factors that impact physical design (e.g., the impact of replication architectures). [3].

Neue Schweizer Forschungen weisen darauf hin, dass die automatische Lösung mittels Tools für grosse Gesamtaufgaben problematisch ist. [1], [2]

Wir haben uns entschlossen, ein halbautomatisches Vorgehen zu wählen. So konnte jeder Schritt erklärt und nachvollzogen werden. Jedes Teammitglied konnte seine Gedanken einbringen. Am Schluss hatten wir eine klar begründete Lösung, zu der jeder stehen konnte.

2 Chronik einer Gesamtindexierung

2.1 Vorbereitungsphase

Zunächst wurde ein Team aus allen Beteiligten gebildet. Dieses bestand aus zwei Vertretern aus dem Entwicklungsteam des Applikationsherstellers und zwei DBAs als Vertreter des Softwarebetreibers und ich als externer Berater für den Datenbankhersteller.

In zwei Sitzungen wurde das grobe Vorgehen bestimmt und die Grundregeln der Indexierung festgelegt:

- Primär - und Fremdschlüssel bekommen grundsätzlich automatisch einen Index
- Namenskonventionen
- Design, das der Applikation Rechnung trägt
- Art der physischen Speicherung (zum Beispiel tablespace)

2.2 Datensammlungsphase

In dieser Phase wurden möglichst viele Informationen über die Abfragebedingungen auf der Datenbank gespeichert. Dabei war es wichtig, möglichst alle wichtigen Verarbeitungen in die Auswertung mit einzu- beziehen. Also nicht nur tägliche Verarbeitungen, sondern auch wöchentliche und monatliche Aktivitäten mussten Berücksichtigung finden.

Diese zeitaufwändige Phase zog sich über mehrere Monate hin. Aber: Die Arbeit wurde von automatischen Sammeltools geleistet, das Team hatte damit relativ wenig zu tun.

Im wesentlichen werden folgende Informationen gesammelt:
- Häufigkeit und die Vergleichsoperatoren, mit der nach einer bestimmten Spalte gesucht wird
- Kombination von Spalten, nach denen gleichzeitig gesucht wird und die Häufigkeit, mit der dies geschieht

Dies sind im wesentlichen die gleichen Informationen, die die Datenbank zur Unterstützung der automatischen Statistikgenerierung in der sys.col_usage$ sammelt. Ab der Version 11.2.0.2 werden auch Spaltenkombinationen unterstützt. Das entsprechende Verfahren beschreibt Maria Colgan - eine Kollegin aus dem Oak Table Netz und bis vor kurzem Produktmanagerin des Optimizers - in ihrem Blog [4].
In früheren Datenbankversionen werden die Spaltenkombinationen nicht unterstützt und müssen daher aus den anderen Quellen hochgerechnet werden.

Als Quellen kommen in Frage:
- die aktuellen Abfragen Shared Pool
  Hier kann man die Suchkriterien ganz einfach aus den Filtern und Access Predicates entnehmen
- die top Statements aus dem AWR (hier muss man leider einen reparse durchführen, da die Filter- die Access Predicates in der DBA_HIST_SQL_PLAN Tabelle leer sind)

Diese Möglichkeiten sind deshalb wichtig, weil man sich auf den Inhalt der col_usage$ **nicht hundertprozentig** verlassen kann. Ich habe Datenbanken gesehen, bei denen der Inhalt der col_usage$ nicht brauchbar war, dies vermutlich aufgrund von Memorymangel. Es ist klar, dass dann auch die Statistikgenerierung in Mitleidenschaft gezogen wird.

Ausserdem werden Informationen aus dem Dictionary verwendet :
- die Selektivität der einzelnen Spalten, sowie die der verwendeten Kombinationen
- Primary Key und Foreign Key constraints

2.3 Auswertungsphase

In dieser Phase werden die gesammelten Informationen zu einem fertigen Index Design verdichtet. Dies geschieht über mehrere Etappen. Bei schwierigen Design Entscheidungen wird immer noch das menschliche Wissen als letzte Instanz hinzugezogen.

Das Vorgehen folgt grob dem Muster des als Merge and Reduction bekannten Algorithmus. [3]
Als Grundlage für die Arbeit dienen uns die in der Vorphase gefundenen Spaltenkombinationen. Im Grunde genommen könnte man aus jeder Spalten-Kombination, die im „Suchen“ auftaucht, einen Index machen.

Das würde aber zu einem Überangebot an Indexen führen. Da jeder Index die DML Operationen verlangsamen kann [z.B. 6], sollen nur so viele Indexe wie nötig und so wenige wie möglich erstellt werden. In den folgenden Schritten wird also versucht, die Index Struktur hinsichtlich Preis/ Leistung zu optimieren.

2.3.1 Grundindexierung

Meist wird man von einer Basisindexierung ausgehen können. Primary Keys bekommen in der Regel ohne grosse Überlegung einen Index.

Foreign Keys sollte man im allgemeinen auch indexieren und zwar aus folgenden Gründen:

1. da es beim Löschen eines Satzen zu unangenehmen Sperren der abhängigen Sätze kommen kann, Sofern ein Foreign Key Constraint existiert. [5]
2. Damit ein nested loop join in jede bestimmte Richtung möglich ist und somit der Optimizer in der Wahl des besten Zugriffsplanes nicht unnötig eingeschränkt wird.

2.3.2 Elimination

In dieser Phase nimmt man alle Suchkombinationen, die man ohne grosse Leistungseinbussen weglassen kann, aus der Betrachtung.

Das sind insbesondere alle:
1. Foreign - und Primary Keys, die bereits im vorigen Schritt indexiert worden sind
2. Suchkombinationen, die so wenig selektiv sind, dass sich kein Index lohnt
3. Suchkombinationen, die bereits in anderen Suchkombinationen gleichwertig enthalten sind

Betrachten wir einige Beispiele aus dem allgemein bekannten Bereich der Adress- und Personendaten:

Zu 2.: ein Index {Geschlecht} lohnt sich auf der Tabelle Person nicht, da die Suchspalte zu wenig selektiv ist. Ausnahme: Eine Anwendung für Frauenfragen beim Militär, da hier häufig nach einem seltenen Wert gesucht wird. Dieses Beispiel zeigt, dass selbst scheinbar einfache Fälle nicht ohne Nachdenken entschieden werden können. Einmal mehr wird klar, vor welchen Schwierigkeiten eine vollautomatische Indexierung steht.

Zu 3.: der Indexkandidat {Ort} auf der Tabelle Adresse wird eliminiert, wenn es eine andere häufige Suchkombination gibt, zum Beispiel {Ort , Strasse} in der {Ort} bereits enthalten ist.

Dies kann man natürlich noch weiter führen: Der dreispaltige Index {Nachname, Vorname, Alter} auf der Tabelle Person kann drei verschiedene Suchkriterien unterstützen:

{Nachname, Vorname, Alter}
{Nachname, Vorname}
{Nachname}

Die entsprechenden Suchkombinationen können also eliminiert werden.
Die folgenden Suchabfragen können nicht unterstützt werden und müssen bestehen bleiben:

{Vorname, Alter}
{Alter}
{Vorname}

Erkenntnis
An diesen Beispielen erkennt man, dass die Wahl der richtigen Reihenfolge der Spalten entscheidend für die Qualität eines flexiblen Indexdesigns ist. Eine optimale Indexierung, unabhängig von der Spaltenreihenfolge, ist mit dem konventionellen B*Index nicht möglich. **Wer die ultimative Flexibilität und Performance haben möchte, muss den mehrdimensionalen Index, beispielsweise den der Firma**
dimensio verwenden. Er kann eine beliebige Kombinationen von Suchkriterien parallel auswerten und verknüpfen. Dies erleichtert die optimale Indexierung wesentlich.

2.3.3 Synthese
In diesem Schritt versucht man übrig gebliebene Suchkombinationen zusammenzulegen und dadurch weitere Index Kandidaten zu eliminieren. So kann man beispielsweise einen Foreign Key um weitere Suchkriterien erweitern. Wenn zum Beispiel in der Tabelle Umsatz häufig nach {Artikelnummer, Verkaufsdatum) gesucht wird, ist es sinnvoll, einen bestehenden Foreign Key Index auf der Artikelnummer um das Feld Verkaufsdatum zu erweitern.

Manchmal kann man auch durch eine leichte Verschlechterung des Index Design einen zusätzlichen Index Kandidaten eliminieren:

Als Beispiel seien die Suchkombinationen gegeben:
{Ort, Nachname, Vorname} und
{Ort, Nachname, Geburtsdatum}

Der Indexkandidat {Ort, Nachname, Vorname, Geburtsdatum} kann die beiden oben gezeigten Kombinationen ersetzen. Zwar ist die Suche nach {Ort, Nachname, Geburtsdatum} nicht mehr so optimal unterstützt wie beim spezialisierten Index, jedoch kann die Verschlechterung wahrscheinlich in Kauf genommen werden, weil Ort, Nachname für sich gesehen, bereits gute Suchkriterien sind.

Man sieht das in diesem Schritt die Vernunft gesteuerte Entscheidung des menschlichen Designers besonders wichtig ist.

2.4 Tests und Umsetzung
Hat man alle diese Schritte durchlaufen, ist es relativ einfach, aus dem Ergebnis ein Index create Script zu bilden. Natürlich muss man das Ergebnis jetzt ausführlich testen. RAT bietet sich dabei als Mittel der Wahl an. Wenn man produktiv geht sollte man auch Überwachungsmechanismen im Einsatz haben, die schnell in der Lage sind, fehlende Indexe zu finden.

Wir hatten ein eigenes Überwachungsskript, welches Ausführungspläne finden kann, welche durch sub- optimale Indexierung entstehen und entsprechende Verbesserungsvorschläge macht. Natürlich kann man auch den Oracle Index Advisor verwenden, der für meinen Geschmack für diesen spezialisierten Zweck etwas umständlicher in der Handhabung ist. Die neue Indexstruktur zeigte sich in der Produktion erstaunlich stabil. In den ersten Monaten musste nur eine einstellige Anzahl Indexe nach erzeugt werden.

2.5 Resultate

In dem hier geschilderten Fall handelt es sich um eine Applikation, die spezialisierte Consultants der Firma Oracle schon seit Monaten optimiert hatten. **Umso erfreulicher war es, dass die Gesamtindexierung zusätzlich eine Platzersparnis von 30 % und eine Performanceverbesserung von ebenfalls 30 % erbrachte.**

Ein sehr erfreulicher Nebeneffekt kam überraschend: Die verbesserte Stabilität der Execution Pläne. Da für einen bestimmten Zweck nur noch ein Index zur Verfügung steht und nicht mehrere, bleiben die Pläne stabiler und verändern sich nicht so leicht in Abhängigkeit von den Werten der bind Variablen.

Last but not least: Die neue Indexstruktur mit ihren klaren Regeln und ihrer einheitlichen Namensgebung erleichtert die tägliche Arbeit der DBAs.

Keine Frage: Faktenbasierte Indexierung ist ein spannendes Feld. Gerne würde ich an meine oben beschriebenen Erfahrungen anknüpfen, jetzt, wo ich wieder im Consulting Bereich tätig bin.

[1] Borovica, Alagiannis, Ailamaki. „Automated Physical Designers: What You See in (Not) What You Get“, DBTest’12, May 21, 2012 Scottsdale, AZ, U.S.A.

[2] Weikum, Moenkeberg, Hasse,,Zabback, „Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering“, Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

[3] Bruno, N. and Chaudhuri, S. 2007. Physical design refinement: The ‘merge-reduce’ approach. ACM Trans. Database Syst. 32, 4 (Nov. 2007), 28.

[4] Colgan M,.How do I know what extended statistics are needed for a given workload? blogentry: https://blogs.oracle.com/optimizer/entry/how_do_i_know_what_extended_statistics_are_needed_for_a_given_workload.

[5] Oracle® Database Concepts 12c Release 1 (12.1) , Kapitel 9, Locks and Foreign Keys und Kapitel 6, Indexes and Foreign Keys)

[6] Oracle® Database 2 Day DBA 12c Release 1 (12.1) , Kapitel 8. managing Indexes

Lothar Flatz

Diso AG
Morgenstrasse 1
CH-3073 Gümligen
Tel. Nr. +41 (0) 31 958 90 90
www.diso.ch