SQLTXPLAIN – Dem Optimizer auf der Spur

Frank Großheim
Dignum GmbH
Hannover

Schlüsselworte

SQLTXPLAIN, SQLT, CBO, Cost Based Optimizer, Execution Plan, Performance Analyse

Einleitung

Die Performance-Analyse von einzelnen SQL-Statements stellt den Oracle DBA immer wieder vor große Herausforderungen. Für die Identifizierung eines problematischen SQL-Statements, die Ermittlung dessen Ausführungsplans und der Ausführungsstatistiken stehen gut bekannte Hilfsmittel und Werkzeuge zur Verfügung, wie z.B. AWR, ASH, Statspack usw.

Wenn die Ausführung des SQL-Statements auf einen nicht-optimalen Ausführungsplan schließen lässt, sind weitere Untersuchungen notwendig. Die Erstellung des Ausführungsplans durch den Cost-Based-Optimizer (CBO) ist eine komplexe Angelegenheit und wird durch eine Vielzahl von Faktoren beeinflusst. Neben den Objekt- und Systemstatistiken, die möglichst die Wirklichkeit abbilden sollten, spielen eine Vielzahl von Parametern und Schemaobjekt-Metadaten eine Rolle.

Um die Entscheidung des Optimizers nachvollziehen und bewerten zu können, ist ein Werkzeug notwendig, welches den ambitionierten DBA oder Entwickler komfortabel und zielgerichtet bei der Analyse unterstützt. Ein oft unbeachtetes und unterschätztes Hilfsmittel ist dabei SQLTXPLAIN, welches 2010 komplett neu entwickelt wurde und im folgenden Artikel eingehend beleuchtet wird.

Beschreibung

SQLTXPLAIN, ebenfalls bekannt als SQLT, ist ein kostenloses Tool, das vom Oracle Server Technologies Center of Expertise (ST CoE) bereitgestellt wird, um langsame SQL-Abfragen effektiv zu analysieren.

Das Tool kann von den Oracle-Supportseiten heruntergeladen werden. SQLT dient in erster Linie dazu, die Oracle Support- und Entwicklungsabteilung nach Aufforderung bei deren Analyse zu unterstützen, ist aber auch hervorragend geeignet um erfahrene DBAs und Entwickler bei eigenen Performance-Analysen behilflich zu sein.

Dem Einsatz von SQLTXPLAIN geht in der Regel eine Gesamtanalyse des Systems voraus, welche Last, Wait-Events, TOP-SQL, Performance-Metriken u.v.m. für einen definierten Zeitraum beleuchtet. Sollten dann ein oder mehrere SQL-Statements als problematisch identifiziert werden und eine Untersuchung im Optimizer-Kontext sinnvoll erscheinen, hilft hier SQLTXPLAIN weiter. Es ist also kein Werkzeug, welches als erstes oder gar in jedem Fall bei einer Performanceanalyse zum Einsatz kommt. Es ist vielmehr, ähnlich wie „Extended SQL Tracing“, ein Spezialwerkzeug, das vom fachkundigen DBA für SQL-Diagnosen in bestimmten Fällen genutzt werden kann. Wer wissen

möchte, warum der Ausführungsplan eines SQL-Statements so ist wie er ist, bekommt die nötigen Informationen geliefert, um diese Frage zu beantworten.

Im April 2010 wurde die dritte Generation des Tools (Release 11.4.0.1) mit einem neuen Datenmodell und neuem Code für Datenbanken ab 10.2 veröffentlicht. Die aktuelle Version 11.4.4.7 ist im Juli 2012 erschienen.

Entwickelt wurde das Werkzeug von Carlos Sierra, Technical Advisor und langjähriges Mitglied von Oracle ST CoE. Neben der Entwicklung von SQLTXPLAIN zeigt er sich auch für den Oracle Trace Analyzer TRCANLRZ verantwortlich.

Nachfolgend wird im Artikel für SQLTXPLAIN die gebräuchliche Kurzform SQLT verwendet.

Funktionsübersicht

Einmal installiert, wird SQLT primär dazu genutzt ein SQL-Statement mit allen, den Optimizer beeinflussenden Faktoren zu analysieren, um diese im Anschluss bewerten zu können. Dabei kann ein SQL-Statement auf verschiedene Weise an SQLT übergeben werden.

Das Tool besteht aus einer Vielzahl von unterschiedlichen Methoden und Modulen, welche durch den Aufruf von verschiedenen SQL-Skripten auf einem Oracle Client gestartet werden.

Als Ergebnis einer erfolgreichen Analyse mit SQLT, wird ein Zip-Archiv erzeugt, welches, abhängig von der ausgewählten Methode, einen Satz von Diagnosedateien enthält. Dies sind in der Regel ein kleiner und ein großer HTML-Bericht, SQL-Trace-Dateien (Event 10053, Event 10046), verschiedene SQL-Skripte, Test-Case-Builder-Dateien und ein Export-Dump des SQLT-Repositories. Bei jedem Aufruf einer SQLT-Methode wird von SQLT eine interne eindeutige Statement-ID vergeben, die sich im Namen des Zip-Files und in den Logfiles wiederfindet.

Die Hauptmethoden (Main Methods) von SQLT verbinden sich mit der Datenbank und lesen Ausführungspläne, Cost-Based-Optimizer-Statistiken, Schemaobjekt-Metadaten, Performance-Statistiken, Konfigurationsparameter und ähnliche Elemente aus, die den Optimizer des zu analysierenden SQL-Statements, beeinflussen. Die Daten werden in dem Schema SQLTXPLAIN gespeichert und daraus die HTML-Berichte generiert, die dem Anwender dann als Basis für seine Auswertung dienen.

Neben den Hauptmethoden (Main Methods) werden noch zusätzlich spezielle Methoden (Special Methods) angeboten. Diese speziellen Methoden dienen z.B. der Differenzanalyse zweier SQL Ausführungen, zur expliziten SQL-Tracedateianalyse Event 10046 und zur Auswertung mehreren SQL-Statements gleichzeitig.

Darüber hinaus gibt es fortgeschrittene Methoden und Module (Advanced Methods and Modules), welche z.B. das Festlegen von Plänen für ein selbsterstelltes SQL-Profil erlauben, das Ändern von Histogrammen ermöglichen u.v.m. Dieser Artikel verzichtet auf die Beschreibung, da die fortgeschrittenen Methoden und Module nur nach Aufforderung vom Oracle Support verwendet werden sollen.

Installation

Nach dem Herunterladen von „My Oracle Support“ wird die 2 MB große Zip-Datei zunächst auf dem Client-PC entpackt. Entpackt besteht SQLT aus 5 Unterverzeichnissen, die wiederum eine Vielzahl von Skripten, Installationsdateien und Dokumenten beinhalten.

Für die Installation der SQLT-Objekte in die zu analysierende Datenbank wird das Skript `sqcreate.sql` mit einem Benutzer mit SYSDBA-Berechtigung gestartet (z.B. SYS).

Das angelegte SQLT-Repository beinhaltet in der aktuellen Version 214 Tabellen, 146 Indizes, 105 Views, 17 Packages, 12 Sequenzen, 1 Prozedur und eine neue Rolle. Darüber hinaus werden 5 Oracle Directories angelegt, die dem Zugriff auf Tracedateien im Filesystem dienen. Es werden Initial ca. 10 MB in der Datenbank belegt.

Die für die Installation notwendigen Informationen werden während der Installation im SQL*Plus-Fenster interaktiv abgefragt. Dazu zählt der optionale Connect Identifier der Zieldatenbank, Ziel-Tablespace, Temp-Tablespace und das Passwort des neuen Users SQLTXPLAIN.

Außerdem muss der Applikationsbenutzer registriert werden, welcher das zu analysierende SQL-Statement ausführt (z.B. SCOTT). Weitere Applikationsbenutzer können nach der Installation noch später durch die Ausführung des Skripts `squser.sql` oder durch das Erteilen der neuen Rolle SQLT_USER_ROLE hinzugefügt werden. Als Ziel-Tablespace für die SQLTXPLAIN-Objekte kann ein eigener Tablespace (default SQLTXPLAIN) ausgewählt werden, welcher vor der Installation angelegt werden muss.

Die Nutzung von Funktionen des Diagnostic- und Tuningpack bei der Berichterstellung kann deaktiviert werden. Wird z.B. das Vorhandensein beider Lizenzen verneint, so macht SQLT keinen Gebrauch von SQL Tuning Advisor (STA), SQL Monitoring, Automatic Workload Repository (AWR) und SQL Tuning Sets (STS).

Deinstallation

Die Deinstallation von SQLT erfolgt mit dem Skript `sqdrop.sql`, welches das Schema SQLTXPLAIN entfernt. Der für das Schema genutzte Tablespace und die Rolle SQLT_USER_ROLE verbleiben jedoch in der Datenbank und müssen manuell gelöscht werden.

Sicherheitsaspekte

Das Tool verwendet ein eigenes, abgegrenztes Schema (SQLTXPLAIN) in der Datenbank, welches, ähnlich wie beim Oracle Statspack, als Repository für die Analyse dient. Es werden keine Objekte in Applikationsschemata installiert. Applikationsdaten werden weder geändert noch gelesen und Low- und High-Werte können bei Histogrammen verborgen werden.

Der neu angelegte Benutzer SQTXPLAIN ist nach der Installation mit den Rollen SELECT_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, GATHER_SYSTEM_STATISTICS und einigen expliziten System- und Objektberechtigungen auf Objekte im SYS und OUTLN-Schema ausgestattet. Objektberechtigungen auf Anwendungsdaten werden nicht erteilt.

Der Applikationsbenutzer, der das zu analysierende SQL ausführt, wird bei SQLT registriert und ihm die Rolle SQLT_USER_ROLE zugewiesen. Diese Rolle ist nicht ganz unkritisch, da diese die Rolle SELECT_CATALOG_ROLE enthält und diese wiederum dem Applikationsbenutzer lesenden Zugriff auf Objekte im Data Dictionary gewährt. Dies ist aus Sicherheitsgründen oft unerwünscht und in einigen Organisationen sogar verboten. Vor der Installation sollte dieser Punkt abgeklärt werden und nach abgeschlossener Analyse die Rolle SQLT_USER_ROLE dem Applikationsbenutzer entzogen und/oder SQLTXPLAIN deinstalliert werden. Alternativ kann man auch auf die Erteilung der Rolle verzichten, da die Auswirkung auf die generierten Berichte eher gering ist.

HTML-Berichte

Der Hauptbericht (Main Report) enthält sämtliche zur Diagnose erforderlichen Informationen, auf die bequem über eine am Anfang des Berichtes erstelle Liste zugegriffen werden kann. Da der Umfang des Berichtes von der gerade verwendeten Methode und der Freischaltung von Diagnostic- und Tuningpack abhängig ist, sind einige Links in der Liste u.U. nicht aktiv und die Information dazu nicht verfügbar. Bewegt man den Mauszeiger über einen Link, erscheint ein Hinweis, welcher View oder welchem Package die Information zu dem jeweiligen Berichtspunkt entnommen wurde.

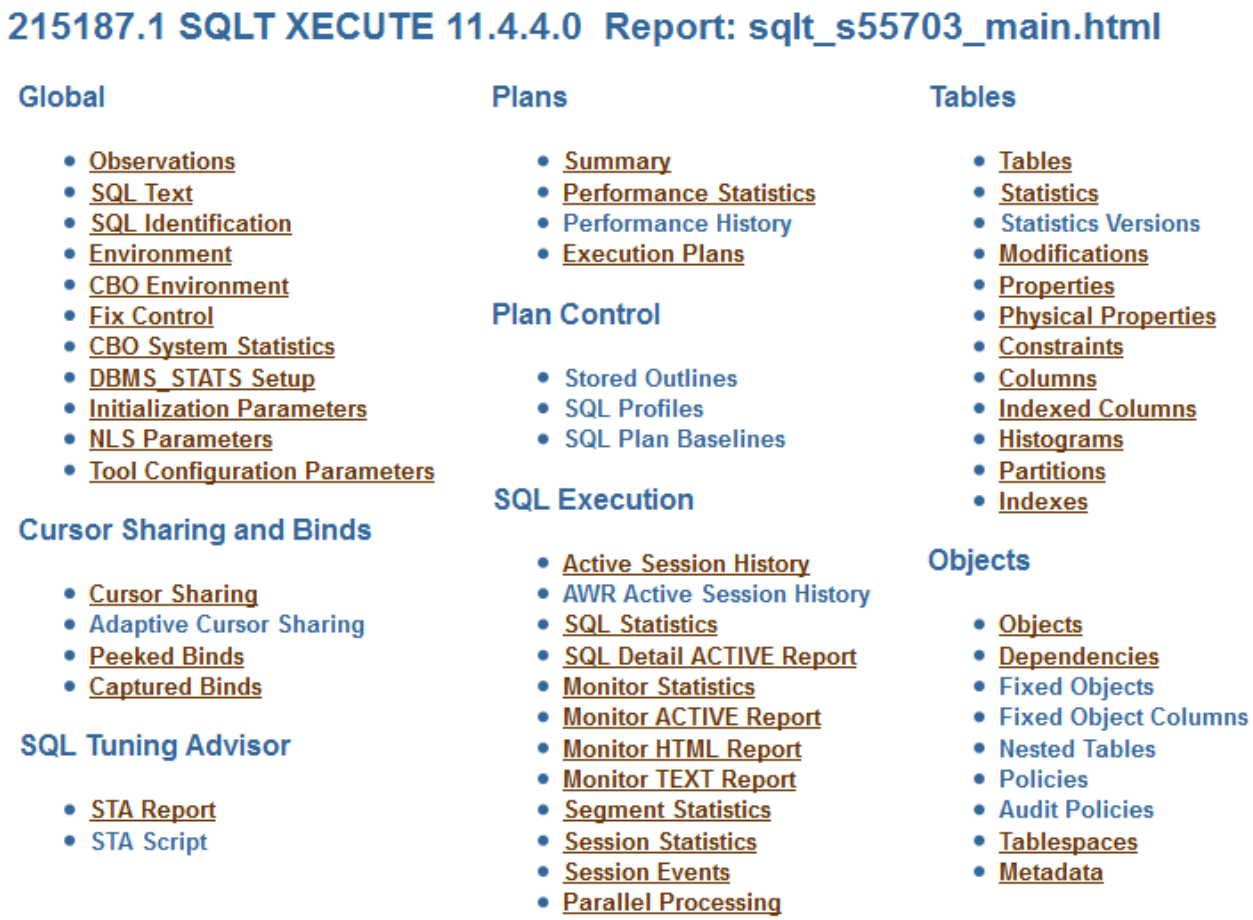

Abb. 1: Beispiel Startseite Main Report der Methode XECUTE

Ein echter Mehrwert ist der erste Punkt im Hauptbericht „Observations“. Dieser wird durch das integrierte Health-Check-Modul erzeugt und listet vom Programm gemachte Beobachtungen auf, die überprüft werden sollten und ggf. weiterer Aufmerksamkeit bedürfen. Es wird nicht nur beschrieben was festgestellt wurde, sondern auch eine Erklärung geliefert warum etwas problematisch sein könnte und was zu tun ist. Wie relevant eine festgestellte Beobachtung tatsächlich ist, muss allerdings vom Nutzer bewertet werden.

Neben dem Hauptreport wird noch ein abgespeckter Light-Report generiert. Dieser enthält lediglich Informationen zum Ausführungsplan, den Tabellen, Tabellenspalten, Indizes und Indexspalten.

Hauptmethoden

SQLT unterstützt 5 Hauptmethoden (Main Methods), welche Diagnosedetails für ein SQL-Statement generieren.

XTRACT - Extrahiert SQL aus dem Memory und/oder dem Automatic Workload Repository (AWR).

XECUTE - Führt SQL tatsächlich aus und findet es dann im Memory.
(Mehr Details, Textfile als Input notwendig)

XTRXEC- Führt beide Methoden XTRACT und XTRACT hintereinander aus.
(Extrahiert den teuersten Plan aus dem Memory und führt das SQL anschließend aus)

XPLAIN - Basiert auf „EXPLAIN PLAN FOR“.
(Nutzung nur empfohlen, wenn XTRACT und XECUTE nicht möglich ist)

XTRSBY- Extrahiert SQL, welches auf einer Read-Only Standby Datenbank ausgeführt wurde.

Alle Methoden müssen als der Applikationsbenutzer ausgeführt werden, unter dem das zu analysierende SQL tatsächlich läuft, damit die Analysen im tatsächlichen Applikationskontext durchgeführt werden.

Während XTRACT, XECUTE, XTRXEC und XTRSBY mit Bind-Variablen umgehen können und BIND PEEKING verstehen, tut dies XPLAIN nicht. Der Grund ist, dass XPLAIN auf EXPLAIN PLAN FOR basiert, welches blind für BIND PEEKING ist. Aus diesem Grund sollten man XPLAIN vermeiden. Auf die drei gängigsten Hauptmethoden XTRACT, XECUTE und XTRXEC wird nachfolgend im Detail eingegangen.

XTRACT-Methode

Die XTRACT-Methode extrahiert und analysiert bereits ausgeführte SQL-Statements. Voraussetzung für die Verwendung der XTRACT-Methode ist, dass das SQL-Statement sich entweder noch im Memory (SQL Area im Library Cache) befindet oder vom AWR eingefangen wurde. Zur eindeutigen Identifikation des SQL-Statements ist die SQL_ID oder der HASH_VALUE notwendig. SQL_SID und/oder HASH_VALUE können z.B. einem AWR-Report, einer Trace-Datei, V$SQL oder V$SQLAREA entnommen werden.

Die XTRACT-Methode ist außerdem die einzige Möglichkeit zur Analyse, wenn das Statement kein weiteres Mal tatsächlich ausgeführt werden kann, da es z.B. Daten ändert (DML) oder einen unerwünschten signifikanten negativen Effekt auf die Performance eines Systems hat.

Wichtige Performance-Statistiken, wie z.B. die tatsächliche Anzahl von Zeilen pro Ausführungsplan-Operation, sind allerdings nur verfügbar, wenn zu dem Zeitpunkt des Hard-Parsens des SQL-Statements der Parameter STATISTIC_LEVEL auf ALL gesetzt war. Das gleiche Ergebnis erreicht man allerdings auch, wenn der Optimizer-Hint `/*+ GATHER_PLAN_STATISTICS */` zum SQL-Text hinzugefügt wurde. Unter 11g kann man außerdem den Optimizer Hint `/*+ GATHER_PLAN_STATISTICS MONITOR */` einsetzen, um erweitere Diagnosen zu erhalten. Der Hint MONITOR erzwingt ein „Real Time SQL Monitoring" und darf nur bei Lizensierung des Diagnostic- und Tuningpack gesetzt werden.

Um die XTRACT-Methode einzusetzen, muss man sich mit SQL*Plus an die Datenbank als der Applikationsbenutzer, der das SQL ausgeführt hat, anmelden. Anschließend wird das Skript `sqltxtract.sql` ausgeführt, wobei SQL_ID oder HASH_VALUE beim Aufruf übergeben werden.

```
sqlplus [application_user]
sqltxtract.sql [SQL_ID]|[HASH_VALUE]
```

Beispiel:

```
cd F:\sqlt\run
sqlplus soe/soe@oradb1
SQL> @sqltxtract.sql gkxxkghxubh1a
```

XECUTE-Methode

Diese Methode führt ein SQL-Statement aus, welches dem Programm in einem Textfile in einem vordefinierten Format übergeben wird, bevor es dann zur Analyse aus dem Memory extrahiert wird. Dies ist natürlich bei ändernden Statements (DML), lastintensiven oder langlaufenden Abfragen auf Produktionssystemen nur bedingt möglich. Diese Methode stellt jedoch mehr Details als XTRACT zur Verfügung.

Vor dem Einsatz von XECUTE muss ein Textfile erstellt werden, welches den SQL-Text enthält. Wenn das SQL-Statement Bind-Variablen enthält, müssen diese im Textfile deklariert und diesen einen Wert zugewiesen werden.

Darüber hinaus sollte es den Hint `/* ^^unique_id */` enthalten, um ein Hard-Parsen bei jeder Ausführung zu erzwingen. Damit auch hier wichtige Performance-Statistiken anschließend zur Verfügung stehen, sollte das Textfile am Anfang um die Zeile `ALTER SESSION SET STATISTICS_LEVEL = 'ALL'` ergänzt werden oder der Hint `/*+ gather_plan_statistics */` hinzugefügt werden.

Beispiel statement1.sql:

```
VAR B1 NUMBER;
EXEC :b1 := 285;

SELECT /*+ gather_plan_statistics monitor */ ORDER_MODE, ORDERS.WAREHOUSE_ID, /*
^^unique_id */ SUM(ORDER_TOTAL), COUNT(1)
FROM ORDERS, WAREHOUSES
WHERE ORDERS.WAREHOUSE_ID = WAREHOUSES.WAREHOUSE_ID
AND WAREHOUSES.WAREHOUSE_ID = :B1
GROUP BY CUBE(ORDERS.ORDER_MODE, ORDERS.WAREHOUSE_ID);
```

Weitere Beispiele befinden sich im Unterverzeichnis \sqlt\input\sample.

Wie bei der XTRACT-Methode, muss man sich zum Starten von XECUTE mit SQL*Plus an die Datenbank als der Applikationsbenutzer, der das SQL normalerweise ausführt, anmelden. Anschließend wird das Skript `sqltxecute.sql` ausgeführt, wobei der Pfad des Textfiles, welches das vorbereitete SQL-Statement enthält, beim Aufruf mitübergeben wird.

```
sqlplus [application_user]
[path]sqltxecute.sql [path]scriptname
```

Beispiel:

```
cd F:\sqlt\run
sqlplus soe/soe@oradb1
SQL> @sqltxecute.sql F:\sqlt\input\statement1.sql
```

XTRXEC-Methode

Diese Methode kombiniert beide Funktionen von XTRACT und XECUTE. Genau genommen werden beiden Methoden seriell hintereinander ausgeführt. Die XTRACT-Phase generiert ein Skript, welches das extrahierte SQL-Statement zusammen mit den Bind-Variablen-Deklarationen und -Zuweisungen des teuersten Plan enthält. Anschließend wird in der XECUTE-Phase das erzeugte Skript ausgeführt.

Die Auswahl der Werte der Bind-Variablen basiert auf den „Peeked"-Werten zu dem Zeitpunkt, als der teuerste Plan generiert wurde. Der teuerste Plan wird dabei auf der Grundlage der durchschnittlichen Ausführungszeit ausgewählt. Auch diese Methode wird unter dem Applikationsbenutzer gestartet, unter dem das problematische SQL ausgeführt wurde. Als Übergabeparameter zum Skript `sqltxtrxec.sql` wird SQL_ID oder HASH_VALUE erwartet.

```
sqlplus [application_user]
sqltxtrxec.sql [SQL_ID]|[HASH_VALUE]
```

Beispiel:

```
cd F:\sqlt\run
sqlplus soe/soe@oradb1
SQL> @sqltxtrxec.sql drvf6vtmwyvtb
```

Logfiles und Monitoring

Während der Ausführung der Methoden, kann über die View SQLTXPLAIN.SQLT$_LOG_V nachverfolgt werden, welcher Einzelschritt gerade ausgeführt wird. Es werden ebenfalls Logfiles von den Methoden ins Filesystem (Verzeichnis /sqlt/run) geschrieben, welche im Fehlerfall bei der Ursachenermittlung behilflich sind.

Spezialmethoden

SQLT bietet 6 Spezialmethoden. Die interessanteren Spezialmethoden COMPARE, TRCANLZR und XTRSET werden nachfolgend kurz beschrieben.

COMPARE-Methode

Die COMPARE-Methode wird eingesetzt, wenn auf zwei ähnlichen Systemen ein und dasselbe SQL-Statement unterschiedlich performt. Diese Methode hilft Unterschiede bezüglich der Ausführungspläne, Metadaten, CBO-Statistiken und Initialisierungsparameter herauszustellen.

SQLT muss dazu auf beiden Systemen installiert und die Analyse des betroffenen SQL-Statements auf beiden Systemen mit einer der Haupt-Methoden durchgeführt worden sein.

Die COMPARE-Methode selber kann auf jedem der beiden Systeme oder auf einem dritten System durchgeführt werden. Damit die COMPARE-Methode in der Lage ist beide Ausführungen zu vergleichen, muss es auf die Informationen in beiden SQLT-Repositories zugreifen können. Da bei jeder Anwendung der Haupt-Methoden, auch das SQLT-Repository automatisch am Ende exportiert wird, können die fehlenden Informationen einfach vor Verwendung der COMPARE-Methode in das COMPARE-System importiert werden, auf dem die COMPARE-Methode ausgeführt werden soll. Bei Verwendung eines dritten Systems, müssten beide SQLT-Repositories in dieses importiert werden. Der Import des SQLT-Repositories wird in der Datei `sqlt_<statement_id>_readme.html` beschrieben.

Sobald das COMPARE-System beide Repositories der Quellsysteme enthält, kann der Vergleich durch den Aufruf von `sqltcompare.sql` gestartet werden. Der User, mit dem das Skript gestartet wird, kann diesmal neben dem Applikationsbenutzer auch SYS oder SQLTXPLAIN sein.

TRCANLZR-Methode

Diese Methode verarbeitet, ähnlich wie TKPROF, eine SQL-Trace-Datei und erstellt anschließend einen ausführlichen Bericht, welcher Top-SQL und die SQL-Genealogie der SQL-Statements enthält, die in der Trace-Datei enthalten sind.

Die Trace-Datei muss sich in den Oracle Directories TRCA$INPUT1 oder TRCA$INPUT2 befinden, um diese verarbeiten zu können (Pfade siehe View dba_direrctories). Beide Pfade zeigen standardmäßig auf die Trace-Verzeichnisse der Datenbank USER_DUMP_DEST und BACKGROUND_DUMP_DEST, können aber bei der Installation angepasst werden.

Als SQL-Trace-Datei kann jede Trace-Datei verwendet werden, die mit dem Event 10046 (Level 1, 4, 8 oder 12) erzeugt wurde. Da die Haupt-Methoden XECUTE und XTRXEC automatisch SQL-Trace-Dateien des Events 10046 erzeugen, können diese ebenfalls ein Input verwendet werden. TRCANLZR ist auch in der Lage mehrere in Beziehung stehende Trace-Dateien zu verarbeiten, wie sie z.B. bei einer Parallelverarbeitung vorkommen (PX-Traces). Für diesen Zweck ist eine Datei `control.txt` im Oracle Directory TRCA$INPUT1 oder TRCA$INPUT1 zu erstellen, die in jeder Zeile den Namen der zu verarbeitenden Trace-Datei enthält.

TRCANLZR ist vergleichbar mit TKPROF, besitzt aber erweiterte Funktionalitäten. Wenn es eine Trace-Datei oder einen Satz von Trace-Dateien analysiert, schließt es auch Schemaobjekt-Charakteristiken, wie CBO-Statistiken und einige andere wertvolle Performance-Kennzahlen mit ein. Wenn TRCANLZR fertig ist, fragt es, ob eine XTRACT-Methode für die Top-SQLs, die in der Trace-Datei gefunden wurden, erwünscht ist. Wenn erwünscht, ruft es die TRCSET-Methode auf und konsolidiert alle erzeugten Berichte.

XTRSET-Methode

XTRSET extrahiert aus dem Memory oder AWR eine Liste von SQL-Statements und führt dann XTRACT für jedes der SQL-Statements aus. Am Ende werden alle SQLT-Dateien zu einer gepackten Datei konsolidiert. XTRSET wird genutzt, um für einen Leistungsvergleich den gleichen Satz von SQL-Statements wiederholen zu können.

Gestartet wird bei XTRSET das Skript `sqltxtrset.sql` unter dem Applikationsbenutzer, der alle oder zumindest die meisten der SQL-Statements ausführt. Bei Aufforderung muss dann eine kommaseparierte Liste von SQL_IDs oder HASH_VALUEs angegeben werden, um die zu analysierenden Statements zu identifizieren. Des Weiteren muss noch das Passwort von SQLTXPLAIN angegeben werden.

Analysen von verteilen Abfragen

Analysen von verteilten Abfragen, die über Datenbanklinks auch auf entfernte Datenbanken zugreifen, sind ebenfalls möglich. Dazu muss SQLT auch auf dem entfernten Knoten installiert werden und der entfernte Knoten auf dem lokalen Knoten durch Registrierung der Datenbanklinks bekannt gemacht werden:

```
SQL> EXEC sqltxplain.sqlt$i.register_db_link('db_link_name');
```

Bonus: SQL Tuning Health-Check Skript

Mit SQLT wird im Verzeichnis /utl das allein lauffähige Health-Check-Skript `sqlhc.sql` mitgeliefert, welches sich auch ohne Installation von SQLT ausführen lässt. Wenn es für eine SQL_ID ausgeführt wird, generiert es, ähnlich wie der Abschnitt Observations in den SQLT-Berichten, einen HTML-Bericht mit den Ergebnissen eines Health-Checks rund um das SQL-Statement.

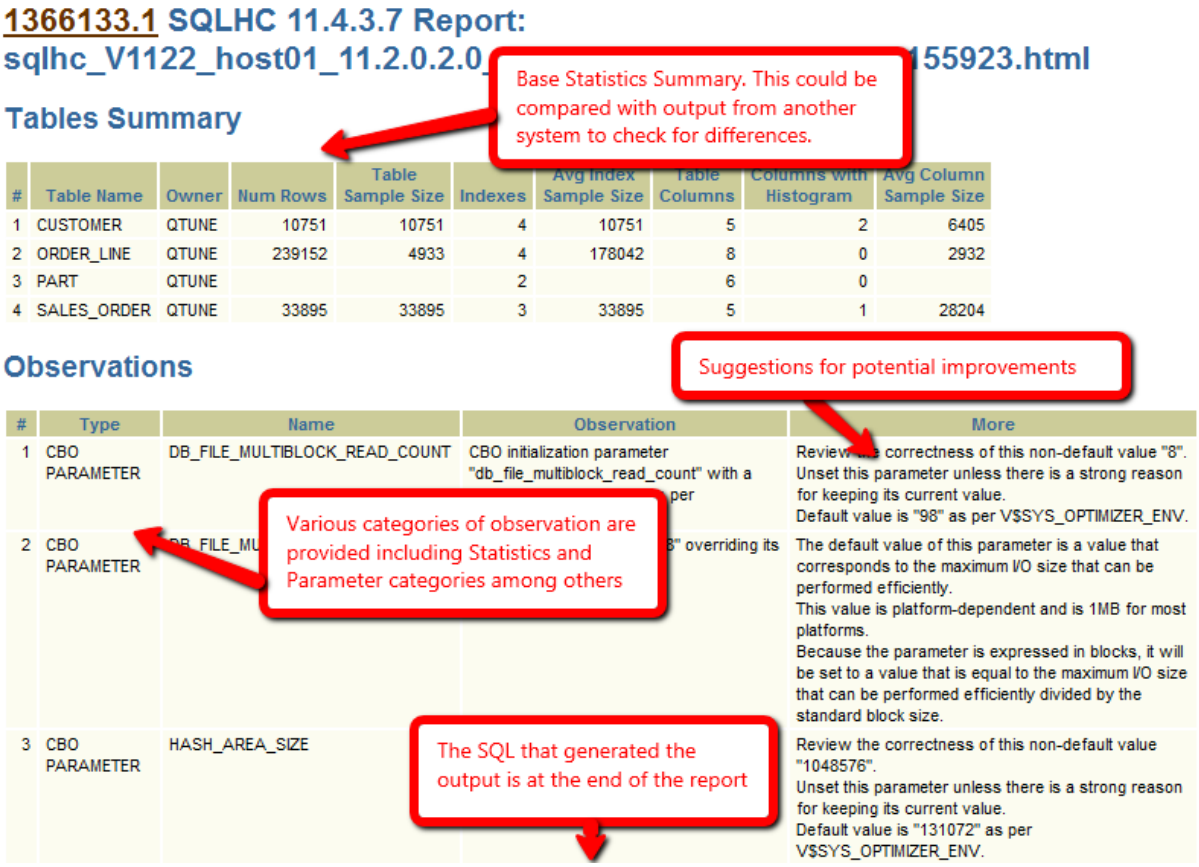

Abb. 2: Beispiel SQL-Heath-Check Report

Folgende Aspekte werden geprüft:

- CBO Statistiken für die Schemaobjekte, auf die von dem SQL-Statement zugegriffen wird
- CBO Parameter
- CBO System Statistiken
- CBO Data Dictionary Statistiken
- CBP Fixed Objects Statistiken

Das Skript muss mit einem SYS-, einem DBA- oder einem Benutzer mit Zugriffsberechtigung auf das Data Dictionary ausgeführt werden. Das Skript legt keinerlei Datenbankobjekte an und muss nicht weiter konfiguriert werden. Es berichtet über mögliche Probleme und empfiehlt dazugehörige Maßnahmen.

```
sqlplus [sys user] [dba user] [select dictionary user]
@sqlhc.sql [SQL_ID]
```

Beispiel:

```
cd F:\sqlt\utl
sqlplus system/mypasswd
SQL> @sqlhc.sql 29qp10usqkqh0
```

Achtung: Das Healthcheck-Skript wird von Oracle offiziell nicht supported.

Die aktuelle Version 11.4.4.7 ist herunterladbar von den Oracle Supportseiten (MOS Not 1366133.1).

Fazit

Wenn die Installation von SQLT in Produktionsumgebungen kein Hindernis ist, dann ist SQLT eine wertvolle Hilfe bei der Analyse von SQL-Statements, bei denen der Verdacht besteht, dass deren Ausführungspläne nicht optimal sind. Insbesondere gefällt die umfassende Betrachtung aller Faktoren, die den Oracle Cost-Based-Optimizer bei seiner Entscheidung beeinflussen.

Auch die Hilfestellung des Heath-Checks-Moduls im Berichtsabschnitt Observation ist gut gemacht und gibt ggf. den entscheidenden Tipp für die Ursache. Alle, die keine Diagnostic- und Tuning-Pack Lizenz erworben haben oder die Standard Edition einsetzen, werden die Möglichkeit der Deaktivierung der beiden Managements Packs zu schätzen wissen.

Die Bedienung und Installation ist sehr einfach und die Verwendung des Tools ist gut dokumentiert. Dennoch bleibt SQLT ein Werkzeug für den erfahrenen DBA oder Entwickler mit sehr guten Kenntnissen. Trotz Hilfestellung, obliegt die Auswertung der Berichte und das Ziehen der richtigen Schlussfolgerungen immer noch dem Anwender. Fast überflüssig zu erwähnen, dass natürlich die Auswertung der Diagnosedateien auch dem Oracle Support, wie von diesem ursprünglich vorgesehen, überlassen werden kann.

Ein Blick in das Änderungslog von SQLT lässt hoffen, dass die Entwickler auch weiterhin Fehler schnell beseitigen und weitere nützliche Funktionen stetig hinzufügen. Wer über entsprechendes Know-how verfügt und den Einsatz von SQLT nicht scheut, wird viel Freude mit dem Tool haben und es schätzen lernen.

Referenzen / Quellen

- My Oracle Support Note: 125187.1
  "SQLT (SQLTXPLAIN) – Tool That Helps To Diagnose SQL Statements Performing Poorly"

- My Oracle Support Note: 1454160.1
  "FAQ:SQLT (SQLTXPLAIN) Frequently Asked Questions"

- My Oracle Support Note: 1366133.1
  "SQLT Tuning Health-Check Script (SQLHC)"

- My Oracle Support Note: 1417774.1
  "FAQ: SQL Health Check (SQLHC) Frequently Asked Questions"

- SQLTXPLAIN Programm-Dokumentation (Verzeichnis sqlt/doc)

Kontaktadresse:

Frank Großheim
Dignum GmbH
Expo Plaza1
D-30539 Hannover

Telefon: +49 (0) 511-165 955-89
Fax: +49 (0) 511-165 055-11
E-Mail frank.grossheim@dignum.de
Internet: www.dignum.de