SQLTXPLAIN und SQLHC:

Performance-Diagnose mit Bordmitteln

Uwe M. Küchler
Opitz Consulting GmbH
Bad Homburg v.d.H.

Schlüsselworte
Datenbank, SQL, Performance, Diagnose, Troubleshooting, Tools

Einleitung
Schon seit den frühen 2000'ern gibt es von Oracle Support zwei sehr nützliche, aber leider ziemlich wenig bekannte Werkzeuge zum Troubleshooting inperformanter oder fehlerhafter SQL-Statements. Die geringe Dokumentation und die Fülle an generierter Information können zunächst abschreckend wirken. Anhand von Beispielen aus der Praxis soll daher ein Einstieg in die effiziente Nutzung dieser Werkzeuge vermittelt werden.

Noch ein Diagnose-Tool?
Performance-Probleme in Datenbank-Anwendungen können sehr unterschiedliche Ursachen haben. Um bei einem konkreten Problem die Ursache(n) zu ermitteln, wird meist eine Top-Down-Vorgehensweise gewählt:

1. Auf welche Systemressourcen wurde am längsten gewartet?
2. Was war für diesen Ressourcenverbrauch am meisten verantwortlich?
3. Warum kam es jeweils zu diesen Wartezeiten und lassen sich diese verkürzen?

Zur Beantwortung dieser Fragen stehen auf dem Markt dafür sehr vielfältige, oft kostenintensive, Werkzeuge zur Verfügung – vom ganzheitlichen Ansatz des Application Performance Monitoring (APM) über Datenbank-zentrierte Tools von Drittherstellern und lizenzpflichtigen Oracle-Features bis hin zu frei verfügbaren Werkzeugen von Oracle oder aus der Oracle-Community.
Lohnt sich da der Aufwand, sich in ein weiteres, neues Werkzeug einzuarbeiten? Decken die verfügbaren Lösungen nicht schon alle denkbaren Herangehensweisen beim Entstören von Performance-Problemen ab?

Um dies zu beantworten, sollen zunächst einige, bekannte Werkzeuge von Oracle dargestellt werden:

- Active Session History (ASH) und Automatic Workload Repository (AWR):
  Seit Oracle 10g gibt es diese beiden, lizenzpflichtigen Features. Sie sammeln in regelmäßigen Abständen Performance-Metriken, historisieren sie und bieten über Views den Zugriff per SQL auf diese Historie sowie über Text- und HTML-Reports einen guten Überblick über alles, was in einem bestimmten Zeitfenster „in der Datenbank passiert ist“. Damit lässt sich eine gute Top-Down-Sicht erreichen.
  Nachteile: Die Lizenzkosten für die Nutzung dieser Features sind nicht unerheblich. Außerdem lässt sich damit zwar problematisches SQL aufspüren, aber nur selten weiter diagnostizieren.

- Statspack:
  Dieser Vorgänger des AWR ist nach wie vor kostenfrei verfügbar und soll in dieser

Aufzählung nicht vernachlässigt werden.
Nachteile: Auch hier gilt: Ressourcenintensives SQL kann hiermit gefunden, aber nicht eigehender untersucht werden.

- Real Time SQL Monitoring:
  Dieses Feature ist seit 11g verfügbar und ebenfalls Teil des Diagnostic- und Tuning-Packs. Der Ablauf eines SQL-Statements kann damit „live“ verfolgt und noch einige Zeit nach dessen Beendigung nachvollzogen werden. Zum Ausführungsplan kann neben den kalkulierten Kosten, Kardinalitäten und Laufzeiten auch der tatsächliche Wert bei der Ausführung dargestellt werden. Somit ist schnell erkennbar, in welchem Teil der SQL-Ausführung ungewöhnlich viele Ressourcen beansprucht und Wartezeiten verursacht wurden.
  Nachteile: Das „Wo“ und das „Wie“ des Ressourcenverbrauchs wird damit zwar aufgedeckt, nicht aber das „Warum“.

- Autotrace:
  Dieses über SQL*Plus, SQL Developer, TOAD und weitere Tools bedienbare Bordmittel zeigt direkt nach der Ausführung eines SQLs etliche Metriken zum Ressourcenbedarf sowie den tatsächlich verwendeten Ausführungsplan.
  Nachteile: Das SQL muss frisch ausgeführt werden; ein Einblick in vergangene SQL-Ausführungen ist damit nicht möglich. Außerdem ist die Ausführung in einem Tool nicht dasselbe wie die Ausführung in der tatsächlichem Anwendungsumgebung.

- SQL-Trace:
  Bei der „Ultima Ratio“ der Diagnose ermöglicht es, quasi jeden einzelnen Schritt der SQL-Ausführung nachzuvollziehen. Zur Auswertung der oft riesigen Tracefiles werden Tools wie tkprof, TRCA oder auch SQL Developer herangezogen.
  Nachteile: Das Tracing muss zur Laufzeit durchgeführt werden; ein Einblick in vergangene SQL-Ausführungen ist damit ebenfalls nicht möglich. Ein Einsatz des Tracings in Produktionsumgebungen verbietet sich zumeist wegen des Aufwands und der generierten Datenmenge.

- Data Dictionary:
  Nicht zuletzt bietet das Data Dictionary einem Performance-Analysten vielfältige Möglichkeiten, Informationen zu bereits ausgeführtem SQL zu sammeln: Über die SQL Area, die Session-Konfiguration, das Optimizer-Environment zum Zeitpunkt der Ausführung, Optimizer-Statistiken aller vom Statement verwendeten Objekte u.v.m.
  Nachteile: Das kann schnell in zeitaufwendige Kleinarbeit ausarten. Expertenwissen ist nötig, um zu wissen, wonach man suchen muss und wo vielleicht spezifische Bugs zuschlagen könnten. Und selbst dann besteht noch die Möglichkeit, dass man eine wertvolle Information übersieht.

Oft handelt es sich bei den Problemfällen um Statements, deren Ausführungszeit schwankt bzw. plötzlich umschlägt. Die Diagnose ist hier oft Detektivarbeit, bei der viele Informationen aus vielen Quellen zusammengetragen werden müssen, z.B.:

- Plötzliche Änderungen am Ausführungsplan
- Änderungen an den Optimizer-Statistiken der vom SQL benutzten Objekte
- Grundsätzlich schlechte oder fehlende Optimizer-Statistiken
- Nicht mehr benutzte (z.B. weil invalide oder gelöschte) Indizes

- Aufruf des SQLs mit außergewöhnlichen Bind-Werten

Genau hier fügt sich SQLTXPLAIN (kurz: SQLT) in das vorhandene Repertoire ein:

Funktionsweise von SQLT

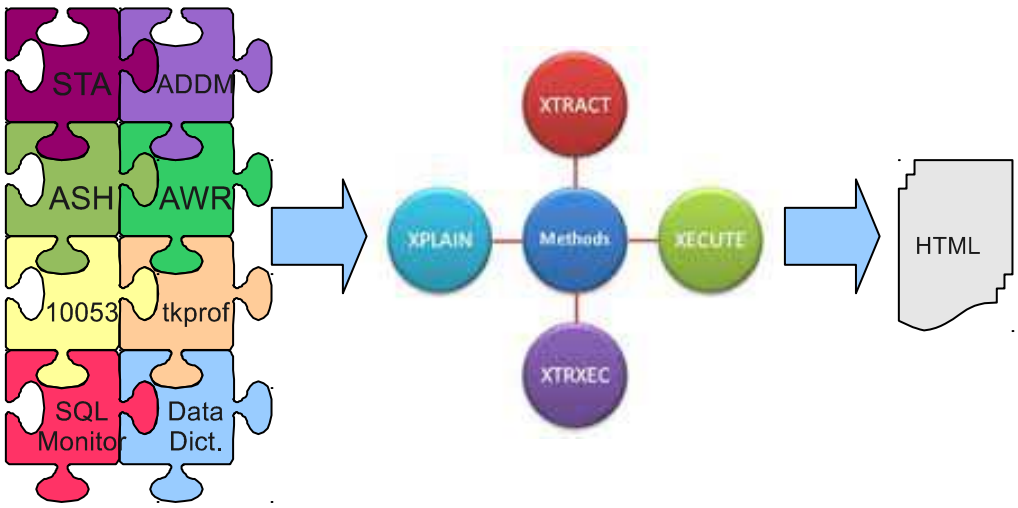


Abb. 1: Von den Quellen zu den Berichten: Modularer Aufbau von SQLT

SQLT

- Besteht aus einer Sammlung von SQL-Scripten, PL/SQL-Packages und Repository-Tabellen.
- Sammelt Informationen zum Tuning eines SQL-Statements ein, wahlweise aus historischen Informationen oder durch Ausführung des Statements.
- Führt über 100 Health Checks rund um das untersuchte Statement aus.
- Kann aus einem Anwendungs-Schema heraus ausgeführt werden.
- Wenn Tuning- oder Diagnostics-Pack lizensiert sind:
    - Wird SQL Tuning Advisor (STA) für das Statement ausgeführt.
    - Werden ASH- und SQL-Monitor-Berichte erzeugt.
    - Können SQL Tuning Test Cases extrahiert werden.
- Kann SQL Tuning Test Cases (TC) extrahieren, die dann in einem Testsytem zu eingehenderen Analysen verwendet werden können.
- Beschleunigt den SQL-Tuning-Prozess durch Automatisierung von ansonsten manuell ausgeführten Recherchen.

Dazu stellt SQLT vier wesentliche Methoden für die Analyse eines SQL-Statements bereit, denen eine SQL-ID, ein Hash oder ein SQL-Text übergeben wird und deren Ergebnis eine ZIP-Datei ist, die einen Bericht im HTML-Format sowie weitere Details in Form von AWR- ASH- und ADDM-Berichten,

10046- und 10053-Traces enthalten.

Installation
SQLT steht zum Download bei "My Oracle Support" unter der Document ID 215187.1 bereit. Die Installation besteht aus wenigen Schritten:

1. Entpacken der ZIP-Datei
2. Start des Installationsskriptes sqlt/install/**sqcreate.sql** als SYS
3. Angabe eines Passwortes, Default Tablespace, Temp Tablespace, eines Benutzerkontos, das SQLT ausführen darf und der lizensierten Optionen (**T**uning, **D**iagnostics oder **N**one).

Danach wird die Einrichtung der beiden Schemata SQLTXPLAIN (Repository) und SQLTXADMIN (Packages) sowie der Rolle SQLT_USER_ROLE durchgeführt. Diese Rolle muss anderen DB-Accounts vergeben werden, wenn diese SQLT ausführen sollen.

Ausführen von SQLT
Die wichtigsten Methoden, die SQLT zur Verfügung stellt, seien hier kurz aufgeführt:

Methode	Script	Features
SQLT XTRACT	sqlt/run/**sqltxtract.sql**	❑ Am häufigsten verwendete Methode ❑ Eingabewert: SQL_ID ❑ SQL-Statement wird *nicht* ausgeführt

SQLT XECUTE	sqlt/run/**sqltxecute.sql**	❑ Eingabe: Scriptname; Inhalte:
		  ❑ ALTER SESSION-Anweisungen
		  ❑ Binds mit Wertzuweisungen
		  ❑ SQL-Statement
		❑ SQL-Statement wird ausgeführt

SQLT XTRXEC	sqlt/run/**sqltxtrxecsql**	❑ Kombiniert XTRACT und XECUTE ❑ Eingabewert: SQL_ID
SQLT XPLAIN	sqlt/run/**sqltxplain.sql**	❑ Eingabe: Dateiname mit SQL-Statement ❑ Wenn SQL Binds enthält: ❑ Unverändert lassen, ODER mit Literalen vom gleichen Datentyp ersetzen
SQLT XTRSBY	sqlt/run/**sqltxtrsby.sql**	❑ Für Read-only-Datenbanken ❑ Ausführung auf Primary, Verbindung zu Stdby. ❑ Wie XTRACT ❑ Eingabe: SQL_ID und DBLINK

SQLHC
SQLT ist ein sehr mächtiges Werkzeug, das allerdings auch eine Installation von DB-Objekten erfordert. Dies ist einerseits in Produktionsumgebungen gelegentlich nicht erwünscht, und andererseits kommt eine Installation von SQLT, wenn das „Kind schon im Brunnen" ist, zu spät.
Hier kommt als Alternative „SQL Health Check" (SQLHC) ins Spiel, das ohne Installation aus SQL*Plus heraus ausgeführt werden kann. Es entspricht im Wesentlichen der Methode XTRACT von

SQLT, muss allerdings mit einem DBA-User ausgeführt werden, um alle notwendigen Abfragen durchführen zu können.
Das Script läuft entweder interaktiv oder wird mit den Parametern für

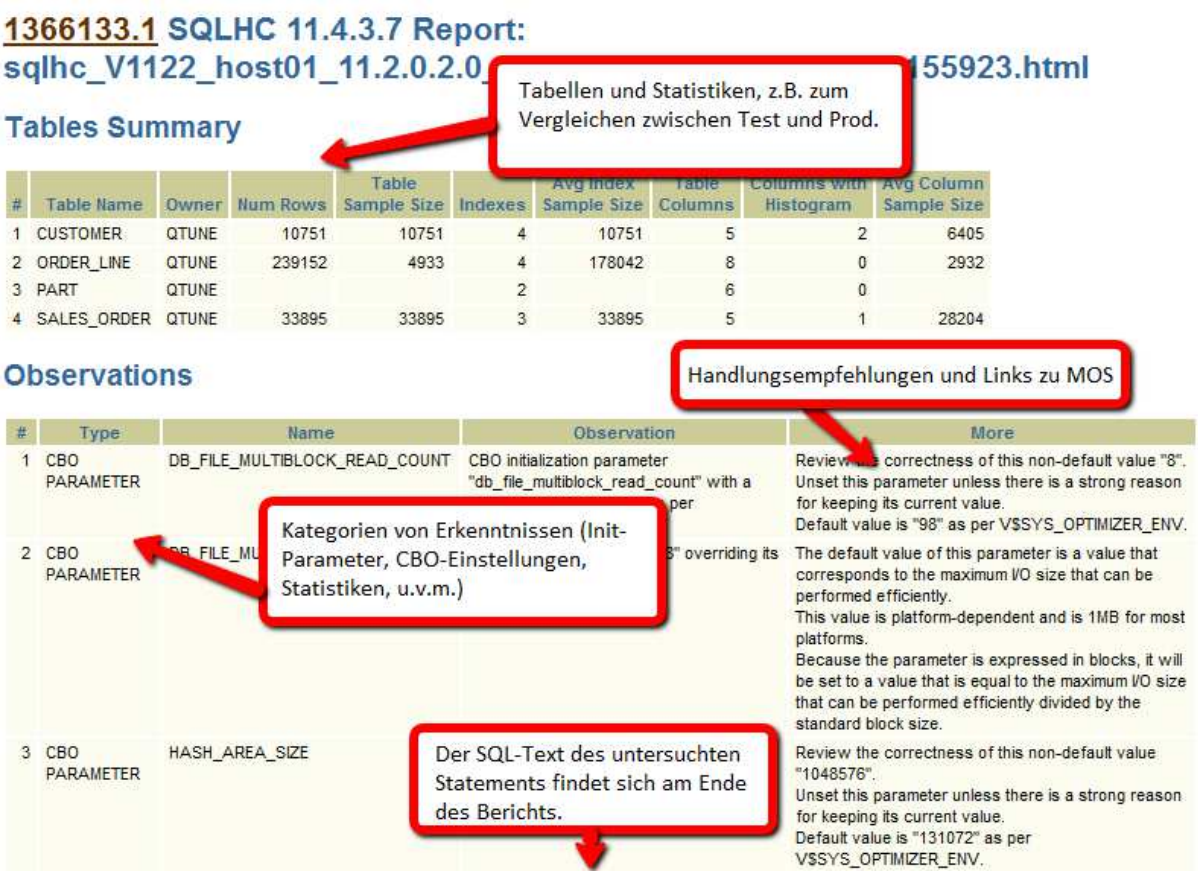
1. Das lizensierte Pack (**T**uning, **D**iagnostics oder **N**one)
2. Die zu untersuchende SQL-ID

gestartet.
Beispiel:

```
# sqlplus / as sysdba
```

```
SQL> @sqlhc T djkbyr8vkc64h
```

Das Endergebnis ist ein ZIP-File mit einem mehrteiligen, auf drei bis fünf HTML-Dateien verteilten Report. Dieser enthält:



- Eine Übersichtsseite mit
  - dem SQL-Text,
  - den wichtigsten Erkenntnissen und Handlungsempfehlungen
  - den an der Abfrage beteiligten Tabellen und Indizes nebst deren elementaren Statistiken
- mehrere Unterseiten mit

- allen für das SQL auffindbaren Ausführungsplänen
- Informationen zum jeweils benutzten Optimizer Environment
- Informationen zu den jeweiligen Inhalten von Bind-Variablen
- Ausführliche Informationen zu Optimizer-Statistiken aller am SQL beteiligten Tabellen und Indizes
- Chronologische Auflistungen von Ausführungsplänen und Laufzeiten
- Verwendete SQL Plan Baselines, SQL Profiles oder SQL Patches
- Informationen zu eventuell eingetretenem Cursor Sharing
- SQL-Monitor-Bericht (Flash-Version) über die letzte Ausführung.

Oft können bereits mit den Erkenntnissen aus dem Abschnitt „Observations“ auf der ersten Seite bestehende Probleme erkannt und Maßnahmen abgeleitet werden.

Ansonsten wird auf den weiteren Seiten nahezu jede, verfügbare Information zu dem untersuchten SQL bereitgestellt, durch deren Fülle man dank diverser Links gut hin- und her-navigieren kann.

Literatur

- SQLTXPLAIN Download und Dokumentation: MOS Doc ID 215187.1
- SQLHC Download und Dokumentation: MOS Doc ID 1366133.1
- SQLTXPLAIN FAQ: MOS Doc ID 1454160.1
- Blog von Carlos Sierra, dem Schöpfer von SQLT+SQLHC: http://carlos-sierra.net/
- Stelios Charalambides: Oracle SQL Tuning with Oracle SQLTXPLAIN, Apress 2013, ISBN13: 978-1-4302-4809-5

Kontaktadresse:
Uwe Küchler
Opitz Consulting Deutschland GmbH
Standort Bad Homburg
Norsk-Data-Str. 6
D-63152 Bad Homburg v.d.H.

Telefon: +49 (0) 6172-662600
E-Mail uwe.kuechler@opitz-consulting.com
Internet: www.opitz-consulting.com