# Macht ADDM wunschlos glücklich?

**Dr. Frank Haney**
**Consultant**
**Jena**

**Schlüsselworte:**

Oracle 10*g* Database, Performance Tuning, Extended SQL-Tracing

## Einleitung

In meinem Vortrag auf der 17. Oracle-Anwenderkonferenz im Vorjahr habe ich versucht herauszuarbeiten, daß eine End-to-End-Analyse von Performanceproblemen auf die Methode des *Extended SQL Tracing* nicht verzichten kann. Extended SQL Tracing ermöglicht es letztlich, die Auswirkungen des einzelnen Database Calls für die Systemperformance zu erfassen und zielführend festzustellen, wo die Antwortzeit bleibt. End-to-End ist hier in zweierlei Hinsicht gemeint, einerseits nicht nur bezogen auf die Leistungsfähigkeit des Datenbankservers, sondern auf alle Schichten einer komplexen Infrastruktur, also von der Benutzereingabe am Frontend bis zur Anzeige des Ergebnisses, und andererseits werden die Benutzeraktionen nicht als isolierte Statements analysiert, sondern in Ihrem sachlichen Zusammenhang als Ganzes betrachtet. Die seinerzeitig Untersuchung basierte auf Oracle 9*i*. Dort hatte sich auch gezeigt, daß man entgegen häufig anzutreffender gegenteiliger Auffassungen die sogenannten *Idle Waits* bei einer ernsthaften Performanceanalyse nicht vernachlässigen darf.
Mit 10*g* hat Oracle eine ganze Reihe von Diagnostikwerkzeugen neu eingeführt bzw. in ihrem Leistungsumfang deutlich erweitert, so daß sich die Frage stellt, ob damit das relativ aufwendige Erstellen und Auswerten von Trace Files nicht obsolet geworden ist. Ausgehend von einem Überblick über die Merkmale dieser Werkzeuge soll untersucht werden, wie zielführend sie sind und wie sie eine End-to-End-Analyse unterstützen. Auf dieser Grundlage kann dann eine eventuell weiterbestehende Bedeutung von Extended SQL Tracing auch unter Oracle 10*g* herausgearbeitet werden.

## Performancediagnostik in Oracle 10*g*

Die Haupteinwände gegen die in Oracle 9*i* zumeist propagierten Methoden der Performancediagnostik waren, daß sie zu wenig deterministisch sind, daß nicht der Beitrag der einzelnen Database Calls zur Antwortzeit, sondern diverse Benchmarks als Maßstab dienen, und die mangelnde Zuordenbarkeit der Daten zu Sessions bzw. konkreten Nutzeraktionen, d.h. die Tatsache, daß in den meisten Fällen auf der Basis systemweiter Statistiken geschlossen werden muß. Das sollte mit 10*g* nun alles anders werden, und nach Auffassung vieler Autoren revolutioniert Oracle damit die Performancediagnostik. Was gibt nun Veranlassung zu derart euphorischen Äußerungen?

1. Es werden viel mehr performancerelevante Daten erfaßt. Es gibt mehr als 300 neue dynamische Performanceviews (V$-Views). Außerdem haben viele eine SID-Spalte bzw. sammeln sessionbezogene Informationen.
2. Gegenüber vorherigen Versionen wurde der Zugriff auf das Wait Interface drastisch erweitert.
3. Früher ermöglichten die Standardmittel zur Performancediagnostik entweder Momentaufnahmen oder Summenbildung über größere Zeiträume. Eine Historie der Ereignisse, die sich dann zur Antwortzeit kumulieren, wurde schmerzlich vermißt. Das ist jetzt in Grenzen gegeben.
4. Wichtige Systemmetriken werden unabhängig von benutzergenerierten Reports automatisch erfaßt, und die Werte in kurzen Intervallen aktualisiert. Auch hier werden für eine beschränkte Dauer historische Daten vorgehalten.
5. Das System speichert performancerelevante Informationen nicht nur flüchtig in den entsprechenden Views, sondern auch permanent im *Automatic Workload Repository* (AWR).
6. Betriebssystemstatistiken werden jetzt standardmäßig gesammelt. Außerdem werden nunmehr mit Oracle-Mitteln auch Statistiken zugänglich, die früher nur mit Betriebssystembefehlen verfügbar waren.
7. Der *Automatic Database Diagnostic Monitor* (ADDM – verballhornt als Adam bezeichnet) erstellt auf der Basis der gesammelten Daten Expertisen und gibt Empfehlungen zur Beseitigung von Performanceengpässen.
8. Die Möglichkeiten für das *Extended SQL Tracing* wurden dahingehend erweitert, daß die Identifizierung bestimmter Nutzeraktionen einfacher ist. Außerdem wird dieses hilfreiche Feature jetzt offenkundig auch offiziell unterstützt.

Wichtige Elemente dieser neuen Diagnostikhilfsmittel sollen im folgenden kurz diskutiert werden. Dabei ist weder Vollständigkeit angestrebt, noch kann auf Details eingegangen werden. Grundlage sind natürlich die gesammelten Statistiken, die durch die diversen V$-Views dem Nutzer direkt und indirekt in der Aufbereitung durch Werkzeuge wie den ADDM zur Verfügung gestellt werden. Diese Statistiken wiederum basieren auf den vom Oracle-Kernel in den X$-Views gesammelten viel umfangreicheren Informationen. In Oracle 10*g* geben die Statistiken neue Sichtweisen, die es dem DBA ermöglichen sollen, sich auf die wesentlichen Performanceprobleme zu konzentrieren.

**Datenquellen für den ADDM**

Dazu hat Oracle zwei neue Hintergrundprozesse eingeführt, die die Daten sammeln und pflegen, den MMON (Manageability Monitor) und den MMNL (Manageability Monitor – Lightweight). Der erste sammelt in festgelegten Zeiträumen die Statistiken und gibt Warnungen aus, wenn bestimmte Schwellenwerte überschritten werden. Diese Schwellenwerte sind im Enterprise Manager individuell anpaßbar. Der zweite ist für die Berechnung und Aktualisierung der verschiedenen Metriken und die Überwachung der aktiven Sessions zuständig.

Eine Neuerung ist, daß die Zeit, die das System mit diversen Operationen beschäftigt ist, in 17 Werten zusammengefaßt wird. Diese Informationen sind systemweit mit

V$SYS_TIME_MODEL oder sessionbezogen mit V$SESS_TIME_MODEL abrufbar (Angabe jeweils in µs). In dieser sogenannten *Time Model Statistics* ist die *DB Time* eine wichtige Größe. In diese gehen die CPU-Zeit und die Summe aller Warteereignisse ein, die mit *Non Idle Waits* verbunden sind. Aus meiner Sicht gibt es hier zwei Probleme: Einerseits wird direkt aus der View nicht deutlich, daß es sich nicht ausschließlich um separate Werte handelt, sondern um eine Hierarchie, bei der einige Werte die Summe anderer sind. Andererseits kann die Fixierung auf die *Non Idle Waits* die Performancediagnostik auch in die Irre führen, wie es das von mir auf der Konferenz 2004 diskutierte Beispiel beeindruckend zeigt.
Auf ähnliche Weise werden die Warteereignisse systemweit von V$SYSTEM_WAIT_CLASS oder sessionbezogen von V$SESSION_WAIT_CLASS zu Klassen zusammengefaßt (Angabe in cs). Auch damit soll sich der Administrator auf die Bereiche konzentrieren können, die vordringlich Tuning erfordern. Insgesamt sind es 806 Warteereignisse, die in 9 Klassen zusammengefaßt werden. Davon werden 58 als *Idle* rubriziert. Auch hier könnte es fatal sein, diese einfach zu ignorieren.

Der mit 10*g* erweiterte Zugriff auf das Wait Interface eröffnet der Performancediagnostik wesentliche Informationsquellen, die in dieser Form bislang nicht verfügbar waren:

- Es wird nicht mehr nur die Dauer des letzten Warteereignisses und die Zeit gespeichert, die das System bzw. die Session aktuell wartet, sondern es ist jetzt auch möglich, sich über die View V$SESSION_WAIT_HISTORY die letzten 10(!) Warteereignisse einer Session anzusehen (Angabe in cs). Das ist natürlich für betriebsame Systeme zu wenig. Deswegen gibt es die View V$ACTIVE_SESSION_HISTORY. Diese kann die Warteereignisse (Angabe in µs) f ür einen längeren Zeitraum speichern, so daß mit ihr sinnvoll Performancediagnostik betrieben werden kann. Dabei wird jede Sekunde eine „Messung“ durchgeführt, praktisch ein Schnappschuß von V$SESSION_WAIT und anderen Views. Allerdings gibt es keinen Zeitraum, für den das Vorhalten der Daten garantiert werden kann, analog zur UNDO_RETENTION bei *Flashback Query*. Wenn der interne Puffer voll ist, werden die Daten überschrieben. Aber der Server speichert die Informationen im *Active Workload Repository*, und zwar in der View DBA_HIST_ACTIVE_SESS_HISTORY (Angabe ebenfalls in µs). Allerdings wird nur jede zehnte „Messung“ von V$ACTIVE_SESSION_HISTORY in diese View übernommen, d.h. Warteereignisse, die dazwischen zu Ende gehen, fallen durch dieses Raster, denn Ereignisse, die länger als eine Sekunde dauern, werden erst nach Abschluß erfaßt. Das könnte bei der Analyse von Anwendungen, die periodische Warteereignisse zwischen den einzelnen Calls verursachen, Probleme bereiten. Eine weitere Einschränkung ist, daß nur solche Sitzungen als aktiv zählen, die entweder gerade die CPU nutzen oder auf ein Ereignis warten, das nicht *idle* ist! Von V$SESSION_WAIT_HISTORY werden die (letzten 10!) *Idle Waits* zwar erfaßt, aber nicht von V$ACTIVE_SESSION_HISTORY. Positiv dagegen ist, daß auch Informationen über die gerade ausgeführte SQL festgehalten werden (Identifikation der SQL, ID des Ausführungsplans, gerade ausgeführtes Kommando).
- Erstmals ist es auch möglich, Informationen darüber zu bekommen, wie die Dauer der Einzelereignisse verteilt ist. Diese Histogramme werden für die verschiedenen Events, einschließlich der *Idle Waits*, geführt und durch die View V$EVENT_HISTOGRAMM dem Administrator zugänglich gemacht. Es wird in Buckets von $<2^0$ ms bis $>=2^{22}$ ms

angezeigt, wie oft das System entsprechend lange auf das jeweilige Ereignis gewartet hat. Allerdings aggregiert diese View die Events systemweit ab Instanzstart und liefert keine sessionbezogenen Informationen.

- Die die SQL-Verarbeitung betreffenden Views V$SQL und V$SQLAREA enthalten jetzt Spalten, die Auskunft darüber geben, wie lange die Anweisung auf Ereignisse wichtiger Klassen gewartet hat. Dazu gehören z.B. die Ereignisklassen *application*, *user I/O* und *concurrency*. Es werden aber keine sessionbezogenen Informationen gegeben und natürlich summiert die Ausgabe über alle Ausführungen des jeweiligen Cursor.

Eine wichtige Rolle für die Performancediagnostik in Oracle 10*g* spielen die *Metriken*. Dabei handelt es sich eigentlich um auf die Spitze getriebenes Benchmarking. 15 Views liefern Kenndaten für den Datenbankbetrieb. Angegeben ist immer die Kenngröße, die Maßeinheit und der Wert. Entweder es handelt sich um zeitbezogene Metriken wie z.B. *Physical Writes Per Second* oder um Verhältniszahlen (Ratios) wie z.B. die *Memory Sorts Ratio* oder die *Library Cache Ratio*. Insgesamt handelt es sich um Metriken, von denen viele früher schon in den Statspack-Reports verfügbar waren. Neu ist daran, daß diese Metriken jetzt vom System automatisch generiert werden, wobei der Meßzeitraum unterschiedlich ist, manche Metriken werden alle 15 Sekunden neu bestimmt, andere alle 60 Sekunden. Die aktuellen Systemmetriken werden von der View V$SYSMETRIC geliefert. Bei vielen Metriken ist auch eine Historie verfügbar. Diese reicht eine Stunde zurück. (Das entspricht dem Standardintervall der im Automatic Workload Repository gespeicherten Snapshots. Siehe dazu den nächsten Abschnitt.) Für die Systemmetriken heißt die entsprechende View V$SYSMETRIC_HISTORY. Speziell für die Systemmetriken wird mit der View V$SYSMETRIC_SUMMARY noch eine Auswertung geboten, die für die letzte Stunde Minimum, Maximum, Durchschnitt und Standardabweichung anzeigt. In diese Metriken sind auch die Warteereignisse einbezogen. Für alle 806 Ereignisse wird in der View V$EVENTMETRIC die kumulierte Wartezeit aller Sessions (Angabe in cs), die Anzahl der wartenden Sessions und die Anzahl der Ereignisse für die letzten 60 Sekunden erfaßt. Es gibt aber im Gegensatz zu anderen Metriken keine Historie für die Warteereignisse und es gibt auch keine Metrik über die Wartezeit einzelner Sessions.

Das *Automatic Workload Repository* (AWR) ersetzt in Oracle 10*g* für den Administrator die Notwendigkeit, manuell mit Statspack oder gar utlbstat/utlestat Snapshots zu erstellen und diese zu Reports zu verarbeiten. Jetzt werden die im Gegensatz zu früheren Releases viel umfangreicheren Snapshots automatisch vom MMON-Prozeß gezogen und permanent in der Datenbank im SYSAUX-Tablespace gespeichert (167 Tabellen und 67 Views). Zur Erleichterung des Zugriffs sind die meisten Tabellen des AWR range-partitioniert. Standardmäßig wird jede Stunde ein Snapshot gezogen und die Daten werden 7 Tage aufbewahrt. Diese Intervalle können aber geändert werden, entweder mit *Database Control* oder mit Prozeduren des Package DBMS_WORKLOAD_REPOSITORY. Außerdem ist es möglich Snapshots manuell anzulegen, Reports zu erstellen und Baselines für das Tuning festzulegen. Baselines stellen gewissermaßen den Ausgangspunkt und Maßstab für den Erfolg von Tuningmaßnahmen dar. Die Baseline-Snapshots werden nicht nach der entsprechenden Frist gelöscht.

## Der Automatic Database Diagnostic Monitor (ADDM)

All die vorstehend umrissenen Datenquellen bezieht ADDM in seine Analysen ein. Am komfortabelsten wird dieser über Database Control (den Enterprise Manager) bedient. Alternativ kann man das Package DBMS_ADVISOR benutzen. Es ist nun weder vom Umfang noch vom Gegenstand dieses Beitrags geboten, die verschiedenen Ausgaben. Einstellungen und sonstigen Features von ADDM vorzustellen oder gar durch Screenshots zu dokumentieren. Hier nur ein kurzer Überblick: Schon auf der Startseite von Database Control wird angezeigt, ob es performancerelevante Meldungen gibt. Über den Link *Advisor Central* kommt man dann zu einer Übersicht der verschiedenen diagnostischen Tasks, z.B. wird angezeigt, auf der Basis welcher Snapshots durch ADDM eine Diagnose durchgeführt wurde. Den Report kann man sich selbstverständlich betrachten. Eine Ebene tiefer (Link *Advisor ADDM*) können dann Tasks unter Angabe des Start- und des End-Snapshot erstellt werden. Als Resultat gibt ADDM eine Liste mit Resultaten seiner Analyse aus und Empfehlungen, wie dem Problem beizukommen sei. Diese Resultate und die darauf fußenden Empfehlungen können auch als Report ausgegeben werden. Allerdings erschließt sich auf der Benutzeroberfläche nicht unmittelbar, daß mit der Erstellung eines Report das eine Mal die ADDM-Findings und -Empfehlungen und das andere Mal die Differenzbildung von zwei AWR-Snapshots gemeint ist. Bei Problemen mit der SQL kann man sich auch die Identifikation der Anweisung verschaffen, die den Ärger verursacht. Die gegebenen Empfehlungen führen oft nicht unmittelbar zum Ziel, da sind weitere Recherchen notwendig. So kann es sein, daß gerade bei Problemen mit der Applikation ADDM empfiehlt, bei deren Ausführung einen Trace File zu erstellen, um die Ursache zu ermitteln. Daran zeigt sich eigentlich schon, daß auch ADDM *Extended SQL Tracing* nicht völlig ersetzen kann. Eine weitere Einschränkung darf hier nicht verschwiegen werden: AWR und ADDM sind Bestandteile des Diagnostic Pack, das zusätzlich zur Enterprise Edition lizenziert werden muß!

## Neue Möglichkeiten des *Extended SQL Tracing*

In der Vergangenheit gab es das Problem, daß das es praktisch keine offiziell unterstützte Möglichkeit gab, das *Extended SQL Tracing* für eine andere Session ein- und auszuschalten. Das ist aber für eine wirkliche End-to-End-Analyse unverzichtbar. Mit Oracle 10*g* hat sich auch das geändert. Es gibt das neues Package DBMS_MONITOR. Mit dessen Prozedur SESSION_TRACE_ENABLE kann man das Tracing für eine bestimmte Session unter Angabe, ob Waits und/oder Binds im Trace File erscheinen sollen, ein und mit einer analogen Prozedur ausschalten. Darüber hinaus bringt dieses Package eine wichtige Neuerung: Man kann das Tracing jetzt auch auf Client-, Service-, Module- und Action-Ebene aktivieren. Service bedeutet hier den Dienst, über den auf die Datenbank zugegriffen wird, Client Module und Action werden vom Entwickler mit den Prozeduren des Packages DBMS_APPLICATION_INFO gesetzt. Mit dem neuen Tool *trcsess* können Trace Files entsprechend dieser Kriterien spezifisch ausgewertet werden. Dazu kommt, daß dieses Tool auch die spezifischen Informationen einer bestimmten Session aus dem Trace File filtern kann. Das ist besonders nützlich bei einer Shared-Server-Konfiguration, wo die Informationen zu einer Session auf mehrere Trace Files verteilt sein können. Auch der TRCA (Trace Analyzer) hat eine Weiterentwicklung erfahren. Zwar wird er nach wie vor nicht supportet,

aber die Tatsache, daß nach zweieinhalb Jahren seit August 2005 eine neue Version (2.1) über Metalink verfügbar ist, macht deutlich, daß Oracle dem *Extended SQL Tracing* neben den vorstehend geschilderten neuen Diagnostikmitteln einen hohen Stellenwert einräumt. Der TRCA gibt jetzt einen HTML-Report aus.

**Die Tests und ihre Resultate**

Für die Untersuchung wurden verschiedene speziell präparierte Anwendungen eingesetzt. Die Präparation bestand darin, daß in den Code Performancebremsen eingebaut wurden, die die Anwendungen gewissermaßen künstlich „entschleunigt" haben, um zu sehen, was die neuen Hilfsmittel im Sinne der eingangs apostrophierten End-to-End-Analyse leisten. Getestet wurde in verschiedenen Konfigurationen, lokal am Server und von einem Client-Rechner aus, mit und ohne zwischengeschalteten Application Server. Zur Laufzeit der Anwendungen war eine Benutzerinteraktion nicht möglich, um ausschließen zu können, daß eventuelle *Idle Waits* auf eine fehlende Eingabe des Benutzers zurückgehen. Im Kern handelte es sich um immer die gleiche Abfrage, die je nach Programmierung eine Antwortzeit von 2 bis 20 Minuten hatte. Vor und nach dem Lauf wurde jeweils ein Snapshot erstellt sowie das Extended SQL Tracing im Level 8 (mit Waits, aber ohne Binds) ein- bzw. ausgeschaltet.

Zur Auswertung wurden AWR- und ADDM-Reports herangezogen. Erstere wurden sowohl für einzelne Ausführungen einer bestimmten Anwendung als auch für einen größeren Zeitraum erstellt. Die Aussagen sind zwar detaillierter als in früheren Releases, tragen aber nur wenig zur Suche nach den Ursachen des angenommenen Performanceproblems bei. Das liegt zum einen daran, daß die Idle Waits wie in früheren Releases eher nebenbei aufgelistet werden, und zum anderen an der Tatsache, daß keine sessionbezogenen Informationen über die Warteereignisse verfügbar sind. Es ist leichter geworden, die problematische SQL zu identifizieren, was aber die Antwortzeit hauptsächlich determiniert hat, muß man aus anderen Quellen erschließen. Die Views V$EVENTMETRIC, V$EVENT_HISTOGRAMM, V$SESSION_WAIT_HISTORY oder V$ACTIVE_SESSION_HISTORY helfen hier auch nicht weiter, denn einerseits fehlt die Korrelation zur Benutzeraktion und andererseits bieten sie entweder nur systemweite Daten, nur die letzten 10 Ereignisse oder gar keine *Idle Waits*.

Ähnlich ist es mit dem ADDM-Report. Hier gab es noch eine Besonderheit. Bei dem Versuch ADDM-Tasks spezifisch für die Laufzeit der verschiedenen Applikationen zu erstellen, kam es zu folgender Meldung: *There was no significant database activity to run the addm*. ADDM braucht also ein Minimum an Informationen, das offenkundig von den genannten Quellen für die Laufzeit der Anwendung, immerhin bis 20 Minuten, nicht ausreichend zur Verfügung gestellt werden kann. (Die entsprechenden Trace Files hatten aber immerhin je ca. 1 MB.) Es mußte also ein größerer Abstand (ca. 90 Minuten) gewählt werden, um von ADDM einen Report zu bekommen. Damit stellt sich die Frage, ob ADDM das geeignete Hilfsmittel für eine End-to-End-Analyse ist. Bei einem längeren Abstand zwischen den Snapshots, auf denen die ADDM-Analyse basiert, geht die Korrelation zur Benutzeraktion wieder verloren.
Die Ergebnisse des Reports und die daraufhin gegebenen Empfehlungen hatten nun aber überhaupt nichts mit dem gesetzten Performanceproblem zu tun. Das ist auch nicht weiter verwunderlich, denn bei der speziellen Zurichtung der Tests äußerte sich dieses hauptsächlich in *Idle Waits* zwischen den Database Calls. Da ADDM aber dazu nur summarische

Informationen zur Verfügung stehen, kann er die Wartezeit und deren Beitrag zur Antwortzeit nicht zuordnen. Es wäre auch unredlich, denn es würde bedeuten, von ADDM eine Entscheidung zu verlangen, ob ein Idle Wait durch fehlende Benutzereingaben oder ähnliches zustande kommt oder Hinweis auf ein echtes Problem der Applikation ist. Unter den gegebenen Verhältnissen gibt es also nur eine Möglichkeit Idle Time mit ihrem signifikanten Beitrag zur Antwortzeit zu erkennen: Extended SQL Tracing. Die Auswertung der Trace Files soll hier nicht dargelegt werden, da gelten die von mir auf der letzten Konferenz herausgearbeiteten Prinzipien für eine deterministische Performanceanalyse.

## Fazit

ADDM ist auf jeden Fall ein mächtiges Tool, das die Performance Diagnostic sehr erleichtern kann. Es gibt aber ein paar Einschränkungen, die einer Verwendung für eine wirkliche End-to-End-Analyse abträglich sind. Diese betreffen insbesondere

- die Eingrenzung der Analyse auf eine bestimmte Benutzeraktion,
- periodisch wiederkehrende Warteereignisse
- das Erkennen von performancerelevanten Problemen, die in der Anwendung zu suchen sind sowie
- Situationen, in denen die Einschätzung der Antwortzeit von der Zuordnung der Warteereignisse zu einzelnen Database Calls abhängt.

In diesen Fällen wird man auf Extended SQL Tracing nicht verzichten können. Auch dieses eröffnet natürlich keinen Universalschlüssel zu allen Performanceproblemen, kann aber auch dann zeigen, in welcher Schicht des Gesamtsystems die Antwortzeit bleibt, wenn ADDM dazu nicht in der Lage ist. Immerhin räumt Oracle selber jetzt dem Extended SQL Tracing nicht nur als Instrument für den Support, sondern auch als Hilfsmittel für die Diagnostik durch den Endanwender einen größeren Stellenwert ein.

Nicht nur, daß Extended SQL Tracing als zum ADDM komplementäre Methode immer mit ins Kalkül gezogen werden muß, auch für diesen selber sind durchaus noch Verbesserungen wünschbar. Zum einen sollte den Idle Waits ein höherer Stellenwert eingeräumt werden. Das betrifft vor allem die Idle Waits zwischen einzelnen Database Calls, die nicht durch den Benutzer beeinflußbar sind, also den Fall, daß die Session de facto aktiv und gleichzeitig idle ist. Die Idle Waits tragen in diesem Fall ja signifikant zur Antwortzeit bei. Wichtig wäre auch, Histogramme für die Warteereignisse auf Sessionebene verfügbar zu machen.

## Literatur


K. Floss: Tracing SQL in Oracle Database 10*g*. Oracle Magazin, September/October 2004, pp. 79-80.

F. Haney: Was bringt Extended SQL-Tracing für das Performance-Tuning? Vortragsband zur 17. Deutschen Oracle-Anwenderkonferenz. DOAG, Berlin 2004, pp.375–386. ISBN 3-928490-15-X.

C. V. Millsap, J. L. Holt: Optimizing Oracle Performance. O'Reilly, Sebastopol (CA) 2003, 390 pp. ISBN 0-596-00527-X.

A. Nanda: Perform without Waiting. Oracle Magazin, July/August 2004, pp. 79–82.

R. J. Niemiec: Oracle 9i Performance Tuning. Tips & Techniques. Mc Graw Hill, New York 2003, 826 pp. ISBN 0-07-222473-8.

R. Schumacher: Vereinfachte Antwortzeitanalysen in Oracle 10*g*. DOAG News, Q3/2005, pp. 52–58.
R. Shee, K. Deshpande, K. Gopalakrishnan: Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning. Mc Graw Hill, New York 2004, 350 pp. ISBN 0-07-222729-X.
Oracle10*g* Database Online Documentation, Release 1.
http://metalink.oracle.com
http://www.hotsos.com (Webseite der Firma von Cary Millsap)

**Kontaktadresse:**

**Dr. Frank Haney**
Anna-Siemsen-Str. 5
D-07745 Jena

Telefon: +49(0)3641-210224
Fax: +49(0) 3641-210224
E-Mail info@it-haney.de
Internet: www.it-haney.de