Unsichere Bindevariablen

Wilhelm Breßer
Berater
Dezember 2008

- **Die Firma Herrman & Lenz wurde 1995 gegründet und hat aktuell 12 Mitarbeiter.**
- **Firmensitz: Burscheid (bei Köln).**
- **Beratung, Schulung und Fernwartung für das Thema *Oracle Datenbanken.***
- **Schwerpunktthemen: Hochverfügbarkeit, Tuning, Migrationen und Troubleshooting.**
- **Viele DOAG-Aktivitäten (Leitung der Regionalgruppe NRW, Vorträge für Regios, SIGs und Konferenzen).**

Weitere Aktivitäten

Agenda

- Einleitung
- Begriffserklärungen
- Initialisierungsparameter
- Optimizer Features
- Cursor Sharing
- Bind Peeking
- Adaptive Cursor Sharing
- Dynamic Sampling
- Ausführungspläne
- Tracing
- Bugs
- Quellen
- Zusammenfassung

Einleitung

- Allgemeines
- Historie
- Problematik

Allgemeines

Bindevariablen werden aufgrund ihrer Vorteile häufig eingesetzt:

- **Fragmentierung des Shared Pool reduzieren**
- **Statements zwischen Sessions "shareable" machen**
- **Speicherplatz im Shared Pool einsparen**
- **Teures Hard-Parsing vermeiden**

Historie

- Bindevariablen werden bereits sehr lange vom Oracle RDBMS unterstützt.
- Andere Features, die für eine performante SQL-Ausführung und / oder eine gute Ausnutzung des Shared Pool wichtig sind, wurden später eingeführt:
  - Cost Based Optimizer in Version 7
  - OPTIMIZER_FEATURES_ENABLE - Release 8.0.4
  - CURSOR_SHARING - Release 8.1.6
  - Bind Peeking in Version 9i
  - Adaptive Cursor Sharing - Release 11.1.0.6.0

Problematik

- Der CBO kann die tatsächlichen Werte von benutzerdefinierten Bindevariablen vor Version 9i überhaupt nicht erkennen
- Die Auswertung des Inhaltes von Bindevariablen erfolgt nur einmalig zur Parsezeit (>= 9i < 11g)
- Die Selektivität von Bereichsprädikaten kann nur sehr bedingt ermittelt werden
- Der CBO muß Annahmen machen bzw. fest implementierte Werte benutzen
- Je genauer Objektstatistiken sind, um so effektiver kann der Ausführungsplan sein

Begriffserklärungen

- Kardinalität
- Selektivität
- Cost
- Clustering-Faktor
- Density
- Cursor
- Unsichere Bindevariablen

Kardinalität

Die Kardinalität ist die Anzahl der aus einem Row Set extrahierten Rows.

Beispiele für mögliche Row Sets sind:

- **Tabelle**
- **View**
- **Materialized View**
- **Ergebnis eines Joins**
- **Ergebnis einer Group By Operation**

Kardinalität wird vom CBO u.a. benutzt, um die Kosten von Nested Loop Joins und Sortieroperationen zu ermitteln.

Selektivität

- Selektivität ist die Anzahl der Datensätze, die von einer Query aus einem Row Set tatsächlich selektiert werden.

- Selektivität hängt von Prädikaten ab, da diese als Filter wirken. Die Selektivität eines Prädikates bestimmt somit, wieviele Zeilen aus einem Row Set zurückgegeben werden.

- Innerhalb des Oracle Kernels wird Selektivität mit Werten zwischen 0 und 1 ausgedrückt. Je kleiner der Wert ist, um so selektiver ist das jeweilige Prädikat.
  0 – es wird keine Zeile zurückgegeben
  1 – es werden alle Zeilen zurückgegeben

Defaultwerte für Prädikat-Selektivität

Bindevariablen mit Equalityprädikat	**→ 1/NDV**
Bindevariablen mit Bereichsprädikaten	**→ 5%**
Bindevariablen mit Histogrammen	**→ 5%**
Bindevariablen mit "like"	**→ 25%**

NDV = Number of Distinct Values

Wenn diese Voraussetzungen aufgrund ungleichmäßiger Datenverteilung nicht der Realität entsprechen, ist die Erstellung aktueller Histogramme angezeigt.

Density

Density ist ein anderer Ausdruck für Selektivität.

Die Basisformel zur Ermittlung der Selektivität lautet:

$$\text{Selectivity} = \frac{\text{Number of records satisfying a condition}}{\text{Total Number of records}}$$

Cost (1)

- Cost steht für Arbeitseinheiten bzw. Einheiten benutzter Ressourcen.
- Einheiten können sein:
  - Anzahl von Disk I/O
  - CPU-Nutzung
  - Speichernutzung

  für die Ausführung von Operationen (z.B. Table Scan, Indexzugriff, Joins, Sortiervorgänge)
- Die in einem Ausführungsplan aufgeführten Kosten geben die Anzahl der Einheiten wieder, die voraussichtlich aufgewendet werden müssen, um eine Query auszuführen und deren Ergebnis zu liefern.

Cost (2)

- Der Zugriffspfad bestimmt die Anzahl der Einheiten, die aufgewendet werden müssen, um Daten zu selektieren.
- Folgende Zugriffspfade sind möglich:
  - Table Scan
  - Fast Full Index Scan
  - Index Scan
- Während eines Table Scan oder eines Fast Full Index Scan werden mit einem Lesevorgang mehrere Blöcke von Platte gelesen. Somit hängen die Kosten dieser Zugriffspfade von der Anzahl der zu lesenden Blöcke und dem Server-Initialisierungsparameter DB_FILE_MULTI_BLOCK_READ_COUNT ab (sofern keine Systemstatistiken benutzt werden).

Cost (3)

- Die Kosten eines Index Scan hängen von folgenden Faktoren ab:
  - Anzahl der Level der benutzten B-Tree-Indizes,
  - Anzahl der zu scannenden Leaf-Blöcke
  - Anzahl der auf Basis der in den Index-Schlüsseln enthaltenen Row-IDs zu lesenden Zeilen
- Die Höhe der Kosten zum Lesen von Zeilen auf Basis von in Index-Schlüsseln enthaltenen Row-IDs hängt vom Clustering-Faktor der beteiligten Indizes ab.

Clustering-Faktor

Der Clustering-Faktor gibt an, wieviele Datenblöcke aufgrund der in Index-Schlüsseln gefundenen Row-Ids tatsächlich gelesen werden müssen, um das Ergebnis einer Query zu produzieren, drückt also ein Sortierverhältnis zwischen Tabelle und Index aus.

Cursor

- Speicherbereich, in dem ein geparstes SQL-Statement oder andere Informationen für die Ausführung eines SQL-Befehles gehalten werden.
- Seit Oracle Release 8.1.6 kann ein solcher Bereich zwischen verschieden Sessions "geshared" werden.
  - Möglich für SQL-Statements und PL/SQL-Blöcke.
  - Der entsprechende Bereich im Shared Pool wird als „shared SQL Area“ bezeichnet.

Unsichere Bindevariablen

- **Bindevariablen werden vom Cost Based Optimizer unter folgenden Voraussetzungen als "unsicher" betrachtet:**
  - **Verschiedene Werte einer Spalte führen zu verschiedenen Ausführungsplänen**
    - Range Type Query (Verwendung von Bereichsprädikaten)
    - Query mit INLIST-Operator
    - Verwendung von Histogrammen
- **Problematisch bei CURSOR_SHARING=SIMILAR, weil der CBO jedes zu ersetzende Literal darauf hin überprüft, ob unterschiedliche Werte zu verschiedenen Ausführungsplänen führen können.**
- **Nicht relevant bei CURSOR_SHARING=FORCE, weil der CBO dann grundsätzlich immer eine Ersetzung vornimmt.**

Initialisierungsparameter (1)

Folgende Initialisierungsparameter haben Einfluß auf das Verhalten des Cost Based Optimizer

- **OPTIMIZER_MODE** >= 7
- **OPTIMIZER_FEATURES_ENABLE** >= 8.0.4
- **OPTIMIZER_DYNAMIC_SAMPLING** >= 9.2.0
- **OPTIMIZER_INDEX_CACHING** >= 8.0.5
- **OPTIMZER_INDEX_COST_ADJ** >= 8.0.5
- **OPTIMIZER_SECURE_VIEW_MERGING** >= 10.2.0
- **OPTIMIZER_MAX_PERMUTATIONS** >= 8.0.4 - < 10.1.0
  - **_OPTIMIZER_MAX_PERMUTATIONS** >= 10.1.0

Initialisierungsparameter (2)

- OPTIMIZER_PERCENT_PARALLEL <= 9.0.1
  - _OPTIMIZER_PERCENT_PARALLEL >= 9.2.0
- CURSOR_SHARING >= 8.1.6
- CURSOR_SPACE_FOR_TIME >= 8.1.6
- DB_FILE_MULTI_BLOCK_READ_COUNT >= 7
- HASH_MULTIBLOCK_IO_COUNT >= 7 – < 9.0.1
  - _HASH_MULTIBLOCK_IO_COUNT >= 9.0.1
- HASH_JOIN_ENABLED >= 7
- PGA_AGGREGATE_TARGET >= 9.0.1
- SORT_AREA_SIZE / HASH_AREA_SIZE >= 7
- SORT_AREA_RETAINED_SIZE >= 7

Initialisierungsparameter (3)

- STAR_TRANSFORMATION_ENABLED >= 8.0.4
- ALWAYS_ANTI_JOIN >= 7 - < 9.0.1
  - _ALWAYS_ANTI_JOIN >= 9.0.1
- ALWAYS_SEMI_JOIN >= 8.0.4 - < 9.0.1
  - _ALWAYS_SEMI_JOIN >= 9.0.1

Initialisierungsparameter (4)

- _OPTIM_PEEK_USER_BINDS
- _PX_BIND_PEEK_SHARING
- _XPL_PEEKED_BINDS_LOG_SIZE
- _CURSOR_CACHE_FRAME_BIND_MEMORY
- _LIKE_WITH_BIND_AS_EQUALITY
- _CURSOR_BIND_CAPTURE_AREA_SIZE
- _CURSOR_BIND_CAPTURE_INTERVAL
- _OPTIMIZER_COST_MODEL
- _OPTIMIZER_ROWNUM_BIND_DEFAULT
- _XSOLAPI_SQL_USE_BIND_VARIABLES

Cost Based Optimizer Features (1)

- Einführung des Cost Based Optimizer mit Oracle 7
- Der Serverinitialisierungsparameter OPTIMIZER_FEATURES_ENABLE ermöglicht, Features des CBO gezielt zu aktivieren bzw. zu deaktivieren.
- Der Parameter wurde mit Release 8.0.4 eingeführt. Seine Defaultwerte bzw. möglichen Werte sind auf der nächsten Folie dargestellt.
- Die Nutzung neuer Optimizer Features kann oftmals bessere Ausführungspläne und somit eine gesteigerte Performance zur Folge haben, aber in Einzelfällen auch Probleme verursachen.
- Seit Release 10.1 mit alter system / alter session dynamisch änderbar

Cost Based Optimizer Features (2)

RDBMS Release	Default-wert	Mögliche Werte
8.0.x	8.0.0	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6
8.1.x	8.1.x	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7
9.0.x	9.0.0	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7, 9.0.0
9.2.x	9.2.0	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.0.7, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7, 9.0.0, 9.0.1, 9.2.0
10.1.x	10.0.0	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.0.7, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7, 9.0.0, 9.0.1, 9.2.0, 10.0.0
10.2.x	10.2.0.1	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.0.7, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7, 9.0.0, 9.0.1, 9.2.0, 10.0.0, 10.1.0, 10.1.0.3, 10.1.0.4, 10.2.0.1
11.1.x	11.1.0.6	8.0.0, 8.0.3, 8.0.4, 8.0.5, 8.0.6, 8.0.7, 8.1.0, 8.1.3, 8.1.4, 8.1.5, 8.1.6, 8.1.7, 9.0.0, 9.0.1, 9.2.0, 10.1.0, 10.1.0.3, 10.1.0.4, 10.1.0.5, 10.1.0.6, 10.2.0.1, 10.2.0.2, 10.2.0.3, 10.2.0.4, 11.1.0.6

Cursor-Sharing (1)

- Nutzung eines Cursors durch verschiedene Sessions.
- Serverinitialisierungsparameter CURSOR_SHARING
  - EXACT
  - FORCE
  - SIMILAR (seit Version 9i)
- Dynamisch änderbar (alter system, alter session)
- Default: EXACT

Cursor-Sharing (2)

- **Sharing möglich, wenn**
  - **Text (einschließlich Groß-/Kleinschreibung und Leerzeichen) absolut identisch ist.**
  - **Referenzen auf Schemaobjekte auf dasselbe Objekt im selben Schema verweisen.**
  - **Bindevariablen in SQL-Befehlen bzw. PL/SQL-Blöcken einen identischen Namen haben und vom selben Datentyp sind.**
  - **Die Statements alle mit demselben Optimizer und bei Verwendung des CBO mit dem identischen OPTIMIZER_GOAL, z.B. ALL_ROWS, optimiert worden sind.**

Cursor-Sharing (3)

- CURSOR_SHARING=FORCE führt zur Ersetzung von Literalen ähnlicher SQL-Befehle durch system-generierte Bindevariablen (z.B. :SYS_B_0) solange dies die Bedeutung der Statements nicht verändert.
- Bei CURSOR_SHARING=EXACT werden Statements nur als "shareable" betrachtet, wenn sie in ihrem Text vollständig übereinstimmen.
- CURSOR_SHARING=SIMILAR bewirkt, daß Statements, die sich in Literalen unterscheiden, aber ansonsten identisch sind, als "shareable" betrachtet werden, solange die Literale keine Auswirkung auf die Bedeutung des Statements oder den Grad der Statement-Optimierung haben. → Bind Peeking

Cursor-Sharing (4)

➢ CURSOR_SHARING FORCE vs. SIMILAR

- **Bei FORCE wird bei der ersten Ausführung ein Bind Peeking durchgeführt und der Ausführungsplan erstellt.**
- **SIMILAR wirkt wie FORCE mit folgender Ausnahme:**

  Der CBO prüft, ob unterschiedliche Werte des übergebenen Literals zu unterschiedlichen Ausführungsplänen führen können. Falls ja, wird die Bindevariable als "unsicher" gekennzeichnet und ein Child-Cursor erzeugt.

Cursor-Sharing (5)

- Der Cost Based Optimizer geht davon aus, daß bei der Nutzung von Bindevariablen
  - ein Cursor-Sharing beabsichtigt ist!
  - verschiedene Varianten eines Statements den selben Ausführungsplan benutzen können
- Falls verschiedene Varianten eines SQL-Befehles von unterschiedlichen Ausführungsplänen signifikant profitieren würden, ist
  - die Nutzung von Bindevariablen wahrscheinlich nicht angemessen (>= 9i <= 10g)
  - Bind-Aware Cursor-Matching erforderlich (= 11g)

Beispiele für Cusor-Sharing-Kriterien (1)

Die Statements

```
select * FROM ma;
select *  FROM ma;
```

bzw.

```
select * from ma;
select * from MA;
```

sind nicht "shareable", weil sie in ihrem Text nicht übereinstimmen.

Beispiele für Cursor-Sharing-Kriterien (2)

Die Statements

```
insert into kunden values (67308, 'Heinz Klein
  GmbH', 3);
insert into kunden values (67309, 'Marion
  Klinger OHG', 2);
```

sind "shareable", weil sie sich nur in Literalen unterscheiden.

Derartige Statements können "geshared" werden, wenn der Parameter CURSOR_SHARING die Ausprägung FORCE hat.

Beispiele für Cusor-Sharing-Kriterien (3)

Das Statement

```
select artikel, menge from bestellung;
```

ist nicht "shareable", wenn es mehrere Tabellen mit dem Namen `bestellung` gibt, kann jedoch "shareable" gemacht werden, wenn in der Abfrage auf das Schema referenziert wird:

```
select artikel, menge from user1.bestellung;
```

Beispiele für Cusor-Sharing-Kriterien (4)

Die Statements

```
select ma_nr, ma_name from mitarbeiter where
  abtno = :abt
select ma_nr, ma_name from mitarbeiter where
  eintrittsdatum = :einst_dat;
```

sind nicht "shareable", weil die Bindevariablen unterschiedliche Namen haben.

Beispiele für Cusor-Sharing-Kriterien (5)

Die Statements

```
select * from mitarbeiter where abtno = :10;
select * from mitarbeiter where abtno = :30;
```

sind ebenfalls nicht "shareable", weil die Bindevariablen unterschiedliche Namen haben, können jedoch so umgeschrieben werden, daß sie "shareable" sind:

```
select * from mitarbeiter where abtno = :abt;
```

Cursor-Sharing

Demonstration

Bind Peeking

- Betrifft benutzerdefinierte Bindevariablen
- Vor dem ersten Hard-Parse ermittelt der CBO den aktuellen Wert der Bindevariablen und benutzt diesen als Basis für die Ermittlung der Selektivität der im Statement enthaltenen where-Bedingung(en).
- Bei erneuter Ausführung des Cursors bzw. einem Soft-Parse findet kein weiteres Peeking statt und der Cursor wird aufgrund der allgemeinen Sharing-Kriterien als "shareable" oder "non-shareable" eingestuft.
- Gilt auch, wenn weitere Ausführungen des Cursors unterschiedliche Bindevariablen benutzen.
- Weiterentwicklung in Version 11g

Adaptive Cursor Sharing (1)

ORACLE DATABASE 11g

- Ein Cursor wird als "Bind Sensitive" gekennzeichnet, wenn der CBO beim Bind Peeking und unter Nutzung eines Histogramms zu dem Schluß kommt, daß der optimale Ausführungsplan von dem Wert der Bindevariablen abhängt.
- Wenn ein Cursor als "Bind Sensitive" gekennzeichnet ist, überwacht das RDBMS das Verhalten desselben bei Nutzung unterschiedlicher Werte von Bindevariablen daraufhin, ob unterschiedliche Ausführungspläne erzeugt werden.
- Trifft dies zu, wird der Cursor als "Bind Aware" markiert und das sogenannte "Bind-Aware Cursor-Matching" aktiviert.

Adaptive Cursor Sharing (2)

ORACLE DATABASE 11g

- Mit aktiviertem "Bind-Aware Cursor-Matching" berücksichtigt der CBO bei der Ermittlung des Ausführungsplanes den tatsächlichen Wert der Bindevariablen und die dafür berechnete Selektivität. Daher können SQL-Statements mit benutzerdefinierten Bindevariablen oder durch CURSOR_SHARING=FORCE erzeugten Bindevariablen in Abhängigkeit von den Werten der Bindevariablen mehrere Ausführungspläne haben.

Adaptive Cursor Sharing

Demonstration

Dynamic Sampling (1)

- **Dynamic Sampling ist ein Verfahren zum Sammeln von fehlenden oder "nicht aktuellen" Statistiken durch den CBO zur Kompilierzeit von SQL-Statements. Der Cost Based Optimizer bestimmt beim Kompilieren, ob für eine Query durch Dynamic Sampling Informationen gewonnen werden können, die die Laufzeit des Statements verbessern.**
  - **Sampling durch Erzeugung von rekursivem SQL**
  - **Verfahren erzeugt Overhead**
  - **Level einstellbar über den Serverinitialisierungsparameter OPTIMIZER_DYNAMIC_SAMPLING**
  - **Zusätzlich nutzbar über den Optimizer-Hint DYNAMIC_SAMPLING**

Dynamic Sampling (2)

- **OPTIMIZER_DYNAMIC_SAMPLING**

  Der Defaultwert dieses Parameters hat sich zwischen Release 9i und 10g geändert.

9.2.x	**OPTIMZER_FEATURES_ENABLE**	**= 9.2.0 → 1**
	OPTIMZER_FEATURES_ENABLE	**<= 9.0.1 → 0**
10.1.x	**OPTIMZER_FEATURES_ENABLE**	**>= 10.0.0 → 2**
	OPTIMZER_FEATURES_ENABLE	**= 9.2.0 → 1**
	OPTIMZER_FEATURES_ENABLE	**<= 9.0.1 → 0**
10.2.x	**wie 10.1.x**	
11.1.x	**wie 10.1.x**	

- **Mögliche Werte: 1 – 10 (0 → kein Dynamic Sampling)**

Dynamic Sampling (3)

- **Die Einstellung des Serverinitialisierungsparameters OPTIMIZER_DYNAMIC_SAMPLING bestimmt den Umfang des Dynamic Sampling:**

  1 – Statistik-Sampling aller nicht analysierten Tabellen, die folgende Bedingungen erfüllen:

  - Die Query enthält mindestens 1 nicht analysierte Tabelle
  - Diese Tabelle ist Bestandteil eines Joins oder steht in einer Subquery oder in einer non-mergeable View
  - Diese Tabelle hat keine Indizes
  - Diese Tabelle hat mehr Blöcke, als die Anzahl der Blöcke beträgt, die für ein Dynamic Sampling gelesen werden (per Default 32 Blöcke)

  2 – Sampling (Statistik und Histogramme) aller nicht analysierten Tabellen mit der doppelten Anzahl der per Default gesampleten Blöcke → 64

Ausführungspläne

- Ausführungspläne sollten bei Nutzung von Bind Peeking stets mit TKPROF auf Basis eines Trace-Files und nicht mit AUTOTRACE oder Explain Plan erstellt werden, weil `AUTOTRACE TRACEONLY EXPLAIN` sowie `EXPLAIN PLAN` kein Bind Peeking durchführen und der zur Parsezeit erzeugte Ausführungsplan aufgrund von Bind Peeking anders aussehen kann.
- Eine alternative Möglichkeit ist ab Release 9.2.0 der Einsatz des Packages DBMS_XPLAN

```
select * from
table(dbms_xplan.display_cursor('<sql
id>',null,'ADVANCED'));
```

SQL-Tracing (1)

- **Für die eigene Session aktivieren:**
  - `alter session set sql_trace = true;`
  - `alter session set events '10046 trace name context forever, level 12';`
  - `alter session set events '10053 trace name context forever, level 2';`

- **Für die eigene Session deaktivieren:**
  - `alter session set sql_trace = false;`
  - `alter session set events '10046 trace name context off;`
  - `alter session set events '10053 trace name context off;`

SQL-Tracing (2)

- **Für andere Sessions aktivieren:**
  - `exec dbms_support.start_trace_in_session (sid, serial#, true, true);`
  - `exec dbms_system.set_ev (sid, serial#, 10046, 12);`
  - `exec dbms_system.set_ev (sid, serial#, 10053, 2);`
  - `exec dbms_monitor.session_trace_enable (sid, serial#, true, true);` **(seit 10g)**

- **Für andere Sessions deaktivieren:**
  - `exec dbms_support.stop_trace_in_session (sid, serial#);`
  - `exec dbms_system.set_ev (sid, serial#, 10046, 0);`
  - `exec dbms_system.set_ev (sid, serial#, 10053, 0);`
  - `exec dbms_monitor.session_trace_disable (sid, serial#);` **(seit 10g)**

SQL-Tracing (3)

- Eine weiter Möglichkeit zum Tracing besteht mit dem nicht vollständig dokumentierten Oracle-Tool ORADEBUG (siehe Metalink)
- Zur Analyse von Trace-Files, die mit Events (10046, 10053) erstellt wurden, eignet sich besonders TKPROF und in begrenztem Umfang auch der über Metalink erhältliche Oracle Trace Analyzer.

Bugs

Zu Cursor-Sharing und Bindevariablen sind etliche Bugs bekannt, z.B.:

- 2992537: 9i CBO uses merge join cartesian inappropriately
- 3132098: Bind Peeking does not work for duplicate Binds
- 3406977: High version count due to binds marked as non-data with cursor_sharing=force
- 3582980: Dynamic_Sampling Hint ignored in some cases
- 4567767: Unexplained plan changes can occur without stats regather
- 4874738: High version count due to unsafe binds
- 5056340: Cursor Sharing behavior and histograms on a column
- 5082178: In some situations bind peeking can occour when it should not
- 5146740: Wrong results with bind variables/CURSOR_SHARING
- 5364143: Unpredictable Change in Query Optimizer Plan
- 5863277: ORA-1008 from SQL on second run when cursor_sharing=similar/force
- 6163785: Intermittent wrong results with dblink and cursor_sharing

Quellen (1)

Literatur:

- Cost Based ORACLE Fundamentals, Jonathan Lewis
- Effective Oracle by Design, Thomas Kyte
- A look under the hood of CBO: The 10053 Event, Wolfgang Breitling

Online-Quellen:

- http://metalink.oracle.com
- http://asktom.oracle.com
- http://otn.oracle.com
- http://www.juliandyke.com
- http://www.hotsos.com

Quellen (2)

Metalink-Notes

- 1031826.6: Histograms: An Overview
- 35934.1: Cost Based Optimizer - Common Misconceptions and Issues
- 68992.1: Predicate Selectivity
- 70075.1: Use of bind variables in queries
- 131272.1: How using synonyms may affect database performance and scalability
- 212809.1: Limitations of the Oracle Cost Based Optimizer
- 261020.1: High Version Count with CURSOR_SHARING = SIMILAR or FORCE
- 296377.1: Handling and resolving unshared cursors/large version_counts

Quellen (3)

Metalink-Notes

- 338113.1: Plans can change despite no stats being regathered
- 377847.1: Unsafe Peeked Bind Variables and Histograms
  387394.1: Query using Bind Variables is suddenly slow
- 401068.1: Possible Poor Runtime Performance for Bind Variables when Compared with Literal Values
- 430208.1: Bind Peeking By Example
- 57309.1: How To Flush an Object out the Library Cache [SGA]
- 604256.1: Why is a Particular Query Slower on One Machine than Another?

Zusammenfassung

- Unsichere Bindevariablen stellen für den Cost Based Optimizer in Verbindung mit Cursor-Sharing nach wie vor ein Problem dar.
- Adaptive Cursor Sharing muß sich im produktiven Einsatz erst noch bewähren.
- Bindevariablen sollten aus Performancegründen wo möglich eingesetzt werden; Literale dort, wo sinnvoll (ungleiche Schlüsselverteilung).
- Der Serverinitialisierungsparameter CURSOR_SHARING sollte grundsätzlich die Default-Ausprägung EXACT haben. Die Nutzung von FORCE und SIMILAR empfiehlt sich nur auf Session- oder Statement-Ebene.

Weitere Fragen?

wilhelm.bresser@hl-services.de
http://www.hl-services.de