Initialisierungsparameter – wieso, weshalb, warum?

Eero Mattila
Quest Software GmbH
Köln

Schlüsselworte
Oracle Initialisierungsparameter, spfile, init.ora

Einleitung

Die Oracle Datenbank kommt mit einer inzwischen schier unüberschaubaren Menge an Initialisierungsparametern daher. Mit jedem Release kommen neue dazu, meist werden gleichzeitig andere wiederum obsolet. Viele davon sind für die allermeisten Instanzen standardmäßig korrekt eingestellt, sodass man sich darüber keine Gedanken zu machen braucht. In manchen Fällen kann aber eine Standardeinstellung eines Parametes verheerende Folgen für die Performance der Datenbank haben, gelegentlich kann eine unglückliche Einstellung unerwartete Folgen haben, oder gar ein falsch gesetzter Parameter das Hochfahren der Datenbank ganz verhindern.

Dieser Vortag gibt vor allem jüngeren Datenbankadministratoren einen Überblick über die Arbeit mit Initialisierungsparametern und die eine oder andere Empfehlung.

Wozu sind Initialisierungsparameter da?

Das Oracle RDMBS verwendet Initialisierungsparameter für verschiedene Zwecke, um z.B. diverse Grenzen für Speicherressourcen, Anzahl von Prozessen, Sessions, regionale Einstellungen wie Sprache, Datums- und Zahlenformate und vieles andere mehr zu setzen. Alle Parameter sind grundsätzlich optional, denn Oracle hat für jeden Parameter einen Standardwert. Diese Parameter können von Datenbankadministratoren verändert werden, um die Leistung der Datenbank zu optimieren.

Die aktuelle Version 12.1.0.2 listet für eine Single Instance Datenbank je nach Betriebssystem etwa 380 verschiedene *dokumentierte* Parameter in der View V$PARAMETER auf. Darüber hinaus gibt es eine Fülle von undokumentierten, sogenannten *hidden* oder auch *underscore* Parameter. Oracle rät dringend davon ab, diese Parameter ohne ausdrückliche Anweisung von Support zu setzen oder zu verändern. Mehr dazu etwas später.

Wo und wie werden Initialisierungsparameter gespeichert?

Bis zur Version Oracle8i wurden die Parameter in einer Textdatei gespeichert, allgemein bekannt als init.ora, üblicherweise init<SID>.ora benannt. Sie wurde nur beim Hochfahren der Instanz gelesen, sodass Änderungen darin einen Neustart der Instanz erforderlich machten. Parameter, die zur Laufzeit der Instanz geändert wurden, mussten manuell in init.ora gepflegt werden, damit die Änderungen auch

nach einem Neustart gültig waren. Da es häufig mehrere Kopien der init.ora-Datei gab, erfordete die Handhabung große Sorgfalt von allen beteiligten DBAs.

Mit Oracle9i wurde ein neues Konzept eingeführt, das sogenannte Server Parameter File oder SPFILE. Wie der Name sagt, liegt das SPFILE immer auf dem Datenbankserver, auf Unix/Linux im Verzeichnis $ORACLE_HOME/dbs, auf Windows im Verzeichnis ORACLE_HOME\database. Laufzeitänderungen von Parametern (per ALTER SYSTEM) können gezielt im SPFILE hinterlegt werden, und Oracle speichert darin auch selbsttätig z.B. aktuelle Größen von SGA-Komponenten – auch dazu später mehr.

Das SPFILE ist eine Binärdatei, das nicht editiert werden kann bzw. darf – manuelle Änderungen darin können eine laufende Instanz zum Absturz bringen und einen Start der Instanz verhindern.

Es ist absolut empfehlenswert, ausschließlich SPFILEs zum Starten der Datenbank zu verwenden. Jedoch ist es nach wie vor möglich – und manchmal auch notwendig – weiterhin PFILEs zu verwenden, zumindest als Sicherheitskopie der SPFILEs, um bei Bedarf ein neues SPFILE aus einem PFILE zu erzeugen.

Aus einem SPFILE erzeugt man ein PFILE mit dem folgenden Befehl (als SYSDBA):

```
SQL> create pfile['/<pfad>/<dateiname>'] from spfile['/<pfad>/<dateiname>'];
```

Falls kein Pfad und Dateiname angegeben wird, erstellt Oracle eine Datei mit dem Namen initSID.ora im Verzeichnis ORACLE_HOME/dbs (Linux/Unix) bzw. ORACLE_HOME/database (Windows).

Umgekehrt wird aus einem PFILE ein SPFILE mit dem folgenden Befehl erzeugt:

```
SQL> create spfile['/<pfad>/<dateiname>'] from pfile['/<pfad>/<dateiname>'];
```

Bei einer heruntergefahrenen Instanz funktioniert beides auch ohne Pfad- und Dateiangabe, ist aber eine laufende Instanz mit einem SPFILE gestartet worden, muss für das neue SPFILE immer ein anderer Zielort definiert werden! Wenn Pfad- und Dateiname weggelassen werden, verwendet Oracle das Standardverzeichnis und den Standardnamen (initSID.ora bzw. spfileSID.ora).

Parameteränderungen im SPFILE können mit zwei Methoden vorgenommen werden. Üblicherweise geschieht dies mit dem Befehl `ALTER SYSTEM SET <parameter>=<wert> SCOPE=SPFILE|BOTH`. Das Argument `SCOPE=SPFILE` ändert den Wert nur im SPFILE, und der neue Wert wird erst einem Neustart wirksam, während `SCOPE=BOTH` den Wert sowohl für die laufende Instanz (sofern für den jeweiligen Parameter erlaubt) als auch im SPFILE festlegt. Die zweite Methode ist, zunächst aus einem SPFILE ein PFILE zu erzeugen (s. oben), dann das PFILE zu editieren und schließlich aus dem editierten PFILE wieder ein neues SPFILE zu generieren.

Möchte man den Wert eines Parameters auf den Standardwert zurücksetzen, muss man folgenden Befehl absetzen:

```
SQL> alter system reset <parameter>;
```

Damit wird der Eintrag für den entsprechenden Parameter aus dem SPFILE gelöscht.

Der aktuelle Wert eines Parameters lässt sich mit dem Befehl `show parameter <parameter>` überprüfen. Lässt man den Parameternamen weg, erhält man eine zwar alphabetisch sortierte, aber eher unübersichtliche Liste. Besser geeignet dafür ist, sich ein PFILE aus der laufenden Instanz generieren zu lassen:

```
SQL> create pfile='dateiname' from memory;
```

Diese lässt sich dann in aller Ruhe mit einem geeigneten Editor betrachten. Diese Ausgabe hat, im Gegensatz zum PFILE, das man aus dem SPFILE generiert, den Vorteil, dass darin sämtliche aktive Parameter aufgelistet werden, d.h. auch alle nicht explizit gesetzten, sondern auch alle systemseitig implizit gesetzten Parameter, inklusive der aktuellen Größen der SGA-Komponenten sowie alle sogenannten „hidden" bzw. „Underscore-Parameter" – s. nächster Abschnitt.

Abhängigkeiten zwischen den Parametern

Einige Parameter erhalten ihre Standardwerte in Abhängigkeit von anderen Parametern. Der Wert von SESSIONS ist zum Beispiel abhängig vom Wert des Parameters PROCESSES. Wird PROCESSES geändert, ändert sich auch der Wert für SESSIONS – es sei denn, es wird ein expliziter Wert für SESSIONS definiert.

Gewisse Parameter sind außerdem abhängig vom verwendeten Betriebssystem. Zum Beispiel die Werte für die Standardgröße und maximale Anzahl von Block Buffern hängt vom Betriebssystem ab.

Zwischen einigen Parametern gibt es Abhängigkeiten, die sogar dazu führen können, dass Parameter bei verschiedenen Konfigurationen unterschiedliche Bedeutungen haben. Ein gutes Beispiel dafür ist die Konfiguration der SGA-Komponenten, die entweder manuell vorgenommen oder aber ganz oder teilweise Oracle überlassen werden kann. Bei der manuellen Konfiguration können alle Komponenten der SGA, also Buffer Cache, Shared Pool, Large Pool, Java Pool, Streams Pool und Result Cache mit entsprechenden Parametern auf konkrete Werte gesetzt werden. Wird aber eine der beiden Möglichkeiten zur automatischen Speicherverwaltung (Automatic Memory Management oder Automatic Shared Memory Management) eingesetzt, wird mit denselben Parametern, sofern sie konfiguriert werden, nunmehr Mindestwerte definiert. Dazu mehr weiter unten.

Was sind „hidden" bzw. „Underscore" Parameter?

Neben den dokumentierten ca. 380 Initialisierungsparametern kennt die Oracle Datenbank eine von Release zu Release wachsende Anzahl – in der aktuellen Version 12.1.0.2 sind es um 3.600 – von sogenannten „versteckten" (*hidden*) Parametern, deren Name mit einem Unterstrich, auf Englisch *Underscore*, beginnt.

Oracle weist ausdrücklich darauf hin, dass diese Parameter ausschließlich auf Aufforderung von Support gesetzt oder angepasst werden dürfen, u.U. droht sogar der Verlust auf den Anspruch auf Support, wenn diese Parameter unaufgefordert verändert werden. Gelegentlich könnte aber ein solcher Parameter den DBA vor einem versehentlichen Lizenzverstoß schützen – hier ein Beispiel:

Mit Oracle Data Guard wird eine physische Standby Datenbank zur aufgesetzt, die im Desasterfall den Betrieb der primären Datenbank übernehmen kann. Data Guard an sich ist im Lizenzumfang der Enterprise Edition enthalten, es müssen lediglich beide Datenbanken lizensiert sein. Seit Oracle11g ist es aber möglich, die Standby Datenbank im Read-Only-Modus zu öffnen, während sie die Redologs

der primären Datenbank entgegennimmt. Dadurch wird aus dem System ein Active Data Guard, was zusätzliche Kosten nach sich zieht. Ein versehentliches `startup open` (statt `startup mount`) genügt, und das Lizenzvergehen ist passiert. Dies kann jedoch mit einem Underscore-Parameter verhindert werden: `_query_on_physical=FALSE` veranlasst die Standby-Datenbank, den Read-Only-Modus zu verweigern. Leider steht dieser Parameter standardmäßig auf TRUE, sodass ein Lizenzverstoß z.B. bei einem Switchover leicht passieren kann. Möchte man sich dagegen absichern und den Parameter auf FALSE setzen, sollte man jedoch sicherheitshalber, wie bei allen Underscore-Parametern, zunächst die Erlaubnis vom Support einholen.

Und was sind „Double Underscore Parameter“?

Sieht man sich ein PFILE an, das man entweder aus dem SPFILE oder aus Memory generiert hat, findet man am Anfang der Datei eine Reihe von Parametern, die mit dem Instanznamen und danach mit zwei anstatt einem Unterstrich beginnen:

```
DOAG.__data_transfer_cache_size=0
DOAG.__db_cache_size=226492416
DOAG.__java_pool_size=4194304
DOAG.__large_pool_size=20971520
DOAG.__pga_aggregate_target=83886080
DOAG.__sga_target=440401920
DOAG.__shared_io_pool_size=12582912
DOAG.__shared_pool_size=167772160
```

Hier handelt es sich bei Verwendung von Automatic Memory Management oder Automatic Shared Memory Management um die aktuellen Werte der entsprechenden Parameter, die in der Instanz aktiv sind. Im Laufe der Zeit können sich die Größen der verschiedenen Speicherbereiche in der SGA und PGA ändern. Netterweise hält Oracle diese Änderungen im SPFILE fest, sodass die Parameter beim Durchstarten der Instanz automatisch wieder auf dieselben Werte gesetzt werden und die Datenbank sofort mit der gewünschten Konfiguration startet und nicht wieder erst lernen muss, was für sie sinnvoll ist. Zusätzlich bietet das dem DBA die Möglichkeit, sich gelegentlich ein Bild über die aktuellen Parameterwerte zu verschaffen und bei Bedarf einzugreifen.

Unzulässige Werte im SPFILE – was tun?

Es kann schon mal vorkommen, dass man, aus welchen Gründen auch immer, einen ungültigen Wert im SPFILE eingetragen hat und die Instanz dies beim Hochfahren moniert:

```
ORA-32004: obsolete or deprecated parameter(s) specified for RDBMS instance.
```

Je nach betreffendem Parameter startet die Datenbank trotzdem, manchmal aber auch nicht. Die Fehlermeldung sagt leider nicht immer, um welchen Parameter es sich handelt. Die exakte Information dazu befindet sich aber in jedem Fall im Alert.log der Instanz, beispielsweise:

```
Deprecated system parameters with specified values:
  sec_case_sensitive_logon
End of deprecated system parameter listing
```

In diesem Fall ist der Grund einfach: Ab Oracle12c sollen Passworte der Datenbankanwender zwischen Groß- und Kleinschreibung unterscheiden, also ist der Parameter

SEC_CASE_SENSITIVE_LOGON abgekündigt worden. Noch kann er gesetzt werden, und mit dem Wert FALSE bewirkt er auch, dass die Passworte nicht auf Groß- bzw. Kleinschreibung geprüft werden, aber die Fehlermeldung sollte dem DBA als Hinweis dienen, dass sich das in künftigen Versionen ändern kann.

Falls aber beispielsweise ein zu hoher Wert für MEMORY_TARGET definiert wurde, ist die Fehlermeldung recht lakonisch und vor allem verwirrend:

```
ORA-00845: MEMORY_TARGET not supported on this system
```

Im Alert.log ist die Auskunft auch in diesem Fall deutlich genauer:

```
WARNING: You are trying to use the MEMORY_TARGET feature. This feature requires the
/dev/shm file system to be mounted for at least 10737418240 bytes. /dev/shm is
either not mounted or is mounted with available space less than this size. Please
fix this so that MEMORY_TARGET can work as expected. Current available is
1333780480 and used is 675536896 bytes. Ensure that the mount point is /dev/shm for
this directory.
```

Hier muss der Wert also korrigiert werden, ehe die Instanz hochgefahren werden kann. Wie oben beschrieben, erzeugt man dafür aus dem SPFILE ein PFILE, korrigiert den Wert im PFILE, generiert daraus wieder ein SPFILE und startet die Instanz.

Was sind die wichtigsten Parameter?

Von den genannten ca. 380 dokumentierten Initialisierungsparametern sind 30 sogenannte *basic* Parameter, mit denen sich jeder DBA vertraut machen sollte. Anstatt sie hier aufzulisten und zu erklären, sei auf die Oracle Dokumentation verwiesen. Die weiteren ca. 350 Parameter dienen speziellen Zwecken, wie beispielsweise Automatic Memory Management, Automatic Storage Management, Auditing und vielen anderen mehr.

Eine Aussage über die „wichtigsten“ Initialisierungsparameter wäre jedoch ein gewagtes Unterfangen – es würde nämlich implizieren, dass es ebenfalls unwichtige Parameter gibt. Fragt man verschiedene Kapazitäten, bekommt man unterschiedliche Antworten, je nach dem speziellen Fokus des Befragten, wie Performance, Sicherheit, Verwaltungsaufwand usw. – oder spezielle Anwendungsgebiete, wie ASM, RAC, DataGuard etc. In diversen Blogs und auf unzähligen Foren und sonstigen Internetseiten findet man Artikel wie „Top 25 Initialization Parameters“ oder „Top 8 init.ora Parameters Affecting Performance“ und ähnliches mehr – und die Meinungen gehen auseinander. Nimmt man dazu die Empfehlungen von Oracle, ist die Verwirrung perfekt.

Zwei Beispiele, eins mit Performance- und Sicherheitsaspekten (CURSOR_SHARING), ein anderes mit Implikationen für die Administration einerseits und Performance andererseits (MEMORY_TARGET).

Der Parameter CURSOR_SHARING bestimmt, wie Oracle mit Statements umgeht, die Bindevariablen oder Literale verwenden. Die möglichen Werte sind ab Oracle12c EXACT (Standardeinstellung) und FORCE – der Wert SIMILAR aus älteren Versionen ist inzwischen obsolet. Der Wert EXACT bewirkt, dass nur textlich identische Statements sich denselben Cursor teilen und somit teures Parsing reduzieren können. Der Wert FORCE dagegen erlaubt der Datenbank, Literale durch automatisch generierte Bindevariablen zu ersetzen. Bei Anwendungen mit sehr vielen Statements, die sich nur durch unterschiedliche Literale unterscheiden (`select spalte from`

`tabelle where spalte = 'A', select spalte from tabelle where spalte = 'B'`) klingt das durchaus verlockend, denn dadurch könnte sich die Performance der Anwendung drastisch erhöhen – und so liest man diese Empfehlung immer wieder in verschiedenen Artikeln. Größer als dieser Vorteil ist jedoch der Umstand, dass dadurch eine enorme Sicherheitslücke eröffnet wird: SQL Injection. Daher sollte CURSOR_SHARING in jedem Fall datenbankweit auf EXACT stehen, und allenfalls auf Session- oder Statement-Ebene (Hint) sollte auf FORCE zurückgegriffen werden, und zwar auch nur solange, bis die Literale durch Bindevariablen beseitigt sind. Mehr dazu findet man u.a. im ausgezeichneten Artikel von Tom Kyte unter http://tkyte.blogspot.de/2012/02/all-about-security-sql-injection.html.

Mit Oracle 11g wurde der Parameter MEMORY_TARGET eingeführt, der das Automatic Memory Management (AMM) aktiviert. Das stellt verheißungsvoll in Aussicht, dass sich der DBA nie wieder um die Konfiguration der verschiedenen Speicherbereiche (SGA und PGA) einer Datenbank kümmern muss – Oracle erledigt alles „automagisch" und immer richtig. Leider ist das nicht unbedingt der Fall: Je nach Anwendungen, die mit der Datenbank arbeiten, kann die Gewichtung der verschiedenen Komponenten extrem ungünstig werden, sodass der PGA deutlich mehr Speicher zugewiesen wird als der SGA, was für die Performance der Datenbank abträglich ist. Um dies zu vermeiden, sollten bei der Verwendung von AMM unbedingt für die wichtigsten SGA-Bereiche *Mindestgrößen* mit entsprechenden Parametern gesetzt werden: zumindest DB_CACHE_SIZE und SHARED_POOL_SIZE, ggf. auch LARGE_POOL_SIZE und JAVA_POOL_SIZE. Viele zuverlässige Kapazitäten raten allerdings generell von der Verwendung von AMM zumindest bei geschäftskritischen Datenbanken ab. Ein Grund dafür ist auch, dass AMM nicht die Hugepages von Linux unterstützt, die bei einer großen SGA erhebliche Performancevorteile bringen können.

Bereits seit Oracle10g steht auch das Automatic Shared Memory Management (ASMM) zur Verfügung. Damit wird die Konfiguration und das Tuning der SGA-Komponenten Buffer Cache, Shared Pool, Large Pool, Java Pool und Streams Pool vereinfacht, aber die PGA ist davon unberührt. ASMM wird mit dem Parameter SGA_TARGET eingeschaltet. Auch hier gilt, dass die Parameter DB_CACHE_SIZE, SHARED_POOL_SIZE usw. Mindestgrößen darstellen und auf sinnvolle Werte gesetzt werden sollten, um eine vernünftige Performance der Datenbank sicherzustellen.

Backup und Recovery von SPFILE

Wie oben beschrieben, ist es sinnvoll, ein PFILE als Sicherheitskopie des SPFILE zu generieren und zu sichern. Sofern man – und das ist dringend zu empfehlen – die Datenbank regelmäßig mit RMAN sichert, kann dabei auch das SPFILE gesichert werden, was aber leider nicht in der Standardkonfiguration von RMAN enthalten ist. Wird die Option `CONFIGURE CONTROLFILE AUTOBACKUP ON` eingeschaltet, sichert RMAN automatisch am Ende eines Backups sowie nach jeder Strukturänderung der Datenbank sowohl das Controlfile als auch das SPFILE. Darüber hinaus kann das SPFILE gezielt mit dem RMAN-Befehl BACKUP SPFILE gesichert werden.

Bei der Wiederherstellung gibt es viele mögliche Szenarien, in Abhängigkeit davon, ob die Instanz noch läuft, ob ein RMAN Catalog verwendet wird, ob automatische Backups konfiguriert sind, auf welchem Medium die Sicherungskopie liegt u.a.m. Daher wird hier ausdrücklich auf die Oracle Dokumentation verwiesen. Im folgenden Beispiel wird davon ausgegangen, dass die Instanz nicht läuft und dass automatische Backups verwendet werden:

```
1) set dbid <DBID>   -- DBID kann aus dem Dateinamen  des Controlfile-Backups ermittelt werden

2) startup nomount
```

```
3) restore spfile from autobackup;

4) shutdown immediate

5) startup
```

Wie immer bei Backup und Recovery, ist es auch beim SPFILE dringend zu empfehlen, vor allem die Wiederherstellung bei verschiedenen Szenarien gründlich zu testen. Was nützt einem ein Backup, wenn man nicht weiß, wie es schnell und sicher wieder hergestellt wird?

Zusammenfassung

Oracle hält für jeden Initialisierungsparameter einen Standardwert bereit. Trotzdem ist es empfehlenswert, sich mit den Parametern auseinanderzusetzen, denn sie können einen erheblichen Einfluss auf die Performance und auch die Sicherheit der Datenbank haben.

Versteckte oder „Underscore“ Parameter können bei richtigem Einsatz Performancesteigerungen herbeiführen oder vor versehentlichen Lizenzvergehen schützen, aber deren Einsatz sollte immer vorab mit Oracle Support abgestimmt werden.

Da fast jede Datenbank und Datenbankanwendung unterschiedlich ist, gibt es leider kein Patentrezept für eine allgemeingültige Konfiguration. Eine solide, optimal auf RMAN basierende Sicherungsstrategie ist auch für die Parameterdatei erforderlich, und die Wiederherstellung muss unbedingt für verschiedene Szenarien getestet und geübt werden.

Quellen:

asktom.oracle.com, Thomas Kyte, Oracle Corporation
carajandb.com, Johannes Ahrends
Oracle Dokumentation (Oracle8i, Oracle9i, Oracle10g, Oracle11g, Oracle12c)

Kontaktadresse:
Eero Mattila
Quest Software GmbH
Im Mediapark 4e
DE-50670 Köln

Telefon: +49 (0) 221-5777 4169
Fax: +49 (0) 221-5777 450
E-Mail eero.mattila@quest.com
Internet: http:// https://www.quest.com/de-de/