Standardisierungen sind die Grundlage für jeden qualitätsgesicherten und kosteneffizienten Betrieb von Oracle-Datenbanken.

Effizienzsteigerung mittels Standardisierung

Christian Wischki und Rainer Hartwig

Egal, wie groß oder klein IT-Abteilungen, die Menge der zu betreibenden Oracle-Datenbanken oder die Zahl der Anwender auch sein mögen, die Definition und Einhaltung von im Configuration Management definierten Standards bietet sowohl für das Unternehmen – in Bezug auf Compliance-Anforderungen unter ITIL und ISO 20000 – und die verantwortlichen Personen als auch für die Entwicklung und den Betrieb der darauf basierenden Applikationen und deren Anwender viele Vorteile. Grundsätzlich ist es dabei im ersten Schritt eher unwichtig, wie genau eine Oracle-Standard-Datenbank aussieht beziehungsweise aussehen soll. Das am Anfang Wichtigste ist, dass ein Standard definiert und vor allem auch überall eingehalten wird. Erst in einem zweiten Schritt sollten dieser dann optimiert und alle entsprechenden Parameter- und Konfigurationseinstellungen vom Configuration Management überprüft sowie, wo es sinnvoll ist, auch weitestgehend standardisiert werden. Dabei spielt es absolut keine Rolle, ob und welches Tool für das Configuration Management zum Einsatz kommt. Es sollte jedoch stets für die jeweiligen Anforderungen des Unternehmens passend und praktikabel sein.

Ist dieser Standard festgelegt und dokumentiert, können auch schon entsprechende Quick-Wins realisiert werden, wie beispielsweise, dass beim Neuaufsetzen eines jeden weiteren Datenbankservers erheblich weniger Dokumentationsaufwand anfällt und eine potenzielle Fehlersuche oder Optimierungsmaßnahmen zielführender und schneller durchgeführt werden können. Zudem wissen dann auch alle DBAs, etwa aufgrund geeigneter Namensgebungen, sofort darüber Bescheid, ob die Oracle-Datenbanken produktiv oder nur Test-Datenbanken sind, womit auch die Einarbeitungszeit für neue DBAs deutlich kürzer und effektiver ist. Weitere nicht zu unterschätzende Vorteile sind unter anderem, dass auszurollende Skripte und Programme nur noch in einer entsprechenden Referenzumgebung getestet werden müssen und der Aufwand bei der Administration signifikant reduziert werden kann.

In einem dritten Schritt sollten dann aus dem Configuration Management auch die stets vollständigen, richtigen und sowohl die aktuellen als auch die historischen Daten (wie Parameter- und Konfigurations-Einstellungen) zur Verfügung gestellt werden – egal, wie klein oder groß die Oracle-Datenbank-Landschaft ist. Diese Informationen sind beispielsweise für Analysen bezüglich der Auswirkungen und Abhängigkeiten von Änderungen sowie für Fehler- und Ursachen-Analysen zwingend erforderlich. Werden vom Configuration Management hierfür falsche Daten zur Verfügung gestellt, liefert eine darauf basierende Analyse sicherlich ebenfalls falsche Ergebnisse. Zusätzlich hierzu sollte das Configuration Management auch immer aktuelle Antworten auf Konfigurationsfragen beantworten können, wie:

- Welche Datenbanken entsprechen nicht den vom Configuration Management vorgegebenen Baselines?
- Welche Konfigurations- und Parameter-Einstellungen waren in der Vergangenheit für welche Datenbank gesetzt?
- Welche Konfigurations- und Parameter-Einstellungen haben sich wie und wann verändert?
- Welche Konfigurations- und Parameter-Einstellungen welcher Datenbanken verändern sich häufiger?
- Wie ist der Trend bezüglich der Veränderungsrate bei den Konfigurations- und Parameter-Einstellungen?

Die Configuration Policy für eine Oracle-Datenbank

Einer der wichtigsten Bestandteile einer Configuration Policy für Oracle-Datenbank-Services stellt die Vorgabe der Server-Auswahl und die des Betriebssystems (OS) dar. Grundsätzlich kann man Oracle-Datenbanken auf fast jedem OS und Server-System installieren und betreiben. Doch führt sowohl eine zu breite Technologie-Diversifikation als auch hier vielleicht unpassende Technologie (neben zu vielen Datenbank-Versionen) zu höheren Aufwänden im Betrieb. Dann muss einerseits das entsprechende Wissen in der definierten Technologie-Breite vorgehalten werden und man erhält andererseits aufgrund einer suboptimalen OS- und Serversystem-Technologieauswahl beziehungsweise -Definition eine im Betrieb instabile Oracle-Datenbank, was einen schlechteren und/oder ineffizienteren Datenbank-Service zur Folge hat.

Nachdem innerhalb der Configuration Policy die unter dem Datenbank-Service notwendigen Komponenten wie OS und Server definiert sind, muss man sich über die Installationsprozeduren und Verzeichnisstrukturen der Datenbanken Gedanken machen. Eine Vereinheitlichung dieser hat unter anderem auch den Vorteil, dass dadurch alle Datenbanken identisch aufgesetzt

sind, was in der Folge auch eine Optimierung der Aufwände im administrativen Bereich bedeutet.

Hierfür hat sich beispielsweise die Oracle Flexible Architecture (OFA) als Best Practice etabliert. OFA ist ein Standard, der von der Oracle Special Performance Group entwickelt wurde und eine allgemeine Richtlinie für die Benennung von Dateien und Verzeichnissen für Oracle-Installationen darstellt. Das Ziel hierbei ist einerseits die Erleichterung von Verwaltungsaufgaben – vor allem, wenn mehrere Oracle-Datenbank-Installationen auf demselben System betrieben werden, oder auch bei mehreren Systemen, die auf verschiedenen Servern laufen, – und andererseits, die Vereinheitlichung von Oracle-Installations- beziehungsweise Datei- und Verzeichnisstrukturen sicherzustellen.

Ein weiterer zwingender Bestandteil einer jeden Configuration Policy ist die Beschreibung der Backup-Mechanismen der Datenbank-Server. In diesem Bereich gibt es in der Praxis verschiedene Methoden – Offline- oder Online-Backups, Backup to Tape, Backup to Disk, Backups durch selbstentwickelte oder gekaufte RMAN-Skripte oder mithilfe von Backup-Agenten diverser Hersteller. Es existiert nicht immer eine einzige richtige Lösung, doch sollte man auch hier nicht verschiedene Backup-Techniken einsetzen, sondern sich für eine passende Lösung entscheiden, diese innerhalb der Configuration Policy definieren und dann auch konsequent verwenden.

Weitere Bestandteile einer Configuration Policy können auch vorgegebene Sollwerte für beispielsweise informationssicherheitsrelevante Konfigurations- und Parameter-Einstellungen der Oracle-Datenbanken darstellen – oder auch allgemeine Best-Practice-Vorgaben für andere Konfigurations- und Parameter-Einstellungen. In der Praxis sollte man hier jedoch zwischen „shall“ und „should“ unterscheiden. Bei den „Shall-Vorgaben“ seitens des Configuration Managements sollte ein Abweichen nicht möglich sein. Bei den „Should-Vorgaben“ sollte man im Bedarfsfall – nach Freigabe durch das Configuration Management – davon abweichen können.

Einen Standard in der Oracle-Datenbank-Praxis ausrollen

Erfahrungsgemäß stößt man in vielen Datenbank-Abteilungen schon nach kurzer Zeit auf folgende Fragen: „Die Theorie dazu ist gut und schön, aber wie kann das Ganze auch in der Praxis skalierbar realisiert werden und wie schafft man es, dass auch wirklich alle Server diesem Standard genügen?“

Der in der Praxis am meisten begangene Fehler ist, dass man gerne gleich eine allumfassende und für jede kommende Eventualität gerüstete Lösung sucht und implementieren will, obwohl man im Grunde noch gar nicht wirklich weiß, wie viel Configuration Management man für den jeweiligen Datenbank-Betrieb wirklich braucht. Solche Big-Bang-Lösungsansätze kosten sehr viel Geld und haben leider keinen guten ROI. Nach Erfahrungen des Autors lautet ein wesentlich besserer Ansatz hierbei: „Think big, but start small.“

Leider gibt es aber für das Configuration Management nicht wirklich die eine für alle Oracle-Datenbanken und jedes Unternehmen passende und einfach zu replizierende Lösung. Jedoch kann man hierfür durchaus Best-Practice-Lösungen als eine Art Framework und Grundlage verwenden – man darf dabei nur nie vergessen, diese auch zwingend immer auf die entsprechenden individuellen Anforderungen und Gegebenheiten anzupassen und zu ergänzen. Es empfiehlt sich hierzu folgendes Framework als Vorgehensweise:

- Festlegen/Festschreiben eines Standards
- Evaluieren des Standards durch
  - Aufbau eines neuen Datenbank-Servers – entsprechend Punkt 1
  - Aufbau neuer (leerer) Datenbanken – entsprechend Punkt 1
  - Kopie von bestehenden Test-Datenbanken auf diesen Server und Migration entsprechend dem neuen Standard nach Punkt 1
  - Sollte in diesen Schritten ein Problem auftreten, sind der Standard entsprechend anzupassen und die genannten Schritte zu wiederholen
- Ab jetzt gilt der Standard für alle neuen Datenbank-Server und alle neuen Datenbanken
- Dann werden alle Test-Datenbanken schrittweise hinsichtlich des neuen Standards migriert – sofern notwendig
- Im nächsten Schritt betrifft es alle QA-Datenbanken
- Danach werden in Absprache mit den Anwendungen möglichst eng verbunden zuerst die INT-Datenbanken und nach Erfolg auch die zugehörigen PROD-Datenbanken migriert

Definition eines Standards beziehungsweise einer Configuration Policy für Oracle-Datenbanken

Die wichtigste Frage ist: „Wie genau definiere ich einen solchen Standard, und muss dieser sehr detailliert sein oder nur ganz grob und dafür möglichst schnell?“ Wie bereits beschrieben, ist es sehr zielführend, möglichst schnell zu einem definierten Standard zu kommen (Schritt 1). Im Rahmen von Schritt 2, der Evaluierung dieses Standards, wird dieser dann auf seine Praxistauglichkeit getestet, optimiert und ergänzt. Eine sehr gute Basis hierbei ist die Verwendung der von Oracle empfohlenen Standards (wie OFA) und der Konfiguration der Server gemäß den Installation- und User Guides (IUG) und My-Oracle-Support–Notes.

Das in diesem Artikel weiter Folgende stellt keineswegs eine starre Vorgabe dar und ist auch nicht die für alle Unternehmen passende, finale Vorgehensweise und Lösung – es ist aber eine gute Option und Orientierungshilfe. Falls jedoch im Unternehmen bereits ein anderer funktionierender Standard etabliert ist, der auch ähnliche Vorteile aufweist, wäre die Einführung eines jetzt wieder komplett neuen Standards (vor allem in Bezug auf das Kosten-/Nutzen-Verhältnis) sicherlich nicht wirklich empfehlenswert. Aber auch in diesem Fall kann das beschriebene Standardisierungs-Vorgehen sehr gut verwendet werden, um den eigenen, bereits definierten Standard gegen diesen zu überprüfen:

- *Datenbank-Server*
  - Für Produktionsserver sollten ausschließlich von Oracle zertifizierte Umgebungen (Hardware) eingesetzt werden

- Auf einer Produktionsumgebung laufen nur produktive Datenbanken. Das gilt auch für virtualisierte Umgebungen. Integrations-, Test- und QA-Datenbanken können dagegen untereinander gemischt werden
- Datenbanken mit unterschiedlichen SLAs sollten nicht gemischt werden. So sollten auf einer Plattform nur eine oder mehrere RAC-Umgebungen laufen, nicht auch noch Single-Datenbanken
- Nach Möglichkeit sollte die Anzahl verschiedener OS und OS-Versionen für Datenbank-Server minimiert werden, um Shell-Skripte zu vereinfachen etc.
- Oracle Enterprise Linux ist ausgereift und wird ebenfalls durch Oracle direkt unterstützt. Damit gibt es für OS und Datenbank nur einen Lieferanten und einen Support
- Optimal Flexible Architecture (OFA, definiert von Oracle) ist eine sehr gute Basis für ein File- und Verzeichnissystem. Unter „/u0x/app/oracle/rdbms/11.2.0.3.0“ befinden sich die verschiedenen ORACLE_HOMEs, unter „/u00/app/oracle/grid/11.2.0.3.0“ liegt das GRID, unter „/u00/app/oracle/agent/11.2.0.3.0“ die Software für den EM-Agent und unter „/uxy (x>1, y>=0)“ sind dann die Datenbank-Dateien der x-ten Datenbank zu finden
- Sollte auch nur eine Datenbank im Unternehmen ASM nutzen, so ist der grundsätzliche Einsatz von ASM für alle Datenbanken empfohlen
- Die OS-Parameter sollten immer entsprechend den aktuellen Notes in My Oracle Support gesetzt werden. Die IUG sind dazu nicht immer aktuell
- Oracle-Binaries wie auch das DIAG-Verzeichnis sollten immer auf lokalen Platten liegen, jedoch in getrennten Filesystemen
- Alle Konfigurationsdateien (SPFILE, orapwd, sqlnet.ora, listener.ora, tnsnames.ora etc.) liegen in einem separaten Filesystem auf lokalen Platten außerhalb der Oracle-Binaries. Über symbolische Links werden diese im ursprünglichen Verzeichnis zur Verfügung gestellt
- Wann immer möglich, sollte man ASM oder RAID 10 im Storage nutzen
- Alle Server eines Unternehmens sollten dieselbe Uhrzeit haben, unabhängig von ihrem lokalen Rechenzentrum, um so besser LOG-Files auf unterschiedlichen Servern vergleichen zu können

- *Oracle-Datenbank*
  - SID = Instance-Name, alles in Großbuchstaben, Präfix = „P“, „I“, „T“ oder „Q“ – je nach Art und Zweck der Datenbank
  - Es empfiehlt sich die Nutzung der Flash Recovery Area
  - Sinnvoll ist eine einheitliche Filegröße für alle Datendateien
  - Bei allen Datendateien sollte „autoextend = OFF“ gesetzt sein
  - Alle unnötigen oder Default-Parameter aus dem SPFILE entfernen
  - Nutzung von Automatic Segment Space Management (ASSM)
  - Aktivieren des Automatic Shared Memory Management (ASMM) und Automatic Memory Management (AMM); verschiedene Bereiche (buffer_cache, shared_pool etc.) sollten jedoch so gesetzt sein, dass die Performance des Servers gut ist
  - Aktivieren des Flashback-Modus, sofern der Performance-Impact nicht zu groß ist
  - OMF sollte nur in Testumgebungen eingesetzt werden
  - Aus lizenztechnischen Gründen nur diejenigen Optionen installieren, die wirklich auch genutzt werden
  - Nutzung von WAR-Reports (sofern Lizenzen vorhanden) oder STATSPACK-Reports. Das Intervall sollte auf 15 Minuten und der Zeitraum auf 33 Tage gesetzt sein, um so einen besseren Vergleich zu haben
  - Erzeugen der SYSTEM-Statistiken
  - Vermeiden des Setzens/Änderns der „OPTIMIZER_INDEX*“-Parameter – wenn irgend möglich
- *RAC*
  - Von der Nutzung des ASM Cluster File Systems (ACFS) ist abzuraten
  - „ASM_PREFERED_READ_FAILURE_GROUPS” im SPFILE setzen
  - Der SCAN-Listener sollte genutzt werden
  - SCAN-Listener-Name = Cluster-Name
  - Die normalen Listener und der Interconnect haben je ein privates Netzwerk
  - „LOCAL_LISTENER“ und „REMOTE_LISTENER“ sollten gesetzt sein
- *ASM*
  - Explizite Angabe von Failure Groups
  - „Normal Redundancy“ innerhalb von ASM verwenden
- *SQL*NET*
  - „DB_DOMAIN“ im SPFILE setzen
  - „GLOBAL_NAMES=TRUE“ im SPFILE setzen
  - Die Verbindung zur Datenbank nur via „SERVICE_NAME“ und nicht mehr über die Oracle System ID (SID) herstellen
  - Es empfiehlt sich nur ein Listener pro Datenbank-Server und pro Oracle-Version
  - Nach Möglichkeit keine statischen Einträge in „listener.ora“
- *Patching/Upgrade/Migration/Deployment*
  - Es empfiehlt sich der Einsatz von „guaranteed restore points“, um möglichst schnell den Ursprungszustand beim Patchen, Upgrade, Migration oder nach einem Deployment wiederherstellen zu können.
- *Täglich anfallende Arbeiten*
  - Grundsätzlich sollte so viel wie möglich automatisiert werden, um manuelle, individuelle Fehler zu vermeiden und um den DBA zu entlasten
  - Index-Reorganisation kann sinnvoll sein, wenn mehr als 25 Prozent der Zeilen der Tabelle gelöscht wurden. Eine grundsätzliche, regelmäßige Index-Reorganisation ergibt keinen Sinn – lediglich in besonderen, begründeten Fällen
  - Eine Reorganisation von Tabellen ist nur dann empfehlenswert, wenn die Anzahl der „migrated rows“ signifikant ist und durch die Anwendung auf diese noch zugegriffen wird

  - Es empfiehlt sich, den DBA_SCHEDULER insbesondere für Datenbank-interne Jobs zu nutzen
- *Monitoring*
  - Zur Überwachung aller Datenbanken ist der Einsatz von nur einem Tool empfohlen
  - Bei allen neu zu schreibenden oder anzupassenden Skripten, die auf das Alert-File zugreifen, sollte „alert.xml“ ausgelesen werden, da ab Version 13 „alert.log“ wahrscheinlich nicht mehr unterstützt wird
- *Dokumentation*
  Jeder Datenbank-Server und jede Datenbank ist ausführlich zu dokumentieren. Ganz besonders wichtig sind dabei folgende Aspekte:
  - Genaue Version des OS und aller eingespielten Patches
  - Hauptverantwortlicher System-Admin und Stellvertreter
  - Hauptverantwortlicher Netzwerk-Admin und Stellvertreter
  - Hauptverantwortlicher Storage-Admin und Stellvertreter
  - Hauptverantwortlicher DBA und Stellvertreter
  - Mögliche Wartungsfenster
  - SLA
  - Wann findet ein Backup statt? Welche Art von Backup?
  - Sonstige Besonderheiten, wie und warum sich diese DB vom Standard abhebt
  - Genaue Version der Datenbank und aller eingespielten Patches
  - Name der Instance und Name der Datenbank
  - Welche Non-Default-Parameter wurden warum gesetzt?
  - Konfiguration von SQL*NET (auf dem Datenbank-Server)
  - Welche Optionen sind installiert?
  - Welche Optionen werden auch genutzt?
  - Wann werden Daten und in welchem Umfang in die DB geladen? Von wo?
  - Wann und in welchem Umfang werden Daten zu anderen DBs verschoben? Und wie?
  - Wann und in welchem Umfang werden Daten im Filesystem gespeichert? Und wohin?

```
Statement s = null;
Connection con = null;

String MyName = "PROGRAMNAME";
String nameForConnect = userId;
String HostName = null;
String Terminal = null;

try {
      HostName = InetAddress.getLocalHost().getCanonicalHostName();
} catch (Exception e) {
}

Terminal = HostName;
int i = HostName.indexOf(".");
if(-1 != i) {
      Terminal = HostName.substring(0, i);
}

String OsUser = System.getProperty("user.name");

OperatingSystemMXBean mxBean = (com.sun.management.OperatingSystemM
XBean) ManagementFactory
                              .getOperatingSystemMXBean();
String ProcessName = "1234@" + mxBean.getName();

if (nameForConnect.equalsIgnoreCase("SYS")) {
      nameForConnect = nameForConnect + " as sysdba";
}

try {
      Properties pro = new java.util.Properties();

      pro.setProperty("password", passWord);
      pro.setProperty("user", nameForConnect);

      try {
              pro.put("ApplicationName", MyName);
              pro.put("ClientHostname", HostName);
              pro.put("ClientUser", OsUser);

              pro.put("v$session.osuser", OsUser);
              pro.put("v$session.machine", HostName);
              pro.put("v$session.program", MyName + ".jar");
              pro.put("v$session.process", ProcessName);
              pro.put("v$session.terminal", Terminal);
      } catch (Exception ignore) {

      }

      DriverManager.registerDriver (new oracle.jdbc.OracleDriver());
      Class.forName(getDriverName()).newInstance();
      con = DriverManager.getConnection(getUrl(), pro);

      s = con.createStatement();
      s.setFetchSize(getFetchedRows());

      return s;

} catch (Exception ignore) {
}
```

Listing 1

  - Welche Anwendungen greifen wie und von wo auf diese DB zu?
  - Wer sind Ansprechpartner und Stellvertreter der jeweiligen Anwendung?
  - Gibt es Abläufe in der Anwendung, die in einem bestimmten Zeitfenster fertig sein müssen? Wie kann man dies überprüfen?
  - Wie ist die Userverwaltung organisiert?
  - Welche Tablespaces werden von dieser Anwendung genutzt?
  - Mit welchem Datenwachstum muss man pro Anwendung für diese Datenbank innerhalb von sechs Monaten rechnen?
- *JDBC-Connections*
  - Um beim Aufbau einer JDBC-Verbindung zur Datenbank möglichst viele Informationen mitzugeben, empfiehlt es sich, ein Java-Codefragment zu implementieren (siehe Listing 1).

Definition der Datenbank- und Datenbankserver-Parameter

Um die notwendigen Attribute der CIs für Datenbank-Services zu definieren, gibt es zwei praktikable Ansätze:

- *Methode 1*
  Es wird vorab ein Set von wahrscheinlich beziehungsweise in der Regel für alle Datenbanken relevanten Parameter- und Konfigurations-Einstellungen definiert, das zu Beginn für jedes CI (in diesem Fall die entsprechende Oracle-Datenbank) am besten in einer Configuration Management Database (CMDB) erfasst, permanent überprüft und bei Bedarf pro CI individuell mit weiteren Parameter- und Konfigurations-Einstellungen erweitert wird. Dafür haben sich in der Praxis bestimmte Parameter- und Konfigurations-Einstellungen als eine Art „Initial-Set" bewährt (siehe Tabellen unter www.doag.org/go/doagnews/wischki_tabelle)
- *Methode 2*
  Es werden am Anfang alle Parameter- und Konfigurations-Einstellungen für jedes CI (in diesem Fall die jeweilige Oracle-Datenbank) in der CMDB erfasst und permanent überprüft. Aus diesem Set werden dann die jeweils nicht relevanten Parameter- und Konfigurations-Einstellungen sukzessive ausgegliedert, sodass schlussendlich die wirklich relevanten übrig bleiben.

Welche Methode wann zu verwenden ist, hängt zumeist vom Wissensgrad hinsichtlich der für die CI relevanten Parameter- und Konfigurations-Einstellungen ab. Sind sie weitgehend bekannt, sollte man Methode 1 verwenden, wenn nicht, Methode 2 bevorzugen.

Fazit

Für ein erfolgreiches Configuration Management im Oracle-Datenbank-Bereich gilt immer: Das richtige Maß und Ziel ist entscheidend! Wie bereits beschrieben, gibt es kein für alle Datenbank-Landschaften passendes Configuration Management, da hier immer individuelle Komponenten wie die Menge und Komplexität der Datenbanken sowie auch die hierauf basierenden Applikationen einen signifikanten Einfluss haben.

Kein oder zu wenig Configuration Management haben eine unnötig hohe Total Cost of Ownership (TCO) für den Datenbank-Betrieb zur Folge – zu viel Configuration Management aber auch. In jedem Fall sollte man mit dem Configuration Management immer tendenziell kleiner anfangen und dieses dann nach Bedarf erweitern – aber auch stets die Skalierbarkeit im Auge behalten. Der umgekehrte Weg, gleich den „Big Bang" zu realisieren und zu implementieren, sowie das von einigen Lösungsanbietern versprochene „quick and easy" oder gar „out of the box" ist in der Regel immer der teurere Weg.

Christian Wischki
cw@christian-wischki.com

Rainer Hartwig
rainer.hartwig@mt-ag.com