Das ist Database-Cloning

Johannes Ahrends, CarajanDB GmbH

Viele von uns sind mit der Filmserie „Star Wars“ aufgewachsen und haben schon diverse Schlachten mit Clone-Kriegern von Lego geschlagen. Das Schaf „Dolly“ hat bereits in den 1990er Jahren für Aufregung gesorgt und die laut Wikipedia „durch ungeschlechtliche Vermehrung entstandene Nachkommenschaft eines Lebewesens“ ist zumindest für Menschen in Deutschland verboten. Der Hintergedanke beim Klonen ist jedes Mal derselbe: Es sollen eine oder mehrere Kopien einer Sache hergestellt werden, die die gleichen Eigenschaften wie das Original besitzen. Das gilt auch für Datenbanken und ist prinzipiell nichts Neues. Wenn es da nicht zwei wichtige Punkte gäbe: Größe und Anzahl von Datenbanken nehmen immer mehr zu.

Kopien von Datenbanken gibt es, seit diese existieren, sei es per RMAN Duplicate, Standby Database oder auf Storage-Ebene über Mirroring. Das Verfahren ist einfach und schnell und wird seit vielen Jahren erfolgreich genutzt. Allerdings kommen wir mehr und mehr an die Grenzen dieser Verfahren, denn es ist nicht selten, dass 20 oder mehr Kopien einer Datenbank für Test und Entwicklung zur Verfügung stehen müssen. Bei Datenbanken, die mehrere Terabyte groß sind, bedeutet dies, dass Unsummen für Plattenspeicher anfallen.

Aber werden diese Datenbanken überhaupt genutzt? In vielen Fällen benötigen die Entwicklungsteams nur einen Bruchteil der Datenbank; manchmal werden zwar viele Daten gelesen, jedoch nur sehr wenige für die neue Entwicklung geändert. Das bedeutet, dass in der Regel 90 Prozent der kopierten Daten gar nicht erforderlich sind und weniger als 5 Prozent tatsächlich geändert werden. Außerdem müssen Datenbanken immer schneller provisioniert beziehungsweise aktualisiert werden. Der Stand der Produktion von vor einem Monat ist für die Entwicklung eventuell noch akzeptabel, für einen Qualitätstest allerdings schon nicht mehr. Das stellt uns vor folgende Herausforderungen:

- Viele Datenbanken aufzubauen, die wenig Speicherplatz benötigen
- Datenbanken schnell zur Verfügung zu stellen
- Möglichst den DBA von dieser Aufgabe zu entlasten, also den Verantwortlichen selbst die Möglichkeit zu geben, solche Datenbanken zu erstellen,

Auch wenn Oracle mit RMAN Duplicate einen einfach zu bedienenden Befehl für das Klonen von Datenbanken zur Verfügung stellt, reicht das für diese Anforderungen nicht aus:

- Es werden zu viele/alle Daten kopiert
- Bei einer Datenbankgröße von einigen Terabyte dauert das etliche Stunden
- Das ist für den Anwender unzumutbar

Als Alternative bietet sich zunächst Data Pump an, das einige der genannten Anforderungen erfüllt:

- Es können Teilmengen übertragen und die gleichen Exports für mehrere Fachbereiche genutzt werden
- Die Teilmengen reduzieren die Dauer für die Erstellung

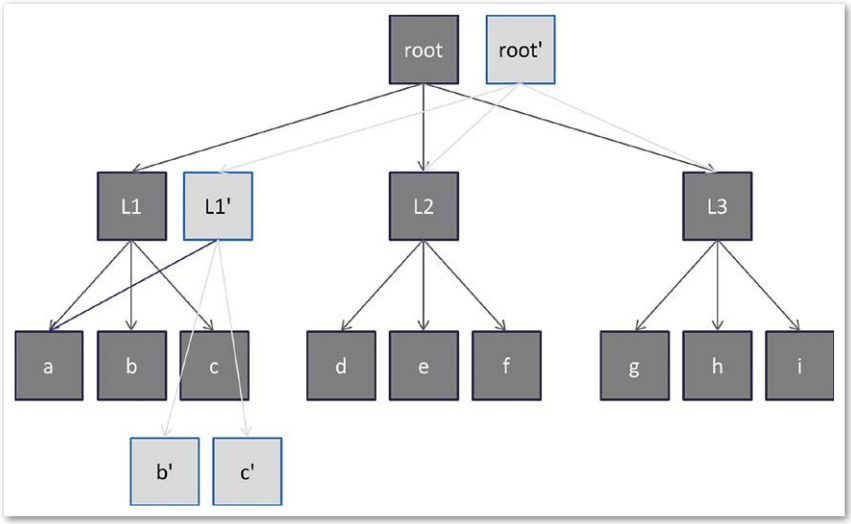

Abbildung 1: Schema des „Copy On Write“

- Wenn schon eine Datenbank existiert, kann der Ex-/Import per Skript erfolgen, sodass sich die Fachabteilung selbst bedienen kann

Das Verfahren wird heute in vielen Unternehmen für den Aufbau von Entwicklungsdatenbanken benutzt, hat aber folgende Einschränkungen:

- Es ist schwierig, die Teilmengen klar zu definieren. Dadurch wächst die Gefahr der Inkonsistenz.
- Auch wenn 90 Prozent der Daten nicht benötigt werden. Wer sagt, welche das sind?
- Die Teilmenge eignet sich nicht für Qualitätstests beziehungsweise Performance-Analysen.
- Der DBA muss zumindest die Datenbank vor dem Import erstellen und dafür sorgen, dass die alten Daten gelöscht sind. Außerdem ist er für den korrekten Export verantwortlich.

Snapshots und Copy-On-Write

Beim Namen „Snapshot“ denkt man oft an Virtualisierungslösungen wie VMware. Dabei wird zu einem bestimmten Zeitpunkt ein Snapshot erstellt, um bei Bedarf das Gastsystem auf diesen Snapshot zurückzusetzen. VMware verwendet dafür ein „Copy On Write“, es werden also für eine bestehende virtuelle Disk (zum Beispiel „disk1.vmdk“) eine neue Version (zum Beispiel „disk1-0001.vmdk“) angelegt und alle Änderungen in dieser Datei gespeichert. Jetzt kann der Gast jederzeit auf den alten Stand zurückgesetzt werden oder der alte Stand wird durch das Löschen des Snapshots auf den aktuellen Stand vorgerollt.

Ähnlich verhält es sich mit Oracle ZFS und „btrfs“-Filesystemen. Allerdings steht dabei nicht die Möglichkeit des Zurücksetzens im Vordergrund, sondern es geht darum, eine gemeinsame Basis der Daten zu haben, auf der beliebig viele Snapshots aufsetzen können. Es gibt also eine Quelle (das Original), auf die lesend zugegriffen wird (entsprechend „disk1.vmdk“ bei VMware), und mehrere unabhängig existierende Dateien, in die die Änderungen geschrieben werden. Auf die Oracle-Datenbank übertragen würde das bedeuten, dass die Datafiles in einen lesenden sowie einen schreibenden Teil aufgeteilt sind und das Filesystem die I/Os entsprechend verteilt. Es gibt unterschiedliche Methoden, wie ein solches Filesystem aufgebaut sein kann, wobei es sich immer um eine Art Indizierung der Blöcke (beispielsweise Filesystem- oder Oracle-Block) handelt *(siehe Abbildung 1)*.

Es wird ähnlich einem „B*Tree“-Index zunächst eine Struktur angelegt, in der die Blöcke referenziert sind. Alle Anwendungen lesen, ausgehend vom „Root“-Block über die Verzweigungen (L1 … L3) die Blätter (a … i), in denen die Zeiger zu den tatsächlichen Datenblöcken stehen. Wenn man jetzt Datenblöcke ändert, entsteht ein neuer Stamm („Root“-Verzweigungen), der auf diese sowie die entsprechenden nicht geänderten Blöcke verweist. Eine Anwendung, die also einen neueren Zustand der Blöcke (Snapshot) benötigt, greift jetzt auf „Root“‘ zu und über L1‘, L2 und L3 auf a, b‘, c‘, d … i.

Copy-On-Write für Datenbanken

Um dieses Verfahren für die Oracle-Datenbank zu nutzen, wird man zunächst sicherlich nicht die Produktionsdatenbank als Quelle nutzen; als Erstes wird also eine konsistente Kopie der Datenbank benötigt. Es liegt nahe, mit RMAN ein Backup

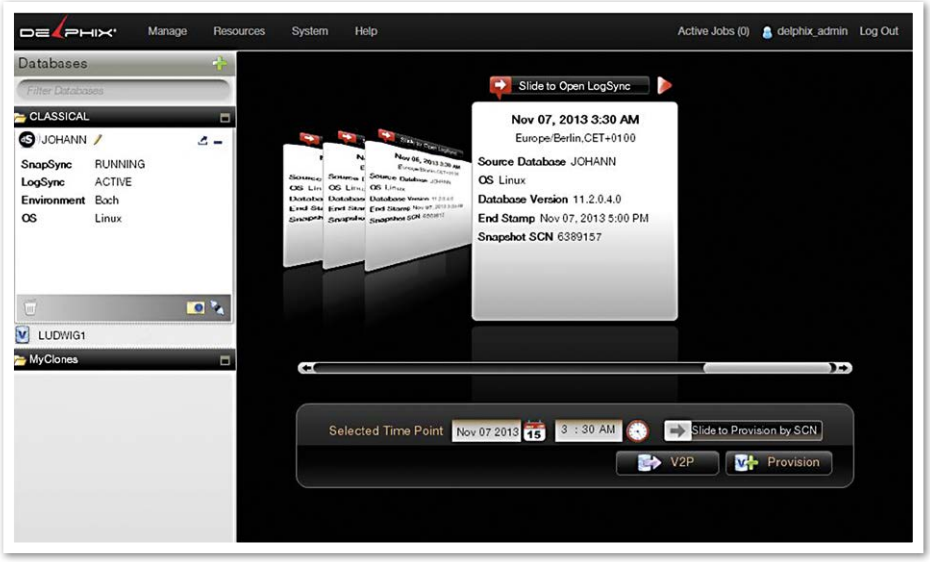

Abbildung 2: Delphix in Aktion

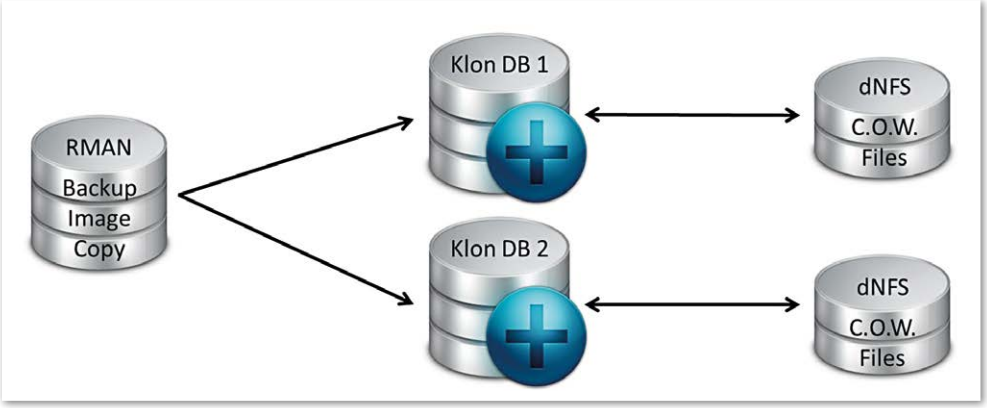

Abbildung 3: Das Prinzip von CloneDB

durchzuführen. Aus diesem konnte man bisher auch mit dem Befehl „DUPLICATE DATABASE" beliebig viele Datenbanken erstellen – aber das war nicht das Ziel: Das Backup selbst sollte die Datenbank sein. Denn nur dann lassen sich der Speicherverbrauch reduzieren und neue Datenbanken schnell aufbauen.

Die Frage ist also: „Wie stelle ich das Backup gleichzeitig mehreren Datenbanken beziehungsweise Servern zur Verfügung?" Egal ob Unix oder Linux, die Lösung ist bekannt und heißt „NFS". In einem derzeit noch andauernden Proof of Concept hatte der Autor die Möglichkeit, zwei Lösungen näher unter die Lupe zu nehmen: Delphix und CloneDB. Die Multitenant-Option von Oracle 12*c* bietet noch eine weitere Lösung, die nicht auf RMAN basiert. Dazu später mehr.

Delphix

In den USA hat sich seit einigen Jahren das Unternehmen „Delphix" mit der gleichnamigen Software einen Namen gemacht *(siehe Abbildung 2)*. Das Unternehmen bietet Copy-On-Write unter anderem auch für Oracle-Datenbanken an. Als Quelle beziehungsweise Startpunkt dienen ein normales RMAN-Backup sowie die archivierten Redolog-Dateien. Für die Clones steht ein spezielles Filesystem (DxFS) zur Verfügung, in dem die Daten-, Redolog-, Temp- und archivierten Redolog-Dateien liegen. Beim Starten der Datenbank-Instanz werden diese Filesysteme eingebunden und die Dateien als normale Datenbank-Dateien sichtbar. Das Copy-On-Write wird intern geregelt, man bekommt also von der ganzen Snapshot-Technologie nichts mit.

Die Technologie erweist sich als ausgereift, und wenn man die Delphix-Appliance einmal installiert hat, ist der Aufbau ausgesprochen einfach. Über eine grafische Oberfläche lassen sich sehr schnell auch von Nicht-DBAs (fast) beliebig viele Klone erstellen, die dann automatisch als Oracle-Datenbanken gestartet werden. Vorteil von Delphix ist, dass zusätzlich zum RMAN-Backup in Intervallen Snapshots der Quelldatenbank erstellt werden, sodass es möglich ist, einen Klon zu jedem beliebigen Zeitpunkt der Quelldatenbank aufzubauen und dann vor- oder zurückzurollen.

CloneDB

Oracle hat mit der Version 11.2.0.2 relativ unauffällig das Feature „CloneDB" auf den Markt gebracht. Auch dieses basiert auf einem RMAN-Backup (in diesem Fall allerdings als „Datafile Copy") und erlaubt es, eine beliebige Anzahl von Datenbanken auf dieser einen Kopie aufzubauen. Die geänderten Daten werden ebenfalls in entsprechende Datafiles geschrieben. Einzige Voraussetzung ist die Nutzung von Direct NFS (dNFS) für die „Client"-Datenbanken *(siehe Abbildung 3)*.

Zwar war die Dokumentation zu Beginn des Tests (Ende 2013) noch sehr übersichtlich; wenn man sich allerdings einmal intensiver mit dem Aufbau von Direct NFS beschäftigt hat, lassen sich Snapshots innerhalb weniger Minuten erstellen. Im Gegensatz zu Delphix muss man sich hierbei aber um das Starten beziehungsweise die Einstellung der Snapshot-Datenbank selbst kümmern.

Oracle 12*c*

Natürlich bleibt Oracle an dieser Stelle nicht stehen. Die Multitenant-Option der Datenbank 12*c* bietet jetzt zusätzlich die Möglichkeit eines „Snapshot Clone" für Pluggable Databases. Anstelle der RMAN-Kopie fungiert hier eine bereits existierende Pluggable Database als Quelle beziehungsweise lesende Version der Datenbank und dementsprechend können bis zu 251 weitere Snapshot-Klone angelegt werden. Voraussetzung hierbei ist die Verwendung eines Filesystems, das Snapshot Copy unterstützt (entweder vom Storage-Hersteller, etwa ZFS, oder ACFS). Diese Methode ist sicherlich von den vorgestellten Verfahren die unkomplizierteste, da man sich durch die Verwendung der Multitenant-Option nicht um die Parametrierung beziehungsweise das Starten der Datenbank-Instanz kümmern muss.

Compliance

Das hört sich alles ganz gut an – und funktioniert übrigens auch ganz hervorragend. Allerdings stellt sich die Frage, ob man solche Kopien überhaupt anlegen darf. Es sollte selbstverständlich sein, dass man Daten der Produktion nicht einfach kopieren und schon gar nicht für Entwicklungssysteme nutzen kann. Insofern ist die ursprüngliche Definition von Klonen als „identische Kopie" gar nicht das, was wir haben wollen. Wichtig ist, dass es ein Data-Masking-Verfahren für die Kopie gibt. Eine Herausforderung, die gar nicht so leicht zu lösen ist, denn damit wird ja schreibend auf den Klon zugegriffen und die ganze Technik ad absurdum geführt.

Delphix bietet daher eine Möglichkeit an, die Daten auf dem Weg zum Klon zu maskieren. Bei CloneDB und Snapshot Clone sind dem Autor solche Verfahren nicht bekannt. Als Alternative bietet sich hier an, eine Kopie der Datenbank zu erstellen, die mithilfe der Oracle-Data-Masking-Option gesäubert und dann als Quelle für die Klone genutzt wird.

Fazit

Generell können die genannten Verfahren den Storage-Bedarf von Datenbank-Kopien drastisch reduzieren. Delphix spricht davon, dass nur noch 5 Prozent der ursprünglichen Ressourcen erforderlich sind. Das ist sicherlich in einigen Umgebungen ein durchaus realistischer Wert. Allerdings eignen sich die hier vorgestellten Verfahren nicht für Performance-Tests, da der zusätzliche I/O die Messungen verfälschen würde.

Die Multitenant-Option eröffnet sicherlich noch weitere Möglichkeiten des Cloning, allerdings beschränkt sich dieses Verfahren auf Datenbanken, die auf einem gemeinsamen Server (natürlich auch als RAC) betrieben werden. Die anderen vorgestellten Lösungen sind mit den aktuellen Oracle-Versionen einsetzbar und können in Zukunft helfen, schneller, einfacher und vor allen Dingen in größerem Maße Datenbanken für Test und Entwicklung zur Verfügung zu stellen.

Johannes Ahrends
johannes.ahrends@carajandb.com

Schnelles Datenbank-Cloning mit Snapshots – ein Überblick

Manuel Hoßfeld, ORACLE Deutschland B.V. & Co. KG

Inhaltlich identische Kopien („Klone“) von Datenbanken sind für verschiedene Zwecke praktisch oder sogar notwendig – sei es als Kopie einer Produktionsdatenbank für Entwicklungszwecke oder als Basis von Tests, die man nicht an einer Produktionsdatenbank durchführen kann oder will.

Normalerweise erfordert das Anlegen eines Datenbank-Klons das vollständige Kopieren aller zum Original gehörenden Datenbank-Dateien. Es gibt inzwischen jedoch verschiedene alternative Möglichkeiten, Datenbanken auch über Snapshot-Technologien zu duplizieren, ohne dass tatsächlich ein physischer Kopiervorgang anfällt. Dieser Artikel zeigt, warum dies sinnvoll ist, und gibt einen Überblick über einige der möglichen Varianten. Da es sich hier nur um eine Übersicht handelt, erhebt er keinen Anspruch auf Vollständigkeit oder eine technisch erschöpfende Darstellung. Für tiefergehende Informationen empfiehlt sich daher der Blick in weiterführende Artikel oder in die jeweilige Dokumentation.

```
create pluggable database
testklon-db from prod-db snapshot copy;
```

Listing 1

Das vollständige Kopieren einer Datenbank mit traditionellen Methoden wird gelegentlich auch als Erzeugen einer „full copy“ oder „fat copy“ bezeichnet. Je nach Größe der Datenbank und Geschwindigkeit des zugrunde liegenden Storage-Systems kann dieser Vorgang sehr zeitraubend sein. Auch ist das Kopieren einer kompletten Datenbank in der Regel nicht sehr effizient, wenn man bedenkt, dass für die meisten Anwendungsfälle nur sehr wenige Änderungen oder neue Daten in der geklonten Datenbank anfallen werden.

Viel sinnvoller erscheint es daher, einen vollständigen Kopiervorgang zu vermeiden, indem man sich die Eigenheiten von Storage-Snapshots zunutze macht. Man könnte dabei von einer logischen Kopie (auch „thin copy“ oder „thin clone“) sprechen. Vorteile dabei sind vor allem der