Let's talk about LOBs please

Beate Künneke
BK Unternehmensberatung
Lichtenau

Schlüsselworte
Long, LOB, CLOB, BFILE, BLOB, basicfile, securefile, flashback, nologging, deduplication, compression, partitioning, chunk, Migration, PL/SQL

Einleitung
Die Speicherung von Daten im LOB-Datentyp (wie z.B. BLOB, CLOB) ist eine wichtige Grundlage, um unstrukturierte und semi-strukturierte Daten wie Dokumente, graphische Daten, XML-Dokumente, geospatial data, biometric information, video files usw. in der Datenbank abzuspeichern. Die Anforderungen an die Datenhaltung dieser Objekte steigen mit Speicherung, Verwaltung und performanter Abfragen immer größerer Datenmengen unter Beachtung von Sicherheitsaspekten.

Historie

1. **Datentyp LONG**

Größere Datenmengen (bis 2 GB) konnte in früheren Oracle-Versionen (< 8i) als LONG oder LONG RAW abgespeichert werden. Hierzu gab es und gibt es heute noch genauso eine Menge von Einschränkungen/Nachteilen: So kann es pro Tabelle maximal eine LONG-Spalte geben, eine Indizierung so einer Spalte ist nicht möglich, die Verwendung in der where-Klausel ist nicht möglich, als Rückgabe in einer Funktion ist es auch nicht möglich usw. Mit Version 8i kamen die LOBs (Large Objects) hinzu mit der ORACLE-Empfehlung LONG-Spalten auf LOB zu migrieren. Ein zusätzlicher Grund war die Einführung von Partitionierung, Partitionierung kann nicht mit LONG-Datentypen in der Tabelle kombiniert werden. Aktuell wird der Datentyp LONG aus Gründen der Kompatibilität noch weiter unterstützt, ist aber abgekündigt. Allerdings sind im Data Dictionary der Datenbank auch in 12c immer noch LONG Datentypen im Einsatz.

2. **LOB**

Lobs wurden mit 8i als der zu verwendende Datentyp eingeführt, dabei wird LONG durch CLOB und LONG RAW durch BLOB (B steht dabei für Binär) ersetzt. Zusätzlich gibt es noch 2 weitere Datentypen: BFILE mit Speicherung einer Referenz in der DB, die zugehörige Datei ist im Filesystem abgelegt und NCLOB zur Aufnahme nationaler Zeichensätze wie Japanisch, Chinesisch, etc. Die Verwendung des NCLOB-Datentyps ist durch die Einführung von UTF-8/16 Charactersets der Datenbank in vielen Fällen obsolet geworden. Für die Speicherung von LOBs stehen 2 Formen der Datenhaltung zur Verfügung.

   a. **Basicfiles**

In der ORACLE Version 8i wurde die Speicherung von großen unstrukturierten Lobs eingeführt. Die hierfür zugrundeliegende Speicherform wird aktuell als „basicfile" bezeichnet. Sie ist auch weiterhin (noch) gueltig.

   b. **Securefiles**

In 11g wurde die LOB-Technologie in Oracle grundlegend überarbeitet: Eine neue effizientere und funktional mehrwertige Speicherstruktur wurde geschaffen, die ORACLE Securefiles.
ASSM Tablespaces sind eine Voraussetzung für das Anlegen dieser neuartigen LOBs. Oracle Securefiles können im Gegensatz zu Basicfiles komprimiert, dedupliziert und/oder verschlüsselt werden. Allerdings ist dies Feature nicht umsonst: Sobald mindestens eine dieser Technologien

eingesetzt wird, wird es kostenpflichtig und in einer EE zu lizensieren. Durch die Komprimierung und/oder Deduplizierung verspricht man sich neben geringerem Speicherplatzbedarf auch Performance-Gewinn. Diese in 11g neue Speicherform ist ab 12c der Standard für die Speicherung von LOBs. Im Weiteren wird es –so angekündigt- basicfiles ersetzen. Empfehlung: die Nutzung von Securefiles bei Erstellung von neuen LOB-Objekten kann in 11g durch einen DB-Parameter forciert werden. In 12c ist es der Standard.

Speicherung (Space management)
Um ein LOB zu speichern, verwendet Oracle auch für Securefiles die gleichen Datenstrukturen analog zu basicfiles, ein LOB Segment und einen LOB Index. Der LOB Index wird benötigt, um auf LOB Chunks (Erklärung folgt unten) aus dem LOB Segment zuzugreifen. LOBs können nach wie vor inline in der Tabelle (enable storage in row) oder auch out-of-line (disable storage in row) gespeichert werden. Inline werden maximal 4000 Bytes gespeichert. Für LOBs, die out-of-line gespeichert werden, speichert Oracle einen LOB Locator in der entsprechenden Row. Der LOB Locator bietet dann den Einstieg, um über den LOB Index auf das LOB im LOB Segment zuzugreifen. Wird hingegen Inline Speicherung verwendet, so können kleine LOBs zusammen mit den anderen Spalten der Tabelle in demselben Block gespeichert werden. Überschreitet das LOB jedoch eine Größe von 4000 Bytes wird es immer komplett out-of-line gespeichert. Inline Speicherung empfiehlt sich nur für kleine LOBs, die häufig zusammen mit den anderen Spalten der Tabelle gelesen werden.

Die Daten in den LOB Segmenten werden in Chunks unterteilt. Sie bilden eine kleinste Einheit für den Zugriff und die Manipulation. Die Definition muss ein Vielfaches der Blockgröße sein. Bei Basicfiles können sie maximal 32K groß sein und sind immer gleich groß.
Im Gegensatz dazu sind die Größen der Chunks bei Securefiles dynamisch und können auch größer als 32K werden. Der DB-Parameter Chunk ist jedoch nach wie vor auch für Securefiles gültig, er dient jedoch nur noch der Abwärtskompatibilität bzw. wird vom Oracle Server als Vorschlag erachtet.
Außer den eigentlichen LOB-Daten werden auch LOB-Indexe gespeichert. Beide sind eng miteinander verbunden und befinden sich immer in demselben Tablespace. Ein LOB-Index kann nicht separat gelöscht oder neu angelegt werden, er ist nicht änderbar.
Falls Lob Segmente andere Anforderungen an die Datenhaltung (Stichwort Größenverhältnis) stellen als die eigentlichen Daten, dann macht es durchaus Sinn sie in einem anderen Tablespace abzulegen. Anwendungen können aber auch nach wie vor von der manuellen Segmentplatzverwaltung oder von separaten Tablespaces profitieren. Hierzu gehören solche Anwendungen, in denen von Zeit zu Zeit größere Datenmengen gelöscht werden. Mit ASSM werden neue Daten oft über teilweise gefüllte Blöcke verteilt die u.U. auch nicht unmittelbar nebeneinander liegen. Neue Daten haben allerdings oft eine Beziehung zueinander, so dass bei einem Zugriff auf verteilte Datenhaltung oft mehr Datenblöcke gelesen werden müssen als bei zusammenhängenden Blöcken.
In 11gR2 wird für eine bessere Performance für LOBS empfohlen, diese in einem separaten Tablespace als die anderen Daten der Tabelle zu halten. Bei häufigem Zugriff auf unterschiedliche LOB-Spalten wird empfohlen auch jede Lob-Spalte einer Tabelle diese in einem separaten Tablespace anzulegen. Das heißt aber nicht, dass mehrere TBS die Performance automatisch verbessern, da sie auch auf derselben Platte liegen können. Sie dienen dann in erster Linie der komfortableren Administration. Es sind allerdings mehr Blöcke zu lesen wenn für eine Abfrage inline LOBs enthalten sind. Maximal 4000 Bytes können inline gespeichert werden. Für ein out-Line gespeichertes LOB wird in dem Tabellenfeld nur die Referenz hinterlegt.

Basicfile versus Securefile
Bezüglich Lesekonsistenz gibt es auch eine Neuerung: Der Parameter **PCTVERSION**, mit dem es für Basicfiles möglich ist, ein Prozentsatz des Speicherplatzes vom LOB Segment für alte LOB Versionen anzugeben, ist für Securefiles nicht mehr gültig. Hier kommt nur noch der Parameter **RETENTION**

zum Einsatz. Hiermit wird eine Zeitperiode definiert für die ältere Versionen noch verfügbar bleiben. Man kann entweder PCTVERSION oder RETENTION wählen.

Migration
Zunächst einmal beginnt oft alles mit einer VARCHAR2-Spalte für die 4000 Byte (bis 11g, in 12c bis 32k) nicht ausreichen. Handelt es sich z.B. um einen Multibyte-Zeichensatz, so können ggfs. nur 1000 Zeichen dort abgespeichert werden, da ein Zeichen max. 4 Byte beansprucht. Eine VARCHAR2-Spalte kann in eine LONG-Spalte modifiziert werden, falls es sich nicht um eine partitionierte Tabelle handelt. Eine Long-Spalte kann wiederum in eine LOB-Spalte modifiziert werden. Um Daten von Basicfiles zu Securefiles zu migrieren, gibt es leider keine „einfachen" Methoden. Die Daten müssen wirklich bewegt werden. Als Möglichkeiten zur Migration möchte ich CTAS, „insert into … select …", „alter table … move lob", Data Pump und Online Table Redefinition (mittels DBMS_REDEFINITION) nennen. Die letztgenannte Methode ist die einzige, die es ermöglicht online Migrationen durchzuführen, ohne zugreifende Applikationen zu behindern.

Partitionierung
Lobs können partitioniert werden wie wir es von „normalen" Tabellen gewohnt sind. Hier gibt es ein nettes Feature: Auch wenn die bestehenden Partitionen eines Lobs vom Typ „Basicfile" sind, können neue Partitionen vom Typ „Securefile" hinzugefügt werden. LOB-Daten und LOB-Indexe werden bei partitionierten Lobs analog zu den Daten-Partitionen angelegt.

Logging/Nologging
Securefiles bieten außerdem eine zusätzliche Logging Option FILESYSTEM_LIKE_LOGGING" mit welcher es nun möglich ist, Redo Informationen für Metadaten zu loggen, für Daten jedoch nicht.

Freigabe von Speicherplatz
Die Freigabe von Speicherplatz ist wie bei den "normalen" Tabellen über einen Shrink-Befehl möglich: Denn der Befehl auf Tabellenebene inkludiert auch die Lob-Segmente. Falls man allerdings nur Speicherplatz von Lob-Segmenten freigeben möchte, kann man das auch entsprechend einschränken:

```
ALTER TABLE employees MODIFY LOB (emp_lob) (SHRINK SPACE);
```

Des Weiteren ist noch hinzufügen, dass leere Segmente auch über die Nutzung der Funktionalität von `DBMS_SPACE_ADMIN.DROP_EMPTY_SEGMENTS` entfernt werden können.

Flashback
Für Flashback Query oder Flashback Table werden im Allgemeinen die überschriebenen Versionen von Daten aus dem UNDO-Tablespace rekonstruiert. Dies gilt jedoch für Lobs nicht, denn diese werden nicht aus dem UNDO-TBS bedient, sondern aus dem LOB-Tablespace. Hier gibt es Speicherbereiche die für die vergangenen Versionen genutzt werden. Potentieller Platz im UNDO-Tablespace unterstützt somit einem Flashback (Query oder Table) für LOBs nicht. Die Aufbewahrung vergangener Versionen wird über die Einstellung PCTVERSION oder RETENTION (eins von beiden, beides geht nicht) gesteuert. Hierbei ist zu beachten wie das Segment-Space-Management (AUTO oder MANUAL) des Tablespace gesetzt ist. Für Segment-Space-Management AUTO (ASSM, diese Option ist vorzuziehen) ist der Parameter RETENTION maßgeblich. Alternativ kann bei Basicfiles anstatt RETENTION der Parameter PCTVERSION genutzt werden. Bei Segment-Space-Management MANUAL (MSSM) ist zu beachten, dass nur über diesen Parameter ein Flashback (Query oder Table) gesteuert werden kann.

Lobs und UTF-8
Bei der Benutzung von Multibyte-Datensätzen ist zu beachten, dass bei der Verarbeitung des LOBs ggfs. die korrekten Längen in Byte ermittelt werden. So führt das Schreiben eines Textes v_data mit

```
dbms_lob.writeappend(v_blob, dbms_lob.getlength(v_data),
utl_raw.cast_to_raw(v_data)); -- falsch
```

zu einem fehlerhaften BLOB. Richtig ist die Benutzung von

```
dbms_lob.writeappend(v_blob, utl_raw.length( utl_raw.cast_to_raw(v_data)),
utl_raw.cast_to_raw(v_data)); -- korrekt
```

SQL und PL/SQL
Ab ORACLE 12c ist die parallele Ausführung von SQL auf partitionierten Tabellen mit Securefiles LOBs und Basicfiles LOBs und auch nicht partitionieren Tabellen mit Securefiles LOBs für folgende Operationen möglich:
INSERT, INSERT AS SELECT, CREATE TABLE AS SELECT, DELETE, UPDATE, MERGE, Multi-table INSERT, SQL*Loader und Import/Export.

Kostenpflichtige Optionen Deduplication/Compression und Encryption
Securefiles können im Gegensatz zu den Basicfiles komprimiert, dedupliziert und/oder verschlüsselt werden. Für den Einsatz von Komprimierung ist die Oracle Advanced Compression zu lizensieren. Dies gilt ebenfalls für die Deduplikation. Für die Verschlüsselung ist die Option Oracle Advanced Security zu lizensieren. Im Hinblick auf immer größere werdende Datenmengen (die entsprechend Speicherplatz) benötigen und auch stetig steigende Sicherheitsaspekte durchaus interessante Optionen.

Kontaktadresse:
Beate Künneke
BK Unternehmensberatung
Schulstraße 13
D-33165 Lichtenau

Telefon: +49 (0) 5295-8015
Fax: +49 (0) 5295-9979885
E-Mail bk@berlin.de