Gestiunea spațiului în Oracle

DBA-ul de servici: optimizarea stocării datelor în Oracle

– *Monica Ibănescu*

Gestiunea spațiului este una din responsabilitățile de bază ale administratorului bazei de date. Modul de stocare a datelor în baza de date influențează performanța execuției cererilor de date. Dacă datele sunt fragmentate în extinderi multiple, atunci, rezolvarea unei cereri poate conduce la căutarea în mai multe locații fizice pentru aducerea rândurilor referite. Fragmentarea spațiului liber poate încetini performanța la introducerea de noi înregistrări, deoarece baza de date va trebui sa combine extinderile vecine libere pentru crearea unei singure extinderi, capabilă sa stocheze noile înregistrări.

Serverul Oracle dispune de facilități care simplifică această activitate, prin definirea unui număr nelimitat de extinderi, posibilitatea dealocării spațiului într-un segment și a compactării spațiului fragmentat dintr-o tabelă de spațiu. Orice operațiune de *„data storage tuning"* presupune defragmentarea segmentelor de date, defragmentarea extinderilor libere, depistarea și înlăturarea liniilor înlănțuite și migratoare. Să începem, deci, cu…

Defragmentarea segmentelor

La crearea unui obiect în baza de date (tabelă sau index de exemplu) acesta este asociat unei tabele de spațiu. Un *segment* este creat în tabela de spațiu pentru a stoca datele asociate obiectului nou creat. Spațiul alocat segmentului nu este eliberat până când asupra acestuia nu se execută una din comenzile: `DROP`, `TRUNCATE` sau `SHRINK`. Un segment este alcătuit din secțiuni numite extinderi(extents). Extinderile sunt seturi contigue de blocuri Oracle. Odată ce o extindere existentă nu mai poate stoca noi date, segmentul obține o nouă extindere. Acest proces de extindere continuă până când nu mai există spațiu disponibil în fișierul de date asociat tabelei de spațiu respective sau pâna când este atins numărul maxim de extinderi admis de segment. Dacă un segment cuprinde mai multe extinderi, nu este garantat că acestea sunt contigue. Este bine ca fiecare segment de date să aibă o singură extindere. Parametrul de stocare `INITIAL` al oricărui segment, cel care specifică mărimea primei extinderi, trebuie setat destul de mare încât să poată cuprinde toate datele segmentului. Numărul mare de extinderi nu are un impact negativ asupra performanței cererilor, dacă extinderile au fost dimensionate corect.

Ce înseamnă o extindere dimensionată corect? Oracle citește datele din tabele în două moduri: prin `RowID`(acces pe bază de index) sau prin full table scan(parcurgerea secvențială a înregistrărilor tabelei). Dacă citirea se face prin `RowID`, atunci numărul de extinderi ale tabelei nu reprezintă un factor de influență al performanței citirii. Oracle va citi fiecare linie de la locația fizică specificată prin `RowID`. În cazul parcurgerii secvențiale a tabelei, mărimea extinderilor are impact asupra performanței. Oracle poate citi mai multe blocuri în același timp. Numărul lor este setat prin parametrul `DB_FILE_MULTIBLOCK_READ_COUNT` din fișierul init.ora și este limitat de mărimea bufferului de I/O a sistemului de operare. De exemplu, dacă mărimea blocului bazei de date este de 4KB iar mărimea bufferului de I/O a sistemului de operare este de 64KB, atunci Oracle poate citi maxim 16 blocuri ale bazei de date în același timp. Setarea cu o valoare mai mare de 16 a parametrului `DB_FILE_MULTIBLOCK_READ_COUNT` nu va schimba cu nimic performanța parcurgerii integrale a tabelei. Mărimea unei extinderi trebuie să profite de abilitatea pe care o are Oracle de a citi mai multe blocuri în același timp. De aceea, dacă mărimea bufferului de I/O a sistemului de operare este de 64KB atunci mărimea unei extinderi trebuie sa fie multiplu de 64KB. Considerând o tabela cu 10 extinderi, fiecare extindere având 64KB, mărimea bufferului de I/O a sistemului de operare de 64KB, pentru a executa o parcurgere secvențială a tabelei, Oracle trebuie să efectueze 10 citiri. Dacă datele din tabelă sunt compresate într-o singură extindere de 640KB, Oracle va efectua tot 10 citiri pentru a parcurge tabela. Dacă mărimea extinderii tabelei nu este multiplu de mărimea bufferului de I/O a sistemului de operare, atunci numărul de citiri efectuat de Oracle va crește. Pentru aceeași tabelă de 640KB, împărțită în 8 extinderi de 80KB, Oracle va efectua 2 citiri pentru fiecare extindere, una pentru a citi 64KB din extindere și una pentru a citi restul de 16KB. Pentru a citi întreaga tabelă, Oracle va trebui să efectueze deci 16 citiri. Astfel, reducerea numărului de extinderi de la 10 la 8 printr-o dimensionare improprie a extinderilor, crește numărul de citiri din baza de date cu 60%. Pentru a evita reducerea performanței cererilor cauzată de mărimea extinderilor, trebuie să se aleagă între următoarele două strategii:

- crearea de extinderi a căror mărime să fie semnificativ mai mare decât mărimea bufferului de I/O a sistemului de operare (dacă extinderea este foarte mare, atunci un număr foarte mic de citiri suplimentare vor fi necesare, chiar dacă mărimea extinderii nu este multiplu de dimensiunea bufferului);
- crearea de extinderi a căror mărime să fie multiplu de dimensiunea bufferului de I/O a sistemului de operare.

Operațiile DDL(Data Definition Language) sunt totuși afectate de numărul mare de extinderi ale unui obiect. Când un obiect aloca o extindere, Oracle modifică intrările în tabela de extinderi utilizate, `SYS.UET$`. În același timp, modifică intrările în tabela de extinderi libere, `SYS.FET$`. De aceea, dacă o tabelă are un număr mare de extinderi, modificările pe care Oracle le face asupra tabelelor `SYS.UET$` și `SYS.FET$` încetinesc execuția comenzilor DDL. De exemplu, o comandă DROP pe o tabelă cu 5000 de extinderi durează 2 minute, aceeași comandă aplicată unei tabele cu 10000 de extinderi durează 10 minute, în timp ce un DROP pe o tabelă cu 60000 de extinderi durează aproape…o zi.

Următoarea frază select aduce informații privind tabela de spațiu, tipul de segment, owner-ul, mărimea în bytes, Kb, Mb, numărul de extinderi, precum si valorile parametrilor de stocare a obiectului pentru un anumit owner.

```
SELECT Segment_name as "Name",
       RTrim(Segment_type) as "Type",
       Tablespace_name as "TableSpace Name",
       Owner as owner,
       Bytes as "Size",
       Round(bytes / 1024) as "Size (Kb)",
       Round(bytes / 1024 / 1024) as "Size (Mb)",
       Extents "Extents",
       initial_extent "Initial Extent",
       next_extent "Next Extent",
       min_extents "Min Extents",
       max_extents "Max Extents",
       pct_increase "Pct Increase"
FROM sys.dba_segments
WHERE owner=owner_name
AND extents > 1
ORDER BY extents DESC;
```

Dacă un segment este fragmentat, cea mai usoară metodă de compresare a datelor pe care le conține într-o singură extindere este reconstruirea lui cu parametri de stocare adecvați. Întrucât parametrul de stocare `INITIAL` nu poate fi modificat după ce tabela a fost creată, o noua tabelă trebuie creată cu parametrii de stocare corecți. Vechile date vor fi apoi inserate în noua tabelă, iar cea veche poate fi ștearsă. Acest proces se face automat prin Export/Import. Parametrul de export `COMPRESS` determina ca în cadrul unui export, la citirea unei tabele, să se determine spațiul total alocat acesteia și să scrie în fișierul dump o nouă valoare pentru parametrul de stocare `INITIAL`, echivalentă cu mărimea spațiului total alocat tabelei. Dacă apoi tabela este ștearsă și recreată prin Import, datele vor încăpea într-o singura extindere, de dimensiune mai mare. E bine de știut că parametrul `COMPRESS` determină spațiul total *alocat* tabelei și nu spațiul total *utilizat*. De asemenea, baza de date Oracle nu va verifica dacă noua extindere inițială depășește mărimea celui mai mare fișier de date asociat tabelei de spațiu. Întrucât o extindere nu poate fi cuprinsă decât de un singur fișier de date, depășirea acestuia va duce la eroare în timpul importului.

Procedura de compresie a unui fișier de date presupune trei pași: exportul tabelei, ștergerea ei și apoi importul aceteia:

```
EXP system/manager FILE=exp.dmp COMPRESS=y GRANTS=y
   INDEXES=y TABLES=owner_name.table_name
DROP TABLE table_name CASCADE CONSTRAINTS;
IMP system/manager FILE=exp.dmp COMMIT=y BUFFER=64000 FULL=y
```

Metoda poate fi folosită și la nivelul întregii baze de date.

Să continuăm cu…

Defragmentarea extinderilor libere

O extindere liberă dintr-o tabelă de spațiu este o colecție de blocuri libere, contigue. Când un segment este șters, extinderile sale sunt dealocate și marcate ca fiind libere. Procesul de background `SMON` alipește periodic extinderile libere vecine, dacă valoarea implicită a parametrului de stocare `PCTINCREASE` al tabelei de spațiu este diferit de zero. În cazul în care acesta are valoarea zero, extinderile libere alăturate nu vor mai fi alipite automat de către baza de date, iar spațiul liber va rămane fragmentat. Lucru neplăcut, care afectează alocarea spațiului la următoarea cerere de spațiu de stocare(cum ar fi crearea sau extinderea unei tabele). În încercarea de a aloca spațiul necasar pentru noul obiect, baza de date nu va putea folosi extinderile libere din tabela de spațiu, care au dimensiunea prea mică în comparație cu cererea de spațiu. În cel mai fericit caz, se va aloca o nouă extindere, dar există și posibilitatea în care, din lipsă de spațiu, de exemplu, un obiect nu va mai putea fi creat deși ar ocupa mai puțin decât spațiul liber total din tabela de spațiu. Concret, avem o tabelă de spațiu cu 90Kb liberi dar care are spațiul fragmentat în extinderi de 30Kb. O instrucțiune DDL, care are nevoie de o extindere inițială de 40Kb nu se poate executa.

Opțiunea `COALESCE` a comenzii `ALTER TABLESPACE` este o încercare de suplinire a procesului `SMON`. Comanda este următoarea:

```
ALTER TABLESPACE tablespace_name COALESCE;
```

Următoarea frază select determină spațiile libere din cadrul unei tabele de spațiu(nume_tabela_de_spațiu):

```
SELECT owner segment_owner, Segment_name, partition_name,
   Block_ID, File_id, Blocks, segment_type
FROM sys.DBA_Extents
WHERE tablespace_name = nume_tabela_de_spațiu
UNION
SELECT NULL, 'Free Space', NULL, Block_ID, File_id, Blocks,
   NULL
FROM sys.DBA_FREE_SPACE
WHERE tablespace_name = nume_tabela_de_spațiu
ORDER BY 5,4
```

Opțiunea COALESCE a comenzii ALTER TABLESPACE alipește extinderile libere adiacente(acest gen de fragmentare este uneori numită *fragmentare honeycomb* -fagure). Ea nu alătură spațiile libere dintr-o extindere, dacă aceste spații sunt separate de extinderi alocate. În mod curent, administratorii de baze de date pot elimina acest tip de fragmentare(numită uneori si *fragmentare bubble*-balon) prin exportarea obiectelor din tabela de spațiu cu optiunea COMPRESS=y, ștergerea și recrearea tabelei de spațiu și apoi importarea tuturor obiectelor înapoi în tabelă.

Pentru a activa procesul SMON să alipească extinderile libere, poate fi modificată valoarea implicită a parametrului PCTINCREASE pentru tabela de spațiu utilizând comanda:

```
ALTER TABLESPACE nume_tabela_de_spațiu
DEFAULT STORAGE(PCTINCREASE 1)
```

În încheiere câteva cuvinte despre…

Rândurile înlănțuite și cele migratoare

Baza de date Oracle nu lucrează la cele mai ridicate performanțe dacă nu se iau măsuri pentru minimizarea operațiilor fizice de intrare/ieșire. Serverul Oracle stochează datele în blocuri a căror mărime este aleasă la crearea bazei de date(de exemplu 2K, 4K, 8K).

La crearea unui segment de date este specificată valoarea parametrului de stocare PCTFREE. Acest parametru indică bazei de date cât spațiu trebuie păstrat liber în fiecare bloc. Acest spațiu este folosit în cazul extinderii rândurilor deja stocate, prin operații de UPDATE.

Dacă în urma modificărilor, dimensiunea tabelei a crescut peste capacitatea blocului de date original, serverul Oracle mută linia în alt bloc și păstreaza în blocul original de date un pointer către noua locație a liniei. Această situație este numită *migrarea liniei(row migration)*. *Înlănțuirea liniei(row chaining)* apare când o linie din tabelă este prea mare ca sa încapă într-un singur bloc al bazei de date, părți din linia respectivă fiind stocate în mai multe blocuri. De exemplu, dacă dimensiunea blocului bazei de date este de 2048 de octeți și o linie are 3000 de octeti, această linie nu va încăpea într-un singur bloc al bazei de date.

În oricare din cele două cazuri, această situație „o linie, mai multe blocuri" impune serverului Oracle să execute o operație fizică de I/O suplimentară pentru a accesa liniile înlănțuite sau migratoare. Înlănțuirea liniilor poate fi evitată prin setarea corectă a valorii parametrului PCTFREE(valoarea implicită a parametrului PCTFREE este 10), la crearea obiectului. Acesta poate fi calculat dupa formula:

```
PCTFREE=(lungimea maximă a unei linii(în bytes)-lungimea
   medie a unei linii(în bytes))/ lungimea maximă a unei
   linii(în bytes) *100
```

Detectarea liniilor înlănțuite se poate face utilizând comanda:

```
ANALYZE TABLE table_name LIST CHAINED ROWS INTO
   CHAINED_ROWS;
```

Tabela CHAINED_ROWS trebuie creată în prealabil executând scriptul utlchain.sql din subdirectorul /rdbms/admin al directorului în care a fost instalat Oracle. Următoarea cerere va selecta datele cele mai semnificative din tabela CHAINED_ROWS:

```
SELECT owner_name, table_name, cluster_name, head_rowid
FROM CHAINED_ROWS;
```

Dacă o tabelă conține linii înlănțuite, aceasta trebuie recreată cu o valoare mai mare a parametrului PCTFREE. Înlănțuirea liniilor datorată depășirii dimensiunii blocului este mai problematică, întrucât singurul mod de a o elimina este reconstruirea bazei de date cu un bloc de dimensiune mai mare. Procesul implică un export total al bazei de date, reconstruirea și apoi importul acesteia. De accea este bine ca, în timpul procesului de creare a bazei de date, administratorul acesteia să aleagă dimensiunea blocului astfel încât în el să încapă cea mai mare linie ce va fi stocată vreodată în orice tabelă. Mărimea blocului bazei de date este setat prin parametrul DB_BLOCK_SIZE din fișierul init.ora al bazei de date.

Migrarea liniilor poate fi rezolvată prin mutarea liniilor din tabelă într-o tabelă temporară, ștergerea lor din tabela de origine și copierea lor înapoi în tabela inițială.

```
DROP TABLE temp;

CREATE TABLE temp AS
SELECT * FROM &tabela_cu_linii_migratoare
WHERE RowID IN
(SELECT head_rowid FROM chained_rows
WHERE table_name=UPPER(&tabela_cu_linii_migratoare));

DELETE FROM &tabela_cu_linii_migratoare
WHERE RowID IN
(SELECT head_rowid FROM chained_rows
WHERE table_name=UPPER(&tabela_cu_linii_migratoare));

INSERT INTO tabela_cu_linii_migratoare
SELECT * FROM temp;

COMMIT;
```

Dacă nu e posibilă crearea unei tabele în care să se păstreze temporar liniile migratoare, facilitățile de import/export rezolvă problema liniilor migratoare.

Concuzii

Monitorizarea fragmentării spațiului în Oracle are implicații directe asupra performanței bazei de date. O bună administrare a spațiului conduce la evitarea numărului mare de I/O pentru o citire logică, reducerea timpului de căutare pe disc, evitarea creșterii aparent nejustificate a dimensiunii obiectelor bazei de date și implicit a spațiului pierdut.

Monica Ibănescu este ORACLE Programmer la Romanian Data Soft. Poate fi contactată pe email la: s_ibanes@yahoo.com. ■ 75

Bibliografie

✓ Oracle8I DBA Handbook: Kevin Loney,Marlene Theriault
✓ Oracle8 Secrete: Edward Honour,Paul Dalberth,Ari Kaplan, Atul Mehta
✓ Oracle7 Administration: David Austin, Laie Caindec, Julie Frazier, Sue Jang