RECLAIM SPACE USED BY TABLE AND INDEX AFTER DATA DELETION

by Snehashish Ghosh

April 7, 2013, in category Storage comments No comments

Scenario

Significant amount of data (more than 80%) has been deleted from a database schema. However size of schema has not come down proportionately with the volume of data deleted.

Solution

---

1. Perform Online Segment Shrink

The Online Segment Shrink process consolidates fragmented free space below the high water mark and compacts the segment. After compaction, the high water mark is moved, resulting in new free space above the high water mark. That space above the high water mark is then deallocated.

1.1 Benefits

- Compaction of data leads to better cache utilization, which in turn leads to better online transaction processing (OLTP) performance.
- The compacted data requires fewer blocks to be scanned in full table scans, which in turns leads to better decision support system (DSS) performance.

1.2 Downtime

Segment shrink is an online, in-place operation. DML operations and queries can be issued during the data movement phase of segment shrink. Concurrent DML operation are blocked for a short time at the end of the shrink operation, when the space is deallocated. Indexes are maintained during the shrink operation and remain usable after the operation is complete. Segment shrink does not require extra disk space to be allocated.

1.3 Limitation

- Shrink operations can be performed only on segments in locally managed tablespaces with automatic segment space management (ASSM). Within an ASSM tablespace, all segment types are eligible for online segment shrink except these:
- IOT mapping tables
- Tables with rowid based materialized views
- Tables with function-based indexes
- `SECUREFILE` LOBs
- Tables with LONG column
- As with other DDL operations, segment shrink causes subsequent SQL statements to be reparsed because of invalidation of cursors unless you specify the`COMPACT` clause.

1.4 Shrink Operation Types

Two optional clauses let you control how the shrink operation proceeds:

- The `COMPACT` clause lets you divide the shrink segment operation into two phases. When you specify `COMPACT`, Oracle Database defragments the segment space and compacts the table rows but postpones the resetting of the high water mark and the deallocation of the space until a future time. This option is useful if you have long-running queries that might span the operation and attempt to read from blocks that have been reclaimed. The defragmentation and compaction results are saved to disk, so the data movement does not have to be redone during the second phase. You can reissue the `SHRINK SPACE` clause without the `COMPACT` clause during off-peak hours to complete the second phase.
- The `CASCADE` clause extends the segment shrink operation to all dependent segments of the object. For example, if you specify `CASCADE` when shrinking a table segment, all indexes of the table will also be shrunk. (You need not specify `CASCADE` to shrink the partitions of a partitioned table.) To see a list of dependent segments of a given object, you can run the `OBJECT_DEPENDENT_SEGMENTS` procedure of the `DBMS_SPACE` package.

1.5 Perform Shrink

Segment shrink requires that rows be moved to new locations. Therefore, you must first enable row movement in the object you want to shrink and disable any rowid-based triggers defined on the object. You enable row movement in a table with the `ALTER TABLE` ... `ENABLE ROW MOVEMENT` command.

Shrink a table and all of its dependent segments (including `BASICFILE` LOB segments):

```
<span style="background-color: #ccffff;">ALTER TABLE employees SHRINK SPACE CASCADE;</span>
```

Shrink a `BASICFILE` LOB segment only:

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

Shrink a single partition of a partitioned table:

```
<span style="background-color: #ccffff;">ALTER TABLE customers MODIFY PARTITION cust_P1 SHRINK SPACE;</span>
```

Shrink an IOT index segment and the overflow segment:

```
<span style="background-color: #ccffff;">ALTER TABLE cities SHRINK SPACE CASCADE;</span>
```

Shrink an IOT overflow segment only:

```
<span style="background-color: #ccffff;">ALTER TABLE cities OVERFLOW SHRINK SPACE;</span>
```

Generate Shrink script for all the tables of a schema

set lines 300
set pages 10000

```
set trimspool on
set feedback off
spool shrink_space.sql
select ‘alter table ‘ || table_name || ‘ enable row movement;’ from dba_tables where owner=’<owner
of the schema/schema name>’;
select ‘alter table ‘ || table_name || ‘ shrink space cascade;’ from dba_tables where owner=’<owner
of the schema/schema name>’;
select ‘alter table ‘ || table_name || ‘ disable row movement;’ from dba_tables where owner=’<owner
of the schema/schema name>’;
spool off
spool shrink_space.txt
@shrink_space.sql
spool off
```

1.6 Space not fully freed after Shrink Operation

It has been observed that in some scenario space is not fully freed from the objects of the schema after shrink is performed.

Also the expected size of the tables and indexes seem to be very high with respect to the amount of data in the tables and indexes.

Check using the following query:

```
select segment_name,sum(bytes)/1024/1024 size_mb from dba_segments where owner=’<owner of
the schema/schema name>’ order by sum(bytes)/1024/1024 desc;
```

To reclaim the space further we need to perform the following:

1.6.1. Perform Index Re-Build

Generate Index Rebuild script for all the tables of a schema

```
set lines 300
set pages 10000
set trimspool on
set feedback off
spool rebuild_indexes.sql
select ‘alter index’ || index_name || ‘ shrink space cascade;’ from dba_indexes where owner=’<owner
of the schema/schema name>’;
spool off
spool rebuild_indexes.txt
@rebuild_indexes.sql
spool off
```