Moving tables from multi-table table spaces to partition-by-growth table spaces

Last Updated: 2022-06-21

FL 508 You can move tables from multi-table simple or multi-table segmented (non-UTS) table spaces, which are deprecated, to partition-by-growth universal table spaces (UTS).

Before you begin

Use the ALTER TABLESPACE statement with the MOVE TABLE option to move tables from a source multi-table table space to target partition-by-growth table spaces. If the data sets of the source table space are already created, a MOVE TABLE operation becomes a pending definition change that must be materialized by running the REORG utility on the source table space. For more information, see Pending data definition changes.

Before you move any tables, complete the following tasks to prepare for a MOVE TABLE operation:

Identify tables to move

Use the following query to identify all multi-table table spaces and the number of tables in each table space:

```
SELECT CURRENT SERVER AS DB2LOC,
       DBNAME, DBID, NAME AS TSNAME, NTABLES
  FROM SYSIBM.SYSTABLESPACE
  WHERE NTABLES > 1
  ORDER BY DBNAME, TSNAME;
```

Then use the following query to identify all tables in a multi-table table space:

```
SELECT CURRENT SERVER AS DB2LOC,
       TSP.DBNAME, TSP.DBID, TSP.NAME as TSNAME, TSP.NTABLES,
```

```
     TAB.CREATOR AS TBCREATOR,
     TAB.NAME AS TBNAME,
     TAB.TYPE AS TBTYPE
FROM SYSIBM.SYSTABLESPACE AS TSP,
     SYSIBM.SYSTABLES AS TAB
WHERE TSP.NTABLES > 1
AND TSP.DBNAME = TAB.DBNAME
AND TSP.NAME = TAB.TSNAME
ORDER BY TSP.DBNAME, TSP.NAME, TAB.CREATOR, TAB.NAME;
```

Create the target table space, if needed

The target table space must already exist as a partition-by-growth UTS in the same database as the source multi-table table space and be defined with the following attributes:

- DEFINE NO attribute
- MAXPARTITIONS value of 1
- LOGGED or NOT LOGGED attribute that matches the source table space
- CCSID values that match the source table space

If you plan to create a new target table space, first determine whether the database has enough available OBIDs to accommodate the table space creation, which requires two OBIDs. For more information, see Identifying databases that might exceed the OBID limit.

> **Remember:** When you create a new table space, consider the possible impact on the number of open data sets (the DSMAX subsystem parameter), the DBD size, the EDM pool size, and so forth.

> **Recommendation:** If the target table space is defined with MAXPARTITIONS 1, it cannot grow a new partition during the materialization of a MOVE TABLE operation. Therefore, if you create a new target table space, choose a maximum partition size (DSSIZE) value that can contain all of the data in the tables that are being moved. Assuming that you create the target table space with the same space attributes (such as page size and SEGSIZE) as the source table space, you can choose one of the following recommended DSSIZE values:
>
> - 64 GB, which is the maximum size for the source table space. This DSSIZE value ensures that the target table space can contain all of the table data. However, consider the following implications of selecting a DSSIZE of 64 GB:

- The DFSMS extended addressability function must be enabled to use a DSSIZE value greater than 4 GB.
- A DSSIZE value of 64 GB limits the maximum number of partitions to which the table space can grow.
- Altering the DSSIZE value later is a pending definition change that requires running the REORG utility to materialize the entire table space.

– A DSSIZE value that is closer to the minimum size that is needed to contain all of the table data. If you collected statistics for the tables, use the following formula to calculate the amount of space that is required for the table data:

```
SYSTABLES.NPAGESF × SYSTABLESPACE.PGSIZE = table-size-in-KB
```

In this formula, *table-size-in-KB* indicates the amount of space occupied by the data in the tables that are being moved, in kilobytes. Choose the closest DSSIZE value that is greater than *table-size-in-KB*. The DSSIZE value must be a power-of-two value that is within the range 1 GB - 256 GB.

If you create the target table space with different space attributes than the source table space, the recommended DSSIZE values must be modified accordingly.

Determine the number of moved tables to materialize in a single REORG job

You can move only one table per ALTER TABLESPACE MOVE TABLE statement, but you can materialize multiple pending MOVE TABLE operations in a single REORG. To help you decide how many tables to move and materialize in a single REORG, consider the following issues:

– Generally, the total processing time for a data definition statement that is a pending definition change for a table space includes the following constituent processing times:

- The processing time for that statement
- The time that it takes to process unmaterialized pending definition changes for the table space. All unmaterialized pending definition changes for the table space are processed, regardless of whether the changes were executed in a single unit of work or across multiple units of work.

Processing includes parsing of the statement, semantic validation, and updating and restoring catalog rows. You can use the following formula to calculate the time that it takes to process a subsequent ALTER TABLESPACE MOVE TABLE statement:

$$(n + 1) \times t$$

Pending definition changes are reapplied during both the REORG UTILINIT phase and the REORG SWITCH phase to become materialized. You can use the following formula to calculate the time that it takes to reprocess all of the pending statements during the REORG:

$$2 \times (n + 1) \times t$$

where:

n
Is the number of unmaterialized pending MOVE TABLE operations

t
Is the processing time for a single ALTER TABLESPACE MOVE TABLE statement

Therefore, each table that is moved linearly increases the time that is required to process the ALTER TABLESPACE statement and the time that is required for the REORG SWITCH phase to complete. The SWITCH phase directly affects the total outage duration during materialization.

- Compared to a REORG on the non-partitioned source table space, this REORG creates one additional shadow data set for each moved table. Generally, moving a large number of tables in a single REORG results in the creation of more shadow data sets and an increase in the duration of the SWITCH phase.
- Regarding the total amount of sort space and data that is processed, this REORG is comparable to a REORG on the non-partitioned source table space that does not materialize any pending changes. The number of tables that are moved does not affect the total amount of sort space and data that is processed. If a table is moved, all data in the table space except XML and LOB data are processed and all indexes are reorganized.
- For each table that is moved during a REORG materialization, a sequential inline image copy is allocated for its respective target table space, which uses virtual storage below the 16-MB line. The more tables that are moved in a single REORG, the more below-the-line virtual storage that is used by the sequential image copy data sets during the REORG. For a REORG that creates only FlashCopy inline image copies (FCIC) without

creating sequential inline image copies, this below-the-line memory consumption does not apply. The FCIC is created at the end of the SWITCH phase and does not affect mainline REORG executions during earlier phases.

– For general considerations for running a REORG to materialize pending definition changes, see Reorganization with pending definition changes.

Based on these considerations, it is recommended that you do not move more than a few hundred tables in a single REORG.

Plan schema changes ahead of time

After you issue an ALTER TABLESPACE MOVE TABLE statement and before you run the REORG utility to materialize it, most data definition changes are restricted against any table in the source table space. If a data definition statement references a table in a table space that has an unmaterialized pending MOVE TABLE operation, all immediate definition changes and a subset of pending definition changes are restricted. For more information, see Restrictions for pending data definition changes.

If any tables in the table space need schema definition changes that can be executed as immediate changes, consider executing these changes before you move the tables.

Plan rebind operations, if needed

The REORG that materializes the ALTER TABLESPACE MOVE TABLE statement invalidates packages that depend on the moved tables. Before you move the tables, you can identify packages that will become invalid and prepare necessary rebind operations ahead of time.

For a query that identifies packages that will be invalidated by a certain change, see Identifying invalidated packages. To identify packages that will be invalidated by materialization of a MOVE TABLE operation, run the query using the following values:

Table 1. Values to use in the query to identify invalidated packages from materialization of a MOVE TABLE operation

Variable in the query	Value
object_qualifier	The schema of the table that is being moved
object_name	The name of the table that is being moved

Variable in the query	Value
object_type	Use one of the following values: **T** Normal table **M** Materialized query table

Tip: Unlike non-UTS table spaces, Db2 supports access to currently committed data in UTS. Applications that use a sequence of FETCH, DELETE, and INSERT statements in the same commit scope instead of an UPDATE statement might need to be modified to use UPDATE statements before the conversion to UTS. The reason is that a row that has been logically updated using DELETE and INSERT can re-appear in the FETCH result set before the application commits.
For more information about currently committed data access, see Accessing currently committed data to avoid lock contention.

Procedure

Attention: Before you run the REORG utility to materialize one or more pending MOVE TABLE operations, the following actions will cause the REORG to fail during the UTILINIT phase:

- Altering the target table space so that its attributes become invalid for a MOVE TABLE operation. In this case, use the ALTER TABLESPACE statement to alter any invalid attributes to be valid again.
- Dropping and re-creating the target table space, regardless of whether the table space attributes are valid. In this case, complete the following steps to re-execute the MOVE TABLE operation:
  1. Issue the ALTER TABLESPACE statement with the DROP PENDING CHANGES option to drop all pending definition changes for the target table space.
  2. Re-create the target table space with the desired attributes.

3. Issue the ALTER TABLESPACE MOVE TABLE statement again.

To move tables from a source multi-table table space, take one of the following actions:

- Move all tables from the source table space to partition-by-growth table spaces, and then drop the source table space.
  1. Issue the ALTER TABLESPACE statement with the MOVE TABLE option to move each table to a partition-by-growth table space.
  2. Run the REORG TABLESPACE utility with the SHRLEVEL REFERENCE or SHRLEVEL CHANGE option on the source table space to materialize the MOVE TABLE operations. All pending definition changes for the source table space are materialized in the same REORG. The source table space then becomes an empty table space.
  3. Issue the DROP statement to drop the source table space.

  > **Note:** Db2 removes any associated history and recovery information for the source table space. References to the source table space must be modified accordingly.

- Move all tables except the last table from the source table space to partition-by-growth table spaces, and then convert the source table space to a partition-by-growth table space. This approach does not require creating a target table space for the last table and saves two OBIDs.
  1. Issue the ALTER TABLESPACE statement with the MOVE TABLE option to move each table to a partition-by-growth table space. Leave the last table in the source table space.
  2. Issue the ALTER TABLESPACE statement with a MAXPARTITIONS value of 1 to convert the source table space to a partition-by-growth table space. This conversion is a pending definition change that can be materialized with any previous pending definition changes for the source table space.
  3. Run the REORG TABLESPACE utility with the SHRLEVEL REFERENCE or SHRLEVEL CHANGE option on the source table space to materialize the MOVE TABLE operations. All pending definition changes for the source table space are materialized in the same REORG.

  > **Note:** Db2 retains all associated history and recovery information for the source table space. References to the source table space must be modified to consider that the table space is now a partition-by-growth table space.

Results

If the moved tables remain empty after materialization, the REORG leaves the target table space as DEFINE NO (that is, no VSAM data sets defined) but it updates the metadata definition of the target table space to reflect the linkage to the moved tables. REORG does not insert any SYSCOPY records for target table spaces that remain as DEFINE NO after materialization of pending MOVE TABLE operations. These target table spaces will not be involved in any future recovery situations because their VSAM data sets do not exist.

What to do next

Packages that depend on the moved tables are invalidated by materialization of the MOVE TABLE operations. Existing table-level statistics still persist after you run the REORG, which results in minimal risk for access path regression from rebinds and autobinds of the invalidated packages. Nevertheless, you can take one of the following actions after you run the REORG to further mitigate risk of access path regression:

- Run a stand-alone RUNSTATS job.
- When you rebind the invalidated packages, issue the REBIND command with the APREUSE(WARN) bind option to minimize access path change. If enabled, an autobind also issues APREUSE(WARN).

After the MOVE TABLE operations are materialized, you can increase the MAXPARTITIONS value of the target table space to any valid value as an immediate change by issuing an ALTER TABLESPACE statement.

If you create partition-level image copies of the target table spaces for backup and recovery, and you use a LISTDEF list with the PARTLEVEL option for the COPY utility, you must also use a LISTDEF list with the PARTLEVEL option for the RECOVER utility.

Parent topic:

→ Altering table spaces

Related tasks

→ Creating partition-by-growth table spaces

Related reference

→ ALTER TABLESPACE

→ DROP