
Options for rebuilding indexes

When you clone table spaces, you might need to rebuild the indexes associated with the table spaces.

There are several methods for rebuilding indexes from which to choose. The table that follows provides a brief overview of each method. Refer to the detailed topics on each method for specific information about setting up index rebuilds.

Table 1. Types of index rebuilds

Type	Description	Considerations	Link?
Dynamic rebuild	Generates a REBUILD utility for each index or index partition, and runs the utilities at the end of the target job.	<ul style="list-style-type: none">• Good for when there are only a few non-partitioned table spaces and only one index on each table.• Generates a separate REBUILD INDEXSPACE command for each index.• Can report on and rebuild indexes that are only on the target.	"Method 1: Dynamic rebuild" on page 2
Intelligent rebuild	Groups together as many indexes as possible into a single REBUILD utility for a given table space, to minimize table read I/O and elapsed time, as well as take advantage of built-in multi-tasking capabilities in the DB2 REBUILD utility. Runs the utilities at the end of the target job.	<ul style="list-style-type: none">• More efficient when there are multiple indexes on tables.• Minimizes table read I/O and elapsed time by grouping indexes where possible.• Can report on and rebuild indexes that are only on the target.• If a CKZINTRB DD is supplied on the target job, a JCL version of the rebuild job stream is written to the specified file and can provide restart capability if the rebuild fails.	"Method 2: Intelligent rebuild" on page 2
Job templates	Job templates use JCL templates with variables to generate REBUILD JCL and write it to a specified file in the source or the target job. User intervention may be required before running the JCL.	Can provide restart capabilities, with manual intervention.	"Method 3: Using job templates in the source job" on page 3 or "Method 4: Using job templates in the target job" on page 4

Recommendations when rebuilding indexes

When rebuilding indexes as part of the cloning process, you may need to plan for restartability in case of rebuild failure. In addition, you might need to consider situations such as indexes that are only located on the source or only on the target, or whether you have partitioned objects that are split across multiple jobs.

Enabling restartability in the event of rebuild failure

With either dynamic or intelligent rebuild, an essentially identical JCL stream can be generated for potential use on a restart after a rebuild failure. In either case, follow this procedure to restart the rebuild after a failure:

1. Determine the problem that caused the index rebuild job to fail.

2. Resolve the problem that caused the job to fail.
3. If necessary, terminate the failed utility ID by using a `-TERM UTIL` command.
4. Edit the JCL that was generated by either job templates or intelligent rebuild to remove the indexes that were successfully built.
 - If JCL was generated by job templates, use the file specified by the `ddnameO` DD in the target job (the `ddname` is taken from the source job keyword `TARGET-JOB-INDEX-REBUILD-DDN(ddname)`).
 - If the JCL was generated by intelligent rebuild, use the file specified by `ddname CKZINTRB` in the target job.
5. Submit the modified index rebuild job manually.

Excluding source-only indexes

To exclude source-only indexes, but copy all indexes that exist on both source and target, use `ALWAYS-COPY-INDEXSPACES(Y)` and exclude the source-only indexes using `LISTDEF EXCLUDE`.

Rebuilding target-only indexes

With dynamic or intelligent rebuild, you can rebuild indexes that are only on the target by copying the related table space and specifying both `PROCESS-UNMATCHED-TARGET-INDEXES(Y)` and `REBUILD-UNMATCHED-TARGET-INDEXES(Y)` in the `SET` command of the source job.

Method 1: Dynamic rebuild

When using `LOG-APPLY` or data masking, or when cloning from image copies, you can use this method to rebuild indexes that are defined as `COPY NO` and to submit the index rebuild as part of the target job.

About this task

To use this method, all index spaces must be explicitly included in the `LISTDEF` or be included by specifying the `COPY` command `ALWAYS-COPY-INDEXSPACES(Y)`.

Procedure

1. Specify `SET` command `REBUILD-INDEXES-EXECUTE(Y)` in the source job.
2. Optional: If you want indexes that are defined as `COPY NO` to be rebuilt, specify the `COPY` command `REBUILD-COPY-NO-INDEXES(Y)` in the source job.
3. Submit the source job.
4. Submit the target job. The target job will dynamically rebuild the indexes.

Method 2: Intelligent rebuild

Intelligent rebuild groups together as many indexes as possible into a single `REBUILD` utility for a given table space. This minimizes table read I/O and elapsed time and takes advantage of built-in multi-tasking capabilities in the DB2 `REBUILD` utility. Like dynamic rebuild, it runs the utility at the end of the target job.

About this task

To use this method, all index spaces must be explicitly included in the LISTDEF or be included by specifying the COPY command ALWAYS-COPY-INDEXSPACES(Y).

Procedure

1. Specify SET command REBUILD-INDEXES-EXECUTE(Y) in the source job.
2. Specify SET command INTELLIGENT-REBUILD (Y) in the source job.
3. Optional: If you want to generate a JCL version of the INTELLIGENT-REBUILD job stream, include ddname CKZINTRB in the target job JCL. The JCL job can be used (with modifications) should a restart be required. The DD that is supplied must be LRECL=80 and RECFM=FB.
4. Optional: If you want indexes that are defined as COPY NO to be rebuilt, specify the COPY command REBUILD-COPY-NO-INDEXES(Y) in the source job.
5. Submit the source job.
6. Submit the target job. The target job will dynamically rebuild the indexes.

Method 3: Using job templates in the source job

When you want to rebuild indexes that are not affected by LOG-APPLY, data masking, or cloning by using image copies, the indexes can be rebuilt from a utility job that is generated in the source job by using job templates.

About this task

This method uses job templates and the JOB-TEMPLATE command to rebuild indexes.

Note: This method may not produce the desired results if the indexes are not explicitly included.

Procedure

1. Specify the COPY command JOB-TEMPLATE (*inddname1*, *outddname1*) in the source job.
2. For *inddname1*, make a copy of one of the following sample templates in the SCKZJCL library and modify it for your site. Instructions for updating the template are contained in the sample templates.
 - CKZJOBI: This template can be used if you specified the COPY command ALWAYS-COPY-INDEXSPACES (Y) in the source job or explicitly specified the indexes by using LISTDEF. The template generates index rebuilds with REBUILD INDEXSPACE *database.indexspace* syntax.
 - CKZJOBR: This template can be used when the indexes are not explicitly included in the cloning, either by using LISTDEF or by using the COPY command keyword ALWAYS-COPY-INDEXSPACES(Y), in the source job. The template generates index rebuilds with REBUILD INDEX (ALL) TABLESPACE *database.table_space* syntax.
3. Submit the source job. The utility JCL to rebuild the indexes is created and placed in *outddname1*, as specified in the JOB-TEMPLATE command.
4. To rebuild the indexes, submit the JCL in *outddname1*.

Method 4: Using job templates in the target job

When using LOG-APPLY or data masking, or when cloning from image copies, you can use job templates to create a job to rebuild the indexes. The job to rebuild the indexes is written to a member in the target job that you can submit.

About this task

This method uses job templates to rebuild indexes.

Note: This method may not produce the desired results if the indexes are not explicitly included.

Procedure

1. Specify the COPY command TARGET-JOB-INDEX-REBUILD-DDN(*ddname*) in the source job.
2. For *inddname1*, make a copy of one of the following sample templates in the SCKZJCL library and modify it for your site. Instructions for updating the template are contained in the sample templates.
 - CKZJOBI: This template can be used if you specify the COPY command ALWAYS-COPY-INDEXSPACES (Y) in the source job or explicitly specify the indexes by using LISTDEF. The template generates index rebuilds with REBUILD INDEXSPACE *database.indexspace* syntax.
 - CKZJOBR: This template can be used when the indexes are not explicitly included in the cloning, either by using LISTDEF or by using the COPY command keyword ALWAYS-COPY-INDEXSPACES(Y), in the source job. The template generates index rebuilds with REBUILD INDEX (ALL) TABLESPACE *database.table_space* syntax.
 - CKZJOBRN: This template can be used to generate partition-level rebuilds, using the automatic &PARTNUM variable. It can be used if you specify the COPY command ALWAYS-COPY-INDEXSPACES (Y) in the source job or explicitly specify the indexes by using LISTDEF. The template generates index rebuilds with REBUILD INDEXSPACE *database.indexspace* PART *partition_number* syntax.

3. Add the following two DDs to the target job for rebuilding indexes:

```
//ddnameI DD DISP=SHR,DSN=hlq.indsn(mbr)
```

where *ddname* is the DD that is specified in the TARGET-JOB-INDEX-REBUILD-DDN keyword. You must append an I to *ddname*. *hlq.indsn(mbr)* is the data set location and member name of the template that you edited.

```
//ddnameO DD DISP=OLD,DSN=hlq.outdsn(mbr)
```

where *ddname* is the DD that is specified in the TARGET-JOB-INDEX-REBUILD-DDN keyword. You must append an O to *ddname*. *hlq.outdsn(mbr)* is the data set location and member name where you want the utility job to be saved. This DD will contain the output from the generated REBUILD INDEX job.

4. Submit the source job.
5. Submit the target job. When the target job is run, the job to rebuild the indexes is written to *ddnameO*.
6. To rebuild the indexes, submit the JCL in *ddnameO*.