TUC Unique Features

1 Overview

This document is describing the unique features of TUC that make this product outstanding in automating the DB2 object maintenance tasks. The document is comparing the various features of TUC with competitive products that provide similar functionality.

2 Objectives

- Automate the generation of utility statements
- Ensure all objects are covered
- Identify candidates for utility processing
- Minimize resource consumption of utilities
- Exploit parallelism and balance workload
- Remove dependency on specific vendor
- Integrate utilities and applications
- Leave control in the hands of administrators
- Protect resources and follow conventions
- Increase administrators productivity

3 Concepts

TUC automatically identifies any newly created objects and refreshes the utility statements and jobs automatically without any need for manual action. New databases are automatically covered. In fact, you must explicitly exclude objects from automatic processing, if desired. Utility jobs are triggered by a set of rules that examine catalog and Real Time Statistics. This approach allow database administrators to set up the maintenance procedures once and allow TUC to automatically pick up any newly created object.

TUC approach to automation is focusing on integration of application jobs and utility jobs to avoid collisions or lack of availability and decrease impact on performance. The application batch jobs can trigger backups in business significant points in time. Executing utilities in predefined batch windows may not always fit current application activity that may change over time or due to unexpected events.

TUC prepares the utility statements and jobs before execution to avoid delays at run time. This approach also allows executing utility statements on a different system or on a remote site at disaster recovery. TUC generates utility statements at run time only when needed.

4 Features

		TUC	Other
1.	**Grouping objects**	Y	Y
1.1.	Grouping objects using generic notations	Y	Y
1.2.	Automatic grouping of newly created databases	Y	N
1.3.	Grouping objects using dynamic SQL expressions	Y	Y
1.4.	Grouping package dependent objects	Y	Y
1.5.	Grouping all candidates for utility execution	Y	N
1.6.	Grouping objects in balanced subgroups based on size	Y	N
2.	**Generating utilities**	Y	Y
2.1.	Generating utility statements	Y	Y
2.2.	Refreshing utility statements for new objects or new options	Y	N
2.3.	Allow defining utility options for the entire DB2 subsystem as global defaults	Y	N
2.4.	Allow defining utility options per group of objects	Y	N
2.5.	Reusing generated utility statements by repeatable executions	Y	N
2.6.	Support for multiple utility vendors in co-existence	Y	N
3.	**Exploit LISTDEF**	Y	Y
3.1.	Allow using an external library for LISTDEF statements	Y	Y
3.2.	Generating LISTDEF for all recoverable objects	Y	N
3.3.	Generating LISTDEF for full copy and incremental copy	Y	N
3.4.	Generating LISTDEF for related indexes	Y	N
3.5.	Generating LISTDEF at partition level and at object level	Y	N
3.6.	Generating LISTDEF for LOB and XML objects	Y	N
4.	**Generating jobs**	Y	Y
4.1.	Generating utility jobs ready for scheduling	Y	Y
4.2.	Generating utility jobs for immediate execution	Y	Y
4.3.	Allow flexible naming of job names using SQL expressions	Y	N
4.4.	Allow controlling job card	Y	Y
4.5.	Executing a quota of jobs	Y	N
4.6.	Mapping legacy utility jobs to identify groups of objects	Y	N
5.	**Triggering utilities**	Y	Y
5.1.	Trigger utilities based on Real Time Statistics	Y	Y
5.2.	Trigger utilities based on catalog statistics	Y	Y
5.3.	Trigger utilities based on SYSCOPY events	Y	N
5.4.	Trigger utilities based on user tables	Y	N
5.5.	Trigger utilities based on statistics trace access performance patterns	Y	N
5.6.	Trigger utilities for tablespaces only	Y	Y
5.7.	Trigger utilities for indexes only	Y	Y
5.8.	Trigger utilities for the entire DB2 subsystem	Y	Y
5.9.	Trigger utilities for a group of objects	Y	Y
5.10.	Trigger backups when updates are no longer recorded in the active logs to avoid accessing archive logs when attempting to recover.	Y	N
5.11.	Trigger REORG based on access performance counters GETPAGES per sync I/O rate decrease since last reorg.	Y	N

	TUC	Other
6. Setting Thresholds	Y	Y
6.1. Define default threshold per triggering rule	Y	Y
6.2. Define threshold per object using generic notations	Y	Y
6.3. Allow processing a quota of objects by number	Y	N
6.4. Allow processing a quota of objects by total size	Y	N
7. Prioritizing objects	Y	Y
7.1. Define priority by object name using generic notations	Y	Y
7.2. Define priority by threshold	Y	Y
7.3. Process objects in order of priority	Y	Y
7.4. Process jobs in order of priority	Y	Y
7.5. Process different lists of objects per priority within same job	Y	N
8. Excluding objects	Y	Y
8.1. Excluding objects by name using generic notations	Y	Y
8.2. Excluding objects by threshold	Y	Y
8.3. Excluding objects by size	Y	Y
8.4. Excluding objects in restricted states	Y	Y
8.5. Excluding empty objects	Y	Y
8.6. Excluding by priority	Y	Y
8.7. Excluding by rule relationships to avoid unnecessary REORG based on performance	Y	N
8.8. Report excluded objects and the reason for being qualified	Y	N
9. Displaying candidates	Y	Y
9.1. Displaying candidates per utility	Y	N
9.2. Displaying candidates per triggering rule	Y	N
9.3. Displaying computed value compared to threshold	Y	Y
9.4. Displaying statistics used to qualify object for processing	Y	Y
9.5. Displaying access performance statistics based on statistics trace records	Y	N
9.6. Allow immediate change of threshold	Y	N
9.7. Allow immediate change of priority	Y	N
9.8. Allow excluding candidate object	Y	N
10. Processing utilities	Y	N
10.1. Continuously process pending requests for utility execution	Y	N
10.2. Handing over jobs to the scheduler	Y	N
10.3. Allow users and jobs to place a request for utility processing	Y	N
10.4. Display status of requests for utility processing	Y	N
10.5. Display history of executions	Y	N
10.6. Allow defining authorizations for requesting utility processing	Y	N
10.7. Protect utilities from an unauthorized request	Y	N

	TUC	Other
11. Executing utilities	Y	Y
11.1. Executing up to 35 parallel jobs	Y	Y
11.2. Executing sync job waiting for parallel jobs to complete	Y	N
11.3. Allow restart from failing step	Y	Y
11.4. Allow rerun to exclude objects processed in previous run	Y	Y
11.5. Allow rerun to resize work datasets without re-blocking	Y	N
11.6. Track job status to identify failing jobs	Y	Y
11.7. Record jobs elapsed time for trend analysis	Y	Y
11.8. Executing online utilities in a worklist	N	Y
12. Space Management	Y	Y
12.1. Calculating space based on estimated rows	Y	N
12.2. Automatic analysis of DSN1COMP output to set compressed average row length	Y	N
12.3. Correcting estimated number of rows using RTS	Y	N
12.4. Resizing objects prior to reorg on the boundary of a cylinder	Y	Y
12.5. Allow resize down to release unused space	Y	Y
12.6. Adjusting free space based on overflows or splits	Y	N
12.7. Propagating estimated space to all partitions	Y	Y
12.8. Map location and size of all DB2 datasets using DCOLLECT	Y	N
13. Monitoring Growth	Y	N
13.1. Collect space growth per object using Real Time Statistics	Y	N
13.2. Calculate growth rate per day or per minute	Y	N
13.3. Predict when an object reaches maximum available space	Y	N
13.4. Allow triggering RUNSTATS based on growth for tables loaded with LOAD RESUME	Y	N
14. Creating utility objects	Y	N
14.1. Creating Online REORG Mapping Tables	Y	N
14.2. Creating CHECK DELETE exceptions tables	Y	N
14.3. Creating target tables based on LOAD specification	Y	N
14.4. Creating missing target auxiliary objects	Y	N
14.5. Altering target tables to add missing columns for LOAD	Y	N
15. Archiving data	Y	N
15.1. Defining discard conditions per table	Y	N
15.2. Resolving symbols imbedded in conditions at run time	Y	N
15.3. Allow complex predicates by using query results	Y	N
15.4. Recording discard datasets	Y	N
15.5. Reloading discarded data back to source tables	Y	N
15.6. Loading discarded data into history tables	Y	N

	TUC	Other
16. Unload and reload	Y	N
16.1. Unload from last image copy	Y	Y
16.2. Unload tables with LOB columns using lob files	Y	N
16.3. Unload tables with LOB columns using SPANNED filed	N	N
16.4. Presort data before reload using sort based on cluster index key columns	Y	N
16.5. Reload data to different target tables	Y	N
17. Editing datasets	Y	N
17.1. Display and edit discard and unload datasets	Y	N
17.2. Show column positions based on associated SYSPUNCH	Y	N
17.3. Allow updating rows using a friendly dialog	Y	N
17.4. Allow scrolling to the next row or skipping to a specific row	Y	N
17.5. Convert decimal and integer values to external format	Y	N
17.6. Adjust length for variable columns	Y	N
18. Rebind Packages	Y	Y
18.1. Automatically rebind all dependent packages after utility	Y	Y
18.2. Collect bind errors into repository for analysis	Y	N
19. Application syncpoints	Y	Y
19.1. Recording syncpoints with meaningful names	Y	N
19.2. Ensuring syncpoint recoverability	Y	N
19.3. Recording syncpoint datasets	Y	N
20. Recovery services	Y	Y
20.1. Recover to current for a group of objects	Y	Y
20.2. Recover by database	Y	N
20.3. Recover tablespace to a point in time	Y	Y
20.4. Recover tablespace to a quiet time	Y	N
20.5. Recover or rebuild index based on available IC	Y	Y
20.6. Recover an entire volume	Y	N
20.7. Recover a single table of a multi table tablespace	Y	N
20.8. Recover all objects included in a UR or LUW	Y	N
20.9. Identifying recover dependencies	Y	Y
20.10. Automate Mass-recovery LOGONLY	Y	N
21. Automating DSN1COPY	Y	Y
21.1. Generate DSN1COPY jobs using source datasets pattern	Y	N
21.2. Relate datasets to objects using template symbols	Y	N
21.3. Match source datasets to target objects by tablespace name	Y	N
21.4. Automatic translation of internal ids	Y	N
21.5. Checking compatibility source and target	Y	N
21.6. Defining missing linear VSAM pieces	Y	N
21.7. Adjusting maximum value for identity columns	Y	N
21.8. Executing DSN1COPY in a single step (worklist)	Y	Y
21.9. Match FLASHCOPY image copies	Y	N

	TUC	Other
22. Issuing display Commands	Y	Y
22.1. Display objects based on -DIS DB	Y	Y
22.2. Allow issuing stop and start commands	Y	Y
22.3. Allow cancelling a stop command if object is in STOPP status	Y	N
22.4. Display utility ids based on -DIS UT	Y	Y
22.5. Allow terminating or altering a utility	Y	Y
22.6. Allow identifying the utility job name	Y	N
22.7. Display active threads based on -DIS TH	Y	Y
22.8. Allow canceling a thread	Y	Y
22.9. Display logging status based on -DIS LOG	Y	N
22.10. Allow archiving the active log	Y	N
22.11. Allow issuing a system checkpoint	Y	N
23. Tracing	Y	N
23.1. Display active traces	Y	N
23.2. Allow starting a new trace, modifying an active trace and stop a trace	Y	N
23.3. Analyze SMF records and collect access performance statistics trace records	Y	N
23.4. Analyze SMF records and collect audit trace records	Y	N
24. Mapping logs	Y	N
24.1. Display the log inventory	Y	N
24.2. Display logging durations	Y	N
24.3. Identifying logs availability	Y	N
24.4. Rename logs and rebuild the BSDS	Y	N
24.5. Convert BSDS using DSNJCNVB	Y	N
24.6. Create conditional restart control record	Y	N
24.7. Allow executing log analysis using DSN1LOGP	Y	N
24.8. Record UR and LUW activity based on DSN1LOGP	Y	N
24.9. Display log RBA reset deadline	Y	N
24.10. Allow adding new active logs	Y	N
25. Automate MODIFY RECOVERY	Y	Y
25.1. MODIFY to retain last image copies	Y	Y
25.2. MODIFY to cleanup recording of deleted image copies	Y	N
25.3. Deleting image copies no longer recorded in SYSCOPY	Y	N
26. Scheduling services	Y	Y
26.1. Define CONTROL-M scheduling tables using CTMBLT	Y	Y
26.2. Define TWS job dependencies using batch loader EQQYLTOP	Y	N
26.3. Define DB2 administrative scheduler tasks	Y	N

	TUC	Other
27. Administrative scheduler	Y	N
27.1. Display DB2 administrative scheduler tasks	Y	N
27.2. Allow adding, updating and removing tasks	Y	N
27.3. Display task status	Y	N
27.4. Display task history	Y	N
28. Autonomic Statistics	Y	N
28.1. Defining autonomic statistics monitoring options	Y	Y
28.2. Calling the autonomic statistics monitor stored procedure	Y	Y
28.3. Scheduling tasks for autonomic statistics monitors	Y	Y
28.4. Displaying alerts created by autonomic statistics	Y	Y
28.5. Defining periods for autonomic statistics	Y	Y
29. Deploying definitions	Y	Y
29.1. Export specific definitions	Y	Y
29.2. Export definitions for the entire subsystem	Y	Y
29.3. Using symbolic variables in exported definitions	Y	N
29.4. Manipulating definitions before import	Y	N
29.5. Importing definitions	Y	Y
29.6. Comparing definitions to identify discrepancies	Y	N
29.7. Unload and reload definitions for cloned subsystems	Y	N
29.8. IVP jobs to verify correct definitions for all functions	Y	N