Global Productivity Solutions

DQDBArchive Release Notes

Subject:	DQDBArchive Release Notes
Date:	02/04/2021
Version:	1.0

DQDBArchive v7.1.18

New version available in downloads - DQDBArchive v7.1.18 (20210204)

1. Purge Tables – added SQL Scheduler History table as option.

DQDBArchive v7.1.17

New version available in downloads – DQDBArchive v7.1.17 (20200415)

1. Purge Tables – new option to delete records from certain tables based on a time frame different than the Clean Trips time frame.
    a. Found under Properties -> Clean Tables tab.
    b. It will run once before the clean orphans and clean trips processes run.

DQDBArchive v7.1.15

New version available in downloads - DQDBArchive v7.1.15 (20180209)

1. Ignore violation errors. If a constraint error is encountered for a specific row, that row will be skipped and not copied over to the archive table and the archiving process will be able to continue on. However, this doesn't prevent the order record from being archived. For example, if a job record is not copied over because it already exists in the archive table with a different pkey, the order record can still be archived. That's because there isn't a parent-child relationship on the order_tbl to the customerjobaccount_tbl, so it's possible to set a non-existent job account on an order record.
2. Disable the ability to clean up customer and job orphans. Customer and Job records get removed from the database because all of their orders have been archived. But then, later, they import another order for that customer or job and then those records are recreated with different pkeys, then when it tries to copy them over to the archive, it can't because of these constraints. So, I'm disabling the ability to clean orphans for these two tables.

DQDBArchive v7.1.14

1. When certain support tables are copied over to the archive db, they should be deleted first, and then copied. This in effect will make the archive table that same as the prod table. Otherwise, constraints on the table can be violated.
    a. The smart_deviceGroup_tbl is one of these tables.

DQDBArchive v7.1.12

1. Remove orphan option added for RoutedDistance_tbl. If the parent record (order, location or job) for a routed distance no longer exists, the record will be removed.
2. New option – 'Delete/Archive VRPSolve tables (using above settings)'. If selected, then the VRPSolve_tbl will be archived (or deleted) according to the archive trip settings. That means that if a trip record is being archived due to the selected trip date range, then VRPSolve_tbl records will be archived based on the same tripdate range. For example, if a trip with a trip date of 2/1/2014 is being archived, then any VRPSolve_tbl records with a startdate of 2/14/2014 will also be archived.
3. Large tables can now be archived without causing memory issues. Support tables, like RouteDistance_tbl, may have 5 million plus rows. These tables will be copied over in chunks of 100,000 rows at a time. This value can be changed in the Admin.

DQDBArchive v7.1.6

1. Feature fix: The Delete Orphans routine would only run when the Clean and the archive options were also selected. Now, they can be run without those options being checked.

DQDBArchive v7.1.12

1. Updated the WT alternative order import to work for Turtle. Basically, this just meant getting some additional fields from the oe record instead of the wt record.

DQDBArchive v7.1.5

1. Signatures:
    a. New option added to delete orphaned records from the ordersignatures_tbl and signatures_tbl tables.
    b. Also, these tables are no longer copied over in full as a support table, they are archived, since they are so large.
2. Backup folder no longer required if the number of backups to keep is set to 0.

DQDBArchive v7.1.4

1. Added windows authentication as an option to connect to databases.
2. These are the permissions required in order for DBArchive to run correctly:
    a. An sql or windows account set up with sysadmin (sa) permissions.
    b. A windows account set up with the following permissions:
        i. Windows permissions:
            1. Local machine administrator.
            2. Set the DBArchive windows service to run under this account.
        ii. SQL Server permissions:
            1. DBOwner (db role on both main db and archive db).
            2. DBCreator (server role).
            3. DBArchive will not be able to delete backups (which requires sysadmin permissions), so the option 'Number of backups to keep' must be set to -1. Then a separate sql server maintenance plan should be set up on the server to delete backups.

DQDBArchive v7.1.2

1. Many improvements and bug fixes to the archiving routines.
2. If you select 0 for the Number of backups to keep, then the backup routine will not run. This may be useful for testing, but is not recommended to be set to 0 when this is set up to run in production.
3. The db backup timeout is now set according to the database timeout settings.
4. Recovery and db script version are now shown for the Delivery db and the archive db.
5. It's not necessarily recommended to clean up orphaned jobs if they want to keep jobs that are not associated to an order record.

DQDBArchive v7.1.1

1. The number of backups to keep can now be specified.
    a. If there are more backups than the specified number, they will be deleted. Oldest deleted first.
2. The recovery model is now indicated in the database name.
3. 'Shrink DB' and 'Rebuild indexes' are now different options instead of being combined together.
4. Shrink DB will not run unless the recovery model is 'Simple'
5. Rebuild indexes will not run unless recovery model is 'Simple'
6. The number of index pages no longer limits whether the index is rebuilt/reorganized. The limit used to be 1000 pages, but that constraint has been removed.
7. The DQScheduler control was updated to latest version. The older version has bugs in it.

DQDBArchive v7.1.0

Prerequisites – these two apps need to be installed in order for this app to work. They are included in the download folder:

1. SQLSysClrTypes.msi
2. SharedManagementObjects.msi

DQDBArchive can perform certain schedulable maintenance functions on a database:

1. Removing/Archiving old trips and orders
2. Removing parentless or childless (orphaned) records
3. Shrinking the size of the database and rebuilding table indexes

Features:

1. Multiple databases can be set up and processed (sequentially)
2. Routines are schedulable.
3. Right now, there is one routine, Clean Database. Basically, this routine can be configured to perform multiple maintenance functions against the database every time it runs:
    a. The first thing that is done, before any records are deleted, is to take a full backup of the database and store it on the server in the specified backup folder. Each backup is timestamp marked so that the database can be rolled back to the time before this routine ran.
    b. Next, if archiving is turned on:
        i. The 'Delete Orphans' routine is run against the prod database, which removes orphaned records from the selected tables. The 'Delete Orphans' routine is also run after the trip records have been processed, but they're done here before the trip records are processed so that orphaned records, which will be deleted anyway, are not needlessly copied over to the archive database first.
        ii. Then, all "support" tables are copied from the prod database to the archive database. Because there are so many intricate relationships between various tables, it's too tedious to try and archive only the individual table rows based on their relationships to other tables. So, instead, all relatively small tables (support tables) are simply "copied" from the prod database to the archive database. "Copied" means that if the row already exists in the archive database it is updated and if it doesn't exist, it is inserted, but rows are never deleted. This means that for a given support table, the archive database will have all the records the production database has, plus possibly more that were subsequently deleted in the prod database because they were no longer referenced. But, they are kept in the archive database because they are probably still referenced by other archive tables. The non-support tables that are not copied during this routine are the larger tables, whose rows will be individually archived during the archiving routine.
    c. Next, trips (and related records) are deleted and archived. The tables that are deleted/archived are the large tables in the database, whose records take up the most space. These include the trip_tbl, order_tbl, orderdetail_tbl, ordercommsoft_tbl, tripaudit_tbl, orderaudit_tbl, statushistory_tbl, builder_tbl, ordertransfer_tbl and ordertransferlog_tbl. Also, these tables are directly relatable to trip records so that specific records can be removed from the table without causing issues in other tables.

i. Based on the selected company and locations, all trips with trip dates prior to the specified trip date are queried from the production database.

ii. These trips are then processed in batches of 100 records. For each batch of trips:

    1. The trips (and related records) are first archived (if archiving turned on). If the record being archived already exists in the archive table, it is not inserted or updated.
    2. Then, the trips are deleted.

iii. After all trips have been processed:

    1. The Remove Orphans routine is run against the prod database and the archive database (if archiving turned on).
    2. The shrink database routine is run against the production database. When large amounts of data are removed from the database during the delete and archive routines, there may be large pockets of freed up space in the database. Running the shrink database can recover that lost space back to the OS. Shrink only works for databases set with a Recovery Model of 'Simple'. If the database is set to anything other than 'Simple', the shrink and rebuild index routines will not be run.
    3. The rebuild indexes routine is run against production tables. After shrinking a database, it is recommended to rebuild or reorganize table indexes that have become excessively fragmented due to the shrink. Each index on each table is analyzed and if the index has some fragmentation but not a lot of fragmentation, it is reorganized. If it has a lot of fragmentation, it is rebuilt. During rebuilding, if the version of SQL server is an Enterprise or Development version, then the index can be built online while the table is being used. However, if it is not an Enterprise or Development version, then the index must be built offline which means the table is not be accessible during that time. This adds more reason as to why this routine should be run during non-business hours.
    4. The shrink database routine is run against the archive database. There should not be many pockets of free space in the archive database, so this may not be necessary.
    5. The rebuild indexes routine is run against archive tables. Not sure how much the indexes will get fragmented during archiving, but it should not hurt to run this.

Notes:

1. Archive database must be at same db script version as prod database. A check is made during setup and when the routine runs. If the db version dates do not match, an error is thrown and the routine will not run. So every time a prod database is updated, the archive db must also be updated or the next time this routine runs, it will throw an error and fail.
2. The archive database can be a database on a different server and even possibly be a different version of Sql Server.
3. The archive and delete routines are done within a transaction so that if an error occurs, changes will be rolled back. For example, if an error occurs while deleting trip records, related order records will not be deleted either. Since this can lock table records and delete records while running, it's best to schedule this routine to run after business hours. Depending on the amount

of data needed to be archived or deleted, the process may take from 1 to several hours to complete. So it could even be scheduled to start Friday night and could go into Saturday to complete if necessary. It should be ok to kill the process should it still be found running the next start of business. It would pick up from where it last was the next time that it runs.

4. Before the archive and delete routines run, all triggers and check constraints are disabled on all tables in the database. This is to speed up the copying of data to the tables any also prevents triggers from unnecessarily firing as data is inserted or updated.
5. SMO, which is used to run the bulk of database procedures in this app often opens up multiple connections to the database but should close all those connections within about 5 minutes of this routine finishing.
6. If wanting to archive data, here are a couple of options to create the archive database.
    a. Copy the production database and restore it to another database and/or server. Then delete out these tables: trip_tbl, order_tbl, orderdetail_tbl, ordercommsoft_tbl, tripaudit_tbl, orderaudit_tbl, statushistory_tbl, builder_tbl, ordertransfer_tbl and ordertransferlog_tbl. When the routine runs, most tables will already be copied over and only the records that need to, will be copied from these tables. Then, when the shrink runs on the archive db, a bunch of space will be freed up.
    b. Create a new, blank database on the server and then create a db script to get it up to the same db version. When the routine runs, all support tables will be copied over, even lookup tables. The only table whose data will need to be manually entered is version_tbl. This is the only support table that is not copied.