Database at risk (Logs)

REASON

SQL Server databases are primarily backed by two categories of files – data files and log files. The data files contain all of the data related to the database, including tables, indexes, and object definitions, while the log files contain a log of all of the changes which have been made to that data. This separation of log and data files is critical to ensuring the recoverability of the data in the case of an unexpected server failure.

RESOLUTION

Under normal circumstances, the space in the log file is regularly reused, with space being made available for reuse through a process called log truncation . When log truncation cannot occur for some reason, the log file will attempt to grow if autogrow is enabled and the file has not reached its maximum size. When a log file becomes full and can no longer grow, attempts to modify or add data to the database will produce an error: the database essentially becomes read-only.

When a database log file is full, it is important to identify what is preventing log truncation from occurring. Two common reasons that the log might not be truncating include:

- For databases which are operating in the full or bulk logged recovery model, the log file will grow indefinitely until a backup is taken. In this case, you need to back up your log file and either schedule regular backups to continue, or consider changing the log truncation of your database to better fit your needs.
- For databases of any recovery model, a long running transaction will prevent the log from being truncated. You may need to identify and either commit or kill the offending transaction, or in the case of an important business transaction, you may need to grow the log file to allow it to finish.

There are other reasons truncation may not occur: full details are available here.

If you are not immediately able to resolve the problem that is preventing log truncation, you may need to take another action to create space for the log file. This can include freeing disk space, manually increasing the log file size, or adding an additional log file. Each of those options, along with their implications, is explained here, and can be summarized thus:

- Back up the database log
- Free disk space to allow autogrow
- Manually increase the size of the log file, in cases where autogrow is disabled
- Move the log file to a new, larger disk
- Add an additional log on a different disk
- Complete or kill the longest running transaction

It's a good practice to keep an eye on the sizes of your log files and the space remaining and to take action before reaching a critical stage. You can read more about managing transaction log sizes here.