Can't delete (doesn't exist) can't create new (already exist)

Last updated by | Vitor Tomaz | Jan 4, 2023 at 6:15 AM PST

Contents

Issue

Customer can't create new database, getting an error saying that the database already exists; nor can they delete a database - error database doesn't exist.

Investigation/Analysis

PS: There might be other reasons, this TSG is focused on backup queue preventing a new database from being created or preventing the deletion of existing database.

Backup design

By design, the first full backup is scheduled immediately after a new database is created or restored, or after backup redundancy changes. This backup usually finishes within 30 minutes, but it can take longer when the database is large.

Also, by design, when a database drop is requested, the remaining tail-end of the transaction log is backed up synchronously before the drop operation can finish. This is to ensure that backups are taken at different intervals to preserve the data so that it would be possible to restore later within the given retention period.

If you try to create or restore a database, and then drop it immediately before the full backup completes, you will not be able to drop the database and create another database with that same name until the full backup is taken (that's by design for create/restore new database). Depending on the size of the database, the full backup could take hours. Once the full backup (for the create/restore) completes, the drop request will initiate a tail-log backup for the same database before it can proceed to drop.

It's not possible to create a database with the same name until the tail-log is backed up and the drop operation completes.

Customer scenario

In some scenarios, the tail-log backup takes more time than expected. Some of the reasons are:

- There are several databases being created and drop in a short period of time.
  - The sequential nature of the drop operation puts databases dropped in short succession into a queue, which can prolong the process of dropping the databases.
  - Until the tail-log backup is executed, customer will not be able to create a new database with the same name. If there are more backup requests on the queue before the tail-log backup of the database that the customer is interested in, they will have to wait for all the backups on the queue to be executed.
- Customer restored a big database and shortly after it was available customer requested the database to be dropped.
  - After the database is created/restored, there will be a full backup request for it placed on the queue. The time this full backup will take, will depend on the requests already placed on the queue and the database size. If the drop request is placed shortly after the database is available, then customer will have to wait for the full backup and the tail-log backup to finish before they are able to create a new database with the same name.

Analysis

Confirm the drop attempt via ASC --> Provisioning --> Management Operations and collect the request_id

Based on the request_id of the drop request, we can confirm the timestamp of the drop request.

```
MonManagement
| where originalEventTimestamp >= {startTime} and originalEventTimestamp <= {endTime}
| where request_id =~ '<drop request_id>'
| where action == 'MarkDatabaseDropped'
| where state_machine_type == 'ManagedDatabaseStateMachine'
| project originalEventTimestamp, request_id, event, operation_type, elapsed_time, operation_parameters, opera
```

Check the current state of database. The state 'TailLogBackupInProgress' confirms that the tail-end of the transaction log is being backed up.

Below kusto will show the Backup start timestamp and if the backup is failing for some reason:

```
let managed_database_id = '<logicalDatabaseGuid>';
let serverName = '<MIName>';
MonBackup
| where TIMESTAMP >= ago(5d)
| where LogicalServerName =~ serverName
| where tolower(logical_database_id) =~ tolower(managed_database_id)
| where event == 'database_backup'
//| where backup_type == 'Full'
| project TIMESTAMP, AppName, event_type, br_exception_message, logical_database_id
```

Below kusto will show backup progress at every 10% interval:

```
let BACKUP_START_TIMESTAMP = ago(5d));
let managed_database_id = '<logicalDatabaseGuid>';
MonSQLSystemHealth
| where TIMESTAMP >= BACKUP_START_TIMESTAMP
|where message contains strcat("Backup(",managed_database_id) and message contains "processed"
|project TIMESTAMP, message
```

Mitigation

INTERNAL ONLY - START

On an average, the backup speed is about 300GB/h

INTERNAL ONLY - END

If the backups are progressing as intended (based on the kusto queries and approx backup speed above), the only mitigation is for the customer to wait for the tail-log backup to finish. Do not raise an ICM in such scenario.

Customer can follow the progress of the backups using the following T-SQL script: sp_mibackupinfo

Only raise ICM **IF** any one of the below conditions are met:

- Customer is facing tight deadlines during their migration window and do not need the taillogbackup to be taken.
- Impacting customer Production environment.
- Backups are failing or restarting or get interrupted or shows exceptions.

Public Doc Reference

- FAQ - Can I drop and recreate a database on a managed instance using the same database name?
- When deleting a database, the service takes a final transaction log backup before deletion, to prevent any data loss.
- The first full backup is scheduled immediately after a new database is created or restored.

How good have you found this content?