

# Resolve bulk insert issues

Last updated by | Peter Hewitt | Dec 16, 2022 at 11:10 AM PST

---

## Contents

- [Customer solution](#)
  - [Resolve bulk insert issues in Azure SQL Database](#)
  - [Common issues and solutions](#)
  - [Resources](#)

## Customer solution



This is the customer solution article for resolving SQL Database bulk insert issues that's displayed to customers in the Azure portal when creating a support ticket.




## Resolve bulk insert issues in Azure SQL Database

Bulk insert problems in SQL Database are commonly due to performance, Azure Blob storage access issues, incorrect configuration settings, and other factors.





Use the following guidance and best practices when using the `BULK INSERT` Transact-SQL (T-SQL) statement, the bulk copy program utility (BCP), and the `OPENROWSET` bulk insert T-SQL command to import data to SQL Database.

## Common issues and solutions

Issue	Recommended solution
Slow data import	<p>Consider adding a <code>TABLOCK</code> query hint to the BULK INSERT statement, the BCP command, or OPENROWSET insert statement. The <code>TABLOCK</code> query hint specifies that a bulk update table level lock is acquired for the duration of the data load operation instead of a row-level lock. This can significantly improve performance, as holding this lock for the duration of the load reduces table lock contention. For more information how to specify a table hint, see <a href="#">Using a Table Hint as a Query Hint</a> .</p> <p>Test various batch sizes with your data load to find out what works best. By default, all the rows in the data file are imported as one batch. To distribute the rows among multiple batches, specify a batch size that is smaller than the number of rows in the data file. If the transaction fails for any batch, only insertions from the current batch are rolled back. Batches already imported by committed transactions are unaffected by a later failure.</p> <p>When loading data into an empty table, drop or disable all indexes on the table. After the load is completed, re-create or re-enable the indexes. Importing data into a table that already contains data (a non-empty table) is known as an incremental bulk import. The key question for an incremental bulk import is whether indexes should be dropped beforehand. The options are to either keep the indexes, or drop them and recreate them afterwards. When importing data into a non-empty table, whether to keep the indexes depends on the amount of new data imported relative to the amount of existing data in the table:</p> <ol style="list-style-type: none"> <li>1. If importing a small amount of new data relative to the amount of existing data, dropping and rebuilding the indexes may be counter productive. The time required to rebuild the indexes is likely to be longer than the time saved during the bulk operation.</li> <li>2. In contrast, if importing a relatively large amount of new data, dropping the indexes on the table before performing the bulk operation can increase performance, without substantially increasing the time required for indexing.</li> </ol> <p><a href="#">Scale the database</a>  to a higher service tier, and compute size, to maximize data loading speed. After the import is complete, scale down as needed.</p>
Error accessing Azure Blob Storage file	<p>Error 5 (Access Denied): This error occurs when the desired file to be imported in Azure Blob Storage can't be opened. A shared access signature (SAS) is used to access the storage account. Confirm that the</p>

Issue	Recommended solution
	<p>SAS identity token is valid, and that access to the account isn't blocked by a storage account firewall. See <a href="#">Delegate access with shared access signatures (SAS)</a>  and <a href="#">Configure Azure Storage firewalls and virtual networks</a>  for more information.</p>
T-SQL command syntax errors	<p>The <code>EXTERNAL DATA SOURCE</code> should be created with type <code>BLOB_STORAGE</code> and point to the URL of the blob storage file container. The <code>DATA_SOURCE</code> is required in the <code>BULK INSERT</code> command and the <code>OPENROWSET</code> function for importing and reading the contents of the file respectively. In addition, the target file must already exist in the blob storage.</p> <p>Confirm the SAS credential token in the <code>SECRET</code> option of the <code>CREATE DATABASE SCOPED CREDENTIAL</code> T-SQL statement is valid. The most common errors in the T-SQL SAS token parameters:</p> <ol style="list-style-type: none"> <li>1. The question mark isn't removed from the beginning of the SAS token. The Azure portal generates an SAS token with a leading question mark. Remove this character as it's not required.</li> <li>2. <code>se</code> (expiry date) property is set to some value in the past (note this is in UTC time).</li> <li>3. <code>st</code> (start date) property is set to a value in the future (note this is in UTC time).</li> <li>4. <code>sp</code> (permission) property should allow reading the file on the storage account.</li> <li>5. <code>sip</code> (ip range) parameter isn't required, so remove this from the SAS token (if present).</li> </ol> <p>Refer to the example, <a href="#">Import data from a file in Azure Blob Storage</a>  for more details.</p>
Unsupported scenarios	Private IPs for blob storage and service endpoints aren't supported.

## Resources

- [BULK INSERT](#) 
- [BCP](#) 
- [Using OPENROWSET with the BULK Option](#) 
- [Examples of bulk access to data in Azure Blob storage](#) 

## How good have you found this content?

