DATA( ) {
EXPOSED;

DATA EXPOSED SPECIAL

Around the Clock with Azure SQL and Azure Data Factory

Americas
February 3, 2021
09:00 - 17:00 PT

Asia
February 4, 2021
09:00 - 17:00 SGT

HOSTED BY
Wee Hyong Tok & Anna Hoffman

16 Sessions | 2 Ask the Expert Panels | 1 Hackathon

DATA() {
EXPOSED;

Best practices using Azure SQL Database as Sink in ADF

Silvano Coriani – Program Manager, Azure SQL

Agenda

- Azure SQL Database connector
- Security
- Copy Activity
- Best practice for loading data into Azure SQL Database
- Scenarios
    - Append only
    - Upsert (insert new rows / update existing rows)
    - Custom bulk load logic
- Mapping data flows

Azure SQL Database connector

Connectivity "from and to" Azure SQL Database for:

- Copy activity
- Mapping data flow
- Lookup (retrieve a dataset to pass to subsequent activities)
- GetMetadata

Security

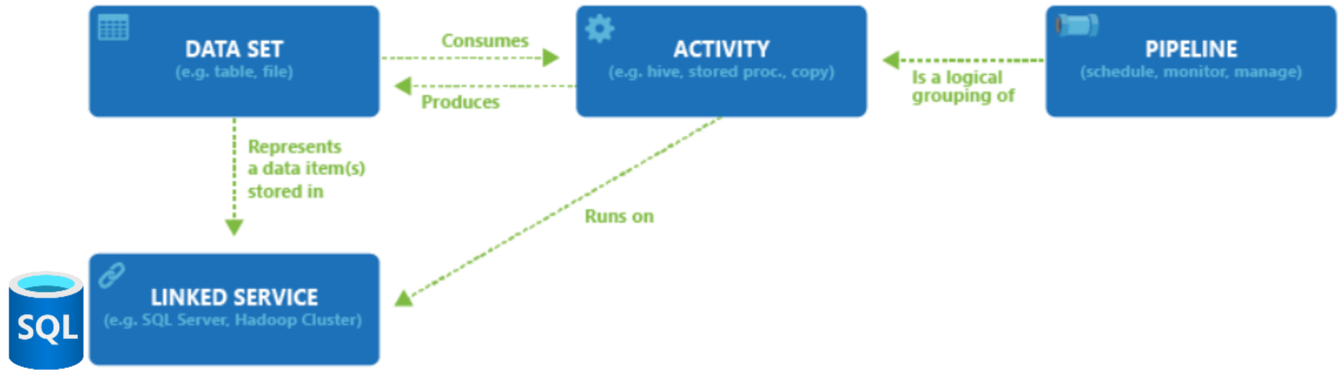
- Supports SQL authentication and Azure Active Directory
- Service principal and Managed Identity

As Source

- Query
- Stored procedure
- Parallel copy

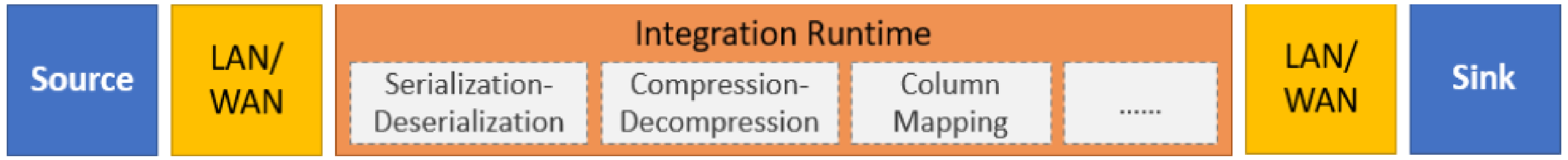
As Sink

- Uses bulk load where possible
- Create destination table if not exists
- Invoke stored procedure with custom logic during copy


DATA SET
(e.g. table, file)
Consumes
Produces
ACTIVITY
(e.g. hive, stored proc., copy)
Is a logical grouping of
PIPELINE
(schedule, monitor, manage)
Represents a data item(s) stored in
Runs on
SQL
LINKED SERVICE
(e.g. SQL Server, Hadoop Cluster)

Copy Activity

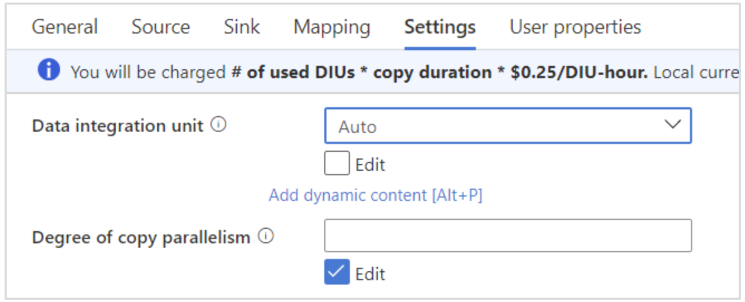
- Copy Activity in Data Factory copies data from a source data store to a sink data store.



- Uses the Integration Runtime specified at the dataset level
  - Can be in Azure or Self-hosted
  - Can use a managed VNET and connect to Azure SQL through Private Link
- Azure SQL Database as source
  - Leverage partitioning and parallel copy
- Consider retry / retry interval for robust copy execution
  - Mandatory for Serverless tier
- Other data transformation activities can target Azure SQL Database as sink
  - Azure Function
  - Stored Procedure
  - Data Flow
  - Databricks Notebook

Best practice for loading data into Azure SQL Database

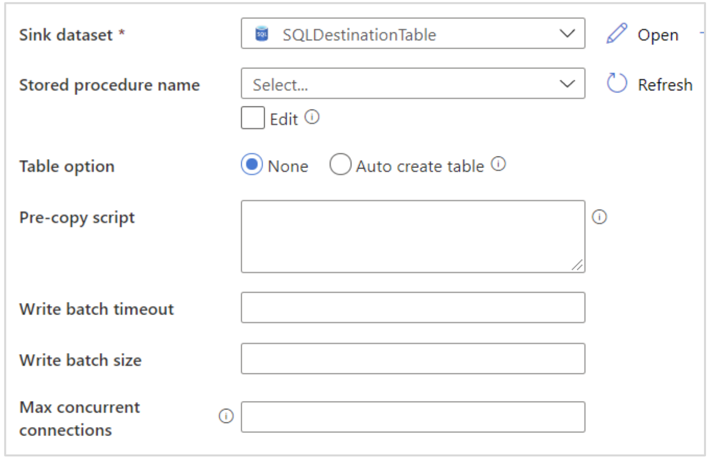
- Consider DIU and parallelism settings
  - Adjust (up to 32) to get required throughput/cost ratio



You will be charged **# of used DIUs * copy duration * $0.25/DIU-hour.** Local curre

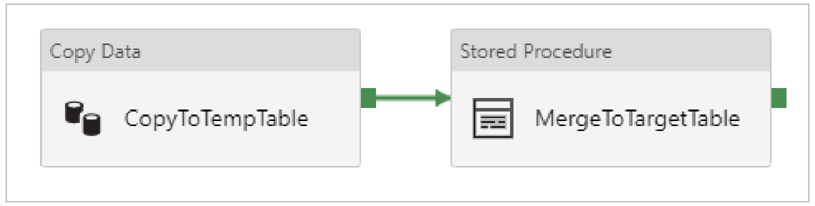
Data integration unit	Auto; [ ] Edit; Add dynamic content [Alt+P]
Degree of copy parallelism	[✓] Edit

- Append data -> uses bulk insert
  - Can control attribute like batch size and max concurrent connections



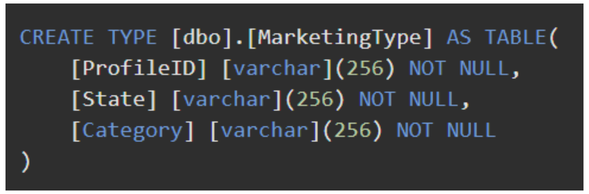
Sink dataset *	SQLDestinationTable
Stored procedure name	Select...; [ ] Edit
Table option	[● None] ○ Auto create table
Pre-copy script	
Write batch timeout	
Write batch size	
Max concurrent connections	

- Upsert -> bulk load in staging table and use MERGE or INSERT/UPDATE
  - Recommended for very large datasets
  - Consider a #temptable if writelog is an critical

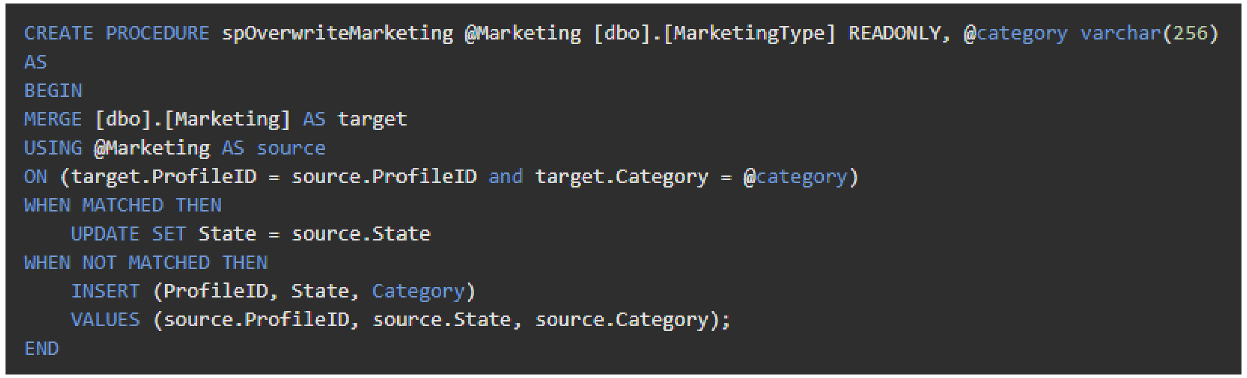


Best practice for loading data into Azure SQL Database

- As an alternative, you define a Stored Procedure, invoked once for each batch and that receives a TVP containing the rows to insert
  - Recommended for mid-large datasets and for complex insert logic where you want to control all aspects of ingestion (e.g. control TABLOCK, etc.)

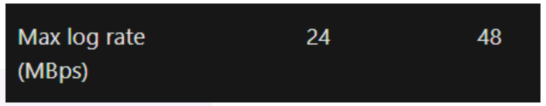


```sql
CREATE TYPE [dbo].[MarketingType] AS TABLE(
    [ProfileID] [varchar](256) NOT NULL,
    [State] [varchar](256) NOT NULL,
    [Category] [varchar](256) NOT NULL
)
```

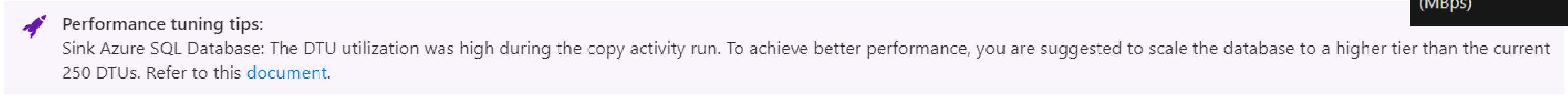


```sql
CREATE PROCEDURE spOverwriteMarketing @Marketing [dbo].[MarketingType] READONLY, @category varchar(256)
AS
BEGIN
MERGE [dbo].[Marketing] AS target
USING @Marketing AS source
ON (target.ProfileID = source.ProfileID and target.Category = @category)
WHEN MATCHED THEN
    UPDATE SET State = source.State
WHEN NOT MATCHED THEN
    INSERT (ProfileID, State, Category)
    VALUES (source.ProfileID, source.State, source.Category);
END
```

- Consider using proper Service Tier and compute size, check [resource limits (e.g. max log rate)](). All data loading best practices and techniques (e.g. indexing, partitioning, etc.) are still relevant!



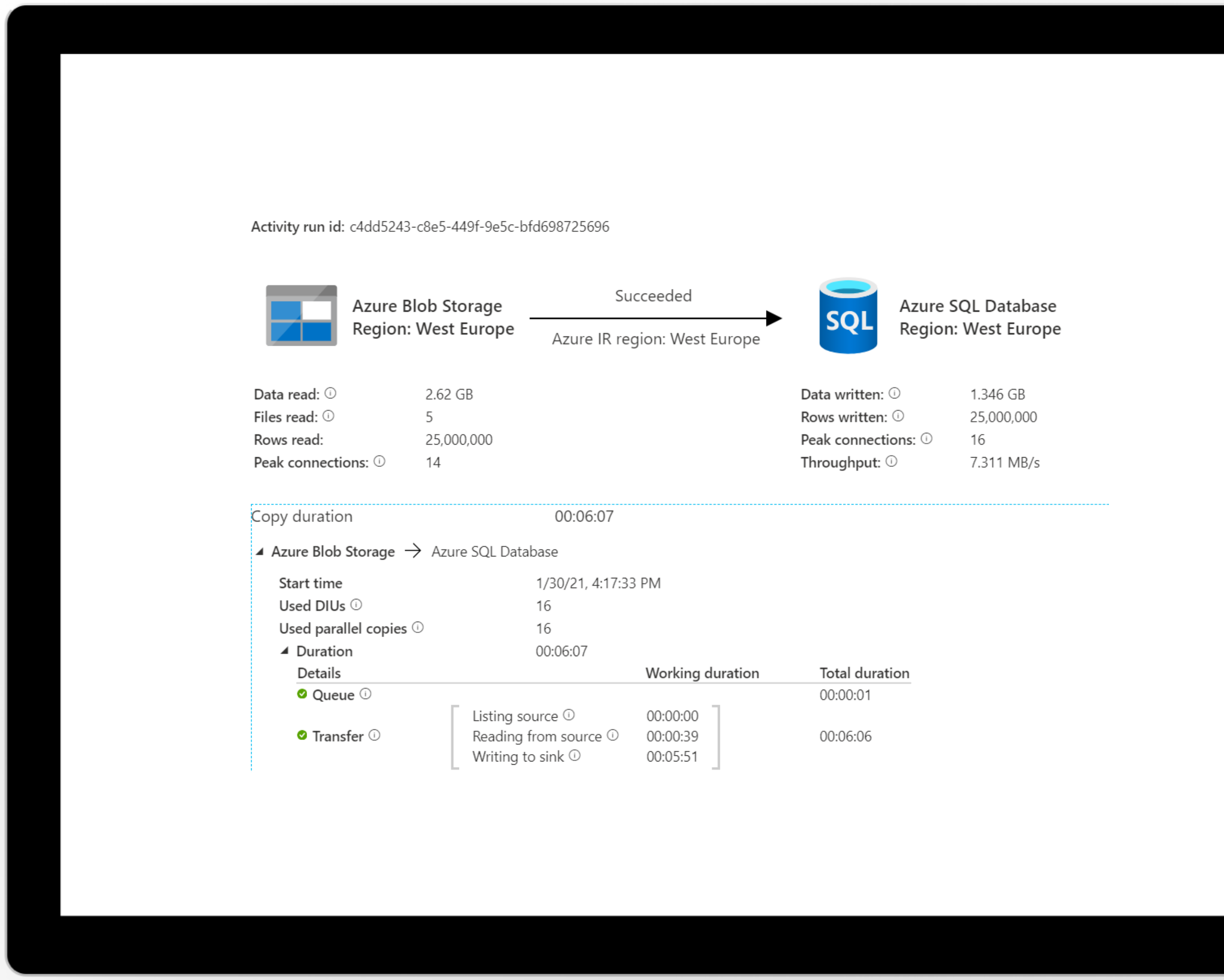
Max log rate (MBps)	24	48



Performance tuning tips:
Sink Azure SQL Database: The DTU utilization was high during the copy activity run. To achieve better performance, you are suggested to scale the database to a higher tier than the current 250 DTUs. Refer to this document.

```sql
SELECT primary_max_log_rate,primary_max_log_rate FROM sys.dm_user_db_resource_governance
```

Demo: optimize data loading in Azure SQL Database



Activity run id: c4dd5243-c8e5-449f-9e5c-bfd698725696

Azure Blob Storage
Region: West Europe

Succeeded
Azure IR region: West Europe

Azure SQL Database
Region: West Europe

Data read:	2.62 GB
Files read:	5
Rows read:	25,000,000
Peak connections:	14

Data written:	1.346 GB
Rows written:	25,000,000
Peak connections:	16
Throughput:	7.311 MB/s

Copy duration 00:06:07

Azure Blob Storage → Azure SQL Database

Start time	1/30/21, 4:17:33 PM
Used DIUs	16
Used parallel copies	16
Duration	00:06:07

Details		Working duration	Total duration
Queue			00:00:01
Transfer	Listing source	00:00:00	00:06:06
	Reading from source	00:00:39	
	Writing to sink	00:05:51	

Mapping Data Flow

Visual data transformation engine based on scaled-out Apache Spark clusters

Recommended for large and complex data transformation tasks

Can read and write to Azure SQL Database tables

As source

- Define a read query (can use UDFs)
- Can define attributes like Batch size and Isolation level

As sink transformation

- Define what operations are allowed (insert, delete, upsert, update)
- Define a key column to determine which row to alter (support composite keys)
- Table action (recreate, truncate, none)
- Batch size (trade off between insert efficiency and memory usage)
- Use TempDB: select between using a global temp table or a persisted table

Error handling

- Control if fail or continue on error
- Determine transactional behavior (single transaction or batches)
- Output rejected data to Azure Blob Storage or ADLS Gen2

Recap

- Azure Data Factory is a great option for creating data movement and transformation solutions from and to Azure SQL Database.
- Copy Activity is indicated for most common data movement scenarios like Append, Upsert and custom logic.
- Understand Data Integration Unit and parallelism impact is critical to get maximum performance during copy operations.
- Prefer bulk insert into staging tables followed by T-SQL based transformations (ELT-like) to get best performance in append or upsert scenarios.
- If complex logic during inserts is still required, leverage invoking Stored Procedure and TVPs and evaluate proper batch size (depending on table size, optimal could be between 50k and 250k rows).
- Consider all usual Azure SQL Database best practices for schema and indexing optimizations (e.g. partitioning, clustered columnstore, etc.)
- Azure SQL Database capacity planning principles will help setting the stage

DATA( ) {
EXPOSED;

Learn with us!

View our on-demand playlist:
aka.ms/azuresqlandadf

@AzureSQL
@AzDataFactory

Microsoft Azure