Lesson 0

Skills Tested

* Design and implement data storage (40–45%)
* Design and develop data processing (25–30%)
* Design and implement data security (10–15%)
* Monitor and optimize data storage and data processing (10–15%)

Study Program for DP-203

<https://www.linkedin.com/pulse/how-prepare-azure-dp-203-exam-week-munish-malhotra/?trk=public_post-content_share-article>

Lesson 1

Azure data store services

<https://www.youtube.com/watch?v=H-3lAESh-9A>

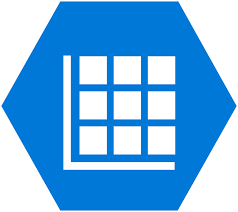
Azure Data Lake Storage Gen2

A picture containing text, clock

Description automatically generated Azure **B**lobs

 Azure **F**iles

 Azure **Q**ueues

 Azure **T**ables



**Non-Relational DB management systems**

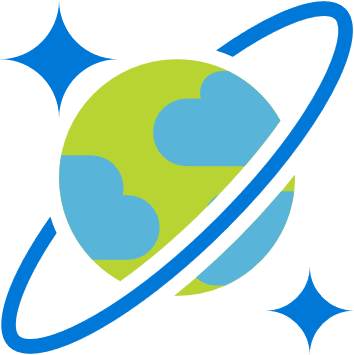
Icon

Description automatically generated Azure Data Lake Storage Gen2

*(Azure Storage Account)*

A picture containing text, clock

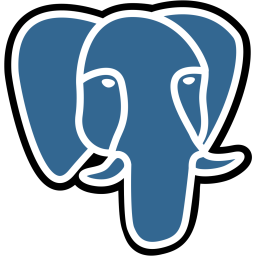
Description automatically generatedAzure **B**lobs

 Azure Cosmos DB

**Relational DB management systems**

 Azure SQL Database

Azure Database for PostgreSQL

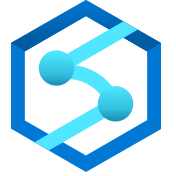


 Azure Database for MySQL

- Azure Database for MariaDB

**Data Analytics**

(NoSQL)

 Azure Synapse Analytics

Azure Data Lake Storage Gen2



*(Azure Storage Account)*

**Key/Value stores**

*(NoSQL)*

- Azure Cosmos DB Table API

- Azure Cache for Redis

- Azure Table Storage

*(Azure Storage Account)*

**Object Storage**

- Azure Cosmos Blob Storage

- Azure Data Lake Storage Gen2

*(Azure Storage Account)*

Graph Databases

(NoSQL)

- Azure Cosmos DB Gremlin API

- SQL Server

[Azure] Four Storage options:

B-F-Q-T

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | | **Ingest** | **Store** | **Prep & Train** | **Model & Serve** |
| Icon  Description automatically generated | **Azure Files** |  | ✓ |  |  |
| Icon  Description automatically generated | **Azure Queue** |  | ✓ |  |  |
| Icon  Description automatically generated | **Azure Data Lake Storage Gen 2**  (Non-relational data store) |  | ✓ |  |  |
| **A picture containing icon  Description automatically generated** | **CosmosDB**  (Non-relational data store) |  | ✓ |  | ✓ |
| Icon  Description automatically generated | **Azure Blob (WASB)**  (Non-relational data store) |  | ✓ |  |  |

**Azure Blob (WASB):** A scalable object store for text and binary data

This is a data store that will store but not query data, your cheapest option is to set up a storage account as a Blob store.

Blob storage works well with images and unstructured data

Flexible pricing options (cold vs hot storage)

Icon

Description automatically generated

Icon

Description automatically generated

**Azure Files:** Managed file shares for cloud or on-premises deployments. Accessible via the industry standard Server Message Block (SMB) protocol

Icon

Description automatically generated

**Azure Queue:** Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world.

Azure Table: A NoSQL store for no-schema storage of structured data

Diagram, icon

Description automatically generated

Azure Storage Tables is aimed at high capacity on a single region (optional secondary read only region but no failover), indexing by PK/RK and storage optimized pricing;

**A picture containing icon

Description automatically generated**

Azure Cosmos DB is a globally distributed database service.

**high throughput** (single-digit millisecond latency),

**global distribution** (multiple failover), SLA-backed predictive performance with automatic indexing of each attribute/property and a pricing model focused on throughput.

Data Lake Layers

Graphical user interface, application

Description automatically generated

Files (Compression Ratios)

Table

Description automatically generated

CSV vs Parquet vs Avro.

<https://medium.com/ssense-tech/csv-vs-parquet-vs-avro-choosing-the-right-tool-for-the-right-job-79c9f56914a8>

**CSV**

- Bulk processing data

- Not always best choice for Spark

**Parquet**

- columnar storage is optimized for picking a section of columns.

- if you only need part of each record, the latency of reads is considerably lower.

**Avro**

*Good for unstructured /schema differentiating data*

- Schema is richer than Parquet’s

- allows us to evolve the schema by adding, removing, or modifying the columns of a record, with much greater ease than Parquet

|  |  |  |  |
| --- | --- | --- | --- |
|  | CSV | Parquet | Avro |
| Read or Write | Write | Read | Write |
| Row or Column |  | Column | Row |
| Most Compatible Platforms | ALL | Spark | Kafka, NiFi |
| Compression Ratio |  | 97.5% | 91.24% |
| Compression |  | Icon  Description automatically generated | Icon  Description automatically generated |
| Schema Evolution Support |  | Icon  Description automatically generated | Shape  Description automatically generated with medium confidence |

Lesson 2

Azure SQL Database

Lesson 3

Cosmos DB

Lesson 4

Azure Stream Analytics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | | **Ingest** | **Store** | **Prep & Train**  **(analyse)** | **Model & Serve** |
| Logo, icon  Description automatically generated | **Azure Stream Analytics** | ✓ |  | ✓x` |  |

Event Processing

Event

Producer

Event

Processor

Event

Consumer

Event producer Generates data continuously

Event Processor An engine to consume event data streams and derive insights

Event Consumer Application that consumes the data and takes specific action based on the insights

Graphical user interface, application

Description automatically generated

How does it work?

<https://www.youtube.com/watch?v=NbGmyjgY0pU>

1. Define Source

. Input Type & Properties

. Input Alias

2. Define Output

. Output Type & Properties

. Output Alias

3. Define Job SQL Query

. Fetch from source

. Select & transform data

. Insert into output

Example:

Data is grouped every 2 seconds, and the aggregate function returns the max of each group

Diagram

Description automatically generated with medium confidence

**Stream Analytics windowing functions [T.H.S.S.S]**

<https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-window-functions>

Timeline

Description automatically generated

Tumbling window functions are used to segment a data stream into distinct time segments and perform a function against them, such as the example below. The key differentiators of a Tumbling window are that they repeat, do not overlap, and an event cannot belong to more than one tumbling window.

Timeline

Description automatically generatedHopping window functions hop forward in time by a fixed period. It may be easy to think of them as Tumbling windows that can overlap and be emitted more often than the window size. Events can belong to more than one Hopping window result set. To make a Hopping window the same as a Tumbling window, specify the hop size to be the same as the window size.

Chart, timeline

Description automatically generatedSliding windows, unlike Tumbling or Hopping windows, output events only for points in time when the content of the window changes. In other words when an event enters or exits the window. So, every window has at least one event. Like Hopping windows, events can belong to more than one sliding window.

Timeline

Description automatically generatedSession window functions group events that arrive at similar times, filtering out periods of time where there is no data. It has three main parameters: timeout, maximum duration, and partitioning key (optional).

Graphical user interface, application

Description automatically generatedSnapshot windows groups events that have the same timestamp. Unlike other windowing types, which require a specific window function (such as [SessionWindow()](https://docs.microsoft.com/en-us/stream-analytics-query/session-window-azure-stream-analytics), you can apply a snapshot window by adding System.Timestamp() to the GROUP BY clause.

Lesson 5

Azure HD Insight

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | | **Ingest** | **Store** | **Prep & Train** | **Model & Serve** |
| Logo  Description automatically generated | **Azure HDInsight**  **(Hadoop for azure)** |  |  | ✓ |  |
| **Icon  Description automatically generated** | **Azure Data Lake Analytics** |  |  | ✓ |  |
| **Icon  Description automatically generated** | **Azure Databricks**  **(python, Scala, Spark SQL, Spark R, Spark Structured Streaming)** |  |  | ✓ |  |

Lesson 6

Azure Synapse Analytics

<https://www.youtube.com/watch?v=WbDqeNsmoL4>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | | **Ingest** | **Store** | **Prep & Train** | **Model & Serve** |
| Icon  Description automatically generated | **Azure Data Factory** | ✓ |  |  |  |
|  | **Azure Synapse Analytics**  **(Azure SQL DW)** | ✓ |  |  | ✓ |

**Azure Synapse Analytics** [Azure]

Azure Synapse Analytics is generalized analytics service.

It uses a clustered architecture.

Azure Synapse is a:

- Data Warehouse solution

- Big Data solution

- Data Integration

… And a lot more!

You can use a variety of languages/frameworks with Azure Synapse Analytics

**Azure Synapse Workspace**

Graphical user interface, application

Description automatically generated

**Azure Synapse Studio**

Azure Synapse Studio offers everything in one place

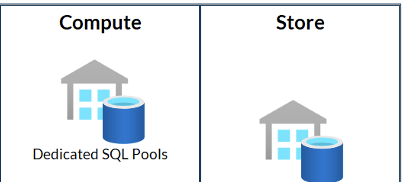
A picture containing text, device

Description automatically generated

Azure Synapse Analytics vs Synapse Dedicated SQL Pool (formerly SQL DW)

Dedicated SQL Pool

(formerly SQL DW)



To load data into a normal DW, there are best practices.  
**M.D.I.P.I**

Azure Synapse Analytics

Graphical user interface, application

Description automatically generated

A table created in any pool will be immediately available to all the others

Lesson 6

Azure Synapse Analytics

**Distributions**

- A distribution is a basic unit of storage and processing.

**Sharding**

- A distributed table appears as a single table, but the rows are stored across 60 distributions.

- When a dedicated SQL Pool (DW) runs a query, the work is divided into 60 smaller queries that run in parallel i.e. it is divided into 60 **distributions**.

- When data is ingested into a dedicated SQL Pool (DW), the data is sharded into distributions to optimize the performance of the system.

The user has the option of which sharding pattern to use when defining the table

**Sharding Patterns**

A **round-robin distributed table**

**QUERY SPEED ↓**

**LOADING SPEED ↑**

Distributes rows evenly across all distributions.

The assignment of rows to distributions is random

A picture containing text

Description automatically generated

A **hash distributed table**

**QUERY SPEED ↑**

*(on large fact tables)*

**LOADING SPEED ↓**

Distributes table rows across the Compute nodes by

using a deterministic hash function.

Assigns each row to one distribution.

A screenshot of a map

Description automatically generated with medium confidence

CREATE TABLE [dbo].[FactInternetSales]

( [ProductKey] int NOT NULL

, [OrderDateKey] int NOT NULL

, [CustomerKey] int NOT NULL

, [PromotionKey] int NOT NULL

, [SalesOrderNumber] nvarchar(20) NOT NULL

, [OrderQuantity] smallint NOT NULL

, [UnitPrice] money NOT NULL

, [SalesAmount] money NOT NULL

)

WITH

( CLUSTERED COLUMNSTORE INDEX

, **DISTRIBUTION = HASH([ProductKey])**

);

<https://learn.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-distribute>

Lesson 7

Moving data in Dedicated SQL Pool (formerly SQL DW)

1. Partitioning tables in dedicated SQL pool

*referred to as* ***switching out to archive data*** *from partitioned tables*

1. CREATE TABLE AS SELECT (CTAS) T-SQL statement in dedicated SQL pool
2. SELECT… INTO T-SQL statement

**CREATE TABLE AS SELECT**

<https://learn.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-develop-ctas>

The [CREATE TABLE AS SELECT](https://learn.microsoft.com/en-us/sql/t-sql/statements/create-table-as-select-azure-sql-data-warehouse?toc=/azure/synapse-analytics/sql-data-warehouse/toc.json&bc=/azure/synapse-analytics/sql-data-warehouse/breadcrumb/toc.json&view=azure-sqldw-latest&preserve-view=true) (CTAS) statement is one of the most important T-SQL features available.

CTAS is a parallel operation that creates a new table based on the output of a SELECT statement.

CTAS is the simplest and fastest way to create and insert data into a table with a single command.

Customizable version of **SELECT… INTO**

CREATE TABLE [dbo].[FactInternetSales\_new]

WITH

(

DISTRIBUTION = ROUND\_ROBIN

,CLUSTERED COLUMNSTORE INDEX

)

AS

SELECT \*

FROM [dbo].[FactInternetSales];

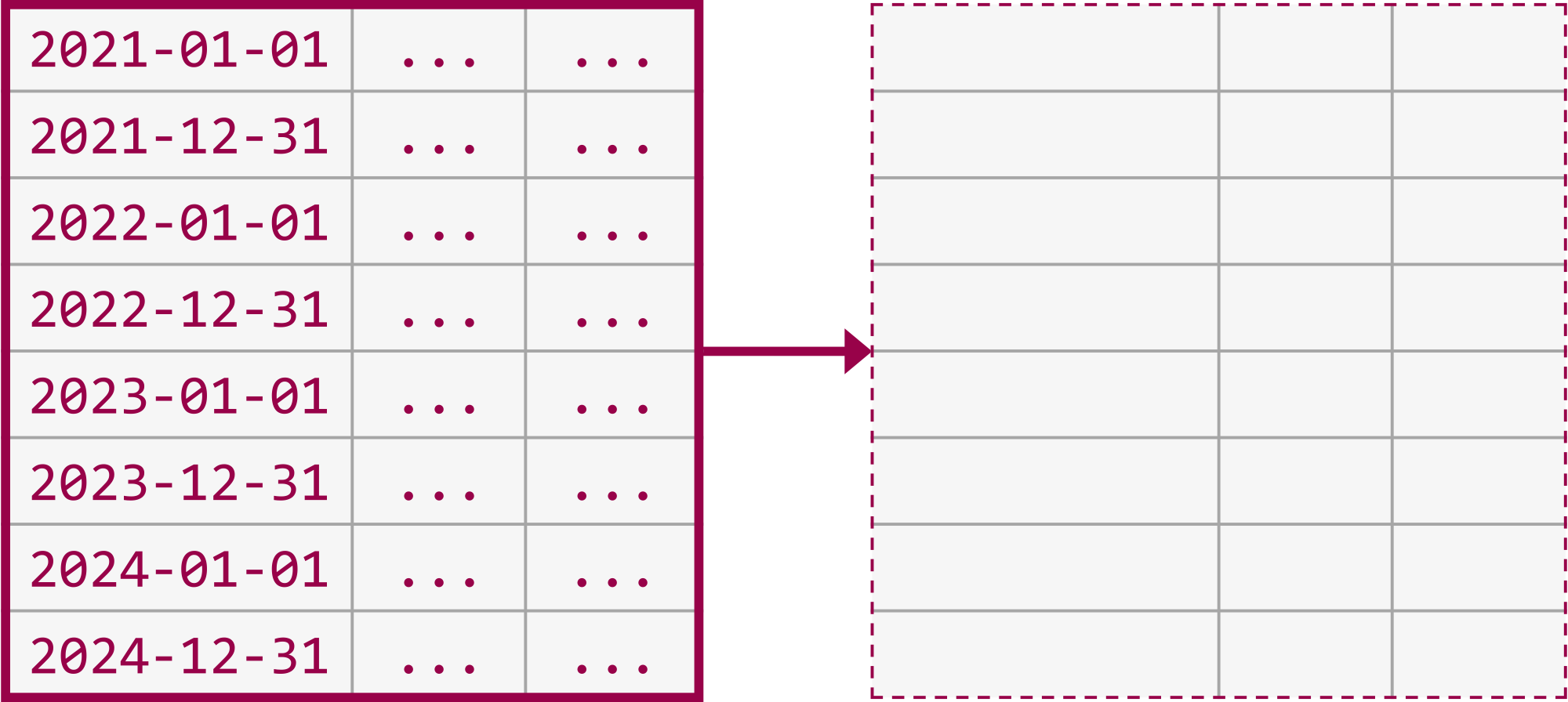
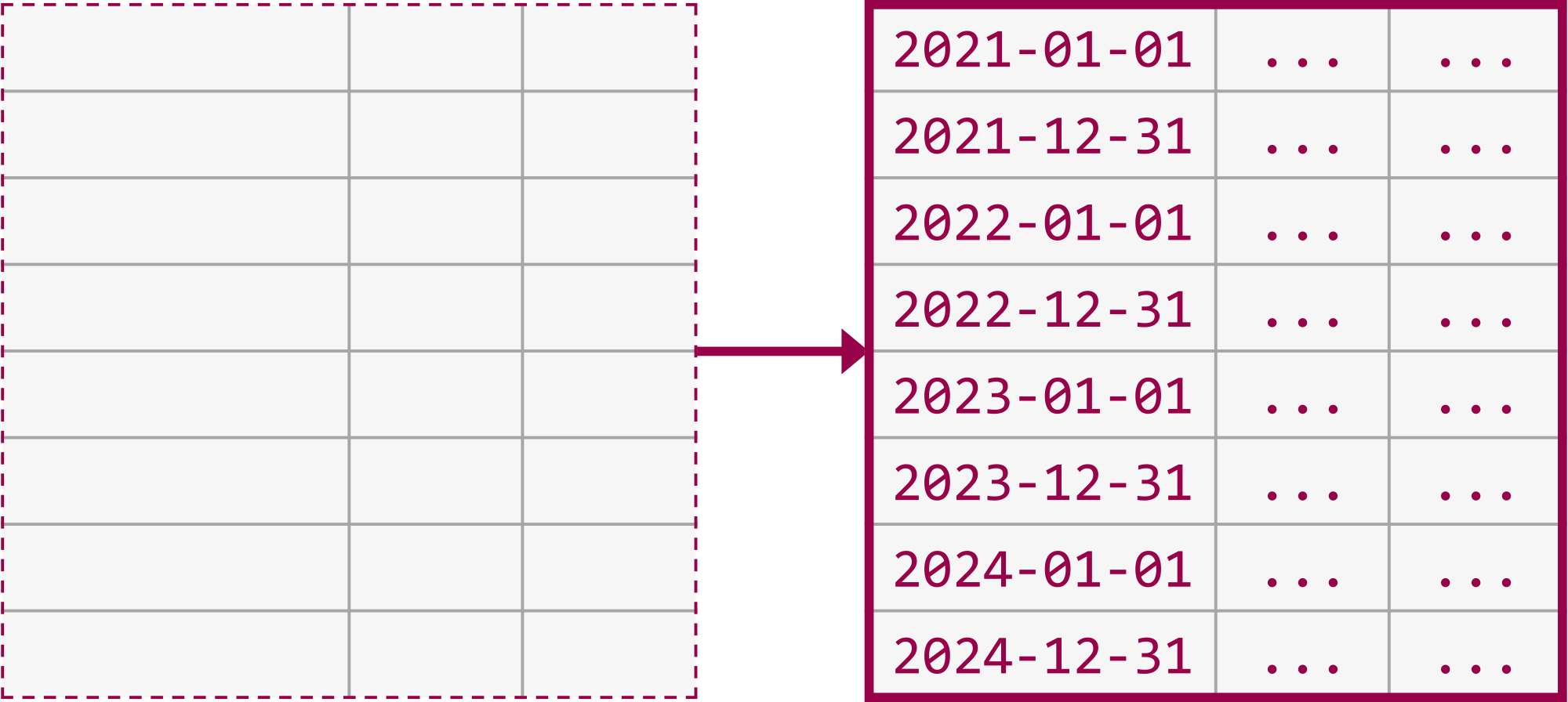
Partitioning tables in dedicated SQL pool

1. Non-partitioned table to another non-partitioned table
2. Non-partitioned table to a partition in a partitioned table (*Load data by switching in*)
3. Partition in a partitioned table to a non-partitioned table (*Archive data by switching out*)
4. Partition in a partitioned table to a partition in another partitioned table

**1**

The first way to use the ALTER TABLE SWITCH statement is to switch all the data from a non-partitioned table to an empty non-partitioned table:

ALTER TABLE SourceTable SWITCH TO TargetTable;

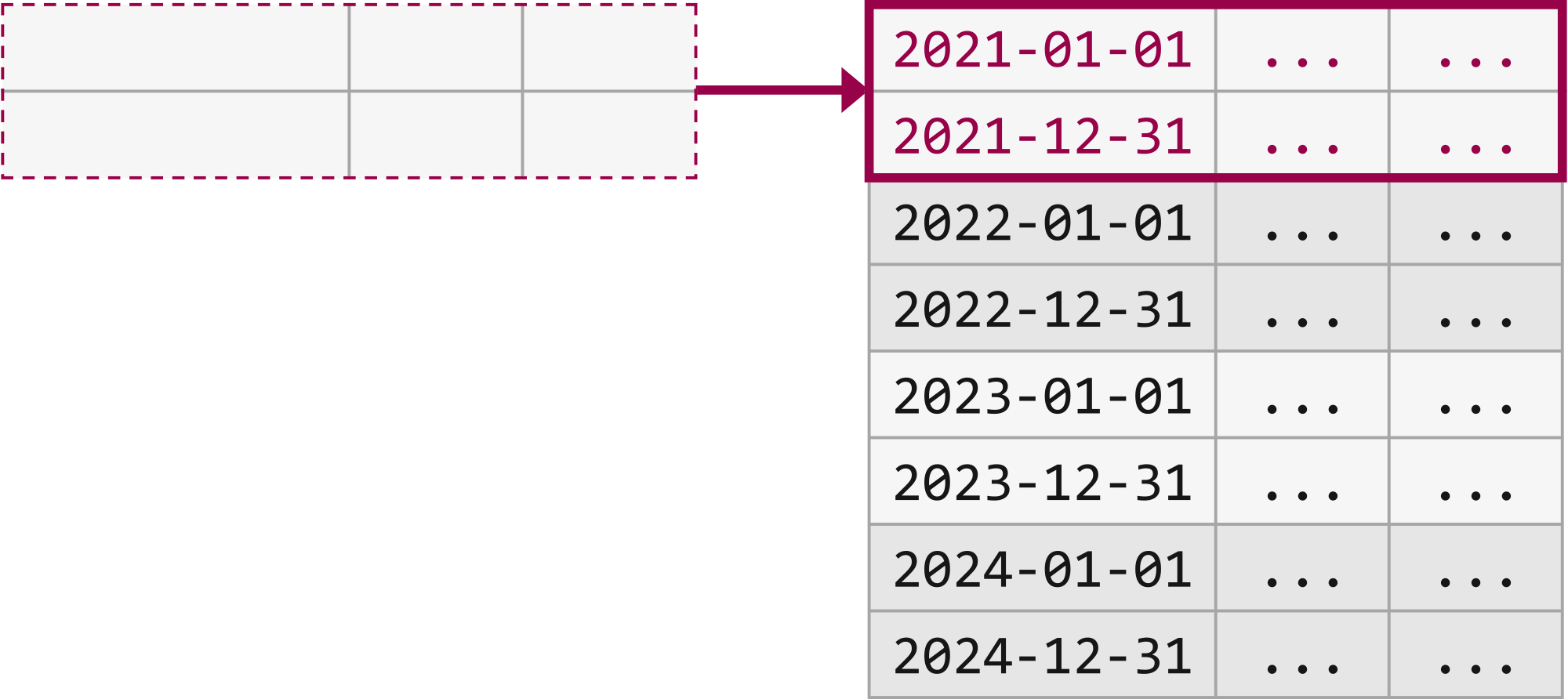
****

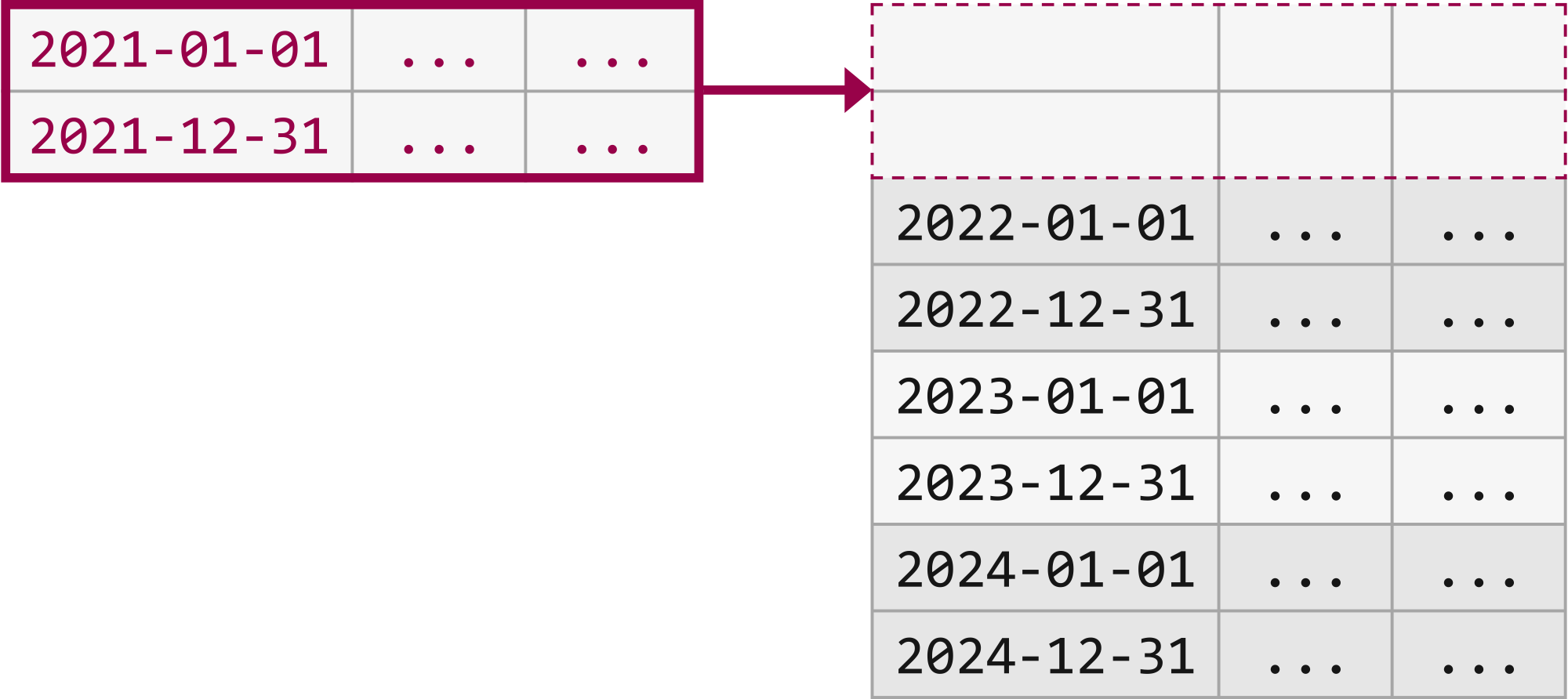
**BEFORE AFTER**

**2**

The second way to use the ALTER TABLE SWITCH statement is to switch all the data from a non-partitioned table to an empty specified partition in a partitioned table:

ALTER TABLE Source SWITCH TO Target PARTITION 1;

****

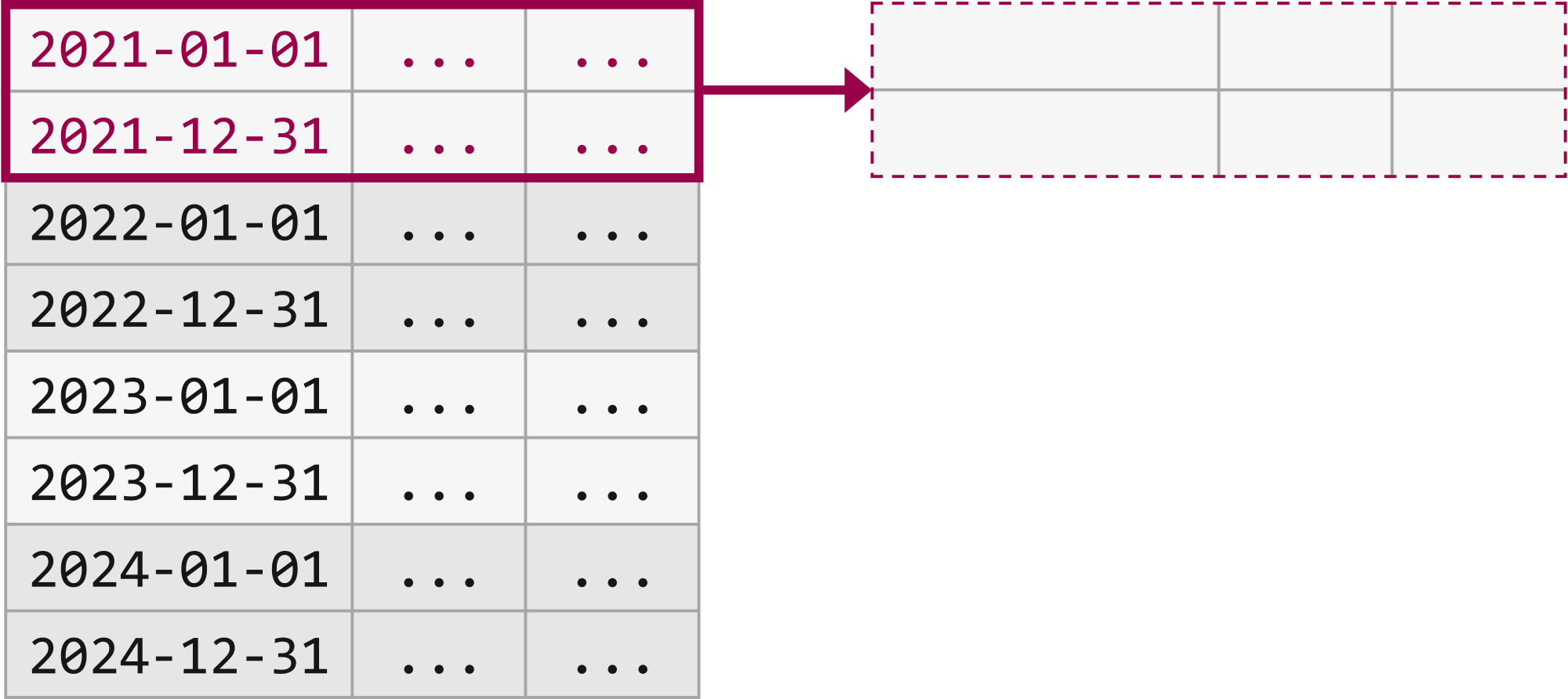
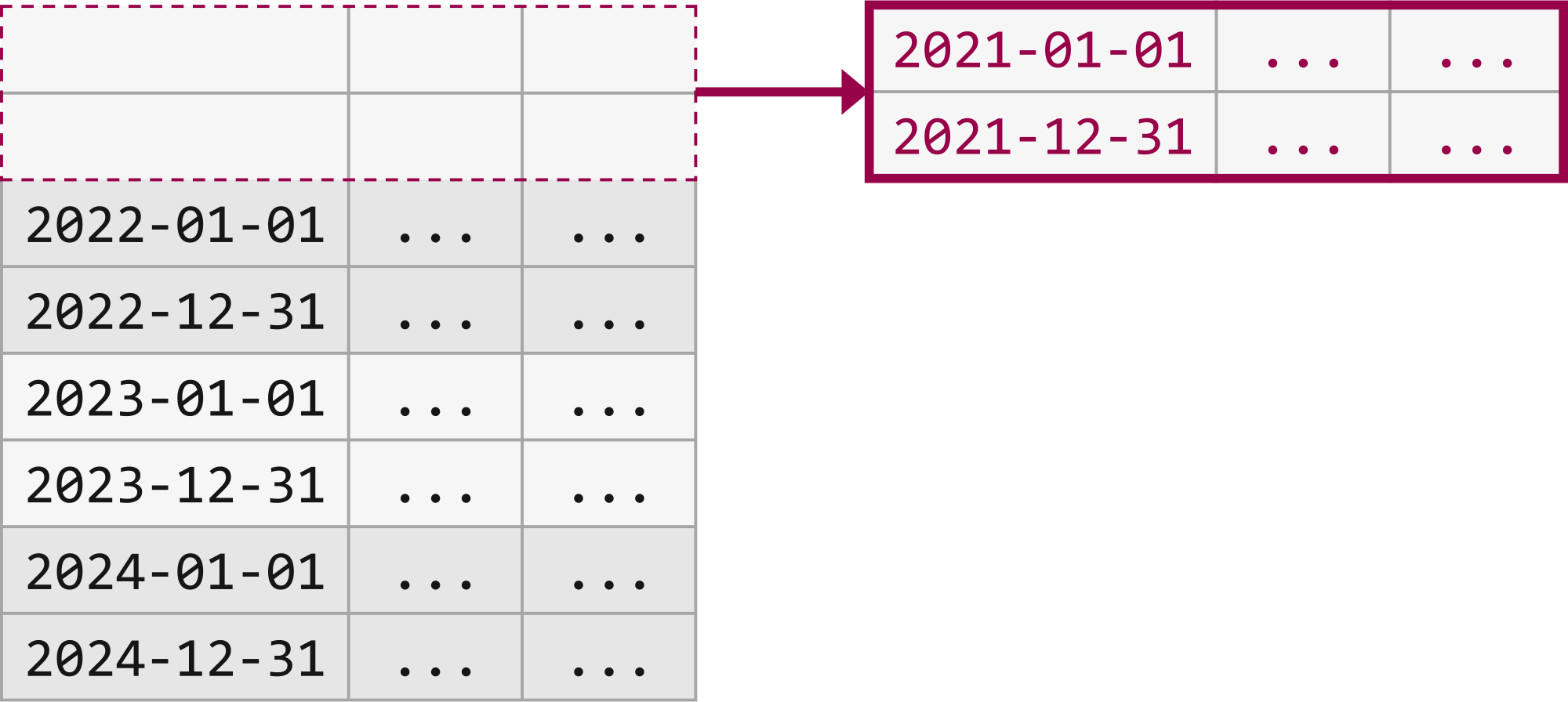
****

**BEFORE AFTER**

The third way to use the ALTER TABLE SWITCH statement is to switch all the data from a specified partition in a partitioned table to an empty non-partitioned table:

**3**

ALTER TABLE Source SWITCH PARTITION 1 TO Target;

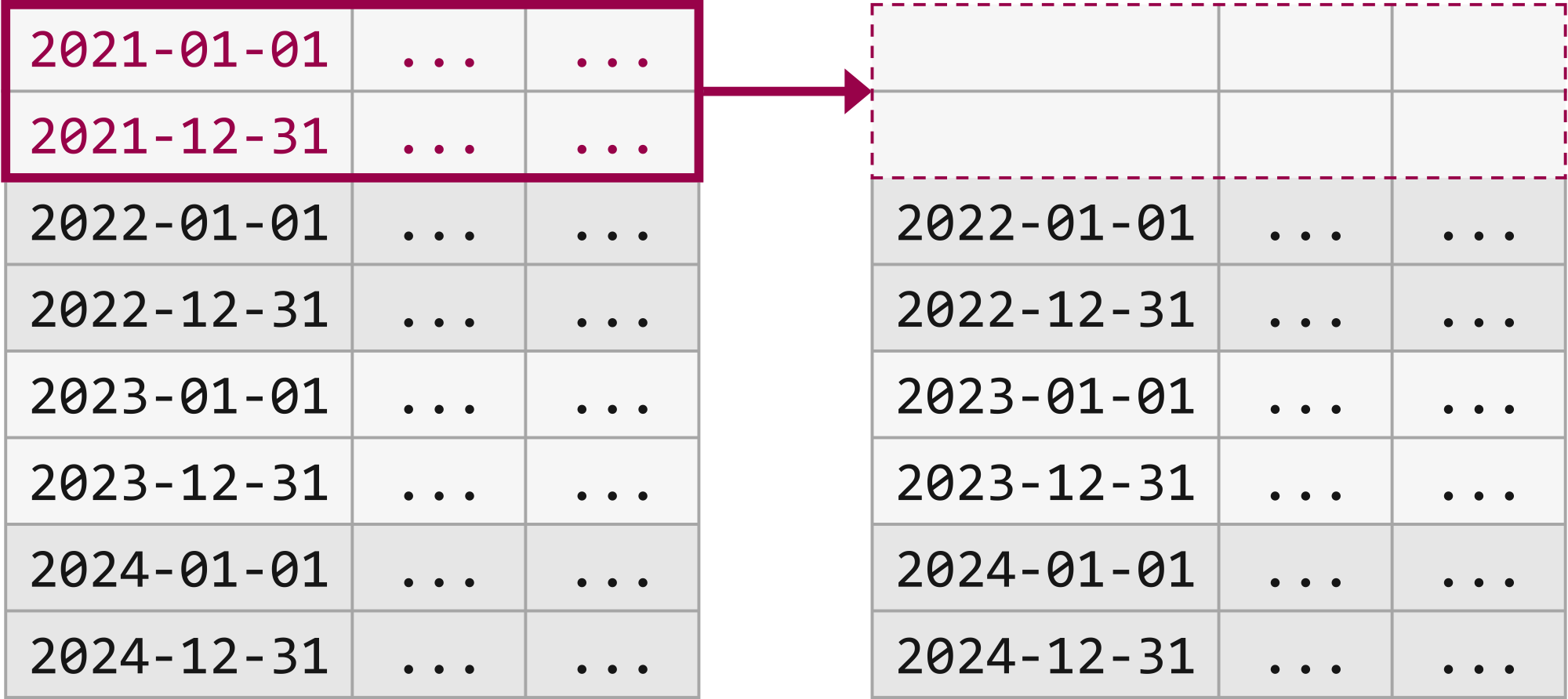
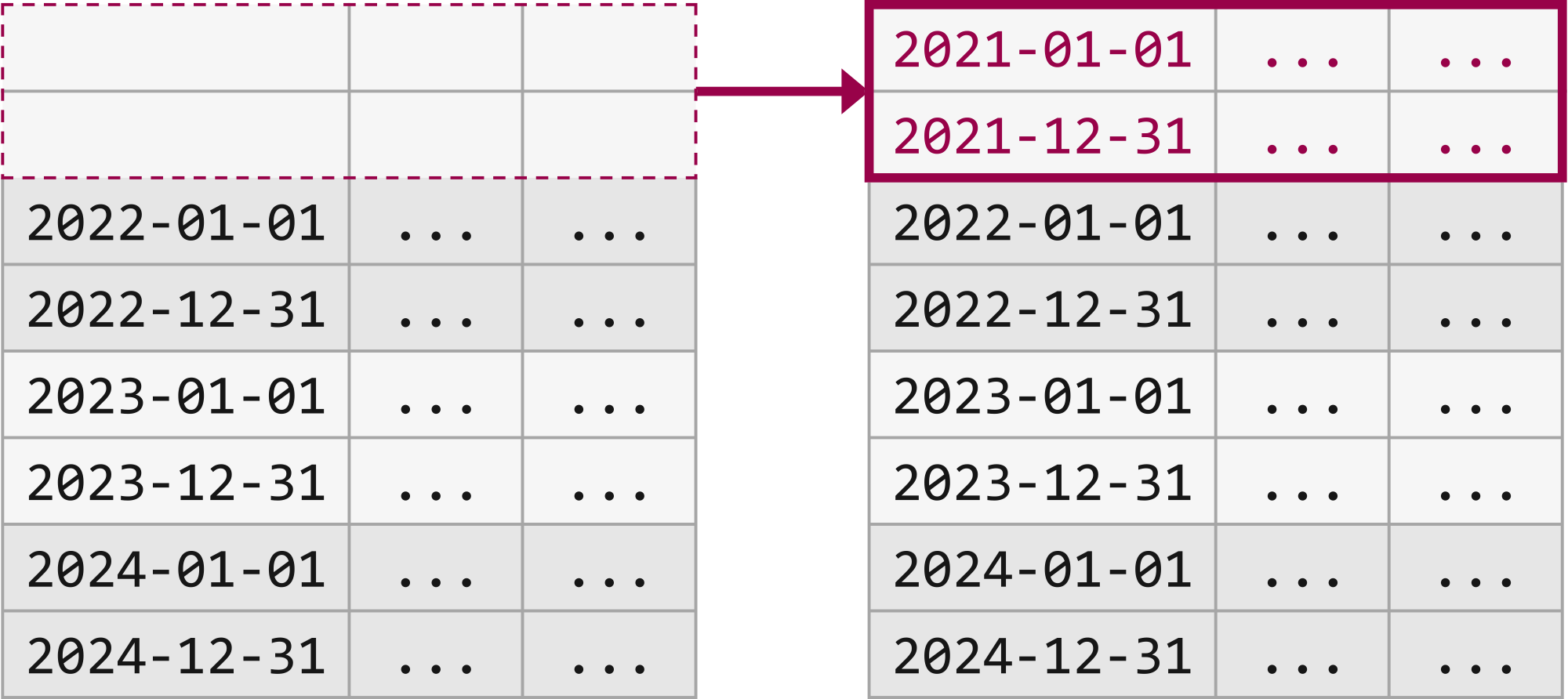
****

**BEFORE AFTER**

The fourth way to use the ALTER TABLE SWITCH statement is to switch all the data from a specified partition in a partitioned table to an empty specified partition in another partitioned table:

**4**

ALTER TABLE Source SWITCH PARTITION 1 TO Target PARTITION 1;

****

**BEFORE AFTER**

Lesson 8

**M**→**D**→**I**→**P**→**I**:

<https://learn.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/cheat-sheet>

Diagram

Description automatically generated

**5**

**3**

**4**

**2**

**1**

**Data migration**:

First, load your data into Azure Data Lake Storage or Azure Blob Storage.

Next, use the COPY statement (preview) to load your data into staging tables.

Use the following configuration:

|  |  |
| --- | --- |
| **Design** | **Recommendation** |
| Distribution | Round Robin |
| Indexing | Heap |
| Partition | None |
| Resource Class | largerc or xlargerc |

**Distributed or replicated tables**:

Start with Round Robin, but aspire to a hash distribution strategy to take advantage of a massively parallel architecture.

|  |  |  |
| --- | --- | --- |
| **Type** | **Great fit for…** | **Watch out if** |
| Replicated | \* Small dimension tables in star schema (<2GB) | \* Many write transactions (insert, upsert, delete, update)  \* You change warehouse units (DWU) and provision frequently |
| Round Robin | \* Temp/Staging Table  No obvious joining key | \* Performance is slow due to data movement |
| Hash | \* Fact Tables  \* Large Dimension Tables | \* The distribution key cannot be updated |

**Index your tables**

Indexing is helpful for reading tables quickly. There is a unique set of technologies that you can use based on your needs:

|  |  |  |
| --- | --- | --- |
| **Type** | **Great fit for…** | **Watch out if** |
| Heap | Temp/Staging Table | \* Any lookup scans the full table |
| Clustered Index  *nonclustered index to a column heavily used for filtering.* | Tables with up to 100 million rows  Large tables (more than 100 million rows) with only 1-2 columns heavily used | \* Used on a replicated table \* You have complex queries involving multiple join and Group By operations \* You make updates on the indexed columns: it takes memory |
| Clustered columnstore index (CCI) (default) | Large tables (more than 100 million rows) | \* Used on a replicated table  \* You make massive update operations on your table |

**Partitioning**

You might partition your table when you have a large fact table (greater than 1 billion rows). In 99 percent of cases, the partition key should be based on date. Be careful to not overpartition, especially when you have a clustered columnstore index.

With staging tables that require ELT, you can benefit from partitioning. It facilitates data lifecycle management. Be careful not to overpartition your data, especially on a clustered columnstore index.

**Incremental load**

If you're going to incrementally load your data, first make sure that you allocate larger resource classes to loading your data.

This is particularly important when loading into tables with clustered columnstore indexes. See resource classes for further details.

For a large batch of updates in your historical data, consider using a [CTAS](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-develop-ctas) to write the data you want to keep in a table rather than using INSERT, UPDATE, and DELETE.

CREATE TABLE [dbo].[FactInternetSales\_new]

WITH

(

DISTRIBUTION = ROUND\_ROBIN

,CLUSTERED COLUMNSTORE INDEX

)

AS

SELECT \*

FROM [dbo].[FactInternetSales];

Customizable version of **SELECT… INTO**