**Designing tables in dedicated SQL pool.**

Note: Serverless SQL pool in Azure Synapse Analytics supports only **external** and **temporary** tables.

## Table persistence

Tables store data either permanently in Azure Storage, temporarily in Azure Storage, or in a data store external to dedicated SQL pool.

## Regular table

A regular table stores data in Azure Storage as part of dedicated SQL pool. The table and the data persist regardless of whether a session is open.

## Temporary table

A temporary table only exists for the duration of the session. You can use a temporary table to prevent other users from seeing temporary results and also to reduce the need for cleanup.

## External table

An external table points to data located in Azure Storage blob or Azure Data Lake Store. When used with the CREATE TABLE AS SELECT statement, selecting from an external table imports data into dedicated SQL pool.

## Distributed tables

A fundamental feature of dedicated SQL pool is the way it can store and operate on tables across distributions. Dedicated SQL pool supports three methods for distributing data: round-robin (default), hash and replicated.

## Hash-distributed tables

A hash distributed table distributes rows based on the value in the distribution column. A hash distributed table is designed to achieve high performance for queries on large tables. There are several factors to consider when choosing a distribution column.

## Replicated tables

A replicated table has a full copy of the table available on every Compute node. Queries run fast on replicated tables since joins on replicated tables don't require data movement. Replication requires extra storage, though, and isn't practical for large tables.

## Round-robin tables

A round-robin table distributes table rows evenly across all distributions. The rows are distributed randomly. Loading data into a round-robin table is fast. Keep in mind that queries can require more data movement than the other distribution methods.

**Determine table category**

A star schema organizes data into fact and dimension tables. Some tables are used for integration or staging data before it moves to a fact or dimension table. As you design a table, decide whether the table data belongs in a fact, dimension, or integration table. This decision informs the appropriate table structure and distribution.

* **Fact tables** contain quantitative data that are commonly generated in a transactional system, and then loaded into the dedicated SQL pool. For example, a retail business generates sales transactions every day, and then loads the data into a dedicated SQL pool fact table for analysis.
* **Dimension tables** contain attribute data that might change but usually changes infrequently. For example, a customer's name and address are stored in a dimension table and updated only when the customer's profile changes. To minimize the size of a large fact table, the customer's name and address don't need to be in every row of a fact table. Instead, the fact table and the dimension table can share a customer ID. A query can join the two tables to associate a customer's profile and transactions.
* **Integration tables** provide a place for integrating or staging data. You can create an integration table as a regular table, an external table, or a temporary table. For example, you can load data to a staging table, perform transformations on the data in staging, and then insert the data into a production table.

## Common distribution methods for tables

The table category often determines which option to choose for distributing the table.

| **COMMON DISTRIBUTION METHODS FOR TABLES** | |
| --- | --- |
| **Table category** | **Recommended distribution option** |
| Fact | Use hash-distribution with clustered columnstore index. Performance improves when two hash tables are joined on the same distribution column. |
| Dimension | Use replicated for smaller tables. If tables are too large to store on each Compute node, use hash-distributed. |
| Staging | Use round-robin for the staging table. The load with CTAS is fast. Once the data is in the staging table, use INSERT...SELECT to move the data to production tables. |

## Unsupported table features

Dedicated SQL pool supports many, but not all, of the table features offered by other databases. The following list shows some of the table features that aren't supported in dedicated SQL pool:

* Foreign key
* Computed Columns
* Indexed Views
* Sequence
* Sparse Columns
* Surrogate Keys. Implement with Identity.
* Synonyms
* Triggers
* Unique Indexes
* User-Defined Types

## Aligning source data with dedicated SQL pool

Dedicated SQL pool tables are populated by loading data from another data source. To perform a successful load, the number and data types of the columns in the source data must align with the table definition in the dedicated SQL pool. Getting the data to align might be the hardest part of designing your tables.

If data is coming from multiple data stores, you load the data into the dedicated SQL pool and store it in an integration table. Once data is in the integration table, you can use the power of dedicated SQL pool to perform transformation operations. Once the data is prepared, you can insert it into production tables.

## Unsupported data types

Dedicated SQL pool (formerly SQL DW) supports the most commonly used data types. For a list of the supported data types, see [data types](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-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#DataTypes) in the CREATE TABLE statement.

The following list shows the data types that dedicated SQL pool (formerly SQL DW) doesn't support and gives useful alternatives for unsupported data types.

| **WORKAROUNDS FOR UNSUPPORTED DATA TYPES** | |
| --- | --- |
| **Unsupported data type** | **Workaround** |
| [geometry](https://docs.microsoft.com/en-us/sql/t-sql/spatial-geometry/spatial-types-geometry-transact-sql?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) | [varbinary](https://docs.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?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) |
| [geography](https://docs.microsoft.com/en-us/sql/t-sql/spatial-geography/spatial-types-geography) | [varbinary](https://docs.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?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) |
| [hierarchyid](https://docs.microsoft.com/en-us/sql/t-sql/data-types/hierarchyid-data-type-method-reference) | [nvarchar](https://docs.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql?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)(4000) |
| [image](https://docs.microsoft.com/en-us/sql/t-sql/data-types/ntext-text-and-image-transact-sql?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) | [varbinary](https://docs.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?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) |
| [text](https://docs.microsoft.com/en-us/sql/t-sql/data-types/ntext-text-and-image-transact-sql?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) | [varchar](https://docs.microsoft.com/en-us/sql/t-sql/data-types/char-and-varchar-transact-sql?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) |
| [ntext](https://docs.microsoft.com/en-us/sql/t-sql/data-types/ntext-text-and-image-transact-sql?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) | [nvarchar](https://docs.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql?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) |
| [sql\_variant](https://docs.microsoft.com/en-us/sql/t-sql/data-types/sql-variant-transact-sql?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) | Split column into several strongly typed columns. |
| [table](https://docs.microsoft.com/en-us/sql/t-sql/data-types/table-transact-sql?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) | Convert to temporary tables. |
| [timestamp](https://docs.microsoft.com/en-us/sql/t-sql/data-types/date-and-time-types) | Rework code to use [datetime2](https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime2-transact-sql?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) and the [CURRENT\_TIMESTAMP](https://docs.microsoft.com/en-us/sql/t-sql/functions/current-timestamp-transact-sql?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) function. Only constants are supported as defaults, so current\_timestamp can't be defined as a default constraint. If you need to migrate row version values from a timestamp typed column, use [BINARY](https://docs.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?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)(8) or [VARBINARY](https://docs.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?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)(8) for NOT NULL or NULL row version values. |
| [xml](https://docs.microsoft.com/en-us/sql/t-sql/xml/xml-transact-sql?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) | [varchar](https://docs.microsoft.com/en-us/sql/t-sql/data-types/char-and-varchar-transact-sql?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) |
| [user-defined type](https://docs.microsoft.com/en-us/sql/relational-databases/native-client/features/using-user-defined-types?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) | Convert back to the native data type when possible. |
| default values | Default values support literals and constants only. |