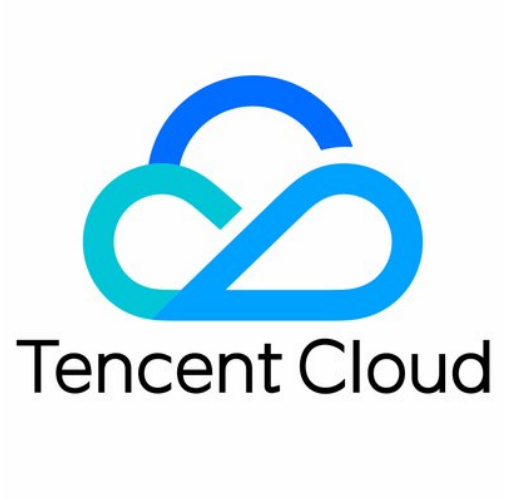
TencentDB for PostgreSQL

MSSQL Compatible Version

Product Documentation



Contents

MSSQL Compatible Version Architecture Introduction

Last updated：2024-04-09 10:31:59

Based on Babelfish for PostgreSQL, TencentDB for PostgreSQL has been adapted to support the data types, syntax, and functions of Microsoft SQL Server, the SQL Server wire-level protocol (TDS), and the communication between SQL Server applications and PostgreSQL. This helps the migration of objects, storage procedures, and application codes from TencentDB for SQL Server to TencentDB for PostgreSQL with minimal changes.
TencentDB for PostgreSQL does not fully support T-SQL, but you can run PostgreSQL commands to perform most tasks that are typically handled by these commands. For example, if you frequently use specific T-SQL commands that are not supported by TencentDB for PostgreSQL, you can connect to the PostgreSQL port and use PostgreSQL commands instead. For more information, see the SQL commands in the PostgreSQL documentation.

Note:

All the subsequent SQL Server versions compatible with TencentDB for PostgreSQL are collectively referred to as the MSSQL Compatible Edition.

Architecture Description

The MSSQL Compatible Edition currently supports TencentDB for PostgreSQL version 14.
The MSSQL Compatible Edition has a new database access port, allowing it to support SQL Server T-SQL and commonly used SQL Server statements, and enable TDS-based client applications to access the TDS listener port of the MSSQL Compatible Edition. Currently, TDS 7.1 and later versions are supported. For more information about the SQL Server wire-level protocol, see [MS-TDS]: Tabular data stream protocol.
You can access data simultaneously by using the TDS connection from an application and the native PostgreSQL connection.
By default, to use different database-specific syntax, please select the following ports:
For SQL Server, clients connect to port 1433.
For PostgreSQL, clients connect to port 5432.
When building a TecentDB for PostgreSQL database instance for the MSSQL Compatible Edition, the system will create a TecentDB for PostgreSQL database named `babelfish_db` for the instance. The database is where all SQL Server objects and structures are migrated to.

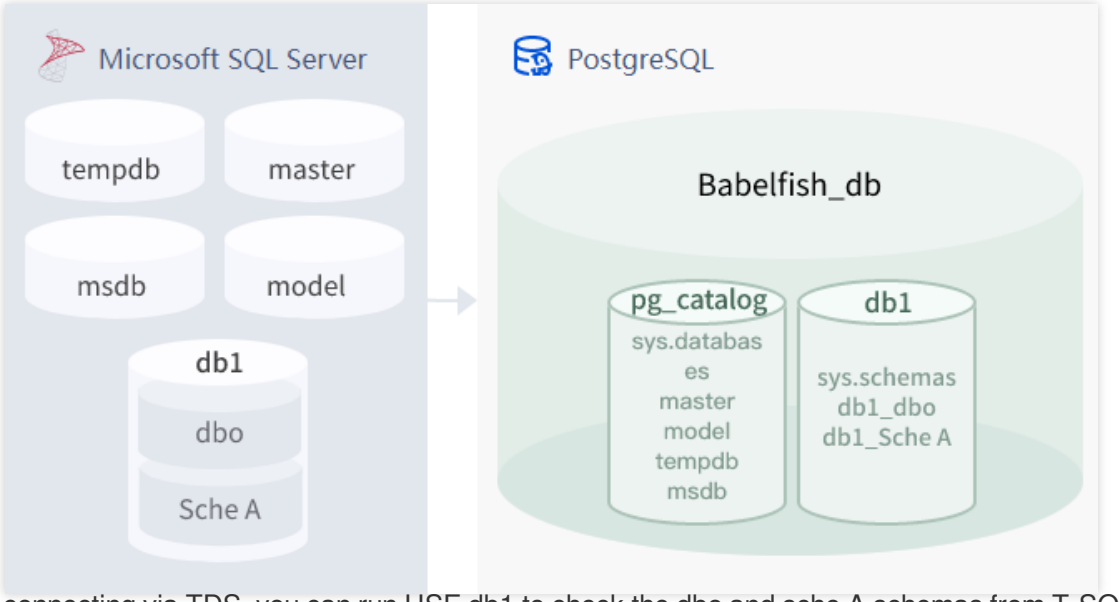
Note:

A database named `babelfish_db` will be reserved for this instance.
If you connect to the TDS port, the session will automatically be switched to the `babelfish_db` database. From the perspective of T-SQL, the architecture is similar to that for connecting a SQL Server instance. You will see the

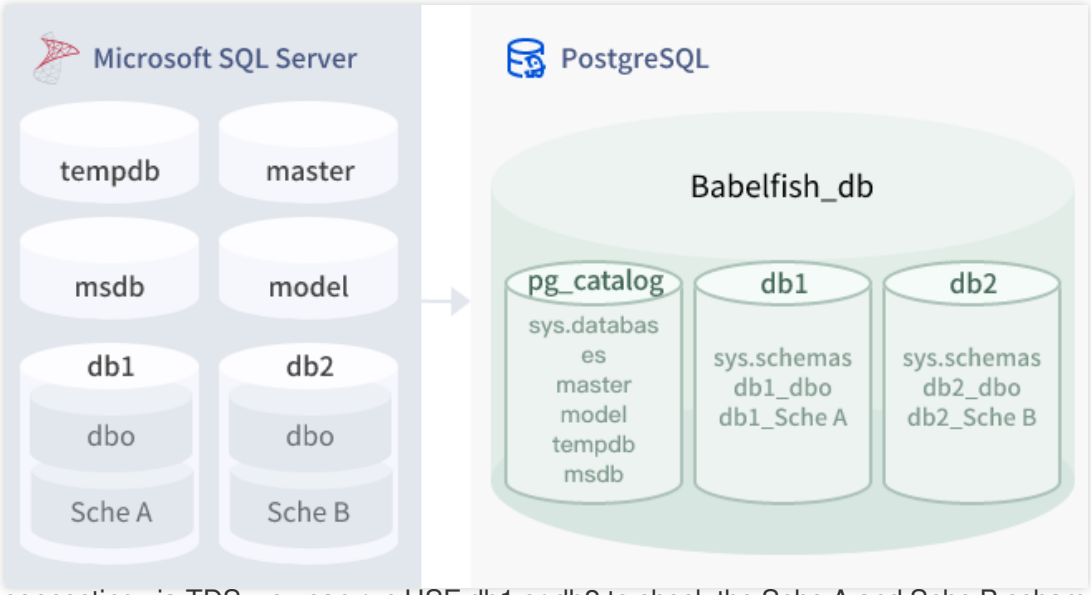
`master` and `tempdb` databases along with the `sys.databases` directory. You can create additional user databases and switch between them through the USE statement.
When you create a SQL Server user database, the database is mapped as a schema in the `babelfish_db` database. The retained cross-database syntax and semantics are equal to or similar to those assigned by SQL Server.

Difference Between Single-Database and Multi-Databases

When you create a MSSQL Compatible Edition instance, you can choose one or more SQL Server databases. You choice will affect how the names of SQL Server schemas within the `babelfish_db` database are displayed in PostgreSQL. The migration mode is specified in the `migration_mode` parameter, which is an initialization parameter and cannot be modified later.
In single-database mode, the schema name of the user database within the `babelfish_db` database is the same as that in the SQL Server database. If you choose to migrate a single database, its schemas will be rebuilt in the database, and you can name them the same as those used in SQL Server. For example, if the `dbo` and sche A schemas are in the db1 database, after migration to PostgreSQL, the schemas are as follows.



When connecting via TDS, you can run USE db1 to check the dbo and sche A schemas from T-SQL, just like in SQL Server. You can also check the same schema names from PostgreSQL.
In multi-database mode, the user database schema name becomes dbname_schemaname in PostgreSQL, while the schema name remains unchanged in T-SQL.



When connecting via TDS, you can run USE db1 or db2 to check the Sche A and Sche B schames from T-SQL, just like in TencentDB for SQL Server. You can also check the names of their mapped schemas, such as db1_dbo and dbA_Sche A, in TencentDB for stgreSQL.

Each database still contains your schemas. Each database's name is prefixed to the SQL Server schema name with an underscore as a separator, for example:

db1 contains db1_dbo and db1_sche A.

db2 contains db2_dbo and db2_Sche B.

In the babelfish_db database, T-SQL users still need to run the USE dbname command to change databases, similar to the operations in TencentDB for SQL Server.

How to Choose a Migration Mode

You can select a proper migration mode based on the number of user databases you have and how you plan to migrate them. Once you create an instance, you will not be able to change the migration mode. Therefore, when choosing a migration mode, please take into account of your user databases and client requirements.

If you create an MSSQL Compatible Edition instance, the master and tempdb system databases are also built. If you already created or modified any objects in the system databases (master or tempdb), please be sure to re-create these objects in the new cluster. Unlike TencentDB for SQL Server, TencentDB for PostgreSQL does not re-initialize tempdb after cluster restart.

Single-database migration mode is recommended in the following scenario:

You intend to migrate a single TecentDB for SQL Server database. In single-database mode, the migrated schema names are the same as the original schema names in the SQL Server, and migrating applications requires minimal changes to SQL codes.

Your ultimate goal is to fully migrate databases to TecentDB for PostgreSQL, and the Compatible Edition is only for transition.

Multi-database migration mode is recommended in the following scenarios:

You are trying out the MSSQL Compatible Edition and are not sure about what you will need.

You need to migrate multiple user databases together, and your ultimate goal is not for a completely native migration to TencentDB for PostgreSQL.

You have a potential need to migrate multiple databases in the future.

Function Description

Last updated：2024-05-16 15:56:58

Differences in Using MSSQL Compatible Edition and T-SQL

TencentDB for PostgreSQL supports most T-SQL syntax. You can find a table below listing the currently supported T-SQL features. It includes notes about differences in behavior compared to SQL Server.

Feature or Syntax	Behavior or Difference Description
\\ (Line Continuation Character)	Currently, line continuation characters (backslashes before newline) for strings and hexadecimal strings are not supported. For strings, the backslash followed by a newline is interpreted as a character in the string. For hexadecimal strings, the backslash followed by a newline will result in a syntax error.
@@version	The format of the value returned by @@version is slightly different from that of the value returned by SQL Server. If your code depends on the formatting of @@version, it may not work properly.
Aggregate Function	Partially support for aggregate functions (supports AVG, COUNT, COUNT_BIG, GROUPING, MAX, MIN, STRING_AGG, and SUM).
ALTER TABLE	Only adding or deleting a single column or constraint is supported.
BACKUP Statement	Backup methods differ. In TencentDB for PostgreSQL's SQL Server Compatible Edition, backup operations can only be conducted via the cloud console.
Blank Column Names Without Aliases	sqlcmd and psql handling columns with empty names differently: SQL Server sqlcmd returns a blank column name. PostgreSQL psql returns a system-generated column name.
Data Type Indexes According to ICU Library Collation Rules	When the library version is changed, indexes of user-defined types dependent on the International Components for Unicode (ICU) collation library (used by the MSSQL Compatible Edition) will not become invalid.
COLLATIONPROPERTY Function	Collation rule properties are only applicable to the supported collation rules in the MSSQL Compatible Edition BBF.
Original Column Settings	When a column with the original setting is being created, the constraint name is ignored. To delete the original setting of a column, please use the following syntax: `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...` .
Constraint	In PostgreSQL, individual constraint conditions cannot be enabled or disabled. The statement will be ignored and a warning will be issued.

Constraints Created Using DESC (Descending Order) Columns	Constraints are created using ASC (ascending order) columns.
Constraints with the IGNORE_DUP_KEY	Creating constraints with this attribute is not supported.
CREATE, ALTER, DROP SERVER ROLE	ALTER SERVER ROLE is only supported for sysadmin. Other syntax is not supported. In MSSQL Compatible Edition, T-SQL user experiences regarding log-ins (server principals), databases, and database users (database principals) are similar to those in SQL Server. In MSSQL Compatible Edition, only the dbo user exists in the user databases. To operate as the dbo user, the log-in name must be a member of the server-level sysadmin role (ALTER SERVER ROLE sysadmin ADD MEMBER log-in). Log-ins that are not of the sysadmin role currently can only access the master and tempdb databases as the guest user. Currently, since the MSSQL Compatible Edition supports only the dbo user in user databases, all application users must use a log-in name that is a member of the sysadmin role. You cannot create users with lower permissions, such as read-only access to certain tables.
CREATE, ALTER LOGIN Clauses Supporting Limited Syntax	The clauses CREATE LOGIN... PASSWORD, ...DEFAULT_DATABASE, and ...DEFAULT_LANGUAGE are supported. The clause ALTER LOGIN... PASSWORD is supported, but the clause ALTER LOGIN... OLD_PASSWORD is not supported. Only log-in names that are members of the system administrator can modify passwords.
CREATE DATABASE Case-Sensitive Collation Rules	The CREATE DATABASE statement does not support case-sensitive collation rules.
CREATE DATABASE Keywords and Clauses	Options other than COLLATE and CONTAINMENT=NONE are not supported. The COLLATE clause can only accept the value set by babelfishpg_tsql.server_collation_name.
CREATE SCHEMA... Supported Clauses	You can use the CREATE SCHEMA command to create an empty schema. Use other commands to create schema objects.
CREATE, ALTER LOGIN Clauses Supporting Limited Syntax	The clauses CREATE LOGIN... PASSWORD, ...DEFAULT_DATABASE, and ...DEFAULT_LANGUAGE are supported. The clause ALTER LOGIN... PASSWORD is supported, but the clause ALTER LOGIN... OLD_PASSWORD is not supported. Only log-in names that are members of the system administrator can modify passwords.
LOGIN Objects	All options for LOGIN objects are supported, except for the following: PASSWORD, DEFAULT_DATABASE, ENABLE, DISABLE.

Database ID Values	The primary database and tempdb database will not be database IDs 1 and 2.
Identifiers Exceeding 63 Characters	PostgreSQL supports up to 63 characters for identifiers. The MSSQL Compatible Edition will convert identifiers exceeding 63 characters into names containing the original name hash.
Support for IDENTITY Columns	IDENTITY columns are supported for data types tinyint, smallint, int, bigint, numeric, and decimal. SQL Server supports up to 38 digits of precision for the data types numeric and decimal in IDENTITY columns. PostgreSQL supports up to 19 digits of precision for the data types numeric and decimal in IDENTITY columns.
Using IGNORE_DUP_KEY in Indexes	The syntax for creating an index with IGNORE_DUP_KEY will create an index as if this attribute is omitted.
Indexes Containing More Than 32 Columns	Indexes cannot contain more than 32 columns. The number of included index columns is counted towards the maximum in PostgreSQL, but not in SQL Server.
Index (Clustered)	The creation of a clustered index is as if NONCLUSTERED is specified.
Index Clause	Ignore the following clauses: FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX, STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUENTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP, and DATA_COMPRESSION.
NEWSEQUENTIALID Function	Implemented as NEWID; does not guarantee sequential behavior. When NEWSEQUENTIALID is called, PostgreSQL generates a new GUID value.
OUTER APPLY	SQL Server's lateral joins are not supported. PostgreSQL offers SQL syntax for lateral joins, but the behavior is different.
OUTPUT Clause Is Supported with the Following Limitations:	Simultaneous use of OUTPUT and OUTPUT INTO in the same DML query is not supported. Referring to non-target tables in the OUTPUT clause for UPDATE or DELETE operations is not allowed. `OUTPUT... DELETED *` , `INSERTED *` in the same query is not supported.
Procedure or Function Parameter Limitations	The MSSQL Compatible Edition supports up to 100 parameters for procedures or functions.
RESTORE Statement	PostgreSQL snapshots of the database are different from the backup files created in SQL Server. Additionally, the granularity of backup and restore between SQL Server and PostgreSQL may also differ.
ROLLBACK: Table variables do not support transaction rollback.	If a rollback occurs in a session containing table variables, the process may be interrupted.
ROWGUIDCOL	This clause is currently ignored. Referencing $GUIDGOL in the query leads to a

	syntax error.
Support for SEQUENCE Objects	Data types tinyint, smallint, int, bigint, numeric, and decimal support SEQUENCE objects. For the data types numeric and decimal in SEQUENCE, PostgreSQL supports a precision of up to 19 digits.
Server-Level Roles	The sysadmin server-level role is supported. Other server-level roles apart from sysadmin are not supported.
Database-Level Roles Other Than db_owner	The db_owner database-level role is supported. Other database-level roles apart from db_owner are not supported.
SQL Keyword SPARSE	The keyword SPARSE is accepted and ignored.
SQL Keyword Clause ON filegroup	This clause is currently ignored.
SQL Keywords CLUSTERED and NONCLUSTERED for Indexes and Constraints	The MSSQL Compatible Edition accepts and ignores the keywords CLUSTERED and NONCLUSTERED.
sysdatabases.cmptlevel	sysdatabases.cmptlevel is always NULL.
tempdb Not Reinitialized upon Restart	Permanent objects (such as tables and procedures) created in tempdb are not deleted when the database is restarted.
TEXTIMAGE_ON Filegroup	The MSSQL Compatible Edition ignores the TEXTIMAGE_ON filegroup clause.
Time Precision	The MSSQL Compatible Edition supports precision up to six decimal places for fractional seconds. It is anticipated that this behavior will not have negative impacts.
Transaction Isolation Levels	Treats READUNCOMMITTED in the same way as READCOMMITTED. REPEATABLEREAD and SERIALIZABLE are not supported.
Virtual Computed Columns (Non-Persisted)	Virtual computed columns are created as persisted columns.
WITHOUT SCHEMABINDING Clause	This clause is not supported for functions, procedures, triggers, or views.

Features with Limited Support

Each new version of the MSSQL Compatible Edition adds support for more features, aligning better with T-SQL features and behaviors. Despite this, there are some unsupported features and differences in the current implementation. The following provides information on the differences in features between the MSSQL Compatible Edition and T-SQL, as well as some solutions or usage instructions.

Starting from MSSQL Compatible Edition 1.2.0, the following features currently have limited implementation:

SQL Server Catalogs (System Views)

The catalogs sys.sysconfigures, sys.syscurconfigs, and sys.configurations support only a single read-only configuration. sp_configure is currently not supported. For more information on some other SQL Server views implemented in the MSSQL Compatible Edition, see querying the database to access object information.

Granting Permissions

GRANT... TO PUBLIC is supported, but currently, GRANT..TO PUBLIC WITH GRANT OPTION is not supported.

SQL Server Ownership Chains and Permission Mechanism Limitations

In the MSSQL Compatible Edition, SQL Server ownership chains apply to views but not to stored procedures. This means to explicitly grant the access permissions to other database objects owned by the same owner as the calling procedure to the procedure itself. In SQL Server, granting the EXECUTE permission to the caller for the procedure is sufficient to invoke other objects owned by the same owner. In the MSSQL Compatible Edition, it is further required to grant the caller direct access permissions to the objects that the procedure accesses.

Resolution of Object (without schema name) References

When a SQL object (procedure, view, function, or trigger) references an object without specifying its schema name, SQL Server resolves the referenced object's schema name using the schema name of the SQL object where the reference occurs. Currently, the MSSQL Compatible Edition resolves the name differently by using the default schema of the database user executing the procedure.

Default Schema Changes, Sessions, and Connections

If a user changes the default schema using ALTER USER...WITH DEFAULT SCHEMA, the change takes effect immediately in that session. However, for other sessions connected under the same user account, the timing differs as follows:

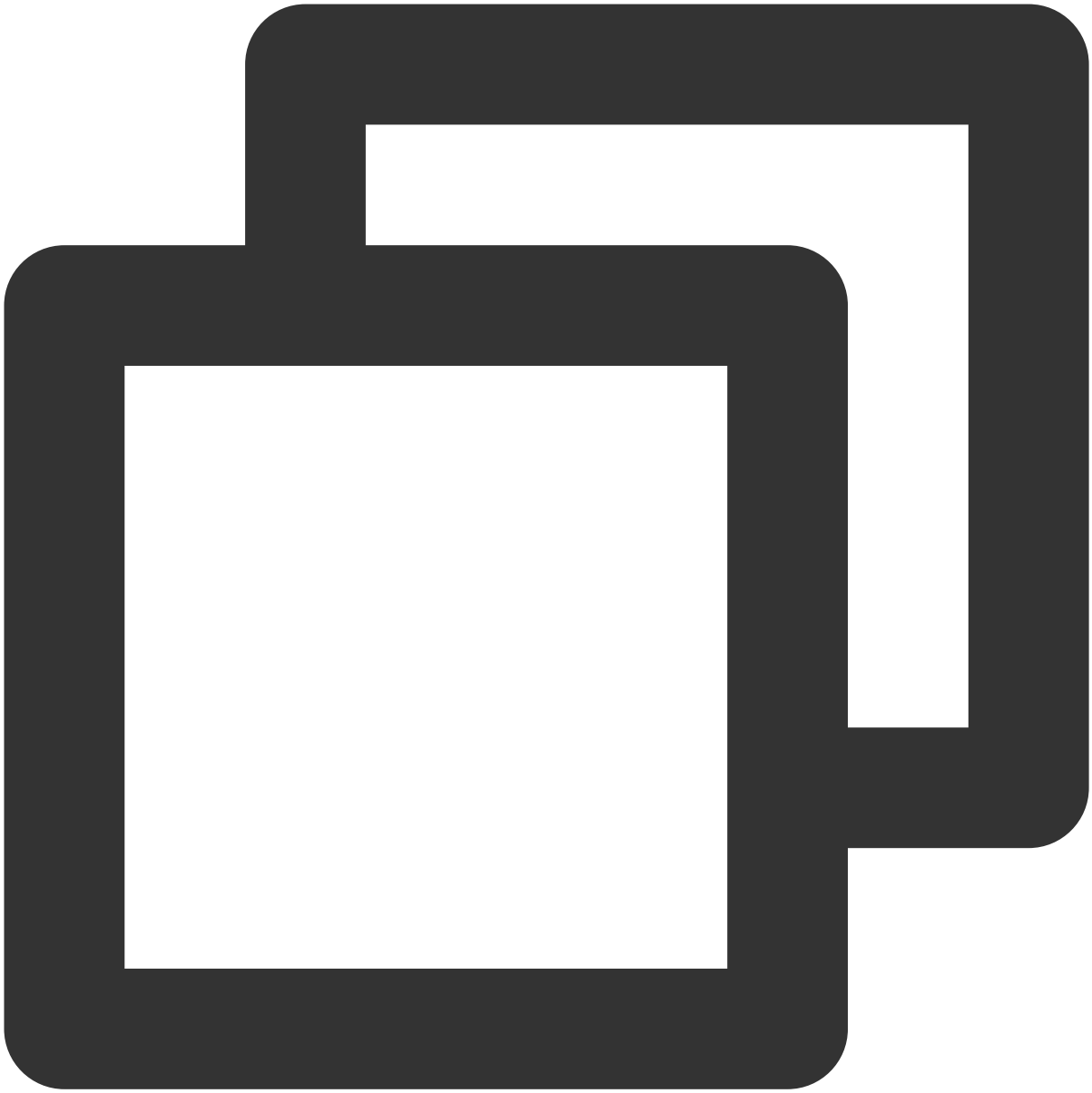
For SQL Server: This change takes effect immediately for this user across all other connections.

For the MSSQL Compatible Edition: This change will only take effect for this user in new connections.

Non-Deterministic Collation Rules and CHARINDEX

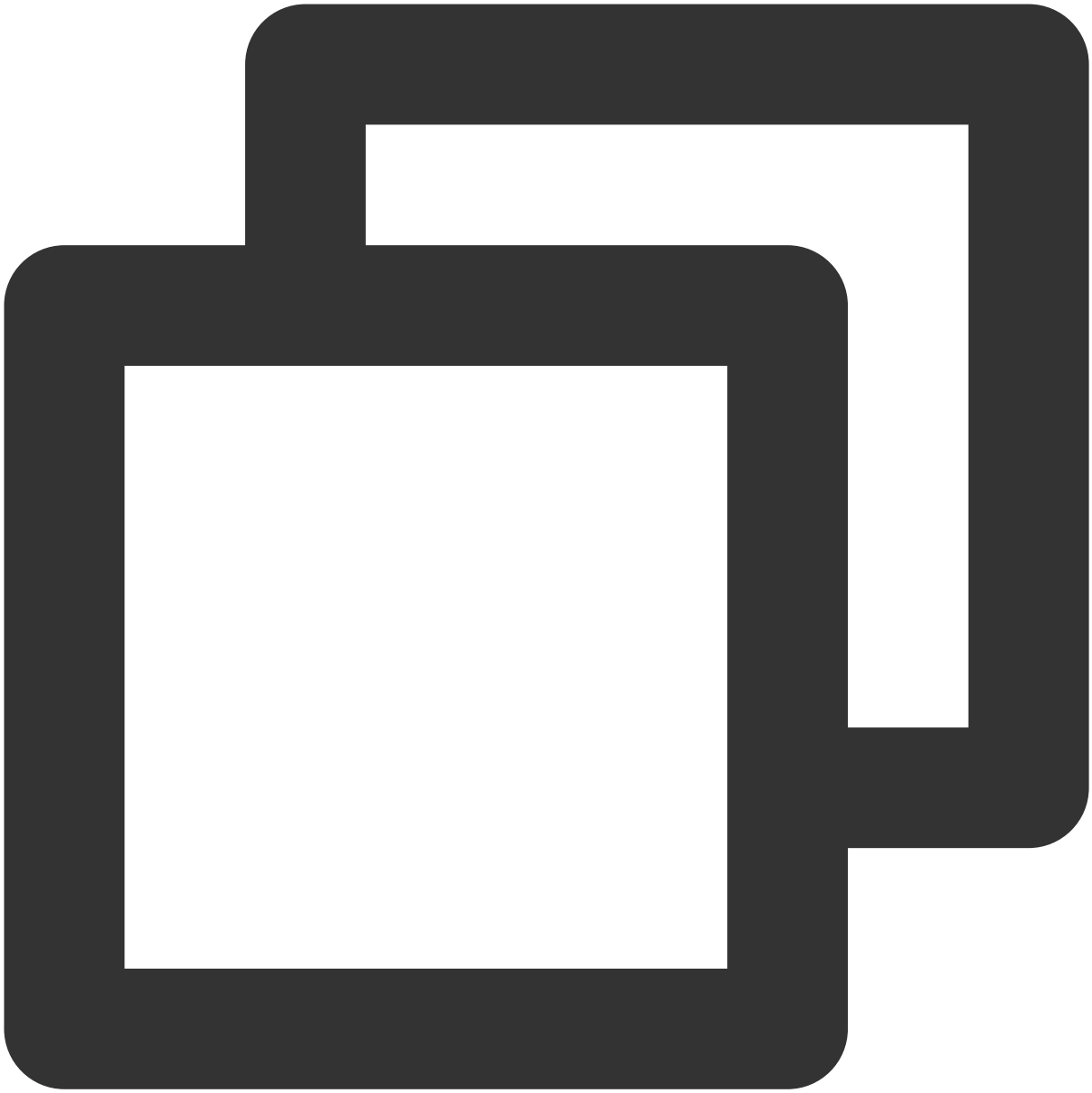
When the applicable collation rules are non-deterministic, currently, CHARINDEX cannot be used. Because the MSSQL Compatible Edition defaults to a case-insensitive collation rule, which is non-deterministic. You may receive a runtime error indicating "Substring search is not supported for non-deterministic collation rules". Until this error is resolved, the issue can be handled by either of the following methods:

Explicitly convert the expression to a case-sensitive collation rule, then convert both arguments to uppercase through applying LOWER or UPER. For example, SELECT charindex('x', a) FROM t1 would become as follows:



```
SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
```

Create a SQL function f_charindex, and then replace the CHARINDEX call with a call to the following function:



```
CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) returns int
AS
BEGIN
declare @i int = 1
WHILE len(@s2) >= len(@s1)
BEGIN
  if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
  set @i += 1
  set @s2 = substring(@s2,2,999999999)
END
return 0
```

```
END
go
```

Implementation of ROWVERSION and TIMESTAMP Data Types and escape hatch Settings

The MSSQL Compatible Edition now supports ROWVERSION and TIMESTAMP data types. To use ROWVERSION or TIMESTAMP in the MSSQL Compatible Edition, the escape hatch setting babelfishpg_tsql.escape_hatch_rowversion must be changed from the default value of strict to ignore. The implementation of ROWVERSION and TIMESTAMP data types in the MSSQL Compatible Edition is semantically similar to that in the SQL Server, with the following exception:

In SQL Server, each inserted or updated row is assigned a unique ROWVERSION/TIMESTAMP value. In the MSSQL Compatible Edition, every row inserted by the same statement is assigned the same ROWVERSION/TIMESTAMP value.

For instance, when an UPDATE statement or an INSERT-SELECT statement affects multiple rows, in SQL Server, the affected rows each have different values in their ROWVERSION/TIMESTAMP column. In the MSSQL Compatible Edition, the rows have the same value.

In SQL Server, when you create a new table using SELECT-INTO, you can convert explicit values (such as NULL) into the ROWVERSION/TIMESTAMP column to be created. When you perform the same operation in the MSSQL Compatible Edition, the system will assign an actual ROWVERSION/TIMESTAMP value to each row in the new table.

Note:

These subtle differences in the ROWVERSION/TIMESTAMP data types should not negatively impact applications running on the MSSQL Compatible Edition.

Pattern Creation, Ownership, and Permissions

SQL Server and the MSSQL Compatible Edition have different permissions for non-DBO users to create objects within schemas created by the database owner (using CREATE SCHEMA…AUTHORIZATION DBO), as shown in the table below:

Database users (non-DBO) can perform the following actions:	SQL Server	Babelfish
Can objects be created in the schema without extra authorization from the DBO?	No	Yes
Can referenced objects created by the DBO in the schema be used without extra authorization?	Yes	No