DBMaker

Stored Procedure Transaction User's Guide

Version: 01.00

Document No: 54/DBM547-T06262025-01-SPTU

Author: Production Team
Syscom Computer Engineering CO.

Print Date: June 26, 2025

Table of Content

1. Introduction

Welcome to the DBMaker Stored Procedure Transaction User's Manual. This guide discusses the details about transaction operations in SQL stored procedure in DBMaker 5.4.7 and later versions. This document will only introduce transaction operations, if users need more information about SQL Stored Procedure, please refer to *SQL Stored Procedure User's Guide*.

After version 5.4.7, A new clause TRANSACTION ON/OFF is added and allowed stored procedure to execute transaction commands such as COMMIT, ROLLBACK, SET TRANSACTION. This can greatly expand the usage scenarios of stored procedures.

Stored procedure that supports transaction is different from a general stored procedure. In the following chapters, they will be called as Transaction Stored Procedure and General Stored Procedure respectively.

1.1. SQL Stored Procedure

A **SQL Stored Procedure** is a precompiled collection of one or more SQL statements that can be executed as a single unit. Stored procedures are created and stored within a relational database management system (RDBMS) and can be called to perform specific operations like querying, inserting, updating, or deleting data. They help improve performance by reducing the amount of repetitive SQL code and promoting code reusability.

1.2. Transaction Commands

In a database, a transaction is a work unit that is composed of one or more SQL statements. It is an atomic operation. That means it should either complete a series of statements entirely or do nothing at all. Serial, atomic, permanent, consistent, and isolated are the properties of a transaction.

The following are related transaction commands, these commands can be executed in the SQL stored procedure in DBMaker 5.4.7 and later versions:

```
SAVEPOINT
COMMIT
ROLLBACK
SET AUTOCOMMIT ON/OFF
```

NOTE In a stored procedure without **TRANSACTION ON** clause, using the syntaxes above will return a syntax error.

2. SQL Stored Procedure with Transaction Commands

This chapter will show users how to create procedure with transaction commands.

2.1. SQL Stored Procedure Syntax

With the new clause TRANSACTION ON/OFF, users can execute transaction commands into SQL stored procedure, the following are SQL stored procedure's syntax

```
CREATE PROCEDURE SP(...)
LANGUAGE SQL
[transaction-clause]
BEGIN
    ...
END;
```

2.1.1. Transaction Clause Syntax

This is a new added keyword in DBMaker 5.4.7. Users can set transaction on/off to decide if this stored procedure contains transaction commands. The default option is ON.

```
<(transaction-clause)>
    transaction-clause ::=
         [TRANSACTION [ON/OFF]]
```

The following procedure shows the how to enable transaction options in stored procedure, this means the stored procedure will be defined as a transaction stored procedure:

```
CREATE PROCEDURE SP_NAME(...)
LANGUAGE SQL
TRANSACTION
BEGIN
    ...
END;

CREATE PROCEDURE SP_NAME(...)
LANGUAGE SQL
TRANSACTION ON
BEGIN
    ...
END;
```

The following procedure shows how to disable transaction options in stored procedure, this means the stored procedure will be defined as a general stored procedure:

```
CREATE PROCEDURE SP_NAME(...)
LANGUAGE SQL
TRANSACTION OFF
BEGIN
    ...
END;
```

2.2. Supported Transaction Commands

The following commands are supported when **TRANSACTION ON** is set.

```
COMMIT [WORK]

ROLLBACK [WORK]

SET AUTOCOMMIT ON

SET AUTOCOMMIT OFF
```

NOTE If users don't set **TRANSACTION ON** in the stored procedure and use the commands above, a syntax error will be returned.

2.2.1. COMMIT

Standard SQL commands. Users can permanently save the modifications made by the current transaction to the database.

Example:

The following syntax shows how to execute **COMMIT** in a stored procedure.

```
CREATE PROCEDURE SP4
LANGUAGE SQL
TRANSACTION
BEGIN
  DECLARE CC INT;
  SET CC=1;
  WHILE (CC <= 1000) DO
    INSERT INTO TAB VALUES(...);
    SET CC = CC+1;
  END WHILE;
  COMMIT;
END;
```

2.2.2. ROLLBACK

Standard SQL commands. Used to undo all changes made in the current transaction and restore the transaction to the state before it started.

Example:

The following syntax shows how to execute ROLLBACK in a stored procedure.

```
CREATE PROCEDURE SP5
LANGUAGE SQL
TRANSACTION ON
BEGIN
  DECLARE CC INT;
  SET CC=1;
  WHILE (CC <= 1000) DO
    INSERT INTO TAB VALUES(...);
    SET CC = CC+1;
  END WHILE;
  ROLLBACK;
END;
```

2.2.3. SET AUTOCOMMIT ON/OFF

The **SET AUTOCOMMIT** command controls the automatic commit behavior of database transactions.

Syntax of SET AUTOCOMMIT:

```
SET AUTOCOMMIT ON/OFF
```

ON/OFF

determines whether to turn on or off automatic submission

The default behavior for transaction-enabled stored procedures is SET AUTOCOMMIT OFF,

Turn off automatic commit, and then you need to explicitly use COMMIT or ROLLBACK to commit or rollback the transaction.

Or manually turn on automatic submission according to the situation

Turning off automatic submission can improve the completeness and consistency of your data, especially in a series of related operations, ensuring that either all of them succeed or all of them fail.

Example:

```
CREATE PROCEDURE SP6
LANGUAGE SQL
TRANSACTION ON
BEGIN
  DECLARE CC INT;
  SET CC=1;

  SET AUTOCOMMIT ON;

  WHILE (CC <= 1000) DO
    INSERT INTO TAB VALUES(...);
    SET CC = CC+1;
  END WHILE;

  SET AUTOCOMMIT OFF;

  SET CC=1;
  WHILE (CC <= 1000) DO
    INSERT INTO TAB VALUES(...);
    SET CC = CC+1;
  END WHILE;

  COMMIT;
END;
```

3. Transaction Stored Procedures and General Stored Procedures

3.1. General Stored Procedures

DBMaker's general stored procedure treats the operations within the stored procedure as one command. From a transaction perspective, every stored procedure is atomic. All operations are inseparable. All operations in the stored procedure will either succeed or fail.

Therefore, when a normal stored procedure encounters an error, it will roll back to the state before the stored procedure was executed, thereby ensuring the atomicity of the transaction.

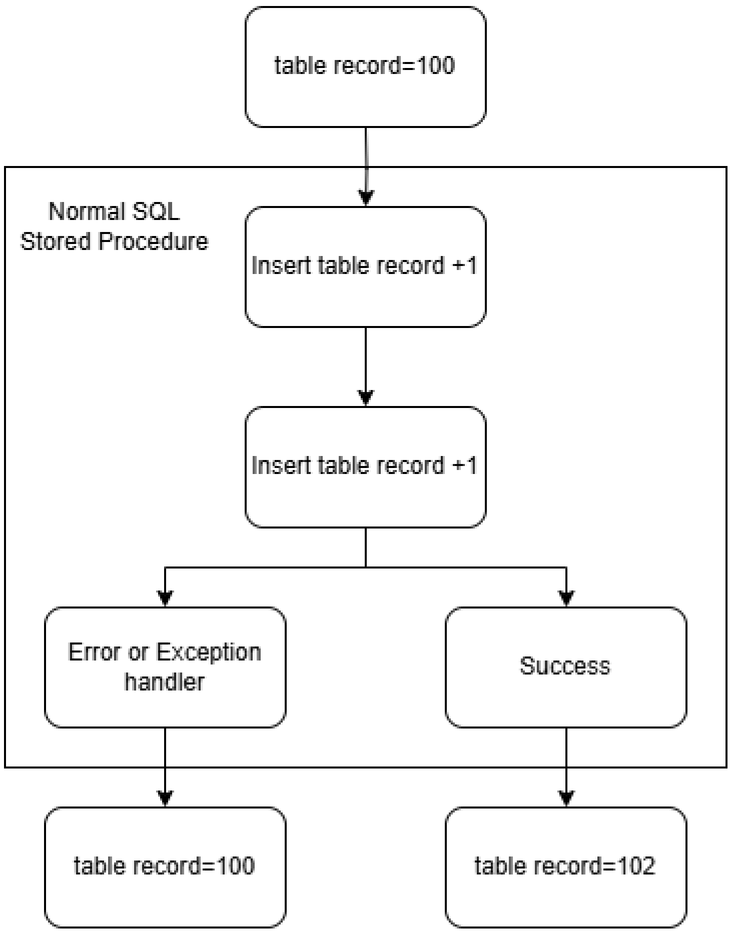

Figure: Normal Stored Procedure flow chart

3.2. Transaction Stored Procedures

The stored procedure that supports transactions can operate each SQL command. The atomic rows of transaction processing will be executed into each SQL command.

Transaction commands can be included in stored procedures that support transactions, and can be logged just like normal operations.

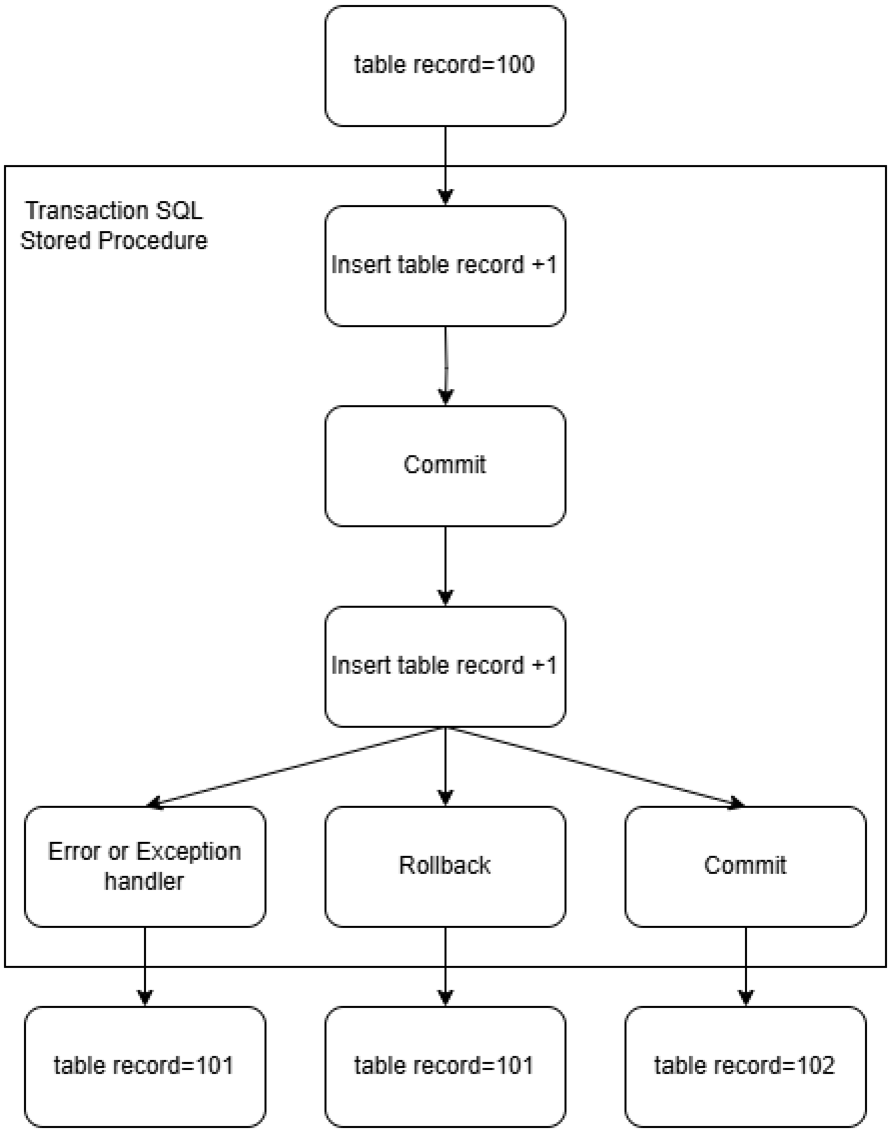

Figure: Transaction Stored Procedure flow chart

Please distinguish between different situations when using stored procedures. If users need to use a transaction processing flow, please use a transaction stored procedure.

Usually, stored procedures can call other stored procedures. But to ensure the consistency for transaction commands, general stored procedures cannot call transaction stored procedures. On the contrary transaction stored procedures can call both general and transaction stored procedures.