Steps to Create and use updatable ledger tables

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:35 AM PDT

Create and use updatable ledger tables

This page shows you how to create an updatable ledger table in Azure SQL Database, insert values into your updatable ledger table, make updates to the data, and view the results using the ledger view. As an example, we will use a simple banking application tracking a banking customers balance in their account. This will give you a practical example of the relationship between the updatable ledger table and it's corresponding history table and ledger view.

Prerequisite

Have an existing Azure SQL Database with SQL Ledger feature turned on .

Create an updatable ledger table

We will create a simple account balance table with the following schema.

```
Column name     Data type       Description
CustomerID      int     Customer ID - Primary key clustered
LastName        varchar (50)    Customer last name
FirstName       varchar (50)    Customer first name
Balance decimal (10,2)  Account balance
```

1. Using either SQL Server Management Studio or Azure Data Studio, create a new schema and table called [Account].[Balance].

```
CREATE SCHEMA [Account]
GO

CREATE TABLE [Account].[Balance]
(
    [CustomerID] INT NOT NULL PRIMARY KEY CLUSTERED,
    [LastName] VARCHAR (50) NOT NULL,
    [FirstName] VARCHAR (50) NOT NULL,
    [Balance] DECIMAL (10,2) NOT NULL
)
WITH
(
        SYSTEM_VERSIONING = ON,
        LEDGER = ON
);
GO
[!NOTE] Specifying the LEDGER = ON argument is optional if you enabled ledger when you created your Azure SQL
```

2. When your updatable ledger table is created, the corresponding history table and ledger view are also created. Execute the following T-SQL to see these new tables.

```
SELECT
ts.[name] + '.' + t.[name] AS [ledger_table_name]
, hs.[name] + '.' + h.[name] AS [history_table_name]
, vs.[name] + '.' + v.[name] AS [ledger_view_name]
FROM sys.tables AS t
JOIN sys.tables AS h ON (h.[object_id] = t.[history_table_id])
JOIN sys.views v ON (v.[object_id] = t.[ledger_view_id])
JOIN sys.schemas ts ON (ts.[schema_id] = t.[schema_id])
JOIN sys.schemas hs ON (hs.[schema_id] = h.[schema_id])
JOIN sys.schemas vs ON (vs.[schema_id] = v.[schema_id])
NEED THIS IMAGE (C:\Users\janders\AppData\Roaming\Typora\typora-user-images\image-20210324104307011.png)
```

3. Insert a customer, "Nick Jones", as a new customer with an opening balance of $50.

```
INSERT INTO [Account].[Balance]
VALUES (1, 'Jones', 'Nick', 50)
```

4. Insert 3 new customers, "John," "Joe" and "Mary" as new customers with opening balances of $500, $30 and $200.

```
INSERT INTO [Account].[Balance]
VALUES (2, 'Smith', 'John', 500),
(3, 'Smith', 'Joe', 30),
(4, 'Michaels', 'Mary', 200)
```

5. View the the [Account].[Balance] updatable ledger table, specifying the system-generated hidden columns added to the table.

```
SELECT *
      ,[ledger_start_transaction_id]
      ,[ledger_end_transaction_id]
      ,[ledger_start_sequence_number]
      ,[ledger_end_sequence_number]
FROM [Account].[Balance]
```

6. In the results window, you will first see the values inserted by your T-SQL commands, along with the system metadata that is used for data lineage purposes.

ledger_start_transaction_id notes the unique transaction ID associated to the transaction that inserted the data. Since John, Joe and Mary were inserted using the same transaction, they share the same transaction ID. **ledger_start_sequence_number** notes the order by which values were inserted by the transaction.

7. Update Nick's balance from $50 to $100.

```
UPDATE [Account].[Balance] SET [Balance] = 100
WHERE [CustomerID] = 1
```

8. Get the unique name of your history table. You will need this for the next step.

```
SELECT
ts.[name] + '.' + t.[name] AS [ledger_table_name]
, hs.[name] + '.' + h.[name] AS [history_table_name]
, vs.[name] + '.' + v.[name] AS [ledger_view_name]
FROM sys.tables AS t
JOIN sys.tables AS h ON (h.[object_id] = t.[history_table_id])
JOIN sys.views v ON (v.[object_id] = t.[ledger_view_id])
JOIN sys.schemas ts ON (ts.[schema_id] = t.[schema_id])
JOIN sys.schemas hs ON (hs.[schema_id] = h.[schema_id])
JOIN sys.schemas vs ON (vs.[schema_id] = v.[schema_id])
```

9. View the the [Account].[Balance] updatable ledger table, along with it's corresponding history table and ledger view.

```
SELECT *
      ,[ledger_start_transaction_id]
      ,[ledger_end_transaction_id]
      ,[ledger_start_sequence_number]
      ,[ledger_end_sequence_number]
FROM [Account].[Balance]
GO

SELECT * FROM <Your unique history table name>
GO

SELECT * FROM Account.Balance_Ledger
ORDER BY ledger_transaction_id
GO
```

10. Nick's account balance has been successfully updated in the updatable ledger table to $100.

The history table now shows the previous balance of $50 for Nick.

11. The ledger view shows that updating the ledger table is a DELETE of the original row with $50 as the balance with a corresponding INSERT of a new row with $100 with the new balance for Nick.

How good have you found this content?