Sebastian ŁACHECIŃSKI

Uniwersytet Łódzki, Instytut Logistyki i Informatyki, Katedra Informatyki Ekonomicznej
ORCID: 0000-0001-8933-5197

doi:10.15199/48.2022.10.05

Składowanie danych temporalnych dla wymiaru czasu transakcyjnego na platformie MariaDB

Streszczenie. *Celem artykułu jest wskazanie zakresu implementacji obsługi danych temporalnych dla czasu transakcyjnego w środowisku MariaDB oraz określenie stopnia zgodności tej implementacji z zapisami dotyczącymi temporalnych rozszerzeń języka SQL zawartych w standardzie ISO/IEC 9075 w wersji SQL:2011, a także prezentacja możliwości składowania danych temporalnych w środowisku MariaDB.*

Abstract*. The aim of the article is an indication the scope of the implementation of temporal data support for transaction time in the MariaDB environment and determining the degree of compliance of this implementation with the provisions on temporal extensions of the SQL language of the ISO/IEC 9075 standard in the SQL: 2011 version, as well as to present possibility storage of temporal data by MariaDB environment. (**Storage of temporal data for the transaction time on the MariaDB platform**).*

Słowa kluczowe: temporalne bazy danych, temporalne tabele, temporalne operatory, czas transakcyjny, SQL:2011, MariaDB.
Keywords: temporal databases, temporal tables, temporal operators, transaction time, SQL:2011, MariaDB.

Wstęp

Artykuł ten stanowi kolejny cykl rozważań na temat obsługi danych temporalnych w systemach baz danych opartych o relacyjny model danych. Poprzednie artykuły poświęcone zostały modelowaniu danych temporalnych w relacyjnym modelu danych [1], rozwojowi języka SQL i standardu ISO/IEC 9075 ze szczególnym uwzględnieniem składni pozwalającej składować i przetwarzać dane w RDBMS [2], a także możliwościom składowania i przetwarzania danych temporalnych na platformach MS SQL Server i Azure SQL Database [3] oraz na platformie Oracle [4,5]. Artykuł ten jest kontynuacją rozważań nt. możliwości składowania i przetwarzania danych temporalnych oferowanych przez serwery SQL. W całości poświęcony został on możliwościom składowania danych temporalnych na platformie MariaDB z uwzględnieniem wymiaru czasu transakcyjnego. Przeprowadzona została analiza zakresu możliwości magazynowania danych temporalnych dostępnych na platformie MariaDB, dla czasu transakcyjnego. Celem tej analizy było zidentyfikowanie stopnia zgodności obsługi danych temporalnych z wymaganiami standardu ISO/IEC 9075, w szczególności z wersją standardu SQL:2011. Ponadto zaprezentowane zostały możliwości i przykłady tworzenia tabel wersjonowanych systemowo, składujących dane temporalne dla wymiaru czasu transakcyjnego w środowisku MariaDB.

Podstawowe wymagania składowania i obsługi danych temporalnych na podstawie specyfikacji standardu SQL:2011

Główne elementy dotyczące obsługi danych temporalnych, które zostały wprowadzone w standardzie SQL 2011 to [6,7,8]:

- definicja okresu czasu,
- tabele temporalne wersjonowane aplikacyjnie lub systemowo,
- tabele bitemporalne (wersjonowane aplikacyjnie i systemowo),
- możliwość aktualizacji i usuwania rekordów z określonego przedziału czasowego,
- temporalne ograniczenie klucza podstawowego,
- temporalne ograniczenie integralności referencyjnej,
- nowe predykaty czasowe dla interwałów czasowych.

Składowanie danych temporalnych w MariaDB

MariaDB to system RDBMS o otwartym kodzie źródłowym na licencji GNU GPL. Zachowuje on kompatybilność wsteczną z serwerem MySQL, ale zawiera także wiele nowych rozwiązań niedostępnych na serwerze MySQL. Serwer MariaDB został opracowany w 2009 roku m.in. przy udziale współtwórców MySQL-a, w celu gwarancji utrzymania otwartego źródła serwera, po przejęciu firmy Sun Microsystems wraz z serwerem MySQL przez korporację Oracle w 2010 roku. Przejęcie serwera MySQL nie dawało pewności, co do dalszego jego istnienia na rynku oraz jego dalszej przyszłości.

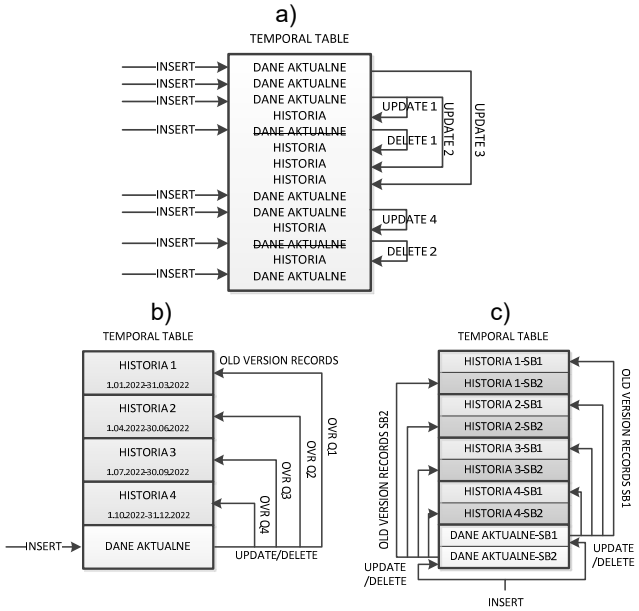
Obsługa danych temporalnych w środowisku MariaDB została wprowadzona w wersji 10.3.4, w 2017 roku. Dotyczyła ona wymiaru czasu transakcyjnego w odniesieniu do tabel wersjonowanych systemowo. Obsługa wymiaru czasu rzeczywistego została zaimplementowana w wersji 10.4.3, dla tabel wersjonowanych aplikacyjnie w 2018 roku. W tej wersji także dostępna stała się obsługa tabel bitemporalnych [9].

Tabele wersjonowane systemowo dla obsługi czasu transakcyjnego przechowują oprócz aktualnych danych także historię wszystkich wprowadzonych zmian. Takie rozwiązanie pozwala analizować dane dla dowolnej chwili czasowej oraz porównywać dane dla wybranych punktów w czasie. Ponadto mechanizm ten oferuje rozbudowaną składnię dla zapytań temporalnych, a co najważniejsze daje gwarancję niezmienności danych historycznych.

Silnik bazodanowy dla każdej instrukcji DML operującej na tabeli temporalnej automatycznie generuje (w zależności od jej konstrukcji) wartości daty i czasu, bądź też numeru identyfikacyjnego transakcji. Użytkownik nie może w żaden sposób ingerować w te wartości.

Istnieje także możliwość oddzielnego składowania danych temporalnych od danych aktualnych za pomocą podziału tabeli temporalnej na partycje. Niewątpliwie taka strategia posiada swoje zalety. Zapytania bazujące tylko na danych aktualnych będą uzyskiwały dostęp tylko do jednej partycji, w obrębie, której są one składowane, co znacznie zwiększy efektywność przetwarzania takiego zapytania. W przeciwnym razie, gdy dane temporalne przechowywane są razem z danymi aktualnymi w jednym obszarze, znacząco zwiększa się rozmiar tabeli, a co za tym idzie wydłuża się czas przetwarzania danych. Separowanie danych aktualnych i temporalnych tabeli temporalnej wersjonowanej systemowo jest zarazem domyślną opcją dostępną od wersji 10.5 serwera MariaDB [10,11].

Na rysunku 1 zaprezentowano mechanizm składowania danych temporalnych na platformie MariaDB dla wariantów: a) składowanie danych aktualnych i temporalnych w jednym wspólnym obszarze, b) składowanie danych aktualnych i temporalnych w oddzielnych partycjach, c) składowanie danych aktualnych i temporalnych w oddzielnych partycjach podzielonych na subpartycje.



Rys.1. Mechanizm składowania danych temporalnych z obsługą czasu transakcyjnego na platformie MariaDB, źródło: opracowanie własne

Tworzenie tabeli temporalnej z obsługą czasu transakcyjnego w MariaDB

Tabela temporalna wersjonowana systemowo musi mieć zdefiniowane dwie kolumny typu TIMESTAMP definiujące odpowiednio początek i koniec przedziału czasowego. Pola te mogą być określane zarówno w sposób jawny jak i niejawny. Jawne znaczniki posiadają nazwy zdefiniowane przez użytkownika. Niejawne znaczniki mają przypisane przez system nazwy: ROW_START i ROW_END [10-12].

W tabeli 1 zestawiono wykaz systemowych typów danych kategorii data i czas dostępnych na platformie MariaDB wraz z zakresem składowanych wartości [10,13].

Tabela 1. Systemowe typy danych kategorii data i czas dostępne na platformie MariaDB

Typ danych	Zakres	Dokładność
Date	1000-01-01 ÷9999-12-31	1 dzień
Time	-838:59:59.999999 ÷838:59:59.999999	0.1s÷1μs
DateTime	1000-01-01 00:00:00.000000 ÷9999-12-31 23:59:59.999999	0.1s÷1μs
TimeStamp	1970-01-01 00:00:00 ÷2038-01-19 03:14:07	0.1s÷1μs
Year	1901÷2155 70-69 w formacie dwucyfrowym	1 rok

W przypadku jawnego sposobu definicji tych pól, ich zawartość domyślnie będzie widoczna w zbiorze wynikowym dla zapytania wybierającego, bazującego na takiej tabeli temporalnej. Jeżeli na etapie konstruowania tabeli temporalnej, dla pól przechowujących początek i koniec przedziału czasowego zastosowany zostanie parametr INVISBILE, wówczas zawartość tych kolumn nie będzie domyślnie dostępna w zbiorze wynikowym zapytania temporalnego. W celu wyświetlenia ich zawartości należy na liście pól w zapytaniu wybierającym wymienić także nazwy pól dla znaczników czasowych.

W przypadku, gdy tabela temporalna tworzona jest z niejawnie definiowanymi kolumnami dla znaczników czasowych, domyślnie ich zawartość nie jest dostępna w zapytaniu wybierającym. Wartości stempla czasowego składowane są w pseudokolumnach ROW_START i ROW_END. Użycie tych pseudokolumn na liście pól w zapytaniu wybierającym pozwala na wyświetlenie ich zawartości. Wartości NULL są niedopuszczalne dla kolumn znaczników czasowych.

Tabele temporalne wersjonowane systemowo, niejawnie dodają kolumnę końca przedziału czasowego zdefiniowaną jawnie przez użytkownika do klucza podstawowego. W przypadku tworzenia tabeli temporalnej z niejawnie definiowanymi znacznikami czasowymi, w skład klucza podstawowego dołączana jest pseudokolumna ROW_END.

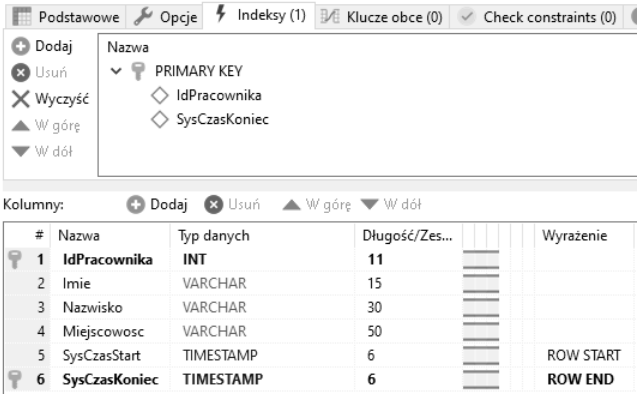
Sposoby tworzenia tabeli temporalnej wersjonowanej systemowo w środowisku MariaDB[10-12]:

- tabela temporalna z jawnie definiowanymi kolumnami dla znaczników czasowych,
- tabela temporalna z jawnie definiowanymi kolumnami dla znaczników czasowych, z jednoczesnym wyłączeniem ich widoczności,
- tabela temporalna z niejawnie definiowanymi kolumnami dla znaczników czasowych,
- adaptacja istniejącej tabeli do tabeli temporalnej z jawnie definiowanymi kolumnami dla znaczników czasowych,
- adaptacja istniejącej tabeli do tabeli temporalnej z niejawnie definiowanymi kolumnami dla znaczników czasowych,
- tabela temporalna z wersjonowaniem tylko wybranych kolumn, a nie całej tabeli,
- partycjonowana tabela temporalna -osobno składująca dane aktualne i dane temporalne na oddzielnych partycjach,
- partycjonowana tabela temporalna z subpartycjami,
- tabela temporalna z dokładnością do transakcji.

Poniżej przedstawiono przykład tworzenia tabeli temporalnej wersjonowanej systemowo z jawnie definiowanymi znacznikami czasowymi:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika),
SysCzasStart TIMESTAMP(6) GENERATED ALWAYS AS ROW
START,
SysCzasKoniec TIMESTAMP(6) GENERATED ALWAYS AS ROW
END,
PERIOD FOR SYSTEM_TIME(SysCzasStart, SysCzasKoniec)
)
WITH SYSTEM VERSIONING;
```

Na rysunku 2 przedstawiono strukturę utworzonej tabeli temporalnej wersjonowanej systemowo z jawnie definiowanymi znacznikami czasowymi. Ponadto uwidoczniona została także struktura indeksu klucza podstawowego tabeli temporalnej zawierająca kolumnę znacznika czasowego składującego koniec przedziału czasowego.



Rys.2. Struktura tabeli temporalnej wersjonowanej systemowo z jawnie definiowanymi znacznikami czasowymi, źródło: opracowanie własne

Poniżej przedstawiono przykład tworzenia tabeli temporalnej wersjonowanej systemowo z jawnie definiowanymi znacznikami czasowymi, ale domyślnie niewidoczną ich zawartością przez dołączenie klauzuli INVISIBLE:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika),
SysCzasStart TIMESTAMP(6) GENERATED ALWAYS AS ROW
START INVISIBLE,
SysCzasKoniec TIMESTAMP(6) GENERATED ALWAYS AS ROW END
INVISIBLE,
PERIOD FOR SYSTEM_TIME(SysCzasStart, SysCzasKoniec)
)
WITH SYSTEM VERSIONING;
```

Możliwe jest także użycie uproszczonej składni bez jawnego definiowania kolumn przechowujących wartości stempla czasowego dla czasookresu. Dostęp do wartości kolumn znaczników czasowych odbywa się za pośrednictwem pseudokolumn ROW_START i ROW_END. Poniżej przedstawiono przykład tworzenia tabeli wersjonowanej systemowo bez jawnie definiowanych znaczników czasowych:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING;
```

Innym sposobem utworzenia tabeli temporalnej jest przekształcenie istniejącej już tabeli na tabelę wersjonowaną systemowo. Poniżej przedstawiono przykład przekształcenia klasycznej tabeli, w tabelę temporalną z jawnie definiowanymi kolumnami znaczników czasowych:

```
ALTER TABLE pracownik
ADD SysCzasStart TIMESTAMP(6) GENERATED ALWAYS AS ROW
START,
ADD SysCzasKoniec TIMESTAMP(6) GENERATED ALWAYS AS ROW
END,
ADD PERIOD FOR SYSTEM_TIME(SysCzasStart,
SysCzasKoniec),
ADD SYSTEM VERSIONING;
```

Przekształcenie istniejącej tabeli, w tabelę temporalną wersjonowaną systemowo bez jawnego określenia kolumn dla znaczników czasowych zostało przedstawione poniżej:

```
ALTER TABLE pracownik ADD SYSTEM VERSIONING;
```

Tabela temporalna może być tworzona także z wersjonowaniem w odniesieniu do wybranych kolumn za pomocą klauzuli WITH SYSTEM VERSIONING. Użycie tej klauzuli skutkuje wersjonowaniem tabeli. Ma to sens, gdy pożądane jest przechowywanie historii tylko wybranych kolumn. Kolejny przykład ilustruje utworzenie tabeli temporalnej z wersjonowaniem tylko jednego pola tabeli:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL WITH SYSTEM
VERSIONING,
PRIMARY KEY(IdPracownika)
);
```

Tworzenie tabeli temporalnej z wersjonowaniem wybranych kolumn może być także definiowane poprzez wykluczenie określonej kolumny bądź kilku kolumn z wersjonowania za pomocą klauzuli WITHOUT SYSTEM VERSIONING. Poniżej przedstawiono przykład wykluczający z wersjonowania pola składujące imiona i nazwiska pracowników (ze względu na fakt, iż dane te bardzo rzadko lub praktycznie nigdy nie ulegają zmianie):

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL WITHOUT SYSTEM VERSIONING,
Nazwisko VARCHAR(30) NOT NULL WITHOUT SYSTEM
VERSIONING,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING;
```

Utworzenie partycjonowanej tabeli temporalnej wersjonowanej systemowo pozwala odseparować dane bieżące od danych temporalnych. Pozwala to zmniejszyć rozmiar tabeli. W przypadku, gdy historia trzymana jest razem z aktualnymi danymi zwiększa się rozmiar tabeli. Ponadto, gdy określone zapytanie bazuje tylko na danych aktualnych, skanowana jest tylko partycja przechowująca te dane, a nie cały obszar tabeli. Dzięki temu czas realizacji takiego zapytania także ulega skróceniu. Partycjonowana tabela temporalna musi posiadać minimum jedną partycję na dane aktualne i minimum jedną partycję na dane temporalne. Poniżej przedstawiono przykład tworzenia partycjonowanej tabeli wersjonowanej systemowo, która zawiera jedną partycje na dane aktualne i jedną na dane temporalne:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME
 (
   PARTITION partycja_historia HISTORY,
   PARTITION partycja_aktualna CURRENT
  );
```

Za pomocą poniższego zapytania można zapoznać się ze strukturą utworzonej partycjonowanej tabeli temporalnej:

```
SELECT partition_name AS PartitionName,
subpartition_name AS
SubpartitionName,partition_ordinal_position AS
PartitionOrdinalPosition,
partition_method AS
PartitionMethod,partition_description AS
PartitionDescription,table_rows AS TableRows FROM
information_schema.partitions
WHERE TABLE_SCHEMA='temporaldatabase2022' AND
TABLE_NAME = 'pracownik'
AND PARTITION_NAME IS NOT NULL;
```

Na rysunku 3 przedstawiono podział tabeli temporalnej na dwie partycje z przeznaczeniem na dane aktualne oraz dane temporalne.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
partycja_historia	(NULL)	1	SYSTEM_TIME	(NULL)	0
partycja_aktualna	(NULL)	2	SYSTEM_TIME	CURRENT	0

Rys.3. Struktura tabeli temporalnej wersjonowanej systemowo z podziałem na partycje, na dane aktualne i temporalne, źródło: opracowanie własne

Tworzone partycje mogą być rotowane wg rozmiaru zgromadzonych danych lub jednostki czasu. Poniżej przedstawiono kolejny przykład, w którym po przekroczeniu 500 000 rekordów w pierwszej partycji historycznej kolejne rekordy będą zapisywane w drugiej partycji historycznej. Po przekroczeniu kolejnych 500 000 rekordów kolejne będą zapisywane w trzeciej partycji historycznej. Z uwagi na fakt, iż nie istnieje więcej partycji historycznych, kolejne rekordy będą składowane w obszarze ostatniej partycji historycznej, pomimo wygenerowania systemowego komunikatu ostrzegawczego.

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME LIMIT 500000
 (
   PARTITION partycja_historia_0 HISTORY,
   PARTITION partycja_historia_1 HISTORY,
   PARTITION partycja_historia_2 HISTORY,
   PARTITION partycja_aktualna CURRENT
  );
```

Na rysunku 4 przedstawiono podział tabeli temporalnej na trzy partycje z przeznaczeniem na dane historyczne oraz jedną partycje na dane aktualne. Partycje rotowane są wg rozmiaru danych.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
partycja_historia_0	(NULL)	1	SYSTEM_TIME	(NULL)	0
partycja_historia_1	(NULL)	2	SYSTEM_TIME	(NULL)	0
partycja_historia_2	(NULL)	3	SYSTEM_TIME	(NULL)	0
partycja_aktualna	(NULL)	4	SYSTEM_TIME	CURRENT	0

Rys.4. Struktura tabeli temporalnej wersjonowanej systemowo z podziałem na trzy partycje na dane temporalne i jedną partycję na dane aktualne. Rotowanie partycji wg rozmiaru danych, źródło: opracowanie własne

Następny przykład ilustruje rotowanie partycji historycznych wg czasu. Dane temporalne z pierwszego kwartału składowane będą w pierwszej partycji, drugiego kwartału w drugiej partycji, trzeciego kwartału w trzeciej partycji, czwartego i kolejnych kwartałów w czwartej partycji historycznej (z uwagi na fakt, iż nie zdefiniowano więcej partycji historycznych). Dopuszczalne jest użycie następujących interwałów czasowych: MICROSECOND, SECOND, MINUTE, DAY, HOUR, WEEK, MONTH, QUARTER, YEAR.

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME INTERVAL 1 QUARTER
 (
   PARTITION partycja_historia_0 HISTORY,
   PARTITION partycja_historia_1 HISTORY,
   PARTITION partycja_historia_2 HISTORY,
   PARTITION partycja_historia_3 HISTORY,
   PARTITION partycja_aktualna CURRENT
  );
```

Na rysunku 5 przedstawiono podział tabeli temporalnej na cztery partycje z przeznaczeniem na dane temporalne oraz jedną partycje na dane aktualne. Partycje rotowane są wg czasu, co kwartał.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
partycja_historia_0	(NULL)	1	SYSTEM_TIME	2022-09-02 00:00:00	0
partycja_historia_1	(NULL)	2	SYSTEM_TIME	2022-12-01 23:00:00	0
partycja_historia_2	(NULL)	3	SYSTEM_TIME	2023-03-01 23:00:00	0
partycja_historia_3	(NULL)	4	SYSTEM_TIME	2023-06-02 00:00:00	0
partycja_aktualna	(NULL)	5	SYSTEM_TIME	CURRENT	0

Rys.5. Struktura tabeli temporalnej wersjonowanej systemowo z podziałem na cztery partycje na dane temporalne i jedną partycję na dane aktualne. Rotowanie partycji wg czasu, źródło: opracowanie własne

Istnieje także możliwość utworzenia partycjonowanej tabeli temporalnej wersjonowanej systemowo z podziałem każdej z partycji na subpartycję. Każda z partycji posiada taką samą liczbę subpartycji. W poniższym przykładzie zarówno partycja przechowująca dane aktualne jak również partycja przechowująca dane temporalne zostały podzielone na cztery subpartycje.

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME
SUBPARTITION BY KEY (IdPracownika)
 SUBPARTITIONS 4
 (
   PARTITION partycja_historia HISTORY,
   PARTITION partycja_aktualna CURRENT
  );
```

Na rysunku 6 przedstawiono podział partycjonowanej tabeli temporalnej na cztery subpartycje.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
partycja_historia	partycja_historiasp0	1	SYSTEM_TIME	(NULL)	0
partycja_historia	partycja_historiasp1	1	SYSTEM_TIME	(NULL)	0
partycja_historia	partycja_historiasp2	1	SYSTEM_TIME	(NULL)	0
partycja_historia	partycja_historiasp3	1	SYSTEM_TIME	(NULL)	0
partycja_aktualna	partycja_aktualnasp0	2	SYSTEM_TIME	CURRENT	0
partycja_aktualna	partycja_aktualnasp1	2	SYSTEM_TIME	CURRENT	0
partycja_aktualna	partycja_aktualnasp2	2	SYSTEM_TIME	CURRENT	0
partycja_aktualna	partycja_aktualnasp3	2	SYSTEM_TIME	CURRENT	0

Rys.6. Struktura tabeli temporalnej wersjonowanej systemowo z podziałem partycji aktualnej i historycznej na cztery subpartycje, źródło: opracowanie własne

Partycjonowanie tabeli wg danych aktualnych i temporalnych jest typowym przypadkiem użycia. Tworzenie tabeli partycjonowanej jest domyślnym sposobem tworzenia tabeli temporalnej wersjonowanej systemowo, dostępnym od wersji 10.5 serwera MariaDB. Można wówczas korzystać z uproszczonej składni tworzenia partycjonowanej tabeli temporalnej, co zostało przedstawione na poniższym przykładzie:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME;
```

Na rysunku 7 przedstawiono podział partycjonowanej tabeli temporalnej na partycje, utworzonej za pomocą uproszczonej składni.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
p0	(NULL)	1	SYSTEM_TIME	(NULL)	0
pn	(NULL)	2	SYSTEM_TIME	CURRENT	0

Rys.7. Struktura partycjonowanej tabeli temporalnej wersjonowanej systemowo, z podziałem na partycję, na dane aktualne i historyczne, źródło: opracowanie własne

Możliwe także jest określenie liczby partycji podczas tworzenia partycjonowanej tabeli temporalnej. Należy pamiętać o określeniu interwału czasowego w oparciu, o który będzie definiowany podział na partycje oraz wskazaniu określonej liczby tworzonych partycji. Przy czym minimalna liczba tworzonych partycji nie może być mniejsza od dwóch. Poniżej przedstawiono przykład tworzenia tabeli temporalnej zawierającej dwanaście partycji, dla każdego kolejnego miesiąca oraz jedną partycję na dane bieżące. Ta jednostka czasu stanowi jednocześnie kryterium podziału tabeli temporalnej na partycje.

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika)
)
WITH SYSTEM VERSIONING
PARTITION BY SYSTEM_TIME
INTERVAL 1 MONTH
PARTITIONS 13;
```

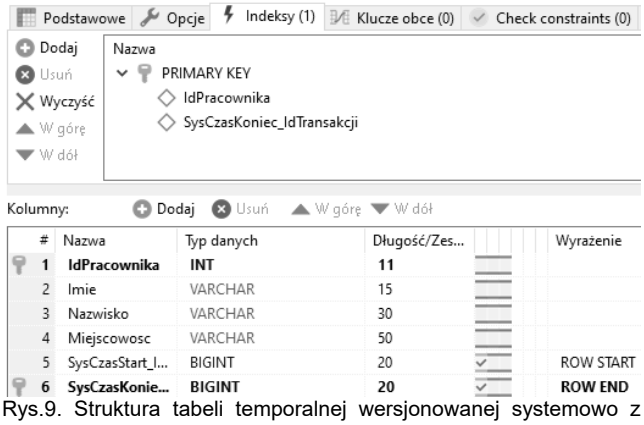
Na rysunku 8 przedstawiono podział partycjonowanej tabeli temporalnej utworzonej za pomocą uproszczonej składni, wersjonowanej systemowo na dwanaście partycji historycznych i jedną partycję na dane bieżące.

PartitionName	SubpartitionName	PartitionOrdinalPosition	PartitionMethod	PartitionDescription	TableRows
p0	(NULL)	1	SYSTEM_TIME	2022-07-02 00:00:00	0
p1	(NULL)	2	SYSTEM_TIME	2022-08-02 00:00:00	0
p2	(NULL)	3	SYSTEM_TIME	2022-09-02 00:00:00	0
p3	(NULL)	4	SYSTEM_TIME	2022-10-02 00:00:00	0
p4	(NULL)	5	SYSTEM_TIME	2022-11-01 23:00:00	0
p5	(NULL)	6	SYSTEM_TIME	2022-12-01 23:00:00	0
p6	(NULL)	7	SYSTEM_TIME	2023-01-01 23:00:00	0
p7	(NULL)	8	SYSTEM_TIME	2023-02-01 23:00:00	0
p8	(NULL)	9	SYSTEM_TIME	2023-03-01 23:00:00	0
p9	(NULL)	10	SYSTEM_TIME	2023-04-02 00:00:00	0
p10	(NULL)	11	SYSTEM_TIME	2023-05-02 00:00:00	0
p11	(NULL)	12	SYSTEM_TIME	2023-06-02 00:00:00	0
pn	(NULL)	13	SYSTEM_TIME	CURRENT	0

Rys.8. Struktura partycjonowanej tabeli temporalnej wersjonowanej systemowo z podziałem na dwanaście partycji na dane temporalne i jedną partycję na dane bieżące, źródło: opracowanie własne

Należy wspomnieć, że wykonanie określonej instrukcji DML na tabeli temporalnej nie zawsze będzie natychmiastowo widoczne w systemie. Może mieć to niepożądane skutki np. w przypadku krótkoterminowych analiz. Nie wszystkie dane będą uwzględnione w takiej analizie, bądź też pod uwagę nie będą wzięte najbardziej aktualne wartości, czy też uwzględnione zostaną wartości już usunięte z systemu. Wszystkie takie przypadki będą zniekształcać końcowy wynik analizy danych. Wyniki takiej analizy mogą prowadzić do przekłamanych wniosków i podjęcia błędnych, nieoptymalnych decyzji. Aby zapobiec takim przypadkom, na serwerze MariaDB wprowadzono możliwość tworzenia tabeli temporalnej z obsługą dokładnej historii transakcji (historia z dokładnością do transakcji). Dzięki temu wszystkie zmiany przeprowadzone w ramach transakcji na danych składowanych w tabeli temporalnej wersjonowanej systemowo, będą natychmiast widoczne w systemie. Dane takie będą widoczne w identycznej postaci, jak dla nowo nawiązanego połączenia wykonującego zapytanie wybierające, dla określonego punktu w czasie. Należy zwrócić uwagę, że np. rekordy dodane do tabeli przed wskazanym punktem w czasie, ale zatwierdzone w ramach transakcji już po tym punkcie, nie będą widoczne. Jednakże należy zaznaczyć, że funkcjonalność ta jest jedynie dostępna dla silnika InnoDB. W odróżnieniu od dotychczasowych konstrukcji tabel temporalnych wersjonowanych systemowo zamiast znaczników czasowych używane są pola składujące identyfikatory transakcji dla każdego rekordu, w oparciu o systemowy typ danych BIGINT UNSIGNED. Poniżej przedstawiono przykład tworzenia tabeli temporalnej wersjonowanej systemowo z dokładną historią transakcji:

```
CREATE TABLE pracownik
(
IdPracownika INT NOT NULL,
Imie VARCHAR(15) NOT NULL,
Nazwisko VARCHAR(30) NOT NULL,
Miejscowosc VARCHAR(50) NOT NULL,
PRIMARY KEY(IdPracownika),
SysCzasStart_IdTransakcji BIGINT UNSIGNED GENERATED
ALWAYS AS ROW START,
SysCzasKoniec_IdTransakcji BIGINT UNSIGNED GENERATED
ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME(SysCzasStart_IdTransakcji,
SysCzasKoniec_IdTransakcji)
)
WITH SYSTEM VERSIONING;
```



Rys.9. Struktura tabeli temporalnej wersjonowanej systemowo z jawnie definiowanymi znacznikami czasowymi, z uwzględnieniem historii zmian z dokładnością do pojedynczej transakcji, źródło: opracowanie własne

Na rysunku 9 przedstawiono strukturę utworzonej tabeli temporalnej wersjonowanej systemowo z jawnie

definiowanymi znacznikami czasowymi, z dokładną historią transakcji. Ponadto uwidoczniona została także struktura indeksu klucza podstawowego tabeli temporalnej zawierająca kolumnę składującą identyfikator transakcji dla ostatniej zmiany rekordu.

Podsumowanie

Dane temporalne dla wymiaru czasu transakcyjnego na platformie MariaDB składowane są w tabelach temporalnych wersjonowany systemowo. Wartości znaczników czasu lub identyfikatorów transakcji generowane są automatycznie przez system.

Platforma MariaDB oferuje szerokie spectrum różnorodnych wariantów tworzenia tabel temporalnych przechowujących dane temporalne w aspekcie czasu transakcyjnego. Oferuje ona możliwość tworzenia tabel temporalnych zarówno z jawnym i niejawnym sposobem definiowania kolumn znaczników czasowych. Pozwala także przekształcać istniejące już tabele do postaci tabeli temporalnej zarówno z jawnie i niejawnie definiowanymi znacznikami czasu. Co więcej pozwala tworzyć i przekształcać tabele temporalne z wybiórczym definiowaniem kolumn, które mają podlegać wersjonowaniu. Jest to rozwiązanie szczególnie przydatne, gdy z punktu widzenia wymagań biznesowych, rejestracji historii zmian podlega tylko pojedyncza kolumna lub niewielka liczba kolumn, w stosunku do całkowitej liczby kolumn tabeli. Kolejnym wartym podkreślenia udogodnieniem jest możliwość oddzielnego składowania danych aktualnych od danych temporalnych. Jest to możliwe poprzez implementację partycjonowania tabeli temporalnej na zadaną ilość partycji, a także subpartycji. Należy zwrócić uwagę, że jest to zarazem domyślna opcja tworzenia tabel temporalnych na platformie MariaDB od wersji 10.5. Funkcjonalność ta pozwala rozdzielić dane temporalne wg ustalonego kryterium podziału np. dane sprzedaży za każdy miesiąc, kwartał lub rok składowane są w oddzielnej partycji, w separacji od danych aktualnych. Dzięki temu podejściu znacznie wzrasta efektywność przeszukiwania i odpytywania takiej tabeli temporalnej. Skanowana jest tylko wybrana partycja historyczna, w której znajdują się dane temporalne należące do określonego przedziału czasowego. W przypadku pracy na danych bieżących skanowaniu podlega tylko i wyłącznie partycja składująca dane aktualne.

Istnieją także pewne ograniczenia w obsłudze tabel temporalnych m.in. kolumny, których wartości są generowane automatycznie nie mogą podlegać wersjonowaniu danych [10,11].

W kwestii zgodności z wytycznymi zawartymi w standardzie SQL:2011 odnośnie składowania i obsługi danych temporalnych dla wymiaru czasu transakcyjnego środowisko MariaDB zapewnia możliwość tworzenia i obsługi tabel wersjonowanych systemowo. Możliwe także jest użycie zarówno jawnych jak i niejawnych znaczników czasu. Ponadto platforma MariaDB obsługuje większość typów danych uwzględnionych w standardzie SQL:2011, istotnych dla przetwarzania danych temporalnych, poza typem INTERVAL. Udostępnia także predykaty czasowe umożliwiające temporalne przetwarzanie danych. Rozszerza ten zbiór o dodatkowy predykat temporalny w stosunku do standardu SQL:2011. Predykaty temporalne oraz przetwarzanie danych temporalnych zostaną szerzej zaprezentowane w kolejnym artykule. Z technicznego punktu widzenia możliwe jest także usunięcie zawartości tabeli temporalnej. Podobne rozwiązania dotyczące zarządzaniem czasem retencji danych dostępne są także m.in. na platformie MS SQL Server[3,14] czy Oracle[4,15]. Niemniej jednak postepowanie takie skutkuje utratą historii, co stanowi kontrę do postulatów standardu SQL:2011, który gwarantuje niezmienność danych temporalnych.

Artykuł ten w całości poświęcony został składowaniu danych temporalnych na platformie MariaDB dla wymiaru czasu transakcyjnego. Kolejnym etapem pracy będzie zbadanie stopnia realizacji założeń temporalnych rozszerzeń standardu SQL:2011, na platformie MariaDB w kontekście przetwarzania danych temporalnych dla wymiaru czasu transakcyjnego oraz prezentacja zaimplementowanych funkcjonalności temporalnych.

***Autorzy**: dr inż. Sebastian Łacheciński, Uniwersytet Łódzki, Instytut Logistyki i Informatyki, Katedra Informatyki Ekonomicznej, ul. Rewolucji 1905 r. 37, 90-214 Łódź, E-mail: sebastian.lachecinski@uni.lodz.pl*

LITERATURA

[1] Łacheciński S., Modelowanie danych temporalnych w relacyjnym modelu danych, Informatyka Ekonomiczna, 4(46) (2017), 90-107
[2] Łacheciński S., Składowanie i przetwarzanie danych temporalnych w świetle wymagań standardu języka SQL ISO-IEC 9075, Przegląd Elektrotechniczny, 96 (2020), nr 10, 184-191
[3] Łacheciński S., Obsługa danych temporalnych na platformie MS SQL Server i Azure SQL Database, Przegląd Elektrotechniczny, 96 (2020), nr 12, 95-101
[4] Łacheciński S., Obsługa danych temporalnych dla wymiaru czasu transakcyjnego na platformie ORACLE, Przegląd Elektrotechniczny, 97 (2021), nr 11, 159-163
[5] Łacheciński S., Obsługa danych temporalnych dla wymiaru czasu rzeczywistego na platformie ORACLE, Przegląd Elektrotechniczny, 97 (2021), nr 12, 86-91
[6] Date C.J., Darwen H., Lorentzos N., Time and relational theory Temporal Databases in the Relational Model and SQL, 2014, Morgan Kaufmann
[7] Kulkarni K., Jan-Eike Michels, 2012, *Temporal features in SQL:2011,* 34-43 https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf
[8] Temporal extension SQL: https://link.springer.com/content/pdf/10.1007%2F978-1-4899-7993-3_80729-1.pdf
[9] MariaDB Temporal Tables: https://mariadb.com/kb/en/temporal-tables/
[10] MariaDB Server Documentation: https://mariadb.org/wp-content/uploads/2022/05/MariaDBServerKnowledgeBase.pdf
[11] MariaDB System-Versioned Tables: https://mariadb.com/kb/en/system-versioned-tables/
[12] MariaDB Temporal Tables for MariaDB Enterprise Server: https://mariadb.com/docs/sql/features/temporal-tables/enterprise-server/
[13] MariaDB Date and Time Data Types: https://mariadb.com/kb/en/date-and-time-data-types/
[14] MS SQL Server Retention: https://docs.microsoft.com/en-us/sql/relational-databases/tables/manage-retention-of-historical-data-in-system-versioned-temporal-tables?view=sql-server-ver15
[15] Database Development Guide, Using Oracle Flashback Technology:https://docs.oracle.com/database/121/ADFNS/adfns_flashback.htm#ADFNS1008