Sebastian ŁACHECIŃSKI

Uniwersytet Łódzki, Instytut Logistyki i Informatyki, Katedra Informatyki Ekonomicznej

doi:10.15199/48.2020.12.18

Obsługa danych temporalnych na platformie MS SQL Server i Azure SQL Database

Streszczenie. *Celem artykułu jest wskazanie zakresu implementacji obsługi danych temporalnych w środowiskach MS SQL Server i Azure SQL Database 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 obsługi danych temporalnych przez wymienione środowiska.*

Abstract*. The aim of the article is an indication the scope of the implementation of temporal data support in the MS SQL Server and Azure SQL Database environments 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 of handling temporal data by these environments. (**Support of temporal data on the MS SQL Server and Azure SQL Database platform**).*

Słowa kluczowe: temporalne bazy danych, temporalne tabele, temporalne operatory, SQL:2011, MS SQL Server, Azure SQL Database.
Keywords: temporal databases, temporal tables, temporal operators, SQL:2011, MS SQL Server, Azure SQL Database.

Wstęp

Artykuł ten stanowi kolejny cykl rozważań na temat obsługi danych temporalnych w systemach baz danych opartych o relacyjny model danych. O ile poprzednie artykuły traktowały o modelowaniu danych temporalnych w relacyjnym modelu danych [1], a także o rozwoju języka SQL i standardu ISO/IEC 9075 ze szczególnym uwzględnieniem składni pozwalającej składować i przetwarzać dane temporalne w RDBMS [2], tak artykuł ten stanowi opis możliwości zarówno składowania jak i przetwarzania danych temporalnych, jakie oferuje platforma MS SQL Server i Azure SQL Database. Analiza taka ma odniesienie do zidentyfikowania stopnia zgodności obsługi danych temporalnych z wymaganiami standardu ISO/IEC 9075 w szczególności do wersji standardu SQL:2011. Ponadto zaprezentowane zostały możliwości i przykłady obsługi danych temporalnych w środowiskach MS SQL Server i Azure SQL Database.

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 [3,4,5]:

- 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 MS SQL Server i Azure SQL Database

Obsługa danych temporalnych do środowiska MS SQL Server została wprowadzona od wersji 2016 [2,6,7]. Serwer MS SQL Server posiada wbudowany mechanizm składowania danych temporalnych oraz ich obsługi dla każdego punktu w czasie. Nie ogranicza się tylko do przechowywania ich ostatniej, aktualnej wartości-jak w większości konwencjonalnych zastosowań baz danych OLTP.

Należy zauważyć, że rozwiązania zaimplementowane na platformie MS SQL Server nie są pełną implementacją podstawowych zapisów wprowadzonych do standardu SQL:2011. Obsługiwane są m.in. tylko tabele wersjonowane systemowo. Niezaimplementowane zostały tabele wersjonowane aplikacyjnie. Rozwiązanie to, zatem nadaje się do obsługi czasu transakcyjnego. Tak więc dobrze będzie się sprawdzać w przypadkach, gdzie bardziej istotny jest sam moment utrwalenia danych w bazie danych. Jako przykład można przytoczyć systemy bankowe np. rejestracja czasu danej transakcji, przelewu, etc. Zestawienie wybranych cech temporalnych standardu SQL:2011 oraz tych zaimplementowanych na platformie MS SQL Server i Azure SQL Database przedstawiono w tabeli 1.

Tabela 1. Spełnienie wybranych wymogów standardu SQL:2011

Wymagania	SQL:2011	SQL Server i Azure SQL Database
Definicja czasookresu	✓	✓
Typ danych PERIOD	-	-
Tabele wersjonowane aplikacyjnie	✓	-
Tabele wersjonowane systemowo	✓	✓
Tabele bitemporlane	✓	-
Aktualizacja i usuwanie rekordów z zadanego przedziału czasowego	✓	-
Temporalne ograniczenie PK	✓	-
Temporalne ograniczenie FK REFERNCES	✓	-
Implementacja predykatów czasowychTWA[1]	✓	-
Implementacja predykatów czasowych TWS[2]	✓	✓
Jawne znaczniki czasowe	✓	✓
Niejawne zanczniki czasowe	✓	✓
Typ danych data	DATE, TIME, TIMESTAMP, INTERVAL	DATE, TIME

[1] TWA-tabela wersjonowana aplikacyjnie.
[2] TWS-tabela wersjonowana systemowo.

W przypadku tego rozwiązania, opartego o tabele wersjonowane systemowo to sam system (silnik bazy danych) odpowiada za generowanie wartości daty i czasu dla zdarzenia dodania, modyfikacji lub usunięcia rekordu. Użytkownik nie ma możliwości wprowadzenia czy też modyfikacji zawartości pól składających się na początek i koniec przedziału czasowego.

Korzystanie z temporalnego mechanizmu składowania wersji danych oferowanych przez SQL Server i Azure SQL Database jest mniej skomplikowane i łatwiejsze w użyciu w odniesieniu do autorskich implementacji mechanizmu składowania i przetwarzania danych temporalnych implementowanych m.in. za pomocą wyzwalaczy. Ponadto mechanizm ten oferuje dodatkowe udogodnienia m.in.:

- gwarancję niezmienności danych historycznych,
- rozszerzoną składnię dla zapytań temporalnych,
- większą wydajność dla poleceń DML,
- niewielkie koszty utrzymania.

Mechanizm obsługi danych temporalnych jest przydatny wszędzie tam, gdzie istnieje konieczność przechowywania całej historii zmian, a więc m.in. przeprowadzanie audytu zmian wszelkich danych, możliwość wyznaczenia różnego rodzaju trendów np. wzrost lub spadek sprzedaży, na potrzeby systemów wspomagania decyzji DSS. Mechanizm ten sprawdzi się także w nieco nietypowych zastosowaniach dla tabel temporalnych m.in.: w wykrywaniu anomalii występujących okresowo lub nieregularnie; czy też w naprawie uszkodzeń danych na poziomie rekordu. Pozwala przywrócić rekordy do wybranego stanu z przeszłości, eliminując w tym przypadku użycie kopii zapasowej. Baza danych jest cały czas dostępna dla użytkowników w trybie on-line. Samo to działanie również jest wersjonowane i składowane w tabeli historycznej.

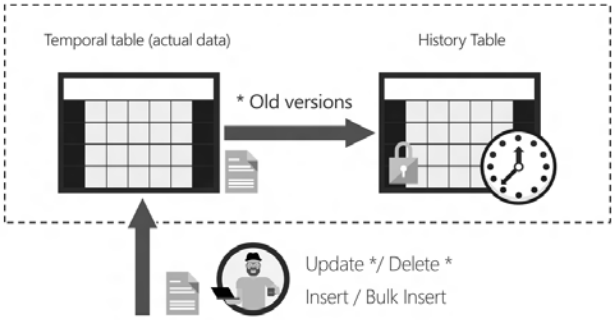
Tworzenie tabel temporalnych w MS SQL Server i Azure SQL Database

Każda tabela temporalna wersjonowana systemowo musi mieć jawnie zadeklarowane dwa pola typu systemowego datetime2. Do zawartości tych kolumn dostęp posiada tylko sam system RDBMS. Niedopuszczalne są wartości NULL dla tych kolumn. W tabeli 2 zestawiono wykaz systemowych typów danych kategorii data i czas dostępnych na platformie MS SQL Server i Azure SQL Database wraz z zakresem składowanych wartości.

Tabela 2. Systemowe typy danych kategorii data i czas dostępne na platformie MS SQL Server i Azure SQL Database

Typ danych	Zakres	Dokładność
Date	0001-01-01 ÷9999-12-31	1 dzień
Datetime	1753-01-01 ÷9999-12-31	zaokrąglone do .000, .003, .007
Datetime2	0001-01-01 ÷9999-12-31	100 ns
Datetimeoffset	0001-01-01 ÷9999-12-31	100 ns
Smalldatetime	1900-01-01 ÷2079-06-06	1 minuta
Time	00:00:00.0000000 ÷23:59:59.9999999	100 ns

Tabela taka musi posiadać zdefiniowane ograniczenie klucza głównego PK (co jest jednym z wymagań 1NF, które de facto w większości przypadków baz transakcyjnych jest spełnione). Ponadto tabela temporalna zawiera referencję do tabeli będącej jej kopią, w której to składowane są poprzednie wersje rekordów sprzed ich modyfikacji lub też sprzed momentu ich usunięcia. Jest ona tabelą historyczną, w której przechowywana jest cała historia zmian rekordów. Tabela historii jest całkowicie transparentna dla użytkowników (choć z technicznego punktu widzenia istnieje możliwość m.in. tworzenia dla niej indeksów celem optymalizacji wydajności zapytań). Użytkownicy mogą samodzielnie zdefiniować tabelę historii lub też może być ona generowana automatycznie przez system. W bieżącej tabeli temporalnej przechowywane są tylko aktualne wartości rekordów, zaś w tabeli historii każda wcześniejsza wersja rekordu wraz z określeniem wartości dla czasookresu, kiedy to dany rekord był rekordem aktualnym. Tabela historii musi posiadać identyczny schemat, jak bieżąca tabela temporalna (identyczna ilość kolumn, te same nazwy kolumn, te same typy danych, w tej samej kolejności). Na rysunku 1 przedstawiony został mechanizm składowania danych temporalnych na platformie MS SQL Server.



Rys.1. Mechanizm składowania danych temporalnych na platformie MS SQL Server, Źródło: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?view=sql-server-ver15

Tabela temporalna może być zbudowana od podstaw lub też może w tym celu być zmodyfikowana już wcześniej istniejąca tabela z danymi poprzez rozszerzenie jej schematu o kolumny składujące początek i koniec czasookresu. Tabele temporalne mogą być tworzone za pomocą narzędzi SSMS (SQL Server Management Studio), SSDT (SQL Server Data Tools), ADS (Azure Data Studio) lub składni języka T-SQL.

Sposoby tworzenia tabeli temporalnej w środowiskach MS SQL Server i Azure SQL Database [6,7]:

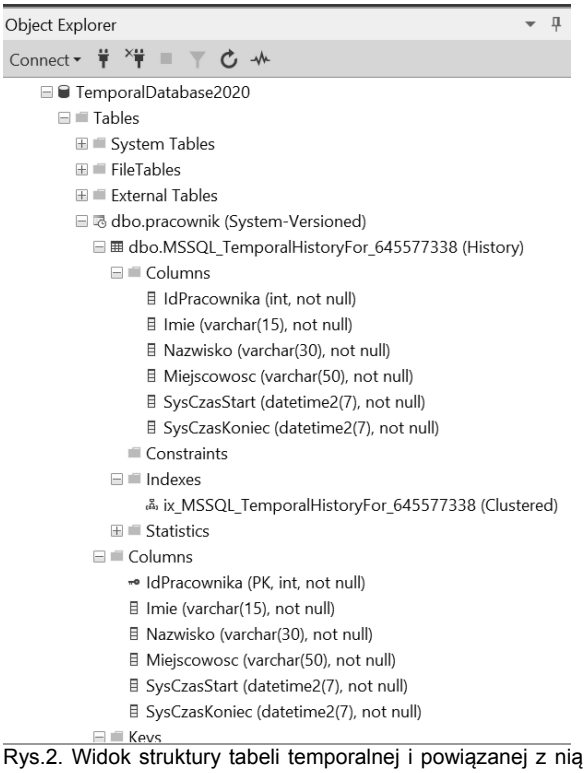
- tabela temporalna z anonimową tabelą historii-użytkownik tworzy schemat bieżącej tabeli a system sam generuje dla niej tabelę historii z automatycznie nadaną dla niej nazwą,
- tabela temporalna z domyślną tabelą historii-użytkownik sam określa nazwę schematu tabeli historii i nazwę tabeli, zaś system generuje tabelę historii w oparciu o ten schemat,
- tabela temporalna z uprzednio utworzoną tabelą historii przez użytkownika-użytkownik tworzy tabelę historii o strukturze wg własnego zapotrzebowania, do której odwołuje się podczas tworzenia tabeli temporalnej,
- adaptacja istniejącej już w bazie tabeli, do postaci tabeli temporalnej.

Tabela temporalna z anonimową tabelą historii jest wygodnym rozwiązaniem w środowiskach prototypowych i testowych. Jest to najprostszy i najszybszy sposób tworzenia tabeli temporalnej. Anonimowa tabela historii jest tworzona automatycznie w tym samym schemacie, co bieżąca tabela temporalna. Dla tabeli historii automatycznie tworzony jest indeks zgrupowany założony na kolumnach przechowujących początek i koniec czasookresu.

Poniżej przedstawiono przykład tworzenia tabeli temporalnej z anonimową tabelą historii:

```
CREATE TABLE dbo.pracownik
(
IdPracownika int CONSTRAINT PK_IdPrac PRIMARY KEY NOT
NULL,
Imie varchar(15) NOT NULL,
Nazwisko varchar(30) NOT NULL,
Miejscowosc varchar(50) NOT NULL,
SysCzasStart datetime2 GENERATED ALWAYS AS ROW START
NOT NULL,
SysCzasKoniec datetime2 GENERATED ALWAYS AS ROW END
NOT NULL,
PERIOD FOR SYSTEM_TIME (SysCzasStart,SysCzasKoniec)
)
WITH (SYSTEM_VERSIONING=ON)
```

Tabele temporalne mają przypisaną wyróżniającą ją od innych tabel ikonę. Powiązana z nią tabela historii jest wyświetlana, jako węzeł podrzędny. Na rysunku 2 przedstawiono widok utworzonej tabeli temporalnej wraz z powiązaną z nią tabelą historii. Ponadto uwidocznione zostały automatycznie utworzone indeksy.



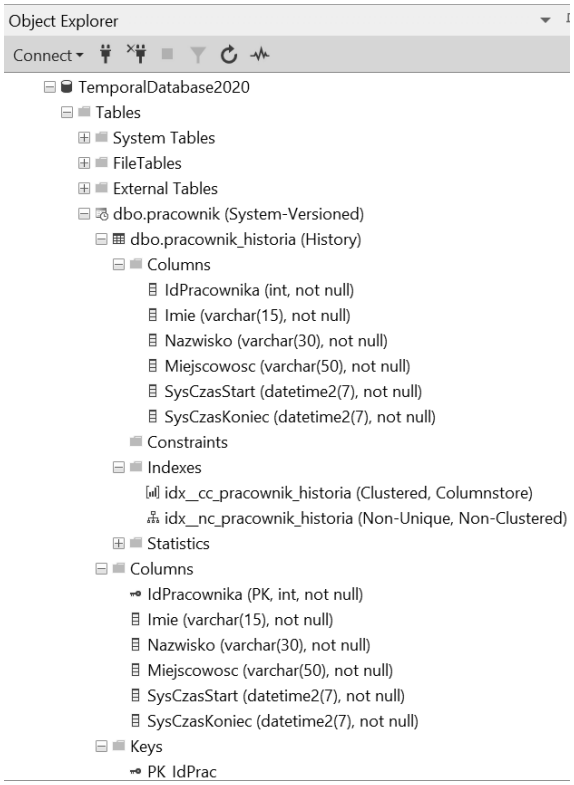
Rys.2. Widok struktury tabeli temporalnej i powiązanej z nią tabeli historii zmian.

Tabela temporalna z domyślną tabelą historii pozwala określić jej nazwę, a system sam tworzy tabelę historii z domyślną konfiguracją. Przy czym obowiązkowe jest podanie nazwy schematu (który musi istnieć) dla parametru HISTORY_TABLE. Jeśli tabela historyczna już istnieje, jest ona weryfikowana pod kątem spójności schematu oraz spójności czasowej danych. Poniżej przedstawiono przykład tworzenia tabeli temporalnej wersjonowanej systemowo z jawnie zdefiniowaną nazwą tabeli historii:

```
CREATE TABLE dbo.pracownik
(
IdPracownika int CONSTRAINT PK_IdPrac PRIMARY KEY NOT
NULL,
Imie varchar(15) NOT NULL,
Nazwisko varchar(30) NOT NULL,
Miejscowosc varchar(50) NOT NULL,
SysCzasStart datetime2 GENERATED ALWAYS AS ROW START
NOT NULL,
SysCzasKoniec datetime2 GENERATED ALWAYS AS ROW END
NOT NULL,
PERIOD FOR SYSTEM_TIME (SysCzasStart,SysCzasKoniec)
)
WITH (SYSTEM_VERSIONING=ON
(HISTORY_TABLE=dbo.pracownik_historia))
```

Na rysunku 3 przedstawiono strukturę tabeli temporalnej z domyślną tabelą historii zmian.

Tabela temporalna z tabelą historii zdefiniowaną przez użytkownika pozwala określić opcje przechowywania oraz daje możliwość utworzenia na niej dodatkowych indeksów. Tabela historii nie może mieć zdefiniowanego klucza podstawowego, kluczy obcych, unikalnych indeksów, ograniczeń czy też wyzwalaczy.



Rys 3. Widok struktury tabeli temporalnej i powiązanej z nią domyślnie utworzoną tabelą historii zmian.

W pierwszej kolejności tworzona jest tabela historii zmian:

```
CREATE TABLE dbo.pracownik_historia
(
IdPracownika int NOT NULL,
Imie varchar(15) NOT NULL,
Nazwisko varchar(30) NOT NULL,
Miejscowosc varchar(50) NOT NULL,
SysCzasStart datetime2  NOT NULL,
SysCzasKoniec datetime2 NOT NULL,
)
```

Następnie tworzony jest na tabeli historii klastrowy indeks kolumnowy:

```
CREATE CLUSTERED COLUMNSTORE INDEX
idx__cc_pracownik_historia
ON dbo.pracownik_historia
```

oraz niezgrupowany indeks dla pól określających początek i koniec czasookresu:

```
CREATE NONCLUSTERED INDEX idx__nc_pracownik_historia
ON dbo.pracownik_historia(SysCzasStart,SysCzasKoniec)
```

Ostatnim etapem jest utworzenie tabeli temporalnej w powiązaniu z uprzednio utworzoną tabelą historii:

```
CREATE TABLE dbo.pracownik
(
IdPracownika int CONSTRAINT PK_IdPrac PRIMARY KEY NOT
NULL,
Imie varchar(15) NOT NULL,
Nazwisko varchar(30) NOT NULL,
Miejscowosc varchar(50) NOT NULL,
SysCzasStart datetime2 GENERATED ALWAYS AS ROW START
NOT NULL,
SysCzasKoniec datetime2 GENERATED ALWAYS AS ROW END
NOT NULL,
PERIOD FOR SYSTEM_TIME (SysCzasStart,SysCzasKoniec)
)
WITH (SYSTEM_VERSIONING=ON
(HISTORY_TABLE=dbo.pracownik_historia))
```

Tabela temporalna tworzona na podstawie wcześniej istniejącej tabeli z danymi wymaga zdefiniowania dwóch dodatkowych pól na składowanie początku i końca czasookresu. Dodatkowo można określić nazwę dla pustej tabeli historii, za utworzenie, której odpowiedzialny będzie SQL Server lub SQL Database. Poniżej przedstawiono przykład adaptacji istniejącej tabeli do rozwiązania tabeli temporalnej:

```
ALTER TABLE dbo.pracownik
ADD
SysCzasStart datetime2(0) GENERATED ALWAYS AS ROW
START HIDDEN
CONSTRAINT df_SysCzasStart DEFAULT SYSUTCDATETIME(),
SysCzasKoniec datetime2(0) GENERATED ALWAYS AS ROW END
HIDDEN
CONSTRAINT df_SysCzasKoniec DEFAULT '9999.12.31
23:59:59:99',
PERIOD FOR SYSTEM_TIME(SysCzasStart,SysCzasKoniec)
```

Po wprowadzonej modyfikacji, tabela zostaje zamieniona w tabelę temporalną wersjonowaną systemowo za pomocą poniższego kodu:

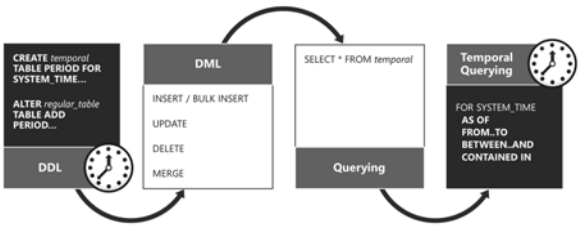
```
ALTER TABLE dbo.pracownik
SET (SYSTEM_VERSIONING=ON
(HISTORY_TABLE=dbo.pracownik_historia))
```

Możliwe także jest utworzenie na tabeli historii indeksu podobnie jak dla przypadku tworzenia tabeli historii bezpośrednio przez użytkownika.

```
CREATE CLUSTERED COLUMNSTORE INDEX
idx__cc_pracownik_historia
ON dbo.pracownik_historia
```

W przypadku tabel temporalnych tworzonych na platformie SQL Server i SQL Database istnieje możliwość określenia czasu przechowywania danych historycznych za pomocą klauzuli HISTORY_RETENTION_PERIOD. Pozwala ona zarządzać cyklem życia danych historycznych i zaoszczędzić miejsce na składowane dane (przy założeniu, że nie narusza to wymagań biznesowych, co do długości czasu składowania tych danych). Oczywiście w prawdziwych modelach temporalnych jest to niedopuszczalne, gdyż wiąże się to z utratą danych historycznych. W wielu zastosowaniach podejście takie nie pozwoli np. na śledzenie trendów sprzedaży, historii zatrudnienia itp. Należy zaznaczyć, że jest to jedna z kilku metod dostępnych na serwerze do zarządzania czasem przechowywania danych historycznych.

Na rysunku 4 przedstawiony został sposób pracy z rozwiązaniem opartym o tabele czasowe w środowisku MS SQL Server i SQL Database.



Rys.4. Sposób pracy z tabelami czasowymi w MS SQL Server, Źródło: https://docs.microsoft.com/en-us/sql/relational-databases/tables/getting-started-with-system-versioned-temporal-tables?view=sql-server-ver15

Manipulacja danymi temporalnymi

Gdy dodawany jest nowy rekord do tabeli temporalnej SQL Server automatycznie ustawia dla niego początek okresu na aktualny czas w strefie UTC. Koniec okresu ustawiany jest na maksymalną wartość dla typu danych datetime2 przypisaną do tego pola (9999-12-31 23:59:59.99), co jednocześnie oznacza, iż jest to aktualna wersja tego rekordu do momentu wystąpienia pierwszej modyfikacji (lub kolejnej) lub też jego usunięcia.

Instrukcja INSERT skutkuje dodaniem nowego rekordu do bieżącej tabeli temporalnej. W powiązanej tabeli historii zmian nie pojawia się żaden rekord. Jeśli pola składujące czasookres są ukryte, wówczas w klauzuli INSERT wystarczy podać wartości dla widocznych kolumn tabeli temporalnej. W przeciwnym razie można na liście pól pominąć pola czasookresu lub w klauzuli VALUES zaproponować dla nich wartości domyślne generowane przez system przy użyciu klauzuli DEFAULT. Poniżej przedstawiono przykład dodania rekordu do tabeli temporalnej.

```
--dodanie rekordów do tabeli temporalnej
INSERT dbo.pracownik
VALUES (1,'Jan','Kowalski','Łódź'),(2,'Katarzyna','Janowska','Kraków')
go
SELECT * FROM dbo.pracownik
go
SELECT * FROM dbo.pracownik_historia
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc
1	1	Jan	Kowalski	Łódź
2	2	Katarzyna	Janowska	Kraków

IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec

Rys.5. Dodanie kilku rekordów do bieżącej tabeli temporalnej.

Instrukcja UPDATE skutkuje dodaniem nowego rekordu do tabeli historii zmian, jego ostatniej aktualnej wersji sprzed modyfikacji. W tabeli bieżącej zaś pozostaje jego zmieniona, aktualna wersja. Znacznik końca okresu dla rekordu historycznego jest równy wartości początku okresu rekordu bieżącego. Instrukcja ta pozwala także przywrócić ostatnią poprawną wersję rekordu do tabeli bieżącej na podstawie zawartości tabeli historycznej (zamiast odtwarzania kopii zapasowej-jest to zdecydowanie szybsze rozwiązanie, należy zaznaczyć, że nie jest to mechanizm mający zastąpić tradycyjne kopie zapasowe, ale w specyficznych zastosowaniach jest znaczenie lżejszy i bardziej opłacalny). Poniżej przedstawiono przykład aktualizacji rekordu w tabeli temporalnej.

```
--aktualizacja rekordów tabeli temporalnej
UPDATE dbo.pracownik
SET Miejscowosc='Wrocław' WHERE IdPracownika=2
go
SELECT * FROM dbo.pracownik
go
SELECT * FROM dbo.pracownik_historia
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc
1	1	Jan	Kowalski	Łódź
2	2	Katarzyna	Janowska	Wrocław

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49

Rys.6. Aktualizacja rekordów w bieżącej tabeli temporalnej

Aby zachowana została cała historia zmian, aktualizacji poddawane mogą być tylko rekordy składowane w tabeli bieżącej (poza wyjątkiem pól definiujących czasokres PERIOD). Niemożliwa jest także modyfikacja zawartości tabeli historycznej. Ustawienie parametru SYSTEM_VERSIONING=OFF pozwoli na wprowadzanie modyfikacji również w tabeli historycznej kosztem utracenia historii zmian.

Instrukcja DELETE skutkuje usunięciem rekordu z bieżącej tabeli temporalnej. W tabeli historii rekord ten nadal pozostaje. Pojawia się natomiast modyfikacja końca okresu czasu na wartość, do kiedy dany rekord istniał w bazie danych. Nie można usunąć rekordów z tabeli historycznej chyba, że zostanie wyłączona opcja wersjonowania systemowego. Wówczas jednak utracona zostanie historia zmian lub też będzie ona niekompletna. Poniżej przedstawiono przykład usunięcia rekordu z tabeli temporalnej wraz z wizualizacją zawartości bieżącej tabeli temporalnej oraz tabeli historii sprzed i po usunięciu rekordu:

```
--usuwanie rekordów temporalnych
SELECT * FROM dbo.pracownik
go
SELECT * FROM dbo.pracownik_historia
go
DELETE FROM dbo.pracownik
WHERE IdPracownika=2
go
SELECT * FROM dbo.pracownik
go
SELECT * FROM dbo.pracownik_historia
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc
1	1	Jan	Kowalski	Łódź
2	2	Katarzyna	Janowska	Wrocław

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49

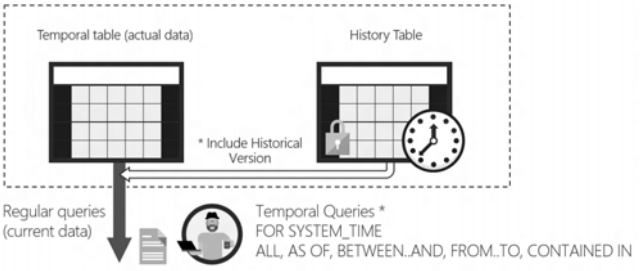
	IdPracownika	Imie	Nazwisko	Miejscowosc
1	1	Jan	Kowalski	Łódź

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49
2	2	Katarzyna	Janowska	Wrocław	2020-08-19 16:18:49	2020-08-19 16:52:30

Rys.7. Usuwanie rekordów z bieżącej tabeli temporalnej

Obsługa zapytań

Za pomocą odpowiedniej składni można manipulować w środowisku SQL Server i SQL Database danymi temporalnymi. Można np. pobierać rekordy, których wartość była aktualna w określonym przedziale czasowym, czy też prześledzić historię zatrudnienia danego pracownika. Zapytanie takie jest wykonywane w sposób automatyczny na obu tabelach: bieżącej tabeli temporalnej i tabeli historii. Użytkownik nie musi w sposób jawny odwoływać się do tabeli historii w konstrukcji zapytania. Na rysunku 8 przedstawiono sposób pracy z zapytaniami temporalnymi na platformie MS SQL Server i SQL Database.



Rys.8. Przetwarzanie zapytań temporalnych środowisku w MS SQL Server, Źródło: https://docs.microsoft.com/en-us/sql/relational-databases/tables/getting-started-with-system-versioned-temporal-tables?view=sql-server-ver15

W środowisku MS SQL Server i SQL Database wprowadzone zostały klauzule do przetwarzania danych temporalnych składowanych w tabelach temporalnych wersjonowanych systemowo. Są one zgodne z zapisem standardu SQL:2011 (który wprowadza trzy klauzule). Na platformie SQL Server dostępne są dwie dodatkowe konstrukcje. Zestawienie tych konstrukcji zawarte zostało w tabeli 3.

Tabela 3. Rozszerzenia składniowe do przetwarzania danych składowanych w tabelach temporalnych wersjonowanych systemowo.

SQL:2011	SQL Server i SQL Database
AS OF	AS OF
FROM TO	FROM TO
BETWEEN AND	BETWEEN AND
-	CONTAINED IN
-	ALL

Rozszerzenie temporalne AS OF pozwala uzyskać stan rekordów na dany moment w czasie, którego wartość jest podana, jako argument predykatu. Poniżej przedstawiono przykład zapytania z użyciem klauzuli AS OF.

```
--odpytywanie tabeli temporalnej
--klauzula AS OF
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME AS OF '2020-08-19 16:10'
go
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME AS OF '2020-08-19 16:20'
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	1	Jan	Kowalski	Łódź	2020-08-19 16:09:44	9999-12-31 23:59:59
2	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	1	Jan	Kowalski	Łódź	2020-08-19 16:09:44	9999-12-31 23:59:59
2	2	Katarzyna	Janowska	Wrocław	2020-08-19 16:18:49	2020-08-19 16:52:30

Rys.9. Zapytanie na tabeli temporalnej z użyciem predykatu AS OF

Predykaty temporalne FROM TO i BETWEEN AND zwracają zbiór rekordów temporalnych z zadanego przedziału czasowego. Jednakże predykat FROM TO w wynikowym zbiorze rekordów nie uwzględnia rekordów, które są aktualne dla chwili czasowej równej górnej granicy czasookresu. Rysunki 10 i 11 stanowią ilustrację użycia omówionych operatorów temporalnych. Na rysunku 11 uwidoczniony jest dodatkowy rekord dla chwili czasowej odpowiadającej górnej granicy wartości przedziału czasokresu, kiedy to nastąpiła zmiana miejscowości zamieszkania pracownika.

```
--klauzula FROM TO
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME FROM '2020-08-19 16:09' TO '2020-08-19 16:18:49'
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	1	Jan	Kowalski	Łódź	2020-08-19 16:09:44	9999-12-31 23:59:59
2	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49

Rys.10. Zapytanie na tabeli temporalnej z użyciem predykatu FROM TO

```
--klauzula BETWEEN AND
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME BETWEEN '2020-08-19 16:09' AND '2020-08-19 16:18:49'
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	1	Jan	Kowalski	Łódź	2020-08-19 16:09:44	9999-12-31 23:59:59
2	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49
3	2	Katarzyna	Janowska	Wrocław	2020-08-19 16:18:49	2020-08-19 16:52:30

Rys.11. Zapytanie na tabeli temporalnej z użyciem predykatu BETWEEN AND

Rozszerzenie temporalne CONTAINED IN zwraca zbiór rekordów otwartych lub zamkniętych w przedziale czasu dla wartości podanych, jako argument tej klauzuli (również z uwzględnieniem rekordów dla obu granicznych wartości czasookresu). Na rysunku 12 przedstawiono przykład zapytania temporalnego z użyciem predykatu CONTAINED IN.

```
--klauzula CONTAINED IN
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME CONTAINED IN ('2020-08-19 16:09','2020-08-19 16:18:49')
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49

Rys.12. Zapytanie na tabeli temporalnej z użyciem predykatu CONTAINED IN

Rozszerzenie temporalne ALL nie wymaga podania żadnego argumentu, zwraca sumę zbiorów rekordów z tabeli bieżącej i tabeli historii (w postaci unii). Na rysunku 13 przedstawiono przykład zapytania temporalnego z użyciem predykatu ALL:

```
--klauzula ALL
SELECT IdPracownika,Imie,Nazwisko,Miejscowosc,SysCzasStart,SysCzasKoniec
FROM dbo.pracownik
FOR SYSTEM_TIME ALL
go
```

100 %

Results | Messages

	IdPracownika	Imie	Nazwisko	Miejscowosc	SysCzasStart	SysCzasKoniec
1	1	Jan	Kowalski	Łódź	2020-08-19 16:09:44	9999-12-31 23:59:59
2	2	Katarzyna	Janowska	Kraków	2020-08-19 16:09:44	2020-08-19 16:18:49
3	2	Katarzyna	Janowska	Wrocław	2020-08-19 16:18:49	2020-08-19 16:52:30

Rys.13. Zapytanie na tabeli temporalnej z użyciem predykatu ALL

Podsumowanie

Platforma SQL Server oraz Azure SQL Database implementują tylko w wąskim zakresie postulaty standardu SQL:2011 w zakresie temporalności danych. Nie zostały zaimplementowane tabele wersjonowane aplikacyjnie, a więc rozwiązanie szczególnie przydatne dla rejestracji w bazie danych czasu rzeczywistego. Konsekwencją realizacji tylko części temporalnego rozszerzenia standardu SQL:2011 jest także brak implementacji specjalnych predykatów temporalnych używanych w trakcie odpytywania tabeli temporalnej z rejestracją czasu rzeczywistego. Brak jest temporalnych ograniczeń dla klucza głównego, klucza obcego, czy też ograniczeń referencyjnych w odniesieniu do tabeli wersjonowanej aplikacyjnie. Niezaimplementowane zostały także specjalne rozszerzenia temporalne dla akcji modyfikacji i usuwania rekordów z tabeli temporalnej wersjonowanej aplikacyjnie. Brak również jest obsługi tabel bitemporalnych z możliwością rejestracji czasu rzeczywistego oraz transakcyjnego. Tak więc, aspekty te nadal wymagają własnoręcznej implementacji np. w postaci specjalnych typów danych, operatorów temporalnych, m.in. za pomocą środowiska CLR lub mechanizmu CDC (Change Data Capture), czy też składni języka T-SQL. Niestety, od momentu wprowadzenia elementów obsługi danych temporalnych do serwera MS SQL Server 2016 i SQL Database, odpowiadających tylko fragmentarycznie zapisom standardu SQL:2011, w kolejnych jego edycjach tj. 2017, 2019 wprowadzono wiele nowych udogodnień oraz usprawnień, ale żadne z nich nie dotyczyło rozszerzenia obsługi danych temporalnych.

Należy zauważyć, że w stosunku do zapisów standardu SQL:2011 platforma SQL Server i SQL Database wprowadziła dwa dodatkowe operatory do odpytywania tabel temporalnych wersjonowanych systemowo: ALL i CONTAINED IN. Ponadto, w środowiskach dostępna jest opcja zarządzania czasem składowania danych historycznych, który może być realizowany na różne sposoby za pomocą różnych mechanizmów tj:. Stretch Database, partycjonowanie tabeli, tworzenia własnych dedykowanych skryptów usuwających dane, czy też skorzystanie z mechanizmu zasad przechowywania danych [8]. Od wersji 2019 w SQL Server możliwe także jest tworzenie tabeli temporalnej optymalizowanej pod kątem pamięci. Usprawnienie to nie dotyczy platformy Azure SQL Database [9]. Ponadto, środowisko SQL Server pozwala także uzyskać informacje nt. tabel temporalnych. Informacje te dostępne są przy użyciu systemowych widoków: sys.tables, sys.columns, sys.periods oraz systemowych funkcji: OBJECTPROPERTY, OBJECTPROPERTYEX, COLUMNPROPERTY [10].

Udostępnione rozwiązania temporalne na platformie SQL Server i Azure SQL Database mają także swoje ograniczenia m.in. nie można używać tabel temporalnych na serwerach połączonych, tabela temporalna i tabela historii muszą być utworzone w tej samej bazie danych, nie mogą być ponadto tabelami FILETABLE, a żadne pole tych tabel nie może być typu FILESTREAM (co eliminuje możliwość składowania danych poza bazą danych, w przestrzeni systemu operacyjnego). W przypadku zapytań temporalnych nie są obsługiwane widoki indeksowane, niedostępne jest także polecenie TRUNCATE TABLE dla tabel wersjonowanych systemowo. W wersji 2016 SQL Server nie są także obsługiwane akcje referencyjne w postaci kaskadowej aktualizacji rekordów i kaskadowego usuwania rekordów dla bieżących tabel temporalnych. Ograniczenie to nie dotyczy nowszych wersji SQL Server oraz SQL Database. Nie można także używać wyzwalaczy typu INSTEAD OF dla bieżącej tabeli temporalnej i tabeli historii zmian. Użycie wyzwalaczy typu AFTER dozwolonej jest tylko dla bieżącej tabeli temporalnej. Istotne ograniczenia dotyczą też użycia mechanizmu replikacji (np. replikacji scalającej w ogóle nie można zastosować dla tabeli temporalnej) [11].

Obsługa danych temporalnych na platformie SQL Server wypada mniej korzystnie w odniesieniu do konkurencyjnych platform bazodanowych. Najbardziej zaawansowaną obsługę danych temporalnych zgodnie ze standardem SQL:2011 zaimplementowano w rozwiązaniach IBM DB2 [12,13,18-20]. Nieco bardziej rozwinięta obsługa danych temporalnych jest także dostępna na platformie ORACLE [14,15,18,19]. Zaawansowana obsługa danych temporalnych jest także zaimplementowana w rozwiązaniach Teradata, niemniej jednak opiera się ona innych fundamentach niż standard SQL:2011 [16-19].

Mimo pewnych ograniczeń rozwiązań temporalnych zaimplementowanych na platformie SQL Server i SQL Database warto rozważyć ich użycie w projektach, które wpisują się w obsługę czasu transakcyjnego. Skorzystanie z temporalnego mechanizmu składowania wersji danych jest znacznie łatwiejsze w użyciu, w odniesieniu do autorskich implementacji mechanizmu składowania i przetwarzania danych temporalnych. Dodatkowo oferuje on niezmienność danych historycznych, rozbudowaną składnię dla zapytań temporalnych jak i większą wydajność dla instrukcji DML.

Kolejnym etapem pracy będzie zbadanie stopnia realizacji założeń temporalnych rozszerzeń standardu SQL:2011 w wybranych wiodących produktach na rynku oraz prezentacja zaimplementowanych w nich 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] Date C.J., Darwen H., Lorentzos N., Time and relational theory Temporal Databases in the Relational Model and SQL, 2014, Morgan Kaufmann
[4] Kulkarni K., Jan-Eike Michels, 2012, *Temporal features in SQL:2011,* 34-43 https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf
[5] Temporal extension SQL: https://link.springer.com/content/pdf/10.1007%2F978-1-4899-7993-3_80729-1.pdf
[6] MS SQL Server: https://docs.microsoft.com/en-us/sql/relational-databases/tables/getting-started-with-system-versioned-temporal-tables?view=sql-server-ver15\
[7] Azure SQL Database: https://docs.microsoft.com/pl-pl/azure/azure-sql/temporal-tables
[8] 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
[9] MS SQL Server MOT: https://docs.microsoft.com/en-us/sql/relational-databases/tables/system-versioned-temporal-tables-with-memory-optimized-tables?view=sql-server-ver15
[10] MS SQL Server Metadata: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-table-metadata-views-and-functions?view=sql-server-ver15
[11] MS SQL Server Limitations: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-table-considerations-and-limitations?view=sql-server-ver15
[12] Saracco M., Nicola M., Gandhi L., 2012, A matter of time: Temporal data management in DB2 10 https://www.yumpu.com/en/document/read/13882386/a-matterof-time-temporal-data-management-in-db2-10-ibm
[13] IBM: https://www.ibm.com/developerworks/data/library/techarticle/dm-1410temporal-tables-db2zos/index.html
[14] Salvisberg P., Multi-temporal Features in Oracle 12c, https://www.salvis.com/blog/2014/01/04/multi-temporal-database-features-in-oracle-12c/
[15] Oracle: https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/ilm/temporal/temporal.html
[16] Snodgrass R.T., 2010, A Case Study of Temporal Data, Teradata Corporation https://cs.ulb.ac.be/public/_media/teaching/infoh415/teradata_temporal_case_study.pdf
[17] Teradata https://docs.teradata.com/reader/vi_E5DAzLxKE_Xhr8yNIWw/BuO5td4SJIxUXyFilG6Geg
[18] Petković D., Support of Temporal Data in Database Systems, International Journal of Computer Applications (0975 –8887), Volume 152 –No.4, October 2016, 26-33
[19] Petković D., Temporal Data in Relational Database Systems: A Comparison, Conference: WorldCIST (1) 2016, At Recife, Volume: 1
[20] Petković, D., Temporal Data in Enterprise Database Systems, The 7th International Conference on Information Technology, 2015,pp 276-282