Sebastian ŁACHECIŃSKI

Uniwersytet Łódzki, Instytut Logistyki i Informatyki, Katedra Informatyki Ekonomicznej

doi:10.15199/48.2020.10.35

Składowanie i przetwarzanie danych temporalnych w świetle wymagań standardu języka SQL ISO/IEC 9075

Streszczenie. *Przechowywanie i przetwarzanie danych temporalnych jest szczególnie istotne tam, gdzie zachodzi konieczność składowania i przetwarzania oprócz danych statycznych również danych, które zmieniają się wraz z upływem czasu. Dane temporalne są danymi specyficznymi, tym samym wymagają także nieco odmiennego podejścia do ich obsługi. W artykule zaprezentowano pokrótce ewolucje standardu języka SQL ze szczególnym naciskiem na jego rozszerzenie związane z temporalnością danych. Przedstawione zostały sposoby składowania danych w tabelach temporalnych z uwzględnieniem czasu rzeczywistego i transakcyjnego oraz w tabelach bitemporalnych. Zaprezentowane zostały sposoby przetwarzania danych z użyciem operatorów temporalnych wprowadzonych do standardu SQL:2011.*

Abstract*. The storage and processing of temporal data is particularly important where it is necessary to store and process not only static data, but also those that change over time. Temporal data is specific data, therefore it requires a slightly different approach to their handling. The article briefly presents the evolution of the SQL language standard with particular emphasis on its extension related to the temporality of data. The methods of data storage in temporal tables, taking into account valid and transaction time, and in bitemporal tables were presented. The methods of data processing with the use of temporal operators introduced into the SQL: 2011 standard were presented. (**Storage and processing of temporal data in the light of SQL language standard ISO/IEC 9075**).*

Słowa kluczowe: temporalne bazy danych, temporalne tabele, temporalne operatory, standard SQL ISO/IEC 9075
Keywords: temporal databases, temporal tables, temporal operators, standard SQL ISO/IEC 9075

Wstęp

Artykuł ten stanowi kontynuację rozważań na temat obsługi danych temporalnych w środowiskach relacyjnych baz danych. O ile w poprzednim artykule [1] skoncentrowano się na możliwościach zamodelowania danych temporalnych w odwzorowaniu do relacyjnego modelu danych, a także przedstawiono porównanie sposobów modelowania wybranych cech encji logicznych zmieniających się wraz z upływem czasu, tak ten artykuł poświęcono zagadnieniom i możliwościom obsługi danych temporalnych składowanych w relacyjnych bazach danych.

Przechowywanie i przetwarzanie danych temporalnych jest szczególnie istotne tam, gdzie zachodzi konieczność składowania i przetwarzania oprócz danych statycznych również danych, które zmieniają się wraz z upływem czasu. Należy zauważyć, że dane temporalne są danymi specyficznymi, tym samym wymagają także nieco odmiennego podejścia do ich obsługi, m.in. poprzez zastosowanie dodatkowych, specjalnych operatorów i klauzul ułatwiających pracę z danymi temporalnymi [2]. Na przestrzeni ostatnich trzech dekad standard języka SQL znacznie ewoluował. W kolejnych latach powstawały jego kolejne, nowsze wersje uwzględniające w swojej specyfikacji nowe rozwiązania i konstrukcje, również te z obszaru składowania, przetwarzania i obsługi danych temporalnych.

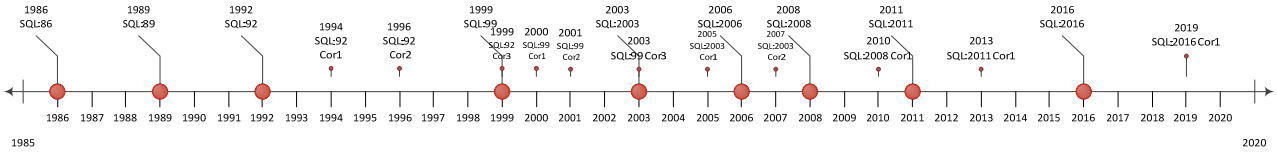
W kolejnych rozdziałach przedstawione zostaną ogólne założenia modelu temporalnego oraz główne operatory i klauzule temporalne, rozwój języka SQL, skutkujący pojawianiem się kolejnych standardów języka SQL ze szczególnym uwzględnieniem rozszerzeń o cechy temporalne.

Rozwój standardu języka SQL a obsługa zmian temporalnych

Pierwowzór strukturalnego języka zapytań SQL (Structured Query Language) został opracowany ponad cztery dekady temu, na początku lat 70-tych ubiegłego wieku w firmie IBM pod nazwą SEQueL (Structured English Query Language). Pierwsze komercyjne wdrożenie miało miejsce w bazie danych ORACLE, w 1979 roku. Jednakże ze względu na fakt, iż nazwa ta została zastrzeżona przez brytyjską firmę przemysłu lotniczego została ona zamieniona na nową dobrze znaną do dnia dzisiejszego nazwę SQL (Structured Query Language). Wraz z upływem czasu język SQL implementowany był przez inne firmy w ich własnych produktach, dzięki czemu zyskał znacznie na popularności. Jednocześnie język SQL zaczął zatracać spójność, ze względu na fakt, że każda z firm implementowała go na swój własny sposób, wprowadzając do niego własne modyfikacje. W celu zagwarantowania jednolitej postaci a tym samym utrzymania spójności języka SQL koniecznością było ustandaryzowanie tego języka. Po raz pierwszy miało to miejsce w 1986 roku, w którym to opracowana została pierwsza wersja standardu języka SQL pod auspicjami ANSI (American National Standards Institute) oraz ISO (International Organization for Standarization). Wersje standardu SQL: 86, 89 i 92 opracowane zostały pod auspicjami ANSI [3]. Natomiast kolejne wersje standardu opracowane były pod auspicjami ISO/IEC (International Electrotechnical Commision).

Powstało wiele dialektów języka SQL, pośród których do najbardziej popularnych zaliczyć można: TSQL (Transact SQL) opracowany przez firmę Sybase a następnie rozwijany także przez firmę Microsoft na platformie SQL Server, PL/SQL (Procedural Language/SQL), jako proceduralne rozszerzenie języka SQL dostępne na platformie ORACLE, SQL PL (SQL Procedural Language) na platformie IBM DB2, SQL/PSM (SQL/Presistent Stored Modules) na platformie MySQL, PL/pgSQL na platformie PostgreSQL, PSQL (Procedural SQL) na platformie Interbase i Firebird. Pomimo istniejących różnic pomiędzy poszczególnymi dialektami języka SQL główne jego założenia zachowały spójność. Było to możliwe dzięki opracowaniu standardu języka SQL i jego implementacji w poszczególnych środowiskach RDBMS. Na rysunku 1 przedstawiono rozwój standardu języka SQL na przestrzeni lat.

Od momentu pojawienia się języka SQL, kilkukrotnie podejmowano próby wprowadzenia do jego standardu rozszerzeń temporalnych. Pierwsza próba oparta była o język TSQL2 wprowadzona do standardu SQL92, jako opcja. Jednakże ostatecznie została z niego wycofana w 2001 roku. Kolejna i jednocześnie udana próba wprowadzenia do standardu języka SQL rozszerzenia temporalnego miała miejsce w 2011 roku, aczkolwiek oparta była na nieco odmiennych założeniach w stosunku do wcześniejszego rozwiązania. W kolejnych podpunktach zaprezentowano krótki przegląd poszczególnych wersji standardu języka SQL z uwzględnieniem najbardziej istotnych aspektów do nich wprowadzonych.



Rys.1. Chronologiczny rozwój standardu języka SQL, Źródło: opracowanie własne

Standard SQL 86

Jest to pierwszy, oficjalny standard, jaki został opracowany po upływie zaledwie nieco ponad dekady od momentu pojawienia się pierwowzoru języka SQL. Stanowił bardzo ogólną i luźną wersję niewprowadzającą w zasadzie żadnych restrykcji z punktu widzenia jego implementacji przez dostawców serwerów baz danych SQL. Dostawcom rozwiązań bazodanowych pozostawiał wiele swobody.

Standard SQL-89

W standardzie tym, wprowadzono w stosunku do jego poprzedniej wersji m.in. ograniczenia integralności. Podobnie, jak jego poprzednik nie była to wersja rygorystyczna dla producentów serwerów SQL.

Standard SQL-92

Standard ten stanowił pierwszą ważną i istotną wersję, która była znacznie bardziej uściślona, a zarazem bardziej restrykcyjna w porównaniu do dwóch poprzednich (mając na uwadze dużą swobodę implementacji, jaką pozostawiały one dostawcom rozwiązań bazodanowych w swoich produktach). Uwzględniono w nim m.in. typy danych tj. DATE, TIME, TIMESTAMP, INTERVAL, BIT, VARCHAR, NATIONAL CHAR. Operatory UNION JOIN, NATURAL JOIN, tworzenie tabel tymczasowych, czy przewijalne kursory.

Pewne zmiany zostały wprowadzone do tego standardu w latach 1994 (Call Level Interface), 1996 (procedury składowane) i 1999.

Standard SQL-99

Do standardu tego wprowadzono m.in.: obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyzwalacze, obsługę kodu SQL osadzonego w języku Java. Standard ten doczekał się także pewnych zmian w latach 2000, 2001 i 2003. Ponadto wprowadzona została instrukcja CREATE LIKE pozwalające kopiować tabelę –jej strukturę (działanie ograniczone do nazw i typów kolumn).

Standard SQL:2003

W standardzie tym dodano obsługę XML. Wprowadzono nową instrukcję scalającą MERGE. Ponadto wprowadzono obsługę generatorów wartości w postaci niezależnych obiektów-sekwencji poprzez wprowadzenie instrukcji CREATE SEQUENCE, a także pól z automatycznym generowaniem wartości dla kolumn tożsamości. Wprowadzono także możliwość tworzenia w definicji tabeli tzw. kolumn generowanych na podstawie uprzednio zdefiniowanego wyrażenia skalarnego. Usunięty ze specyfikacji został typ danych BIT i BIT VARYING. Jednocześnie wprowadzone zostały do standardu nowe typy danych: BIGINT, MULTISET (nowa kolekcja, nieuporządkowany zbiór elementów tego samego typu, z możliwością zawierania duplikatów) i XML. Ponadto wprowadzono obsługę funkcji tabelowych, rozszerzono klauzulę CREATE TABLE o możliwość definiowania kolumn tożsamości oraz kolumn generowanych na podstawie wyrażenia. Dla obu przypadków wartości w tych kolumnach generowane są automatycznie. Rozszerzono możliwości instrukcji CREATE TABLE LIKE o przenoszenie dodatkowych informacji tj. wartości domyślne, kolumna tożsamości, czy wyrażenia będące podstawą utworzenia nowej kolumny tabeli. Możliwość utworzenia nowej tabeli na podstawie częściowej definicji innej tabeli lub tabel za pomocą instrukcji CREATE TABLE AS na podstawie odpowiednio zbudowanej treści zapytania SELECT [4].

Pewne korekty wprowadzono w 2005 roku.

Standard SQL:2006

Standard ten rozszerza możliwości obsługi danych XML. M.in. określa sposoby ładowania i składowania danych w bazie danych, manipulowania danymi XML po stronie bazy danych.

Standard SQL:2008

W standardzie tym dodano obsługę wyzwalaczy typu „zamiast” INSTEAD OF, jak również wprowadzono klauzulę FETCH i TRUNCATE TABLE. Rozszerzono klauzulę MERGE o możliwość obsługi wielu klauzul MATCHED i NOT MATCHED, co daje znaczenie szersze możliwości w tworzeniu złożonych instrukcji MERGE z wieloma różnymi warunkami m.in. do obsługi konfliktów podczas aktualizacji. Ponadto znacznie wsparto typy binarne BINARY i VARBINARY [5]. Standard ten został rozszerzony przez wprowadzenie dodatkowych korekt w 2010 roku.

Standard SQL:2011

Standard SQL określa zarówno obowiązkowe jak i opcjonalne cechy języka SQL. Wiele cech obowiązkowych nie uległo znaczącym zmianom od wersji 92 czy też 99. Najwięcej nowości dotyczyło opcjonalnych elementów składniowych SQL. M.in. wprowadzono obsługę klauzuli DELETE w instrukcji MERGE, możliwość użycia instrukcji INSERT, UPDATE, DELETE, MERGE wewnątrz instrukcji SELECT, polepszono instrukcję CALL. Ponadto wprowadzono także usprawnienia na poziomie składni DML w kontekście wygodniejszej obsługi kolumn identyfikacyjnych oraz kolumn o generowanych wartościach. Wprowadzono udogodnienia w kontekście stronicowania danych m.in. poprzez implementację klauzuli OFFSET. Poprawiona także została funkcjonalność funkcji okna OVER poprzez wprowadzenie do standardu dodatkowych instrukcji tj. NTILE, opcji grupowania, czy też nawigowania wewnątrz okna jak również zagnieżdżone nawigowanie [3].

Jest to także wersja standardu SQL, w której wprowadzono wiele nowości w kontekście obsługi danych temporalnych m.in. możliwość tworzenia tabel temporalnych, z uwzględnieniem okresów czasowych oraz modyfikowania ich zawartości. Wprowadzono m.in. klauzulę PERIOD FOR [6]. Do standardu tego zostały wprowadzone pewne korekty w roku 2013 i 2015.

Standard SQL:2016

W standardzie tym wprowadzono m.in. obsługę plików JSON, polimorficznych funkcji tabelowych. Stanowi on ósmą rewizję standardu ISO (z 1986 roku) oraz ANSI (1987). Przyjęty przez komisję w grudniu 2016 roku, składa się z 9 części. Standard ten wprowadza 44 nowe opcjonalne funkcjonalności, z czego połowa dotyczy obsługi plików JSON m.in. możliwość tworzenia dokumentów JSON (Java Script Object Notation), dostęp i przeszukiwanie wybranych części dokumentu JSON. Rozpoznawanie wzorca rekordów,

formatowanie daty i czasu. Wprowadzono nową funkcję LISTAGG, która transformuje grupy rekordów w łańcuch tekstowy, gdzie poszczególne elementy oddzielone są odpowiednim separatorem oraz nowy typ danych DECFLOAT. Ponadto w standardzie tym uwzględniono polimorficzne funkcje tabelowe bez jawnie określonego typu zwracanej wartości. Do standardu tego pierwsza korekta została wprowadzona w roku 2019 [7,8,9].

Temporalne języki zapytań

W latach 80 i 90-tych ubiegłego stulecia powstało wiele konstrukcji, które pozwalały manipulować danymi temporalnymi. Wśród nich wyróżnić można: HSQL (Historical Query Language), HQUEL, Tquel (Temporal Query Language), czy TSQL2 [10].

To właśnie język TSQL2 stanowił ostateczną wersję będącą próbą połączenia różnych podejść i koncepcji zawartych we wcześniejszych językach. Był on także na owe czasy propozycją zbudowania spójnego, temporalnego rozszerzenia standardu SQL 92.

Jednakże ze względu na liczne kontrowersje oraz różnicę zdań wśród komisji ostatecznie został wycofany ze specyfikacji standardu języka SQL w roku 2001. Powrót do uwzględnienia temporalności w standardzie języka SQL nastąpił dopiero po kolejnej dekadzie w roku 2011, ale już ze zmienioną koncepcją w stosunku do założeń, na których oparty był język TSQL2.

Temporalne rozszerzenia w kontekście standardu języka SQL

Główne elementy dotyczące obsługi danych temporalnych, które zostały wprowadzone w standardzie SQL 2011 to [3,6,11,12]:

- 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

Definicja okresu czasu

Okres czasu definiowany jest, jako interwał na osi czasu o długości wyznaczonej przez czas rozpoczęcia i zakończenia, który jest odpowiednio wiązany z poszczególnymi rekordami składowanymi w tabeli. Stanowi on dodatkowy element tabeli, który posiada swoją własną nazwę i określa kolumny w tabeli, w których składowany jest jego początek i koniec. Standard SQL 2011 wprowadza definicję okresu czasowego w postaci metadanych do tabel. Jednakże nie zakłada on wprowadzenia nowego typu danych, jako okresu czasowego, co z drugiej strony wydaje się być usprawiedliwione, ponieważ skutkowałoby to dużą kosztochłonnością i czasochłonnością adaptacji już istniejących na rynku rozwiązań (m.in. zmiany w specyfikacji interfejsów dostępowych do baz danych, w środowiskach programistycznych, w których tworzona jest aplikacja komunikująca się z serwerem SQL i przetwarzająca dane na nim się znajdujące). W efekcie końcowym, możliwość użycia takiego typu danych w praktycznych zastosowaniach znacznie byłaby oddalona w czasie.

Tabele temporalne

Tworzone tabele temporalne mogą być dwojakiego rodzaju: tabele wersjonowane przez aplikację (aplikacyjnie) i tabele wersjonowane systemowo (przez system RDBMS). Standard SQL:2011 dopuszcza użycie jednego okresu czasu aplikacyjnego i jednego okresu czasu systemowego. Obsługa czasu transakcyjnego (czas, kiedy rekord jest utrwalany w bazie) jest realizowana poprzez tabele wersjonowane systemowo zaś obsługa czasu rzeczywistego (czas, kiedy dany rekord jest aktualny w modelowanym świecie rzeczywistym) realizowana jest za pomocą tabel wersjonowanych aplikacyjnie.

Tabele temporalne wersjonowane przez aplikację

Tabele wersjonowane przez aplikacje używane są wszędzie tam, gdzie istotne jest rejestrowanie okresów czasowych, w których dane rekordy były aktualne. Na użytkowniku spoczywa wymóg zdefiniowania granic okresu czasowego, w którym dany rekord jest aktualny (mogą to być dowolne wartości z przeszłości, teraźniejszości lub z przyszłości). Nazwa definiowanego czasokresu przez użytkownika jest dowolna. Podobnie nazwy pól przechowujących początek i koniec zdefiniowanego przedziału czasowego także są dowolne. Natomiast istotne jest to, że oba te pola muszą być tego samego typu DATE lub TIMESTAMP. Dzięki temu możliwe jest bezpośrednie użycie klauzuli INSERT do dodawania rekordów do tabeli czasowej a także ich modyfikowania i usuwania za pomocą standardowej postaci instrukcji UPDATE i DELETE. Natomiast nazwa czasokresu musi być różna od nazwy kolumn tabeli.

Przykładowa składnia tworzenia tabeli temporalnej wersjonowanej przez aplikację:

```
CREATE TABLE tabela_wersjonowana_aplikacyjnie
(
pole typ danych, … [n],
początek typ_danych {DATE,TIMESTAMP}
koniec typ_danych {DATE,TIMESTAMP},
PERIOD FOR okres_czasowy (poczatek, koniec)
)
```

Przykład dodania rekordu do tabeli czasowej za pomocą standardowej postaci instrukcji INSERT

```
INSERT INTO tabela_wersjonowana_aplikacyjnie
VALUES (wartość_1, wartość_2, …[n])
```

Tabele temporalne wersjonowane systemowo.

Używane w aplikacjach, które muszą składować w bazie danych szczegółową i dokładną historię zmian danych ze względu na narzucone wymagania biznesowe, uregulowania prawne, czy też uwarunkowania medyczne etc.

Jednym z głównych wymogów takich aplikacji jest to, że nie może być utracona informacja o stanie poprzednim sprzed aktualizacji lub w momencie usunięcia rekordu. Przed wykonaniem aktualizacji, czy też przed usunięciem rekordu musi być zapamiętany jego stan poprzedni.

Wartość dla kolumn definiujących początek przedziału czasowego jest generowana automatycznie (przez system baz danych) w momencie użycia klauzuli INSERT INTO dla znacznika czasowego transakcji. Wartość dla kolumny końca przedziału czasowego ustawiana jest automatycznie, jako maksymalna wartość dopuszczona dla typu danych przypisanego do tej kolumny (w zależności od użytego środowiska RDBMS).

Użytkownicy nie mogą zmieniać ani przypisywać wartości dla kolumn zdefiniowanych, jako początek i koniec przedziału czasowego. Ponadto użytkownicy nie mogą także w żaden sposób modyfikować zawartości rekordów historycznych.

Podejście takie gwarantuje niezmienność przechowywanej w bazie historii zmian. Aktualizacje wierszy wykonywane są przez system w momencie aktualizacji lub usunięcia danych z kolumn nieokresowych, które nie definiują przedziału czasowego dla okresu.

Dopuszczalna jest dowolna nazwa dla kolumn, które stanowią początek i koniec przedziału czasowego SYSTEM_TIME. Obie kolumny muszą mieć przypisany ten sam typ danych DATE lub TIMESTAMP. Jednakże w praktycznych rozwiązaniach, ze względu na specyfikę użycia (rejestracja czasu transakcyjnego) znacznie lepiej sprawdzi się typ TIMESTAMP pozwalający przechowywać wartości z dokładnością do ułamkowych części sekundy.

Przykładowa składnia tworzenia tabeli temporalnej wersjonowanej systemowo:

CREATE TABLE *tabela_wersjonowana_systemowo*
(pole typ_danych,…[n],
poczatek typ_danych {DATE,TIMESTAMP} GENERATED ALWAYS AS ROW START,
koniec typ_danych {DATE,TIMESTAMP} GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME (poczatek, koniec)) WITH SYSTEM VERSIONING

Aktualizacja i usuwanie rekordów z określonego przedziału czasowego-tabele wersjonowane aplikacyjnie

Rekordy w tabeli czasowej mogą być aktualizowane za pomocą standardowej postaci instrukcji UPDATE i DELTE. Jednakże wersja standardu SQL:2011 wprowadza rozszerzenia klauzuli UPDATE z punktu widzenia temporalnej aktualizacji poprzez wprowadzenie klauzuli FOR PORTION OF, dzięki której możliwe jest aktualizowanie rekordów, które są zawarte w przedziale czasowym podanym, jako parametr wymienionej klauzuli. W przypadku istnienia rekordów, dla których punkt początkowy lub końcowy przedziału czasowego wykraczają poza granicę czasokresu zdefiniowanego w klauzuli FOR PORTION OF, takie rekordy są dzielone odpowiednio na dwa lub trzy ciągłe rekordy (zależnie od zakresu pokrywania się czasookresów zdefiniowanych dla wybranego rekordu oraz określonego w klauzuli FOR PORTION OF) oraz rekord, którego czasookres całkowicie zawiera się w przedziale zdefiniowanym w klauzuli FOR PORTION OF.

UPDATE tabela_wersjonowana_aplikacyjnie
FOR PORTION OF okres_czasowy FROM DATE data_1 TO DATE data_2
SET pole=wartość WHERE warunek

W ten sam sposób standard SQL 2011 ubogaca klauzulę DELETE, aby efektywniej usuwać rekordy, które są „ważne" w okresie czasu wyrażonego w klauzuli FOR PORTION OF. (Usuwane są rekordy całkowicie zwierające się w zdefiniowanym czasookresie, natomiast w przypadku istnienia zbioru rekordów, dla których czasookres wykracza poza granice „ważności" wykonywana jest akcja wstawienia tych rekordów do tabeli temporalnej).

DELETE tabela_wersjonowana_aplikacyjnie
FOR PORTION OF okres_czasowy FROM DATE data_1 TO DATE data_2
WHERE warunek

Aktualizacja i usuwanie rekordów z określonego przedziału czasowego-tabele wersjonowane systemowo

Klauzula UPDATE i DELETE w odniesieniu do tabeli wersjonowanej systemowo działają tylko na bieżących (aktualnych na daną chwilę) rekordach. Użytkownicy nie mogą modyfikować ani też usuwać systemowych rekordów historycznych. Nie mogą też modyfikować czasookresu systemowego (początku i końca przedziału) dla rekordów bieżących oraz rekordów historycznych. Uruchomienie polecenia UPDATE lub DELETE na tabelach wersjonowanych systemowo skutkuje automatycznym utworzeniem rekordu historycznego dla każdego aktualizowanego bądź usuwanego rekordu.

Temporalne ograniczenie klucza podstawowego-tabele wersjonowane aplikacyjnie

Podczas definiowania ograniczenia PRIMARY KEY dla pól składających się na klucz główny tabeli czasowej (zwykle zawierający identyfikator oraz początek i koniec przedziału czasowego) możliwe jest wymuszenie ograniczenia, które nie zezwala na nakładanie się czasookresów dla pól wchodzących w skład klucza podstawowego (dzięki czemu możliwe jest wymuszenie stanu, gdzie np. dany wykładowca w określonym czasookresie jest zatrudniony tylko i wyłącznie w jednej jednostce). Implementowane jest to za pomocą klauzuli WITHOUT OVERLAPS, bez użycia, której możliwy byłby scenariusz zatrudnienia wykładowcy w tym samym czasookresie w kilku jednostkach (przy czym np. jeden wakat byłby aktywny przez cały czasookres, a drugi mógłby się zaczynać lub wygasać w trakcie obowiązywania aktualnego czasookresu, bądź też oba byłyby aktywne przez cały czasookres)

ALTER TABLE tabela_wersjonowana_aplikacyjnie
ADD PRIMARY KEY (pole_id, okres_czasowy WITHOUT OVERLAPS)

Temporalne ograniczenie klucza podstawowego - tabela wersjonowana systemowo

Ograniczenia takie są egzekwowane tylko dla aktualnych rekordów. Innymi słowy ograniczenia te były wymuszane, kiedy historyczny już rekord (na ten moment) był tworzony i wówczas był rekordem bieżącym (w przeszłości). Zatem nie ma konieczności uwzględniania kolumn będących początkiem i końcem przedziału czasowego, czy też nazwy okresu w definicji klucza podstawowego. Ograniczenie klucza podstawowego przyjmuje identyczną postać jak w konwencjonalnym zapisie dla klasycznej tabeli nietemporalnej.

Temporalne ograniczenie integralności referencyjnej-tabele wersjonowane aplikacyjnie

Ograniczenia te mają zastosowanie w przypadkach implementacji referencji pomiędzy temporalną tabelą nadrzędną a temporalną tabelą podrzędną. Ograniczenie to wymusza istnienie rekordu w tabeli podrzędnej, którego okres zawiera się w pełni, w okresie dołączanego rekordu z tabeli nadrzędnej (okres „obowiązywania" rekordu potomnego zawiera się w okresie „istnienia" rekordu nadrzędnego). Dopuszczalne jest istnienie więcej niż jednego okresu, a więc kilku sąsiadujących ze sobą okresów w tabeli nadrzędnej dla rekordu z tabeli podrzędnej wersjonowanej aplikacyjnie.

Przykładowa składnia wymuszenia temporalnej integralności referencyjnej:

ALTER TABLE tabela_wersjonowana_aplikacyjnie
ADD FOREIGN KEY (pole_id, PERIOD okres_czasowy_tabeli_podrzędnej)
REFERENCES (pole_id_tabeli_nadrzędnej, PERIOD okres_czasowy_tabeli_nadrzędnej)

Temporalne ograniczenie integralności referencyjnej-tabele wersjonowane systemowo

Podobnie jak w przypadku definiowania ograniczenia klucza podstawowego dla tabeli wersjonowanej systemowo także dla ograniczenia referencyjnego nie stosuje się ograniczeń temporalnych. Ograniczenia te są egzekwowane tylko w odniesieniu do bieżących rekordów.

Predykaty czasowe i zapytania na tabelach temporalnych wersjonowanych aplikacyjnie

Tabele temporalne mogą być odpytywane w tradycyjny sposób przy użyciu tradycyjnej składni.

SELECT lista_pól FROM tabela_wersjonowana_aplikacyjnie WHERE warunek=wartość AND początek<=data_1 AND koniec> data_1

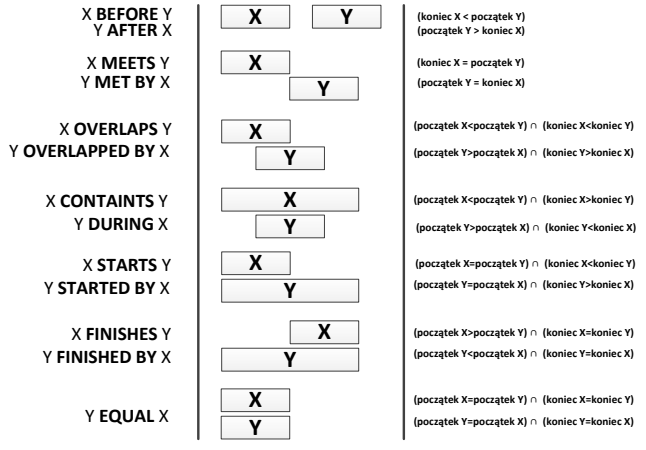
Można także posłużyć się w tym celu nowymi predykatami wprowadzonymi do standardu SQL:2011 tj.: CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES, IMMEDIATELY SUCCEEDS. Dostępne są one tylko dla zapytań tworzonych w oparciu o tabele temporalne wersjonowane aplikacyjnie.

W tabeli 1 przedstawione zostały interwały czasowe zaproponowane przez Allena [13].

Tabela 1. Zestawienie relacji pomiędzy interwałami czasowymi wg Allena

Lp	Relacja	Relacja odwrotna
1	BEFORE (x,y)	AFTER (y,x)
2	MEETS (x,y)	MET BY (y,x)
3	OVERLAPS (x,y)	OVERLAPPED BY (y,x)
4	DURING (x,y)	CONTAINS(y,x)
5	STARTS (x,y)	STARTED BY (y,x)
6	FINISHES (x,y)	FINISHED BY (y,x)
7	EQUAL (x,y)	EQUAL (y,x)

Na rysunku 2 zaprezentowana została interpretacja graficzna interwałów czasowych zaproponowanych przez Allena.



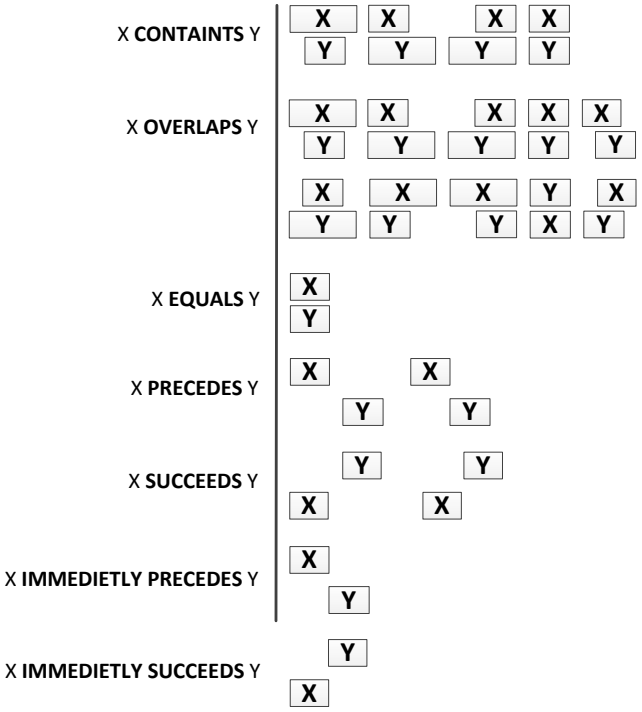
Rys.2. Interpretacja graficzna operatorów temporalnych zaproponowanych przez Allena, Źródło: opracowanie własne

Predykaty czasookresu uwzględnione w standardzie SQL:2011 są podobne funkcjonalnie do interwałów zaproponowanych przez Allena (aczkolwiek nieidentyczne) [12]. Mają one nieco zmodyfikowaną postać w stosunku do predykatów zaproponowanych przez Allena. W tabeli 2 przedstawiono porównanie operatorów temporalnych zaproponowanych przez Allena i tych wprowadzonych do standardu SQL:2011, wskazując tym samym równoważność wybranych operatorów.

Tabela 2. Porównanie operatorów temporalnych zaproponowanych przez Allena i wprowadzonych do standardu SQL:2011

Lp	Operatory Allena	Operatory SQL:2011
1	BEFORE (x,y)	PRECEDES
2	MEETS (x,y)	PRECEDES, IMMEDIETLY PRECEDES
3	OVERLAPS (x,y)	OVERLAPS
4	DURING (x,y)	OVERLAPS
5	STARTS (x,y)	OVERLAPS, CONTAINTS
6	FINISHES (x,y)	OVERLAPS, CONTAINTS
7	EQUAL (x,y)	OVERLAPS, CONTAINTS, EQUALS
8	AFTER (y,x)	SUCCEEDS
9	MET BY (y,x)	SUCCEEDS, IMMEDIETLY SUCCEEDS
10	OVERLAPPED BY (y,x)	OVERLAPS
11	CONTAINS (y,x)	OVERLAPS, CONTAINS
12	STARTED BY (y,x)	OVERLAPS
13	FINISHED BY (y,x)	OVERLAPS

Na rysunku 3 zaprezentowana została interpretacja graficzna operatorów temporalnych wprowadzonych do standardu SQL:2011.



Rys.3. Interpretacja graficzna operatorów temporalnych wprowadzonych do standardu SQL:2011, Źródło: opracowanie własne

Poniżej przedstawiono alternatywny zapis przy użyciu predykatu zawierania CONTAINS, w odniesieniu do tradycyjnego zapisu:

SELECT lista_pól FROM tabela_wersjonowana_aplikacyjnie
WHERE warunek=wartość AND okres_czasu CONTAINS DATE data

Postać zapisu zwracającego zbiór rekordów z zadanego okresu czasu przy użyciu predykatu OVERLAPS:

SELECT lista_pól FROM tabela_wersjonowana_aplikacyjnie
WHERE warunek=wartość AND okres_czasu OVERLAPS PERIOD (DATE data_1, DATE data_2)

Zapytania na tabelach wersjonowanych systemowo

Tabele wersjonowane systemowo używane są przede wszystkim do śledzenia danych historycznych. Standard SQL:2011 wprowadza trzy dodatkowe rozszerzenia składniowe dla zapytań bazujących na tabelach wersjonowanych systemowo:

1. FOR SYSTEM_TIME AS OF pozwala odpytywać tabelę od określonego momentu w czasie

SELECT lista_pól, początek, koniec FROM tabela_wersjonowana_aplikacyjnie FOR SYSTEM_TIME AS OF TIMESTAMP znacznik_daty_i_czasu

2. FOR SYSTEM_TIME FROM … TO … pozwala pobierać dane pomiędzy dwoma punktami w czasie, przy czym wartość końcowa przedziału czasowego nie jest dołączana-przedział domknięty-otwarty <początek,koniec)

SELECT lista_pól, początek, koniec FROM tabela_wersjonowana_aplikacyjnie FOR SYSTEM_TIME FROM OF TIMESTAMP znacznik_daty_i_czasu_1 TO znacznik_daty_i_czasu_2

3. FOR SYSTEM_TIME BETWEEN … AND … pozwala pobierać dane pomiędzy dwoma punktami w czasie, przy czym wartość końcowa przedziału czasowego jest dołączana – przedział domknięty-domknięty <początek,koniec>.

SELECT lista_pól, początek, koniec FROM tabela_wersjonowana_aplikacyjnie FOR SYSTEM_TIME BETWEEN TIMESTAMP znacznik_daty_i_czasu_1 AND znacznik_daty_i_czasu_2

W przypadku, gdy zapytanie na tabeli wersjonowanej systemowo nie określa jawnie w/w opcji składniowych, wówczas domyślnie zakłada się FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP w skutek, czego zwracane są tylko bieżące rekordy. Jest to dobre rozwiązanie wszędzie tam gdzie najczęściej pobierane są aktualne rekordy. W celu pobrania aktualnych i historycznych rekordów zapytanie jest parametryzowane w taki sposób, że wartościom początku i końca przedziału czasowego przypisuje się odpowiednio wartość minimalną i maksymalną dostępną dla typu danych przypisanego do tych kolumn.

Bitemporalne tabele

Tabela bitemporalna może być wersjonowana zarówno systemowo oraz aplikacyjnie. Pozwala to jednocześnie zapamiętać czas, kiedy dane te są aktualne (prawdziwe) w danym przedziale czasowym oraz dodatkowo odnotować fakt utrwalenia tych danych w bazie danych [1,10]. Tabele bitemporalne łączą oba rozwiązania, gdzie dla wersjonowania przez aplikację użytkownik jest odpowiedzialny za wprowadzenie wartości czasu dla początku i końca przedziału czasowego, dla wersjonowania systemowego wartości te są generowane automatycznie. Dla czasu rzeczywistego możliwe jest wykonanie aktualizacji i usuwania rekordów temporalnych, natomiast dla czasu systemowego modyfikowane i usuwane mogą być tylko rekordy aktualne. Zapytania na tabelach bitemporalnych pozwalają na użycie predykatów określonych dla tabel wersjonowanych przez aplikacje jak i tabel wersjonowanych systemowo.

Podstawowa składnia tworzenia tabeli bitemporalnej ma postać:

CREATE TABLE *tabela_bitemporalna*
(pole typ_danych,…[n],
poczatek_app typ_danych {DATE,TIMESTAMP},
koniec_app typ_danych {DATE,TIMESTAMP},
PERIOD FOR okres_app (poczatek_app, koniec_app)
poczatek_sys typ_danych {DATE,TIMESTAMP} GENERATED ALWAYS AS ROW START,
koniec_sys typ_danych {DATE,TIMESTAMP} GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME (poczatek_sys, koniec_sys)
) WITH SYSTEM VERSIONING

Implementacja temporalnych rozszerzeń standardu SQL:2011 w RDBMS

Kilku wiodących dostawców środowisk RDBMS zaimplementowało w swoich flagowych produktach wybrane rozwiązania zawarte w standardzie języka SQL 2011. Przedstawia się to w różny sposób się u poszczególnych producentów. W różnym zakresie i stopniu zostały zaimplementowane wybrane funkcjonalności, nie zawsze w sposób w pełni zgodny z postacią ujętą w standardzie SQL. Jednym z pierwszych produktów, w którym zawarta została funkcjonalność składowania i przetwarzania danych temporalnych była baza danych IMB DB2 od wersji 10 [14-16]. Należy zaznaczyć, że jest to najbardziej obszerna realizacja standardu SQL:2011 [24-26]. Najbardziej znaczącą różnicą jest to, że czas rzeczywisty w standardzie na platformie DB2 nazywany jest czasem biznesowym, a czas transakcyjny platforma DB2 nazywa czasem systemowym. Ponadto platforma IBM DB2 nie obsługuje temporalnych ograniczeń referencyjnych w oparciu o tabele z czasem rzeczywistym. Ograniczenia te muszą być zaimplementowane samodzielnie m.in. za pomocą procedur składowanych czy też wyzwalaczy. DB2 obsługuje także jawne (opcja domyślna) i niejawne (nie jest wyświetlana zawartość kolumn czasowych) znaczniki czasowe. Platforma ORACLE obsługuje dane temporalne od wersji 10g/11g, choć obsługa danych temporalnych w oparciu o standard SQL 2011 uwzględniona została w wersji 12c [17-18,24-26,29]. Obsługuje jawne i niejawne znaczniki czasowe. Co ciekawe dopuszcza użycie wartości NULL dla określenia końca czasookresu. Nie obsługuje temporalnych rozszerzeń instrukcji UPDATE i DELETE dla czasu rzeczywistego. Nie obsługuje także temporalnego ograniczenia PK [23]. Teradata obsługuje dane czasowe od wersji 10. W kolejnych wersjach nieco poszerzona została funkcjonalność temporalna. Należy zaznaczyć, iż model temporalny w tym środowisku oparty jest na modelu TSQL2. Teradata (jako jedyne środowisko) obsługuje typ danych PERIOD. Domyślnie obsługuje niejawne znaczniki czasowe, aby wyświetlić zawartość pól czasowych należy wymienić je jawnie na liście pól klauzuli SELECT [20-21,24-26,29]. Platforma MS SQL Server od wersji 2016 [18], realizuje tylko fragmentarycznie obsługę standardu SQL:2011, w oparciu o tabele wersjonowane systemowo (dla czasu transakcyjnego). Bardziej zaawansowaną i złożoną funkcjonalność użytkownik musi implementować samodzielnie m.in. za pomocą typów danych oraz funkcji definiowanych przez użytkownika. Ponadto funkcjonalność temporalna jest także dostępna w produktach: SAP HANA 2.0. [22] oraz w wybranych otwartych produktach m.in. PostgreSQL, gdzie m.in. został wprowadzony nowy typ danych zakresu wraz z odpowiednimi funkcjami i predykatami [28], czy też MariaDB [23]. Nie brak także nowatorskich rozwiązań m.in. w oparciu o platformę SAP HANA autorzy zbudowali własny prototyp oparty o pamięciowy magazyn kolumnowy [27] oraz opracowali nową

strukturę indeksu czasowego TimeLine Index. Dzięki czemu znacząco poprawili wydajność zapytań temporalnych.

Podsumowanie

Udogodnienia wprowadzone do standardu SQL:2011 wychodzą naprzeciw oczekiwaniom użytkowników relacyjnych systemów bazodanowych. Oczywiście przy założeniu, że korzystają oni ze środowisk, w których zaimplementowane zostało temporalne rozszerzenie standardu SQL:2011. „Odpowiedzialność" za implementację przechowywania historii zmian, jak również temporalnego przetwarzania takich danych jest przeniesiona z projektanta bazy danych i/lub programisty takiej aplikacji na konkretny RDBMS posiadający zaimplementowane temporalne rozszerzenie standardu SQL:2011. Programista jest zwolniony z implementacji całego mechanizmu przetwarzania danych temporalnych, czy to po stronie serwera baz danych, w postaci odpowiednio implementowanych procedur składowanych, funkcji, wyzwalaczy, czy też odpowiednich typów danych stworzonych w zewnętrznych środowiskach deweloperskich (w przypadku programisty serwera baz danych), czy po stronie serwera aplikacji (w przypadku programisty aplikacji bazodanowych). Korzysta on z gotowych rozwiązań wkomponowanych w RDBMS, którego składnia jest zgodna z rozszerzeniem temporalnym języka SQL wprowadzonym do standardu SQL:2011. Wcześniej wymagało to np. na platformie SQL Server stworzenia własnych typów danych UDT (np. dla okresu czasu) za pomocą środowiska CLR (Common Language Runtime) oraz specjalnych operatorów temporalnych za pomocą specjalnie implementowanych funkcji użytkownika UDF CLR, procedur składowanych czy też wyzwalaczy. Alternatywnym rozwiązaniem było użycie mechanizmu CDC (Change Data Capture), który jest dostępny także m.in. na platformie ORACLE czy IBM DB2.

Na przestrzeni kilku dziesięcioleci standard języka SQL znacznie ewoluował. W kolejnych jego rewizjach wprowadzane były kolejne udogodnienia i nowości. Przez cały ten okres rozwoju standardu języka SQL podjęto dwukrotną próbę jego rozszerzenia o elementy temporalności.

We wcześniejszym zapisie dotyczącym temporalnego rozszerzenia standardu SQL tabela miała ukrytą kolumnę składującą czas, do której nie można było uzyskać dostępu za pomocą zwykłego, standardowego zapytania wybierającego. Możliwe to było za pomocą specjalnych, wbudowanych funkcji. W standardzie SQL:2011 wprowadzono niewielki zestaw klauzul i predykatów, których zakres działania jest jasno sprecyzowany, a sama konstrukcja tabel temporalnych pozwala w bezproblemowy sposób uzyskać dostęp do danych czasookresu, które są jawnie składowane, jako kolejne pola tabeli temporalnej.

Pomimo wielu udogodnień wprowadzonych w standardzie SQL:2011 w kontekście obsługi danych temporalnych wciąż istnieje jeszcze wiele nieuwzględnionych funkcjonalności o czym wspominają także inni autorzy [6]. M.in. nadmienić można:

- Możliwość łączenia rekordów z dwu tabel w taki sposób, aby czas aplikacyjny i systemowy nakładały się (standard SQL:2011 dopuszcza użycie predykatu OVERLAPS jako złączenia wewnętrznego ale nie zewnętrznego)
- Obsługa agregatów okresu czasu i grupowania okresów czasu z uwzględnieniem czasu aplikacji i systemowego
- Temporalne wsparcie dla operatów operacji na zbiorach UNION, EXCEPT, INTERSECT
- Obsługa wielu okresów czasowych na tabeli
- Wsparcie dla okresów czasowych nieprzewidywalnych w czasie

Wiodący na rynku dostawcy platform RDBMS zaimplementowali nowe rozszerzenia temporalne w swoich produktach. Nie zawsze jednak jest to pełna implementacja standardu SQL:2011, a także nie zawsze w pełni zgodna z tym, co zostało zawarte w specyfikacji standardu. Środowisko SQL Server obsługuje tylko czas transakcyjny i tabele temporalne wersjonowane systemowo [19], a Teradata oparte jest na temporalnym modelu TSQL2 [24-26,29]. Z jednej strony należy oczekiwać, że wraz z upływem kolejnych lat dostawcy RDBMS będą implementować w swoich produktach kolejne funkcjonalności zawarte w standardzie SQL:2011 (w przypadku niepełnej jej implementacji), dostosowywać już istniejące pod kątem zgodności ze standardem, a także wprowadzać swoje własne usprawnienia, jak ma to miejsce w przypadku serwera PostgreSQL [28]. Z drugiej strony należy także spodziewać się kolejnych udogodnień temporalnych wprowadzanych w następnych wersjach standardu SQL.

Artykuł ten stanowi kontynuację poprzednich rozważań [1] i zarazem punkt wyjścia do dalszych poszukiwań i prac badawczych. Jego kontynuacją będzie prezentacja wyników analizy porównawczej ze szczególnym uwzględnieniem implementacji temporalnego rozszerzenia standardu w wybranych komercyjnych i niekomercyjnych rozwiązaniach dostępnych na rynku.

***Autor**: 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] Date C.J., Darwen H., Lorentzos N., Time and relational theory Temporal Databases in the Relational Model and SQL, 2014, Morgan Kaufmann
[3] Zemke F., What's new in SQL:2011, SIGMOD Record, Vol. 41, No. 1, March 2012, 67-73, https://sigmodrecord.org/publications/sigmodRecord/1203/pdfs/10.industry.zemke.pdf
[4] Eisenberg A., Kulkarni K., Melton J., Michels J., Zemke F., SQL:2003 Has Been Published, SIGMOD Record, Vol. 33, No. 1, March 2004, SQL, 119-126 2013 http://www09.sigmod.org/sigmod/record/issues/0403/E.JimAndrew-standard.pdf
[5] SQL:2008: https://web.archive.org/web/20110628130925/http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/
[6] Kulkarni K., Jan-Eike Michels, 2012, *Temporal features in SQL:2011,* 34-43 https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf (1.12.2017)
[7] SQL:2016: https://www.iso.org/obp/ui/#iso:std:iso-iec:9075:-2:ed-5:v1:en
[8] SQL:2016: https://modern-sql.com/blog/2017-06/whats-new-in-sql-2016
[9] Michels J., Hare K., Kulkarni K., Zuzarte C., Liu Z. H., Hammerschimdt B., Zemke F., The New and Improved SQL:2016 Standard, 51-60 https://sigmodrecord.org/publications/sigmodRecord/1806/pdfs/08_Industry_Michels.pdf
[10] Kania K., 2004, Temporalne bazy danych w systemach informatycznych zarządzania, Wydawnictwo Akademii Ekonomicznej w Katowicach, Katowice
[11] https://link.springer.com/content/pdf/10.1007%2F978-1-4899-7993-3_80729-1.pdf
[12] Date C.J., Darwen H., Lorentzos N., Temporal Data and the Relational Model, Morgan Kaufman (2003)

[13] James F. Allen, “Maintaining knowledge about temporal intervals”, *Communications of ACM*, Vol. 26, No. 11, November 1983
[14] 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-matter-of-time-temporal-data-management-in-db2-10-ibm
[15] IBM: https://www.ibm.com/developerworks/data/library/techarticle/dm-1410temporal-tables-db2zos/index.html
[16] IBM: https://www.ibm.com/support/knowledgecenter/en/SSEPEK_11.0.0/admin/src/tpc/db2z_creatingtemporal.html
[17] Salvisberg P., *Multi-temporal Features in Oracle 12c*, https://www.salvis.com/blog/2014/01/04/multi-temporal-database-features-in-oracle-12c/
[18] Oracle: https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/ilm/temporal/temporal.html
[19] SQL Server: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?view=sql-server-ver15
[20] 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
[21] Teradata https://docs.teradata.com/reader/vi_E5DAzLxKE_Xhr8yNIWw/BuO5td4SJIxUXyFilG6Geg
[22] SAP HANA: https://help.sap.com/viewer/6a504812672d48ba865f4f4b268a881e/Cloud/en-US/cf3523ab01834f5e84a32164c1fd597a.html
[23] MariaDB: https://mariadb.com/kb/en/temporal-data-tables/
[24] Petković D., Support of Temporal Data in Database Systems, International Journal of Computer Applications (0975 –8887), Volume 152 –No.4, October 2016, 26-33
[25] Petković D., Temporal Data in Relational Database Systems: A Comparison, Conference: WorldCIST (1) 2016, At Recife, Volume: 1
[26] Petković, D., Temporal Data in Enterprise Database Systems, The 7th International Conference on Information Technology, 2015,pp 276-282
[27] Kaufmann M., Fisher P., Farber F., Vagenas P., Kossmann D., Comprehensive and Interactive Temporal Query Processing with SAP HANA, Conference Paper in Proceedings of the VLDB Endowment 6(12) August 2013,
[28] Böhlen M., Hanspeter M.; Dignös A. ; Gamper J. ; Jensen Ch., Database Technology for Processing Temporal Data (Invited Paper), University of Zurich, 2018 https://www.zora.uzh.ch/id/eprint/161083/1/LIPIcs-TIME-2018-2.pdf
[29] Owusuaa Mensah R., Amankona V., SQL and Temporal Database Research: Unified Review and Future Directions, International Research Journal of Engineering and Technology(IRJET), Volume: 04 Issue: 09 Sep 2017, 1160-1168 https://www.irjet.net/archives/V4/i9/IRJET-V4I9207.pdf