Biuletyn WAT
Vol. LXV, Nr 1, 2016

Projektowanie baz danych z pełną historią zmian danych — model bitemporalnej bazy danych i operacje zapisu

STEFAN ROZMUS

Wojskowa Akademia Techniczna, Wydział Cybernetyki, Instytut Systemów Informatycznych,
00-908 Warszawa, ul. gen. S. Kaliskiego 2, stefan.rozmus@wat.edu.pl

Streszczenie. Znane i sprawdzone są dobre praktyki projektowania baz danych. Jednak w przypadku gdy baza danych musi gromadzić pełną historię zmian danych, wykonanie projektu bazy danych staje się zdecydowanie trudniejsze. Uwzględnienie aspektów temporalnych ze swej natury zamienia związki pomiędzy powiązanymi obiektami na związki pomiędzy stanami tych obiektów. Dodatkowo, możliwość różnej interpretacji zależności czasowych powoduje, że nie wypracowano jeszcze ogólnie przyjętej metodyki projektowania temporalnych baz danych. Artykuł jest pierwszym z serii artykułów podsumowujących doświadczenia Autora zdobyte w ramach prac nad systemem Centralnej Ewidencji Pojazdów i Kierowców (CEPiK). W artykule przedstawiono sposób podejścia do projektowania relacyjnej bazy danych przechowującej pełną historię zmian stanów obiektów. Opisano model bazy danych, szczegółowo wyjaśniono wpływ zmian stanów obiektów na historię życia obiektów oraz zaprezentowano algorytmy operacji zapisu zmieniających stan bazy. Przedstawione podejście zostało praktycznie wykorzystane do budowy jednej z baz danych CEPiK.
Słowa kluczowe: informatyka, relacja bitemporalna, stan obiektu, znacznik czasu ważności, znacznik czasu transakcji
DOI: 10.5604/12345865.1197981

1. Wprowadzenie

Większość aplikacji wykorzystujących technologie baz danych posiada temporalną naturę. Wynika ona głównie z faktu, że cechy (atrybuty) obiektów utrwalanych w bazach danych w ogólnym przypadku ulegają naturalnym zmianom w cyklu swojego życia. Oznacza to, że w danym momencie czasu ustalone są wartości wszystkich atrybutów obiektu. Zbiór tych wartości tworzy stan obiektu, natomiast

w całym cyklu życia obiektu może pojawić się wiele stanów. O ile wymagane jest utrwalenie w relacyjnej bazie danych wszystkich stanów obiektu[1], pojawia się podstawowy problem związany z ustaleniem klucza głównego w relacji. Znany z klasycznego podejścia klucz główny traci, w takim przypadku, unikalność w relacji i staje się swoistego rodzaju spinaczem stanów obiektu. Nie jest to jedyny problem utrudniający lub wręcz uniemożliwiający zastosowanie klasycznego podejścia do projektowania temporalnej bazy danych. Obiekty z reguły występują w określonych związkach. Jeśli uwzględnimy historię zmian, w relacyjnej bazie danych pojawią się w istocie związki pomiędzy stanami obiektów, a nie pomiędzy obiektami. Należy zwrócić uwagę na jeszcze jeden aspekt historii zmian w cyklu życia obiektu. Mogą w nim wystąpić okresy nieciągłości. Przyczyna tego zjawiska może wynikać z samej natury życia obiektu lub też z ustalonych zasad postrzegania cyklu życia obiektu przez obserwatorów zewnętrznych. Przykładem pierwszego przypadku może być istnienie państwa polskiego jako niezależnego kraju w nienakładających się interwałach czasowych, tzn. w latach 1025-1795, 1918-1939 i ponownie od roku 1945 do chwili obecnej. Drugą przyczynę można zilustrować przykładem, niejednokrotnie przywoływanym w literaturze przedmiotu, odnoszącym się do postrzegania osoby jako klienta firmy na podstawie jego aktywności w dokonywaniu zakupów ([3], [20]). Klient pojawia się tylko wtedy, gdy wartość zakupionych przez niego towarów osiągnie ustalony próg, i znika, gdy w określonym przedziale czasu wartość ta spadnie poniżej ustalonego progu. Uwzględnienie okresów nieciągłości w cyklu życia obiektu przysparza kolejnych poważnych problemów, związanych z identyfikacją obiektów, zapisami do bazy danych, interpretacją wyników zapytań i wydajnością bazy danych. Stąd też decyzje w tym zakresie należy podejmować po dokładnym zbadaniu skali tego zjawiska i oszacowaniu stopnia spodziewanych korzyści w stosunku do niezbędnych nakładów.

Kolejne problemy dla baz danych z pełną historią wynikają z następujących faktów (których w ogólnym przypadku nie można wykluczyć):

- pojawiła się informacja o stanie obiektu, który nie został utrwalony przed utrwaleniem stanów późniejszych,
- w momencie utrwalania stanu obiektu nie były znane wartości wszystkich jego atrybutów,
- stwierdzono, że wartości niektórych atrybutów w utrwalonym stanie są błędne i wymagają skorygowania,
- stwierdzono, że pewne utrwalone stany obiektu są błędne (przyczyna błędów jest bez znaczenia) i należy je usunąć.

[1] W niniejszym artykule utrwalenie stanu obiektu rozumiane jest jako trwały zapis (do tabeli relacyjnej bazy danych) rekordu, zawierającego wartości atrybutów obiektu.

2. Przegląd literatury

Problemy projektowania relacyjnych baz danych zajmowały i nadal zajmują znaczące miejsce w literaturze przedmiotu. Od roku 1970, w którym E.F. Codd zaproponował model relacyjny bazy danych, opublikowano wiele prac z tego zakresu. Rozszerzono w nich nie tylko opisy podstaw teoretycznych, lecz także podano wskazówki praktycznego wykorzystania założeń teoretycznych do budowy rzeczywistych systemów bazodanowych ([1], [6], [23]). Temporalna natura systemów informatycznych zaowocowała pojawieniem się nowego nurtu rozwoju baz danych ujmującego problemy wskazane w poprzednim rozdziale. Obejmuje on prace zmierzające do wypracowania standardu projektowania temporalnych baz danych. Znaczący wkład w opracowanie podwalin pod temporalne bazy danych można przypisać kilku autorom ([16], [21], [22]). Kolejne prace, w tym najnowsze (ostatnie dwa lata) poświęcone rozwiązaniom wskazanych powyżej problemów, w większości przypadków albo pochodzą od tych samych autorów, albo koncentrują się wokół koncepcji zaproponowanych przez tych autorów ([4], [5], [7], [8], [10], [19]). W ostatnich latach widoczny jest postęp rozwiązań problemów temporalnych w komercyjnych systemach zarządzania systemami relacyjnych baz danych. Początkowo, jedyna komercyjnie znacząca nakładka na konwencjonalny system zarządzania bazą danych, oferująca rozszerzenia temporalne ([17], [18]), posiadała ograniczenia, które znacząco zawężały zakres możliwości jej wykorzystania. Obecnie została zdominowana przez systemy zarządzania bazami danych oferujące właściwości obsługi aspektów temporalnych na poziomie systemu zarządzania bazą danych. Dotyczy to jednak tylko trzech uznanych producentów oprogramowania tej klasy: IBM [14], Oracle [13], Terradata [15].

Problematyka temporalnych baz danych poruszana jest również w literaturze dotyczącej hurtowni danych. Koncentruje się ona głównie na uwzględnieniu aspektów temporalnych bazy danych opartej o wielowymiarowy model danych. Problematyka ta wykracza poza ramy niniejszego artykułu. Szczegółowy opis podejścia do budowy hurtowni danych można znaleźć np. w [9]. W pracy tej zawarto również szereg przykładów budowy hurtowni danych do zastosowań w różnych obszarach biznesowych. Niemniej jednak aspekty temporalne ujęte w pracy ograniczają się jedynie do uwzględnienia wymiarów wolnozmiennych w wielowymiarowym modelu danych. Godne polecenia są prace [11] i [12], w których zaproponowano rozszerzenie temporalne modelu wielowymiarowego. Model ten opiera się na wynikach badań zarówno w obszarze temporalnych baz danych, jak i w obszarze hurtowni danych.

3. Baza danych i aplikacje temporalne

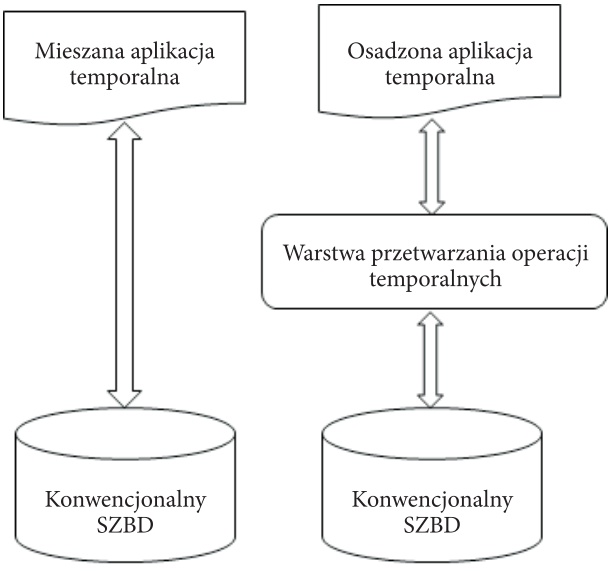
Baza danych jest oparta na modelu danych, który definiuje konstrukcje i formalizmy pozwalające jednolicie opisać oraz udostępniać dane. Model ten dostarcza

zestawu pojęć do opisu struktury danych, ograniczeń integralności oraz operacji pobierania i aktualizacji danych. Model danych, M = (DS, OP, C), składa się z trzech części: struktur danych DS, operacji OP i ograniczeń integralności C.

Temporalny model danych $M^T = (DS^T, OP^T, C^T)$ rozszerza wszystkie trzy powyższe elementy modelu danych w odniesieniu do czasu. Struktury danych DS^T powinny być dostosowane tak, aby można było przechowywać informacje o danych zmiennych w czasie, pozostawiając użytkownikowi wybór danych, dla których te informacje mają być uwzględnione. Operacje OP^T powinny oferować wsparcie dla specjalnych zapytań uwzględniających okresy czasu, ograniczenia integralności C^T powinny obejmować horyzont przeszłych, aktualnych i przyszłych zdarzeń [16].

Jeżeli modyfikacja lub rozszerzenia modelu danych do obsługi danych zmiennych w czasie oznaczają, że realizacja tego modelu ma być zapewniona przez System Zarządzania Bazą Danych (SZBD), to wszelkie zmiany mogą zostać wprowadzone tylko przez dostawcę SZBD. Inne podejście do wsparcia obsługi danych zmiennych w czasie opiera się na wykorzystaniu właściwości konwencjonalnego SZBD i zbudowaniu temporalnej funkcjonalności w aplikacji korzystającej z tego SZBD. Możliwe są w tym przypadku dwa rozwiązania (rys. 1):

a) Mieszana aplikacja temporalna (a),
b) Osadzona aplikacja temporalna (b).



Rys. 1. Aplikacje temporalne oparte o konwencjonalny SZBD

Mieszana aplikacja temporalna zawiera obsługę wszystkich operacji, zarówno nietemporalnych, jak i temporalnych. Operacje te odwołują się bezpośrednio do bazy danych. Należy przy tym zaznaczyć, że implementacja operacji temporalnych bazuje na języku zapytań nieposiadającym rozszerzeń temporalnych. Wykorzystywane jest natomiast specyficzne podejście do konstrukcji schematu bazy danych i określenie jednolitych reguł tworzenia operacji temporalnych.

Implementacja operacji temporalnych w osadzonej aplikacji temporalnej sprowadza się do wywołania usług rozszerzeń temporalnych udostępnianych przez warstwę przetwarzania operacji temporalnych. Warstwa ta pośredniczy w komunikacji pomiędzy aplikacją a bazą danych, pełniąc rolę tłumacza rozszerzeń temporalnych na język bazy danych.

Koncepcja warstwy przetwarzania operacji temporalnych doczekała się niewielu rozwiązań komercyjnych. Jednym z najbardziej znanych jest TimeDB ([17], [18]) stosowany jako rozszerzenie temporalne do baz danych firm Oracle i IBM. Jednak ze względu na swoje ograniczenia, TimeDB nie stanowi platformy do budowy dowolnej aplikacji temporalnej. W konkretnym przypadku zwykle pozostaje opracowanie własnej aplikacji temporalnej. Jeżeli jest to przedsięwzięcie jednostkowe, należy skłaniać się ku mieszanej aplikacji temporalnej. Konieczne jest wtedy ustalenie zasad konstrukcji schematu bazy danych i semantyki operacji temporalnych. Ich zrozumienie przez zespół projektowy, a następnie konsekwentne stosowanie w całym cyklu budowy aplikacji pozwoli zminimalizować wysiłek i koszty jej opracowania oraz ułatwi jej późniejszą konserwację.

W dalszej części artykułu zostanie przedstawiony sposób podejścia do projektowania konwencjonalnej relacyjnej bazy danych przechowującej pełną historię zmian stanów obiektów oraz zasad tworzenia operacji temporalnych mieszanej aplikacji temporalnej, który z powodzeniem został praktycznie wykorzystany przez Autora do budowy jednej z baz danych Centralnej Ewidencji Pojazdów i Kierowców. Podejście to opiera się na modelu bitemporalnej bazy danych. Zapewnia on możliwość rejestrowania aktualizacji czasu ważności, co jest warunkiem koniecznym zachowania pełnej historii zmian stanów obiektów.

Zdaniem Autora zastosowanie zaproponowanego podejścia może przynieść równie dobry skutek w innych obszarach informatyzaji administracji publicznej, np. w tych, o których wspomniano w [24] i [25].

4. Model bitemporalnej bazy danych

Z każdym elementem temporalnej bazy danych związany jest znacznik (stempel) czasu. Jest on wartością czasową powiązaną z wartością danych. Może być reprezentowany za pomocą stałej czasowej lub przedziału czasu. Każdy z nich zapisywany

jest za pomocą chrononu[2] lub chrononów przyjmujących jakąś wartość o zadanej granulacji czasu.

Do zapisania wartości chrononu można wykorzystać jeden z kilku czasowych typów danych (dostępnych w większości SZBD), takich jak time, date, timestamp, datetime, year. Stałe czasowe są punktami w czasie odzwierciedlającymi moment wystąpienia zdarzenia, w wyniku którego podjęto pewne działania skutkujące zmianą stanu obiektu świata rzeczywistego. Zdarzenia zapisywane są za pomocą jednego chrononu. Stan obiektu trwa do chwili, w której pojawi się nowy chronon reprezentujący nową zmianę. Należy tutaj wyraźnie zaznaczyć, że chronon jest niepodzielną jednostką czasu, stąd zmiany stanu obiektu zachodzące w „obrębie" tego samego chrononu skutkują wystąpieniem więcej niż jednego stanu obiektu w tym samym, tak rozumianym czasie.

Przedział czasu reprezentowany jest przez dwa chronony, gdzie jeden z nich jest początkiem (t_p) — dolnym ograniczeniem, a drugi końcem (t_k) — górnym ograniczeniem przedziału. Reprezentuje on czas trwania, w którym obiekt znajdował się w jakimś stanie. Ze względu na charakter najczęściej zadawanych zapytań do bazy danych z pełną historią (zapytania o stan obiektu, który istniał w pewnym momencie czasu) wygodnie jest przyjąć reprezentację przedziałową znacznika czasowego[3].

Jeżeli znacznik czasowy określa czas, w którym obiekt się znajduje (lub się znajdował) w określonym stanie, w świecie rzeczywistym, nazywany jest znacznikiem czasu ważności (ang. *Valid Time*). Zbiór wszystkich stanów obiektu, opatrzonych znacznikami czasu ważności, stanowi jego historię. Biorąc jednak pod uwagę fakt, że dane mogą być korygowane, z każdym stanem konieczne jest również związanie informacji o czasach utrwalenia danego stanu w bazie danych oraz pozostawania w nim bez zmian. Uzyskuje się to poprzez dodatkowe opatrzenie stanów kolejnym znacznikiem czasu, czasu transakcji (ang. *Transaction Time*). W dalszych rozważaniach przyjęto przedziałową reprezentację czasu transakcji z tych samych względów co dla czasu ważności.

Relacja konwencjonalnej bazy danych rozszerzona o znacznik czasu ważności i znacznik czasu transakcji[4] staje się relacją bitemporalną. W relacji tej identyfikator obiektu traci swą unikalność, zatem konieczne jest wprowadzenie identyfikatora zastępczego (identyfikator pozycji), przyjmującego unikalną wartość dla każdego ze stanów obiektu. W ogólnym przypadku relacja bitemporalna zawiera:

— identyfikator pozycji (ang. *item surrogate*),
— identyfikator obiektu (ang. *object surrogate*),

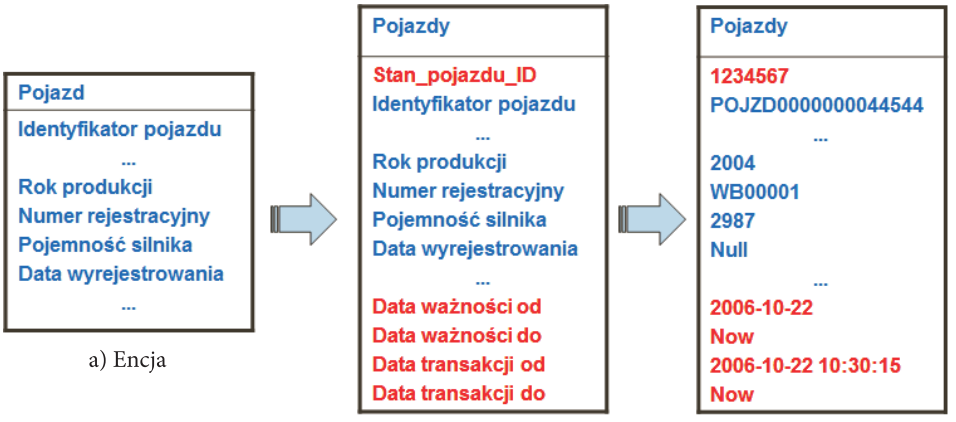
[2] Chronon — niepodzielna jednostka czasu.

[3] W dalszej części artykułu wykorzystywana jest tylko taka reprezentacja znaczników czasowych.

[4] W ogólnym przypadku w relacji bitemporalnej może występować więcej niż jeden znacznik czasu transakcji, np. jeżeli dane o stanie obiektu są przekazywane pomiędzy kolejnymi bazami wraz ze znacznikami czasu transakcji.

— znacznik czasu transakcji (ang. *transaction time-stamp*),
— znacznik czasu ważności (ang. *valid time-stamp*),
— atrybuty o wartościach stałych, niezmiennych w czasie (ang. *time-invariant attribute values*),
— atrybuty, których wartości mogą ulegać zmianom w czasie (ang. *time-varying attribute values*),
— czasy definiowane przez użytkownika (ang. *user defined times*).

Pojazdy (rys. 2b) to przykład relacji bitemporalnej. Została ona utworzona w oparciu o atrybuty obiektu *Pojazd* (rys. 2a), każdemu atrybutowi odpowiada kolumna w relacji bitemporalnej. Przedziałowy znacznik czasu ważności stanowią kolumny *Data ważności od* (początek przedziału czasu obowiązywania stanu w świecie rzeczywistym) oraz *Data ważności do* (koniec przedziału czasu obowiązywania stanu w świecie rzeczywistym), natomiast przedziałowy znacznik czasu transakcji — kolumny *Data transakcji od* (początek przedziału czasu obowiązywania stanu w bazie danych) i *Data transakcji do* (koniec przedziału czasu obowiązywania stanu w bazie danych). Przykładowy stan obiektu zapisany w tej relacji (rys. 2c) jest stanem aktualnie obowiązującym. Fakt ten określa się poprzez nadanie symbolicznej wartości *Now* końcom przedziałów czasów obowiązywania, odpowiednio, ważności i transakcji.



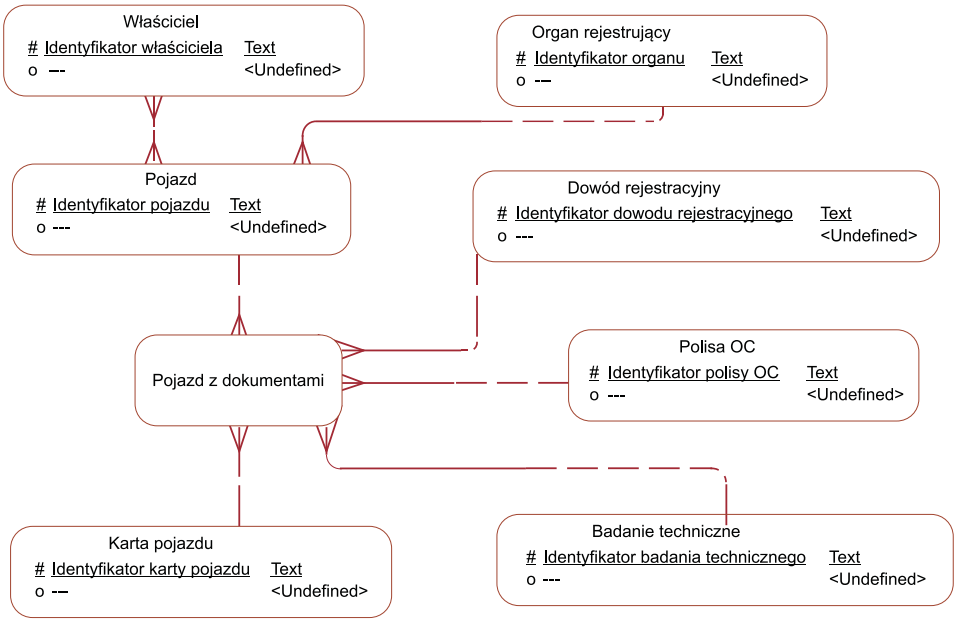
a) Encja b) Relacja bitemporalna c) Stan obiektu

Rys. 2. Atrybuty obiektu (a); relacja bitemporalna (b); stan obiektu w relacji bitemporalnej (c) — przykład

Implementacja wartości *Now* zależy od przyjętego typu danych, przy czym muszą zostać spełnione następujące warunki:
— dla czasu ważności — zaimplementowana wartość nie zostanie osiągnięta podczas całego życia obiektu,
— dla czasu transakcji — zaimplementowana wartość nie zostanie osiągnięta podczas okresu eksploatacji bazy danych.

Opisana dotychczas natura zmian stanów obiektów ma istotny wpływ na realizację powiązań pomiędzy relacjami w bitemporalnej bazie danych. Weźmy pod uwagę przykład logicznego modelu danych (rys. 3), w którym występują różne rodzaje związków pomiędzy wyszczególnionymi w nim encjami[5]:

— związek *jeden-do-wielu* pomiędzy encjami *Organ rejestrujący* i *Pojazd*,
— związek *wiele-do-wielu* pomiędzy encjami *Właściciel* i *Pojazd*,
— związek *n-arny* pomiędzy encjami *Pojazd*, *Dowód rejestracyjny*, *Polisa OC*, *Karta pojazdu* i *Badanie techniczne*.



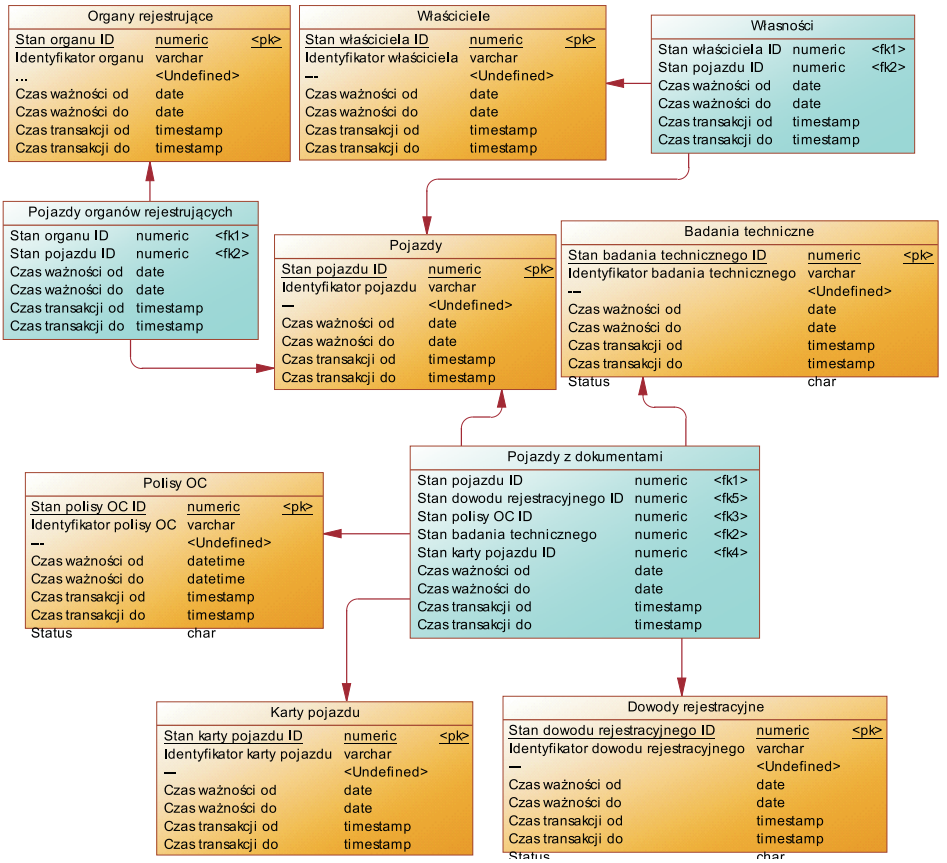
Rys. 3. Logiczny model danych uwzględniający różne związki pomiędzy encjami — przykład

W przypadku klasycznej bazy danych, na podstawie tego modelu, w prosty sposób można utworzyć model fizyczny, postępując zgodnie z ustalonymi regułami transformacji modelu logicznego na model fizyczny ([1], [6], [23]). Opracowując model fizyczny bitemporalnej bazy danych, należy pamiętać o możliwości zmiany wartości cech obiektów[6] (z przyczyn naturalnych lub poprzez korektę). Stąd też, w ogólnym przypadku, realizacjami wszystkich związków będą bitemporalne relacje

[5] Atrybuty encji, nieistotne z puntu widzenia prowadzonych rozważań, oznaczono jako „--- <Undefined>".

[6] W niniejszym artykule rozważania ograniczono do typowych sytuacji, w których wartości identyfikatorów nie ulegają zmianom.

intersekcji, z których każda musi zawierać znacznik czasu ważności, znacznik czasu transakcji oraz identyfikatory pozycji wszystkich powiązanych relacji bitemporalnych pełniących role kluczy obcych (rys. 4).



Rys. 4. Model bitemporalny utworzony dla przykładu logicznego modelu danych

Podsumowując, przy opracowaniu bitemporalnego modelu bazy danych zasadne jest postępowanie zgodnie z następującym scenariuszem:

- opracuj logiczny model danych,
- w modelu logicznym oznacz encje zawierające co najmniej jeden atrybut, którego wartości mogą się zmieniać w czasie życia obiektów reprezentowanych przez te encje,
- opracuj wstępny model fizyczny, tworząc relację bitemporalną dla każdej oznaczonej encji oraz relację zwykłą dla każdej z pozostałych encji,

— rozszerz model fizyczny o bitemporalne relacje intersekcji dla związków, w których uczestniczy co najmniej jedna oznaczona encja oraz dla związków *wiele-do-wielu*[7],
— uzupełnij klucze obce w relacjach zwykłych dla pozostałych związków.

Należy wyraźnie podkreślić, że w tak opracowanym modelu zarówno znaczniki czasu, jak i identyfikatory pozycji w relacjach bitemporalnych stanowią elementy techniczne dla operacji temporalnych i nie są dostępne dla użytkowników aplikacji wykorzystujących bitemporalną bazę danych.

5. Operacje zapisu do bitemporalnej bazy danych

Operacje zapisu do bitemporalnej bazy danych wymagają podania wartości czasów określających krańce przedziałów czasu ważności i czasu transakcji. Istnieją cztery możliwe sposoby zapisu przedziału czasu[8]:

— $[t_p, t_k]$ — przedział czasu jest domknięty od dołu i od góry,
— $[t_p, t_k)$ — przedział czasu jest domknięty od dołu i otwarty od góry,
— $(t_p, t_k]$ — przedział czasu jest otwarty od dołu i domknięty od góry,
— oraz (t_p, t_k) — gdzie przedział czasu jest obustronnie otwarty.

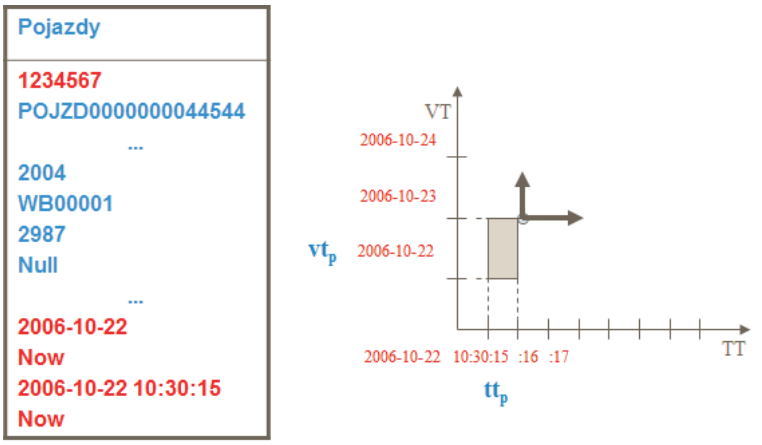
W przypadku bazy danych z pełną historią zmian przyjmuje się, że przedział czasu jest obustronnie domknięty, pozostawiając tym samym swobodę wyboru interpretacji domknięcia przedziału na poziomie zapytań. Rozwiązanie takie pozwala na elastyczną implementację zapytań, z założenia zależnych od interpretacji biznesowej. Przykładowo, w zapytaniu o właściciela pojazdu w dniu jego zbycia, w jednym przypadku może chodzić tylko o nabywcę pojazdu, ale w innym — również o zbywającego pojazd.

Kolejnym zagadnieniem, które należy rozpatrzyć w odniesieniu do operacji zapisu, jest kwestia ciągłości okresu życia obiektu, wyznaczona następującymi po sobie stanami obiektu. W przypadku baz danych z pełną historią obiekt pojawia się w bazie danych po raz pierwszy, gdy zostanie w niej utrwalony pierwszy stan obiektu (rys. 5). Początek okresu ważności obiektu określa chronon vt_p, początek czasu transakcji określa chronon tt_p. Oznacza to, że w momencie zapisu do bazy danych stan obiektu trwał, trwa lub będzie trwał przez okres odpowiadający co najmniej jednemu chrononowi dla każdego z czasów. Stan ten staje się stanem aktualnie obowiązującym dla obiektu (zwanym dalej stanem aktualnym obiektu)

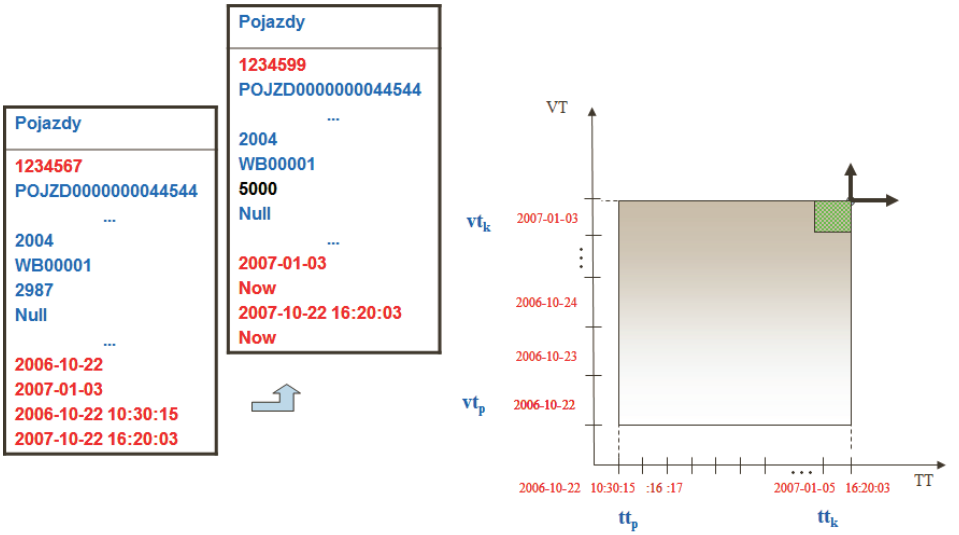
[7] Takie związki przeważnie wymagają dodatkowych atrybutów określających, od kiedy do kiedy istniał dany związek. Reprezentacja bitemporalna takich związków pozwala na ograniczenie liczby reguł opisujących operacje temporalne.

[8] A. Tansel, *Adding Time Dimension to Relational Model and Extending Relational Algebra*, IS nr 4, 1986.

i trwa do chwili zmiany wartości co najmniej jednego z atrybutów obiektu (zmiana stanu) lub zakończenia życia obiektu.

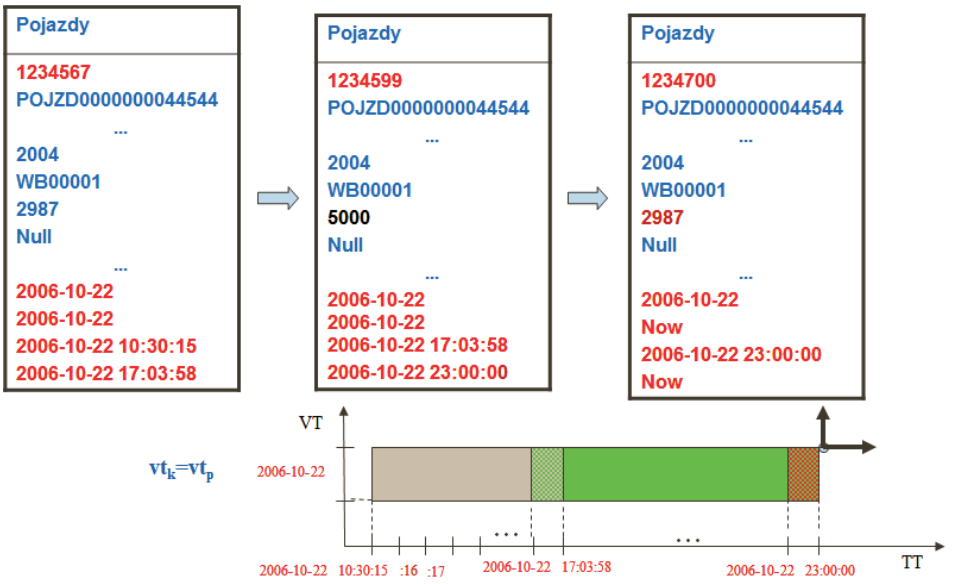


Rys. 5. Utworzenie pierwszego stanu obiektu — przykład



Rys. 6. Zmiana stanu obiektu — przykład ($vt_k > vt_p$)

Z chwilą zajścia zmiany pojawia się nowy stan obiektu, który staje się stanem aktualnym, podczas gdy stan poprzedni staje się w tym samym momencie stanem historycznym (rys. 6, rys. 7). Chronon vt_e, w którym nastąpiła zmiana ważności stanu obiektu, stanowi ograniczenie górne przedziału ważności stanu poprzedniego, ale również ograniczenie dolne przedziału ważności stanu aktualnego. Podobnie



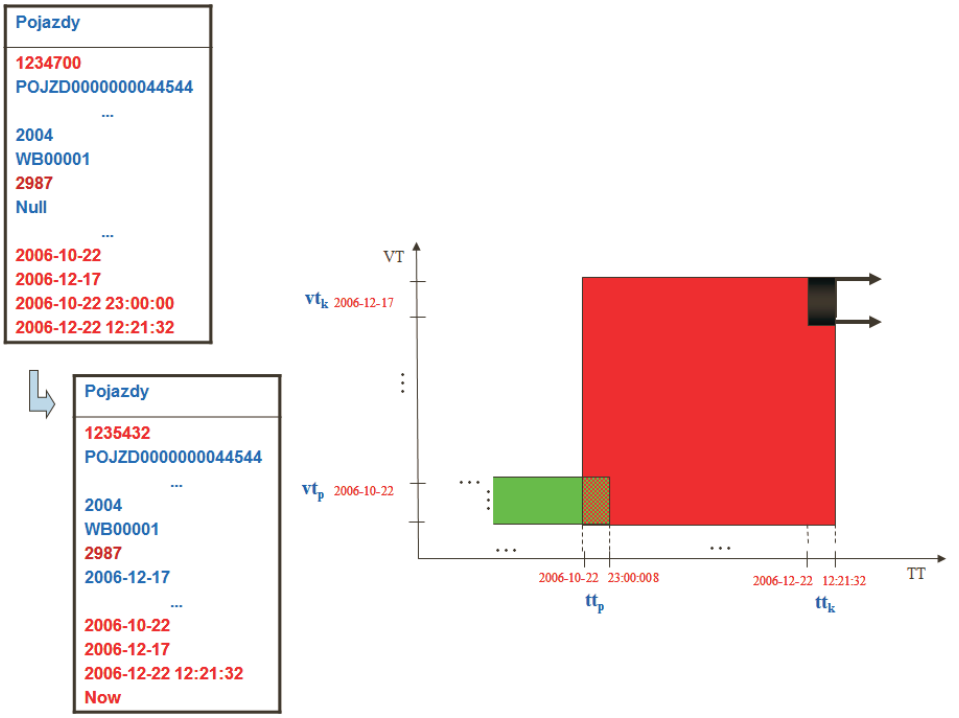
Rys. 7. Zmiana stanu obiektu — przykład ($vt_k = vt_p$)

chronon tt_e, w którym nastąpił zapis do bazy danych, stanowi ograniczenie górne przedziału transakcji stanu poprzedniego, ale również ograniczenie dolne przedziału transakcji stanu aktualnego. Należy tutaj wyraźnie podkreślić, że w przypadku zmian stanu obiektu w tym samym chrononie czasu ważności ($vt_k = vt_p$), w ramach tego chrononu wystąpi więcej niż jeden stan obiektu (rys. 7). Stosowane niekiedy przez programistów typy danych o większej dokładności niż chronon (np. *datatime* w miejsce *data*) nie zmieniają faktu, że w świecie rzeczywistym takiej dokładności nie uzyskamy. Przyjmując tego typu rozwiązania ze względów technicznych, należy mieć na uwadze, że stanowią one potencjalne źródło błędnych wyników zapytań[9].

W przypadku zakończenia życia obiektu (rys. 8) utworzenie nowego stanu jest wymuszane zdarzeniem, które nie zmienia wartości żadnego z atrybutów obiektu (np. może to być decyzja administracyjna o wyrejestrowaniu pojazdu z powodu trwałej utraty pojazdu). W nowo utworzonym stanie obiektu wartości atrybutów będą identyczne jak w stanie poprzednim, natomiast oba krańce przedziału ważności przyjmą tę samą wartość vt_k. Pozostałe zasady są analogiczne jak przy zmianie stanu obiektu.

Zakończenie życia obiektu nie oznacza usunięcia go z bazy danych. Obiekt pozostaje w niej nadal, natomiast upływ czasu dotyczy tylko czasu transakcji dla stanu, w którym obiekt zakończył życie. W ogólnym przypadku obiekt może zostać

[9] Wykorzystując w zapytaniach znaczniki czasu, należy konsekwentnie zapewniać użycie formatu odpowiadającego właściwemu chrononowi.



Rys. 8. Zakończenie życia obiektu

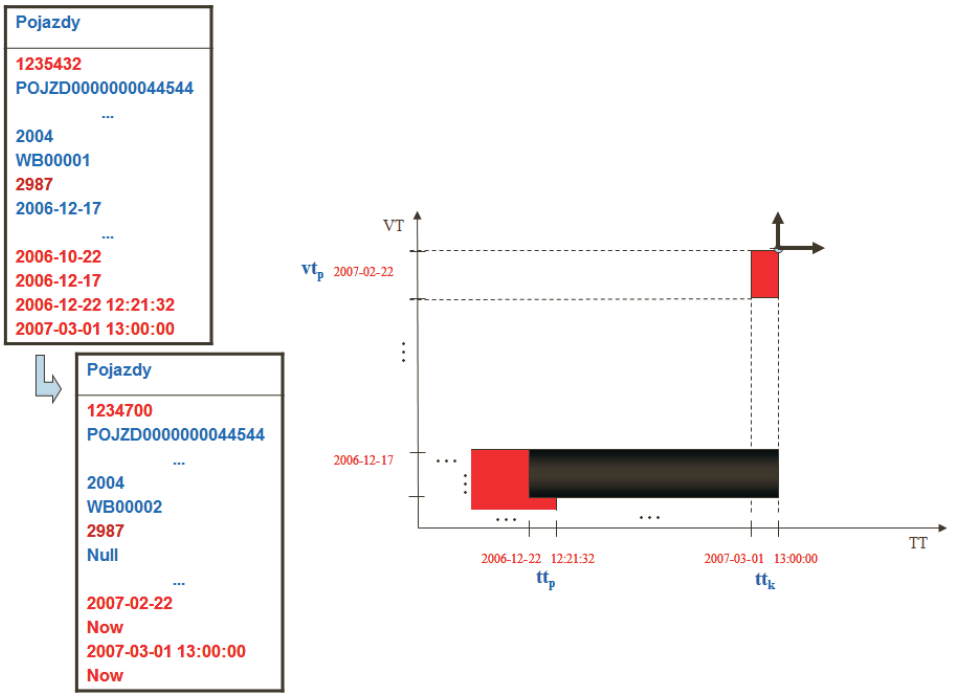
przywrócony do życia (np. ponowne zarejestrowanie odnalezionego pojazdu, który został wcześniej wyrejestrowany na podstawie stosownej decyzji administracyjnej). W nowo utworzonym stanie (rys. 9) wartości atrybutów mogą, ale nie muszą być identyczne jak w stanie poprzednim. Należy przy tym zaznaczyć, że zmianie może ulec identyfikator obiektu[10] (np. wymiana uszkodzonej karoserii w odnalezionym pojeździe skutkuje zmianą numeru VIN, czyli zmianą wartości identyfikatora pojazdu). Widoczna przerwa w życiu obiektu oznacza, że w tym czasie nie było wiadomo, co działo się z obiektem w rzeczywistości.

Opierając się na powyższych spostrzeżeniach, można opracować algorytm operacji zapisu stanu obiektu do bitemporalnej bazy danych[11]. Operacja zapisu musi uwzględniać weryfikację możliwości utworzenia nowego stanu obiektu (rys. 10)

[10] Nie jest to jedyna sytuacja, w której wartość identyfikatora obiektu może ulec zmianie. W niniejszym artykule przyjęto, że wartości identyfikatorów obiektów nie podlegają zmianom, co nie zmniejsza ogólności prowadzonych rozważań.

[11] Przyjęta notacja, bazująca na koncepcji pseudokodu i wykorzystaniu ogólnie znanych konstrukcji języków programowania, zdaniem Autora nie tylko upraszcza opis, lecz także podkreśla istotne reguły dla operacji, niezależne od implementacji algorytmu.

na podstawie podanych parametrów: identyfikatora obiektu *Id* i znacznika czasu ważności określonego przez vtp i vtk.



Rys. 9. Przywrócenie życia obiektu

```
sprawdź istnienie obiektu (Id; vtp; vtk; operacja);
  BEGIN
    IF NOT istnieje obiekt (Id) THEN
      operacja := nowy
    ELSE
      BEGIN
        pobierz znacznik czasu z aktualnego stanu obiektu (Id, znacznik);
        IF (znacznik.vtp > vtp) OR ((vtk ≠ PUSTY AND
          (znacznik.vtk > vtk) OR vtp ≠ PUSTY)) THEN RAISE błąd
        ELSE
          IF vtk ≠ PUSTY THEN operacja := zakończenie
          ELSE
            IF znacznik.vtk = VTNOW THEN operacja := dopisanie
            ELSE operacja := przywrócenie
      END
  END;
```

Rys. 10. Algorytm weryfikacji możliwości utworzenia nowego stanu obiektu

Jeżeli wynik weryfikacji jest pozytywny, określony zostaje rodzaj operacji zapisu:

- *nowy* — utworzenie pierwszego stanu obiektu,
- *dopisanie* — utworzenie kolejnego stanu obiektu,
- *zakończenie* — utworzenie stanu obiektu wskazującego na zakończenie życia obiektu,
- *przywrócenie* — utworzenie stanu obiektu wskazującego na przywrócenie obiektu do życia.

W przypadku przeciwnym operacja zapisu zostaje przerwana z powodu braku spełnienia warunków wymaganych do zachowania przyjętych reguł integralności bazy danych[12]. Algorytm zapisu stanu obiektu do bitemporalnej bazy danych prezentuje rysunek 11. Dla przejrzystości zapisu szczegóły tworzenia nowego stanu obiektu zostały przedstawione oddzielnie (rys. 12, rys. 13).

```
zapisz stan obiektu (Id; vtp; vtk; wartości atrybutów; operacja);
  BEGIN
    CASE operacja OF
      nowy:
        utwórz nowy stan obiektu (Id, vtp, wartości atrybutów);
      dopisanie:
        BEGIN
          utwórz nowy stan obiektu (Id, vtp, wartości atrybutów);
          poprzedni_akt.vtk := nowy.vtp;
          poprzedni_akt.ttk := nowy.ttp
        END;
      zakończenie:
        BEGIN
          utwórz nowy stan obiektu na podstawie stanu aktualnego (Id, PUSTY, vtk);
          poprzedni_akt.vtk := nowy.vtk;
          poprzedni_akt.ttk := nowy.ttp
        END;
      przywrócenie:
        BEGIN
          utwórz nowy stan obiektu na podstawie stanu aktualnego (Id, vtp, PUSTY);
          uaktualnij wartości atrybutów, o ile zostały zmienione;
          poprzedni_akt.ttk := nowy.ttp
        END
    END
  END;
```

Rys. 11. Algorytm operacji zapisu stanu obiektu

[12] Założenie przyjęte na tym etapie rozważań. W ogólnym przypadku zaistniała sytuacja nie musi oznaczać błędu. Może oznaczać np. uzupełnienie zdarzeń historycznych bądź korektę uprzednio wprowadzonych wartości jednego lub większej liczby stanów.

```
utwórz nowy stan obiektu (Id, vtp, wartości atrybutów);
    BEGIN
        utwórz stan (nowy);
        nowy.Id_pozycji := pobierz kolejny numer pozycji;
        nowy.Id_obiektu := Id;
        wypełnij nowy stan podanymi wartościami atrybutów;
        nowy.vtp := vtp;
        nowy.vtk := VTNOW;
        nowy.ttp := pobierz czas transakcji;
        nowy.ttk := TTNOW;
    END;
```

Rys. 12. Algorytm tworzenia nowego stanu obiektu

```
utwórz nowy stan obiektu na podstawie stanu aktualnego (Id, vtp, vtk);
    BEGIN
        utwórz stan (nowy);
        nowy.Id_pozycji := pobierz kolejny numer pozycji;
        nowy.Id_obiektu := Id;
        IF vtp = PUSTY THEN
            BEGIN
                do nowego stanu skopiuj pozostałe wartości z poprzedniego stanu aktualnego;
                nowy.vtk := vtk;
                nowy.ttp := pobierz czas transakcji;
                nowy.ttk := TTNOW;
            END
        ELSE
            BEGIN
                do nowego stanu skopiuj pozostałe wartości z poprzedniego
                stanu aktualnego lub wprowadź nowe wartości atrybutów;
                nowy.vtp := vtp;
                nowy.vtk := VTNOW;
                nowy.ttp := pobierz czas transakcji;
                nowy.ttk := TTNOW;
            END;
    END;
```

Rys. 13. Algorytm tworzenia stanu obiektu na podstawie jego stanu aktualnego

Przedstawiony algorytm wykorzystuje się również w operacjach zapisu obejmujących obiekty pozostające w związkach, przy czym w każdej z takich operacji:

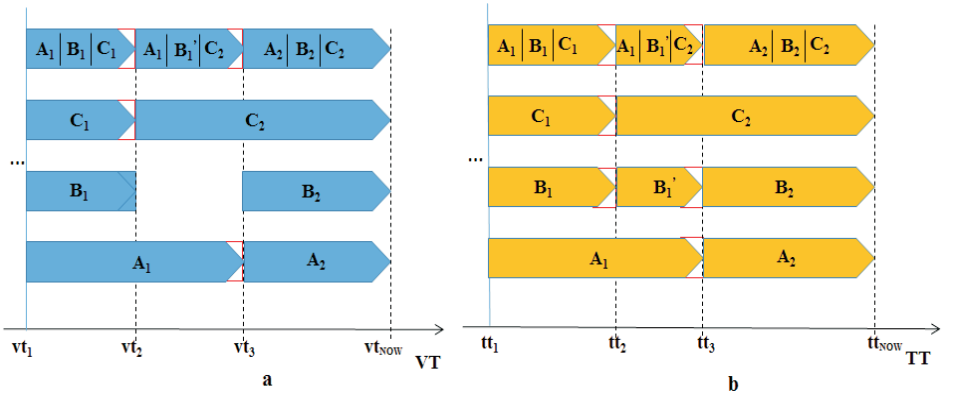
— dodatkowo tworzony jest nowy zapis do relacji intersekcyjnej określającej związki pomiędzy obiektami,
— w bazie zostają utrwalone wszystkie zapisy (nowe i zmienione stany obiektów oraz związki pomiędzy tymi stanami) albo żaden z nich.

Przed prezentacją algorytmu operacji zapisu stanów powiązanych obiektów celowe jest przybliżenie natury zjawiska w oparciu o reprezentatywny przykład.

Niech *A*, *B*, C oznaczają obiekty pozostające w związku[13], natomiast przez A_i, B_i, C_i oznaczmy *i*-te stany tych obiektów. Załóżmy, że:

- w momencie vt_1 zostały utworzone nowe obiekty *A*, *B*, *C*. Informacja o tym fakcie została utrwalona w bazie danych w momencie tt_1, tzn. zostały zapisane stany A_1, B_1, C_1 utworzonych obiektów oraz nowy stan związku pomiędzy tymi obiektami wskazujący na utworzone stany,
- w momencie vt_2 obiekt *B* zakończył życie, a obiekt *C* zmienił stan. Informacja o tym fakcie została utrwalona w bazie danych w momencie tt_2, tzn. został utworzony stan B_1' (kopia stanu B_1, w której vt_k określa moment zakończenia życia obiektu *B*), został utworzony nowy stan C_2 oraz nowy stan związku pomiędzy obiektami wskazujący na stany A_1, B_1' i C_2,
- w momencie vt_3 obiekt *B* został przywrócony do życia, a obiekt *A* zmienił swój stan. Informacja o tym fakcie została utrwalona w bazie danych w momencie tt_3, tzn. został utworzony stan B_2 (przyjęto, że część wartości została zmieniona w stosunku do stanu, w którym obiekt zakończył życie), został utworzony nowy stan A_2 oraz nowy stan związku pomiędzy obiektami wskazujący na stany A_2, B_2 i C_2.

Okresy trwania tak tworzonych stanów z punktu widzenia czasu ważności oraz czasu transakcji przedstawia rysunek 14. Można zauważyć, że w tym przypadku zmiany stanów związku pomiędzy obiektami wymuszane są przez zmianę stanu co najmniej jednego z powiązanych obiektów.

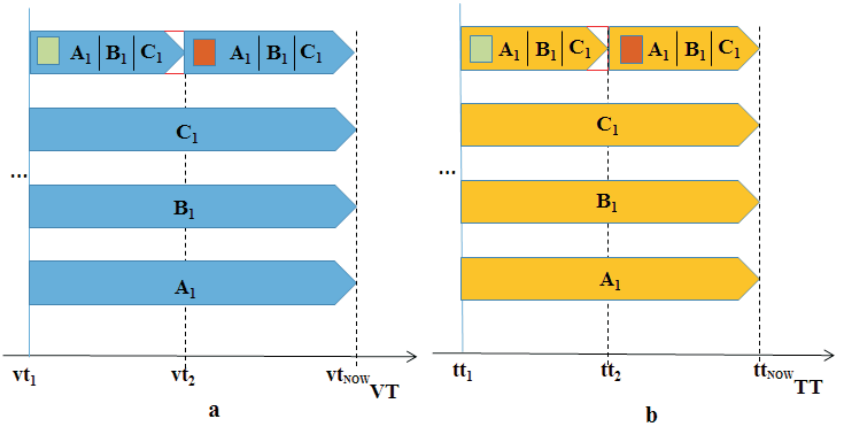


Rys. 14. Zmiany stanów powiązanych obiektów — przykład

Jeżeli związek posiada atrybuty, których wartości mogą ulegać zmianom w czasie trwania związku, każda zmiana wartości któregokolwiek z tych atrybutów spowoduje utworzenie nowego stanu związku, nawet jeżeli stany powiązanych obiektów

[13] Należy zaznaczyć, że związek (tutaj terarny) może mieć atrybuty charakteryzujące ten związek.

pozostają niezmienione (rys. 15). Należy zaznaczyć, że tak wymuszane zmiany stanu związku mogą mieć miejsce tylko w przypadku już istniejącego związku. Pierwszy stan związku pomiędzy danymi obiektami może pojawić się jedynie w wyniku utworzenia wszystkich obiektów pozostających w tym związku. Algorytm operacji zapisu stanów powiązanych obiektów (rys. 16) uwzględnia przyjęte założenia. Wykorzystano w nim wcześniej opisane operacje sprawdzenia istnienia obiektu o podanym identyfikatorze oraz zapisu jego stanu. Uszczegółowienia dodatkowych operacji tworzenia nowego stanu związku zawarte zostały na oddzielnych rysunkach (rys. 17, rys. 18).



Rys. 15. Zmiany stanu związku pomiędzy obiektami bez zmiany stanów powiązanych obiektów — przykład

```
zapisz stan powiązanych obiektów (wo; vtp; wymuszenie);
    SET TRANSACTION
        IF wymuszenie.rodzaj = obiekty THEN
            BEGIN
                WHILE nie został sprawdzony ostatni obiekt z wykazu obiektów wo DO
                    sprawdź istnienie obiektu (wo.Id; vtp; wo.vtk; operacja);
                    zapisz stan obiektu (wo.Id; vtp; wo.vtk; wo.wartości atrybutów; operacja);
                END;
                utwórz nowy stan związku (wo, vtp, wymuszenie.wartości atrybutów związku);
            END
        ELSE
            utwórz nowy stan związku na podstawie stanu aktualnego (
                                vtp, wymuszenie.wartości atrybutów związku);
    END;
```

Rys. 16. Ogólny algorytm operacji zapisu stanów powiązanych obiektów

```
utwórz nowy stan związku na podstawie stanu aktualnego (
                vtp, wartości atrybutów związku);
    BEGIN
        utwórz stan (nowy);
        nowy.Id_pozycji := pobierz kolejny numer pozycji;
        wypełnij nowy stan podanymi wartościami atrybutów związku;
        nowy.vtp := vtp;
        nowy.vtk := VTNOW;
        nowy.ttp := pobierz czas transakcji;
        nowy.ttk := TTNOW;
        poprzedni_akt.vtk := vtp;
        poprzedni_akt.ttk := nowy.ttp;
    END;
```

Rys. 17. Utworzenie nowego stanu związku wymuszonego zmianą wartości atrybutów związku

```
utwórz nowy stan związku (wo; vtp; wartości atrybutów związku);
    BEGIN
        utwórz stan (nowy);
        nowy.Id_pozycji := pobierz kolejny numer pozycji;
        nowy.vtp := vtp;
        nowy.ttp := pobierz czas transakcji;
        nowy.ttk := TTNOW;
        IF wartości atrybutów związku ? PUSTY THEN
            wypełnij nowy stan podanymi wartościami atrybutów związku
        ELSE
            IF istnieje stan poprzedni aktualny THEN
                BEGIN
                    skopiuj do nowego stanu wartości atrybutów związku ze stanu
                        poprzedniego, o ile występują;
                    skopiuj do nowego stanu wartości identyfikatorów pozycji
                        powiązanych obiektów;
                END;
        WHILE nie został sprawdzony ostatni obiekt z wykazu obiektów wo DO
            pobierz identyfikator obiektu (wo.Id, id_pozycji);
            uzupełnij nowy stan o pobrany identyfikator pozycji
        END;
        poprzedni_akt.vtk := vtp;
        poprzedni_akt.ttk := nowy.ttp;
        IF wszystkie powiązane obiekty zakończyły życie THEN
            nowy.vtk := maksymalny vtk ze stanów aktualnych powiązanych obiektów
        ELSE
            nowy.vtk := VTNOW
    END;
```

Rys. 18. Utworzenie nowego stanu związku wymuszonego zmianą stanu obiektów

6. Podsumowanie

Jednym z częstych wymagań stawianych twórcom systemów informatycznych jest konieczność zapewnienia przechowywania pełnej historii zmian stanów obiektów utrwalanych w bazie danych wytworzonego systemu. Ze względu na znikomą liczbę i ograniczone możliwości komercyjnych narzędzi wspierających wytworzenie systemu tej klasy, nie jest to zadanie trywialne. Niezbędne staje się bowiem opracowanie jednolitych zasad konstrukcji schematu bazy danych i semantyki operacji temporalnych. Ich zrozumienie przez zespół projektowy, a następnie konsekwentne stosowanie w całym cyklu budowy aplikacji, pozwoli zminimalizować wysiłek i koszty jej opracowania oraz ułatwia jej późniejszą konserwację. W artykule zostały poruszone podstawowe zagadnienia dotyczące projektowania bitemporalnych baz danych. Obejmują one wyjaśnienie przyczyn i natury zmian stanów obiektów, opisanie sposobu tworzenia relacji i związków bitemporalnych oraz algorytmy operacji zapisu do bitemporalnej bazy danych, zapewniające jej integralność. Pozostałe zagadnienia, takie jak korekty, anulowania czy też operacje odczytu wykraczają poza ramy niniejszego artykułu i będą przedmiotem dalszych prac.

Praca została sfinansowana ze środków własnych.

Artykuł wpłynął do redakcji 30.11.2015 r. Zweryfikowaną wersję po recenzjach otrzymano 2.02.2016 r.

LITERATURA

[1] Davies P.B., *Systemy baz danych*, WNT, 2003.

[2] Christy A., Gandhi G.M., *Combining bitemporal conceptual datamodel with multiway join relations for forecasting*, Procedia Computer Science, 57, 2015, 1104-1114.

[3] Cichosz M., *Lojalność klienta wobec firmy*, PWE, Marketing i rynek, 8, 2003, 8-13.

[4] Darwen H., *An Introduction to Relational Database Theory*, Frederiksberg, Denmark: Ventus, 2012.

[5] Date C.J., Darwen H., Lorentzos N.A., *Temporal Databases in the Relational Model and SQL*, Elsevier Inc., 2014.

[6] Garcia-Molina H., Ullman J.D., Widom J., *Systemy baz danych*, *Pełny wykład*, WNT, 2006.

[7] Jensen C.S., *Temporal Database Management*, IBM Press, 2005.

[8] Johnston T., *Bitemporal Data. Theory and Practice*, Elsevier Inc., 2014.

[9] Kimball R., Ross M., *The Data Warehouse Toolkit. The Definitive Guide to Dimensional Modeling*, Third Edition, John Wiley & Sons, Inc., 2013.

[10] Kulkarni K., Michels J.E., *Temporal Features in SQL:2011*. ACM SIGMOD Record 41, No. 3, September 2012.

[11] Malinowski E., Zimanyi E., *A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models*, Data Knowl. Eng., 64(1), 2008.

[12] Malinowski E., Zimanyi E., *Advanced Data Warehouse Design. From Conventional to Spatial and Temporal Applications*, Data-Centric Systems and Applications, Springer, 2008.

[13] Oracle Corporation, *Oracle Database Concepts, 11g Release 2 (11.2)*, 2015, http://docs.oracle.com/cd/E11882_01/server.112/e40540/title.htm.

[14] Saracco C., Nicola M., Gandhi L., *A matter of time: Temporal data management in DB2 10*, April 2012, http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/.

[15] Snodgrass R.T., *A Case Study of Temporal Data*, Teradata Corporation, 2010.

[16] Steiner A., *A Generalisation Approach to Temporal Data Models and their Implementations*, Departement Informatik, ETH Zürich, Szwajcaria, 1997.

[17] Steiner A., *TimeDB 2.0 Version Beta 4*, A TimeConsult Product, 1999, www.TimeConsult.com.

[18] *TimeDB*, http://www.timeconsult.com/Software/Software.html.

[19] Tang Y., Ye X., Tang N., *Temporal Information Processing Technology and Its Applications*, Tsinghua University Press, Beijing, 2011.

[20] Todman C., *Projektowanie hurtowni danych. Zarządzanie kontraktami z klientami (CRM)*, WNT, 2005.

[21] Zaniolo C., Ceri S., Faloutsos C., Snodgrass R.T., Subrahmanian V.S., Zicari R., *Advanced Database Systems*, The Morgan Kaufmann Series in Data Management Systems, 1997.

[22] Torp K., Jensen K., Snodgrass R.T., *Stratum Approaches to Temporal DBMS Implementation*, 1998.

[23] Ullman J.D., *Podstawowy wykład z systemów baz danych*, WNT, 2000.

[24] Górski T., *The use of Enterprise Service Bus to transfer large volumes of data*, Journal of Theoretical and Applied Computer Science, 8 (4), 2014, 72-81.

[25] Wieczorkowski J., *Wykorzystanie koncepcji big data w administracji publicznej*, Roczniki Kolegium Analiz Ekonomicznych, 33, 2014, 567-579.

S. ROZMUS

Database design with full history of data changes — the bitemporal database model and write operations

Abstract. Good database design practices are well-known and proven. However, when a particular base has to include also a complete history of changes introduced to the data, the project realization becomes a much more complex task. If we take into account temporal aspects, we naturally transform the relations between the interlinked objects into the relationships between the states of these objects. Additionally, the possibility of various interpretations of time dependencies prevented us so far from developing a generally accepted methodology for designing temporal databases. This article is the first in a series of articles published within the framework of the work carried out on the system of the Central Register of Vehicles and Drivers (in Polish: CEPiK). It presents the approach to the design of a relational database, which stores a complete history of the changes made to the states of these objects. It describes a database model, which explains in detail the influence of the changes made to the states of these objects, whilst also containing the life history of these objects and presents algorithms of the right operation modifying the state of the database. The presented approach has been used in practice to create one of CEPiK's databases.

Keywords: informatics, bitemporal relation, state of the object, valid time-stamp, transaction time--stamp

DOI: 10.5604/12345865.1197981