ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1999
Seria: INFORMATYKA z. 37 Nr kol. 1421

Aleksandra WERNER
Politechnika Śląska, Instytut Informatyki

ROZSZERZANIE NIETEMPORALNYCH SYSTEMÓW BAZ DANYCH O INFORMACJE CZASOWE

Streszczenie. W artykule przedstawiono przegląd zagadnień, dotyczących rejestracji i obsługi wiedzy umiejscowionej w czasie, w systemach baz danych. Główny nacisk położono na zasadność implementacji informacji czasowej w relacyjnych bazach danych. Zaprezentowano również przykłady użycia eksperymentalnych temporalnych systemów zarządzania bazami danych.

TEMPORAL DATABASES

Summary. This paper contains review of knowledge area in representing and service time-varying information, in database management systems. The main area of interests is focused on the usefulness of implementing some aspects of time information in relational database systems. This article also include examples of using experimental temporal database management systems.

1. Wprowadzenie

Czas – według encyklopedycznej definicji – jest oprócz przestrzeni jedną z podstawowych form bytu materii, co w konsekwencji powoduje, że jest on mniej lub bardziej obecny praktycznie we wszystkich dziedzinach działalności człowieka. Jest też powszechnie uznawany za jeden z istotniejszych parametrów w szeregu badań związanych z informatyką. Źródłem jednego z pierwszych bodźców współczesnych rozważań dotyczących czasu stało się w latach 60 naszego stulecia spostrzeżenie, że modalne logiki temporalne, służące do reprezentacji wiedzy związanej z czasem, mogą również służyć do formalnego opisu struktury czasu. Ogólność tego podejścia otworzyła nowe kierunki badań, mających początkowo przede wszystkim charakter filozoficzny, a od drugiej połowy lat 70

coraz bardziej skłaniający się w kierunku zastosowań informatycznych. Aktualnie prace poświęcone problematyce czasu obejmują następujące obszary badań:

- budowanie algorytmów, umożliwiających operowanie na strukturach czasowych,
- przetwarzanie języka naturalnego,
- automatyczne wnioskowanie i planowanie akcji (sztuczna inteligencja),
- temporalne bazy danych,
- systemy rozproszone i czasu rzeczywistego.

2. Struktura czasu

Nie ma zgody wśród fizyków, czy czas jest ciągły czy dyskretny, ale są oni zgodni co do tego, że czas jest ograniczony. To z kolei implikuje model dyskretny jako adekwatny do konstruowania i implementacji algorytmów, umożliwiających operowanie na (odpowiednio zareprezentowanych) strukturach czasowych w systemach informatycznych.

W terminologii matematycznej linia czasu jest izomorficzna do skończonej sekwencji liczb naturalnych, co pociąga za sobą wniosek, że czas tworzą pewne uporządkowane względem siebie obiekty, dzielące linię czasu rzeczywistego na niepodzielne segmenty równego rozmiaru. Wydaje się więc, że podstawą przedstawienia struktury czasu powinno być pojęcie odległości, niosące ze sobą ważną informację, dotyczącą położenia obiektów w czasie. Jest to zasadne podejście, chociażby ze względu na to, że w życiu codziennym również posługujemy się pojęciami odległości przedstawianymi w pewnych określonych jednostkach i tworzących pewną hierarchię (lata, sekundy, minuty, godziny itd).

Można wyodrębnić pewne charakterystyczne cechy, decydujące o pełnym dopasowaniu zaproponowanej struktury czasowej do jej rzeczywistego wzorca. Według D. McDermotta najważniejszymi własnościami, które powinna spełniać struktura czasu, jest: ciągłość czasu, „otwartość" przyszłości, rozumiana jako możliwość rozwoju podjętych akcji na wiele różnych sposobów oraz – najbardziej oczywista własność czasu – jego upływ.

W celu przedstawienia wiedzy umiejscowionej w czasie opracowano grupę specjalnych logik temporalnych, wśród których można – w zależności od budowy użytych w nich struktur czasu – wyodrębnić logiki: punktowe, punktowo-przedziałowe oraz przedziałowe.

2.1. Czas punktowy

Struktury temporalne dla czasu złożonego z punktów budowane są w oparciu o relację poprzedzania: <. Dlatego struktura czasu punktowego to para: $(T,<)$, gdzie: T – zbiór punktów czasu, < – relacja binarna na T zwana relacją poprzedzania.

Wszystkie punkty czasu są generalnie traktowane jednolicie, często jednak wśród punktów wyróżnia się daty. Data to specyfikacja czasu chronologicznie ustalona, a jej zapis jest ścisły i jednoznacznie ustalony. Przy takim podejściu potrzebna jest funkcja, która każdemu wydarzeniu przyporządkuje datę jego wystąpienia. Według K. Kahna i G. Gorry'ego założenie, że znamy dokładny czas zajścia każdego zdarzenia, jest zbyt rygorystyczne, gdyż w praktyce często znamy jedynie jego czas przybliżony.

2.2. Czas punktowo-przedziałowy

W dziedzinie reprezentowania wiedzy zależnej od czasu często można się zetknąć z operowaniem na strukturach punktowych, w których jako pojęcia pierwotne są zdefiniowane przedziały określane jako zbiory punktów (propozycja D. McDermotta) lub jako pary punktów (teoria Y. Shohama). Formuły logiczne, zdefiniowane w takiej strukturze, interpretowane są zarówno w przedziałach domkniętych (znany punkt początkowy i końcowy), jak i jednostronnie domkniętych lub otwartych.

2.3. Czas przedziałowy

Najbardziej znaną koncepcję przedziałowej struktury czasowej sformułował J.F. Allen, który za podstawę swych rozważań przyjął przedziały, powiązane ze sobą za pomocą trzynastu relacji: *przed, po, spotyka, spotkane przez, zachodzi na, przecięte przez, rozpoczyna, rozpoczęte przez, równe, kończy, kończone przez*. Relacje te określają wszystkie możliwe położenia przedziałów względem siebie na osi czasu. Istnieje wiele formalizmów zainspirowanych teorią Allena, z których najbardziej istotna jest metryzacja struktury czasu zaproponowana przez K.D. Forbusa. W swej teorii proponuje on rozważenie specjalnych funkcji, przyporządkowujących każdemu przedziałowi punkt czasu rozpoczynający bądź kończący go. Dodatkowo autor wyróżnia przedziały o zerowym czasie trwania, nazywając je *momentami*.

Reasumując, omawiana struktura czasowa to para: $(J,][)$, gdzie J – zbiór przedziałów czasu, $][$ – relacja stykania się przedziałów z J.

3. Fakty, zdarzenia, procesy

Chociaż terminologia używana w literaturze na określenie pewnych pojęć funkcjonujących w czasie jest niespójna, możemy przyjąć, że podstawowymi uniwersaliami,

reprezentującymi zmiany wiedzy o stanie świata w zależności od czasu, są: fakty, zdarzenia i procesy.

Fakt reprezentuje sobą pewien ustalony stan rzeczywistości i jest rezultatem wcześniej podjętych akcji (np. stwierdzenie: *Kowalski pracuje w firmie „ABC"* mogło być faktem spowodowanym przez akcję: *Kowalski ukończył kurs obsługi komputera*). Fakty są czasowo homogeniczne, tzn. zachodzą we wszystkich podprzedziałach swych przedziałów zachodzenia, a ich zmiany występują stosunkowo rzadko. Aby fakt przestał zachodzić, musi zajść pewne zdarzenie (np. opisem zdarzenia jest zdanie: *Kowalski został zwolniony z firmy „XYZ"*), przy czym akcja może być szczególnym, bo celowo spowodowanym przez agenta (człowieka), rodzajem zdarzenia. Każde zdarzenie powoduje rozpoczęcie kilku faktów i zakończenie innych. Na przykład zdarzenie *Kowalski został zwolniony z firmy „XYZ"* kończy fakty: *Kowalski pracuje w firmie „XYZ"*, *„Kowalski jest pracownikiem technicznym"* i rozpoczyna fakty: *Kowalski pracuje w firmie „ABC", Kowalski jest pracownikiem administracyjnym.* Mimo iż czas trwania zdarzeń jest relatywnie krótki w odniesieniu do czasu trwania faktów, upływ czasu ma miejsce wyłącznie tam, gdzie występują zdarzenia.

Między zdarzeniami zachodzi pewna korelacja czasowa: po pewnym zdarzeniu zawsze występuje inne zdarzenie. Można w związku z tym powiedzieć, że są one elementem wykonawczym, a biorąc pod uwagę informacje zawarte w bazach danych, że są one równoznaczne z poleceniem (ciągiem poleceń) zmiany zawartości bazy.

Dynamiczny obraz świata tworzą procesy, traktowane jako sekwencje drobnych zdarzeń, których skutki widoczne są jako zmiany ciągłe. W odróżnieniu od faktów procesy muszą zachodzić w większości podprzedziałów każdego przedziału swego zachodzenia (niektóre informacje są, z punktu widzenia całości procesu, nieistotne – np. w czasie trwania procesu: *Kowalski jest zwalniany z pracy.* Kierownik Kowalskiego miał ważną rozmowę ze swoim zastępcą; nie zlikwidowało to jednak identyczności procesu).

W języku naturalnym procesy wyrażane są zwykle za pomocą czasowników niedokonanych, a zdarzenia za pomocą czasowników dokonanych. Na przykład *Kierownik zwalnia pracownika* jest procesem, natomiast *Kierownik zwolnił pracownika* jest zdarzeniem.

4. Temporalne bazy danych

W bardzo wielu dziedzinach życia codziennego (inżynieria, nauka, medycyna) konieczne jest pamiętanie danych zmieniających się w czasie (ang. time varying data). W związku z tym, że analiza takich danych dostarcza więcej informacji potrzebnych do podejmowania taktycznych decyzji lub decydujących o strategii przedsiębiorstw, w ostatniej dekadzie odnotowano znaczny wzrost zainteresowania możliwością przedstawienia wiedzy

umiejscowionej w czasie w systemach baz danych. Tymczasem w rozpowszechnionych obecnie klasycznych relacyjnych bazach danych czas traktuje się w sposób uproszczony, sprowadzając go wyłącznie do atrybutu reprezentowanego obiektu, a nie traktując jako naturalny wymiar rzeczywistości.

W tradycyjnych Systemach Zarządzania Bazami Danych (ang. DBMS) najczęściej do dyspozycji programisty są:

- typy danych umożliwiające reprezentowanie wartości opisujących czas (*DATE, TIME*),
- zbiory podstawowych funkcji operujących na wartościach tego typu danych, realizujących porównywanie czy obliczanie dat.

Wymienione możliwości okazują się jednak być niewystarczającymi na potrzeby dzisiejszych użytkowników baz danych:

- niejasna jest kwestia interpretacji zapisu wartości atrybutów czasowych. Zapis ten traktuje się z reguły jako moment, w którym dana wielkość powstała w realnym świecie (np. data produkcji wyrobu, data ustalenia ceny, data dostawy towaru), choć równie dobrze może to być moment wpisania do bazy lub moment, w którym towar znalazł się w magazynie,
- konieczne jest przyjęcie założenia, że wartości z krotki, dotyczącej danego obiektu, nie podlegały żadnej (nawet chwilowej) zmianie (istnienie nieprawidłowych danych w bazie, pociągających za sobą podjęcie błędnych działań może być trudne do wykrycia, jeżeli dane te uległy późniejszej modyfikacji),
- utrudniona jest integracja systemów (aplikacji) ze względu na różne możliwości reprezentacji czasu przez projektantów konstruujących własne procedury zapisu/odczytu danych historycznych,
- brak możliwości zapytań o dynamikę zjawisk.

Aby uniknąć ograniczenia, programiści stosują nienaturalne rozwiązania, tworząc przykładowo kopie archiwalne relacji (np. oznaczające sprzedaż w miesiącu *X*).

W chwili obecnej brak jest na rynku jakiegokolwiek komercyjnego systemu zarządzania temporalną bazą danych (ang. Temporal Database Management System - TDBMS). Wiąże się to przede wszystkim z różnicami pomiędzy poszczególnymi danymi zmieniającymi się w czasie oraz z faktem stosowania w systemach informatycznych różnych wariantów reprezentacji czasu. Niemniej jednak zostały podjęte pewne działania, koordynowane przez naukowców tworzących grupę *TimeCenter*, które zaowocowały powstaniem różnorodnych niekomercyjnych, eksperymentalnych systemów TDBMS, takich jak: Postgres95, TimeIt, TimeDB i wiele innych, jak również przyczyniły się do opracowania nowych standardów języka SQL (np. TSQL2, SQL+i, Transac SQL).

4.1. Prosty przykład, ilustrujący trudności uwzględnienia czasu w modelu konceptualnym

W rzeczywistym systemie możemy się spodziewać znacznie bardziej skomplikowanych struktur niż pokazane poprzednio. Oto przykład bazy danych dla magazynu artykułów spożywczych (uproszczony do celów dydaktycznych; nie bierze się pod uwagę np. zwrotów, korekt i innych dokumentów, które mogą wystąpić w rzeczywistości).

Baza danych składa się z tabel:

TOWARY (identyfikator towaru, nazwa towaru, jednostka miary)

STANY_MAGAZYNOWE (identyfikator_towaru, ilość, cena, data ważności)

DOKUMENTY_POZYCJE (identyfikator_nagłówka, identyfikator towaru, data ważności, cena, ilość)

DOKUMENTY_NAGŁÓWKI (identyfikator_nagłówka, nr_dokumentu, przyjęcie/wydanie_towaru, data_dokumentu, kontrahent)

Jeśli do magazynu przywożony jest towar, wówczas do systemu wprowadzany jest dokument (jest to równoznaczne z uzupełnieniem relacji: *DOKUMENTY_NAGŁÓWKI* i *DOKUMENTY_POZYCJE*), który jest następnie księgowany, co powoduje, że do stanu magazynowego dodawane są kolejne pozycje z dokumentu (wypełniana jest relacja *STANY_MAGAZYNOWE*). W chwili sprzedaży do odpowiedniej tabeli wprowadzany jest dokument, ewidencjonujący wydanie towaru konkretnemu klientowi. Dokument jest księgowany, co oznacza odjęcie towaru ze stanu magazynu.

Wykorzystując bazy temporalne, można zadać następujące zapytania:

Jaki był zapas magazynowy na dany dzień dla danego asortymentu?

Jaka była wartość magazynu na konkretny dzień (ilość * cena)?

Uzyskanie odpowiedzi na powyższe pytania w relacyjnej bazie danych byłoby bardzo czasochłonne.

Poczyńmy teraz dodatkowe założenie, że towary dzielą się na grupy asortymentowe. Potrzebna jest dodatkowa tabela:

GRUPY(identyfikator_grupy, nazwa_grupy)

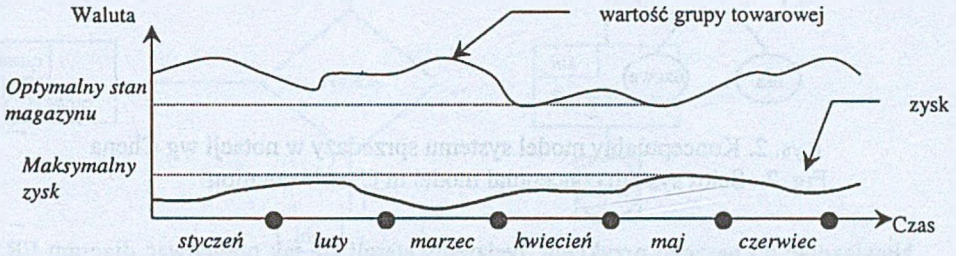
oraz dodanie atrybutu *identyfikator_grupy* do tabeli TOWARY.

Podstawowym pytaniem stawianym sobie przez kierownika magazynu jest: *Jak ma się kształtować struktura i liczba zamawianego towaru w ramach grupy, żeby osiągnąć największy zysk*[1]*?*

[1] Rozumiany jako różnica między wartością sprzedaży a wartością zakupu, z uwzględnieniem strat oraz kosztów zamrożenia kapitału (inflacja).

Możliwe są dwa skrajne podejścia, przy czym oba mają zarówno pewne wady, jak i zalety. Jeżeli oferta będzie uboga (wydana zostanie stosunkowo niewielka ilość gotówki), to klienci mogą się zniechęcić (nie wszystkie towary z poszczególnych grup będą dostępne na bieżąco), a w rezultacie obniżyć zysk; z kolei różnorodność i wzrost liczby zamawianego asortymentu może zwiększyć zyski firmy, ale tylko pod warunkiem, że sprzedaż wzrośnie w równym stopniu jak zaangażowany kapitał. W przeciwnym przypadku zysk będzie się obniżał.

Dla każdej grupy towarowej można by wygenerować wykres, przedstawiający (z uwzględnieniem inflacji) wartość towaru w magazynie wraz z odpowiadającym mu zyskiem (rys. 1).



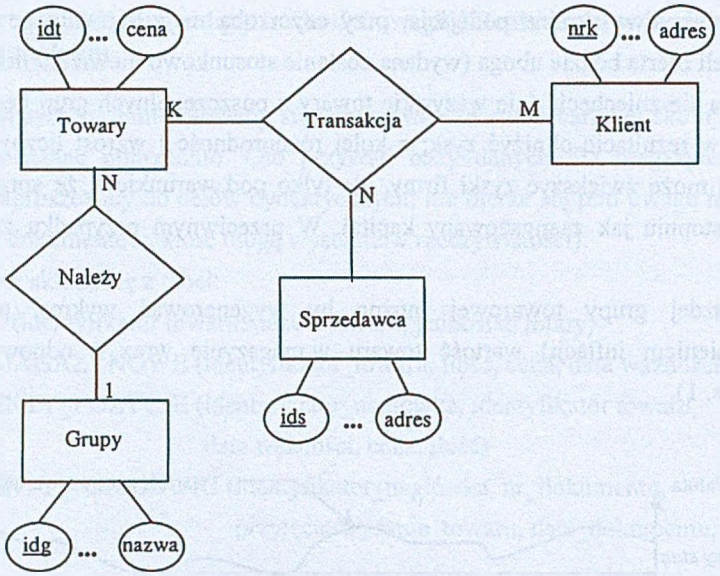
Rys. 1. Czasowe zmiany wartości zysku w odniesieniu do zmian wartości grupy towarowej
Fig. 1. Time-varying value of profit with reference to article's value changing

Analiza takiego wykresu, jak przedstawiony na rys. 1, pozwoliłaby kierownikowi zoptymalizować ilość zapasów magazynowych w ramach poszczególnych grup asortymentowych.

Niestety, model konceptualny omawianej rzeczywistości nie zawiera żadnych informacji o możliwości zmiany asortymentu w czasie sprzedaży (rys. 2). Tracona jest również możliwość analizy wpływu poprzednich działań na obecne. Niezbędne staje się zatem wprowadzenie dodatkowych konstrukcji, ujmujących w sposób jawny czasowy wymiar zagadnienia.

Ze względu na dużą popularność i łatwość stosowania diagramów ER do projektowania schematu relacyjnej bazy danych oraz analizy opisywanej rzeczywistości, opracowano wiele modeli związków encji zdolnych do utrwalenia informacji czasowej.

Jednym z nich jest model RAKE (ang. Relationships, Attributes, Keys and Entity) najbardziej, biorąc pod uwagę stosowaną w nim konwencję, zbliżony do klasycznego modelu ER. Dodatkowe komponenty diagramu RAKE to: *ramka kluczy* w lewym górnym rogu encji z wyspecyfikowanym w niej kluczem głównym oraz *okrąg wpisany w kwadrat*, reprezentujący typ encji, na który składa się zakres wartości atrybutu.



Rys. 2. Konceptualny model systemu sprzedaży w notacji wg Chena
Fig. 2. Sales system conceptual model in Chen's notation

Nawiązując do naszego przykładu, będziemy starali się tak opracować diagram ER, aby uchwycić czasowe aspekty zarówno transakcji *zamówień-kupna-sprzedaży*, zachodzących między obiektami: *Towary - Sprzedawca – Klient*, jak i samych obiektów.

Zastosowanie temporalnego typu związku (ang. Time-varying Relationship Type) pozwoli na wyraźne zakotwiczenie w czasie faktu np. kupna przez klienta określonego towaru, natomiast określenie temporalnych atrybutów obiektów (ang. Time-varying attributes) precyzyjnie wskaże, które parametry obiektów podlegają czasowym zmianom (np. mogą pojawić się różnice w cenie konkretnego towaru, wynikające z przejściowego wzrostu kosztów transportu) (rys. 3).

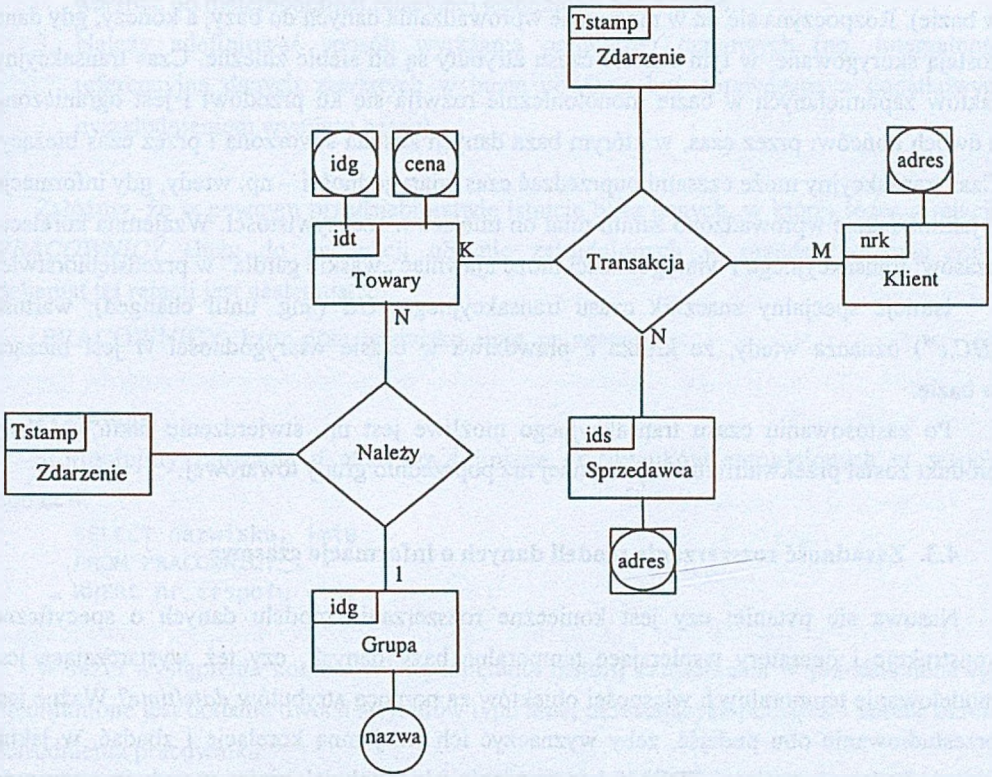
4.2. Warianty reprezentacji czasu

Jak – projektując bazę danych dla magazynu artykułów spożywczych – zakotwiczyć w czasie transakcje *zamówień/sprzedaży* między obiektami: *klient – sprzedawca*, a zarazem umożliwić wykonywanie zapytań o historię samych obiektów? Są trzy możliwości:

4.2.1. Czas zdefiniowany przez użytkownika (ang. user defined time)

Czas ten jest jednym z atrybutów bazy danych; pozostałe atrybuty nie są od niego zależne. Przykładem czasu definiowanego przez użytkownika może być: data urodzenia, data

produkcji wyrobu czy, nawiązując do przedstawionego w punkcie 4.1 przykładu, data przyjścia faktury, czyli czas definiowany indywidualnie w zależności od projektu.



Rys. 3. Diagram RAKE dla rozważanego przykładu
Fig. 3. The running example modeled using RAKE diagram

4.2.2. Czas wiarygodności (ang. valid time)

Jest to czas, w którym wartość danego atrybutu – według naszej najlepszej wiedzy – występowała w rzeczywistości. Dziedziną tego czasu jest okres od początku do końca okresu trwania modelowanej rzeczywistości. Atrybuty są funkcjonalnie czasowo od siebie zależne. Przykładem ilustrującym ten typ czasu może być fakt: *„Anna mieszkała w Gliwicach od 1 kwietnia 1995 do 13 marca 1996"*. Ten czas mógłby również zostać użyty do reprezentacji atrybutu *data_ważności* wyrobu z tabeli *STANY_MAGAZYNOWE*.

Dla czasu wiarygodności wprowadza się zwykle dwie specyficzne wartości: *begining* oraz *forever* (w dosłownym tłumaczeniu: *do nadal*), oznaczające odpowiednio: najmniejszą i największą dozwoloną wartość.

4.2.3. Czas transakcyjny (ang. transaction time)

Czas transakcyjny jest to czas, w którym dane są prawdziwe w bazie (zapamiętywane w bazie). Rozpoczyna się on w momencie wprowadzania danych do bazy, a kończy, gdy dane zostają skorygowane; w tym okresie czasu atrybuty są od siebie zależne. Czas transakcyjny faktów zapamiętanych w bazie monotonicznie rozwija się ku przodowi i jest ograniczony z dwóch końców: przez czas, w którym baza danych została stworzona i przez czas bieżący. Czas transakcyjny może czasami poprzedzać czas wiarygodności – np. wtedy, gdy informację o jakimś fakcie wprowadzono zanim miał on miejsce w rzeczywistości. Wzajemna korelacja czasów: transakcyjnego i wiarygodności może ujawniać „wąskie gardła" w przedsiębiorstwie.

Istnieje specjalny znacznik czasu transakcyjnego: *UC* (ang. until changed); wartość (UC,c^{vt}) oznacza wtedy, że krotka *c* prawdziwa w czasie wiarygodności *vt* jest bieżąca w bazie.

Po zastosowaniu czasu transakcyjnego możliwe jest np. stwierdzenie faktu, że dany produkt został przekwalifikowany do innej niż poprzednio grupy towarowej.

4.3. Zasadność rozszerzania modeli danych o informacje czasowe

Nasuwa się pytanie: czy jest konieczne rozszerzanie modelu danych o specyficzne konstrukcje i operatory wspierające temporalne bazy danych, czy też wystarczające jest modelowanie temporalnych własności obiektów za pomocą atrybutów *date/time*? Ważne jest przestudiowanie obu podejść, żeby wyznaczyć ich wzajemną korelację i zbadać, w jakim stopniu będą one wspierać TDBMS bez czynienia jakichkolwiek zmian w wykorzystywanym systemie i modelu danych.

Żeby zaimplementować temporalną aplikację, nietemporalne systemy baz danych należy rozszerzyć w trzech kierunkach:

1. Struktury danych muszą być przystosowane do rejestracji informacji czasowej.

   To wymaganie nie stwarza obecnie żadnych problemów. Jeżeli dane będą czasowo znacznikowane czasami: wiarygodności i transakcyjnym, struktura danych powinna być rozszerzona o dwa dodatkowe atrybuty typu *date*, odnotowujące punkt początkowy i końcowy przedziału czasu (transakcyjnego lub wiarygodności), w którym dane zawarte w bazie są prawdziwe. Ważne jest poczynienie założeń, która część struktury danych powinna być znacznikowana czasem: krotka czy atrybut?
2. Muszą zostać udostępnione nowe operatory, działające na dodatkowej temporalnej semantyce danych, pozwalające na modyfikację i wyszukiwanie danych temporalnych.

To żądanie wydaje się być problematyczne, gdyż operacje algebry temporalnej muszą być zaimplementowane albo bezpośrednio w systemie, albo jako dodatkowa warstwa do nietemporalnej relacyjnej technologii baz danych.

3. Należy zdefiniować sposób wyrażania ograniczeń czasowych (np. integralność referencyjna danych zawartych w bazie powinna być sprawdzana z dodatkowym uwzględnieniem wymiaru czasu).

Załóżmy, że w pewnym przedsiębiorstwie istnieje baza danych, w której jedna z relacji: *PRACOWNICY* służy do ewidencji obecnie zatrudnionych w przedsiębiorstwie osób. Schemat tej relacji jest następujący:

PRACOWNICY_1 (nr_prac, nazwisko, imię, nr_zespołu) (1)

Przykład 1

Sformułujmy zapytanie o nazwiska i imiona pracowników zatrudnionych w zespole numer 4:

```
SELECT nazwisko, imię
FROM PRACOWNICY_1
WHERE nr_zespołu = 4;
```

W razie wystąpienia konieczności pamiętania historii zatrudnienia w przedsiębiorstwie, nieuniknione jest dodanie dwóch atrybutów typu *date*, określających początek i koniec okresu zatrudnienia pracownika:

PRACOWNICY_2 (nr_prac, nazwisko, imię, nr_zespołu, pocz_zatr, koniec_zatr) (2)

Dla relacji, o podanym wyżej schemacie (2), można teraz zadać (np. w języku PL/SQL systemu Oracle) zapytanie o nazwiska/imiona pracowników zatrudnionych w zespole numer 4 np. w ciągu ostatnich 2 lat. W związku z tym, że w relacji temporalnej możliwe jest pamiętanie faktów (zdarzeń), które dopiero *wystąpią* w świecie rzeczywistym (np. pozytywne rozpatrzenie */w czerwcu/* podania o angaż nowego nauczyciela biologii *od września br.*), konieczne jest ograniczenie zbioru wszystkich rekordów do tych, których atrybut *pocz_zatr* jest mniejszy bądź równy *data_bieżąca*:

```
SELECT nazwisko, imię
FROM PRACOWNICY_2
WHERE nr_zespołu = 4 and pocz_zatr <= SYSDATE and
      SYSDATE <= koniec_zatr and TO_NUMBER(TO_CHAR(pocz_zatr,'yy'))>=
      (TO_NUMBER(TO_CHAR(SYSDATE,'yy'))-2);
```

Decydując się na użycie języka SQL/Temporal, nasze zapytanie uległoby znacznemu uproszczeniu i byłoby postaci:

```
VALIDTIME PERIOD '[data_bieżąca - 1996-10-27]'
SELECT nazwisko, imię
FROM PRACOWNICY_1
WHERE nr_zespołu = 4;
```

Analogiczne zapytanie w języku PSQL (język zapytań eksperymentalnego systemu Postgres95) miałoby formę:

```
SELECT nazwisko, imię
FROM PRACOWNICY_2 ['data_bieżąca','October 27,1996']
WHERE nr_zespołu = 4;
```

Przykład 2

Prześledźmy różnice w zapytaniach agregujących, dotyczących relacji o podanych wyżej schematach, na przykładzie zapytania o liczbę pracowników zatrudnionych obecnie w przedsiębiorstwie.

Dla relacji o schemacie (1):

```
SELECT COUNT(*)
FROM PRACOWNICY_1;
```

Dla relacji o schemacie (2) identyczne zapytanie przybiera bardziej skomplikowaną formę:

```
SELECT COUNT(*)
FROM PRACOWNICY_2
WHERE TO_CHAR(koniec_zatr,'dd-mon-yy') < TO_CHAR(SYSDATE,'dd-mon-yy')
      and TO_CHAR(pocz_zatr,'dd-mon-yy') <=
          TO_CHAR(SYSDATE,'dd-mon-yy');
```

Posługując się językiem SQL/Temporal, zapytanie o liczbę pracowników w przedsiębiorstwie należałoby sformułować następująco:

```
VALIDTIME SELECT COUNT(*)
FROM PRACOWNICY_1;
```

Najprostszą formę zapytania o stan obecny zatrudnienia w przedsiębiorstwie oferuje nam jednak język PSQL systemu Postgres95:

```
SELECT COUNT(*)
FROM PRACOWNICY_1;
```

Przykład 3

Niech w bazie danych rozważanego przez nas przedsiębiorstwa istnieje dodatkowo relacja: *PŁACE* zawierająca informacje o aktualnych zarobkach poszczególnych

pracowników przedsiębiorstwa, zajmujących pewne stanowiska. Schemat relacji *PŁACE* jest następujący:

PŁACE_1 (nrp, pensja, stanowisko) (3)

Znalezienie aktualnej pensji pracownika o numerze 21 w tabeli *PŁACE* jest równoznaczne z zadaniem prostego zapytania w języku SQL:

```
SELECT pensja
FROM PŁACE_1
WHERE nrp = 21;
```

Po pewnym czasie, chcąc zapamiętać wszystkie zmiany pensji pracowników wynikające np. z ich kolejnych awansów na wyższe stanowiska, schemat relacji *PŁACE* zmieniono, dodając dwa dodatkowe atrybuty typu *date*: atrybut *start* – pamiętający dzień, w którym pracownik zaczął otrzymywać daną pensję oraz atrybut *stop* – pamiętający dzień, do którego dana pensja pozostaje aktualna.

Zapytanie o aktualną pensję pracownika o numerze 21 przyjmie teraz postać:

```
SELECT pensja
FROM PŁACE_2
WHERE    nrp = 21
         and TO_CHAR(start,'dd-mon-yy')<=TO_CHAR(SYSDATE,'dd-mon-yy')
         and TO_CHAR(stop,'dd-mon-yy')>=TO_CHAR(SYSDATE,'dd-mon-yy');
```

Przykładowe wypełnienie tabeli *PŁACE_2* przedstawiono w tabeli 1.

Tabela 1

Przykładowe dane zapisane w tabeli *PŁACE_2*

Nrp	Stanowisko	pensja	start	Stop
21	Stażysta	1100	01.01.1994	30.05.1995
21	Asystent	1100	31.05.1995	06.04.1996
21	Doktor	2000	07.04.1996	30.10.1998
30	Doktor	2000	12.02.1993	06.12.1997
30	Profesor	3000	07.12.1997	20.12.1998

Jeżeli zaszłaby konieczność pokazania pracownikowi o numerze 21 historii jego zarobków, wtedy zapytanie:

```
SELECT pensja, start, stop
FROM PŁACE_2
WHERE nrp = 21;
```

zwróciłoby informacje w formacie przedstawionym w tabeli 2.

Tabela 2
Wynik zapytania o historię zarobków

pensja	Start	stop
1100	01.01.1994	30.05.1995
1100	31.05.1995	06.04.1996
2000	07.04.1996	30.10.1998

Zmianie uległ niewidoczny w wyniku atrybut: *stanowisko*

Tymczasem intuicyjny wynik jest taki, jak zaprezentowano w tabeli 3.

Tabela 3
Spodziewany rezultat zapytania

pensja	Start	stop
1100	01.01.1994	06.04.1996
2000	07.04.1996	30.10.1998

Jak zrealizować to żądanie?

Wariant 1:

Zreorganizować schemat relacji *PŁACE*, dzieląc go na dwie odrębne relacje (4) (5):

PŁACE_3 (nrp, pensja, start, stop) (4)
TYTUŁ (nrp, stanowisko, start, stop) (5)

Powtórzenie atrybutów: *start, stop* jest konieczne, gdyż zmiana pensji może być niezależna od zmiany stanowiska

Wariant 2

Posługując się językiem SQL, stworzyć tabelę tymczasową, zawierającą identyczne informacje jak te zawarte w tabeli *PŁACE_2*, a następnie (w pętli) znaleźć te okresy czasu, które się nakładają lub spotykają ze sobą, uaktualnić atrybut *stop*, usuwając w następnym kroku niepotrzebne już (bo scalone z innymi) wiersze tabeli.

W systemie ORACLE (w języku PL/SQL) realizacja dwóch pierwszych żądań wyglądałaby następująco:

```
CREATE TABLE TEMP (pensja, start, stop)
AS SELECT pensja, start, stop
FROM PŁACE_2
WHERE nrp = 21;
UPDATE TEMP T
SET (T.stop) =
  (SELECT MAX(TT.stop)
   FROM TEMP TT
```

```
WHERE T.pensja = TT.pensja
      AND TO_CHAR(T.start) < TO_CHAR(TT.start)
      AND TO_CHAR(T.stop) >= TO_CHAR(TT.start)
      AND TO_CHAR(T.stop) < TO_CHAR(TT.stop)
WHERE EXISTS (SELECT *
              FROM TEMP TT
              WHERE T.pensja = TT.pensja
                    AND TO_CHAR(T.start) < TO_CHAR(TT.start)
                    AND TO_CHAR(T.stop) >= TO_CHAR(TT.start)
                    AND TO_CHAR(T.stop) < TO_CHAR(TT.stop);
```

Empirycznie sprawdzono, że maksymalny koszt scalenia ze sobą N wierszy tabeli wynosi: $\log_2 N$.

Wariant 3

Posługując się językiem TSQL2 zadać zapytanie[1]:

```
SELECT pensja
FROM PŁACE_2;
```

4.4. Idea bazy bitemporalnej

W większości przypadków wymagane jest, aby baza danych wspierała zarówno czas transakcyjny, jak i wiarygodności. Takie bazy bywają nazywane bazami bitemporalnymi.

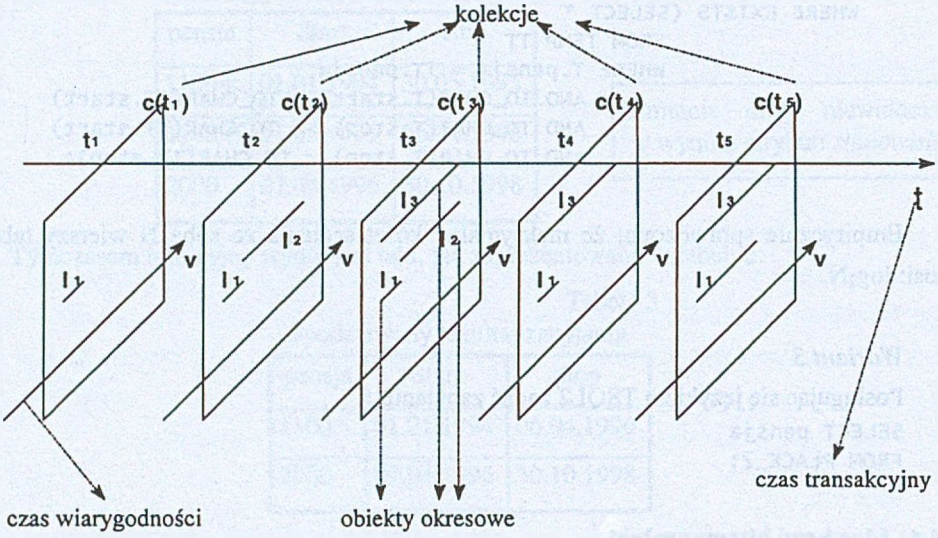
Idea bazy bitemporalnej może być zobrazowana jako zachowywanie (poprzez wsparcie czasu transakcyjnego) zmian dynamicznych kolekcji obiektów okresowych (określonych przez czas wiarygodności). Rysunek 4 przedstawia konceptualną perspektywę bazy bitemporalnej. Zamiast pojedynczej kolekcji obiektów przedstawiona jest sekwencja kolekcji obiektów, indeksowana przez czas transakcyjny.

Załóżmy, że każdy obiekt okresowy reprezentuje jakiś kontrakt w przedsiębiorstwie. W modelu bitemporalnym możemy więc zareprezentować, jak zmieniała się nasza wiedza o tych kontraktach.

Każdy obiekt okresowy jest wprowadzany do bazy w czasie transakcyjnym *t*. Zapis wiedzy o obiekcie do bazy danych obejmuje: surogat obiektu (unikalny identyfikator obiektu, przy czym wartość tego identyfikatora nie jest widziana przez użytkownika), np.

[1] Problem łączenia (ang. coalescing) tych wierszy, których czasy wiarygodności nakładają się lub zachodzą na siebie, jest w większości TDBMS rozwiązany, zwykle przez jawne dostarczenie użytkownikowi systemu specjalnych konstrukcji językowych, umożliwiających włączanie/wyłączanie scalania odpowiednich wierszy.

id_kontraktu, atrybut np. *kwota_kontraktu*, okres wiarygodności (czyli czas trwania kontraktu) oraz początkowy okres czasu transakcyjnego *[t,now)*[1].



Rys. 4. Konceptualny model bazy bitemporalnej
Fig. 4. Conceptual model of bitemporal database

Przykładowo: zapis dla obiektu *I2* ma czas transakcyjny *[t2, t4)*, ponieważ obiekt *I2* był wprowadzony do bazy w chwili *t2*, a w chwili *t4* został skasowany (może to oznaczać, że kontrakt był źle wprowadzony).

4.5. Modele danych temporalnych

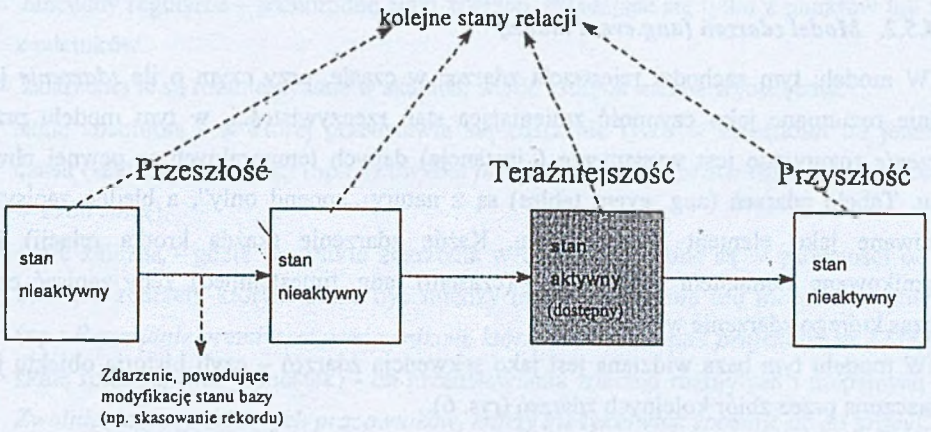
W literaturze przedmiotu spotyka się trzy zasadnicze modele, różniące się między sobą sposobem ujęcia czasu.

4.5.1. Model „migawka" (ang. snapshot model) – rys. 5

W tym modelu reprezentowane są dane z pewnej ściśle określonej chwili. Migawka utrwala stan danych (obecny, przeszły lub przyszły) w określonej chwili czasu, przy czym migawka z jednej chwili czasu może być różna od migawki wziętej z innej chwili.

Model relacyjny (w ujęciu klasycznym) używa migawki do reprezentacji bieżącego stanu danych. Możliwe jest więc zadanie pytania migawkowego (ang. snapshot query).

[1] Punkt końcowy okresu czasu transakcyjnego *now* zostanie zmieniony w chwili aktualizacji obiektu.



Rys. 5. Graficzna ilustracja modelu „migawka"
Fig. 5. Snapshot model graphic image

Przykładowe wypełnienie tabeli typu migawka przedstawia tabela 4.

Tabela 4

Hipotetyczne dane relacji PŁACE_1

nrp	stanowisko	pensja
21	Asystent	1100
30	Doktor	2000

W takiej tabeli historia nie jest zapamiętywana, a zmiany i uaktualnienia są destrukcyjne. Relacje o tej strukturze pozwalają uzyskać odpowiedź na jeden typ pytania: *Jaka informacja o danym atrybucie została aktualnie wpisana do bazy?*

Zakładając, że korzystamy z relacji o schemacie (3), posiadającej dwa rekordy, dotyczące płac pracowników o numerach 21 i 30 (tabela 4), wykonanie jakichkolwiek zmian w tej relacji (np. poleceniami *insert* czy *delete*) spowoduje zmianę zawartości relacji PŁACE_1 (tabela 5).

```
delete from PŁACE_1 where nrp = 30;
insert into PŁACE_1 values (30,profesor,00);
```

Tabela 5

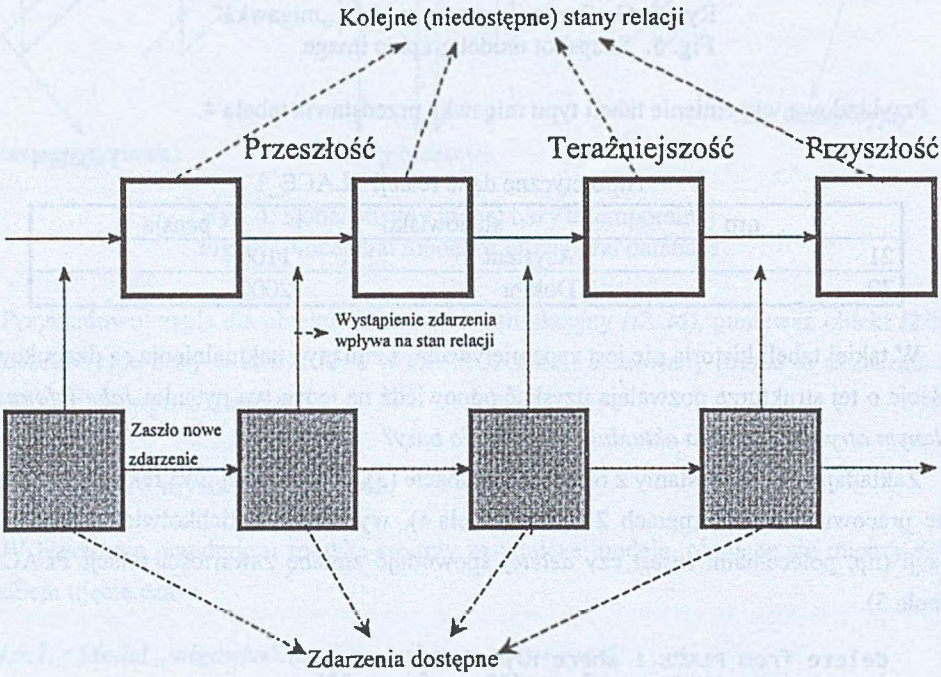
Zmiana danych w tabeli PŁACE_1 po wykonaniu poleceń *delete/insert*

nazwisko	wydział	pensja
21	Asystent	1100
30	Profesor	3000

4.5.2. Model zdarzeń (ang.event model)

W modelu tym zachodzi rejestracja zdarzeń w czasie, przy czym o ile *zdarzenie* jest ogólnie rozumiane jako czynność zmieniająca stan rzeczywistości, w tym modelu przez *zdarzenie* rozumiane jest wystąpienie (instancja) danych temporalnych w pewnej chwili czasu. Tabele zdarzeń (ang. event tables) są z natury „append only", a błędne zapisy są traktowane jako element historii opisu. Każde zdarzenie (każda krotka relacji) jest znacznikowane elementem temporalnym (czasem) (ang. timestamped), żeby zapisać czas, podczas którego zdarzenie wystąpiło.

W modelu tym baza widziana jest jako sekwencja zdarzeń – czyli historia obiektu jest wyznaczona przez zbiór kolejnych zdarzeń (rys. 6).



Rys. 6. Graficzna ilustracja modelu zdarzeń
Fig. 6. Event model graphic image

Wyróżnia się trzy podstawowe temporalne typy zdarzeń:

- punkt - w zależności od zastosowanej skali czasu może być opisany np. jako dzień miesiąca, godzina, dzień roku,
- odcinek (inaczej: interwał) - tej strukturze odpowiada zdarzenie, odbywające się w pewnym okresie odcinka czasu, którego początek i koniec można opisać jako punkty,

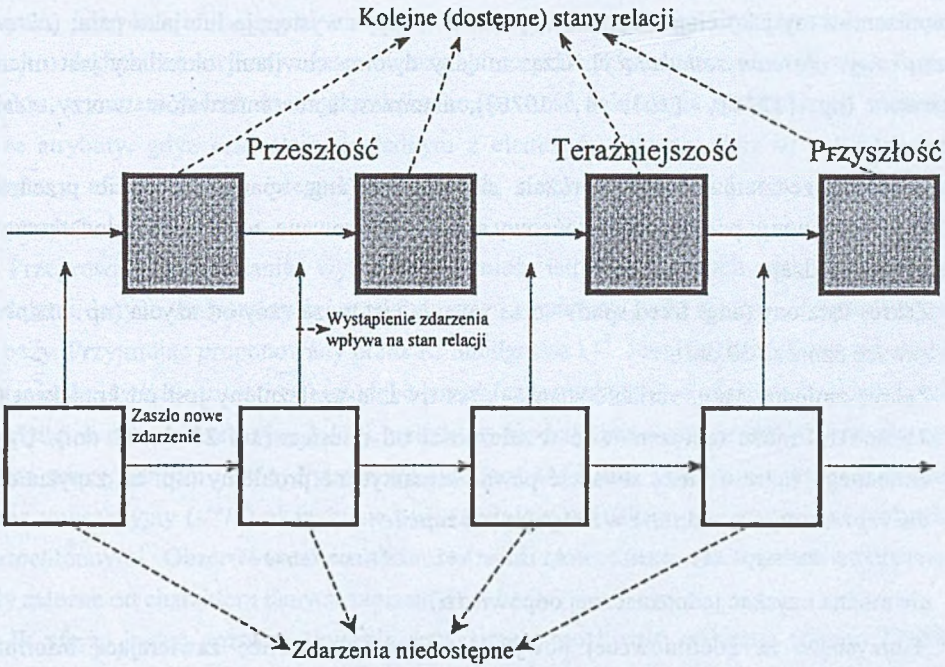
- łańcuchy regularne – jednorodne ciągi zdarzeń, składające się tylko z punktów lub tylko z odcinków.

Zdarzenia te są rozmieszczane w skalach, wśród których można wyodrębnić:

- skalę absolutną - w której przedstawia się zdarzenie tylko w zależności od jednostek czasu (sekundy, roku itd.) (np.: *Podwyżka pensji obejmująca pracowników zatrudnionych w 1980 roku*),
- skalę zależną - gdzie wszystkie zdarzenia w czasie rozłożone są w zależności od tzw. markera zdarzeń, którym może być między innymi zdarzenie lub moment wypowiedzi (np.: *Przyznanie premii tym pracownikom, którzy pracowali nad projektem nr 1234/95*),
- skalę rozmytą (bez jednostek) - do przedstawiania zdarzeń rozmytych i niepełnych (np.: *Zwolnienie wszystkich tych pracowników, którzy kiedykolwiek spóźnili się do pracy*).

Przykładem danych, które mogą być zamodelowane za pomocą zdarzeń,są np. transakcje finansowe, rezerwacje, sprzedaż.

4.5.3. Model stanu (ang. state model) – rys. 7



Rys. 7. Graficzna ilustracja modelu stanu
Fig. 7. State model graphic image

W modelu stanu zachodzi rejestracja stanu bazy w czasie (z definicji: *stan* jest to instancja danych, które istnieją lub trwają przez pewien okres czasu), czyli baza widziana jest

jako sekwencja stanów relacji. Każdy zapisany stan jest znacznikowany okresem czasu, w którym stan istnieje (rys. 7). Ten model ma zastosowanie np. w bazach przechowujących informacje o ratach kredytowych, polisach ubezpieczeniowych itp.; jest też pomocny przy wykonywaniu bilansów materiałowych.

Modele: stanu i zdarzeń są komplementarne. Dostając znany stan początkowy i końcowy, model zdarzeń może być przekonwertowany do modelu stanu przez chronologiczne użycie zdarzeń do konstruowania nowych stanów (przejście odwrotne można uzyskać przez wyliczenie różnic pomiędzy chronologicznie przyległymi, graniczącymi stanami).

4.6. Model czasu

Model linii czasu rzeczywistego w terminologii Temporalnych Systemów Zarządzania Bazą Danych to skończona sekwencja *chrononów*, będących reprezentacją podziału linii czasu rzeczywistego na niepodzielne segmenty równego rozmiaru (przy czym rozmiar zależy od potrzeb przetwarzania poszczególnych danych). Sekwencja chrononów to ciąg *chwil* (ang. instants), nie znajdujący swego odzwierciedlenia na linii czasu rzeczywistego, a reprezentowany jako ciąg chrononów, podczas których występuje lub jako para: (*chronon początkowy, chronon zamykający*). Czas między dwoma chwilami określany jest mianem *interwału* (np. `[1776], [July 4, 1976]`), natomiast suma interwałów tworzy *element temporalny.*

Wśród przedziałów czasu wyróżnia się *zakresy* (ang. span), tzn. takie przedziały, w których chronony: początkowy i końcowy są niesprecyzowane, rozróżniając dodatkowo ich następujące rodzaje:

- Zakres ustalony (ang. fixed span) - czas trwania jest niezależny od użycia (np. `%April%` zawsze oznacza 30 dni),
- Zakres zmienny (ang. variable span) – czas trwania uzależniony jest od kontekstu (np. `%1 month%` może reprezentować w zależności od miesiąca :od 28 do 31 dni). Użycie zmiennego zakresu może stwarzać pewne semantyczne problemy (np. na zapytanie: *ile dni reprezentuje* `%1month%` *w natępującym zapisie* :

  `(|12:00 PM May 31, 1991| + %1 month%) - %1 month%)?`

  nie można uzyskać jednoznacznej odpowiedzi).

Korzystając ze zdefiniowanej powyżej terminologii, tablicę zawierającą informacje o pracownikach biura można by zdefiniować w następujący sposób:

```
create table PRACOWNIK (imie character(20), id character(3),
                        data_ur event, wiek span,
                        kiedy_zatr interval)
```

5. Podsumowanie

W literaturze przedmiotu wymienia się kilka zagadnień (dziedzin), wymagających uwzględnienia osi czasu:

- automatyczne śledzenie danych zapamiętanych w bazie (np. transakcje finansowe),
- wzrost integralności danych,
- zachowywanie historii zmian atrybutów,
- lepsze uszczegółowienie danych potrzebnych dla hurtowni danych (ang. data mining),
- możliwość tworzenia bilansów materiałowych i tendencji.

Nie bez znaczenia jest również fakt, że temporalny model danych jest niezależny od granularności czasu (gdzie granularność jest to taki rozmiar każdej cząstki czasu, który może być określony przez projektanta systemu, a jest ograniczony właściwościami środowiska; np. sekunda, milisekunda, dzień, tydzień). Przykłady przedstawione w punkcie 4.3 jednoznacznie wykazują, że dodanie wsparcia czasów: wiarygodności i transakcyjnego do implementacji Systemów Zarządzania Relacyjnymi Bazami Danych może znacząco uprościć niektóre z zapytań, jak również wpływać na poprawę wydajności systemu. Rozszerzanie schematu relacji o dodatkowe atrybuty czasowe pozwala co prawda na zadawanie zapytań o historię obiektu, wymusza jednak na użytkowniku konieczność nakładania dodatkowych warunków na te atrybuty, gdyż czas staje się jednym z elementów klucza. Daje to z jednej strony pożądany efekt wzrostu integralności danych, natomiast z drugiej wydłuża sam zapis zapytania.

Przeprowadzone badania wykazały również istnienie ścisłych zależności między modelami danych temporalnych i strukturą czasu wspieraną przez system a typami zapytań do bazy. Przyjmując proponowany przez R. Snodgrassa i C. Jensena podział zapytań do bazy (notacja: *key/valid/transaction*[1]), model zdarzeń (czas punktowy) wydaje się być dedykowany dla pytań o czas transakcyjny (-/-/*), natomiast model stanu (czas przedziałowy) dla pytań o czas wiarygodności (-/*/-). Pytania o klucz (*point/-/-*) oraz pytania o czas wiarygodności i czas transakcyjny (-/*/*) okazały się być, zgodnie z oczekiwaniami, pytaniami najbardziej czasochłonnymi. Obserwowane rozbieżności czasu potrzebnego do uzyskania odpowiedzi były zależne od charakteru danych zapisanych w bazie.

W sferze badań pozostaje kwestia, czy istnieje możliwość realizacji takiego TDBMS, który umożliwi efektywne rozwiązywanie szerokiej klasy zadań, dotyczących danej dziedziny, nie wymuszając na programiście ścisłego dopasowywania struktury systemu do konkretnego zagadnienia.

---

[1] Stosowane oznaczenia: *point* (pojedyncza wartość pola), * (dowolna wartość pola), - (brak jakiejkolwiek wartości), *range* (zakres wartości).

LITERATURA

1. Snodgrass R., Ahn I.: Temporal Databases. IEEE Computer. Vol. 19. Nr 9. 1986.
2. Jensen Ch., Clifford J., Soo M.: A Consensus Glossary of Temporal Database Concepts. Aalborg University, 1993. Adres internetowy: tdbglossary@cs.arizona.edu.
3. Snodgrass R., Jajodija S., Clifford J.: Temporal Databases: Theory, Design and Implementation. Benjamin/Cummings Publishing Company, 1993.
4. Böhlen M., Snodgrass R.: Coalescing in Temporal Databases. 22 VLDB Conference. India, 1996. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
5. Böhlen M., Snodgrass R., Jensen C.: Adding Valid Time to SQL/Temporal. ANSI Expert's Contribution, 1996. Adres strony internetowej: http://www.timeconsult.com.
6. Kania K., Gołuchowski J.: Systemy temporalnych baz danych. Informatyka nr 9, 1996.
7. Hajnicz E.: Reprezentacja logiczna wiedzy zmieniającej się w czasie. Akademicka Oficyna Wydawnicza. Warszawa 1996.
8. Torp K., Jensen Ch., Böhlen M.: Layerd Implementation of Temporal DBMSs. TimeCenter Technical Report. 1997. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
9. Torp K., Jensen Ch., Snodgrass R.: Correct and Efficient Timestamping of Temporal Data. TimeCenter Technical Report, 1997. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
10. Jensen Ch., Snodgrass R.: Temporal Data Management. TimeCenter Technical Report, 1997. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
11. Steiner A.: A Generalization Approach to Temporal Data Models and their Implementations. Informatik Dept. Zurich 1997. Adres strony internetowej: http://www.timeconsult.com.
12. Jensen Ch., Snodgrass R.: Semantics of Time-Varying Attributes and Their Use for Temporal Database Design. TimeCenter Technical Report, 1997. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
13. Lin H.: Efficient Conversion Between Temporal Granularity. TimeCenter Technical Report, 1997. Adres strony internetowej: http://www.cs.auc.dk/research/DBS/tdb/TimeCenter.
14. Bach M., Werner A.: Rozszerzenia modelu relacyjnego w eksperymentalnym systemie Postgres95. ZN Pol. Śl. s. Informatyka z. 33, Gliwice 1997.

Recenzent: Dr hab. inż. Stanisław Wołek
Prof. Pol. Rzeszowskiej

Wpłynęło do Redakcji 12 stycznia 1999 r.

Abstract

The topic of this article is the overview of Temporal Database Management Systems. The article is organized as follows. The first part is connected with fundamental concepts taken from logical formalizations, which are specific to representing time structure. This is the basic platform to construct temporal databases which are specified in the second part. Finally, conclusions and directions for future are offered in section 5.

The main purpose of the paper is to identify the difficult problem of supporting and implementing some aspects of time in relational databases. Time-varying data is manipulated in most database applications. But, many temporal queries are either difficult to simulate in SQL, or require embedding SQL in a procedural language (section 4.8, example 3). In this article, alternatives are expressed in PSQL, SQL/T, TSQL2 and SQL/Temporal (section 4.8, examples: 1-3). The tutorials show needs for additional time support in SQL as well as the necessity for developing commercial TDBMS.