History and Point in Time in Enterprise Applications

Prof.dr. Constantin-Gelu APOSTOL, Catedra de Informatică Economică, A.S.E. Bucureşti
ec. Daniel BĂLĂCEANU, TotalSoft S.A., Bucureşti

First part points out the main differences between temporal and non-temporal databases. In the second part, based on identification of the three main categories of time involved in database applications: user-defined time, valid time and transaction time, some relevant solutions for their implementation are discussed, mainly from the point of view of database organization and data access level of enterprise applications. The final part is dedicated to the influences of historical data in the business logic and presentation levels of enterprise applications and in application services, as security, workflow, reporting.
Keywords: *temporal databases, non-temporal databases, user-define time, valid time, transaction time, enterprise application architecture, application services*

Pentru proiectarea bazelor de date şi a aplicațiilor de întreprindere o anumită viziune asupra dimensiunii temporale a datelor este obligatorie.

Baze de date temporale şi non-temporale

După modul cum tratează atributul timp, bazele de date pot fi împărțite în două mari categorii: **temporale**, respectiv **non-temporale**.

Majoritatea sistemelor de gestiune a bazelor de date (SGBD) comerciale sunt în esență non-temporale, adică nu tratează distinct dimensiunea temporală a datelor. Ca urmare, într-o abordare simplistă, se poate ajunge ca toate datele memorate să fie considerate ca valide numai la timpul prezent. Datele trecute au fost suprascrise sau şterse, iar cele viitoare sunt considerate ca valide la un timp viitor, dar nu în prezent. Această caracteristică este valabilă atât pentru SGBD relaționale (SGBDR), cât şi pentru SGBD orientate obiect (SGBD-OO).

În consecință, considerând exemplul tabelei *Persoana* (figura 1), dintr-o bază de date relațională, absența oricăror atribute (coloane) de timp conduce implicit la o abordare non-temporală. Eventualele schimbări de nume, prenume sau salariu se fac prin suprascriere, devenind imposibil răspunsul la întrebări de tipul: „Câte persoane şi-au schimbat numele în ultimul an ?" sau „De când are Ionescu Mihai salariul actual?" sau „Cine avea cel mai mare salariu acum două luni?" etc.

IDPersoana	Nume	Prenume	Salariu
1000	Popescu	Ioana	250
1001	Ionescu	Mihai	125
1000	Mihnea	Ioana	250

Fig.1. Exemplu de tabelă relațională non-temporală

În cazul unui SGBD-OO, persoanele vor deveni obiecte de un anumit tip, grupate în colecții, caracterizate prin proprietăți ca *Nume*, *Prenume, Salariu* etc. În absența unor proprietăți temporale ale obiectelor de diverse tipuri se ajunge însă, ca şi pentru SGBDR, în situația în care multe cerințe de analiză a datelor nu pot fi satisfăcute.

Pentru modelarea atributului de timp pot fi identificate trei tipuri de date temporale fundamentale (Snodgrass, 2000):

- **Moment**: un moment sau punct în timp, când se întâmplă ceva (de exemplu, 20 iulie 1969, 20:17:40 GMT, când astronautul Neil Armstrong a pus piciorul pe Lună);
- **Interval**: o durată nelocalizată în timp (de exemplu, 12 luni, 15 minute, 30 secunde etc.);
- **Perioadă**: o durată localizată în timp (de exemplu, 1 decembrie 2005 la 31 ianuarie 2006).

În acord cu terminologia folosită în (Goralwalla et al., 2001), perioadele corespund unor **date temporale ancorate**, în timp ce intervalele sunt **date temporale neancorate**.

Din perspectiva aplicațiilor cu baze de date, este importantă distincția între trei categorii fundamentale de timp:

- **Timpul utilizator** (*user-defined time*), reprezentând uzual un moment de timp (*point in time*) a cărui semnificație este determinată de utilizator (de exemplu, momentul când se execută o aplicație).
- **Perioada de validitate** (*valid time*), desemnând perioada de timp pe parcursul căreia un fapt este adevărat în raport cu situația din lumea reală.
- **Perioada de tranzacționare** (*transaction time*), desemnând perioada de timp pe parcursul căreia un fapt este reflectat informațional în baza de date.

Așa cum evidențiază abordarea propusă de TimeConsult (1998), un prim pas spre trecerea la baze de date temporale îl reprezintă asocierea unor mărci de timp (*timestamp*) datelor, ceea ce permite realizarea distincției între diferitele stări ale bazei de date. Între soluțiile posibile se remarcă marcarea temporală a entităților, respectiv marcarea temporală a valorilor proprietăților acestor entități. În modelul de date relațional sunt marcate tuplurile, iar în modelele de date orientate obiect sunt marcate obiectele şi/sau valorile proprietăților.

În funcție de modul de folosire a perioadelor de timp pentru marcare se distinge între mai multe forme de baze de date temporale:

- **Baze de date istorice** (*historical database*), care memorează datele în raport cu perioada de validitate.
- **Baze de date tranzacționale** (*rollback database*), care memorează datele în raport cu perioada de tranzacționare.
- **Baze de date bitemporale** (*bitemporal database*), care memorează datele în raport cu ambele perioade de timp.

În raport cu această clasificare, bazele de date non-temporale pot fi considerate ca fiind doar „instantanee" (*snapshot databases*), având capacitatea să reflecte o singură stare a lumii reale, uzual pe cea mai recentă.

Fără a ne propune să analizăm în ce măsură diversele SGBD comerciale actuale oferă soluții pentru implementarea caracteristicilor bazelor de date temporale, remarcăm că discuțiile pe această temă sunt în continuare actuale, aşa cum evidențiază, între altele, o recentă abordare a acestui subiect (Haughey et al., 2006).

Din perspectiva reflectării atributului timp în aplicații, se remarcă faptul că majoritatea arhitecturilor structurează aplicațiile pe mai multe niveluri (figura 2).

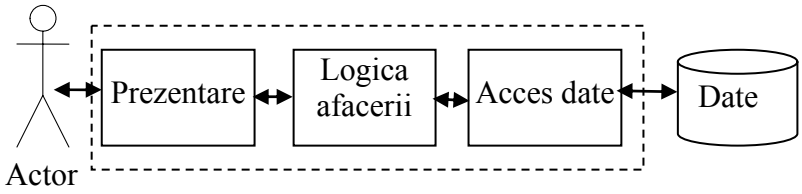

Fig.2. Arhitectura generală a aplicațiilor de întreprindere

Aspectele legate de istoric apar pe ultimele două niveluri, de acces la date şi de stocare a datelor. Nivelurile de logica afacerii şi de prezentare, plus alte posibile niveluri superioare, folosesc (sau ar trebui să folosească) nivelul de acces la date pentru orice probleme legate de istoric, mecanismul fiind transparent pentru ele.

Înțelegerea corectă a celor trei categorii de timp evidențiate anterior este esențială pentru dezvoltatori, având în vedere că în bazele de date şi în aplicații pot interveni - succesiv sau simultan - una, două sau chiar toate cele trei categorii de timp.

Exemplificările folosite în continuare în acest scop se bazează atât pe diagrame UML, cât şi pe valori concrete din tuplurile unor tabele ale bazei de date pentru gestiunea angajaților.

Utilizarea perioadei de validitate

Pentru aplicația de gestiune a angajaților este necesară înregistrarea datelor personale şi a

celor legate de angajare. Fiecare angajat este unic identificat prin marca sa, iar între atributele asociate uzual clasei *Angajat* se regăsesc (figura 3): *Nume*, *Prenume*, *Adresa*, *tipul de contract*, *perioada de angajare*, *salariul net lei*, *salariul brut lei*, *salariul net USD*, *salariul brut USD.*

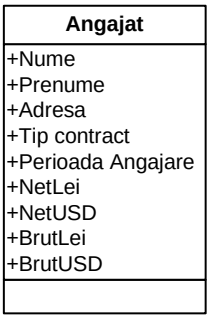

Fig.3. Diagrama clasei *Angajat*

Un angajat poate avea una sau mai multe adrese: de bază, flotantă, temporară etc. Angajatul îşi poate schimba numele; aceste schimbări trebuie înregistrate. Tipul de contract şi perioada de angajare sunt relativ stabile, putându-se modifica doar în cazuri excepționale. Salariul angajatului se poate modifica în timp. Angajatul poate opta pentru o negociere a salariului în lei sau în valută, la nivel brut sau net. Funcție de opțiunea sa, toate celelalte sume (*NetLei*, *NetUSD*, *BrutLei*, *BrutUSD*) se calculează în fiecare lună.

În baza acestor cerințe, modelul de mai sus se rafinează (figura 4), introducerea dimensiunii temporale determinând separarea într-o clasă distinctă a atributelor asupra cărora se aplică. În exemplul considerat, atributele *Nume* şi *Prenume* evoluează diferit de datele de angajare, fapt ce a determinat izolarea lor în clasa *Persoana*. Similar, datele de salariu au propria lor evoluție, lunară, şi au fost izolate în clasa *Salariu*.

O altă rafinare o reprezintă apariția celor două atribute, *DataStart* şi *DataFinal,* cu ajutorul cărora este înregistrată perioada de validitate a înregistrărilor.

Un *Angajat* poate avea unul sau mai multe salarii de-a lungul timpului. Dacă se adaugă însă calificatorii temporali, atunci un angajat are un singur *Salariu* la un anumit moment de timp.

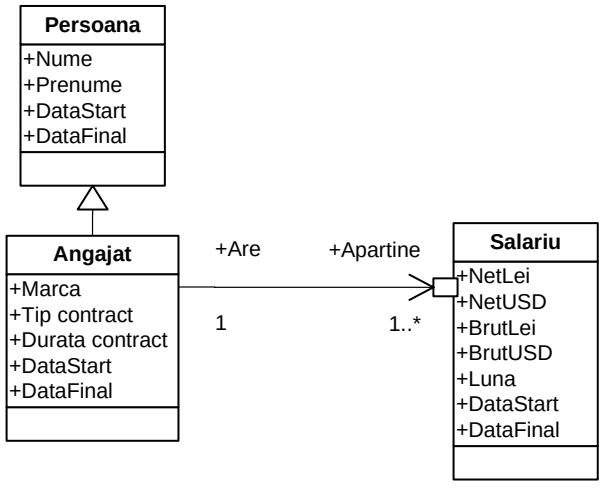

Fig.4. Rafinarea modelului prin includerea dimensiunii temporale

Transpunerea acestui model obiectual într-un model relațional ridică unele probleme. De regulă, introducerea dimensiunii temporale adaugă un grad de multiplicitate, în sensul că acolo unde există o singură înregistrare pe obiect, odată cu istoricul vor exista mai multe.

Există mai multe abordări posibile pentru transpunerea dimensiunii temporale în termenii modelului relațional.

De exemplu, considerând clasele *Persoana* şi *Angajat*, în plus față de modelul obiectual vor apare cheile primare (PK- *Primary Key*), cheile externe (FK – *Foreign Key*), relațiile dintre tabele. Necesitatea cheilor explică prezența atributelor de tip ID (*IDPersoana*, *IDAngajat*, *IDSalariu* etc.).

În literatura de specialitate sunt propuse di-

verse tehnici de transpunere a modelelor obiect în structuri relaţionale (Blaha & Rumbaugh, 2004) precum şi pattern-uri temporale (Fowler, 2005) asociate modelelor de istoric al datelor.

În funcţie de modelul de istoric ales, modelul relaţional poate arăta diferit. În cazul unui sistem simplu bazat pe înregistrarea modificărilor (*audit log*), cea mai bună soluţie este adăugarea unei noi tabele în care sunt păstrate înregistrările istorice (figura 5). În acest scenariu, în *tblPersoana* sunt păstrate datele curente, iar în *tblPersoanaLog* datele istorice. Modelul este bun atunci când aplicaţia foloseşte uzual doar datele valide la momentul curent (sau un alt moment ales de aplicaţie); datele istorice sunt folosite doar izolat, pentru consultări de genul „cum au evoluat datele personale?"

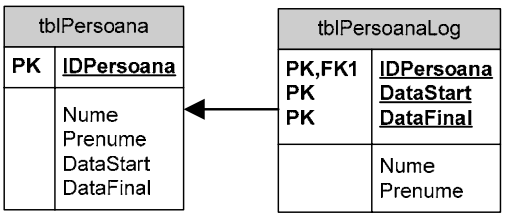

Fig.5. Model relaţional cu tabelă pentru date istorice

Într-un alt scenariu posibil (figura 6), toate datele sunt păstrate într-o singură tabelă.

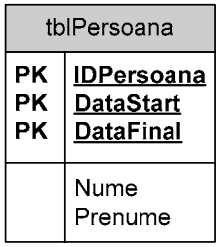

Fig.6. Model relaţional cu o singură tabelă

Fără cele două atribute (coloane) temporale, *IDPersoana* (vrzi figura 5) poate forma singură cheia primară, cu consecinţa că fiecărei persoane îi corespunde un singur tuplu. Prin adăugarea coloanelor temporale, *DataStart* şi *DataFinal*, care împreună cu *IDPersoana* formează cheia primară, fiecărei persoane îi pot corespunde mai multe înregistrări, dar, într-un moment de timp dat, îi corespunde una singură, indiferent de scenariu.

Fără a ne propune, în acest cadru, ilustrarea diverselor interogări posibile prin implementarea acestor modele, recomandăm una din cele mai complete lucrări de specialitate dedicată tratării istoricului în baze de date (Snodgrass, 2000).

Utilizarea perioadei de tranzacţionare

Tabelele analizate surprind doar una din dimensiunile temporale: durata de validitate, dar nu conţin date despre istoricul tranzacţiilor. Dacă, spre exemplu, datele personale sunt folosite pentru a tipări „fluturaşii" de salarii, este important de ştiut care erau datele valide cunoscute de aplicaţie la momentul tipăririi lor. În acest scop sunt adăugate două noi coloane temporale, *DeLa* şi *PanaLa*, care specifică perioada în care sistemul a cunoscut şi folosit acea înregistrare (figura 7), corespunzând perioadei de tranzacţionare.

În exemplul din figura 7, în tabel sunt surprinse ambele dimensiuni temporare, perioada de validitate şi perioada de tranzacţionare. O mulţime de informaţii pot fi extrase dintr-o astfel de structură, dacă este analizată cu atenţie.

IDPersoana	Nume	Prenume	DataStart	DataFinal	DeLa	PanaLa
1000	Popescu	Ioana	15.03.1976	10.10.2005	05.06.2004	20.10.2005
1001	Ionescu	Mihai	20.10.1976	31.12.9999	06.06.2004	31.12.9999
1000	Mihnea	Ioana	10.10.2005	31.12.9999	20.10.2005	31.12.9999

Fig.7. Includerea de atribute pentru perioada de tranzacţionare

În cele ce urmează este descrisă evoluţia înregistrărilor din acest tabel.

Pasul 1: Pe 5.06.2004 s-a înregistrat *Popescu Ioana*, înregistrare validă din 15.03.1976. (Asupra coloanelor *DataFinal* şi *PanaLa* se revine la pasul 3).

IDPersoana	Nume	Prenume	DataStart	DataFinal	DeLa	PanaLa
1000	Popescu	Ioana	15.03.1976	31.12.9999	05.06.2004	31.12.9999

Pasul 2: Pe 6.06.2004 s-a înregistrat Ionescu Mihai, înregistrare validă din 15.03.1976 pâna la infinit (corespunzând, în standardul SQL, datei de 31.12.9999).

IDPersoana	Nume	Prenume	DataStart	DataFinal	DeLa	PanaLa
1000	Popescu	Ioana	15.03.1976	31.12.9999	05.06.2004	31.12.9999
1001	Ionescu	Mihai	20.10.1976	31.12.9999	06.06.2004	31.12.9999

Pasul 3: Pe 20.10.2005 sistemul înregistrează modificarea numelui persoanei 1000, din Popescu în Mihnea, începând cu 10.10.2005.

IDPersoana	Nume	Prenume	DataStart	DataFinal	DeLa	PanaLa
1000	Popescu	Ioana	15.03.1976	10.10.2005	05.06.2004	20.10.2005
1001	Ionescu	Mihai	20.10.1976	31.12.9999	06.06.2004	31.12.9999
1000	Mihnea	Ioana	10.10.2005	31.12.9999	20.10.2005	31.12.9999

Odată cu apariția noilor informații, primul rând a fost modificat şi el. *DataFinal* s-a modificat din 31.12.9999 în 10.10.2005, data pâna la care este validă înregistrarea, iar *PanaLa* s-a modificat în 20.10.2005, data la care sistemul a fost conştient de această modificare.

Utilizarea timpului definit de utilizator

Aşa cum reflectă soluțiile analizate, folosirea istoricului poate merge de la o simplă înregistrare a datelor istorice şi pâna la consultări ale întregii evoluții a unui obiect. Ele acoperă partea de înregistrare şi consultare, dar procesele de afaceri pot necesita răspunsuri la întrebări de tipul: „Care a fost starea întregului sistem cu două luni în urmă?” sau „Care va fi aceasta peste 3 luni?”.

Asemenea întrebări cer mai mult decât înregistrarea şi consultarea datelor istorice şi se bazează pe conceptul de **timp definit de utilizator** (*point in time*), care desemnează capacitatea unei aplicații de a funcționa în mod nativ la orice moment de timp ales.

Pentru a ilustra implicațiile implementării în aplicații a acestui concept se consideră o tabelă *Persoana*, cu înregistrările din figura 8.

RecordId	EntityId	Nume	Marca	DataStart	DataFinal	Salariu
1	1	Mihai	10000	01.01.2005	01.03.2005	100
2	2	Dan	10001	01.05.2005		150
3	1	Mihai	10000	01.03.2005		125
4	3	Vasile	10002	01.01.2005	01.03.2005	100
5	3	Vasile	10002	01.03.2005		130

Fig.8. Exemplu de tupluri dintr-o tabelă de personal

Pot fi imaginate mai multe scenarii de utilizare a acestei tabele în aplicația de gestiune a personalului.

Scenariul 1: Lista angajaților. La data de 01.02.2005 utilizatorul deschide aplicația şi solicită lista angajaților din companie. Apar *Mihai* şi *Vasile* reprezentând mulțimea înregistrărilor valide la data curentă.

Scenariul 2: Detalii angajat. Este selectat angajatul *Mihai*. Salariul său este de 100, reprezentând salariul valid la 01.02.2005.

Scenariul 3: Istoric angajat. Se apelează opțiunea *Istoric*, obținându-se cele două înregistrări ale angajatului, cu salariul de 100, respectiv 125.

De remarcat data de 01.02.2005, care este un element foarte important, ea dând momentul definit de utilizator pentru care sunt furnizate toate informațiile. Modul de tratare a acestei date, numită uzual **data curentă a aplicației**, poate fi diferit, cu următoarele variante de abordare:

- **Data curentă**. Există aplicații la care se consideră întotdeauna data curentă, aşa cum este ea furnizată de serverul de aplicații sau de stația client. De regulă, aceste aplicații au nevoie de date istorice cel mult pentru operații de audit.
- **Mulțime de valori**. Există aplicații în care această dată poate lua valori dintr-o mulțime strictă de valori. Pentru o aplicație de salarii,

de exemplu, sau de contabilitate, luna de calcul este foarte importantă. Din acest motiv, de regulă, utilizatorii au posibilitatea să aleagă luna pentru care vor să lucreze, aplicația furnizându-le datele valide în acea lună.

• **Orice dată.** Există aplicații în care utilizatorul poate alege orice dată doreşte, aplicația furnizându-i datele valide la acel moment de timp. Toate interogările sunt făcute la data aleasă de utilizator.

Implicații ale utilizării datelor istorice pe alte niveluri ale aplicațiilor

Modelele şi soluțiile evidențiate au analizat implicațiile dimensiunii temporale pentru nivelurile de acces la date şi de stocare a datelor (conform figurii 1). Cerințe de considerare a evoluției în timp pot apare însă şi în celelalte niveluri ale aplicației:

• **Nivelul de logică a afacerii**. Dacă se consideră, de exemplu, o aplicație de salarizare, trebuie avut în vedere că legislația din domeniu, cel puțin în România, este destul de dinamică. Regulile după care s-au calculat salariile anul trecut pot fi diferite de cele de astăzi. Dacă vrem să ne întoarcem la acel moment de timp şi să refacem calculul de salarii, vom avea nevoie de componenta de logică a afacerii de la acel moment.

• **Nivelul de prezentare**. Există aplicații dinamice în care interfața client este construită la momentul interogării: meniurile disponibile se afişează în funcție de drepturi, acțiunile pe care le poate face utilizatorul sunt funcție de starea sistemului etc. Toate aceste elemente pot fi şi ele afectate de istoric.

Implicațiile dimensiunii temporale asupra serviciilor din aplicații

De regulă, diferitele servicii în cadrul unei aplicații folosesc şi ele structuri proprii de date. Problema istoricului în cadrul acestor servicii înseamnă de fapt semnificația şi modul cum folosesc ele datele istorice. Câteva dintre tipurile de servicii afectate de atributul timp sunt:

• **Sistemul de securitate.** De regulă, în orice model de securitate apar elemente ca utilizatori, roluri, acțiuni, obiecte. Modelul de securitate poate merge de la unul simplu, bazat pe roluri, până la unul complex, cu acțiuni şi obiecte care trebuie securizate. Şi aici pot apare întrebări legate de istoric: „Ce permisiuni avea George la 1 iulie 2005?"; „Exista utilizatorul George la 1 iulie 2005?"; „Cum a putut George extrage acest raport, pentru că acum nu are aceste permisiuni?" etc. Dacă se reia exemplul aplicației de personal, în cadrul ei există rolul de manager, care oferă utilizatorului dreptul de a vizualiza angajații din cadrul departamentului său. Dacă utilizatorul se întoarce în timp, modificând data aplicației, ce angajați va vedea el? Cei existenți la acel moment? Era el manager la momentul de timp pe care şi l-a ales?

• **Mecanisme de flux de lucru.** Există aplicații care oferă facilități de flux de lucru (*workflow*), date de posibilitatea ca un obiect să treacă printr-o succesiune de stări ca urmare a unei acțiuni a utilizatorului sau a unor evenimente în cadrul aplicației. În acest caz, succesiunea de stări prin care a trecut un document reprezintă istoricul acelui document în raport cu fluxul de lucru. Mai mult, se paote face distincție între istoricul stărilor în flux şi istoricul conținutului documentului, care pot fi diferite.

• **Istoricul în sistemul de raportare.** Scenariile analizate anterior au evidențiat câteva tipuri de interogări legate de istoric. O problemă care poate apare aici este combinarea istoricului mai multor obiecte.

Bibliografie

1. Blaha, M.R., Rumbaugh, J.R. (2004). *Object-Oriented Modeling and Design with UML* (2nd Edition), Prentice Hall
2. Martin, F. (2005). *Temporal Patterns*, http://martinfowler.com/eaaDev/timeNarrative.html
3. Goralwalla, I.A., Leontiev, I., Özsu, M.T., Szafron, D., Combi, C. (2001). *Temporal Granularity: Completing the Puzzle*, Journal of Intelligent Information Systems, Vol.16, Nr.1, pag. 41-63
4. Haughey, T., Kelley, C., Oates, J. (Ianuarie, 2006). *Is there any database management system which supports temporal databases?* www.dmreview.com/editorial/dmreview/articleID=1045306
5. Snodgrass, R.T. (2000). *Developing Time-Oriented Database Applications in SQL*, Morgan Kaufmann Publishers, Inc., San Francisco
6. TimeConsult (1998). *What are temporal Databases?* www.timeconsult.com/TemporalData/TemporalDB.html