Time is Money for Nothing

Fiola de SQL (12) – *Marin Fotache*

De mult timp căutam o formulă care să constituie sinteza filosofică a modurilor de viață american și românesc. *Time is Money* este una din cele mai celebre vorbe de duh ale americanilor. Cât despre expresia *Money for Nothing*, deși am auzit-o prin a doua parte a anilor '80 la Dire Straits (albumul *Brother In Arms*), să recunoaștem că se pliază grozav pe insula noastră de latinitate și peninsula de balcanitate.

Odată trecut cu bine preludiul filosofic, să precizăm că subiectul fiolei de astăzi ține de modul în care SQL tratează datele calendaristice. Știu, știu, acum doi ani, când combinația Nostradamus-Year 2K făcea furori, articolul ar fi avut ceva mai multă căutare.

SQL-92 a luat în discuție destul de serios tipurile dată-timp, operând o delimitare între:

- evenimente, pentru care există tipurile `DATE`, `TIME`, `TIMESTAMP`;
- intervale – perioade de timp dintre două evenimente (momente) - tipul `INTERVAL` (`YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`) și
- perioade – intervale cu un moment fix de început, pentru care este necesară o combinație de două câmpuri, fie `TIMESTAMP - TIMESTAMP`, fie `TIMESTAMP - INTERVAL`.

Generozitatea SQL-92 nu se regăsește, din păcate, în prea multe SGBD-uri, astfel încât funcțiile pentru lucrul cu date calendaristice sunt destul de eterogene. Tipul DATE este unul general, iar `TIME` și `TIMESTAMP` cvasi-generale.

Formatul datei calendaristice în SQL-92 este `yyyy-mm-dd`, în conformitate cu standardul ISO 8601. `TIMESTAMP(n)` este definit ca o marcă temporală cu *n* poziții zecimale, specificațiile FIPS-127 cerând ca *n* să fie minimum 5, lucru esențial pentru jurnalizarea tranzacțiilor. Spre exemplu, pentru baza de date a unei bănci este vital ca pentru orice operațiune cu un cont (depunere, virament, retragere, închidere) să se consemneze cu exactitate momentul consumării operațiunii. De asemenea, pentru urmărirea modificărilor petrecute în liniile unei tabele, se poate folosi o tabelă-jurnal care înregistrează tipul și momentul actualizării.

Pentru data/ora curentă, standadul SQL pune la dispoziție funcțiile `CURRENT_DATE`, `CURRENT_TIME (n)`, `CURRENT_TIMESTAMP (n)`, însă titulatura și sintaxa acestora depind de la SGBD la SGBD: `SYSDATE`, `TODAY`, `DATE()`, `CURRENT DATE`.

Pentru cele ce urmează, luăm drept cobai o tabelă ce consemnează facturile emise: `FACTURI {NrFact, DataFact, CodClient, ValoareTotala}`.

Figura 1. O expresie de tip dată calendaristică

FACTURA	DATA_FACTURARE	SCADENTA_INCASARE
1111	2000-08-01	2000-08-15
1112	2000-08-01	2000-08-15
1113	2000-08-01	2000-08-15
1114	2000-08-01	2000-08-15
1115	2000-08-02	2000-08-16
1116	2000-08-02	2000-08-16
1117	2000-08-03	2000-08-17
1118	2000-08-04	2000-08-18
1119	2000-08-07	2000-08-21
1120	2000-08-07	2000-08-21
1121	2000-08-07	2000-08-21
1122	2000-08-07	2000-08-21

Constante de tip dată calendaristică

Să începem cu un exemplu simplu:
Care sunt facturile emise în perioada 2-5 august 2000 ?
Soluție DB2:

```
SELECT *
FROM FACTURI
WHERE DataFact BETWEEN '2000-08-02' AND '2000-08-05'
```

Soluția Oracle utilizează `TO_DATE`:

```
SELECT *
FROM FACTURI
WHERE DataFact BETWEEN TO_DATE('02/08/2000','DD/MM/YYYY')
AND TO_DATE('05/08/2000','DD/MM/YYYY') ;
```

Constantele de tip dată calendaristică trebuie încadrate de acolade în Visual FoxPro. În plus, sintaxa versiunii 6 este una specială, după cum se observă în interogarea:

```
SELECT * ;
FROM FACTURI  ;
WHERE DataFact BETWEEN {^2000/08/02} AND {^2000/08/05}
```

Adunări și scăderi cu date calendaristice

Să presupunem că orice factură trebuie încasată în maximum două săptămâni de la data emiterii. Să se afișeze scadențele.
Soluția DB2 este:

```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   DataFact + 14 DAY AS Scadenta_Incasare
FROM FACTURI
```

Rezultatul din figura 1 se obține grație expresiei `DataFact + 14 DAY`. Într-o expresie de tip `DATE` sau `DATETIME`, imediat după număr, trebuie indicată semnificația acestuia: `YEAR` (`YEARS`), `MONTH` (`MONTHS`), `DAY` (`DAYS`) sau, după caz, `HOUR` (`HOURS`), `MINUTE` (`MINUTES`), `SECOND` (`SECONDS`), `MICROSECOND` (`MICROSECONDS`).

La aceeași problemă, Oracle și Visual FoxPro permit o variantă de interogare mai simplă (de data aceasta); `DataFact` fiind un atribut de tip `DATE`, în SELECT-ul următor 14 reprezintă numărul zilelor:

```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   DataFact + 14 AS Scadenta_Incasare
FROM FACTURI
```

Figura 2. Intervalul dintre 5 februarie (data execuției interogării) și data fiecărei facturi (DB2)

FACTURA	DATA_FACTURARE	TIMP_SCURS
1111	2000-08-01	604
1112	2000-08-01	604
1113	2000-08-01	604
1114	2000-08-01	604
1115	2000-08-02	603
1116	2000-08-02	603
1117	2000-08-03	602
1118	2000-08-04	601
1119	2000-08-07	529
1120	2000-08-07	529
1121	2000-08-07	529
1122	2000-08-07	529

Dacă însă am presupune că *scadența este peste două luni de la facturare*, interogările ar trebui modificate astfel:

DB2:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   DataFact + 2 MONTHS AS Scadenta_Incasare
FROM FACTURI
```

Oracle:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
    ADD_MONTHS(DataFact,2) AS Scadenta_Incasare
FROM FACTURI
```

Funcția **Visual FoxPro** corespondentă ADD_MONTHS este GOMONTH:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
GOMONTH(DataFact,2)  AS Scadenta_Incasare
FROM FACTURI
```

Complicându-ne și mai zdravăn, dorim să afișăm pentru fiecare factură ce dată va fi peste 1 an, două luni și 25 de zile de la momentul emiterii.

Soluția DB2:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   DataFact + 1 YEARS + 2 MONTHS + 25 DAYS AS
   O_Data_Viitoare
FROM FACTURI
```

Soluția Oracle necesită transformarea anilor în luni:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
    ADD_MONTHS(DataFact,14)+15 AS O_Data_Viitoare
FROM FACTURI
```

În **VFP**:
```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
GOMONTH(DataFact,14)+15 AS O_Data_Viitoare
FROM FACTURI
```

Cât privește operațiunile de adunare și scădere între două date calendaristice, aici lucrurile se prezintă și mai diferențiat. Spre exemplu, interesează *intervalul scurs între momentul curent și cel al emiterii fiecărei facturi*. Interogarea DB2 de mai jos:

```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   CURRENT DATE - DataFact AS Timp_Scurs
FROM FACTURI
```

furnizează valorile din figura 2. Trebuie avut în vedere că interogarea a fost executată pe 5 februarie 2001.

Rezultatul scăderii a două date calendaristice conține numărul de ani, luni și zile. În figura 2 valoarea 604 înseamnă 6 luni și 4 zile, iar 529 - 5 luni și 29 de zile. Bineînțeles că apelând la funcții de conversie și extragere se poate obține un format de prezentare ceva mai agreabil.

Oracle necesită câteva artificii. Funcția MONTHS_BETWEEN calculează numărul lunilor cuprinse între două date calendaristice, iar TRUNC asigură obținerea valorilor întregi în rezultat (fără rotunjiri):

```
SELECT NrFact AS Factura, DataFact AS Data_Facturare,
   TRUNC(MONTHS_BETWEEN(TRUNC(SYSDATE), DataFact),0)
AS Luni_Scurse,
ADD_MONTHS(TRUNC(SYSDATE), -
         TRUNC(MONTHS_BETWEEN(TRUNC(SYSDATE), DataFact),0))
         - DataFact AS Zile_Scurse
FROM FACTURI
```

Interogarea generează o listă de forma celei din figura 3.

Alte funcții pentru zile

Scadența fiecărei facturi emise este 20 de zile. Dacă însă data limită cade într-o sâmbătă sau duminică, atunci scadența se mută în lunea următoare. Care sunt noile scadențe în aceste condiții ?

Soluția DB2 se bazează pe funcția DAYOFWEEK, care întoarce 1, dacă data-argument se referă la duminică, 2 pentru luni… 6 pentru sâmbătă. DAYNAME afișează numele zilei, iar interogarea următoare va genera rezultatul din figura 4.

```
SELECT NrFact, DataFact,
DataFact + 20 DAYS AS Scadenta1,
DAYNAME(DataFact + 20 DAYS) AS NumeZi1,
CASE
   WHEN DAYOFWEEK(DataFact + 20 DAYS)  = 6
```

Figura 3. Intervalul dintre 5 febr. 2001 (data execuției interogării) și data fiecărei facturi (Oracle)

FACTURA	DATA_FACTURARE	LUNI_SCURSE	ZILE_SCURSE
1111	2000-08-01	6	4
1112	2000-08-01	6	4
1113	2000-08-01	6	4
1114	2000-08-01	6	4
1115	2000-08-02	6	3
1116	2000-08-02	6	3
1117	2000-08-03	6	2
1118	2000-08-04	6	1
1119	2000-08-07	5	29
1120	2000-08-07	5	29
1121	2000-08-07	5	29
1122	2000-08-07	5	29

Figura 4. Scadența rectificată a încasării facturilor

NRFACT	DATAFACT	SCADENTA1	NUMEZI1	SCADENTA	ZI_SCADENTA
1111	2000-08-01	2000-08-21	MONDAY	2000-08-21	MONDAY
1112	2000-08-01	2000-08-21	MONDAY	2000-08-21	MONDAY
1113	2000-08-01	2000-08-21	MONDAY	2000-08-21	MONDAY
1114	2000-08-01	2000-08-21	MONDAY	2000-08-21	MONDAY
1115	2000-08-02	2000-08-22	TUESDAY	2000-08-22	TUESDAY
1116	2000-08-02	2000-08-22	TUESDAY	2000-08-22	TUESDAY
1117	2000-08-03	2000-08-23	WEDNESDAY	2000-08-23	WEDNESDAY
1118	2000-08-04	2000-08-24	THURSDAY	2000-08-24	THURSDAY
1119	2000-08-07	2000-08-27	SUNDAY	2000-08-28	MONDAY
1120	2000-08-07	2000-08-27	SUNDAY	2000-08-28	MONDAY
1121	2000-08-07	2000-08-27	SUNDAY	2000-08-28	MONDAY
1122	2000-08-07	2000-08-27	SUNDAY	2000-08-28	MONDAY

```
THEN DataFact + 22 DAYS
   ELSE
      CASE WHEN DAYOFWEEK(DataFact + 20 DAYS)  = 1
         THEN DataFact + 21 DAYS
         ELSE DataFact + 20 DAYS
      END
END AS Scadenta,
DAYNAME (CASE WHEN DAYOFWEEK(DataFact + 20 DAYS)  = 6
   THEN DataFact + 22 DAYS
   ELSE
      CASE WHEN DAYOFWEEK(DataFact + 20 DAYS)=1
      THEN DataFact + 21 DAYS
      ELSE DataFact + 20 DAYS
      END
END ) AS Zi_Scadenta
FROM FACTURI
```

Soluția Oracle 8i2 este una ceva mai simplă, un rol decisiv avându-l, pe lângă structura `CASE`, puternica funcție `TO_CHAR`:

```
SELECT NrFact, TO_CHAR(DataFact,'YYYY-MM-DD') AS DataFact,
   TO_CHAR(DataFact + 20,'YYYY-MM-DD') AS Scadenta1,
   TO_CHAR(DataFact + 20,'DAY') AS NumeZi1,
   CASE WHEN TO_CHAR(DataFact + 20,'DAY')  = 'SATURDAY'
THEN TO_CHAR(DataFact + 22,'YYYY-MM-DD')
      ELSE
         CASE WHEN TO_CHAR(DataFact + 20,'DAY')  = 'SUNDAY'
            THEN TO_CHAR(DataFact +
21,'YYYY-MM-DD')
            ELSE TO_CHAR(DataFact +
20,'YYYY-MM-DD')
         END
   END AS Scadenta,
   TO_CHAR(
CASE WHEN TO_CHAR(DataFact + 20,'DAY')
= 'SATURDAY'
THEN DataFact + 22
      ELSE
         CASE WHEN TO_CHAR(DataFact +
         20,'DAY')  = 'SUNDAY'
            THEN DataFact + 21
            ELSE DataFact + 20
         END
   END,
'DAY')  AS Zi_Scadenta
FROM FACTURI
```

Visual FoxPro prezintă funcțiile `CDOW` (Character Day Of the Week) și `DOW` (Day Of the Week), rezultatul din figura 4 fiind obținut după cum urmează:

```
SELECT NrFact AS Factura, DataFact, ;
   DataFact + 20  AS Scadenta1, ;
   CDOW(DataFact+20)  AS NumeZi1, ;
   IIF(DOW(DataFact+20)=6, ;
      DataFact+22, ;
      IIF(DOW(DataFact+20)=1, ;
         DataFact+21, ;
         DataFact+20) ) AS Scadenta, ;
   CDOW(IIF(DOW(DataFact+20)=6, ;
      DataFact+22, ;
      IIF(DOW(DataFact+20)=1, ;
         DataFact+21, ;
         DataFact+20) ) ) AS Zi_Scadenta ;
FROM FACTURI
```

Asta-i tot pentru azi. Într-un posibil episod viitor vom vedea cum se pot calcula sporuri de vechime, reduceri și penalități pe baza expresiilor de tip dată calendaristică. Până atunci, toate bune!

Marin Fotache este conferențiar la Catedra de Informatică Economică, UAIC Iași, Facultatea de Economie și Administrarea Afacerilor. Poate fi contactat pe email la: fotache@uaic.ro.