Fiola de SQL (21)

Un an după – *Marin Fotache*

Fiola de astăzi este una tipică de vacanță. După atâtea impozite, taxe și alte probleme care mai de care mai serioase, ne ocupăm luna aceasta de o chestiune care era de actualitate acum un an: alegerile prezedențiale. Nu de la noi, însă, că nu prea am avut de ales, ci sistemul electoral american ne roade pe noi acum. Bine, veți spune, am terminat cu problemele noastre și ne apucăm de probleme altora ? Fiți fără grijă ! Nu cred c-ar interesa pe cineva din SUA ceea ce discutăm noi în acest episod. În plus, alegerele prezidențiale din SUA au fost apropiate calendaristic de cele românești și marcate de oarecare dramatism, dar nu din același cauze. Oricum, gândindu-mă că C.V. Tudor (scuzați cacofonia, nu e din pricina mea) era să ajungă președinte, vă propun o secundă de reculegere. Mulțumesc !

În numărul din ianuarie 2001 al revistei *Intelligent Enterprise* Joe Celko formula o problemă pentru amatorii și profesioniștii de SQL: se dă un sistem electoral (american) în care la alegerile prezidențiale candidează patru oameni ai muncii: Gore, Bush, Nader și Celko. Știind că buletinele de vot pot fi preluate sub forma unei tabele de forma: `BALLOTS {B_Id, Bush, Gore, Nader, Celko}` să se scrie secvența SQL pentru aflarea câștigătorului.

Mai întâi însă să lămurim schema tabelei `BALLOTS`:

```
CREATE TABLE BALLOTS (
    b_id INTEGER NOT NULL PRIMARY KEY,
    bush INTEGER CHECK (bush BETWEEN 1 AND 4),
    gore INTEGER CHECK (gore BETWEEN 1 AND 4),
    nader INTEGER CHECK (nader BETWEEN 1 AND 4),
    celko INTEGER CHECK (celko BETWEEN 1 AND 4)
    )
```

Alegătorii completează buletinul îndicând în dreptul fiecărui candidat o cifră între 1 și 4 ce arată ordinea preferințelor. O linie de genul `(12345, 2, 1, 4, 3)` are următoarea semnificație: alegătorul care a completat buletinul de vot cu identificatorul 12345 îl dorește drept „prezident" pe Gore. Dacă Gore nu întrunește numărul de opțiuni, atunci preferabil ar fi să iasă Bush. În cazul eliminării și lui Gore, și lui Bush, atunci Celko ar fi alesul, iar în ultimă instanță Nader (nu atât alesul, cât culesul).

Figura 1. Primele 20 de linii din VOTURI

b_id	nume	opt
1001	BUSH	2
1001	CELKO	3
1001	GORE	1
1001	NADER	4
1002	BUSH	2
1002	CELKO	3
1002	GORE	1
1002	NADER	
1003	BUSH	2
1003	CELKO	3
1003	GORE	1
1003	NADER	4
1004	BUSH	1
1004	CELKO	3
1004	GORE	2
1004	NADER	4
1005	BUSH	1
1005	CELKO	3
1005	GORE	2
1005	NADER	4

Dacă alegătorul nu vrea cu nici un chip să dea vreo șansă vreunora dintre candidați, atunci nu completează nimic în dreptul lor, ceea ce pentru baza de date reprezintă o valoare `NULL`. De exemplu, dacă alegătorul consideră ca SUA nu trebuie să ajungă pe mâinile lui Celko sau Nader, poate completa un buletin de vot astfel: `(12345, 2, 1, NULL, NULL)`.

Ei, și acum vine partea interesantă. Dacă după numărarea primelor opțiuni din buletinele de vot nici unul dintre candidați nu întrunește majoritatea, atunci se elimină din cursă candidatul cu cele mai puține prime opțiuni. În toate buletinele de vot în care prima opțiune era candidatul eliminat se operează astfel: opțiunea 2 devine opțiunea 1, opțiunea 3 devine 2 și cea cu numărul 4 devine 3.

După această mașinațiune, se verifică dacă în urma redistribuirii voturilor, vreun candidat întrunește majoritatea din voturile rămase valabile (se elimină buletinele în care nici unul dintre candidații rămași nu au vreo opțiune nenulă).

Listing 1. Popularea tabelei ce conține buletinele de vot – sintaxa Oracle/PostgreSQL

```
DELETE FROM ballots ;
INSERT INTO ballots VALUES (1001, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1002, 2, 1, NULL, 3) ;
INSERT INTO ballots VALUES (1003, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1004, 1, 2, 4, 3) ;
INSERT INTO ballots VALUES (1005, 1, 2, 4, 3) ;
INSERT INTO ballots VALUES (1006, 1, 2, 4, 3) ;
INSERT INTO ballots VALUES (1007, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1008, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1009, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1010, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1011, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1012, 4, 2, 1, 3) ;
INSERT INTO ballots VALUES (1013, 4, 2, 1, 3) ;
INSERT INTO ballots VALUES (1014, 4, 2, 1, 3) ;
INSERT INTO ballots VALUES (1015, 4, 2, 1, 3) ;
INSERT INTO ballots VALUES (1016, 1, 2, NULL, 3) ;
INSERT INTO ballots VALUES (1017, 1, 2, NULL, 3) ;
INSERT INTO ballots VALUES (1018, 1, 2, NULL, 3) ;
INSERT INTO ballots VALUES (1019, 1, 2, NULL, 3) ;
INSERT INTO ballots VALUES (1020, 2, 4, 1, 3) ;
INSERT INTO ballots VALUES (1021, 2, 4, 1, 3) ;
INSERT INTO ballots VALUES (1022, 2, 4, 1, 3) ;
INSERT INTO ballots VALUES (1023, 4, 3, 2, 1) ;
INSERT INTO ballots VALUES (1024, 2, 3, NULL, 1) ;
INSERT INTO ballots VALUES (1025, 2, 3, NULL, 1) ;
INSERT INTO ballots VALUES (1026, 4, 2, 3, 1) ;
INSERT INTO ballots VALUES (1027, 2, 4, 1, 3) ;
INSERT INTO ballots VALUES (1028, 2, 1, 4, 3) ;
INSERT INTO ballots VALUES (1029, 2, 1, 4, 3) ;
COMMIT ;
```

Popularea tabelei BALLOTS

În partea de răspunsuri a articolului din Intelligent Enterprise, Joe prezintă clauza `CHECK` pentru ca un buletin să nu conțină erori de genul repetării unei opțiuni. Trecem discret peste acest aspect și ne vom concentra pe cel mai incitant aspect al problemei: aflarea câștigătorului. Pentru a putea testa soluția pe care o vom discuta în continuare, luăm în discuție 29 de buletine de vot – vezi *listing 1*.

Verticalizarea orizontalității

Este foarte dificil de formulat o soluție pe structura tabelei `BALLOTS`. Iar dacă se va întâmpla vreodată ca pe la noi, să fie vreo zece candidați la gloria neamului, atunci situația devine insuportabilă (mă refer la probema SQL...). Astfel încât este de preferat verticalizarea datelor după o structură de genul `VOTURI {b_id, nume, opt}`, unde `b_id` „rămâne" identificatorul buletinului de vot, `nume` reprezintă candidatul iar `opt` – opțiunea – este numărul între 1 și 4 sau `NULL`. Iată comanda (PostgreSQL/Oracle) de creare a tabelei `VOTURI` și primele 20 de linii (din cele 116 ale exemplului nostru) ale acesteia în *figura 1*.

```
CREATE TABLE  voturi AS
    SELECT b_id, 'BUSH' AS nume, bush AS opt
    FROM ballots UNION SELECT b_id, 'GORE', gore
    FROM ballots UNION SELECT b_id, 'NADER', nader
    FROM ballots UNION SELECT b_id, 'CELKO', celko
    FROM ballots ;
```

Listing 2. Popularea tabelei ce conține buletinele de vot – sintaxa Oracle/PostgreSQL

```
UPDATE voturi
SET
/* daca buletinul are toate optiunile NULL, id-ul sau se
    NULL-izeaza */
    b_id =
        CASE WHEN (SELECT SUM(NVL(opt,0)) FROM voturi v2 WHERE
                  v2.b_id=voturi.b_id) = 0
            THEN NULL
            ELSE b_id
        END,
/* urmeaza partea cea mai dificila - modificarea atributului OPT
    */
    opt =
/* daca OPT sau B_ID sunt nule, nu se face mai nimic */
        CASE WHEN opt IS NULL OR b_id IS NULL
            THEN NULL
            ELSE
/* este un buletin de vot in care optiunea 1 este a
    candidatului ce trebuie eliminat    */
        CASE WHEN b_id IN
            (SELECT b_id FROM voturi v2 WHERE opt = 1 AND nume
                IN
               (SELECT nume FROM voturi WHERE  b_id IS NOT NULL
                    AND opt=1
               GROUP BY nume HAVING COUNT(opt)
                     = (SELECT MIN(Nr) FROM
                        (SELECT COUNT(opt) AS Nr FROM voturi
                         WHERE b_id IS NOT NULL AND opt=1
                         GROUP BY Nume) YY1
                       )
                )
            )
            THEN   /* se decrementeaza OPT */
                CASE WHEN opt - 1 = 0
                    THEN NULL
                    ELSE opt - 1
                END
            ELSE
/* linia curenta nu priveste un buletin la care optiunea 1
    reprezinta un candidat de eliminat (nr minim de optiuni 1).
    Daca insa linia se refera la un candidat eliminat, optiunea
    sa se NULL-izeaza */
                CASE WHEN nume IN
                    (SELECT nume FROM voturi
                     WHERE  b_id IS NOT NULL AND opt=1
                     GROUP BY nume HAVING COUNT(opt) =
                       (SELECT MIN(Nr)
                        FROM (SELECT COUNT(opt) AS Nr
                           FROM voturi  WHERE b_id IS NOT NULL
                                AND opt=1
                           GROUP BY Nume) YY2
                       )
                    )
                   THEN NULL
                   ELSE opt
                END
            END
        END
WHERE
    NOT EXISTS
        (SELECT nume FROM voturi WHERE opt = 1
         GROUP BY nume
         HAVING COUNT(DISTINCT b_id) >=
             (SELECT COUNT(DISTINCT b_id) / 2 FROM voturi
              )
        )
;

/* daca nu exista optiunea 1 intr-un pachet, se decrementeaza
    optiunea 2 (sau 3)*/
UPDATE voturi
SET opt = opt - (SELECT MIN(opt)
         FROM voturi v2
         WHERE opt IS NOT NULL AND b_id IS NOT NULL
               AND  voturi.b_id = v2.b_id )
      + 1
WHERE b_id IS NOT NULL
    AND opt IS NOT NULL
    AND b_id IN
        (SELECT b_id
         FROM voturi WHERE b_id IS NOT NULL AND opt IS NOT NULL
         GROUP BY b_id
         HAVING MIN(opt) > 1)
;
```

Numărăm situația inițială a primelor opțiuni:

```
SELECT nume, COUNT(*) AS NrPO
FROM voturi WHERE opt=1 GROUP BY nume
```

Iată și rezultatul: Bush – 7, Celko - 4, Gore - 10 și Nader – 8. Nici unul dintre cei patru candidați nu întrunește majoritatea simplă. Celko are cele mai puține voturi, așa că trebuie eliminat. Cele patru voturi ale sale (buletinele 1023, 1024, 1025 și 1026) trebuie redistribuite astfel: 1023 lui Nader, 1024 lui Bush, 1025 lui Bush și 1026 lui Gore.

Repetabilul script

Beneficiind de noua structură, formulăm o soluție alcătuită din două comenzi `UPDATE` de lungime considerabilă – vezi listing 2. Sintaxa este cea din PostgreSQL 7.1.3, dar funcționează identic și în Oracle 8i înlocuind funcția `COALESCE` cu `NVL`.

Prima dintre cele două comenzi `UPDATE` are ca efect eliminarea din cursă a candidatului cu cel mai mic număr de voturi. Se `NULL`-izează toate liniile corespunzătoare buletinelor în care prima opțiune aparține candidatului de eliminat (cu cele mai puține prime opțiuni), iar celelalte opțiuni corespunzătoare acestor buletine de vot se decrementează cu 1. Dacă după lansarea repetată a scriptului celor două comenzi, este posibil ca să nu mai existe nici o opțiune valabilă, iar buletinul respectiv trebuie scos de la numărătoare, lucru realizat prin chiar primul `SET` care `NULL`-izează identificatoarele acestor buletine.

Prin prezența opțiunii `EXISTS` în `WHERE`-ul primei fraza `UPDATE` se asigură prelucrarea liniilor din `VOTURI` numai atât timp cât nici un candidat nu are majoritatea voturilor valabile. Așa încât scriptul se poate lansa de oricâte ori fără a afecta corectitudinea rezultatului.

Al doilea `UPDATE` are un rol mai degrabă corector. Să presupunem că suntem la a doua execuție a scriptului. Pot exista buletine la care prima opțiune aparține candidatului tocmai eliminat, iar a doua a candidatului eliminat la prima execuție a scriptului. În această situație ar exista riscul ca buletinul respectiv să rămână după cele două execuții fără o primă opțiune, deși alegătorul a completat toate „căsuțele“ cu numere de la 1 la 4. Astfel încât acest `UPDATE` va decrementa opțiunea cea mai mică până se va ajunge la 1.

Comentarii dezinteresate

După prima execuție a celor două comenzi, obținem următoarea configurație a primelor opțiuni: Bush – 9, Gore - 11 și Nader – 9. Aici apare un impas, deoarece Joe nu evocă nici un criteriu de balotaj. Atunci când numărul buletinelor de vot este mare, șansele apariției a doi candidați cu același număr de voturi este infim, deci putem considera soluția acceptabilă.

Următoarea lansare a scriptului va NULL-iza primele opțiuni ale lui Nader și ale lui Bush și va pasa lui Bush patru voturi care reprezintă a doua opțiune a voturilor lui Nader, astfel încât victorios este Gore. Cer scuze administrației americane, dar n-a fost cu intenție.

După execuția repetată a scriptului (număr execuțiilor poate fi cel mult egal cu numărul candidaților minus 1), obținem victoriosul prin interogarea:

```
SELECT nume, COUNT(*) AS Voturi
FROM voturi WHERE OPT=1 GROUP BY NUME
HAVING COUNT(*) >= ALL
    (SELECT COUNT(*) FROM voturi WHERE OPT=1 GROUP BY NUME)
```

Ca o concluzie, soluția prezentată ar putea fi ușor modificată pentru a funcționa perfect dacă am ști criteriul de balotaj aplicabil la eliminarea unuia dintre candidații cu număr egal de opțiuni, atunci când acest număr este cel mai mic prin comparație cu cele ale celorlalți candidați (it sounds well, n'est pas?). Cu oarecare îngăduință, vom presupune că asemenea situații sunt aproape imposibile. Plus faptul că sunt și detectabile, pentru a evita frauda electorală SQL-istă.

Marin Fotache este conferențiar dr. la Catedra de Informatică Economică, UAIC Iași, Facultatea de Economie și Administrarea Afacerilor. Poate fi contactat pe email la: fotache@uaic.ro. ■ 98