Fiola de SQL (8)

Corelez, deci, în general, EXISTS

Interogările corelate reprezintă piatra de încercare a oricărui SQL-ist ce bate sau se milogeşte la porţile profesionismului. Logica acestora este una dintre cele mai dificile, cu atât mai mult cu cât corelarea poate fi dublă, triplă… Deşi nu deţine exclusivitatea, operatorul cel mai apropiat de acest subiect este EXISTS.

Ca şi IN, EXISTS permite legarea frazei SELECT principale de una sau mai multe sub-consultări şi astfel formularea unor interogări complexe. Utilizând operatorul IN, în tabela rezultat se extrag acele linii din tabela (tabelele) din fraza SELECT principală care satisfac o condiţie relativă la liniile din sub-consultare. Cu EXISTS se extrag linii tot din tabela din fraza SELECT principală, dar pe baza existenţei a cel puţin o linie în tabela (tabelele) din fraza SELECT secundară (subconsultare), care satisface (satisfac), pe lângă restricţii proprii subconsultării, şi o condiţie ce face trimitere la tabele din fraza principală.

Logica sa este la nivel de linie şi nu la nivel de ansamblu (seturi de înregistrări), ceea ce Joe Celko consideră o deviere de la spiritul relaţional. Pe de altă parte, Sheryl Larsen pledează pentru folosirea lui EXISTS în detrimentul IN-ului pe considerentul optimizării. IN presupune crearea unor tabele temporare ce, uneori, reclamă timp de execuţie sensibil mai mare prin comparaţie cu logica tuplu-cu-tuplu EXISTS-enţială.

Chiar dacă este denormalizată, şi constituie probabil un prost exemplu pentru tineretul SQList din ziua de azi, vom folosi tot tabela VINZARI_CARTI prezentată în premieră în materialul dedicat funcţiilor OLAP din SQL99 şi Oracle 8i2 din actualul număr al PCReport şi cel precedent. Pentru cei care au avut norocul să nu parcurgă cele două articole, prezentăm comanda de creare (în Oracle) a tabelei şi, în figură, conţinutul acesteia.

```
CREATE TABLE vinzari_carti (
        NrFact DECIMAL(6) NOT NULL,
        DataFact DATE NOT NULL,
        Client VARCHAR(15),
        Loc VARCHAR(15),
        Jud CHAR(2),
        ISBN CHAR(9),
        Cantit DECIMAL (6),
        PretUn DECIMAL (8),
        ValTotala DECIMAL (14)   );
```

NRFACT	DATAFACT	CLIENT	LOC	JUD	ISBN	CANTIT	PRETUN	VALTOTALA
1111	01-NOV-00	CLIENT1	Iasi	IS	ISBN1	500	75000	44625000
1111	01-NOV-00	CLIENT1	Iasi	IS	ISBN2	150	60000	10710000
1111	01-NOV-00	CLIENT1	Iasi	IS	ISBN3	225	125000	33468750
1112	01-NOV-00	CLIENT2	Pascani	IS	ISBN1	100	80000	9520000
1112	01-NOV-00	CLIENT2	Pascani	IS	ISBN2	300	55000	19635000
1112	01-NOV-00	CLIENT2	Pascani	IS	ISBN3	50	115000	6842500
1112	01-NOV-00	CLIENT2	Pascani	IS	ISBN4	175	40000	8330000
1113	04-NOV-00	CLIENT1	Iasi	IS	ISBN1	100	80000	9520000
1113	04-NOV-00	CLIENT1	Iasi	IS	ISBN4	275	40000	13090000
1114	04-NOV-00	CLIENT3	Suceava	SV	ISBN1	500	74000	44030000
1114	04-NOV-00	CLIENT3	Suceava	SV	ISBN2	150	60000	10710000
1114	04-NOV-00	CLIENT3	Suceava	SV	ISBN3	100	120000	14280000
1114	04-NOV-00	CLIENT3	Suceava	SV	ISBN4	225	39000	10442250
1115	05-NOV-00	CLIENT2	Pascani	IS	ISBN2	75	61000	5444250
1115	05-NOV-00	CLIENT2	Pascani	IS	ISBN4	100	40000	4760000
1116	05-NOV-00	CLIENT4	Iasi	IS	ISBN1	500	76000	45220000
1116	05-NOV-00	CLIENT4	Iasi	IS	ISBN2	60	65000	3867500
1116	05-NOV-00	CLIENT4	Iasi	IS	ISBN4	50	45000	2677500

Conţinutul tabelei denormalizate VINZARI_CARTI

- *Care sunt clienţii din acelaşi judeţ ca şi Client 4 ?*

Cu ajutorul operatorului IN se poate redacta o soluţie ce foloseşte subconsultări necorelate:

```
SELECT DISTINCT Client
FROM vinzari_carti
WHERE Jud IN
      (SELECT DISTINCT Jud
       FROM vinzari_carti
       WHERE Client = 'CLIENT4')
```

Execuţia acestei fraze SELECT începe cu subconsultarea al cărei rezultat este o tabelă intermediară ce prezintă o singură coloană – Jud şi o singură linie în care apare valoarea ‘IS’. În etapa a doua, rezultatul subconsultării este folosit ca argument al SELECT-ului principal, extrăgându-se din VINZARI_CARTI toate liniile în care valoarea atributului Jud este ‘IS’.

Cu EXISTS se poate formula o soluţie bazată două SELECT-uri corelate:

```
SELECT DISTINCT Client
FROM vinzari_carti v1
WHERE EXISTS
      (SELECT 1
       FROM vinzari_carti v2
       WHERE  v2.Client = 'CLIENT4' AND v1.Jud=v2.Jud)
```

Interogarea foloseşte două instanţe, V1 şi V2, ale tabelei VINZARI_CARTI şi se derulează în 18 etape, câte luna pentru fiecare linie din V1:

Faza 1:

- se “încarcă” prima linie din V1: (1111, ‘01-NOV-00’, ‘CLIENT1’, ‘Iasi’, ‘IS’, ‘ISBN1’, 500, 75000, 44625000)
- se verifică dacă există măcar o linie în V2 în care v2.Client = 'CLIENT4' şi v2.Jud=’IS’ (‘IS’ este valoarea atrubutului v1.Jud). Există ! Este linia 16 din V2. Prin urmare,
- valoarea curentă a v1.Client (‘CLIENT1’) se include în rezultat.

….

Faza 10:

- se "încarcă" a zecea linie din v1: (1114, '04-NOV-00', 'CLIENT3', 'Suceava', 'SV', 'ISBN1', 500, 74000, 44030000)
- se verifică dacă există măcar o linie în V2 în care v2.Client = 'CLIENT4' şi v2.Jud='SV'. Nu există ! Prin urmare,
- valoarea curentă a v1.Client ('CLIENT3') nu se include în rezultat.

….

Ceea ce face atât de dificil acest operator şi, implicit, interogarea corelată este tocmai includerea în rezultat a unei linii din V1 nu direct, pe baza unui predicat din clauza WHERE (şi/sau HAVING) proprie, ci dacă există măcar o linie în V2 care îndeplineşte condiţia specificată în subconsultare.

Prin EXISTS se poate realiza şi *intersecţia* a două relaţii: *Care sunt clienţii care au cumpărat cel puţin cărţile ISBN1 şi ISBN3*.

- Soluţia 1:

```
SELECT Client
FROM vinzari_carti
WHERE ISBN='ISBN1'
INTERSECT
   SELECT Client
   FROM vinzari_carti
   WHERE ISBN='ISBN3'
```

- Soluţia 2:

```
SELECT DISTINCT Client
FROM vinzari_carti v1
WHERE ISBN='ISBN1' AND EXISTS
   (SELECT 1
    FROM vinzari_carti v2
    WHERE ISBN='ISBN3' AND v1.Client=v2.Client)
```

Deşi pare (şi chiar este) mai complicată, a doua soluţie are un grad mai mare de "portabilitate", deoarece destule SGBD-uri (precum VFP) nu au încă implementat operatorul INTERSECT.

- *Pentru care dintre clienţi s-au întocmit numai două facturi ?*

Este o problemă banală pentru care se pot formula soluţii interesante. Fireşte, cea mai la îndemână variantă este:

```
SELECT Client, COUNT(DISTINCT NrFact) AS NrFacturi
FROM vinzari_carti
GROUP BY Client
HAVING COUNT(DISTINCT NrFact) = 2
```

Interogarea pe care o consider cea mai interesantă nu foloseşte însă, pentru corelarea subconsultării cu fraza SELECT principală, nici EXISTS, nici IN. Din acest punct de vedere, reiese că titlul prezentei fiole e puţin exagerat, deoarece *pot să corelez şi fară să EXISTS* !

```
SELECT DISTINCT Client
FROM vinzari_carti v1
WHERE 2 =
        (SELECT COUNT(DISTINCT NrFact)
         FROM vinzari_carti v2
        WHERE v1.Client=v2.Client)
```

Logica interogării:

Faza 1:

- se "încarcă" prima linie din V1: (1111, '01-NOV-00', 'CLIENT1', 'Iasi', 'IS', 'ISBN1', 500, 75000, 44625000)
- se numără, prin funcţia COUNT, valorile distincte ale atributului NrFact din V2 pentru liniile în care v2.Client = 'CLIENT1'. Rezultatul este 2.
- se compară 2 din SELECT-ul principal cu rezultatul subconsultării corelate; cele două valori fiind egale, 'CLIENT1' se include în rezultat.

…

Faza 10:

- se "încarcă" a zecea linie din v1: (1114, '04-NOV-00', 'CLIENT3', 'Suceava', 'SV', 'ISBN1', 500, 74000, 44030000)
- se numără, prin funcţia COUNT, valorile distincte ale atributului NrFact din V2 pentru liniile în care v2.Client = 'CLIENT1'. Rezultatul este 1.
- se compară 2 din SELECT-ul principal cu rezultatul subconsultării corelate;$2 \neq 1$, deci valoarea curentă a v1.Client ('CLIENT3') nu se include în rezultat.

….

- *Care sunt cele mai mari cinci valori care apar în facturile emise ?*

Formulăm o soluţie care seamănă oarecum cu precedenta, deşi problema este cu mult mai complicată. Iniţial, am vrut să o "plasez" în fiola trecută, cea dedicată clasamentelor. Am reuşit însă să mă abţin, fiind nimerită în prezentarea interogărilor corelate.

```
SELECT ValTotala
FROM vinzari_carti v1
WHERE 5 >
        (SELECT COUNT(v2.ValTotala)
        FROM vinzari_carti v2
        WHERE v2.ValTotala > v1.ValTotala)
ORDER BY ValTotala DESC
```

Ideea este grozav de ingenioasă (păcat că nu este a mea): se parcurge linie cu linie prima instanţă a tabelei VINZARI_CARTI – V1. Pentru fiecare linie de V1 se numără, în a doua instanţă a VINZARI_CARTI – V2, câte linii prezintă valoarea totală mai mare

decât cea de pe linia curentă din V1. Dacă numărul calculat prin COUNT(*) este mai mic decât 5, atunci în rezultat se extrage valoarea LF1.ValTotala.

Păcat că VFP nu suportă aşa găselniţă, sau cel puţin aşa zice mesajul cu care suntem întâmpinaţi la execuţia acestei interogări.

Nu am încheiat aici problematica interogărilor corelate. Vom vedea luna viitoare cum se derulează interogările dublu corelate şi câteva chestiuni legate de diviziunea relaţională.

Bibliografie

Larsen, S. – POWERFUL SQL: BEYOND THE BASICS, DB2 Magazine On Line, Winter 1996

Erată

În chiar prima propoziţie a fiolei proaspăt expirate, încercând să suprind o porţiune cât mai semnificativă din complexul peisaj al muzicii actuale, am pus faţă în faţă doi titani contemporani: Puff Daddy şi Nelu Vijelie. Trecând întâmplător pe lângă o tarabă cu muzică de profil, am descoperit, cu mare tulburare, că artistul evocat se numeşte Vali Vijelie, iar nu Nelu Vijelie. Drept care mă grăbesc să cer scuze artistului, casei de casete, precum şi celor de la Yes, Camel, Pink Floyd, EL&P, Beatles…

Marin Fotache