SELECT FROM AUSTIN, TEXAS

Joe Celko over SQL en andere database-zaken

Denken in sets

Alleen als u wiskundestudent bent, weet u waarschijnlijk hoe sets echt werken. De belangrijkste conceptuele bijdrage van de set-theorie is dat u afgeronde verzamelingen als één item kunt behandelen, in plaats van de verzameling onderdeel voor onderdeel te verwerken. Wiskundig gezien geldt dit ook voor oneindige sets. Bij SQL is het verdraaid lastig om genoeg geld te krijgen voor de aanschaf van oneindige hoeveelheden diskruimte.

Op 7 september 2001 stuurde Henri Tuthill mij een vraagstuk, waar hij zijn tanden op stuk beet. En hij zond ook goede specificaties. Het probleem behandelt een tabel met laboratoriumresultaten, met de volgende omschrijving.

- In een kolom, 'samres' geheten, komen drie typen resultaten voor, die worden voorgesteld als i, n en p.
- Voor nadere analyse zijn de wetenschappers alleen geïnteresseerd in selectie van de i- of p-rijen.
- De tabel bevat 800.000 rijen en meer kolommen, maar dit zijn de belangrijke, en we bouwen ter lering een kleine versie van de tabel.

```
CREATE TABLE Tests (sample_nbr INTEGER NOT NULL,
samres CHAR(1) NOT NULL DEFAULT 'n'
      CHECK (samres IN ('i', 'p', 'n')),
inhib INTEGER NOT NULL,
PRIMARY KEY (sample_nbr, samres, inhib);

INSERT INTO Tests VALUES (1, 'i', 79);
  INSERT INTO Tests VALUES (1, 'p',
63); INSERT INTO Tests VALUES (2, 'i', 46);
  INSERT INTO Tests VALUES (2,
'i', 48); INSERT INTO Tests VALUES (2, 'n', 15);
  INSERT INTO Tests VALUES
 (3, 'n', 79); INSERT INTO Tests VALUES (3, 'p', 70);
  INSERT INTO Tests
VALUES (3, 'p', 72);
```

Iemand anders had een PL/SQL-routine geschreven om dit probleem binnen 24 uur op te lossen. Henri kwam direct op de gedachte om net zo'n VIEW te maken.

```
CREATE VIEW Tests_summary
(sample_nbr, samres, inhib)
   AS SELECT sample_nbr,
samres, MAX(inhib)
     FROM Tests
   WHERE samres IN ('p', 'i');
   GROUP BY sample_nbr, samres;
```

Zo komen we tot de volgende set resultaten:

sample_nbr	samres	inhib
1	i	79
1	p	63
2	i	48
3	p	72

Maar hier zit een akelige rij in (1, i, 79). De sample-groep #1 is een mix van p- en i-waarden, dus moeten we alleen de rij (1, p, 63) terug zien te krijgen.

Ik geef toe dat ik een valse start maakte, maar omdat dit nu eenmaal mijn column is, vertel ik u niet waardoor dat kwam. Wat we nodig hadden, was een analyse van de data in termen van subsets. We hebben alle rijen met een n weggehaald, dus weten we dat de sample-nummers de volgende subsets vormen:

- alle p's
- alle i's
- de p's en de i's

Hoe komen we erachter of de groep voldoet aan de criteria? Dat kan bijvoorbeeld met de cursor; scan elke rij en loop de gegevens na. Dit is *sequential thinking*. Probeer in plaats daarvan tot de gewenste subset-karakteristiek te komen, zonder naar details van

de subset te hoeven kijken. De volgende predicaten doen dat:

```
(MIN(samres) = 'i' AND MAX(samres) = 'i')
```

betekent dat de set alleen uit i's bestaat.

```
(MIN(samres) = 'p' AND MAX(samres) = 'p')
```

betekent dat de set alleen uit p's bestaat.

```
(MIN(samres) = 'i' AND MAX(samres) = 'p')
```

betekent dat de set gemengd is. Nu brengen we deze predicaten onder in HAVING-clausules, om zo te komen tot een UNION-ed resultaat:

```
  SELECT sample_nbr, 'i',  MAX(inhib) – all i's
    FROM Tests AS T1
  WHERE samres IN ('i', 'p')
  GROUP BY sample_nbr HAVING (MIN(samres) = 'i' AND
MAX(samres) = 'i') UNION
 ALL – all p's
  SELECT sample_nbr, 'p',  MAX(inhib)  @l:
    FROM Tests AS T1
  WHERE samres IN ('i', 'p')
  GROUP BY sample_nbr HAVING (MIN(samres) = 'p' AND
MAX(samres) = 'p') UNION
 ALL  – mixed
   SELECT sample_nbr, 'p',  MAX(inhib)
    FROM Tests AS T1
   WHERE samres IN ('i', 'p')
   GROUP BY sample_nbr HAVING (MIN(samres) = 'i' AND
MAX(samres) = 'p');
```

Maar het werkt nog steeds niet; de laatste regel is nog steeds de i's niet kwijt voordat de MAX () berekend wordt. Laten we dat voor nu even vergeten. We bekijken de eerste twee regels, en wat opvalt is dat ze bijna identiek zijn; feitelijk kunnen ze in elkaar geschoven worden tot het predicaat (MIN(samres) = MAX(samres)) om aan te geven dat het of allemaal i's of allemaal p's zijn.

Als u een UNION ziet die gebouwd is op subsets van dezelfde basistabel, kunt u bijna altijd de UNION vervangen door een enkele query. Wat we eigenlijk willen is de hele rij in de oorspronkelijke tabel, niet alleen het sample-nummer.

Verbetering #1: Schuif de eerste twee zaken in elkaar en herschrijf aldus de UNION.

```
SELECT sample_nbr, samres, inhib
 FROM Tests AS T1
WHERE samres IN ('i', 'p')
  AND sample_nbr
    IN (SELECT sample_nbr
         FROM Tests AS T2
         WHERE samres IN ('i', 'p')
         GROUP BY sample_nbr
        – all i's or all p's
         HAVING (MIN(samres) = MAX(samres)
              AND MAX(inhib) = T1.inhib)
        – mixed
         OR (MIN(samres)  MAX(samres)
              AND MAX(CASE WHEN T2.samres = 'p'
                      THEN inhib ELSE NULL END)
    = T1.inhib));
```

De Case-regel in de laatste MAX () verwijdert de i's en raakt de ongewenste rij kwijt. Dat ziet er goed uit, maar er is nog een andere heuristiek; hebt u een OR-ed predicaat, dan kunt u die vervangen door één enkele Case-regel.

Verbetering #2: schuif de zaken verder in elkaar tot één Case-regel:

```
SELECT sample_nbr, samres, inhib
   FROM Tests AS T1
WHERE sample_nbr
    IN (SELECT sample_nbr
        FROM Tests AS T2
        WHERE samres IN ('i', 'p')
        GROUP BY sample_nbr
        HAVING (CASE WHEN MIN(samres) = MAX(samres)
               THEN MAX(inhib)
                 ELSE MAX(CASE WHEN T2.samres = 'p'
                          THEN inhib
                            ELSE NULL END)
               END)
          = T1.inhib);
```

Joe Celko (www.celko.com) is werkzaam bij Data Junction in Austin als consultant en trainer. Hij is tevens lid van het ANSI X3H2 Database Standards Committee en auteur van diverse boeken over SQL. Als SQL-specialist schrijft hij behalve voor Database Magazine voor het blad *Intelligent Enterprise* (voorheen DBMS).