SELECT FROM AUSTIN, TEXAS

Joe Celko over SQL en andere database-zaken

Neem de tijd!

Berekeningen met de factor tijd zijn ingewikkeld. Het eerste probleem dat we tegenkomen is dat de rekeneenheden niet decimaal zijn, en een enorme verscheidenheid te zien geven. En nog helemaal los hiervan, tijd is een speciaal gegeven. Het is continu en niet een afgerond vaststaand getal, het is een dimensie die elke gebeurtenis betreft. En er zijn ook taalkundige verwarringen.

We kunnen het begrip tijd gebruiken als aanduiding van een bepaald punt in een continu proces (de trein komt om 15.00 uur aan), we kunnen er tijdsduur mee bedoelen (de treinreis duurt twee uur) maar ook een periode (de treinreis duurt van 13.00 uur tot 15.00 uur).

Als we tijd als gegeven vast leggen in een database, gebruiken we vaak niet het juiste tijdsbegrip. Bijvoorbeeld, een gebeurtenis heeft een begin- en een eindtijd, maar de meeste programmeurs leggen niet beide gegevens vast, vaak is alleen de begintijd vastgelegd. Dit is een probleem van de Domain-Key normalisatie; een rij in een tabel moet een enkelvoudig feit bevatten, dus als we het verkeerde Domain hebben, hebben we ook niet het gehele gegeven.

Het probleem met het vastleggen van tijdperioden is dat ze elkaar kunnen overlappen. We bekijken de volgende tabel.

```
CREATE TABLE Events
(event CHAR(2) NOT NULL,
start_date DATE NOT NULL,
end_date DATE NOT NULL,
CHECK (start_date < end_date),
PRIMARY KEY (event, start_date, end_date));

INSERT INTO Events VALUES ('A1', '1997-05-01', '1998-07-01');
INSERT INTO Events VALUES ('A1', '1998-02-01', '1999-11-30');
INSERT INTO Events VALUES ('A1', '1999-01-01', '1999-12-31');
INSERT INTO Events VALUES ('A1', '1999-03-01', '1999-11-30');
INSERT INTO Events VALUES ('A1', '2000-02-01', '2000-12-31');
INSERT INTO Events VALUES ('A1', '2000-03-01', '2000-11-28');
INSERT INTO Events VALUES ('A1', '2002-02-01', '2002-04-29');
INSERT INTO Events VALUES ('A1', '2002-03-01', '2002-11-28');
```

Het meest voor de hand liggende is om de perioden weg te halen, die volledig binnen een andere periode liggen.

```
DELETE FROM Events
WHERE EXISTS
(SELECT *
FROM Events AS E1
WHERE E1.event = Events.event
AND E1.start_date < Events.start_date
AND E1.end_date > Events.end_date);
```

Alleen is nu nog niet voorzien in de situatie dat twee gebeurtenissen een gemeenschappelijke begin- en eind-datum hebben. We passen de code enigszins aan.

```
DELETE FROM Events
WHERE EXISTS
(SELECT *
FROM Events AS E1
WHERE E1.event = Events.event
AND ((E1.start_date <= Events.start_date
AND E1.end_date > Events.end_date)
OR (E1.start_date < Events.start_date
AND E1.end_date >= Events.end_date));
```

Of als alternatieve oplossing:

```
DELETE FROM Events
WHERE EXISTS
(SELECT *
FROM Events AS E1
WHERE Events.start_date
BETWEEN E1.start_date + INTERVAL 1 DAY
AND E1.end_date
AND Events.end_date
BETWEEN E1.start_date
AND E1.end_date - INTERVAL 1 DAY);
```

Er blijven nu twee soorten van tijdsperioden over om vast te leggen; opeenvolgende perioden en overlappende perioden.

Of zouden dit twee speciale gevallen zijn van een meer algemene situatie? Hieronder volgt een query die een tweetal perioden in één tijdsperiode samenvoegt en dan in de tabel zet. Als de nieuwe periode is vastgelegd in de tabel, kunnen we het overbodige oorspronkelijke paar elimineren.

```
INSERT INTO Events
SELECT E1.event, E1.start_date, MAX(E2.end_date)
FROM Events AS E1,
Events AS E2,
WHERE E1.event = E2.event
AND E2.start_date BETWEEN E1.start_date
AND E1.end_date GROUP BY E1.event, E1.start_date;
```

Dit werkt toch niet helemaal zoals we willen, we krijgen teveel begintijden. Bekijk de gebeurtenissen van 2002 in deze tabel;

```
event  start_date  end_date
============================
A1     1997-05-01  1999-11-30
A1     1998-02-01  1999-12-31
A1     1999-01-01  1999-12-31
A1     1999-03-01  1999-11-30
A1     2000-02-01  2000-12-31
A1     2000-03-01  2000-11-28
A1     2002-02-01  2002-11-28
A1     2002-03-01  2002-11-28
```

We proberen het opnieuw, en gebruiken het eerste resultaat als een afgeleide tabel.

```
SELECT X.event, MIN (X.start_date), X.end_date FROM
(SELECT E1.event, E1.start_date, MAX(E2.end_date)
FROM Events AS E1, Events AS E2
WHERE E1.event = E2.event
AND E1.start_date <= E2.start_date
AND E2.start_date BETWEEN E1.start_date
AND E1.end_date GROUP BY E1.event, E1.start_date)
AS X (event, start_date, end_date)
GROUP BY X.event, X.end_date;
event  start_date  end_date
============================
A1     1997-05-01  1999-11-30
A1     1998-02-01  1999-12-31
A1     2000-03-01  2000-11-28
A1     2000-02-01  2000-12-31
A1     2002-02-01  2002-11-28
```

Dit is veel beter, maar kijk eens naar de gegevens van 2000. Er is nog een periode over, die volledig binnen een andere valt.
We hebben nu code om overbodige rijen weg te halen, maar het zou makkelijker zijn om het in een keer te doen.

```
SELECT X.event, MIN (X.start_date), X.end_date FROM
(SELECT E1.event, E1.start_date, MAX(E2.end_date)
FROM Events AS E1, Events AS E2
WHERE E1.event = E2.event
AND E1.start_date <= E2.start_date
AND E2.start_date BETWEEN E1.start_date
AND E1.end_date GROUP BY E1.event, E1.start_date)
AS X (event, start_date, end_date)
GROUP BY X.event, X.end_date
HAVING NOT EXISTS
(SELECT *
FROM Events AS E3
WHERE E3.start_date
BETWEEN MIN (X.start_date) + INTERVAL 1 DAY
AND X.end_date
AND E3.end_date
BETWEEN MIN (X.start_date)
AND X.end_date - INTERVAL 1 DAY);
```

En dat geeft ons wat we willen zien.

```
event  start-date  end_date
============================
A1     1997-05-01  1999-11-30
A1     2000-03-01  2000-11-28
A1     2002-02-01  2002-11-28
```

Zoals ik boven al schreef; berekeningen met tijd zijn ingewikkeld. ●

Joe Celko (www.celko.com) is onafhankelijk consultant en lid van het ANSI X3H2 Database Standards Committee. Hij is auteur van diverse boeken over SQL. Als SQL-specialist schrijft hij behalve voor Database Magazine voor het blad *Intelligent Enterprise* (voorheen DBMS).

De laatste bijdrage

Dit is voorlopig de laatste column van Joe Celko. Hij heeft dan ook direct zijn kans gegrepen en is verhuisd naar Salt Lake City..... De redactie dankt Joe Celko voor zijn gewaardeerde en leerzame bijdragen aan Database Magazine in de afgelopen jaren.