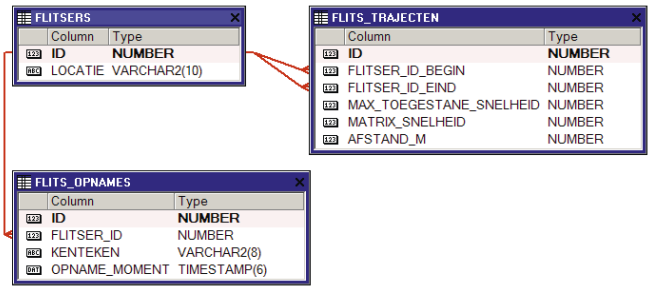
Puzzelen met SQL

Trajectcontrole: van A naar Bekeuring

Op twaalf plaatsen in Nederland worden trajectcontroles uitgevoerd. Een trajectcontrole is een snelheidsmeting die niet op één punt wordt gedaan, maar op verschillende punten over een langere afstand. Aan het begin van het traject wordt er een opname gemaakt, net als op het einde van het traject. Aan de hand van deze gegevens kun je de gemiddelde snelheid uitrekenen. Tenminste, als je weet wat de afstand is tussen begin- en eindpunt.

Voor deze puzzel hanteren we – gefingeerde - gegevens van trajectcontroles. Hiervoor maken we gebruik van het volgende datamodel.



Van iedere camera is de locatie bekend. Voor deze puzzel maken we gebruik van de naam van een locatie. We hadden hier ook de echte geografische locatie kunnen kiezen en deze als Spatial-type kunnen opnemen. Het werken met Spatial-data is een vak op zich en nog steeds geen gemeengoed voor de gemiddelde Oracle-ontwikkelaar. Het zou voor dit artikel te veel ruimte in beslag nemen om alle Spatial syntax uit te leggen. Wellicht besteden we hier in een toekomstige puzzel aandacht aan.
Camera's boven de weg nemen deel aan een meting aan het begin van een traject en aan het einde daarvan. Voor de trajectcontrole is het van belang om de afstand tussen de twee camera's te weten. Vandaar dat de afstand in meters erbij staat. Nu schijnt het zo te zijn dat de trajectcontrole rekening houdt met de aangegeven snelheid op de matrix borden. Dus wees gewaarschuwd, de matrix borden boven de snelweg geven een maximum snelheid aan. Dit in tegenstelling tot wat je geleerd hebt toen je je theorieexamen deed. Daarbij werd verteld dat het een adviessnelheid was. De aangegeven snelheid dient dus opgevolgd te worden.
Als laatste, maar zeker niet onbelangrijk, de opnamen zelf. Van iedere auto die een camera passeert, wordt een opname gemaakt, met de nodige informatie zoals tijdstip van passeren en het kenteken.

Nu zijn alle gegevens bekend. We weten nu wie waar is gepasseerd en op welk tijdstip. Daarvan kan de gemiddelde snelheid over de bekende afstand worden afgeleid en aan de hand van het kenteken kunnen de bekeuringen worden verstuurd. Het doel van de trajectcontrole is natuurlijk niet om zoveel mogelijk bekeuringen te kunnen innnen, maar om het veiliger te maken op de weg.

Nu zou je natuurlijk kunnen bedenken dat het kenteken alleen niet genoeg is. Je kunt tenslotte meerdere keren over dezelfde weg rijden op verschillende tijden. Op de website van het Openbaar Ministerie staat echter dat gegevens van de trajectcontrole alleen worden opgeslagen indien er sprake is van een verkeersovertreding. De gegevens worden dan doorgestuurd naar het CJIB en de bekeuring valt automatisch op de mat. Het is dan nog maar een kleine stap naar het SMS-bericht waarin staat dat je bekeuring reeds van je bankrekening is afgeschreven. Dat is dan €50 per ontvangen bericht.

In deze puzzel hebben we niet alleen gegevens in de tabellen staan van snelheidsovertreders, maar van alles wat.

De puzzels

1) Bereken de gemiddelde snelheid.

Laten we maar beginnen met de meest voor de hand liggende puzzel: het bepalen van de gemiddelde snelheid.
Om de gemiddelde snelheid te kunnen berekenen moeten we

gebruik maken van gegevens uit de volgende tabellen: Flits Trajecten en de Flits Opnames. Dit omdat er twee relaties tussen de tabellen liggen: één voor het registreren van het begin van het traject en ééntje voor het passeren van het einde van het traject. Omdat de afstand tussen de camera's bekend is, kan eenvoudig de gemiddelde snelheid berekend worden.

```
SQL> select (select f.locatie
  2             from flitsers f
  3            where f.id = opnbegin.flitser_id
  4          ) locatie
  5        , opnbegin.kenteken
  6        , tjt.afstand_m
  7        , (opneind.opname_moment - opnbegin.opname_moment) tijdsduur
  8        , tjt.max_toegestane_snelheid
  9        , tjt.afstand_m /
 10            (extract(second from (opneind.opname_moment - opnbegin.
opname_moment)) +
 11            (extract(minute from (opneind.opname_moment - opnbegin.
opname_moment))) * 60 )
 12            * 3.6 km_per_uur
 13    from flits_trajecten tjt
 14    join flits_opnames opnbegin
 15      on (tjt.flitser_id_begin = opnbegin.flitser_id)
 16    join flits_opnames opneind
 17      on (tjt.flitser_id_eind = opneind.flitser_id)
 18   where opnbegin.kenteken = opneind.kenteken
 19  ;

LOCATIE    KENTEKEN  AFSTAND_M TIJDSDUUR                    MAX_
TOEGESTANE_SNELHEID KM_PER_UUR
---------- -------- ---------- --------------------------- -------------
---------- ----------
A4-1       XD-ZN-44       5000 +000000000 00:02:00.000000
120         150
A10-1      93-JN-JT       1000 +000000000 00:00:25.000000
80         144
A12-1      XD-ZN-44       3000 +000000000 00:17:00.000000
80 10.5882353
A12-3      93-JN-JT       1000 +000000000 00:00:30.000000
100         120
```

De locatie van de flitsers wordt bepaald door gebruik te maken van een Scalar Subquery. Een Scalar Subquery is een query die maar één waarde teruggeeft. Indien deze Scalar Subquery geen waarde zou opleveren, dan wordt het resultaat NULL.
De opname wordt in de tabel geregistreerd in een kolom van het type TIMESTAMP. De TIMESTAMP behoort bij de DATE familie, maar heeft een veel grotere precisie dan de DATE. Waar de DATE tot op seconden nauwkeurig is, kan de TIMESTAMP een nauwkeurigheid hebben tot op nano-seconden nauwkeurig.
Het verschil tussen twee DATE variabelen is het verschil in dagen, een numerieke waarde. Het verschil tussen twee TIMESTAMP variabelen is een INTERVAL. In bovenstaand overzicht is in de kolom 'tijdsduur' een INTERVAL te zien. Hier staat heel nauwkeurig hoe lang het heeft geduurd om van het eerste naar het laatste camerastandpunt te komen. Voor de eerste rij uit het resultaat is dat twee minuten, de tweede rij 25 seconden en zo verder.

Van deze INTERVAL hebben we de minuten en de seconden nodig. Deze halen we uit de INTERVAL met behulp van de EXTRACT functie. Het gebruik hiervan is te vinden in regel 10 en 11 van de query. Met de EXTRACT functie kun je de afzonderlijke delen van een INTERVAL halen.
De afstand is bekend, evenals de tijd die een auto daarover doet. Nu is het dus eenvoudig uit te rekenen wat de gemiddelde snelheid is. Met behulp van de EXTRACT functies berekenen we eerst het aantal meters per seconde en vervolgens de kilometers per uur.

In het overzicht is nu duidelijk te zien wie er een bekeuring thuis kan verwachten. Degene die op de A12 reed heeft 17 minuten over een afstand van 3000 meter gedaan, zeker geen snelheidsrecord. Die zal wel last van file hebben gehad.

2) Wat is de gemiddelde afwijking ten opzichte van de toegestane snelheid?

Voor het beantwoorden van deze vraag, gaan we gebruik maken van de vorige query. Om het resultaat van de vorige query te kunnen gebruiken, kunnen we deze in een inline view plaatsen, in de FROM clause. Maar onze voorkeur gaat toch uit naar de WITH clause. In de documentatie te vinden onder Subquery Factoring.

```
SQL> with t as
  2  (select opnbegin.kenteken
  3        , tjt.id traject_id
  4        , tjt.afstand_m
  5        , (opneind.opname_moment - opnbegin.opname_moment) tijdsduur
  6        , tjt.max_toegestane_snelheid
  7        , tjt.afstand_m /
  8          (extract(second from (opneind.opname_moment - opnbegin.
opname_moment)) +
  9          (extract(minute from (opneind.opname_moment - opnbegin.
opname_moment))) * 60 )
 10          * 3.6 km_per_uur
 11    from flits_trajecten tjt
 12    join flits_opnames opnbegin
 13      on (tjt.flitser_id_begin = opnbegin.flitser_id)
 14    join flits_opnames opneind
 15      on (tjt.flitser_id_eind = opneind.flitser_id)
 16   where opnbegin.kenteken = opneind.kenteken
 17  )
 18  select traject_id
 19       , avg(max_toegestane_snelheid - km_per_uur) gemiddelde_
afwijking
 20    from t
 21   group by traject_id
 22  ;

TRAJECT_ID GEMIDDELDE_AFWIJKING
---------- --------------------
         1                  -30
         2                  -64
         4                  -20
         3            69.4117647
```

De query uit de eerste opdracht hebben we de naam 'T' gegeven en deze gebruiken we in de hoofdquery die begint op regel 18. De eigenlijke query is nu heel eenvoudig geworden. De negatieve waarden in de kolom 'GEMIDDELDE_AFWIJKING' duiden op overschrijding van de maximumsnelheid. Als we het traject_id uit de query halen, dan berekenen we de gemiddelde afwijking over alle trajecten.
Wat grappig is om te zien, is de volgorde van de trajecten. Al heel lang gaan geruchten, mythes, de ronde die vertellen dat een GROUP BY ook zou sorteren. Dit is niet waar en is ook nooit waar geweest. Zou je erop vertrouwen dat je GROUP BY altijd op een bepaalde manier gesorteerd wordt, dan kun je wel eens bedrogen uitkomen. Hier een kort stukje uit het EXPLAIN PLAN van deze query:

```
-------------------------------------------------------------------------
---------------
| Id  | Operation             | Name             | Rows  | Bytes | Cost
(%CPU)| Time     |
-------------------------------------------------------------------------
---------------
|   0 | SELECT STATEMENT      |                  |     4 |   516 |
11  (19)| 00:00:01 |
|   1 |  HASH GROUP BY        |                  |     4 |   516 |
11  (19)| 00:00:01 |
|*  2 |   HASH JOIN           |                  |     4 |   516 |
10  (10)| 00:00:01 |
|*  3 |    HASH JOIN          |                  |     8 |   776 |
7  (15)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| FLITS_OPNAMES    |     8 |   256 |
3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| FLITS_TRAJECTEN |    12 |   780 |
3   (0)| 00:00:01 |
|   6 |    TABLE ACCESS FULL | FLITS_OPNAMES    |     8 |   256 |
3   (0)| 00:00:01 |
-------------------------------------------------------------------------
---------------
```

Zoals je bij ID 1 kunt zien, vindt er een HASH GROUP BY plaats. Er vind geen sortering plaats. Wil je het resultaat in een bepaalde volgorde hebben, dan zul je dit expliciet moeten aangeven met een ORDER BY.

3) Welk kenteken heeft de grootste snelheidsovertreding gemaakt?

Geen leuker vermaak dan leedvermaak. Dus gaan we bepalen wie de hoogste bekeuring op de mat krijgt.

```
SQL> with t as
  2  (select opnbegin.kenteken
  3         , tjt.afstand_m
  4         , (opneind.opname_moment - opnbegin.opname_moment) tijdsduur
  5         , tjt.max_toegestane_snelheid
  6         , tjt.afstand_m /
  7           (extract(second from (opneind.opname_moment - opnbegin.
```

```
opname_moment)) +
  8           (extract(minute from (opneind.opname_moment - opnbegin.
opname_moment))) * 60 )
  9           * 3.6 km_per_uur
 10      from flits_trajecten tjt
 11      join flits_opnames opnbegin
 12        on (tjt.flitser_id_begin = opnbegin.flitser_id)
 13      join flits_opnames opneind
 14        on (tjt.flitser_id_eind = opneind.flitser_id)
 15     where opnbegin.kenteken = opneind.kenteken
 16  )
 17  , afwijkingen as
 18  (select t.kenteken
 19        , (max_toegestane_snelheid - km_per_uur) afwijking
 20     from t
 21    order by afwijking
 22  )
 23  select kenteken
 24       , afwijking
 25    from afwijkingen
 26   where rownum = 1
 27  ;

KENTEKEN  AFWIJKING
-------- ----------
93-JN-JT         -64
```

Een van de krachtigste mogelijkheden van Subquery Factoring is dat je de query steeds verder kunt uitbreiden. Het geeft een beetje een procedurele manier van een query opbouwen. Stukje bij beetje breiden we op deze manier de query uit. De query van de vorige opgave hebben we nu gedeclareerd in het Subquery Factoring deel, met toevoeging van een ORDER BY. Degene met de grootste afwijking komt in het tussenresultaat bovenaan te staan en in de 'echte' query zijn we alleen maar geïnteresseerd in de eerste. Vandaar de ROWNUM=1.

4) Toon in een matrix het aantal bekeuringen op basis van verschillende categorieën (0-10, 11-20, 21-30 te snel)

Om een matrixoverzicht te tonen, maken we gebruik van de PIVOT functie. Om niet weer dezelfde query helemaal te tonen, heb ik er een stukje tussenuit gelaten, namelijk de eerste 24 regels.

```
...
 25  cats as
 26  (select max_toegestane_snelheid
 27        , km_per_uur
 28        , afwijking
 29        , case
 30            when afwijking < 0
 31            then 'geen categorie'
 32            when afwijking between 0 and 10
 33            then 'categorie 1'
 34            when afwijking between 11 and 20
 35            then 'categorie 2'
 36            when afwijking between 21 and 30
 37            then 'categorie 3'
 38            when afwijking between 31 and 40
 39            then 'categorie 4'
 40            else 'categorie 5'
 41            end cat
 42     from res
 43  )
 44  select *
 45  from cats
 46       pivot (count(*)
```

```
 47               for cat in ('categorie 1'
 48                           ,'categorie 2'
 49                           ,'categorie 3'
 50                           ,'categorie 4'
 51                           ,'categorie 5'
 52                           )
 53             )
 54  /

MAX_TOEGESTANE_SNELHEID KM_PER_UUR  AFWIJKING 'categorie 1' 'categorie
2' 'categorie 3' 'categorie 4' 'categorie 5'
----------------------- ---------- ---------- ------------- ------------
- ------------- ------------- -------------

120         150          30              0             0             1
        0              0


100         120          20              0             1             0
 0              0

80         144          64              0             0             0
0              1

80 10.5882353 -69.411765              0             0             0
0              0
```

Wederom maken we gebruik van Subquery Factoring om eerder genoemde redenen. Op basis van de afwijking van de toegestane snelheid kennen we een categorie toe. Dit doen we met behulp van het CASE statement. De PIVOT functie is een nieuwe functie die in Oracle 11g is toegevoegd. Met deze functie kun je rijen data transformeren naar kolommen. De namen van de nieuwe kolommen worden bepaald door de namen die je opgeeft in de FOR clause van de PIVOT. Zo kun je redelijk eenvoudig een pivot operatie uitvoeren. Redelijk eenvoudig, want de syntax is even wennen.

File

Kwam dit artikel nu voort uit frustratie voor de vele boetes die intussen van mijn bankrekening zijn afgeschreven? Nee hoor. Het kwam vooral door de file waar ik iedere dag in sta. Dat geeft je een heleboel tijd om dit soort dingen uit te denken. De frustratie zit hem dus niet in de boetes, maar vooral in de file.


Patrick Barel (l.) en Alex Nuijten,
AMIS Services BV