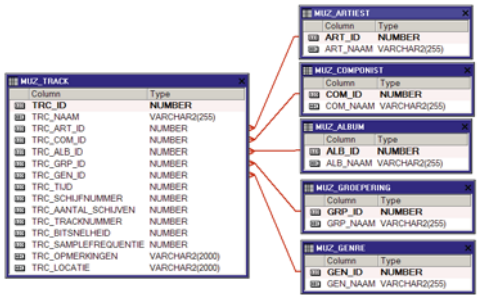
Puzzelen met SQL

De Muzieklijst, deel 2

In het vorige nummer is een start gemaakt met het bijhouden van een muzieklijst, die is aangeleverd in een tekstbestand, in Oracle. Toen lag de nadruk meer op het invoeren van gegevens in een database. Deze keer gaan we kijken wat er verder voor mogelijkheden zijn met de ingevoerde data.

Het datamodel dat we in deze puzzel gebruiken, ziet er als volgt uit:



1) Welke artiesten hebben muziekstukken uitgebracht met dezelfde naam?

Om deze vraag te kunnen beantwoorden moeten we de MUZ_TRACK tabel gebruiken. Hier staat tenslotte de naam van de verschillende muziekstukken in. Het antwoord is te vinden door in deze tabel te groeperen op de naam van het muziekstuk, zodat alleen de titels eruit komen die meerdere keren voorkomen.
In onderstaande query is een klein overzichtje van muziekstukken te zien die meerdere keren voorkomen. Hiermee is deels de vraag beantwoord. Er wordt gevraagd naar de namen van de artiesten, maar die staan nog niet in de resultaatset.

```
select t.trc_naam
     , count (distinct t.trc_art_id)
  from muz_track t
 group by t.trc_naam
 having count(distinct t.trc_art_id) > 1
;
```

A deeper love	3
A different corner	2
A groovy kind of love	2
A little bit more	2
A night to remember	2
A spaceman came travelling	2
ABC	2

Nu we een overzichtje hebben van muziektitels die meerdere keren voorkomen, kunnen we verder bouwen om in kaart te brengen welke artiesten deze titels nu geschreven hebben.

```
select trc_art_id
  from muz_track
 where trc_naam in (select t.trc_naam
                      from muz_track t
                     group by t.trc_naam
                    having count(distinct t.trc_art_id) > 1
                    );
```

Aan een lijstje met alleen maar Artiesten ID's hebben we natuurlijk ook niet veel. Dus laten we ook de naam van de artiest en de titel van het liedje erbij zien.

```
select trc.trc_art_id
     , (select art.art_naam
          from muz_artiest art
         where art.art_id = trc.trc_art_id
        ) artiest_naam
     , trc.trc_naam
  from muz_track trc
 where trc.trc_naam in (select t.trc_naam
                          from muz_track t
                         group by t.trc_naam
                        having count(distinct t.trc_art_id) > 1
                        );
```

85	30 seconds from Mars	From yesterday
86	30 seconds to Mars	The story
86	30 seconds to Mars	From yesterday
86	30 seconds to Mars	Attack
105	3rd Bass	Pop goes the weasel
105	3rd Bass	Pop goes the weasel
107	3rd bass	Pop goes the weasel

Zoals in de resultaatset te zien is, is het lastig om in een MP3-speler de data eenduidig in te voeren. Heet de band nu '30 seconds *from* Mars' of is het '30 seconds *to* Mars'? Ook het verschil in kapitalisatie leid tot een 'nieuwe' band.

2) Welke artiest heeft de langste nummers?
In de MUZ_TRACK tabel wordt de tijd van een nummer geregistreerd. Dit wordt gedaan in seconden. Het datatype van het veld is NUMBER.
Als we op zoek zijn naar het langste muziek nummer in onze database, dan is het voldoende om de MUZ_TRACK tabel te sorteren op de tijd, van groot naar klein, en vervolgens alleen de eerste te laten tonen. Laat je echter niet verleiden om deze query te schrijven:

```
select trc_id
     , trc_naam
     , trc_tijd
  from muz_track
 where rownum = 1
 order by trc_tijd desc
;
    TRC_ID TRC_NAAM                                  TRC_TIJD
---------- ------------------------------------ ----------
         4 A touch of jazz                             177
```

Deze query kiest een willekeurige rij en sorteert deze. Alleen indien de MUZ_TRACK tabel een enkele rij bevat, levert deze query het gewenste resultaat. Voordat de sortering zijn werk kan doen, als laatste stap in de verwerking van een query, is de filtering 'WHERE ROWNUM = 1' al geweest. Door deze filtering hebben we dus één rij, die vervolgens door de ORDER BY clause wordt gesorteerd.
Om ervoor te zorgen dat er eerst een sortering plaatsvindt voordat de where-clause wordt uitgevoerd, is het noodzakelijk om een inline-view op te nemen.

```
select trc_id
     , trc_naam
     , trc_tijd
  from (select trc_id
             , trc_naam
             , trc_tijd
          from muz_track
         order by trc_tijd desc nulls last
       )
 where rownum = 1
/
    TRC_ID TRC_NAAM                                  TRC_TIJD
---------- ------------------------------------ ----------
     11871 Radio 538 - Grandmix 2003                  10834
```

De inline-view sorteert eerst de data zoals deze in de MUZ_TRACK tabel staat op basis van de tijdsduur van lang naar kort.
Het buitenste deel van de query zorgt ervoor dat alleen de 'bovenste' rij getoond wordt, en dat is het nummer dat het langst duurt.
De 'NULLS LAST' in de ORDER BY clausule van de inline-view zorgt ervoor dat NULL helemaal onderaan in de (tussen) resultaatset komt te staan.

Als we op zoek gaan naar de artiest die gemiddeld de langste muziekstukken maakt, dan kunnen we een query als deze schrijven:

```
select *
  from (select avg (trc.trc_tijd)
             , trc.trc_art_id
          from muz_track trc
         group by trc.trc_art_id
         order by avg (trc.trc_tijd) desc
        )
 where rownum = 1
;
AVG(TRC.TRC_TIJD) TRC_ART_ID
----------------- ----------
             4681       3356
```

Nu rest de vraag nog hoe lang is 10834 seconden - het langste nummer uit de muziekcollectie - eigenlijk? Om inzichtelijker te maken hoeveel minuten en seconden dit eigenlijk is, kunnen we uiteraard heel ingewikkeld gaan rekenen. Maar we kunnen ook gebruik maken van de functie NumToDSInterval. Met behulp van deze functie is het mogelijk om een nummer om te zetten naar een Day to Second Interval datatype.
Oracle 9i heeft het INTERVAL datatype geïntroduceerd. Dit datatype geeft een bepaalde tijdsduur aan. Er zijn twee varianten: de INTERVAL DAY TO SECOND en de INTERVAL YEAR TO MONTH. Met de INTERVAL DAY TO SECOND kun je een periode uitdrukken in aantal dagen, uren, minuten, seconden tot aan nano-seconden. Het INTERVAL YEAR TO MONTH geeft een periode in jaren en maanden weer.

In de volgende query gaan we gebruiken maken van de NumToDSInterval functie.

```
SQL> select numtodsinterval (10834, 'second')
  2    from dual
  3  /

NUMTODSINTERVAL(10834,'SECOND')
---------------------------------------------
+000000000 03:00:34.000000000
```

Het eerste argument voor de functie is een numerieke waarde. In ons geval het aantal seconden. Het tweede argument vertelt iets over het eerste argument. In ons geval moet het getal worden geïnterpreteerd als seconden, dus gebruiken we SECOND.

Het nummer dat het langste duurt in onze database duurt dan ook nul dagen, drie uur, nul minuten en 34 seconden, een behoorlijk lang nummer.
Een handige tip: Het verschil tussen twee datum velden is het aantal dagen dat er tussen ligt. Het resultaat is dan een numerieke waarde. Om deze eenvoudig te formatteren kun je ook gebruik maken van de functie numtodsinterval, maar nu geef je als tweede argument 'day' op. Het resultaat is dan keurig geformatteerd:

```
SQL> select numtodsinterval (sysdate - sysdate + dbms_random.value
  2                          ,'day'
  3                          )
  4    from dual
  5  /

NUMTODSINTERVAL(SYSDATE-SYSDATE+DBMS_RANDOM.VALUE,'DAY')
---------------------------------------------------------------------------
+000000000 09:43:27.866137827
```

Om nu niet iedere keer de tijd (in seconden) om te hoeven rekenen naar een Interval om het netjes geformatteerd te krijgen, zouden we ervoor kunnen kiezen een view aan te maken waarin we de functie verbergen. Oracle 11g biedt ook een andere mogelijkheid om iets vergelijkbaars te doen: de Virtual Column.
Een Virtual Column is gebaseerd op een expressie. De syntax om een Virtual Column aan te maken, op basis van de functie die we net gebruikt hebben, is als volgt:

```
alter table muz_track
add (trc_tijd_interval as (numtodsinterval (trc_tijd, 'second')))
/
```

Nu is er een kolom aan de tabel toegevoegd met de naam TRC_TIJD_INTERVAL, met als datatype een INTERVAL.
Het is mogelijk om een Virtual Column te indexeren, met alle voordelen vandien. Om voorbeelden te zien hoe je een Virtual Column zou kunnen gebruiken voor het implementeren van Business Rules verwijzen we naar de AMIS Technology Blog.

3) Hoeveel nummers passen er gemiddeld op een CD van 74 minuten?

Laten we er een keertje vanuit gaan dat we dit moeten doen aan de hand van de daadwerkelijke tijd dat een muzieknummer duurt en niet van de grootte van het bestand. Deze vraag is eenvoudig te beantwoorden: Een kwestie van delen.

```
select (74 * 60) /avg (trc.trc_tijd)
  from muz_track trc
/
(74*60)/AVG(TRC.TRC_TIJD)
-------------------------
               13.2622016
```

Maar echte Hollanders als we zijn, er zouden toch meer nummers op een CD van 74 minuten moeten passen? Welke nummers zouden we dan moeten kiezen?
Om deze vraag te beantwoorden, gebruiken we Analytische Functies. We gaan een zogenaamde "Running Total" gebruiken. Een Running Total is een subtotaal dat meeloopt met de resultaatset.

```
select trc.trc_naam
     , trc.trc_tijd
     , trc.trc_tijd_interval
     , sum (trc.trc_tijd) over (order by trc.trc_tijd
                                       , trc.trc_naam
                               ) totale_tijd
  from muz_track trc
 order by trc.trc_tijd
        , trc.trc_naa
```

bonus track -	4	+000000000 00:00:04	4
Interlude: Let's dance	4	+000000000 00:00:04	8
Interlude: Race	4	+000000000 00:00:04	12
Intro	4	+000000000 00:00:04	16
Overheard in a wawa parking lot	4	+000000000 00:00:04	20
Yo Vanilla	4	+000000000 00:00:04	24
[Untitled]	4	+000000000 00:00:04	28
17 - Jingle 'Soul Show Classic' fas	5	+000000000 00:00:05	33
ID Tania	5	+000000000 00:00:05	38
Interlude: No acid	5	+000000000 00:00:05	43

De vierde kolom uit het bovenstaande voorbeeld laat het 'Running Total' zien. De eerste rij heeft een tijdsduur van 4 seconden, een erg kort nummertje - te zien in de tweede en derde kolom. Het subtotaal is dan ook 4 seconden.
Voor de tweede rij, ook een tijdsduur van 4 seconden, is het subtotaal 8 seconden. Vier seconden van de eerste rij plus vier seconden van de tweede rij. En zo wordt een 'Running Total' gecreëerd.
Omdat we maar plaats hebben voor 74 minuten, moet op dit tussenresultaat nog een filtering worden toegepast, zodat we maximaal 74 minuten kunnen vullen.

```
select trc_naam
     , trc_tijd
     , totale_tijd
  from (select trc.trc_naam
             , trc.trc_tijd
             , trc.trc_tijd_interval
             , sum (trc.trc_tijd) over (order by trc.trc_tijd
                                              , trc.trc_naam
                                      ) totale_tijd
          from muz_track trc
         order by trc.trc_tijd
                , trc.trc_naam
       )
 where totale_tijd <= 74 *60;
```

Het uiteindelijke resultaat levert 150 rijen op, veelal intro's van muziekstukken. Niet veel aan om naar te luisteren, maar wel lekker veel.

Omdat kwantiteit ook niet alles is, introduceren we een limiet van liedjes met een minimale duur van 1 minuut. Dit extra criterium voegen we toe aan de inline query. De totale query komt er dan als volgt uit te zien:

```
select trc_naam
     , trc_tijd
     , totale_tijd
  from (select trc.trc_naam
             , trc.trc_tijd
             , trc.trc_tijd_interval
             , sum (trc.trc_tijd) over (order by trc.trc_tijd
                                                , trc.trc_naam
                                        ) totale_tijd
          from muz_track trc
         where trc.trc_tijd >= 60
         order by trc.trc_tijd
                , trc.trc_naam
       )
 where totale_tijd <= 74 *60;
```

4) Wie zouden we volgend jaar naar het Eurovisie Song Festival kunnen sturen?

Een van de eisen die gesteld wordt aan de inzendingen voor het Eurovisie Song Festival is dat een liedje niet langer duurt dan drie minuten. In eerste instantie moeten we dus op zoek gaan naar muziekstukken, die korter zijn dan drie minuten. Aangezien we in een vorige opgave de MUZ_TRACK tabel al hebben uitgebreid met een Virtual Column die de tijdsduur in een net formaat laat zien, gaan we deze gebruiken.
Om een string om te zetten naar het juiste formaat - we hebben een INTERVAL DAY TO SECOND nodig - gebruiken we de conversie functie TO_DSINTERVAL.
Het argument voor deze TO_DSINTERVAL functie dient als volgt te worden gevuld:

```
to_dsinterval ('0 0:03:00')
```

De eerste nul geeft het aantal dagen weer. Gevolgd door een spatie en dan het aantal uren, minuten en seconden gescheiden door een dubbele punt (:). Bij het INTERVAL DAY TO SECOND is het ook mogelijk om fracties van seconden weer te geven. Deze komen, eventueel, na de seconden te staan gescheiden door een punt.
Voor deze opdracht willen we alle liedjes tonen die korter zijn dan drie minuten en de query komt er dan als volgt uit te zien:

```
select t.trc_naam
     , t.trc_tijd_interval
  from muz_track t
 where t.trc_tijd_interval <= to_dsinterval ('0 0:03:00')
;
```

A touch of jazz	+000000000 00:02:57
Lady Madonna	+000000000 00:02:16
The ballad of John and Yoko	+000000000 00:02:59
I feel fine	+000000000 00:02:18
Eight days a week	+000000000 00:02:44
Love me do	+000000000 00:02:20
From me to you	+000000000 00:01:56
She loves you	+000000000 00:02:21

Zoals je in het overzicht kunt zien, staan er heel veel liedjes in van buitenlandse artiesten. Helaas hebben we de nationaliteit van de artiesten niet geregistreerd, en kunnen we nu dus niet verder filteren.
Misschien dat we 'De Toppers' één van deze liedjes kunnen laten uitvoeren volgend jaar tijdens het Eurovisie Song Festival. Aan de andere kant, het maakt helemaal niet uit welk liedje we sturen, we winnen toch niet. De tijd van 'Ding-a-Dong' en 'Kleine Kokette Katinka' is lang vervlogen.

Wijzigingen doorvoeren in de muzieklijst

Sinds de vorige aflevering van deze puzzel is er een update van de lijst gekomen. Deze lijst bevat meer artiesten en vooral meer tracks. De MP3-verzameling is natuurlijk niet statisch. Heel af en toe verdwijnen er ook nummers van de MP3-speler wegens gebrek aan ruimte. Maar om bij iedere wijziging alle tabellen opnieuw aan te maken en te vullen is natuurlijk niet de bedoeling. Het datamodel kan natuurlijk uitgebreid worden met andere tabellen, waarin extra informatie kan worden opgenomen. Gelukkig biedt Oracle ook hiervoor een oplossing, namelijk het Mergen van data.

Het Merge statement is een combinatie van een INSERT, een UPDATE en - sinds Oracle 10g - een DELETE. Op basis van het matching-algorithme wat in de ON-clause wordt gedefinieerd, wordt een keuze gemaakt of er een INSERT moet plaatsvinden of een UPDATE en/of DELETE.

Wanneer we een nieuw bestand krijgen aangeleverd met MP3-gegevens, gaan we de naam van de artiest gebruiken als criterium om te bepalen wat we moeten doen: een INSERT of UPDATE/DELETE. Goed beschouwd zullen we geen gebruik maken van de UPDATE of DELETE functionaliteit van de MERGE, we willen tenslotte wel nieuwe artiesten kunnen toevoegen en bestaande ongemoeid laten. De overweging om hiervoor een MERGE te gebruiken ten opzichtte van een 'gewone' INSERT - SELECT wordt ingegeven door een performance overweging. Het kan zijn dat een MERGE die alleen maar een INSERT doet een betere performance heeft dan een INSERT-SELECT. Voordat je hier een uitspraak over kunt doen, dien je dit wel te onderzoeken.

De opbouw voor het MERGE-statement is als volgt: eerst geven we aan in welke tabel we de actie willen uitvoeren:

```
merge into muz_artiest a
```

Vervolgens geven we aan wat we als brondata willen gebruiken, in ons geval het nieuw aangeleverde bestand met MP3 gegevens:

```
using (select distinct artiest
          from muziek_ext
       ) muz
```

Nu wordt het tijd om aan te geven op basis van welke criteria er een vergelijk moet worden gedaan tussen de brondata en de doeldata. We doen het hier op basis van de naam van de artiest:

```
on (a.art_naam = muz.artiest)
```

En als laatste dienen we aan te geven wat er moet gebeuren op basis van bovengenoemde criteria:

```
when not matched
then insert (a.art_naam) values (muz.artiest);
```

Omdat we hier alleen maar gebruik willen maken van het INSERT deel van het MERGE-statement, is hiermee het statement compleet. Mocht je nu wel gebruik willen maken van het UPDATE of DELETE deel van de MERGE, dan kun je een WHEN MATCHED THEN clausule toevoegen. In Oracle 9i, toen het MERGE statement werd geïntroduceerd, was het echter verplicht om zowel een MATCHED als een NOT MATCHED clausule te hebben. Sinds Oracle 10g is deze verplichting komen te vervallen.

Per look-up tabel dienen we nu zo'n zelfde soort MERGE uit te voeren als hierboven beschreven. Wat nog niet eerder is verteld, is dat er op iedere tabel een trigger staat die zorgt voor het vullen van het ID indien deze nog niet is gevuld. Het ID wordt gevuld aan de hand van een sequence. Deze sequence start niet op 1, maar op het hoogste nummer dat in de desbetreffende tabel staat. Dit houdt dus in dat de sequences pas worden aangemaakt na het initiële laden van de stamtabellen.

Foto: Evelina Wellman

Wanneer alle look-up tabellen gewijzigd zijn met gelijksoortige MERGE-statements, kunnen we ook de nieuwe tracks gaan aanvullen in de MUZ_TRACKS tabel.

Dit doen we met het volgende statement:

```
1  merge into muz_track t
2  using
3    (select naam
4          , (select art_id
5               from muz_artiest
6              where art_naam = artiest
7            ) artiest_id
8       from muziek_ext) muz
9  on
10  (t.trc_naam = muz.naam)
11 when not matched
12 then insert (trc_naam
13              ,trc_art_id
14              )
15      values (naam
16              ,artiest_id
17              );
```

Om te bepalen wat het ID is van een artiest maken we gebruik van een Scalar Subquery, de SELECT in het SELECT statement - regel 4, 5 en 6. Een Scalar subquery is een query die maar één kolomwaarde teruggeeft voor een rij. Dit is een hele eenvoudige manier om het ID van een artiest te bepalen.

Zwembad

In de vorige 'Puzzelen met SQL' noemden we een speciaal hoesje waarmee je kunt zwemmen en tegelijkertijd naar je favoriete muziek kunt luisteren. Op onderstaande foto's is te zien hoe je van je favoriete muziek kunt genieten en tegelijkertijd een paar baantjes kunt trekken om in conditie te blijven.

Patrick Barel en **Alex Nuijten**, AMIS Services BV