Puzzelen met SQL

De muzieklijst, deel 1

In de trein, in het park, in de stad, op de fiets – voorop en ook achterop – tijdens het hardlopen en zelfs met zwemmen, overal kom je ze tegen. Of het nu verhalen zijn, of muziek, steeds meer zie je mensen met MP3 spelers op. Bij ieder MP3-bestand kun je een heleboel informatie ingeven. Niet alleen de naam, de artiest, componist of genre. Maar ook het jaar, de bitsnelheid, aantal tracks, equalizer, tijdsduur en bewerking.

Maar ja, wie vult dat nu allemaal in? Welnu, een vriend van mij heeft een enorme verzameling MP3-bestanden. Bij ieder bestand zijn alle gegevens ingevuld in iTunes. Voor hem is dit natuurlijk heel handig. Zo kan hij zien welk nummer op dat moment wordt afgespeeld, maar heel veel meer kan hij met die data niet doen.
Aangezien hij de data heeft, en wij de database... je voelt hem al aankomen. Dit leent zich uitstekend voor een Puzzel met SQL. Vragen als:
- Welke artiest heeft de langste nummers?
- Zijn de liedjes van vroeger inderdaad veel korter dan die tegenwoordig worden gemaakt?
- Worden er inderdaad geen originele liedjes meer geschreven, maar alle liedjes gerecycled?
- Hoeveel liedjes van een bepaalde artiest kun je eigenlijk op een CD branden?
- Welke liedjes moet je kiezen om er zoveel mogelijk van op een CD te zetten?

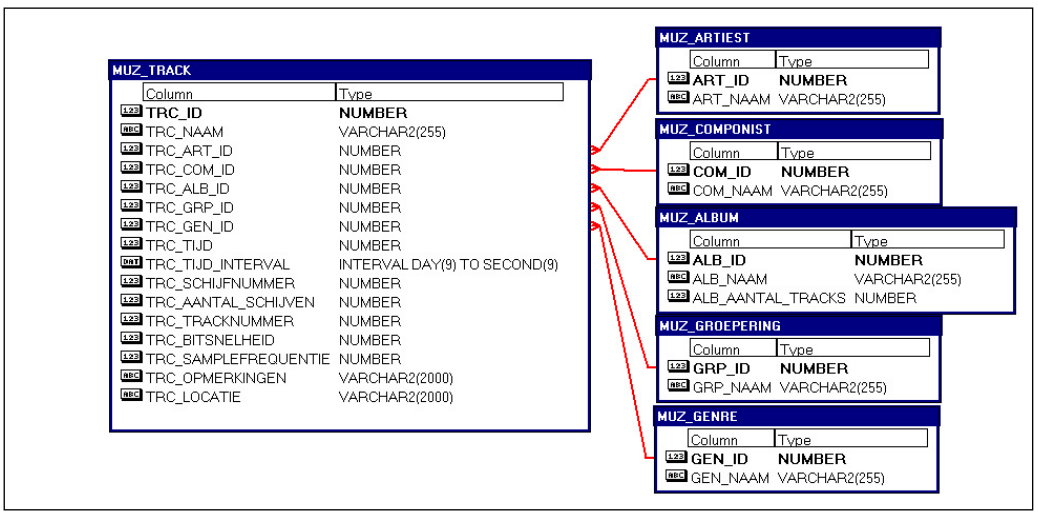
Deze keer is de puzzel iets anders van opzet dan gebruikelijk. In deze aflevering gaan we stap voor stap door het laden en bijwerken van onze stamgegevens heen. Voor de volgende aflevering lossen we verschillende vraagstukken op aan de hand van deze stamgegevens.

De tabellen

De tabellen die we gaan gebruiken in deze puzzel zijn weergegeven in afbeelding 1.

In afbeelding 1 komen gegevens van het specifieke track te staan, bijvoorbeeld de naam van het liedje. In de Artiest, Componist, Album, Groepering en Genre staan de lookupgegevens voor ieder track. Hoe gaan we onze tabellen nu vullen met de gegevens? We krijgen een bestand aangeleverd met alle data die we bij een MP3 kunnen opslaan. Traditioneel gezien zou je deze gegevens met behulp van SQL*Loader kunnen inlezen. Gegevens worden in een tijdelijke tabel gelezen, waar je vervolgens met verschillende DML-statements de Lookup-tabellen vult, en vervolgens de Tracks.

Maar dat gaan we nu eens niet doen. We gaan hier gebruikmaken van External Tables en een Multitable Insert.



Afbeelding 1

External Table

Sinds Oracle 9i is het mogelijk een External Table te definiëren. Met een External Table kun je data uit externe bronnen, zoals een tekstbestand, benaderen alsof het een database-tabel is. DML-operaties op een External Table zijn echter niet mogelijk. Het bestand met alle MP3-gegevens erin plaatsen we op de database server. Oracle moet tenslotte wel bij het bestand kunnen komen om het vervolgens te kunnen lezen. Als er een nieuwe versie van het exportbestand komt, kunnen we deze eenvoudig vervangen en wordt de nieuwe data automatisch beschikbaar. Het werkt dus geheel transparant.
Voordat we de definitie van de External Table kunnen maken, is het noodzakelijk een Directory aan te maken. De Directory waar ik het hier over heb, is niet de fysieke directory waar je het tekstbestand in zet, maar een Oracle Directory. Eigenlijk gaat het om een database object waarin het pad staat naar het fysieke pad op de database server. Vroeger moest je in de init.ora-parameter utl_file_dir aangeven in welke directory geschreven en gelezen mocht worden. Dit was een heel grove manier om met lees- en schrijfrechten om te gaan. Nu er Oracle Directories zijn, kun je heel specifiek lees- en schrijfrechten uitdelen aan bepaalde gebruikers.
Om een Oracle Directory aan te maken is het voldoende dit commando uit te voeren:

```
CREATE OR REPLACE DIRECTORY EXT_TABLES AS 'D:\MySources\SQL\puzzler\
muziek'
```

Nu hebben we een Oracle Directory met de naam EXT_TABLES. Het aanmaken van de fysieke folder moet je zelf doen, dit gebeurt niet met dit commando.
Dit is de eerste stap. We moeten nu nog rechten krijgen op deze directory voordat we er echt iets mee kunnen doen.

```
GRANT READ, WRITE ON DIRECTORY ext_tables TO patrick;
```

Voor deze puzzel is het niet echt nodig om schrijfrechten te hebben op de directory, we hoeven tenslotte alleen maar het bestand met MP3-gegevens te lezen.
Nu we de fysieke folder hebben aangemaakt, er een Oracle Directory is en we rechten hebben op deze directory, kunnen we de External Table gaan definiëren.
Hier de verkorte versie van de External Table-definitie, de gehele External Table staat in afbeelding 2 (Muziek_Ext):

```
SQL> create table muziek_ext
  2  ( naam char(255)
  3  , artiest char(255)
  4  , componist char(255)
  5  , album char(255)
  6  , bewerking date
  7  )
  8  organization external
  9  (
 10   type oracle_loader
 11   default directory ext_tables
 12   access parameters
 13   (records delimited by newline
 14    fields terminated by 0x'09'
 15    missing field  values are null
 16      ( naam
 17      , artiest
 18      , componist
 19      , album
 20      , bewerking char date_format date mask "dd-mm-yyyy hh24:mi"
 21      )
 22   )
 23   location ('muziek.txt')
 24   )
 25  reject limit unlimited;
```

Hoewel de syntax eenvoudig begint, gewoon met een CREATE TABLE, wordt het al snel ingewikkelder. Als je in het verleden wel eens SQL*Loader hebt gebruikt, ziet de syntax er bekender uit. Een paar dingetjes willen we wat nader toelichten. Voor meer informatie verwijzen we naar de Oracle Documentatie over External Tables.
Op regel 8 wordt al meteen duidelijk dat we hier te maken hebben met een External Table door de "ORGANIZATION EXTERNAL" syntax.
Het directory object dat we in een eerdere stap hebben aangemaakt, gebruiken we in de regel 11.
Afhankelijk van het tekstbestand kun je aangeven hoe de individuele velden worden gescheiden, in het geval van een CSV file is het een komma, in ons geval het TAB-karakter. Aangezien je dit niet zomaar kan intypen, moeten we Oracle vertellen dat het gaat om asciicode 9: 0x'09'.
Vaak zijn het niet alleen karaktervelden in een tekstbestand, maar staan er ook datums of numerieke waarden in. Het is mogelijk om dit aan te geven bij de definitie van de External Table zoals te zien is op regel 20.

De echte External Table die we in deze puzzel gebruiken zie je in afbeelding 2.

De definitie van de External Table is nu helemaal gereed. Maar hoe gebruik je zo'n External Table nu? Je typt dit commando:

```
select *
  from muziek_ext t;
```

Eenvoudig niet waar? Zie afbeelding 3.

MUZIEK_EXT

	Column	Type
ABC	NAAM	CHAR(255)
ABC	ARTIEST	CHAR(255)
ABC	COMPONIST	CHAR(255)
ABC	ALBUM	CHAR(255)
ABC	GROEPERING	CHAR(255)
ABC	GENRE	CHAR(255)
123	GROOTTE	NUMBER
123	TIJD	NUMBER
123	SCHIJFNUMMER	NUMBER
123	AANTAL_SCHIJVEN	NUMBER
123	TRACKNUMMER	NUMBER
123	AANTAL_TRACKS	NUMBER
123	JAAR	NUMBER
DAT	BEWERKING	DATE
DAT	TOEGEVOEGD	DATE
123	BITSNELHEID	NUMBER
123	SAMPLEFREQUENTIE	NUMBER
ABC	VOLUMEAANPASSING	CHAR(255)
ABC	SOORT	CHAR(255)
ABC	EQUALIZER	CHAR(255)
ABC	OPMERKINGEN	CHAR(2000)
123	AFGESPEELD	NUMBER
DAT	LAATSTE_KEER	DATE
123	OVERGESLAGEN	NUMBER
DAT	LAATST_OVERGESLAGEN	DATE
123	BEOORDELING	NUMBER
ABC	LOCATIE	CHAR(2000)

Afbeelding 2

Multi Table Insert

Nu de tabellen helemaal gereed zijn, wordt het tijd om deze te gaan vullen met gegevens uit onze External Table. Uiteraard is het mogelijk dit per tabel uit te voeren, eerst alle componisten, artiesten, albums, groeperingen en genre, en vervolgens de Tracks te gaan vullen.

Het grote nadeel van deze methode is dat we de External Table, waar de brongegevens in staan, zes keer moeten benaderen. Het zou een stuk fraaier, en uiteindelijk ook een beter performend zijn, om het in één statement te kunnen doen.

Hiervoor gebruiken we de Multi Table Insert. Met een enkel statement kun je vanuit één bron verscheidene tabellen gelijktijdig vullen. Dit gaat zowel op basis van gestelde voorwaarden als onvoorwaardelijk.

Er zitten echter wat haken en ogen aan. Zo kun je bijvoorbeeld niet gebruikmaken van een Oracle sequence binnen het statement, en dat is toch wel een groot gemis. Aangezien we geen gebruik kunnen maken van sequences, gaan we zelf de Primary Keys van de lookup-tabellen vullen in het statement. Omdat we dit doen, kunnen we ook zelf de foreign keys vullen van de MUZ_TRACKS -tabel.

We willen uiteraard artiesten maar één keer opnemen in de MUZ_ARTIEST-tabel. Hetzelfde geldt voor de Albums, Componisten, et cetera. In de subquery die we als bron gaan gebruiken, moeten we dus een manier vinden om artiesten maar één keer te laten voorkomen. Dit kunnen we doen door gebruik te maken van Analytische Functies.

De Analytische Functie die we hiervoor gebruiken is ROW_NUMBER. De PARTITION BY clause die we gebruiken is gebaseerd op de artiest. Met als resultaat dat je per artiest een nummer één, twee, drie, et cetera krijgt. Aangezien we maar één artiest in de MUZ_ARTIEST-tabel willen zien, zijn we alleen geïnteresseerd in de eerste.

Door gebruik te maken van een CASE-expressie kunnen we de eerste eruit filteren.

Omdat we al geen gebruik kunnen maken van sequences binnen een Multi Table Insert, zullen we dus zelf een ID moeten verzinnen. Ook hiervoor gebruiken we een Analytische Functie. Hier hebben we gekozen voor de RANK-functie. Net zo goed hadden we DENSE_RANK of ROW_NUMBER kunnen kiezen, alle drie rankingfuncties met subtiele verschillen. De verschillen zitten voornamelijk in de manier waarmee zij omspringen met gelijke waarden. De gekozen rankingfunctie is in dit geval niet van belang. Wel van belang, van groot belang zelfs, is dat we geen PARTITION BY clause opnemen. Over de gehele resultaatset moeten alle artiesten een uniek nummer hebben. Zo krijgt bijvoorbeeld Queen nummer 35, en voor alle tracks zal Queen dit nummer hebben.

```
select naam
     , case
       when artiest is not null then
         rank () over (order by artiest
                       )
       end artiest_id
     , case row_number() over (partition by artiest order by null)
       when 1 then artiest
       end artiest
  from muziek_ext
;
```

	NAAM	ARTIEST	COMPONIST	ALBUM	BEWERKING
1	Honeysuckle rose	Robert Palmer		Acoustic love	2/15/0008
2	Sugar mice [Acoustic version]	Marillion		Acoustic love	2/15/0008
3	Your love is king	Sade		After midnight - 19 jazzy lovesongs	1/11/0008
4	The age of the understatement	The Last Shadow Puppets		The age of the understatement	5/8/0008

Afbeelding 3

Ce matin la	78	
Meditation	98	Air of gloom
Wild oak tree	99	Airmate
Let it go	99	
Born under a bad sign	101	Albert King
Luna	102	Alessandro Safina
Halo of flies	103	Alice Cooper
Elected	103	
Hello hurray	103	
Poison	103	
Big in Japan	107	Alphaville
A wish for somethng more	108	Amy MacDonald
L.A.	108	
Barrowland ballroom	108	
Let's start a band	108	
Footballer's wife	108	
Youth of today	108	
Poison Prince	108	
This is the life	108	
Mr Rock & Roll	108	
Run	108	
Some unholy war	118	Amy Winehouse

Afbeelding 4

In afbeelding 4 zien we dat er per artiest een ID is 'verzonnen' en dat bij het merendeel de artiestennaam niet is ingevuld. Een Multi Table Insert maakt het mogelijk condities op te geven. Het vullen van de MUZ_ARTIEST-tabel doen we dan ook met behulp van zo'n conditie. Als de artiestennaam is ingevuld, voegen we een rij toe in de MUZ_ARTIEST-tabel.

```
when artiest is not null
then
  into muz_artiest (art_id, art_naam)
  values (artiest_id, artiest)
```

Alle andere rijen in de resultaatset moeten een rij opleveren in de MUZ_TRACKS-tabel, de conditie kan er dan zo uit zien:

```
when 1 = 1
then
 into muz_track (trc_naam, trc_art_id, trc_alb_id)
 values (naam, artiest_id, album_id)
```

Het totale Multi Table Insert-statement komt er dan als volgt uit te zien.

```
insert all
 when 1 = 1
 then
  into muz_track (trc_naam, trc_art_id, trc_alb_id)
  values (naam, artiest_id, album_id)
 when artiest is not null
 then
   into muz_artiest (art_id, art_naam)
   values (artiest_id, artiest)
```

```
select naam
     , case
       when artiest is not null then
         rank () over (order by artiest
                       )
       end artiest_id
     , case row_number() over (partition by artiest order by null)
       when 1 then artiest
       end artiest
  from muziek_ext
;
```

Deferrable Foreign Keys

Als we het Multi Table Insert-statement zo uitvoeren, krijgen we een Integrity Constraint Violation. Blijkbaar vindt er geen validatie op statementniveau plaats bij een Multi Table Insert, of worden de individuele delen gezien als aparte statements.
Dit hoeft geen probleem te zijn. De oplossing zit hem in de foreign key. De foreign keys die tussen MUZ_TRACKS en de lookup-tabellen zitten zijn deferrable foreign keys.
Deferrable Constraints zijn constraints die je tijdelijk kunt 'uit' zetten, zodat een tijdelijke overtreding van constraints toegestaan is. De nadruk ligt op tijdelijk, ten tijde van een COMMIT moet iedere vorm van datacorruptie opgelost zijn. Je kan dus een rij aanmaken in de MUZ_TRACKS-tabel met een verwijzing naar een artiest die nog niet in de MUZ_ARTIEST-tabel staat. Als je er maar voor zorgt dat de betreffende artiest wel aanwezig is op het moment dat je de data probeert te committen.
Hoe geef je aan dat de Foreign Key deferrable is? Met de volgende syntax:

```
alter table MUZ_TRACK
  add constraint fk_track_artiest foreign key (TRC_ART_ID)
  references muz_artiest (ART_ID) deferrable initially immediate;
```

Op de derde regel van het statement staat DEFERRABLE INITIALLY IMMEDIATE. Dit betekent dat deze foreign key constraint het toelaat tijdelijk uitgezet te worden (DEFERRABLE). Maar om te beginnen wordt hij altijd onmiddellijk gevalideerd (INITIALLY IMMEDIATE).
Het tijdelijk uitzetten van de validatie van de constraints doen we met dit commando:

```
set constraints all deferred
/
```

Uiteraard is het ook mogelijk deze actie per constraint uit te voeren.
Om de constraints weer terug te zetten, zodat de validatie

onmiddellijk wordt uitgevoerd, kunnen we dit commando gebruiken:

```
set constraints all immediate
/
```

Dit laatste commando is met name handig om te controleren of alle data weer aan alle constraint-regels voldoet. In geval van een overtreding, stel dat je een verwijzing hebt in MUZ_TRACKS naar een artiest die niet aanwezig is in de MUZ_ARTIEST-tabel, krijg je een Integrity Constraint violation te zien. De data in de tabellen is dan nog gewoon aanwezig, je kunt er dus correcties op aanbrengen net zolang tot de data weer helemaal intact is.
Uitvoeren van een COMMIT gaat ook. Deze zal de constraints meteen naar IMMEDIATE zetten. De Integrity Constraint kun je zien, maar door deze foutmelding is er een impliciete ROLLBACK opgetreden. Dit maakt het dan dus onmogelijk om correcties uit te voeren.
Het uiteindelijke script dat we gaan gebruiken om onze doeltabellen vanuit de External Table met behulp van een Multi Table Insert te vullen, komt er dan zo uit te zien:

```
set constraints all deferred
/

insert all
 when 1 = 1
 then
  into muz_track (trc_naam, trc_art_id, trc_alb_id)
  values (naam, artiest_id, album_id)
 when artiest is not null
 then
   into muz_artiest (art_id, art_naam)
   values (artiest_id, artiest)
select naam
     , case
       when artiest is not null then
         rank () over (order by artiest
                       )
       end artiest_id
     , case row_number() over (partition by artiest order by null)
       when 1 then artiest
       end artiest
  from muziek_ext
;

set constraints all immediate
/
```

Als er op dit moment geen exceptions zijn opgetreden, weten we dat de data aan alle integriteitsregels voldoet, en dan kunnen we een COMMIT uitvoeren.

Volgende keer

In tegenstelling tot vorige afleveringen van 'Puzzelen met SQL' hebben we deze keer geen vraagstukken opgelost, maar laten zien hoe je van een tekstbestand kunt komen tot het vullen van stamtabellen in de database. Het gebruik van geavanceerde functionaliteit zoals External Tables, Multi Table Insert en de MERGE stonden centraal in deze aflevering.
In de volgende aflevering van 'Puzzelen met SQL' gaan we gebruikmaken van de data die we in dit artikel hebben geladen. We gaan een antwoord zoeken op de vraag welke artiest gemiddeld de langste liedjes schrijft of welke liedjes je moet kiezen voor het zo efficiënt mogelijk branden van een CD. Ook gaan we nieuwe functionaliteit toepassen die in Oracle 11g beschikbaar is, zoals de Virtual Column.
De online-versie van dit artikel is te vinden op het AMIS Technology Weblog http://technology.amis.nl/blog/?p=3353. Hier zijn ook de scripts te downloaden om zelf met deze puzzel aan de slag te gaan.

Zwembad

Zoals aan het begin van dit artikel genoemd, zie je tegenwoordig mensen zelfs in het zwembad met een MP3-speler. In het water, wel te verstaan. Dat kan door gebruik te maken van een speciaal, waterdicht hoesje voor de MP3-speler. Bijvoorbeeld voor een IPod.

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