De vele gezichten van DUAL

Mystiek rond een vreemde databasetabel

Al heel lang vervult de tabel DUAL in veel Oracle databases een belangrijke rol. Het is vaak de meest benaderde tabel in een database. Maar hoe komt dat eigenlijk?

De tabel DUAL bestaat uit een kolom en een rij. Waarom heet die dan DUAL? Daarover gaan veel verhalen en geruchten. Er wordt beweerd dat DUAL vroeger twee rijen had. Een ander gerucht is dat DUAL werd gebruikt om demotabellen te vullen (te verdubbelen). De waarheid zal wel in het midden liggen. Tegenwoordig wordt DUAL veel door PL/SQL en andere applicaties gebruikt (en misbruikt). DUAL blijkt ook een heel vreemde tabel te zijn. Laten we eens kijken hoe DUAL is gecreëerd in de database.

```
Sqlplus "/ as sysdba"

SQL> desc DUAL;

Name                           Null?    Type
------------------------------ -------- ----
DUMMY                          VARCHAR2(1)

Er is dus 1 kolom en in die kolom is er ruimte voor 1 karakter.

SQL> select * from DUAL;

D
-
X
```

De waarde in de kolom DUMMY is 'X'. Tot zover niks bijzonders, maar nu komt de magie:

```
SQL> insert into dual values ('Y');

1 row created.

SQL> select * from DUAL;

D
-
X
```

Waar is de waarde 'Y' gebleven? De insert is gedaan en er waren geen fouten! Is die 'Y' echt niet in de DUAL tabel aanwezig?

```
SQL> select count(*) from DUAL;

COUNT(*)
--------
       2
```

Dus de 'Y' is wel aanwezig, maar we zien de 'Y' niet.
Hoe kunnen we die te voorschijn toveren?

```
SQL> delete from DUAL;

1 row deleted.
```

Dit zou eigenlijk moeten zeggen: 2 rows deleted. Dus er is maar een van de twee rijen verwijderd uit de tabel.

```
SQL> select * from DUAL;

D
-
Y
```

Er zijn veel verklaringen en geruchten, waarom een tabel die bestaat uit één kolom en één rij toch DUAL heet

Hoe kan dit? Laten we dit alles nog eens doen maar dan niet als SYS maar als SCOTT, maar dan moeten we eerst de DUAL tabel creëren:

```
SQL> connect SCOTT/TIGER

SQL> create table DUAL (dummy varchar2(1));

Table created.

SQL> insert into DUAL values ('X');

1 row created.

SQL> commit;

Commit complete.

SQL> desc DUAL;


Name                           Null?    Type
_____________________- ____ ____
DUMMY                          VARCHAR2(1)
```

Er is dus weer één kolom en in die kolom is weer ruimte voor één karakter.

```
SQL> select * from DUAL;

D
-
X
```

De waarde in de kolom DUMMY is 'X'. Wat gebeurt er nu als we de waarde 'Y' weer toevoegen:

```
SQL> insert into dual values ('Y');

1 row created.

SQL> select * from DUAL;

D
-
X
Y
```

Dus nu zien we wel 2 rijen, maar dat zagen we niet met de DUAL tabel onder SYS! Wat zou er met de DELETE gebeuren op de DUAL van SCOTT?

```
SQL> delete from DUAL;

2 rows deleted.
```

Nu verdwijnen er wel twee rijen. Er is dus een verschil. Het maakt uit onder welke gebruiker DUAL wordt gecreëerd! Dat betekent dat Oracle (de optimizer om precies te zijn), moet weten met welke DUAL het te maken heeft. Als het met SYS.DUAL te maken heeft gaat ORACLE er vanuit dat er maar één rij aanwezig is, ook al zijn er meer.

Wanneer we nu onder SYSTEM in SQL*Plus autotrace aanzetten, dan zien we het volgende:

```
SQL> set autotrace on
SQL> select * from dual;
D
-
X


Execution Plan
_____________________________________________
0              SELECT STATEMENT (Optimizer=CHOOSE)
1     0        TABLE ACCESS (FULL) OF 'DUAL'


Statistics
_____________________________________________
0              recursive calls
0              db block gets
3              consistent gets
0              physical reads
0              redo size
375            bytes sent via SQL*Net to client
499            bytes received via SQL*Net from client
2              SQL*Net roundtrips to/from client
0              sorts (memory)
0              sorts (disks)
1              row processed
```

Hier kunnen we zien dat de SELECT * FROM DUAL drie consistent buffer gets doet om één rij terug te brengen. Dat lijkt niet veel, maar wanneer DUAL vaak wordt gebruikt kan het

In volgende releases zal de fysieke tabel DUAL verdwijnen en door een virtuele tabel worden overgenomen

aantal consistent gets aardig oplopen. Wat kunnen we dan doen om DUAL goedkoper te maken? Oracle heeft sinds 8.0.3 een nieuwe tabel: X$DUAL. Dit is een memory resident tabel. Een describe vertelt ons het volgende:

```
SQL> connect / as sysdba
SQL> desc X$DUAL;
Name                           Null?    Type
_____________________          ____     ___________

ADDR                                    RAW(4)
INDX                                    NUMBER
INST_ID                                 NUMBER
DUMMY                                   VARCHAR2(1)

SQL> select * from x$dual;



ADDR                           INDX       INST_ID  D
_______ _______ __________  -
0A3A2640                                0                       1 X
```

Deze X$DUAL ziet er dus net zo uit als DUAL, alleen ligt deze tabel in het de SGA. Het grote voordeel hiervan is dat er geen consistent gets meer nodig zijn. Om X$DUAL te kunnen gebruiken dient er een view te worden gecreëerd:

```
SQL> create view dualv as select dummy from x$dual

View created.

SQL> grant select on sys.dualv to public;

Grant succeeded.

SQL> connect scott/tiger
Connected.
SQL> select * from dualv;

D
-
X
```

Het is natuurlijk ook mogelijk om de table DUAL te droppen en die dan te vervangen door een view DUAL (op select dummy from x$dual). Dit kan grote voordelen geven in PL/SQL.
De meeste functies in PL/SQL worden namelijk vertaald naar:

```
SELECT <FUNCTIE> INTO <VAR> FROM DUAL;
```

Met de originele DUAL betekent dat altijd drie buffer gets. Met X$DUAL zijn er geen buffer gets meer. Eén van de meest voorkomende PL/SQL functies is SYSDATE. SYSDATE werd tot Oracle Release 8.1.7.2 altijd vertaald naar:

```
SELECT SYSDATE INTO <VAR> FROM DUAL;
```

Sinds 8.1.7.2 is deze functie in PL/SQL een 'ingebouwde' of native C functie geworden. Vanaf deze versie kunnen PL/SQL routines die veelvuldig SYSDATE aanroepen sneller geworden zijn. Er kan daardoor ook een vermindering van de consistent gets hebben plaatsgevonden. De PL/SQL functie SYSDATE is de enige functie die veranderd is. De andere functies zoals USER worden nog steeds vertaald naar SELECT … DUAL.

Conclusie

De dual tabel is een hele speciale tabel in Oracle. De Oracle kernel gaat er altijd vanuit dat er maar één rij in deze tabel zit, maar dan moet de tabel wel onder het schema van SYS gecreëerd zijn en niet onder een andere gebruiker. X$DUAL is nog specialer en kan wel voor performance verbeteringen zorgen voor mensen die veel PL/SQL draaien. Oracle Corporation heeft zelf ook wel in de gaten dat DUAL veel gebruikt wordt en toch ook een performance probleem kan zijn. Dit komt uit de release notes van Oracle9 Release 2:

> **6. Using SYS.DUAL for Updates**
>
> The use of table SYS.DUAL for updates (including SELECT FOR UPDATES) will be prohibited in a future release. If you need to update SYS.DUAL to enforce concurrency control of your application, see dbmslock.sql as an alternative. SYS.DUAL will still be available for selection.

Volgens mij betekent dat in volgende releases de fysieke tabel DUAL zal verdwijnen en door een virtuele tabel zal worden overgenomen. Dit zal alleen maar mystiek aan de toch vreemde DUAL tabel toevoegen.

Anjo Kolk is werkzaam bij Precise Software Solutions. Voordat hij bij Precise in dienst trad werkte hij meer dan zestien jaar bij Oracle Corporation, waar hij zich vooral bezig hield met rdbms-performance. Hij is per e-mail te bereiken op akolk@precise.com.

Met dank aan Peter Gram van Miracle A/S.