RuleGen testen met Codetester

De proef op de som

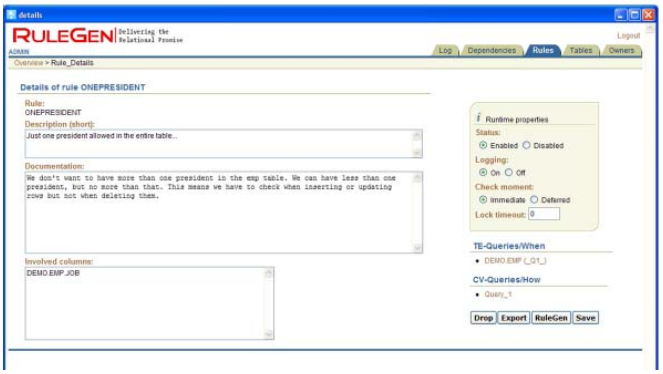
Met de mogelijkheden van RuleGen in het achterhoofd, vroeg Patrick Barel zich af hoe deze regels te testen zijn. Hij bouwde zijn eigen testset om het antwoord op die vraag te vinden. Hoe het hem vergaan is, beschrijft hij voor Optimize.

RuleGen is een tool om business rules in de database te maken op basis van SQL-queries. RuleGen genereert dan een paar triggers en een package om deze business rules af te dwingen. Zo worden ze afgedwongen in de database, onafhankelijk van de gebruikte applicatie, om de data te wijzigen. Het is een beetje zoals CDM Ruleframe, zoals beschikbaar bij Oracle Designer.

Terug naar de vraag hoe deze regels getest kunnen worden. Ik heb wat ervaring met 'Quest CodeTester for Oracle' (http://unittest.inside.quest.com), maar zoals de naam al zegt, is dit een testtool voor (PL/SQL)code, terwijl RuleGen code genereert die alleen getest kan worden met behulp van SQL-statements (insert, update en delete).
Als ik toch gebruik wil maken van Codetester om het testen uit te voeren, moet ik de insert, update en delete opdrachten in eenvoudige procedures zien te vangen. Ik wil niet alle waarden voor het record als parameters meesturen, maar ik bedacht dat ik een PL/SQL Record kan maken gebaseerd op de tabel. Hiermee kan ik in de 'pre-execution' code van CodeTester dit record vullen met de gewenste waarden die dan in de test gebruikt gaan worden.

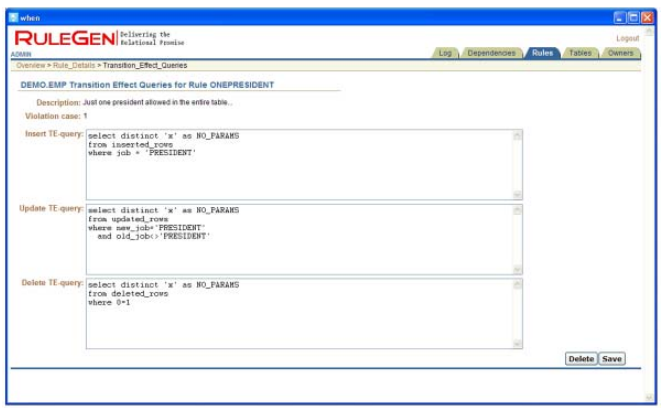
Ik heb mijn testset gebouwd op basis van een eenvoudige tabel met een eenvoudige rule die gecontroleerd moet worden. Een en ander op basis van de presentatie van Toon Koppelaars zoals beschikbaar op de site van RuleGen.

Ik heb deze regel in de eenvoudige (Apex) frontend opgenomen:



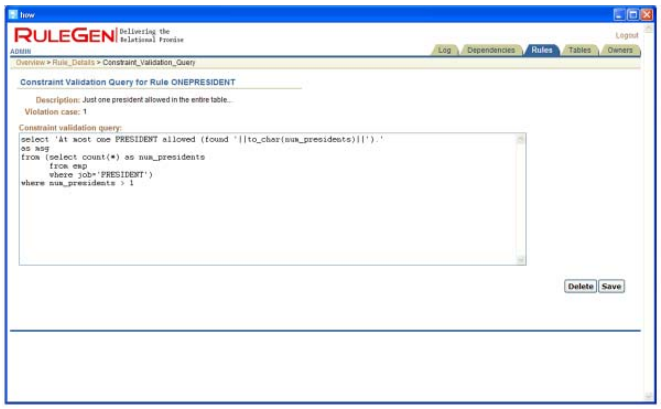
Details van de regel ONEPRESIDENT:

Rule:	ONEPRESIDENT
Description (short):	Just one president allowed in the entire table...
Documentation:	We don't want to have more than one president in the emp table. We can have less than one president, but no more than that. This means we have to check when inserting or updating rows but not when deleting them.
Involved columns:	DEMO.EMP.JOB



Transition Effect Queries for Rule ONEPRESIDENT

Description:	Just one president allowed in the entire table...
Violation case:	1
Insert TE-query:	select distinct 'x' as NO_PARAMS from inserted_rows where job = 'PRESIDENT'
Update TE-query:	select distinct 'x' as NO_PARAMS from updated_rows where new_job='PRESIDENT' and old_job<>'PRESIDENT
Delete TE-query:	select distinct 'x' as NO_PARAMS from deleted_rows where 0=1



Constraint Validation Query for Rule ONEPRESIDENT

Description:	Just one president allowed in the entire table...
Violation case:	1
Constraint validation query:	select 'At most one PRESIDENT allowed (found '||to_char(num_presidents)||').' as msg from (select count(*) as num_presidents from emp where job='PRESIDENT') where num_presidents > 1

Nu moet ik een package maken met een record gebaseerd op deze tabel:

```
TYPE EMP_rt IS RECORD(
    EMPNO EMP.EMPNO%type,
    ENAME EMP.ENAME%type,
    JOB EMP.JOB%type,
    MGR EMP.MGR%type,
    HIREDATE EMP.HIREDATE%type,
    SAL EMP.SAL%type,
    COMM EMP.COMM%type,
    DEPTNO EMP.DEPTNO%type
  );
```

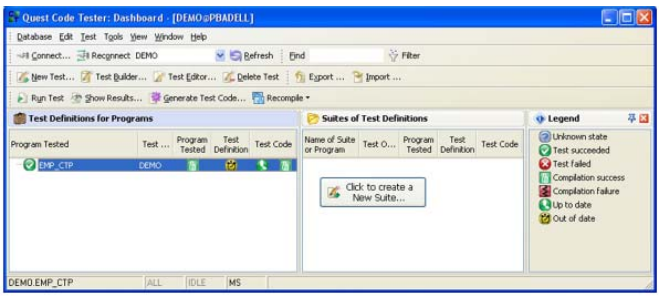
En ik moet een paar procedures maken die de DML voor me uitvoeren.

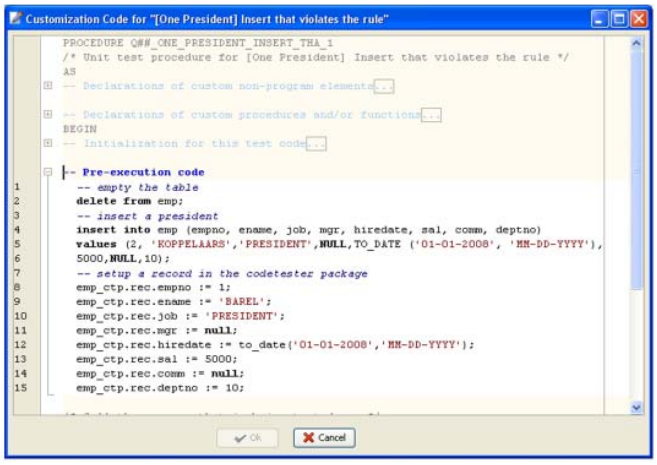
```
  PROCEDURE ins;
  PROCEDURE upd;
  PROCEDURE del;
```

De implementatie van deze procedures is als volgt:

```
PROCEDURE ins
IS
BEGIN
  -- insert the values in the rec record into the table
  INSERT INTO EMP (
                EMPNO,
                ENAME,
                JOB,
                MGR,
                HIREDATE,
                SAL,
                COMM,
                DEPTNO
               ) VALUES (
                rec.EMPNO,
                rec.ENAME,
                rec.JOB,
                rec.MGR,
                rec.HIREDATE,
                rec.SAL,
                rec.COMM,
                rec.DEPTNO
               );
END ins;
PROCEDURE upd
IS
BEGIN
  -- update the values in the table with the values in the rec record
  -- based on the primary key
  UPDATE EMP
     SET
         EMPNO = rec.EMPNO,
         ENAME = rec.ENAME,
         JOB = rec.JOB,
         MGR = rec.MGR,
         HIREDATE = rec.HIREDATE,
         SAL = rec.SAL,
         COMM = rec.COMM,
         DEPTNO = rec.DEPTNO
   WHERE
         EMPNO = rec.EMPNO
   ;
END upd;
PROCEDURE del
IS
BEGIN
  -- remove a record from the table
  -- based on the primary key
  DELETE FROM EMP
   WHERE
         EMPNO = rec.EMPNO
  ;
END del;
```

Nu kan ik een paar tests maken in CodeTester (dit is de pre-execution code, die moet worden toegevoegd in CodeTester):



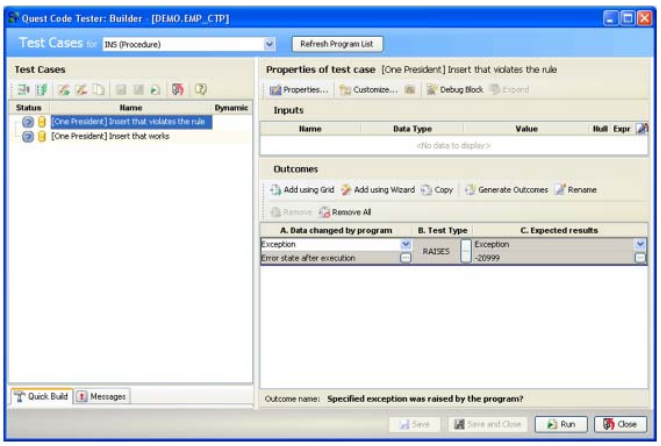


• ins

```
[One President] Insert that violates the rule   -- Pre-execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','PRESIDENT',NULL,TO_DATE ('01-01-2008 ,
'MM-DD-YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
  emp_ctp.rec.ename := 'BAREL';
  emp_ctp.rec.job := 'PRESIDENT';
  emp_ctp.rec.mgr := null;
  emp_ctp.rec.hiredate := to_date('01-01-2008 ,'MM-DD-YYYY');
  emp_ctp.rec.sal := 5000;
  emp_ctp.rec.comm := null;
  emp_ctp.rec.deptno := 10;
[One President] Insert that works   -- Pre-execution code
  -- empty the table
  delete from emp;
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
  emp_ctp.rec.ename := 'BAREL';
  emp_ctp.rec.job := 'PRESIDENT';
  emp_ctp.rec.mgr := null;
  emp_ctp.rec.hiredate := to_date('01-01-2008 ,'MM-DD-YYYY');
  emp_ctp.rec.sal := 5000;
  emp_ctp.rec.comm := null;
  emp_ctp.rec.deptno := 10;
```

• upd

```
[One President] Update that fails (promote to president)   -- Pre-
execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (1, 'BAREL','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','PRESIDENT',NULL,TO_DATE ('01-01-2008 ,
'MM-DD-YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
  emp_ctp.rec.ename := 'BAREL';
  emp_ctp.rec.job := 'PRESIDENT';
  emp_ctp.rec.mgr := null;
  emp_ctp.rec.hiredate := to_date('01-01-2008 ,'MM-DD-YYYY');
  emp_ctp.rec.sal := 5000;
  emp_ctp.rec.comm := null;
  emp_ctp.rec.deptno := 10;
[One President] Update that works (demote the president)   -- Pre-
execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (1, 'BAREL','PRESIDENT',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
```




```
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
  emp_ctp.rec.ename := 'BAREL';
  emp_ctp.rec.job := 'MANAGER';
  emp_ctp.rec.mgr := null;
  emp_ctp.rec.hiredate := to_date('01-01-2008 ,'MM-DD-YYYY');
  emp_ctp.rec.sal := 5000;
  emp_ctp.rec.comm := null;
  emp_ctp.rec.deptno := 10;
[One President] Update that works (promote to president)   -- Pre-
execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (1, 'BAREL','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
  emp_ctp.rec.ename := 'BAREL';
  emp_ctp.rec.job := 'PRESIDENT';
  emp_ctp.rec.mgr := null;
  emp_ctp.rec.hiredate := to_date('01-01-2008 ,'MM-DD-YYYY');
  emp_ctp.rec.sal := 5000;
  emp_ctp.rec.comm := null;
  emp_ctp.rec.deptno := 10;
```

• del

```
[One President] Delete that works (remove the manager)   -- Pre-
execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (1, 'BAREL','PRESIDENT',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 2;
```

```
[One President] Delete that works (remove the president)   -- Pre-
execution code
  -- empty the table
  delete from emp;
  -- insert a president
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (1, 'BAREL','PRESIDENT',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  values (2, 'KOPPELAARS','MANAGER',NULL,TO_DATE ('01-01-2008 , 'MM-DD-
YYYY'),
  5000,NULL,10);
  -- setup a record in the codetester package
  emp_ctp.rec.empno := 1;
```

Natuurlijk kan ik deze code steeds met de hand schrijven als ik hem nodig heb, maar als programmeur ben ik in principe lui en houd ik er niet van om hetzelfde steeds maar opnieuw te doen. Dat is precies waar CodeGen erg goed in is.
CodeGen is een gratis beschikbare tool om op basis van gegevens in de databasetekst bestanden te genereren. Dit kan documentatie zijn, Java, C#, Delphi source code en natuurlijk ook PL/SQL code. De tool is ontwikkeld door Steven Feuerstein en de generatiekracht vormt ook de basis voor CodeTester (de code voor de testcases wordt gegenereerd).
Door gebruik te maken van de generatietool kan ik op eenvoudige wijze de packages maken gebaseerd op de tabel structuur.

(De ctp extensie in de package naam betekent Code Tester Package)
CodeGen script om de package header te maken:

```
CREATE OR REPLACE PACKAGE [objname]_ctp
IS
   -- Author  : [thisuser]
   -- Created : [rightnow]
   -- Purpose : CodeTester Package for [objname]
   -- Public type declarations
  TYPE [objname]_rt IS RECORD(
[foreach]col[between],
    [colname] [objname].[colname]%type
[endforeach]
  );
   -- Public variable declarations
  rec                                          [objname]_rt;
   -- Public function and procedure declarations
  PROCEDURE ins;
  PROCEDURE upd;
  PROCEDURE del;
END [objname]_ctp;
```

CodeGen script om de package body te maken:

```
CREATE OR REPLACE PACKAGE BODY [objname]_ctp
IS
   -- Function and procedure implementations
  PROCEDURE initialization
```

```
  IS
  BEGIN
    NULL;
  END initialization;
  PROCEDURE ins
  IS
  BEGIN
    -- insert the values in the rec record into the table
    INSERT INTO [objname] (
[foreach]col[between],
                   [colname]
[endforeach]
                  ) VALUES (
[foreach]col[between],
                   rec.[colname]
[endforeach]
                 );
  END ins;
  PROCEDURE upd
  IS
  BEGIN
    -- update the values in the table with the values in the rec
record
    -- based on the primary key
    UPDATE [objname]
       SET
[foreach]col[between],
            [colname] = rec.[colname]
[endforeach]
     WHERE
[foreach]pkycol[between],
            [colname] = rec.[colname]
[endforeach]
     ;
  END upd;
  PROCEDURE del
  IS
  BEGIN
    -- remove a record from the table
    -- based on the primary key
    DELETE FROM [objname]
     WHERE
[foreach]pkycol[between],
            [colname] = rec.[colname]
[endforeach]
    ;
  END del;
BEGIN
  initialization;
END [objname]_ctp;
```

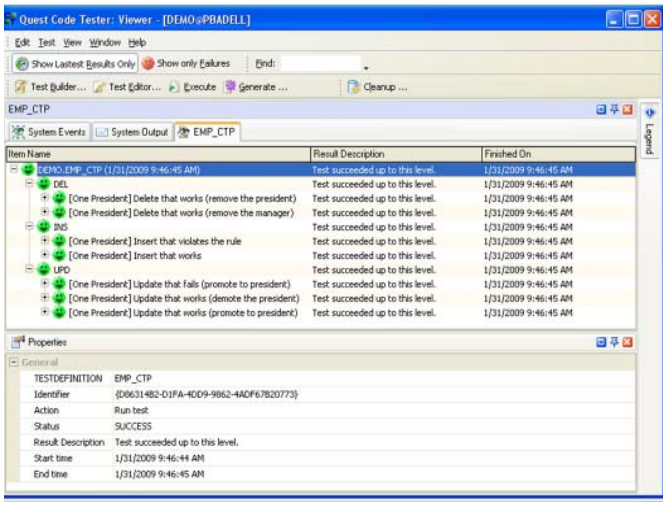
De complete (gegenereerde) code voor het package is:
Specificatie:

```
CREATE OR REPLACE PACKAGE EMP_ctp
IS
  -- Author  : DEMO
  -- Created : Jan 31, 2009 1:23:45
  -- Purpose : CodeTester Package for EMP
  -- Public type declarations
  TYPE EMP_rt IS RECORD(
    EMPNO EMP.EMPNO%type,
    ENAME EMP.ENAME%type,
    JOB EMP.JOB%type,
    MGR EMP.MGR%type,
    HIREDATE EMP.HIREDATE%type,
    SAL EMP.SAL%type,
    COMM EMP.COMM%type,
    DEPTNO EMP.DEPTNO%type
  );
  -- Public variable declarations
  rec                                    EMP_rt;
  -- Public function and procedure declarations
  PROCEDURE ins;
  PROCEDURE upd;
  PROCEDURE del;
END EMP_ctp;
```

Implementatie:

```
CREATE OR REPLACE PACKAGE BODY EMP_ctp
IS
  -- Function and procedure implementations
  PROCEDURE initialization
  IS
  BEGIN
    NULL;
  END initialization;
  PROCEDURE ins
  IS
  BEGIN
    -- insert the values in the rec record into the table
    INSERT INTO EMP (
                  EMPNO,
                  ENAME,
                  JOB,
                  MGR,
                  HIREDATE,
                  SAL,
                  COMM,
                  DEPTNO
                 ) VALUES (
                  rec.EMPNO,
                  rec.ENAME,
                  rec.JOB,
                  rec.MGR,
                  rec.HIREDATE,
                  rec.SAL,
                  rec.COMM,
                  rec.DEPTNO
                );
  END ins;
  PROCEDURE upd
  IS
  BEGIN
    -- update the values in the table with the values in the rec
record
    -- based on the primary key
    UPDATE EMP
       SET
           EMPNO = rec.EMPNO,
           ENAME = rec.ENAME,
           JOB = rec.JOB,
           MGR = rec.MGR,
           HIREDATE = rec.HIREDATE,
           SAL = rec.SAL,
           COMM = rec.COMM,
           DEPTNO = rec.DEPTNO
     WHERE
           EMPNO = rec.EMPNO
     ;
  END upd;
  PROCEDURE del
  IS
  BEGIN
    -- remove a record from the table
    -- based on the primary key
    DELETE FROM EMP
     WHERE
           EMPNO = rec.EMPNO
    ;
  END del;
BEGIN
  initialization;
END EMP_ctp;
```

Na het runnen van de tests in CodeTester krijg ik het volgende resultaat:



Dit betekent dat RuleGen-regels getest kunnen worden door gebruik te maken van CodeTester om de tests uit te voeren en CodeGen om de eenvoudige package rondom de tabel die getest wordt te maken. Het zal wel wat tijd kosten om alle business rules op de database vast te leggen, maar zoals Toon Koppelaars zegt: "Dat is waar ze horen. Het is waarschijnlijk de enige constante factor in een systeem." En, als je de regels op tabelniveau vastlegt, dan maakt het niet uit welk front-end je er tegenaan bouwt en in welke taal dit gebeurt. Er zal ook wel wat tijd gaan zitten in het maken van de testgevallen, maar ze zijn in ieder geval vastgelegd en eenvoudig te herhalen.
Let op: Alle andere regels moeten ook valide zijn om een correct resultaat te krijgen. Probeer slechts één regel per keer te testen. Als je een regel aan de RuleGen repository toevoegt, dan is het mogelijk dat je bepaalde bestaande testgevallen moet aanpassen om de wijziging mee te nemen.

Patrick Barel, AMIS Services BV