SQL Assertions / Declarative multi-row constraints

[edit: ^^ Grey vote-up/down arrows will appear after login to OTN (create account here: https://profile.oracle.com/myprofile/account/create-account.jspx) ^^]

We are considering building support for the **CREATE ASSERTION** command in a next release of the Oracle database. Assertions have been part of the SQL standard since SQL-92. You can find the BNF definition for SQL assertions here: https://github.com/ronsavage/SQL/blob/master/sql-92.bnf (search for "assertion definition").

SQL assertions can be used to implement what's commonly called **cross-row constraints**, or **multi-table check constraints**. In short a SQL assertion is a CHECK constraint at the database level that is allowed to contain queries. Support request for SQL assertions has come up on Asktom several times:
https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:21389386132607
https://asktom.oracle.com/pls/apex/f?p=100:11:0::::p11_question_id:4233459000346171405
https://asktom.oracle.com/pls/apex/f?p=100:11:0::::P11_QUESTION_ID:698031000346429496

Any arbitrary (static) constraint can be specified as a SQL assertion. With support for SQL assertions, there would be no longer a need to build the notoriously complex and error-prone database triggers for cross-row constraints. Once a SQL assertion has been declared to the RDBMS, it is the task of the RDBMS to ensure its continued validity during transactions that change the involved tables/columns.

A few examples.

This SQL statement creates an assertion to demand that there's no more than a single president among the employees:

```
create assertion AT_MOST_ONE_PRESIDENT as CHECK
((select count(*)
    from EMP e
   where e.JOB = 'PRESIDENT') <= 1
)
```

This SQL statement creates an assertion to demand that Boston based departments do not employ trainers:

```
create assertion NO_TRAINERS_IN_BOSTON as CHECK
   (not exists
    (select 'trainer in Boston'
       from EMP e, DEPT d
      where e.DEPTNO = d.DEPTNO
        and e.JOB    = 'TRAINER'
        and d.LOC    = 'BOSTON')
   )
```

This SQL statement creates an assertion to demand that vacation records cannot be outside of one's employment period:

```
create assertion VACATION_DURING_EMPLOYMENT as CHECK
(not exists
     (select 'vacation outside employment'
        from EMP e
            ,EMP_VACATION ev
       where e.EMPNO = ev.EMPNO
         and (ev.FIRST_DATE < e.HIRE_DATE or
              ev.LAST_DATE  > e.TERMINATION_DATE))
   )
```

This SQL statement creates an assertion to demand that every department employs a clerk:

```
create assertion AT_LEAST_ONE_CLERK_PER_DEPT as CHECK
(not exists
        (select 'a department without a clerk'
           from DEPT d
          where not exists
              (select 'a clerk in d'
                 from EMP e
                where e.DEPTNO = d.DEPTNO
                  and e.JOB    = 'CLERK'))
    ) deferrable initially deferred
```

This SQL statement creates an assertion to disallow suppliers based in cities of Albany, Palo Alto, or Portland from supplying, in quantities higher than 50, all the parts that are red or cost $10.00 or more:

```
create assertion AllPartSupp as CHECK
   (not exists
     (select 'an s shipping all parts'
        from SUPPLIER s
       where s.CITY in ('Albany', 'Palo Alto', 'Portland')
         and not exists
               (select 'a p not shipped'
                  from PART p
                 where (p.COLOR = 'red' or p.PRICE >= 10)
                   and not exists
                        (select 'a connecting sh'
                           from SHIPMENT sh
                          where sh.QUANTITY > 50
                            and sh.SNO = s.SNO
```

```
                    and sh.PNO = p.PNO)))
  )
```

The implementation of SQL assertions would be such that only when a transaction changes involved data in such a manner that it could potentially violate the SQL assertion, would the RDBMS perform a re-validation. For instance, the first example above concerning the number of presidents, would not be revalidated on insert of a CLERK, as the SQL assertion is immune to this kind of insert. Furthermore, whenever this is possible, the RDBMS would perform a **delta-check** and not reevaluate the whole SQL assertion expression. For instance, for the second example above concerning the trainers not allowed in Boston, the RDBMS would only revalidate department, say 10, if a trainer were to be inserted into department 10.

As can be seen in the fourth example above, SQL assertions could be defined such that semantically the revalidation takes place at the end of the transaction, thereby allowing temporary violations during a transaction. Such support for SQL assertions would also undo the need for a before-commit trigger, which has been proposed here: BEFORE-COMMIT trigger

All the familiar attributes that are currently available for declarative constraints, such as rely/norely, enable/disable, validate/novalidate and the exceptions-into clause, are under consideration to support.

Please let us know whether you would like to have support for SQL assertions in the future.