

Practice Trigger_2

1. Employees receive an automatic increase in salary if the minimum salary for a job is increased to a value larger than their current salary. Implement this requirement through a package procedure called by a trigger on the JOBS table. When you attempt to update the minimum salary in the JOBS table and try to update the employees' salary, the CHECK_SALARY trigger attempts to read the JOBS table, which is subject to change, and you get a mutating table exception that is resolved by creating a new package and additional triggers.
 - a. Update your EMP_PKG package as follows:
 - i. Add a procedure called SET_SALARY that updates the employees' salaries.
 - ii. The SET_SALARY procedure accepts the following two parameters:
 - The job ID for those salaries that may have to be updated, and
 - The new minimum salary for the job ID
 - iii. The procedure sets all the employees' salaries to the minimum for their jobs if their current salaries are less than the new minimum value.
 - b. Create a row trigger named UPD_MINSALARY_TRG on the JOBS table that invokes the EMP_PKG.SET_SALARY procedure, when the minimum salary in the JOBS table is updated for a specified job ID.
 - c. Write a query to display the employee ID, last name, job ID, current salary, and minimum salary for employees who are programmers—that is, their JOB_ID is 'IT_PROG'. Then update the minimum salary in the JOBS table to increase it by \$1,000. What happens?
2. To resolve the mutating table issue, create a JOBS_PKG to maintain in memory a copy of the rows in the JOBS table. Next, modify the CHECK_SALARY procedure to use the package data rather than issue a query on a table that is mutating to avoid the exception. However, you must create a BEFORE INSERT OR UPDATE statement trigger on the EMPLOYEES table to initialize the JOBS_PKG package state before the CHECK_SALARY row trigger is fired.
 - a. Create a new package called JOBS_PKG with the following specification:

```
PROCEDURE initialize;
FUNCTION get_minsalary(jobid VARCHAR2) RETURN NUMBER;
FUNCTION get_maxsalary(jobid VARCHAR2) RETURN NUMBER;
PROCEDURE set_minsalary(jobid VARCHAR2,min_salary NUMBER);
PROCEDURE set_maxsalary(jobid VARCHAR2,max_salary NUMBER);
```

Practice (continued)

- b. Implement the body of the JOBS_PKG, as follows:
 - i. Declare a private PL/SQL index-by table called `jobs_tab_type` that is indexed by a string type based on the `JOBS.JOB_ID%TYPE`.
 - ii. Declare a private variable called `jobstab` based on the `jobs_tab_type`.
 - iii. The `INITIALIZE` procedure reads the rows in the `JOBS` table by using a cursor loop, and uses the `JOB_ID` value for the `jobstab` index that is assigned its corresponding row.
 - iv. The `GET_MINSALARY` function uses a `p_jobid` parameter as an index to the `jobstab` and returns the `min_salary` for that element.
 - v. The `GET_MAXSALARY` function uses a `p_jobid` parameter as an index to the `jobstab` and returns the `max_salary` for that element.
 - vi. The `SET_MINSALARY` procedure uses its `p_jobid` as an index to the `jobstab` to set the `min_salary` field of its element to the value in the `min_salary` parameter.
 - vii. The `SET_MAXSALARY` procedure uses its `p_jobid` as an index to the `jobstab` to set the `max_salary` field of its element to the value in the `max_salary` parameter.
- c. Copy the `CHECK_SALARY` procedure from this Practice, Exercise 1 a, and modify the code by replacing the query on the `JOBS` table with statements to set the local `minsal` and `maxsal` variables with values from the `JOBS_PKG` data by calling the appropriate `GET_*SALARY` functions. This step should eliminate the mutating trigger exception.
- d. Implement a `BEFORE INSERT OR UPDATE` statement trigger called `INIT_JOBPKG_TRG` that uses the `CALL` syntax to invoke the `JOBS_PKG.INITIALIZE` procedure to ensure that the package state is current before the DML operations are performed.
- e. Test the code changes by executing the query to display the employees who are programmers, and then issue an update statement to increase the minimum salary of the `IT_PROG` job type by 1000 in the `JOBS` table, followed by a query on the employees with the `IT_PROG` job type to check the resulting changes. Which employees' salaries have been set to the minimum for their jobs?

Practice (continued)

3. Because the `CHECK_SALARY` procedure is fired by the `CHECK_SALARY_TRG` before inserting or updating an employee, you must check whether this still works as expected.
 - a. Test this by adding a new employee using `EMP_PKG.ADD_EMPLOYEE` with the following parameters: ('Steve', 'Morse', 'SMORSE', and `sal => 6500`). What happens?
 - b. To correct the problem encountered when adding or updating an employee:
 - i. Create a `BEFORE INSERT OR UPDATE` statement trigger called `EMPLOYEE_INITJOBS_TRG` on the `EMPLOYEES` table that calls the `JOBS_PKG.INITIALIZE` procedure.
 - ii. Use the `CALL` syntax in the trigger body.
 - c. Test the trigger by adding employee Steve Morse again. Confirm the inserted record in the `EMPLOYEES` table by displaying the employee ID, first and last names, salary, job ID, and department ID.