

[PL/SQL Exercise - triggers](#)

1. Using triggers to maintain business rules

Suppose that the Middlesex Transport has a rule stating that a bus driver's salary cannot be changed by more than 20% of the original salary. Create a trigger 'salary_change_monitoring' to enforce this constraint. The trigger fires whenever there is an update to the Busdriver table and outputs a suitable error message when the rule is violated.

2. Creating triggers to prevent updates and deletions

In the BusDrivers' database, we can see that the rows in the Depot table are often referenced by many child rows in a number of other tables (e.g., Bus, Cleaner and Busdriver). Although there are FOREIGN KEY constraints declared on the child tables to maintain the referential integrity, we can still define a trigger in the parent table (i.e., Depot) to stop any attempt to change the name of the depot and/or to remove any of the depot rows. This is corresponding to the business rule stating that once a depot is established, it will be there 'forever' and will not be allowed to change name (although unrealistic, we assume that such a rule is necessary).

Write appropriate PL/SQL statements to create the trigger. Note that the trigger you create is a **statement level trigger** so the 'for each row' statement should not be used. After the trigger is created, try to change the name of some depots and delete a row from depot, and see what will happen.

3. Creating triggers to maintain data validity

In your previous database module you may have encountered CHECK constraints. This is similar to a validation rule and is an option in the CREATE TABLE command whereby you can specify what data may be entered into a particular column. So if we wanted to add a constraint in an Cleaner table that salary must be within certain limits we could create the table thus:

```
create table Cleaner
  (cno          varchar2(5),
   cname        varchar2(20),
   csalary      number(6,2),
   dno          varchar2(5),
   constraint pk_clno primary key(cno),
   constraint fk_deno1 foreign key(dno) references depot(dno),
   check (csalary > 0 and csalary < 5000) );
```

Here any salary that is less than zero and greater than 5000 will cause a violation of the constraint.

Applying the CHECK constraint, however, we would not know whether the salary is greater than 5000 or smaller than 0 (i.e., a negative number).

We can create a trigger instead of a CHECK constraint, which can tell us how the restriction on 'csalary' is violated. Whenever the value of 'csalary' is beyond the valid range (0 – 5000), the trigger will generate an error message informing users whether it is greater than 5000 or a negative number. (If a check constraint already exists it must be dropped first.)

Write PL/SQL statements to create the trigger, and use some SQL **update** and **insert** statements to test it.