Triggers

The following relations are given (primary keys are underlined, optional attributes are denoted with *):

ATHLETE(AthleteCode, TeamName)
ATHLETE_ARRIVAL(AthleteCode, Time)
TEAM_ARRIVAL(TeamName, NumberArrivedAthletes)
RANKING(AthleteCode, Position, Time)

Write triggers to *update* TEAM_ARRIVAL and RANKING tables when a new row is inserted in the ATHLETE_ARRIVAL table.

- For the update of the TEAM_ARRIVAL table, consider also the case of a team not yet inserted in the table.
- For the update of RANKING table, consider that the Time field can assume the same value for two different athletes.

Draft solution

```
CREATE OR REPLACE TRIGGER UPDATE_RANKING
AFTER INSERT ON ATHLETE_ARRIVAL
FOR EACH ROW
DECLARE
        APos NUMBER;
        X NUMBER;
        ATime NUMBER;
        Draw BOOLEAN;
BEGIN
        -- verify if the ranking is empty and compute the position for the new athlete
        SELECT MAX(Time), MAX(Position) INTO ATime, X
        FROM RANKING
        WHERE Time <= :NEW.Time;

        IF (X IS NULL) THEN
        --- ranking empty or athlete in the first position without any other athlete with the same time
                APos := 1;
                Draw := FALSE;
        ELSE
        --- verify if there is another athlete with the same time
                IF (ATime < :NEW.Time) THEN
                        APos := X +1;
                        Draw := FALSE;
                ELSE
                        APos := X;
                        Draw := TRUE;
                END IF;
        END IF;


        INSERT INTO RANKING (Position, AthleteCode, Time)
        VALUES (APos, :NEW.AthleteCode, :NEW.Time);

        IF (Draw = FALSE) THEN
                UPDATE RANKING SET Position=Position+1
                WHERE Time > :NEW.Time;
        END IF;
END;


CREATE OR REPLACE TRIGGER UPDATE_TEAM_ARRIVAL
AFTER INSERT ON ATHLETE_ARRIVAL
FOR EACH ROW
DECLARE
        Team VARCHAR(10);
        X NUMBER;
BEGIN
        ---read the team of the new athlete
        SELECT TeamName INTO Team
        FROM ATHLETE
        WHERE AthleteCode = :NEW.AthleteCode;

        ---check if the new athlete is the first athlete for the team
        SELECT COUNT(*) INTO X
        FROM TEAM_ARRIVAL
        WHERE TeamName = Team;

        IF (X=0) THEN
        ---the new athlete is the first athlete for the team
                INSERT INTO TEAM_ARRIVAL (TeamName, NumberArrivedAthletes)
                VALUES (Team,1);
        ELSE
                UPDATE TEAM_ARRIVAL
                SET NumberArrivedAthletes = NumberArrivedAthletes + 1
                WHERE TeamName= Team;
        END IF;
END;
```

Triggers

The following relations are given (primary keys are underlined, optional attributes are denoted with *):

COURSE(<u>CourseCode</u>, CourseName, Credits)
STUDENT(<u>RegNum</u>, StudentName, YearFirstEnrollment)
EXAM (<u>CourseCode</u>, <u>RegNum</u>, <u>Date</u>, Score)
GRANT_APPLICATION(<u>RegNum</u>, RequestDate)
STUDENT_RANKING(<u>RegNum</u>, TotalPoints)
GRANT_AVAILABILITY(<u>Grant#</u>, CourseCode, TeachingHours)
GRANT_ASSIGNMENT(<u>Grant#</u>, RegNum, TeachingHours)
NOTIFICATION(<u>Not#</u>, Grant#, RegNum*, Message)

The trigger application deals with the assignment of student grants for supporting teaching activities. Students applying for a student grant are inserted into a ranking (table STUDENT_RANKING). When a new grant becomes available (table GRANT_AVAILABILITY), the student recipient of the grant is selected from the ranking. The same student may be the recipient of more than one grant, provided that she/he does not exceed 150 hours of teaching activities. Write the triggers managing the following tasks for the automatic assignment of student grants.

(1) *Grant application.* A student applies for the assignment of a student grant (insertion into table GRANT_APPLICATION). The application is accepted if (i) the student has acquired at least 120 credits on passed exams (i.e., on exams with score above 17) and (ii) the student is not yet in the ranking (i.e., in table STUDENT_RANKING). If any of the two requirements is not satisfied, the application is rejected. If the application is accepted, the student is inserted in the ranking. The total points (attribute TotalPoints) of the student are given by the average score computed only on passed exams divided by the years elapsed from the student first enrollment (the current year is given by the variable YEAR(SYSDATE)).

(2) *Grant assignment.* When a new grant becomes available (insertion into table GRANT_AVAILABILITY), the recipient student is selected from the ranking. The recipient is the student with the highest ranking that satisfies the following requirements: (i) she/he has passed the exam for the course on which the grant is available, and (ii) she/he does not exceed 150 teaching hours overall (including also the new grant). Suppose that at most one student satisfies the above requirements. If the grant is assigned, table GRANT_ASSIGNMENT should be appropriately modified. The result of the assignment process must be notified both in the positive case (the grant is assigned) and in the negative case (no appropriate student is available, in this case the RegNum attribute takes the NULL value). The Not# attribute is a counter, which is incremented for each new notification.

Trigger exercises: Event calendar management

Given the following relational schema (primary keys are underlined):

EVENT(EvID, EventName, EventCategory, EventCost, EventDuration)
CALENDAR(EvID, Date, StartTime, Location)
CATEGORY_SUMMARY(EventCategory, Date, TotalNumberEvents, TotalEventCost)

You are requested to manage the planning of events in the city of Turin for the anniversary of the 150th anniversary of the unification of Italy (Italia 150).
Events belong to different categories (EventCategory attribute), such as exhibitions, debates, screenings, and are characterized by a cost of realization (EventCost attribute). Each event can be repeated several times on different dates. The CALENDAR table shows the planning of events on different days and places in the city. Write triggers to handle the following tasks.

(1) *Updating the table CATEGORY_SUMMARY*. The table CATEGORY_SUMMARY shows, for each category of event and for each date, the total number of events planned and the total cost for their realization. Write the trigger to propagate changes to the CATEGORY_SUMMARY table when a new calendar event is inserted (CALENDAR table insertion).

(2) *Integrity constraint on the maximum cost of the event*. The cost of an event in the film screening category (EventCategory attribute) cannot exceed 1500 euros. If a cost value greater than 1500 is entered in the EVENT table, the EventCost attribute must be assigned a value of 1500. Write the trigger for handling the integrity constraint.

(3) *Constraint on the maximum number of events per date*. No more than 10 events can be scheduled on any given date. Any changes to the table CALENDAR that cause the constraint violation should not be performed.

Trigger Exercises: Leave request management

The following relations are given (primary keys are underlined, optional attributes are denoted with *):

PERSON (RegNumber, Job)
SHIFT (RegNumber, Date, TCode)
SHIFT-TYPE (TCode, StartTime, Duration)
LEAVE-REQUEST(RCode, RegNumber, Date)
NOTIFICATION(RegNumber, Date, RequestOutcome)

Write a trigger to manage one-day leave requests by people working in a hospital (insert into the LEAVE-REQUEST table).

A leave request is accepted if the person requesting it is not on duty in the requested date (SHIFT table). If instead the person is on duty, the request is accepted only if another person is available to fill the requester's shift. Otherwise, the request is declined. A person may substitute another person on a shift if they both have the same job and the substitute is not on duty in the same date.

The outcome of the request (accepted or declined) must be notified by means of an insert into the NOTIFICATION table.