Corrigé du TD BDA (Séance n° 2)
Triggers

Exercice 1

Q1 Lors de la création des contrôles, on souhaite que :
- lorsque la date de fin n'est pas spécifiée alors elle doit être égale à la date de début ;
- la date de fin doit être égale à la date de début pour les types de contrôle concernés (tests, tests rattrapage, interrogations) ;
- par défaut la note maximale du barème doit être à 20.

Définir un trigger se déclenchant automatiquement à l'insertion de nouveaux contrôles, ce qui permet de donner des valeurs par défaut. Il faut également penser aux opérations de modification affectant les attributs `TypeC, DateDeb, DateFin ou NoteMax.`

```
CREATE OR REPLACE TRIGGER Ins_controle
BEFORE INSERT OR UPDATE OF TypeC, DateDeb, dateFin, NoteMax ON CONTROLE
FOR EACH ROW
BEGIN
   IF :new.TYPEC IN ('TEST', 'TEST RATTRAPAGE', 'INTERROGATION') OR
   :NEW.DateFin IS NULL  THEN
        :NEW.DateFin := :NEW.DateDeb ;
   END IF ;
   IF :NEW.NoteMax IS NULL  THEN
        :NEW.NoteMax := 20 ;
   END IF ;
END;
```

Q2 Lors de la saisie des notes, on veut pouvoir calculer la note définitive à partir de l'attribut Note (qui n est pas évalué sur 20 mais sur `NoteMax` de `CONTROLE`). Donnez le Trigger associé. NoteDef = Note *20 / NoteMax

```
CREATE OR REPLACE TRIGGER Ins_notation
BEFORE INSERT OR UPDATE OF Note ON NOTATION
FOR EACH ROW
DECLARE
 Bareme CONTROLE.NoteMax%TYPE;
BEGIN
   SELECT NoteMax INTO bareme FROM CONTROLE WHERE IdC = :NEW.IdC;
  :NEW.NoteDef := :NEW.Note * 20 / bareme;
   EXCEPTION
   WHEN NO_DATA_FOUND THEN
     RAISE_APPLICATION_ERROR(-20000,'Le contrôle est inexistant');
END;
```

Exercice 2

Q1 On souhaite mettre en œuvre la contrainte suivante : un abonné ne peut faire un nouvel emprunt que si son nombre d'emprunts courants est inférieur à 5.
Ecrire un trigger permettant de vérifier la contrainte automatiquement si le nombre d'emprunts est trop grand et de faire échouer l'insertion.

```
CREATE TRIGGER t_autoriser_emprunt
BEFORE INSERT ON EMPRUNT
FOR EACH ROW
DECLARE
Nb ABONNE.NbEmprunt%TYPE ;
BEGIN
SELECT NbEmprunt INTO Nb FROM ABONNE WHERE IdA = :NEW.IdA ;
IF Nb >= 5
   THEN RAISE_APPLICATION_ERROR(-20000, 'Trop d''emprunts en cours');
END IF ;
EXCEPTION
     WHEN NO_DATA_FOUND THEN
     RAISE_APPLICATION_ERROR(-20000, 'Pas d''abonné de ce numéro')
END ;
```

Q2 Au fur et à mesure que des emprunts sont créés ou achevés (valorisation de l'attribut `DateRet`), on souhaite mettre à jour l'attribut calculé `NbEmprunt` dans la relation `ABONNE`. On suppose qu'il est impossible de modifier par **`UPDATE`** l'attribut `DateRet` en le mettant à **`NULL`**. On lui donnera forcément la date du jour.

Définir un trigger sur la relation EMPRUNT permettant d'effectuer la propagation automatique des mises à jour sur l'attribut `NbEmprunt` qui est incrémenté après la création d'un emprunt et décrémenté dès qu'un emprunt est terminé.

```
CREATE TRIGGER t_calcul_nbemprunt
AFTER INSERT OR UPADTE OF DateRet ON EMPRUNT
FOR EACH ROW
BEGIN
IF INSERTING THEN
     UPDATE ABONNE SET NbEmprunt = NbEmprunt + 1
     WHERE IdA = :NEW.IdA ;
ELSE
     UPDATE ABONNE SET NbEmprunt = NbEmprunt - 1
     WHERE IdA = :OLD.IdA ; (ou :NEW.IdA)
END IF ;
END ;
```

Q3 Lorsqu'un livre est réservé et qu'un exemplaire de ce livre est rapporté à la bibliothèque par un abonné qui l'avait emprunté, on souhaite effectuer les opérations suivantes :

- Contacter l'abonné prioritaire parmi tous ceux qui ont réservé le livre considéré. L'abonné prioritaire est celui dont la date de réservation est la plus ancienne. S'il y en avait plusieurs, le choix est fait, parmi eux, de manière arbitraire.
- Mettre à jour l'attribut `Contact` de la relation `RESERV` qui prendra la valeur 'Oui' pour la réservation sélectionnée.

Ecrire un trigger sur la relation `EMPRUNT` permettant d'effectuer le traitement demandé de manière automatique dès qu'un emprunt est achevé (valorisation de l'attribut `DateRet`).
Vous supposez qu'il existe une procédure `Avertir` en admettant en paramètres le nom, le prénom et l'adresse de l'abonné ainsi que le titre du livre. Cette procédure éditera un courrier pour l'abonné.

```
CREATE TRIGGER t_avert_reservation
AFTER UPDATE OF DateRet ON EMPRUNT
FOR EACH ROW
DECLARE
     NomA  ABONNE.Nom%TYPE ;
     PrenomA  ABONNE.Prenom%TYPE ;
     AdresseA  ABONNE.Adresse%TYPE ;
     TitreR  LIVRE.Titre%TYPE ;

     CURSOR c_reserv IS
          SELECT * FROM RESERV WHERE Contact <> 'OUI' AND IdL IN
              (SELECT IdL FROM EXEMPLAIRE WHERE CodBar = :NEW.CodBar)
          ORDER BY DateR ;

     reservation  RESERV%ROWTYPE ;

BEGIN
     OPEN c_reserv ;
     FETCH c_reserv INTO reservation ;
     IF c_reserv%FOUND THEN
     BEGIN
          SELECT Nom, Prenom, Adresse INTO NomA, PrenomA, AdresseA
          FROM ABONNE WHERE IdA = reservation.IdA ;

          SELECT Titre INTO TitreR
          FROM LIVRE WHERE IdL = reservation.IdL ;

          Avertir(NomA, PrenomA, AdresseA, TitreR) ;

          UPDATE RESERV SET Contact = 'OUI'
          WHERE IdA = reservation.IdA AND IdL = reservation.IdL
          AND DateR = reservation.DateR ;
     END ;
     ELSE
          DBMS_OUTPUT.PUT_LINE('Pas de réservation pour ce livre') ;
     END IF ;
     CLOSE c_reserv ;
END ;
```

Version sans curseur (avec rownum)

```
CREATE TRIGGER t_avert_reservation
AFTER UPDATE OF DateRet ON EMPRUNT
FOR EACH ROW
DECLARE
     NomA  ABONNE.Nom%TYPE ;
     PrenomA  ABONNE.Prenom%TYPE ;
     AdresseA  ABONNE.Adresse%TYPE ;
     TitreR  LIVRE.Titre%TYPE ;

     reservation  RESERV%ROWTYPE ;




BEGIN
     SELECT * INTO reservation
     FROM (SELECT * FROM RESERV WHERE Contact <> 'OUI' AND IdL IN
               (SELECT IdL FROM EXEMPLAIRE WHERE CodBar = :NEW.CodBar)
          ORDER BY DateR))
     WHERE ROWNUM = 1 ;

     SELECT Nom, Prenom, Adresse INTO NomA, PrenomA, AdresseA
     FROM ABONNE WHERE IdA = reservation.IdA ;

     SELECT Titre INTO TitreR
     FROM LIVRE WHERE IdL = reservation.IdL ;

     Avertir(NomA, PrenomA, AdresseA, TitreR) ;

     UPDATE RESERV SET Contact = 'OUI'
     WHERE IdA = reservation.IdA AND IdL = reservation.IdL
     AND DateR = reservation.DateR ;


EXECEPTION
     WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE('Pas de réservation pour ce livre') ;
END ;
```