Exercices sur les contraintes et déclencheurs

Les exercices de ce laboratoire sont à réaliser dans la base de données Hôpital.

Contrainte 1

Ajouter, dans la table Patients, les contraintes applicatives suivantes :

- Le sexe doit toujours être connu et doit être égal à `F` ou `M`
- L'état civil doit toujours être connu et doit être égal à `C`, `M`, `D` ou `V`
- Le groupe sanguin doit être égal à `A`, `B`, `O` ou `AB`
- Les deux derniers caractères du compte bancaire sont le reste de la division des dix premiers caractères par 97
- Construire un jeu de commandes permettant de tester ces contraintes.

Afficher toutes les contraintes spécifiées sur la table Patients.

```
-- Afficher les contraintes définies sur la table Patients

SELECT CONSTRAINT_NAME,
       CONSTRAINT_TYPE,
       SEARCH_CONDITION,
       STATUS,
       DEFERRABLE,
       DEFERRED,
       VALIDATED
FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'PATIENTS';

-- Ajouter les nouvelles contraintes :

ALTER TABLE Patients
    ADD CONSTRAINT PatientsSexeNotNull CHECK (Sexe IS NOT NULL)
    ADD CONSTRAINT PatientsSexeMF CHECK (Sexe IN ('M', 'F'))
    ADD CONSTRAINT PatientsEtatCivilNotNull
        CHECK (EtatCivil IS NOT NULL)
    ADD CONSTRAINT PatientsEtatCivilCM
        CHECK (EtatCivil IN ('C', 'M'))
    ADD CONSTRAINT PatientsGrpSanguin
        CHECK (GrpSanguin IN ('A', 'B', 'O', 'AB'))
    ADD CONSTRAINT PatientsCptBancaire
        CHECK (SUBSTR(CptBancaire, 11, 2) =
               MOD(SUBSTR(CptBancaire, 1, 10), 97));

-- Réexécuter la requête pour afficher les contraintes définies sur la table Patients
et voir ce qui a été ajouté.
```

Déclencheur 1

Écrire et tester un déclencheur `PatientsInsertNrSis` qui permet d'initialiser automatiquement le NrSis d'un patient.
Le `NrSis` est constitué de la date de naissance du patient `YYMMDD` suivi d'un numéro de 5 chiffres. Soit, par exemple `75123000001`. Ce numéro de 5 chiffres est le dernier numéro enregistré dans la table Patients + 1. Cette
contrainte sera vérifiée lors de l'insertion d'un patient.

```
CREATE OR REPLACE TRIGGER PatientsNrSis
    BEFORE INSERT
    ON Patients
    FOR EACH ROW
DECLARE
    DateNaissNull Exception;
BEGIN
    IF(:New.DateNaissance IS NULL)
    THEN
        Raise DateNaissNull;
    END IF;
    SELECT TO_CHAR(:NEW.DateNaissance, 'YYMMDD') ||
           COALESCE(LPAD(TO_CHAR(1 + (SELECT MAX(Substr(NrSis, 7))
                                      FROM Patients)), 5, '0'), '00001')
    INTO :NEW.NrSis
    FROM DUAL;

EXCEPTION
    WHEN DateNaissNull THEN
        Raise_application_error(-20001,
                                'La date de naissance ne peut être nulle');
    WHEN OTHERS THEN Raise;
END;
```

> Faire un test avec toutes valeurs correctes, sans préciser de `NrSis` :
>
> ```
> INSERT INTO PAtients Values (NULL,'De Dijcker', 'Sebastien',
> 'M','M','BE',TO_DATE('15051990','DDMMYYYY'),NULL,NULL,NULL,NULL,NULL,179);
> ```
>
> Faire un test avec toutes valeurs correctes en précisant un NrSis bidon, vérifier résultat :
>
> ```
> INSERT INTO PAtients Values ('123456789','De Dijcker', 'Sebastien',
> 'M','M','BE',TO_DATE('15051990','DDMMYYYY'),NULL,NULL,NULL,NULL,NULL,179);
> ```
>
> Faire un test avec date de naissance nulle :

```
INSERT INTO PATIENTS
Values ('123456789', 'De Dijcker', 'Sebastien', 'M', 'M', 'BE', NULL, NULL, NULL,
NULL, NULL, NULL, 179);
```

> Idéalement, il faudrait également faire un test quand il n'y a pas encore de patients, histoire de vérifier le COALESCE ...

Déclencheur 2

Écrire et tester un déclencheur `HopitauxServicesNbreMed` qui va permettre de mettre à jour automatiquement la
colonne `NbreMedecins` de la table `HopitauxServices` à chaque insertion, modification ou suppression sur la
table `MedecinsServices`. Construire un jeu de commandes permettant de tester ce déclencheur.

```
CREATE OR REPLACE TRIGGER HopitauxServicesNbreMed
    BEFORE INSERT OR DELETE OR UPDATE OF NrHopital, NrService
    ON MedecinsServices
    FOR EACH ROW
BBEGIN
    IF INSERTING THEN
        UPDATE HopitauxServices
        SET NbreMedecins = COALESCE(NbreMedecins, 0) + 1
        WHERE NrHopital = :new.NrHopital
          AND NrService = :new.NrService;
    ELSIF DELETING THEN
        UPDATE HopitauxServices
        SET NbreMedecins = COALESCE(NbreMedecins, 0) - 1
        WHERE NrHopital = :old.NrHopital
          AND NrService = :old.NrService;
    ELSIF UPDATING THEN
        UPDATE HopitauxServices
        SET NbreMedecins = COALESCE(NbreMedecins, 0) + 1
        WHERE NrHopital = :new.NrHopital
          AND NrService = :new.NrService;
        UPDATE HopitauxServices
        SET NbreMedecins = COALESCE(NbreMedecins, 0) - 1
        WHERE NrHopital = :old.NrHopital
          AND NrService = :old.NrService;
    END IF;
Eend;
```

Ajouter un enregistrement dans `MedecinsServices`, voir changement dans `HopitauxServices`

```
select *
from hopitauxservices;

select *
```

```
from medecins;

select *
from medecinsservices;

insert into medecinsservices
values ('0034000', 'S017', '10003681001');
insert into medecinsservices
values ('0034000', 'S017', '10003780001');
insert into medecinsservices
values ('0034000', 'S017', '10003881001');
```

Supprimer un enregistrement de `MedecinsServices` et voir changement dans `HopitauxServices` (Rollback)

```
delete
from medecinsservices
where nrmedecin = '10003881001';
```

Modifier un enregistrement de `MedecinsServices` (NrHopital et/ou NrService) et voir changements dans `HopitauxServices` (Rollback)

```
update medecinsservices
set nrservice='S030'
WHERE nrmedecin = '10003780001';
```

Déclencheur 3

Écrire et tester un déclencheur PatientsUpdateNrSis qui va permettre de gérer la mise à jour du numéro de SIS du patient
sachant que l'option `ON UPDATE CASCADE` n'existe pas en Oracle. Cette mise à jour du numéro de SIS sera exécutée lors
de la modification de la date de naissance du patient. Construire un jeu de commandes permettant de tester ce
déclencheur.

```
CREATE OR REPLACE TRIGGER PatientsUpdNrSis
    BEFORE UPDATE OF DateNaissance
    ON Patients
    FOR EACH ROW

DECLARE
    DateNaissNull Exception;
BEGIN
    IF (:New.DateNaissance IS NULL)
    THEN
```

```
        Raise DateNaissNull;
    END IF;

    :NEW.NrSis := TO_CHAR(:NEW.DateNaissance, 'YYMMDD') ||
                  SUBSTR(:Old.NrSis, 7);

    UPDATE PatientsMedecins
    SET NrSis = :NEW.NrSis
    WHERE NrSis = :OLD.NrSis;

    UPDATE Allergies
    SET NrSis = :NEW.NrSis
    WHERE NrSis = :OLD.NrSis;

EXCEPTION
    WHEN DateNaissNull THEN
        Raise_application_error(-20001,
                                'La date de naissance ne peut être nulle');
    WHEN OTHERS THEN Raise;
END;
```

Modifier la date de naissance d'un patient et constater les modifications dans les tables `Patients`, `PatientsMedecins` et `Allergies`. Mettre la date de naissance d'un patient à `NULL` et vérifier que l'on récupère
bien le message d'erreur et qu'aucune modification n'est effectuée dans les 3 tables.

```
select *
from patients
where nom like 'De %';
UPDATE PAtients
set datenaissance = to_date('01012000', 'DDMMYYYY')
WHERE nrsis = '90051500680';

UPDATE PAtients
set datenaissance = NULL
WHERE nrsis = '90051500680';
commit;
```

Déclencheur 4

Écrire et tester un déclencheur `PatientsDelNrSIS` qui va permettre de gérer l'option `ON DELETE CASCADE` lors de la
suppression d'un patient. Construire un jeu de commandes permettant de tester ce déclencheur.

```
CREATE OR REPLACE TRIGGER PatientsDelNrSis
    BEFORE DELETE
    ON Patients
    FOR EACH ROW
```

```
DECLARE

BEGIN
    DELETE FROM PatientsMedecins WHERE NrSis = :OLD.NrSis;
    DELETE FROM ALLERGIES WHERE NrSis = :OLD.NrSis;

EXCEPTION
    WHEN OTHERS THEN Raise;
END;
```

> Les instructions `DELETE` ne généreront pas d'exception si aucun tuple n'est supprimé => pas de soucis, rien de plus à faire. Le `DELETE` va générer une erreur si on viole une intégrité référentielle ce qui ne saurait être le cas ici !
> !!! Utiliser `OLD` et non `NEW` : il n'y a pas de `new`, on doit se servir des données de l'enregistrement d'origine que l'on souhaite supprimer ...

```
select *
from patientsmedecins;
delete
from patients
where nrsis = '84090100068';
rollback;
```

Synthèse

Afficher le nom, le type, la date de création et le statut de tous les objets et de toutes les contraintes que vous avez
créés dans votre schéma hôpital.

```
SELECT Object_name as Nom,
       Object_Type as TypeObjet,
       Created     as DateCreation,
       Status
FROM USER_OBJECTS
UNION
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, Last_Change, Status
FROM USER_CONSTRAINTS;
```