

**Département d’informatique et de mathématiques**

**8TRD157 – Bases de données avancées**

Travail pratique #2 : Conception physique

|  |  |
| --- | --- |
| **Étudiant 1** | Samuel Dallaire |
| **Étudiant 2** | Steven Duquette |
| **Date** | 31/10/2018 |

|  |  |
| --- | --- |
| **Rapport** |  |
| **Contraintes** | **/ 35** |
| **Procédures** | **/ 35** |
| **Script de création de tables** | **/ 15** |
| **Question théorique** | **/ 15** |
|  |  |
| **Total** | **/ 100** |

# Règles d’affaires (contraintes)

FOREIGN KEY :

Nous avons utiliser des foreign key pour les tables Employe, Client, Abonnement, Item, Location, Personnage, ParcipantFilm, GenreFilm et PaysProduction. Cette contraite previent la cretient d’un enfant si son parent n’existe pas.

CHECK :

On les retrouve dans les tables Abonnement, Client, Employe, Film, Participant. Elle nous permette de constamment tester certaines conditions nécessaires au bon fonctionnement de notre systèmes. En voici quelques-unes :

CONSTRAINT enumType CHECK( typeParticipant IN ('Acteur','Realisateur',NULL) )

CONSTRAINT bool CHECK(actif=0 OR actif=1)

TRIGGER :

Nos Trigger sont sur nos deux vues, Personne, ainsi que item. Dans le cas des Vue le triggeur permettait de cacher les nos insert nécessaire pour la création de ceux si et de l’applications à appliquer sur ceux-ci

# Opérations à encapsuler

Nous avons encapsulé avec des procédures tous ce qui concerne l’ajout de et les délaites dans la base de données (pour qu’il y ait entre autres l’auto-incrémentassions de la clé primaire) nous avons également fait ça avec les views étant donnée qu’il fallait que l’utilisateur entre exemple personne et client ou personne et employé en même temps.

# Planification des tâches

Nous avons entièrement effectué ce travail en équipe résolvant à deux nos problèmes et s’appuyant l’un l’autre lors des tâches de modifications dans la BD

# Question théorique

Voici nos 2 vue nous permettant d’accéder à une interface plus complète (‘fusion’ de personne et de l’autre table). Par ce fait nous avons été en mesure d’appliquer un trigger spécifique sur l’âge seulement pour le client.

CREATE VIEW ClientView AS

SELECT p.personid,p.prenom,p.nom,p.courriel,c.mdp,c.numclient,c.cartecredit,c.dateexpiration,p.adresse,p.datenaissance

FROM personne p ,client c

WHERE p.personid=c.personid;

commit;

CREATE VIEW EmployeViewView AS

SELECT p.personid,e.numemploye,p.prenom,p.nom,e.identifiant,e.mdp,p.courriel,p.adresse,p.datenaissance

FROM personne p ,employe e

WHERE p.personid=e.personid;

commit;

# Création des TRIGGERS

create or replace TRIGGER deletePersonView

INSTEAD OF DELETE ON clientview

FOR EACH ROW

BEGIN

delete from client where personid=:OLD.personid;

delete from personne where personid=:OLD.personid;

END;

create or replace TRIGGER verificationAgeClient

INSTEAD OF INSERT ON clientview

FOR EACH ROW

DECLARE

ageClient NUMBER;

BEGIN

SELECT MONTHS\_BETWEEN(SYSDATE,:NEW.datenaissance)/12 INTO ageClient FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(ageClient);

IF (ageClient < 18) THEN

RAISE\_APPLICATION\_ERROR(-20000,'L age du client doit être suppérieur a 17.');

ELSE

INSERT INTO personne values (:NEW.personid,:NEW.nom,:NEW.prenom,:NEW.courriel,:NEW.adresse,:NEW.datenaissance);

INSERT INTO client values (:NEW.numclient,:NEW.personid,:NEW.mdp,:NEW.cartecredit,:NEW.dateexpiration);

END IF;

END;

create or replace TRIGGER insertEmployeView

INSTEAD OF INSERT ON employeviewview

FOR EACH ROW

DECLARE

ageClient NUMBER;

BEGIN

INSERT INTO personne values (:NEW.personid,:NEW.nom,:NEW.prenom,:NEW.courriel,:NEW.adresse,:NEW.datenaissance);

INSERT INTO employe values (:NEW.numemploye,:NEW.personid,:NEW.identifiant,:NEW.mdp);

END;

create or replace TRIGGER verificationMail

BEFORE INSERT ON personne

FOR EACH ROW

DECLARE

ageClient NUMBER;

BEGIN

IF(NOT(REGEXP\_LIKE (:NEW.courriel, '^[A-Za-z]+[A-Za-z0-9.]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$'))) THEN

RAISE\_APPLICATION\_ERROR(-20000,'Format mail invalde');

END IF;

END;

create or replace TRIGGER verificationDeleteItem

BEFORE DELETE ON item

FOR EACH ROW

BEGIN

IF (:old.qttotal != :old.qtrestante)then

RAISE\_APPLICATION\_ERROR(-20000,'Impossible a supprimer ,car encore des item en circulation');

END IF;

END;

# Création des procédures stockées

Dans les procédures nous utilisons des séquences.nextval qui s’occupe du auto incrémente et donc l’utilisateur ne peut mettre nimportequele chiffre.

Voici un exemple de création de séquence

create sequence item\_sequence;

create or replace procedure addclient(prenom VARCHAR, nom varchar,courriel varchar, mdp varchar,cartecredit INTEGER,

dateexpiration DATE,adresse varchar,datenaissance date) is

begin

insert into clientview values (personne\_sequence.nextval,addclient.prenom,addclient.nom,addclient.courriel,addclient.mdp,numclient\_sequence.nextval,

addclient.cartecredit,addclient.dateexpiration,addclient.adresse,addclient.datenaissance);

end;

create or replace procedure addemploye(prenom VARCHAR, nom varchar,identifiant INTEGER, mdp varchar,courriel

varchar,adresse varchar,datenaissance date) is

begin

insert into employeviewview values (personne\_sequence.nextval,numemploye\_sequence.nextval,addEmploye.prenom,addEmploye.nom,

addEmploye.identifiant,addEmploye.mdp,addEmploye.courriel,addEmploye.adresse,addEmploye.datenaissance);

end;

create or replace procedure deleteclient (personid INTEGER)

is

begin

delete from clientview where personid = deleteclient.personid;

end;

create or replace procedure deleteEmploye (personid INTEGER)

is

begin

delete from employeviewview where personid = deleteemploye.personid;

end;