IUT d'Orsay
Projet Tutoré
Groupe 3A2
Zixuan FENG & Hector de Framond

Rapport

I. Introduction

a) Résumé

Dans le cadre d'une association de sport pour des jeunes, nous avons mis en place un système informatique de payement des gouters pour les enfants qui souaitent un manger à la fin de leur séance. Ce système informatique remplace la gestion des gouters par les parents bénévoles de l'association qui devaient s'occuper des achats et des comptes à la main.

b) Acteurs

Les différents acteurs sont :

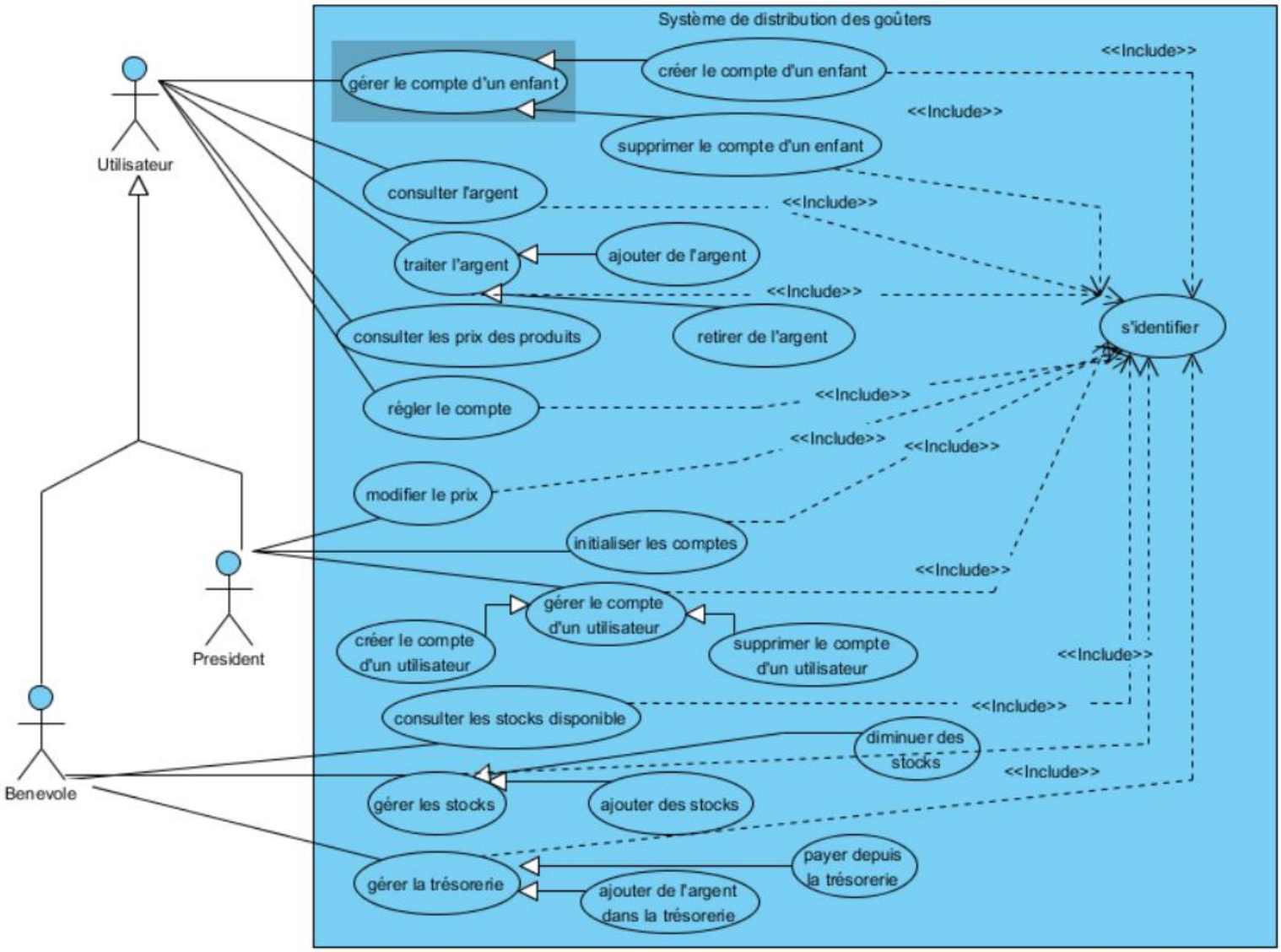
-Les utilisateurs qui fournissent les comptes des enfants.

-Les parents bénévoles qui s'occupent de la distribution des goûters le jour même et qui peuvent effectuer les achats de nouveaux goûters et créditer le compte des enfants.

-Le président de l'association qui gère les comptes des autres utilisateurs et modifier les prix des produits.

On considère que tous les parents sont des utilisateurs et que tous les utilisateurs sont des parents. Les parents sont des parents bénévoles dans le système lorsqu'ils distribuent les goûters le jour même.

c) Diagramme de cas d'utilisation

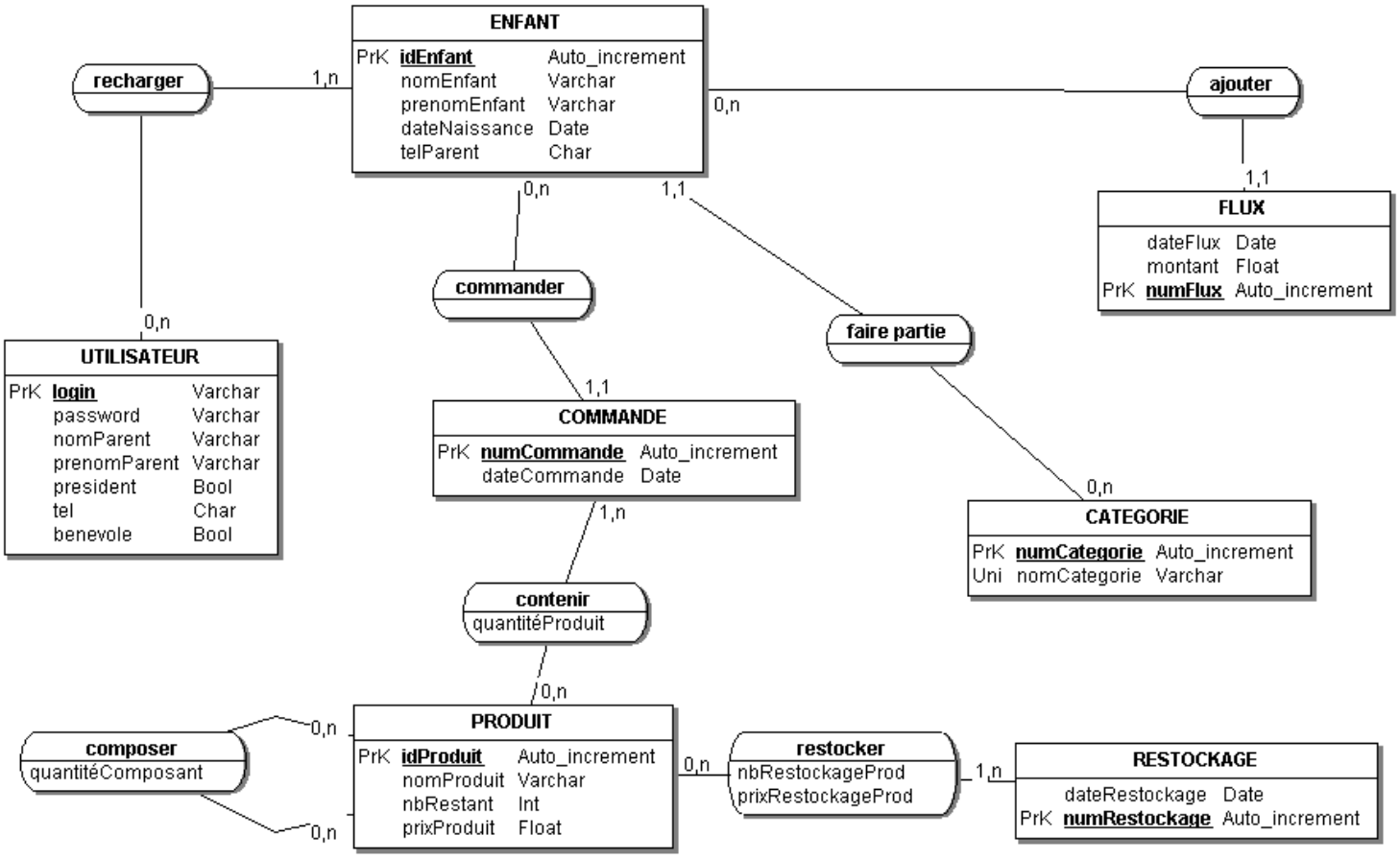


II. Analyse et conception du site Web

a) Dictionnaire des données

NOM RUBRIQUE	TYPE	DESCRIPTION
login	A	login des utilisateurs parent,c'est également leur adresse Email
password	A	
nomParent	A	
prenomParent	A	
president	A	détermine qui est le président de l'association
tel	A	
benevole	A	détermine qui des parents sont des parents bénévoles.
idEnfant	A	numéro identifiant des enfants
nomEnfant	A	
prenomEnfant	A	
dateNaissance	A	
telParent	A	
solde	C	= somme des montants des flux de ce compte (valeur initialisée=0)
numCategorie	A	numéro identifiant des catégories
nomCategorie	A	
numFlux	A	numéro identifiant des flux
dateFlux	A	
montant	A	>0 si on ajoute de l'argent ; <0 si on paye les goûters
numCommande	A	numéro identifiant des commandes
totalPrix	C	= somme totale de prixProduit*quantitéProduit d'une commande
dateCommande	A	
quantitéProduit	A	
idProduit	A	numéro identifiant des produits
nomProduit	A	
nbRestant	A	le nombre restant des produits en stock
prixProduit	A	le prix unitaire du produit
quantitéComposant	A	la quantité des produits composants
apport	C	= somme de totalPrix
depense	C	= somme de prixTotalRestock
soldeTresorerie	C	= apport-depense
numRestockage	A	numéro identifiant des restockages
dateRestockage	A	
prixTotalRestock	C	= somme de nbRestockageProd*prixRestockageProd d'une seule restockage
nbRestockageProd	A	le nombre de chaque produit achetés
prixRestockageProd	A	le prix unitaire du produit acheté

b)Schéma entité/association

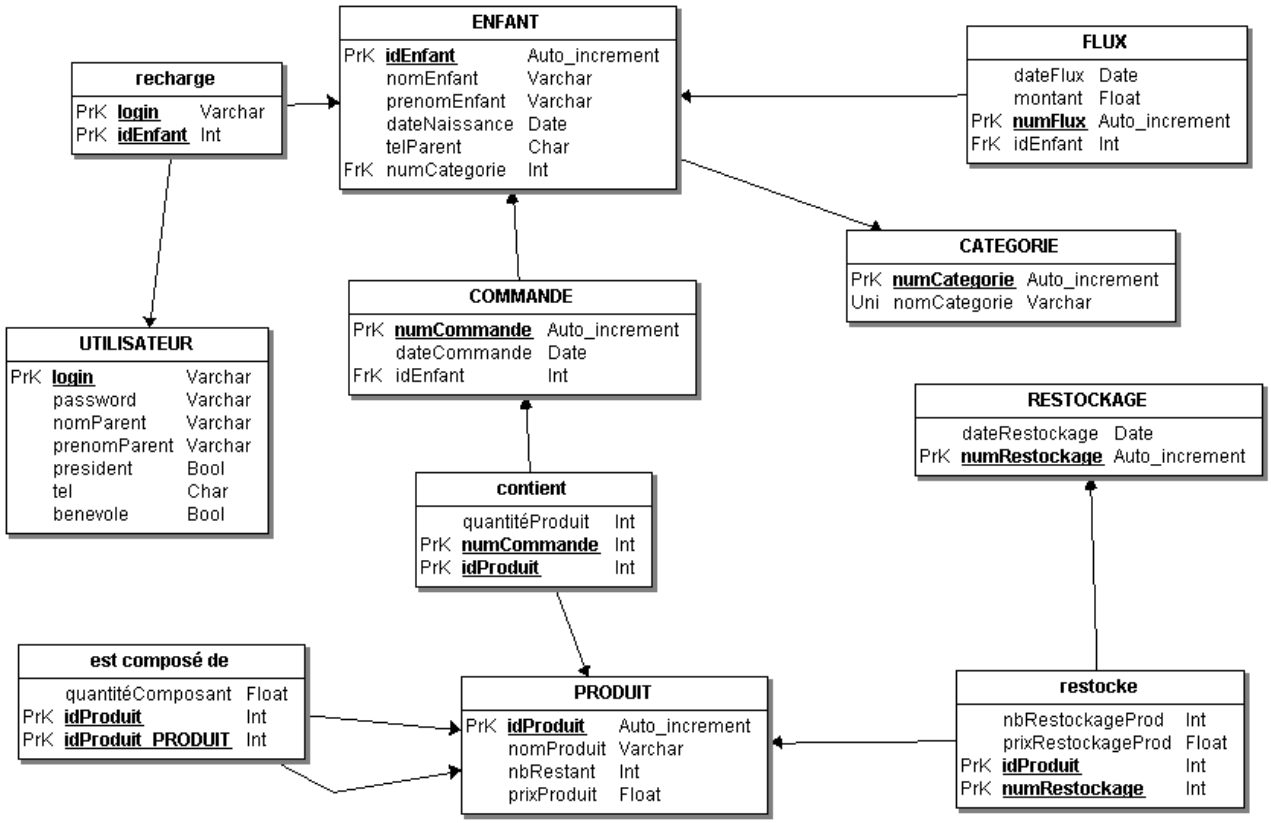


Nous avons mis en place deux bouléens dans la table utilisateur pour déterminer si le parent est un des parents bénévoles qui distribue les goûters le jour même ou s'il est le président.

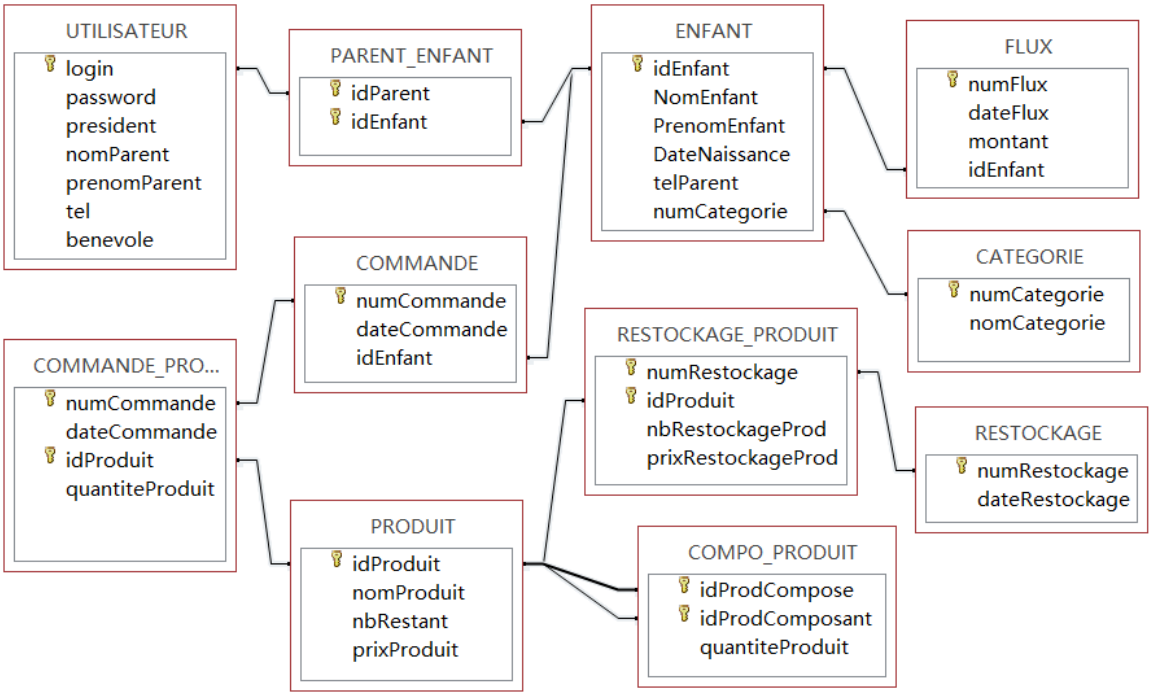
Tous les flux d'argent du compte enfant sont stockés grâce à la table Flux.

III. Implémentation

a) Schéma relationnel



(SR créé automatiquement par Jmerise, les noms des tableaux ne sont pas modifiables)



(SR créé avec acces, les fleches sont peu visibles mais le code provient de ce schéma)

b) Code SQL de la création de la base de données

```
CREATE TABLE UTILISATEUR(
    login       VARCHAR2 (50) NOT NULL ,
    password    VARCHAR2 (30) NOT NULL ,
    nomParent   VARCHAR2 (25) NOT NULL ,
    prenomParent VARCHAR2 (50) ,
    president   NUMBER (1) NOT NULL ,
--il n'existe pas le type boolean en SQL, donc on utilise NUMBER(1)
--1=TRUE et 0=FALSE
    tel         CHAR (10)  NOT NULL ,
    benevole    NUMBER (1) NOT NULL ,
    CONSTRAINT cleUtilisateur PRIMARY KEY (login) ,
    CONSTRAINT CHK_BOOLEAN_president CHECK (president IN (0,1)),
    CONSTRAINT CHK_BOOLEAN_benevole CHECK (benevole IN (0,1))
);

CREATE TABLE CATEGORIE(
    numCategorie  NUMBER(2) NOT NULL ,
    nomCategorie  VARCHAR2 (25)NOT NULL ,
    CONSTRAINT cleCategorie PRIMARY KEY (numCategorie) ,
    CONSTRAINT CATEGORIE_Uniq UNIQUE (nomCategorie)
);

CREATE TABLE ENFANT(
    idEnfant    NUMBER(5) NOT NULL ,
    nomEnfant   VARCHAR2 (25) NOT NULL ,
    prenomEnfant  VARCHAR2 (50) NOT NULL ,
    dateNaissance  DATE ,
    telParent   CHAR (10)  NOT NULL ,
    numCategorie  NUMBER(2)  NOT NULL ,
    CONSTRAINT cleEnfant PRIMARY KEY (idEnfant),
 FOREIGN KEY (numCategorie) REFERENCES CATEGORIE(numCategorie)
);

CREATE TABLE COMMANDE(
```

```
    numCommande   NUMBER(10) NOT NULL ,
    dateCommande  DATE  NOT NULL ,
    idEnfant     NUMBER(5)  NOT NULL ,
    CONSTRAINT cleCommande PRIMARY KEY (numCommande),
 FOREIGN KEY (idEnfant) REFERENCES ENFANT(idEnfant)
);

CREATE TABLE PRODUIT(
    idProduit    NUMBER(5) NOT NULL ,
    nomProduit   VARCHAR2 (25) NOT NULL ,
    nbRestant    NUMBER(5,2)  NOT NULL ,
    prixProduit  NUMBER(4,2)  NOT NULL ,
    CONSTRAINT cleProduit PRIMARY KEY (idProduit),
 CONSTRAINT clePrixPositif CHECK (prixProduit>=0),
 CONSTRAINT clePrixMax CHECK (prixProduit<100),
);
CREATE TABLE RESTOCKAGE(
    dateRestockage  DATE  NOT NULL ,
    numRestockage   NUMBER(10) NOT NULL ,
    CONSTRAINT cleRestockage PRIMARY KEY (numRestockage)
);

CREATE TABLE FLUX(
    dateFlux  DATE  NOT NULL ,
    montant   NUMBER(5,2) NOT NULL ,
    numFlux   NUMBER(10) NOT NULL ,
    idEnfant  NUMBER(5)  NOT NULL ,
    CONSTRAINT cleFlux PRIMARY KEY (numFlux),
 FOREIGN KEY (idEnfant) REFERENCES ENFANT(idEnfant) ON DELETE CASCADE
);

CREATE TABLE PARENT_ENFANT(
    idParent  VARCHAR2 (50) NOT NULL ,
    idEnfant  NUMBER(5)  NOT NULL ,
    CONSTRAINT clePE PRIMARY KEY (idParent,idEnfant),
 FOREIGN KEY (idParent) REFERENCES UTILISATEUR(login),
```

```
 FOREIGN KEY (idEnfant) REFERENCES ENFANT(idEnfant)
);

CREATE TABLE COMMANDE_PRODUIT(
    quantiteProduit  NUMBER(5,2)  NOT NULL ,
    numCommande     NUMBER(10)  NOT NULL ,
    idProduit       NUMBER(5)  NOT NULL ,
    CONSTRAINT cleCP PRIMARY KEY (numCommande,idProduit),
 FOREIGN KEY (numCommande) REFERENCES COMMANDE(numCommande),
 FOREIGN KEY (idProduit) REFERENCES PRODUIT(idProduit),
 CONSTRAINT cleNbPositif CHECK (quantiteProduit>=0)
);

CREATE TABLE RESTOCKAGE_PRODUIT(
    nbRestockageProd    NUMBER(5,2)  NOT NULL ,
    prixRestockageProd  NUMBER(5,2)  NOT NULL , --prix unitaire
    idProduit         NUMBER(5)  NOT NULL ,
    numRestockage      NUMBER(10)  NOT NULL ,
    CONSTRAINT cleRP PRIMARY KEY (idProduit,numRestockage),
 FOREIGN KEY (idProduit) REFERENCES PRODUIT(idProduit),
 FOREIGN KEY (numRestockage) REFERENCES RESTOCKAGE(numRestockage),
 CONSTRAINT clePrixRPositif CHECK (prixRestockageProd>=0),
 CONSTRAINT clePrixRMax CHECK (prixRestockageProd<100),
 CONSTRAINT cleNbRPositif CHECK (nbRestockageProd>=0)
);
CREATE TABLE COMPO_PRODUIT(
    quantiteComposant  NUMBER(5,2) NOT NULL ,
    idProdCompose    NUMBER(5)  NOT NULL ,
    idProdComposant  NUMBER(5)  NOT NULL ,
    CONSTRAINT cleCompoProd PRIMARY KEY (idProdCompose,idProdComposant),
 FOREIGN KEY (idProdCompose) REFERENCES PRODUIT(idProduit),
 FOREIGN KEY (idProdComposant) REFERENCES PRODUIT(idProduit),
 CONSTRAINT cleQuanPositif CHECK (quantiteComposant>=0)
);
```

c) Code SQL de l'insertion de données exemples

```
--ajoute 4 parents
INSERT INTO Utilisateur VALUES('ADF','123456','De Framond','A',1,'0600000000',0);
INSERT INTO Utilisateur VALUES('BDF','123456','De Framond','B',0,'0600000001',0);
INSERT INTO Utilisateur VALUES('CF','123456','Feng','C',0,'0600000003',0);
INSERT INTO Utilisateur VALUES('DF','123456','Feng','D',0,'0600000004',0);

--ajoute 2 categories
INSERT INTO Categorie VALUES(1,'<8ans');
INSERT INTO Categorie VALUES(2,'8-10ans');

--ajoute 2 flux sur le compte 1
INSERT INTO Flux VALUES(sysdate,50,1,1);
INSERT INTO Flux VALUES(sysdate,-2.2,2,1);

--ajoute 2 enfants
INSERT INTO Enfant VALUES(1,'De Framond','Hector',TO_DATE('29-11-2011','DD-MM-
YYYY'),'0600000000',1);
INSERT INTO Enfant VALUES(2,'Feng','Zixuan',TO_DATE('29-11-2008','DD-MM-
YYYY'),'0600000003',2);

--ajoute 5 produits
INSERT INTO Produit VALUES(1,'Donut chocolat',3,1);
INSERT INTO Produit VALUES(2,'pain au chocolat',5,1.2);
INSERT INTO Produit VALUES(3,'croissant',10,1.2);
INSERT INTO Produit VALUES(4,'Coca Zero',12,1);
INSERT INTO Produit VALUES(5,'produit speciale chocolat',10,1.2);

--ajoute 2 lignes dans COMPO_PRODUIT
INSERT INTO Compo_Produit VALUES(0.5,5,1);
INSERT INTO Compo_Produit VALUES(0.5,5,2);

--ajoute 1 commande
INSERT INTO Commande VALUES(1,sysdate,1);
```

```
--ajoute 2 produits dans cette commande
INSERT INTO Commande_Produit VALUES(1,1,5);
INSERT INTO Commande_Produit VALUES(1,1,4);

--ajoute 1 ligne dans restockage
INSERT INTO Restockage VALUES(TO_DATE('28-11-2017','DD-MM-YYYY'),1);

--ajoute 2 lignes dans restockage_produit
INSERT INTO Restockage_Produit VALUES(10,0.7,1,1);
INSERT INTO Restockage_Produit VALUES(5,1,2,1);
```

d) Code des requêtes

```
--creer le compte d'un enfant
--ici on prend un enfant du parent numero1 comme l'exemple
INSERT INTO Enfant VALUES((SELECT MAX(idEnfant) FROM
ENFANT)+1,'Nom','Prenom',sysdate,'0699999999',1);
INSERT INTO Flux VALUES(sysdate,0,(SELECT MAX(numFlux) FROM FLUX)+1,(SELECT idEnfant
FROM ENFANT WHERE nomEnfant='Nom' and prenomEnfant='Prenom'));

--supprimer les compte d'un enfant
DELETE FROM Enfant WHERE nomEnfant='Nom' and prenomEnfant='Prenom';

--consulter l'argent du compte de l'enfant 1
SELECT SUM(montant)
FROM FLUX
WHERE idEnfant=1;

--traiter l'argent
--ajouter de  l'argent
INSERT INTO Flux VALUES(sysdate,30,(SELECT MAX(numFlux) FROM Flux)+1,1);
--retirer de l'argent
INSERT INTO Flux VALUES(sysdate,-30,(SELECT MAX(numFlux) FROM Flux)+1,1);

--consulter les prix des produits
SELECT nomProduit,prixProduit
FROM Produit;
```

```
--consulter les stocks disponibles
SELECT nomProduit,nbRestant
FROM Produit;

--Gerer les stocks
--ajouter les stocks
UPDATE Produit
SET nbRestant=nbRestant+2
WHERE nomProduit='Donut chocolat';
--diminuer les stocks
UPDATE Produit
SET nbRestant=nbRestant-2
WHERE nomProduit='Donut chocolat';

--modifier le prix
UPDATE Produit
SET prixProduit=1.2
WHERE nomProduit='Donut chocolat';
```

--Les codes pour gérer les comptes utilisateurs sont les mêmes que les codes des fonctions de creation et suppressions avec des parametres differents.

e) Code des vue

```
--la vue pour voir le solde de tous les enfants
CREATE OR REPLACE VIEW soldeE(idEnfant,nomEnfant,prenomEnfant,solde)
AS SELECT idEnfant,nomEnfant,prenomEnfant,SUM(montant)
   FROM ENFANT NATURAL  JOIN FLUX
   GROUP BY idEnfant,nomEnfant,prenomEnfant;

--si on veux voir le solde d'un enfant , ici on prend l'enfant 'Hector De Framond'
CREATE OR REPLACE VIEW soldeUnE(idEnfant,nomEnfant,prenomEnfant,solde)
AS SELECT idEnfant,nomEnfant,prenomEnfant,SUM(montant)
   FROM ENFANT NATURAL JOIN FLUX
   WHERE nomEnfant='De Framond' and prenomEnfant='Hector'
   GROUP BY idEnfant,nomEnfant,prenomEnfant;
```

f) Code des triggers

```
--1 trigger permet de gerer des alertes quand on n'a pas assez de produits
CREATE OR REPLACE TRIGGER trigPasAssez
BEFORE Update ON Produit FOR EACH ROW WHEN(new.nbRestant<0)
BEGIN
  RAISE_APPLICATION_ERROR(-20001,'Pas assez de produit!');
END;
```

IV. Documentation technique

Les utilisateurs doivent s'inscrire en fournissant un login, un mot de passe, leur nom, prenom et leur telephone. Il doivent ensuite remplir un formulaire pour leur enfant, comprenant le nom de l'enfant, le prenom de l'enfant, sa date de naissance et le numéro de telephone des parents.

Pour que la base fonctionne il faut que le président de l'association le soit indiqué dans la table utilisateur. Les parents bénévoles qui s'occupent de la distribution des gouters le jour même doivent l'etre aussi.

Les différents menus, produits doivent etre initialisés avec un prix et la quantité de produit dont ils sont composés

V. Conclusion

La base de donnée est opérationnelle pour la suite du projet et la mise en place du site web. Nous avons appris à nous servir du logiciel JMerise sans réelles difficultés et revu les techniques apprises au s1 et au s2. Le code des triggers était difficile à compléter, l'actualisation des données produisait des erreurs dans le code. Il était également difficile de trouver toutes les differentes interactions entre les utilisateurs et le gestionnaire. On a longement hesité entre plusieurs possibilités de MCDs, avec plus ou moins d'acteurs.