RAPPORT DE PROJET

HAI502I GROUPE B

Numéro de carte étudiant : 22010095
Nom : FAURA BEHAGUE
Prenom : MATTEO

Numéro de carte étudiant : 22020352
Nom : DJUNAEDI
Prenom : Rendra

Groupe du projet : GROUPE 3

Sujet du projet : GESTION DE DON DE SANG

Professeur de TD : Mr.Guillaume Perution

Base de données : Oracle Database

Le sujet que nous avons choisi pour ce projet est la gestion de dons de sang. Nous nous sommes intéressés à ce sujet car on s'est souvent demandé comment était gérée la base de données pour la gestion des dons de sang en voyant les camions de don de sang devant la fac. On a trouvé intéressant d'essayer de répondre à ce problème de gestion de don de sang, de traitement et de stockage en hôpital des dons.

(Les fichiers test_Groupe3.sql et creation_Groupe3.sql contiennent des commentaires, des affichages et des mises en scénario qui viennent appuyer les informations contenues dans ce rapport. L'exécution de ces fichiers est donc recommandée) ⚠

➤ SPÉCIFICATIONS :

Notre base de données contiendra des **collectes** qui auront une date de début, une date de fin, une localisation et les collectes sont identifiées par une id de collecte.

Les collectes récoltent des dons de sang qui seront par la suite envoyés dans des **hôpitaux**. Chaque don est effectué par un **donneur** caractérisé par son numéro de donneur, ses coordonnées et son numéro de contact. **-** *Le groupe sanguin de ce dernier n'est connu qu'après le traitement/analyse de son don* **-**.

Un **don** est caractérisé par son id_don, la quantité de sang donnée, la date à laquelle le don a été effectuée pour pouvoir en déduire sa date de péremption (validité du sang) .

Les dons sont envoyés aux hôpitaux qui sont caractérisés par le code hôpital, le nom et l'adresse.

Le don sera analysé par le **personnel** de l'hôpital et validé ou pas **-** *auquel cas un message sera envoyé au donneur***-**.

Chaque hôpital reçoit des **patients** pouvant bénéficier de dons de sang et chaque patient est caractérisé par son id patient, son groupe sanguin et ses coordonnées. **-** *La transfusion n'est faite que si le don est compatible avec le groupe sanguin du patient (après analyse et validation du don par le personnel)* **-**

➤ SCHÉMA PHYSIQUE :

(Le schéma physique pourra être consulté dans notre fichier creation_Groupe3.sql de nôtre projet).

➤ DICTIONNAIRE DES DONNÉES :

PERSONNE				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
NUM_PERSO	NUMERIC	(10,0)	CLE PRIMAIRE	Numero personne
NOM	VARCHAR	32	NOT NULL	Nom personne
PRENOM	VARCHAR	32	//	Prenom personne
CONTACT	NUMERIC	(10,0)	//	Numero de telephone
DATE_NAISSANCE	DATE	'DD-MM-YYYY'	//	Date de naissance

DONNEUR				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
NUM_DONNEUR	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero Donneur
EMAIL	VARCHAR	64	NOT NULL	Adresse mail

PATIENT				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
NUM_PATIENT	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero patient
GROUPE_SANG	VARCHAR	3	(O-,O+,A-,A+,B-,B+,AB-,AB+)	Adresse mail

PERSONNEL				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
NUM_PERSONNEL	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero Personnel
FONCTION	VARCHAR	32	//	Adresse mail
DATE_ANCIENNETE	DATE	'DD-MM-YYYY'	//	Date de recrutement

HOPITAL				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
CODE_HOPITAL	NUMERIC	(10,0)	CLE PRIMAIRE	Code de l'hôpital
NOM	VARCHAR	128	//	Nom de l'hôpital
ADRESSE	VARCHAR	128	//	Adresse de l'hôpital

COLLECTE				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
ID_COLLECTE	NUMERIC	(10,0)	CLE PRIMAIRE	ID de la collecte
DATE_DEBUT	DATE	'DD-MM-YYYY'	//	Date de début de la collecte
DATE_FIN	DATE	'DD-MM-YYYY'	//	Date de fin de la collecte
LOCALISATION	VARCHAR	128	//	Adresse du lieu collecte
URGENCE	VARCHAR	10	(URGENT,NORMAL)	Motif pour dons de -18 ans

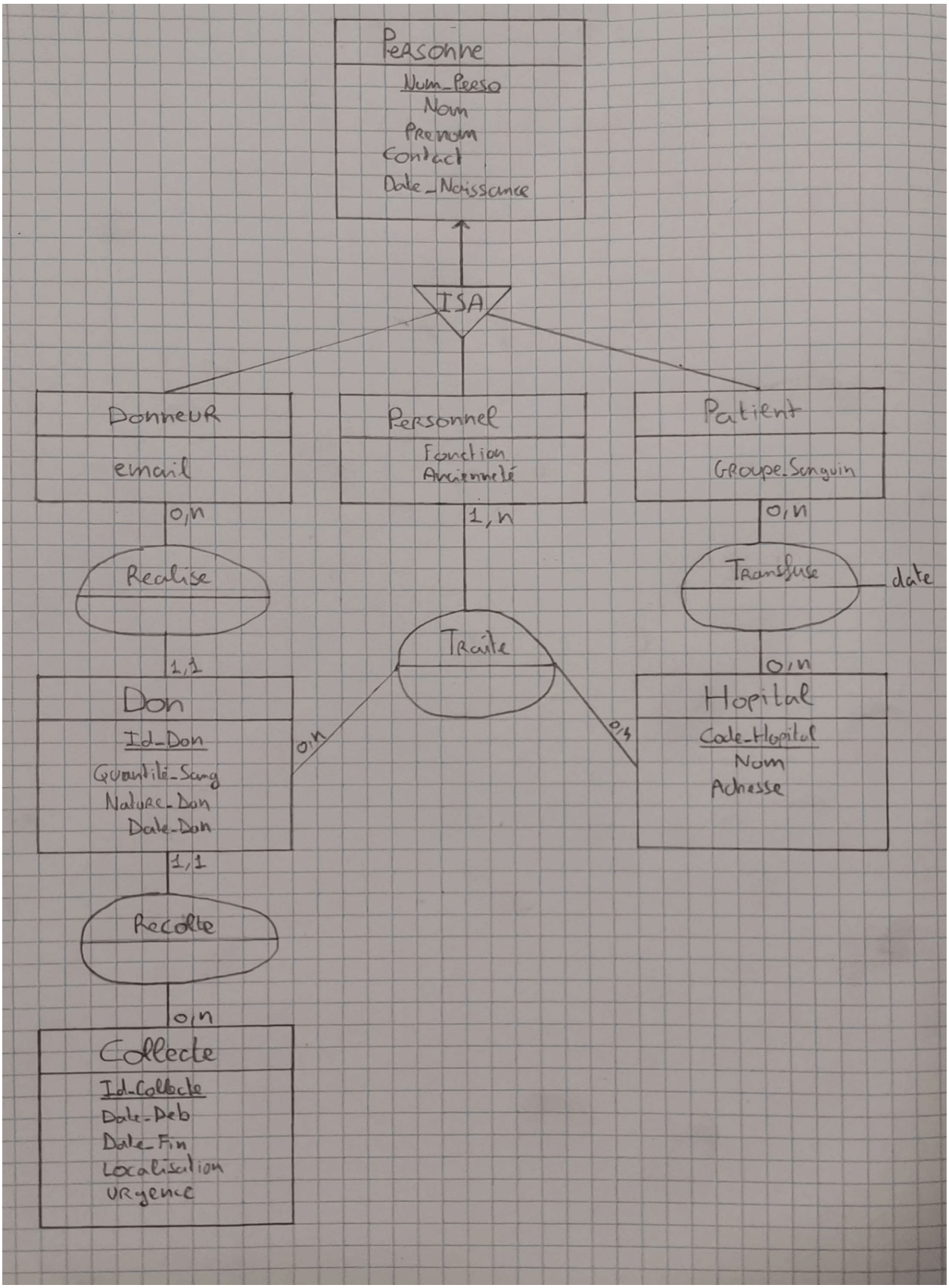
DON				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
ID_DON	NUMERIC	(10,0)	CLE PRIMAIRE	ID du don
QUANTITE	INT	//	ENTRE 400 et 500	Quantité de sang donnée
NATURE_DON	VARCHAR	10	(URGENT,NORMAL)	Nature du don
DATE_DON	DATE	'DD-MM-YYYY'	//	Date du don
NUM_D	NUMERIC	(10,0)	CLE ETRANGERE	Numero du Donneur
ID_C	NUMERIC	(10,0)	CLE ETRANGERE	ID de la collecte

TRAITE				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
NUM_PER	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero du Personnel
ID_D	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero du Donneur
CODE_H	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	ID de l'hôpital
DATE_TRAITEMENT	DATE	'DD-MM-YYYY'	//	Date du traitement
VALIDITE	VARCHAR	15	VALIDE ou NON_VALIDE	Validité du sang
TYPE_SANG	VARCHAR	3	(O-,O+,A-,A+,B-,B+,AB-,AB+)	Type du sang du don

TRANSFUSE				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
ID_D	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Numero du Donneur
NUM_P	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	ID du patient
CODE_H	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	Code de l'hopital
D_TRANSFUSION	DATE	'DD-MM-YYYY'	//	Date de la transfusion

MSG_REFUS (pour simuler nos triggers)				
FIELD	**TYPE**	**TAILLE**	**CONTRAINTE**	**DESCRIPTION**
ID_MSG	NUMERIC	(10,0)	CLE PRIMAIRE	ID du message
ID_D	NUMERIC	(10,0)	CLE PRIMAIRE CLE ETRANGERE	ID du Don NON_VALIDE

➤ **SCHÉMA ENTITÉ ASSOCIATION :**



➤ SCHÉMA RELATIONNEL :

PERSONNE (**NUM_PERSO**, NOM, PRENOM, CONTACT, DATE_NAISSANCE)
DONNEUR (***#NUM_DONNEUR***, EMAIL)
PATIENT (***#NUM_PATIENT***, GROUPE_SANG)
PERSONNEL (***#NUM_PERSONNEL***, FONCTION, DATE_ANCIENNETE)
HOPITAL (**CODE_HOPITAL**, NOM, ADRESSE)
COLLECTE (**ID_COLLECTE**, DATE_DEBUT, DATE_FIN, LOCALISATION, URGENCE)
DON (**ID_DON**, QUANTITE, NATURE_DON, DATE_DON, *#NUM_D*, *#ID_C*)
MSG_REFUS (**ID_MSG**, *#ID_D*)
TRAITE (***#NUM_PER***, ***#ID_D***, ***#CODE_H***, DATE_TRAITEMENT, VALIDITE, TYPE_SANG)
TRANSFUSE (***#ID_D***, ***#NUM_P***, ***#CODE_H***, D_TRANSFUSION)

➤ FONCTIONS, TRIGGERS ET PROCEDURES :

(plus de détails durant l'exécution de test_Groupe3.sql avec un scénario et affichage bien détaillé)

Les **fonctions** qui interviennent dans notre base de donnée sont : **Get_index()** et **Get_index_MSG()** qui tout simplement comme leur nom l'indique, vont aller chercher le dernier ID de la table PERSONNE ou MSG_REFUS à l'aide d'une requête qui retourne le max(ID) de la Table. (SELECT COUNT(*) INTO MSG_INDEX FROM MSG_REFUS;) pour voir si notre table est vide et qu'aucun ID n'est utilisé alors retourner l'ID = 1. Ou Sinon (SELECT MAX(ID_MSG) INTO MSG_INDEX FROM MSG_REFUS/PERSONNE;) si notre table n'est pas vide puis par la suite incrémenter cet ID et le retourner. Cet ID va être utilisé pour pouvoir ensuite insérer des nouveaux tuples dans ces Tables (fonctions utilisées dans nos TRIGGERS) sans avoir de problèmes au niveau des ID.

En ce qui concerne les **TRIGGERS**, **Insert_donneur, Insert_patient, Insert_personnel et Check_valid_don** sont les triggers utilisés par notre base de données. Insert_donneur(), Insert_patient() et Insert_personnel() utilisent les **VIEW ALL_DONNEURS, ALL_PATIENTS et ALL_PERSONNELS** qui eux vont retourner les tables avec toutes les informations en plus de la table PERSONNE (NOM, PRENOM, CONTACT, DATE_NAISSANCE) grâce à une requête utilisant un JOIN sur les ID des tables.

Ces 3 **VIEWS** sont créées pour permettre le remplissage des tables Personne et de ces tables filles en même temps. Lors de l'insertion d'une ligne dans l'une de ces tables, le trigger **Insert_[Type_de_Personne]** est appelé pour compléter à la fois la table Personne est la table fille Donneur,

Patient ou Personnel. Ainsi on n'insère jamais directement dans Les tables mais plutôt dans ces 3 views, puis les triggers ce charge du reste.

Du côté des **PROCÉDURES**, **Make_non_valide** est la procédure qu'on a mis en place pour être utilisée dans notre base de données. Make_non_valide comme le nom l'indique va aller update notre table TRAITE et plus précisément la VALIDITE des dons traités en mettant cette donnée à NON_VALIDE lorsque le nombre de mois entre la date à laquelle on a effectué le traitement et la date à laquelle la PROCÉDURE est lancée est supérieur à 2 mois.

```
CREATE OR REPLACE PROCEDURE Make_non_valide AS
BEGIN
    UPDATE TRAITE SET VALIDITE = 'NON_VALIDE'
        WHERE (SELECT MONTHS_BETWEEN((SELECT SYSDATE FROM DUAL),DATE_TRAITEMENT) FROM DUAL) > 2
        OR TRAITE.ID_D IN (SELECT TRANSFUSE.ID_D FROM TRANSFUSE);
END;
/
```

➤ REQUÊTES SUR NOTRE TABLE :

Requête utilisant une division : Les collectes où tous les Donneurs sont allés.
C'est à dire, les collectes où il n'y a pas eu pas tous les donneurs.

```
SELECT DISTINCT C.ID_COLLECTE
FROM COLLECTE C
WHERE NOT EXISTS (SELECT * FROM DONNEUR D
                  WHERE NOT EXISTS (SELECT * FROM DON D1
                                    WHERE C.ID_COLLECTE = D1.ID_C
                                      AND D1.NUM_D = D.NUM_DONNEUR));
```

Requête utilisant une division un peu plus compliquée : Les collectes où tous les Donneurs nommés "Sanchez" sont allés après le 1er janvier 2019.
C'est à dire, les collectes après 2018 où il n'y a pas eu pas tous les donneurs nommé "Sanchez".

```
SELECT DISTINCT C.ID_COLLECTE
FROM COLLECTE C
WHERE DATE_DEBUT > TO_DATE('01-01-2019') AND NOT EXISTS (SELECT * FROM DONNEUR D JOIN PERSONNE P ON P.NUM_PERSO = D.NUM_DONNEUR
                  WHERE P.NOM = 'SANCHEZ' AND NOT EXISTS (SELECT * FROM DON D1
                                    WHERE C.ID_COLLECTE = D1.ID_C
                                      AND D1.NUM_D = D.NUM_DONNEUR));
```

Requête contenant 2 sous requêtes : Id collecte, id donneur, nom et prénom des donneurs où il y a eu le plus de dons.

Ici, une requête pour donner le nombre de dons par collecte. On utilise ce résultat pour sélectionner la collecte où il a le plus de dons. Et ensuite, on sélectionne les personnes qui ont participé à cette collecte.

```
SELECT ID_COLLECTE, NUM_D, NOM, PRENOM
FROM COLLECTE
JOIN DON ON ID_COLLECTE = ID_C
JOIN PERSONNE ON NUM_PERSO = ID_DON
WHERE ID_COLLECTE IN ( SELECT ID_C
        FROM DON
        GROUP BY ID_C
        HAVING COUNT(*) > = ALL (SELECT COUNT(*)
                                 FROM DON
                                 GROUP BY ID_C) );
```

Requête avec un groupe by : Nombre de Don par collecte.
Dans cette requête, on fait l'union entre les collectes à plus d'un don est celle qui ne sont pas dans la liste des requêtes avec au moins un don. Le left join mettait qu'il y avait un Don dans les requêtes où il n'y avait pas donc on a procédé comme suivant.

```
SELECT ID_COLLECTE, COUNT(*) AS NB_DONS
FROM COLLECTE
JOIN DON ON ID_COLLECTE = ID_C
GROUP BY ID_COLLECTE
UNION
SELECT ID_COLLECTE, 0 AS NB_DONS
FROM COLLECTE WHERE ID_COLLECTE NOT IN ( SELECT ID_COLLECTE
                                         FROM COLLECTE
                                         JOIN DON ON ID_COLLECTE = ID_C
                                         GROUP BY ID_COLLECTE);
```

Requête avec une sous requête correlative : Pour chaque collecte, le nom du personnel ayant traité le plus de don. (Il peut y en avoir plusieurs)
On a besoin d'une corrélation car pour chaque membre du personnel on doit vérifier si son nombre de traitements pour chaque collecte à laquelle il a participé est le plus grand. Donc quand on prend un membre du personnel pour une collecte, on doit vérifier que son nombre de traitements est plus grand que tous les nombres de traitement des autres membres du personnel.

```sql
SELECT C1.ID_COLLECTE, P1.NOM, P1.PRENOM, COUNT(*) AS NB_TRAITEMENT
FROM PERSONNE P1
JOIN TRAITE T1 ON P1.NUM_PERSO = T1.NUM_PER
JOIN DON D1 ON T1.ID_D = D1.ID_DON
JOIN COLLECTE C1 ON D1.ID_C = C1.ID_COLLECTE
GROUP BY C1.ID_COLLECTE, P1.NOM, P1.PRENOM
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM COLLECTE C2
                        JOIN DON D2 ON D2.ID_C = C1.ID_COLLECTE
                        JOIN TRAITE T2 ON D2.ID_DON = T2.ID_D
                        WHERE C1.ID_COLLECTE = C2.ID_COLLECTE
                        GROUP BY T2.NUM_PER);
```

➤ PROBLÈMES RENCONTRÉS:

L'un des gros problème que nous avons rencontré est l'impossibilité d'utiliser la bibliothèque DBMS_OUTPUT. En effet, même en utilisant la commande : **EXECUTE DBMS_OUTPUT.ENABLE;** non avons été dans l'incapacité de créer des procédures intéressantes.

Nous avons aussi d'autres procédures mises en commentaires dans notre projet qui ne fonctionnent pas, nous aimerons bien une correction ou une indication pour pouvoir comprendre d'où le problème de "Avertissement : Procédure créée avec erreurs de compilation." provient.
Nous avons essayé à plusieurs reprises de modifier ces procédures mais rien ne semble les rendre fonctionnelles.