Aline BOULANGER Louise DELPIERRE Eline MOTA
Youlan COLLARD Aurélie GENOT Yannis VAN ACHTER

Conception de la base de données Groupe 7

Présentation du domaine d'application de AdopteTonMort (ATM)

AdopteTonMort est une application proposant l'achat et la transplantation d'organes et de sang à des personnes majeures.

Les organes sont définis par un identifiant propre. Ils sont proposés dans un certain état (allant de très mauvais à très bon). Chaque organe possède deux dates d'expiration: la première désigne la date maximale à laquelle il peut être transplanté, la seconde la date maximale à laquelle il peut être commercialisé. De plus, chaque organe est d'un type (le CEO définit la liste des types d'organes mis à la vente) et est à un prix unique.

Chaque poche de sang est elle aussi définie par un identifiant propre. Elle est d'un type (A, B, O ou AB) et d'un signe (+ ou -). Elle possède une date unique d'expiration.

Que ce soit pour les organes ou pour le sang, on retrouve un donateur dont les informations sont anonymes. Cela permet notamment de pouvoir relier plusieurs organes qui viendraient d'un même donateur. Une tranche d'âge et le genre du donateur sont demandés.

Le premier service proposé est l'achat d'organes ou de sang. Pour ça, un client doit faire une commande. Chaque commande est liée à une livraison contenant un destinataire, une date de départ et une date d'arrivée. Chaque commande contient également un ou plusieurs détails avec le contenu de l'achat (un organe ou une poche de sang). A chaque livraison est également lié un type de livraison. Celui-ci détaille le prix de la livraison et la livraison choisie.

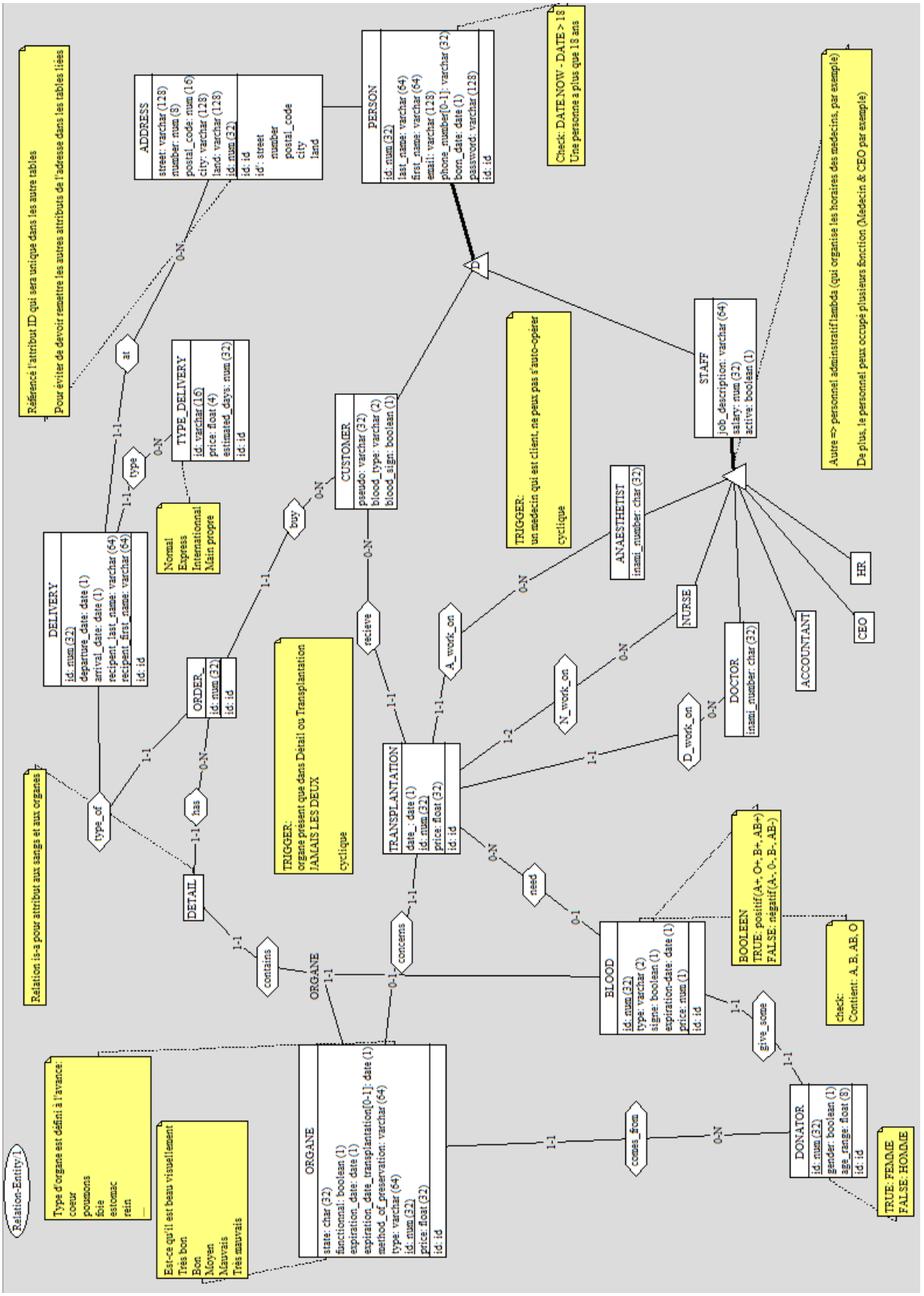
Le deuxième service permet d'avoir une transplantation de sang ou d'organes. Celle-ci doit avoir une date et un prix. Elle est liée au client demandeur. Pour effectuer une transplantation, un anesthésiste et un docteur sont nécessaires. Il y a également au choix un ou deux infirmiers.

Pour pouvoir passer commande, il faut donc être client de l'entreprise. Pour se faire, un client doit s'enregistrer avec un pseudonyme, son nom et son prénom. Son groupe sanguin est demandé. Il doit aussi fournir une adresse mail et une date de naissance. Finalement, il peut s'il le désire fournir un numéro de téléphone.

Le deuxième type de personnes de l'entreprise sont les employés. Ceux-ci doivent également entrer nom, prénom, adresse mail, numéro de téléphone et date de naissance. En plus, chaque employé est accompagné d'une description de son travail, de son salaire et de la notion d'actif/ inactif afin de savoir si l'employé travaille toujours actuellement au sein de l'entreprise (il est donc inactif s'il l'a quitté). L'employé peut être dans une catégorie spécifique s'il est anesthésiste, infirmier, docteur, comptable, membre des ressources humaines ou CEO. S'il est anesthésiste ou docteur, son numéro INAMI est demandé.

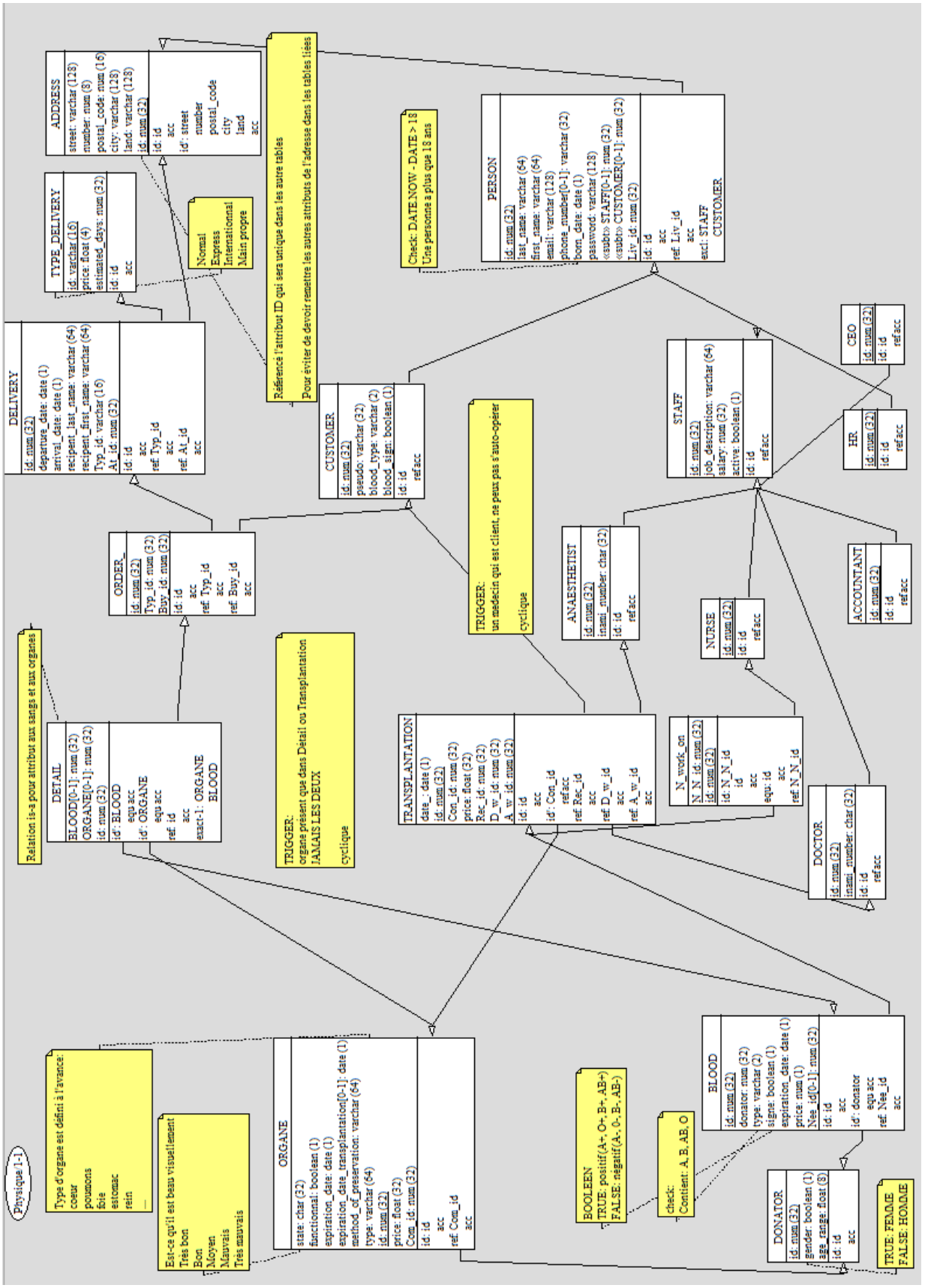
Aline BOULANGER
Youlan COLLARD
Louise DELPIERRE
Aurélie GENOT
Eline MOTA
Yannis VAN ACHTER

Le schéma conceptuel équivalent



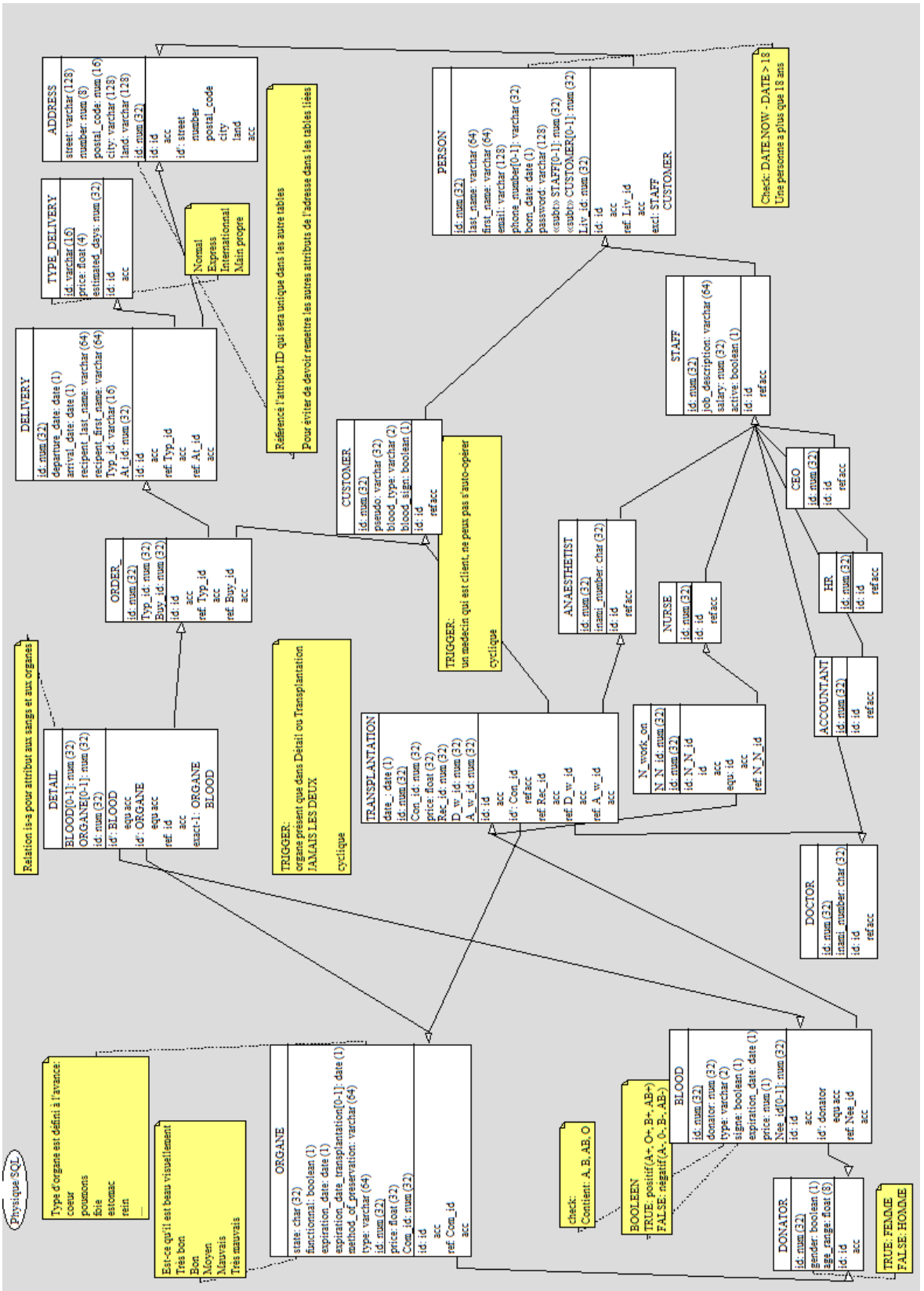
Aline BOULANGER
Youlan COLLARD
Louise DELPIERRE
Aurélie GENOT
Eline MOTA
Yannis VAN ACHTER

Le schéma logique équivalent



Aline BOULANGER
Youlan COLLARD
Louise DELPIERRE
Aurélie GENOT
Eline MOTA
Yannis VAN ACHTER

Le schéma physique équivalent (représentation selon le modèle relationnel)



Code DDL:

```
-- ********************************************
-- * Standard SQL generation
-- *------------------------------------------
-- * DB-MAIN version: 11.0.2
-- * Generator date: Sep 14 2021
-- * Generation date: Fri Apr  7 14:07:35 2023
-- * LUN file: .\INFOB212-BD2-project\schema\conceptual-schema.lun
-- * Schema: physical v4/SQL1
-- ********************************************


-- Database Section
-- ________________

-- create database db; -- As we user Docker we does not need this line

-- DBSpace Section
-- ________________


-- Tables Section
-- _____________

create table IF NOT EXISTS ADDRESS (
    street varchar(128) not null,
    number numeric(8) not null,
    postal_code numeric(16) not null,
    city varchar(128) not null,
    land varchar(128) not null,
    id INT unsigned not null AUTO_INCREMENT,
    constraint ID_ADDRESS_ID primary key (id),
    constraint SID_ADDRESS_ID unique (street, number, postal_code, city, land));

create table IF NOT EXISTS PERSON (
    id INT unsigned not null AUTO_INCREMENT,
    last_name varchar(64),
    first_name varchar(64),
    email varchar(128) not null,
    phone_number varchar(32),
    born_date date not null,
    password varchar(128) not null,
    Liv_id INT unsigned not null,
```

```
        constraint ID_PERSON_ID primary key (id),
        constraint FK_PersonAddress foreign key (Liv_id) references ADDRESS(id)
        );

    create table IF NOT EXISTS CUSTOMER (
        id INT unsigned not null,
        pseudo varchar(32) not null,
        blood_type varchar(2) not null,
        blood_sign char not null,
        constraint ID_CUSTO_PERSO_ID primary key (id),
        constraint FK_CustomerPerson foreign key (id) references PERSON(id)
        );

    create table IF NOT EXISTS TYPE_DELIVERY (
        id varchar(16) not null,
        price float(4) not null check(price > 0),
        estimated_days INT not null,
        constraint ID_TYPE_DELIVERY_ID primary key (id));

    create table IF NOT EXISTS DELIVERY (
        id INT unsigned not null AUTO_INCREMENT,
        departure_date date not null,
        arrival_date date not null,
        recipent_last_name varchar(64) not null,
        recipent_first_name varchar(64) not null,
        Typ_id varchar(16) not null,
        At_id INT unsigned not null,
        constraint ID_DELIVERY_ID primary key (id),
                constraint FK_DeliveryTypeDelivery foreign key (Typ_id) references
TYPE_DELIVERY(id),
        constraint FK_DeliveryAddress foreign key (At_id) references ADDRESS(id));

    create table IF NOT EXISTS ORDER_ (
        id INT unsigned not null AUTO_INCREMENT,
        Typ_id INT unsigned not null,
        Buy_id INT unsigned not null,
        constraint ID_ORDER_ID primary key (id),
        constraint FK_OrderDelivery foreign key (Typ_id) references DELIVERY(id),
        constraint FK_OrderCustomer foreign key (Buy_id) references CUSTOMER(id));

    create table IF NOT EXISTS STAFF (
        id INT unsigned not null,
        salary numeric(32) not null,
        job_description varchar(64),
        active boolean not null,
        constraint ID_STAFF_PERSO_ID primary key (id),
        constraint FK_StaffPerson foreign key (id) references PERSON(id));
```

```
create table IF NOT EXISTS CEO (
    id INT unsigned not null,
    constraint ID_CEO_STAFF_ID primary key (id),
    constraint FK_CEOStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS DOCTOR (
    id INT unsigned not null,
    inami_number varchar(32) not null,
    constraint ID_DOCTO_STAFF_ID primary key (id),
    constraint FK_DoctorStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS NURSE (
    id INT unsigned not null,
    constraint ID_NURSE_STAFF_ID primary key (id),
    constraint FK_NurseStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS ACCOUNTANT (
    id INT unsigned not null,
    constraint ID_ACCOU_STAFF_ID primary key (id),
    constraint FK_AccountantStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS ANAESTHESIST (
    id INT unsigned not null,
    inami_number varchar(32) not null,
    constraint ID_ANAES_STAFF_ID primary key (id),
    constraint FK_AnaesthesistStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS HR (
    id INT unsigned not null,
    constraint ID_HR_STAFF_ID primary key (id),
    constraint FK_HRStaff foreign key (id) references STAFF(id));

create table IF NOT EXISTS DONATOR (
    id INT unsigned not null AUTO_INCREMENT,
    gender char not null,
    age_range float(8) not null,
    constraint ID_DONATOR_ID primary key (id));

create table IF NOT EXISTS BLOOD (
    id INT unsigned not null AUTO_INCREMENT,
    type varchar(2) not null check(type = "A" or type = "B" or type = "O" or type = "AB"),
    signe boolean not null,
    expiration_date date not null,
    donator INT unsigned not null,
    price float(32) not null check(price > 0),
    Nee_id INT unsigned,
```

```
        constraint ID_BLOOD_ID primary key (id),
        constraint FK_BloodDonator foreign key (donator) references DONATOR(id)
        -- Foreign key Nee_id -> Transplantaton in alter tables
    );

    create table IF NOT EXISTS ORGANE (
        state char(32) not null,
        functionnal char not null,
        expiration_date date not null,
        expiration_date_transplantation date,
        method_of_preservation varchar(64) not null,
        type varchar(64) not null,
        id INT unsigned not null AUTO_INCREMENT,
        price float(32) not null check(price > 0),
        Com_id INT unsigned not null,
        constraint ID_ORGANE_ID primary key (id),
        constraint FK_OrganeDonator foreign key (Com_id) references DONATOR(id));

    create table IF NOT EXISTS TRANSPLANTATION (
        date_ date not null,
        id INT unsigned not null AUTO_INCREMENT,
        Con_id INT unsigned not null,
        price float(32) not null check(price > 0),
        Rec_id INT unsigned not null,
        D_w_id INT unsigned not null,
        A_w_id INT unsigned not null,
        constraint ID_TRANSPLANTATION_ID primary key (id),
        constraint SID_TRANS_ORGAN_ID unique (Con_id),
            constraint FK_TransplantationCustomer foreign key (Rec_id) references
CUSTOMER(id),
            constraint FK_TransplantationOrgane foreign key (Con_id) references
ORGANE(id),
        constraint FK_TransplantationDoctor foreign key (D_w_id) references DOCTOR(id),
            constraint FK_TransplantationAnaesthesist foreign key (A_w_id) references
ANAESTHESIST(id));

    create table IF NOT EXISTS DETAIL (
        BLOOD INT unsigned,
        ORGANE INT unsigned,
        id INT unsigned not null,
        constraint SID_DETAIL_ID unique (BLOOD, ORGANE, id),
        foreign key (BLOOD) references BLOOD(id),
        foreign key (ORGANE) references ORGANE(id),
        foreign key (id) references ORDER_(id),
        constraint EXTONE_DETAIL check(
            (ORGANE is not null and BLOOD is null)
            or (ORGANE is null and BLOOD is not null))
```

```
    );

create table IF NOT EXISTS N_work_on (
    N_N_id INT unsigned not null,
    id INT unsigned not null,
    constraint ID_N_work_on_ID primary key (N_N_id, id),
        constraint FK_NWorkOnTransplantation foreign key (id) references
TRANSPLANTATION(id),
    constraint FK_NWorkOnNurse foreign key (N_N_id) references NURSE(id));


-- Constraints Section - Checks
-- _____________________________

alter table BLOOD add
        constraint FK_BloodTransplantation foreign key (Nee_id) references
TRANSPLANTATION(id);

-- Index Section
-- _____________

create unique index ID_ADDRESS_IND
    on ADDRESS (id);

create unique index SID_ADDRESS_IND
    on ADDRESS (street, number, postal_code, city, land);

create unique index ID_ANAES_STAFF_IND
    on ANAESTHESIST (id);

create unique index ID_CUSTO_PERSO_IND
    on CUSTOMER (id);

create unique index ID_ORDER_IND
    on ORDER_ (id);

create index REF_ORDER_DELIV_IND
    on ORDER_ (Typ_id);

create index REF_ORDER_CUSTO_IND
    on ORDER_ (Buy_id);

create unique index ID_ACCOU_STAFF_IND
    on ACCOUNTANT (id);

create index EQU_DETAI_ORGAN_IND
    on DETAIL (ORGANE);
```

```
create index REF_DETAI_ORDER_IND
    on DETAIL (id);

create unique index SID_DETAIL_IND
    on DETAIL (BLOOD, ORGANE, id);

create unique index ID_DONATOR_IND
    on DONATOR (id);

create unique index ID_NURSE_STAFF_IND
    on NURSE (id);

create unique index ID_DELIVERY_IND
    on DELIVERY (id);

create index REF_DELIV_TYPE__IND
    on DELIVERY (Typ_id);

create index REF_DELIV_ADDRE_IND
    on DELIVERY (At_id);

create unique index ID_DOCTO_STAFF_IND
    on DOCTOR (id);

create unique index ID_ORGANE_IND
    on ORGANE (id);

create index REF_ORGAN_DONAT_IND
    on ORGANE (Com_id);

create unique index ID_CEO_STAFF_IND
    on CEO (id);

create unique index ID_PERSON_IND
    on PERSON (id);

create index REF_PERSO_ADDRE_IND
    on PERSON (Liv_id);

create unique index ID_STAFF_PERSO_IND
    on STAFF (id);

create unique index ID_HR_STAFF_IND
    on HR (id);

create unique index ID_BLOOD_IND
```

```
    on BLOOD (id);

create index REF_BLOOD_DONATOR_IND
    on BLOOD (donator);

create index REF_BLOOD_TRANS_IND
    on BLOOD (Nee_id);

create unique index ID_TRANSPLANTATION_IND
    on TRANSPLANTATION (id);

create index REF_TRANS_CUSTO_IND
    on TRANSPLANTATION (Rec_id);

create unique index SID_TRANS_ORGAN_IND
    on TRANSPLANTATION (Con_id);

create index REF_TRANS_DOCTO_IND
    on TRANSPLANTATION (D_w_id);

create index REF_TRANS_ANAES_IND
    on TRANSPLANTATION (A_w_id);

create unique index ID_TYPE_DELIVERY_IND
    on TYPE_DELIVERY (id);

create unique index ID_N_work_on_IND
    on N_work_on (N_N_id, id);

create index EQU_N_wor_TRANS_IND
    on N_work_on (id);

-- create personnal index
create unique index PERSON_email
    on PERSON (email);

create index ORGANES_Types
    on ORGANE (type);

-- View Section
-- _____________
create or replace view ACC_BLOOD_PRICE(id_blood, type_blood, signe_blood)
    -- View goal : view accountable to view the price of the organ and the blood
    -- Author: "The Blood" team
    as  select B.id, B.type, B.signe
    from BLOOD B;
```

```
create or replace view ACC_ORGANE_PRICE(id, price)
    -- View goal : view accountable to view the price of the organ
    -- Author: "The Blood" team
    as  select O.id, O.price
    from ORGANE O;


create or replace view DEL_ORDER(first_name, last_name, order_id, street, number,
postal_code, city, country)
    -- View goal : view deliverers to view information for the delivery
    -- Author: "The Blood" team
     as  select D.recipent_first_name, D.recipent_last_name, O.id, A.street, A.number,
A.postal_code, A.city, A.land
    from DELIVERY D, TYPE_DELIVERY TD, ADDRESS A, ORDER_ O
    where
    O.Typ_id = D.id and
    D.Typ_id = TD.id and
    TD.id != "main propre" and
    D.At_id = A.id;


create or replace view RH (ID, SALARY, FIRST_NAME, LAST_NAME, EMAIL,
PHONE, JOB_DESCRIPTION)
    -- View goal : view RH, to view the staff, the wages, jobs
    -- Author: Louise DELPIERRE et Aline Boulanger
       as select S.id, S.salary, P.first_name, P.last_name, P.email, P.phone_number,
S.job_description
    from  STAFF S, PERSON P
    where S.id = P.id ;

create or replace view MEDECIN (organe, organe_id, client, type_blood, signe_blood,
anesthesiste_id, medecin_id, date_, id, customer_id)
    -- View goal: view information on the customer and the organ the doctor will have to
transplant on him
    -- Authors: Eline Mota
     as select O.type, O.id, C.pseudo, C.blood_type, C.blood_sign, A.id, D.id, T.date_,
T.id, C.id
      from ORGANE O, TRANSPLANTATION T, CUSTOMER C, ANAESTHESIST A,
DOCTOR D
    where
    T.Rec_id = C.id and
    T.Con_id = O.id and
    T.D_w_id = D.id and
    T.A_w_id = A.id;


-- Trigger Section
```

```
-- _____

-- create trigger TRG_DELIVERY_DATES_EXPIRATION_CONTROL_INSERT
--       -- Trigger goal: Check if the date of delivery is before the date expiration of the
organe
--     -- Author: Yannis Van Achter
--     before insert on DELIVERY
--     for each row
--     begin
--         SELECT expiration_date INTO expiration
--             FROM ORGANE
--             WHERE ORGANE.id IN (SELECT DETAIL.organe
--                         FROM DETAIL
--                         WHERE DETAIL.id in (SELECT ORDER.id
--                                     FROM ORDER
--                                     WHERE ORDER.Typ_id = new.id)
--                         );

--         if (new.arrival_date < expiration) then
--             signal sqlstate '45000'
--               set message_text = 'The date of delivery must be after the date expiration
of the organe';
--         end if;
--     end;

-- create trigger TRG_DELIVERY_DATES_EXPIRATION_CONTROL_UPDATE
--       -- Trigger goal: Check if the date of delivery is before the date expiration of the
organe
--     -- Author: Yannis Van Achter
--     before update on DELIVERY
--     for each row
--     begin
--         SELECT expiration_date INTO expiration
--             FROM ORGANE
--             WHERE ORGANE.id IN (SELECT DETAIL.organe
--                         FROM DETAIL
--                         WHERE DETAIL.id in (SELECT ORDER.id
--                                     FROM ORDER
--                                     WHERE ORDER.Typ_id = new.id)
--                         );

--         if (new.arrival_date < expiration) then
--             signal sqlstate '45000'
--               set message_text = 'The date of delivery must be after the date expiration
of the organe';
--         end if;
--     end;
```

```
-- create trigger TRG_CHECK_AVAILABILITY_ORGAN_TO_SELL_INSERT
--     -- Trigger goal: Checks if the organ is available before accept to sell it
--     -- Author: Aurélie Genot
--     before insert on DETAIL
--     for each row
--     begin
--         if (new.ORGANE in (SELECT TANSPLANTATION.Con_id
--                                     FROM TRANSPLANTATION
--                                                 WHERE TANSPLANTATION.Con_id =
ORGANE.id))
--         then
--             signal sqlstate '45000'
--                 set message_text = 'The organ that you want to sell is not available
anymore';
--         end if;
--     end;

-- create trigger TRG_CHECK_AVAILABILITY_ORGAN_TO_SELL_UPDATE
--     -- Trigger goal: Checks if the organ is available before accept to sell it
--     -- Author: Aurélie Genot
--     before update on DETAIL
--     for each row
--     begin
--         if (new.ORGANE in (SELECT TANSPLANTATION.Con_id
--                                     FROM TRANSPLANTATION
--                                                 WHERE TANSPLANTATION.Con_id =
ORGANE.id))
--         then
--             signal sqlstate '45000'
--                 set message_text = 'The organ that you want to sell is not available
anymore';
--         end if;
--     end;

-- create trigger TRG_CHECK_AVAILABILITY_ORGAN_TO_TRANSPLANT_INSERT
--     -- Trigger goal: Checks if the organ is available before accept to transplant it
--     -- Author: Aurélie Genot
--     before insert on TRANSPLANTATION
--     for each row
--     begin
--         if (new.ORGANE in (SELECT DETAIL.ORGANE
--                     FROM DETAIL
--                     WHERE DETAIL.ORGANE is not null))
--         then
--             signal sqlstate '45000'
```

```
	--		set message_text = 'The organ that you want to transplant is not available
anymore';
	--	end if;
	--	end;

	-- create trigger TRG_CHECK_AVAILABILITY_ORGAN_TO_TRANSPLANT_UPDATE
	--	-- Trigger goal: Checks if the organ is available before accept to transplant it
	--	-- Author: Aurélie Genot
	--	before update on TRANSPLANTATION
	--	for each row
	--	begin
	--	if (new.ORGANE in ( SELECT DETAIL.ORGANE
	--			FROM DETAIL
	--			WHERE DETAIL.ORGANE is not null))
	--	then
	--		signal sqlstate '45000'
	--		set message_text = 'The organ that you want to transplant is not available
anymore';
	--	end if;
	--	end;

	-- create trigger TRG_CHECK_AVAILABILITY_BLOOD_TO_SELL_INSERT
	--	-- Trigger goal: Checks if the blood is available before accept to sell it
	--	-- Author: Aurélie Genot
	--	before insert on DETAIL
	--	for each row
	--	begin
	--		if (new.BLOOD in (SELECT BLOOD.id FROM BLOOD WHERE
BLOOD.Nee_id is not null))
	--	then
	--		signal sqlstate '45000'
	--			set message_text = 'The blood that you want to sell is not available
anymore';
	--	end if;
	--	end;

	-- create trigger TRG_CHECK_AVAILABILITY_BLOOD_TO_SELL_UPDATE
	--	-- Trigger goal: Checks if the blood is available before accept to sell it
	--	-- Author: Aurélie Genot
	--	before update on DETAIL
	--	for each row
	--	begin
	--		if (new.BLOOD in (SELECT BLOOD.id FROM BLOOD WHERE
BLOOD.Nee_id is not null))
	--	then
	--		signal sqlstate '45000'
```

```
--                 set message_text = 'The blood that you want to sell is not available
anymore';
--         end if;
--     end;


-- create trigger TRG_UPDATE_MEDECINS
--     -- Trigger goal: Before update a medecin checks if this employee has to do a
transplantation (in the future)
--   -- Author: Aurélie Genot
--   before update on DOCTOR
--   for each row
--     begin
--         SELECT count(date_) into transplantation_in_future
--             FROM TRANSPLATATION
--                                 WHERE TANSPLANTATION.D_w_id = new.id and
TANSPLANTATION.date_ > CURRENT_DATE()
--             ORDER BY date DESC
--             LIMIT 1;
--         if (transplantation_in_future > 0) then
--             signal sqlstate '45000'
--              set message_text = 'This person cannot be deleted because she has to do
a transplatation';
--         end if;
--     end;


-- create trigger TRG_CHECK_STAFF_NOT_OPERATING_THEMSELF_INSERT
--     -- Tigger goal: Checks if the receiver of an organ is not operating themself
--     -- Author: Youlan Collard
--     before insert on TRANSPLANTATION
--     for each row
--     begin
--         if (new.Rec_id = new.D_w_id or new.Rec_id = new.A_w_id or exists (
--             SELECT * FROM N_work_on
--             WHERE id = new.id and N_N_id = new.Rec_id
--         ))
--         then
--             signal sqlstate '45000'
--               set message_text = 'The receiver of an organ can not be in the transplant
team';
--         end if;
--     end;

-- create trigger TRG_CHECK_STAFF_NOT_OPERATING_THEMSELF_UPDATE
--     -- Tigger goal: Checks if the receiver of an organ is not operating themself
--     -- Author: Youlan Collard
```

```
	--	before update on TRANSPLANTATION
	--	for each row
	--	begin
	--		if (new.Rec_id = new.D_w_id or new.Rec_id = new.A_w_id or exists (
	--			SELECT * FROM N_work_on
	--			WHERE id = new.id and N_N_id = new.Rec_id
	--		))
	--		then
	--			signal sqlstate '45000'
	--			  set message_text = 'The receiver of an organ can not be in the transplant
team';
	--		end if;
	--	end;

	-- Role section
	-- _____

	-- create section
	create role IF NOT EXISTS CEO;
	create role IF NOT EXISTS HR;
	create role IF NOT EXISTS DOCTOR;
	create role IF NOT EXISTS ACCOUNTANT;
	create role IF NOT EXISTS ADMINISTRATIVE;
	create role IF NOT EXISTS CUSTOMERS;

	-- Grant section
	grant select, insert, update, delete on ADDRESS to CUSTOMERS;
	grant select, insert, update, delete on PERSON to CUSTOMERS;
	grant select, insert, update, delete on CUSTOMER to CUSTOMERS;
	grant select, insert, update, delete on ORDER_ to CUSTOMERS;
	grant select, insert, update, delete on DELIVERY to CUSTOMERS;
	grant select, insert, update, delete on DETAIL to CUSTOMERS;
	grant select on BLOOD to CUSTOMERS;
	grant select on ORGANE to CUSTOMERS;
	grant select on TRANSPLANTATION to CUSTOMERS;


	grant select, insert, update, delete on ADDRESS to CEO;
	grant select, insert, update, delete on ADDRESS to HR;

	grant select, insert, update, delete on PERSON to CEO;
	grant select, insert, update, delete on PERSON to HR;

	grant select, insert, update, delete on CUSTOMER to CEO;

	grant select, insert, update, delete on TYPE_DELIVERY to CEO;
	grant select, insert, update on TYPE_DELIVERY to ACCOUNTANT;
```

```
grant select, insert, update, delete on DELIVERY to CEO;
grant select, insert, update, delete on DELIVERY to ADMINISTRATIVE;

grant select, insert, update, delete on ORDER_ to CEO;
grant select, insert, update, delete on ORDER_ to ADMINISTRATIVE;

grant select, insert, update, delete on STAFF to CEO;
grant select, insert, update, delete on STAFF to HR;

grant select, insert, update, delete on CEO to CEO;
grant select, insert, update, delete on HR to CEO;
grant select, insert, update, delete on ANAESTHESIST to CEO;
grant select, insert, update, delete on NURSE to CEO;
grant select, insert, update, delete on DOCTOR to CEO;
grant select, insert, update, delete on ACCOUNTANT to CEO;
grant select, insert, update, delete on STAFF to CEO;

grant select, insert, update, delete on DOCTOR to HR;
grant select, insert, update, delete on NURSE to HR;
grant select, insert, update, delete on ANAESTHESIST to HR;
grant select, insert, update, delete on ACCOUNTANT to HR;
grant select, insert, update, delete on STAFF to HR;
grant select, insert, update, delete on HR to HR;
grant select on CEO to HR;

grant select, insert, update, delete on DETAIL to ADMINISTRATIVE;
grant select, insert, update, delete on DETAIL to CEO;

grant select, insert, update, delete on BLOOD to ADMINISTRATIVE;
grant select, insert, update, delete on BLOOD to CEO;
grant select, insert, update, delete on BLOOD to ACCOUNTANT;

grant select, insert, update, delete on DONATOR to ADMINISTRATIVE;
grant select, insert, update, delete on DONATOR to CEO;
grant select, insert, update, delete on DONATOR to ACCOUNTANT;

grant select, insert, update, delete on ORGANE to ADMINISTRATIVE;
grant select, insert, update, delete on ORGANE to CEO;
grant select, insert, update, delete on ORGANE to ACCOUNTANT;

grant select, insert, update, delete on TRANSPLANTATION to ADMINISTRATIVE;
grant select, insert, update, delete on TRANSPLANTATION to CEO;
grant select, insert, update, delete on TRANSPLANTATION to ACCOUNTANT;
grant select on TRANSPLANTATION to HR;

grant select, insert, update, delete on N_work_on to ADMINISTRATIVE;
```

```
grant select, insert, update, delete on N_work_on to CEO;
grant select on N_work_on to HR;

grant select, insert, update, delete on ACC_BLOOD_PRICE to ACCOUNTANT;
grant select, insert, update, delete on ACC_ORGANE_PRICE to ACCOUNTANT;

grant select, insert, update, delete on DEL_ORDER to ADMINISTRATIVE;

-- Init Section
-- _____________

insert into TYPE_DELIVERY (id, price, estimated_days) values ('normal', 5.0, 10);
insert into TYPE_DELIVERY (id, price, estimated_days) values ('express', 10.0, 3);
insert into TYPE_DELIVERY (id, price, estimated_days) values ('international', 15.0,
7);
insert into TYPE_DELIVERY (id, price, estimated_days) values ('main propre', 3.0, 1);

-- Init for tests (base of data)
-- ___________________________________

-- Anonymization of the database
insert into ADDRESS (street, number, postal_code, city, land, id) values
("Anonymized", 1, 1, "Anonymized", "Anonymized", 1);
insert into PERSON (id, last_name, first_name, email, phone_number, password,
born_date, Liv_id)
        values (1, "Anonymized", "Anonymized", "anonymized@anonymized.com",
"Anonymized", "Anonymized", "0001-01-01", 1);
insert into CUSTOMER (id, blood_type, blood_sign, pseudo) values (1, 'A', 1,
"Annonimous");

-- insert of personn, customers and staff members
insert into ADDRESS (id, street, number, city, postal_code, land) values (2, 'Rue de la
Loi', 16, 'Bruxelles', 1000, 'Belgique');
insert into PERSON (id, last_name, first_name, email, phone_number, password,
born_date, Liv_id) values (2, 'Van Achter', 'Yannis', "yannis.van.achter@test.gmail.com",
"+32 470 00 00 00", "password", "1997-01-01", 2);
insert into CUSTOMER (id, blood_type, blood_sign, pseudo) values (2, 'A', 1,
"Yannis");
insert into STAFF (id, salary, active, job_description) values (2, 2000, true, "CEO");
insert into CEO (id) values (2);

insert into ADDRESS (id, street, number, city, postal_code, land) values (3, 'Rue des
Anges', 16, 'Bruxelles', 1000, 'Belgique');
insert into PERSON (id, last_name, first_name, email, phone_number, password,
born_date, Liv_id) values (3, 'Genot', 'Aurélie', "aurelie.genot@test.gmail.com", "+32 470 00
00 01", "password", "1997-05-07", 3);
```

```
insert into STAFF (id, salary, active, job_description) values (3, 2000, true, "HR Manager");
insert into HR (id) values (3);

insert into ADDRESS (id, street, number, city, postal_code, land) values (4, "Rue de l'amitié", 16, 'Bruxelles', 1000, 'Belgique');
insert into PERSON (id, last_name, first_name, email, phone_number, password, born_date, Liv_id) values (4, 'Collard', 'Youlan', "youlan.collard@test.gmail.com", "+32 302 08 08 02", 'password', "1997-05-07", 4);
insert into STAFF (id, salary, active, job_description) values (4, 2000, true, "Doctor general");
insert into DOCTOR (id, inami_number) values (4, '83678643923');

insert into ADDRESS (id, street, number, city, postal_code, land) values (5, "Rue des amies vocal", 1, 'Bruxelles', 1000, 'Belgique');
insert into PERSON (id, last_name, first_name, email, phone_number, password, born_date, Liv_id) values (5, 'Boulanger', 'Aline', "aline.boulanger@test.gmail.com", "+32 903 22 20 01", "password", "1997-05-07", 5);
insert into STAFF (id, salary, active, job_description) values (5, 2000, true, "General nurse");
insert into NURSE (id) values (5);

insert into PERSON (id, last_name, first_name, email, phone_number, password, born_date, Liv_id) values (6, 'Delpierre', 'Louise', "louise.delpierre@test.gmail.com", "+32 032 83 92 78", "password", "1997-05-07", 5);
insert into STAFF (id, salary, active, job_description) values (6, 2000, true, "General anaesthesist");
insert into ANAESTHESIST (id, inami_number) values (6, '29878470982');

insert into ADDRESS (id, street, number, city, postal_code, land) values (6, "Rue de la bonté", 92, "Paris", 9000, "France");
insert into PERSON (id, last_name, first_name, email, phone_number, password, born_date, Liv_id)
values (7, "Mota", "Eline", "mota.eline@test.gmail.com", "+32 032 83 92 78", "password", "1997-05-07", 6);
insert into STAFF (id, salary, active, job_description) values (7, 2000, true, "General accountant");
insert into ACCOUNTANT (id) values (7);

insert into PERSON (last_name, first_name, email, phone_number, password, born_date, Liv_id)
values ('Boulanger2', 'Aline2', "aline.boulanger@test2.gmail.com", "+32 903 22 20 01", "password", "1997-05-07", 5);
insert into PERSON (last_name, first_name, email, phone_number, password, born_date, Liv_id)
values ('Boulanger3', 'Aline3', "aline.boulanger@test3.gmail.com", "+32 903 22 20 01", "password", "1997-05-07", 5);
```

```
insert into PERSON (last_name, first_name, email, phone_number, password, born_date, Liv_id)
values ('Boulanger4', 'Aline4', "aline.boulanger@test4.gmail.com", "+32 903 22 20 01", "password", "1997-05-07", 5);
insert into PERSON (last_name, first_name, email, phone_number, password, born_date, Liv_id)
values ('Boulanger5', 'Aline5', "aline.boulanger@test5.gmail.com", "+32 903 22 20 01", "password", "1997-05-07", 5);

-- insert blood and organs
insert into DONATOR (id, gender, age_range) values (1, False, 32);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (1, 'A', True, '2020-01-01', 250, 1);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (2, 'A', True, '2030-01-01', 250, 1);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (3, 'A', True, '2030-01-01', 250, 1);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (4, 'A', True, '2030-01-01', 250, 1);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (5, 'A', True, '2030-01-01', 250, 1);
insert into BLOOD (id, type, signe, expiration_date, price, donator) values (6, 'A', True, '2030-01-01', 250, 1);
insert into ORGANE (id, state, functionnal, expiration_date, expiration_date_transplantation, method_of_preservation, type, price, Com_id)
  values (1, "very well", True, "2024-05-04", "2023-11-10", "Dry at ambiant temperature", "heart", 2000000, 1);
insert into ORGANE (id, state, functionnal, expiration_date, expiration_date_transplantation, method_of_preservation, type, price, Com_id)
  values (2, "very well", True, "2024-05-04", "2023-11-10", "Dry at ambiant temperature", "foot", 5000, 1);
insert into ORGANE (id, state, functionnal, expiration_date, expiration_date_transplantation, method_of_preservation, type, price, Com_id)
  values (3, "very well", True, "2024-05-04", "2023-11-10", "Dry at ambiant temperature", "large intestine", 13000, 1);
insert into ORGANE (id, state, functionnal, expiration_date, expiration_date_transplantation, method_of_preservation, type, price, Com_id)
  values (4, "very well", True, "2024-05-04", "2023-11-10", "Dry at ambiant temperature", "liver", 2000000, 1);
insert into ORGANE (id, state, functionnal, expiration_date, expiration_date_transplantation, method_of_preservation, type, price, Com_id)
  values (5, "very well", True, "2024-05-04", "2023-11-10", "Dry at ambiant temperature", "ear", 2000000, 1);


 insert into TRANSPLANTATION(id, date_, Con_id, price, Rec_id, D_w_id, A_w_id)
 values(1, '2021-01-03', 1, '45.6', 2, 4, 6);
```

```
insert into N_work_on(N_N_id, id)
 values(5, 1);
insert into BLOOD (type, signe, expiration_date, price, donator, Nee_id) values ('A', True, '2020-01-01', 250, 1, 1);
```