```
CREATE TABLE utilisateur (
  "idutil" bigint NOT NULL,
  "nomutil" character varying,
  "prenomutil" character varying,
  "mailutil" character varying,
  "numtel" character varying,
  "loginutil" character varying,
  "mdputil" character varying,
  "adresseutil" character varying,
  "dateembauche" date,
  "idville" bigint NOT NULL
);

CREATE TABLE profil (
  "idprofil" bigint NOT NULL,
  "nomprofil" character varying
);

CREATE TABLE ville (
  "idville" bigint NOT NULL,
  "nomville" character varying
);

CREATE TABLE droit (
  "iddroit" bigint NOT NULL,
  "nomdroit" character varying
);

CREATE TABLE profil_utilisateur (
  "idutil" bigint NOT NULL,
  "idprofil" bigint NOT NULL
);

CREATE TABLE profil_droit (
  "idprofil" bigint NOT NULL,
  "iddroit" bigint NOT NULL
);

CREATE TABLE trace (
  "id" serial NOT NULL,
  "tablename" character varying,
  "idoccurence" bigint NOT NULL,
  "datetrace" date,
  "idtypemodif" bigint NOT NULL
);

CREATE TABLE type_modif (
  "idtypemodif" serial NOT NULL,
  "libelle" character varying
);

--
-- Primary Keys
--

ALTER TABLE ONLY utilisateur ADD CONSTRAINT utilisateurs_pk PRIMARY KEY ("idutil");

ALTER TABLE ONLY profil ADD CONSTRAINT profil_pk PRIMARY KEY ("idprofil");

ALTER TABLE ONLY ville ADD CONSTRAINT ville_pk PRIMARY KEY ("idville");

ALTER TABLE ONLY droit ADD CONSTRAINT droit_pk PRIMARY KEY ("iddroit");

ALTER TABLE ONLY profil_droit ADD CONSTRAINT "profil_droit_PK" PRIMARY KEY ("idprofil",
"iddroit");

ALTER TABLE ONLY profil_utilisateur ADD CONSTRAINT "profil_utilisateur_PK" PRIMARY KEY (
"idprofil", "idutil");
```

```
ALTER TABLE ONLY type_modif ADD CONSTRAINT "type_modif_PK" PRIMARY KEY ("idtypemodif");

ALTER TABLE ONLY trace ADD CONSTRAINT "trace_PK" PRIMARY KEY ("id");


--
-- Foreign Keys
--

ALTER TABLE ONLY profil_droit ADD CONSTRAINT "profil_droit_droit_FK" FOREIGN KEY ("iddroit")
REFERENCES droit("iddroit");

ALTER TABLE ONLY profil_droit ADD CONSTRAINT "profil_droit_profil_FK" FOREIGN KEY ("idprofil"
) REFERENCES profil("idprofil");

ALTER TABLE ONLY profil_utilisateur ADD CONSTRAINT "profil_utilisateur_util_FK" FOREIGN KEY (
"idutil") REFERENCES utilisateur("idutil");

ALTER TABLE ONLY profil_utilisateur ADD CONSTRAINT "profil_utilisateur_profil_FK" FOREIGN KEY
 ("idprofil") REFERENCES profil("idprofil");

ALTER TABLE ONLY trace ADD CONSTRAINT "trace_type_modif_FK" FOREIGN KEY ("idtypemodif")
REFERENCES type_modif("idtypemodif");

ALTER TABLE ONLY utilisateur ADD CONSTRAINT "ville_utilisateur_FK" FOREIGN KEY ("idville")
REFERENCES ville("idville");


--
-- Procédures stockées
--

--
-- Création de n ville(s)
--

CREATE OR REPLACE FUNCTION create_ville(nombre bigint) RETURNS void AS
$$
DECLARE
  id bigint;
  i int;
BEGIN
  SELECT INTO id max(idville) FROM ville;
  IF id ISNULL THEN
    id:=0;
  END IF;

  FOR i IN 1..nombre LOOP
    id:=id+1;
    INSERT INTO ville (idville, nomville) VALUES(id, 'ville' || i);
  END LOOP;
END;
$$ LANGUAGE plpgsql;

--
-- Création de n utilisateur(s)
--

CREATE OR REPLACE FUNCTION create_user(nombre bigint) RETURNS void AS
$$
DECLARE
  id bigint;
  i int;
BEGIN
    SELECT INTO id max(idutil) FROM utilisateur;
    IF id ISNULL THEN
      id:=0;
    END IF;

    FOR i IN 1..nombre LOOP
      id:=id + 1;
```

```
      INSERT INTO utilisateur (idutil, nomutil, prenomutil, mailutil, numtel, loginutil,
      mdputil, adresseutil, dateembauche, idville) VALUES (id, 'nom' || i, 'prenom1' || i,
      'mail1@mail.fr', '0600' || i, 'login' || i, 'mdp' || i, 'adresse' || i, to_date(
      '2015-06-01', 'YYYY-MM-DD'), 1);
    END LOOP;
END;
$$ LANGUAGE plpgsql;

--
-- Création de n profil(s)
--

CREATE OR REPLACE FUNCTION create_profil(nombre bigint) RETURNS void AS
$$
DECLARE
  id bigint;
  i int;
BEGIN
    SELECT INTO id max(idprofil) FROM profil;
    IF id ISNULL THEN
      id:=0;
    END IF;

    FOR i IN 1..nombre LOOP
      id:=id + 1;
      INSERT INTO profil (idprofil, nomprofil) VALUES (id, 'profil' || i);
    END LOOP;
END;
$$ LANGUAGE plpgsql;

--
-- Création de n droit(s)
--

CREATE OR REPLACE FUNCTION create_droit(nombre bigint) RETURNS void AS
$$
DECLARE
  id bigint;
  i int;
BEGIN
    SELECT INTO id max(iddroit) FROM droit;
    IF id ISNULL THEN
      id:=0;
    END IF;

    FOR i IN 1..nombre LOOP
      id:=id + 1;
      INSERT INTO droit (iddroit, nomdroit) VALUES (id, 'droit' || i);
    END LOOP;
END;
$$ LANGUAGE plpgsql;

--
-- Creation des utilisateurs associés aux profils
--

CREATE OR REPLACE FUNCTION create_profil_utilisateur() RETURNS void as
$$
DECLARE
  maxprofil bigint;
  maxuser bigint;
  i bigint;
  l_idprofil bigint;
BEGIN
  SELECT INTO maxprofil max(idprofil) FROM profil;
  SELECT INTO maxuser max(idutil) FROM utilisateur;

  FOR i IN 1..maxprofil LOOP
    IF i = 1 THEN
      INSERT INTO profil_utilisateur (idutil, idprofil) VALUES (i, i);
    ELSE
```

```
      INSERT INTO profil_utilisateur (idutil, idprofil) VALUES (i, i);
      INSERT INTO profil_utilisateur (idutil, idprofil) VALUES (i, i - 1);
    END IF;
  END LOOP;

    l_idprofil:=1;
    FOR k IN 11..maxuser LOOP
        INSERT INTO profil_utilisateur (idutil, idprofil) VALUES (k, l_idprofil);

        IF l_idprofil = maxprofil THEN
            l_idprofil:=1;
        ELSE
            l_idprofil:= l_idprofil + 1;
        END IF;
    END LOOP;

END;
$$ LANGUAGE plpgsql;

--
-- Création des droits associés aux profils
--

CREATE OR REPLACE FUNCTION create_profil_droit() RETURNS void as
$$
DECLARE
  maxprofil bigint;
  maxdroit bigint;
  l_iddroit bigint;
BEGIN
    SELECT INTO maxprofil max(idprofil) FROM profil;
    SELECT INTO maxdroit max(iddroit) FROM droit;

    l_iddroit:=1;
    FOR i IN 1..maxprofil LOOP
        IF l_iddroit = maxdroit + 1 THEN
            l_iddroit:= 1;
        END IF;

        IF l_iddroit = 1 THEN
            INSERT INTO profil_droit (idprofil, iddroit) VALUES (i, l_iddroit);
            l_iddroit:= l_iddroit + 1;
        ELSE
            INSERT INTO profil_droit (idprofil, iddroit) VALUES (i, l_iddroit);
            INSERT INTO profil_droit (idprofil, iddroit) VALUES (i, l_iddroit - 1);
            l_iddroit:= l_iddroit + 1;
        END IF;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION create_type_modif() RETURNS void as
$$
BEGIN
    INSERT INTO type_modif (libelle) VALUES ('INSERT');
    INSERT INTO type_modif (libelle) VALUES ('UPDATE');
    INSERT INTO type_modif (libelle) VALUES ('DELETE');
END;
$$ LANGUAGE plpgsql;

select create_type_modif();
select create_ville(500);
select create_user(1000);
select create_profil(50);
select create_droit(20);
select create_profil_utilisateur();
select create_profil_droit();

--
-- TRIGGERS POUR LA TABLE DE LOG
--
```

```
CREATE OR REPLACE FUNCTION log() RETURNS trigger AS $log$
    BEGIN
        IF (TG_OP = 'INSERT') THEN
            INSERT INTO trace (tablename, idoccurence, datetrace, idtypemodif) SELECT
            TG_TABLE_NAME, TG_RELID, now(), 1;
        ELSIF (TG_OP = 'UPDATE') THEN
            INSERT INTO trace (tablename, idoccurence, datetrace, idtypemodif) SELECT
            TG_TABLE_NAME, TG_RELID, now(), 2;
        ELSIF (TG_OP = 'DELETE') THEN
            INSERT INTO trace (tablename, idoccurence, datetrace, idtypemodif) SELECT
            TG_TABLE_NAME, TG_RELID, now(), 3;
        END IF;
        RETURN NULL;
    END;
$log$ LANGUAGE plpgsql;

DO $$
DECLARE
    tables CURSOR FOR SELECT tablename FROM pg_tables WHERE schemaname = 'public' ORDER BY
    tablename;
    table_record name;
    id bigint;
BEGIN

    OPEN tables;

    id:=0;
    LOOP
        FETCH tables INTO table_record;
            IF table_record IS NOT NULL THEN
                IF table_record <> 'trace' THEN
                    EXECUTE 'CREATE TRIGGER log AFTER INSERT OR UPDATE OR DELETE ON ' ||
                    table_record || ' FOR EACH ROW EXECUTE PROCEDURE log()';
                END IF;
            END IF;
        EXIT WHEN NOT FOUND;
    END LOOP;

    CLOSE tables;
END$$;
```