Rapport SQL

Mahdi Larbi et Mohamed Ramdane Debiane

1 Décembre 2019

Contents

1 Modification du modèle E/A

Le modèle E/A de départ a été modifié, car certaines erreures s'étaient glissés dans la conception.
Certaines entités on été supprimés:
L'adiministrateur, étant unique, n'a pas besoin d'être representé dans le modèle, les entités représentant la visite (A_visiter, Visite, _A_visiter) on été supprimés.
Nous avons remplacé les utilisateurs visteurs et annonceurs par une généralisation des utilisateurs en visteurs et_ou Annonceurs.
Les produits aussi on été spécialisés en sous catégories de produits.
Certaines informations de ces entités une fois le modèle relationnel crée seront masquées par l'utilisations de vues (cf. Section 2)
L'entité TypeAnnonce disparait et deviant un attribut dans l'entité annonce.
Ville elle aussi disparait pour devenir un attribut dans les entités Annonceurs et Annonce.

L'ajout d'une entité Photo va servir à stocker et retrouver les photos correspondant à chaque annonce.
Le fait que les utilisateurs peuvent tous consulter des annonces alors que seulent les annonceurs peuvent en publier explique les relations Consulter entre Utilisateur et Annonces, Annonceurs et Annonces .

2 Modèle Relationnel

Nous avons appliqué les régles de transformations suivantes pour obtenir le nouveau modèle relationnel:
Les relations 1..N on vu les clé primaires des relations du coté 1 exporté dans les tables cibles (Voir annonces et photos).
Les clés primaires des tables Annonceurs et produits on été exportés comme clés étrangères dans la table Annonce .

2.1 Liste des tables

Nous obtenons ainsi le modèle relationnel suivant:
Utilisateur(id_user, time_access, adresse_ip, type_user)
Visiteur(id_visiteur)
Annonceur(id_annonceur , login , password, mail, nom, prenom, ville , téléphone , annonceur_photo)
Produit(id_produit , marque, modèle, poids , état , catégorie)
Annonce(id_annonce, titre_annonce, prix , ville, type_annonce, time_pub , id_annonceur, id_produit
Consulter(id_user, id_annonce , date_consultation
Publier(d_annonce , id_annonceur, date_publication)
Photo(id_photo , id_annonce , photo)
PC(id_produit, diagonale, processeur, c_g, ram, type_disque, taille_disque, batterie)
TV(id_produit, diagonale , définition, tech , os, connectique)
Téléphonie(id_produit, diagonale, processeur, ram , taille_disque, os , batterie, nb_sim, type_sim, res_app_arr, res_app_av, nfc)
App_photo(id_produit, resolution, format_cap , definition , type_memoire, type_ecran , tech)

3 Implémentaion en SQL

Est joint en annexe de se document le script SQL prêt à être importé dans un SGBD.

3.1 Création de la BD et des tables

La création de la base de donnée se fait avec les commandes suivantes:

Listing 1: Création

```
DROP DATABASE IF EXISTS techstore;
create database techstore;
USE techstore;
```

La création des tables se fait selon les modèle relationnel. (cf Techstor.sql) lignes 10 à 175
Plus des vues seront ajoutés afin de filtrer les information accessibles au utilisateurs (cf. 3.4)

3.2 Contraintes de clés étrangères

Les clés étrangères sont gérés par des ALTER TABLE sur les tables correspondantes:
Les table accessoires,annonce app_photo, pc, téléphonie, tv contiennent une clé étrangère référençant la table produit

Listing 2: Clé étrangères produit

```
ALTER TABLE accesoires
  ADD CONSTRAINT accesoires_fk_produit FOREIGN KEY ( id_produit ) REFERENCES
produit ( id_produit );

ALTER TABLE annonce
  ADD CONSTRAINT annonce_fk_annonceur FOREIGN KEY ( id_annonceur ) REFERENC
annonceur ( id_annonceur ),
  ADD CONSTRAINT annonce_fk_produit FOREIGN KEY ( id_produit ) REFERENCES
produit ( id_produit );

ALTER TABLE app_photo
  ADD CONSTRAINT app_photo_fk_produit FOREIGN KEY ( id_produit ) REFERENCES
produit ( id_produit );

ALTER TABLE pc
  ADD CONSTRAINT pc_fk_produit FOREIGN KEY ( id_produit ) REFERENCES
produit ( id_produit );

ALTER TABLE photo
  ADD CONSTRAINT photo_fk_annonce FOREIGN KEY ( id_annonce ) REFERENCES
annonce ( id_annonce );

ALTER TABLE telephonie
```

```
  ADD CONSTRAINT telephonie_fk_produit FOREIGN KEY ( id_produit ) REFERENCE
produit ( id_produit );

ALTER TABLE tv
  ADD CONSTRAINT tv_fk_produit FOREIGN KEY ( id_produit ) REFERENCES
produit ( id_produit );
```

La clé primaire des tables annonce à été exportée comme clé étrangère dans les tables publier photo consulter

Listing 3: Clé étrangères Annonce

```
ALTER TABLE consulter
  ADD CONSTRAINT consulter_fk_user FOREIGN KEY ( id_user ) REFERENCES
user ( id_user ),
  ADD CONSTRAINT consulter_fk_annonce FOREIGN KEY ( id_annonce ) REFERENCES
annonce ( id_annonce );

ALTER TABLE photo
  ADD CONSTRAINT photo_fk_annonce FOREIGN KEY ( id_annonce ) REFERENCES
annonce ( id_annonce );

 ALTER TABLE publier
  ADD CONSTRAINT publier_fk_annonceur FOREIGN KEY ( id_annonceur ) REFERENC
annonceur ( id_annonceur ),
  ADD CONSTRAINT publier_fk_annonce FOREIGN KEY ( id_annonce ) REFERENCES
annonce ( id_annonce );
```

Pour finir la clé id_user est exportée dans les tables Visiteur et Annonceur

Listing 4: Clé étrangères User

```
ALTER TABLE annonceur
  ADD CONSTRAINT annonceur_fk_user FOREIGN KEY ( id_annonceur ) REFERENCES
user ( id_user );
ALTER TABLE visiteur
  ADD CONSTRAINT visiteur_fk_user FOREIGN KEY ( id_visiteur ) REFERENCES
user ( id_user );
```

3.3 Gestion des droits

3.3.1 Administrateur

La gestion des droits se fait de telle sorte à ce que l'administrateur possède tout les droits (Insert, Delete , Update, Create) sur toutes les tables de la base.
Pour des raison de sécurité il travaillera sur des vues des Annonceurs pour conserver la confidentialité.

Listing 5: Droits Administrateurs

```
DROP USER IF EXISTS 'admin'@'localhost';
```

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON techstore.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;
```

3.3.2 Annonceur

Posséde le droit de Select sur toutes les vues concernant les produits et toutes les sous catégories de produits et les annonceurs.
Ainsi que le droit de DELETE INSERT et UPDATE sur les annonces et photos lui appartenant.

Listing 6: Droits Annonceur

```
DROP USER IF EXISTS 'annonceur'@'localhost';

CREATE USER 'annonceur'@'localhost' IDENTIFIED BY 'password';

GRANT SELECT ON techstore.v_acc TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.v_annonceur TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.v_app_photo TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.v_pc TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.v_telephonie TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.v_tv TO 'annonceur'@'localhost';
GRANT SELECT, INSERT,UPDATE ON techstore.annonce TO 'annonceur'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON techstore.photo TO 'annonceur'@'loca
GRANT SELECT ON techstore.publier TO 'annonceur'@'localhost';
GRANT SELECT ON techstore.consulter TO 'annonceur'@'localhost';

FLUSH PRIVILEGES;
```

3.3.3 Visiteur

Le visiteur quand a lui n'a que le droit d'acceder au annonces, produits et annonceurs qui l'intéressent dans pouvoir ni ajouter ni supprimer ni modifier de tuples

Listing 7: Droits Visiteur

```
DROP USER IF EXISTS 'visiteur'@'localhost';

CREATE USER 'visiteur'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT ON techstore.v_acc TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.v_annonceur TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.v_app_photo TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.v_pc TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.v_telephonie TO 'visiteur'@'localhost';
```

```
GRANT SELECT ON techstore.v_tv TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.annonce TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.photo TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.publier TO 'visiteur'@'localhost';
GRANT SELECT ON techstore.consulter TO 'visiteur'@'localhost';
FLUSH PRIVILEGES;
```

3.4 Création des vues

Ne pouvant pas representer l'héritage dans un modèle relationel, nous avons décider de conserver les tables concernés et de monter_cacher des attributs grace à la mise en place de vues.
La relation de Généralisation_Spécialisation entre les tables User et Annonceurs et Visiteurs et représenté par les vues suivantes permettant la duplication d'infomrtaions et le masquage d'informations sensibles telles que les paires de login_mdp

Listing 8: Vue Visiteur Annonceur

```
DROP VIEW IF EXISTS v_visiteur;

CREATE VIEW v_visiteur AS (
        SELECT id_visiteur, time_access, adress_ip
    FROM user u , visiteur v
    WHERE u.id_user = v.id_visiteur

);
DROP VIEW IF EXISTS v_annonceur;

CREATE VIEW v_annonceur AS (
        SELECT  id_annonceur,time_access,adress_ip,
                    mail,nom,prenom,ville,
                    telephone,annonceur_photo
        FROM user u , annonceur a
    WHERE u.id_user = a.id_annonceur
    );
```

Nous avons procédé de la même manière pour les produit et les sous catégories de produits

Listing 9: Vue Visiteur Annonceur

```
DROP VIEW IF EXISTS v_pc;

CREATE VIEW v_pc AS (
        SELECT p.id_produit, marque, modele, poids, etat
                diagonale,processeur,c_g,ram,type_disque,taille_disque,batter
    FROM produit p, pc
```

```
        WHERE p.id_produit = pc.id_produit
);

DROP VIEW IF EXISTS v_telephonie;

CREATE VIEW v_telephonie AS (
        SELECT p.id_produit, marque, modele, poids, etat
                diagonale,processeur,ram,taille_disque,os,batterie,
                nb_sim,type_sim,res_app_arr,res_app_av, nfc
    FROM produit p, telephonie t
        WHERE p.id_produit = t.id_produit
);

DROP VIEW IF EXISTS v_tv;

CREATE VIEW v_tv AS (
        SELECT p.id_produit, marque, modele, poids, etat
                diagonale,definition,tech,os,connectique
    FROM produit p, tv
        WHERE p.id_produit = tv.id_produit
);

DROP VIEW IF EXISTS v_app_photo;

CREATE VIEW v_app_photo AS (
        SELECT p.id_produit, marque, modele, poids, etat
                resolution,format_cap,definition,type_memoire,type_ecran,tech
    FROM produit p, app_photo ap
        WHERE p.id_produit = ap.id_produit
);

DROP VIEW IF EXISTS v_acc;

CREATE VIEW v_acc AS (
        SELECT p.id_produit, marque, modele, poids, etat
    FROM produit p, app_photo ap
        WHERE p.id_produit = ap.id_produit
);
```

Cette vue particuliere permet de regrouper chaque utilisateur en fonction de son nombre d'annonce dans chaque catégorie:

3.5 Requêtes

Nous avons pensé à un ensemble de requêtes statistiques pour suivres divers métriques quand à l'utilisation de note site:

3.5.1 Requêtes simples

Elles calculent respectivement le nombre d'annonces par villes, par catégories et le nombre de consultation pour chaque annonce

Listing 10: Requêtes simples

```
/* Nombre d'annonces pour chaque ville */
SELECT ville, COUNT(*) FROM annonce
GROUP BY ville

/* Nombre d'annonces par catgorie */
SELECT categorie, COUNT(*)
FROM produit p, annonce a
where p.id_produit = a.id_produit
GROUP BY categorie

/* Nombre de consultation de chaque annonce*/
SELECT id_annonce, COUNT(*) nbr_consultation
from consulter
GROUP by id_annonce;
```

3.5.2 Requêtes complexes

Les requêtes suivantes permettent de determiner le nombre de consultation de chaque utilisateur par catégorie et la catégorie la plus visitée pour permettre de proposer des annonces de plus en plus pertinantes et suceptibles de les interesser

Listing 11: Proposition d'annonces

```
SELECT u.id_user, categorie , COUNT(*) nbr_consultation
FROM consulter c , user u, annonce a , produit p
WHERE c.id_user = u.id_user
and c.id_annonce = a.id_annonce
and a.id_produit = p.id_produit
and u.id_user = /*Id utilisateur*/
GROUP by categorie;

SELECT u.id_user, categorie , COUNT(*) nbr_consultation
FROM consulter c , user u, annonce a , produit p
WHERE c.id_user = u.id_user
and c.id_annonce = a.id_annonce
```

```
and a.id_produit = p.id_produit
GROUP by u.id_user, categorie;
```

Les annonces suivantes servent à la gestion de la base de donnée:
Trouver les annonceurs inactifs et lister toutes les annonces (Utilisée dans une recherche par exemple)

Listing 12: Autres

```
SELECT id_annonce
FROM annonce a , produit p
WHERE a.id_produit = p.id_produit
and categorie = 'Choisir une categorie';

SELECT a2.id_annonceur
FROM annonce a, annonceur a2
WHERE a2.id_annonceur NOT IN (SELECT a3.id_annonceur
                              FROM publier p , annonceur a3
                              WHERE p.id_annonceur = a3.id_annonceur)
```

Cette requête donne le classement des annonceurs par nombre d'annonces et par catégorie

Listing 13: Classement des annonceurs

```
SELECT a.id_annonceur, pr.categorie ,COUNT(*) AS Nombre
FROM annonceur a , annonce an , publier p , produit pr
WHERE a.id_annonceur = p.id_annonceur
AND p.id_annonce = an.id_annonce
AND pr.id_produit = an.id_produit
GROUP BY a.id_annonceur, pr.categorie
```