2022 – 2023

Groupe G2A-9

Récapitulatif SGBD SAÉ S3

Thomas Demeyere

Anton Xu

Louis Yvelin

Oryann Prochaska

Sommaire

Retour au sommaire

Introduction

Durant cette SAÉ, nous dûmes coder un site web d'e-commerce pour une entreprise cliente. Il nous fallut ainsi concevoir une base de données séante à ses besoins. À savoir : créer des comptes utilisateurs·rices, ajouter des produits, stocker des commandes, etc.

Pour ce faire, nous entreprîmes d'esquisser un diagramme de classes (Annexe 1) qui prit en compte toutes les spécificités du site web. Bien que cette version ne fût pas la dernière ébauche, nous entamâmes l'élaboration de la base de données en SQL Oracle (Annexe 2).

C'est M. Demeyere qui prit la main sur l'écriture du script d'insertion des données dont je donne un exemplaire abrégé ci-dessous (Annexe 3). Il est ici question d'insérer des données validant le fonctionnement des dépendances de la base et complétant le site internet, afin de le construire sur des exemples réels et par-dessus tout fonctionnels !

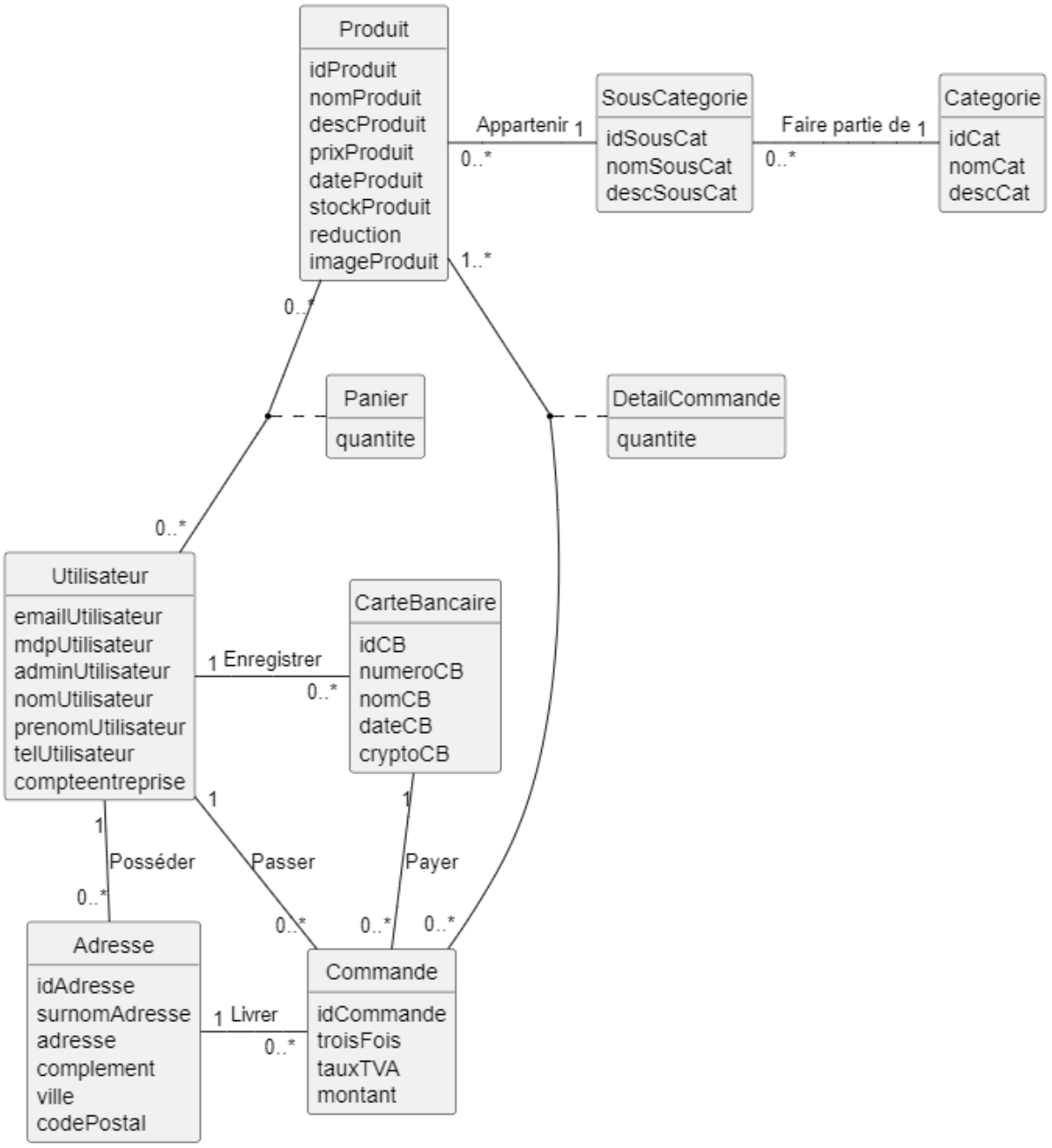
Dans le cadre de l'utilisation des données au sein du site web, il est plus aisé et parfois nécessaire de faire appel à des procédures. Ces dernières, couchées sur le papier par M. Xu et Mlle Prochaska (Annexe 4), permettent — à l'issue de l'ajout d'un article au panier ou à la validation d'une commande — au serveur PHP d'exécuter des requêtes SQL complexes gracieusement. Cela rend le code plus lisible et modulaire, les procédures étant stockées dans la base de données.

Néanmoins, il arrive qu'il faille mettre à jour des données automatiquement, c'est le cas du nombre de produits total en stock après le passage d'une commande. Il est alors impératif d'employer des déclencheurs (Annexe 5).

Plus de détails étudiants des points à ne pas rater vous sont offerts sous chaque annexe.

Retour au sommaire

1 — Diagramme de classes



L'utilisateur est au centre du fonctionnement du site web, il peut enregistrer plusieurs cartes bleues, posséder plusieurs adresses et passer plusieurs commandes. Celles-ci doivent néanmoins, si elles existent, être associées à un utilisateur.
Il possède également plusieurs produits et pour chacun, sa quantité associée, simulant un panier (quantité qui va par ailleurs être reprise au passage d'une commande). Une commande doit, elle, posséder plusieurs produits.
Chaque produit appartient à une sous-catégorie qui fait elle-même partie d'une catégorie.

Retour au sommaire

2 — Script de création de la base de données

```
drop table DetailCommande;
drop table commande;
drop table carteBancaire;
drop table panier;
drop table produit;
drop table sousCategorie;
drop table categorie;
drop table adresse;
drop table utilisateur;

create table utilisateur (
    emailUser varchar(64),
    mdpUser varchar(256),
    adminUser number(1),
    nomUser varchar(64),
    prenomUser varchar(64),
    telUser char(10),
    COMPTEENTREPRISE NUMBER(1) CHECK (COMPTEENTREPRISE between 0 and 1),
    CONSTRAINT pk_utilisateur PRIMARY KEY (emailUser)
);

create table adresse(
    idAdresse varchar(20),
    surnomAdresse varchar(64),
    adresse varchar(128),
    complement varchar(64),
    ville varchar(45),
    codePostal char(5),
    emailUser varchar(64),
    CONSTRAINT pk_adresse PRIMARY KEY (idAdresse),
    CONSTRAINT fk_adresse_user FOREIGN KEY (emailUser) REFERENCES utilisateur(emailUser)
);

create table categorie(
    idCat VARCHAR(20),
    nomCat VARCHAR(64),
    descCat VARCHAR(256),
    CONSTRAINT pk_idCat PRIMARY KEY (idCat)
);

create table sousCategorie(
    idSousCat VARCHAR(20),
    nomSousCat VARCHAR(64),
    descSousCat VARCHAR(256),
    idCat VARCHAR(20),
    CONSTRAINT pk_sousCat PRIMARY KEY (idSousCat),
    CONSTRAINT fk_sousCat_idCat FOREIGN KEY (idCat) REFERENCES categorie(idCat)
);
create table produit(
    idProduit  VARCHAR(20),
    nomProduit VARCHAR(64),
    descProduit VARCHAR(256),
```

Retour au sommaire

```
    prixProduit DECIMAL(9,2),
    dateProduit DATE,
    stockProduit DECIMAL(9,2),
    reduction DECIMAL(9,2),
    imageProduit DECIMAL(9,2),
    idSousCat VARCHAR(20),
    CONSTRAINT pk_produit PRIMARY KEY (idProduit),
    CONSTRAINT fk_produit_sousCat FOREIGN KEY (idSousCat) REFERENCES sousCategorie(idSousCat)
);
create table panier(
    emailUser varchar(64),
    idProduit VARCHAR(20),
    quantite DECIMAL(4),
    CONSTRAINT pk_panier PRIMARY KEY (idProduit, emailUser),
    CONSTRAINT fk_panier_idProduit FOREIGN KEY (idProduit) REFERENCES produit(idProduit),
    CONSTRAINT fk_panier_emailUser FOREIGN KEY (emailUser) REFERENCES utilisateur(emailUser)
);
create table carteBancaire (
    idCB VARCHAR(20),
    numeroCB CHAR(16),
    nomCB VARCHAR(64),
    dateCB DATE,
    cryptoCB VARCHAR(4),
    emailUser varchar(64),
    CONSTRAINT pk_carteBancaire PRIMARY KEY (idCB),
    CONSTRAINT fk_carteBancaire_emailUser FOREIGN KEY (emailUser) REFERENCES utilisateur(emailUser)
);
create table commande (
    idCommande VARCHAR(20),
    emailUser varchar(64),
    idCB VARCHAR(20),
    idAdresse varchar(20),
    troisFois DECIMAL(1),
    tauxTVA DECIMAL(3),
    montant DECIMAL(11,2),
    CONSTRAINT pk_commande PRIMARY KEY (idCommande),
    CONSTRAINT fk_commande_emailUser FOREIGN KEY (emailUser) REFERENCES utilisateur(emailUser),
    CONSTRAINT fk_commande_idCB FOREIGN KEY (idCB) REFERENCES carteBancaire(idCb),
    CONSTRAINT fk_commande_idAdress FOREIGN KEY (idAdresse) REFERENCES adresse(idAdresse)
);
CREATE TABLE DetailCommande (
    idCommande VARCHAR(20),
    idProduit VARCHAR(20),
    quantite NUMBER(4),
    CONSTRAINT pk_detailcommande PRIMARY KEY (idCommande, idProduit),
    CONSTRAINT fk_detailcommande_idcomm FOREIGN KEY (idCommande) REFERENCES
Commande(idCommande),
    CONSTRAINT fk_detailcommande_idprod FOREIGN KEY (idProduit) REFERENCES Produit(idProduit)
);
```

L'ordre d'insertion des tables évite les problèmes de contraintes de clés étrangères et de clés primaires. Comme nous réinsérions souvent les tables pour faire des modifications, ces dernières doivent être supprimées avant chaque réinsertion.

Retour au sommaire

3 — Extraits du script d'insertion des données

Exemple d'insertion d'un·e utilisateur·rice :

```
INSERT INTO utilisateur (emailUser,mdpUser,adminUser,nomUser,prenomUser,telUser)
VALUES
('gravida.sagittis@outlook.ca','AFU52JUB6EJ',0,'Benton','Faith','0738755996');
```

Exemple d'insertion d'une catégorie :

```
INSERT INTO categorie
VALUES ('00001', 'Processeurs','Un processeur (ou unité centrale de calcul, UCC ;
en anglais central processing unit, CPU) est un composant présent dans de nombreux
dispositifs électroniques qui exécute les instructions machine des programmes
informatiques.');
```

Exemple d'insertion d'une sous-catégorie :

```
INSERT INTO sousCategorie
VALUES ('0100A', 'AMD','Processeurs de la marque AMD', '00001');
```

Exemple d'insertion d'un produit :

```
INSERT INTO produit
VALUES ('00001', 'AMD Ryzen 9 7950X', 'Le processeur pour PC de bureau AMD Ryzen 9
7950X a été conçu pour vous permettre de jouer dans les meilleures conditions.',
799.99, TO_DATE('01/01/2022', 'dd/mm/yyyy'), 10, 0,'','0100A');
```

Les données d'insertion ont été faites avec un générateur spécialisé dans les bases de données. Néanmoins, elles correspondent tout à fait à des cas d'usage du site web. Pour valider l'usage du site web, nous avons nous-mêmes créé des utilisateurs sur lesquels nous pouvions nous connecter.

Retour au sommaire

4 — Script d’insertion des procédures

Procédure d’ajout d’un produit au panier :

```
CREATE OR REPLACE PROCEDURE AjouterPanier
(
    p_user       Panier.emailuser%TYPE,
    p_idProd     Produit.idproduit%TYPE,
    p_quant      Panier.quantite%TYPE
)
IS
BEGIN
    INSERT INTO Panier (emailuser, idproduit, quantite)
    VALUES (p_user, p_idProd, p_quant);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Article ajouté au panier');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Cet article est déjà dans le panier');
        UPDATE Panier
        SET quantite = quantite + p_quant
        WHERE idproduit = p_idProd
        AND emailuser = p_user;
        COMMIT;
END;
/
```

Procédure du passage d’une commande :

```
CREATE OR REPLACE PROCEDURE PasserCommande(p_emailUser VARCHAR, p_idCB VARCHAR,
p_idAdr VARCHAR, p_troisFois NUMBER) IS
    email_exists NUMBER;
BEGIN
    SELECT COUNT(*) INTO email_exists
    FROM Utilisateur
    WHERE emailUser = p_emailUser;

    IF email_exists != 1 THEN
        raise_application_error(-20001,'p_emailUser ('||p_emailUser||') ne
correspond à aucun email dans la base');
    END IF;
```

Retour au sommaire

```
    IF p_troisFois NOT BETWEEN 0 AND 1 THEN
        raise_application_error(-20002,'p_troisFois ('||p_troisFois||') doit être
compris entre 0 et 1');
    END IF;

    INSERT INTO COMMANDE(IDCOMMANDE, EMAILUSER, IDCB, IDADRESSE, TROISFOIS,
TAUXTVA, MONTANT)
    VALUES (COMMANDE_SEQ.NEXTVAL, p_emailUser, p_idCB, p_idAdr, p_troisFois, 15,
0);

    FOR produit IN (SELECT * FROM Panier WHERE Panier.EMAILUSER = p_emailUser) LOOP

        INSERT INTO DETAILCOMMANDE (
            IDCOMMANDE,
            IDPRODUIT,
            QUANTITE
        )
        VALUES
        (
            COMMANDE_SEQ.CURRVAL,
            produit.idProduit,
            produit.quantite
        );

    END LOOP;

    DELETE FROM Panier
    WHERE Panier.EMAILUSER = emailUser;
END;
/
```

Ces procédures sont exécutées dans le PHP grâce au plugin oci qui met en relation le PHP et la base de données oracle, fournie par l'IUT. C'est un gain de temps et d'efficacité, les procédures stockées dans la base étant compilées. Il sera aussi plus facile de les mettre à jour, cela évitera de devoir modifier le code de la page web.

Retour au sommaire

5 — Script d’insertion des déclencheurs

Déclencheur de mise à jour du montant de la commande :

```
CREATE OR REPLACE TRIGGER t_iud_commande_maj_montant
AFTER INSERT OR DELETE OR UPDATE OF quantite
ON DetailCommande
FOR EACH ROW
DECLARE
    v_prix Produit.prixproduit%TYPE;
BEGIN
    IF DELETING OR UPDATING THEN
        SELECT prixproduit INTO v_prix
        FROM Produit P
        WHERE P.idproduit = :OLD.idproduit;
        UPDATE Commande
        SET montant = montant - (:OLD.quantite * v_prix)
        WHERE Commande.idcommande = :OLD.idcommande;
    END IF;

    IF INSERTING OR UPDATING THEN
        -- on supprime et on remet la ligne car quantite varie mais sans
suppression
        SELECT prixproduit INTO v_prix
        FROM Produit P
        WHERE P.idproduit = :NEW.idproduit;
        UPDATE Commande
        SET montant = montant + (:NEW.quantite * v_prix)
        WHERE Commande.idcommande = :NEW.idcommande;
    END IF;
END;
/
```

Déclencheur de mise à jour du stock :

```
CREATE OR REPLACE TRIGGER t_i_produit_maj_stock
AFTER INSERT
ON DetailCommande
FOR EACH ROW
BEGIN
    IF INSERTING THEN
```

Retour au sommaire

```
        UPDATE Produit
        SET stockproduit = stockproduit - :NEW.quantite
        WHERE Produit.idproduit = :NEW.idproduit;
    END IF;
END;
/
```

Déclencheur de validation des informations de la commande et de son·sa client·e:

```
CREATE OR REPLACE TRIGGER t_i_commande_check
BEFORE INSERT
ON Commande
FOR EACH ROW
DECLARE
    cb_appartient NUMBER;
    adr_appartient NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO cb_appartient
    FROM CarteBancaire CB
    WHERE CB.idCB = :NEW.IDCB
    AND CB.emailUser = :NEW.emailUser;

    IF cb_appartient != 1 THEN
        raise_application_error(-20003,'La carte '||:NEW.idCB||' n''existe pas ou
n''appartient pas à l''utilisateur');
    END IF;

    SELECT COUNT(*)
    INTO adr_appartient
    FROM Adresse ADR
    WHERE ADR.idAdresse = :NEW.idAdresse
    AND ADR.emailUser = :NEW.emailUser;

    IF ADR_appartient != 1 THEN
        raise_application_error(-20004,'L''adresse '||:NEW.idAdresse||' n''existe
pas ou n''appartient pas à l''utilisateur');
    END IF;
END;
/
```

Retour au sommaire

Ces déclencheur sont stockés et exécutés dans la base de données oracle, fournie par l'IUT. C'est un gain de temps et d'efficacité, les déclencheurs stockés mettant à jour les données automatiquement. Il sera aussi plus facile de les modifier, cela évitera de devoir changer le code de la page web.

Retour au sommaire