INF 4163 - PROJET 1 TECHNIQUES DE BASE DE DONNÉES

Concevoir une base de données relationnelle pour optimiser la gestion des stocks et des transactions commerciales

Durée : du 15-11-2024 au 30-11-2024

superviseur : Etienne Tajeuna

par l'équipe 3

Anis Davidson ANID69360004

Gabriel-Atangana Mboa Bryan GABB78300209

Tresor Megane Tambat TAMT79360604

Tables des matières

INTRODUCTION

Dans le cadre du premier projet sql de la session d'automne 2024, un jeu de données portant sur la gestion des transaction commerciales et du stock d'un entrepôt à été mis à notre à disposition. Ce jeu de données contenait des informations sur les transactions réalisées par les clients, les produits achetés, le lieu de l'achat, ainsi que les méthodes de paiements utilisées. Notre objectif principal était donc de concevoir une base de données relationnelle permettant de gérer ces informations de manière efficace.

Pour y parvenir, nous avons d'abord mis en place un modèle conceptuel et un schéma relationnel en utilisant la méthode UML, puis nous avons créé la base de données en utilisant SQL dans le système de gestion de base de données MySQL.

Ce document présente les différentes étapes réalisées dans le cadre ce projet. Il détaille l'analyse du jeu de données fournies ainsi que toutes les étapes que nous avons suivies pour concevoir une base de données relationnelle permettant d'optimiser la gestion des stocks, l'analyse des achats, et la prise de décisions commerciales au sein l'entreprise du client.

I. PROBLÉMATIQUE ET HYPOTHÈSES

Afin de mener à bien notre mission, nous avons jugé bon d'établir une problématique principale, ainsi que plusieurs problématiques secondaires.

A. Problématique principale

Comment concevoir une base de données relationnelle permettant une gestion optimale des transactions commerciales et du stock de l'entrepôt de l'entreprise de notre client, afin d'optimiser les processus d'achat, de vente et de gestion des stocks, et ainsi faciliter la prise de décision ?

B. Problématiques secondaires

1. **Gestion des transactions commerciales:**

Comment structurer les informations liées aux translations de manière à garantir la fiabilité des données ?

2. **Gestion des stocks:**

Comment assurer un suivi efficace des stock afin d'être certain de la quantité de stock restante, pour éviter toute rupture de stock ainsi le surstockage ?

3. **Analyse des données:**

Comment concevoir une base de données qui facilite l'analyse du comportement d'achats des clients et des tendances de consommation pour permettre des décisions stratégique basées sur les données.

4. **Optimisation des processus commerciaux:**

Comment utiliser l'automatisation des opérations commerciales pour gagner en temps et éviter les risques d'erreur ?

C. Hypothèses

1. Si les données liées aux transactions sont correctement structurées et normalisées, alors la fiabilité des informations sera meilleure, ce qui permettra une meilleure traçabilité et un meilleur suivi des ventes.
2. Si un suivi en temps réel des niveaux de stock est mis en place dans la base de données, alors les risques de rupture de stock et de surstockage pourront être réduits, garantissant ainsi une gestion plus efficace.
3. Si les relations dans la base de données permettent une analyse détaillée des comportements des clients, alors de meilleures décisions stratégiques pourront être prises.
4. Si les opérations récurrentes sont automatisées, alors le temps consacré à la gestion manuel des transactions sera réduit, ce qui diminue les risques d'erreurs potentielles.

La rédaction de la problématique et des hypothèses nous a permis de mieux comprendre ce que nous devions faire pour répondre à la problématique principale, ainsi que de détailler une liste de tâches à partir de cette problématique et des attendues du projet.

II. GESTION DE PROJET

Pour accomplir toutes les tâches liées à ce projet, nous avons établi une liste de toutes les tâches avec des codes, afin de faciliter la traçabilité de chaque tâche.

A. Répartition des tâches

Code	Tâche
T1	**Table "toutes transactions"**
T1.1	Création de la base de données
T1.2	Importation des données dans une table appelées "Toutes transactions"
T1.3	Requête : Combien y-a-t-il de produits distincts dans ce jeu de données ?
T1.4	Requête : Quels sont les différents lieux de vente ?
T1.5	Requête : Quels sont les différents modes de paiement ?

T1.6	Requête : Quelles sont les différentes catégories de clients et combien de clients avons nous de clients par catégorie ?
T2	**Analyse du problème**
T2.1	Description détaillée du problème
T2.2	Énumération des entitées qui pourront aider à la gestion
T3	**Construction du MCD**
T3.1	Construction du modèle entité relation (MERISE, UML, Chen)
T3.2	Documentation du modèle
T4	**Implémentation de la base de données (MySQL)**
T4.1	Création des tables
T4.2	Insertion des données dans les tables identifiées
T4.3	Requêtes SQL
T5	**Gestion des stocks**
T5.1	Création des transactions
T5.2	Simulation des 5 premières transactions

Tableau 1: Liste des tâches

B. Outils de gestion de projet

Pour mener à bien ce projet et pour assurer une bonne collaboration entre les membres de l'équipe nous avons utilisé les outils suivants:

- **Visual Studio Code** comme interface de développement pour l'écriture du code et l'organisation des fichiers;
- **MySQL server** comme système de gestion de base de données relationnelle;
- **WhatsApp** pour la communication entre les membres de l'équipe;
- **Github** pour le versionnement du code;

- **Google Docs** pour la rédaction du rapport;
- **Draw.io** et **Figma** pour la création de la représentation visuel du schéma.

C. Organisation du travaille

Pour garantir une fluidité dans l'avancement du projet, nous avons fonctionné ainsi :

Organisation de réunions hebdomadaires: Des réunions hebdomadaires ont été organisées pour faire le point sur l'avancement du projet et discuter des problèmes rencontrés.

Division des tâches: Le travail a été réparti entre les membres de l'équipe en fonction de leurs préférences.

Revues d'avancement: Nous avons utilisé le service github pour voir l'avancement des tâches liées au code SQL ainsi que pour donner des feedbacks.

La gestion de projet s'est avérée efficace car elle nous a permis d'effectuer le travail demandé et d'obtenir des résultats concrets.

III. RÉSULTATS

A. T1 : Toute transactions

Cette tâche consistait d'abord en l'importation des données du dataset *"Retail Transactions Dataset"* dans une table nommée *"toutes_transactions"*, dans le but d'effectuer certaines requêtes demandées dans l'énoncé du projet. Le dataset "Retail Transactions Dataset" a été créé pour simuler un ensemble de données de panier d'achat. Il contient des informations détaillées sur les transactions, les clients, les produits, et le comportement d'achat. Ce dataset comprend un million d'entrées et 13 colonnes décrites comme suit :

- **Transaction_ID :** Identifiant unique pour chaque transaction, représenté par un entier à 10 chiffres. Cette colonne permet d'identifier chaque achat de manière unique;
- **Date :** Date et heure auxquelles la transaction a eu lieu;
- **Customer_Name :** Nom du client ayant effectué l'achat. Cette colonne fournit des informations sur l'identité du client;

- **Product :** Liste des produits achetés lors de la transaction. Elle inclut les noms des produits achetés;
- **Total_Items :** Nombre total d'articles achetés dans la transaction. Elle représente la quantité de produits achetés;
- **Total_Cost :** Coût total de l'achat. Elle représente le coût de la transaction.
- **Payment_Method :** Méthode de paiement utilisée pour la transaction, ça peut être : une carte de crédit, une carte de débit, en espèces ou un paiement mobile;
- **City :** Ville où l'achat a été effectué. Cette colonne indique la localisation de la transaction;
- **Store_Type :** Type de magasin où l'achat a été réalisé, tel qu'un supermarché, une supérette, un grand magasin, etc;
- **Discount_Applied :** Indicateur binaire (vrai/faux) indiquant si un rabais a été appliqué à la transaction;
- **Customer_Category :** Catégorie représentant le profil du client, comme son âge ou son groupe démographique;
- **Season :** Saison au cours de laquelle l'achat a été effectué, comme le printemps, l'été, l'automne ou l'hiver;
- **Promotion :** Type de promotion appliquée à la transaction, cette colonne peut prendre les valeurs : “None” (aucune), "BOGO (Buy One Get One)" (un acheté, un offert), ou “Discount on Selected Items” (réduction sur certains articles).

Pour cette tâche, nous avons tout d'abord créé une base de données nommée *"uqo_projet_sql_1"*, ensuite nous avons créé la table *"toutes_transactions"*, et finalement nous avons importé les données du dataset qui était en format *.csv* dans notre base de données. À cette étape, nous avons rencontré un défi lors de l'importation des données, car le problème résidait dans la taille du dataset, qui comptait un million d'entrées. Pour contourner ce problème, MySQL Server devait être installé directement sur la machine, et les données devaient ensuite être importées localement.

Après avoir importé les données le requêtes suivantes demandées ont été réaliser et nous avons obtenues les résultats suivants :

Requête	Résultat
Combien y-a-t-il de produits distincts dans	81

ce jeu de données ?		
Quels sont les différents lieux de vente ?	10 villes : Los Angeles, San Francisco, Houston, Chicago, Boston, New York, Seattle, Miami, Dallas, Atlanta.	
Quels sont les différents modes de paiement ?	4 option de paiement : Mobile Payment, Cash, Credit Card, Debit Card	
Quelles sont les différentes catégories de clients et combien de clients avons nous de clients par catégorie ?	Catégorie de client	Nombre total (personnes)
	Homemaker	125418
	Professional	124651
	Young Adult	124577
	Retiree	125072
	Student	124842
	Middle-Aged	124636
	Senior Citizen	125485
	Teenager	125319

Tableau 2: Requêtes réalisées et résultats obtenues sur la table 'toutes transactions

Après avoir accompli ce qui était attendu de nous sur la table contenant toutes les transactions, nous nous sommes lancés à la recherche d'une solution potentielle à notre problématique en commençant par l'analyse du problème.

B. T2 : Analyse du problème

Pour rappel, notre objectif est de concevoir une base de données relationnelle pour optimiser la gestion des stocks et des transactions commerciales partant d'un dataset contenant l'information sur les transactions effectuées. Pour ce faire, il y a plusieurs entités qui peuvent être ajoutées afin de structurer les données. Le tableau ci-dessous présente l'entité existante ainsi que celles que nous avons jugées pertinentes à ajouter pour mieux structurer les données.

Entité existante	Entités créer
Toutes transactions	Transaction Client Produit Magasin Promotion

Tableau 3: Liste des tâches

Nous justifierons le choix des entités sélectionnées à l'aide de notre MCD (Modèle Conceptuel de Données) et notre MLD (Modèle Logique des Données).

C. T3 : MCD et MLD

1. Présentation des modèles MCD et MLD

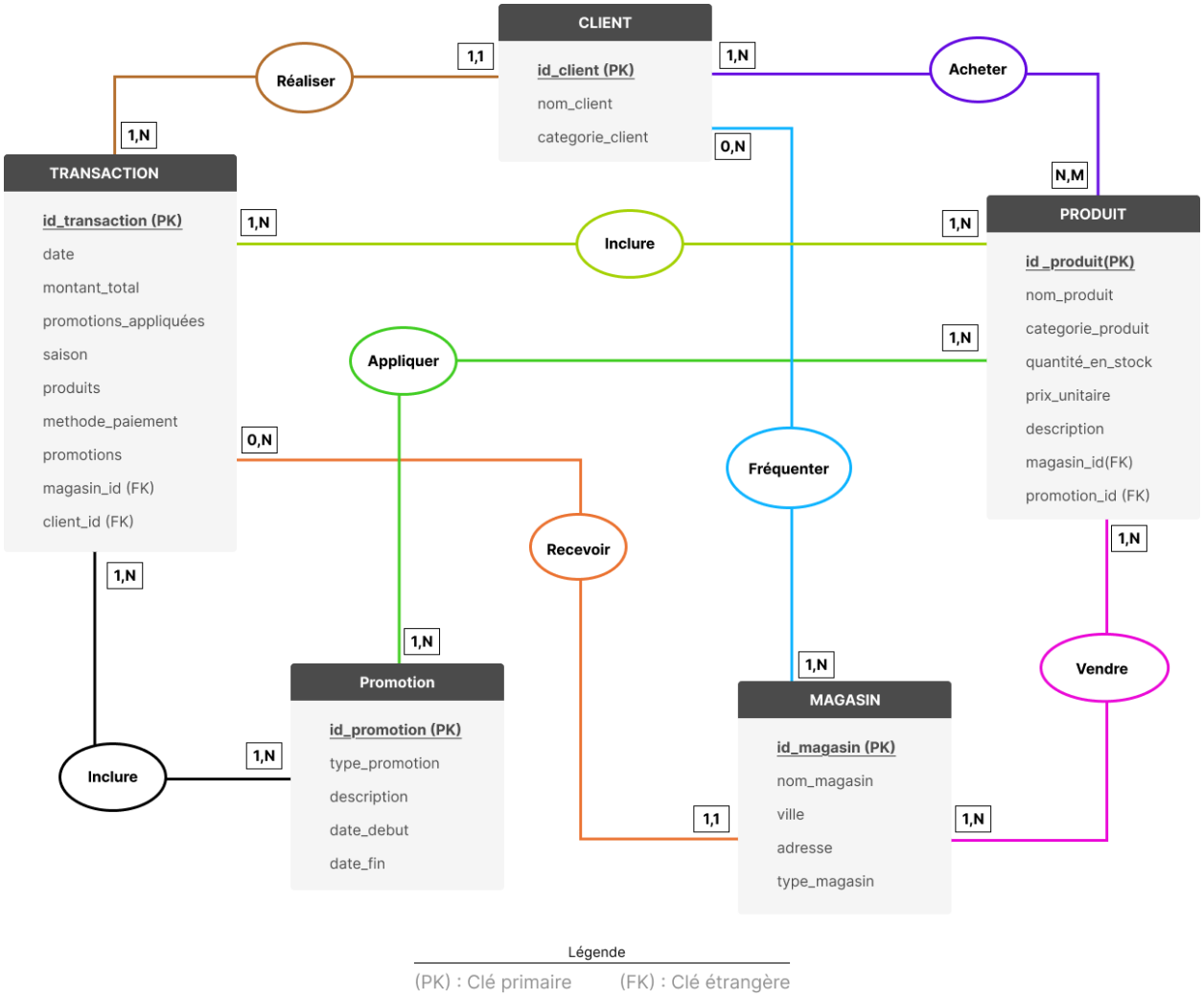

Figure 1 : Modèle Conceptuel de Données (MCD) pour la gestion des transactions

Après avoir fait le MCD, nous avons trouvé pertinent par la suite de faire un MLD, pour montrer plus de détails sur nos entités ainsi que nos attributs.

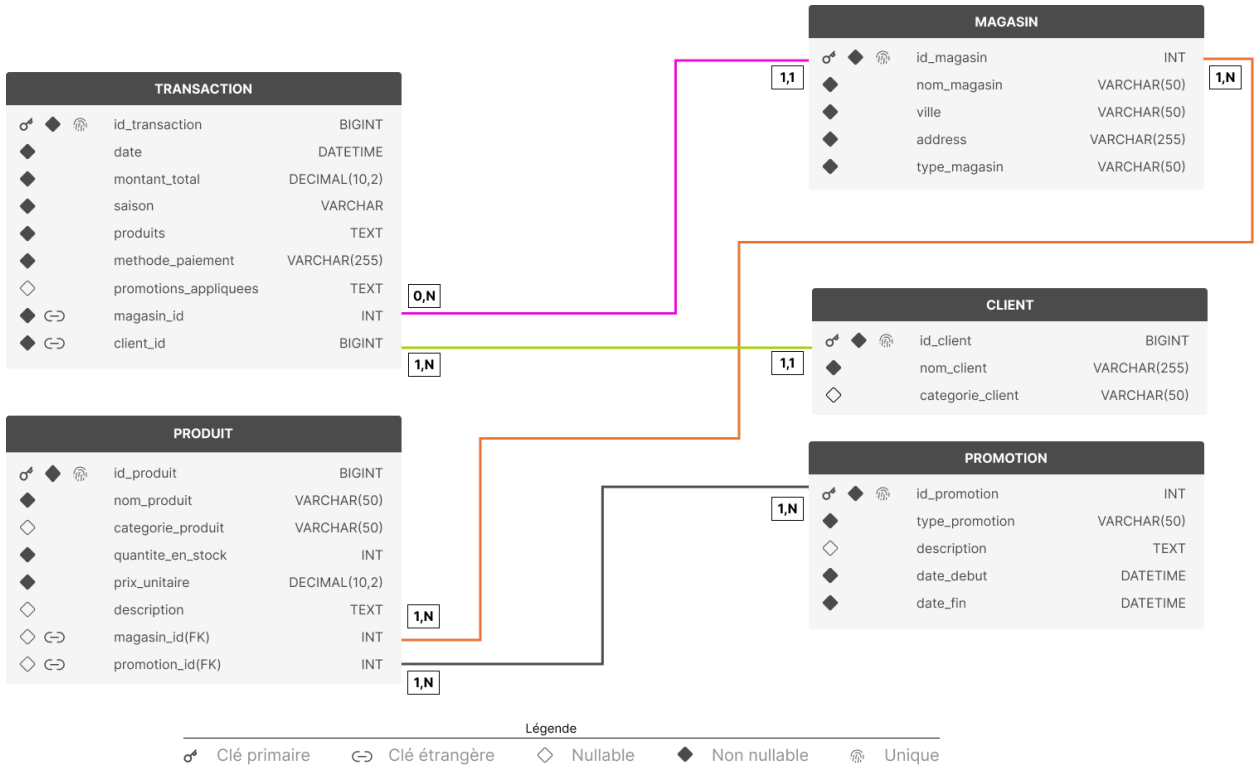

Figure 2 : Modèle Logique de Données (MLD) pour la gestion des transactions

2. Description des entités

Pour la documentation de notre MCD nous avons rédigé une description détaillée de chaque entité impliquée dans la gestion de données.

CLIENT

Cette entité représente les clients qui effectuent des achats dans les magasins.

Attributs :

- **id_client :** Identifiant unique du client (clé primaire) – Type : Entier, Contrainte : Obligatoire, unique.
- **nom :** Nom du client – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **catégorie_client :** Catégorie du client – Type : Chaîne de caractères, Contrainte : Facultatif.

PRODUIT

Cette entité représente les produits disponibles dans les magasins.

Attributs :

- **id_produit :** Identifiant unique du produit (clé primaire) – Type : Entier, Contrainte : Obligatoire, unique.
- **nom_produit :** Nom du produit – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **catégorie_produit :** Catégorie du produit – Type : Chaîne de caractères, Contrainte : Facultatif.
- **prix_unitaire :** Prix de vente du produit – Type : Décimal, Contrainte : Obligatoire.
- **quantité_en_stock :** Quantité disponible en stock – Type : Entier, Contrainte : Obligatoire.
- **description :** Description détaillée du produit – Type : Texte, Contrainte : Facultatif.
- **promotion_id :** Promotion appliquée au produit – Type : Entier, Contrainte : Clé étrangère
- **magasin_id :** Magasin dans lequel le produit est vendu - Type : Entier, Contrainte: Clé étrangère

MAGASIN

Cette entité représente les différents magasins.

Attributs :

- **id_magasin :** Identifiant unique du magasin (clé primaire) – Type : Entier, Contrainte : Obligatoire, unique.
- **nom_magasin :** Nom du magasin – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **adresse :** Adresse du magasin – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **Ville :** Ville dans laquelle est situé le magasin – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **type_magasin :** Type de magasin – Type : Chaîne de caractères, Contrainte : Facultatif.

TRANSACTION

Cette entité représente une transaction effectuée par un client dans un magasin.

Attributs :

- **id_transaction :** Identifiant unique de la transaction (clé primaire) – Type : Entier, Contrainte : Obligatoire, unique.
- **date_transaction :** Date et heure de la transaction – Type : Date/Heure, Contrainte : Obligatoire.
- **montant_total :** Montant total de la transaction – Type : Décimal, Contrainte : Obligatoire.
- **mode_paiement :** Mode de paiement utilisé – Type : Chaîne de caractères, Contrainte : Obligatoire.
- **promotions_appliqués :** Toutes les promotions appliquées à cette transaction – Type : Texte, Contrainte : Facultatif.

- **id_client :** Lien vers le client ayant effectué la transaction (clé étrangère) – Type : Entier, Contrainte : Obligatoire.
- **id_magasin :** Lien vers le magasin où la transaction a eu lieu (clé étrangère) – Type : Entier, Contrainte : Obligatoire.

PROMOTION

Cette entité représente les promotions qui peuvent être appliquées aux transactions ou produits.

Attributs :

- id_promotion : Identifiant unique de la promotion (clé primaire) – Type : Entier, Contrainte : Obligatoire, unique.
- type_promotion : Type de la promotion – Type : Chaîne de caractères, Contrainte : Obligatoire.
- description : Description détaillée de la promotion – Type : Texte, Contrainte : Facultatif.
- date_début : Date de début de la promotion – Type : Date, Contrainte : Obligatoire.
- date_fin : Date de fin de la promotion – Type : Date, Contrainte : Obligatoire.

Dans la suite de la documentation de notre MCD, nous avons défini les relations entre les différentes entités de notre base de données.

3. Description des relations

Les relations qui alimentent les interactions entre les entités de notre base de données sont énumérées ci-dessous.

CLIENT (1,1) ⟷ (1,N) TRANSACTION : un client peut effectuer plusieurs transactions, mais une transaction est associée à un seul client.

CLIENT (1,N) ⟷ (N,M) PRODUIT : un client peut acheter plusieurs produits, et un produit peut être acheté par plusieurs clients et par un même client plusieurs fois.

CLIENT (0,N) ⟷ (1,N) MAGASIN : un client peut fréquenter plusieurs magasins, et un magasin peut avoir plusieurs clients.

PRODUIT (0,N) ⟷ (1,N) MAGASIN : un magasin peut vendre plusieurs produits, et un produit peut être vendu dans plusieurs magasins.

PRODUIT (1,N) ⟷ (1,N) PROMOTION : un produit peut bénéficier de plusieurs promotions, et une promotion peut s'appliquer à plusieurs produits.

TRANSACTION (0,N) ⟷ (1,1) MAGASIN : une transaction a lieu dans un seul magasin, mais un magasin peut enregistrer aucune ou plusieurs transactions.

TRANSACTION (1,N) ⟷ (1,N) PRODUIT : une transaction peut inclure plusieurs produits, et un produit peut être inclus dans plusieurs transactions.

TRANSACTION (1,N) ⟷ (1,N) PROMOTION : une transaction peut bénéficier de plusieurs promotions, et une promotion peut être appliquée à plusieurs transactions.

Après avoir documenté le MCD, nous avons procédé à la création des tables et à l'insertion des données dans les différentes entités créées.

4. Implémentation MySQL

4.1 - Création des tables

Nous nous sommes basés sur les modèles MCD et MLD pour créer les tables.

Nom	Code SQL
Transaction	CREATE TABLE transaction (id_transaction BIGINT PRIMARY KEY AUTO_INCREMENT NOT NULL, id_magasin INT NOT NULL, id_client BIGINT NOT NULL, date_transaction DATETIME NOT NULL, montant_total DECIMAL(10, 2) NOT NULL, saison VARCHAR(50) NOT NULL, produits TEXT NOT NULL, methode_paiement VARCHAR(255) NOT NULL, promotion_appliquee TEXT, FOREIGN KEY (id_magasin) REFERENCES magasin(id_magasin) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (id_client) REFERENCES client(id_client) ON DELETE CASCADE ON UPDATE CASCADE);
Client	CREATE TABLE client (id_client BIGINT PRIMARY KEY AUTO_INCREMENT NOT NULL, nom_client VARCHAR(255) NOT NULL, categorie_client VARCHAR(50));

Produit	CREATE TABLE produit (id_produit INT PRIMARY KEY AUTO_INCREMENT NOT NULL, nom_produit VARCHAR(50) NOT NULL, category_produit VARCHAR(50), quantite_en_stock INT NOT NULL, prix_unitaire DECIMAL(10, 2) NOT NULL, description_produit TEXT, id_promotion INT, FOREIGN KEY (id_promotion) REFERENCES Promotion(id_promotion) ON DELETE SET NULL ON UPDATE CASCADE);
Promotion	CREATE TABLE promotion (id_promotion INT PRIMARY KEY AUTO_INCREMENT NOT NULL, type_promotion VARCHAR(50) NOT NULL, description_promotion TEXT, date_debut DATETIME NOT NULL, date_fin DATETIME NOT NULL);
Magasin	CREATE TABLE magasin (id_magasin INT PRIMARY KEY AUTO_INCREMENT NOT NULL, nom_magasin VARCHAR(50) NOT NULL, ville VARCHAR(255) NOT NULL, adresse VARCHAR(255) NOT NULL, type_magasin VARCHAR(50) NOT NULL);

Tableau 4: Code pour la création des tables

4.2 - Insertion des données

Pour l'insertion des données, nous avons utilisé la bibliothèque "pymysql" de Python afin d'extraire une partie du dataset contenant toutes les transactions et de l'insérer dans les tables que nous avons créées. Des valeurs aléatoires ont été insérées dans les champs obligatoires tandis que les champs facultatifs ont été laissés NULL. Nous avons opté pour cette méthode car plusieurs attributs qui n'étaient pas présents dans la table originale ont été ajoutés pour optimiser la structure des données. Cette optimisation vise à améliorer la gestion des stocks et des transactions.

4.3 - Requêtes pour la gestion des transactions

Pour les requêtes, nous avons choisi 10 requêtes qui nous paraissaient pertinentes dans la gestion des transactions.

Requête	Code SQL
Lister toutes les transactions dans une ville spécifique.	select * from transaction t join magasin m on t.id_magasin = m.id_magasin where ville = 'Houston';
Obtenir les descriptions de toutes les promotions entre deux dates.	SELECT * FROM promotion WHERE date_debut <= '2024-01-29' AND date_fin >= '2024-11-20';
Lister tous les produits dont le stock est en dessous d'un seuil donné.	SELECT * FROM produit WHERE quantite_en_stock < 10;
Calculer le revenu total généré par chaque magasin.	SELECT m.nom_magasin, SUM(t.montant_total) AS revenu_total FROM magasin m INNER JOIN transaction t ON m.id_magasin = t.id_magasin GROUP BY m.nom_magasin;
Trouver la valeur moyenne des transactions pour chaque méthode de paiement.	select t.methode_paiement, avg(t.montant_total) as valeur_moyenne from transaction t group by t.methode_paiement;
Identifier les clients ayant dépensé le plus.	SELECT c.nom_client, SUM(t.montant_total) AS total_depense FROM client c INNER JOIN transaction t ON c.id_client = t.id_client GROUP BY c.nom_client ORDER BY total_depense DESC LIMIT 10;
Déterminer les magasins les plus performants dans chaque ville en termes de revenus.	with revenu_magasin as(select m.ville, m.nom_magasin, sum(t.montant_total) as revenu_total from transaction t join magasin m on t.id_magasin = m.id_magasin group by m.ville, m.nom_magasin) select ville, nom_magasin, revenu_total

	from revenu_magasin r where revenu_total= (select max(revenu_total) from revenu_magasin where ville = r.ville);
Identifier l'impact des promotions sur les ventes.	select p.type_promotion, count(t.id_transaction) as nb_ventes, sum(t.montant_total) as revenu_total from transaction t left join promotion p on t.promotion_appliquee = p.id_promotion group by p.type_promotion;
Analyser les tendances saisonnières en comparant les ventes entre les saisons.	select saison, sum(montant_total) as revenu_total from transaction group by saison order by revenu_total desc;
Analyser l'efficacité des promotions en calculant les dépenses moyennes par type de promotion.	select p.type_promotion, avg(t.montant_total)as depense_moyenne from transaction t join promotion p on t.promotion_appliquee = p.id_promotion group by p.type_promotion;

Tableau 4 : Requêtes SQL pour la gestion des transaction

5. Gestion des stocks

5.1 - Création des transactions pour l'automatisation des tâches

Nous avons réalisé des transactions pour aider dans l'automatisation de certaines tâches visant à optimiser la gestion des stocks.

Transaction	Code MySQL	Coût
Création d'un client qui effectue sa première commande	START TRANSACTION; -- Insertion du nouveau client INSERT INTO client (nom_client, categorie_client) VALUES ('Jean Mathieu de la franchise', 'Nouveau'); SET @new_client_id = LAST_INSERT_ID(); -- Création de la nouvelle transaction INSERT INTO transaction (id_magasin,	35ms

	id_client, date_transaction, montant_total, saison, produits, methode_paiement) VALUES (1, @new_client_id, NOW(), 250.50, 'Printemps', '1,2,3', 'Carte Bancaire'); -- Mis à jour du stock des produits UPDATE produit SET quantite_en_stock = quantite_en_stock - 1 WHERE id_produit IN (1, 2, 3); COMMIT; SELECT 'Création client réussie' AS result;	
Création d'un nouveau magasin	START TRANSACTION; -- Création du nouveau magasin INSERT INTO magasin (nom_magasin, ville, adresse, type_magasin) VALUES ('George Weya Centre-Ville', 'Gatineau', '12 Rue Principale', 'Luxe'); -- Récupération de l'ID du nouveau magasin SET @new_store_id = LAST_INSERT_ID(); COMMIT;	1ms
Ajout d'un nouveau produit ou augmentation de stock d'un produit existant	-- Transaction pour ajouter des produits DELIMITER // CREATE PROCEDURE AjoutProduitEtUpdateStock(IN new_produit_nom VARCHAR(50), IN new_produit_categorie VARCHAR(50), IN new_produit_qte INT, IN new_produit_prix DECIMAL(10,2),	4ms

```
    IN new_produit_description TEXT,
    IN id_produit_existant INT,
    IN update_stock INT
)
BEGIN
    START TRANSACTION;

    INSERT INTO produit (
        nom_produit,
        category_produit,
        quantite_en_stock,
        prix_unitaire,
        description_produit
    ) VALUES (
        new_produit_nom,
        new_produit_categorie,
        new_produit_qte,
        new_produit_prix,
        new_produit_description
    );

    IF id_produit_existant IS NOT NULL THEN
        UPDATE produit
        SET quantite_en_stock = quantite_en_stock +
update_stock
        WHERE id_produit = id_produit_existant;
    END IF;

    COMMIT;
END //

DELIMITER ;

CALL AjoutProduitEtUpdateStock('Écouteurs sans fil',
'Audio', 50, 79.99, 'Écouteurs Bluetooth avec réduction
de bruit', 123, 50);
```

Tableau 5 : Transaction SQL pour l'automatisation dans la gestion des stocks

CONCLUSION

Face à la problématique de concevoir une base de données relationnelle pour optimiser la gestion du stock et des transactions commerciales de notre client, nous avons mené une analyse des besoins spécifiques de l'entreprise. Après avoir étudié les besoins, nous avons élaboré une structure de base de données répondant aux exigences :

- Les informations relatives aux transactions ont été structurées de manière à garantir leur intégrité et leur cohérence;
- Le système de gestion des stocks peut être utilisé pour prévenir les ruptures de stock et les surstocks;
- L'implémentation de transactions automatisées contribue au gain en productivité et à la réduction des risques d'erreurs manuelles.

Grâce à cette solution, notre client bénéficie désormais d'un système de gestion d'opérations lui permettant d'optimiser la gestion des transactions ainsi que la gestion des stocks.

En conclusion, ce projet a été l'occasion de mettre en pratique nos connaissances en conception de bases de données. En partant d'un cas concret, nous avons pu démontrer comment une base de données bien structurée peut transformer des données peu structurées en informations précieuses pour la prise de décision, soulignant ainsi l'importance de cette compétence dans le monde professionnel.

ANNEXES

Patil, Prasad. “Retail Transactions Dataset.” *Kaggle*, 18 May 2024, www.kaggle.com/datasets/prasad22/retail-transactions-dataset/data

Etienne, Tajeuna. INF4163 - “Techniques de bases de données. Projet P1 : Gestion des achats et du stock.” Automne 2024, https://moodle.uqo.ca/pluginfile.php/1684463/mod_resource/content/1/INF4163-Projet-P1.pdf

Code source : https://github.com/ticorail/inf4163-p1-equipe3

Transaction 1

id_client bigint	nom_client varchar(255)	categorie_client varchar(50)
95289	Cheryl Nelson	Young Adult
95290	Lisa Nunez	Teenager
95291	Julie Calderon	Homemaker
95292	Kerry Edwards	Retiree
95293	Derek Perry	Retiree
95294	Jean Mathieu dela franc	Nouveau

Transaction 2

id_magasin int	nom_magasin varchar(50)	ville varchar(255)	adresse varchar(255)	type_magasin varchar(50)
Filter	Filter	Filter	Filter	Filter
101	George Weya Centre-V	Gatineau	12 Rue Principale	Luxe

Transaction 3

81	Rice	(NULL)	46	285.62	(NULL)	(NULL)
82	Écouteurs sans fil	Audio	50	79.99	Écouteurs Bluetooth avec rédu	(NULL)