Rapport de Conception et d'Exploitation d'une Base de Données E-Commerce

Auteurs : Aya LAHOUEL, Mohamed SAKHO

Introduction

Ce rapport présente la démarche complète de conception, de peuplement et d'interrogation d'une base de données relationnelle pour "Impact", une entreprise de vente en ligne. Le projet, mené en suivant la méthodologie MERISE, a abouti à la création d'un système d'information robuste et normalisé, prêt à répondre à des scénarios métier complexes.

Ce rapport s'articule autour de trois axes majeurs :

I. **Justification des choix de modélisation**

II. **Scénario d'utilisation :** Une démonstration concrète de l'exploitation de la base de données à travers un scénario d'analyse marketing.

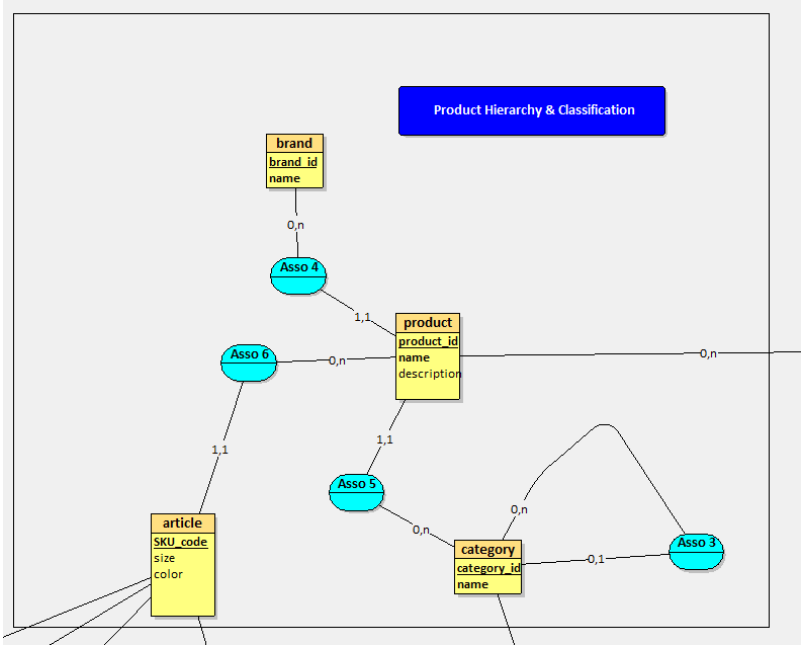
III. **Bilan et perspectives**

I. Démarche de Modélisation et Justifications Architecturales

L'objectif principal de la conception était de créer un modèle de données qui élimine la redondance et prévient les anomalies de mise à jour, tout en restant flexible et performant. Pour cela, nous avons structuré notre modèle en trois unités fonctionnelles interdépendantes.

1. Classification et Hiérarchie des Produits

Ce bloc constitue le cœur du catalogue de l'entreprise. Il définit ce qui est vendu et comment cela est présenté au client.



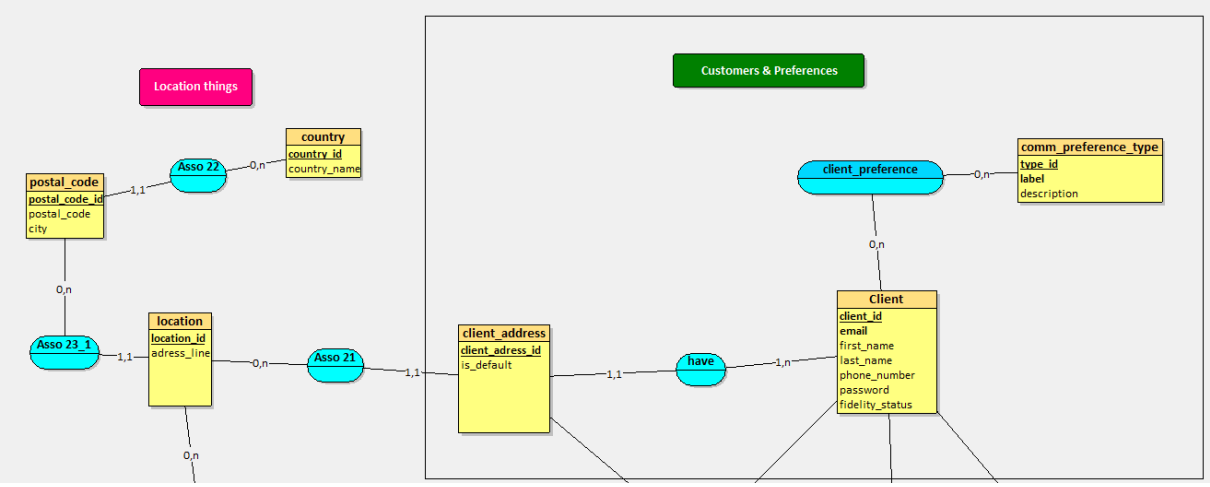
Choix Architecturaux et Justifications :

- **Séparation Product / Article (SKU) :**
  - **Problématique :** Un produit conceptuel (ex : "Veste Urban Explorer") existe en plusieurs variations physiques (taille M/couleur Noire, taille L/couleur Olive). Avoir ces deux concepts dans une seule table entraînerait une répétition massive des informations marketing (nom, description, marque), violant la **2ème Forme Normale (2FN)** car ces données ne dépendent que de l'ID du produit, et non de la variation.
  - **Solution :** Nous avons créé une entité Product pour les informations générales et une entité Article pour chaque variation unique.
  - **Bénéfice :** Cette structure garantie la 3FN. Elle permet une gestion fine des stocks et des prix au niveau de l'Article (l'unité transactionnelle) tout en centralisant l'information marketing au niveau du Product.

- **Hiérarchie des Catégories (Category) :**
    - **Problématique :** Les produits sont organisés dans une arborescence (ex: Vêtements > Manteaux > Vestes). Une structure figée avec des colonnes categorie_1, categorie_2 serait une violation de la **1ère Forme Normale (1FN)**.
    - **Solution :** Nous avons opté pour une **association récursive** : une catégorie peut avoir un category_parent_id qui référence une autre catégorie qui serait l'id de son parent.
    - **Bénéfice :** Cette solution offre une flexibilité totale pour la profondeur de la hiérarchie et simplifie les requêtes de navigation.

2. Gestion des Clients, Adresses et Préférences

Ce bloc gère toutes les informations relatives aux utilisateurs du service.



Choix Architecturaux et Justifications :

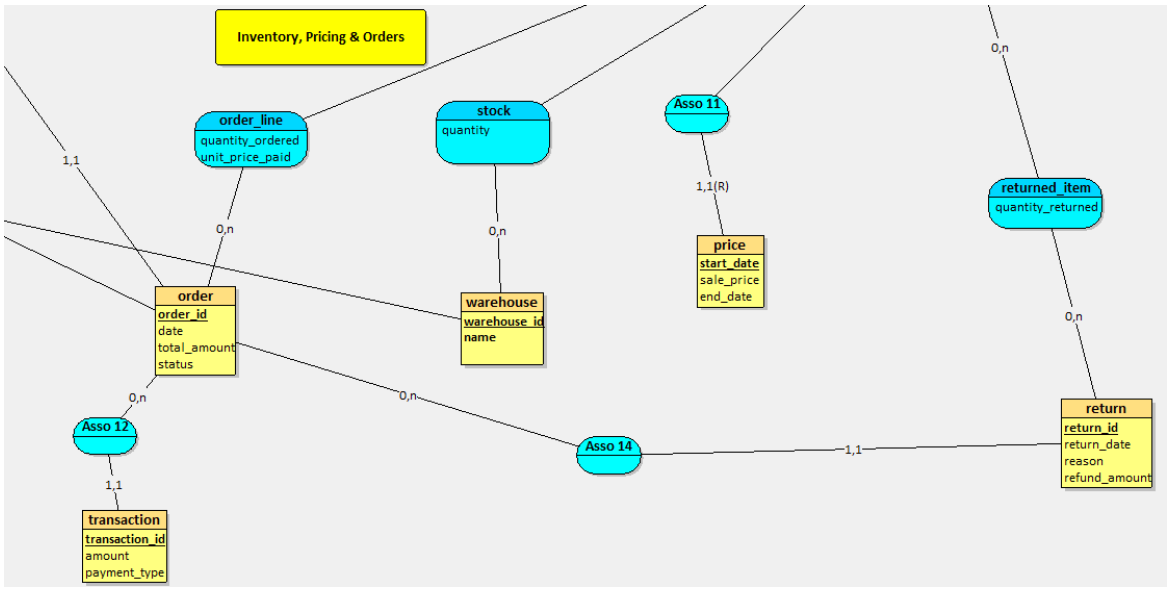
- **Normalisation Poussée des Données Géographiques :**
    - **Problématique :** Un client peut avoir plusieurs adresses. Les stocker dans la table Client violerait la **1FN**. De plus, dans une même table d'adresse, la ville dépend du code postal, qui lui-même dépend du pays. Ces dépendances transitives sont une violation de la **3FN**.
    - **Solution :** Nous avons mis en place un modèle en cascade :

      Client -> Client_Address -> Location -> Postal_Code -> Country
    - **Bénéfice :** Ce modèle garantit une intégrité absolue des données géographiques, élimine toute redondance et optimise les performances pour les analyses de ventes par région.
- **Gestion des Préférences (Client_Preference) :**
    - **Justification :** La table Client_Preference est une table associative qui modélise une relation plusieurs-à-plusieurs entre Client et Comm_Preference_Type. Cela

permet à un client de souscrire à plusieurs types de communication (Email, SMS, etc.) sans avoir à ajouter des colonnes booléennes à la table Client.

3. Cycle de Vie Transactionnel

Ce bloc est le plus complexe car il gère toutes les opérations dynamiques de l'entreprise : ventes, paiements, retours, et service client.



Choix Architecturaux et Justifications :

- **Utilisation d'Entités Faibles pour l'Intégrité :**
  - **Concept :** Une entité faible ne peut exister sans son entité forte parente. Sa clé primaire est composée, incluant celle de son parent.
  - **Application** :
    - Order_Line (ligne de commande) est une entité faible de Order_. Il est structurellement impossible d'avoir une ligne de commande "orpheline", garantissant la cohérence de chaque commande.
    - Price est une entité faible de Article, permettant d'historiser les prix.
    - Ticket_History est une entité faible de Support_Ticket, assurant l'intégrité des conversations du service client.

- **Immutabilité Financière dans Order_Line :**
  - **Problématique :** Le prix d'un article peut changer. Si une commande se contentait de référencer le prix actuel, son montant total pourrait varier rétrospectivement, ce qui est inacceptable.
  - **Solution :** La table Order_Line possède un champ unit_price_paid qui "gèle" le prix de l'article au moment exact de la transaction.

- **Bénéfice :** Cette copie garantit l'immutabilité de l'historique financier. Le montant facturé est préservé, ce qui est crucial pour la comptabilité, les retours et la résolution de litiges.

- **Séparation Commande / Transaction (Order_ vs Transaction) :**
  - **Justification :** L'acte commercial (la commande) est distinct de l'acte financier (la transaction). Cette séparation permet de gérer des scénarios complexes comme un paiement échoué pour une commande qui existe toujours, ou un paiement en plusieurs fois.

II. Scénarios d'Utilisations

Pour démontrer la **PUISSANCE** et la flexibilité de notre base de données, nous avons simulé trois scénarios métier concrets. Les requêtes SQL correspondantes se trouvent dans le fichier 4_interrogation.sql.

Scénario 1 : Analyse Stratégique (Rôle : Analyste Marketing)

Objectif : Préparer la campagne promotionnelle du prochain trimestre.

- **Analyse :** L'analyste identifie les clients à plus forte valeur (HAVING total_spent > 500), analyse la répartition géographique de la clientèle (DISTINCT city), et détermine les marques les plus rentables (SUM(revenue) GROUP BY brand). Il vérifie également quels produits sont les plus appréciés en cherchant ceux qui n'ont jamais été retournés (NOT EXISTS).
- **Décision :** Lancer un programme VIP pour les meilleurs clients, concentrer les publicités sur la France, et mettre en avant les produits des marques "Adibas" et "Nikéa". Les produits jamais retournés feront l'objet d'une campagne "Satisfaction Garantie".

Scénario 2 : Investigation de Fraude (Rôle : Analyste Fraude)

Objectif : Confirmer ou infirmer une suspicion de fraude aux retours suite à une alerte de l'entrepôt (ticket TICKET011).

- **Démarche d'investigation :**
  1. **Identification du suspect :** L'analyste utilise une jointure pour retrouver le client (CL106) associé au retour suspect (RET016).

```sql
-- on retrouve d'abord l'id du client qui a fait le retour
SELECT *
FROM return_ r
JOIN order_ o ON r.order_id = o.order_id
WHERE return_id = 'RET016';
```

  2. **Analyse comportementale :** Il examine l'historique des commandes et des retours du client via une jointure externe (LEFT JOIN). Il découvre un schéma

suspect : deux commandes de grande valeur, passées à quelques jours d'intervalle, et toutes deux intégralement retournées.

```sql
-- Étape 2 : Examiner l'historique complet des commandes et des retours du client.
-- L'analyste cherche des montants élevés, des retours fréquents, ou tout autre schéma inhabituel
-- ------------------------------------------------------------------------------------
SELECT
    o.order_id,
    o.date_ AS order_date,
    o.total_amount,
    o.status,
    r.return_id,
    r.return_date,
    r.reason AS return_reason,
    r.refund_amount
FROM order_ o
LEFT JOIN return_ r ON o.order_id = r.order_id
WHERE o.client_id = 'CL106'
ORDER BY o.date_ DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_date	total_amount	status	return_id	return_date	return_reason	refund_amount
ORD032	2024-03-10 15:20:00	699.99	DELIVERED	RET017	2024-03-15 14:00:00	Changed mind	699.99
ORD031	2024-03-05 11:00:00	899.99	DELIVERED	RET016	2024-03-12 10:00:00	Item defective	899.99

3. **Corrélation des preuves :** Il consulte les tickets de support associés à ce client et trouve le TICKET011, qui confirme que l'entrepôt a reçu une boîte vide pour l'un des retours.

```sql
SELECT
    st.ticket_id,
    st.subject,
    st.description,
    st.status,
    th.date_time,
    th.exchange_content
FROM support_ticket st
JOIN ticket_history th ON st.ticket_id = th.ticket_id
WHERE st.client_id = 'CL106'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_id	subject	description	status	date_time	exchange_content
TICKET011	Warehouse Alert: Empty Box for Return RET016	The warehouse team has flagged return RET01...	OPEN	2024-03-13 09:05:00	System: Ticket created automatically from ware...
TICKET011	Warehouse Alert: Empty Box for Return RET016	The warehouse team has flagged return RET01...	OPEN	2024-03-13 09:10:00	Fraud Analyst: Acknowledged. Starting investig...

The warehouse team has flagged return RET016. The box for the iScroll Pro arrived empty. Investigating potential fraud.

4. **Recherche de réseau :** Une sous-requête avec IN est utilisée pour vérifier si les adresses de livraison du suspect ont été utilisées par d'autres comptes. Le résultat est négatif.

```sql
SELECT
    c.client_id,
    c.email,
    ca.location_id
FROM client c
JOIN client_address ca ON c.client_id = ca.client_id
WHERE ca.location_id IN (
    SELECT location_id FROM client_address WHERE client_id = 'CL106'
    )
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

client_id	email	location_id

- **Décision :** Les preuves sont suffisantes pour confirmer la fraude. L'analyste décide de bloquer le compte client et d'annuler tout remboursement en attente.

Scénario 3 : Résolution de Plainte Client (Rôle : Agent de Support)

Objectif : Résoudre les problèmes soulevés par la cliente fidèle Frieren (CL001) dans le ticket TICKET012 : une surcharge sur la commande ORD001 et un remboursement manquant pour son retour RET018.

- **Démarche de résolution :**

1. **Vérification du statut client :** L'agent vérifie le profil de Frieren et confirme son statut de cliente fidèle, justifiant un traitement prioritaire.

```sql
SELECT
    client_id,
    email,
    first_name,
    last_name,
    fidelity_status,
    subscribtion_date
FROM client
WHERE client_id = 'CL001';
-- Observation : Cliente fidèle, inscrite depuis 2022. Sa plainte doit être traitée avec la plus haute priorité :)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

client_id	email	first_name	last_name	fidelity_status	subscribtion_date
CL001	frieren@mage.com	Frieren	The Mage	1	2022-01-15 10:30:00
NULL	NULL	NULL	NULL	NULL	NULL

2. **Diagnostic de la surcharge :** L'agent compare le montant total de la commande ORD001 avec la somme calculée des lignes de commande correspondantes. Il confirme une surcharge.

```
291     -- 2.a: Voir le montant qui a été facturé.
292 •   SELECT order_id, total_amount FROM order_ WHERE order_id = 'ORD001'; -- 189.97
293     -- 2.b Calculer le montant qui aurait dû être facturé
294 •   SELECT
295         SUM(ol.quantity_ordered * ol.unit_price_paid) AS correct_total
296     FROM order_line ol
297     WHERE ol.order_id = 'ORD001'; -- 149.97
298     -- Conclusin: La commande a un total de 189.97€ alors que la somme des lignes fait 149.97€. Il y a bien eu une erreur
```

3. **Diagnostic du remboursement manquant :** L'agent utilise une requête avec NOT EXISTS pour prouver l'absence d'une transaction de remboursement (montant négatif) associée au retour RET018.

```
308 •   SELECT
309         r.return_id,
310         r.refund_amount
311     FROM return_ r
312     WHERE r.return_id = 'RET018'
313         AND NOT EXISTS (
314             SELECT 1
315             FROM transaction t
316             -- On cherche une transaction négative pour le meme order_id que celui de la cliente
317             WHERE t.order_id = r.order_id AND t.amount = -r.refund_amount
318         );
319     -- Conclusion: Le retour RET018 d'un montant de 29.99€ a bien été enregistré. mais aucune transaction de remboursement n'a été effectuée
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

return_id	refund_amount
RET018	29.99
NULL	NULL

4. **Vue d'ensemble :** Pour s'assurer qu'il n'y a pas d'autres problèmes, l'agent utilise une requête UNION pour afficher l'historique complet des commandes, retours et transactions de la cliente dans un seul flux chronologique.

```
325 •   SELECT 'Order' AS type, order_id AS id, date_ AS event_date, total_amount AS amount FROM order_ WHERE client_id = 'CL001'
326     UNION ALL
327     SELECT 'Return' AS type, return_id AS id, return_date AS event_date, -refund_amount AS amount FROM return_ r JOIN order_ o ON r.order_id = o.order_id WHERE o.client_id = 'CL001'
328     UNION ALL
329     SELECT 'Transaction' AS type, transaction_id AS id, o.date_ AS event_date, amount FROM transaction t JOIN order_ o ON t.order_id = o.order_id WHERE o.client_id = 'CL001'
330     ORDER BY event_date DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

type	id	event_date	amount
Order	ORD026	2024-04-21 09:55:00	299.99
Transaction	TXN026	2024-04-21 09:55:00	299.99
Return	RET018	2024-01-25 10:00:00	-29.99
Return	RET001	2024-01-20 10:30:00	-89.99
Order	ORD001	2024-01-10 14:30:00	189.97
Transaction	TXN_REF_001	2024-01-10 14:30:00	-89.99
Transaction	TXN001	2024-01-10 14:30:00	189.97

- **Décision :** L'agent contacte la cliente, s'excuse pour les deux erreurs, et procède manuellement au remboursement du montant total dû. En guise de geste commercial, il lui offre un code de réduction personnel.

III. Bilan Critique et Perspectives d'Amélioration

Ce projet nous a permis de mettre en application le cycle complet de conception d'une base de données, en nous confrontant à des choix architecturaux concrets. Le modèle actuel est robuste, normalisé et répond aux besoins définis. Cependant, tout système est perfectible right ?

Axes d'Amélioration Potentiels :

1. **Automatisation de la Logique Métier via des Triggers :**
    - **Constat :** Actuellement, la mise à jour des stocks après une vente ou un retour doit être gérée par l'application qui interagit avec la base (hypothèse). Cela crée un risque d'incohérence si une transaction n'est pas correctement traitée.
    - **Amélioration :** Par manque de temps, nous n'avons pas implémenté de **triggers**. L'ajout de triggers au niveau de la base de données aurait permis d'automatiser ces processus. Par exemple, un trigger AFTER INSERT sur order_line aurait pu décrémenter le stock de manière atomique et fiable. De même, un trigger sur la mise à jour du statut d'une commande à "CANCELLED" aurait pu ré-incrémenter le stock.
    - **Bénéfice :** L'intégration de triggers aurait renforcé l'intégrité des données en rendant la base de données moins dépendante de la logique applicative, garantissant que les règles métier critiques sont toujours respectées.
2. **Gestion Avancée des Promotions :**
    - Notre système de discount_code est fonctionnel mais simple. Une évolution majeure serait de permettre des règles promotionnelles plus complexes (ex : "2 articles achetés, le 3ème offert", réductions conditionnelles au montant du panier), ce qui nécessiterait des tables supplémentaires pour modéliser ces règles.
3. **Internationalisation :**
    - Le modèle gère plusieurs pays, mais pas les complexités associées comme la gestion de multiples devises pour les prix ou le calcul de la TVA en fonction du pays de livraison. L'intégration de ces aspects rendrait le système plus complet.

Si nous devions refaire ce projet, nous allouerions plus de temps (si jamais nous avions le temps oui puisque c'est un « mini projet ») à l'anticipation de ces règles métier avancées dès la phase de conception. Néanmoins, le modèle actuel constitue une fondation solide et évolutive, prête à intégrer ces améliorations futures.

Merci, désolé pour la longueur.