Proposition D’Audit et D’Optimisation de la Base de Données EMF-TONTINE

1. Introduction

Dans le cadre de l’amélioration des performances de la base de données, un audit a été réalisé afin d’identifier les pistes d’optimisation possibles. Cette analyse s’est appuyée sur le dictionnaire des données, le modèle conceptuel (MCD), les index existants, ainsi que les requêtes métier les plus courantes.

---

2. Objectifs de l’audit

- Identifier les colonnes fréquemment sollicitées dans les clauses `WHERE`, `JOIN` ou `ORDER BY`.
- Proposer des index pertinents pour améliorer les temps de réponse.
- Concevoir des vues stratégiques pour simplifier les requêtes métier complexes.
- Fournir des recommandations de monitoring pour le suivi post-optimisation.

---

3. Propositions d’Index

3.1 Index sur les clés étrangères

But : Accélérer les recherches fréquentes et les tris sur colonnes uniques.

NOM DE LA TABLE	SCRIPTE DE CRÉATION DES INDEXES
-- Adresse	CREATE INDEX idx_adresse_code_pays ON tbl_adresse(code_pays);

	CREATE INDEX idx_adresse_code_ville ON tbl_adresse(code_ville);
-- Affectation caisse	CREATE INDEX idx_affectation_caisse_id_caisse ON tbl_affectation_caisse(id_caisse);
	CREATE INDEX idx_affectation_caisse_ancien_user ON tbl_affectation_caisse(ancien_user);
	CREATE INDEX idx_affectation_caisse_nouveau_user ON tbl_affectation_caisse(nouveau_user);
-- Opérations	CREATE INDEX idx_operation_date_operation ON tbl_operation(date_operation);
	CREATE INDEX idx_operation_type_operation ON tbl_operation(type_operation);
	CREATE INDEX idx_operation_id_caisse ON tbl_operation(id_caisse);
	CREATE INDEX idx_operation_id_carnet ON tbl_operation(id_carnet);
-- Comptes	CREATE INDEX idx_compte_id_client ON tbl_compte(id_client);
	CREATE INDEX idx_compte_id_agence ON tbl_compte(id_agence);
-- Collectes	CREATE INDEX idx_collecte_date_collecte ON tbl_collecte(date_collecte);
	CREATE INDEX idx_collecte_id_souscription ON tbl_collecte(id_souscription);

3.2 Index sur colonnes de recherche courantes

But : Accélérer les recherches fréquentes et les tris sur colonnes uniques.

NOM DE LA TABLE	SCRIPTE DE CRÉATION DES INDEXES

-- Personnes	CREATE INDEX idx_personne_telephone ON tbl_personne(telephone);
	CREATE INDEX idx_personne_nif ON tbl_personne(nif);
	CREATE INDEX idx_personne_date_naissance ON tbl_personne(date_naissance);
-- Produits souscrits	CREATE INDEX idx_produit_souscrit_numero_carnet ON tbl_produit_souscrit(numero_carnet);
	CREATE INDEX idx_produit_souscrit_date_souscription ON tbl_produit_souscrit(date_souscription);
-- Caisses	CREATE INDEX idx_caisse_statut ON tbl_caisse(statut);
	CREATE INDEX idx_caisse_id_agence ON tbl_caisse(id_agence);
-- Articles	CREATE INDEX idx_article_code_interne ON tbl_article(code_interne);
	CREATE INDEX idx_article_libelle ON tbl_article(libelle);

3.3 Index composites

But : Optimiser les requêtes avec conditions multiples ou tris complexes.

CREATE INDEX idx_operation_type_date ON tbl_operation(type_operation, date_operation);
CREATE INDEX idx_personne_nom_prenom ON tbl_personne(nom, prenom);
CREATE INDEX idx_mouvement_compte_date ON tbl_mouvement(id_compte_gestion, date_mouvement);

3.4 Index sur colonnes booléennes (soft deletes)

But : Optimiser les filtres fréquents sur les enregistrements actifs.

```
CREATE INDEX idx_tbl_adresse_deleted ON tbl_adresse(deleted);
```

```
CREATE INDEX idx_tbl_agence_deleted ON tbl_agence(deleted);
```

```
CREATE INDEX idx_tbl_compte_deleted ON tbl_compte(deleted);
```

4. Propositions de Vues Stratégiques

4.1 Rapport des Transactions

```
CREATE VIEW vw_transactions_details AS
SELECT
  o.id, o.date_operation, o.type_operation, o.montant_operation,
  c.numero_compte, p.nom_complet AS client,
  a.nom_agence, ca.nom_caisse
FROM tbl_operation o
JOIN tbl_compte c ON o.id_compte_gestion = c.id
JOIN tbl_role_personne rp ON c.id_client = rp.id
JOIN tbl_personne p ON rp.id_personne = p.id
JOIN tbl_agence a ON o.id_agence_operation = a.id
JOIN tbl_caisse ca ON o.id_caisse = ca.id;
```

Justification : Facilite la génération de rapports de transactions sans refaire les jointures complexes.

4.2 État des Caisses

```
CREATE VIEW vw_etat_caisses AS
SELECT
  c.id, c.nom_caisse, c.montant_solde,
  MAX(o.date_operation) AS derniere_operation,
  COUNT(o.id) AS nb_operations_jour
```

```
FROM tbl_caisse c
LEFT JOIN tbl_operation o ON c.id = o.id_caisse
GROUP BY c.id;
```

Justification : Vue utile pour le monitoring quotidien des activités de caisse.

4.3 Produits souscrits par client

```
CREATE VIEW vw_clients_produits AS
SELECT
  p.id AS client_id, p.nom_complet, p.telephone,
  ps.numero_carnet, ps.date_souscription, pr.nom AS produit
FROM tbl_personne p
JOIN tbl_role_personne rp ON p.id = rp.id_personne
JOIN tbl_produit_souscrit ps ON rp.id = ps.id_client
JOIN tbl_produit pr ON ps.id_produit = pr.id;
```

Justification : Permet aux équipes marketing de suivre les souscriptions client.

5. Justification des Optimisations

Type d'optimisation	Bénéfices
Index sur FK	Accélère les jointures (ex : tbl_operation ↔ tbl_caisse)
Index composites	Optimise les requêtes avec plusieurs filtres ou tris
Index booléens	Rend les filtres sur deleted = false beaucoup plus rapides
Vues stratégiques	Simplifient les requêtes métier complexes et améliorent la lisibilité

6. Recommandations de Monitoring

6.1 Analyse de l’utilisation des index

```
SELECT * FROM pg_stat_all_indexes WHERE schemaname = 'public';
```

6.2 Suivi des requêtes lentes

```
SELECT * FROM pg_stat_statements ORDER BY total_time DESC;
```

6.3 Revue régulière des performances

Utiliser `EXPLAIN ANALYZE` sur les requêtes critiques pour ajuster les index si nécessaire.

---

7. Remarque

- Les index améliorent la lecture mais peuvent ralentir les écritures.
- Il est important d’évaluer la fréquence des mises à jour pour éviter une surcharge.
- Un équilibrage est nécessaire entre la performance en lecture et la performance en écriture.