TP #03 : Indexation fondamentale et avancée

Prérequis

- PostgreSQL installé et accessible
- Base de données IMDb importée (ou une autre base de données volumineuse)

Exercice 1: Index B-tree

1.1 Analyse sans index

Écrivez une requête avec EXPLAIN ANALYZE qui recherche dans la table title_basics tous les titres commençant par "The" (utilisez LIKE).

1.2 Création d'un index B-tree

Créez un index B-tree sur la colonne primary_Title de la table title_basics.

1.3 Analyse après indexation

Exécutez à nouveau la même requête qu'en 1.1 et analysez les différences de performance et de plan d'exécution.

1.4 Test des différentes opérations

Testez plusieurs types de conditions et notez lesquelles bénéficient de l'index B-tree:

1. Égalité exacte (WHERE primary_Title = '...')
2. Préfixe (WHERE primary_Title LIKE 'The%')
3. Suffixe (WHERE primary_Title LIKE '%The')
4. Sous-chaîne (WHERE primary_Title LIKE '%The%')
5. Ordre (ORDER BY primary_Title)

1.5 Analyse et réflexion

Répondez aux questions:

1. Pour quels types d'opérations l'index B-tree est-il efficace?
2. Pourquoi l'index n'est-il pas utilisé pour certaines opérations?
3. Dans quels cas un index B-tree est-il le meilleur choix?

Exercice 2: Index Hash

2.1 Requête d'égalité exacte

Écrivez une requête avec EXPLAIN ANALYZE recherchant un film précis par son identifiant tconst.

2.2 Création d'un index Hash

Créez un index de type Hash sur la colonne tconst de la table title_basics.

2.3 Comparaison avec B-tree

Créez également un index B-tree sur la même colonne, puis comparez les performances:

1. Mesurez le temps d'exécution avec chaque index
2. Comparez la taille des deux index (utilisez pg_indexes_size)
3. Testez une recherche par égalité et une recherche par plage

2.4 Analyse et réflexion

Répondez aux questions:

1. Quelles sont les différences de performance entre Hash et B-tree pour l'égalité exacte?
2. Pourquoi l'index Hash ne fonctionne-t-il pas pour les recherches par plage?
3. Dans quel contexte précis privilégier un index Hash à un B-tree?

Exercice 3: Index composites

3.1 Requête avec plusieurs conditions

Écrivez une requête qui recherche des films par genre et année (par exemple, tous les drames de 1994).

3.2 Test sans index

Analysez les performances sans index spécifique.

3.3 Index sur colonnes individuelles

Créez des index séparés sur les colonnes de genre et d'année, puis mesurez l'impact.

3.4 Index composite

Créez un index composite sur les deux colonnes (genre, année) et mesurez les performances.

3.5 Test de l'ordre des colonnes

Créez un deuxième index composite avec l'ordre inverse (année, genre) et comparez les performances pour:

1. Filtrer uniquement sur le genre
2. Filtrer uniquement sur l'année
3. Filtrer sur les deux colonnes
4. Trier par genre puis par année
5. Trier par année puis par genre

3.6 Analyse et réflexion

Répondez aux questions:

1. Comment l'ordre des colonnes dans l'index composite affecte-t-il son utilisation?
2. Quand un index composite est-il préférable à plusieurs index séparés?
3. Comment choisir l'ordre optimal des colonnes dans un index composite?

Exercice 4: Index partiels

4.1 Identifier un sous-ensemble fréquent

Analysez la distribution des années de sortie dans title_basics pour identifier une décennie récente avec beaucoup de films.

4.2 Requête sur ce sous-ensemble

Écrivez une requête qui recherche des films uniquement dans cette période récente.

4.3 Création d'un index partiel

Créez un index partiel qui couvre uniquement les films de cette période.

4.4 Comparaison avec index complet

Créez un index complet sur la même colonne, puis comparez:

1. Les performances pour les requêtes dans la période ciblée
2. Les performances pour les requêtes hors de cette période
3. La taille des deux index

4.5 Analyse et réflexion

Répondez aux questions:

1. Quels sont les avantages et inconvénients d'un index partiel?
2. Dans quels scénarios un index partiel est-il particulièrement utile?
3. Comment déterminer si un index partiel est adapté à votre cas d'usage?

Exercice 5: Index d'expressions

5.1 Recherche insensible à la casse

Écrivez une requête qui recherche des titres indépendamment de la casse (par exemple, trouver "Star Wars" même si saisi comme "star wars").

5.2 Mesure des performances sans index adapté

Analysez les performances avec l'index B-tree standard sur primary_Title.

5.3 Création d'un index d'expression

Créez un index sur l'expression LOWER(primary_Title).

5.4 Mise à jour de la requête et test

Modifiez votre requête pour utiliser LOWER() et mesurez l'amélioration des performances.

5.5 Autre exemple d'expression

Créez un index sur une autre expression (par exemple, sur l'extraction de l'année à partir d'une date) et testez son efficacité.

5.6 Analyse et réflexion

Répondez aux questions:

1. Pourquoi l'expression dans la requête doit-elle correspondre exactement à celle de l'index?
2. Quel est l'impact des index d'expressions sur les performances d'écriture?
3. Quels types de transformations sont souvent utilisés dans les index d'expressions?

Exercice 6: Index couvrants (INCLUDE)

6.1 Requête fréquente avec projection

Identifiez une requête qui sélectionne uniquement quelques colonnes de title_basics en filtrant sur une seule colonne (par exemple, récupérer primary_Title et start_Year en filtrant sur genre).

6.2 Index standard

Créez un index standard sur la colonne de filtrage et analysez si PostgreSQL peut réaliser un "Index Only Scan".

6.3 Index couvrant

Créez un index couvrant qui inclut les colonnes supplémentaires nécessaires.

6.4 Comparaison des performances

Mesurez la différence de performances entre:

1. Aucun index
2. Index standard
3. Index couvrant

6.5 Analyse et réflexion

Répondez aux questions:

1. Qu'est-ce qu'un "Index Only Scan" et pourquoi est-il avantageux?
2. Quelle est la différence entre ajouter une colonne à l'index et l'inclure avec INCLUDE?
3. Quand privilégier un index couvrant par rapport à un index composite?

Exercice 7: Recherche textuelle

7.1 Requête de recherche simple

Écrivez une requête qui recherche tous les titres contenant un mot spécifique (par exemple "love").

7.2 Test de différentes approches

Mesurez les performances avec:

1. LIKE sans index
2. LIKE avec index B-tree
3. Index trigram (GIN)

7.3 Recherche full-text

Créez un index de recherche full-text sur primaryTitle:

1. Ajoutez une colonne de type tsvector

2. Remplissez-la avec le résultat de to_tsvector
3. Créez un index GIN sur cette colonne
4. Réécrivez la requête avec to_tsquery

7.4 Analyse et réflexion

Répondez aux questions:

1. Quelles sont les différences entre LIKE, trigram et full-text search?
2. Quels compromis faites-vous en termes de précision, performance et espace?
3. Pour quels volumes de données et types de recherches chaque approche est-elle adaptée?

Exercice 8: Indexation de données JSON/JSONB

8.1 Préparation

Créez une table avec une colonne JSONB contenant des informations structurées comme par exemple une table contenant l'intégralité des informations sur un film dans une seule ligne qui permet de récupérer le contenu sans jointure.

8.2 Requêtes sur données JSON

Écrivez diverses requêtes qui:

1. Filtrent sur une propriété de premier niveau
2. Filtrent sur une propriété imbriquée
3. Vérifient l'existence d'une clé

8.3 Index sur JSONB

Créez plusieurs types d'index sur vos données JSONB:

1. Index GIN standard
2. Index GIN avec jsonb_path_ops
3. Index B-tree sur une propriété extraite

8.4 Analyse et réflexion

Répondez aux questions:

1. Quels types d'opérations bénéficient le plus des index GIN sur JSONB?
2. Quand utiliser jsonb_path_ops plutôt que l'index GIN standard?
3. Comment indexer efficacement une propriété spécifique dans une structure JSONB complexe?

Exercice 9: Analyse et maintenance des index

9.1 Étude de l'utilisation des index

Utilisez les vues système pg_stat_user_indexes et pg_stat_user_tables pour identifier:

1. Les index fréquemment utilisés
2. Les index rarement utilisés
3. Les index qui occupent beaucoup d'espace

9.2 Observation de la fragmentation

Analysez la fragmentation d'un index fréquemment mis à jour:

1. Mesurez sa taille avant opération

2. Effectuez un grand nombre d'insertions/mises à jour
3. Mesurez à nouveau sa taille

9.3 Maintenance des index

Effectuez des opérations de maintenance:

1. VACUUM sur une table
2. REINDEX sur un index fragmenté
3. ANALYZE pour mettre à jour les statistiques

9.4 Mesure de l'impact

Comparez les performances des requêtes avant et après maintenance.

9.5 Analyse et réflexion

Répondez aux questions:

1. Comment déterminer si un index est utile ou superflu?
2. Quels signes indiquent qu'un index nécessite une maintenance?
3. Quelle est la stratégie optimale pour planifier la maintenance des index?

Exercice 10: Synthèse et cas pratiques

10.1 Scénario: Application de streaming

Imaginez que vous concevez la base de données d'une plateforme de streaming vidéo. Identifiez les index nécessaires pour:

1. Recherche de contenu par titre/acteur/genre
2. Recommandations personnalisées
3. Historique de visionnage
4. Statistiques de popularité

10.2 Scénario: Système de réservation

Concevez une stratégie d'indexation pour un système de réservation hôtelière avec:

1. Recherche de disponibilité par date et lieu
2. Historique des réservations par client
3. Rapports d'occupation
4. Tarification dynamique

10.3 Réflexion globale

Répondez aux questions:

1. Comment équilibrer les performances de lecture et d'écriture?
2. Comment décider quels index créer en priorité?
3. Quelles métriques utiliser pour évaluer l'efficacité de votre stratégie d'indexation?
4. Comment faire évoluer votre stratégie d'indexation avec la croissance des données?