HEIG VD | MEI
Media Engineering Institute

05 - Optimisation

Infrastructure de données 1

Noemi Romano | 27 mars 2025

Indexation

- Structure de données qui **accélère** les recherches dans une table.
- Utilisé pour optimiser les requêtes avec **WHERE**, **JOIN**, **ORDER BY**, etc.

Avantage : améliore la vitesse de lecture

Inconvénient : ralentit les écritures (**INSERT**/ **UPDATE**) et consomme de l'espace

Un **index** en base de données, c'est comme l'index à la fin d'un livre : au lieu de lire toutes les pages une par une, on va directement à l'endroit où se trouve l'information.

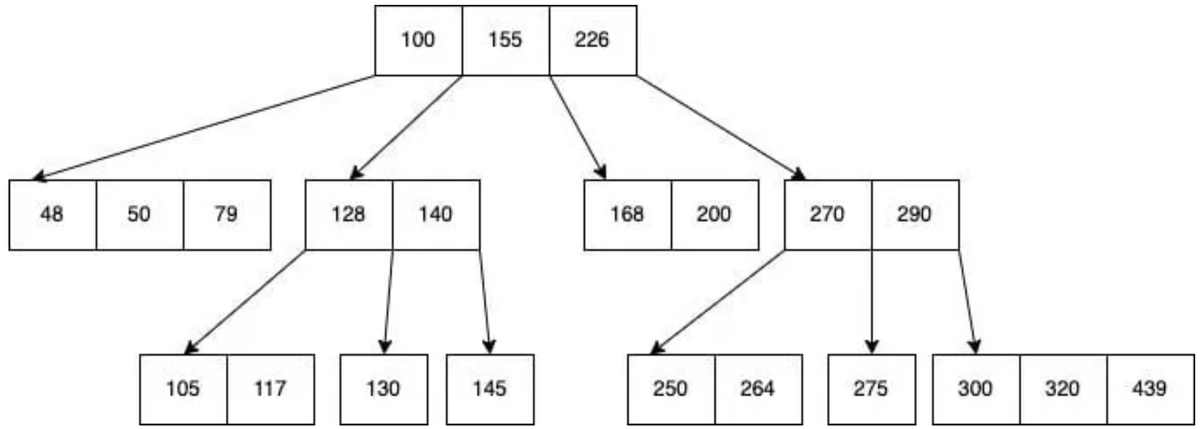
Types d’index

Index PostgreSQL

Type	Description	Syntaxe
btree (défaut)	Ordonné, efficace pour la plupart des requêtes	`CREATE INDEX idx_name ON table(col);`
hash	Utilisé pour les égalités (=)	`CREATE INDEX idx_hash ON table USING hash(col);`
GIN (Generalized Inverted Index)	Index sur tableaux, JSONB, texte plein	`CREATE INDEX idx_gin ON table USING gin(col);`
GiST	Index spatial, texte approximatif, etc.	`CREATE INDEX idx_gist ON table USING gist(col);`

B-Tree index

- Type d’index par défaut dans PostgreSQL.
- Fonctionne très bien pour les comparaisons classiques : **=**, **<**, **>**, **BETWEEN**, **ORDER BY**, etc.
- Structure sous forme d’arbre équilibré → accès rapide à la donnée triée.
- Très efficace pour des recherches sur des colonnes de type **INT**, **TEXT**, **DATE**, etc.
- Utilisé automatiquement pour les clés primaires et uniques

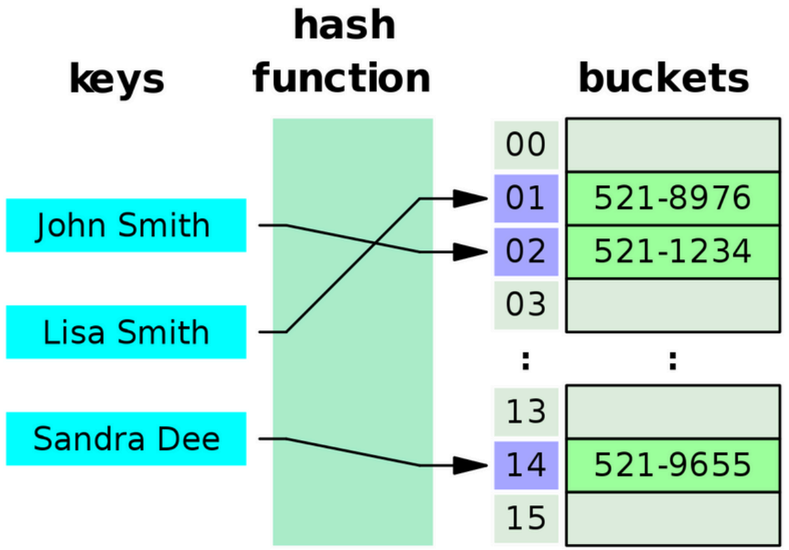


B-Tree (Source: Dhanushka Madushan)

B-Tree index PostgreSQL

Hash index

- Optimisé uniquement pour les recherches par égalité (=)
- Ne peut pas être utilisé pour des tris ou intervalles (<, >, etc.)
- Moins d’espace que B-tree, mais aussi moins polyvalent
- Cas d’usage : recherche exacte sur une clé alternative très utilisée

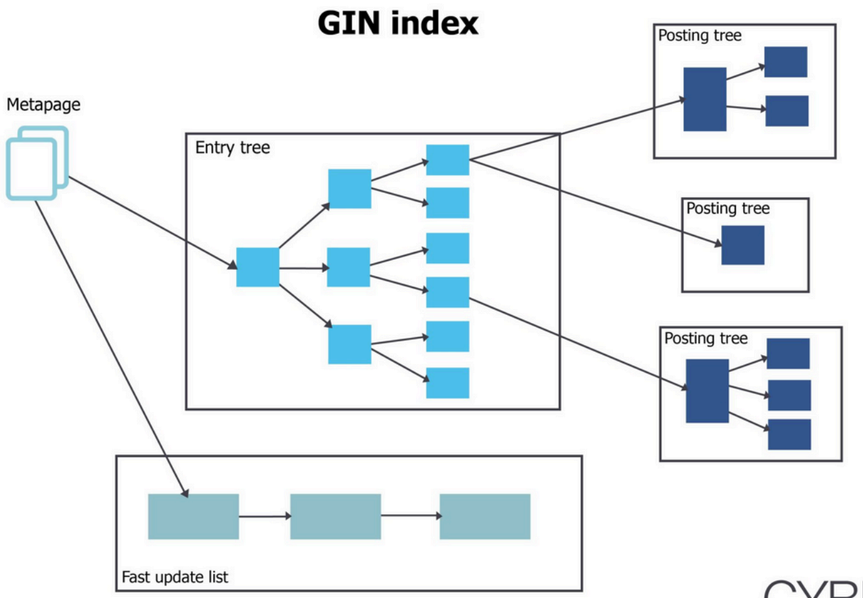


Source: B+ Tree vs Hash Index (and when to use them), SQL Pipe

GIN index

Generalized Inverted Index (GIN)

- Très performant pour faire des recherches dans :
  - Du texte (avec to_tsvector, full-text search)
  - Des colonnes de type ARRAY
  - Des champs JSONB avec opérateurs @>, ?, etc.
- Permet de savoir dans quelles lignes un mot ou une valeur apparaît.
- Plus lent à écrire et plus lourd, mais indispensable pour les recherches complexes.



www.cybertec-postgresql.com

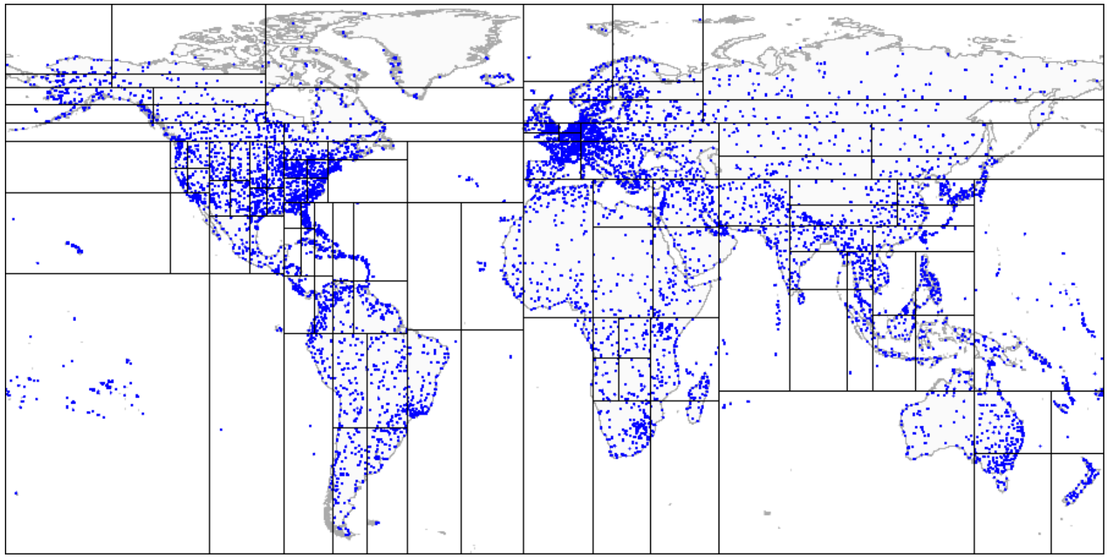
CYBERTEC
The PostgreSQL Database Company

(SP-)GiST index

(Space Partioned) Generalized Search Tree

Très flexible, peut indexer :

- Des données géospatiales (PostGIS)
- Des recherches approximatives (recherche floue, distances, etc.)
- Structure adaptable à plusieurs types de logique (proximité, similarité…).
- Un peu comme le "couteau suisse" des index PostgreSQL.



Source: Indexes in PostgreSQL — 6 (SP-GiST), Habr

Impact

✅ Avantages :

- Accélèrent les requêtes avec **WHERE**, **JOIN**, **ORDER BY**, etc.
- Améliorent les performances sur les grandes tables
- Réduisent le temps de traitement côté serveur

❌ Inconvénients :

- Ralentissent les écritures (**INSERT**, **UPDATE**, **DELETE**)
- Peuvent devenir inutiles si mal choisis
- Consomment de l’espace disque supplémentaire

Créer un index si...

- Tu filtres souvent une colonne dans WHERE → **B-Tree**
- Tu fais des jointures fréquentes sur une colonne → **B-Tree** / **Hash**
- Tu travailles sur des plages de dates ou des intervalles → **B-Tree**
- Tu cherches dans du texte ou des tableaux → **GIN**
- Tu manipules des données spatiales ou géographiques → **GIST**

Bonnes pratiques générales

- Évite de créer trop d'index → chaque écriture les met à jour !
- Utilise **EXPLAIN (ANALYZE)** pour mesurer les performances
- Supprime les index inutilisés ou redondants
- Adapte le type d'index à la nature des données et des requêtes

Révisions

GitHub