Les Vues

Introduction

Imaginez que nous travaillons dans une banque avec une base de données qui contient plusieurs tables, dont :

Table COMPTE : Informations sur chaque compte bancaire

```
Compte (numéro de compte, type de compte, solde, et identifiant du client).
```

Table CLIENT : Informations personnelles des clients

```
Client (identifiant, nom, adresse).
```

Table TRANSACTION : Historique des opérations effectuées sur les comptes bancaires

```
Transaction (date, montant, type de transaction, numéro de compte).
```

Le service de Comptabilité doit générer des rapports quotidiens pour analyser :

- Les comptes avec un solde supérieur à un certain montant.
- Les clients ayant effectué des transactions importantes (dépôts/retraits) au cours de la dernière semaine.

Introduction

Par exemple, pour afficher les clients ayant un solde élevé et les informations associées à leurs transactions récentes,

```
SELECT c.nom, c.adresse, cp.numero_compte, cp.solde, t.date_transaction, t.montant
FROM CLIENT c
JOIN COMPTE cp ON c.id_client = cp.id_client
JOIN TRANSACTION t ON cp.numero_compte = t.numero_compte
WHERE cp.solde > 5000
AND t.date_transaction >= SYSDATE - 7;
```

Cette requête est :

- Complexe (plusieurs jointures et des conditions)
- Lourde à maintenir
- Inefficace pour les utilisateurs récurrents

Solution : Créer des Vues

Pour simplifier ce processus, nous décidons de créer deux vues.

Vue `vue_comptes_soldes_eleves` : Une vue pour afficher uniquement les comptes avec un solde supérieur à 5000.

Avantage : Les comptables peuvent consulter cette vue directement sans avoir à écrire une requête complexe.

Vue `vue_transactions_recentes` : Une vue pour afficher les transactions de la dernière semaine.

Avantage : En utilisant cette vue, les comptables peuvent facilement obtenir un sous-ensemble des transactions récentes sans devoir filtrer manuellement les dates.

Définition

Une **vue** en SQL est une structure virtuelle dans une base de données qui permet d'accéder à un sous-ensemble de données sans en stocker directement les valeurs. Contrairement aux tables, qui contiennent des données stockées de manière physique, les vues ne retiennent que la logique de sélection des données, définie par une requête SQL sous-jacente.

En d'autres termes, une vue est le résultat d'une requête enregistrée, qui est actualisée chaque fois qu'on y accède. Elle présente une perspective filtrée ou réorganisée des informations disponibles dans une ou plusieurs tables de la base de données.

Avantages

- **Simplicité d'accès aux données :** grâce à la création de vues spécifiques, il est plus facile pour les utilisateurs finaux d'interroger les données sans avoir besoin de connaître les structures complexes des tables sous-jacentes.
- **Meilleure organisation des données :** les vues permettent de créer des structures logiques, regroupant les données par thème ou fonctionnalité.
- **Contrôle des accès :** en restreignant l'accès aux données par le biais de vues, il est possible de sécuriser et protéger certaines informations sensibles ou confidentielles.
- **Compatibilité descendante :** dans certains cas, il peut être nécessaire de modifier la structure d'une table existante. Si cette table est utilisée par de nombreuses requêtes et applications, ces modifications peuvent entraîner de nombreux problèmes. En utilisant des vues pour masquer ces changements, on assure une compatibilité descendante avec les requêtes et applications existantes.

Tables vs Vue

Tables

- Contient les données de manière physique, stockées dans la base de données.
- Les données de la table sont permanentes jusqu'à ce qu'elles soient modifiées ou supprimées.

Vue

- Ne contient pas de données réelles ; c'est une représentation virtuelle des données stockées dans une ou plusieurs tables.
- Les données de la vue changent dynamiquement en fonction des données présentes dans les tables source.
- Ne prend pas d'espace supplémentaire pour stocker des données (à l'exception des vues matérialisées).

Syntaxe

La requête SQL qui définit les données et la structure de la vue. Elle peut inclure des filtres, des jointures, et des conditions complexes.

```
CREATE OR REPLACE VIEW <nom vue> [(liste des attributs)]
AS <requête de sélection>
[WITH CHECK OPTION | WITH READ ONLY]
```

- **[(LISTE_D_ATTRIBUTS)]** : Optionnel, cela permet de spécifier des noms d'attributs personnalisés pour les colonnes de la vue.
- **[WITH CHECK OPTION]** : Optionnel, cette clause garantit que toutes les modifications (INSERT, UPDATE) faites via la vue respectent la condition définie dans la requête de la vue. Si une modification tente de violer cette condition, elle sera rejetée.
- **WITH READ ONLY :** Aucune modification (INSERT, UPDATE) n'est possible

Types de Vues

Vues simples :

- Basées sur une seule table.
- Permettent souvent les opérations de modification (INSERT, UPDATE, DELETE), sous certaines conditions.

Une vue affichant uniquement les produits en stock supérieur à 50 unités dans la table PRODUIT.

Vues complexes :

- Basées sur plusieurs tables ou utilisent des fonctions avancées comme GROUP BY, DISTINCT, UNION, etc.
- Souvent non modifiables (impossibilité d'INSERT, UPDATE, DELETE).

Une vue combinant les données des tables CLIENT et COMMANDE pour afficher les clients ayant passé une commande.

Règles d'utilisations de vues

Le SELECT principal de la vue contient	SELECT	UPDATE	DELETE	INSERT
Plusieurs tables	OUI	NON	NON	NON
GROUP BY	OUI	NON	NON	NON
DISTINCT	OUI	NON	NON	NON
fonction de groupe	OUI	NON	NON	NON
Attribut calculé	OUI	NON	OUI	NON
Attribut NOT NULL pas dans le SELECT	OUI	OUI	OUI	NON
UNION, INTERSETC, MINUS	OUI	NON	NON	NON

Types d'opérations autorisées

- Les vues peuvent être interrogées avec des requêtes SELECT pour afficher les données, tout comme une table.
- Dans certains cas, il est possible d'effectuer des opérations de modification (INSERT, UPDATE, DELETE) via une vue, mais sous certaines conditions.
- Les opérations d'INSERT, UPDATE, et DELETE sur une vue sont restreintes selon la structure de la vue Si la vue est basée sur plusieurs tables (JOIN), elle n'est pas modifiable.
- GROUP BY, DISTINCT, fonctions de groupe : Les vues qui incluent ces éléments ne permettent pas les opérations d'INSERT, UPDATE ou DELETE.
- UNION, INTERSECT, MINUS : Les vues créées avec ces opérateurs de combinaison ne sont pas modifiables.
- Les vues ne peuvent pas inclure des colonnes qui ne respectent pas les contraintes NOT NULL de la table sous-jacente si elles doivent permettre des opérations d'insertion.