Université Abou Bakr Belkaid Tlemcen
Faculté des Sciences
Département d'informatique

Administration des Bases de Données

L3 2017-2018

Mr H.MATALLAH

Administration des Bases de Données

1. Notions fondamentales
2. SQL Avancé
3. Gestion d'intégrité et cohérence
4. Vues et Index
5. Optimisation des requêtes
6. Gestion des transactions : Gestion des accès concurrents

CHAPITRE 4

VUES & INDEX

Vues

Problématique

- Une BD peut contenir des centaines de tables avec des milliers d'attributs
- Une BD peut contenir plusieurs utilisateurs avec différents droits d'accès
- Les requêtes d'interrogation peuvent être assez complexes
  - Difficulté de formuler ces requêtes
  - Lenteur pour exécuter ces mêmes requêtes

Vues

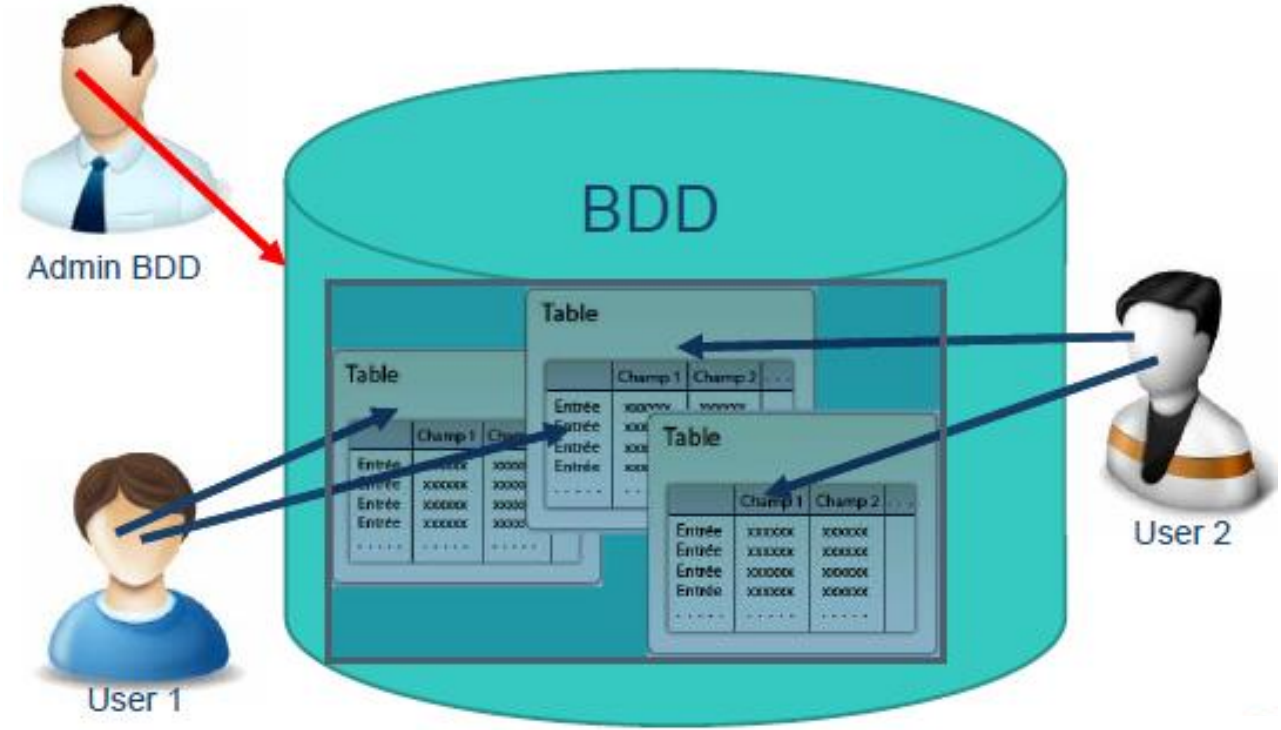
Solution

- Pour simplifier la visibilité de certaines données
- Pour assurer la confidentialité de certaines données
- Pour simplifier la formulation et l'exécution des requêtes complexes
- Solution : Adapter le schéma et les données à des applications spécifiques

VUES

Vues

- ***Définition***



Vues

Définition

- Table virtuelle qui n'a pas d'existence propre, résultant d'une requête SELECT, à laquelle nous attribuons un nom
- **Table** dont les données ne sont pas **stockées physiquement** mais qui se réfère à des données stockées dans leurs **tables d'origine**
- Il n'y a pas de duplication des informations mais stockage de la définition de la vue
- L'utilisateur qui interroge une vue à l'illusion d'accéder à une (ou plusieurs) table(s) contenant réellement des données, alors qu'il exécute une instruction SQL faisant référence à une ou plusieurs tables réelles

Vues

Définition

- Une **fenêtre** sur la **BD** permettant à chacun de voir les données comme il souhaite
- Une vue constitue une manière différente de consulter des données issues d'une ou de plusieurs tables de la base de données
- Calculée et interprétée dynamiquement à chaque exécution d'une requête qui y fait référence
- Une fois créée, une vue s'utilise comme une table avec : SELECT, UPDATE, DELETE, INSERT, GRANT

Vues

Avantages

- **Confidentialité**
  - Les utilisateurs n'ont le droit d'accéder qu'aux données autorisées par les vues
- **Cacher la complexité des données aux utilisateurs**
  - Vue simplifiée des données
  - Chaque utilisateur pourra avoir sa vision propre des données (indépendance logique)
  - Simplifier les requêtes des utilisateurs non spécialistes
- **Requêtes complexes**
  - Réduire la complexité syntaxique des requêtes
  - Sauvegarder les définitions et les plans d'exécution des requêtes fréquemment utilisées
- **Présenter différentes perspectives sur les données aux utilisateurs**
  - Renommage des colonnes
- **Mise à jour des tables transparente aux utilisateurs**

Vues

Création

- Remplace la vue si elle existe

CREATE [**OR REPLACE**] **VIEW** Nom_Vue [(Nom_col1, Nom_col2, ….)]

AS Sous-Requête

[**WITH CHECK OPTION** [CONSTRAINT nom_contrainte]]

[**WITH READ ONLY**]

- interdit la MAJ

- vérifie que la mise à jour faite à travers la vue ne produise que des lignes qui font partie de la sélection de la vue

- Par défaut, les noms des colonnes de la vue sont les mêmes que les noms des colonnes résultat du SELECT
- Si certaines colonnes résultat du SELECT sont des expressions, il faut renommer ces colonnes dans le SELECT ou spécifier les noms de colonne de la vue

Vues

Création : Exemple

EMPLOYE

Nom	Sal_Mens (DA)	Nbr_Jour_Trav	Dep
BENAISSA	49600	16	MTN
BELMOKHTAR	67200	21	PRD
DAOUDI	43200	18	COM
SALHI	50000	20	ADM
SELADJI	49300	17	PRD
MAHIEDDINE	53200	19	MTN
MOKRI	66000	22	PRD

DEPARTEMENT

Dep	Designation
MTN	Maintenance
PRD	Production
COM	Commercial
ADM	Administration

- Création d'une vue contenant le nom et le nombre de jours travaillés des employés du Dép de Production
  - **CREATE VIEW** EMP_PROD
    AS SELECT Nom, Nbr_Jour_Trav **FROM** EMPLOYE
    WHERE Dep = PRD ;
  - **SELECT** * **FROM** Emp_Prod

EMP_PROD

Nom	Nbr_Jour_Trav
BELMOKHTAR	21
SELADJI	17
MOKRI	22

Vues

Création : Exemple

EMPLOYE

Nom	Sal_Mens (DA)	Nbr_Jour_Trav	Dep
BENAISSA	49600	16	MTN
BELMOKHTAR	67200	21	PRD
DAOUDI	43200	18	COM
SALHI	50000	20	ADM
SELADJI	49300	17	PRD
MAHIEDDINE	53200	19	MTN
MOKRI	66000	22	PRD

DEPARTEMENT

Dep	Designation
MTN	Maintenance
PRD	Production
COM	Commercial
ADM	Administration

- Création d'une vue sur Nom, Département et Salaire journalier

```
CREATE VIEW SAL_JOUR (Nom, Departement, Sal_J)
AS SELECT Nom, Designation, Sal_Mens/Nbr_Jour_Trav
FROM EMPLOYE, DEPARTEMENT
WHERE EMPLOYE.Dep = DEPARTEMENT.Dep ;
```

SAL_JOUR

Nom	Departement	Sal_J
BENAISSA	Maintenance	3100
BELMOKHTAR	Production	3200
DAOUDI	Commercial	2400
SALHI	Administration	2500
SELADJI	Production	2900
MAHIEDDINE	Maintenance	2800
MOKRI	Production	3000

Vues

Requête avec Vue

- Calculer la moyenne des salaires journaliers par département

```
SELECT Departement, AVG (Sal_J)
FROM SAL_JOUR
GROUP BY Departement ;
```

Vue d'une Vue

- Créer une vue qui fait la restriction sur Nom, Département dont le Salaire Journ est < 3000

```
CREATE VIEW PETIT_SAL
AS SELECT Nom, Departement
FROM SAL_JOUR
WHERE Sal_J < 3000 ;
```

Vues

Re-Création

- CREATE VIEW EMP_PROD

  AS SELECT * FROM EMPLOYE

  WHERE Dep=PRD or Dep=MTN ;

  - Erreur ! : **Vue existe déjà**

- CREATE OR REPLACE VIEW EMP_PROD

  AS SELECT * FROM EMPLOYE

  WHERE Dep=PRD or Dep=MTN ;

  - **La Vue est remplacée par cette nouvelle définition**

Vues

Création

- CREATE VIEW EMP_TRAV

  AS SELECT * FROM EMPLOYE

  WHERE Nbr_Jour_Trav < 18

  WITH CHECK OPTION CONSTRAINT Ck_JourTrav ;

- INSERT INTO EMP_TRAV

  VALUES ('AZZOUZI ', 58500, 22, 'ADM')

✗ Impossible ! : Nbr_Jour_Trav > 18

Vues

Création

- CREATE VIEW PETIT_SAL

  AS SELECT * FROM EMPLOYE

  WHERE Sal_Mens < 55000

  WITH READ ONLY ;

- INSERT INTO PETIT_SAL

  VALUES ('ABDERRAHMANE', 47000, 18, 'COM')

✘ Impossible ! : Pas de mise à jour de la table autorisée à travers la vue

Vues

Mise à jour de la vue

- Lorsqu'il est possible d'exécuter des instructions INSERT, UPDATE ou DELETE sur une vue, cette dernière est dite « modifiable » (updatable view)

INSERT INTO Nom_Vue **VALUES** ...

UPDATE Nom_Vue **SET** ... **WHERE** ...

DELETE FROM Nom_Vue **WHERE** ...

Vues

Mise à jour de la vue

- **CREATE OR REPLACE VIEW** EMP_SAL
  AS SELECT * **FROM** EMPLOYE **WHERE** Sal_Mens < 55000 ;

  - **INSERT INTO** EMP_SAL
    VALUES ('BRAHIM', 39000, 14, 'PRD') ;
    - Ligne aussi insérée dans la table EMPLOYE

  - **UPDATE** EMP_SAL **SET** Sal_Mens=Sal_Mens * 1.1
    WHERE Sal_Mens < 48000 ;
    - Ligne aussi modifiée dans la table EMPLOYE

  - **DELETE FROM** EMP_SAL
    WHERE Nom='BRAHIM' ;
    - Ligne aussi supprimée de la table EMPLOYE

Vues

Conditions de Mise à jour de la vue

Pour qu'une vue soit modifiable, sa requête de définition SELECT ne doit pas contenir :

- **Distinct**
- Fonctions : **Avg, Count, Max, Min, Sum**
- **Group By, Having**
- **Order By**
- **Attributs Calculés** *(Ex : TTC=THT+THT*TVA, Sal=Sal*1.1)*
- **Jointure** (Sauf Exceptions)
- **Opérateurs Ensemblistes** : Union, Intersect, Minus
- **Sous-requête**

Vues

Suppression et Renommage

- **DROP VIEW** Nom_Vue ;

- **RENAME** Ancien_Nom **TO** Nouveau_Nom ;

Vues

Vue matérialisée

- Crée une nouvelle table contenant les résultats de la requête utilisée dans la vue
- On stocke à la fois la définition et le résultat
- Duplication des données
- Vue physique d'une table (Snapshot ou Réplique)
- Nécessité de synchroniser les données
- La fréquence des mises à jour de la vue matérialisée est à préciser

Vues

Vue matérialisée

- **Contexte d'utilisation**

  - **BDs Distribuées** : Pour répliquer les données sur les différents sites

  - **Entrepôts de données** (DataWarehouse) : Pour précalculer et stocker les données agrégées comme somme et moyenne

Vues

Vue matérialisée

- **Create Materialized View** Nom Vue Matérialisée

  Build [Immediate | Deferred]

  Refresh [On Commit | On Demand | Start With … Next …]

  As (Select … From … Where …) ;

- **2 Options de la clause Build**
  - **Immediate** : La VM est immédiatement remplie
  - **Deferred** : La VM est remplie lors du premier rafraichissement demandé

Vues

Vue matérialisée

- **CREATE MATERIALIZED VIEW** <Nom Vue Matérialisée>

  BUILD [Immediate | Deferred]

  REFRESH [On Commit | On Demand | Start With … Next …]

  AS (Select … From … Where …) ;

- **3 Modes de déclenchement de Refresh**
  - **Synchrone** - clause **On Commit** : Mise à jour de la table maître qui déclenche automatiquement le REFERESH
  - **Asynchrone**, à la demande - clause **On Demand** : C'est le mode par défaut
  - **Asynchrone**, cyclique - clauses **Start With** et **Next**, qui précise une date de début et une période de temps intermédiaire

Vues

Vue matérialisée

- Création d'une vue matérialisée sur Nom, Département et Nombre de jours travaillés

```
CREATE MATERIALIZED VIEW EMPLOYE2
BUILD Immediate
REFRESH START sysdate NEXT sysdate+1
AS (SELECT Nom, Designation, Nbr_Jour_Trav
FROM EMPLOYE, DEPARTEMENT
WHERE Employe.Dep = Departement.Dep) ;
```

- Cette VM est immédiatement remplie juste après l'exécution
- Cette réplique se synchronisera tous les 24h

Index

Introduction

- **Modèle relationnel** : les sélections peuvent être faites en utilisant le contenu de n'importe quelle colonne et les lignes sont stockées dans **n'importe quel ordre**
- Pour retrouver les employés qui travaillent en administration et qui ont travaillé plus de 20 Jours par ex**, il faut balayer toute la table**
- Un tel moyen d'accès conduit à **des temps excessifs** pour des tables dépassant quelques centaines de lignes
- **Solution** : **création d'index** (clé secondaire), qui permettra de satisfaire aux **requêtes les plus fréquentes** avec des temps de réponse acceptables
- Permet **d'accéder rapidement et directement** aux lignes des tables

Index

Définition

- Un index sera matérialisé par la création de blocs disque contenant des couples (valeurs d'index, numéro de bloc) donnant le numéro de bloc disque dans lequel se trouvent les lignes correspondant à chaque valeur d'index
- Les index sont des structures permettant de retrouver une ligne dans une table à partir de la valeur d'une colonne ou d'un ensemble de colonnes
- Un index contient la liste triée des valeurs des colonnes indexées avec les adresses des lignes (numéro de bloc dans la partition et numéro de ligne dans le bloc) correspondantes.
- On peut indexer une table sur un ou plusieurs noms de colonnes cités par ordre d'importance
- On peut créer plusieurs index indépendants sur une même table

Index

Avantages et inconvénients

- Un index est un objet différent à la table auquel il se rapporte, il est mis à jour automatiquement lors de chaque mise à jour d'une table
- L'adjonction d'un index à une table ralentit les mises à jour (insertion, suppression, modification de la clé) mais accélère beaucoup la recherche d'une ligne dans la table
- L'index accélère la recherche d'une ligne à partir d'une valeur donnée de clé, mais aussi la recherche des lignes ayant une valeur d'index supérieure ou inférieure à une valeur donnée, car les valeurs de clés sont triées dans l'index
- **Ex** : Les requêtes suivantes bénéficieront d'un index sur le champ Nbr_Jour_Trav :

```
SELECT * FROM EMPLOYE WHERE Nbr_Jour_Trav = 21 ;
SELECT * FROM EMPLOYE WHERE Nbr_Jour_Trav >= 18 ;
SELECT * FROM EMPLOYE WHERE Nbr_Jour_Trav BETWEEN 19 AND 22 ;
```

Index

Principe

- Un index est utilisable même si le critère de recherche est constitué seulement du début de la clé:
  - **Ex** : La requête suivante bénéficiera d'un index sur la colonne Nom :

    SELECT * FROM EMPLOYE WHERE Nom LIKE 'M%' ;
  - Par contre si le début de la clé n'est pas connu, l'index est sans intérêt (Nom LIKE '???????')
- **Valeurs NULL** : Elles ne sont pas représentées dans l'index, ceci afin de minimiser le volume nécessaire pour stocker l'index
- L'index n'est utilisable que si le critère de sélection est le contenu de la colonne indexée, sans aucune transformation
  - **Ex** : Un index sur le Salaire ne sera pas utilisé pour la requête suivante :

    SELECT * FROM EMPLOYE WHERE Sal_Mens * 12 > 600000 ;

Index

Choix des index

- **Indexer en priorité :**
  1. Les clés primaires
  2. Les colonnes servant de critère de jointure
  3. Les colonnes servant souvent de critère de recherche

- **Ne pas indexer :**
  1. Les colonnes contenant peu de valeurs distinctes (index alors peu efficace)
  2. Les colonnes fréquemment modifiées

Index

Création

- **CREATE [UNIQUE] INDEX** Nom_Index
  ON Table (Col1 [Asc | Desc] , Col2, ...)
  [**PCTFREE** Nombre]
  [**ROWS** = Nombre_lignes] ;
  - **UNIQUE** : On interdit que deux lignes aient la même valeur dans la colonne indexée
  - **PCTFREE** : Précise le pourcentage de place laissée libre dans les blocs d'index à la création de l'index. Cette place libre évitera une réorganisation de l'index des les premières insertions de nouvelles clés. La valeur par défaut est 20%
  - **ROW** : Une estimation du nombre de lignes, permettant d'optimiser l'algorithme de classement
- Les requêtes SQL sont transparentes au fait qu'il existe un index ou non. C'est l'optimiseur du SGBD qui, au moment de l'exécution de chaque requête, recherche s'il peut s'aider ou non d'un index

Index

Exemple de création

- **CREATE INDEX** Index-Des
  ON DEPARTEMENT (Designation) ;

- **CREATE INDEX** Index-ND
  ON EMPLOYE (Nom, Dep) ;

Suppression

- **DROP INDEX** Nom_Index ;