Chapitre II :

VUES ET SEQUENCES

Plan du chapitre

I- Les vues

- Principes
- Création d'une vue
- Vue simple modifiable
- Vue multitable modifiable
- Suppression d'une vue

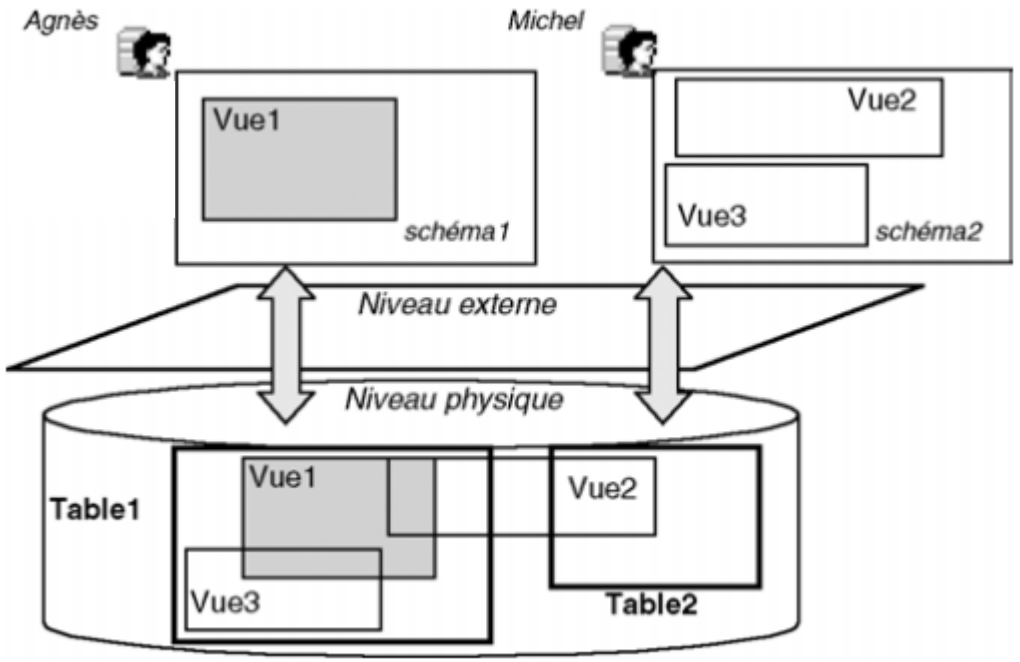
II - Les séquences

- Principes
- Création d'une séquence
- Manipulation d'une séquence

I. Les vues
Principes

- Une vue est une table virtuelle créée à partir d'autres tables grâce à une requête SELECT
  - ne nécessite aucune allocation en mémoire pour contenir les données
  - seule sa structure est stockée dans le dictionnaire de données
  - Elle se recharge chaque fois qu'elle est interrogée (avec SELECT)

CREATE VIEW *nomVue* **AS** *requêteSELECT* **;**



I. Les vues
Principes

- **Pourquoi les vues ?**
  - Réaliser le **niveau externe** des SGBD

    ➔ propose à l'utilisateur une perception de la BD plus **proche de ses besoins**, en termes de structures et de formats de données
  - Garantir une **indépendance logique** des programmes par rapport aux données

    ➔ le programme restera invariant aux modifications de schéma s'il accède à la BD via une vue
  - Définir un niveau additionnel **de sécurité**

    ➔ en restreignant l'accès à un sous ensemble de lignes et/ ou de colonnes
  - Permettre **certains contrôles d'intégrité**

    ➔ lorsqu'elles sont utilisées pour les mises à jour
  - Réduire la **complexité syntaxique des requêtes**

I. Les vues

Création d'une vue

CREATE [OR REPLACE] [FORCE] VIEW ***nomVue*** **[(*alias*$_1$, *alias*$_2$, …)]**

AS *requêteSELECT*

[WITH READ ONLY | WITH CHECK OPTION] ;

- **OR REPLACE :** remplace la vue par la nouvelle définition même si elle existait déjà
- **FORCE :** crée la vue sans vérifier si les tables qui l'alimentent existent ou si les privilèges (SELECT, INSERT, …) sur ces tables sont acquis par l'utilisateur qui crée la vue
- **alias :** nom de chaque colonne de la vue ➔ sert à masquer les noms des colonnes de l'objet source Quand un alias n'est pas présent la colonne prend le nom de l'expression renvoyée par la requête SELECT.
- **WITH READ ONLY :** déclare la vue non modifiable par INSERT, UPDATE, ou DELETE
- **WITH CHECK OPTION :** garantit que toute mise à jour de la vue par INSERT ou UPDATE s'effectuera conformément au prédicat contenu dans la requête de définition

I. Les vues

Exemples

```
VENTE(NUMV, #NUM_PROD, #NUM_FOUR, DATE_VENTE, QUANTITE, PRIX_VENTE)
PRODUIT(NUM_PROD, NOM_PROD, MARQUE, PRIX_U)
FOURNISSEUR(NUM_FOUR, NOM_FOUR, VILLE_FOUR, TEL_FOUR)
```

Créer une vue nommée Détail_Vente comportant :

- NUMÉRO DE VENTE
- NOM PRODUIT
- MARQUE DE PRODUIT
- NOM DU FOURNISSEUR
- VILLE DU FOURNISSEUR
- DATE DE VENTE
- QUANTITE
- PRIX DE VENTE

```
CREATE VIEW DETAIL_VENTE
AS SELECT
NUMV,NOM_PROD,MARQUE,NOM_FOUR,VILLE_FOUR,
DATE_VENTE, QUANTITE, PRIX_VENTE
FROM VENTE V, PRODUIT P, FOURNISSEUR F
WHERE V.NUM_PROD =P.NUM_PROD
AND V.NUM_FOUR = F.NUM_FOUR ;
```

I. Les vues

Exemples

```
VENTE(NUMV, #NUM_PROD, #NUM_FOUR, DATE_VENTE, QUANTITE, PRIX_VENTE)
PRODUIT(NUM_PROD, NOM_PROD, MARQUE, PRIX_U)
FOURNISSEUR(NUM_FOUR, NOM_FOUR, VILLE_FOUR, TEL_FOUR)
```

- Créer la vue VENTES_GROUPEES qui indique les nombres de ventes et les quantités totales vendues par produit et par date

```
CREATE VIEW VENTES_GROUPEES(NUM_PROD,DATE_VENTE,NBRE_VENTES,QTITE_TOT)
AS SELECT NUM_PROD, DATE_VENTE, COUNT(*), SUM(QUANTITE)
FROM VENTE
GROUP BY NUM_PROD, DATE_VENTE ;
```

Remarque : Une fois créée, une vue s'interroge comme une table

➢ `Select * from VENTES_GROUPEES ;`

```
CREATE VIEW VENTES_GROUPEES(NUM_PROD,DATE_VENTE,NBRE_VENTES,QTITE_TOT)
AS SELECT NUM_PROD, DATE_VENTE, COUNT(*), SUM(QUANTITE)
FROM VENTE
GROUP BY NUM_PROD, DATE_VENTE ;
```

VENTE

NUMV	NUM_PROD	NUM_FOUR	DATE_VENTE	QUANTITE	PRIX_VENTE
1001	P1	F1	20/10/2021	20	520
1022	P1	F1	20/10/2021	5	130
1013	P2	F4	20/10/2021	15	1500
1002	P2	F3	20/10/2021	10	900
1005	P2	F3	20/10/2021	5	475
1004	P3	F1	20/10/2021	13	260
1015	P1	F1	21/10/2021	7	145
1019	P1	F1	21/10/2021	5	120
1037	P1	F2	21/10/2021	2	50
…	…	…	…	…	…

Count(*) : 2
Sum(quantite) : 25

Count(*) : 3
Sum(quantite) : 30

Count(*) : 1
Sum(quantite) : 13

Count(*) : 3
Sum(quantite) : 14

I. Les vues

Exemples

VENTES_GROUPEES

NUM_PROD	DATE_VENTE	NBRE_VENTE	QTITE_TOT
P1	20/10/2021	2	25
P2	20/10/2021	3	30
P3	20/10/2021	1	13
P1	21/10/2021	3	14
…	…	…	…

- En utilisant la vue `VENTES_GROUPEES`, afficher le produit ayant réalisé le nombre de vente le plus élevé ainsi que la date correspondante

```
Select num_prod, date_vente  from VENTES_GROUPEES
where nbre_vente = (select max(nbre_vente)
                    from VENTE_GROUPEES);
```

I. Les vues
Vue simple modifiable

- Lorsqu'il est possible d'exécuter des instructions INSERT, UPDATE ou DELETE sur une vue, cette vue est dite modifiable (*updatable view*)
- Pour mettre à jour une vue, il doit exister une correspondance biunivoque entre les lignes de la vue et celles de la table source
- Pour qu'une vue simple soit modifiable, sa requête de définition doit respecter les critères suivants :

  • pas de directive DISTINCT, de fonction (AVG, COUNT, MAX, MIN, STDDEV, SUM, ou VARIANCE) dans le SELECT.

  • pas de GROUP BY - HAVING
- **Remarque :** Si un attribut, déclaré NOT NULL dans la table source, n'est pas repris dans la vue, aucun INSERT n'est possible sur cette vue

I. Les vues
Vue simple modifiable

- **Exemples**

Pilote (id_pilote, nom, adresse, nbhvol, compa#)

Create view ***Etat_civil*** *as select nom, adresse, nbhvol , compa from pilote ;*

- **Suppression :** DELETE FROM Etat_civil WHERE compa = 'ASO'; ✓
- **Modification :** UPDATE Etat_civil SET nbHVol = nbHVol*2 WHERE nom = 'Ali Bali'; ✓
- **Insertion :** INSERT INTO Etat_civil VALUES('Salah Ahmed', 'Sfax', 10,'TUN'); ➔ Erreur ✘

CREATE VIEW ***PilotesTN***
*AS SELECT * FROM pilote* ***WHERE compa = 'TUN' WITH CHECK OPTION ;***

- **Insertion :**

INSERT INTO PilotesTN VALUES ('PL-10', 'Ali Ali', 'SFAX', 10,**'SYPH') ;** ➔ Erreur ✘

ORA-01402: vue WITH CHECK OPTION - violation de clause WHERE

I. Les vues

Vue multitable modifiable

- Une table est dite protégée par sa clé si sa clé primaire est préservée dans la vue et peut jouer le rôle de clé primaire de la vue
- Pour qu'une vue multitable soit modifiable, sa requête de définition doit respecter les critères suivants :
  - La mise à jour (INSERT, UPDATE, DELETE) n'affecte qu'une seule table
  - Seuls des enregistrements de la table protégée peuvent être insérés. Si la clause WITH CHECK OPTION est utilisée, aucune insertion n'est possible
  - Seules les colonnes de la table protégée peuvent être modifiées
  - Seuls les enregistrements de la table protégée peuvent être supprimés

→ Afin de savoir dans quelle mesure les colonnes d'une vue sont modifiables (en insertion, en modification ou en suppression), il faut interroger la vue USER_UPDATABLE_COLUMNS du dictionnaire de données

I. Les vues

Vue multitable modifiable

Exemple : **COMPAGNIE** (ID_COMP, NOM_COMP, ADRESSE)

PILOTE (NUM_PIL, NOM_PIL, NBHVOL, #ID_COMPA)

- Créer la vue VPIL_PRO (NUM_PIL, NOM_PIL, ID_COMPA, NOM_COMP) qui renseigne sur les pilotes professionnels, ayant effectué plus que 100 heures de vols

```
create view  VPIL_PRO
as select num_pil,  nom_pil,  id_compa,  nom_comp
from pilote , compagnie
where id_compa = id_comp and nbhvol > 100 ;
```

- Pour cette vue la table protégée est la table Pilote

  ➔ Possibilité d'insertion, de mise à jour, de suppression dans la table Pilote, à travers la vue VPIL_PRO

  ➔ Pas d'insertion, de mise à jour, de suppression dans la table Compagnie, à travers la vue VPIL_PRO

I. Les vues

Vue multitable modifiable

Suite exemple :

```
select column_name, insertable, updatable, deletable
from user_updatable_columns
where table_name = 'V_PIL_PRO';
```

```
5  select column_name, insertable, updatable, deletable from user_updatable_columns where table_name = 'V_PIL_PRO';
```

Résultats | Expliquer | Décrire | SQL enregistré | Historique

COLUMN_NAME	INSERTABLE	UPDATABLE	DELETABLE
NUM_PIL	YES	YES	YES
NOM_PIL	YES	YES	YES
ID_COMPA	YES	YES	YES
NOM_COMP	NO	NO	NO

I. Les vues

Renommer / Supprimer une vue

- **Renommer une vue**

RENAME *nom_vue* **TO** *nouveau_nom* ;

- **Supprimer une vue**

DROP VIEW *nom_vue* **;**

- ***Exemple :***

```
drop view VPIL_PRO ;
```
➔ vue supprimée

- ***Remarques :***
  - La suppression d'une vue n'entraîne pas la perte des données qui résident toujours dans les tables
  - Une vue devient inutilisable si la table source utilisée dans sa définition n'existe plus

II - Les séquences

II. Les séquences

Principes

- Une séquence est un objet virtuel qui ne contient aucune donnée mais qui s'utilise pour générer automatiquement des valeurs numériques (NUMBER)
- Elles sont utiles pour composer des clés primaires de tables
- Les séquences sont gérées indépendamment des tables

II. Les séquences

Création de séquence

CREATE SEQUENCE *nomSéquence*

[INCREMENT BY *entier*]

[START WITH *entier*]

[{ MAXVALUE *entier* | NOMAXVALUE }]

[{ MINVALUE *entier* | NOMINVALUE }]

[{ CYCLE | NOCYCLE }] ;

- Si aucune option n'est précisée, la séquence commencera à 1 et augmentera sans fin (la limite réelle d'une séquence est de 10^{29}-1). En spécifiant seulement « INCREMENT BY –1 » la séquence créée commencera à -1 et sa valeur diminuera sans limites (la borne inférieure réelle d'une séquence est de - 10^{27}-1).
- **INCREMENT BY :** donne l'intervalle entre deux valeurs de la séquence (entier positif ou négatif mais pas nul). La valeur absolue de cet intervalle doit être plus petite que MAXVALUE-MINVALUE. L'intervalle par défaut est 1.
- **START WITH :** précise la première valeur de la séquence à générer. Pour les séquences ascendantes, la valeur par défaut est égale à la valeur minimale de la séquence. Pour les séquences descendantes la valeur par défaut est égale à la valeur maximale de la séquence.

II. Les séquences

Création de séquence

- **MAXVALUE :** donne la valeur maximale de la séquence (ne pas dépasser 10^{29}-1). Cette limite doit être supérieure ou égale à l'entier défini dans START WITH et supérieure à MINVALUE.
- **NOMAXVALUE :** (par défaut) fixe le maximum à 10^{29}-1 pour une séquence ascendante et à -1 pour une séquence descendante.
- **MINVALUE :** précise la valeur min de la séquence (ne pas dépasser la valeur -10^{27}-1). Cette limite doit être inférieure ou égale à l'entier défini dans START WITH et inférieure à MAXVALUE.
- **NOMINVALUE :** (par défaut) fixe le minimum à 1 pour une séquence ascendante et à la valeur -10^{27}-1 pour une séquence descendante.
- **CYCLE :** indique que la séquence doit continuer de générer des valeurs même après avoir atteint sa limite. Au-delà de la valeur maximale, la séquence générera la valeur minimale et incrémentera comme cela est défini dans la clause concernée. Après la valeur minimale, la séquence produira la valeur maximale et décrémentera comme cela est défini dans la clause concernée.
- **NOCYCLE** (par défaut) indique que la séquence ne doit plus générer de valeurs une fois la limite atteinte.

II. Les séquences

Manipulation d'une séquence

- Seules deux directives peuvent être appliquées à une séquence :
  - *nom_séquence*.**CURRVAL** : retourne la valeur courante
  - *nom_séquence*.**NEXTVAL** : incrémente la séquence et retourne la valeur obtenue

II. Les séquences

Exemples

- Créer une séquence masequence qui commence par 1000, incréméntée par 30, sans valeur maximale, sans cycle.
-
```
CREATE SEQUENCE masequence
START WITH 1000
INCREMENT BY 30
NOMAXVALUE
NOCYCLE ;
```
- `select masequence.nextval from dual ;` ➔ 1000
- `select masequence.nextval from dual ;` ➔ 1030
- `select masequence.nextval from dual ;` ➔ 1060
- `select masequence.currval from dual ;` ➔ 1060

II. Les séquences

Exemples

Film (numFilm, titre, genre, année)

Exemplaire (num_exemplaire, #numFilm)

- ```
  INSERT INTO film (numFilm, titre)
  VALUES(masequence.NEXTVAL, 'Kill Bill')
  ```
- ```
  INSERT INTO exemplaire (numExemplaire,numFilm)
  VALUES (290870, masequence.CURRVAL)
  ```