Cours Bases de données 2ème année IUT

Cours 1 : Vues et Index

Anne Vilnat
http://www.limsi.fr/Individu/anne/cours

Plan

Les Vues : définition et intérêt

Définition

- Une vue est une table virtuelle
- résultat d'une requête à laquelle on a donné un nom

Pour quoi faire?

Les vues permettent de

- Simplifier des requêtes complexes
- Résoudre des problèmes de confidentialité
- Garantir l'intégrité d'une base
- ...

Création

Pour créer une vue :

CREATE [OR REPLACE][FORCE|NO FORCE]
VIEW <nom_de_vue> [<liste_attributs>)]
AS <clause_select>
[WITH CHECK OPTION [CONSTRAINT <nom_de_contrainte>]]
[WITH READ ONLY]

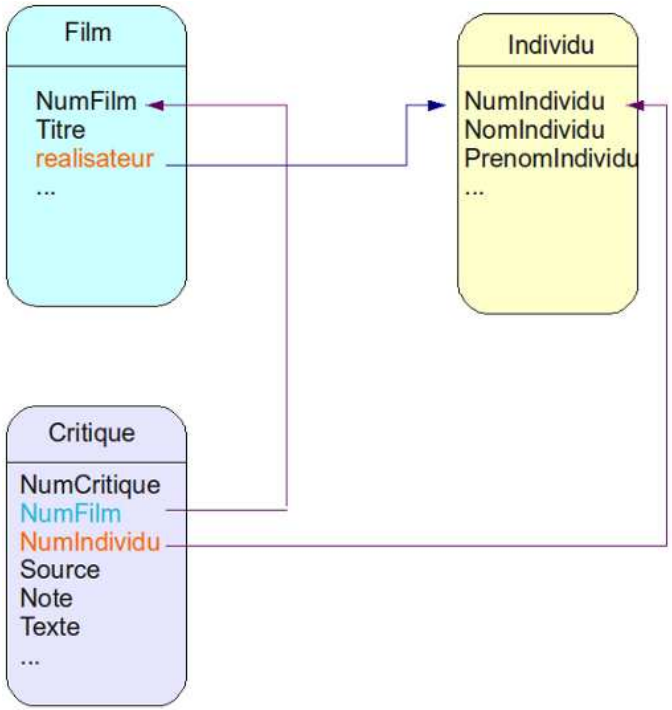
Et s'en servir :

SELECT ... FROM <nom_de_vue> WHERE ...

Le nom d'une vue

Utilisé partout où on peut mettre le nom d'une table :
SELECT, UPDATE, DELETE, INSERT, GRANT

Exemple




```
CREATE VIEW CritiquesLeMonde AS
SELECT * FROM Critique WHERE Source='Le Monde';
```

Suppression, renommage

Syntaxe :

- pour supprimer une vue :
  DROP VIEW <nom_de_vue>
- pour renommer une vue :
  RENAME <ancien_nom_de_vue> TO
  <nouveau_nom_de_vue>

Remarques

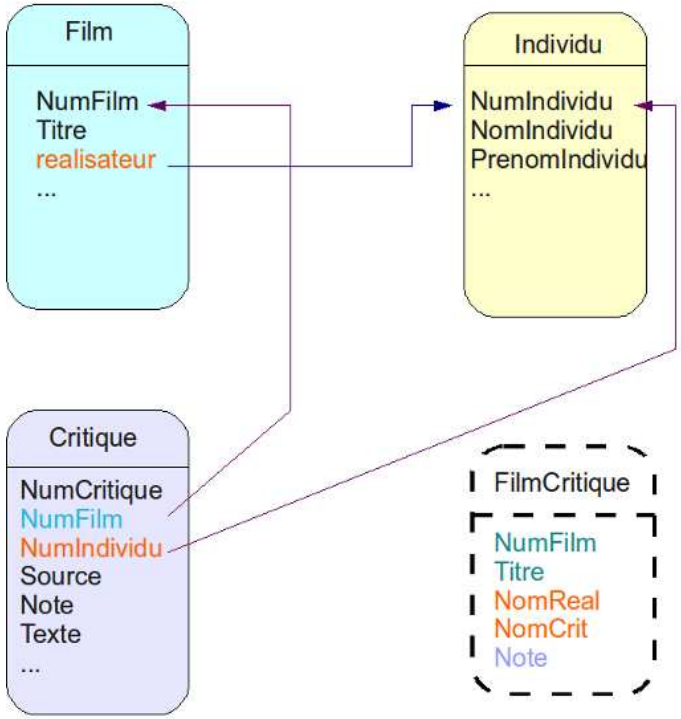
- Suppression d'une vue mais pas suppression des données
- Pour connaître les vues, on peut les retrouver dans les tables système :
  ALL-CATALOG, USER_VIEWS et ALL_VIEWS

Une vue : pour simplifier des requêtes complexes

Exemple: avoir facilement, pour un film, les critiques "en clair"

```
CREATE VIEW FilmCritique (NumFilm, Titre, NomReal,
NomCrit, Note) AS
            SELECT F.NumFilm, Titre, I1.NomIndividu,
I2.NomIndividu, Note
            FROM Film F, Individu I1; Individu I2, Critique C
            WHERE C.NumFilm = F.NumFilm
                AND C.NumIndividu = I2.NumIndividu
                AND F.NumRealisateur = I1.NumIndividu
```

Exemple



Exemple : on veut tout ce qui a été dit des films de Rohmer

```
SELECT NumFilm, Titre, NomCrit, Note
            FROM FilmCritique
            WHERE NomReal='ROHMER'
```

Une vue : pour assurer la confidentialité

permet de ne donner accès qu'à une partie des informations

Exemple:

```
CREATE VIEW IndividuPublic
                SELECT NumIndividu, NomIndividu,
PrenomIndividu
                FROM INDIVIDU;
```

Seul le créateur de IndividuPublic pourra avoir accès à des informations complètes, et il ne donnera accès aux autres qu'à la vue (donc aux noms et prénoms, mais pas adresse, téléphone,....

Une vue : pour assurer des contraintes d'intégrité

Exemple : éviter les critiques "vides"

```
CREATE VIEW CritiqueSansAvis AS
                SELECT * FROM Critique
                WHERE Texte IS NULL
```

Mise à jour :

```
UPDATE CritiqueSansAvis
                SET Texte='Sans interêt'
                WHERE Note = 0
```

Supprime des lignes de la vue "sans le dire"

Une vue : pour assurer des contraintes d'intégrité (2)

Impossible si :

Exemple:

```
CREATE VIEW CritiqueSansAvis AS
        SELECT * FROM Critique
        WHERE Texte IS NULL
        WITH CHECK OPTION CONSTRAINT CO_CRIT_SA
```

l'erreur CO_CRIT_SA sera alors déclenchée

Une vue : pour augmenter l'indépendance logique

permet de distinguer le point de vue logique (la vue) des tables physiques

Exemple:

Reprenons les critiques sur les films :

```
CREATE VIEW FilmCritique (NumFilm, Titre, NomReal,
NomCrit, Note) AS
            SELECT F.NumFilm, Titre, I1.NomIndividu,
I2.NomIndividu, Note
            FROM Film F, Individu I1; Individu I2, Critique C
            WHERE C.NumFilm = F.NumFilm
                AND C.NumCritique = I2.NumIndividu
                AND F.NumRealisateur = I1.NumIndividu
```

La vue logique de l'utilisateur ne change pas si une table physique change (par exemple, on décide de mettre les sources dans une table avec un code).

Conditions générales de mise à jour

UPDATE, DELETE, INSERT possibles si :

On peut retrouver la ou les lignes de la table originale concernée.
La vue ne doit pas contenir :

- Un opérateur ensembliste (UNION, EXCEPT, INTERSECT)
- Un opérateur DISTINCT
- Une fonction d'aggrégation comme attribut
- Une clause GROUP BY, ORDER BY ou CONNECT BY
- Une jointure (vue construite sur une seule table)

Conditions de mise à jour : cas particulier Oracle

Jointures possibles si il y a préservation de la clé.
Une table *préserve sa clé au travers d'une jointure* si celle-ci reste une clé de la table résultant de la jointure.

Exemple

```
CREATE OR REPLACE VIEW VueFilmsComplets AS
        SELECT numFilm, titre, NumIndividu as NumRealisateur,
nomIndividu as NomRealisateur, prenomIndividu as PreReal
        FROM Film, Individu
        WHERE Realisateur = numindividu
```

Possible

```
UPDATE VueFilmsComplets
        SET Titre = Upper(Titre)
```

Conditions de mise à jour : cas particulier Oracle (2)

Impossible

```
UPDATE VueFilmsComplets
        SET NomRealisateur = 'ROHMER'
        WHERE Titre = Upper('genou de Claire')
```

car NumIndividu ne pourrait pas être clef primaire de la vue

Impossible

```
INSERT INTO VueFilmsComplets
        (Numfilm, Titre, NomRealisateur, PreReal)
        VALUES (200000, Upper('Les Amours d'Astrée et de
Céladon'), 'ROHMER', 'ERIC')
```

Les attributs n'appartiennent pas à une seule table

Index : Généralités

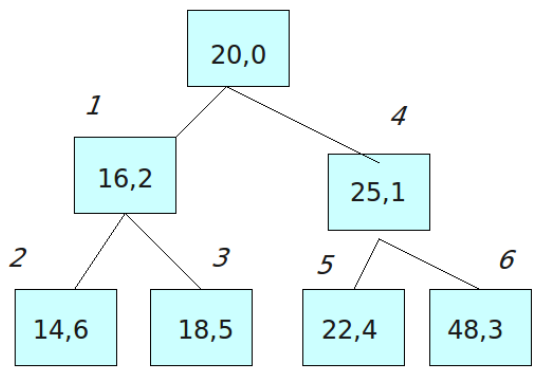
Pourquoi

- Pour vérifier des contraintes d'unicité : clé, contrainte explicite sur un attribut
  => index créés automatiquement (implicites)
- Pour accéder plus vite aux lignes d'une table à partir :
  - de la clé d'une table
  - d'autres attributs : index créés manuellement (explicites)
    oui... mais avec vigilance! car prend de l'espace disque

  => Indexer ce qui est nécessaire!

Index : Comment ça marche ?

Age	nom personne	prénom personne
20	Durand	Jules
25	Dupont	Alfred
16	Dupond	Ernest
48	Durant	Julie
22	Martin	Ernestine
18	Durand	Emilie
14	Martin	Zoé



1	4	20	0	2	3	16	2	-1	-1	14	6	-1	-1	18	5
5	6	25	1	-1	-1	22	4	-1	-1	48	3	...			

Syntaxe

Création d'un index

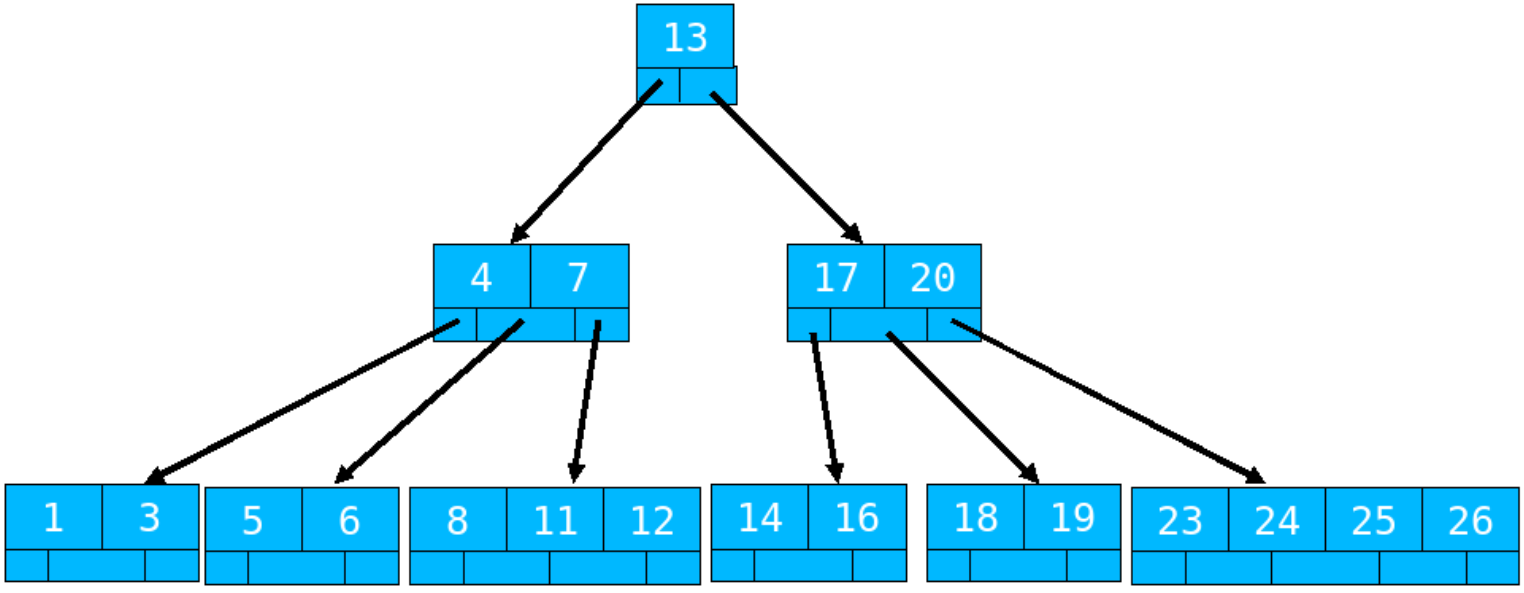
```
CREATE [UNIQUE | BITMAP] INDEX [<schema>.]<nom_index>
        ON <nom_de_table>
                (<nom_de_colonne> [ASC|DESC]
                [,<nom_de_colonne> [ASC|DESC]]
, ...)
CREATE INDEX IndNomIndividu
                ON Individu(nomIndividu);
```

Organisation de l'index

2 types d'index dans Oracle :

- arbres équilibrés (B-arbre) : toutes les branches de l'arbre ont la même longueur (mieux que les ABR...)
- Bitmap : si peu de valeurs sont possibles, une chaîne de bits pour chaque valeur

Exemple d'index : B-arbre d'ordre 5



Création d'un index

- Suite de clés : 3, 14, 7, 1, 8, 5, 11, 17, 13, 6, 23, 12, 20, 26, 4, 16, 18, 24, 25, 19
- Toute clé est atteinte en 2 accès maximum

Exemple d'index : Bitmap

Création d'un index

Une table contenant des élèves repérés par leur numéro (clé primaire), avec un attribut qui est leur classe : 1, 2 ou 3.

numeleve	...	numclasse	...
54		1	
18		2	
26		1	
30		3	
15		3	
12		2	
61		2	
44		1	

Exemple d'index : Bitmap (2)

numeleve	numclasse=1	numclasse=2	numclasse=3
12	0	1	0
15	0	0	1
18	0	1	0
26	1	0	0
30	0	0	1
44	1	0	0
54	1	0	0
61	0	1	0

Création d'un index

- Pour numclasse = 2, chaîne de bits : 10100001
- Peu de place occuppée

Pour aller plus loin : table organisée en index

Définition

Une table organisée en index maintient ses données triées sur la clé primaire.
Uniquement si la clé primaire constitue l'essentiel des attributs.

Création d'une table index

```
CREATE TABLE estActeurType
    numInd...
    numFilm...
     type CHAR CHECK type IN ('C', 'T', 'S', 'P')
     CONSTRAINT PKEAT PRIMARY KEY (numInd, numFilm)
     ORGANIZATION INDEX
```

Quand créer un index

- Attributs utilisés dans des jointures,
- Attributs servant souvent pour les sélections,
- Table de gros volume dont la majorité des interrogations sélectionne - de 15% des lignes,
- Bitmap : attribut à peu de valeurs distinctes,
- B-arbre : attribut à beaucoup de valeurs distinctes.

Quand ne pas créer un index

- Attributs souvent modifiés (index à recréer...),
- Table de petit volume,
- si requêtes sur NULL car les NULL, non stockés dans l'index. (ex : WHERE ... IS NULL).
- Dans certains dialectes SQL, quand la clause WHERE fait appel à une fonction (mais pas dans Oracle où c'est possible)

Exemple

La requête suivante peut utiliser l'index ci-dessous en Oracle, mais pas en SQL standard :

```
SELECT *
        FROM Individu
        WHERE UPPER(nomIndividu)='SPIELBERG'

CREATE INDEX individuNomMajuscule
        ON Individu (UPPER(nomIndividu))
```