Chapitre 6 - Vues (Views)

1. SQLDDL-DML- Vues
   - Définition et intérêt des vues
   - Création d'une vue
   - Suppression une vue
   - Représentation au niveau conceptuel et logique
   - Conclusion

Vue : définition

Une vue est une relation dont les données sont *dynamiquement* calculées par une requête.
Une vue peut être pensée comme

- une représentation personnalisée de données stockées dans une ou plusieurs relations
- une *requête sauvegardée* qui définit une relation dérivée.

Vue : Exemple

Table Employes

num	prenom	nom	annee
1	Marie	Smith	1970
2	Adrian	Johnsson	1960
3	Alizee	Jones	1988

Table Etudiants

num	prenom	nom	annee
100	Lauranne	Rodriguez	2000
200	Tom	Lee	2001

View Personnes *au 31 Décembre 2021*

num	prenom	nom	annee	age
1	Marie	Smith	1970	51
2	Adrian	Johnsson	1960	61
3	Alizee	Jones	1988	33
100	Lauranne	Rodriguez	2000	21
200	Tom	Lee	2001	20

View Personnes *au 31 Décembre 2025*

num	prenom	nom	annee	age
1	Marie	Smith	1970	55
2	Adrian	Johnsson	1960	65
3	Alizee	Jones	1988	37
100	Lauranne	Rodriguez	2000	25
200	Tom	Lee	2001	24

Le code SQL

ORACLE:

```
CREATE VIEW Personnes (num, prenom, nom, annee, age) AS
   SELECT num, prenom, nom, annee, TRUNC(MONTHS_BETWEEN(SYSDATE, annee)/12) AS age
   FROM Employes
   UNION
   SELECT num, prenom, nom, annee, TRUNC(MONTHS_BETWEEN(SYSDATE, annee)/12) AS age
   FROM Etudiants;
```

SQLITE:

```
CREATE VIEW Personnes (num, prenom, nom, annee, age) AS
   SELECT num, prenom, nom, annee, strftime('%Y', 'now') - annee AS age
   FROM Employes
   UNION
   SELECT num, prenom, nom, annee, strftime('%Y', 'now') - annee AS age
   FROM Etudiants;
```

Syntaxe

```
CREATE VIEW [IF NOT EXISTS] Nom_View [(liste de colonnes)]
AS
-- Une requête SQL
SELECT STATEMENT;
```

Autre exemple

/ <nu, no, p, d, a, nb> $\in$ *Adherents_View* $\Longleftrightarrow$ *l'adhérent avec numéro nu, nom no, prénom p et date de naissance d et age a, possède actuellement nb bateaux */*

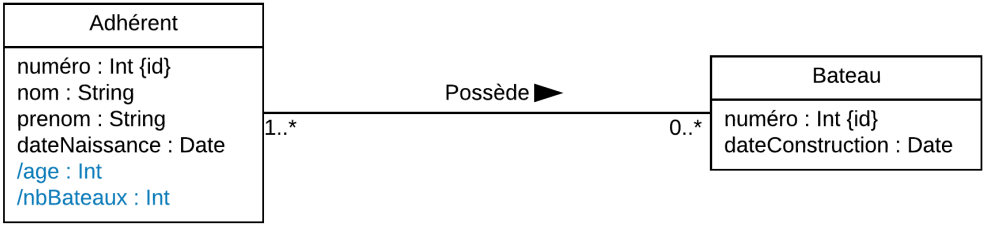
```
CREATE VIEW Adherents_View (
   numero_adherent,
   nom_adherent,
   prenom_adherent,
   date_naissance_adherent,
   age_adherent,
   nb_bateaux_adherent) AS
      SELECT
         numero_adherent,
         nom_adherent,
         prenom_adherent,
         date_naissance_adherent,
         -- attribut age calculé à partir de la date de naissance
         (strftime('%Y', 'now') - strftime('%Y', date_naissance_adherent))
         - (strftime('%m-%d', 'now') < strftime('%m-%d', date_naissance_adherent)) AS age_adherent,
         -- attribut nb bateaux calculé à graçe à un COUNT
         COUNT (numero_bateau) AS nb_bateaux_adherent
      FROM Adherents JOIN Proprietaires USING (numero_adherent)
      GROUP BY numero_adherent, nom_adherent, prenom_adherent, date_naissance_adherent;
```

Supprimer une vue

DROP VIEW V; supprime la vue V (mais pas les n-uplets des relations de base!)

```
DROP VIEW Adherents_View ;
```

Et au niveau conceptuel (UML) et logique (relationnel)?



Adherents_base (numero_adherent, nom_adherent, prenom_adherent, date_naissance_adherent)
Bateaux (numero_bateau, date_construction_bateau, nom_modele, numero_emplacement)
Proprietaires (numero_adherent, numero_bateau) /* suite à l'association many-to-many */

Proprietaires[numero_adherent] ⊆ Adherents_**base** [numero_adherent]
Proprietaires[numero_bateau] = Bateaux[numero_bateau]

(View) Adherents (numero_adherent, nom_adherent, prenom_adherent, date_naissance_adherent, **age_adherent, nb_bateaux_adherent**)

Le nouveau code de la vue sera :

```sql
-- Notez les changements de noms de la vue et la table de base
CREATE VIEW Adherents (
        numero_adherent,
        nom_adherent,
        prenom_adherent,
        date_naissance_adherent,
        age_adherent,
        nb_bateaux_adherent) AS
        SELECT
           numero_adherent,
           nom_adherent,
           prenom_adherent,
           date_naissance_adherent,
           -- attribut age calculé à partir de la date de naissance
           (strftime('%Y', 'now') - strftime('%Y', date_naissance_adherent))
           - (strftime('%m-%d', 'now') < strftime('%m-%d', date_naissance_adherent)) AS age_adherent,
           -- attribut nb bateaux calculé à partir graçe à un COUNT
           COUNT (numero_bateau) AS nb_bateaux_adherent
        FROM Adherents_base JOIN Proprietaires USING (numero_adherent)
        GROUP BY numero_adherent, nom_adherent, prenom_adherent, date_naissance_adherent;
```

Avantages

Les vues sont souvent utilisées pour :

- Cacher la complexité des données,
- Simplifier des requêtes,
- Présenter des données sous différents angles,
- Sauvegarder des requêtes complexes,
- Offrir un niveau supplémentaire de confidentialité

Comme dans une relation, les données associées à une vue peuvent être interrogées
Les modifications doivent se réaliser dans les relations de base de la vue