Le langage SQL

Base de donnes Cinéma

Considérons les tables suivantes :
Film(Titre, Réalisateur, Acteur)

Programme(NomCiné, Titre, Horaire)

Film contient des infos sur tous les films et Programme concerne le programme à Bordeaux

Le langage SQL

C'est un langage fourni avec tout SGBD relationnel commercialisé.

C'est un standard reconnu par l'ISO depuis 87 (standard⇒ portabilité)

On en est à la version 3

Structure de base

Une requête SQL typique est de la forme :

$$\begin{array}{ll} \textbf{SELECT} & A_1, \ldots, A_n \\ \textbf{FROM} & r_1, \ldots, r_m \\ \textbf{WHERE} & P \end{array}$$

Les A_i sont des attributs, les r_j sont des noms de relations et P est un prédicat.

Cette requête est équivalente à

$$\pi_{A_1,\ldots,A_n}\Big(\sigma_P(r_1 \times \cdots \times r_m)\Big)$$

La clause SELECT

La clause SELECT correspond à la projection de l'algèbre.

Les titres des films :

```
SELECT  Titre
FROM    film
```

L'utilisation de le symbôle $*$ permet de sélectionner tous les attributs

```
SELECT  *
FROM    film
```

La clause SELECT (suite)

SQL autorise par défaut les doublons. Pour le forcer à les éliminer, on utilise la clause **DISTINCT**

```
SELECT DISTINCT  Titre
FROM             film
```

La clause **SELECT** peut contenir des expressions arithmétiques ainsi que le renommage d'attributs

```
SELECT  Prix_HT * 1.206 AS Prix_TTC
FROM    produit
```

La clause WHERE

Correspond au prédicat de sélection dans l'algèbre.

La condition porte sur des attributs des relations qui apparaissent dans la clause **FROM**

```
SELECT DISTINCT  Titre
FROM             film
WHERE            Réalisateur = "Bergman"
                 AND Acteur = "Stewart"
```

SQL utilise les connecteurs **AND**, **OR** et **NOT**

Pour simplifier la clause **WHERE**, on peut utiliser la clause **BETWEEN**

```
SELECT  Num
FROM    compte
WHERE   Solde BETWEEN 0 AND 10000
```

La clause FROM

Elle correspond au produit cartésien de l'algèbre.

Le titre et le réalisateur des films programmés à l'UGC de Bordeaux.

```
SELECT  Titre, Réalisateur
FROM    film, programme
WHERE   film.titre=programme.titre AND
        programme.NomCiné= "UGC"
```

Les variables n-uplets

Elles sont définies dans la clause **FROM**

```
SELECT  Titre, Réalisateur
FROM    film AS f, programme AS p
WHERE   f.titre=p.titre AND p.NomCiné="UGC"
```

Soit Emp(Id,Nom,Id_chef)

```
SELECT  e1.Nom, e2.Nom AS Nom_Chef
FROM    emp e1, emp e2
WHERE   e1.Id_chef = e2.Id
```

La clause ORDER BY

SQL permet de trier les résultats de requête

SELECT	*
FROM	programme
WHERE	NomCiné=“UGC”
ORDER BY	Horaire **ASC**, Titre **DESC**

Opérateurs ensemblistes

SELECT ...
...
UNION/ INTERSECT/ EXCEPT
SELECT ...

Attention : Ces opérations éliminent les doublons, pour pouvoir les garder, utiliser à la place **INTERSECT ALL** ...

Si t apparaît m fois dans r et n fois dans s alors il apparaît

- $m+n$ fois dans r **UNION ALL** s
- $\min(m,n)$ fois dans r **INTERSECT ALL** s
- $\max(0,m-n)$ fois dans r **EXCEPT ALL** s

Les fonctions d'aggrégation

Ce sont des fonctions qui agissent sur des ensembles (multi-ensembles) de valeurs :

AVG : la valeur moyenne de l'ensemble
MIN : la valeur minimale
MAX : la valeur maximale
SUM : le total des valeurs de l'ensemble
COUNT : le nombre de valeur dans l'ensemble

Les fonctions d'aggrégation (suite)

SELECT COUNT(Titre)
FROM Programme

Cette requête retourne le nombre de films programmés à Bordeaux.

Attention : Un même titre peut être compté plusieurs fois s'il est programmé à des heures différentes et dans des salles différentes.

SELECT COUNT(**DISTINCT** Titre)
FROM Programme

Aggrégation et GROUP BY

Le nombre de films programmés dans chaque salle

```
SELECT NomCiné, COUNT(DISTINCT Titre)
FROM Programme
GROUP BY NomCiné
```

Les attributs apparaissant dans la clause **SELECT** en dehors des aggrégats *doivent* être associés à la clause **GROUP BY**

Aggrégats et la clause HAVING

Les salles où sont programmés plus de 3 films

```
SELECT NomCiné, COUNT(DISTINCT Titre)
FROM Programme
GROUP BY NomCiné
HAVING COUNT(DISTINCT Titre) > 3
```

Le prédicat associé à la clause **HAVING** est testé après la formation des groupes définis dans la clause **GROUP BY**

Requêtes imbriquées

SQL fournit un mécanisme qui permet d'imbriquer les requêtes.

Une sous requête est une requête SQL (SELECT-FROM-WHERE) qui est incluse dans une autre requête. Elle apparaît au niveau de la clause **WHERE** de la première requête.

Les films programmés à l'UGC non programmés au Triaon

SELECT Titre
FROM Programme
WHERE NomCiné="UGC" and Titre **NOT IN** (
    SELECT Titre
    FROM Programme
    WHERE NomCiné="Trianon")

Requêtes imbriquées (suite)

Compte(Num, Solde, NomTit)

Trouver les comptes dont les soldes sont supérieurs aux soldes des comptes de Durand

```
SELECT *
FROM Compte
WHERE Solde > ALL (
      SELECT Solde
      FROM Compte
      WHERE NomTit="Durand" )
```

En remplaçant **ALL** par **SOME**, on obtient les comptes dont les soldes son sup. au solde d'au moins un compte de Durand.

Requêtes imbriquées (suite)

Les cinémas qui passent tous les films programmés à l'UGC

```
SELECT NomCiné
FROM programme p1
WHERE NOT EXISTS (
      (SELECT DISTINCT Titre
      FROM programme
      WHERE NomCiné="UGC")
      EXCEPT
      (SELECT DISTINCT Titre
      FROM programme p2
      WHERE p1.NomCiné=p2.NomCiné))
```