Cours de base de données

SQL-Interrogation des bases de données

Par: Kamal BAL
Université AMOB de Bouira
Facultie des sciences et des sciences appliquées
Département d'informatique

Le langage de requêtes SQL

Le langage de requêtes SQL

1. Introduction
2. Langage déclaratif
3. Syntaxe générale d'une requête SQL
4. Syntaxe détaillée
   - 4.1. Expression de la projection
   - 4.2. Expression des restrictions
   - 4.3. Expression des jointures
   - 4.4. Expression des unions, intersections, différences
5. Travail sur le résultat d'une requête SQL
   - 5.1. Tri du résultat
   - 5.2. Effectuer un calcul sur le résultat
   - 5.3. Effectuer des sous-totaux
   - 5.4. Restrictions sur les sous-totaux
6. Exemple de requêtes

4 types de fonctions (briques) :

Extraction des données	• SELECT
Langage de définition des données LDD	• CREATE, ALTER, DROP , RENAME
Langage de manipulation des données LMD	• INSERT, UPDATE, DELETE, MERGE
Contrôle des transactions	• SAVEPOINT; COMMIT, ROLLBACK
Langage de contrôle des données	• GRANT, REVOKE

4 types de fonctions (briques) :

Extraction des données	• SELECT

Ce chapitre concerne uniquement l'instruction la brique : Extraction des données et l'instruction **SELECT**

Introduction

SQL permet de communiquer avec un SGBD

```
SELECT matriculate, nom
FROM   etudiant;
```

Etudiant	
Matricule	Nom
Q0001	Boussiga
Q0002	Chibane
Q0003	Kaidi
Q0004	Mosteghanemi

SQL est un Langage déclaratif

- SQL est un langage déclaratif :
  - L'utilisateur définit le résultat à obtenir à l'aide de prédicats (*QUOI*), le système optimise la procédure de recherche (*COMMENT*)
    - Par opposition à l'algèbre relationnelle qui introduit un ordre dans l'enchaînement des opérations
    - Par opposition aux langages impératifs : Basic, Pascal, C, Java ...

Syntaxe d'une requête SQL

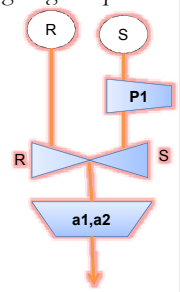
SELECT <liste des attributs du résultat>
FROM <liste des tables impliquées dans la requête>
WHERE <formule de sélection> ;

- La formule de sélection utilise les constituants (colonnes) des tables présentes dans la clause **FROM** ;
- La formule de sélection peut contenir :
  - Les opérateurs de comparaison : >,<,>=,<=,<>
  - Les opérateurs logiques : AND, OR, NOT
  - Les prédicats : ALL, ANY, EXISTS, BETWEEN, LIKE, IN (à combiner éventuellement avec NOT)
  - Des sous-requêtes

Syntaxe d'une requête SQL

- Correspondance entre le SQL et le langage algébrique

```
SELECT   a1,a2
FROM     R, S
WHERE    P1
         And R ⋈ S
```

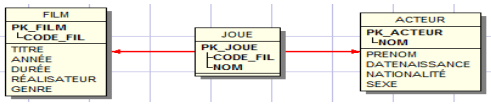


- **Sémantique**
  1. Faire le produit cartésien de toutes les tables de la clause FROM
  2. Appliquer à ce résultat tous les prédicats de la clause WHERE
  3. Afficher les colonnes de la clause SELECT

Syntaxe d'une requête SQL

- On se basera sur cet exemple de Schéma pour illustrer les commandes SQL:

FILM(CodeFilm, Titre, Année, Durée, Réalisateur, Genre)
ACTEUR(Nom, Prénom, DateNaissance, Sexe, Nationalité)
JOUE(*CodeFilm, NomActeur*)



Expression de la projection

Expression de la projection

- NomRelation.NomAttribut : permet de désigner un attribut
  - Ex : afficher les noms des Réalisateurs

```
SELECT Film.Réalisateur
FROM   Film ;
```

- NomAttribut : permet de désigner un attribut s'il n 'y a pas d'ambiguité sur sa provenance.

```
SELECT   Réalisateur
FROM     Film ;
```

réalisateur
Zucker
Hamina
Montecorvo
Hamina

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

Expression de la projection

- **DISTINCT** : permet d'éliminer les lignes en double dans le résultat. Ex : chaque Réalisateurs n'est affiché qu'une seule fois

```
SELECT  DISTINCT  Réalisateur
FROM  Film ;
```

réalisateur
Zucker
Hamina
Montecorvo

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

`DISTINCT` agit sur toute la ligne du résultat

Expression de la projection

- **'*' :** permet de désigner dans la clause SELECT tous les attributs d'une table
  - Ex : afficher la table Film

```
SELECT  Film.*
FROM  Film ;
```

ou

```
SELECT  *
FROM  Film ;
```

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

Expression de la projection

- **Expression d'un calcul** :
  - Permet de spécifier un calcul appliqué à une colonne
  - Ex : afficher les titres des Film et les deux derniers chiffre de leur année de parution

```
SELECT  Titre, Année MOD 100
FROM  Film ;
```

Titre	annee
Lancelo	99
Vent du sud	67
Batille d'Alger	67
Année de braise	65

Expression de la projection

- **Spécifier des alias de colonnes avec AS** : permet de spécifier l'entête d'une colonne affichée dans le résultat
  - **Ex** : afficher pour chaque film, la durée en heure (durée/60) avec l'alias '**durée en heures**'

```
SELECT  titre, duree/60 AS 'Durée en heures'
FROM  Film;
```

Titre	Durée en heures
Lancelo	2
Vent du sud	2
Batille d'Alger	2
Année de braise	3

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

Expression de la projection

- **Projection d'une constante** :
  - **Ex** : afficher réalisateur 'a réalisé ' titre

```
SELECT realisateur,' A réalisé', titre
FROM  Film
```

Réalisateur	A réalisé	Titre
Zucker, ,	A réalisé	Lancelo
Hamina	A réalisé	Vent du sud
Montecorvo,	A realisé	Bataille d'Alger
Hamina,	A réalisé	Années de braise

Expression des restrictions

Expression des restrictions

- Spécification d'un **Prédicat** dans la clause « **WHERE** »
- **Notion de Prédicat :**
  - Expression logique à laquelle on peut répondre par VRAI ou par FAUX pour chaque ligne de la table opérande.
  - Une restriction ne garde que les lignes pour lesquelles le prédicat est vrai.
- **Prédicat simple :**
  - NomRelation.NomAttribut θ Valeur
    ou
    NomAttribut θ Valeur
    θ est un comparateur : =, <, <=, >, >=, <>
  - Ex : titre des films de l'année 1998 ?

```
SELECT titre
FROM Film
WHERE année = 1998 ;
```


Expression des restrictions

- **Prédicat composé :**
  - Combinaison de prédicats reliés par les connecteur AND (ET), OR (OU)
  - L'utilisation de parenthèses peut être utile pour préciser comment combiner des prédicats reliés par des AND et des OR
  - **Ex** : titre des films policiers de l'année 1960 et de l'année 1968 ?

```
SELECT   Titre
FROM     Film
WHERE    (Année=1960 OR Année=1968)
         AND  Genre = 'policier';
```


Expression des restrictions

- **NOT : inverse le résultat d'un prédicat**
  - Ex : titre des films policier produits avant 1980 ?

```
SELECT    Titre
FROM      Film
WHERE     Année <1980 AND genre='Policier' ;
```

peut aussi s'écrire :

```
SELECT    Titre
FROM    Film
WHERE   NOT (Année >1980 AND genre<>'Policier') ;
```

- **BETWEEN :** Pour spécifier un prédicat comportant une intervalle
  - **Ex** : titre des films produits entre **1980** et **1983** ?

```
SELECT    Titre
FROM    Film
WHERE   Année BETWEEN 1980 AND 1983 ;
```

Expression des restrictions

- **Prédicat avec IN** : teste si la valeur d'une colonne est incluse dans un ensemble
  - Ex : titre des films produits entre 1980 et 1983 ?

```
SELECT    Titre
FROM    Film
WHERE   Année IN (1980, 1981, 1982, 1983) ;
```

  - Ex : Titre des films réalisés par : PARKER, MALLE, ou VERNEUIL ?

```
SELECT  Titre
FROM    Film
WHERE   Réalisateur IN
            ('PARKER','MALLE','VERNEUIL');
```

  - Ex : Films **qui n'ont** pas été réalisés par : PARKER, MALLE, ou VERNEUIL ?

```
SELECT Titre
FROM    Film
WHERE   Réalisateur NOT IN
                ('PARKER','MALLE','VERNEUIL') ;
```


Expression des restrictions

- **Prédicat avec LIKE :** Comparer la chaîne de caractère d'une colonne à un Modèle
  - Modèle : chaîne de caractère avec des **jokers** :
    '_' (souligné) → remplace tout caractère isolé
    '%' (pourcentage) → remplace Zéro, un ou plusieurs caractères
  - Ex : les films dont le nom du réalisateur contient la lettre 'P' en 2eme position ?

```
SELECT Titre
FROM    Film
WHERE   Réalisateur LIKE '_P%';
```

- **UPPER, LOWER :** transformer la chaîne de caractère d'une colonne majuscules ou en minuscules
  - Ex : titre des films de Malle ?

```
SELECT    Titre
FROM    Film
WHERE UPPER(Réalisateur) ='MALLE'
```


Expression des jointures

Expression des jointures

- S'évalue comme une restriction sur le produit cartésien des deux tables

```
SELECT  *
FROM    acteur, joue
```

Produire toutes les combinaisons possibles de acteur et de joue

- **Jointure classique en SQL :**
  - Exprimée par un prédicat dans la clause **WHERE**

NomRel1.NomAtt1 θ NomRel2.NomAtt2

θ est un comparateur : =, <, <=, >, >=, <>

Expression des jointures

- Ex : Afficher les réalisateurs avec qui les acteurs ont joué ?

```
SELECT Joue.NomActeur, Film.Réalisateur
FROM Film, Joue
WHERE Film.Codefilm = Joue.Codefilm;
```

Nom Acteur	Realisateur
Acteur 1	Zucker
Acteur 2	Zucker
ACteur3	Montecorvo
Acteur 4	Hamina

film

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

joue

NomActeur	code
Acteur 1	F1
Acteur 2	F1
ACteur3	F3
Acteur 4	F4

Expression des jointures

- Ex : Qui a joué les films réalisés par « Zucker»

```
SELECT Joue.NomActeur
FROM   Joue, Film
WHERE  Joue.Codefilm = Film.Codefilm
       AND Réalisateur = 'Zucker'  ;
```

NomActeur
Acteur 1
Acteur 2

film

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

joue

Nom Acteur	code
Acteur 1	F1
Acteur 2	F1
ACteur3	F3
Acteur 4	F4

Expression des jointures

- **IN** : expression d'une Jointure avec une sous-requête
  - Une sous-requête produit un ensemble de valeurs compatibles avec l'attribut de jointure de la relation dite externe (**RelExt.Attj IN (SELECT …)**)
  - Cet ensemble de valeurs est extrait parmi les valeurs de l'attribut de jointure de la relation dite interne (**(SELECT Attj FROM RelInt …)**)
  - Ex : Qui a joué les films réalisés par 'Hamina'?

```
SELECT NomActeur
FROM   Joue
WHERE  Joue.Codefilm IN
                (SELECT Film.Codefilm
                  FROM Film
                  WHERE Réalisateur='Hamina') ;
```

Code_film
F2
F4

```
WHERE  Joue.Codefilm IN (F2,F4)
```


Expression des jointures

- **EXISTS** : tester l'existence d'un lien via une sous-requête
  - Le prédicats EXISTS est VRAI si la sous-requête associée ramène **au moins** une ligne résultat.
  - La sous-requête est exécutée pour chaque ligne de la relation dite externe. A chaque sous-requête l'attribut de jointure de la relation externe est considéré comme une constante (restriction)
  - La sous-requête contient un prédicat de jointure entre la relation externe et la relation dite interne. Cette jointure est traitée comme une restriction.
  - **Ex : Qui a joué dans des films de durée supérieure à 120 mn?**

```
SELECT NomActeur
FROM   Joue
WHERE  EXISTS (SELECT *
                 FROM Film
                 WHERE duree>= 120
                 AND Film.Codefilm = Joue.Codefilm);
```


Expression des jointures

- **EXISTS** : tester l'existence d'un lien via une sous-requête

```
SELECT NomActeur
FROM   Joue
WHERE  EXISTS (SELECT *
                 FROM Film
                 WHERE duree>= 120
                 AND Film.Codefilm = Joue.Codefilm);
```

Pour nomActeur=Acteur 1

```
SELECT NomActeur
FROM   Joue
WHERE  EXISTS (SELECT *
                 FROM Film
                 WHERE duree>= 120
                 AND Film.Codefilm = F1);
```

Nom Acteur	code
Acteur 1	F1
Acteur 2	F1
ACteur3	F3
Acteur 4	F4

Cod	Titre	annee	réalisateur	duree
F1	Lancelo	1999	Zucker	120
F2	Vent du sud	1967	Hamina	120
F3	Batille d'Alger	1967	Montecorvo	120
F4	Année de braise	1965	Hamina	180

Expression des unions, intersections, différences

- **Composition de requêtes SQL :**
  - Opérateurs de composition (OC)
    - UNION
    - INTERSECT
    - MINUS
  - Syntaxe des requêtes composées

```
SELECT Colonnes-1
FROM …
OC
SELECT Colonnes-2
FROM … ;
```

    - Attention ! Les colonnes spécifiées dans Colonnes-1 et Colonnes-2 doivent être compatibles deux à deux

Expression des unions, intersections et différences

Expression des unions, intersections, différences

- Ex : noms et prénoms des acteurs qui n'ont joué ni dans des comédies ni dans des films policiers

```
SELECT Nom, Prénom
From acteur
MINUS
(     SELECT Nom, Prénom
      FROM    Acteurs , Joue , Film
      WHERE  Genre = 'Comédie'
        AND Film.CodeFilm = Joue.CodeFilm
        AND Joue.NomActeur = Acteur.Nom
 UNION
      SELECT Nom, Prénom
      FROM    Acteurs A, Joue J, Film F
      WHERE  Genre = 'Policier'
        AND Film.CodeFilm = Joue.CodeFilm
        AND Joue.NomActeur = Acteur.Nom
) ;
```


Expression des unions

- UNION : questions avec OU
  - Ex : noms et prénoms des acteurs qui ont joué dans des comédies ou dans des films policiers

```
SELECT Nom, Prénom
   FROM    Acteurs , Joue , Film
   WHERE  Genre = 'Comédie'
     AND Film.CodeFilm = Joue.CodeFilm
     AND Joue.NomActeur = Acteur.Nom
UNION
    SELECT Nom, Prénom
    FROM    Acteurs A, Joue J, Film F
    WHERE  Genre = 'Policier'
       AND Film.CodeFilm = Joue.CodeFilm
       AND Joue.NomActeur = Acteur.Nom
```


Expression des intersections

- INTERSECT : questions avec ET
  - Ex : Nom et prénom des acteurs qui ont joué dans des comédies et des policiers

```
SELECT Nom, Prénom
   FROM    Acteurs , Joue , Film
   WHERE  Genre = 'Comédie'
     AND Film.CodeFilm = Joue.CodeFilm
     AND Joue.NomActeur = Acteur.Nom
INTERSECT
   SELECT Nom, Prénom
   FROM    Acteurs A, Joue J, Film F
   WHERE  Genre = 'Policier'
      AND Film.CodeFilm = Joue.CodeFilm
      AND Joue.NomActeur = Acteur.Nom ;
```


Expression des différences

- MINUS : questions avec NE … PAS,
  - Ex : Nom et prénom des acteurs qui n'ont pas joué dans des comédies

```
SELECT Nom, Prénom
FROM Acteurs
MINUS
SELECT Nom, Prénom
   FROM    Acteurs , Joue , Film
   WHERE  Genre = 'Comédie'
     AND Film.CodeFilm = Joue.CodeFilm
     AND Joue.NomActeur = Acteur.Nom
```

Principe : pour **tester l'absence de lien** entre un acteur et une comédie, on sélectionne tous les acteurs sauf (moins) ceux qui sont **reliés** a un comédie (jointure)

Expression des différences

- NOT IN : La différence peut aussi s'exprimer à l 'aide du prédicat NOT IN
  - La sous-requête est exécutée une seule fois.
  - Le prédicats NOT IN est appliqué à toutes les lignes la relation externe avec l'ensemble de valeurs produit par la sous-requête.
  - ⇒ résultat = relation externe - sous-requête
  - ⇒ Ex : Nom et prénom des acteurs qui n'ont pas joué dans des comédies

```
SELECT Nom, Prénom
FROM Acteurs
WHERE    Nom NOT  IN (
          SELECT Nom
          FROM    Acteurs , Joue , Film
          WHERE  Genre = 'Comédie'
          AND Film.CodeFilm = Joue.CodeFilm
         AND Joue.NomActeur = Acteur.Nom ) ;
```


Tester l'absence de lien

- NOT EXISTS : tester l'absence de lien via une sous-requête
  - Le prédicats NOT EXISTS est VRAI si la sous-requête associée ne ramène aucune ligne résultat.
  - La sous-requête est exécutée pour chaque ligne de la relation dite externe. A chaque sous-requête l'attribut de jointure de la relation externe est considéré comme une constante (restriction)
  - La sous-requête contient un prédicat de **jointure entre la relation externe et la relation dite interne**. Cette jointure est traitée comme une restriction.
  - Ex : Qui n'a jamais joué avec Alain?

```
SELECT NomActeur
FROM   Joue
WHERE NOT EXISTS (
                  SELECT *
                  FROM Film
                  WHERE Réalisateur = 'Alain'
                  AND Film.Codefilm = Joue.Codefilm) ;
```


Travailler sur le résultat d'une requête SQL

Travailler sur le résultat d'une requête SQL

- **Travail sur le résultat d'une requête SQL**

Les clauses suivantes sont interprétées après la clause WHERE. Elles permettent d'effectuer un traitement supplémentaires avant d'afficher le résultat

- Tri du résultat
- Effectuer un calcul sur le résultat
- Effectuer des sous-totaux
- Restrictions sur les sous-totaux

Tri du résultat

- **ORDER BY** : permet de trier le résultat
  - Dernière clause d'une requête SQL.
  - Les colonnes sur lesquelles le résultat est trié sont désignées par leur nom ou leur n° d'apparition dans la clause SELECT
  - ASC : tri croissant, DESC : tri décroissant
  - Ex : afficher les pays dans l'ordre alphabétique

```
SELECT NomP
FROM   Pays
ORDER  BY NomP ASC ;
```

  - Ex : afficher pour chaque pays le ratio superficie/nombre d'habitants. Trier les pays par ordre décroissant de ce ratio.

```
SELECT  NomP AS Pays, Superficie/NbHab AS Occupation
FROM        Pays
ORDER BY 2 DESC, 1 ASC ;
```

Remarque : Le tri sur la colonne 1 indique que les pays ayant le même ration sont affichés dans l'ordre alphabétique

Effectuer un calcul sur le résultat

- Fonctions Agrégat : SQL propose 5 fonctions permettant d'effectuer un calcul sur une colonne du résultat
  - **SUM(Att)** effectue la somme des valeurs de la colonne Att
  - **MIN(Att)** restitue la plus petite valeur de la colonne Att
  - **MAX(Att)** restitue la plus grande valeur de la colonne Att
  - **AVG(Att)** effectue la moyenne des valeurs de la colonne Att
  - **COUNT(*)** compte le nombre de lignes du résultat.
    COUNT(DISTINCT (Att) compte le nombre de valeurs différentes dans la colonne Att

Remarque : l'utilisation de ces fonctions agrège le résultat de la requête en une seule ligne qui affiche le résultat des fonctions.

Effectuer un calcul sur le résultat

- Ex : afficher le nombre total de films ainsi que leur durée moyenne

FILM

CodeFilm	Durée	Réalisateur	Année
F1	2	Fellini	82
F2	2	Verneuil	83
F3	1h30	Fellini	80

```
SELECT Count(*) AS NbFilms, AVG(Durée) AS DuréeMoyenne
FROM  Film ;
```

NbFilms	DuréeMoyenne
3	1h50

Effectuer des sous-totaux

- **GROUP BY** : permet de partitionner le résultat de la clause **WHERE** en sous ensembles sur lesquels on peut appliquer des fonctions agrégat
  - Cette clause se place derrière la clause **WHERE**
  - Les lignes du résultat sont triées sur les colonnes indiquées dans la clause GROUP BY. Un nouveau sous-ensemble est créé à chaque changement de valeur dans les colonnes du tri
  - Les fonctions agrégat peuvent être appliquées aux sous-ensembles de valeurs des colonnes
  - Remarque : l'utilisation de cette clause agrège le résultat de la requête en une ligne par paquet
  - Lorsque la requête comporte une clause GROUP BY, la clause SELECT spécifie généralement les colonne de groupement plus des fonctions agrégat

Effectuer des sous-totaux

- Ex : afficher pour chaque acteur le nombre total de films dans lesquels il a joué

JOUE

CodeFilm	NomActeur
F2	Signoret
F2	Montant
F3	Mastroiani
F3	Montant

```
SELECT  NomActeur, Count(DISTINCT CodeFilm)
FROM   Joue
GROUP  BY NomActeur ;
```

=>

NomActeur	COUNT(DISTINCT CodeFilm)
Mastroiani	1
Montant	2
Signoret	1

Restrictions sur les sous-totaux

- **HAVING** : permet d'éliminer des sous-ensembles du résultat lorsqu'il est partitionné par une clause GROUP BY
  - Cette clause se place derrière la clause GROUP BY
  - Permet d'effectuer des restriction sur les résultats fonctions agrégat appliquées aux sous-ensembles de valeurs des colonnes
  - Lorsqu'un prédicats est évalué à FAUX, le sous-ensemble est éliminé

Restrictions sur les sous-totaux

- Ex : afficher le nom des acteurs qui ont joué dans au moins deux films

```
SELECT  NomActeur, COUNT(DISTINCT CodeFilm) as nb
FROM  Joue
GROUP BY NomActeur
HAVING  COUNT(DISTINCT CodeFilm)>=2 ;
```

NomActeur	nb
Montant	Montant

JOUE

CodeFilm	NomActeur
F2	Signoret
F2	Montant
F3	Mastroiani
F3	Montant

```
SELECT  NomActeur, Count(DISTINCT CodeFilm)
FROM   Joue
GROUP  BY NomActeur ;
```

NomActeur	COUNT(DISTINCT CodeFilm)
Mastroiani	1
Montant	2
Signoret	1