Bases de données SQL

DIU

Sofian MAABOUT

maabout@labri.fr

Au menu

- C’est quoi une Base de Données (BD) ?
- C’est quoi un Système de Gestion de Bases de Données (SGBD) ?
- Le cas particulier des BD’s *relationnelles*
  - Structure
  - Interrogation
    - Algèbre relationnelle (langage théorique)
    - **SQL (langage utilisé en pratique)**
  - Aspects “avancés”
    - Mise à jour des données
    - Définition/création d’une BD
    - Contraintes sur les données
    - Accès concurrents à une BD

SQL

- Structured Query Language ou Simple Query Language
- Inspiré de QueL et SEQUEL langages qui l'ont précédé
- Actuellement, le langage est standardisé
- Utilisé dans tous les SQBD's relationnels
  - Access, Postgres, Oracle, MySQL, SQLServer, …

- SQL permet de
  - Définir et créer une base de données
  - **Interroger et modifier le contenu d'une BD**
  - Interroger et modifier la structure d'une BD
  - Définir et modifier le stockage physique d'une BD
  - Définir et attribuer/retirer des droits à des utilisateurs de la BD
  - Définir la politique de gestions des accès concurrents
  - Définir des contraintes d'intégrité sur une BD
  - …

Requête SQL

- La forme générale d'une requête SQL est :

```
SELECT attributs
FROM   relations
[WHERE  condition]
```

Dans la clause SELECT, on mentionne les attributs que l'on veut retourner (afficher)

Pour FROM, on mentionne les relations que l'on utilise

Pour WHERE, qui est une clause facultative, on exprime une condition

Requêtes SQL

SELECT attributs	Correspond à la projection de l'algèbre
FROM relations	Correspond au produit de l'algèbre
[WHERE condition]	Correspond à la sélection de l'algèbre

L'ordre d'exécution est : FROM (produit) puis WHERE (sélection) puis SELECT (projection)

SELECT A,B FROM R1, R2 WHERE C=c1 est « équivalente » à

$\pi_{A, B}(\sigma_{C=c1}(R1 \times R2))$

SQL : particularités vs algèbre

- Les doublons ne sont pas automatiquement éliminés

- Pour SQL, l'attribut A de la table R s'appelle en réalité R.A

$\Rightarrow$ on peut toujours effectuer le produit de R1 par R2 car si l'attribut A est présent dans R1 et R2, alors ce sont R1.A et R2.A

SQL : Projection

Livre

N°Livre	Titre	NomAuteur	Editeur
137	La peste	Camus	Gallimard
11	Le monde selong Garp	Irving	Folio
46	L'assomoire	Zola	Livre de Poche
12	L'étranger	Camus	Gallimard

Les noms des auteurs

```
SELECT    NomAuteur
FROM      Livre
```

NomAuteur
Camus
Irving
Zola
Camus

Noter que Camus apparaît deux fois

SQL: Elimination des doublons

```
SELECT    DISTINCT NomAuteur
FROM      Livre
```

NomAuteur
Camus
Irving
Zola

La clause DISTINCT permet d'éliminer les doublons

```
SELECT    DISTINCT N°Livre, Editeur
FROM      Livre
```

N°Livre	Editeur
137	Gallimard
11	Folio
46	Livre de Poche
12	Gallimard

Gallimard apparaît deux fois. On raisonne au niveau des enregistrements: il n'y a pas de doublons

SQL: Sélection et projection

- Le Titre des livres d'Irving

```
SELECT      Titre
FROM        Livre
WHERE       NomAuteur = « Irving »
```

Titre
Le monde selong Garp

SQL: Sélection sans projection

- Les livres d'Irving

```
SELECT    *
FROM      Livre
WHERE     NomAuteur = « Irving »
```

* Remplace tous les attributs de ou des tables mentionnées dans FROM

N°Livre	Titre	NomAuteur	Editeur
11	Le monde selong Garp	Irving	Folio

SQL: Renommage

- Les titres d'Irving

```
SELECT    Titre AS TitreIrving
FROM      Livre
WHERE     NomAuteur=« Irving »
```

TitreIrving
Le monde selong Garp

SQL: Jointure et projection

- Prêt(N°Adhérent, N°Livre, N°Exemplaire, DatePrêt, DateRetour)
- Livre(N°Livre, Titre, NomAuteur, Editeur)

- Titre des Livres empruntés

```
SELECT  Titre
FROM    Livre, Prêt
WHERE Livre.N°Livre=Prêt.N°Livre
```

La condition « Livre.N°Livre=Prêt.N°Livre » exprime une jointure: une ligne de la table Livre n'est associée à une ligne de la table prêt que si elles portent sur le même numéro de livre.

Et si un livre a été emprunté plusieurs fois ?

SQL: Jointure et projection « partielle »

- Afficher les livres (tous leurs attributs) qui ont été empruntés au moins une fois.

```
SELECT    DISTICT Livre.*
FROM      Livre, Prêt
WHERE     Livre.N°Livre = Prêt.N°Livre
```

Livre.* désigne tous les attributs de la table livre

On ajoute le DISTINCT pour éviter d'afficher 100 fois un même livre s'il a été emprunté 100 fois.

SQL: Jointure, projection et sélection

- Afficher les titres des livres empruntés par l'adhérent 145

```
SELECT      DISTINCT Titre
FROM        Livre, Prêt
WHERE       N°Adhérent=145 AND Livre.N°Livre=Prêt.N°Livre
```

Noter que Titre et N°Adhérent ne sont pas précédés des nom des tables où ils se trouvent (Livre et Prêt respectivement). On n'est pas obligé car aucune confusion n'est possible.

Règle: on doit faire précéder par le nom de la table quand un même attribut se trouve dans 2 tables mentionnées dans FROM.

SQL: Union, Intersection, Différence

- Enseignants(N°SS, Nom, Salaire) Etudiants(N°SS, Nom, AnnéeEtude)

A l'université, certains étudiants sont aussi enseignants.

Le N°SS et le nom des étudiants qui sont enseignants

SELECT N°SS, NOM FROM Etudiant ← requête1 (N°SS, Nom)

INTERSECT

SELECT N°SS, NOM FROM Enseignant ← requête2 (N°SS, Nom)

SQL: Union, Intersection, Différence

- Les étudiants qui ne sont pas enseignants

```
SELECT N°SS, NOM FROM Etudiant
MINUS                                   ←certains systèmes utilisent EXCEPT
SELECT N°SS, NOM FROM Enseignant
```

- Les enseignants et les étudiants

```
SELECT N°SS, NOM FROM Etudiant
UNION
SELECT N°SS, NOM FROM Enseignant
```

SQL: Union, Intersection, Différence

- **Pour le cas spécifique des opérations ensemblistes, les doublons sont automatiquement éliminés (pour rester conforme au concept d'ensemble)**

- Pour tenir compte des doublons, il faut ajouter la clause ALL

SELECT N°SS, Nom FROM Etudiants

UNION ALL

SELECT N°SS, Nom FROM Enseignants

Quelqu'un qui est enseignant et en même temps étudiant, figurera deux fois dans le résultat

SQL: Requête imbriquée avec All

- Dans la clause WHERE on peut exprimer des conditions basées sur le résultat de requêtes

- Afficher les prêts les plus récents (leur date de prêt est la plus grande)

```
SELECT      *
FROM        Prêt
WHERE       DatePrêt >= ALL      (SELECT DatePrêt FROM   Prêt)
```

Un prêt sera affiché ssi sa date de prêt est supérieure ou égale à toutes les dates de prêt.

SQL: Requête imbriquée avec Some

- Les prêts qui ne sont pas les plus anciens

```
SELECT    *
FROM      Prêt
WHERE     DatePrêt > SOME (SELECT DatePrêt FROM Prêt)
```

Un prêt est retourné si sa date est strictement supérieure à celle d'au moins un prêt.

SQL: Requête imbriquée avec IN

- Les titres des livres qui ont été empruntés

```
SELECT      Titre
FROM        Livre
WHERE       N°Livre  IN (SELECT N°Livre FROM Prêt)
```

Le titre d'un livre est affiché ssi son numéro figure dans la table Prêt

Exo: reprendre les requêtes SQL qu'on a vues et les exprimer en algèbre.

SQL: Variable de relation

- Pour simplifier l'écriture de certaines requêtes, on définit des alias (variables) qu'on associe à des enregistrements d'une table.

- Titres des livres empruntés:

```
SELECT      Titre
FROM        Livre AS L, Prêt AS P
WHERE       L.N°Livre=P.N°Livre
```

- Pour certaines requêtes, ce n'est pas juste pour simplifier leur écriture mais on est obligé d'utiliser des variables

SQL: variable de relation

- Afficher les N°Livre qui ont été empruntés au moins deux fois par le même adhérent.

```
SELECT    DISTINCT P1.N°Livre
FROM      Prêt as P1, Prêt as P2
WHERE     P1.N°Livre = P2.N°Livre AND P1.N°Adhérent=P2.N°Adhérent AND
          P1.DatePrêt <> P2.DatePrêt
```

Un numéro de livre P1.N°Livre est affiché ssi il y a un autre prêt P2 avec le même numéro de livre, le même adhérent mais avec une date différente.

On ajoute le DISTINCT pour éviter d'afficher plusieurs fois le même N°Livre

SQL: Fonction d'agrégation

- SQL, contrairement à l'algèbre relationnelle, permet d'effectuer des calculs en utilisant des fonctions d'agrégation

- Une fonction d'agrégation prend une collection de valeurs et retourne une unique valeur

- Exemples: Max (retourne la valeur maximale), MIN, AVG(la moyenne), COUNT (retourne le nombre de valeurs)

SQL: COUNT

Afficher le nombre de livres dont la bibliothèque dispose:

```
SELECT    COUNT(N°Livre)
FROM      Livre
```

Cette requête va d'abord extraire l'ensemble de tous les numéros de livre puis applique la fonction COUNT à cet ensemble.

SQL: MAX

- Afficher la date du dernier prêt

```
SELECT MAX(DatePrêt)
FROM Prêt
```

- Afficher les prêts les plus récents

```
SELECT      *
FROM        Prêt
WHERE       DatePrêt = (SELECT MAX(DatePrêt) FROM Prêt)
```

SQL: GROUP BY

- Pour certaines requêtes, on veut
  - partitionner un ensemble d'enregistrements en sous-ensembles, puis
  - à chaque sous-ensemble, appliquer une fonction d'agrégation

- Pour chaque auteur, afficher son nom ainsi que le nombre de ses livres disponibles à la bibliothèque.

- On va d'abord regrouper les livres par auteur puis pour chacun des groupes, on compte le ombre de livres qu'il contient.

SQL: GROUP BY

```
SELECT    NomAuteur, COUNT(N°Livre)
FROM      Livre
GROUP BY  NomAuteur
```

Livre

N°Livre	Titre	NomAuteur	Editeur
137	La peste	Camus	Gallimard
11	Le monde selong Garp	Irving	Folio
46	L'assomoire	Zola	Livre de Poche
12	L'étranger	Camus	Gallimard

NomAuteur	Count(N°Livre)
Camus	2
Irving	1
Zola	1

SQL: GROUP BY

- Attention: mentionner un attribut dans la clause SELECT qui n'est pas dans GROUP BY est forcément une erreur

```
SELECT      NomAuteur, Editeur, COUNT(N°Livre)
FROM        Livre
GROUP BY    NomAuteur
```

SQL: GROUP BY avec HAVING

- Pour certaines requêtes, nous avons besoin de poser des conditions sur des groupes.

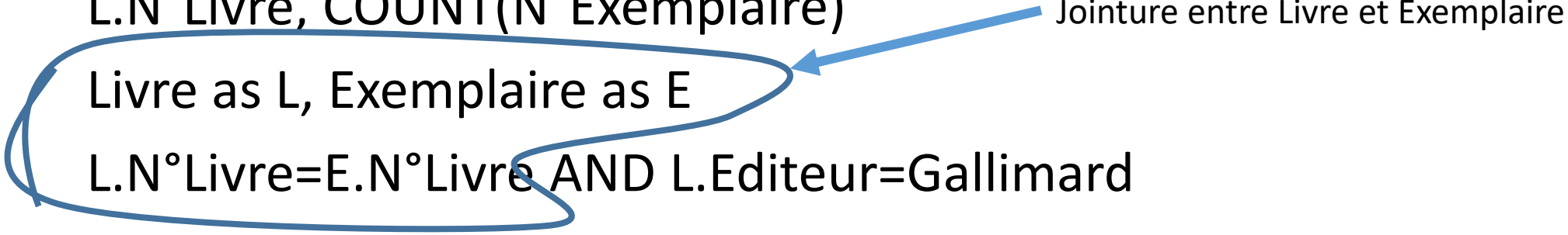
- Afficher le nom des auteurs qui ont au moins 2 livres à la bibliothèque.

```
SELECT      NomAuteur
FROM        Livre
GROUP BY    NomAuteur
HAVING      Count(N°Livre) > 1
```

SQL: HAVING vs WHERE

- La condition HAVING est évaluée sur des groupes entiers (il y a forcément un GROUP BY qui précède)
- La condition WHERE est évaluée sur des enregistrements individuels non pas des groupes.
- Ex: Pour chaque livre des éditions Gallimard, afficher son numéro ainsi que le nombre de ses exemplaires

```
SELECT    L.N°Livre, COUNT(N°Exemplaire)
FROM      Livre as L, Exemplaire as E
WHERE     L.N°Livre=E.N°Livre AND L.Editeur=Gallimard
GROUP BY  L.N°Livre
```



SQL vs Algèbre

- Les agrégats, le GROUP BY et le HAVING sont inexprimables en algèbre.
- Toute requête en algèbre peut être exprimée en SQL
- Par défaut, SQL garde les doublons sauf avec les opérations ensemblistes. L'algèbre est un langage ensembliste pur: pas de doublons.
- Pas de secret: maitriser SQL vient avec la pratique sur machine qui elle va soit dire « erreur de syntaxe » soit « retourne un résultat erroné due à une requête erronée.

Conclusion

- SQL est un langage déclaratif: on précise ce qui nous intéresse mais pas la manière dont le système doit procéder
  - Pas d'algorithme/programme.

- Le SGBD se charge de traduire la requête SQL en un programme *optimisé*

- D'autres aspects de SQL seront abordés au prochain cours.