PHP
Partie 4 - Le SQL

•••

Mathieu Nibas
B3 - ECV 2023

Le programme de cette partie

Etat des lieux

Base de données relationnelles

Langage de manipulation de données

Etat des lieux

Langage informatique normalisé servant à exploiter des **bases de données relationnelles**

Créé en 1974, il est depuis devenu incontournable

L'immense majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBD-R) du marché le supporte, **à quelques variations près**
** MySQL, Oracle Database, Microsoft Access, PostgreSQL, SQlite.*

Les choses bougent avec la tendance « NoSQL » (Not Only SQL)

Base de données relationnelles

L'information est organisée dans des tableaux à deux dimensions appelés des **relations** ou **tables**

- ➔ Les lignes de ces relations sont appelées des n-uplets ou enregistrements
- ➔ Les colonnes sont appelées des attributs

Base de données relationnelles

Une relation (au sens du modèle de Codd) est constituée de deux choses :

- ➔ Un Schéma : Le format de la table (incluant types et contraintes).
  Le schéma est fixé ;
- ➔ Une Extension : Le contenu de la table, qui est un ensemble de n-uplets dont l'ordre n'a pas d'importance, et sans doublons

Le qualificatif « relationnelle » d'une BDD ne fait ainsi aucunement référence au fait que les tables peuvent être reliées entre elles.

Base de données relationnelles

Dossard	Nom	Prenom	Club
503	Afond	Jocelyne	AP
123	Véloce	Marc	NULL
229	Le Goaër	Olivier	TDB
303	Speedy	Nadine	AP
271	Pressé	Marc	E64
404	Asphalte	Didier	NULL

Relation « Coureur »

Code	Intitule	Pro?
AP	Aigles de Pau	No
E64	Endurance 64	Yes
ASM	ASM Mourenx	No
TDB	Touristes du Béarn	No

Relation « Club »

Langage de manipulation de données

Langage de manipulation de données

Sous-ensemble du SQL utilisé pour ajouter, consulter, modifier, et supprimer des données des tables existantes

➔ Correspond aux commandes SQL: INSERT, SELECT, UPDATE, DELETE

Syntaxe générale d'une **requête** pour consulter des données :

```
SELECT attributs                -- ce que l'on cherche
FROM relations                  - - où on cherche
[WHERE condition]               - - avec quelles conditions
[GROUP BY attributs]            -- en utilisant
[HAVING condition]              - - des sous-ensembles
[ORDER BY attributs]            -- résultat dans quel ordre
```

Sélection d'attributs

Le mot clé SELECT vous permet de ne conserver que les attributs qui vous intéressent

```
SELECT Dossard, Nom                    --ou * si vous les voulez tous
```

Vous pouvez en profitez pour les renommer à l'affichage : les « alias »

```
SELECT Dossard, Nom as NomPatronimique
```

Vous pouvez appliquer des fonctions scalaires et des calculs sur les attributs

```
SELECT (Dossard * 3) + 1, LEFT (Nom, 3) as PetitNomPatronimique
```

Vous pouvez forcer l'élimination des doublons apparus potentiellement

```
SELECT DISTINCT Prenom
```

Table(s) Source(s)

Il faut évidemment définir d'où proviennent vos attributs avec le mot clé FROM

```
SELECT Dossard, Nom FROM Coureur
```

```
SELECT * FROM Club
```

Il faudra lever les ambiguïtés quand il y aura plusieurs tables en jeu

- ➔ Vous pouvez donner un alias à la table et s'y référer au niveau des attributs

  ```
  SELECT c1.Dossard, c1.Nom FROM Coureur as c1
  ```

- ➔ Ou vous pouvez utiliser une notation explicite en préfixant l'attribut par sa table d'appartenance

  ```
  SELECT Coureur.Dossard, Coureur.Nom FROM Coureur
  ```

Tri des résultats

Vous pouvez utiliser les attributs pour trier les résultats avec le mot clé ORDER BY

```
SELECT Dossard, Nom FROM Coureur ORDER BY Dossard
```

Vous pouvez choisir de trier dans l'ordre croissant (par défaut), ou décroissant

```
SELECT Dossard, Nom FROM Coureur ORDER BY Dossard ASC       --tri croissant

SELECT Dossard, Nom FROM Coureur ORDER BY Dossard DESC      --tri décroissant
```

Vous pouvez opérer le tri à plusieurs niveaux, en cas de valeurs identiques

```
SELECT Dossard, Nom FROM Coureur ORDER BY LEFT (Nom, 3), Dossard ASC
```

Filtrage des résultats

Vous pouvez éliminer les n-uplets qui ne vous intéressent pas avec le mot clé WHERE

➔ Dis autrement, vous cherchez à conserver ceux qui satisfont une condition logique déterminée

```
SELECT Dossard, Nom FROM Coureur WHERE Dossard > 102
```

Vous pouvez exprimer des conditions d'inégalité, de ressemblance ou de nullité

```
SELECT Dossard, Nom FROM Coureur WHERE Nom = "Le Goaër"

SELECT Dossard, Nom FROM Coureur WHERE Nom like "*Go??r"      --Cf. RegExp

SELECT Dossard, Nom FROM Coureur WHERE Club is NULL
```

Vous pouvez construire des conditions complexes à l'aide de connecteurs logiques

```
SELECT Dossard, Nom FROM Coureur WHERE (not (LEFT (Nom, 3) = "Le ") or (Dossard >= 50 and Dossard <= 300))
```

Jointure (interne)

Sert à reconstituer l'information éclatée entre deux tables, à l'aide du mot clé JOIN

➔ Produit cartésien : combinaison de chaque n-uplet d'une table avec ceux de l'autre table

➔ Un critère spécial doit être satisfait pour les apparier deux à deux : la condition de jointure

```
SELECT * FROM Coureur INNER JOIN Club ON Coureur.club = Club.code
```

La condition de jointure après le mot clé ON n'est pas forcément basique ni naturelle

```
SELECT * FROM Coureur INNER JOIN Club ON LEFT (Coureur .nom, 3) = Club. code
```

Ce mécanisme est généralisable à n tables, mais exige une rigueur syntaxique

```
SELECT * FROM (((T1 INNER JOIN T2 ON Cond1)
        INNER JOIN T3 ON Cond2)
                INNER JOIN T4 ON Cond3)
```

Jointure (Externe)

Parfois, ne pas réussir à reconstituer l'information éclatée entre deux tables, est une information en elle-même ! *
La non-satisfaction du critère de jointure fait apparaître des valeurs nulles (NULL) sur des attributs appariés, qui pourront par exemple servir à filtrer

Une table parmi les 2 est considérée comme table de référence lors de cette jointure, selon qu'elle se trouve à gauche ou à droite dans le sens d'écriture de votre requête

```
SELECT * FROM Coureur RIGHT OUTER JOIN Club ON Coureur.Club = Club.code WHERE Coureur. Dossard is NULL
```

```
SELECT * FROM Club LEFT OUTER JOIN Coureur ON Coureur. Club = Club.code WHERE Coureur. Dossard is NULL
    --100% équivalent à ci-dessus
```

Groupes et fonctions de groupes

L'ensemble des n-uplets forme un groupe par défaut, sur lequel il est possible d'appliquer des « fonctions de groupes »

- ➔ Il s'agit de fonctions spéciales d'agrégation statistiques
- ➔ Fonctions natives : AVG(), COUNT(), MIN(), MAX(), SUM(), STDEV(), VAR()

```
SELECT MAX(Dossard) FROM Coureur
```

Vous pouvez explicitement former des sous-groupes, à plusieurs niveaux, avec GROUP BY

```
SELECT Club, COUNT(Dossard) FROM Coureur GROUP BY Club
```

La sélection peut contenir tout au plus les attributs ayant servis à grouper, mais autant de fonctions de groupes que vous le souhaitez.

Groupes et fonctions de groupes

Si un filtrage doit porter sur une fonction de groupe, elle ne peut figurer derrière WHERE.

Elle est introduite par la clause HAVING

```
SELECT Club, COUNT(Dossard) as Inscrits FROM Coureur GROUP BY Club HAVING COUNT(Dossard) > 2
```

Il est fréquent d'appliquer des alias aux fonctions de groupe.

Toutefois, ces alias ne peuvent pas être référencés depuis le HAVING.

Sous-requête

Les expressions à l'intérieur d'une requête peuvent être remplacée par...une requête

```
SELECT * FROM Coureur WHERE Dossard < (SELECT MAX(Dossard) FROM Coureur)
```

Si la sous-requête ne renvoie pas une valeur unique mais des n-uplets, alors il faut utiliser les opérateurs IN, EXISTS, ANY, ALL

```
SELECT * FROM Coureur WHERE Club IN (SELECT Code FROM Club WHERE Pro?=Yes)
```

Parfois, la sous-requête peut être corrélée : elle ne s'exécute pas indépendamment

- ➔ Ses valeurs dépendent alors de celles de la requête hôte,
- ➔ Les alias de tables sont incontournables pour lever les ambiguïtés.

Exécution des requêtes

SQL est un langage **déclaratif**: il dit ce qu'il faut faire, mais pas comment le faire !

C'est le SGBD-R qui choisira la façon optimale d'exécuter votre requête. Néanmoins, l'ordre logique de traitement est grosso modo le suivant :

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. HAVING
7. SELECT
8. DISTINCT
9. ORDER BY