Introduction à la SGBD

Base de données & SQL

COLLEGE DE PARIS SUPÉRIEUR DAKARTECH

Dr Pape Abdoulaye **BARRO**

Enseignant – Chercheur

Spécialiste en Télémétrie et Systèmes Intelligents

Plan

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d’une base de données
- La sécurité des données

Objectif

Dans ce chapitre, nous aborderons les concepts et approches du langages SQL.

- Nous présenterons également les quelques opérations regroupées en 3 catégories de familles :
  - L'interrogation et la recherche dans les tables;
  - La gestion de tables et de vues;
  - La manipulation de données.

Le langage SQL
concepts

- Le langage SQL est un langage, dite assertionnel, permettant de manipuler des *bases de données relationnelles*. Il est issu des résultats du groupe de travail *System-R* qui travaillait sur la réalisation pratique des concepts de l'*approche relationnelle* chez IBM.
- C'est, en effet, une évolution du langage **SEQUEL**, qui est lui-même dérivé du langage de recherche **SQUARE**.
- Aujourd'hui, le langage SQL est normalisé [ISO89, ISO92] et constitue le *standard d'accès aux bases de données relationnelles*.
  - Le **SQL1**[ISO89] correspondant au norme de base, a été accepté en 1989. Il permet l'expression des requêtes composées d'opérations de l'algèbre relationnelle et d'agrégats. *En plus des fonctionnalités de définition, de recherche et de mise à jour, il comporte aussi des fonctions de contrôle qui n'appartiennent pas vraiment au modèle relationnel, mais qui sont nécessaires pour programmer des applications transactionnelles*.
  - Le **SQL2** [ISO92] a été adoptée en 1992, et est sous-divisé en trois niveaux, respectivement entrée, intermédiaire et complet.
    - *Le niveau entrée peut être perçu comme une amélioration de SQL1, alors que les niveaux intermédiaire et complet permettent de supporter totalement le modèle relationnel avec des domaines variés, tels date et temps.*
  - Le **SQL3** quant à lui a été normalisé en 1999, et est essentiellement constitué d'un ensemble de propositions nouvelles traitant plus particulièrement des *fonctionnalités objets et déductives*.

Le langage SQL

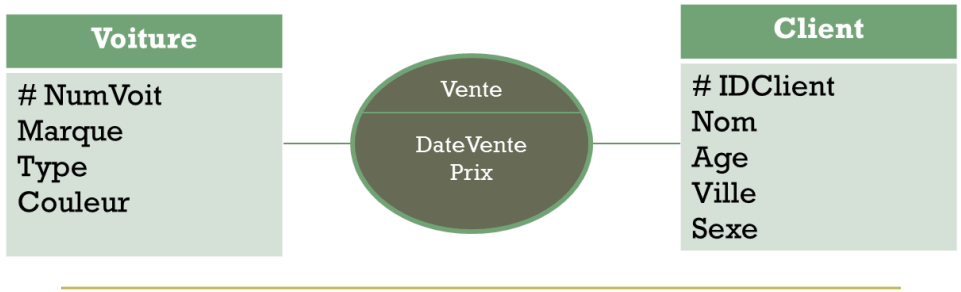
concepts

De manière générale, SQL utilise des critères de recherche (encore appelés qualifications) construits à partir de la logique ***des prédicats du premier ordre*** qui comportent quatre opérations de base à savoir:

- **L'insertion** (mot clé INSERT) permet d'ajouter des tuples dans une relation;
- **La recherche** (mot clé SELECT) permet de retrouver des tuples ou parties de tuples vérifiant la qualification citée en arguments;
- **La suppression** (mot clé DELETE) permet de supprimer d'une relation les tuples vérifiant la qualification citée en argument ;
- **La modification** (mot clé UPDATE) permet de mettre à jour les tuples vérifiant la qualification citée en argument à l'aide de nouvelles valeurs d'attributs ou de résultats d'opérations arithmétiques appliquées aux anciennes valeurs.

Opérations relationnelles avec SQL

- Nous allons créer et manipuler la base de données ‘casse’ ci-dessous:



Modèle entité-association

NumVoit	Marque	Type	Type
1	Peugeot	404	Rouge
2	Citroen	SM	Noire
3	Opel	GT	Blanche
4	Peugeot	403	Blanche
5	Renault	Alpine A310	Rose
6	Renault	Floride	Bleue

voiture

DateVente	Prix	NumVoit	NumAch
1985-12-03	10 000	1	1
1996-03-30	70 000	2	4
1998-06-14	30 000	4	1
2000-04-02	45 000	5	2

vente

NumAch	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

client

Les relations ou tables avec les enregistrements

- **Outils de travail:**
  - Wampserver
  - HeidiSQL (gestionnaire de session)

Créer une base de données MySQL ?

Pour créer une base de données et des utilisateur MySQL, il est possible de le faire soit par ligne de commande ou via une interface graphique.

- En ligne de commande,
  - il va d'abord falloir se connecter en tant qu'utilisateur root:
    - mysql -u root –p # entrer le mot de passe et puis appuyer sur ENTRER.
    - (Taper \q pour **quitter** le programme mysql)
  - Il faut ensuite créer un utilisateur de base de données en tapant la commande suivante:
    - CREATE USER 'username'@'localhost' IDENTIFIED BY 'password'
  - Pour lui accorder tous les privilèges, il faut exécuter la commande suivante:
    - GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost'
    - FLUSH PRIVILEGES ;
  - Pour afficher les privilèges d'un utilisateur on fait:
    - SHOW GRANTS FOR 'username'@'localhost'

Créer une base de données MySQL ?

- Il est possible de révoquer tous les privilèges d'un utilisateur non-root en faisant:
  - REVOKE ALL PRIVILEGES ON *.* FROM 'username'@'localhost';
- Ou certains privilèges:
  - REVOKE TYPE_DE_PERMISSION ON database.table FROM 'username'@'localhost';
- Ou supprimer entièrement un utilisateur:
  - DROP USER 'username'@'localhost';

Les principaux TYPE_DE_PERMISSION sont:

- **CREATE** – Permet aux utilisateurs de créer des bases de données/tables
- **SELECT** – Permet aux utilisateurs de récupérer des données
- **INSERT** – Permet aux utilisateurs d'ajouter de nouvelles entrées dans les tables
- **UPDATE** – Permet aux utilisateurs de modifier les entrées existantes dans les tables
- **DELETE** – Permet aux utilisateurs de supprimer les entrées de la table
- **DROP** – Permet aux utilisateurs de supprimer des bases de données/tables entières
- …

Créer une base de données MySQL ?

- On pourra alors se connecter à MySQL en tant qu'utilisateur (que nous venez de créer), en tapant la commande suivante:
  - mysql -u username –p
- Pour connaitre les bases disponibles, on utilise la commande:
  - SHOW DATABASES;
- Pour créer une base de données, on doit taper la commande suivante:
  - CREATE DATABASE dbname;
- Oubien:
  - CREATE DATABASE IF NOT EXISTS dbname ; # pour éviter des erreurs

  # ici *dbname* sera remplacée par notre base de données (*casse*)
- Pour pouvoir travailler avec la nouvelle base de données, on doit taper:
  - USE dbname;

Nous serons maintenant capable de créer des *tables* de la base et d'y insérer des *données*.

Gestion de tables et de vues

Les TYPES

Notre base de données ainsi créer est vide. Il faut donc y **ajouter des tables** (ou objets).

- Chaque table est composée de plusieurs champs et chacun de ces champs doit avoir un *type* pour spécifier la nature de la donnée qui sera stockée dans ce champ (*chiffres*, *texte*, *etc*.). Parmi les types, nous avons principalement:
  - INTEGER (contiendra des chiffres numériques sous forme de nombre entier « 32bits »);
  - SMALLINT (nombre entier «16 bits»);
  - FLOAT (contiendra des chiffres décimaux);
  - BOOLEAN (Il ne peut stocker que les valeurs true(vrai) ou false(faux)) ;
  - CHAR(n) (contiendra une chaîne de caractères de longueur « n »);
  - VARCHAR(n) (Chaîne de caractères de longueur maximale « n »);
  - DATE (pour les dates);
  - TIME[(n)] (pour les heures, n (optionnel) est le nombre de décimales représentant la fraction de secondes);
  - BLOB (Binary Large Object) pour stocker tout type binaire (photo, fichier texte, …);
  - …

Gestion de tables et de vues
CONTRAINTES D'INTÉGRITÉ

Nous avons eu à voir la notion de « domaine », dans la partie conceptualisation, qui permet de décrire l'ensemble des valeurs que peut prendre un attribut. Le langage SQL permettra de définir ces conditions plus finement lors de la création de la table. Une première approche du domaine étant établie lors du choix du type de la colonne.

- On distingue différents types de contraintes sur les colonnes:
  - les propriétés générales comme l'unicité:
    - La valeur de la colonne doit être renseignée absolument (NOT NULL). Si une telle propriété n'est pas renseignée sur une colonne, cela veut dire qu'elle peut ou pas contenir de données (NULL). Elle peut alors contenir « 0 » pour une colonne de type « entier » ou un espace pour une colonne de type « caractère ».
    - La valeur doit être unique comparée à toutes les valeurs de la colonne de la table (UNIQUE);

    Lorsque les deux conditions précédentes sont réunies, la colonne peut servir à identifier un enregistrement et constitue donc une « clé candidate ». La clé sera désignée en SQL par le mot clé PRIMARY KEY.

Gestion de tables et de vues
CONTRAINTES D'INTÉGRITÉ

- les restrictions d'appartenance à un ensemble ;

Il s'agit de décrire le domaine dans lequel la colonne pourra prendre ses valeurs. Un ensemble peut être décrit :

- En donnant la liste de tous ses éléments constitutifs (IN). L'ensemble des jours de la semaine ne peut être exprimé que de cette manière : « lundi », « mardi », etc.
  - ✓ On vérifie que la colonne 'couleur' ne peut prendre que des valeurs « normalisées » : 'Rouge','Vert' ou 'Bleu':
    - CHECK (Couleur in ('Rouge','Vert','Bleu'));
- Par une expression (>, < , BETWEEN…). Par exemple, le prix doit être supérieur à 1 000.
  - ✓ On vérifie que l'âge est compris entre 1 et 80 :
    - CHECK (Age BETWEEN 1 AND 80).
- Par une référence aux valeurs d'une colonne d'une autre table (REFERENCES).
  - ✓ On vérifie que les valeurs identifiantes des personnes 'NumAch' et des voitures 'NumVoit' de la table 'vente' existent bien dans les tables de référence 'personne' et 'voiture':
    - NumAch INT NOT NULL REFERENCES personne(NumAch),
    - NumVoit INT NOT NULL REFERENCES voiture(NumVoit),
    - PRIMARY KEY (NumAch, NumVoit)

Gestion de tables et de vues
CONTRAINTES D'INTÉGRITÉ

- les dépendances entre plusieurs colonnes (contrainte de table):
  - Lorsque l'on désire exprimer des contraintes plus élaborées impliquant plusieurs colonnes, on peut définir une contrainte de table en utilisant le mot clé CONSTRAINT.
    - ✓ On vérifie que la colonne 'Age' et la colonne 'Ville' doivent être renseignées ou vides en même temps:
      - CONSTRAINT la_contrainte CHECK ((Age IS NOT NULL AND Ville IS NOT NULL) OR (Age IS NULL AND Ville IS NULL))

Gestion de tables et de vues

CREATION DE TABLES

Lors de la création de table, il faut donc définir le type de données, la clé, les index éventuels et les contraintes de validation éventuelles garantissant la bonne qualité des informations entrées.

- Syntaxe :
  - CREATE TABLE <Nom de la table> (liste des colonnes avec leur type séparé par ,) ;

```
CREATE TABLE voiture (
   NumVoit INT PRIMARY KEY,
   Marque CHAR(40) NOT NULL,
   Type CHAR(30),
   Couleur CHAR(20),
   CHECK (Couleur in ('Rouge','Vert','Bleu'))
) ;
```

```
CREATE TABLE personne (
   NumAch INT PRIMARY KEY,
   Nom CHAR(40) NOT NULL,
   Ville CHAR(40),
   Age INT NOT NULL,
   CHECK (Age BETWEEN 1 AND 80)
);
```

```
CREATE TABLE vente (
   DateVente DATE,
   Prix INT,
   NumAch INT NOT NULL REFERENCES personne(NumAch),
   NumVoit INT NOT NULL REFERENCES voiture(NumVoit),
   PRIMARY KEY (NumAch, NumVoit)
) ;
```

- Le nom de la table ou d'une colonne ne doit pas dépasser 128 caractères. Il commence par une lettre, contient des chiffres, des lettres et le caractère « _ ».
- SHOW tables (permet d'afficher toutes les tables présentes dans notre base).
- SHOW COLUMNS FROM le_nom_de_ma_table (permet d'afficher le schéma de chaque table)

Gestion de tables et de vues

CREATION DE VUES

- Une « vue » est le résultat d'une requête que l'on peut manipuler de la même façon qu'une table. C'est une sorte de table dynamique dont le contenu peut être exprimé à chaque utilisation.
- Elle est utilisée juste par commodité car, soit:
  - parce qu'il n'est pas nécessaire que certains utilisateurs voient le modèle complet qui peut parfois être complexe;
  - pour restreindre l'accès à certains données (pour des raisons de sécurité/confidentialité).
    - CREATE VIEW personne_bis (NumAch, Nom, Age) AS SELECT NumAch, Nom, Age FROM personne ;

Gestion des données
INSERTION (*INSERT INTO*)

On dispose classiquement de trois opérations pour gérer les données d'une table: l'insertion, la suppression et la mise à jour.

- Pour insérer des données dans une table, ont utilise la formule générale suivante :
  - INSERT INTO <nom_de_la_table> [liste_des_colonnes] VALUES <liste_des_valeurs>
  - ➢ Par exemple, pour l'insertion d'un enregistrement dans la table 'voiture', on peut avoir :
  - INSERT INTO voiture (NumVoit, Marque, Type, Couleur) VALUES (1,'Peugeot',404,'Rouge');

Gestion des données
INSERTION (*INSERT INTO*)

NumVoit	Marque	Type	Type
1	Peugeot	404	Rouge
2	Citroen	SM	Noire
3	Opel	GT	Blanche
4	Peugeot	403	Blanche
5	Renault	Alpine A310	Rose
6	Renault	Floride	Bleue

voiture

NumAch	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

personne

DateVente	Prix	NumVoit	NumAch
1985-12-03	10 000	1	1
1996-03-30	70 000	2	4
1998-06-14	30 000	4	1
2000-04-02	45 000	5	2

vente

Gestion des données
SUPPRESSION, MODIFICATION

- L'opération de suppression permet de supprimer un ensemble d'enregistrements (lignes) que l'on identifiera avec la close WHERE :
  - DELETE FROM voiture WHERE Couleur='Rouge' ;
  - si l'on ne spécifie aucune condition, tous les enregistrements sont supprimés.
  - DELETE FROM personne ;
- Pour effectuer la mise à jour, il faut préciser les colonnes concernées, les nouvelles valeurs et les enregistrement pour lesquels on modifiera ces valeurs. on identifiera les enregistrements concernés avec la close WHERE:
  - UPDATE personne SET Ville='Paris-Centre' WHERE Ville='Paris' ;

Interrogation dans les tables
Projection

- L'opération de projection consiste à sélectionner la (les) colonne(s) dans une table. On spécifie la liste des colonnes après l'instruction SELECT en les séparant par des virgules. Si l'on désire afficher toutes les colonnes, on les désigne par le caractère «*».

  - SELECT Nom, Ville FROM personne ;

Nom	Ville
Nestor	Paris
Irma	Lille
Henri	Paris
Josette	Lyon
Jacques	Bordeaux

  - SELECT * FROM personne ;

NumAch	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

  - Les colonnes peuvent être renommées par le mot clé AS:
  - SELECT Ville AS City FROM personne ;

City
Paris
Lille
Paris
Lyon
Bordeaux

Interrogation dans les tables
Projection

- Il est possible d'afficher les valeurs distinctes d'une colonne Afin d'éliminer les doublons éventuels. on fait précéder le nom de la colonne par le mot clé DISTINCT:

- SELECT DISTINCT Marque FROM voiture ;

Marque
Peugeot
Citroen
Opel
Renault

- Il est aussi possible d' utiliser des expressions pour créer une colonne (vue que l' on ne stocke pas dans une table ce qui peut être calculé). Les opérateurs suivants peuvent nous permettre d'arriver à ces résultats.

+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

- SELECT Prix, DateVente, (Prix / 6.5596) AS Prix_Euros FROM vente ;

Prix	DateVente	Prix_Euros
10 000	1985-12-03	1 524.483 200
70 000	1996-03-30	10 671.382 401
30 000	1998-06-14	4 573.449 601
45 000	2000-04-02	6 860.174 401

Interrogation dans les tables
Projection

- SQL dispose de nombreuses autres fonctions intégrées, parfois dépendantes du SGBD utilisé, pour permettre par exemple de traiter des colonnes de types caractères, date…

- SELECT UPPER(Nom) AS NomMajuscule FROM personne ;

NomMajuscule
NESTOR
IRMA
HENRI
JOSETTE
JACQUES

- SELECT MONTH(DateVente) AS Mois FROM vente ;

Mois
12
3
6
4

- Les colonnes peuvent être constituées de résultats de fonctions statistiques intégrées à SQL. Voici une liste (non exhaustive) des opérateurs statistiques de SQL:

COUNT	Comptage du nombre d'éléments (lignes) de la table
MAX	Maximum des éléments d'une colonne
MIN	Minimum des éléments d'une colonne
AVG	Moyenne des éléments d'une colonne
SUM	Somme des éléments d'une colonne

- SELECT AVG(Prix) AS Prix_Moyen FROM vente ;

Prix_Moyen
38 750.000 0

- SELECT COUNT(*) AS Nombre_Personne FROM personne ;

Nombre_Personne
5

Interrogation dans les tables

SÉLECTION OU RESTRICTION (*WHERE*)

- L'opération de sélection (ou restriction) consiste à indiquer un ou plusieurs critères (sur le contenu des colonnes) pour choisir les lignes à inclure dans la table « résultat ». Cette critère de sélection est indiqué à la suite du mot clé WHERE et est constitué d'expressions de conditions composées (des opérateurs de comparaison et des connecteurs logiques). Ci-dessous:
  - la liste des opérateurs de comparaison:

=	Égal
<>	Différent
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

BETWEEN <valeur> AND <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères

- SELECT * FROM vente WHERE Prix > 50 000 ;

DateVente	Prix	NumVoit	NumAch
1996-03-30	70 000	2	4

- SELECT * FROM voiture WHERE Couleur IN ("Blanc","Rouge") ;

NumVoit	Marque	Type	Couleur
1	Peugeot	404	Rouge

Interrogation dans les tables

SÉLECTION OU RESTRICTION (*WHERE*)

- la liste des connecteurs logiques:

AND	Et : les deux conditions sont vraies simultanément
OR	Ou : l'une des deux conditions est vraie
NOT	Inversion de la condition

- SELECT * FROM voiture WHERE Couleur="Blanche" OR Marque="Peugeot" ;

NumVoit	Marque	Type	Couleur
1	Peugeot	404	Rouge
3	Opel	GT	Blanche
4	Peugeot	403	Blanche

- SELECT * FROM personne WHERE NOT (Ville='Paris') ;

NumAch	Nom	Age	Ville	Sexe
2	Irma	20	Lille	F
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

Interrogation dans les tables

AGRÉGATS OU GROUPAGE (*GROUP BY*)

- Les opérations d'« agrégation » ou de « groupage » regroupent les lignes d'une table par valeurs contenues dans une colonne. Pour réaliser cette opération, on utilise le mot clé GROUP BY suivi du nom de la colonne sur laquelle s'effectue l'agrégat.

  - SELECT Marque FROM voiture GROUP BY Marque ;

Marque
Citroen
Opel
Peugeot
Renault

  - On applique généralement des opérations de type statistique sur les « sous-tables » ainsi créées.

  - SELECT Marque, COUNT(*) AS Compte FROM voiture GROUP BY Marque ;

Marque	Compte
Citroen	1
Opel	1
Peugeot	2
Renault	2

Interrogation dans les tables

REQUÊTES SUR PLUSIEURS TABLES

- Lorsque l'on utilise plusieurs tables dans une requête SQL, il peut exister une ambiguïté dans les expressions sur les noms de colonnes. Comme deux tables peuvent avoir des noms de colonne identique, il est dans ce cas permis de lever cette ambiguïté en préfixant le nom de la colonne par le nom de la table en question.

  - SELECT voiture.Marque, voiture.Couleur FROM voiture ;

Marque	Couleur
Peugeot	Rouge
Citroen	Noire
Opel	Blanche
Peugeot	Blanche
Renault	Rose
Renault	Bleue

  - Il est commode de désigner la table par un alias pour peut être réduit son nom ou autres. Pour cela, il suffit simplement de mettre à la suite du nom de la table le mot clé AS.

  - SELECT Vo.Marque, Vo.Couleur FROM voiture AS Vo ;

Marque	Couleur
Peugeot	Rouge
Citroen	Noire
Opel	Blanche
Peugeot	Blanche
Renault	Rose
Renault	Bleue

Interrogation dans les tables

REQUÊTES SUR PLUSIEURS TABLES

- Produit cartésien:
  - SELECT * FROM personne, voiture ;
- Jointure interne (INNER JOIN):
  - SELECT voiture.Marque, voiture.Couleur, vente.Prix FROM voiture, vente WHERE voiture.NumVoit=vente.NumVoit ;
  - ➢ Oubien par un opérateur de jointure spécifique JOIN:
  - SELECT voiture.Marque, voiture.Couleur, vente.Prix FROM vente INNER JOIN voiture ON voiture.NumVoit=vente.NumVoit ;

Marque	Couleur	Prix
Peugeot	Rouge	10 000
Citroen	Noire	70 000
Peugeot	Blanche	30 000
Renault	Rose	45 000

  - SELECT vo.Marque, vo.Couleur, ve.Prix, pe.Nom, pe.Age FROM voiture AS vo JOIN vente AS ve JOIN personne AS pe ON (vo.NumVoit=ve.NumVoit) AND (pe.NumAch=ve. NumAch);

Marque	Couleur	Prix	Nom	Age
Peugeot	Rouge	10 000	Nestor	96
Citroen	Noire	70 000	Josette	34
Peugeot	Blanche	30 000	Nestor	96
Renault	Rose	45 000	Irma	20

Interrogation dans les tables

REQUÊTES SUR PLUSIEURS TABLES

- Jointure externe (OUTER JOIN): L'opération de jointure interne ne permet pas de répondre à des questions du type : « Quelles sont les voitures qui n'ont pas été vendues ? » . Il s'agit des 'voiture' qui n'ont pas de correspondance dans la table « vente ».
  - Cette opération n'est pas symétrique: soit on inclut toutes les lignes d'une table, soit toutes celles de l'autre. On précise cela à l'aide des mots clés LEFT et RIGHT ou en inversant simplement l'ordre des tables dans l'expression de l'instruction de jointure.
  - SELECT voiture.NumVoit, vente.NumVoit, voiture.Marque, voiture.Couleur, vente.Prix FROM voiture LEFT OUTER JOIN vente ON voiture.NumVoit=vente.NumVoit ;

NumVoit	NumVoit	Marque	Couleur	Prix
1	1	Peugeot	Rouge	10 000
2	2	Citroen	Noire	70 000
3	NULL	Opel	Blanche	NULL
4	4	Peugeot	Blanche	30 000
5	5	Renault	Rose	45 000
6	NULL	Renault	Bleue	NULL

  - SELECT voiture.NumVoit, vente.NumVoit, voiture.Marque, voiture.Couleur, vente.Prix FROM vente LEFT OUTER JOIN voiture ON voiture.NumVoit=vente.NumVoit ;

NumVoit	NumVoit	Marque	Couleur	Prix
1	1	Peugeot	Rouge	10 000
2	2	Citroen	Noire	70 000
4	4	Peugeot	Blanche	30 000
5	5	Renault	Rose	45 000

Interrogation dans les tables

TRI DU RÉSULTAT D'UNE REQUÊTE

- On utilise le mot clé ORDER BY pour spécifier la (les) colonne(s) sur laquelle (lesquelles) on souhaite trier le résultat.

Marque	Type
Citroen	SM
Opel	GT
Peugeot	404
Peugeot	403
Renault	Alpine A310
Renault	Floride

  - SELECT Marque, Type FROM voiture ORDER BY Marque ;

  - Il est possible de préciser l'ordre de tri par les mots clés ASC (croissant par défaut) ou DESC (décroissant).

Prix	DateVente
70 000	1996-03-30
45 000	2000-04-02
30 000	1998-06-14
10 000	1985-12-03

  - SELECT Prix, DateVente FROM vente ORDER BY Prix DESC ;

  - On peut indiquer plusieurs critères de tri, qui sont lus et traités de gauche à droite (ici, on trie d'abord par villes puis par âges).

Nom	Age	Ville
Jacques	50	Bordeaux
Irma	20	Lille
Josette	34	Lyon
Henri	45	Paris
Nestor	96	Paris

  - SELECT Nom, Age, Ville FROM personne ORDER BY Ville, Age ;

FIN

Feedback: pape.abdoutaye.barro@gmail.com

COLLEGE DE PARIS SUPÉRIEUR
DAKARTECH

Dr Pape Abdoulaye **BARRO**
Enseignant – Chercheur
Spécialiste en Télémétrie et Systèmes Intelligents