



Requêtage des données avec les instructions DQL de base

420-1B2-SW Développement avec bases de
données



Manipulation des données

- Ces commandes/instructions SQL sont principalement classées en cinq catégories :
- **DDL (Data Definition Language)** Langage de définition de données
- **DQL (Data Query Language)** Langage de requête de données
- **DML (Data Manipulation Language)** Langage de manipulation de données
- **DCL (Data Control Language)** Langage de contrôle des données
- **TCL (Transaction Control Language)** Langage de contrôle des transactions



Requêtage des données

Instruction SELECT

L'instruction **SELECT** est l'instruction DQL qui permet de récupérer des enregistrements à partir d'une table.

```
SELECT [ALL|DISTINCT]
    { [NomTable.|AliasDeTable.]*|NomColonne|Expression
    [AS AliasDeColonne][, ...n]}
FROM NomTable [AliasDeTable]
{[ORDER BY NomColonne, [, ...n]]}
```



Requêtage des données

Instruction SELECT

Exemples :

```
SELECT * FROM Employees
```

EmployeeID	LastName	FirtsName	...
1	Davolio	Nancy	...
2	Fuller	Andrews	...
3	Leverling	Janet	...

```
SELECT firstName, lastName FROM Employees
```

firstName	lastName
Nancy	Davolio
Andrews	Fuller
Janet	Leverling



Requêtage des données

Instruction SELECT

Il est possible de spécifier le titre que possédera une colonne en appliquant un alias à l'aide du mot-clé **AS** :

```
SELECT firstName AS Prénom, lastName AS Nom  
FROM Employees
```

Prénom	Nom
Nancy	Davolio
Andrews	Fuller
Janet	Leverling



Requêtage des données

Instruction SELECT

L'utilisation de la clause **DISTINCT** permet de limiter les enregistrements en supprimant les doublons :

```
SELECT Country  
FROM customers
```

Country
Canada
USA
Canada
Canada

```
SELECT DISTINCT Country  
FROM customers
```

Country
Canada
USA



Requêtage des données

Instruction **SELECT**

L'utilisation de la clause **ORDER BY** permet de spécifier l'ordre de tri des enregistrements :

```
SELECT EmployeeID, LastName, FirstName, Country  
FROM employees  
ORDER BY LastName, FirstName
```

EmployeeID	LastName	FirtsName	Country
5	Buchanan	Steven	UK
8	Callahan	Laura	USA
1	Davolio	Nancy	USA



Requêtage des données

Instruction **SELECT**

Il est possible de spécifier les colonnes de tri de la clause **ORDER BY** en indiquant leur position ordinal où 1 représente la première colonne.

```
SELECT EmployeeID, LastName, FirstName, Country  
FROM employees  
ORDER BY 2, 3
```




Requêtage des données

Instruction **SELECT**

Les clauses **ASC** et **DESC** permettent de spécifier l'ordre de tri ascendant ou descendant respectivement :

```
SELECT ProductID, ProductName, UnitPrice  
FROM Products  
ORDER BY UnitPrice DESC
```

ProductID	ProductName	UnitPrice
38		263.5000
29		123.7900
9		97.0000



Requêtage des données

Instruction **SELECT**

La clause **DESC** est d'ailleurs très utile pour trier des dates de la plus récente à la plus ancienne :

```
SELECT OrderID, OrderDate, ShipName  
FROM Orders  
ORDER BY OrderDate DESC
```

OrderID	OrderDate	ShipName
11074	1998-05-06 00:00:00.000	Simons bistro
11075	1998-05-06 00:00:00.000	Richter Supermarkt
11076	1998-05-06 00:00:00.000	Bon app'



Requêtage des données

Instruction SELECT

Les colonnes utilisées pour trier n'ont pas nécessairement à apparaître dans les colonnes retournées.

```
SELECT OrderID, ShipName FROM Orders  
ORDER BY OrderDate DESC
```

Vous ne pouvez utiliser la clause **ORDER** avec des colonnes de type text.



Requêtage des données

La clause WHERE

Adjointe à l'instruction **SELECT**, la clause **WHERE** permet de filtrer les enregistrements sur la base d'un ou plusieurs critères.

```
SELECT {*|NomColonne|Expression [, ...n]}  
FROM NomTable  
WHERE <condition>
```



Requêtage des données

La clause WHERE

Exemples :

```
SELECT * FROM Employees WHERE EmployeeID = 5
```

EmployeeID	LastName	FirstName
5	Buchanan	Steven

```
SELECT * FROM Customers WHERE Country = 'Canada'
```

CustomerID	CompanyName	...
BOTTM	Bottom-Dollar Markets	...
LAUGB	Laughing Bacchus Wine Cellars	...
MEREP	Mère Paillarde	...



Requêtage des données

La clause WHERE

Les opérateurs disponibles :

Opérateur	Description
=, <, >, <=, >=, <>	Comparaison d'égalité, d'infériorité, de supériorité, etc.
LIKE, NOT LIKE	Comparaison de chaînes de caractères
NOT, AND, OR	Opérateurs logiques
BETWEEN, NOT BETWEEN	Comparaison de plages
IN, NOT IN	Présence au sein d'une liste de valeurs
IS NULL, IS NOT NULL	Absence de valeur



Requêtage des données

La clause WHERE

Les opérateurs de comparaison nécessitent que les chaînes et dates soient spécifiées entre apostrophes :

```
SELECT OrderID, CustomerID, OrderDate FROM Orders  
WHERE OrderDate > '1998-05-01'
```

OrderID	CustomerID	OrderDate
11067	DRACD	1998-05-04 00:00:00.000
11068	QUEEN	1998-05-04 00:00:00.000
11069	TORTU	1998-05-04 00:00:00.000



Requêtage des données

La clause WHERE

Les opérateurs de comparaison de chaînes permettent l'utilisation de frimes :

```
SELECT CompanyName  
FROM Customers  
WHERE CompanyName LIKE '%restaurant%'
```

CompanyName
GROSELLA-Restaurante
Lonesome Pine Restaurant
Tortuga Restaurante



Requêtage des données

La clause WHERE

Les frimes disponibles pour MySQL :

Frime	Description
%	Quelconque chaîne de zéro ou plusieurs caractères.
_	Quelconque unique caractère.



Requêtage des données

La clause WHERE

Exemples :

- LIKE 'BR%' Tout ce qui commence par '*BR*'
- LIKE '%een' Tout ce qui se termine par '*een*'
- LIKE '%en%' Tout ce qui contient '*en*'
- LIKE '_en' Tout ce qui contient trois lettres et qui termine par '*en*'



Requêtage des données

La clause WHERE

Les opérateurs logiques permettent de combiner plusieurs éléments de condition.

```
SELECT ProductID, ProductName, SupplierID, UnitPrice
FROM products
WHERE (productname LIKE 'T%' OR productid = 46)
      AND (unitprice > 16.00)
```

ProductID	ProductName	SupplierID	UnitPrice
14	Tofu	6	23.25
29	Thuringer Rostbratwurst	12	123.79
62	Tarte au sucre	29	49.3



Requêtage des données

La clause WHERE

Prenez garde de bien parenthéser vos conditions. Comparez les résultats de ces deux requêtes :

```
SELECT ProductID, ProductName, SupplierID, UnitPrice
FROM products
WHERE (productname LIKE 'T%' OR productid = 46)
    AND (unitprice > 16.00)
```

```
SELECT ProductID, ProductName, SupplierID, UnitPrice
FROM products
WHERE (productname LIKE 'T%')
    OR (productid = 46 AND unitprice > 16.00)
```



Requêtage des données

La clause WHERE

L'opérateur **BETWEEN** permet de retourner les enregistrements se trouvant dans une plage de valeurs :

```
SELECT ProductID, ProductName, UnitPrice  
FROM products  
WHERE unitprice BETWEEN 10 AND 20
```

ProductID	ProductName	UnitPrice
	Chai	18.0000
	Chang	19.0000
	Aniseed Syrup	10.0000



Requêtage des données

La clause WHERE

L'opérateur **IN** permet de retourner les enregistrements se trouvant dans une liste de valeurs :

```
SELECT CompanyName, Country  
FROM suppliers  
WHERE Country IN ('Japan', 'Italy')
```

CompanyName	Country
Tokyo Traders	Japan
Mayumi's	Japan
Formaggi Fortini s.r.l	Italy



Requêtage des données

La clause WHERE

L'opérateur **IS NULL** permet de retourner les enregistrements dont le champs spécifié possède une valeur nulle :

```
SELECT CompanyName, Fax  
FROM Suppliers  
WHERE Fax IS NULL
```

CompanyName	Fax
Exotics Liquid	Null
New Orleans Cajun Delight	Null
Tokyo Traders	Null



Requêtage des données

La clause CASE

La clause **CASE** permet de piloter la valeur d'une colonne selon une condition.

```
SELECT
{[NomTable.|AliasDeTable.]*|NomColonne|Expression
  [AS AliasDeColonne][, ...n]},

CASE expression
  WHEN when_expression THEN result_expression [ ...n ]
  [ ELSE else_result_expression ]
  END [AS AliasDeColonne]
FROM NomTable
```




Requêtage des données

La clause CASE

L'exemple suivant retourne une salutation selon le sexe du client :

```
SELECT *,  
CASE Sexe  
  WHEN 1 THEN 'Mr'  
  ELSE 'Mme'  
END AS Salutations  
FROM Clients
```

Nom	Prénom	Sexe	Salutations
Chabot	Sylvain	1	Mr
Chaput	Claude	2	Mme
Chartier	Solange	2	Mme
Chagnon	Stéphane	1	Mr
...
Chassé	Sophie	2	Mme



Requêtage des données

La clause CASE

L'exemple suivant s'assure qu'une autre valeur que 1 et 2 ne retourne pas la valeur par défaut 'Mme':

```
SELECT *,  
CASE Sexe  
  WHEN 1 THEN 'Mr'  
  WHEN 2 THEN 'Mme'  
  ELSE ''  
END AS Salutations  
FROM Clients
```

Nom	Prénom	Sexe	Salutations
Chabot	Sylvain	1	Mr
Chaput	Claude	<i>null</i>	
Chartier	Solange	2	Mme
Chagnon	Stéphane	1	Mr



Requêtage des données

La clause CASE

La clause **CASE** peut également être construite comme suit :

```
SELECT *,
```

```
CASE
```

```
  WHEN Sexe=1 THEN 'Mr'
```

```
  WHEN Sexe=2 THEN 'Mme'
```

```
  ELSE "
```

```
  END AS Salutations
```

```
FROM Clients
```

Nom	Prénom	Sexe	Salutations
Chabot	Sylvain	1	Mr
Chaput	Claude	<i>null</i>	
Chartier	Solange	2	Mme
Chagnon	Stéphane	1	Mr



Requêtage des données

La clause CASE

Des clauses **CASE** peuvent être imbriquées

```
SELECT Nom, Prénom, Sexe,  
  
CASE  
  WHEN Sexe=1 AND Age>18 THEN 'Mr'  
  WHEN Sexe=2 THEN  
    CASE WHEN EtatCivil=1 THEN 'Mme'  
    ELSE 'Mlle'  
    END  
  ELSE ''  
END AS Salutations  
  
FROM Clients
```

Nom	Prénom	Sexe	Salutations
Chabot	Sylvain	1	Mr
Chaput	Claude	null	
Chartier	Solange	2	Mme
Chassé	Sophie	2	Mlle



Requêtage des données

Les fonctions d'agrégation

Les fonctions d'agrégation permettent d'effectuer des calculs sur des calculs afin d'en retourner un sommaire. Les fonctions d'agrégation retournent une valeur scalaire numérique.

```
USE northwind  
SELECT AVG(unitprice) FROM products
```

```
USE northwind  
SELECT SUM(quantity) FROM products
```



Requêtage des données

Les fonctions d'agrégation

Les fonctions d'agrégation retournent *NULL* lorsqu'aucune ligne ne respecte la requête. Seule **COUNT**(*) retournera zéro si aucune ligne n'est rencontrée.

```
SELECT COUNT( * ) FROM employees  
WHERE FirstName = 'Bill'
```

(no column name)

0



Requêtage des données

Les fonctions d'agrégation

De plus, les fonctions d'agrégation ne considèrent pas les valeurs *NULL* rencontrées alors que **COUNT**(*) les considère.

```
SELECT COUNT( ReportsTo )  
FROM employees
```

(no column name)

8

```
SELECT COUNT( * )  
FROM employees
```

(no column name)

9



Requêtage des données

Les fonctions d'agrégation

Fonctions :

Fonction	Description
AVG	Moyenne des valeurs.
COUNT	Décompte des valeurs.
COUNT(*)	Décompte des valeurs en considérant les valeurs NULL.
MAX	Valeur maximale parmi les valeurs.
MIN	Valeur minimale parmi les valeurs.
SUM	Sommes des valeurs.



Requêtage des données

Les fonctions d'agrégation

- La fonction **COUNT** est la seule à prendre en charge les types de données *text*.
- Les fonction **SUM AVG MIN** et **MAX** ne prennent en charge que les valeurs numériques.



Requêtage des données

La clause **GROUP BY**

La clause **GROUP BY** permet de regrouper les enregistrement mais, contrairement à la clause **ORDER BY**, ne garantie pas l'ordre dans laquelle les enregistrements seront retournés.

```
SELECT OrderID FROM orderDetails  
GROUP BY OrderID
```

OrderID
10248
10249
10250

(822 row(s) affected)



Requêtage des données

La clause **GROUP BY**

Lorsque combinée à une fonction d'agrégation, la clause **GROUP BY** permet de calculer en fonction d'un regroupement.

```
SELECT OrderID, COUNT(ProductID) AS NombreProduits  
FROM OrderDetails  
GROUP BY OrderID
```

OrderID	NombreProduits
10248	3
10249	2
10250	3
10251	3

(822 row(s) affected)



Requêtage des données

La clause GROUP BY

L'exemple suivant retourne le nombre de fois que chaque produit a fait l'objet d'une commande :

```
SELECT ProductID, COUNT(ProductID)  
FROM OrderDetails  
GROUP BY ProductID  
ORDER BY COUNT(ProductID) DESC
```

ProductID	Nombre
56	50
62	48
41	47
(73 row(s) affected)	



Requêtage des données

La clause **HAVING** (de **GROUP BY**)

La clause **HAVING** permet d'agir un peu comme une clause **WHERE** traitant au niveau des regroupements. La clause **HAVING** se place après la clause **GROUP**.

```
SELECT ProductID, SUM(quantity) AS QuantTotale  
FROM OrderDetails  
GROUP BY ProductID  
HAVING SUM(quantity) >=300
```

ProductID	QuantTotale
56	50
62	48
41	47

(62 row(s) affected)