

## Initiation



### Introduction

Le SQL (Structure Query Language) permet de créer, de modifier et gérer des informations dans des bases de données.

C'est un langage universel d'interrogation des bases de données qui permet à différents systèmes d'échanger des données entre eux. La plupart des requêtes programmées en ASP utilisent ce langage, que se soit pour lire le contenu d'une base, le modifier, ajouter ou supprimer des données.

A partir d'une table sous ACCESS, nous vous proposons un exemple qui permettra de voir ici en détail la syntaxe des requêtes les plus utilisées : SELECT, DELETE, UPDATE, INSERT INTO...la clause WHERE et ses éléments, ORDER BY, GROUP BY et DISTINCT.

TableActeurs	nom	prenom	nationalite	recompenses	age	genre	films
--------------	-----	--------	-------------	-------------	-----	-------	-------

Chaque instruction SQL doit se terminer par un point-virgule.

L'instruction SELECT >> permet d'afficher les champs ou enregistrements à récupérer dans la table.

```
SELECT * FROM TableActeurs ;
```

! Afficher tous les champs et tous les enregistrements de cette table !

```
SELECT prenom, nom FROM TableActeurs ;
```

! Précise les champs à récupérer (prenom et nom) de cette table !

```
SELECT * FROM TableActeurs WHERE nom='Depardieu' ;
```

! Couplé à la clause WHERE, SELECT limite les données à récupérer !

---

L'instruction DELETE >> permet d'effacer des enregistrements dans la table.

```
DELETE * FROM TableActeurs ;
```

! Détruit tous les enregistrements de cette table, les données sont définitivement perdues !

```
DELETE nationalite FROM TableActeurs ;
```

! Précise les champs à détruire (nationalite) de cette table !

```
DELETE * FROM TableActeurs WHERE nationalite='américaine' ;
```

! Couplé à la clause WHERE, DELETE limite les données à détruire !

---

L'instruction UPDATE >> permet la mise à jour de la table.

```
UPDATE TableActeurs SET genre='comique' ;
```

! Met à jour cette table en définissant un champ (genre) par une nouvelle valeur ("comique") pour tous les enregistrements !

```
UPDATE TableActeurs SET genre='comique' WHERE nom='Bourvil' ;
```

! Couplé à la clause WHERE, UPDATE limite les enregistrements à modifier !

---

L'instruction INSERT TO >> permet d'ajouter un enregistrement dans la table.

```
INSERT INTO TableActeurs (nom, prenom, nationalite, recompenses, age, genre, films) VALUES ('Depardieu', 'Gérard', 'française', 'César', '55', 'historique', 'Cyrano de Bergerac') ;
```

! Ajoute un enregistrement dans cette table en insérant une valeur pour chaque champ cité !

La clause **WHERE** >> permet de limiter les données à récupérer dans la table, elle s'utilise avec SELECT, DELETE ou UPDATE.

SELECT \* FROM TableActeurs WHERE films='action' ;

#### Les éléments de la clause WHERE

Ils permettent de définir la condition dans cette clause. La condition peut-être accompagnée des opérateurs logiques **AND**, **OR** ou **NOT**.

•	Comparaison à une valeur ( =, <>, <, >, <=, >= )
•	Comparaison à une fourchette de valeurs ( BETWEEN )
•	Comparaison à une liste de valeur ( IN )
•	Comparaison à un filtre ( LIKE )
•	Test 'tous' ou 'au moins' ( ALL , ANY )
•	Test existentiel ( EXIST )

#### Exemples

SELECT nom FROM TableActeurs WHERE age > 25

! Afficher tous les noms d'acteur dont l'âge est supérieur à 25 !

SELECT nom FROM TableActeurs WHERE (nationalite='américaine') **AND** (age > 25)

! Afficher tous les noms d'acteur dont la nationalité est américaine et l'âge supérieur à 25 !

SELECT nom FROM TableActeurs WHERE (nationalite='américaine') **OR** (age >= 25)

! Afficher tous les noms d'acteur dont la nationalité est américaine ou l'âge supérieur ou égal à 25 !

SELECT nom, recompenses FROM TableActeurs WHERE (nationalite='américaine') **AND** (age >= 25) **OR** (genre='drame')

! Afficher tous les noms et récompenses d'acteurs dont la nationalité est américaine et l'âge supérieur ou égal à 25, ou alors le genre est drame !

SELECT \* FROM TableActeurs WHERE age **BETWEEN** 35 **AND** 50 ;

! Afficher l'age entre 35 et 50, on peut utiliser **NOT BETWEEN** pour inverser cette sélection !

SELECT \* FROM TableActeurs WHERE nationalite **IN** ('américaine', 'française') ;

! Permet de trouver les nationalités d'acteur se trouvant dans cette liste de valeurs !

SELECT \* FROM TableActeurs WHERE nom **LIKE** 'DE%' ;

! Pour retrouver le nom d'un acteur qui commencerait par DE, le % représente une chaîne de caractères ; sous Access le % est remplacé par \* !

SELECT \* FROM TableActeurs WHERE nom **NOT LIKE** 'D\_\_' ;

! Pour afficher tous les noms d'acteur ne commençant par D de 4 lettres, le \_ représente un caractère !

La clause **ORDER BY** >> permet de trier les données dans la table, elle s'utilise avec SELECT.

SELECT \* FROM TableActeurs WHERE nationalite='française' ORDER BY age DESC ;

! Sélectionne la nationalité demandée, puis trie les enregistrements par ordre décroissant d'âge (DESC), par défaut le tri est croissant ASC !

La clause **GROUP BY** >> permet de regrouper les données dans la table, elle s'utilise avec SELECT.

```
SELECT nom, prenom, nationalite FROM TableActeurs GROUP BY nationalite ;  
! Regroupe les acteurs par nationalité !
```

---

La clause DISTINCT >> permet d'éviter la redondance dans les résultats de la requête, elle s'utilise avec SELECT.

```
SELECT DISTINCT nationalite FROM TableActeurs ;  
! Renvoie toutes les nationalités différentes sans doublons même si plusieurs acteurs ont la même nationalité !
```

En ASP, les apostrophes sont remplacées par des guillemets pour le texte, par ' ' pour des variables de type texte.  
Exemple : <%variable="Pierre"  
SELECT \* FROM Table WHERE prenom=' ' & variable & " ' %>



[www.asp-irine.com](http://www.asp-irine.com)