
2
SQL (suite)

Ordre SELECT (selection)

- Clause **WHERE** correspond à une sélection**.**
- Restreint la requête aux enregistrements qui respectent les conditions.

Syntaxe:

SELECT liste_attributs FROM table WHERE condition(s);

- Une condition est une expression composée d'opérateurs (**arithmétiques**, **logiques, …**) et d'opérandes (constantes ou attributs).

Exemple:

- **SELECT * FROM etudiant WHERE nom='mohamed';**

Opérateurs de comparaison

Opérateur BETWEEN

- Permet d'afficher des enregistrements basés sur une tranche de valeurs

Syntaxe:

WHERE *nom_colonne* **BETWEEN** *limite_inf* **AND** *limite_sup*

- Les limites peuvent être des nombres, des caractères, des dates. (limites incluses)

N.B: les limites doivent être placées entre simples côtes (caractères ou dates).

Exemple:

```
SELECT nom, age FROM etudiant WHERE age BETWEEN 20
AND 22;
```

Opérateurs de comparaison (2)

Opérateur LIKE

- Permet de faire des recherches de caractères spécifiques dans une chaîne de caractères données.

Syntaxe:

WHERE *nom_colonne* **LIKE** *'expression'*

- L'expression contient les symboles:

Symbole	Description
%	Représente une série de caractères
_	Représente un caractère quelconque

Exemple:

SELECT nom, age FROM etudiant WHERE nom LIKE 'moha%';

Opérateurs de comparaison (3)

Opérateur IN

- Permet d'afficher des enregistrements appartenant à une liste de valeurs.

Syntaxe:

WHERE *nom_colonne* **IN** *(valeur_1, valeur_2,..,valeur_n)*

N.B: Les valeurs de la liste peuvent être des nombres, des caractères, des dates.

- Les caractères ou les dates doivent être placées entre simples côtes.

Exemple:

SELECT nom, age FROM etudiant WHERE age IN(20, 21, 22);

Opérateurs de comparaison (4)

Opérateur IS NULL

- Permet d'afficher les enregistrements dont le champ en terme contient une valeur nulle.

Syntaxe:

WHERE *nom_colonne* **IS NULL;**

Exemple:

SELECT nom, age FROM etudiant WHERE adresse IS NULL;

Opérateurs logiques

Opérateur AND

- Permet d'afficher les enregistrements vérifiant toutes les conditions impliquées dans la clause WHERE:

Syntaxe:

WHERE condition1 AND condition2;

Exemple:

SELECT nom, prenom, age FROM etudiant WHERE prénom='Mohamed' AND age=20;

Nom	Prénom	Age
Alaoui	Mohamed	20
Ismaili	Mohamed	20

Opérateurs logiques (2)

Opérateur OR

- Permet d'afficher les enregistrements vérifiant une des conditions impliquées dans la clause WHERE:

Syntaxe:

WHERE condition1 OR condition2;

Exemple:

SELECT nom, prenom, age FROM etudiant WHERE prénom='mohamed' OR age=20;

Nom	Prénom	Age
Alaoui	Mohamed	20
Ismaili	Mohamed	20
Karimi	Karim	20
Benjelloun	Mohamed	19

Opérateurs logiques (3)

Opérateur NOT:

- Permet d'afficher les enregistrements ne vérifiant pas la condition impliquée dans la clause **WHERE**:

  Syntaxe:

  WHERE nom_colonne NOT opérateur_comparaison valeur;

- Opérateurs de comparaison:

  NOT IN, NOT LIKE, NOT BETWEEN ou IS NOT NULL.

Exemple: SELECT nom, prenom, age FROM etudiant WHERE age NOT IN (20,21,22);

Nom	Prénom	Age
Benjelloun	Mohamed	19
Hachimi	Rita	24

Ordre SELECT (2)

Exercice:

Soit la relation **Etudiant(**Id, nom, prenom, age, ville, tel**)**.

1. Quels sont les étudiants âgés de 20 ans ou plus ?
2. Quels sont les étudiants âgés de 19 à 23 ans ?
3. Quels sont les étudiants ayant un numéro de tel commençant par 0535?
4. Quels sont les étudiants dont la ville est inconnue (connue) ?

Ordre SELECT (3)

Ordonner la clause SELECT

- Afficher des enregistrements sélectionnés dans l'ordre croissant ou décroissant.

N.B: L'ordre par défaut est croissant.

Syntaxe:

SELECT colonne 1, colonne 2 FROM table WHERE condition(s) ORDER BY {colonne } [ASC | DESC]

Exemple:

SELECT nom, prenom, age FROM etudiant ORDER BY age;

Ordre SELECT (4)

Ordonner la clause SELECT(suite)

- Possibilité de trier sur une colonne qui n'a pas été sélectionnée dans la clause SELECT.

Syntaxe:

```
SELECT  colonne1, colonne2 FROM table WHERE Condition(s)
ORDER BY colonne3;
```

- Les alias de colonnes sont permises.

Syntaxe:

```
SELECT  attribut_1 AS alias_1 FROM table WHERE condition(s)
ORDER BY alias_1;
```

Ordre SELECT (5)

Ordonner la clause SELECT(suite)

- Possibilité de trier sur plusieurs colonnes.

Syntaxe:

SELECT col1, col2, col3 FROM table WHERE condition(s) ORDER BY col1[ASC |DESC], col3[ASC | DESC]

Exemple:

SELECT nom, prénom, age FROM etudiant ORDER BY prénom, age DESC;

nom	prénom	age
Karimi	Karim	20
Alaoui	Mohamed	21
Ismaili	Mohamed	20
Benjelloun	Mohamed	19

Fonctions SQL

- **Fonction:** programme qui effectue des opérations sur des données.

Syntaxe:

nom_fonction(arg_1,…,arg_n){ valeur_retour}

- Utilisation des fonctions:
  - ✓ Exécuter des calculs sur des données.
  - ✓ Formater des dates et des nombres pour l'affichage.
  - ✓ Convertir des types de données de colonne.

Fonctions de chaînes de caractères

❖ Fonctions de manipulation de casse

- Manipulation des données dans une casse différente.

Fonction	Définition	Exemples	Résultat
INITCAP(col | expr)	Convertit la première lettre de chaque mot d'une chaîne de caractères en majuscule et les autres lettres en minuscule.	**INITCAP**('cours de SQL')	Cours De Sql
LOWER(col | expr)	Convertit une chaîne de caractères en minuscule.	**LOWER**('Cours de SQL')	cours de sql
UPPER(col | expr)	Convertit une chaîne de caractères en majuscule.	**UPPER** ('Cours de SQL')	COURS DE SQL
LENGHT(col | expr)	Permet de récupérer le nombre de caractères d'une chaîne. LENGTH retourne une valeur de type NUMBER.	**LENGTH** ('Bonjour')	7
SUBSTR(col | expr, m,n)	Permet d'extraire une chaîne de caractères de la chaîne de caractère **col**(ou **expr**) sur une longueur **n** à partir de la position **m**.	**SUBSTR**('Bonjour',1,3)	Bon

Fonctions de chaînes de caractères (2)

❖ Fonctions de manipulation de casse (suite)

Fonctions	Définition	Exemples	Résultat
INSTR(col | expr, C)	Permet de récupérer la position de la première occurrence du caractère **C** dans la chaîne de caractères **col** (ou **expr**)	**INSTR**('Bonjour','j')	4
LPAD(col | expr, n,'String')	Permet de compléter une chaîne de caractère jusqu'à ce qu'elle atteigne la taille souhaitée (**n**), en ajoutant des caractères au début de cette chaîne	**LPAD**(sal, 10,'*')	******5000
RPAD(col | expr,n,'String')	Permet de compléter une chaîne de caractère jusqu'à ce qu'elle atteigne la taille souhaitée (n), en ajoutant des caractères à la fin de cette chaîne.	**RPAD**(sal, 10,'*')	5000******
CONCAT(col1 | expr1, col2 | expr2)	Permet de concaténer la valeur de la première chaîne à la valeur de la seconde chaîne. (équivalent à l'opérateur "||").	**CONCAT** ('Bon','jour')	Bonjour
TRIM(leading | trailing| both, trim_caract **FROM** trim_source)	Permet de couper les caractères « trim_character » en entête (leading), en fin (trailing) ou les deux (both) d'une chaîne de caractère « trim_source».	**TRIM** ('S' FROM 'SSMITH')	MITH

Fonctions des nombres

Fonction	Définition	Exemples	Résultat
ROUND(*col*| *expr* [,*n*])	Permet d'arrondir une valeur de ***col*** *(*ou ***expr****)* à ***n*** décimales près	ROUND (45.926, 2) ROUND (45.923, 0) ROUND (45.923,-1)	45,93 46 50
TRUNC(*col* | *expr* [,*n*])	Permet de tronquer la valeur de ***col*** *(*ou ***expr****)* à ***n*** décimales près.	TRUNC (45.923, 2) TRUNC (45.923, 0) TRUNC (45.923,-1)	45,92 45 40
MOD(*m*,*n*)	Permet de retourner le reste de la valeur de ***m*** divisé par *n*.	MOD(3,2)	1

Fonctions des dates

❖ Fonction SYSDATE

- Permet de retourner la date et l'heure du système.

N.B: par défaut le résultat est sous la forme DD-MON-YY

❖ Opérations arithmétiques

Opération	Résultat	Description
date + nombre	date	Ajoute un nombre de jours à une date
date - nombre	date	Soustrait un nombre de jours à une date
date – date	Nombre de jours	Soustrait une date d'une autre date
date + nombre/24	date	Ajoute un nombre d'heures à une date

Fonctions des dates (2)

Fonction	Définition	Exemples	Résultat
LAST_DAY(*date*)	Trouve la date du dernier jour du mois qui contient *date*.	LAST_DAY('01-SEP-95')	'30-SEP-95'
ROUND(*date*,'*format*')	Retourne la *date* arrondie à l'unité spécifiée par *format*. Si le format est omis, *date* est arrondie au jour le plus près.	ROUND('25-JUL-95','MONTH')	'01-AUG-95'
		ROUND('25-JUL-95','YEAR')	'01-JAN-96'
TRUNC(*date* ,'*format*')	Retourne la *date* tronquée à l'unité spécifiée par *format*. Si le format est omis, *date* est tronquée au jour le plus près.	TRUNC('25-JUL-95','MONTH')	'01-JUL-95'
		TRUNC('25-JUL-95','YEAR')	'01-JAN-95'

Fonctions des dates (3)

Fonction	Définition	Exemples	Résultat
MONTHS_BETWEEN (*date1*,*date2*)	Retourne le nombre de mois séparant deux dates.	MONTHS_BETWEEN ('01-SEP 95', '11-JAN-94')	19.6774194
ADD_MONTHS(*date*,*n*)	Ajoute ***n*** mois à une date. ***n*** doit être un entier.	ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY(*date*, '*day of week*')	Trouve la date du prochain jour de la semaine (*day of week*) suivant *date*. La valeur de "*day of week*" doit être un nombre représentant le jour ou une chaîne de caractères.	NEXT_DAY ('01-SEP- 95', 'FRIDAY')	'08-SEP-95'

Fonctions de conversion

- Le serveur Oracle peut automatiquement convertir les types de données suivants :

DE	A
VARCHAR2 OU CHAR	NUMBER
VARCHAR2 OU CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Fonctions de conversion (2)

- **Les trois principales fonctions de conversion explicite:**

Fonction	Définition
TO_CHAR(number|date, *'format'*)	Convertit un nombre ou une date en une chaîne de caractères
TO_NUMBER *(char ,'format')*	Convertit une chaîne de caractères en un nombre.
TO_DATE *(char ,'format')*	Convertit une chaîne de caractères en une date

N.B: La fonction **TO_DATE** possède le modificateur **fx** (format exact) qui vérifie la concordance entre la chaîne de caractère et le format date.

Chaîne de caractère	Format date	Resultat
'15-JAN-1998'	'fxDD-MON-YYYY'	Pas d'erreur
'1-JAN-1998'	'fxDD-MON-YYYY'	Erreur

Exemples:

TO_CHAR(sysdate, 'yyyy/mm/dd') ***Résultat***: '2018/03/07'
TO_CHAR(sysdate, 'Month DD, YYYY') ***Résultat*** : 'Mars 07, 2018.
TO_Number ('1210.73', '$9999.9') ***Résultat*** : $1210.7

TO_DATE('2017/03/07', 'Month DD, YYYY') ***Résultat*** : March 07, 2017
TO_DATE('2017/03/07', 'DAY DD MONTH YYYY') ***Résultat*** : Tuesday 07 March 2017

Fonctions générales

❖ La fonction: NVL

- Retourne *expr2* si *expr1* est NULL, sinon retourne *expr1*.
- Convertit *expr1* susceptible d'être nulle en une valeur réelle *expr2*.

Syntaxe: **NVL** *(expr1, expr2)*

❖ La fonction: NVL2

- Retourne *expr3* si *expr1* est NULL sinon retourne *expr2*.
- Convertit *expr1* susceptible d'être null en une valeur réelle *expr3 ou expr2*.

Syntaxe: **NVL2** *(expr1, expr2, expr3)*

Fonctions générales (2)

❖ La fonction: NULLIF

- Compare deux expressions, si elles sont **identiques** la fonction retourne **NULL**, dans le cas contraire la fonction retourne la **première expression**.

Syntaxe: **NULLIF** *(expr1, expr2)*

❖ La fonction: COALESCE

- Retourne la première expression non nulle rencontré dans la liste.

Syntaxe: **COALESCE** *(expr1, expr2, … exprn)*

Expressions conditionnelles

❖ Expression: CASE

- Permet d'utiliser la structure de **IF-THEN-ELSE** sans recourir à l'utilisation des procédures.

Syntaxe 1:

```
CASE
WHEN condition1 THEN expression1
WHEN condition2 THEN expression2
. . .
ELSE expression_défaut
END
```

Exemple

```
CASE
WHEN (poste = 'Président') THEN 1
WHEN (poste = 'Directeur') THEN 2
ELSE 3
END
```

Expressions conditionnelles (2)

Syntaxe 2:

CASE expression
WHEN valeur1 **THEN** expression1
WHEN valeur2 **THEN** expression2
. . .
ELSE expression_défaut
END

Exemple

CASE poste
WHEN 'Président' **THEN** 1
WHEN 'Directeur' **THEN** 2
ELSE 3
END