Universite 20 Aout 1955- Skikda Licence TC informatique 2 année

TP 4 BDD : ACCESS 07 - REQUETES DE SELECTION EN MODE SQL

Qu'elle soit créée avec un Assistant ou en mode Création, toute requête est générée en code SQL. C'est un langage de création de requête qui est puissant, tout en étant simple.

FENÊTRE DE SAISIE D'UNE REQUÊTE EN SQL

Pour accéder à la *fenêtre dans laquelle on tape une requête en mode SQL* :

- A l'onglet Créer (situé à droite de l'onglet Accueil), dans le groupe Autre, activez le *bouton « Création de requête »*. Fermez la fenêtre « Afficher la table ».
- Puis : soit dans le groupe Résultats, soit dans la barre d'état, cliquez sur le *bouton SQL* SQL .

INDICATIONS PRELIMINAIRES A LA CREATION D'UNE REQUETE

Taille des caractères et police du texte de la requête

Lors de la saisie d'une requête, si la *taille des caractères* vous semble trop petite ou si vous souhaitez changer de*police*, vous pouvez *modifier ces paramètres* :
Activez le bouton Office (en haut, à gauche de l'écran) > *Options Access*. Cliquez sur « Concepteurs d'objets ». Dans la zone titrée « *Création de requête* », vous pouvez changer la taille des caractères ainsi que la police des textes de requêtes. Validez.
Les nouveaux paramètres seront pris en compte lors des prochaines requêtes.

Casse, espaces et paragraphes

- *Casse* : SQL ne distingue pas les lettres minuscules des lettres majuscules.
- *Espaces et paragraphes* : on peut insérer à volonté des espaces et des sauts de paragraphes (en appuyant sur la touche Entrée) pour rendre plus lisible le texte de la requête.

Le point-virgule final ;

A la fin d'une requête, il est nécessaire de taper un *point-virgule*, pour signaler la fin de la saisie du texte de la requête.

Sommaire...

1 Présentation de la commande SQL SELECT.

SELECT est la seule commande du SQL permettant **d'interroger une base de données**. Il existe d'autres commandes pour **manipuler les données**(ajouter, modifier), **manipuler les structures** (bases, vues, tables), **gérer les utilisateurs et leurs droits**, **créer des procédures stockées**.

Le résultat d'un **SELECT** est appelé un **jeu d'enregistrements** (**recordset** en anglais).

2 Afficher l'ensemble d'une table.

```
SELECT *
FROM user;
```

usr_id	usr_name	usr_first_name	usr_login	usr_password	usr_year_study	...
1	PASTORE	Sébastien	spastore	**********	6	
540	DETEST	Ysser	testeleve1	**********	1	
539	DUBUISSON	Christophe	cdubuisson	**********	2	
538	BONVALOT	Romain	rbonvalot	**********	2	
537	LUCK	Florent	fluck	**********	1	
536	TRAN	Alexandre	atran	**********	2	
534	ESQUIROL	Georges	gesquirol	**********	6	
533	LEGRAND	Marie-Pierre	mplegrand	**********	6	
532	TEXTORIS	André	atextoris	**********	6	
...						

3 Afficher seulement certains champs.

Au lieu d'afficher toutes les **colonnes** (ou **champs**), on veut afficher uniquement le nom et le prénom (c'est à dire les colonnes qui ont été coloriées en orange sur le schéma ci-dessous).

usr_id	usr_name	usr_first_name	usr_login	usr_password	usr_year_study	...
1	PASTORE	Sébastien	spastore	**********	6	
540	DETEST	Ysser	testeleve1	**********	1	
539	DUBUISSON	Christophe	cdubuisson	**********	2	
538	BONVALOT	Romain	rbonvalot	**********	2	
537	LUCK	Florent	fluck	**********	1	
536	TRAN	Alexandre	atran	**********	2	
534	ESQUIROL	Georges	gesquirol	**********	6	
533	LEGRAND	Marie-Pierre	mplegrand	**********	6	
532	TEXTORIS	André	atextoris	**********	6	
...						

On utilise pour ça :

```
SELECT usr_first_name, usr_name
FROM user ;
```

usr_first_name	usr_name
Sébastien	PASTORE
Ysser	DETEST
Christophe	DUBUISSON
Romain	BONVALOT
Florent	LUCK
Alexandre	TRAN
Georges	ESQUIROL
Marie-Pierre	LEGRAND
André	TEXTORIS
...	...

exécuté en 0.005 sec.

4 Renommer les colonnes.

Très utile pour clarifier l'affichage ou exploiter les données dans un langage de programmation :

```
SELECT usr_first_name as Prenom, usr_name as Nom
FROM user ;
```

Prenom	Nom
Sébastien	PASTORE
Ysser	DETEST
Christophe	DUBUISSON
Romain	BONVALOT
Florent	LUCK
Alexandre	TRAN
Georges	ESQUIROL
Marie-Pierre	LEGRAND
André	TEXTORIS
...	...

exécuté en 0.004 sec.

Le **AS** est facultatif, on peut donc écrire :

```
SELECT usr_first_name Prénom, usr_name Nom
FROM user ;
```

Prénom	Nom
Sébastien	PASTORE
Ysser	DETEST
Christophe	DUBUISSON
Romain	BONVALOT
Florent	LUCK
Alexandre	TRAN
Georges	ESQUIROL
Marie-Pierre	LEGRAND
Rémi	FORT

exécuté en 0.004 sec.

5 Afficher seulement certains enregistrements : WHERE.

Au lieu d'afficher tous les **enregistrements** (ou **lignes**), on veut afficher uniquement les utilisateurs qui s'appellent *Romain* (en orange sur la capture ci-dessous).

usr_id	usr_name	usr_first_name	usr_login	usr_password	usr_year_study	...
1	PASTORE	Sébastien	spastore	**********	6	
540	DETEST	Ysser	testeleve1	**********	1	
539	DUBUISSON	Christophe	cdubuisson	**********	2	
538	BONVALOT	Romain	rbonvalot	**********	2	
537	LUCK	Florent	Fluck	**********	1	
536	TRAN	Alexandre	Atran	**********	2	
534	ESQUIROL	Georges	gesquirol	**********	6	
533	LEGRAND	Marie-Pierre	mplegrand	**********	6	
532	TEXTORIS	André	atextoris	**********	6	
...						

On utilise pour ça la clause **WHERE** :

```
SELECT usr_first_name as Prenom, usr_name as Nom
FROM user
WHERE usr_first_name='Romain';
```

Prenom	Nom
Romain	BONVALOT
Romain	DUMAS
Romain	DI FELICE
Romain	BARIAL
Romain	VIALA
Romain	CONSTANT
Romain	SICAUD
Romain	VANDERHAEGHEN
Romain	Chaix

exécuté en 0.002 sec.

La condition **WHERE** est suivie de n'importe quelle expression booléenne, c'est à dire qui a pour résultat "vrai" ou "faux". Par exemple :

```
WHERE condition1 AND condition2
WHERE condition1 OR condition2
WHERE NOT condition
WHERE champ1 = "valeur"
WHERE champ1 > champ2 AND champ3 > champ4
etc.
```

6 Trier les résultats.

Pour que les résultats soient triés, on utilise la clause **ORDER BY**, et les paramètres **ASC** (pour un tri croissant, ***asc**endant* en anglais) ou **DESC** (pour un tri décroissant, ***desc**endant* en anglais). Par exemple :

```
SELECT usr_first_name as Prenom, usr_name as Nom
FROM user
WHERE usr_name > ''
ORDER BY Nom ASC;
```

Prenom	Nom
Isis	ABD RABO
Chakib	ABDALLAH
Hasna	ABDELADIM
Malik	ABDESSLEM
Nathaniel	ABENSOUR
Olivier	Abran
Morad	ACHACHERA
Mimouna	ADDA-BENAMEUR
Ophélie	ADOLPHE
...	...

exécuté en 0.02 sec.

```
SELECT usr_first_name as Prenom, usr_name as Nom
FROM user
ORDER BY Nom DESC;
```

Prenom	Nom
Florent	ZAMUNER
Detest	Ysser
Anthony	XUEREF

Mehdi	WEIBEL
Louis	VOLLEKINDT
Matthieu	VOIGNIER
Mickaël	VOGEL
Thomas	VIVIEN
Carole	VIQUESNEL
...	...

exécuté en 0.023 sec.

Le tri par défaut est croissant, on n'est donc pas obligé de préciser **ASC**. Par exemple :

```
SELECT usr_first_name as Prenom, usr_name as Nom
FROM user
WHERE usr_name > ''
ORDER BY Nom;
```

Prenom	Nom
Isis	ABD RABO
Chakib	ABDALLAH
Hasna	ABDELADIM
Malik	ABDESSLEM
Nathaniel	ABENSOUR
Olivier	Abran
Morad	ACHACHERA
Mimouna	ADDA-BENAMEUR
Ophélie	ADOLPHE
...	...

exécuté en 0.02 sec.

Cela revient au même de trier sur le nom original du champ (**usr_name**), sur le nom renommé (**Nom**) ou sur la position du champ dans le SELECT (**2**).

On peut enchaîner plusieurs tris. Par exemple : utilisateurs nés en février, triés sur la date d'anniversaire. Pour ceux qui sont nés le même jour, on veut un classement par nom :

```
SELECT usr_date_of_birth, usr_first_name as Prenom, usr_name
as Nom
FROM user
WHERE usr_date_of_birth LIKE '%-02-%'
ORDER BY DAY(usr_date_of_birth), Nom;
```

usr_date_of_birth	Prenom	Nom
1994-02-01	Ilyes	BALI
1994-02-03	Baptiste	Messines
1991-02-06	Edouard	CABOT-REVERSAC
1988-02-08	Aurore	BENGUIGUI
2019-02-09	Nathaël	COSTES
1994-02-11	Florian	SANDRAL
1991-02-12	Adrien	BERTHIER
1990-02-13	Thibaud	MUSSO
1994-02-14	Vincent	HINAUT
...	...	...

exécuté en 0.003 sec.

7 La jointure : afficher des données éparpillées dans différentes table

Si je regarde la table des petites annonces, je vois qu'il n'y figure que le numéro de l'utilisateur qui a déposé l'annonce, mais pas son nom ni son prénom. C'est normal : si on stockait son nom et son prénom pour chaque petite annonce, on l'aurait en plusieurs exemplaires. C'est ce qu'on appelle la **redondance**, et on essaie la plupart du temps de l'éviter dans les bases de données (ça fait perdre de la mémoire et ça rend les mises à jour plus difficiles).

```
SELECT * FROM cours_annonce ;
```

ann_id	ann_user_id	ann_titre	ann_prix	ann_texte	ann_is_html	ann_date_depot
204	442	<script>alert('Je suis un boulet')</script>	test	test	0	2015-01-27 00:16:27
206	401	Vend ordinateur	150€	Je vend un ordinateur fixe de la marque HP, possède un lecteur de carte mémoire, 3 port USB frontaux et 6 a	0	2015-10-27 12:22:56

	l'arrière, lecteur CD/DVD, écran, clavier, souris et son tapis inclus. Vous pouvez le tester en Salle C352, c'est celui au fond à coté de la fenêtre.

exécuté en 0.001 sec.

Toutefois, je peux afficher cette information puisque le nom et le prénom sont stockés dans la table **user**, et que ces deux tables sont reliées par le numéro de l'utilisateur qui a déposé l'annonce :

```
SELECT usr_id, usr_first_name, usr_name, ann_user_id,
ann_titre
FROM user, cours_annonce
WHERE usr_id = ann_user_id ;
```

usr_id	usr_first_name	usr_name	ann_user_id	ann_titre
442	Steeven	Thein	442	<script>alert('Je suis un boulet')</script>
401	Florent	VALAY	401	Vend ordinateur

exécuté en 0.001 sec.

Notez bien l'importance du `WHERE usr_id = ann_user_id`. Si vous l'oubliez, vous obtenez des résultats incohérents *(remarquez que chaque annonce a été affichée à côté de chaque utilisateur : SQL a fait **toutes** les combinaisons possibles, c'est ce qu'on appelle un **produit cartésien** et c'est rarement ce qu'on souhaite)* :

```
SELECT usr_id, usr_first_name, usr_name, ann_user_id,
ann_titre
FROM user, cours_annonce ;
```

usr_id	usr_first_name	usr_name	ann_user_id	ann_titre
1	Sébastien	PASTORE	442	<script>alert('Je suis un boulet')</script>
1	Sébastien	PASTORE	401	Vend ordinateur
77	Ysser	DETEST	442	<script>alert('Je suis un boulet')</script>
77	Ysser	DETEST	401	Vend ordinateur
76	Christophe	DUBUISSON	442	<script>alert('Je suis un boulet')</script>
76	Christophe	DUBUISSON	401	Vend ordinateur
75	Romain	BONVALOT	442	<script>alert('Je suis un boulet')</script>

75	Romain	BONVALOT	401	Vend ordinateur
74	Florent	LUCK	442	<script>alert('Je suis un boulet')</script>
...	...	...	...	...

exécuté en 0.013 sec.

8 La clause DISTINCT.

Parfois, on est susceptible d'écrire une requête qui répond plusieurs fois la même réponse. Par exemple : qui a déposé une annonce ?

```
SELECT usr_first_name, usr_name
FROM user, cours_annonce
WHERE usr_id = ann_user_id ;
```

usr_first_name	usr_name
Steeven	Thein
Florent	VALAY

exécuté en 0.001 sec.

Pour éviter que la même réponse soit donnée deux fois, on utilise la clause **DISTINCT**.

```
SELECT DISTINCT usr_first_name, usr_name
FROM user, cours_annonce
WHERE usr_id = ann_user_id ;
```

usr_first_name	usr_name
Steeven	Thein
Florent	VALAY

exécuté en 0.001 sec.

Question à se poser

Il faut arriver à savoir quand le DISTINCT est obligatoire ou non, sinon vous serez pénalisés à l'examen. Essayez donc d'évaluer si la requête risque de multiplier les résultats ou pas. Par exemple :

- Si je demande des **clients** dans une table de **clients**, sans jointure, ils seront forcément uniques.
- Si je demande des **catégories** dans une table de **produits**, chaque catégorie risque d'être en plusieurs exemplaires => DISTINCT.
- Si je demande des **clients** dans une table de **clients** jointe à une table d'**achats**, chaque client risque d'être en plusieurs exemplaires => DISTINCT. *Par exemple : Quel client a déjà acheté des produits de telle catégorie ?*

9 La valeur NULL

On peut affecter une valeur spéciale à un champ : la valeur **NULL**.

A ne pas confondre avec une chaîne de caractères vide :

- sémantiquement : une chaîne vide peut signifier *"je connais cette information, mais elle est vide", alors que la valeur **NULL** signifie "je ne connais pas cette information"* *(par exemple pour un numéro de téléphone)*.
- *techniquement : une chaîne vide se teste avec* `champ=""`, *une valeur **NULL** se teste avec* `champ IS NULL`. *Une chaîne non vide se teste avec* `champ <> ""`, *une valeur non nulle se teste avec* `champ IS NOT NULL`.

*Quand on construit une base de données, on doit préciser pour chaque champ s'il peut recevoir la valeur **NULL** ou pas* *démonstration*.

10 Les expressions : pour afficher autre chose qu'un champ brut.

*On n'est pas obligé d'afficher des **champs bruts**, on peut afficher des **expressions basées** (ou non) **sur les champs**.*

```
SELECT CONCAT(usr_first_name, ' ', usr_name) AS Nom
FROM user
WHERE usr_special & 256 ;
```

Nom
Sébastien PASTORE
Georges ESQUIROL
Marie-Pierre LEGRAND
Alexa STROZZI
Gilles GIRARD
Georghy FUSCO
Véronique GAUBERT
Stéphane LOFFREDA
Judes Gisors

Michael DEBORT
Sophie PERIERAS

exécuté en 0.002 sec.

Afficher l'année de naissance d'une personne (remarquez le IS NOT NULL qui permet de n'afficher que les utilisateurs dont on connaît la date de naissance) :

```
SELECT usr_login, YEAR(usr_date_of_birth) AS `annee de naissance`
FROM user
WHERE usr_date_of_birth IS NOT NULL AND usr_special & 5 = 5;
```

usr_login	annee de naissance
gcarpentier	1997
qfeuillye	1996
yguillaume	1997
kmontes	1997
mpinet	1996
rtixador	1997
talvernhe	1995
kaityous	1995
nboidard	1995
...	...

exécuté en 0.002 sec.

Afficher un prix TTC à partir d'un prix hors taxe : `SELECT prix_HT * 1.196 FROM produits ;`

11 Utiliser SELECT sans table (sans FROM).

*Parfois, on a besoin d'utiliser SELECT simplement pour **évaluer une expression**, sans aller chercher des données dans une table.*

*L'utilisation la plus courante est pour **vérifier le fonctionnement des fonctions SQL**. Par exemple, pour vérifier des fonctions de date ou de cryptage :*

```
SELECT CURDATE();
```

CURDATE()
2016-04-07

exécuté en 0.001 sec.

```
SELECT NOW();
```

NOW()
2016-04-07 21:41:46

exécuté en 0 sec.

```
SELECT PASSWORD('mypassword');
```

PASSWORD('mypassword')
*FABE5482D5AADF36D028AC443D117BE1180B9725

exécuté en 0 sec.

12 Le comparateur de chaînes de caractères LIKE.

*Très utile pour les recherches, **LIKE** permet de comparer des chaînes entre elles en utilisant des caractères jokers, c'est à dire des caractères qui remplacent un ou plusieurs caractères.*

% remplace 0, 1 ou plusieurs caractères. Par exemple :

- `champ like "méd%"` *est vrai pour les champs ayant la valeur **méd**, **méd**ecin, **méd**ical, **méd**ée, etc.*
- `champ like "%test%"` *est vrai pour les champs ayant la valeur gréa**test** hits, **test**, con**test**, **test**ament, etc.*

_ (underscore) remplace 1 caractère. Par exemple :

- `champ like "par_"` *est vrai pour les champs ayant la valeur **par**t, **par**s, **par**e, etc.*
- `champ like "199_-12-31"` *est vrai pour les champs ayant la valeur 1990-12-31, 1991-12-31, ... 1999-12-31.*

*Pour compliquer un peu : sous Access, le % s'écrit * et le _ s'écrit ? (respectant ainsi le formalisme des caractères jokers sous DOS, et dans l'explorateur de fichiers en général).*