Les requêtes multi-tables

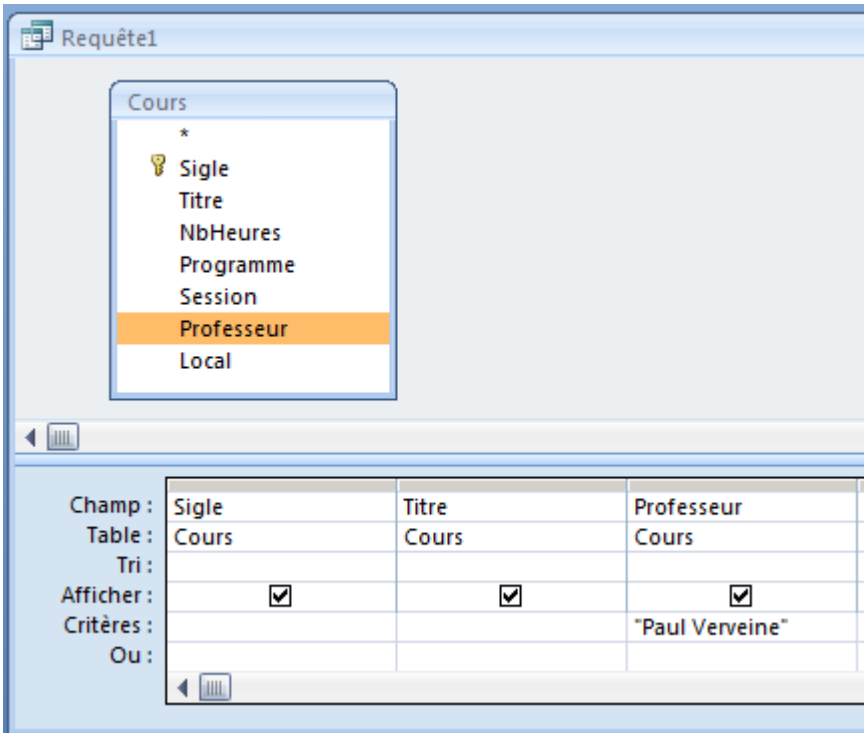
1. *Les requêtes à plusieurs tables – justification et première approche*

Jusqu'à présent, lorsque nous avons créé des requêtes, nous y avons ajouté une seule table de laquelle aller chercher des données. Nous avons ensuite choisi certains champs à afficher (en choisissant des colonnes) et certains enregistrements à afficher (en entrant des critères dans certaines colonnes).

Nous allons voir aujourd'hui comment ajouter plusieurs tables à une requête et surtout pourquoi le faire. Nous utiliserons à titre d'exemple tout au long de cette séance la base de données « Le petit collège Lionel-Groulx ». Cette base de données contient déjà plusieurs tables : les programmes, les groupes, les cours, les professeurs, les locaux, etc.

Supposons pour commencer que nous voulions aller chercher le ***titre*** et le ***sigle*** des **cours** donnés par Paul Verveine.

Le premier réflexe est de créer la requête suivante :



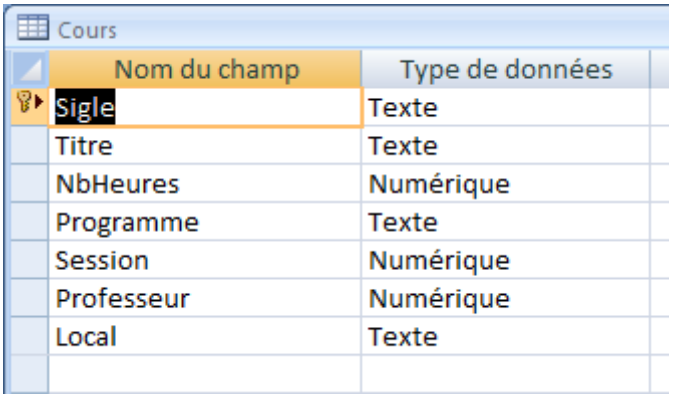
Ça semble assez simple : on part de la table Cours (qui contient toutes les informations sur les cours) et on demande d'afficher le ***sigle*** et le ***titre***. On utilise le champ Professeur comme critère, en demandant de n'afficher que les enregistrements dont le professeur est "Paul Verveine".

(Rappel : si vous ne mettez pas de guillemets, Access les ajoutera pour vous.)

On exécute donc notre requête en s'attendant voir la liste des cours de Paul apparaître devant nos yeux éblouis. Malheureusement, ce n'est pas ce qui arrive. On se retrouve plutôt confronté à un message d'erreur qui dit : « Type de données incompatible dans l'expression du critère ». Quoi?

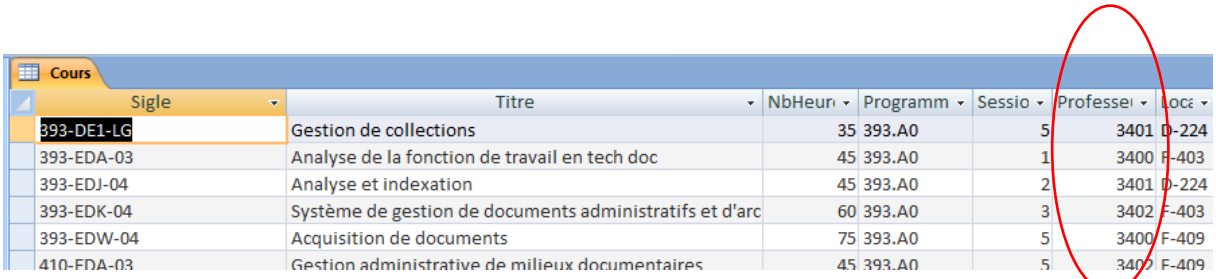
Le type de données est la sorte de données qui peut se retrouver dans un champ. Rappelez-vous, lorsqu'on regarde la structure d'une table, chaque champ a un type de données : du texte, des nombres, des dates, un simple oui ou non, etc. On ne peut pas mettre de lettres dans un champ numérique, ni un chiffre en guise de oui ou de non.

C'est le problème ici. Ouvrons la table Cours, pour voir. En fait, c'est toujours une excellente idée lorsque l'on crée des requêtes de se familiariser avec les tables sur lesquelles on se basera afin de savoir ce qui s'y trouve et ce qui nous sera utile. Commençons par l'ouvrir en mode Création pour voir sa structure :


Cours

Nom du champ	Type de données
Sigle	Texte
Titre	Texte
NbHeures	Numérique
Programme	Texte
Session	Numérique
Professeur	Numérique
Local	Texte

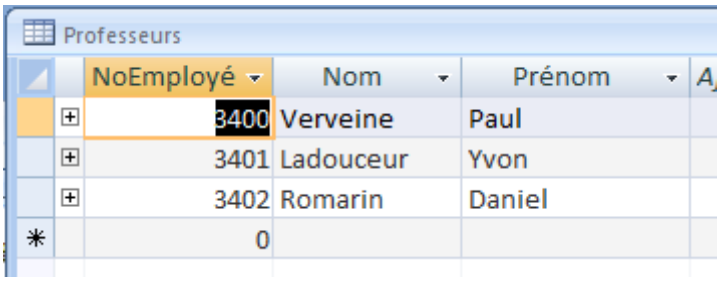
Voyez le champ Professeur : son type de données est numérique. Ça veut dire qu'il ne peut pas y avoir d'autres choses que des nombres dans ce champ et ce, pour tous les enregistrements de la table. Mais comment mettre le nom des profs, alors? Voyons voir le contenu de la table (mode « Feuille de données ») :


Cours

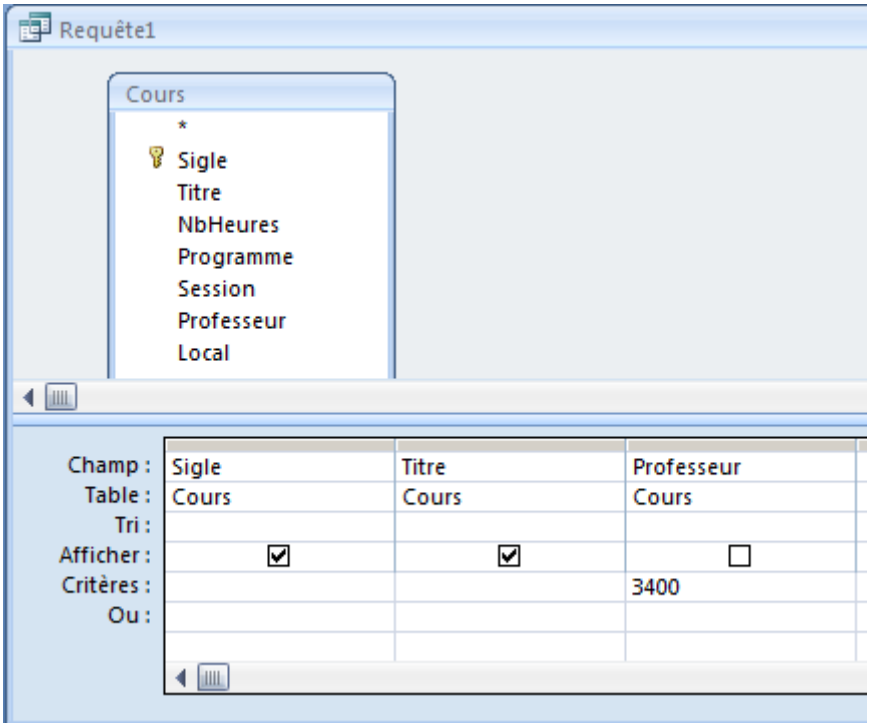
Sigle	Titre	NbHeur	Programm	Sessio	Professe	Loca
393-DE1-LG	Gestion de collections	35	393.A0	5	3401	D-224
393-EDA-03	Analyse de la fonction de travail en tech doc	45	393.A0	1	3400	F-403
393-EDJ-04	Analyse et indexation	45	393.A0	2	3401	D-224
393-EDK-04	Système de gestion de documents administratifs et d'arc	60	393.A0	3	3402	F-403
393-EDW-04	Acquisition de documents	75	393.A0	5	3400	F-409
410-EDA-03	Gestion administrative de milieux documentaires	45	393.A0	5	3402	F-409

Ça semble bien clair maintenant : le champ professeur contient non pas le nom des profs, mais plutôt leur numéro (possiblement un numéro d'employé). Comment faire pour savoir lequel correspond à Paul Verveine? En regardant les autres tables

disponibles, on suppose que la table Professeurs doit sûrement pouvoir nous aider. Ouvrons-la :



Bingo! Paul Verveine porte le numéro 3400. On peut donc modifier notre requête :



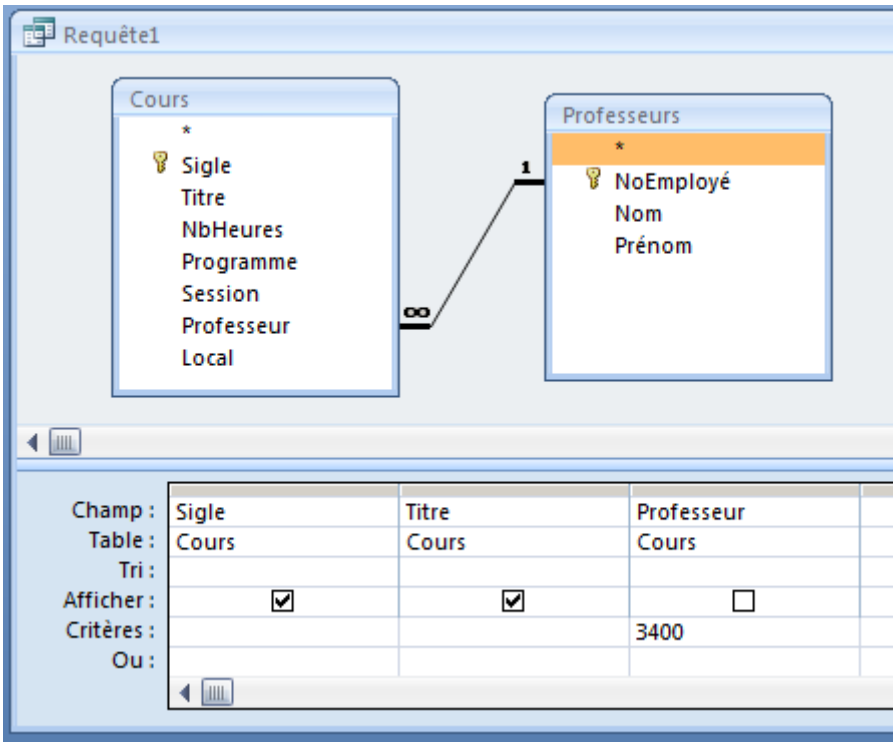
Cette fois-ci, ça fonctionne. Toutefois, le fait d'aller écrire nous-mêmes 3400 après avoir été fouiller manuellement dans une table n'est pas la solution la plus efficace. En effet, Access est supposé être là pour faire ce genre de trucs pour nous. Si on doit chercher nous-mêmes dans des tables, trouver manuellement des enregistrements et retenir des valeurs, on n'utilise pas Access à son plein potentiel.

En plus, que se passera-t-il si jamais Paul change de numéro? On ne sait pas au juste d'où ils viennent ni si c'est possible, mais si oui, notre requête ne fonctionnera plus. Également, si on veut faire une requête « générale », avec paramètres (voir chapitre précédent), qui est capable d'aller chercher les cours de n'importe quel prof, c'est beaucoup plus pratique d'y aller par le nom que par le numéro d'employé, que peu (ou pas) d'élèves connaissent.

La solution à ceci est l'utilisation de **plusieurs tables** dans la requête, ainsi que de **relations** entre les tables.

2. *Les relations*

Ajoutons donc la table « Professeurs » dans notre requête, puisque c'est elle qu'on a consulté pour trouver le numéro du prof. On obtient alors :



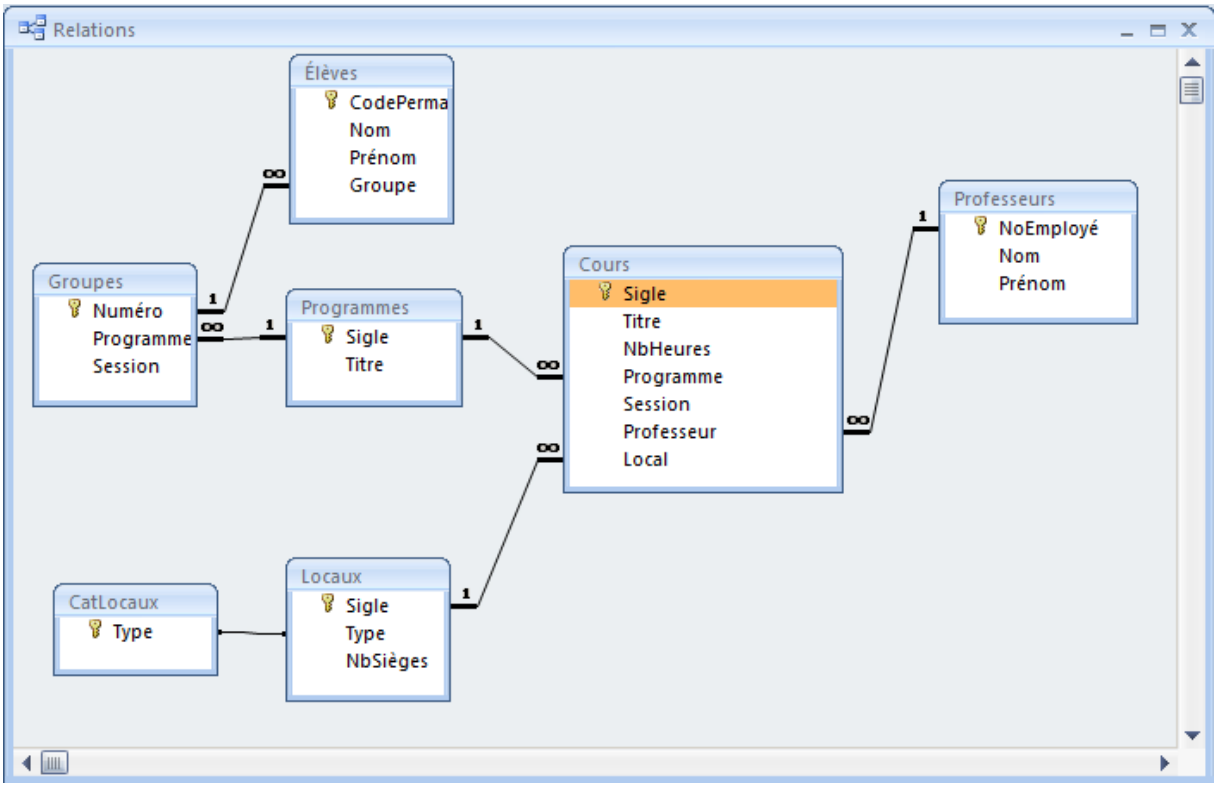
Notez que la ligne qui relie les deux tables s'ajoute automatiquement. Il s'agit de la relation entre les tables. Elle fait en réalité le lien entre deux champs bien précis des tables : le champ Professeur de la table Cours et le champ NoEmployé de la table Professeurs.

Rappel : une relation indique que ces champs contiennent des valeurs qui correspondent. Il est impossible, lorsqu'une telle relation a été définie, d'entrer un cours donné par un professeur dont le numéro ne correspond pas à un enregistrement de la table Professeurs. Autrement dit : il est impossible qu'un cours soit donné par un prof qui n'existe pas.

Cette relation a été mise en place de façon permanente par le créateur de la base de données. Elle a été définie après que les tables aient été créées et existe parce que, conceptuellement, elle a du sens : chaque cours est donné par un prof qui doit exister. Et plutôt que de répéter le nom du prof à chacun de ses cours, on met juste le numéro, ce qui est moins long.

Notez aussi les symboles au-dessus de la ligne de relation : un « 1 » du côté de la table Professeurs et un « ∞ » (symbole d'infinité) du côté de la table Cours. Cela signifie qu'un seul professeur pourra donner autant de cours que désiré. S'il y avait eu un « 1 » de chaque côté, ça aurait voulu dire que chaque professeur aurait eu un et un seul cours attitré – il aurait été impossible d'en entrer plus.

Rappelez-vous qu'on peut voir toutes les relations entre les tables en cliquant sur le bouton « Relations » qui se trouve dans l'onglet « Outils de base de données » du ruban. On obtient ceci :



En plus de la relation entre un cours et un prof qu'on vient d'examiner, on constate aussi une relation entre un cours et un programme (un programme correspond à plusieurs cours), ainsi qu'entre un cours et un local, un programme et un groupe, un groupe et des élèves… Les relations sont nombreuses, mais importantes pour l'intégrité des données.

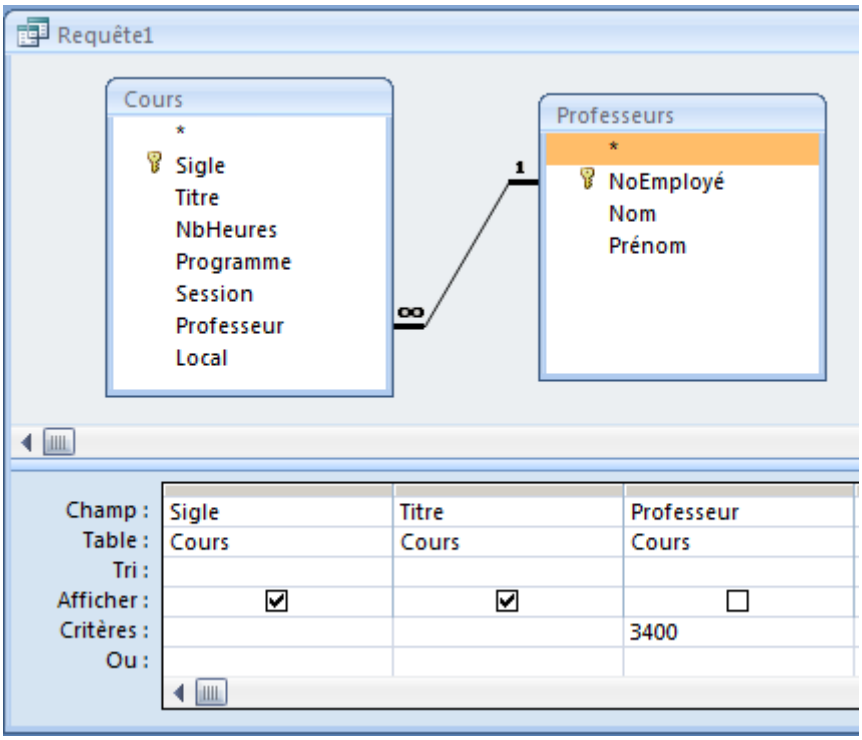
Notez le lien plus pâle et sans symbole entre la table Locaux et CatLocaux. Elle signifie qu'il y a une relation entre ces deux tables, mais une relation qui n'est pas forte : il est possible alors d'entrer une catégorie de locaux qui ne correspond à aucun local, ou d'entrer un local qui possède une catégorie inexistante. On dira que cette relation **ne renforce pas l'intégrité référentielle**, contrairement aux autres.

Notez aussi que le côté « 1 » d'une relation pointe toujours sur le champ qui sert de **clé primaire** à la table – c'est très courant et souhaitable.

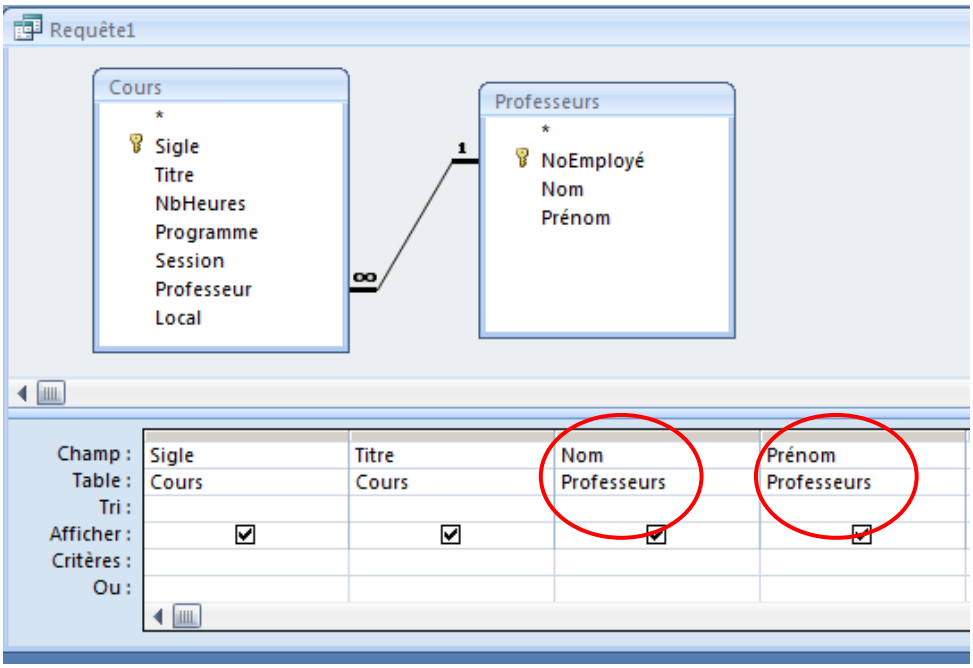
Le fait que les relations existent entre les tables fait en sorte qu'**elles apparaîtront automatiquement** lorsqu'on utilisera ces tables dans des requêtes.

3. Les requêtes à plusieurs tables – suite et fin

Reprenons notre requête à la recherche des cours de Paul Verveine :



Pour l'instant, on a ajouté la table Professeurs, mais on n'a rien changé aux critères, ce qui n'est donc pas très utile. Si on veut chercher par le nom, on sait qu'on ne pourra pas mettre ce nom dans la colonne Professeur. On va donc enlever cette colonne et la remplacer par les colonnes Nom et Prénom de la table Professeurs, de cette façon :



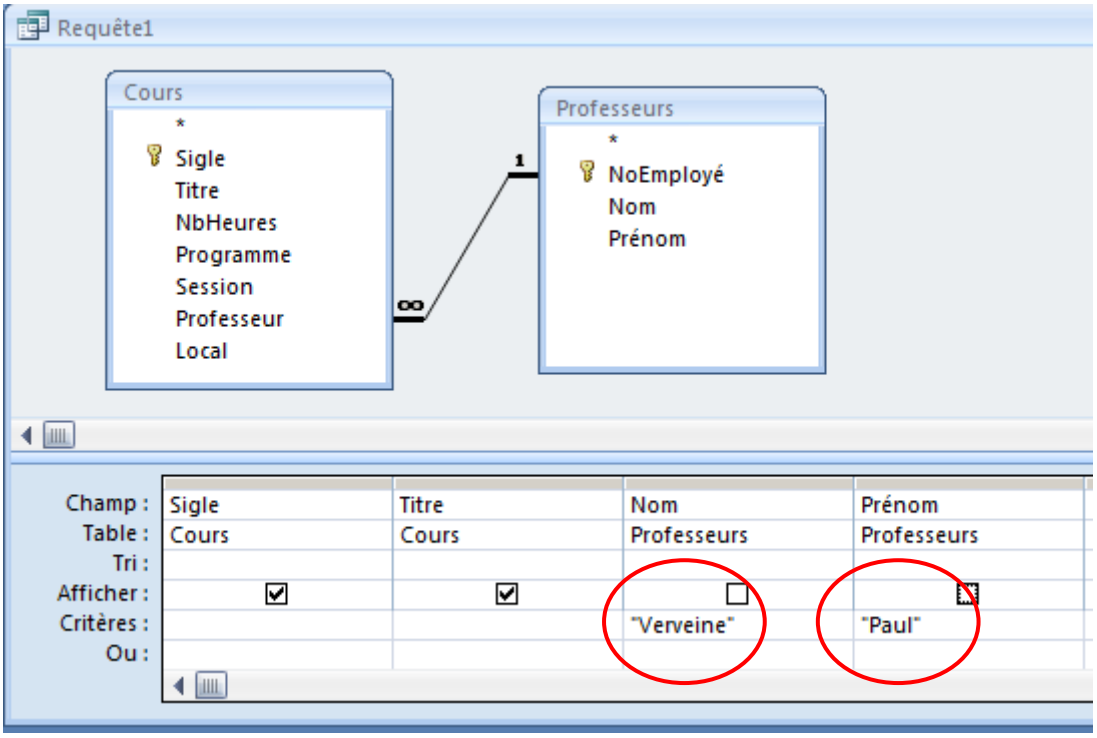
Si on exécute cette requête, qu'obtient-on? Ceci :

Requête1

Sigle	Titre	Nom	Prénom
393-EDA-03	Analyse de la fonction de travail en tech doc	Verveine	Paul
393-EDW-04	Acquisition de documents	Verveine	Paul
393-DE1-LG	Gestion de collections	Ladouceur	Yvon
393-EDJ-04	Analyse et indexation	Ladouceur	Yvon
393-EDK-04	Système de gestion de documents administratifs et d'arc	Romarin	Daniel
410-EDA-03	Gestion administrative de milieux documentaires	Romarin	Daniel

C'est comme si les deux tables avaient été fusionnées pour n'en former qu'une seule. Pour chaque cours de la table Cours, le nom et le prénom du professeur a été trouvé dans la table Professeurs en utilisant le numéro d'employé. Exactement ce qu'on a fait manuellement tantôt!

Maintenant, on peut ajouter les critères pour aller chercher uniquement le cours de Paul Verveine :



Notez qu'on n'affiche pas ces nom et prénom puisque ce n'est pas demandé (et pas vraiment utile, on sait qu'on verra toujours Paul Verveine là-dedans pour chaque enregistrement).

Les requêtes par agrégats (groupes)

1. *L'agrégation : regrouper nos données à des fins statistiques*

Le principe de l'agrégation est le suivant:

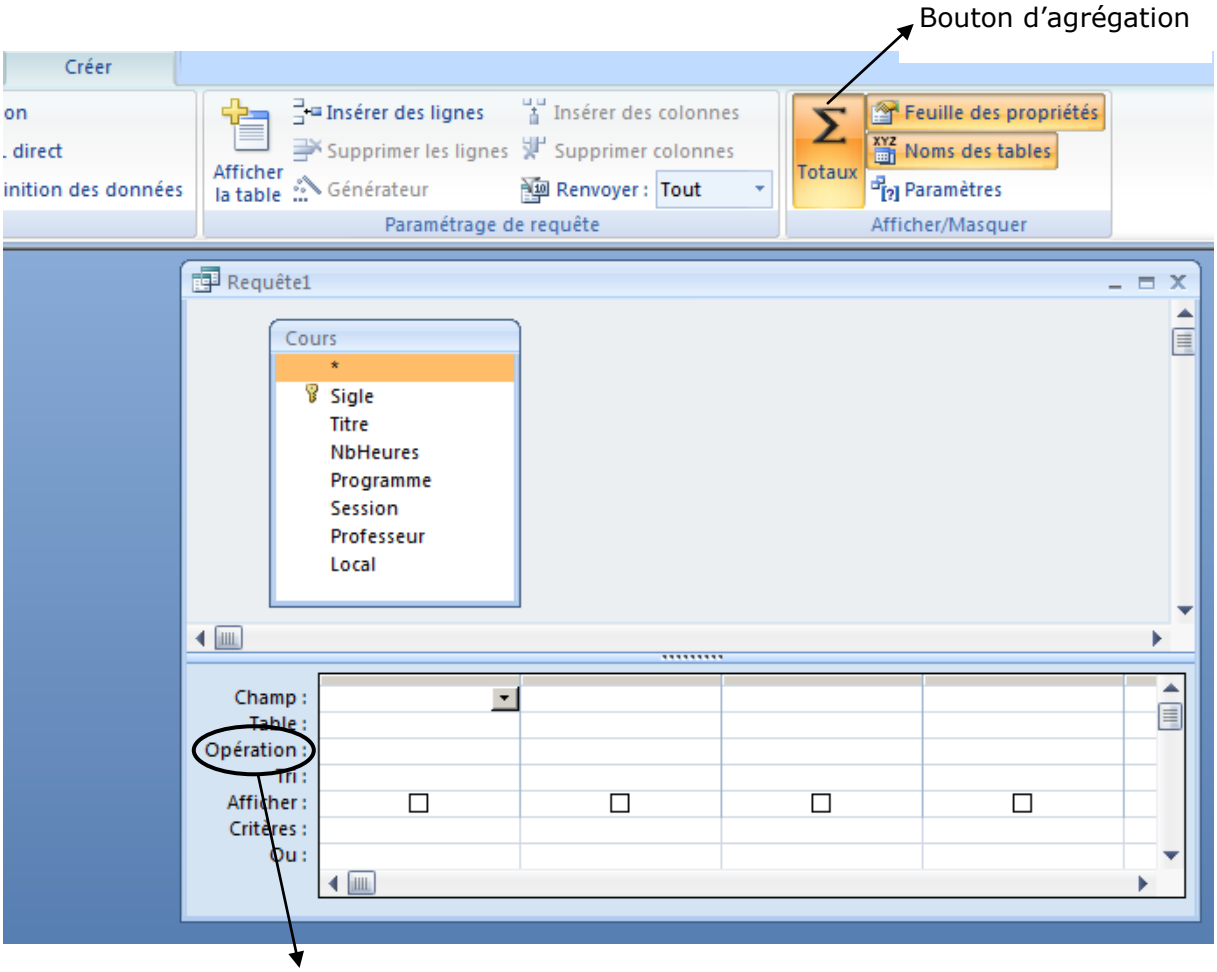
- On prend tous les enregistrements d'une table et on forme des groupes;
- Ces groupes sont formés des enregistrements qui ont **la même valeur** pour une colonne donnée;
- On retourne **une ligne par groupe**;
- On retourne habituellement en plus des **statistiques** sur le groupe (nombre de lignes dans le groupe, somme d'un champ, moyenne d'un champ, etc.)

Pour utiliser l'agrégation dans une requête, on doit appuyer sur le bouton « Totaux » qui apparaît dans le deuxième onglet **Créer** lorsque l'on construit une requête. Ceci ajoutera une nouvelle ligne **Opération** dans notre fenêtre de requête.

Voyons voir comment construire une requête qui fait de l'agrégation en créant une nouvelle requête dans notre base de données du « Petit collège Lionel-Groulx ».

Ajoutons la table *Cours* comme source de données de la requête. Jusque là, rien de nouveau.

Appuyons maintenant sur le bouton **Totaux** mentionné ci-haut. Nous obtiendrons alors la fenêtre suivante:



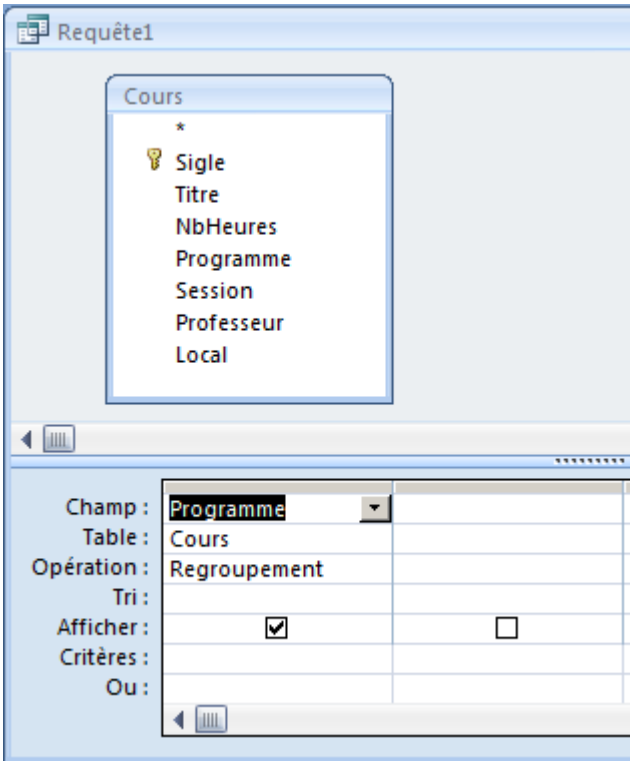
C'est dans la nouvelle ligne **Opération** que nous allons définir quel champ permet de créer les groupes et sur quels champs on va faire des statistiques.

Par exemple, si l'on voulait connaître le nombre de cours dans chaque programme, on déciderait de grouper par *Programme*, puis de compter n'importe quel autre champ. Essayons-le.

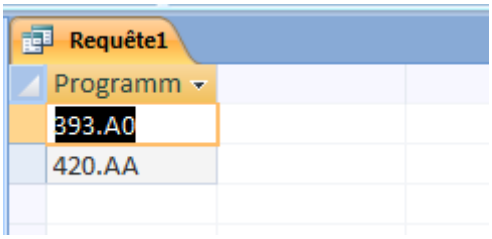
Sélectionnons d'abord le champ *Programme* comme champ de la requête (n'oubliez pas: vous pouvez double-cliquer sur le champ dans la table, le glisser vers une colonne ou carrément sélectionner le champ dans la liste déroulante de la colonne où vous voulez l'envoyer).

À la ligne **Opération**, dans notre nouvelle colonne *Programme*, on choisira **Regroupement** (c'est ce qui va être mis là par défaut de toute façon). Ceci signifie qu'Access va créer un groupe pour chaque programme différent trouvé dans ce champ.

À ce stade notre requête ressemble à ceci:



Si on l'exécute dès maintenant, on va voir tous les groupes qu'Access a créés, c'est-à-dire tous les programmes différents qui existent dans la table *Cours* :



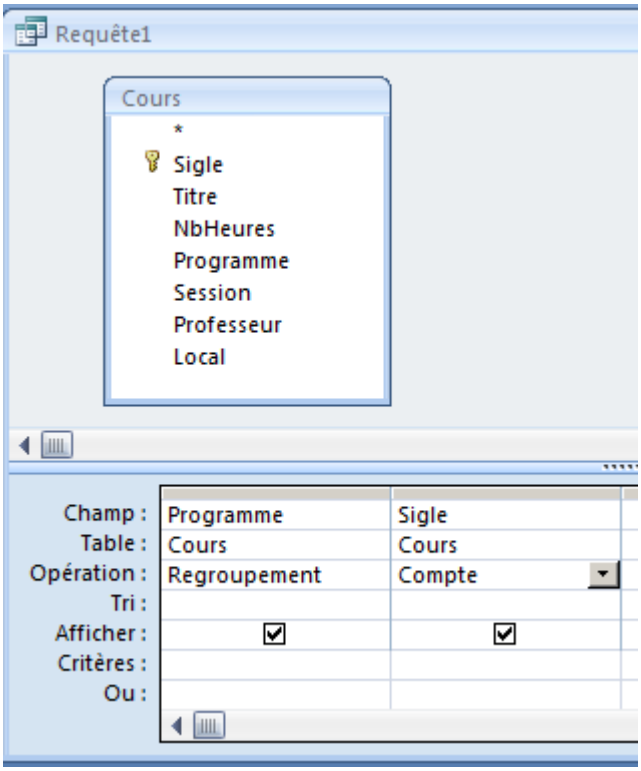
Cela confirme donc ce que l'on savait déjà : il y a deux programmes dans notre table, soit 393.A0 et 420.AA. C'est déjà une information intéressante en soi, mais rappelez-vous que nous voulions savoir combien de cours existent pour chaque programme.

Revenons donc à la construction de la requête et modifions-la pour atteindre ce but. Pour ce faire, on va ajouter n'importe quel champ à la requête. Prenons *Sigle*. À la ligne **Opération**, on choisira cette fois **Compte**.

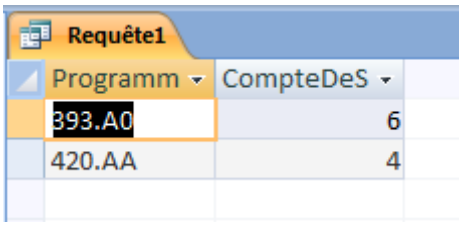
Ceci demandera à Access de regrouper les cours par programme et d'afficher, pour chaque programme, combien de sigles il y a (donc de faire un compte des sigles). Il **ne s'agit pas** ici de compter combien il y a de sigles **différents**, mais simplement « **combien de sigles existent en tout dans la table** ». Access comptera tous les

enregistrements qui ont une valeur dans ce champ – les champs vides ne sont pas comptés. Pour obtenir le nombre de cours par programme, on pourrait donc compter n'importe quel champ, tant qu'on est certain que tous les cours ont une valeur pour ce champ. Le champ *Sigle* est clairement le plus approprié : comme c'est une clé primaire, on est certain que chaque cours aura un sigle, donc on comptera bien tous nos cours.

Notre requête ressemble donc maintenant à ceci :



Et en l'exécutant, on obtient ceci :

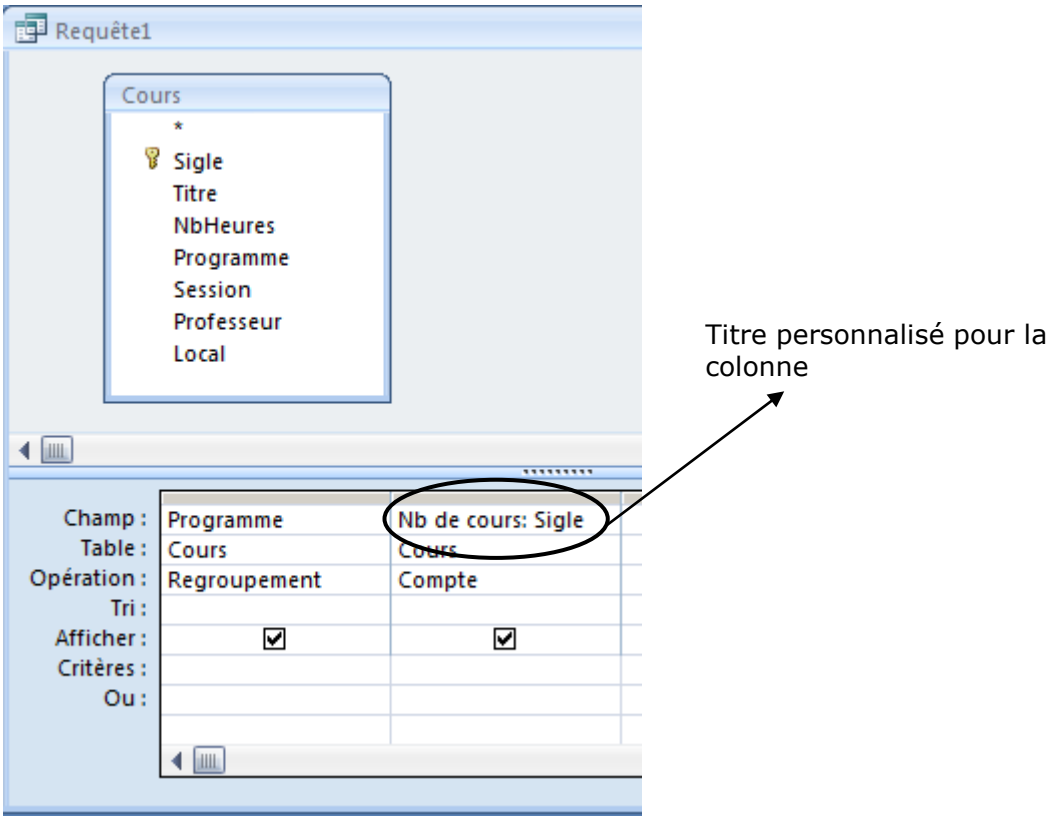


On sait donc maintenant qu'il y a 6 cours de techniques de la documentation et 4 cours d'informatique de gestion dans notre base de données. Notez qu'on aurait obtenu le même résultat en comptant le nombre de titres ou de locaux...

Titres personnalisés

Remarquez en passant le titre de la colonne de chiffres : « CompteDeSigle »... Pas très élégant. C'est dans ce genre de situation qu'il est intéressant de définir un titre nous-mêmes plutôt que de laisser Access en créer un.

Pour ce faire, on retournera dans la création de notre requête, et on ajoutera un titre, suivi d'un deux-points (:) dans la case **Champ**, juste avant le nom du champ lui-même :



Lorsqu'Access voit quelque chose avant le nom du champ, il sait qu'il s'agit d'un titre de colonne et il l'affichera à la place du titre standard.

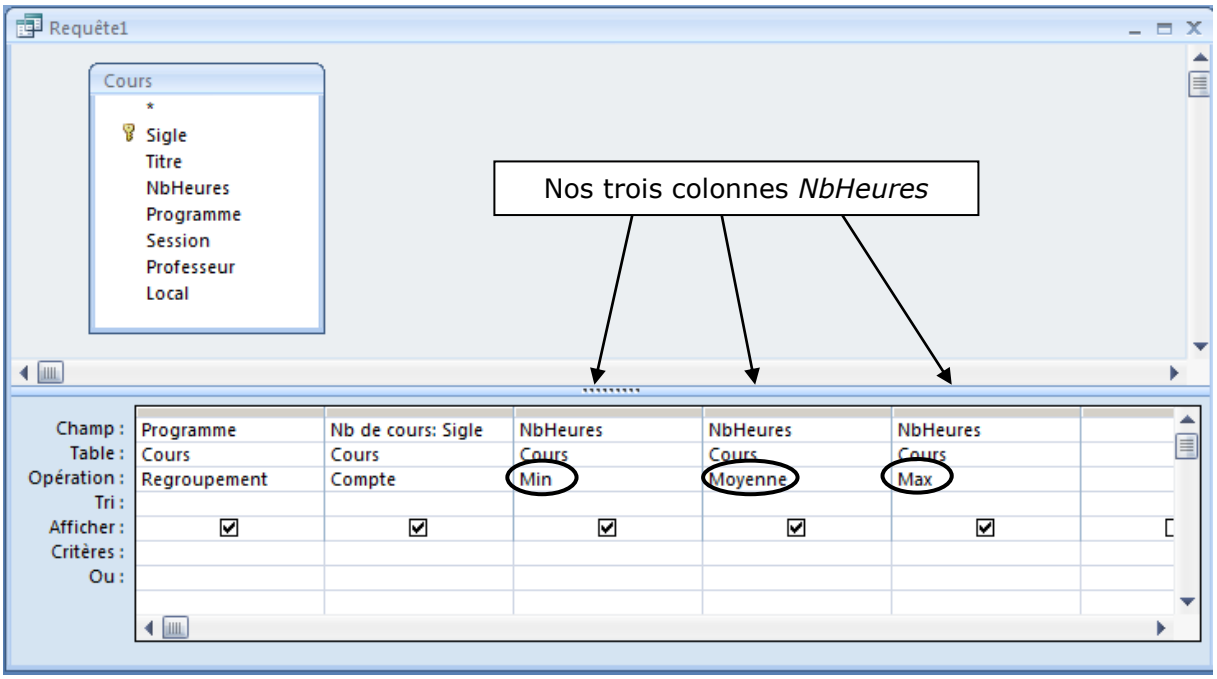
Plus de statistiques

Il est possible de créer des requêtes de groupage pour obtenir toutes sortes de statistiques intéressantes. On peut bien sûr afficher plusieurs statistiques pour le même groupe dans une même requête.

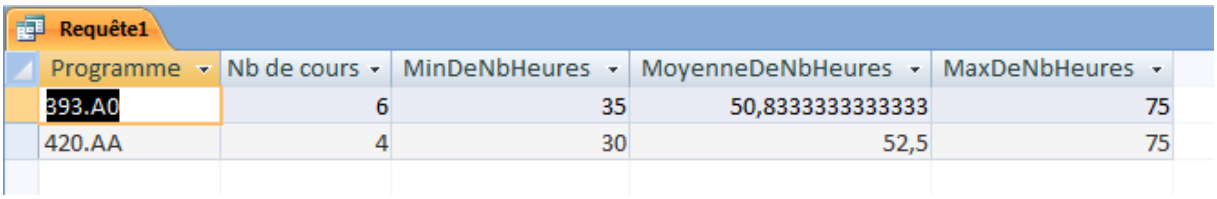
Par exemple, essayons de faire une seule requête qui donne d'un seul coup, pour chaque programme :

- Le nombre de cours définis pour le programme
- La durée minimale, moyenne et maximale des cours (en nombre d'heures)

Pour ce faire, on va partir de la requête qu'on a déjà construite et on ajoutera une colonne pour chacune des trois nouvelles statistiques qu'on veut. Comme elles concernent toutes le nombre d'heures, on ajoutera trois fois le champ *NbHeures*. Par contre, dans la ligne **Opération**, on choisira une statistique différente pour chacune : **Min**, **Moyenne** et **Max**. On obtient alors :

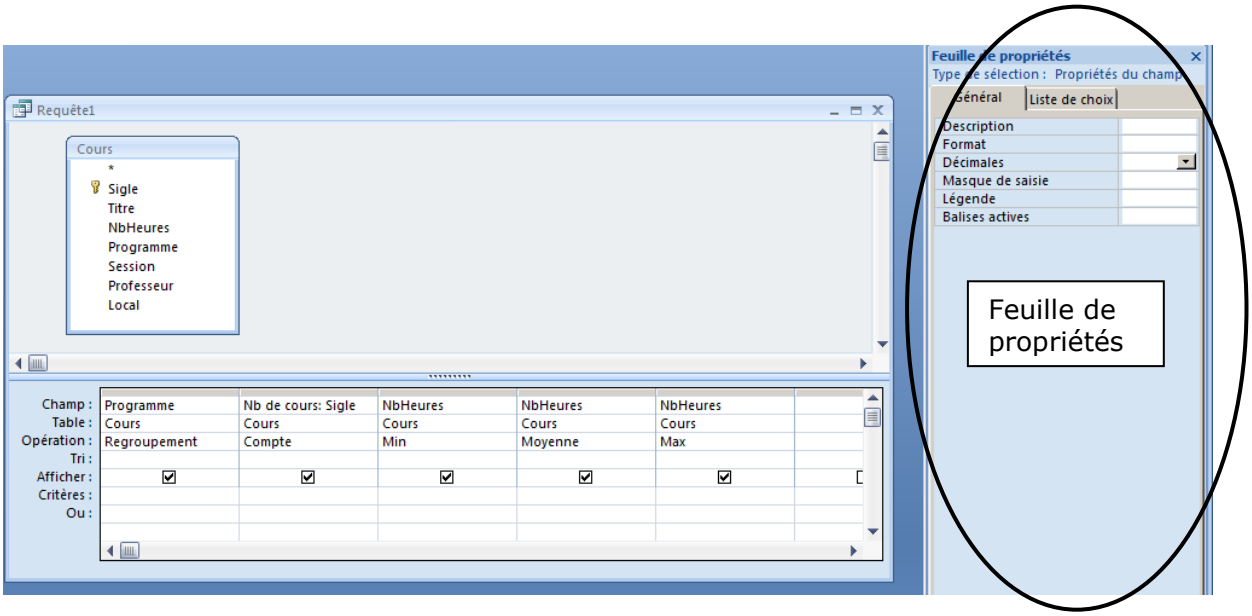


Son exécution nous donne le résultat suivant :



On pourrait améliorer l'apparence en ajoutant des titres personnalisés (on a vu comment tantôt) et en diminuant le nombre de décimales de la moyenne. Pour réaliser cette dernière opération, rien de plus simple :

- On revient en mode création
- On clique dans la colonne de moyenne
- Si la fenêtre de propriétés n'est pas déjà affichée à droite, on clique avec le bouton de droite et on choisit **Propriétés**. Le but est d'arriver à ceci :



- Dans l'onglet **Général** de la feuille de propriétés, on met **Standard** dans le champ **Format** et 2 dans le champ **Décimales**.

Le format **Standard** est en effet ce qu'on utilisera pour avoir un format numérique « joli », avec des espaces pour séparer les milliers et un nombre de décimales déterminé d'avance. Notez qu'on aurait pu aussi choisir monétaire ou pourcentage, si ça avait été approprié.

Nos modifications donnent finalement le résultat suivant :

Requête1

Programme	Nb de cours	MinDeNbHeures	MoyenneDeNbHeures	MaxDeNbHeures
393.A0	6	35	50,83	75
420.AA	4	30	52,50	75

Grouper par plusieurs colonnes

On peut créer des groupes définis par plusieurs colonnes si on a besoin de certaines précisions supplémentaires. Par exemple, on pourrait vouloir savoir combien de cours on a dans chaque programme, mais aussi pour chaque professeur. Il suffit alors d'utiliser l'opération **Regroupement** sur plusieurs colonnes à la fois.

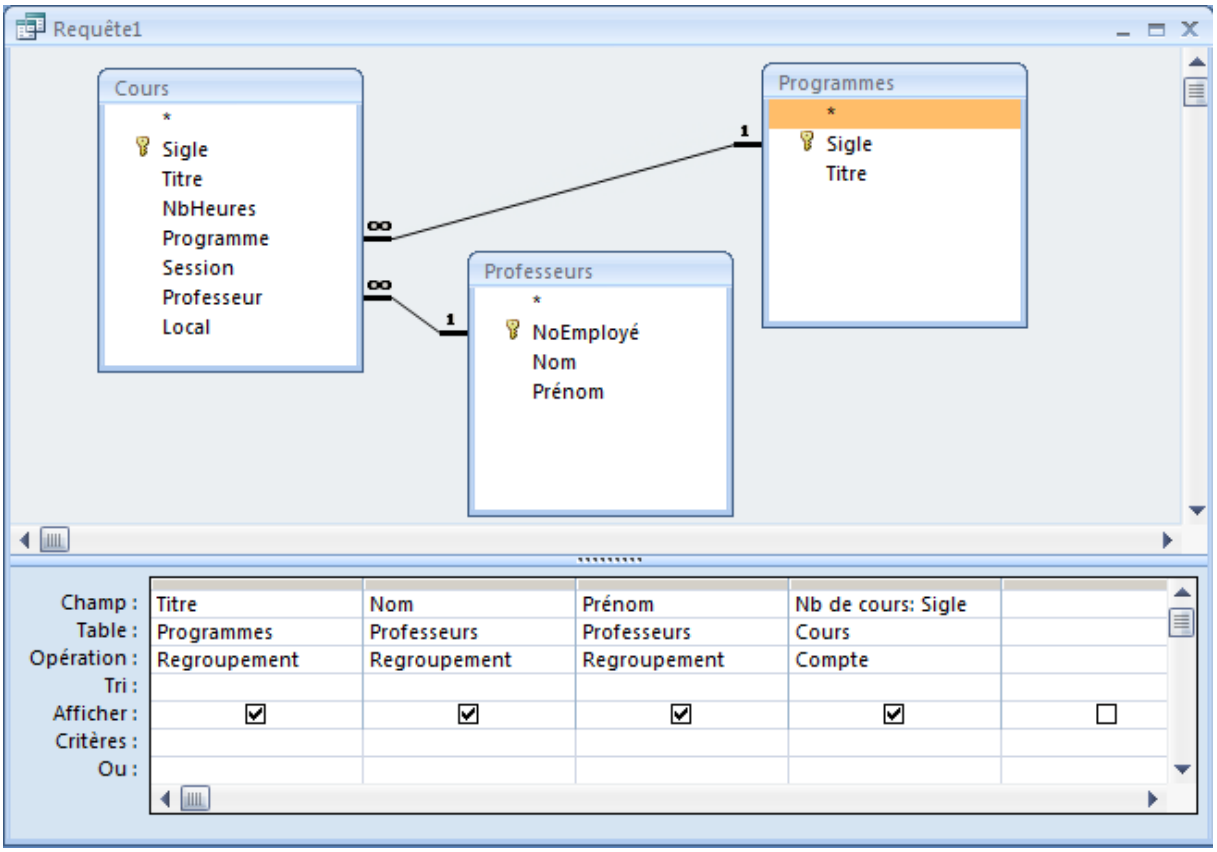
On peut par exemple choisir de regrouper sur *Programme* et *Professeur*, puis de faire un compte sur le *Sigle*, afin d'obtenir les résultats suivants:

Requête1

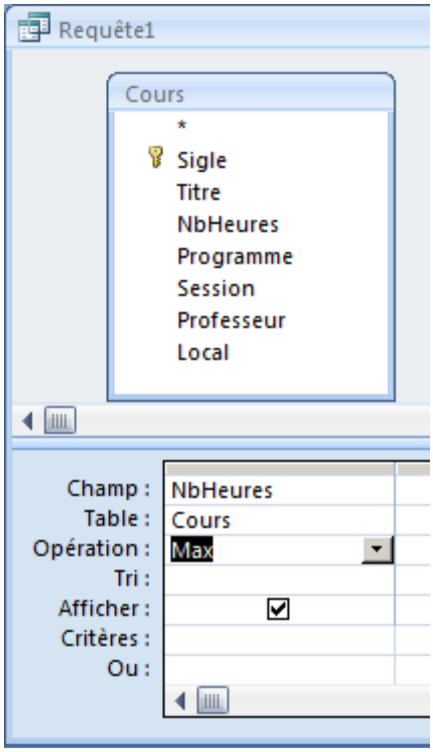
Programme	Professeı	Nb de cours
393.A0	3400	2
393.A0	3401	2
393.A0	3402	2
420.AA	3402	1
420.AA	3500	1
420.AA	3510	1
420.AA	3511	1

On voit donc que le professeur #3400 donne deux cours dans 393.A0. Le professeur #3402 en donne deux dans 393.A0 et un dans 420.AA, etc.

Notez que rien ne nous empêche de mettre en pratique les notions apprises au chapitre précédent et d'ajouter des tables à notre requête afin de regrouper par nom de programme (plutôt que d'utiliser son sigle), et par nom de professeur (plutôt que par son numéro d'employé). On obtient alors la requête suivante :



Des statistiques sans groupe



Lorsque l'on ne fait aucun regroupement, mais que l'on demande quand même des statistiques, Access traite alors toute la table comme un seul grand groupe. Par exemple, si l'on veut connaître le cours le plus long de la table, on peut simplement faire une requête avec une seule colonne : le *NbHeures*, avec **Max** comme opération.

À ce moment, Access groupe tous les cours de la table en un seul groupe, puis trouve la plus longue durée du groupe : 75 heures.

Mais comment peut-on connaître le titre de ce cours de 75 heures?

Qu'arrive-t-il si on ajoute le titre du cours dans notre requête, selon vous? Pourquoi?

La bonne solution à ce problème passe par une requête multi-tables comme nous l'avons vu au chapitre précédent. N'oubliez pas qu'il est possible de baser une requête sur une autre requête et de faire nous-mêmes des liens entre une requête et une table, comme vous l'avez fait au dernier numéro du labo précédent! Ce défi pour les pros est laissé en exercice à l'étudiant.