Partie 3

DRL – DATA RETRIEVAL LANGUAGE

La clause « SELECT »

Limiter et ordonner

Les fonctions

GROUP BY

Jointures

Sous-requêtes

La clause « SELECT »

SELECT *colonne1*, *colonne2*, *colonne3*, …
FROM *nom_table*

- La casse n'a pas d'importance, mais on prendra l'habitude d'écrire les mots-clés du langage en majuscules

- On écrira généralement les clauses (***« SELECT »***, ***« FROM »***, etc.) sur des lignes différentes afin d'indenter au mieux le code

La clause « SELECT »

Sélectionner toutes les colonnes et toutes les lignes

```
SELECT *
  FROM student
```

Sélectionner toutes les lignes de certaines colonnes uniquement

```
SELECT first_name, last_name, year_result
  FROM student
```

La clause « SELECT » : **Alias**

```
SELECT first_name as [Prénom]
       , last_name 'Nom de famille'
       , section_id Section
       , year_result as "Résultat annuel"
  FROM student
```

- Les alias servent à renommer l'intitulé des colonnes à l'affichage des données. Cela ne change rien au niveau des données contenues dans les tables, bien entendu
- Le mot-clé ***« as »*** n'est pas obligatoire
- ***Sous SQL-Server***, les alias doivent être accompagnés de *guillemets simples*, *doubles* ou de *crochets* s'ils contiennent des caractères spéciaux, des accents ou des espaces

Prénom	Nom de famille	Section	Résultat annuel
Georges	Lucas	1320	10
Clint	Eastwood	1010	4
Sean	Connery	1020	12

La clause « SELECT » : Alias

ORACLE

```
SELECT first_name AS Prénom
      , last_name "Nom de famille"
  FROM student ;
```

Sous Oracle, les alias tenant en un seul mot ne doivent pas être délimités. Il sera par contre nécessaire d'utiliser les doubles-guillemets lorsqu'ils en contiennent plus d'un

PRÉNOM	Nom de famille
Georges	Lucas
Clint	Eastwood
Sean	Connery
Robert	De Niro
Kevin	Bacon
Kim	Basinger
Johnny	Depp
Julia	Roberts
Natalie	Portman

La clause « SELECT » :
Opérations Arithmétiques

```
SELECT first_name, year_result
      , (year_result/20)*100 as [Nouveau Résultat]
  FROM student
```

Opérateurs autorisés

Opérateur	Signification
/	Division
*	Multiplication
+	Addition **/ Concaténation**
-	Soustraction

first_name	year_result	Nouveau Résultat
Georges	10	50
Clint	4	20
Sean	12	60
Robert	3	15
Kevin	16	80
Kim	19	95
Johnny	11	55
Julia	17	85
Natalie	4	20
Georges	4	20

La clause « SELECT » : **Concaténation**

```
SELECT first_name + ' ' + last_name as [Nom complet]
      , [login] + student_id as 'Code étudiant'
  FROM student
```

Nom complet	Code étudiant
Georges Lucas	glucas1
Clint Eastwood	ceastwoo2
Sean Connery	sconnery3
Robert De Niro	rde niro4
Kevin Bacon	kbacon5
Kim Basinger	kbasinge6
Johnny Depp	jdepp7
Julia Roberts	jroberts8
Natalie Portman	nportman9
Georges Clooney	gclooney10
Andy Garcia	agarcia11

La clause « SELECT » : Concaténation

ORACLE

```
SELECT first_name || ' ' || last_name AS "Nom complet"
      , login || student_id "Code étudiant"
  FROM student ;
```

Sous Oracle, la concaténation se fait grâce au symbole **« || »**

Nom complet	Code étudiant
Georges Lucas	glucas1
Clint Eastwood	ceastwoo2
Sean Connery	sconnery3
Robert De Niro	rde niro4
Kevin Bacon	kbacon5
Kim Basinger	kbasinge6
Johnny Depp	jdepp7
Julia Roberts	jroberts8
Natalie Portman	nportman9
Georges Clooney	gclooney10

La clause « SELECT » : **DISTINCT**

```
SELECT DISTINCT first_name
  FROM student
```

Sans DISTINCT

first_name
Alyssa
Andy
Bruce
Clint
David
Georges
Georges
Halle
Jennifer
Johnny

first_name
Alyssa
Andy
Bruce
Clint
David
Georges
Halle
Jennifer
Johnny
Julia

Avec DISTINCT

La clause « SELECT » : DISTINCT

```
SELECT DISTINCT first_name, year_result
  FROM student
```

Sans DISTINCT

first_name	year_result
Tom	4
Tom	4
Sophie	6
Shannen	2
Sean	12
Sarah	7
Sandra	2
Robert	3
Reese	7
Natalie	4

Avec DISTINCT

first_name	year_result
Tom	4
Sophie	6
Shannen	2
Sean	12
Sarah	7
Sandra	2
Robert	3
Reese	7
Natalie	4
Michael J	3

La clause « SELECT » : **SANS FROM**

SELECT *col1* **as** *alias_col1*, *col2*, *col3*, …

Sous SQL Server, la clause ***« SELECT »*** peut s'utiliser seule, si le but est simplement d'afficher un résultat structuré dans un tableau

```
SELECT GETDATE() as [Date du jour]
      , 'Vive le SQL !'
```

Date du jour	(No column name)
2014-06-30 15:18:00.263	Vive le SQL !

La clause « SELECT » : **DUAL**

ORACLE

SELECT *col1* **as** *alias_col1*, *col2*, *col3*, … **FROM DUAL**

Sous Oracle, la clause ***« SELECT »*** ne peut pas s'utiliser seule, mais il est également possible d'afficher un résultat divers structuré dans un tableau, grâce à la table temporaire ***« DUAL »***

```
SELECT CURRENT_TIMESTAMP AS "Date du jour"
     , 'Vive le SQL !'
  FROM DUAL ;
```

Date du jour	'VIVELESQL!'
30/06/14 19:15:06,195000000 EUROPE/BERLIN	Vive le SQL !

Limiter et ordonner

SELECT *colonne1*, *colonne2*, *colonne3*, …
FROM *nom_table*
WHERE *conditions*
ORDER BY *liste_colonnes*

- La clause ***« WHERE »*** permet de limiter le nombre de lignes sélectionnées
- La clause ***« ORDER BY »*** permet de trier les résultats affichés, selon une ou plusieurs colonnes données
- Comme vu précédemment, ***ces clauses ne sont pas obligatoires*** mais si elles sont présentes, elles apparaissent dans cet ordre
- ***Il n'y a qu'une seule clause de chaque type.*** Il n'y aura donc jamais deux ***« WHERE »*** dans une même requête

Limiter et ordonner : « **WHERE** »

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result >= 16
```

Opérateurs de comparaison

Opérateur	Signification
>	Plus grand
<	Plus petit
>=	Plus grand ou égal
<=	Plus petit ou égal
<>	Différent
!	Négation *(« !> » = « pas plus grand »)*

student_id	first_name	last_name	year_result
5	Kevin	Bacon	16
6	Kim	Basinger	19
8	Julia	Roberts	17
11	Andy	Garcia	19
18	Jennifer	Gamer	18
25	Halle	Berry	18

Limiter et ordonner : **BETWEEN**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result BETWEEN 10 AND 16
```

Liste des étudiants ayant obtenu un résultat annuel compris entre 10 et 16, ces valeurs incluses

Cela revient à demander la liste des étudiants qui ont un résultat
plus grand ou égal à 10 ET ***plus petit ou égal à 16***

student_id	first_name	last_name	year_result
1	Georges	Lucas	10
3	Sean	Connery	12
5	Kevin	Bacon	16
7	Johnny	Depp	11
23	Keanu	Reeves	10

Limiter et ordonner : BETWEEN

```
SELECT first_name, last_name, birth_date
  FROM student
 WHERE birth_date BETWEEN '1960-01-01' AND '1970-12-31'
```

Les bornes de l'intervalle doivent être du même type que la valeur comparée. Dans cet exemple, les chaînes de caractères ***'1960-01-01'*** et ***'1970-12-31'*** seront automatiquement converties en dates afin de pouvoir être comparées aux valeurs de la colonne ***« birth_date »***

first_name	last_name	birth_date
Johnny	Depp	1963-06-09 00:00:00.000
Julia	Roberts	1967-10-28 00:00:00.000
Georges	Clooney	1961-05-06 00:00:00.000
Tom	Cruise	1962-07-03 00:00:00.000
Sophie	Marceau	1966-11-17 00:00:00.000
Michael J.	Fox	1969-06-20 00:00:00.000
Sandra	Bullock	1964-07-26 00:00:00.000
Keanu	Reeves	1964-09-02 00:00:00.000
Halle	Berry	1966-08-14 00:00:00.000

Limiter et ordonner : **IN**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result IN(12,16,18)
```

Liste des étudiants dont le résultat annuel est ***égal à 12*** OU ***égal à 16*** OU ***égal à 18***

student_id	first_name	last_name	year_result
3	Sean	Connery	12
5	Kevin	Bacon	16
18	Jennifer	Garner	18
25	Halle	Berry	18

Limiter et ordonner : IN

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE first_name IN('Tom','Jennifer','Halle')
```

L'opérateur ***« IN »*** permet de comparer tous types de données, tant que les valeurs entre parenthèses sont bien du même type que la valeur comparée. ***La casse n'a PAS d'importance***

student_id	first_name	last_name	year_result
13	Tom	Cruise	4
18	Jennifer	Garner	18
20	Tom	Hanks	8
25	Halle	Berry	18

Limiter et ordonner : **LIKE**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE first_name LIKE 'j%'
```

L'opérateur ***« LIKE »*** est utilisé pour comparer des chaînes de caractères entre elles
Le symbole ***« % »*** peut être utilisé pour remplacer ***de 0 à N caractères***
Le symbole ***« _ »*** peut être utilisé pour remplacer ***1 caractère***

student_id	first_name	last_name	year_result
7	Johnny	Depp	11
8	Julia	Roberts	17
18	Jennifer	Gamer	18

Limiter et ordonner : LIKE

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE last_name LIKE '%oo_'
```

Liste des étudiants dont les trois dernières lettres du nom de famille sont « o », « o » et un caractère indéfini

student_id	first_name	last_name	year_result
2	Clint	Eastwood	4
14	Reese	Witherspoon	7

Limiter et ordonner : **NOT**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result NOT BETWEEN 10 AND 15
```

L'opérateur ***« NOT »*** marque la négation des opérateurs ***« BETWEEN »***, ***« IN »*** et ***« LIKE »***

student_id	first_name	last_name	year_result
2	Clint	Eastwood	4
4	Robert	De Niro	3
5	Kevin	Bacon	16
6	Kim	Basinger	19
8	Julia	Roberts	17
9	Natalie	Portman	4
10	Georges	Clooney	4
11	Andy	Garcia	19
12	Bruce	Willis	6
13	Tom	Cruise	4

Limiter et ordonner : NOT

Liste des étudiants dont le nom de famille ne contient pas de « e »

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE last_name NOT LIKE '%e%'
```

Liste des étudiants dont le résultat annuel est différent de 12, 16 ou 18

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result NOT IN (12,16,18)
```

Limiter et ordonner : **IS (NOT) NULL**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE year_result IS NULL
```

Afin de déterminer si une valeur est ***« NULL »*** ou non, il faudra utiliser la syntaxe ***« IS NULL »*** dont la négation sera ***« IS NOT NULL »***

student_id	first_name	last_name	year_result
5	Kevin	Bacon	NULL
7	Johnny	Depp	NULL
10	Georges	Clooney	NULL

Limiter et ordonner : **AND**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE first_name like 'J%'
       AND year_result >= 10
```

L'opérateur ***« AND »*** permet de combiner plusieurs conditions en même temps

Une ligne doit répondre simultanément à toutes les conditions pour faire partie du résultat

student_id	first_name	last_name	year_result
7	Johnny	Depp	11
8	Julia	Roberts	17
18	Jennifer	Gamer	18

Limiter et ordonner : **OR**

```
SELECT student_id, first_name, last_name, year_result
  FROM student
 WHERE first_name like 'J%'
       OR year_result >= 10
```

Tout comme son compère ***« AND »***, l'opérateur ***« OR »*** permet également de combiner plusieurs conditions en même temps

Il suffit qu'une ligne réponde à l'une des conditions pour faire partie du résultat

student_id	first_name	last_name	year_result
1	Georges	Lucas	10
3	Sean	Connery	12
5	Kevin	Bacon	16
6	Kim	Basinger	19
7	Johnny	Depp	11
8	Julia	Roberts	17
11	Andy	Garcia	19

Limiter et ordonner : **Précédence**

Il est conseillé d'***utiliser des parenthèses*** afin de forcer le système à tester les conditions dans l'ordre souhaité. Sous SQL-Server, si aucune parenthèse n'est utilisée, l'ordre de précédence suivant est appliqué

Ordre	Opérateurs évalués
1	Multiplication, Division, Modulo
2	Addition, Soustraction, Concaténation
3	Opérateurs de comparaisons (=,>,<,!=,…)
4	NOT
5	AND
6	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
7	Affectation (=)

http://msdn.microsoft.com/fr-fr/library/ms190276.aspx

Limiter et ordonner : « **ORDER BY** »

```
SELECT student_id, first_name, last_name, year_result
FROM student
ORDER BY last_name ASC
```

La clause ***« ORDER BY »*** permet de trier le résultat d'une requête selon une ou plusieurs colonnes

Le tri peut se faire de façon croissante (***« ASC »*** – ascendant) ou décroissante (***« DESC »*** – descendant) sur les valeurs. La valeur par défaut est ***« ASC »***

Il est possible de trier selon une colonne qui n'est pas affichée

student_id	first_name	last_name	year_result
5	Kevin	Bacon	16
6	Kim	Basinger	19
25	Halle	Berry	18
22	Sandra	Bullock	2
10	Georges	Clooney	4
3	Sean	Connery	12
12	Tom	Cruise	4

Limiter et ordonner : « ORDER BY »

```
SELECT section_id
      , first_name + ' ' + last_name as 'Nom complet'
FROM student
ORDER BY section_id, 'Nom complet' DESC
```

Il est possible de trier les résultat ***en fonction d'un alias*** de colonne ainsi que sur ***une combinaison de plusieurs colonnes***

section_id	Nom complet
1010	Sandra Bullock
1010	Natalie Portman
1010	Clint Eastwood
1010	Bruce Willis
1020	Tom Hanks
1020	Tom Cruise
1020	Sean Connery
1020	Sarah Michelle Gellar

Auto-Evaluation

N'oubliez pas de prendre le temps d'évaluer le niveau de maîtrise que vous estimez avoir acquis personnellement concernant les notions abordées dans ce module !

Rappel de la signification des lettres dans les tableaux d'auto-évaluation :

- **Parfait (P) :** vous avez parfaitement compris cette notion et vous vous sentez à votre aise
- **Satisfaisant (S) :** vous avez compris de quoi il s'agit mais la pratique vous manque
- **Vague (V) :** vous savez de quoi il s'agit, mais cela reste un peu vague dans votre esprit. Une explication supplémentaire du formateur ou une bonne révision de votre part s'impose
- **Insatisfaisant (I) :** Vous n'avez pas du tout compris la notion abordée, il faut tout faire pour y remédier !

Auto-Evaluation

Notions à évaluer

Notions	P	S	V	I
Ordre « SELECT … FROM »				
Alias				
Concaténation et conversion de type de données (CONVERT)				
Clause « WHERE »				
Opérations arithmétiques				
Opérateurs « BETWEEN », « IN », « LIKE » et leur négation				
Comparaison avec une valeur « NULL »				
Opérateurs « AND » et « OR »				
Clause « ORDER BY »				