Module 3 : Le langage SQL et les sous-langages DDL et DML

Vérifier et pratiquer

Présentation

Dans cette activité, nous vous proposons une série de questions. Le questionnaire vous permettra de consolider vos connaissances et de revenir, au besoin, sur certains passages des textes que vous avez étudiés au cours de l'activité précédente.

Étape par étape…

1. **Accédez au Questionnaire sur le langage SQL.**

   Ce questionnaire se compose de deux parties. La première partie comprend des questions conceptuelles sur les sujets abordés dans le module. La deuxième partie, plus pratique, contient des exercices de formulation ou d'analyse d'expressions en langage SQL.

Première partie

Dans cette partie, nous vous proposons une série de questions qui vous permettront de valider votre compréhension des notions traitées dans les textes du module. Nous vous invitons à bien réfléchir à la question et à sa réponse avant de regarder la solution et, au besoin, à revenir sur le texte pour consolider vos connaissances. Les questions sont indépendantes les unes des autres; vous n'êtes donc pas tenu d répondre dans l'ordre de leur présentation.

Question 1 : Parmi les affirmations suivantes, laquelle décrit le mieux le langage SQL?

1. SQL est un langage de programmation dédié au développement de bases de données.
2. SQL est un langage pour définir des requêtes sur des bases de données relationnelles.
3. SQL est un langage de définition, de mise à jour, d'interrogation, etc. de bases de données relationnelles.

Réponse : *c*

Explication

Le langage SQL est introduit dans le texte 3.1. À proprement parler, SQL n'est pas un langage de programmation (réponse *a*) comme le sont les langages C et JAVA. De plus, il n'est pas seulement utilisé pour définir des requêtes sur des bases de données relationnelles (réponse *b*) la définition des requêtes est une des facettes du langage SQL.

Question 2 : Qu'est-ce qui différencie les langages SQL 1, SQL 2 et SQL 3?

1. SQL 1 constitue le *niveau d'entrée* du langage, SQL 2 le *niveau intermédiaire* et SQL 3 le *niveau avancé*.
2. SQL 3 est une version corrigée du langage SQL 2 et ce dernier est une version corrigée du langage SQL 1.
3. SQL 1 est la première version du langage et les versions 2 et 3 sont des extensions de cette première version.

Réponse : *c*

Explication

L'évolution du langage SQL est présentée dans le texte 3.1, à la page 205 (introduction).

Question 3 : Parmi les propositions suivantes sur les *vues*, lesquelles sont vraies?

1. Une vue correspond à une table *virtuelle* dont seule la définition, sous la forme d'une requête SFW (select from where), est stockée et no le contenu.
2. Une vue correspond à une table dont la définition et le contenu sont stockés, dans la base et sous la forme d'une requête SFW.
3. Il est possible sous certaines conditions de modifier les données d'une vue.

Réponse : *a et c sont vraies.*

Explication : les vues sont définies et expliquées à la page 282.

Question 4 : Parmi ces propositions sur les *vues*, lesquelles sont fausses?

1. Le but d'une vue est d'offrir aux utilisateurs de la base de données une présentation des données qui soient adaptée à ses besoins en lui évitant la complexité d'une base de données.
2. Les vues, contrairement à toutes les tables, ne sont pas soumises à un contrôle d'accès.
3. Lorsque le schéma d'une base de données évolue, certaines requêtes ne sont plus valides. Les vues ne permettent cependant pas de restaurer la structure précédente des données.
4. Les vues permettent l'expression en SQL de requêtes complexes à rédiger sous la forme SFW.

Réponse : *b et c sont fausses.*

Explication : (b) Les vues sont soumises à un contrôle d'accès. On peut définir les vues qui ne sont accessibles qu'à une certaine classe d'utilisateurs, et on accorde alors ensuite à ces utilisateurs l'utilisation d'utiliser ces vues, mais en leur interdisant l'accès aux tables de la base. De plus, (c) les vues permettent de protéger les utilisateurs contre l'effet de certaines modifications d'une base de données. Une vue permet de restaurer la structure précédente des données, n'affectant pas les utilisateurs si les structures sont modifiées (p.284).

Question 5 : Une jointure se définit comme une opération qui consiste à coupler les lignes de deux ou plusieurs tables afin d'en extraire les données corrélées.

1. Vrai
2. Faux

Réponse : *a*

Explication : La jointure est expliquée à la page 242.

Question 6 : La jointure de deux tables exclut dans chacune d'elle les lignes célibataires, c'est-à-dire, les lignes qui n'ont pas de correspondant dans l'autre table.

1. Vrai
2. Faux

Réponse : *a*

Explication : En général, la proposition est vraie. Il est néanmoins possible de faire apparaître toutes les lignes dans le résultat d'une jointure en ajoutant spécifiquement les ligne célibataires, les données non pertinentes sont alors remplacées par des valeurs null. (Un exemple est proposé à la page 244).
Question 7 : Écrire la requête suivante en utilisant une jointure plutôt qu'une sous-requête (une requête imbriquée) :

```
select NCOM, DATECOM
fromCOMMANDE
whereNCLI in (select NCLI
                from CLIENT
                whereLOCALITE = "Poitiers")
```

Réponse :

Sous la forme d'une jointure, cette requête s'écrit :

```
select NCOM, DATECOM
from COMMANDE, CLIENT
where COMMANDE.NCLI = CLIENT.NCLI
and LOCALITE = "Poitiers"
```

Explication : voir l'exemple à la page 254.

Question 8 : Quel est le résultat retourné par la requête suivante en SQL?

```
SELECT NOM, AGE+1
FROM PERSONNES
WHERE AGE BETWEEN 0 AND 17;
```

1. Le nom de toutes les personnes ayant entre 0 et 17 ans.
2. Le nom de toutes les personnes ayant entre 1 et 18 ans.
3. Le nom et l'âge de toutes les personnes ayant entre 0 et 17 ans.
4. Le nom et l'âge de toutes les personnes ayant entre 1 et 18 ans.
5. Aucune des réponses précédentes.

Réponse : *e*

Explication :

Cette question retourne le nom de toutes les personnes ayant entre 0 et 17 ans ainsi que leur âge incrémenté par 1 (l'âge qu'elles auront l'année prochaine).

Deuxième partie

Cette partie vise à vous préparer davantage à la réalisation du travail pratique ainsi qu'aux exercices pratiques de l'examen sous surveillance. Les exercices de cette partie portent tous sur un même cas et doivent être résolus dans l'ordre. Nous vous encourageons à utiliser le logiciel MySQL pour valider vos réponses et pour vous familiariser davantage avec la syntaxe de SQL. N'hésitez pas à aller au-delà des situations présentées en imaginant et en testant d'autres contraintes ou requêtes.

Exercice 1

Une agence de voyages propose des forfaits touristiques à différents endroits et pour différentes catégories de voyage. Pour chaque endro l'agence procure les informations suivantes : le nom du lieu, le nom du pays dans lequel se trouve ce lieu, et le niveau de confort disponible dans ce lieu. Une catégorie de voyage est définie en fonction du type de voyage (d'aventure, culturel, de loisir, etc.), de l'organisation du groupe (groupe, famille, individuel, sur mesure) et de l'âge minimal requis pour participer au voyage de cette catégorie. Enfin, un forfait est un type de voyage pour un lieu donné pendant une période particulière de l'année.

Donnez les commandes de SQL permettant de créer la base de données de l'agence de voyages avec les informations précédentes (sans inclure des contraintes).

Réponse :

```
CREATE SCHEMA AGENCE;
USE AGENCE;

CREATE TABLE LIEU (
IDLIEU INT,
NOM CHAR(30),
PAYS CHAR(20),
FACILITES INT);

CREATE TABLE CATEGORIE (
ID INT,
NOM CHAR (30),
TYPE CHAR(20),
ORGANISATION INT,
AGE INT);

CREATE TABLE FORFAIT (
IDLIEU INT,
CATEGORIE INT,
DATEDEBUT DEC(6),
DATEFIN DEC(6));
```

Explication

Le modèle de la base comporte deux entités principales : LIEU et CATEGORIE, ainsi qu'une entité qui établit la relation entre elles, FORFAIT. En effet, un forfait est la combinaison d'un lieu et d'une catégorie de voyage. Le schéma se définit donc par ces trois tables. Pour la table LIEU, il y a le nom du lieu ainsi qu'un code numérique l'identifiant de façon unique et, éventuellement, le pays dans lequel se trouve ce lieu. L dernier attribut, FACILITES, permet de représenter, par un code numérique, le niveau de confort du lieu (par exemple 0 pour un endroit très sauvage et 10 pour un lieu avec toutes les facilités). Les attributs TYPE et ORGANISATION de la table CATEGORIE peuvent, eux aussi, être représentés par un entier ou par une chaîne de caractères (par exemple l'organisation peut être un entier entre 1 et 4 ou un texte pouvant comporter les mots : « groupe », « famille », « individuel » ou « sur mesure »). Les attributs DATEDEBUT et DATEFIN sont décrits par 6 décimales, mais pourraient aussi être définis par le type DATE.

Exercice 2

Déclarez une vue sur la table LIEU pour ne voir que les lieux touristiques au Canada.

Réponse :

```
CREATE VIEW NATIONAL (NOM,FACILITES)
AS SELECT NOM,FACILITES
FROM LIEU
WHERE PAYS="CANADA";
```

Explication : p. 282.

Exercice 3

Écrivez une requête en SQL pour présenter les forfaits disponibles pour un pays donné, par exemple la France. La requête doit retourner l'identification de la catégorie, le nom du lieu de vacances et les dates de début et de fin du forfait.

Réponse :

```
SELECT F.CATEGORIE, L.NOM, F.DATEDEBUT, F.DATEFIN
FROM FORFAIT F, LIEU L
WHERE F.IDLIEU=L.IDLIEU AND L.PAYS="FRANCE";
```

Explication : La requête fait intervenir deux tables : FORFAIT et LIEU. Cette deuxième est nécessaire pour trouver le nom du lieu.

Exercice 4

Modifiez la requête précédente pour inclure dans la réponse, au lieu de l'identifiant de la catégorie du forfait, le nom donné à cette catégorie son type et son organisation.

Réponse :

```
SELECT C.NOM, C.TYPE, C.ORGANISATION, L.NOM, F.DATEDEBUT, F.DATEFIN
FROM FORFAIT F, LIEU L, CATEGORIE C
WHERE F.IDLIEU=L.IDLIEU
AND L.PAYS="FRANCE"
AND C.ID=F.CATEGORIE;
```

Explication

Cette nouvelle requête fait intervenir les trois tables, car les nouvelles informations à afficher se trouvent dans la table CATEGORIE.

Exercice 5

Faites la même requête que celle demandée dans les exercices 3 et 4, mais ne présentez que les informations concernant la catégorie, soit son nom, son type et son organisation.

Réponse :

```
SELECT C.NOM, C.TYPE, C.ORGANISATION
FROM FORFAIT F, LIEU L, CATEGORIE C
WHERE F.IDLIEU=L.IDLIEU
AND L.PAYS="FRANCE"
AND C.ID=F.CATEGORIE;
```

Explication

Bien que pour cette nouvelle requête les informations demandées ne concernent que la table CATEGORIE, la requête fait intervenir les trois tables.

Exercice 6

Faites la même requête que celle demandée dans l'exercice 5, mais en utilisant des sous-requêtes. Est-ce que cette requête produit le mêm nombre de réponses que la requête précédente? Pourquoi? Laquelle devrait être modifiée et comment?

Réponse :

La requête avec des sous-requêtes pourrait avoir la forme suivante (d'autres options sont possibles) :

```
SELECT C.NOM, C.TYPE, C.ORGANISATION
FROM CATEGORIE C
WHERE C.ID IN
(SELECT CATEGORIE
FROM FORFAIT F
WHERE F.IDLIEU IN
(SELECT IDLIEU
FROM LIEU
WHERE PAYS="FRANCE"));
```

La solution présentée dans l'exercice précédent n'élimine pas les réponses dupliquées, alors que celle-ci le fait. Pour modifier la requête précédente, il suffit d'ajouter le mot DISTINCT après le premier mot SELECT.

Explication

La requête avec requête imbriquée parcourt tous les n-uplets de la table CATEGORIE et garde le nom, le type et l'organisation que pour la requête dont l'identifiant (unique) satisfait certaines conditions. Puisque dans la table CATEGORIE il n'y a pas d'éléments dupliqués, la réponse ne contient pas non plus de dupliqués. La requête de l'exercice précédent, par contre, fait une jointure de trois tables sur trois attributs. Puisque les valeurs de certains de ces attributs ne sont pas uniques dans toutes les tables (par exemple la catégorie dans la table FORFAIT), la réponse peut donner des éléments dupliqués.

Exercice 7

Écrivez une requête en SQL pour présenter tous les lieux qui ont des facilités de niveau 7 ou plus pour lesquels l'agence propose des forfait

---

Réponse :

```
SELECT NOM
FROM LIEU
WHERE (FACILITES >=7) AND
IDLIEU IN (SELECT DISTINCT IDLIEU FROM FORFAIT);
```

Ou encore

```
SELECT DISTINCT NOM
FROM LIEU, FORFAIT
WHERE (LIEU.FACILITES>=7) AND (LIEU.IDLIEU = FORFAIT.IDLIEU);
```

Explication

Dans le premier cas, la requête imbriquée retourne la liste des identifiants de lieu pour tous les lieux pour lesquels l'agence offre des forfaits. La question globale retourne le nom des lieux ayant des facilités de 7 ou plus et qui figure dans la liste résultante de la question imbriquée. Dans le deuxième cas, la question effectue la jointure de LIEU et FORFAIT sur l'attribut IDLIEU conservant les lieux avec des facilités de 7 ou plus.

2. Répondez aux questions et analysez attentivement vos réponses en vous demandant : « Quels sont les arguments qui font que cette réponse pourrait être la bonne? » et « Quels sont les arguments qui font que cette réponse pourrait être une mauvaise réponse? »
3. Vérifiez votre réponse en cliquant sur le bouton Solution. Pour chaque question, la réponse et une explication sont fournies.
4. Terminez le travail avec l'activité d'autoévaluation suivante.

Autoévaluation

Pour vous préparez un travail noté, faites l'activité qui suit.

Soit le modèle relationnel suivant :

Etudiant (numetu_, nom, prenom, datenaiss, rue, cp, ville)
Matiere (codemat_, libelle, coeff)
Epreuve (numepreuve_, dateepreuve, lieu, codemat#)
Notation (numetu#, numepreuve#, note)

(Le symbole # indique une clef étrangère alors que le symbole _ indique une clef primaire.)

La valeur de coeff est un nombre de 0 à 1 indiquant
le poids de la matière correspondante dans le calcul
de la note globale.

Répondez aux questions suivantes avec la requête SQL correspondante.

Question 1 : Contenu de la table etudiant

Réponse :

```
SELECT * FROM etudiant
```

Question 2 : Contenu de la table etudiant, classé par ordre alphabétique inverse des noms d'étudiants (Indice : on peut utiliser la syntaxe ORDER BY (https://www.w3schools.com/sql/sql_orderby.asp))

Réponse :

```
SELECT * FROM etudiant ORDER BY nom DESC
```

Question 3 : Libellé et coefficient (exprimé en pourcentage) de chaque matière

---

Réponse :

```
SELECT libelle, coef*100 FROM matiere
```

Question 4 : Nom et prénom de chaque étudiant

Réponse :

```
SELECT nom, prenom FROM etudiant
```

Question 5 : Nom et prénom des étudiants domiciliés à Lyon

Réponse :

SELECT nom, prenom FROM etudiant WHERE ville='Lyon'
Question 6 : Liste des notes supérieures ou égales à 10

Réponse :

SELECT note FROM notation WHERE note>=10

Question 7 : Liste des épreuves dont la date se situe entre le 1er janvier et le 30 juin 2004 (on peut utiliser la syntaxe BETWEEN AND (https://dev.mysql.com/doc/refman/8.4/en/comparison-operators.html#operator_between) et représenter le 1er janvier sous la forme 2004-01 01)

Réponse :

SELECT * FROM epreuve WHERE datepreuve BETWEEN '2004-01-01' AND '2004-06-30'

Question 8 : Nom, prénom et ville des étudiants dont la ville contient la chaîne "ll" (Indice : on peut utiliser l'instruction LIKE (https://dev.mysql.com/doc/refman/8.4/en/string-comparison-functions.html#operator_like))

Réponse :

SELECT nom, prenom, ville FROM etudiant WHERE ville LIKE '%ll%'

Question 9 : Prénoms des étudiants de nom Dupont, Durand ou Martin (Indice : on peut utiliser l'instruction IN (https://dev.mysql.com/doc/refman/8.4/en/comparison-operators.html#function_in))

Réponse :

SELECT prenom FROM etudiant WHERE nom IN ('Dupont', 'Durand', 'Martin')

Question 10 : Somme des coefficients de toutes les matières (indice : la somme est calculée avec la fonction SUM)

Réponse :

SELECT SUM(coef) FROM matiere

Question 11 : Nombre total d'épreuves (indice : on dénombre les enregistrements avec la fonction COUNT)

Réponse :

SELECT COUNT(*) FROM epreuve

Question 12 : Nombre de notes indéterminées (NULL) (indice : on vérifie qu'une valeur est nulle est l'expression IS NULL (https://dev.mysql.com/doc/refman/8.4/en/comparison-operators.html#operator_is-null))

Réponse :

SELECT COUNT(*) FROM notation WHERE note IS NULL

Question 13 : Liste des épreuves (numéro, date et lieu) incluant le libellé de la matière (Indice : il faut faire une jointure)

Réponse :

SELECT numepreuve, datepreuve, lieu, libelle FROM epreuve, matiere WHERE epreuve.codemat=matiere.codemat

Question 14 : Liste des notes en précisant pour chacune le nom et le prénom de l'étudiant qui l'a obtenue

---

Réponse :

SELECT nom, prenom, note FROM etudiant, notation WHERE etudiant.numetu=notation.numetu

Question 15 : Liste des notes en précisant pour chacune le nom et le prénom de l'étudiant qui l'a obtenue et le libellé de la matière concernée (Indice : il faut faire trois jointures)

Réponse :

SELECT nom, prenom, note, libelle FROM etudiant, notation, epreuve, matiere WHERE etudiant.numetu=notation.numetu AND notation.numepreuve=epreuve.numepreuve AND epreuve.codemat=matiere.codemat

Question 16 : Nom et prénom des étudiants qui ont obtenu au moins une note égale à 20

Réponse :

SELECT DISTINCT nom, prenom FROM etudiant, notation WHERE etudiant.numetu=notation.numetu AND note=20

Question 17 : Moyennes des notes de chaque étudiant (indiquer le nom et le prénom) (Indice : On peut utiliser la fonction AVG pour calculer la moyenne)

Réponse :

SELECT nom, prenom, AVG(note) FROM etudiant, notation WHERE etudiant.numetu=notation.numetu GROUP BY nom, prenom

Question 18 : Moyennes des notes de chaque étudiant (indiquer le nom et le prénom), classées de la meilleure à la moins bonne (on peut utiliser la syntaxe AS (https://www.w3schools.com/sql/sql_alias.asp))

Réponse :

SELECT nom, prenom, AVG(note) AS moyenne FROM etudiant, notation WHERE etudiant.numetu=notation.numetu GROUP BY nom, prenom ORDER BY moyenne DESC

Question 19 : Moyennes des notes pour les matières (indiquer le libellé) comportant plus d'une épreuve (Indice : il faut utiliser un GROUP BY (https://www.w3schools.com/sql/sql_groupby.asp) ... HAVING (https://www.w3schools.com/sql/sql_having.asp) ainsi que COUNT (https://www.w3schools.com/sql/func_sqlserver_count.asp) et DISTINCT (https://www.w3schools.com/sql/sql_distinct.asp))

Réponse :

SELECT libelle, AVG(note) FROM matiere AS m, epreuve AS e, notation AS n WHERE m.codemat=e.codemat AND e.numepreuve=n.numepreuve GROUP BY libelle HAVING COUNT(DISTINCT e.numepreuve)>1

Question 20 : Moyennes des notes obtenues aux épreuves (indiquer le numéro d'épreuve) où moins de 6 étudiants ont été notés

---

Réponse :

SELECT numepreuve, AVG(note) FROM notation WHERE note IS NOT NULL GROUP BY numepreuve HAVING COUNT(*)<6

Question 21 : Étant donné les tuples suivants :

nom	salaire
jean	4
pierre	5
jean	2

Quelle est la différence entre

SELECT nom, sum(salaire) FROM table WHERE salaire >3 GROUP BY nom;

et

SELECT nom, sum(salaire) FROM table GROUP BY nom HAVING SUM(salaire) >3;

?

Réponse :

Prenons l'instruction

SELECT nom, sum(salaire) FROM table WHERE salaire >3 GROUP BY nom;

Le moteur de base de données fait d'abord une sélection en se basant sur l'instruction WHERE avant de traiter l'instruction GROUP BY. Ainsi, il va d'abord la restriction (salaire >3) et obtiendra

jean	4
pierre	5

Il va ensuite faire le group by et obtiendra

jean	4
pierre	5

Prenons l'instruction

SELECT nom, sum(salaire) FROM table GROUP BY nom HAVING salaire >3;

Dans ce cas, on fait d'abord l'aggrégation avant de considérer l'instruction HAVING :

jean	6
pierre	5

Dans ce cas, l'instruction HAVING n'a aucun effet et le résultat final sera

jean	6
pierre	5

Question 22 : Étant donné les tuples suivants :

nom	salaire
jean	4
pierre	5
jacques	2

Trouvez le nom de tous les individus ayant un salaire supérieur à la moyenne.

Réponse :

On peut calculer la moyenne facilement :

SELECT avg(salaire) FROM table;

(On obtient 3.66666666666667.) On peut ensuite faire la requête suivante :

SELECT nom FROM table WHERE salaire>3.66666666667;

Ou mieux encore, on peut utiliser une sous-requête (http://download.nust.na/pub6/mysql/doc/refman/5.0/fr/subqueries.html) :

SELECT nom FROM table WHERE salaire>(SELECT AVG(salaire) FROM table);

Question 23 : Étant donné les tuples suivants :

nom	salaire
jean	4
pierre	5
jacques	2

Trouvez le nom de tous les individus ayant un salaire maximal.

Réponse :

On peut le faire ainsi avec une sous-requête (http://download.nust.na/pub6/mysql/doc/refman/5.0/fr/subqueries.html) :

SELECT nom FROM table WHERE salaire=(SELECT MAX(salaire) FROM table) ;

Question 24 : (Question avancée) En supposant que l'instruction « except » retourne la différence entre deux tables en SQL (ce qui est le cas avec SQLite, Oracle préférant Minus), donnez une instruction SQL qui retourne « vide » si la table est vide (COUNT(*)=0) et « non vide » autrement. Le nom de la table est « test ».

Réponse :

SELECT "is empty" EXCEPT SELECT "is empty" FROM test UNION SELECT "is not empty" FROM test;

Note : Je n'ai pas réussi à trouver une solution aussi simple pour MySQL car MySQL n'a pas l'équivalent de EXCEPT. Il faudrait sans doute faire une jointure pour y arriver avec MySQL ce qui manque sérieusement d'élégance.

Ce questionnaire est basé en partie sur un tutoriel mis au point par Jérôme Darmont (https://eric.univ-lyon2.fr/jdarmont/tutoriel-sql/) de Lyon 2

Autres exercices

On trouve sans mal d'autres activités d'apprentissage sur le Web. Par exemple, SQL Zoo (https://sqlzoo.net/wiki/SQL_Tutorial) est un site populaire pour approfondir sa compréhension du SQL.