Examen théorique

Système de Gestion de Bases de données – Module 1

Question 1

Pour assurer la cohérence d'une base de données, on évite la redondance d'information : chaque donnée est stockée une seule fois dans la base.
Il arrive cependant que l'on accepte certaines redondances.

a. Expliquez dans quel but.
b. Que fait-on pour éviter que ces redondances provoquent des incohérences dans la base ?
c. Comment qualifie-t-on cette redondance ?
d. Donnez un exemple (pas besoin du code, expliquez le principe)

Question 2

Dans les bases de données relationnelles, on parle de ''catalogue''

a. Donnez 2 synonymes
b. Par qui est-il mis à jour ?
c. Où est-il stocké ?
d. Que contient-il ?

Question 3

Les différents niveaux d'une base de données :

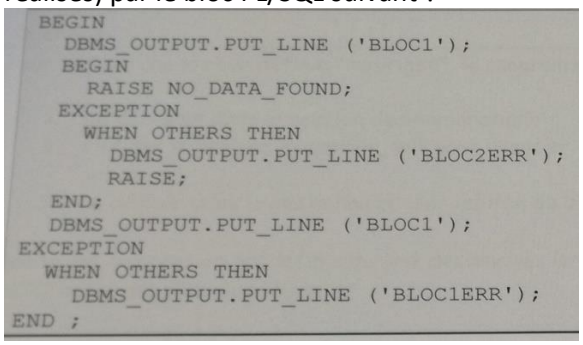
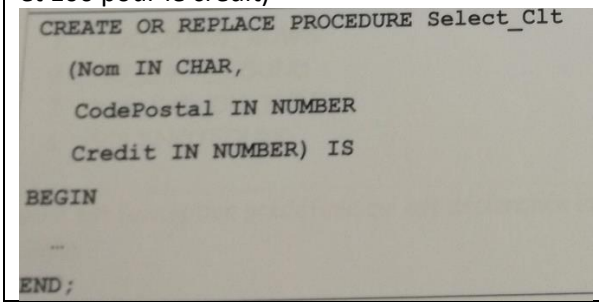
a. Citez les différents niveaux dites ce qu'ils contiennent
b. Citez les différents niveaux d'indépendances et expliquez à quoi ils correspondent

Question 4

VRAI FAUX

Question	V	F
Une exception définie ''non rédigée'' doit être déclarée par le programmeur dans la section des déclarations d'un bloc PL/SQL		
Une exception définie rédigée'' doit faire l'objet d'une instruction ''FRAGMEA EXCEPTION_INIT'' par le programmeur dans la section des déclarations d'un bloc PL/SQL		
En PL/SQL dans une clause WHEN de la section EXCEPTION, il est possible de ''relancer'' une nouvelle exception		
Une exception prédéfinie doit être lancée explicitement		
Une exception définie ''non rédigée'' n'est pas déclenchée automatiquement par Oracle		
Toutes les erreurs détectées par Oracle sont des exceptions prédéfinies		
En PL/SQL il est possible d'écrire des procédures polymorphes		
En PL/SQL, par défaut, le passage de paramètre s'effectue par adresse		
Une collection est un ensemble, éventuellement ordonné, d'éléments de types différents		
Une collection de type, ''tableau associatif'' peut posséder des indices non-consécutifs		

Peut-on dire que le PL/SQL est un langage orienté objet ? oui, non, pourquoi ?

Questions	Réponses
Quel est le résultat d'exécution (les affichages réalisés) par le bloc PL/SQL suivant ? 	
Quelle est la clause qui permet de modifier un tuple tout en retournant des données de ce tuple ?	
Quelle est la directive de compilation qui permet de passer des paramètres par référence en PL/SQL	
Écrivez la commande permettant de déclarer l'association du code d'erreur -2291 à l'exception ExcIntRef	
Réécrivez le code suivant en donnant des valeurs par défaut à chacun des paramètres ('Dupont' pour le nom, 4000 pour le code postal et 100 pour le crédit) 	

```
BEGIN
  DBMS_OUTPUT.PUT_LINE ('BLOC1');
  BEGIN
    RAISE NO_DATA_FOUND;
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE ('BLOC2ERR');
      RAISE;
  END;
  DBMS_OUTPUT.PUT_LINE ('BLOC1');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('BLOC1ERR');
END ;
```

```
CREATE OR REPLACE PROCEDURE Select_Clt
  (Nom IN CHAR,
   CodePostal IN NUMBER
   Credit IN NUMBER) IS
BEGIN
  ...
END;
```

Citez la bonne réponse, en plus des choix proposés, quatre possibilités de réponses supplémantaires sont possibles :

6 = Toutes les réponses sont correctes
7 = Aucune réponse proposé n'est correcte
8 = Il manque des informations pour pouvoir répondre à la question
9 = La question comporte une absurdité

Question	Choix
Pour traiter une erreur définie ''rédigée'', le programmeur doit 1. La déclarer dans la section des déclarations 2. Utiliser le pragma exception_init pour associer l'exception à un code d'erreur Oracle 3. La définir dans le gestionnaire d'exception du bloc	
Quelle est l'exception prédéfinie qui est déclenchée lorsqu'une instruction SELECT INTO retourne plus d'une ligne 1. TOO_MANY_ROWS 2. NO_DATA_FOUNDS 3. DUP_VAL_ON_INDEX 4. SQL%NOTFOUND	
Quelle clause spécifiée dans un gestionnaire d'exceptions permet d'intercepter n'importe quelle exception ? 1. WHEN OTHERS 2. WHEN NO_SPECIFIED 3. WHEN NULL 4. LA DERNIERE CLAUSE SPECIFIEE DANS LE BLOC D'EXCPTIONS	
Quelle est l'exception prédéfinie qui est déclenchée lorsqu'une instruction SELECT…BULK COLLECT…INTO… ne retourne pas de ligne ? 1. TOO_MANY_ROWS 2. NO_DATA_FOUND 3. DUP_VAL_ON_INDEX 4. SQL%NOTFOUND	
Quelle est l'exception prédéfinie qui est déclenchée lorsqu'un curseur ne retounre pas de ligne ? 1. TOO_MANY_ROWS 2. NO_DATA_FOUND 3. DUP_VAL_ON_INDEX 4. SQL%NOTFOUND	

Soit le déclencheur suivant défini sur la table Clients :

```
SELECT matricule, nom, prenom FROM clients;
1234 Thiry              Christiane
1235 Herbiet            Laurence
1236 Delmal             Pierre

CREATE TRIGGER BEF_UPD_NOM_TRIG
BEFORE update OF nom ON clients
FOR EACH ROW
BEGIN
   dbms_output.put_line('BEFORE');
END;
```

Quel sera le résultat de la requête suivante :
Update clients set nom = 'X' ;

Choix :

1. BEFORE
2. BEFORE BEFORE
3. BEFORE BEFORE BEFORE
4. GENERERA UNE ERREUR

Soit le déclencheur suivant défini sur la table Clients :

```
SELECT matricule, nom, prenom FROM clients;
1234 Thiry              Christiane
1235 Herbiet            Laurence
1236 Delmal             Pierre

CREATE TRIGGER BEF_UPD_NOM_TRIG
BEFORE update of prenom on clients FOR EACH ROW
BEGIN
      dbms_output.put_line('BEFORE');
END;
```

Quel sera le résultat de la requête suivante :
Update clients set nom = 'X' ;

Choix :

1. BEFORE
2. BEFORE BEFORE
3. BEFORE BEFORE BEFORE
4. GENERERA UNE ERREUR

Quel est le résultat d'éxcecution (les affichages réalisés) par le bloc PL/SQL suivant ?

```
DECLARE
  Nom1 CHAR(30) DEFAULT 'SGBD';
  Nom2 VARCHAR2(20) DEFAULT 'SGBD';
  Nom3 VARCHAR2(20);
BEGIN
  IF Nom1 = Nom2
    THEN DBMS_OUTPUT.PUT_LINE ('Nom1 = Nom2');
    ELSE DBMS_OUTPUT.PUT_LINE ('Nom1 <> Nom2');
  END IF;
  IF Nom2 <> Nom3
    THEN DBMS_OUTPUT.PUT_LINE ('Nom2 <> Nom3');
    ELSE DBMS_OUTPUT.PUT_LINE ('Nom2 = Nom3');
  END IF;
EXCEPTION
  WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('ERREUR');
END ;
```

1. Nom1 = Nom2
   Nom2 <> Nom3
2. Nom1 <> Nom2
   Nom2 <> Nom 3
3. Nom 1 <> Nom2
   Nom 2 = Nom 3
4. Nom 1 = Nom2
   Nom2 = Nom3

Soient les tables suivantes :

```
CREATE TABLE Departements (
 Deptno NUMBER(2) CONSTRAINT CPDeptartements PRIMARY KEY,
 Name VARCHAR2(30));

CREATE TABLE Employes (
 EmpNo NUMBER(4) CONSTRAINT CPEmployes PRIMARY KEY,
 Name VARCHAR2(10),
 Deptno NUMBER(2)
  CONSTRAINT RefEmployesDepartementsDepno
    REFERENCES Departements(Deptno));
```

Soit le déclencheur suivant :

```
CREATE OR REPLACE TRIGGER DepartementsUpd
BEFORE UPDATE OF Deptno ON departements
FOR EACH ROW
BEGIN
  UPDATE Employes SET Deptno = :NEW.Deptno
  WHERE  Deptno = :OLD.Deptno;
END;
```

Soit le déclencheur suivant :

```
CREATE OR REPLACE TRIGGER DepartementsUpd
BEFORE UPDATE OF Deptno ON departements
FOR EACH ROW
BEGIN
  UPDATE Employes SET Deptno = :NEW.Deptno
  WHERE  Deptno = :OLD.Deptno;
END;
```

Le trigger DepartementUdp :

1. S'exécutera lors de la mise à jour du Deptno dans la table employes
2. Est invalide, il est impossible de modifier une clé étrangère dans un déclancheur
3. Est invalide car il n'y a pas de COMMIT après la commande UPDATE
4. Est invalide car la variable :OLD ne peut être référencée dans un UPDATE

Soit le code suivant, quel sera le résultat de son exécution ?

```
DECLARE
   TYPE list_t IS TABLE OF VARCHAR2(100)
      INDEX BY PLS_INTEGER;
   l_list list_t;
BEGIN
   IF l_list (100) = 'Steven' THEN
      DBMS_OUTPUT.PUT_LINE ('Hurray!');
   END IF;
END;
```

1. ‘’Hurray !’’ sera affiché
2. Rien ne sera affiché, le contrôle sera simplement rendu à l’environnement d’appel
3. Une exception NO_DATA_FOUND sera lancée
4. Une exception VALUE_ERROR sera lancée

Analysez le code du déclencheur ci-dessous. Expliquez son rôle. Est-il possible de le rendre plus efficace en améliorant le code ? Si oui, faites les modifications nécessaires.

```
CREATE TRIGGER upd_salaire_personnel
BEFORE UPDATE OF salaire ON personnel
FOR EACH ROW
DECLARE
  salaire_diminue EXCEPTION;
BEGIN
  IF (:OLD.salaire > :NEW.salaire)
  THEN
    RAISE salaire_diminue;
  END IF;
EXCEPTION
  WHEN salaire_diminue THEN
    raise_application_error (-20001,
          'Le salaire ne peut diminuer')
  WHEN OTHERS THEN RAISE;
END;
```

Analysez le code ci-dessous. Expliquez ce qu'il réalise ;
Modifier le code de manière à obtenir le même résultat sans utiliser le BULK INTO.
Est-il possible de le rendre plus efficace en améliorant le code ?

```
DECLARE
  TYPE t_employe IS TABLE OF Employes%ROWTYPE;
  CURSOR CurEmp IS
    SELECT *
    FROM Employes
    WHERE job LIKE 'A%';
  TousLesEmployes t_employe;
BEGIN
  OPEN CurEmp;
  FETCH CurEmp BULK COLLECT INTO TousLesEmployes;
  FOR i in 1..TousLesEmployes.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE (TousLesEmployes(i).nom);
  END LOOP;
  CLOSE CurEmp;
EXCEPTION
  WHEN INVALID_CURSOR
    THEN DBMS_OUTPUT.PUT_LINE ('Erreur Curseur CurEmp');
  WHEN OTHERS
    THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;
```

1. Analysez le code ci-dessous. Expliquez ce qu'il réalise (soyez précis !).
2. Donnez la portion de MRD correspondant à ce code.
3. Quel est le type de contrainte utilisé à la ligne 6 ; a la ligne 17 ? Dans quel cas est-il nécessaire d'utiliser le type de contrainte de la ligne 17 ?
4. Aurait-il été possible de donner l'ensemble de ces instructions dans un ordre différent ? Expliquez.
5. Quel est l'intérêt de donner un nom à chacune des contraintes ? Expliquez.

```
 1  DROP TABLE Coureur CASCADE CONSTRAINTS;
 2  DROP TABLE CLub CASCADE CONSTRAINTS;
 3
 4  CREATE TABLE Coureur
 5  (IdCoureur      VARCHAR2(6)
 6     CONSTRAINT CPCoureur PRIMARY KEY,
 7   Nom            VARCHAR2(30)
 8     CONSTRAINT CoureurNomNotNull NOT NULL
 9     CONSTRAINT CoureurNomUnique UNIQUE,
10   DateNaissance DATE
11     CONSTRAINT DateNaisanceNotNull NOT NULL,
12   Sexe           CHAR(1)
13     CONSTRAINT SexeNotNull NOT NULL,
14     CONSTRAINT CoureurSexe CHECK (Sexe IN ('F', 'M'))
15   IdClub         VARCHAR2(4)
16     CONSTRAINT IdClub NOT NULL,
17   CONSTRAINT CoureurIdCoureur
18     CHECK (IdCoureur = Sexe || UPPER(SUBSTR (Nom, 1, 3)) ||
19           TO_CHAR (DateNaissance, 'RR')));
20
21 CREATE TABLE Club
22 (IdClub         VARCHAR2(4)
23    CONSTRAINT CpClub PRIMARY KEY,
24  Nom            VARCHAR2(50)
25    CONSTRAINT ClubNomNotNull NOT NULL,
26  Responsable    VARCHAR2(6)
27    CONSTRAINT REFClubIdCoureurINCoureur
28    REFERENCES Coureur (IdCoureur));
29
30 ALTER TABLE Coureur
31    ADD CONSTRAINT REFCoureurIdClubINClub
32    FOREIGN KEY (IdClub) REFERENCES Club (IdClub);
```

Comment le SGBD exécute-t-il

1. Des requêtes (simplement) imbriquées
2. Des requêtes corrélées ?

Question 12

Vrai/Faux

Question	V	F
En Oracle, une lecture bloque une modification et une modification bloque une lecture		
En Oracle, une écriture bloque une écriture, il est impossible de provoquer une perte de mise à jour		
En Oracle le niveau d'isolation par défaut est REPEATABLE READ		
Le niveau d'isolation REPEATABLE READ peut produire des lectures impropres lors de l'exécution de transactions concurrentes		
Par défaut, en Oracle, il est possible d'obtenir des lectures non reproductibles et des lectures impropres lors de l'exécution de transactions concurrentes		