Correction exercices les procédures stockées MySQL : Série 01

1. ## Objectifs
   - Connaitre les procédures stockées de base de données MySQL
2. ## Exercice 01
   - **Énoncé**
     - Vous pouvez visualiser **l'énoncé de l'exercice**
   - **Solution**
     - **Q_01: Procedure**

```
CREATE OR REPLACE PROCEDURE update_salaire(idemp IN Employe.nuempl%type,
sal IN number)
IS
BEGIN
    UPDATE EMPLOYE SET salaire=salaire+sal WHERE nuempl=idemp;
END;
```

     - **Q_02: Compter le nombre d'employés participant à un projet donné :**

```
CREATE FUNCTION count_employes_proj (nuproj integer) RETURNS integer AS
$$
    SELECT COUNT(*) FROM Travail WHERE nuproj = $1;
$$ LANGUAGE SQL;
```

     - **Q_03: Compter le nombre de projets supervisés par les employés d'un service donné :**

```
CREATE FUNCTION count_projets_supervises (nuserv integer) RETURNS
```

apcpedagogie.com

```
integer AS $$
    SELECT COUNT(*) FROM Projet WHERE resp IN (
        SELECT nuempl FROM Employe WHERE #affect = $1
    );
$$ LANGUAGE SQL;
```

- **Q_04: Compter le nombre de projets auxquels participe l'employé donné :**

```
CREATE FUNCTION count_projets_employe (nuempl integer) RETURNS integer
AS $$
    SELECT COUNT(*) FROM Travail WHERE nuempl = $1;
$$ LANGUAGE SQL;
```

- **Q_05: Retourner "Salaire faible" si le salaire de l'employé donné est supérieur à 2000$ sinon retourner "Bon salaire" :**

```
CREATE FUNCTION check_salaire (nuempl integer) RETURNS varchar(20) AS $$
    SELECT CASE WHEN salaire > 2000 THEN 'Salaire faible' ELSE 'Bon
salaire' END FROM Employe WHERE nuempl = $1;
$$ LANGUAGE SQL;
```

- **Q_06: Compter le nombre d'employés qui prennent en charge plus que le nombre de projets donné :**

```
CREATE FUNCTION count_employes_plus_proj (n integer) RETURNS integer AS
$$
    SELECT COUNT(*) FROM (
        SELECT COUNT(*) as projets, nuempl FROM Travail GROUP BY nuempl
    ) as count_projets WHERE projets > $1;
$$ LANGUAGE SQL;
```

- **Q_07: Insérer l'employé donné dans une table de sauvegarde nommée 'ALERT_EMPLOYE' :**

```
CREATE PROCEDURE insert_alert_employe (nuempl varchar(20), nomempl
varchar(20), hebdo integer, salaire integer, #affect integer) AS $$
    INSERT INTO ALERT_EMPLOYE (nuempl, nomempl, hebdo, salaire, #affect)
VALUES ($1, $2, $3, $4, $5);
```

apcpedagogie.com

```
$$ LANGUAGE SQL;
```

3. Exercice 02

- **Énoncé**
  - Vous pouvez visualiser **l'énoncé de l'exercice**
- **Solution**
  - **Q_02: Écrire une fonction qui calcule, pour un adhérent donné, le nombre de jours restant avant d'être en retard.**

```
CREATE FUNCTION jours_restants_avant_retard(numab VARCHAR(3)) RETURNS
INT
BEGIN
  DECLARE nb_emprunts INT;
  DECLARE nb_retards INT;
  DECLARE min_jours_restants INT;
  DECLARE max_retard INT;
  DECLARE min_date_retour DATE;
  DECLARE min_duree INT;
  DECLARE i INT DEFAULT 0;

  -- compter le nombre d'emprunts en cours de l'adhérent
  SELECT COUNT(*) INTO nb_emprunts FROM pret WHERE numab=numab AND
dateretour IS NULL;

  -- si l'adhérent n'a pas d'emprunts en cours, on renvoie NULL
  IF nb_emprunts = 0 THEN
    RETURN NULL;
  ELSE
    SET max_retard = 0;
    SET min_jours_restants = NULL;

    -- parcourir tous les emprunts en cours de l'adhérent
    WHILE i < nb_emprunts DO
      SELECT MIN(date_add(datepret, INTERVAL duree DAY)), MIN(duree)
      INTO min_date_retour, min_duree
      FROM pret, livre
```

apcpedagogie.com

```
      WHERE pret.numinv = livre.numinv AND numab = numab AND dateretour
IS NULL;

      IF min_date_retour < CURDATE() THEN
        SET nb_retards = DATEDIFF(CURDATE(), min_date_retour);
        IF nb_retards > max_retard THEN
          SET max_retard = nb_retards;
        END IF;
      ELSE
        SET nb_retards = DATEDIFF(min_date_retour, CURDATE());
        IF min_jours_restants IS NULL OR nb_retards < min_jours_restants
THEN
          SET min_jours_restants = nb_retards;
        END IF;
      END IF;

      SET i = i + 1;
    END WHILE;

    IF max_retard > 0 THEN
      RETURN -max_retard;
    ELSE
      RETURN min_jours_restants;
    END IF;
  END IF;
END;
```

- **Q_03: Utiliser cette fonction pour afficher la situation de tous les adhérents.**

```
SELECT numab, nom, prenom, jours_restants_avant_retard(numab) AS
jours_restants FROM abonne;
```

- **Q_04: Écrire une procédure qui permette de lister les emprunts d'un adhérent identifié par son numéro.**

```
CREATE PROCEDURE liste_emprunts(numab VARCHAR(3))
BEGIN
  SELECT livre.numinv, titre, datepret, duree
  FROM pret, livre
```

apcpedagogie.com

```
  WHERE pret.numinv = livre.numinv AND numab = numab AND dateretour IS
NULL;
END;
```

- **Q_05: Écrire une procédure qui affiche les exemplaires disponibles d'un titre (on fera une version OUTER JOIN et une version NOT IN). Pour se faciliter la tâche, on a intérêt à d'abord traiter la question : "combien y a-t-il exemplaires disponibles du titre 'NARCISSE ET GOLDMUND'" avec les deux versions demandées, pour ensuite passer à l'écriture de la procédure stockée.**
  Version OUTER JOIN :

```
CREATE PROCEDURE ExemplairesDispoOuterJoin (IN titreRecherche
VARCHAR(20))
BEGIN
    SELECT exemplaire.numinv, livre.titre, livre.auteur
    FROM livre
    LEFT OUTER JOIN exemplaire
    ON livre.numinv = exemplaire.numinv
    WHERE livre.titre = titreRecherche
    AND exemplaire.numinv IS NULL;
END
```

  Version NOT IN :

```
CREATE PROCEDURE ExemplairesDispoNotIn (IN titreRecherche VARCHAR(20))
BEGIN
    SELECT numinv, titre, auteur
    FROM livre
    WHERE titre = titreRecherche
    AND numinv NOT IN
    (SELECT numinv FROM pret WHERE dateret IS NULL);
END
```

- **Q_06: Procédure pour afficher les titres d'un auteur et le nombre d'exemplaires disponibles par titre :**

```
CREATE PROCEDURE ExemplairesDispoAuteur (IN nomAuteur VARCHAR(15))
```

apcpedagogie.com

```
BEGIN
    DECLARE auteurRecherche VARCHAR(15);
    DECLARE curTitres CURSOR FOR SELECT DISTINCT titre FROM livre WHERE
auteur = nomAuteur;
    DECLARE curTitresCount CURSOR FOR SELECT COUNT(*) FROM exemplaire
WHERE numinv IN (SELECT numinv FROM livre WHERE auteur = auteurRecherche
AND titre = titreRecherche AND numinv NOT IN (SELECT numinv FROM pret
WHERE dateret IS NULL));
    DECLARE titreRecherche VARCHAR(20);
    DECLARE nbExemplaires INT(2);

    SET auteurRecherche = nomAuteur;

    OPEN curTitres;
    titreLoop: LOOP
        FETCH curTitres INTO titreRecherche;
        IF (titreRecherche IS NULL) THEN
            LEAVE titreLoop;
        END IF;
        OPEN curTitresCount;
        FETCH curTitresCount INTO nbExemplaires;
        CLOSE curTitresCount;
        SELECT CONCAT(titreRecherche, ' (', nbExemplaires, ' exemplaires
disponibles)') AS result;
    END LOOP;
    CLOSE curTitres;
END
```

- **Q_07: Procédure pour enregistrer un emprunt :**

```
CREATE PROCEDURE EnregistrerEmprunt (IN numAbonne VARCHAR(3), IN numInv
VARCHAR(4), IN datePret DATE)
BEGIN
    INSERT INTO pret (numab, numinv, datepret) VALUES (numAbonne,
numInv, datePret);
    UPDATE exemplaire SET qte = qte - 1 WHERE numinv = numInv;
END
```

- **Q_07: Procédure pour afficher les titres d'un auteur et le nombre d'exemplaires disponibles par titre :**

apcpedagogie.com

```
SELECT numab, nom, prenom, jours_restants_avant_retard(numab) AS
jours_restants FROM abonne;
```

- **Q_08: Modifier la table des emprunts :**

```
ALTER TABLE pret ALTER COLUMN dureemax SET DEFAULT 14;
```

Cours MySQL

Source : https://apcpedagogie.com/correction-exercices-les-procedures-stockees-mysql-serie-01 · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com