TP6 - Création de procédures stockées

B2 - Base de données - BTS SIO2

Préambule: *L'objectif de ce TP est de découvrir et de mettre en place des procédures stockées afin de mettre à jour une base de données donnée.*

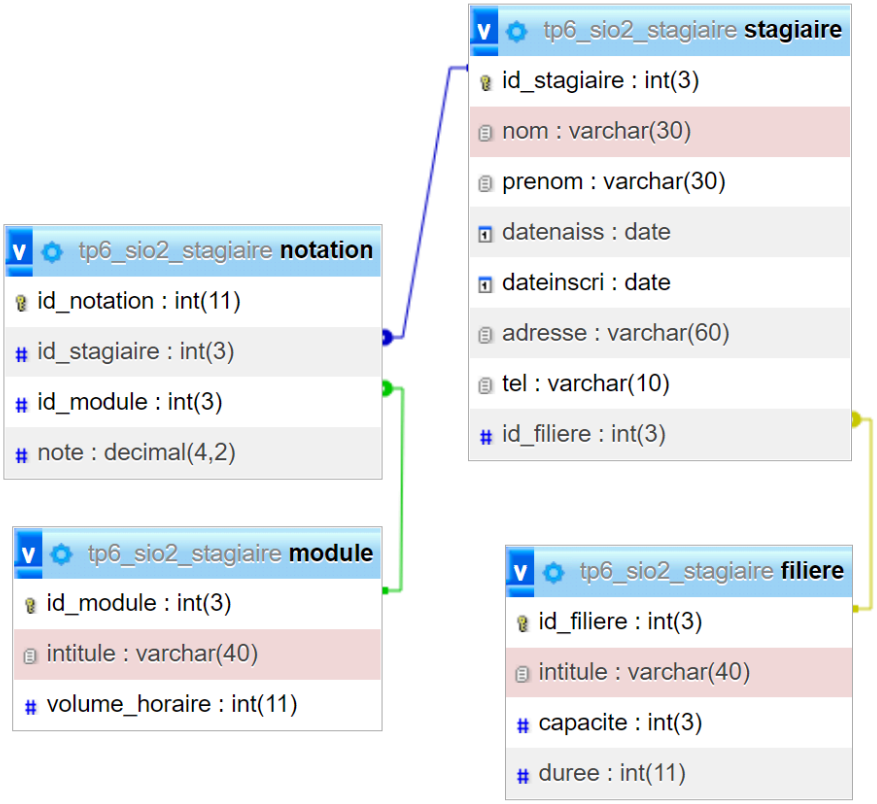
1) Vision globale

- [ ] **Récupérer des données sur un site**
- [ ] **Créer la base de données**
- [ ] **Créer les procédures stockées**

2) Présentation du contexte

a) La base de données

Cette nouvelle base représente la gestion des stagiaires au sein d'un établissement scolaire.



b) Le jeu de données

Cette

3) Travail à faire

1) La procédure permettant de lister les stagiaires d'une filière donnée

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `01_recherche_filiere`(IN
`filiere_choisie` VARCHAR(50))
BEGIN
SELECT *
FROM etudiant
INNER JOIN filiere ON etudiant.id_filiere = filiere.id_filiere
WHERE filiere.intitule = filiere_choisie;
END$$
DELIMITER ;
```

2) La procédure permettant d'afficher les stagiaires ayant l'âge dans la tranche précisé par l'utilisateur

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `02_age_etudiant` (IN `age` INT(2))
BEGIN
SELECT nom, prenom, DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(),datenaiss)),'%Y') + 0 AS
age_etudiant
FROM etudiant
WHERE DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(),datenaiss)), '%Y') + 0 = age;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `02_age_etudiant_BIS` (IN `age_choisie`
INT(2))   BEGIN
SELECT nom, prenom, DATEDIFF(NOW(), datenaiss) / 365 AS 'Age'
FROM etudiant
WHERE DATEDIFF(NOW(), datenaiss) / 365 LIKE CONCAT(age_choisie, ".%");
END$$
```

3) Augmenter d'un point les notes des stagiaires dans le module 4

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `03_update_note` ()   BEGIN
UPDATE notation
SET note = note + 1
WHERE (SELECT id_module
FROM module
WHERE id_module = 4);
END$$
```

4) La liste des stagiaires dont le nom commence par une lettre spécifiée par l'utilisateur

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `04_liste_etu_commence_par` (IN `lettre`
CHAR(1))   BEGIN
SELECT nom, prenom
FROM etudiant
WHERE nom LIKE CONCAT(lettre, '%');
END$$
```

5) Le bulletin de notes d'un stagiaire donné

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `05_bulletin` (IN `name` CHAR(20))   BEGIN
SELECT etudiant.nom, etudiant.prenom, note, id_module
FROM notation
INNER JOIN etudiant ON notation.id_etudiant = etudiant.id_etudiant
WHERE etudiant.nom = name;
END$$
```

6) Liste des stagiaires inscrits entre deux dates

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `06_liste_etu_dateinscription` (IN
`date_debut` DATE, IN `date_fin` DATE)   BEGIN
SELECT nom, prenom, dateinscri
FROM etudiant
WHERE dateinscri BETWEEN date_debut AND date_fin;
END$$
```

INSERT INTO `etudiant` (`id_etudiant`, `nom`, `prenom`, `datenaiss`, `dateinscri`, `adresse`, `tel`, `id_filiere`) VALUES (NULL, 'Bernard', 'Claude', '2002-08-23', '2022-09-01', '10 rue Jean Jaures', '0625698574', '1');

7) Avant de supprimer un stagiaire, vérifier s'il existe et vérifier s'il a des notes

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `07_check_supp_etudiant_ID` (IN `id` INT)
BEGIN
SELECT *
FROM etudiant
WHERE id_etudiant = id;
IF FOUND_ROWS() > 0 THEN
DELETE FROM etudiant WHERE id_etudiant = id;
ELSE
SELECT "L'étudiant n'existe pas, impossible de le supprimer." AS message;
END IF;
END$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `07_check_supp_etudiant_NOM` (IN
`etudiant_nom` VARCHAR(30))   BEGIN
IF EXISTS (SELECT nom FROM etudiant WHERE nom = etudiant_nom)
THEN
IF EXISTS (SELECT id_etudiant FROM notation WHERE id_etudiant = (SELECT
id_etudiant FROM etudiant WHERE nom = etudiant_nom))
THEN
DELETE FROM notation WHERE id_etudiant = (SELECT id_etudiant FROM
etudiant WHERE nom = etudiant_nom);
END IF;
DELETE FROM etudiant WHERE nom = etudiant_nom;
END IF;
END$$
```

8) Procédure qui supprime une filière avec l'ensemble des stagiaires affectés à cette filière

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `08_delete_fil_ID` (IN `numeroFil`
INT(2))    BEGIN
DELETE
FROM etudiant
WHERE etudiant.id_filiere = numeroFil;
DELETE
FROM filiere
WHERE id_filiere = numeroFil;
END$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `08_delete_fil_NOM` (IN `filiere_select`
VARCHAR(40))    BEGIN
DELETE FROM notation WHERE id_etudiant = (SELECT id_etudiant FROM etudiant WHERE
id_filiere = (SELECT id_filiere FROM filiere WHERE intitule = filiere_select));
DELETE FROM etudiant WHERE id_filiere = (SELECT id_filiere FROM filiere WHERE intitule
= filiere_select);
DELETE FROM filiere WHERE id_filiere = (SELECT id_filiere FROM filiere WHERE intitule =
filiere_select);
END$$
```

9) Affecter une note pour un stagiaire : vérifier l'existance du stagiaire et du module. Vérifier si le stagiaire est déjà noté pour ce module

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `09_affecter_note` (IN `nomEtudiant`
VARCHAR(20), IN `idModule` INT(2), IN `note` FLOAT)    BEGIN
DECLARE nbModules INT DEFAULT 0;
DECLARE nbEtudiants INT DEFAULT 0;
SELECT COUNT(*) INTO nbEtudiants
FROM etudiant
WHERE nom = nomEtudiant;
SELECT COUNT(*) INTO nbModules
FROM module
WHERE id_module = idModule;
IF nbEtudiants > 0 AND nbModules > 0 THEN
INSERT INTO notation (id_etudiant, id_module, note)
VALUES ((SELECT id_etudiant FROM etudiant WHERE nom = nomEtudiant),
idModule, note);
ELSE
SELECT "Etudiant ou module inexistant" AS message;
END IF;
END$$
```

10) Supprimer les 3 premiers stagiaires

Pour ce faire, 3 étapes:

- créer une vue “top_notes” récupérant les 3 étudiants ayant la meilleure moyenne
- modifier la structure de la table étudiant - vue relationnelle - on delete -> cascade
- supprimer les étudiants de la table étudiants à partir des ID de la vue “top_notes”

```
CREATE VIEW `top_notes` AS
SELECT `notation`.`id_etudiant` AS `id_etudiant`, avg(`notation`.`note`) AS `moyenne`
FROM `notation` GROUP BY `notation`.`id_etudiant`
ORDER BY avg(`notation`.`note`) DESC LIMIT 0, 33  ;


CREATE DEFINER=`root`@`localhost` PROCEDURE `10_delete_3prem_etudiant_VUE` ()   BEGIN
DELETE FROM etudiant
WHERE id_etudiant IN (
SELECT id_etudiant
FROM top_notes);
END$$
DELIMITER ;



VERSION BIS 10


DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `supp_3premier`()

DELETE etudiant

FROM etudiant

JOIN (

  SELECT id_etudiant

  FROM (

      SELECT etudiant.id_etudiant

      FROM etudiant

      JOIN notation ON etudiant.id_etudiant = notation.id_etudiant

      GROUP BY etudiant.id_etudiant

      ORDER BY AVG(notation.note) DESC

      LIMIT 3

  ) AS requete

) AS supprimer ON etudiant.id_etudiant = supprimer.id_etudiant$$

DELIMITER ;
```

11) Supprimer les stagiaires inscrits l'année dernière et stocker les dans la table archive

```
if exists (select * from stagiaire where DATEDIFF(YEAR,dateinscrip,GETDATE())=1)
begin
select * into archive from stagiaire where DATEDIFF(YEAR,dateinscrip,GETDATE())=1
delete from stagiaire where DATEDIFF(YEAR,dateinscrip,GETDATE())=1
end
else
begin
print 'aucun stagiaire'
end
```

12) Afficher les informations des stagiaires qui ont plus de deux notes

```
select * from stagiaire
where n_stagiaire in (select n_stagiaire from notation group by n_stagiaire having
COUNT (n_stagiaire)>=2)
```