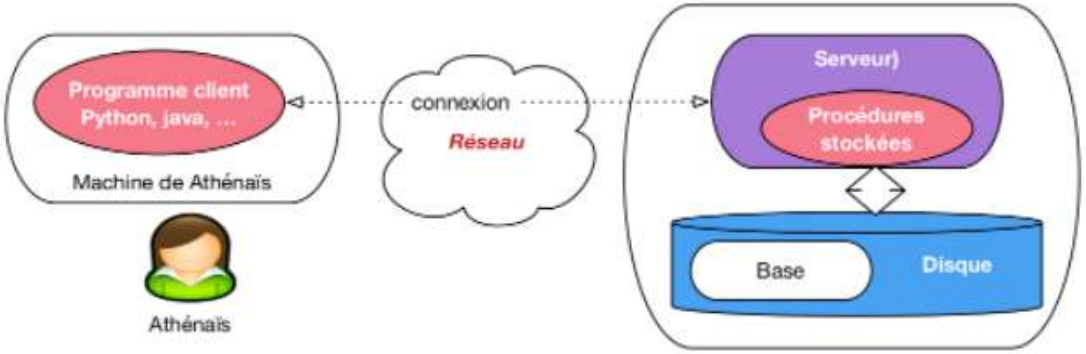
PL/SQL: Procédures et Fonctions Stockées (MySQL)

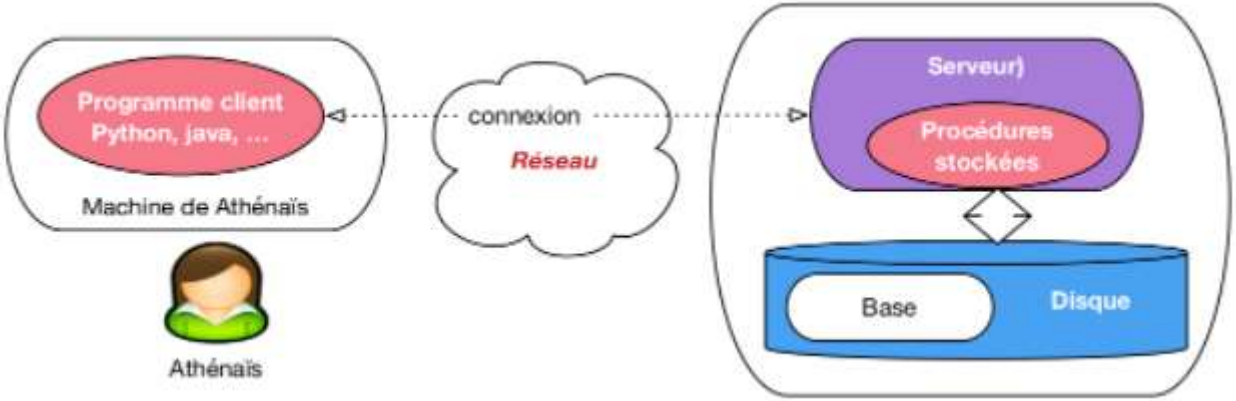
SupNum 2023

- Est-ce que SQL est un langage de programmation complet ?
- Les variables
- Les structures conditionnelles
- Les boucles



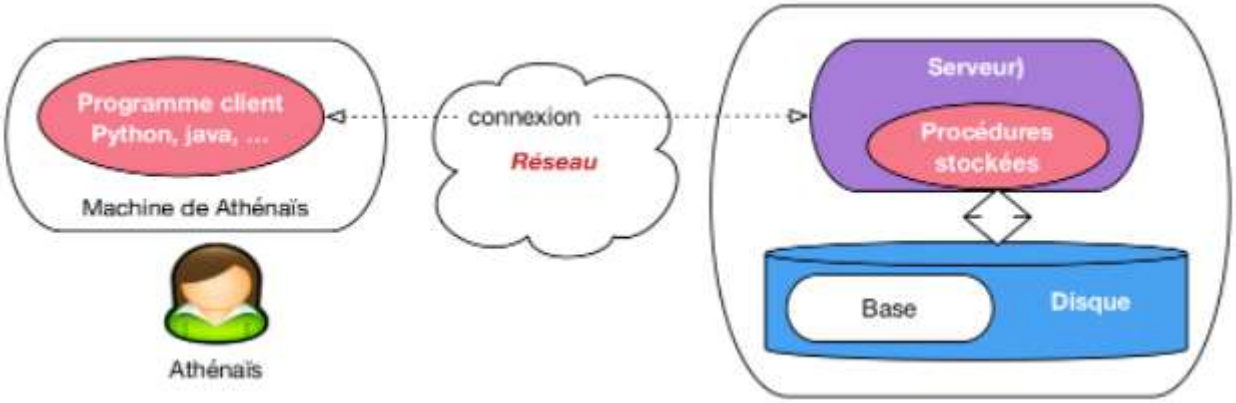
Motivation

- ✓ Une procédure stockée s'exécute au sein du SGBD, ce qui évite les échanges réseaux qui sont nécessaires quand les mêmes fonctionnalités sont implantées dans un programme externe (Python, PHP, Java,…) communiquant en mode client/serveur avec la base de données.
- ✓ Le programme s'exécute alors en communiquant avec le serveur pour exécuter les requêtes et récupérer les résultats.
- ✓ Chaque demande d'exécution d'un ordre SQL implique une transmission sur le réseau, du programme vers le client, suivie d'une analyse de la requête par le serveur, de sa compilation et de son exécution



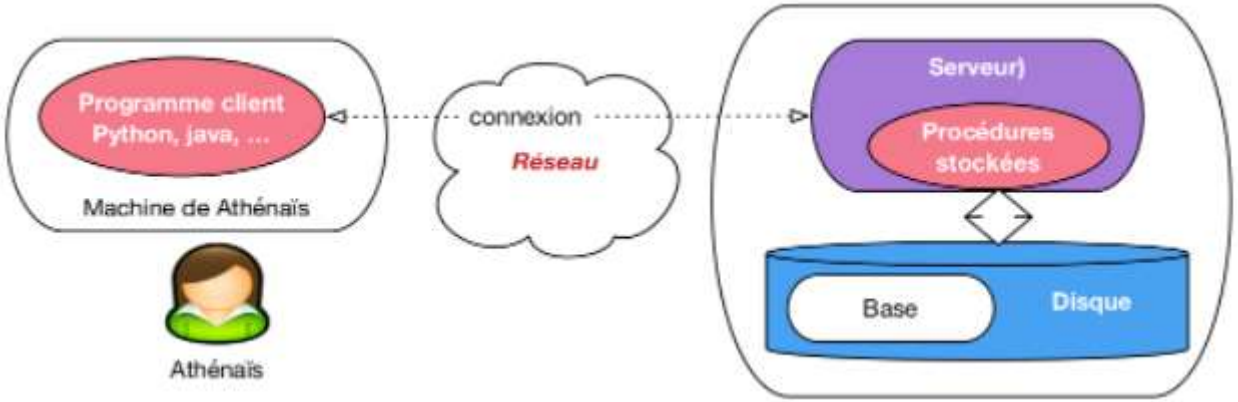
Motivation

- ✓ Ensuite, chaque fois que le programme client souhaite récupérer un n-uplet du résultat, il doit effectuer un appel externe, via le réseau.
- ✓ Tous ces échanges interviennent de manière non négligeable dans la performance de l'ensemble, et cet impact est d'autant plus élevé que les communications réseaux sont lentes et/ou que le nombre d'appels nécessaires à l'exécution du programme est important.



Solution

- ✓ Le recours à une procédure stockée permet de regrouper du côté serveur l'ensemble des requêtes SQL et le traitement des données récupérées.
- ✓ La procédure est compilée une fois par le SGBD, au moment de sa création, ce qui permet de l'exécuter rapidement au moment de l'appel.



Encapsulation, Performance, Efficacité, Réutilisabilité, Réduction du trafic réseau

PL/SQL: les procédures stockées

Procedural Language

Le PL/SQL permet de combiner des requêtes SQL, des boucles, des tests, ... pour définir des procédures, des fonctions,

```
DROP PROCEDURE IF EXISTS PROC ;
DELIMITER //
CREATE  PROCEDURE  PROC(p_1,..., p_n)
  BEGIN (obligatoire)
    DECLARE
     -- bloc d'instructions SQL
     -- instructions PL/SQL ou sous-blocs
  END // (obligatoire)
DELIMITER ; ← A ajouter obligatoirement dans l'exécution d'un script
```

Exemples *de bloc PL/SQL*

```
DELIMITER //
DROP PROCEDURE IF EXISTS getFromParis//
CREATE PROCEDURE getFromParis()
 BEGIN
     DECLARE  V_VILLE  VARCHAR(20) DEFAULT 'PARIS' ;
    select plnum, plnom from pilote where VILLE=V_VILLE;
 END//
```

Afficher les pilotes originaires de la ville de PARIS.

Afficher les pilotes originaires de la ville city en paramètre.

```
DELIMITER //
DROP PROCEDURE IF EXISTS getPiloteVille //
CREATE PROCEDURE getPiloteVille(city varchar(20))
 BEGIN
      select plnum, plnom from pilote where VILLE=city;
 END//
```

Exemple de bloc PL/SQL

```
-- afficher le numero et le nom des pilotes ayant comme salaire 26 000

DELIMITER //
create procedure getNumNom()
 BEGIN
   DECLARE PNOM VARCHAR(20);
   DECLARE PNUM INTEGER;

   select plnum, plnom INTO PNUM, PNOM from pilote where SALAIRE=26000;

   SELECT CONCAT(PNUM," ", PNOM);
 END//

 DELIMITER ;

CALL  getNumNom();
```

- - un script qui affiche le nombre d'avions localisés dans une ville donnée

```
CREATE PROCEDURE getNbVilles(LOC VARCHAR(20))
BEGIN
DECLARE nb INT;
 SELECT COUNT(*) INTO @nb FROM AVION WHERE LOCALISATION=LOC;
 SELECT @nb AS 'NB D''AVIONS';
END
```

Call getNbVilles(«NICE»);

@ indique que c'est une variable de session

Select @nb;

```
-- script de définition d'une procédure
-- procédure « insert_Pilote » : permet d'insérer un pilote avec :
        -- son numéro, son nom et son numéro de département
        -- les autres attributs doivent être non obligatoires !

drop procedure if exists insert_pilote;
delimiter //

create procedure insert_pilote (v_plnum integer, v_plnom varchar(14), v_salaire integer)


  begin
     /* on insert dans la BD,  c'est juste un exemple de code */
    insert into pilote(plnum, plnom, salaire) values (v_plnum, v_plnom, v_salaire);
  end //

delimiter ;

call insert_pilote(200, 'TOTO',40000);
```

```
- - script de définition d'une procédure pour insérer les pilotes dont leurs salaires<=19000
- - dans une nouvelle  table

drop procedure if exists insert_pilote2;
drop table pilotsalaire;   - - table resulat

create table pilotsalaire(num integer, nom varchar(20), sal integer);
delimiter //

create procedure insert_pilote2 ()
begin
     declare num integer;
     declare nom varchar(20);
     declare sal integer;

    SELECT plnum, plnom, salaire INTO num, nom, sal  FROM pilote WHERE salaire<=19000;

      insert into pilotsalaire values (num, nom, sal);
  end //
delimiter ;

call insert_pilote2();

select * from pilotsalaire;
```

```
- - cette fonction renvoie le double de l'entier en
-- paramètre

DELIMITER //

 create function doubleNB(nb INT) RETURNS integer
    BEGIN
      DECLARE  newNB  integer;
      SET newNB=nb*2;
      RETURN newNB;
    END //

   DELIMITER ;
```

SELECT PLNOM, SALAIRE, doubleNB(SALAIRE) **FROM** PILOTE

Résultat ?

SELECT PLNOM, SALAIRE **FROM** PILOTE

PLNUM	PLNOM	SALAIRE
1	MIRANDA	26000
2	LETHANH	21000
3	TALADOIRE	18000
4	CHARBONNIER	17000
5	REY	19000
6	CHARBONNIER	18000
7	PENAULD	17000
8	FOUILHOUX	15000
9	GANNAT	18000
10	GADAIX	20000

SELECT PLNOM, SALAIRE, doubleNB(SALAIRE) **FROM** PILOTE

PLNOM	SALAIRE	SALAIREDOUBLE
MIRANDA	26000	52000
LETHANH	21000	42000
TALADOIRE	18000	36000
CHARBONNIER	17000	34000
REY	19000	38000
CHARBONNIER	18000	36000
PENAULD	17000	34000
FOUILHOUX	15000	30000
GANNAT	18000	36000
GADAIX	20000	40000

```
-- calculer les moyennes des salaires des pilotes de codes pairs et de codes impairs

DELIMITER //
DROP PROCEDURE IF EXISTS MOYENNES//

CREATE PROCEDURE MOYENNES(OUT M1 DECIMAL(7,2), OUT M2 DECIMAL(7,2))
BEGIN
        BEGIN
          SELECT AVG(SALAIRE) INTO M1  FROM PILOTE WHERE PLNUM%2=0;
        END;

        BEGIN
          SELECT AVG(SALAIRE) INTO M2  FROM PILOTE WHERE PLNUM%2<>0;
        END;
END//
DELIMITER ;

CALL MOYENNES(@M1,@M2);

SELECT @M1 AS 'moyenne1', @M2 AS 'moyenne2';
```

La variable en sortie peut être récupérée comme variable globale de mysql.

```
CREATE PROCEDURE JOURS(d INT)
BEGIN
 IF(D=1) THEN
     SELECT "LUNDI";
  ELSEIF(D=2) THEN
      SELECT "MARDI";
    ELSEIF(D=3) THEN
      SELECT "MERCREDI";
      ELSEIF(D=4) THEN
          SELECT "JEUDI";
         ELSEIF(D=5) THEN
               SELECT "VENDREDI";
             ELSEIF(D=6) THEN
                SELECT "SAMEDI";
              ELSEIF(D=7) THEN
                SELECT "DIMANCHE";
               ELSE
                 SELECT "ERREUR, CE JOUR N'EXISTE PAS";
  END IF;
END
```

Une procédure qui reçoit un entier et affiche le jour correspondant.

VERSION IF

```
CREATE PROCEDURE `DAYS_Case`(IN `D` INT)
BEGIN
 CASE D
   WHEN 1 THEN
                SELECT "LUNDI";
   WHEN 2  THEN
                 SELECT "MARDI";
   WHEN 3 THEN
               SELECT "MERCREDI";
   WHEN 4 THEN
                 SELECT "JEUDI";
   WHEN 5 THEN
                SELECT "VENDREDI";
   WHEN 6 THEN
                  SELECT "SAMEDI";
   WHEN 7 THEN
                  SELECT "DIMANCHE";
   ELSE
                SELECT « ERREUR!";
  END CASE;
END//
```

Une procédure qui reçoit un entier et affiche le jour correspondant.

VERSION CASE

EXERCICE

L’objectif est de créer un script qui permet d’orienter

Les etudiants en fonctions des notes dans les matieres:

- DSI: avoir une note >=14 dans les matieres de code DEV
- RSS: avoir une note>=13 dans les matieres de code SYR
- CNM: avoir une note >=13 dans les autres matieres

RELEVE

matricule	code	note
1	DEV12	14.5
2	DEV11	15
3	SYR11	13
4	SYR12	15
5	INFO	13
6	PRESTA	17
7	SYR11	10
8	IMAGE	16

```
DROP PROCEDURE IF EXISTS DISPATCHER;
DROP TABLE IF EXISTS DSI;
DROP TABLE IF EXISTS RSS;
DROP TABLE IF EXISTS CNM;

DELIMITER //
CREATE PROCEDURE DISPATCHER()
BEGIN
 CREATE TABLE DSI AS SELECT * FROM RELEVE WHERE NOTE>=14 AND MATIERE LIKE 'dev%';
 CREATE TABLE RSS AS SELECT * FROM RELEVE WHERE NOTE>=13 AND MATIERE LIKE 'syr%';
 CREATE TABLE CNM AS SELECT * FROM RELEVE WHERE NOTE>=13 AND MATIERE NOT LIKE 'dev%' AND
                                                          MATIERE NOT LIKE 'syr%';
END//
DELIMITER ;
CALL DISPATCHER()
```

On peut passer toutes les commandes du DDL, du DML
et du DCL (grant, revoke, commit, rollback) à une procédure.

FIN