

Cours de bases de données, <http://sql.bdpedia.fr>

Procédures stockées

Pourquoi un langage de programmation ?

SQL **n'est pas** un langage de programmation

- SQL est un langage pensé pour permettre l'interrogation d'une base de manière **déclarative**, et non procédurale
- Pas de variable, pas de boucle, pas de test

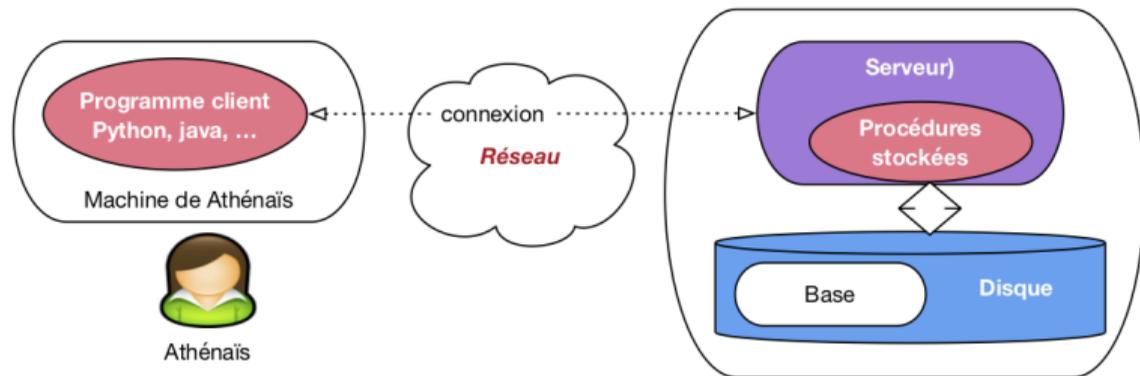
Pour écrire des applications, on utilise SQL associé à un langage de programmation

L'association :

- SQL pour les accès à la base
- Python, Java, PHP, C++, etc. pour tout le reste

Deux architectures possibles

Premier cas (à gauche) : les requêtes SQL sont intégrées à un **programme client** et transmises au serveur de données.



Second cas (à droite) : requêtes SQL intégrées au **serveur** (procédures stockées) moins d'échanges réseaux, plus de sécurité.

Premier exemple, mode interactif (PL/SQL)

```
DECLARE
  -- Quelques variables
  v_nbFilms    INTEGER;
  v_nbArtistes INTEGER;

BEGIN
  -- Compte le nombre de films
  SELECT COUNT(*) INTO v_nbFilms FROM Film;
  -- Compte le nombre d'artistes
  SELECT COUNT(*) INTO v_nbArtistes FROM Artiste;

  -- Affichage des résultats
  DBMS_OUTPUT.PUT_LINE ('Nombre de films: ' || v_nbFilms);
  DBMS_OUTPUT.PUT_LINE ('Nombre d''artistes: ' || v_nbArtistes);
END;
```

Notez : les variables, la clause `into`.

Procédure stockée

```
CREATE OR REPLACE PROCEDURE InsereGenre (p_genre VARCHAR) AS
  v_genre_majuscules VARCHAR(20);
  v_count INTEGER;
BEGIN
  -- On met le paramètre en majuscules
  v_genre_majuscules := UPPER(p_genre);

  -- On vérifie que le genre n'existe pas
  SELECT COUNT(*) INTO v_count
  FROM Genre WHERE code = v_genre_majuscules;

  -- Si on n'a rien trouvé: on insère
  IF (v_count = 0) THEN
    INSERT INTO Genre (code) VALUES (v_genre_majuscules);
  END IF;
END;
```

Notez : le test.

Second exemple, fonction et itérateur

```
CREATE OR REPLACE FUNCTION MesActeurs(v_idFilm INTEGER)
  RETURN VARCHAR IS
  resultat VARCHAR(255);
BEGIN
  -- Boucle prenant tous les acteurs du films
  FOR art IN
    (SELECT Artiste.* FROM Role, Artiste
     WHERE idFilm = v_idFilm AND idActeur=idArtiste)
  LOOP
    IF (resultat IS NOT NULL) THEN
      resultat := resultat || ', ' || art.prenom || ', ' || art.nom;
    ELSE
      resultat := art.prenom || ', ' || art.nom;
    END IF;
  END LOOP;
  return resultat;
END;
```

Notez : le type automatique de art, la boucle

Utilisation

- En interactif

```
SQL> start StatsFilms.sql
```

- Avec l'ordre execute, placé dans un autre langage (C, Java, PHP)

```
execute insereGenre('Policier')
```

- Dans une requête SQL

```
SELECT titre, MesActeurs(idFilm)  
FROM Film WHERE idFilm=5;
```

TITRE	MESACTEURS(IDFILM)
Volte/Face	John Travolta, Nicolas Cage

Dérivation de types depuis le schéma

Deux exemples pour illustrer.

- `Film.titre%TYPE` est le titre de l'attribut titre de la table Film;
- `Artiste%ROWTYPE` est un type RECORD correspondant aux attributs de la table Artiste.

Le même principe pour les requêtes SQL définies dans le cadre des curseurs (à suivre).

Beaucoup plus difficile avec un langage de programmation externe.

À retenir

Principales difficultés SQL/programmation : **typage** et **conversion**

Typage et conversion des nuplets

- Définir des variables du langage (Python, Java) correspondant aux types SQL
- Convertir le résultat d'une requête en variables du langage

Typage et conversion des tables

Ce sont des **ensembles** (ou séquences si **order by**), on doit les parcourir avec des **boucles**.