Principes d'utilisation des systèmes de gestion de bases de données

JDBC

L3 Informatique

Emmanuel Waller, LRI, Orsay
Université Paris-Sud

JDBC

- le mode programme
- JDBC : qu'est-ce que c'est ? Avantages ?
- devant les machines
- interface, portabilité, connexion
- exécution d'ordres BD : principe, sans paramètres

le mode programme : cas général

- qu'est-ce que c'est ?
- pour faire quoi ? pour qui ?
- comment ?
- Java et les problèmes du mode programme : JDBC

qu'est-ce que c'est ?

- lors écriture application :
  - Mêmes ordres SQL qu'en interactif
  - + programme autour
  - comme PL/SQL

pour faire quoi ? pour qui ?

- connu :
  - gérer problèmes BD : ok avec SQL
  - code :
    - dédié à la gestion BD
    - local à la base

    : ok avec PL/SQL
- but : accès BD par application généraliste
  - qui doit faire : calculs, conversationnel, interface, réseau, etc.
- pour : informaticiens (comme PL/SQL)

comment ?

- programme normal (Java, PHP, C, Cobol, etc.)
- + connexion
  - rappel :
    - Interactif : commande Unix
    - PL/SQL : déjà connecté
  - Transformation du processus prog. en client SGBD
- + n'importe quels ordres SQL
  - avec gestion des problèmes du mode programme

Les problèmes du mode programme

- Il faut lancer des ordres BD depuis un programme (Java, PHP, Ocaml, etc.)
- La gestion d'ordre BD dans un programme pose les problèmes suivants

- Interface : concrètement, comment placer « insert into t values (7, 'ok') » dans un programme ?
- Connexion : comment indiquer qu'on démarre une connexion BD (+login, base, etc.) ?

Java et les pbs MP : JDBC

- un programme Java souhaitant accéder à BD doit gérer ces problèmes
- le package JDBC fournit les outils nécessaires pour gérer les problèmes du mode programme

JDBC : qu'est-ce que c'est ? avantages ?

- API = ensemble de classes et interfaces (package) Java
- Java DataBase Connectivity
- permet d'accéder à des SGBD par SQL
- application JDBC = programme Java utilisant JDBC
- indépendant
  - d'un SGBD particulier (grâce drivers)
  - d'une architecture matérielle (grâce Java)

exemple

- situation :
  - c'est l'anniversaire de Jeanne
  - on veut exécuter :

    update personne

    set age = age + 1

    where nom = 'Jeanne'
- regardons intuitivement premier programme Java/JDBC complet qui tourne (voir ci-joint)
- puis nous reviendrons systématiquement sur tous les points

devant les machines

- compilation
  - standard (cause import package)
  - + fournir les classes du driver choisi
- concrètement :
  - indiquer au compilateur Java où se trouve le driver utilisé : CLASSPATH Unix
  - javac
  - java

démonstration

pbs MP : interface

- comment exécuter un ordre BD depuis Java ?
- Java appelle des fonctions d'une bibliothèque (JDBC)
- ces fonctions encapsulent les accès au serveur BD
- les ordres BD sont :
  - certaines fonctions de la bibliothèque
  - chaînes de caractères paramètres de fonctions de la bibliothèque

pbs MP : connexion

- Connection x = DriverManager.getConnection(url)
- fonction getConnection :
  - récupère driver indiqué par url (=> déjà chargé)
  - l'utilise pour établir connexion avec le serveur
  - renvoie objet correspondant (classe Connection) : nécessaire toutes interactions avec le serveur

- url : String
  - = jdbc:oracle:drivertype:user/password@database
  - drivertype : oci7, oci8, thin (applet : thin)
  - @database :
    - optionnel, sinon defaut
    - oci : ligne de tnsnames.ora, thin : host:port:sid, SQL*Net

l'objet de la classe Connection

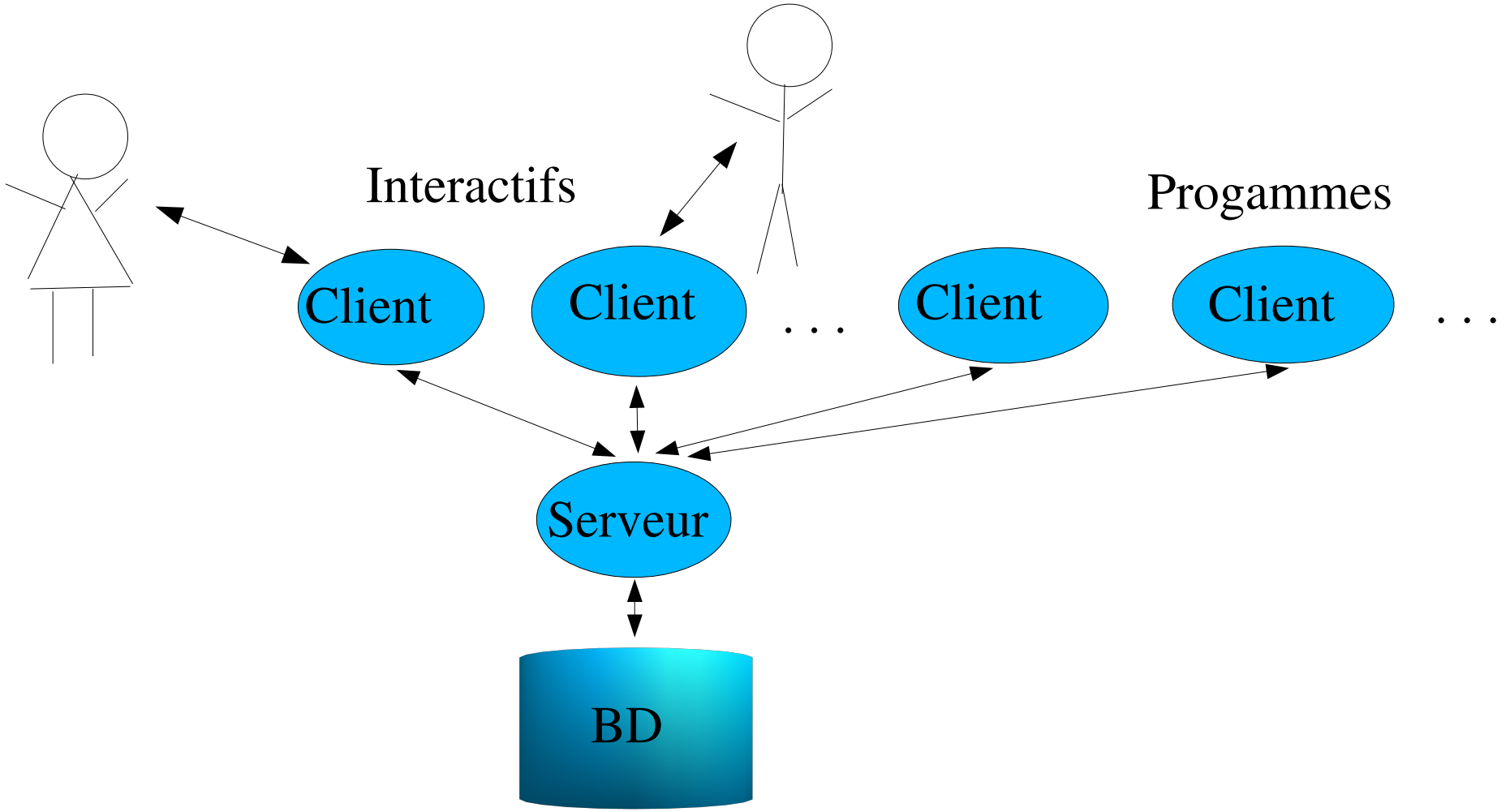
- « session » =
  1. connexion à un serveur donné
  2. séquence d'ordres SQL
  3. déconnexion
- possible : une application JDBC a plusieurs sessions avec un ou plusieurs serveurs
  - (chaque session est alors un client)

- un objet Connection
  - définit une session
  - utilisé via ses méthodes pour
    - envoi ordres BD au serveur
    - gestion transactions
  - fournit informations sur tables, procédures stockées, etc.
  - créable que par getConnection (Connection: interface)

déconnexion

- méthode de Connection :
  - void close() throws SQLException
  - effectue déconnexion normale du serveur qui termine la session
  - utile si on veut déconnexion immédiate au lieu attendre que automatique
- déconnexion implicite automatique lorsque objet Connection libéré par garbage collector

Vue d'ensemble et architecture



pbs MP : exécution d'ordres BD

- principe et déroulement
- ordres sans paramètre

Principe et déroulement

- création d'un « objet ordre » générique (ordre SQL non fixé)
- envoi de l'ordre au serveur par méthode de cet objet
- si select : récupération du résultat (curseur)
- gestion erreurs

- 3 catégories :
  - ordres BD sans ou avec paramètres
    - paramètre : valeur d'une colonne
  - appel procédure stockée PL/SQL

ordres SQL sans paramètre

- = tout ordre SQL copié-collable sous SQL*Plus
- 2 catégories :
  - select
  - tous les autres
- objet classe Statement
- créé par méthode de Connection :

  Statement createStatement() throws SQLException
- ne pas utiliser si ordre exécuté plusieurs fois avec paramètres différents, car moins efficace

- un seul objet Statement en général (plusieurs possible : hors module)
- envoyer l'ordre au serveur par la méthode de Statement (si ordre non requête) :

  int executeUpdate(String sql) throws SQLException

  - renvoie
    - nombre de lignes traitées pour insert, update, delete
    - 0 sinon
  - String sql : sans le point-virgule de SQL*Plus

démonstration