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

JDBC (3)

L3 Informatique

Emmanuel Waller, LRI, Orsay

JDBC

- le mode programme
- JDBC : qu'est-ce que c'est ? Avantages ?
- devant les machines
- interface, portabilité, connexion
- gestion erreurs BD (intuition), transactions
- exécution d'ordres BD : principe, avec/sans paramètres, curseurs, SQL dynamique
- JDBC et l'architecture en couches
- Ref cursor, gestion communication programme-serveur, métadonnées

pbs MP : portabilité

- driver = module logiciel de communication entre un logiciel et
  - du matériel (ex : imprimante)
  - un autre logiciel (ex : entre Java et Oracle)
- = implantation des interfaces de l'API JDBC
- chaque constructeur de SGBD fournit un driver à Java
- conséquence : un programme Java/JDBC qui utilise SQL standard tourne au-dessus de n'importe quel SGBD

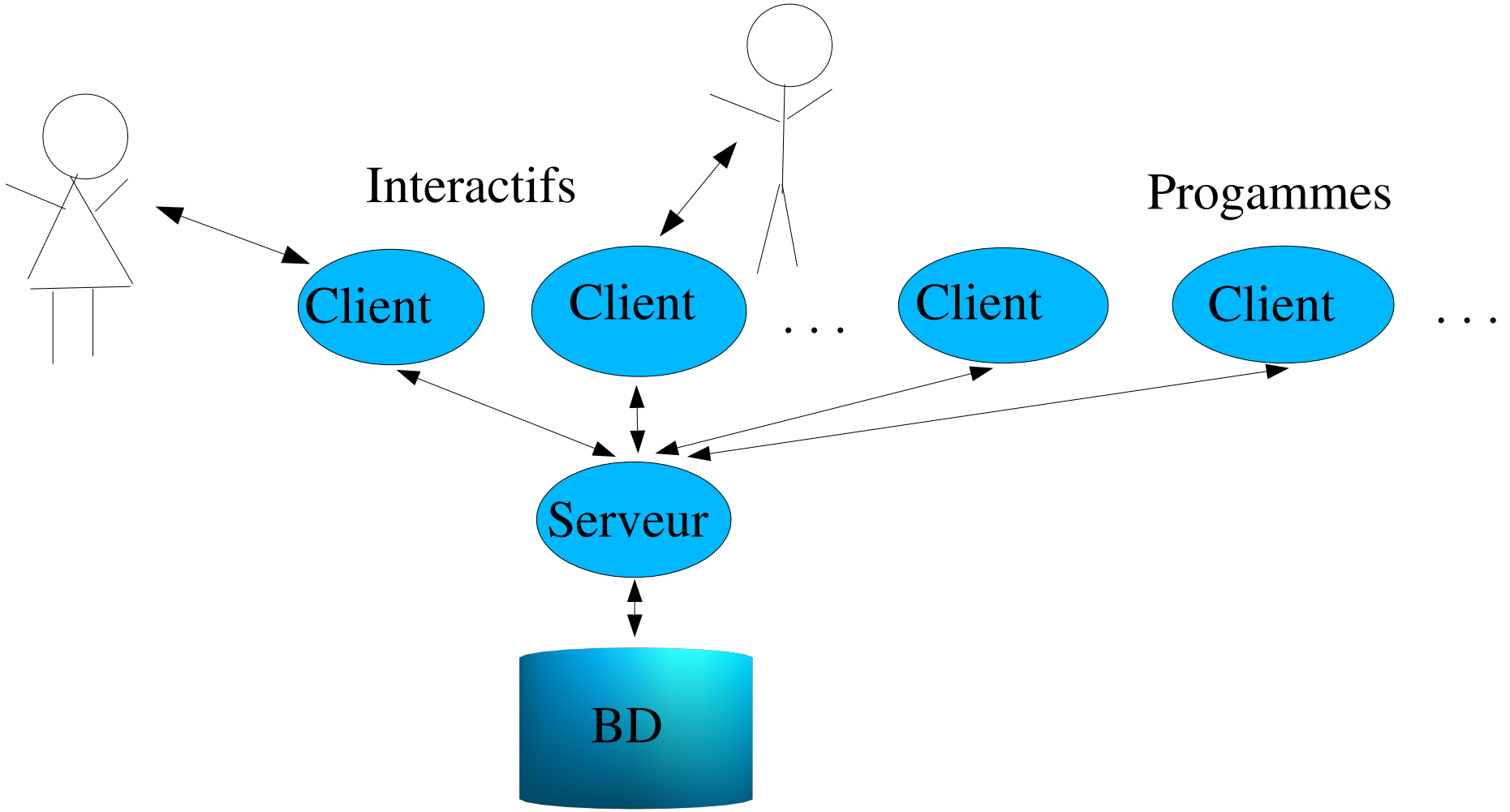
chargement driver(s)

- Class.forName(« oracle.jdbc.driver.OracleDriver ») : recherche cette classe Java
- possible plusieurs dans même programme : interagir avec différents serveurs de différents constructeurs (Oracle, MySQL, Sybase, Informix, etc.)
- (variante : DriverManager.registerDriver(new Oracle.jdbc.driver.OracleDriver()))
- (si applet : « dnlddriver » au lieu « driver »)

déroulement d'un programme

- importer classes JDBC (statique)
- lancement et début de l'exécution d'un programme JDBC : inconnu du serveur
- charger driver(s) voulus(s) par le programmeur (=choix du SGBD)
- connexion (JDBC) : devient client
- envoi ordres SQL au serveur (JDBC)
- déconnexion (JDBC) : termine en tant que client
- inconnu du serveur : continue son exécution, puis termine comme programme (voir sur croquis)

Vue d'ensemble et architecture



pbs MP : gestion des erreurs BD (intuition)

- rappel : un ordre BD envoyé au serveur peut générer une erreur
  - (ou une demande BD non SQL : mode confirmation automatique, méta-données, etc.)
- fonction JDBC ayant demandé cet ordre lève automatiquement une exception de la classe SQLException
  - => throws SQLException si pas gérée

utilisation

```
try {
    . . . appel JDBC . . .
} catch (SQLException e) {
    . . . e . . .
}
```

- Sinon (Java) :
  - retour au premier catch (Exception)
  - Sinon : arrêt brutal programme
    - BD : Annulation transaction