ESIR1 BD
Bases de données

programmation

Olivier Ridoux

Problème à résoudre

- Les BD ne connaissent pas les langages de programmation
- Les langages de programmation ne connaissent pas les BD
- Utiliser un connecteur BD-programme
  - **JDBC** pour Java

Plan

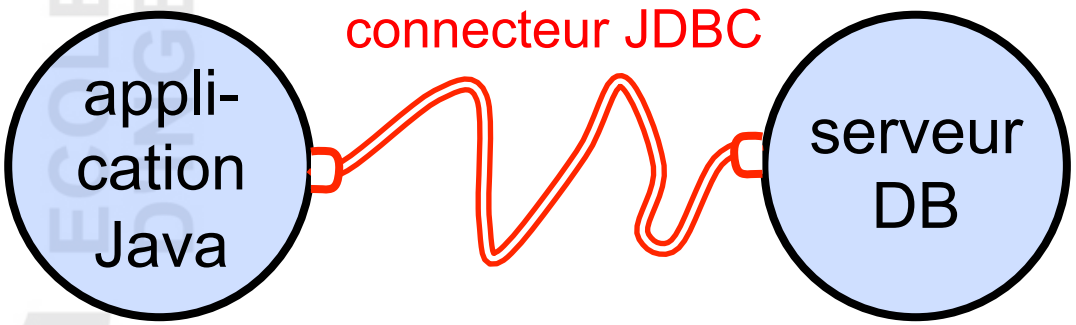
- Connexion avec une BD
- Requête
- Manipulation d'une relation
- Gestion des erreurs

Connexion avec une BD

Objectif

- Offrir une interface indépendante des fournisseurs de BD...

...mais fournie par ces mêmes fournisseurs



L'interface standard de la connexion

```
import java.sql.*;
```

- Interface abstraite…

…donne un accès standardisé à une BD SQL de fournisseur quelconque

La sélection du connecteur (1)

```
Class.forName(
        "com.mysql.jdbc.Driver");
```

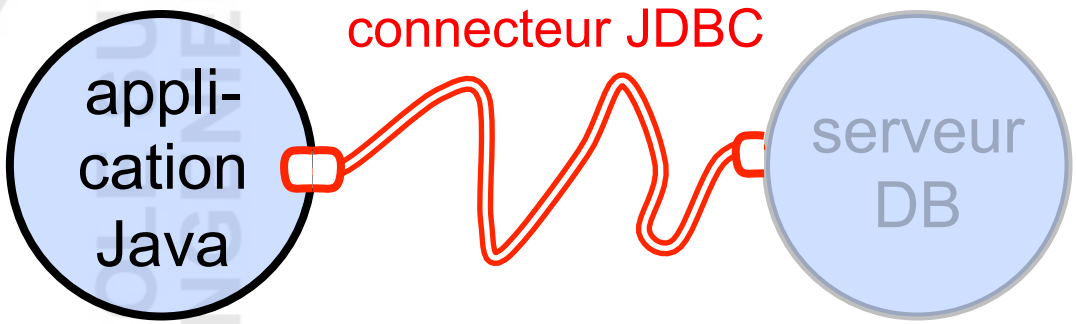
- Lie dynamiquement l'application avec le connecteur concrêt désiré…

…choix du fournisseur de BD

Le connecteur doit exister dans l'environnement de programmation

La sélection du connecteur (2)

```
Class.forName(
        "com.mysql.jdbc.Driver");
```



La connexion (1)

```
Connection conn = null ;
...
conn =
 DriverManager.getConnection(
      "anteros.ifsic.univ-rennes1.fr",
      USER, PASS ) ;
```

- Connexion à une BD…

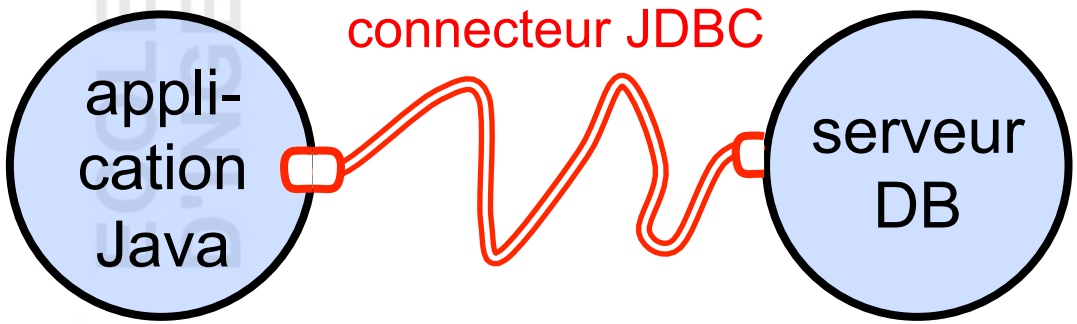
…choix d'un serveur particulier

Le serveur doit exister à l'adresse indiquée

Les `USER` et `PASS` doivent être corrects

La connexion (2)

```
Connection conn = null ;
...
conn =
 DriverManager.getConnection(
     "anteros.ifsic.univ-rennes1.fr",
     USER, PASS ) ;
```



La déconnexion

```
conn.close();
```

La connexion doit exister

Conclusion connexion

- À faire une fois
- Ne pas se tromper de
  - fournisseur
  - serveur
  - USER
  - PASS

Requête

Création d'une requête (1)

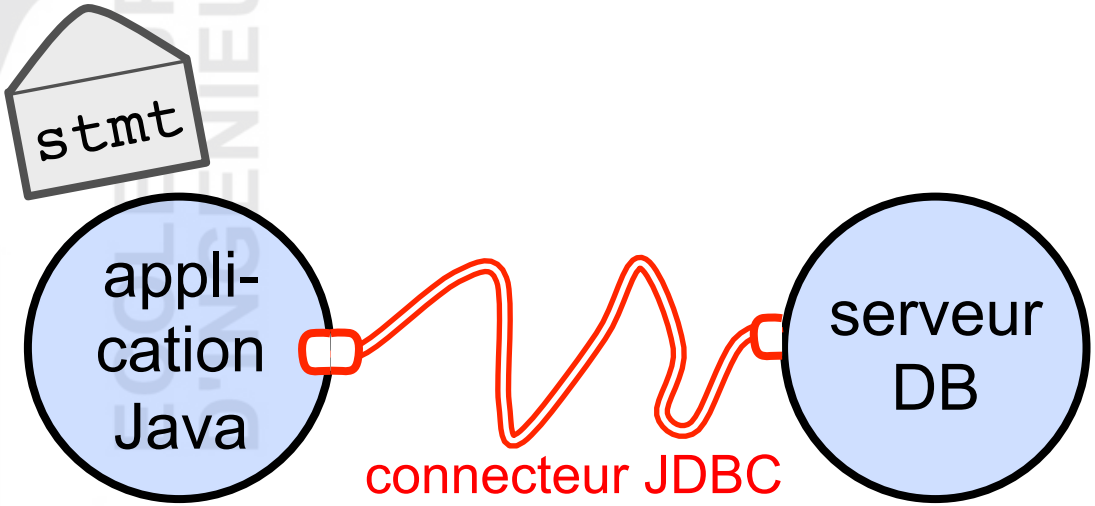
```
Statement stmt = null ;
...
stmt = conn.createStatement() ;
```

- Crée une enveloppe pour requête SQL

La connexion doit être valide

Création d'une requête (2)

```
Statement stmt = null ;
...
stmt = conn.createStatement() ;
```



Création d'une requête (3)

```
String sql = "...";
```

- Crée une requête SQL particulière

Le texte de la requête doit être valide

Envoi d'une requête (1)

- Requête d'interrogation

```
resultSet rs =
        stmt.executeQuery(sql);
```

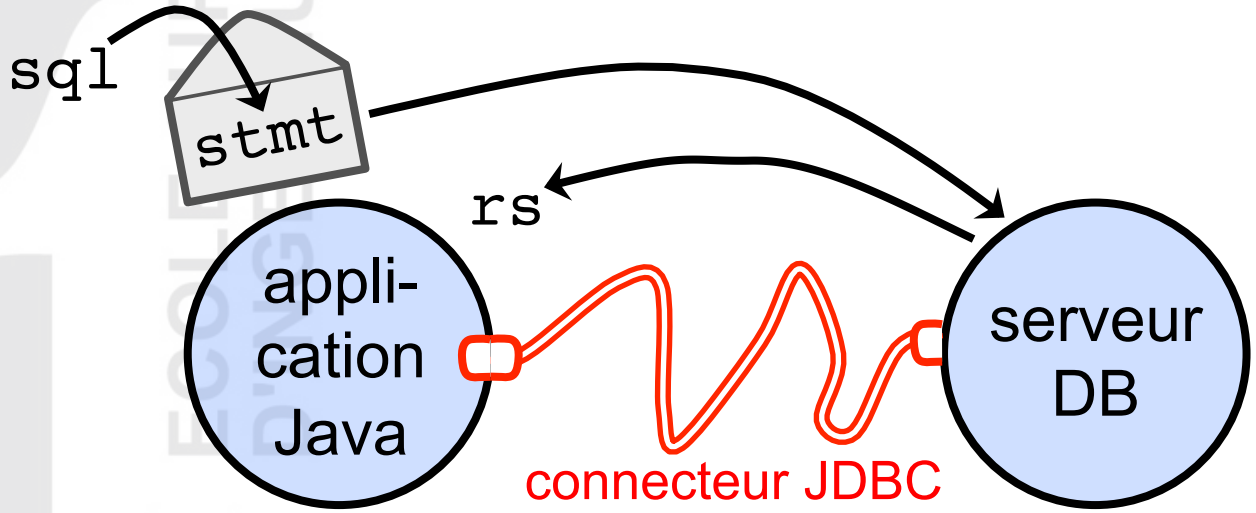
- Soumet une requête d'interrogation et reçoit le résultat dans `rs`

`stmt` doit être un `Statement` valide

Envoi d'une requête (2)

- Requête d'interrogation

```
resultSet rs =
        stmt.executeQuery(sql);
```



Envoi d'une requête (3)

- Requête de mise à jour

```
stmt.executeUpdate(sql);
```

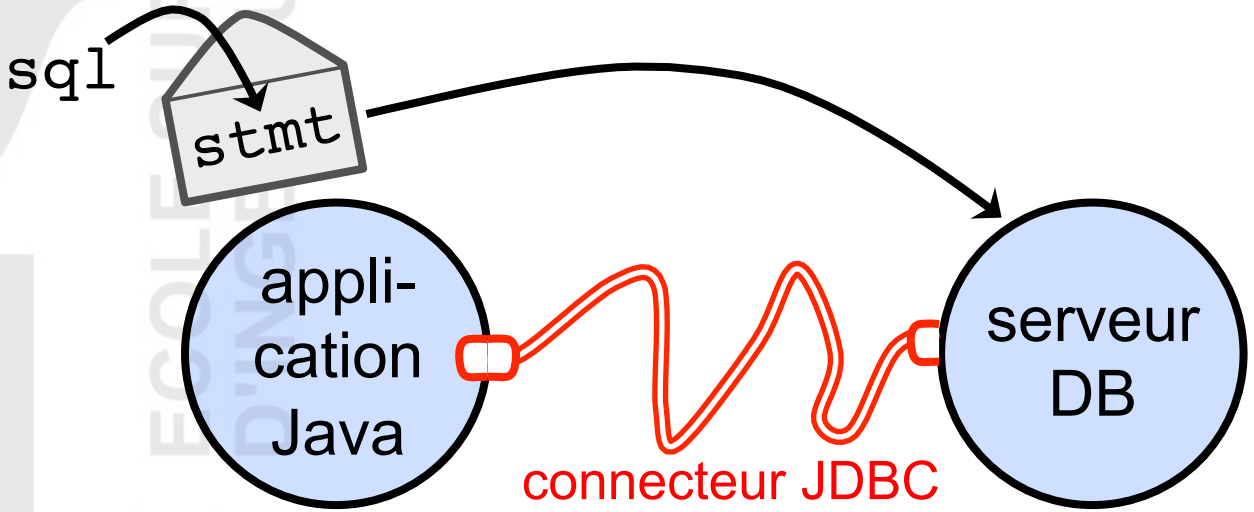
- Soumet une requête de mise à jour

`stmt` doit être un `Statement` valide

Envoi d'une requête (4)

- Requête de mise à jour

```
stmt.executeUpdate(sql);
```



Fermeture d'une requête

```
stmt.close();
```

`stmt` doit être un `Statement` valide

Manipulation d'une relation

Résultat d'une interrogation (1)

- Le résultat d'une requête d'interrogation est une relation…

…un ensemble de lignes construites selon un schéma donné

- `rs` est un **itérateur** qui donne accès tour à tour à chacune des lignes

```
rs.next()
rs.getInt("attr")
rs.getString("attr")
```

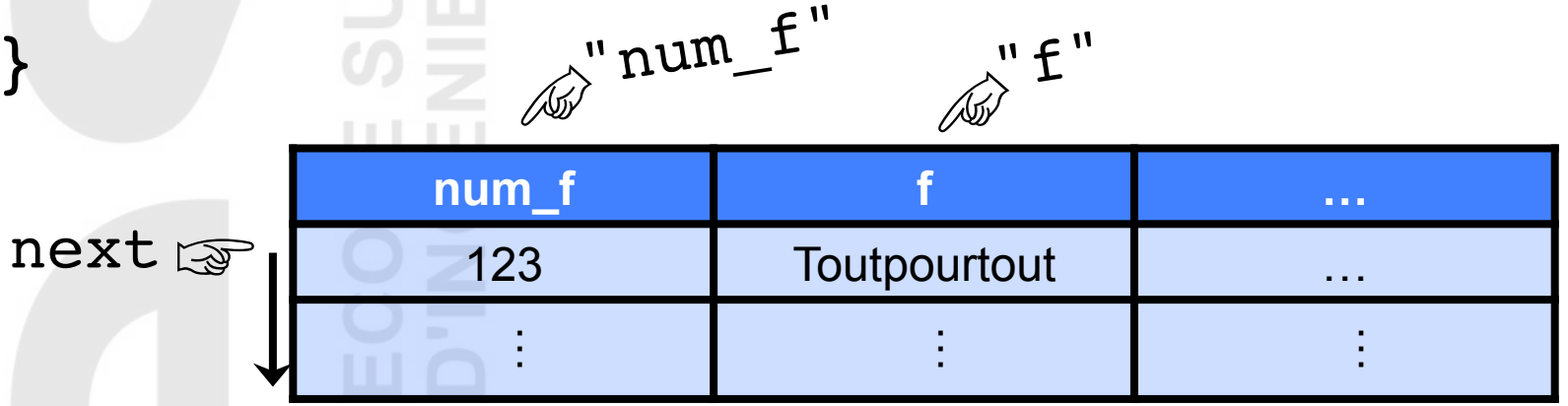
Résultat d'une interrogation (2)

```
while ( rs.next() ) {
  int num_f = rs.getInt( "num_f" ) ;
  String f = rs.getString( "f" ) ;
  ...
}
```

Les attributs de `rs.get…(…)` doivent être des attributs du schéma de la relation résultat

Résultat d'une interrogation (3)

```
while ( rs.next() ) {
  int num_f = rs.getInt( "num_f" ) ;
  String f = rs.getString( "f" ) ;
  ...
}
```



Résultat d'une interrogation (4)

```
rs.close() ;
```

Le `resultSet rs` doit être valide

Gestion des erreurs

Pas de vérification à la compilation

- Liaison dynamique avec le connecteur
- Liaison dynamique avec la BD
- Requêtes SQL traitées comme du texte
- Attributs du résultats traités comme du texte

Les erreurs ne se voient qu'à l'exécution

Grand principe

- **Ne jamais laisser une erreur prévisible non gérée**
  - aider le développeur / ses erreurs
  - aider l'utilisateur / ses erreurs
  - ne pas inquiéter l'utilisateur / erreurs du développeur

Gestion des erreurs en Java (1)

- Les erreurs sont signalées par des **exceptions**
- Un type générique **`Exception`**
- Plusieurs types d'exceptions définis par les applications
  - `ZeroDivide`
  - `ArithmeticException`
  - **`SQLException`**

Gestion des erreurs en Java (2)

```
try{...un calcul...}
catch(AnException e){...}
catch(AnotherException e){...}
...
catch(Exception e){...}
finally{..}
```

- Lance un `calcul` et prévoit un traitement pour différents types d'exception

Gestion des erreurs en Java (3)

```
try{...un calcul...}
catch(AnException e){...}
```

- Attrape une exception signalée par `un calcul`
- En général…

```
System.out.println("Goodbye!");
```

…ou

```
e.printStackTrace();
```

Gestion des erreurs en Java (4)

```
try{...un calcul...}
catch(AnException e){...}
catch(Exception e){...}
```

- Les catch successifs sont traités dans l'ordre…

…finir par un

```
catch (Exception e){...}
```

Gestion des erreurs en Java (5)

```
try{...un calcul...}
...
finally{...}
```

- Lance un `calcul` et prévoit un traitement à exécuter à l'issue du calcul qu'il y ait erreur ou non
- En général…
  - `conn.close()`
  - `stmt.close()`
  - `rs.close()`

Conclusion (1)

`Connection conn`

- Lien vers une base de données particulière

`Statement stmt`

- Requête particulière

`resultSet rs`

- Résultat d'une interrogation

Conclusion (2)

- Erreurs probables…

…ne pouvant être détectées qu'à l'exécution

```
try / catch / finally
```

Conclusion (3)

- Il existe des méthodes de consultation du schéma d'une base de données ou d'une table…

```
DataBaseMetaData dbmd = conn.getMetaData
ResultSetMetaData rsmd = rs.getMetaData
```

…permet d'ajuster les requêtes/accès aux schémas

plus d'erreurs !

…permet de programmer un **client universel** d'accès/manipulation de bases de données