

Les vues dans Oracle

SES – Université de Genève

Les vues - Description

- Les vues sont utilisées dans un SGBD pour remplacer certaines requêtes SQL pour simplifier les tâches d'interrogation.
- Les vues ont le même comportement que les tables:
 - On leur attribue un nom.
 - Elles ont la structure d'une table.
 - Les données sont traitées comme celles des tables.

Les vues - Description

- Une vue ne contient pas de données, il s'agit d'une table sous-jacente qui stock les données présentées par une vue.
- Dynamisme: Quand les données d'une table sont modifiées, les vues fondées dessus sont mises à jour
→ Reflètent toujours l'état actuel des données.

Exemple:

Gestion d'un parc de véhicules (PARCVEH)

- VOITURE (NOV,MV,KM,PSG)
- CH (NCH,CHAUFFEUR)
- V_CH (NOV,NCH,NKM)
- REPARATION (NOREP,NOV,NOG,TYPREP,PX,KMCPT)
- TRAJET
(NOTRAJ,VILLEDEP,VILLEARR,DATETRAJET,NBKM)
- TR_NOV (NOTRAJ,NOV,NCH,NBPERSTR)

Requête VS Vue

- Requete:

```
SELECT trn.nov, c.chauffeur, t.notraj, t.villedep, t.villearr, t.datetrajet FROM  
trajet t, tr_nov trn, ch c WHERE t.notraj=trn.notraj AND trn.nch=c.nch  
ORDER BY trn.nov;
```

- Vue:

- Création:

```
CREATE OR REPLACE VIEW InfoTrajet AS
```

```
SELECT trn.nov, c.chauffeur, t.notraj, t.villedep, t.villearr, t.datetrajet  
FROM trajet t, tr_nov trn, ch c WHERE t.notraj=trn.notraj AND  
trn.nch=c.nch;
```

Requête VS vue

- Vue:
 - Interrogation de la vue:
`SELECT * FROM InfoTrajet;`
`SELECT chauffeur, villedep, villearr, datetrajet FROM infotrajet;`
 - Effacer la vue:
`DROP VIEW InfoTrajet;`
 - Jointures sur les vues
 - Vues sur les vues
 - ...

Connectivité - JDBC

SES – Université de Genève

Introduction

- JDBC(Java DataBase Connectivity):
 - Il s'agit du standard Java pour connecter une application Java à une base de donnée
 - Permet la génération de requêtes dynamiques.
 - Peut être utilisée dans les applications Java, les applets ou les formulaires (JSP).

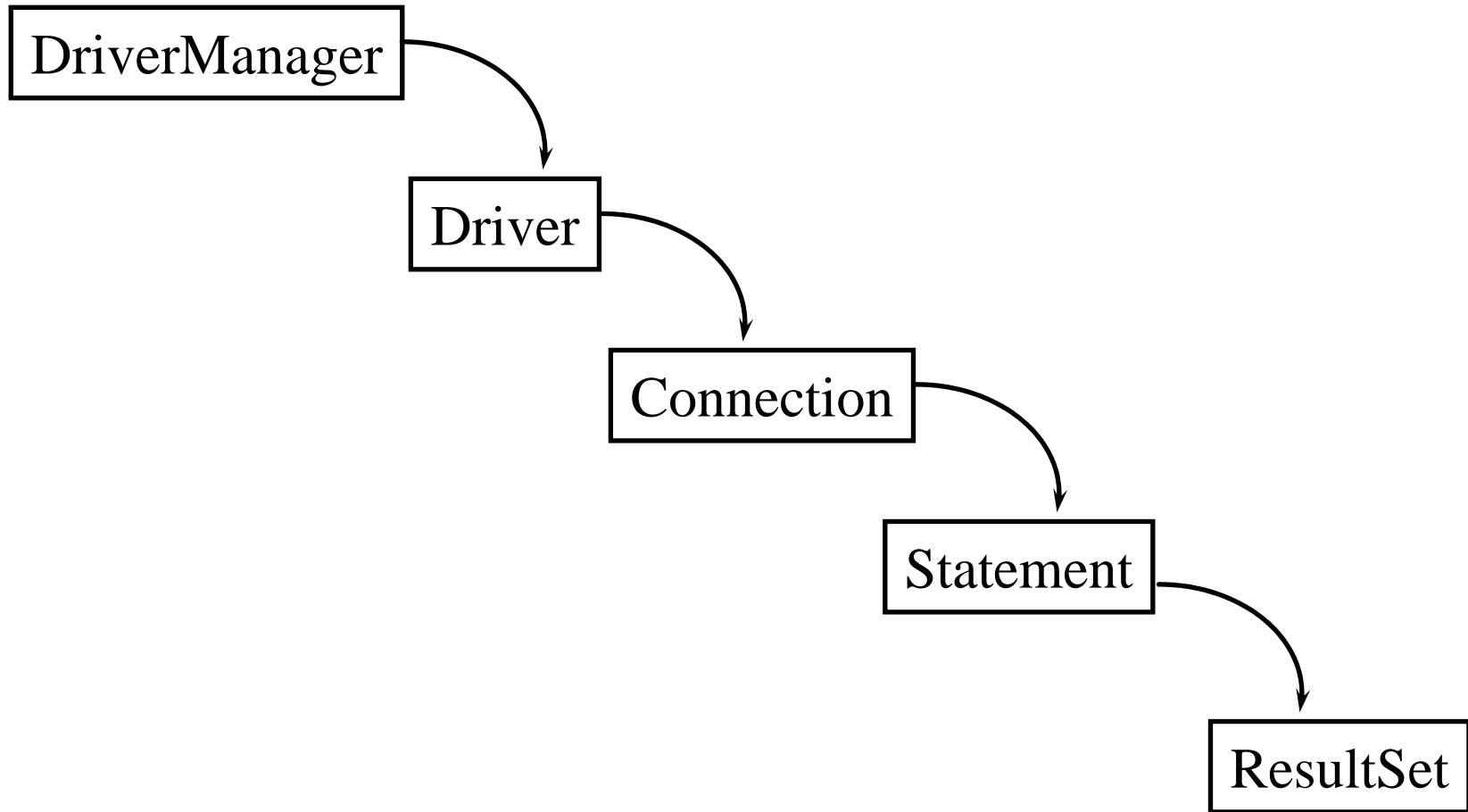
Introduction

- Lorsqu'on utilise une requête SQL dans du code Java, sa syntaxe n'est pas vérifiée à la compilation mais à l'exécution.

Vue d'ensemble

- Package `java.sql.*`;
- `DriverManager`
 - Charge les drivers choisis
- `Driver`
 - connexions à la bd
- `Connection`
 - série de d'instructions SQL de et vers la BD
- `Statement`
 - une instruction SQL
- `ResultSet`
 - le resultat retourné par une instruction (`Statement`)

Utilisation des classes JDBC



JDBC URL

`jdbc:subprotocol:source`

- chaque driver a son propre subprotocol
- chaque subprotocol a sa propre syntaxe pour la source
- Exemple de subprotocol
 - oracle
 - `jdbc:oracle:thin:@hostname:port:dbname`
 - mysql
 - `jdbc:mysql://[hostname][:port]/dbname`

Obtenir une connexion du DriverManager

- Obtenir les archives des drivers :
 - `ojdbc14.jar, classes12.rar` (Oracle)
 - `mysql-connector-java-3.0.11-stable-bin.jar` (MySQL)
- Charger les drivers :
 - `Class.forName("oracle.jdbc.driver.OracleDriver");`
 - `Class.forName("org.gjt.mm.mysql.Driver");`
 - Peut Générer une `ClassNotFoundException`
- Créer la connexion :
 - `Connection con = DriverManager.getConnection(url, "myLogin", "myPassword");`
 - Peut Générer une `SQLException`
 - url : `jdbc:oracle:thin:@hostname:port:dbname`
 - `jdbc:mysql://[hostname][:port]/dbname`

Obtenir un connexion Exemple Oracle

[illegible]

Obtenir un connexion Exemple MySql

[illegible]

Les Statement

- `Statement stmt = con.createStatement();`
- **executeUpdate**
 - `stmt.executeUpdate("insert into ch (nch, chauffeur) values (120, 'Dupond') ");`
- **executeQuery**
 - `stmt.executeQuery("select * from ch");`

Les ResultSet

```
ResultSet rs = stmt.executeQuery ("select * from  
Person");
```

- **next()**

```
rs.next();  
rs.getString (« lastname »);
```

- **while**

```
while(rs.next()){  
    rs.getString (« lastname »);  
}
```

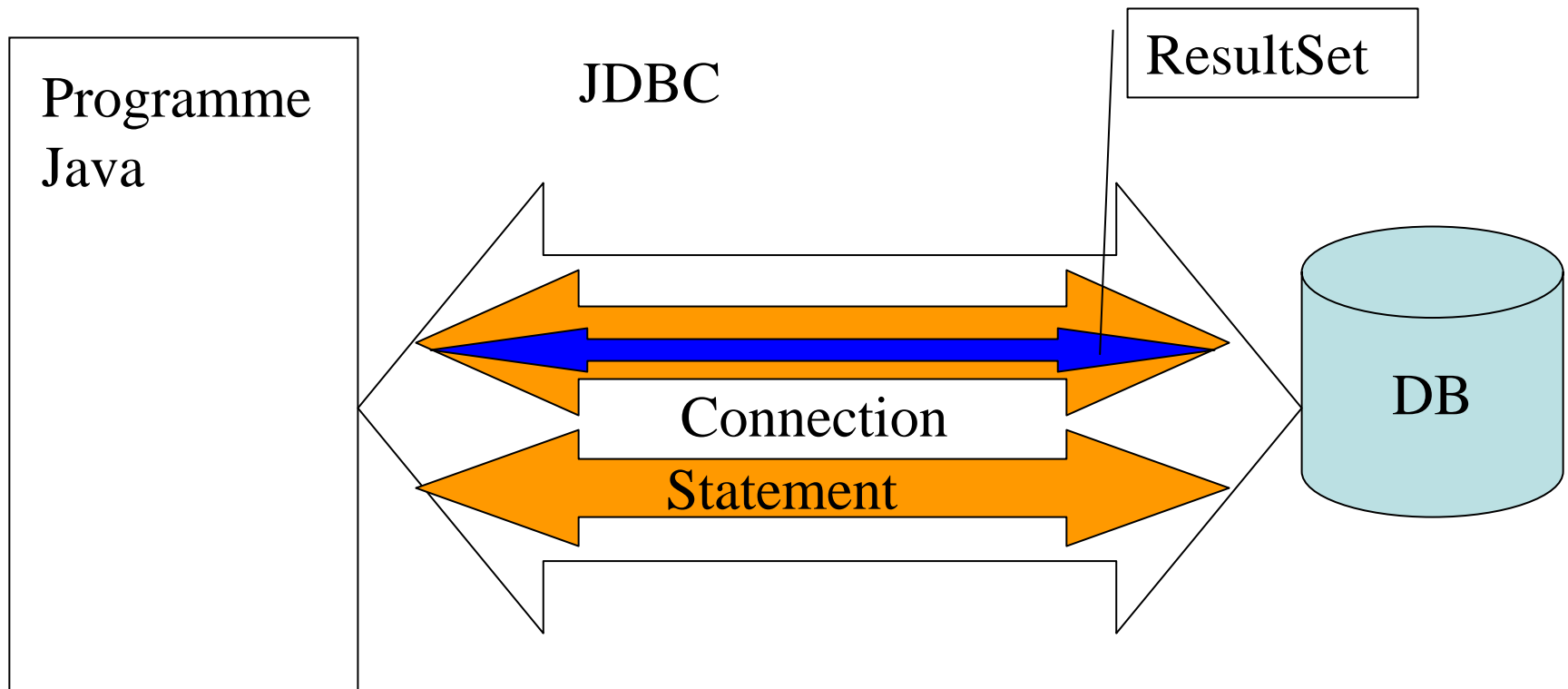
- **ResultSet.getXXX**

PreparedStatement

```
String insert = "INSERT INTO ch (nch,chauffeur)
VALUES (?,?)";
PreparedStatement st =
    conn.prepareStatement(insert);
st.setInt(1, 150);
st.setString(2, "Dupond" );
st.executeUpdate();
```

close() !!!

- Connection, Statement, ResultSet



Ressources limitées

- Connexion
 - cher à la création
 - limité en nombre
- Statement
 - limité en nombre
- Pool de connexion

Mapping type Java Sql

<u>SQL type</u>	<u>Java Type</u>
CHAR, <u>VARCHAR</u> , LONGVARCHAR	String
<u>NUMERIC</u> , DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT, <u>DOUBLE</u>	double
BINARY, <u>VARBINARY</u> , LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

Le temps dans les BD

- Temps en SQL n'est pas standard
- Java définis trois classes pour aider avec le temps
- `java.sql.Date`
 - year, month, day
- `java.sql.Time`
 - hours, minutes, seconds
- `java.sql.Timestamp`
 - year, month, day, hours, minutes, seconds, nanoseconds

Transaction

- Commit et Rollback
- Par défaut les statements sont en auto- commit
`con.setAutoCommit(false);`
- Les statements ne sont validés qu'avec un appel à commit.
- Quand appeler rollback.
 - Lorsqu'une SQLException est levée.

Exemple: Chauffeur.java

- `import java.sql.*;`
- `class Chauffeur`
- `{`
- `public static void main (String args [])`
- `throws SQLException`
- `{`
- `try{`
- `// Charger le driver JDBC`
- `Class.forName("oracle.jdbc.driver.OracleDriver");`
- `// Connection à la base`
- `// syntaxe: <hote>:<port>:<sid>.`
- `String url = "jdbc:oracle:thin:@129.194.69.101:1721:cuiprd";`
- `String userName = "scott";`
- `String password = "tiger";`
- `if (args.length > 0) url = args[0];`
- `if (args.length > 1) userName = args[1];`
- `if (args.length > 2) password = args[2];`
- `}`

Exemple: Chauffeur.java

- `Connection conn =`
- `DriverManager.getConnection (url, userName, password);`
- `// Create a Statement`
- `Statement stmt = conn.createStatement ();`
-
- `ResultSet rset = stmt.executeQuery ("select nch, chauffeur from ch");`
- `//on récupère les informations et les affiche`
- `while (rset.next ())`
- `{`
- `System.out.print (rset.getInt (1));`
- `System.out.println (rset.getString (2));`
- `}`

Exemple: Chauffeur.java

- `catch (ClassNotFoundException cnfe)`
- `{`
- `System.out.println("Error loading driver");`
- `}`
- `catch (SQLException cnfe)`
- `{`
- `System.out.println("SQL Error");`
- `}`
- `catch (Exception e)`
- `{`
- `System.out.println("General Error");`
- `}`
- `}`
- `}`

Exemple - JSP

- `<%@ page import="java.sql.*" %>`
- `<%`
- `Class.forName("oracle.jdbc.driver.OracleDriver");`
- `String url = "jdbc:oracle:thin:@129.194.69.101:1521:cuiprd";`
- `String userName = "scott";`
- `String password = "tiger";`
- `Connection connection =`
- `DriverManager.getConnection (url, userName, password);`
- `Statement stmt = connection.createStatement ();`
- `String query = "Select * From ch";`
- `java.sql.ResultSet rs = stmt.executeQuery(query);`
- `%>`
- `<TABLE border='1'>`
- `<TR>`
- `<TD> Numéro`
- `</TD>`
- `<TD> Nom`
- `</TD>`
- `</TR>`

Exemple - JSP

- <%
- String colorPa = "#CCCCCC";
- while(rs.next()){
- %>
- <TR bgcolor='<%=colorPa%>'>
- <TD><%= rs.getString(1) %>
- </TD>
- <TD><%= rs.getString(2) %>
- </TR>
-
- <%
- if(colorPa.equals("#CCCCCC")) colorPa = "white";
- else colorPa = "#CCCCCC";
- } //fin while
- connection.close();
- %>
- </table>