Récupérer des objets complexes d'oracle depuis Java

Hyacinthe MENIET

3 août 2019

Je vais indiquer, dans ce document, comment depuis une classe Java, récupérer un objet complexe SQL. Pour y parvenir je vais utiliser l'API JDBC (Java DataBase Connectivity) qui permet aux applications Java d'accéder par le biais d'une interface commune à des bases de données. Pour être le moins théorique possible, ce didacticiel consiste à réaliser un exemple.

1. Pré-requis

- Vous êtes familier du SQL et du PL/SQL
- Vous êtes familier de JDBC
- Vous disposez d'une base de données Oracle 8i minimum
- Vous disposez du JDK 8

2. Mapping Oracle vers Java et vice versa

Voici un tableau qui indique en fonction de ce qui est retourné par le SQL l'objet Java le mieux adapté :

Types SQL	Types Java
CHAR	String
VARCHAR2	String
LONG	String
NUMBER	java.lang.Byte, java.lang.Short, java.lang.Integer, java.lang.Long, java.lang.Float, java.lang.Double, java.math.BigDecimal, byte, short, int, long, float, double
RAW	byte[]
LONGRAW	byte[]
DATE	java.sql.Date ou java.sql.Timestamp
ROWID	oracle.sql.ROWID
REF CURSOR	java.sql.ResultSet
BLOB	oracle.sql.BLOB
CLOB	oracle.sql.CLOB
BFILE	oracle.sql.BFILE
Oracle object	Une classe spécifié dans le « *type map* » sinon oracle.sql.STRUCT
Oracle object reference	oracle.sql.REF
collection (varray ou nested table)	oracle.sql.ARRAY

Je me concentre sur les types complexes, ce qui correspond aux lignes « *Oracle object* » et « *REF CURSOR* ». Pour comprendre ce didacticiel vous devez impérativement posséder les bases

de JDBC c'est-à-dire savoir récupérer des types simples (VARCHAR2, NUMBER, DATE ...).

3. Les Oracle object

Les *« Oracle object »* sont des objets SQL. Ils peuvent être stockés dans des tables ou instanciés à la volée dans du PL/SQL. Pour illustrer cette partie, vous allez créer des objets dans la base de donnée. Vous allez ensuite les récupérer par JDBC et les stocker dans des JavaBeans. C'est le sens Oracle vers Java. Pour le sens Java vers Oracle vous allez instancier des JavaBean et vous allez insérer leurs données dans Oracle via JDBC.

3.1 Côté Oracle

Créez l'objet *« PERSON »* ainsi :

```
CREATE OR REPLACE TYPE "PERSON" as object (
firstname   varchar2(25),
name        varchar2(25),
birthyear   number
)
```

Ensuite créez la table *« FRIENDS »* qui contiendra des *« PERSON »* :

```
create table FRIENDS (
  f_id   number      primary key,
  f_pers PERSON
)
```

Insérez dans la table *« FRIENDS »* quelques *« PERSON »* :

```
INSERT INTO FRIENDS VALUES (1,PERSON( 'Hyacinthe','MENIET'));
INSERT INTO FRIENDS VALUES (2,PERSON( 'Larry','ELLISON'));
INSERT INTO FRIENDS VALUES (3,PERSON( 'Bill','JOY'));
```

3.2 Côté Java

Créez la classe PersonBean ainsi :

```
package net.dotmyself.j2sql;
import java.io.Serializable;
import java.sql.SQLData;
import java.sql.SQLException;
import java.sql.SQLInput;
import java.sql.SQLOutput;
/**
* Maps Oracle "PERSON" object
```

```
* @author Hyacinthe MENIET
*/
public class PersonBean implements SQLData, Serializable {
private static final long serialVersionUID = 1L;
private String firstname;
private String name;
private int birthYear;
@Override
public void readSQL(SQLInput stream, String typeName) throws
   SQLException {
name = stream.readString();
firstname = stream.readString();
birthYear = stream.readInt();
}
@Override
public void writeSQL(SQLOutput stream) throws SQLException {
stream.writeString(name);
stream.writeString(firstname);
stream.writeInt(birthYear);
}
@Override
public String getSQLTypeName() throws SQLException {
return "PERSON";
}
public int getBirthYear() {
return birthYear;
}
public void setBirthYear(int birthYear) {
this.birthYear = birthYear;
}
public String getFirstname() {
return firstname;
}
public void setFirstname(String firstname) {
this.firstname = firstname;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
}
```

A chaque *« PERSON »* chez Oracle correspondra un JavaBean *« PersonBean »*. Pour que ça fonctionne, PersonBean doit implémenter SQLData et Serializable. La méthode getSQLTypeName() indique le type SQL correspondant au JavaBean. La méthode writeSQL() est utilisée dans le sens Java vers Oracle et La méthode readSQL() dans l'autre sens. L'ordre de lecture et d'écriture dans *« stream »* est très important. Il correspond à celui que vous avez utilisé pour déclarer les variables

dans « *PERSON* ».

3.3 Echange de données

Créez la classe SQL2Java ainsi :

```
package net.dotmyself.j2sql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
* Creates JavaBeans which contain data from Oracle.
*
* @author Hyacinthe MENIET
*/
public class SQL2Java {
public static void main(String[] args) {
try {
int pid;
PersonBean person;
Connection conn;
conn = DriverManager.getConnection("jdbc:oracle:thin:@192
    .168.0.8.1:1520:JBD",
"login", "password");
Map<String, Class< ?>> map = conn.getTypeMap();
map.put("PERSON", Class.forName("net.dotmyself.j2sql.PersonBean")
    );
conn.setTypeMap(map);
try (Statement stmt = conn.createStatement()) {
ResultSet rs = stmt.executeQuery("SELECT f_id,f_pers FROM FRIENDS
    ");
while (rs.next()) {
pid = rs.getInt("F_ID");
person = (PersonBean) rs.getObject("F_PERS");
System.out.println(pid + ". " + person.getFirstname() + " "
+ person.getName() + " ( " + person.getBirthYear() + " )");
}
rs.close();
}
conn.close();
} catch (SQLException | ClassNotFoundException ex) {
Logger.getLogger(SQL2Java.class.getName()).log(Level.SEVERE, null
    , ex);
```

```
}
}
}
```

Cette classe instancie des JavaBean à partir de données Oracle. Elle réalise le sens Oracle vers Java.
Créez la classe Java2SQL ainsi :

```
package net.dotmyself.j2sql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
* Creates Oracle objects which contain data from JavaBeans.
* @author Hyacinthe MENIET
*/
public class Java2SQL {
public static void main(String[] args) {
try {
PersonBean person = new PersonBean();
Connection conn;
conn = DriverManager.getConnection("jdbc:oracle:thin:@192
    .168.0.8.1:1520:JBD", "login", "password");
Map<String, Class<?>> map = conn.getTypeMap();
map.put("PERSON", Class.forName("net.dotmyself.j2sql.PersonBean")
    );
conn.setTypeMap(map);
person.setBirthYear(2005);
person.setFirstname("dot");
person.setName("Myself");
try (PreparedStatement pstmt = conn.prepareStatement("INSERT INTO
    FRIENDS VALUES (?,?)")) {
pstmt.setInt(1, 4);
pstmt.setObject(2, person);
pstmt.executeUpdate();
}
conn.close();
} catch (SQLException | ClassNotFoundException ex) {
Logger.getLogger(Java2SQL.class.getName()).log(Level.SEVERE, null
    , ex);
}
}
}
```

Cette Classe créée des objets Oracle à partir de JavaBeans. Elle réalise le sens Java vers Oracle.
Il ne vous reste plus qu'à modifier les chaînes connections respectives et à les exécuter.

4. Oracle REF CURSOR

Les *« Oracle REF CURSOR »* sont des références sur des curseurs, ils s'utilisent généralement dans du code PL/SQL. Pour illustrer cette partie, vous allez créer une référence sur un curseur dans un package PL/SQL puis vous récupérerez les données référencées, dans une classe Java.

4.1 Côté Oracle

Créez une table *« CONTACTS »* ainsi :

```
create table CONTACTS (
  c_id number primary key,
  c_firstname varchar2(25),
  c_name varchar2(25)
)
```

Insérez quelques contacts dans *« CONTACTS »* :

```
INSERT INTO CONTACTS VALUES (1,'Hyacinthe','MENIET');
INSERT INTO CONTACTS VALUES (2,'Larry','ELLISON');
INSERT INTO CONTACTS VALUES (3,'Bill','JOY');
```

Créez le package DEMO PL/SQL. La déclaration est la suivante :

```
CREATE OR REPLACE PACKAGE DEMO AS
TYPE ref_cursor IS REF CURSOR;
PROCEDURE get_contacts (p_ref OUT ref_cursor);
END DEMO;
```

Le body ressemble à ça :

```
CREATE OR REPLACE PACKAGE BODY DEMO AS
PROCEDURE get_contacts (p_ref OUT ref_cursor)
IS
BEGIN
OPEN p_ref for
SELECT c_id,c_firstname,c_name
FROM CONTACTS
ORDER BY c_name,c_firstname;
END get_contacts;
END DEMO;
```

4.2 Echange de données

Créez la classe Ref2Java qui permet d'appeler le code PL/SQL ci-dessus :

```
package net.dotmyself.j2sql;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.JDBCType;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
* Displays data from Oracle.
* @author Hyacinthe MENIET
*/
public class Ref2Java {
public static void main(String[] args) {
try {
Connection conn;
conn = DriverManager.getConnection("jdbc:oracle:thin:@192
    .168.0.8.1:1520:JBD",
"login", "password");
CallableStatement cstmt;
cstmt = conn.prepareCall("{call DEMO.get_contacts(?) }");
cstmt.registerOutParameter(1, JDBCType.REF_CURSOR); // p_ref
cstmt.executeQuery();
try (ResultSet rs = (ResultSet) cstmt.getObject(1)) {
while (rs.next()) {
System.out.println(rs.getInt("C_ID") + ". " +
rs.getString("C_FIRSTNAME") + " " + rs.getString("C_NAME"));
}
}
cstmt.close();
conn.close();
} catch (SQLException ex) {
Logger.getLogger(Ref2Java.class.getName()).log(Level.SEVERE, null
    , ex);
}
}
}
```

Il ne vous reste plus qu'à modifier la chaîne de connexion et à exécuter la classe.