SQL

JDBC For Access

前言

jdk1.8之后取消了 jdbc-odbc 的桥接器, 因为Oracle 认为 odbc 只能作为连接 Access 的中间解决方案, odbc 以 jdbc 为基础, 因此性能比 jdbc 更慢, 且依赖与 Windows 平台, 与 java 理念不符.

解决方案

因此, jdk1.8 之后对于 Access 数据库的访问有一下几种解决方案:

1. 从 jdk1.7 拷贝 jdbc-odbc 的 jar 包到jdk1.8

1.1 优缺点

- 能够兼容老代码, 且不会引进新的问题
- 存在 odbc 存在的问题
- 对于产品而言依赖于用户jdk 版本

2. jackcess

2.1 优缺点

- 没有SQL 引擎, 不支持程序员自己写 SQL

3. ucanaccess

3.1 优缺点

- 能够让程序员自己写 SQL
- 开源
- 功能不完善, 还有很多 feature 没有实现, 将抛出 `FeatureNotSupportedException`

```
@Override
public ResultSet getCatalogs() throws SQLException {
    if (this.connection.isShowSchema()) {
        try {
            return wrapped.getCatalogs();
        } catch (SQLException e) {
            throw new UcanaccessSQLException(e);
        }
    }
    throw new FeatureNotSupportedException();
}
```

3.2 示例代码

```
@Test
public void testFindUser() throws Exception {
    Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
    Connection con = DriverManager.getConnection("jdbc:ucanaccess://c:/db
2.accdb", "", "");
    Statement stmt = con.createStatement();
    DatabaseMetaData meta = con.getMetaData();
    System.out.println(meta.getCatalogs());

    ResultSet rs = stmt.executeQuery("select * from user");
    while (rs.next()) {
        System.out.println(rs.getString("name"));
    }
}
```

4.caigen

4.1 优缺点

- 功能完善
- 不支持 jdbc4
  `conn.isValid(5);`会抛错.
- 未开源, 收费

4.2 示例代码

```
@Test
public void test01() throws Exception {
    // inetsoft.uql.jdbc.pool.access2.connectionTestQuery=select 1
    String jdbURL = "jdbc:access:/c:/order.mdb";
```

```
    Class.forName("com.caigen.sql.access.AccessDriver");
    Properties props = new Properties();
    props.setProperty("delayedClose", "0");
    props.setProperty("url", jdbURL);
    DataSource poolDataSource = new CaigenConnectionPoolDataSource(prop
s);
    // new HikariDataSource();

//        Connection conn = DriverManager.getConnection(jdbURL, props);
    Connection conn = poolDataSource.getConnection();
    System.out.println(conn.getClass());
//        conn.isValid(5);
    Statement stmt = conn.createStatement();

    String sql = "SELECT * FROM orders";
    System.out.println(sql);
    ResultSet rs = stmt.executeQuery(sql);
    ResultSetMetaData resultSetMetaData = rs.getMetaData();

    DatabaseMetaData meta = conn.getMetaData();
    System.out.println(meta.getCatalogs());

    int iNumCols = resultSetMetaData.getColumnCount();
    for (int j = 1; j <= iNumCols; j++) {
        System.out.println(resultSetMetaData.getColumnName(j) + "  " + re
sultSetMetaData.getColumnTypeName(j) + "  "
                + resultSetMetaData.getPrecision(j) + "  " + resultSetMet
aData.getScale(j));
    }

    Object colval;

    while (rs.next()) {
        for (int j = 1; j <= iNumCols; j++) {
            colval = rs.getObject(j);
            System.out.print(colval + "\t");
        }
        System.out.println();
    }
}
```