[**如何获取新插入Oracle数据库Sequence值的5种方法**](http://blog.csdn.net/linwei_1029/article/details/7010539)

分类： [Oracle知识](http://blog.csdn.net/linwei_1029/article/category/655442)2011-11-25 09:14939人阅读[评论](http://blog.csdn.net/linwei_1029/article/details/7010539#comments)(0)[收藏](javascript:void(0);)[举报](http://blog.csdn.net/linwei_1029/article/details/7010539#report)

Oracle的sequence实现非常灵活，所以也带来一些易用性问题，如何取到新插入记录生成的sequence值与其它数据库有较大差别，下面介绍了5种实现读取新插入记录sequence值的方法。

测试用的数据库脚本：

[view plain](http://blog.csdn.net/yzsind/article/details/6918506)

1. SQL> **create** **table** T1
2. 2 (
3. 3 ID NUMBER
4. 4 );
5. **Table** created
6. SQL> **create** **sequence** SEQ\_T1;
7. **Sequence** created

[view plain](http://blog.csdn.net/yzsind/article/details/6918506)

1. //公共代码：得到数据库连接
2. **public** Connection getConnection() **throws** Exception{
3. Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
4. Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@127.0.0.1:1521:dbname", "username", "password");
5. **return** conn;
6. }
7. //方法一
8. //先用select seq\_t1.nextval as id from dual 取到新的sequence值。
9. //然后将最新的值通过变量传递给插入的语句：insert into t1(id) values(?)
10. //最后返回开始取到的sequence值。
11. //这种方法的优点代码简单直观，使用的人也最多，缺点是需要两次sql交互，性能不佳。
12. **public** **int** insertDataReturnKeyByGetNextVal() **throws** Exception {
13. Connection conn = getConnection();
14. String vsql = "select seq\_t1.nextval as id from dual";
15. PreparedStatement pstmt =(PreparedStatement)conn.prepareStatement(vsql);
16. ResultSet rs=pstmt.executeQuery();
17. rs.next();
18. **int** id=rs.getInt(1);
19. rs.close();
20. pstmt.close();
21. vsql="insert into t1(id) values(?)";
22. pstmt =(PreparedStatement)conn.prepareStatement(vsql);
23. pstmt.setInt(1, id);
24. pstmt.executeUpdate();
25. System.out.print("id:"+id);
26. **return** id;
27. }
28. //方法二
29. //先用insert into t1(id) values(seq\_t1.nextval)插入数据。
30. //然后使用select seq\_t1.currval as id from dual返回刚才插入的记录生成的sequence值。
31. //注：seq\_t1.currval表示取出当前会话的最后生成的sequence值，由于是用会话隔离，只要保证两个SQL使用同一个Connection即可，对于采用连接池应用需要将两个SQL放在同一个事务内才可保证并发安全。
32. //另外如果会话没有生成过sequence值，使用seq\_t1.currval语法会报错。
33. //这种方法的优点可以在插入记录后返回sequence，适合于数据插入业务逻辑不好改造的业务代码，缺点是需要两次sql交互，性能不佳，并且容易产生并发安全问题。
34. **public** **int** insertDataReturnKeyByGetCurrVal() **throws** Exception {
35. Connection conn = getConnection();
36. String vsql = "insert into t1(id) values(seq\_t1.nextval)";
37. PreparedStatement pstmt =(PreparedStatement)conn.prepareStatement(vsql);
38. pstmt.executeUpdate();
39. pstmt.close();
40. vsql="select seq\_t1.currval as id from dual";
41. pstmt =(PreparedStatement)conn.prepareStatement(vsql);
42. ResultSet rs=pstmt.executeQuery();
43. rs.next();
44. **int** id=rs.getInt(1);
45. rs.close();
46. pstmt.close();
47. System.out.print("id:"+id);
48. **return** id;
49. }
50. //方法三
51. //采用pl/sql的returning into语法，可以用CallableStatement对象设置registerOutParameter取得输出变量的值。
52. //这种方法的优点是只要一次sql交互，性能较好，缺点是需要采用pl/sql语法，代码不直观，使用较少。
53. **public** **int** insertDataReturnKeyByPlsql() **throws** Exception {
54. Connection conn = getConnection();
55. String vsql = "begin insert into t1(id) values(seq\_t1.nextval) returning id into :1;end;";
56. CallableStatement cstmt =(CallableStatement)conn.prepareCall ( vsql);
57. cstmt.registerOutParameter(1, Types.BIGINT);
58. cstmt.execute();
59. **int** id=cstmt.getInt(1);
60. System.out.print("id:"+id);
61. cstmt.close();
62. **return** id;
63. }
64. //方法四
65. //采用PreparedStatement的getGeneratedKeys方法
66. //conn.prepareStatement的第二个参数可以设置GeneratedKeys的字段名列表，变量类型是一个字符串数组
67. //注：对Oracle数据库这里不能像其它数据库那样用prepareStatement(vsql,Statement.RETURN\_GENERATED\_KEYS)方法，这种语法是用来取自增类型的数据。
68. //Oracle没有自增类型，全部采用的是sequence实现，如果传Statement.RETURN\_GENERATED\_KEYS则返回的是新插入记录的ROWID，并不是我们相要的sequence值。
69. //这种方法的优点是性能良好，只要一次sql交互，实际上内部也是将sql转换成oracle的returning into的语法，缺点是只有Oracle10g才支持，使用较少。
70. **public** **int** insertDataReturnKeyByGeneratedKeys() **throws** Exception {
71. Connection conn = getConnection();
72. String vsql = "insert into t1(id) values(seq\_t1.nextval)";
73. PreparedStatement pstmt =(PreparedStatement)conn.prepareStatement(vsql,**new** String[]{"ID"});
74. pstmt.executeUpdate();
75. ResultSet rs=pstmt.getGeneratedKeys();
76. rs.next();
77. **int** id=rs.getInt(1);
78. rs.close();
79. pstmt.close();
80. System.out.print("id:"+id);
81. **return** id;
82. }
83. //方法五
84. //和方法三类似，采用oracle特有的returning into语法，设置输出参数，但是不同的地方是采用OraclePreparedStatement对象，因为jdbc规范里标准的PreparedStatement对象是不能设置输出类型参数。
85. //最后用getReturnResultSet取到新插入的sequence值，
86. //这种方法的优点是性能最好，因为只要一次sql交互，oracle9i也支持，缺点是只能使用Oracle jdbc特有的OraclePreparedStatement对象。
87. **public** **int** insertDataReturnKeyByReturnInto() **throws** Exception {
88. Connection conn = getConnection();
89. String vsql = "insert into t1(id) values(seq\_t1.nextval) returning id into :1";
90. OraclePreparedStatement pstmt =(OraclePreparedStatement)conn.prepareStatement(vsql);
91. pstmt.registerReturnParameter(1, Types.BIGINT);
92. pstmt.executeUpdate();
93. ResultSet rs=pstmt.getReturnResultSet();
94. rs.next();
95. **int** id=rs.getInt(1);
96. rs.close();
97. pstmt.close();
98. System.out.print("id:"+id);
99. **return** id;
100. }

以上5种方法都可以实现功能，以下是5种方法的优缺点汇总，个人推荐性能要求一般的业务采用第一种方法，性能要求非常高业务采用第五种方法。

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **简介** | **优点** | **缺点** |
| **方法一** | 先用seq.nextval取出值，然后用转入变量的方式插入 | 代码简单直观，使用的人也最多 | 需要两次sql交互，性能不佳 |
| **方法二** | 先用seq.nextval直接插入记录，再用seq.currval取出新插入的值 | 可以在插入记录后返回sequence，适合于数据插入业务逻辑不好改造的业务代码 | 需要两次sql交互，性能不佳，并且容易产生并发安全问题 |
| **方法三** | 用pl/sql块的returning into语法，用CallableStatement对象设置输出参数取到新插入的值 | 只要一次sql交互，性能较好 | 需要采用pl/sql语法，代码不直观，使用较少 |
| **方法四** | 设置PreparedStatement需要返回新值的字段名，然后用getGeneratedKeys取得新插入的值 | 性能良好，只要一次sql交互 | 只有Oracle10g才支持，使用较少 |
| **方法五** | returning into语法，用OraclePreparedStatement对象设置输出参数，再用getReturnResultSet取得新增入的值 | 性能最好，因为只要一次sql交互，oracle9i也支持 | 只能使用Oracle jdbc特有的OraclePreparedStatement对象 |

**分享到：**

 上一篇：[httpclient通过POST来上传文件，而不是通过流的形式，并在服务端进行解析(通过httpmime.jar来操作)](http://blog.csdn.net/linwei_1029/article/details/6990971)

 下一篇：[struts1.2中ActionForm和ServletFileUpload.parseRequest(request)不能同时使用](http://blog.csdn.net/linwei_1029/article/details/7010561)