

知识点列表

编号	名称	描述	级别
1	分页策略 – 基于缓存的分页 功能增强-容错	增加对于用户错误输入的处理	**
2	分页策略 – 基于查询的分页	学习 Oracle 中利用 rownum 伪劣获取部分数据的方式。	**
3	分页策略 – 基于查询的分页 功能增强-排序	学习 Oracle 中利用 rownum 伪劣排序后获取部分数据的方式。	**
4	通过 JDBC 调用存储过程	学习 JDBC 调用存储过程的相关 API	*

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. 知识点回顾.....	3
2. 完善基于缓存的分页策略(增加功能) **	3
3. 基于查询的分页 **	9
3.1. 基于查询的分页演示	9
3.2. Mysql 数据库中的分页.....	12
3.3. 两种分页策略总结	12
4. 实现排序后再分页 **	12
4.1. Rownum 伪列的特点	12
5. 存储过程 && 通过 JDBC API 调用存储过程 *	21

1. 知识点回顾

- 1) **JDBC 原理** 面向接口编程
- 2) **JDBC 常用 API**
 - ✓ DriverManager 类 驱动管理器，用于获得 Connection 对象
 - ✓ Connection 接口 连接对象，用于获得获得 Statement 对象
 - ✓ Statement 用于执行静态 SQL 并可返回结果的对象
 - PreparedStatement 继承自 Statement，预编译的 Sql 语句对象
 - CallableStatement 继承自 PreparedStatement，用于执行存储过程(了解)
 - ✓ ResultSet 查询返回的数据库结果集
 - 结果集相当于数据库中的游标(cursor)
 - next()方法的功能 包括两个动作：下移一行，并判断下一行有没有数据
- 3) **JDBC 中的特性**
 - ✓ 事务
 - JDBC 中的事务是自动提交的(autoCommit = true)
 - 程序员如果想控制事务，需要调用方法 setAutoCommit(false/true)进行显式控制
 - ✓ 批处理
stmt.addBatch() / stmt.executeBatch()
 - ✓ 可滚动的结果集 实现分页
 - 如果没有必要，不要使用可滚动结果集，维护指针会消耗额外资源
比如遍历，使用单步单向的结果集 ;需要分页则使用可滚动的结果集

2. 完善基于缓存的分页策略(增加功能)**

【案例 1】基于缓存的分页策略演示(如果用户输入页数不规范：超过最大页/负数/非数字)

- 实现功能
 - 1) 当用户输入超过最大页时： 跳到最后一页
 - 2) 当用户输入为负数时： 跳到第一页
 - 3) 当用户输入为非数字时： 跳到第一页
- 参考代码

```

1 package day03;
2 import java.sql.*;
3 /**
4  * 基于缓存的分页策略
5  * @author teacher
6  */
7 public class BufferPageDemo {
8     public static void main(String[] args) {
9         //从第3页开始, 打印10条记录
10        getPage(10, "3"); // 21-30
11    }
12
13    /**
14     * 打印指定页的数据,用mytemp_xxx表做测试
15     *
16     * begin: 在结果集中第page页的记录起点
17     * begin = (page - 1) * pageSize + 1;
18     *
19     * @param pageSize 每页多少条记录
20     * @param page 第几页
21     */
22    public static void getPage(
23        int pageSize, String strPage){
24
25        int page = 1;
26
27        /* 实现功能3 */
28        //防止用户输入类似"abc@!"等字符作为页数
29        //强制转到第一页
30        try{
31            page = Integer.parseInt(strPage);
32        }catch (NumberFormatException e){
33            page = 1;
34        }
35
36        /* 实现功能1 */
37        //防止用户输入页数超过最大数
38        int totalPage =
39            getTotalPage(pageSize); //获取总页数
40        if (page > totalPage){
41            //当用户输入超出最大页数, 打印最后一页
42            page = totalPage;
43        }

```

```

44
45     /* 实现功能2 */
46     //防止用户输入负数
47     if (page < 1){
48         page = 1;
49     }
50
51     //在结果集中，第page页的记录起点
52     int begin = (page - 1) * pageSize + 1;
53
54     String sql = "select * from mytemp_xxx";
55     Connection conn = ConnectionUtils.getConnection()
56     Statement stmt = null;
57     ResultSet rs = null;
58
59     try {
60         //约定结果集的类型和并发性
61         stmt = conn.createStatement(
62             //1) 设置类型为可滚动的结果集(可跳步的)
63             ResultSet.TYPE_SCROLL_INSENSITIVE,
64             //2) 设置并发性为其他用户只读
65             ResultSet.CONCUR_READ_ONLY);
66
67         //结果集为可滚动的，可进行分页操作
68         rs = stmt.executeQuery(sql);
69
70
71         //指针跳到结果集中的起点begin
72         rs.absolute(begin);
73         //循环pageSize次，取数据并打印
74         for (int i = 0; i < pageSize; i++){
75             System.out.println(rs.getInt("id"));
76             if (!rs.next()){
77                 break;
78             }
79         }
80
81     } catch (SQLException e) {
82         e.printStackTrace();
83     } finally{
84         ConnectionUtils.close(rs);
85         ConnectionUtils.close(stmt);
86         ConnectionUtils.close(conn);
    
```

```

87     }
88 }
89
90 /**
91  * 获得表得到总页数TotalPage
92  * 根据每页的记录数,计算共多少页
93  *
94  * @param pageSize 每页多少条
95  * @return 总页数
96  */
97 private static int getTotalPage(int pageSize) {
98     int totalTableCount = 0; //总记录数共105条
99     totalTableCount = getTotalTableCount();
100    int totalPage = 0;
101    int mode = totalTableCount % pageSize;
102    if (mode == 0){//刚好除尽了
103        totalPage =
104            totalTableCount / pageSize;
105    }
106    else{//没有除尽,加1页
107        totalPage =
108            totalTableCount / pageSize + 1;
109    }
110    return totalPage;
111 }
112
113 /**
114  * 获得表的总记录数TotalTableCount
115  * @return 表的总记录数
116  */
117 private static int getTotalTableCount() {
118     int count = 0;
119     String sql =
120         "select count(*) num from mytemp_xxx";
121     Connection conn =
122         ConnectionUtils.getConnection();
123     Statement stmt = null;
124     ResultSet rs = null;
125

```

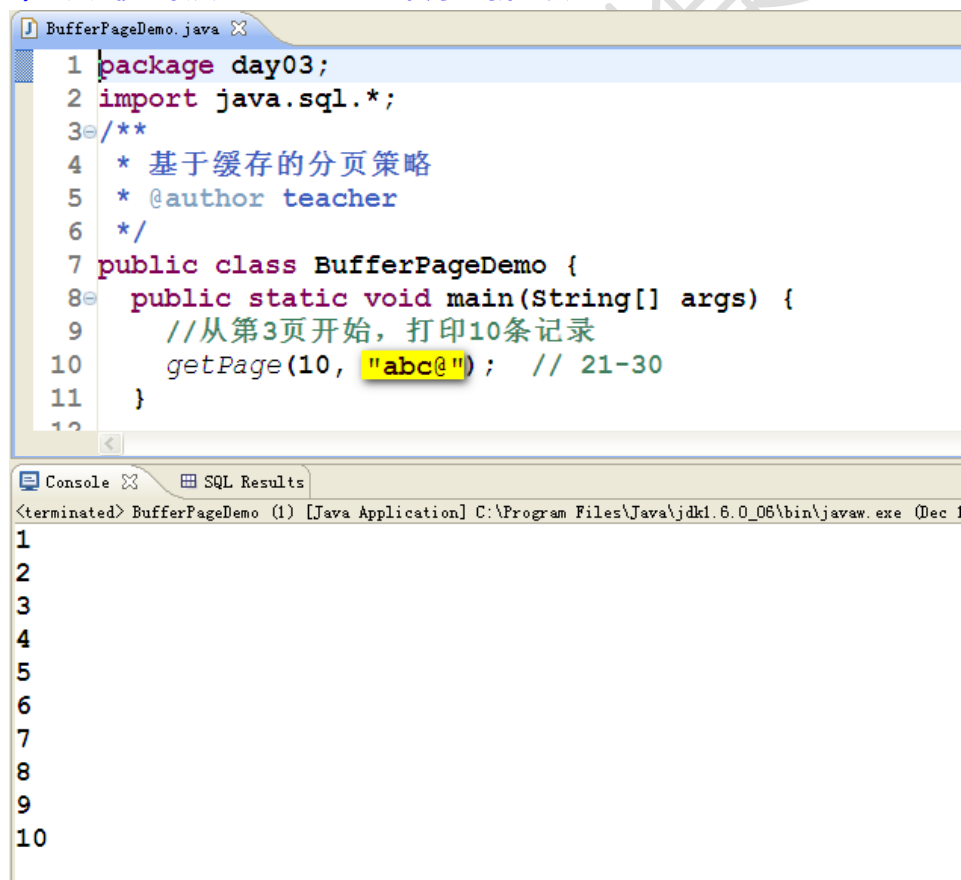
```

126     try{
127         stmt = conn.createStatement();
128         rs = stmt.executeQuery(sql);
129         rs.next();
130         count = rs.getInt("num");//取结果集的第1列
131
132     }catch(Exception e){
133         e.printStackTrace();
134     }finally{
135         ConnectionUtils.close(stmt);
136         ConnectionUtils.close(conn);
137     }
138     return count;
139 }

```

● 结果演示

1) 用户输入不规范字符“abc@”，则跳到第一页



```

BufferPageDemo.java
1 package day03;
2 import java.sql.*;
3 /**
4  * 基于缓存的分页策略
5  * @author teacher
6  */
7 public class BufferPageDemo {
8     public static void main(String[] args) {
9         //从第3页开始，打印10条记录
10        getPage(10, "abc@"); // 21-30
11    }
12

```

```

Console  SQL Results
<terminated> BufferPageDemo (1) [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 1
1
2
3
4
5
6
7
8
9
10

```

2) 用户输入页数超过最大页数，跳转到最后一页

```

BufferPageDemo.java
1 package day03;
2 import java.sql.*;
3 /**
4  * 基于缓存的分页策略
5  * @author teacher
6  */
7 public class BufferPageDemo {
8     public static void main(String[] args) {
9
10         getPage(10, "666");
11     }
12

```

Console SQL Results

```

<terminated> BufferPageDemo (1) [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 1
101
102
103
104
105

```

3) 用户输入负数，则跳转到第一页

```

BufferPageDemo.java
1 package day03;
2 import java.sql.*;
3 /**
4  * 基于缓存的分页策略
5  * @author teacher
6  */
7 public class BufferPageDemo {
8     public static void main(String[] args) {
9
10         getPage(10, "-2");
11     }
12

```



```

Console  SQL Results
<terminated> BufferPageDemo (1) [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 1
1
2
3
4
5
6
7
8
9
10

```

【思考题】如果数据表没有记录，如何避免程序出错？

3. 基于查询的分页 **

3.1. 基于查询的分页演示

基于查询的分页方式，获得某页数据的算法是和基于缓存的方式是一样的：

1 页： 1-10

2 页： 11-20

...

n 页： [起点] $x = (n - 1) * pageSize + 1$ -- [终点] $y = 起点 + pageSize - 1$

【案例 2】基于查询的分页演示

- 问题提出

如果要查询的是第 n 页，如何只把数据表中从[起点]到[终点]之间的数据返回？

即数据表中取起点(x)至终点(y)共(x-y)条数据的方式？

- 方案 1：rownum(直接使用不可行)

rownum 的特性：必须从第 1 行开始获取数据，不能从中间“截取”数据

演示如下：

```
Telnet 192.168.0.26
SQL> select rownum, id from mytemp_ning
2  where rownum < 30
3  and rownum > 20;

no rows selected

SQL> select * from mytemp_ning where rownum > 2;

no rows selected

SQL> █
```

- **方案 2：取差集(不可行)**

如果表中数据量不大是可以的 ;如果表中数据量过大 , 差集效率会很低(比如我们从 100 万条数据中取其中 10 条)

```
SQL> select id from mytemp_ning where rownum < 30
2  minus
3  select id from mytemp_ning where rownum < 20;

      ID
-----
      20
      21
      22
      23
      24
      25
      26
      27
```

- **方案 3：匿名视图(可行)**

```
SQL> select * from ( select id , rownum rn from mytemp_xxx )
      where rn between 21 and 30 ;

-- 功能上等价于
SQL> create view myview
as
```

```
select id , rownum rn from mytemp_xxx ;
SQL> select * from myview ;
```

参考代码

```
SelectPageDemo.java
1 package day03;
2 import day02.ConnectionUtils;
3
4 /**
5  * 基于查询的分页策略
6  * @author teacher
7  */
8 public class SelectPageDemo {
9     public static void main(String[] args) {
10         getPage(10,8); //71-80
11     }
12
13     /**
14      * 根据每页记录数和要查看的页数，显示数据
15      * @param pageSize 每页多少条
16      * @param page 要看的是第几页
17      */
18     public static void getPage(int pageSize, int page){
19         int begin = (page - 1) * pageSize + 1;
20         int end = begin + pageSize - 1;
21         String sql = "select id from (" +
22             "select id, rownum rn from mytemp_xxx)" +
23             " where rn between ? and ?";
24         Connection conn = ConnectionUtils.getConnection();
25         PreparedStatement stmt = null;
26         ResultSet rs = null;
27
28         try{
29             stmt = conn.prepareStatement(sql);
30             stmt.setInt(1, begin);
31             stmt.setInt(2, end);
32             rs = stmt.executeQuery();
33             while(rs.next()){
34                 System.out.println(rs.getInt("id"));
35             }
36         }catch(Exception e){
37             e.printStackTrace();
38         }finally{
39             ConnectionUtils.close(rs);
40             ConnectionUtils.close(stmt);
41             ConnectionUtils.close(conn);
42         }
43     }
44 }
```

```

Console  SQL Results
<terminated> SelectPageDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 9:41:50)
71
72
73
74
75
76
77
78
79
80

```

3.2. Mysql 数据库中的分页

- 1) rownum 是 Oracle 独有的，其他数据库不能用
- 2) 不同的数据库取 x-y 条数据的方式不同
- 3) mysql 中的分页较简单，方法如下：

```

--从第 21 条开始，取 10 条
SQL> select * from mytemp_xxx limit 21, 10 ;

```

3.3. 两种分页策略总结

- 1) 基于缓存的分页策略
 - ✓ 技术解决核心： 获得数据库表全部数据，得到可滚动结果集，通过移动指针从结果集中取出部分数据
 - ✓ 适用于查询数据量小的表
- 2) 基于查询的分页策略
 - ✓ 技术解决核心： 直接从数据表中取出部分数据(x 到 y 条的数据)
 - ✓ 适用于任何数据量的表

4. 实现排序后再分页 **

● 提出疑问

如果数据表中的数据是乱序的，如何实现排序后再分页？

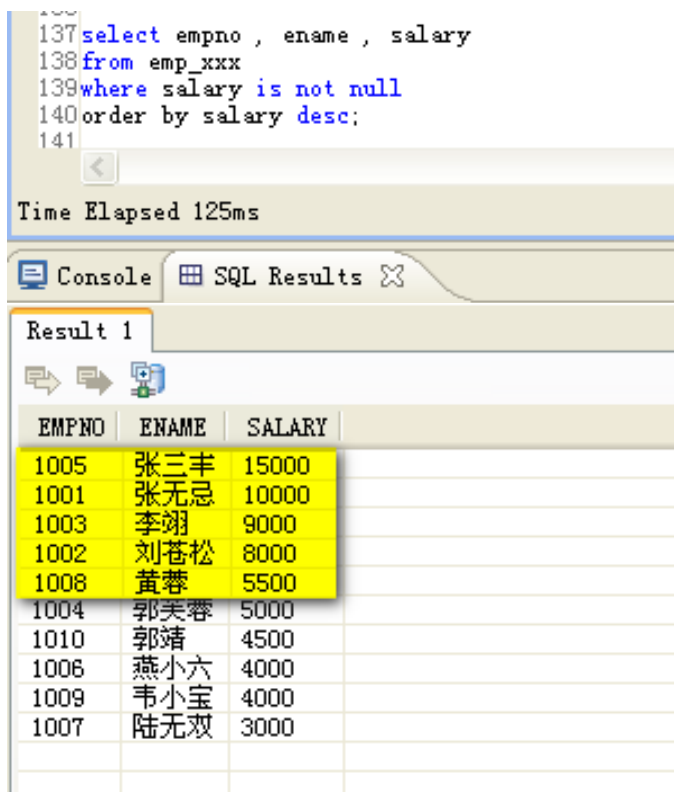
4.1. Rownum 伪列的特点

【案例 3】薪水最高的 5 个人是谁？

- Sql 语句演示

1) 按薪水降序排序

```
SQL> select empno, ename, salary
      from emp_xxx
      where salary is not null
      order by salary desc;
```



Time Elapsed 125ms

Console SQL Results

Result 1

EMPNO	ENAME	SALARY
1005	张三丰	15000
1001	张无忌	10000
1003	李翎	9000
1002	刘苍松	8000
1008	黄蓉	5500
1004	郭芙蓉	5000
1010	郭靖	4500
1006	燕小六	4000
1009	韦小宝	4000
1007	陆无双	3000

2) 如果直接使用 rownum, 则先取前 5 条, 再把这 5 条按薪水倒序排列 (得不到想要的结果)

```
-- 先取出了前五个人( rownum=1 至 rownum=5 ), 再按薪水排序
-- 无法达到效果
SQL> select rownum, empno, ename, salary
      from emp_xxx
      where salary is not null and rownum<6
      order by salary desc;
```

```

142 select rownum , empno , ename , salary
143 from emp_xxx
144 where salary is not null and rownum<6
145 order by salary desc;

```

Time Elapsed 110ms

Console SQL Results

Result 1

ROWNUM	EMPNO	ENAME	SALARY
5	1005	张三丰	15000
1	1001	张无忌	10000
3	1003	李翔	9000
2	1002	刘苍松	8000
4	1004	郭芙蓉	5000

3) 每条数据有固定的 rownum

-- 每条数据有固定的 rownum，是每条数据在表中的顺序标号

```

SQL> select rownum , empno , ename , salary
      from emp_xxx
      where salary is not null ;

```

```

136
137 select rownum , empno , ename , salary
138 from emp_xxx
139 where salary is not null;
140

```

Time Elapsed 141ms

Console SQL Results

Result 1

ROWNUM	EMPNO	ENAME	SALARY
1	1001	张无忌	10000
2	1002	刘苍松	8000
3	1003	李翊	9000
4	1004	郭芙蓉	5000
5	1005	张三丰	15000
6	1006	燕小六	4000
7	1007	陆无双	3000
8	1008	黄蓉	5500
9	1009	韦小宝	4000
10	1010	郭靖	4500

-- rownum 作为每条数据的一部分，每条数据有固定的 rownum

```
SQL> select rownum, empno, ename, salary
      from emp_xxx
      where salary is not null
      order by salary desc;
```

```
142 select rownum, empno, ename, salary
143 from emp_xxx
144 where salary is not null
145 order by salary desc;
146
```

Time Elapsed 141ms

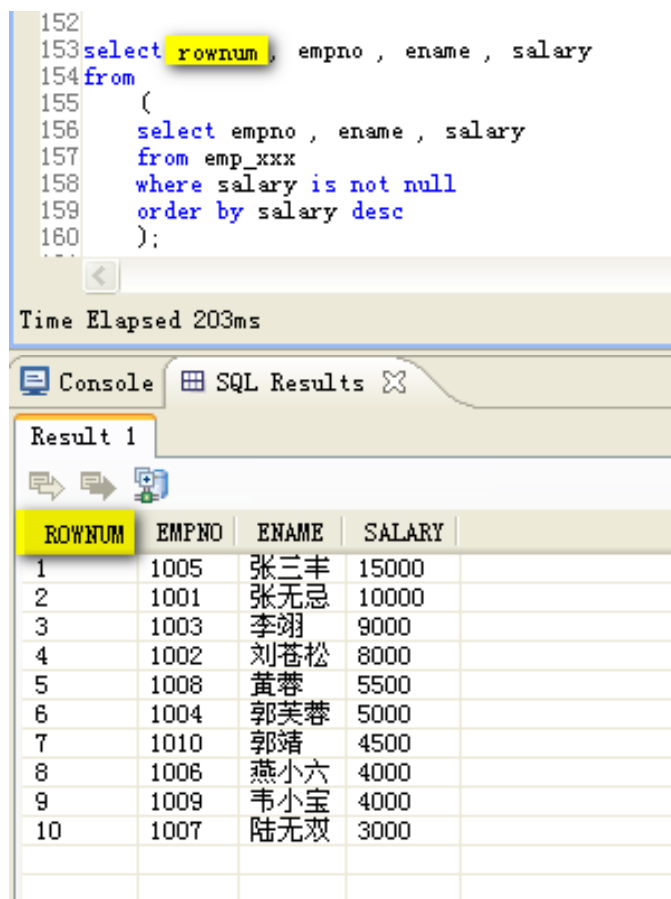
Console SQL Results

Result 1

ROWNUM	EMPNO	ENAME	SALARY
5	1005	张三丰	15000
1	1001	张无忌	10000
3	1003	李翊	9000
2	1002	刘苍松	8000
8	1008	黄蓉	5500
4	1004	郭芙蓉	5000
10	1010	郭靖	4500
6	1006	燕小六	4000
9	1009	韦小宝	4000
7	1007	陆无双	3000

4) 匿名视图(行内视图)的方式, 按薪水降序排序

```
-- 匿名视图( 行内视图 )的方式
-- 在 from 后面出现的子查询语句
SQL> select rownum , empno , ename , salary
      from
      (
        select empno , ename , salary
        from emp_xxx
        where salary is not null
        order by salary desc
      );
```



The screenshot shows a SQL IDE interface. The top pane displays the following SQL query:

```
152
153 select rownum , empno , ename , salary
154 from
155 (
156   select empno , ename , salary
157   from emp_xxx
158   where salary is not null
159   order by salary desc
160 );
```

Below the query editor, the status bar indicates "Time Elapsed 203ms". The bottom pane is split into "Console" and "SQL Results" tabs. The "SQL Results" tab is active, showing "Result 1" with a table of 10 rows. The table has columns ROWNUM, EMPNO, ENAME, and SALARY. The data is sorted by salary in descending order.

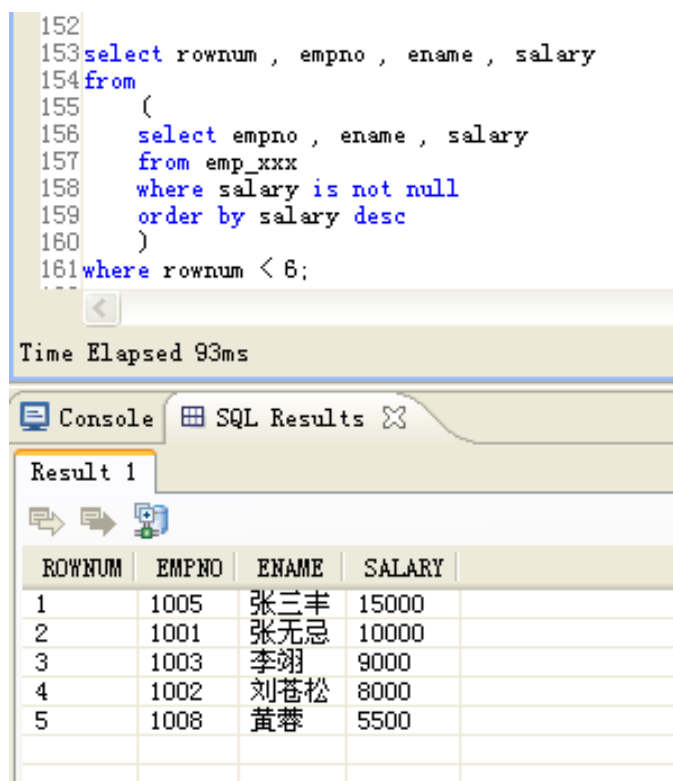
ROWNUM	EMPNO	ENAME	SALARY
1	1005	张三丰	15000
2	1001	张无忌	10000
3	1003	李翔	9000
4	1002	刘苍松	8000
5	1008	黄蓉	5500
6	1004	郭芙蓉	5000
7	1010	郭靖	4500
8	1006	燕小六	4000
9	1009	韦小宝	4000
10	1007	陆无双	3000

5) 查询出薪水最高的前五位(正确写法, 但只能查第 1 页)

```
-- 匿名视图( 行内视图 )的方式
-- 在 from 后面出现的子查询语句
```


-- 薪水由高到低排序，前 5 个员工，只是第一页

```
SQL> select rownum , empno , ename , salary
      from
        (
          select empno , ename , salary
          from emp_xxx
          where salary is not null
          order by salary desc
        )
      where rownum < 6 ;
```



The screenshot shows a SQL IDE interface. The top pane displays the following SQL query:

```
152
153 select rownum , empno , ename , salary
154 from
155 (
156   select empno , ename , salary
157   from emp_xxx
158   where salary is not null
159   order by salary desc
160 )
161 where rownum < 6;
```

Below the query editor, it indicates "Time Elapsed 93ms". The bottom pane is titled "SQL Results" and shows "Result 1" with a table of data:

ROWNUM	EMPNO	ENAME	SALARY
1	1005	张三丰	15000
2	1001	张无忌	10000
3	1003	李翎	9000
4	1002	刘苍松	8000
5	1008	黄蓉	5500

- 参考代码

```

1  SelectPageDemo.java X  0 emp.sql
5  * 基于查询的分页策略
6  * @author teacher
7  */
8  public class SelectPageDemo {
9      public static void main(String[] args) {
10         //取出工资最高的前5位
11         getOrderPage();
12     }
13
14     /**
15      * 基于查询的分页策略，排序后再分页的实现方式
16      * @param pageSize 每页多少条
17      * @param page 查第几页
18      */
19     public static void getOrderPage(){
20         String sql = "select rownum , empno , ename , salary"+
21             " from " +
22             " ( " +
23             "     select empno , ename , salary" +
24             "     from emp_xxx " +
25             "     where salary is not null" +
26             "     order by salary desc" +
27             " )" +
28             " where rownum < 6";
29         Connection conn = ConnectionUtils.getConnection();
30         PreparedStatement stmt = null;
31         ResultSet rs = null;
32         try{
33             stmt = conn.prepareStatement(sql);
34             rs = stmt.executeQuery();
35             while(rs.next()){
36                 System.out.println(rs.getInt("empno") + ", "
37                     + rs.getString("ename") + ", "
38                     + rs.getDouble("salary"));
39             }
40         }catch(Exception e){
41             e.printStackTrace();
42         }finally{
43             ConnectionUtils.close(rs);
44             ConnectionUtils.close(stmt);
45             ConnectionUtils.close(conn);
46         }
47     }

```

Console X SQL Results

```

<terminated> SelectPageDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 3:34:24)
1005, 张三丰, 15000.0
1001, 张无忌, 10000.0
1003, 李翔, 9000.0
1002, 刘苍松, 8000.0
1008, 黄蓉, 5500.0

```

【案例 4】薪水降序排序，找出第 5-10 个员工

- Sql 语句演示

薪水由高到低排序，找出第 5-10 个员工(可查第 2 页)

```
SQL> select empno, ename, salary
      from
        (
          select rownum rn, empno, ename, salary
            from
              (
                select empno, ename, salary
                  from emp_xxx
                 where salary is not null
                 order by salary desc
              )
        )
      where rn between 6 and 10 ;
```

```
164
165 select empno, ename, salary
166      from
167        (
168          select rownum rn, empno, ename, salary
169            from
170              (
171                select empno, ename, salary
172                  from emp_xxx
173                 where salary is not null
174                 order by salary desc
175              )
176        )
177 where rn between 6 and 10;
178
```

Time Elapsed 172ms

Console SQL Results

Result 1

EMPNO	ENAME	SALARY
1004	郭芙蓉	5000
1010	郭靖	4500
1006	燕小六	4000
1009	韦小宝	4000
1007	陆无双	3000

● 参考代码

```
SelectPageDemo.java
5  * 基于查询的分页策略
6  * @author teacher
7  */
8  public class SelectPageDemo {
9      public static void main(String[] args) {
10         //getPage(5, 3);
11         //按工资降序排序, 取出第5-10条
12         getOrderPage(5, 2);
13     }
14
15     /**
16      * 基于查询的分页策略, 排序后再分页的实现方式
17      * @param pageSize 每页多少条
18      * @param page 查第几页
19      */
20     public static void getOrderPage(int pageSize, int page){
21         int begin = (page - 1) * pageSize + 1;
22         int end = begin + pageSize - 1;
23         String sql = "select empno, ename, salary" +
24             " from" +
25             " (select rownum rn, empno, ename, salary" +
26             " from" +
27             "     (select empno, ename, salary " +
28             "       from emp_xxx" +
29             "      where salary is not null" +
30             "     order by salary desc)" +
31             " )" +
32             " where rn between ? and ?";
33         Connection conn = ConnectionUtils.getConnection();
34         PreparedStatement stmt = null;
35         ResultSet rs = null;
36         try{
```

```

37      stmt = conn.prepareStatement(sql);
38      stmt.setInt(1, begin);
39      stmt.setInt(2, end);
40      rs = stmt.executeQuery();
41      while(rs.next()){
42          System.out.println(rs.getInt("empno") + ", "
43                          + rs.getString("ename") + ", "
44                          + rs.getDouble("salary"));
45      }
46      }catch(Exception e){
47          e.printStackTrace();
48      }finally{
49          ConnectionUtils.close(rs);
50          ConnectionUtils.close(stmt);
51          ConnectionUtils.close(conn);
52      }
53  }

```

Console SQL Results

<terminated> SelectPageDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 3:29:06)

```

1004, 郭芙蓉, 5000.0
1010, 郭靖, 4500.0
1006, 燕小六, 4000.0
1009, 韦小宝, 4000.0
1007, 陆无双, 3000.0

```

5. 存储过程 && 通过 JDBC API 调用存储过程 *

- 1) 存储过程(Stored Procedure,简称 SP, 也称为 Procedure, 过程)
存储过程(Stored Procedure)是在大型数据库系统中, 一组完成特定功能的 SQL 语句集, 经编译后存储在数据库中, 用户通过指定存储过程的名字及参数来执行它。
- 2) 函数(function)
 - ✓ to_char / nvl / coalesc 等都是 Oracle 数据库的内置函数
 - ✓ 自己写的函数和 Procedure 功能类似, 类似于 java 中的方法。
 - ✓ 过程和函数都是驻留在数据库中的程序块
 - ✓ 过程可以有返回值也可以没有, 函数必须有返回值
- 3) 数据库编程
 - ✓ 在数据库中编译好程序块, 实现特定的功能
- 4) 优势: 提高效率
 - ✓ 最大限度的减少 I/O
 - ✓ 已经编译过的过程, 减少编译时间
- 5) 每条 sql 语句对应一个执行计划(Explain Plan), 缓存中保持一定数量的 sql 语句解析结果和它的执行计划

【案例 5】实现 1 个过程

● 要求

- 1) 输入参数：用户 id / 用户 pwd
- 2) 输出参数：
 - ✓ 正确：则返回 1(flag = 1)
 - ✓ 有用户 id，密码错：则返回 0(flag = 0)
 - ✓ 没有这个用户：则返回-1(flag = -1)

● 数据准备

1) 创建 user_xxx 表

```
SQL> create table user_xxx(
      id          number(4),
      password    char(4),
      name        char(20),
      phone       char(20),
      email       varchar2(50)
    );

SQL> insert into user_xxx
      values( 1001, '1234', 'liucs', '13600000000', 'liucs@sina.com' );
```

2) 创建存储过程 checkUser_xxx

```
SQL> create or replace procedure checkUser_xxx(
      p_userid in char,      -- in 表示输入( 入口，默认值 )
      p_pwd in char,
      flag out number        -- out 表示输出( 出口 )
    )

  is                          -- 定义变量
      v_password char(4);

  begin                      -- begin 和 end 之间为程序体
      select password into v_password -- 过程中的 select 语句的格式
      from user_xxx
      where id = p_userid;      -- 当且仅当查询出来一条记录时，不会出异常
      if v_password = p_pwd then
          flag := 1;           -- " := " 表示赋值，" = " 表示比较
```

```

else
    flag := 0;
end if;
exception
    when others then
        flag := -1;
end ;
/

```

--当出现异常，程序会跳到这里执行

--flag 赋值

-- "/"表示执行创建过程的语句

提示 Warning: Procedure created with compilation errors. 表示过程有错

```

27
28 /

```

Warning: Procedure created with compilation errors.

SQL> _

调试错误的方法

```

SQL> show errors
或者
SQL> show error

```

```

Telnet 192.168.0.26
SQL> show error
Errors for PROCEDURE CHECKUSER:

LINE/COL ERROR
-----
16/3      PL/SQL: Statement ignored
16/19     PLS-00201: identifier 'PWD' must be declared
SQL>

```

提示 Procedure created. 表示创建成功

3) 在 sqlplus 中用匿名块测试过程

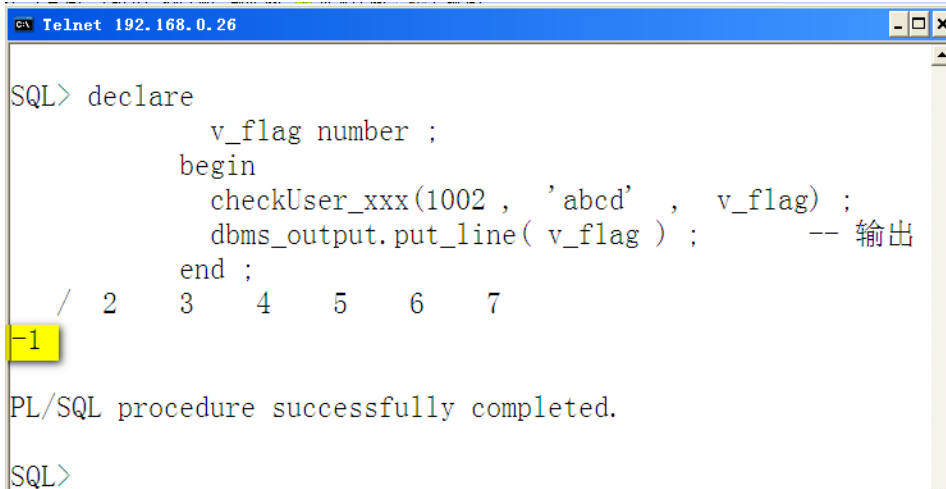
没有找到，返回-1

```

--下面这条语句是打开 sqlplus 工具的输出模式，默认是关闭的
SQL> set serveroutput on

```

```
SQL> declare
      v_flag number ;
    begin
      checkUser_xxx( 1002 , 'abcd' , v_flag );
      dbms_output.put_line( v_flag );      -- 输出功能默认是关闭的
    end ;
  /
```



```
SQL> declare
      v_flag number ;
    begin
      checkUser_xxx(1002 , 'abcd' , v_flag) ;
      dbms_output.put_line( v_flag );      -- 输出
    end ;
  /
  2      3      4      5      6      7
  1
PL/SQL procedure successfully completed.
SQL>
```

找到结果，返回 1

```
SQL> set serveroutput on
SQL> declare
      v_flag number ;
    begin
      checkUser_xxx( 1001 , '1234' , v_flag );
      dbms_output.put_line( v_flag );      -- 输出默认是关闭的
    end ;
  /
```



```

SQL> declare
        v_flag number ;
    begin
        checkUser_xxx(1001 , '1234' , v_flag) ;
        dbms_output.put_line( v_flag ) ;
    end ;
/ 2 3 4 5 6 7
1
PL/SQL procedure successfully completed.
SQL>

```

存储过程准备就绪

● 参考代码

```

ProcDemo.java
4=/**
5 * 测试使用JDBC API调用过程：checkUser
6 * @author teacher
7 */
8 public class ProcDemo {
9     public static void main(String[] args) {
10         int flag =
11             checkUser(1001, "1234");
12         if (flag == 1)
13             System.out.println("登录成功!");
14         else if (flag == 0)
15             System.out.println("密码错误!");
16         else if (flag == -1)
17             System.out.println("帐号不存在!");
18         else
19             System.out.println("其他错误");
20     }
21
22     public static int checkUser(int id, String pwd){
23         int flag = -2;
24
25         //前2个"?"代表in类型,通过参数列表传入
26         //第3个"?"代表out类型,是输出参数
27         String sql =
28             "{call checkUser_xxx(?,?,?)}"; //调用Procedure的语法

```

```

29 Connection conn = ConnectionUtils.getConnection();
30 CallableStatement stmt = null;
31
32 try{
33     stmt = conn.prepareCall(sql);
34     stmt.setInt(1, id);
35     stmt.setString(2, pwd);
36     stmt.registerOutParameter(3, Types.INTEGER);
37
38     stmt.execute();
39     flag = stmt.getInt(3); //取得Procedure的返回值
40 }catch(Exception e){
41     e.printStackTrace();
42 }finally{
43     ConnectionUtils.close(stmt);
44     ConnectionUtils.close(conn);
45 }
46 return flag;
47 }

```

Console SQL Results Oracle Source Errors
 <terminated> ProcDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 6:50:06 PM)
 登录成功!

测试 1

```

1 7 */
2 8 public class ProcDemo {
3 9     public static void main(String[] args) {
4 10         int flag =
5 11             checkUser(1002, "1234");
6 12         if (flag == 1)
7 13             System.out.println("登录成功!");
8 14         else if (flag == 0)
9 15             System.out.println("密码错误!");
10 16         else if (flag == -1)
11 17             System.out.println("帐号不存在!");
12 18         else
13 19             System.out.println("其他错误");
14 20     }
15 21

```

Console SQL Results Oracle Source Errors
 <terminated> ProcDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 6:53:49 PM)
 帐号不存在!

测试 2

```

ProcDemo.java X
7  */
8  public class ProcDemo {
9      public static void main(String[] args) {
10         int flag =
11             checkUser(1001, "12322");
12         if (flag == 1)
13             System.out.println("登录成功!");
14         else if (flag == 0)
15             System.out.println("密码错误!");
16         else if (flag == -1)
17             System.out.println("帐号不存在!");
18         else
19             System.out.println("其他错误");
20     }
21
Console X SQL Results X Oracle Source Errors
<terminated> ProcDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 6:55:56 PM)
密码错误!

```

【案例 6】创建 1 个过程，更新 1 条记录

- 要求

把指定员工的薪水更新为指定值

- 数据准备

1) 存储过程 update_emp

```

SQL> create or replace procedure update_emp(
    p_id in number ,
    p_new_salary in number ,
    flag out number
)
is
begin
    -- DML 语句和 sqlplus 中完全一样
    update emp_xxx set salary = p_new_salary
    where empno = p_id ;
    flag := SQL%ROWCOUNT ;    -- 返回上条 SQL 语句影响的记录数
    commit ;                    -- 注意: 和 Sqlplus 一样，需要自己 commit
end ;
/

-- SQL%ROWCOUNT 固定写法，存放执行的上条 Sql 语句所影响的记录数

```

-- Exception 可写可不写

```

运行 SQL 命令行
SQL> create or replace procedure update_emp(
2      p_id in number ,
3      p_new_salary in number ,
4      flag out number
5  )
6  is
7  begin
8      update emp_xxx set salary = p_new_salary
9      where empno = p_id ;
10     flag := SQL%ROWCOUNT ;
11     commit ;
12     end ;
13 /

过程已创建。

```

2) 在 sqlplus 中测试 update_emp

```

SQL> select empno , ename , salary from emp_xxx ;
SQL> declare                                -- 匿名块( 用于测试 )
      v_flag number ;
      begin
          update_emp( 1002 , 99.99 , v_flag ) ;
          dbms_output.put_line( v_flag ) ;
      end ;

```

```

运行 SQL 命令行
SQL> select empno , ename , salary from emp_xxx ;

  EMPNO ENAME          SALARY
-----
  1002  刘苍松          8000
  1003  李翊             9000
  1004  郭芙蓉             5000
  1005  张三丰            15000
  1006  燕小六             4000
  1007  陆无双             3000
  1008  黄蓉              5500
  1009  韦小宝             4000
  1010  郭靖              4500

已选择9行。

SQL> declare
2   v_flag number ;
3   begin
4   update_emp( 1002 , 99.99 , v_flag ) ;
5   dbms_output.put_line( v_flag ) ;
6   end ;
7   /

PL/SQL 过程已成功完成。

SQL> select empno , ename , salary from emp_xxx ;

  EMPNO ENAME          SALARY
-----
  1002  刘苍松          99.99
  1003  李翊             9000
  1004  郭芙蓉             5000
  1005  张三丰            15000
  1006  燕小六             4000

```

3) 在 java 程序中调用过程 update_emp

```

1 package day03;
2 import java.sql.*;
3
4 /**
5  * 测试使用JDBC API调用过程: checkUser
6  * @author teacher
7  */
8 public class ProcDemo {
9     public static void main(String[] args) {
10         int flag1 = updateEmp(1002, 1);
11         System.out.println(
12             flag1 == 1 ? "修改成功" : "修改失败");
13     }
14
15     public static int updateEmp(int id, double salary){
16         int flag = -2;
17         //调用Procedure的语法
18         String sql = "{call update_emp(?,?,?)}";
19         Connection conn = ConnectionUtils.getConnection();
20         CallableStatement stmt = null;
21         try{
22             stmt = conn.prepareCall(sql);
23             stmt.setInt(1, id);
24             stmt.setDouble(2, salary);
25             stmt.registerOutParameter(3, Types.INTEGER);
26             stmt.execute();
27             flag = stmt.getInt(3);
28         }catch(Exception e){
29             e.printStackTrace();
30         }finally{
31             ConnectionUtils.close(stmt);
32             ConnectionUtils.close(conn);
33         }
34         return flag;
35     }
36 }

```

Console SQL Results

<terminated> ProcDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 20, 2011 11:58:57 AM)

修改成功