数据库第16周作业

19336035 陈梓乐

1. **求各表空间的容量（注意一个表空间对应多个数据文件的情况），剩余空间和使用率，要求一条 SQL 语句完成**

```
select  a.tablespace_name,
        to_char(sum(a.bytes)/1024/1024, '999999.99')||'MB' Total,
        to_char(sum(b.bytes)/1024/1024, '999999.99')||'MB' Free,
        to_char((sum(a.bytes) - sum(b.bytes)) * 100 / sum(a.bytes), '99.99')||'%' rate
from            dba_data_files a
full outer join dba_free_space b
on              a.tablespace_name = b.tablespace_name
group by        a.tablespace_name;
```

```
SQL =  select  a.tablespace_name,
  2          to_char(sum(a.bytes)/1024/1024, '999999.99')||'MB' Total,
  3          to_char(sum(b.bytes)/1024/1024, '999999.99')||'MB' Free,
  4          to_char((sum(a.bytes) - sum(b.bytes)) * 100 / sum(a.bytes), '99.99')||'%' rate
  5  from            dba_data_files a
  6  full outer join dba_free_space b
  7  on              a.tablespace_name = b.tablespace_name
  8  group by        a.tablespace_name;

TABLESPACE_NAME                TOTAL        FREE         RATE
------------------------------ ------------ ------------ -------
SYSAUX                        44530.00MB      51.63MB  99.88%
UNDOTBS1                       6920.00MB     836.75MB  87.91%
USERS                           603.75MB      10.56MB  98.25%
SYSTEM                         1440.00MB       1.63MB  99.89%
EXAMPLE                         300.00MB      21.25MB  92.92%
TEST01                           10.00MB       8.00MB  20.00%

已选择6行。

SQL =
```

2. **查出 emp 表中每一行的 rowid，并查出它们分别在哪个数据文件，第几个 block，第几个slot（row number），属于哪个对象？提示：利用 dbms_rowid 包，用法可以参考课程资源中的 DSI401 第 4 章有关内容，或《PL/SQL packages》**

```
with tmp as
(
    select  rowid,
            dbms_rowid.rowid_relative_fno(rowid) relative_fno,
            dbms_rowid.rowid_block_number(rowid) block_number,
            dbms_rowid.rowid_row_number(rowid) row_number,
            dbms_rowid.rowid_object(rowid) obj
    from    scott.emp
)
select          rowid,
                substr(file_name, 1, 37) file_name,
                block_number,
                row_number,
                obj
from            tmp
natural join    dba_data_files;
```

```
SQL =  with tmp as
  2  (
  3      select  rowid,
  4              dbms_rowid.rowid_relative_fno(rowid) relative_fno,
  5              dbms_rowid.rowid_block_number(rowid) block_number,
  6              dbms_rowid.rowid_row_number(rowid) row_number,
  7              dbms_rowid.rowid_object(rowid) obj
  8      from    scott.emp
  9  )
 10  select          rowid,
 11                  substr(file_name, 1, 35) file_name,
 12                  block_number,
 13                  row_number,
 14                  obj
 15  from            tmp
 16  natural join    dba_data_files;

ROWID              FILE_NAME                                                                    BLOCK_NUMBER ROW_NUMBER        OBJ
------------------ -------------------------------------------------------------------------- ------------ ---------- ----------
AAAR3sAAEAAAACXAAA D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          0      73196
AAAR3sAAEAAAACXAAB D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          1      73196
AAAR3sAAEAAAACXAAC D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          2      73196
AAAR3sAAEAAAACXAAD D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          3      73196
AAAR3sAAEAAAACXAAE D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          4      73196
AAAR3sAAEAAAACXAAF D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          5      73196
AAAR3sAAEAAAACXAAG D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          6      73196
AAAR3sAAEAAAACXAAH D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          7      73196
AAAR3sAAEAAAACXAAI D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          8      73196
AAAR3sAAEAAAACXAAJ D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151          9      73196
AAAR3sAAEAAAACXAAK D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151         10      73196
AAAR3sAAEAAAACXAAL D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151         11      73196
AAAR3sAAEAAAACXAAM D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151         12      73196
AAAR3sAAEAAAACXAAN D:\APP\CZILE\ORADATA\ORCL\USERS01.D                                                 151         13      73196

已选择14行。
```

3. **分别创建本地管理和字典管理的表空间（给它适当的缺省 storage 参数），分别在上面建表，并且指出这些表的 storage 参数。通过 minextents 参数或 insert 行使这些表大小增长到一定规模。再通过观察 dba_extents 数据字典视图看下实际分配的 extent 情况，与你设计的storage 参数是否相符?**

本地管理

首先，我们创建一个本地管理的表空间，不妨名称设置为 `test01`：

```
create tablespace test01
    datafile 'D:/app/czile/oradata/orcl/test01.dbf' size 10M
    extent management local uniform size 256k;
```

接着，我们在表空间创建一个至少 1000kb 的表（因此 storage 参数当然是 `initial 1000k next 1000k`）：

```
create table test01t(x number)
    tablespace test01
    storage (initial 1000k next 1000k);
```

然后我们来看看表参数：

```
select  table_name,
        initial_extent,
        next_extent
from    user_tables
where   table_name = 'TEST01T';
```

```
SQL =  select  table_name,
  2          initial_extent,
  3          next_extent
  4  from    user_tables
  5  where   table_name = 'TEST01T';

TABLE_NAME                     INITIAL_EXTENT NEXT_EXTENT
------------------------------ -------------- -----------
TEST01T                               1024000     1024000

SQL =
```

接着，我们来查看对应的 extent：

```
select  tablespace_name
        extent_id,
        file_id,
        block_id,
        bytes
from    dba_extents
where   tablespace_name = 'TEST01';
```

```
SQL =  select  tablespace_name
  2          extent_id,
  3          file_id,
  4          block_id,
  5          bytes
  6  from    dba_extents
  7  where   tablespace_name = 'TEST01';

EXTENT_ID                      FILE_ID   BLOCK_ID      BYTES
------------------------------ ---------- ---------- ----------
TEST01                                6        128     262144
TEST01                                6        160     262144
TEST01                                6        192     262144
TEST01                                6        224     262144

SQL =
```

可以看到 oracle 创建了4个 `extent` ，每个 `extent` 大小是 `uniform size`，但为了储存 `test01t`，事实上，最后一个 `extent` 只需要 237568 个字节的空间即可，然而 oracle 的 `extent` 仍然以 `uniform size` 创建 `extent`。因此总结：本地管理的表空间的 `extent` 以 `uniform size` 扩展，直到第一次能放下全部数据。

字典管理

我们先来创建字典管理的表空间，不妨称为 `test02`：

```
create tablespace test02
    datafile 'D:/app/czile/oradata/orcl/test02.dbf' size 10M
    extent management dictionary
    default storage (initial 256k next 512k);
```

```
SQL =  create tablespace test02
  2      datafile 'D:/app/czile/oradata/orcl/test02.dbf' size 10M
  3      extent management dictionary
  4      default storage (initial 256k next 256k);
create tablespace test02
*
第 1 行出现错误:
ORA-12913: 无法创建字典管理的表空间
```

发现无法创建字典管理的表空间，查找相关 oracle 文档知要创建字典管理的表空间，需要 `system` 表空间是字典管理的，这需要在安装开始时选定，故无法对比。

> Ref:
> ORA-12913: 无法创建字典管理的表空间 https://blog.csdn.net/OsbornHuo/article/details/6340420
> oracle创建本地表空间,本地管理表空间——大家继续讨论！ https://blog.csdn.net/weixin_35706255/article/details/116342171

4. **参考课程资源中 DSI402e 中第一章内容，分别 dump 出 emp，指出它们是怎么 encode的，比如 ename 和 dump(ename)、sal 和 dump(sal)、hiredate 和 dump(hiredate)分别是怎样 encode 的?**

```
select  ename, dump(ename),
        sal, dump(sal),
        hiredate, dump(hiredate)
from    scott.emp
where   rownum <= 3;
```

```
ENAME
----------
DUMP(ENAME)
--------------------------------------------------------------------------------
       SAL
----------
DUMP(SAL)
--------------------------------------------------------------------------------
HIREDATE
-------------
DUMP(HIREDATE)
--------------------------------------------------------------------------------
SMITH
Typ=1 Len=5: 83,77,73,84,72
      1020
Typ=2 Len=3: 194,11,21
17-12月 -80
Typ=12 Len=7: 119,180,12,17,1,1,1
```

可以看到，oracle 的数据 encode 格式为：数据类型-数据长度-数据编码，其中：

- 字符串类型
  类型为1， 长度为实际长度，编码方式为 ASCII 编码，为了验证该编码格式，我们使用 python 从编码逆推原文：

```
Czile  D: > Work > 数据库  (latest)  16:00:54  Using 21.876s
 pwsh > python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> code = [83,77,73,84,72]
>>> list(map(chr, code))
['S', 'M', 'I', 'T', 'H']
>>>
```

- 数字类型
  类型为2，一般情况下，若只考虑正数，则储存格式为：

位置	类型	数值	备注
1	int	0b11000011	前两位储存正负号，正数是11，负数是00，0是10， 后六位是接下来有多少个数用来表示小数点之前的数值
2	int	??	数值，把原数字拆成两位两位储存，如果是正数，则该数值是`原数字`+1， 否则是100-`原数字`+1，若接下来都是0，则缺省该数值
3	int	??	数值
...			
n	int	102	若是负数，则存在此结束标志

例如图中的1020，对应 194, 11, 21，对应解释为：

数值	二进制	解释
194	0b11000010	前两位是11，代表正数，后六位是10，代表接下来将有2个数字去描述小数点前的位数
11	从高位起第一个两位数是10 = 11 - 1	
21	从高位起第二个两位数是20 = 21 - 1	

- 日期类型
  对应类型为12，长度为7，分别储存世纪、年月日时分秒，其中为了避免负数的产生，世纪和年的储存具体数值为100+真实数值。