oracle常用经典SQL查询(转贴)

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

oracle常用经典SQL查询
常用SQL查询:

1、查看表空间的名称及大小

```
select t.tablespace_name, round(sum(bytes/(1024*1024)),0) ts_size
from dba_tablespaces t, dba_data_files d
where t.tablespace_name = d.tablespace_name
group by t.tablespace_name;
```

2、查看表空间物理文件的名称及大小

```
select tablespace_name, file_id, file_name,
round(bytes/(1024*1024),0) total_space
from dba_data_files
order by tablespace_name;
```

3、查看回滚段名称及大小

```
select segment_name, tablespace_name, r.status,
(initial_extent/1024) InitialExtent,(next_extent/1024) NextExtent,
max_extents, v.curext CurExtent
From dba_rollback_segs r, v$rollstat v
Where r.segment_id = v.usn(+)
order by segment_name;
```

4、查看控制文件

```
select name from v$controlfile;
```

5、查看日志文件

```
select member from v$logfile;
```

6、查看表空间的使用情况

```
select sum(bytes)/(1024*1024) as free_space,tablespace_name
from dba_free_space
group by tablespace_name;

SELECT A.TABLESPACE_NAME,A.BYTES TOTAL,B.BYTES USED, C.BYTES FREE,
(B.BYTES*100)/A.BYTES "% USED",(C.BYTES*100)/A.BYTES "% FREE"
FROM SYS.***$TS_***AIL A,SYS.***$TS_USED B,SYS.***$TS_FREE C
WHERE A.TABLESPACE_NAME=B.TABLESPACE_NAME AND
A.TABLESPACE_NAME=C.TABLESPACE_NAME;
```

7、查看数据库库对象

```
select owner, object_type, status, count(*) count# from all_objects group by owner, object_type, status;
```

8、查看数据库的版本

```
Select version FROM Product_component_version
Where SUBSTR(PRODUCT,1,6)="Oracle";
```

9、查看数据库的创建日期和归档方式

```
Select Created, Log_Mode, Log_Mode From V$Database;
```

10、捕捉运行很久的SQL

```
column username format a12
column opname format a16
column progress format a8

select username,sid,opname,
    round(sofar*100 / totalwork,0) || "%" as progress,
    time_remaining,sql_text
from v$session_longops , v$sql
where time_remaining <> 0
and sql_address = address
and sql_hash_value = hash_value
/
```

11。查看数据表的参数信息

```
SELECT   partition_name, high_value, high_value_length, tablespace_name,
       pct_free, pct_used, ini_trans, max_trans, initial_extent,
       next_extent, min_extent, max_extent, pct_increase, FREELISTS,
       freelist_groups, LOGGING, BUFFER_POOL, num_rows, blocks,
       empty_blocks, ***g_space, chain_cnt, ***g_row_len, sample_size,
       last_analyzed
  FROM dba_tab_partitions
 --WHERE table_name = :tname AND table_owner = :towner
ORDER BY partition_position
```

12.查看还没提交的事务

```
select * from v$locked_object;
select * from v$transaction;
```

13。查找object为哪些进程所用

```
select
p.spid,
s.sid,
s.serial# serial_num,
s.username user_name,
a.type  object_type,
s.osuser os_user_name,
a.owner,
a.object object_name,
decode(sign(48 - command),
1,
to_char(command), "Action Code #" || to_char(command) ) action,
p.program oracle_process,
s.terminal terminal,
s.program program,
s.status session_status
from v$session s, v$access a, v$process p
```

```
where s.paddr = p.addr and
    s.type = "USER" and
    a.sid = s.sid   and
 a.object="SUBSCRIBER_ATTR"
order by s.username, s.osuser
```

14。回滚段查看

```
select rownum, sys.dba_rollback_segs.segment_name Name, v$rollstat.extents
Extents, v$rollstat.rssize Size_in_Bytes, v$rollstat.xacts XActs,
v$rollstat.gets Gets, v$rollstat.waits Waits, v$rollstat.writes Writes,
sys.dba_rollback_segs.status status from v$rollstat, sys.dba_rollback_segs,
v$rollname where v$rollname.name(+) = sys.dba_rollback_segs.segment_name and
v$rollstat.usn (+) = v$rollname.usn order by rownum
```

15。耗资源的进程（top session）

```
select s.schemaname schema_name,  decode(sign(48 - command), 1,
to_char(command), "Action Code #" || to_char(command) ) action,  status
session_status,  s.osuser os_user_name,  s.sid,  p.spid ,  s.serial# serial_num,
nvl(s.username, "[Oracle process]") user_name,  s.terminal terminal,
s.program program,  st.value criteria_value  from v$sesstat st,  v$session s  , v$process p
where st.sid = s.sid and  st.statistic# = to_number("38") and  ("ALL" = "ALL"
or s.status = "ALL") and p.addr = s.paddr order by st.value desc,  p.spid asc, s.username asc, s.osuser asc
```

16。查看锁（lock）情况

```
select /*+ RULE */ ls.osuser os_user_name,  ls.username user_name,
decode(ls.type, "RW", "Row wait enqueue lock", "TM", "DML enqueue lock", "TX",
"Transaction enqueue lock", "UL", "User supplied lock") lock_type,
o.object_name object,  decode(ls.lmode, 1, null, 2, "Row Share", 3,
"Row Exclusive", 4, "Share", 5, "Share Row Exclusive", 6, "Exclusive", null)
lock_mode,  o.owner,  ls.sid,  ls.serial# serial_num,  ls.id1,  ls.id2
from sys.dba_objects o, (  select s.osuser,  s.username,  l.type,
l.lmode,  s.sid,  s.serial#,  l.id1,  l.id2  from v$session s,
v$lock l  where s.sid = l.sid ) ls  where o.object_id = ls.id1 and  o.owner
<> "SYS"  order by o.owner, o.object_name
```

17。查看等待（wait）情况

```
SELECT v$waitstat.class, v$waitstat.count count, SUM(v$sysstat.value) sum_value
FROM v$waitstat, v$sysstat WHERE v$sysstat.name IN ("db block gets",
"consistent gets") group by v$waitstat.class, v$waitstat.count
```

18。查看sga情况

```
SELECT NAME, BYTES FROM SYS.V_$SGASTAT ORDER BY NAME ASC
```

19。查看catched object

```
SELECT owner,  name,  db_link,  namespace,
    type,  sharable_mem,  loads,  executions,
   locks,  pins,  kept  FROM v$db_object_cache
```

20。查看V$SQLAREA

```
SELECT SQL_TEXT, SHARABLE_MEM, PERSISTENT_MEM, RUNTIME_MEM, SORTS,
VERSION_COUNT, LOADED_VERSIONS, OPEN_VERSIONS, USERS_OPENING, EXECUTIONS,
USERS_EXECUTING, LOADS, FIRST_LOAD_TIME, INVALIDATIONS, PARSE_CALLS, DISK_READS,
BUFFER_GETS, ROWS_PROCESSED FROM V$SQLAREA
```

21。查看object分类数量

```
select decode (o.type#,1,"INDEX" , 2,"TABLE" , 3 , "CLUSTER" , 4, "VIEW" , 5 ,
"SYNONYM" , 6 , "SEQUENCE" , "OTHER" ) object_type , count(*) quantity from
```

```
sys.obj$ o where o.type# > 1 group by decode (o.type#,1,"INDEX" , 2,"TABLE" , 3
, "CLUSTER" , 4, "VIEW" , 5 , "SYNONYM" , 6 , "SEQUENCE" , "OTHER" ) union select
"COLUMN" , count(*) from sys.col$ union select "DB LINK" , count(*) from
```

22。按用户查看object种类

```
select u.name schema,  sum(decode(o.type#, 1, 1, NULL)) indexes,
sum(decode(o.type#, 2, 1, NULL)) tables,  sum(decode(o.type#, 3, 1, NULL))
clusters,  sum(decode(o.type#, 4, 1, NULL)) views,  sum(decode(o.type#, 5, 1,
NULL)) synonyms,  sum(decode(o.type#, 6, 1, NULL)) sequences,
sum(decode(o.type#, 1, NULL, 2, NULL, 3, NULL, 4, NULL, 5, NULL, 6, NULL, 1))
others   from sys.obj$ o, sys.user$ u   where o.type# >= 1 and   u.user# =
o.owner# and   u.name <> "PUBLIC"   group by u.name   order by
sys.link$ union select "CONSTRAINT" , count(*) from sys.con$
```

23。有关connection的相关信息

1）查看有哪些用户连接

```
select s.osuser os_user_name,   decode(sign(48 - command), 1, to_char(command),
"Action Code #" || to_char(command) ) action,    p.program oracle_process,
status session_status,   s.terminal terminal,   s.program program,
s.username user_name,   s.fixed_table_sequence activity_meter,   "" query,
0 memory,   0 max_memory,   0 cpu_usage,   s.sid,  s.serial# serial_num
from v$session s,   v$process p   where s.paddr=p.addr and   s.type = "USER"
order by s.username, s.osuser
```

2）根据v.sid查看对应连接的资源占用等情况

```
select n.name,
 v.value,
 n.class,
 n.statistic#
from  v$statname n,
 v$sesstat v
where v.sid = 71 and
 v.statistic# = n.statistic#
order by n.class, n.statistic#
```

3）根据sid查看对应连接正在运行的sql

```
select /*+ PUSH_SUBQ */
 command_type,
 sql_text,
 sharable_mem,
 persistent_mem,
 runtime_mem,
 sorts,
 version_count,
 loaded_versions,
 open_versions,
 users_opening,
 executions,
 users_executing,
 loads,
 first_load_time,
 invalidations,
 parse_calls,
 disk_reads,
 buffer_gets,
 rows_processed,
 sysdate start_time,
 sysdate finish_time,
 ">" || address sql_address,
```

```
 "N" status
from v$sqlarea
where address = (select sql_address from v$session where sid = 71)
```

24．查询表空间使用情况

```
select a.tablespace_name "表空间名称",
100-round((nvl(b.bytes_free,0)/a.bytes_alloc)*100,2) "占用率(%)",
round(a.bytes_alloc/1024/1024,2) "容量(M)",
round(nvl(b.bytes_free,0)/1024/1024,2) "空闲(M)",
round((a.bytes_alloc-nvl(b.bytes_free,0))/1024/1024,2) "使用(M)",
Largest "最大扩展段(M)",
to_char(sysdate,"yyyy-mm-dd hh24:mi:ss") "采样时间"
from  (select f.tablespace_name,
  sum(f.bytes) bytes_alloc,
  sum(decode(f.autoextensible,"YES",f.maxbytes,"NO",f.bytes)) maxbytes
from dba_data_files f
group by tablespace_name) a,
(select  f.tablespace_name,
   sum(f.bytes) bytes_free
from dba_free_space f
group by tablespace_name) b,
(select round(max(ff.length)*16/1024,2) Largest,
  ts.name tablespace_name
from sys.fet$ ff, sys.file$ tf,sys.ts$ ts
where ts.ts#=ff.ts# and ff.file#=tf.relfile# and ts.ts#=tf.ts#
group by ts.name, tf.blocks) c
where a.tablespace_name = b.tablespace_name and a.tablespace_name = c.tablespace_name
```

25. 查询表空间的碎片程度

```
select tablespace_name,count(tablespace_name) from dba_free_space group by tablespace_name
h***ing count(tablespace_name)>10;

alter tablespace name coalesce;
alter table name deallocate unused;

create or replace view ts_blocks_v as
select tablespace_name,block_id,bytes,blocks,"free space" segment_name from dba_free_space
union all
select tablespace_name,block_id,bytes,blocks,segment_name from dba_extents;

select * from ts_blocks_v;

select tablespace_name,sum(bytes),max(bytes),count(block_id) from dba_free_space
group by tablespace_name;
```

26。查询有哪些数据库实例在运行

```
select inst_name from v$active_instances;
```

```
=========================================================
######### 创建数据库----look $ORACLE_HOME/rdbms/admin/buildall.sql #############

create database db01
maxlogfiles 10
maxdatafiles 1024
maxinstances 2
logfile
```

```
GROUP 1 ("/u01/oradata/db01/log_01_db01.rdo") SIZE 15M,
GROUP 2 ("/u01/oradata/db01/log_02_db01.rdo") SIZE 15M,
GROUP 3 ("/u01/oradata/db01/log_03_db01.rdo") SIZE 15M,
datafile "u01/oradata/db01/system_01_db01.dbf") SIZE 100M,
undo tablespace UNDO
datafile "/u01/oradata/db01/undo_01_db01.dbf" SIZE 40M
default temporary tablespace TEMP
tempfile "/u01/oradata/db01/temp_01_db01.dbf" SIZE 20M
extent management local uniform size 128k
character set AL32UTE8
national character set AL16UTF16
set time_zone="America/New_York";

############### 数据字典 ##########

set wrap off

select * from v$dba_users;

grant select on table_name to user/rule;

select * from user_tables;

select * from all_tables;

select * from dba_tables;

revoke dba from user_name;

shutdown immediate

startup nomount

select * from v$instance;

select * from v$sga;

select * from v$tablespace;

alter session set nls_language=american;

alter database mount;

select * from v$database;

alter database open;

desc dictionary

select * from dict;

desc v$fixed_table;

select * from v$fixed_table;

set oracle_sid=foxconn
```

```
select * from dba_objects;

set serveroutput on

execute dbms_output.put_line("sfasd");

############# 控制文件 ###########

select * from v$database;

select * from v$tablespace;

select * from v$logfile;

select * from v$log;

select * from v$backup;

/*备份用户表空间*/
alter tablespace users begin backup;

select * from v$archived_log;

select * from v$controlfile;

alter system set control_files="$ORACLE_HOME/oradata/u01/ctrl01.ctl",
"$ORACLE_HOME/oradata/u01/ctrl02.ctl" scope=spfile;

cp $ORACLE_HOME/oradata/u01/ctrl01.ctl $ORACLE_HOME/oradata/u01/ctrl02.ctl

startup pfile="../initSID.ora"

select * from v$parameter where name like "control%" ;

show parameter control;

select * from v$controlfile_record_section;

select * from v$tempfile;

/*备份控制文件*/
alter database backup controlfile to "../filepath/control.bak";

/*备份控制文件，并将二进制控制文件变为了asc 的文本文件*/
alter database backup controlfile to trace;

############### redo log ##############

archive log list;

alter system archive log start;--启动自动存档

alter system switch logfile;--强行进行一次日志switch

alter system checkpoint;--强制进行一次checkpoint

alter tablspace users begin backup;
```

```
alter tablespace offline;

/*checkpoint 同步频率参数FAST_START_MTTR_TARGET,同步频率越高，系统恢复所需时间越短*/
show parameter fast;

show parameter log_checkpoint;

/*加入一个日志组*/
alter database add logfile group 3 ('/$ORACLE_HOME/oracle/ora_log_file6.rdo" size 10M);

/*加入日志组的一个成员*/
alter database add logfile member "/$ORACLE_HOME/oracle/ora_log_file6.rdo" to group 3;

/*删除日志组:当前日志组不能删；活动的日志组不能删；非归档的日志组不能删*/
alter database drop logfile group 3;

/*删除日志组中的某个成员，但每个组的最后一个成员不能被删除*/
alter databse drop logfile member "$ORACLE_HOME/oracle/ora_log_file6.rdo";

/*清除在线日志*/
alter database clear logfile "$ORACLE_HOME/oracle/ora_log_file6.rdo";

alter database clear logfile group 3;

/*清除非归档日志*/
alter database clear unarchived logfile group 3;

/*重命名日志文件*/
alter database rename file "$ORACLE_HOME/oracle/ora_log_file6.rdo" to
"$ORACLE_HOME/oracle/ora_log_file6a.rdo";

show parameter db_create;

alter system set db_create_online_log_dest_1="path_name";

select * from v$log;

select * from v$logfile;

/*数据库归档模式到非归档模式的互换,要启动到mount状态下才能改变;startup mount;然后再打开数据库.*/
alter database noarchivelog/archivelog;

achive log start;---启动自动归档

alter system archive all;- - 手工归档所有日志文件

select * from v$archived_log;

show parameter log_archive;

###### 分析日志文件logmnr ##############

1) 在init.ora中set utl_file_dir 参数
2) 重新启动oracle
3) create 目录文件
desc dbms_logmnr_d;
```

dbms_logmnr_d.build;
4) 加入日志文件 add/remove log file
dhms_logmnr.add_logfile
dbms_logmnr.removefile
5) start logmnr
dbms_logmnr.start_logmnr
6) 分析出来的内容查询 v$logmnr_content --sqlredo/sqlundo

实践:

desc dbms_logmnr_d;

/*对数据表做一些操作，为恢复操作做准备*/
update 表 set qty=10 where stor_id=6380;

delete 表 where stor_id=7066;
/**********************************/
utl_file_dir的路径
execute dbms_logmnr_d.build("foxdict.ora","$ORACLE_HOME/oracle/admin/fox/cdump");

execute dbms_logmnr.add_logfile("$ORACLE_HOME/oracle/ora_log_file6.log",dbms_logmnr.newfile);

execute dbms_logmnr.start_logmnr(dictfilename=>"$ORACLE_HOME/oracle/admin/fox/cdump/foxdict.ora");

######### tablespace ##############

select * form v$tablespace;

select * from v$datafile;

/*表空间和数据文件的对应关系*/
select t1.name,t2.name from v$tablespace t1,v$datafile t2 where t1.ts#=t2.ts#;

alter tablespace users add datafile "path" size 10M;

select * from dba_rollback_segs;

/*限制用户在某表空间的使用限额*/
alter user user_name quota 10m on tablespace_name;

create tablespace xxx [datafile "path_name/datafile_name"] [size xxx] [extent management local/dictionary] [default storage(xxx)];

exmple: create tablespace userdata datafile "$ORACLE_HOME/oradata/userdata01.dbf" size 100M AUTOEXTEND ON NEXT 5M MAXSIZE 200M;
create tablespace userdata datafile "$ORACLE_HOME/oradata/userdata01.dbf" size 100M extent management dictionary default storage(initial 100k next 100k pctincrease 10) offline;
/*9i以后，oracle建议使用local管理，而不使用dictionary管理，因为local采用bitmap管理表空间 ，不会产生系统表空间的自愿争用;*/
create tablespace userdata datafile "$ORACLE_HOME/oradata/userdata01.dbf" size 100M extent management local uniform size 1m;
create tablespace userdata datafile "$ORACLE_HOME/oradata/userdata01.dbf" size 100M extent management local autoallocate;
/*在创建表空间时，设置表空间内的段空间管理模式，这里用的是自动管理*/
create tablespace userdata datafile "$ORACLE_HOME/oradata/userdata01.dbf" size 100M extent management local uniform size 1m segment space management auto;

```
alter tablespace userdata mininum extent 10;

alter tablespace userdata default storage(initial 1m next 1m pctincrease 20);

/*undo tablespace(不能被用在字典管理模下) */
create undo tablespace undo1 datafile "$ORACLE_HOME/oradata/undo101.dbf" size 40M extent management local;

show parameter undo;

/*temporary tablespace*/
create temporary tablespace userdata tempfile "$ORACLE_HOME/oradata/undo101.dbf" size 10m extent management
local;

/*设置数据库缺省的临时表空间*/
alter database default temporary tablespace tablespace_name;

/*系统/临时/在线的undo表空间不能被offline*/
alter tablespace tablespace_name offline/online;

alter tablespace tablespace_name read only;

/*重命名用户表空间*/
alter tablespace tablespace_name rename datafile "$ORACLE_HOME/oradata/undo101.dbf" to
"$ORACLE_HOME/oradata/undo102.dbf";

/*重命名系统表空间 ,但在重命名前必须将数据库shutdown,并重启到mount状态*/
alter database rename file "$ORACLE_HOME/oradata/system01.dbf" to "$ORACLE_HOME/oradata/system02.dbf";

drop tablespace userdata including contents and datafiles;---drop tablespce

/*resize tablespace,autoextend datafile space*/
alter database datafile "$ORACLE_HOME/oradata/undo102.dbf" autoextend on next 10m maxsize 500M;

/*resize datafile*/
alter database datafile "$ORACLE_HOME/oradata/undo102.dbf" resize 50m;

/*给表空间扩展空间*/
alter tablespace userdata add datafile "$ORACLE_HOME/oradata/undo102.dbf" size 10m;

/*将表空间设置成OMF状态*/
alter system set db_create_file_dest="$ORACLE_HOME/oradata";

create tablespace userdata;---use OMF status to create tablespace;

drop tablespace userdata;---user OMF status to drop tablespace;

select * from dba_tablespace/v$tablespace/dba_data_files;

/*将表的某分区移动到另一个表空间*/
alter table table_name move partition partition_name tablespace tablespace_name;

###### ORACLE storage structure and relationships #########

/*手工分配表空间段的分区(extend)大小*/
alter table kong.test12 allocate extent(size 1m datafile "$ORACLE_HOME/oradata/undo102.dbf");

alter table kong.test12 deallocate unused; ---释放表中没有用到的分区
```

```
show parameter db;

alter system set db_8k_cache_size=10m; ---配置8k块的内存空间块参数

select * from dba_extents/dba_segments/data_tablespace;

select * from dba_free_space/dba_data_file/data_tablespace;

/*数据对象所占用的字节数*/
select sum(bytes) from dba_extents where onwer="kong" and segment_name ="table_name";

############ UNDO Data ################

show parameter undo;

alter tablespace users offline normal;

alter tablespace users offline immediate;

recover datafile "$ORACLE_HOME/oradata/undo102.dbf";

alter tablespace users online ;

select * from dba_rollback_segs;

alter system set undo_tablespace=undotbs1;

/*忽略回滚段的错误提示*/
alter system set undo_suppress_errors=true;

/*在自动管理模式下,不会真正建立rbs1;在手工管理模式则可以建立,且是私有回滚段*/
create rollback segment rbs1 tablespace undotbs;

desc dbms_flashback;

/*在提交了修改的数据后,9i提供了旧数据的回闪操作,将修改前的数据只读给用户看,但这部分数据不会又恢复在表中,而是旧数据的一个映射*/
execute dbms_flashback.enable_at_time("26-JAN-04:12:17:00 pm");

execute dbms_flashback.disable;

/*回滚段的统计信息*/
select end_time,begin_time,undoblks from v$undostat;

/*undo表空间的大小计算公式: UndoSpace=[UR * (UPS * DBS)] + (DBS * 24)
UR :UNDO_RETENTION 保留的时间(秒)
UPS :每秒的回滚数据块
DBS:系统EXTENT和FILE SIZE(也就是db_block_size)*/

select * from dba_rollback_segs/v$rollname/v$rollstat/v$undostat/v$session/v$transaction;

show parameter transactions;

show parameter rollback;

/*在手工管理模式下,建立公共的回滚段*/
```

create public rollback segment prbs1 tablespace undotbs;

alter rollback segment rbs1 online;----在手工管理模式

/*在手工管理模式中,initSID.ora中指定 undo_management=manual 、rollback_segment=("rbs1","rbs2",...)、
transactions=100 、transactions_per_rollback_segment=10
然后 shutdown immediate ,startup pfile=....\???.ora */

########## Managing Tables ###########

/*char type maxlen=2000;varchar2 type maxlen=4000 bytes
rowid 是18位的64进制字符串 (10个bytes 80 bits)
rowid组成: object#(对象号)--32bits,6位
rfile#(相对文件号)--10bits,3位
block#(块号)--22bits,6位
row#(行号)--16bits,3位
64进制: A-Z,a-z,0-9,/,+ 共64个符号

dbms_rowid 包中的函数可以提供对rowid的解释*/

select rowid,dbms_rowid.rowid_block_number(rowid),dbms_rowid.rowid_row_number(rowid) from table_name;

create table test2
(
id int,
lname varchar2(20) not null,
fname varchar2(20) constraint ck_1 check(fname like "k%"),
empdate date default sysdate)
) tablespace tablespace_name;

create global temporary table test2 on commit delete/preserve rows as select * from kong.authors;

create table user.table(...) tablespace tablespace_name storage(...) pctfree10 pctused 40;

alter table user.tablename pctfree 20 pctused 50 storage(...);---changing table storage

/*手工分配分区,分配的数据文件必须是表所在表空间内的数据文件*/
alter table user.table_name allocate extent(size 500k datafile "...");

/*释放表中没有用到的空间*/
alter table table_name deallocate unused;

alter table table_name deallocate unused keep 8k;

/*将非分区表的表空间搬到新的表空间,在移动表空间后，原表中的索引对象将会不可用，必须重建*/
alter table user.table_name move tablespace new_tablespace_name;

create index index_name on user.table_name(column_name) tablespace users;

alter index index_name rebuild;

drop table table_name [CASCADE CONSTRAINTS];

alter table user.table_name drop column col_name [CASCADE CONSTRAINTS CHECKPOINT 1000];---drop column

/*给表中不用的列做标记*/

```
alter table user.table_name set unused column comments CASCADE CONSTRAINTS;

/*drop表中不用的做了标记列*/
alter table user.table_name drop unused columns checkpoint 1000;

/*当在drop col是出现异常，使用CONTINUE，防止重删前面的column*/
ALTER TABLE USER.TABLE_NAME DROP COLUMNS CONTINUE CHECKPOINT 1000;

select * from dba_tables/dba_objects;

######## managing indexes ##########

/*create index*/
example:
/*创建一般索引*/
create index index_name on table_name(column_name) tablespace tablespace_name;
/*创建位图索引*/
create bitmap index index_name on table_name(column_name1,column_name2) tablespace tablespace_name;
/*索引中不能用pctused*/
create [bitmap] index index_name on table_name(column_name) tablespace tablespace_name pctfree 20 storage(inital
100k next 100k) ;
/*大数据量的索引最好不要做日志*/
create [bitmap] index index_name table_name(column_name1,column_name2) tablespace_name pctfree 20 storage(inital
100k next 100k) nologging;
/*创建反转索引*/
create index index_name on table_name(column_name) reverse;
/*创建函数索引*/
create index index_name on table_name(function_name(column_name)) tablespace tablespace_name;
/*建表时创建约束条件*/
create table user.table_name(column_name number(7) constraint constraint_name primary key deferrable using index
storage(initial 100k next 100k) tablespace tablespace_name,column_name2 varchar2(25) constraint constraint_name not
null,column_name3 number(7)) tablespace tablespace_name;

/*给创建bitmap index分配的内存空间参数，以加速建索引*/
show parameter create_bit;

/*改变索引的存储参数*/
alter index index_name pctfree 30 storage(initial 200k next 200k);

/*给索引手工分配一个分区*/
alter index index_name allocate extent (size 200k datafile "$ORACLE/oradata/..");

/*释放索引中没用的空间*/
alter index index_name deallocate unused;

/*索引重建*/
alter index index_name rebuild tablespace tablespace_name;

/*普通索引和反转索引的互换*/
alter index index_name rebuild tablespace tablespace_name reverse;

/*重建索引时，不锁表*/
alter index index_name rebuild online;

/*给索引整理碎片*/
alter index index_name COALESCE;
```

/*分析索引,事实上是更新统计的过程*/
analyze index index_name validate structure;

desc index_state;

drop index index_name;

alter index index_name monitoring usage;-----监视索引是否被用到

alter index index_name nomonitoring usage;----取消监视

/*有关索引信息的视图*/
select * from dba_indexes/dba_ind_columns/dbs_ind_expressions/v$object_usage;

########## 数据完整性的管理(Maintaining data integrity) ##########

alter table table_name drop constraint constraint_name;----drop 约束

alter table table_name add constraint constraint_name primary key(column_name1,column_name2);-----创建主键

alter table table_name add constraint constraint_name unique(column_name1,column_name2);---创建唯一约束

/*创建外键约束*/
alter table table_name add constraint constraint_name foreign key(column_name1) references
table_name(column_name1);

/*不效验老数据，只约束新的数据[enable/disable：约束/不约束新数据;novalidate/validate:不对/对老数据进行验证]*/
alter table table_name add constraint constraint_name check(column_name like "B%") enable/disable
novalidate/validate;

/*修改约束条件，延时验证，commit时验证*/
alter table table_name modify constraint constraint_name initially deferred;

/*修改约束条件，立即验证*/
alter table table_name modify constraint constraint_name initially immediate;

alter session set constraints=deferred/immediate;

/*drop一个有外键的主键表,带cascade constraints参数级联删除*/
drop table table_name cascade constraints;

/*当truncate外键表时，先将外键设为无效，再truncate;*/
truncate table table_name;

/*设约束条件无效*/
alter table table_name disable constraint constraint_name;

alter table table_name enable novalidate constraint constraint_name;

/*将无效约束的数据行放入exception的表中，此表记录了违反数据约束的行的行号；在此之前，要先建exceptions表*/
alter table table_name add constraint constraint_name check(column_name >15) enable validate exceptions into
exceptions;

/*运行创建exceptions表的脚本*/
start $ORACLE_HOME/rdbms/admin/utlexcpt.sql;

/*获取约束条件信息的表或视图*/
select * from user_constraints/dba_constraints/dba_cons_columns;

################## managing password security and resources ###################

alter user user_name account unlock/open;----锁定/打开用户;

alter user user_name password expire;---设定口令到期

/*建立口令配置文件,failed_login_attempts口令输多少次后锁，password_lock_times指多少天后口令被自动解锁*/
create profile profile_name limit failed_login_attempts 3 password_lock_times 1/1440;
/*创建口令配置文件*/
create profile profile_name limit failed_login_attempts 3 password_lock_time unlimited password_life_time 30
password_reuse_time 30 password_verify_function verify_function password_grace_time 5;
/*建立资源配置文件*/
create profile prfile_name limit session_per_user 2 cpu_per_session 10000 idle_time 60 connect_time 480;

alter user user_name profile profile_name;

/*设置口令解锁时间*/
alter profile profile_name limit password_lock_time 1/24;

/*password_life_time指口令文件多少时间到期，password_grace_time指在第一次成功登录后到口令到期有多少天时间可改变口令*/
alter profile profile_name limit password_lift_time 2 password_grace_time 3;

/*password_reuse_time指口令在多少天内可被重用,password_reuse_max口令可被重用的最大次数*/
alter profile profile_name limit password_reuse_time 10[password_reuse_max 3];

alter user user_name identified by input_password;-----修改用户口令

drop profile profile_name;

/*建立了profile后，且指定给某个用户，则必须用CASCADE才能删除*/
drop profile profile_name CASCADE;

alter system set resource_limit=true;---启用自愿限制,缺省是false

/*配置资源参数*/
alter profile profile_name limit cpu_per_session 10000 connect_time 60 idle_time 5;
/*资源参数(session级)
cpu_per_session 每个session占用cpu的时间 单位1/100秒
sessions_per_user 允许每个用户的并行session数
connect_time 允许连接的时间 单位分钟
idle_time 连接被空闲多少时间后，被自动断开 单位分钟
logical_reads_per_session 读块数
***_sga 用户能够在SGA中使用的私有的空间数 单位bytes

(call级)
cpu_per_call 每次(1/100秒)调用cpu的时间
logical_reads_per_call 每次调用能够读的块数
*/

alter profile profile_name limit cpu_per_call 1000 logical_reads_per_call 10;

desc dbms_resouce_manager;---资源管理器包

/*获取资源信息的表或视图*/
select * from dba_users/dba_profiles;

Managing users

show parameter os;

create user testuser1 identified by kxf_001;

grant connect,createtable to testuser1;

alter user testuser1 quota 10m on tablespace_name;

/*创建用户*/
create user user_name identified by password default tablespace tablespace_name temporary tablespace tablespace_name quota 15m on tablespace_name password expire;

/*数据库级设定缺省临时表空间*/
alter database default temporary tablespace tablespace_name;

/*制定数据库级的缺省表空间*/
alter database default tablespace tablespace_name;

/*创建os级审核的用户，需知道os_authent_prefix，表示oracle和os口令对应的前缀,"OPS$"为此参数的值，此值可以任意设置*/
create user user_name identified by externally default OPS$tablespace_name tablespace_name temporary tablespace tablespace_name quota 15m on tablespace_name password expire;

/*修改用户使用表空间的限额,回滚表空间和临时表空间不允许授予限额*/
alter user user_name quota 5m on tablespace_name;

/*删除用户或删除级联用户(用户对象下有对象的要用CASCADE，将其下一些对象一起删除)*/
drop user user_name [CASCADE];

/*每个用户在哪些表空间下有些什么限额*/
desc dba_ts_quotas;select * from dba_ts_quotas where username="...";

/*改变用户的缺省表空间*/
alter user user_name default tablespace tablespace_name;

######### Managing Privileges #############

grant create table,create session to user_name;

grant create any table to user_name; revoke create any table from user_name;

/*授予权限语法,public 标识所有用户,with admin option允许能将权限授予第三者的权限*/
grant system_privs,[.....] to [user/role/public],[....] [with admin option];

select * from v$pwfile_users;

/*当 O7_dictionary_accessiblity参数为True时，标识select any table时，包括系统表也能select ,否则，不包含系统表;缺省为false*/
show parameter O7;

/*由于 O7_dictionary_accessiblity为静态参数，不能动态改变，故加scope=spfile,下次启动时才生效*/

```
alter system set O7_dictionary_accessiblity=true scope=spfile;

/*授予对象中的某些字段的权限，如select 某表中的某些字段的权限*/
grant [object_privs(column,....)],[...] on object_name to user/role/public,... with grant option;

/*oracle不允许授予select某列的权限,但可以授insert ,update某列的权限*/
grant insert(column_name1,column_name2,...) on table_name to user_name with grant option;

select * from dba_sys_privs/session_privs/dba_tab_privs/user_tab_privs/dba_col_privs/user_col_privs;

/*db/os/none 审计被记录在 数据库/操作系统/不审计 缺省是none*/
show parameter audit_trail;

/*启动对表的select动作*/
audit select on user.table_name by session;

/*by session在每个session中发出command只记录一次，by access则每个command都记录*/
audit [create table][select/update/insert on object by session/access][whenever successful/not successful];

desc dbms_fga;---进一步设计，则可使用dbms_fgs包

/*取消审计*/
noaudit select on user.table_name;

/*查被审计信息*/
select * from all_def_audit_opts/dba_stmt_audit_opts/dba_priv_audit_opts/dba_obj_audit_opts;

/*获取审计记录*/
select * from dba_audit_trail/dba_audit_exists/dba_audit_object/dba_audit_session/dba_audit_statement;

########### Managing Role #################

create role role_name; grant select on table_name to role_name; grant role_name to user_name; set role role_name;

create role role_name;
create role role_name identified by password;
create role role_name identified externally;

set role role_name ; ----激活role
set role role_name identified by password;

alter role role_name not identified;
alter role role_name identified by password;
alter role role_name identified externally;

grant priv_name to role_name [WITH ADMIN OPTION];
grant update(column_name1,col_name2,...) on table_name to role_name;
grant role_name1 to role_name2;

/*建立default role,用户登录时，缺省激活default role*/
alter user user_name default role role_name1,role_name2,...;
alter user user_name default role all;
alter user user_name default role all except role_name1,...;
alter user user_name default role none;

set role role1 [identified by password],role2,....;
set role all;
```

```
set role except role1,role2,...;
set role none;

revoke role_name from user_name;
revoke role_name from public;

drop role role_name;

select * from dba_roles/dba_role_privs/role_role_privs/dba_sys_privs/role_sys_privs/role_tab_privs/session_roles;

########## Basic SQL SELECT ################

select col_name as col_alias from table_name ;

select col_name from table_name where col1 like "_o%"; ----"_"匹配单个字符

/*使用字符函数(右边截取,字段中包含某个字符,左边填充某字符到固定位数,右边填充某字符到固定位数)*/
select substr(col1,-3,5),instr(col2,"g"),LPAD(col3,10,"$"),RPAD(col4,10,"%") from table_name;

/*使用数字函数(往右/左几位四舍五入,取整,取余)*/
select round(col1,-2),trunc(col2),mod(col3) from table_name ;

/*使用日期函数(计算两个日期间相差几个星期,两个日期间相隔几个月,在某个月份上加几个月,某个日期的下一个日期,
某日期所在月的最后的日期,对某个日期的月分四舍五入，对某个日期的月份进行取整)*/
select (sysdate-col1)/7
week,months_between(sysdate,col1),add_months(col1,2),next_day(sysdate,"FRIDAY"),last_day(sysdate),
round(sysdate,"MONTH"),trunc(sysdate,"MONTH") from table_name;

/*使用NULL函数(当expr1为空取expr2/当expr1为空取expr2,否则取expr3/当expr1=expr2返回空)*/
select nvl(expr1,expr2),nvl2(expr1,expr2,expr3),nullif(expr1,expr2) from table_name;

select column1,column2,column3, case column2 when "50" then column2*1.1
when "30" then column2*2.1
when "10" then column3/20
else column3
end as ttt
from table_name ; ------使用case函数

select table1.col1,table2.col2 from table1
[CROSS JOIN table2] | -----笛卡儿连接
[NATURAL JOIN table2] | -----用两个表中的同名列连接
[JOIN table2 USING (column_name)] | -----用两个表中的同名列中的某一列或几列连接
[JOIN table2
ON (table1.col1=table2.col2)] |
[LEFT|RIGHT|FULL OUTER JOIN table2 ------相当于(+)=,=(+)连接,全外连接
ON (table1.col1=table2.col2)]; ------SQL 1999中的JOIN语法;

example:
select col1,col2 from table1 t1
join table2 t2
on t1.col1=t2.col2 and t1.col3=t2.col1
join table3 t3
on t2.col1=t3.col3;

select * from table_name where col1 < any (select col2 from table_name2 where continue group by col3);
```

select * from table_name where col1 < all (select col2 from table_name2 where continue group by col3);

insert into (select col1,col2,col3 form table_name where col1> 50 with check option) values (value1,value2,value3);

MERGE INTO table_name table1
USING table_name2 table2
ON (table1.col1=table2.col2)
WHEN MATCHED THEN
UPDATE SET
table1.col1=table2.col2,
table1.col2=table2.col3,
...
WHEN NOT MATCHED THEN
INSERT VALUES(table2.col1,table2.col2,table2.col3,...); -----合并语句

###################### CREATE/ALTER TABLE ######################

alter table table_name drop column column_name ;---drop column

alter table table_name set unused (col1,col2,...);----设置列无效，这个比较快。
alter table table_name drop unused columns;---删除被设为无效的列

rename table_name1 to table_name2; ---重命名表

comment on table table_name is "comment message";----给表放入注释信息

create table table_name
(col1 int not null,col2 varchar2(20),col3 varchar2(20),
constraint uk_test2_1 unique(col2,col3))); -----定义表中的约束条件

alter table table_name add constraint pk_test2 primary key(col1,col2,...); ----创建主键

/*建立外键*/
create table table_name (rid int,name varchar2(20),constraint fk_test3 foreign key(rid) references other_table_name(id));

alter table table_name add constraint ck_test3 check(name like "K%");

alter table table_name drop constraint constraint_name;

alter table table_name drop primary key cascade;----级联删除主键

alter table table_name disable/enable constraint constraint_name;----使约束暂时无效

/*删除列，并级联删除此列下的约束条件*/
alter table table_name drop column column_name cascade constraint;

select * from user_constraints/user_cons_columns;---约束条件相关视图

############## Create Views ####################

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view_name [(alias[,alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint_name]]
[WITH READ ONLY [CONSTRAINT constraint_name]]; ------创建视图的语法

example: Create or replace view testview as select col1,col2,col3 from table_name; ------创建视图
/*使用别名*/

Create or replace view testview as select col1,sum(col2) col2_alias from table_name;
/*创建复杂视图*/
Create view view_name (alias1,alias2,alias3,alias4) as select d.col1,min(e.col1),max(e.col1),***g(e.col1) from table_name1 e,table_name2 d where e.col2=d.col2 group by d.col1;
/*当用update修改数据时，必须满足视图的col1>10的条件，不满足则不能被改变.*/
Create or replace view view_name as select * from table_name where col1>10 with check option;

/*改变视图的值.对于简单视图可以用update语法修改表数据，但复杂视图则不一定能改。如使用了函数，group by ,distinct等的列*/
update view_name set col1=value1;

/*TOP-N分析*/
select [column_list],rownum from (select [column_list] from table_name order by Top-N_column) where rownum<=N;

/*找出某列三条最大值的记录*/
example: select rownum as rank ,col1 ,col2 from (select col1 ,col2 from table_name order by col2 desc) where rownum<=3;

############ Other database Object ###############

CREATE SEQUENCE sequence_name [INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCEL | NOCYCLE}]
[{CACHE n | NOCACHE}]; -----创建SEQUENCE

example:
CREATE SEQUENCE sequence_name INCREMENT BY 10
START WITH 120
MAXVALUE 9999
NOCACHE
NOCYCLE;

select * from user_sequences ;---当前用户下记录sequence的视图

select sequence_name.nextval,sequence_name.currval from dual;-----sequence的引用

alter sequence sequence_name INCREMENT BY 20
MAXVALUE 999999
NOCACHE
NOCYCLE; -----修改sequence,不能改变起始序号

drop sequence sequence_name; ----删除sequence

CREATE [PUBLIC] SYNONYM synonym_name FOR object; ------创建同义词

DROP [PUBLIC] SYNONYM synonym_name;----删除同义词

CREATE PUBLIC DATABASE LINK link_name USEING OBJECT;----创建DBLINK

select * from object_name@link_name; ----访问远程数据库中的对象

/*union 操作，它将两个集合的交集部分压缩，并对数据排序*/
select col1,col2,col3 from table1_name union select col1,col2,col3 from table2_name;

/*union all 操作，两个集合的交集部分不压缩，且不对数据排序*/

```
select col1,col2,col3 from table1_name union all select col1,col2,col3 from table2_name;

/*intersect 操作，求两个集合的交集,它将对重复数据进行压缩，且排序*/
select col1,col2,col3 from table1_name intersect select col1,col2,col3 from table2_name;

/*minus 操作，集合减,它将压缩两个集合减后的重复记录,且对数据排序*/
select col1,col2,col3 from table1_name minus select col1,col2,col3 from table2_name;

/*EXTRACT 抽取时间函数.此例是抽取当前日期中的年*/
select EXTRACT(YEAR FROM SYSDATE) from dual;
/*EXTRACT 抽取时间函数.此例是抽取当前日期中的月*/
select EXTRACT(MONTH FROM SYSDATE) from dual;

######################### 增强的 group by 子句 ########################

select [column,] group_function(column)...
from table
[WHERE condition]
[GROUP BY [ROLLUP] group_by_expression]
[H***ING h***ing_expression];
[ORDER BY column]; -------ROLLUP操作字，对group by子句的各字段从右到左进行再聚合

example:
/*其结果看起来象对col1做小计*/
select col1,col2,sum(col3) from table group by rollup(col1,col2);
/*复合rollup表达式*/
select col1,col2,sum(col3) from table group by rollup((col1,col2));

select [column,] group_function(column)...
from table
[WHERE condition]
[GROUP BY [CUBE] group_by_expression]
[H***ING h***ing_expression];
[ORDER BY column]; -------CUBE操作字，除完成ROLLUP的功能外，再对ROLLUP后的结果集从右到左再聚合

example:
/*其结果看起来象对col1做小计后，再对col2做小计，最后算总计*/
select col1,col2,sum(col3) from table group by cube(col1,col2);
/*复合rollup表达式*/
select col1,col2,sum(col3) from table group by cube((col1,col2));
/*混合rollup,cube表达式*/
select col1,col2,col3,sum(col4) from table group by col1,rollup(col2),cube(col3);

/*GROUPING(expr)函数，查看select语句种以何字段聚合，其取值为0或1*/
select [column,] group_function(column)...,GROUPING(expr)
from table
[WHERE condition]
[GROUP BY [ROLLUP] group_by_expression]
[H***ING h***ing_expression];
[ORDER BY column];

example:
select col1,col2,sum(col3),grouping(col1),grouping(col2) from table group by cube(col1,col2);

/*grouping sets操作，对group by结果集先对col1求和，再对col2求和，最后将其结果集并在一起*/
select col1,col2,sum(col3) from table group by grouping sets((col1),(col2));
```

ORACLE的基本语法集锦----简单却是最基本的

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

```
-- 表
create table test (names varchar2(12),
                   dates date,
                   num   int,
                   dou   double);
-- 视图
create or replace view vi_test as
select * from test;

-- 同义词
create or replace synonym aa
for dbusrcard001.aa;

-- 存储过程
create or replace produce dd(v_id in employee.empoy_id%type)
as
begin

end
dd;

-- 函数
create or replace function ee(v_id in employee%rowtype) return varchar(15)
is
var_test varchar2(15);
begin
  return var_test;
exception when others then

end

-- 三种触发器的定义
create or replace trigger ff
alter delete
on test
for each row
declare
begin
   delete from test;
   if sql%rowcount < 0 or sql%rowcount is null then
      rais_replaction_err(-20004,"错误")
   end if
end
```

```
create or replace trigger gg
alter insert
on test
for each row
declare
begin
  if :old.names = :new.names then
    raise_replaction_err(-2003,"编码重复");
  end if
end


create or replace trigger hh
for update
on test
for each row
declare
begin
 if updating then
    if :old.names <> :new.names then
 reaise_replaction_err(-2002,"关键字不能修改")
    end if
 end if
end

-- 定义游标
declare
  cursor aa is
    select names,num from test;
begin
  for bb in aa
  loop
      if bb.names = "ORACLE" then

      end if
  end loop;

end

-- 速度优化，前一语句不后一语句的速度快几十倍
select names,dates
from test,b
where test.names = b.names(+) and
     b.names is null and
     b.dates > date("2003-01-01","yyyy-mm-dd")


select names,dates
from test
where names not in ( select names
                     from b
                    where dates > to_date("2003-01-01","yyyy-mm-dd"))


-- 查找重复记录
select names,num
from test
where rowid != (select max(rowid)
```

```
                from test b
               where b.names = test.names and
                   b.num = test.num)


-- 查找表TEST中时间最新的前10条记录
select * from (select * from test order by dates desc) where rownum < 11

-- 序列号的产生
create sequence row_id
minvalue 1
maxvalue 999999999999999999999
start with 1
increment by 1

insert into test values(row_id.nextval,....)
```

Oracle的优化器(Optimizer)

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

Oracle在执行一个SQL之前,首先要分析一下语句的执行计划,然后再按执行计划去执行。分析语句的执行计划的工作是由优化器(Optimizer)来完成的。不同的情况,一条SQL可能有多种执行计划,但在某一时点,一定只有一种执行计划是最优的,花费时间是最少的。

相信你一定会用Pl/sql Developer、Toad等工具去看一个语句的执行计划,不过你可能对Rule、Choose、First rows、All rows这几项有疑问,因为我当初也是这样的,那时我也疑惑为什么选了以上的不同的项,执行计划就变了?

1、优化器的优化方式

Oracle的优化器共有两种的优化方式,即基于规则的优化方式(Rule-Based Optimization,简称为RBO)和基于代价的优化方式(Cost-Based Optimization,简称为CBO)。

A、RBO方式：优化器在分析SQL语句时,所遵循的是Oracle内部预定的一些规则。比如我们常见的,当一个where子句中的一列有索引时去走索引。

B、CBO方式：依词义可知,它是看语句的代价(Cost)了,这里的代价主要指Cpu和内存。优化器在判断是否用这种方式时,主要参照的是表及索引的统计信息。统计信息给出表的大小 、有少行、每行的长度等信息。这些统计信息起初在库内是没有的,是你在做analyze后才出现的,很多的时侯过期统计信息会令优化器做出一个错误的执行计划,因些我们应及时更新这些信息。在Oracle8及以后的版本,Oracle列推荐用CBO的方式。

我们要明了,不一定走索引就是优的 ,比如一个表只有两行数据,一次IO就可以完成全表的检索,而此时走索引时则需要两次IO,这时对这个表做全表扫描(full table scan)是最好的。

新版本的oracle逐渐抛弃对Rule方式的支持,即使是Rule方式,最后sql执行效率的衡量标准都是,sql执行消耗了多少资源?对代价(COST)的优化方式,需要表,索引的统计信息,需要每天多表和索引进行定时的分析,但是统计信息也是历史的,有时候也不一定是最优的,统计信息等于就是一个人的经验,根据以前的经验来判断sql该怎么执行(得到优化的sql执行路径),所以具体优化执行的时候,先手工分析sql,看是用RBO方式消耗大,还是CBO消耗大;DBA的工作就是要根据当前oracle的运行日志,进行各种调整,使当前的oracle运行效率尽量达到最优.可以在运行期间,采用hint灵活地采用优化方式.

2、优化器的优化模式(Optermizer Mode)

优化模式包括Rule,Choose,First rows,All rows这四种方式,也就是我们以上所提及的。如下我解释一下：

Rule:不用多说,即走基于规则的方式。 (RBO优化方式)

Choolse:这是我们应观注的,默认的情况下Oracle用的便是这种方式。指的是当一个表或或索引有统计信息,则走CBO的方式,如果表或索引没统计信息,表又不是特别的小,而且相应的列有索引时,那么就走索引,走RBO的方式。
在缺省情况下，ORACLE采用CHOOSE优化器，为了避免那些不必要的全表扫描（full table scan），你必须尽量避免使用CHOOSE优化器，而直接采用基于规则或者基于成本的优化器。

First Rows:它与Choose方式是类似的,所不同的是当一个表有统计信息时,它将是以最快的方式返回查询的最先的几行,从总体上减少了响应时间。 (CBO优化方式,提供一个最快的反应时间,根据系统的需求,使用情况)

All Rows:也就是我们所说的Cost的方式,当一个表有统计信息时,它将以最快的方式返回表的所有的行,从总体上提高查询的吞吐量。没有统计信息则走基于规则的方式。 (CBO优化方式,提供最大的吞吐量,就是使执行总量达到最大)

First Rows和All Rows是有冲突的.如果想最快第返回给用户,就不可能传递更多的结果,这就是First Rows返回最先检索到的行(或记录);而All Rows是为了尽量将所有的结果返回给用户,由于量大,用户就不会很快得到返回结果.就象空车能跑得很快,重装车只能慢慢地跑;

3、如何设定选用哪种优化模式

a、Instance级别

我们可以通过在init<SID>.ora文件中设定OPTIMIZER_MODE=RULE、OPTIMIZER_MODE=CHOOSE、OPTIMIZER_MODE=FIRST_ROWS、OPTIMIZER_MODE=ALL_ROWS去选用3所提的四种方式,如果你没设定OPTIMIZER_MODE参数则默认用的是Choose这种方式。
init.ora和init<SID>.ora都在$ORACLE_HOME/dbs目录下,可以用find $ORACLE_HOME -name init*.ora查看该目录下的init文件.
init.ora是对全体实例有效的;init<SID>.ora只对指定的实例有效.

B、Sessions级别

通过SQL> ALTER SESSION SET OPTIMIZER_MODE=<Mode>;来设定。 将覆盖init.ora,init<sid>.ora设定的优化模式,也可以在sql语句中采用hint强制选定优化模式.如下:

C、语句级别

这些需要用到Hint,比如:
SQL> SELECT /*+ RULE */ a.userid,
2 b.name,
3 b.depart_name
4 FROM tf_f_yhda a,
5 tf_f_depart b
6 WHERE a.userid=b.userid;
在这儿采用hint,强制采用基于规则(rule)的优化模式;
hint语法,/*+开头,*/结尾,中间填写强制采用的优化模式.

4、为什么有时一个表的某个字段明明有索引,当观察一些语的执行计划确不走索引呢? 如何解决呢?

A、不走索引大体有以下几个原因
♀你在Instance级别所用的是all_rows的方式
♀你的表的统计信息(最可能的原因)
♀你的表很小,上文提到过的,Oracle的优化器认为不值得走索引。
B、解决方法
♀可以修改init<SID>.ora中的OPTIMIZER_MODE这个参数,把它改为Rule或Choose,重起数据库。也可以使用4中所提的Hint.
♀删除统计信息
SQL>analyze table table_name delete statistics;
♀表小不走索引是对的,不用调的。

5、其它相关

A、如何看一个表或索引是否是统计信息

SQL>SELECT * FROM user_tables
2 WHERE table_name=<table_name>
3 AND num_rows is not null;

SQL>SELECT * FROM user_indexes
2 WHERE table_name=<table_name>
3 AND num_rows is not null;

b、如果我们先用CBO的方式,我们应及时去更新表和索引的统计信息,以免生形不切合实的执行计划。
SQL> ANALYZE TABLE table_name COMPUTE STATISTICS;
SQL> ANALYZE INDEX index_name ESTIMATE STATISTICS;

oracle函数大全-数字处理函数

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

数字函数

函数接受NUMBER类型的参数并返回NUMBER类型的数值.超越函数和三角函数的返回值精确到36位.ACOS、ASIN、ATAN、ATAN2的结果精确到36位.

l ABS

语法: ABS(x)

功能: 得到x的绝对值.

使用位置: 过程性语言和SQL语句。

l ACOS

语法: ACOS(x)

功能: 返回x的反余弦值. x应该从0到1之间的数,结果在0到pi之间,以弧度为单位.

使用位置: 过程性语言和SQL语句。

l ASIN

语法: ASIN(x)

功能: 计算x的反正弦值. X的范围应该是- 1到1之间,返回的结果在- pi/2到pi/2之间,以弧度为单位.

使用位置: 过程性语言和SQL语句。

l ATAN

语法: ATAN(x)

功能: 计算x的反正切值.返回值在- pi/2到pi/2之间,单位是弧度.

使用位置: 过程性语言和SQL语句。

l ATAN2

语法: ATAN2(x,y)

功能: 计算x和y的反正切值.结果在负的pi/2到正的pi/2之间,单位是弧度.

使用位置: 过程性语言和SQL语句。

l CEIL

语法: CEIL(x)

功能: 计算大于或等于x的最小整数值.

使用位置: 过程性语言和SQL语句。

l COS

语法: COS(x)

功能: 返回x的余弦值. X的单位是弧度.

使用位置: 过程性语言和SQL语句。

l COSH

语法: COSH(x)

功能: 计算x的双曲余弦值.

l EXP

语法: EXP(x)

功能: 计算e的x次幂. e为自然对数,约等于2.71828.

使用位置: 过程性语言和SQL语句。

l FLOOR

语法: FLOOR(x)

功能: 返回小于等于x的最大整数值.

使用位置: 过程性语言和SQL语句。

l LN

语法: LN(x)

功能: 返回x的自然对数. x必须是正数,并且大于0

使用位置: 过程性语言和SQL语句。

l LOG

语法: LOG(x)

功能: 计算以x为底的y的对数.底必须大于0而且不等于1, y为任意正数.

使用位置: 过程性语言和SQL语句。

l MOD

语法: MOD(x,y)

功能: 返回x除以y的余数.如果y是0,则返回x

使用位置: 过程性语言和SQL语句。

l POWER

语法: POWER(x,y)

功能: 计算x的y次幂.

使用位置: 过程性语言和SQL语句。

l ROUND

语法: ROUND(x[,y])

功能: 计算保留到小数点右边y位的x值. y缺省设置为0,这会将x保留为最接近的整数.如果y小于0,保留到小数点左边相应的位. Y必须是整数.

使用位置: 过程性语言和SQL语句。

l SIGN

语法: SIGN(x)

功能: 获得x的符号位标志.如果x<0返回- 1.如果x=0返回0.如果x>0返回1.

使用位置: 过程性语言和SQL语句。

l SIN

语法:SIN(x)

功能:计算x的正弦值. X是一个以弧度表示的角度.

使用位置: 过程性语言和SQL语句。

l SINH

语法:SINH(x)

功能:返回x的双曲正弦值.

使用位置: 过程性语言和SQL语句。

l SQRT

语法: SQRT(x)

功能: 返回x的平方根. x必须是正数.

使用位置: 过程性语言和SQL语句。

l TAN

语法: TAN(x)

功能: 计算x的正切值, x是一个以弧度位单位的角度.

使用位置: 过程性语言和SQL语句。

TANH
语法: TANH(x)

功能: 计算x的双曲正切值.

使用位置: 过程性语言和SQL语句。

l TRUNC

语法: TRUNC(x[,y])

功能: 计算截尾到y位小数的x值. y缺省为0,结果变为一个整数值.如果y是一个负数,那么就截尾到小数点左边对应的位上.

使用位置: 过程性语言和SQL语句。

oracle函数大全-字符处理函数

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

1字符函数——返回字符值

这些函数全都接收的是字符族类型的参数(CHR除外)并且返回字符值.除了特别说明的之外,这些函数大部分返回VARCHAR2类型的数值.字符函 数的返回类型所受的限制和基本数据库类型所受的限制是相同的,比如:VARCHAR2数值被限制为2000字符(ORACLE 8中为4000字符),而CHAR数值被限制为255字符(在ORACLE8中是2000).当在过程性语句中使用时,它们可以被赋值给VARCHAR2 或者CHAR类型的PL/SQL变量.

l CHR

语法: chr(x)

功能：返回在数据库字符集中与X拥有等价数值的字符。CHR和ASCII是一对反函数。经过CHR转换后的字符再经过ASCII转换又得到了原来的字符。

使用位置：过程性语句和SQL语句。

l CONCAT

语法： CONCAT（string1,string2）

功能：返回string1，并且在后面连接string2。

使用位置：过程性语句和SQL语句。

l INITCAP

语法：INITCAP（string）

功能：返回字符串的每个单词的第一个字母大写而单词中的其他字母小写的string。单词是用.空格或给字母数字字符进行分隔。不是字母的字符不变动。

使用位置：过程性语句和SQL语句。

l LTRIM

语法：LTRIM（string1,string2）

功能：返回删除从左边算起出现在string2中的字符的string1。String2被缺省设置为单个的空格。数据库将扫描string1，从最左边开始。当遇到不在string2中的第一个字符，结果就被返回了。LTRIM的行为方式与RTRIM很相似。

使用位置：过程性语句和SQL语句。

l NLS_INITCAP

语法：NLS_INITCAP（string[,nlsparams]）

功能：返回字符串每个单词第一个字母大写而单词中的其他字母小写的string，nlsparams

指定了不同于该会话缺省值的不同排序序列。如果不指定参数，则功能和INITCAP相同。Nlsparams可以使用的形式是：

‘NLS_SORT=sort’

这里sort制订了一个语言排序序列。

使用位置：过程性语句和SQL语句。

l NLS_LOWER

语法：NLS_LOWER（string[,nlsparams]）

功能：返回字符串中的所有字母都是小写形式的string。不是字母的字符不变。

Nlsparams参数的形式与用途和NLS_INITCAP中的nlsparams参数是相同的。如果nlsparams没有被包含，那么NLS_LOWER所作的处理和LOWER相同。

使用位置；过程性语句和SQL语句。

l NLS_UPPER

语法：nls_upper（string[,nlsparams]）

功能：返回字符串中的所有字母都是大写的形式的string。不是字母的字符不变。nlsparams参数的形式与用途和NLS_INITCAP中的相同。如果没有设定参数，则NLS_UPPER功能和UPPER相同。

使用位置：过程性语句和SQL语句。

l REPLACE

语法：REPLACE（string，search_str[,replace_str]）

功能：把string中的所有的子字符串search_str用可选的replace_str替换，如果没有指定replace_str，所有的string中的子字符串search_str都将被删除。REPLACE是TRANSLATE所提供的功能的一个子集。

使用位置：过程性语句和SQL语句。

l RPAD

语法：RPAD（string1,x[,string2]）

功能：返回在X字符长度的位置上插入一个string2中的字符的string1。如果string2的长度要比X字符少，就按照需要进行复制。如 果string2多于X字符，则仅string1前面的X各字符被使用。如果没有指定string2，那么使用空格进行填充。X是使用显示长度可以比字符 串的实际长度要长。RPAD的行为方式与LPAD很相似，除了它是在右边而不是在左边进行填充。

使用位置：过程性语句和SQL语句。

l RTRIM
语法: RTRIM（string1,[,string2]）

功能: 返回删除从右边算起出现在string1中出现的字符string2. string2被缺省设置为单个的空格.数据库将扫描string1,从右边开始.当遇到不在string2中的第一个字符,结果就被返回了RTRIM的行为方式与LTRIM很相似.

使用位置：过程性语句和SQL语句。

l SOUNDEX

语法: SOUNDEX（string）

功能: 返回string的声音表示形式.这对于比较两个拼写不同但是发音类似的单词而言很有帮助.

使用位置：过程性语句和SQL语句。

l SUBSTR

语法: SUBSTR（string,a[,b]）

功能: 返回从字母为值a开始b个字符长的string的一个子字符串.如果a是0,那么它就被认为从第一个字符开始.如果是正数,返回字符是从左边向右边进行计 算的.如果b是负数,那么返回的字符是从string的末尾开始从右向左进行计算的.如果b不存在,那么它将缺省的设置为整个字符串.如果b小于1,那么 将返回NULL.如果a或b使用了浮点数,那么该数值将在处理进行以前首先被却为一个整数.

使用位置：过程性语句和SQL语句。

l TRANSLATE

语法: TRANSLATE(string,from_str,to_str)

功能: 返回将所出现的from_str中的每个字符替换为to_str中的相应字符以后的string. TRANSLATE是REPLACE所提供的功能的一个超集.如果from_str比to_str长,那么在from_str中而不在to_str中而外 的字符将从string中被删除,因为它们没有相应的替换字符. to_str不能为空.Oracle把空字符串认为是NULL,并且如果TRANSLATE中的任何参数为NULL,那么结果也是NULL.

使用位置：过程性语句和SQL语句。

l UPPER

语法: UPPER（string）

功能: 返回大写的string.不是字母的字符不变.如果string是CHAR数据类型的,那么结果也是CHAR类型的.如果string是VARCHAR2类型的,那么结果也是VARCHAR2类型的.

使用位置: 过程性语句和SQL语句。

F.2 字符函数——返回数字

这些函数接受字符参数回数字结果.参数可以是CHAR或者是VARCHAR2类型的.尽管实际下许多结果都是整数值,但是返回结果都是简单的NUMBER类型的,没有定义任何的精度或刻度范围.

l ASCII

语法: ASCII（string）

功能: 数据库字符集返回string的第一个字节的十进制表示.请注意该函数仍然称作为ASCII.尽管许多字符集不是7位ASCII.CHR和ASCII是互为相反的函数.CHR得到给定字符编码的响应字符. ASCII得到给定字符的字符编码.

使用位置: 过程性语句和SQL语句。

l INSTR

语法: INSTR（string1, string2[a,b]）

功能: 得到在string1中包含string2的位置. string1时从左边开始检查的,开始的位置为a,如果a是一个负数,那么string1是从右边开始进行扫描的.第b次出现的位置将被返回. a和b都缺省设置为1,这将会返回在string1中第一次出现string2的位置.如果string2在a和b的规定下没有找到,那么返回0.位置的 计算是相对于string1的开始位置的,不管a和b的取值是多少.

使用位置: 过程性语句和SQL语句。

l INSTRB

语法: INSTRB（string1, string2[a,[b]]）

功能: 和INSTR相同,只是操作的对参数字符使用的位置的是字节.

使用位置: 过程性语句和SQL语句。

l LENGTH

语法: LENGTH（string）

功能: 返回string的字节单位的长度.CHAR数值是填充空格类型的,如果string由数据类型CHAR,它的结尾的空格都被计算到字符串长度中间.如果string是NULL,返回结果是NULL,而不是0.

使用位置: 过程性语句和SQL语句。

l LENGTHB

语法: LENGTHB（string）

功能: 返回以字节为单位的string的长度.对于单字节字符集LENGTHB和LENGTH是一样的.

使用位置: 过程性语句和SQL语句。

l NLSSORT

语法: NLSSORT（string[,nlsparams]）

功能: 得到用于排序string的字符串字节.所有的数值都被转换为字节字符串,这样在不同数据库之间就保持了一致性. Nlsparams的作用和NLS_INITCAP中的相同.如果忽略参数,会话使用缺省排序.

使用位置: 过程性语句和SQL语句。

oracle积累

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

1. 网站资料
   http://www.oracle.com/technology/global/cn/obe/11gr1_db/bidw/partition/partition.htm
2. 计算本星期的起始结束日期
   --得到星期一的日期
   select trunc(sysdate,"DD")-to_char(sysdate,"D")+2 from dual;
   --得到星期天的日期
   select trunc(sysdate,"DD")-to_char(sysdate,"D")+8 from dual;
3. 使用explain plan分析sql;
   使用如下的两个命令显示查询执行的路径:
   set autotrace on 命令
   explain plan 命令
   set autotrace on 命令返回查询数据,显示sql执行路径;
   explain plan 显示sql执行路径,不执行查询操作,可见,explain plan更具实用性;

   下面是我的操作记录
   登录
   sqlplus dev/chenli@testdb;
   执行
   set autotrace on;
   提示错误信息:
   SP2-0613: Unable to verify PLAN_TABLE format or existence
   SP2-0611: Error enabling EXPLAIN report
   错误信息告诉我们,没有plan_table,由于plan_table表结构在不同版本的oracle下会有所不同,所以不要随便创建这个表,在oracle的安装路径下面,提供了创建plan_table表的sql脚本,该sql脚本文件的路径是:$ORACLE_HOME/rdbms/admin/utlxplan.sql
   在sqlplusq模式下,执行如下命令,创建plan_table
   @$ORACLE_HOME/rdbms/admin/utlxplan.sql
   返回
   Table created.
   也可以把这个文件中的建表sql拷贝出来,到plsqldever中执行,创建plan_table表;
   sqlplus模式下,再次执行set autotrace on;
   没有错误信息返回;
   这时再执行sql语句,就会在返回结果后面,显示sql执行路径,和分析结果;

   用explain plan;不先运行查询sql的情况下,生成查询的执行路径;
   命令格式如下:
   explain plan
   for
   sql;
4. 查看组合分区建表脚本
   在plsqldever中,只能看到组合分区表的表结构,看不到建表脚本,为了查看建表脚本,可以通过一个oracle自带的程序包来完成,具体操作sql如下:
   SELECT DBMS_METADATA.GET_DDL("TABLE","TABLE_NAME","USER_NAME") FROM DUAL;
   在上面的参数中,TABLE_NAME是被查询的表的名字,USER_NAME表归属的用户名字.
   如果如上格式填写正确的参数,执行sql,就可以抓去到组合分区表的建表脚本.
   如果执行如上sql的时候,出现如下错误:
   ORA-31603: object "CP_EXT_PKGINFO_HIS_T" of type TABLE not found in schema
   "SCHEMA"
   ORA-06512: at "SYS.DBMS_SYS_ERROR", line 105

ORA-06512: at "SYS.DBMS_METADATA_INT", line 3209
ORA-06512: at "SYS.DBMS_METADATA_INT", line 3594
ORA-06512: at "SYS.DBMS_METADATA_INT", line 4483
ORA-06512: at "SYS.DBMS_METADATA", line 326
ORA-06512: at "SYS.DBMS_METADATA", line 410
ORA-06512: at "SYS.DBMS_METADATA", line 449
ORA-06512: at "SYS.DBMS_METADATA", line 615
ORA-06512: at "SYS.DBMS_METADATA", line 1221
ORA-06512: at line 1
那么,请检查传入的三个参数,最应该注意的是最后一个参数,他是一个数据库用户名;有些参考书上将上面描述成SELECT DBMS_METADATA.GET_DDL("TABLE","TABLE_NAME","SCHEMA") FROM DUAL;
其实SCHEMA在oracle中就是一个oracle用户

5. 取消日志记录,提高性能
在海量插入数据和创建表的时候,不记录日志比记录日志在性能上有很大的提高,使用方法
insert into table_a nologging select * from table_b;
create table_a nologging as select * from table_b;

oracle中sql*plus命令大全

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

在中国ORACLE用户讨论组（http://www.cnoug.org）中发现了这篇文章，很不错，转过来，与大家共享，也方便自己查看。

oracle的sql*plus是与oracle进行交互的客户端工具。在sql*plus中，可以运行sql*plus命令与sql*plus语句。

我们通常所说的DML、DDL、DCL语句都是sql*plus语句，它们执行完后，都可以保存在一个被称为sql buffer的内存区域中，并且只能保存一条最近执行的sql语句，我们可以对保存在sql buffer中的sql 语句进行修改，然后再次执行，sql*plus一般都与数据库打交道。

除了sql*plus语句，在sql*plus中执行的其它语句我们称之为sql*plus命令。它们执行完后，不保存在sql buffer的内存区域中，它们一般用来对输出的结果进行格式化显示，以便于制作报表。

下面就介绍一下一些常用的sql*plus命令:

1. 执行一个SQL脚本文件
SQL>start file_name
SQL>@ file_name
我们可以将多条sql语句保存在一个文本文件中，这样当要执行这个文件中的所有的sql语句时，用上面的任一命令即可，这类似于dos中的批处理。

@与@@的区别是什么?
@等于start命令，用来运行一个sql脚本文件。
@命令调用当前目录下的，或指定全路径，或可以通过SQLPATH环境变量搜寻到的脚本文件。该命令使用是一般要指定要执行的文件的全路径，否则从缺省路径(可用SQLPATH变量指定)下读取指定的文件。
@@用在sql脚本文件中，用来说明用@@执行的sql脚本文件与@@所在的文件在同一目录下，而不用指定要执行sql脚本文件的全路径，也不是从SQLPATH环境变量指定的路径中寻找sql脚本文件，该命令一般用在脚本文件中。
如：在c:\temp目录下有文件start.sql和nest_start.sql，start.sql脚本文件的内容为:
@@nest_start.sql - - 相当于@ c:\temp\nest_start.sql
则我们在sql*plus中，这样执行:
SQL> @ c:\temp\start.sql

2. 对当前的输入进行编辑
SQL>edit

3. 重新运行上一次运行的sql语句
SQL>/

4. 将显示的内容输出到指定文件
SQL> SPOOL file_name
在屏幕上的所有内容都包含在该文件中，包括你输入的sql语句。

5. 关闭spool输出
SQL> SPOOL OFF
只有关闭spool输出，才会在输出文件中看到输出的内容。

6．显示一个表的结构
SQL> desc table_name

7. COL命令:
主要格式化列的显示形式。
该命令有许多选项，具体如下:
COL[UMN] [{ column|expr} [option ...]]
Option选项可以是如下的子句:
ALI[AS] alias
CLE[AR]
FOLD_A[FTER]
FOLD_B[EFORE]
FOR[MAT] format
HEA[DING] text
JUS[TIFY] {L[EFT]|C[ENTER]|C[ENTRE]|R[IGHT]}
LIKE { expr|alias}

NEWL[INE]
NEW_V[ALUE] variable
NOPRI[NT]|PRI[NT]
NUL[L] text
OLD_V[ALUE] variable
ON|OFF
WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]

1). 改变缺省的列标题
COLUMN column_name HEADING column_heading
For example:
```
Sql>select * from dept;
    DEPTNO DNAME                    LOC
---------- ------------------------------ ---------
        10 ACCOUNTING                 NEW YORK
sql>col  LOC heading location
sql>select * from dept;
   DEPTNO DNAME                    location
--------- ------------------------------ -----------
       10 ACCOUNTING                 NEW YORK
```

2). 将列名ENAME改为新列名EMPLOYEE NAME并将新列名放在两行上:
```
Sql>select * from emp
Department  name          Salary
---------- ---------- ----------
        10 aaa               11
SQL> COLUMN ENAME HEADING 'Employee|Name'
Sql>select * from emp
          Employee
Department  name          Salary
---------- ---------- ----------
        10 aaa               11
```
note: the col heading turn into two lines from one line.

3). 改变列的显示长度:
FOR[MAT] format
```
Sql>select empno,ename,job from emp;
    EMPNO ENAME      JOB
---------- ----------     ---------
      7369 SMITH      CLERK
      7499 ALLEN      SALESMAN
7521 WARD       SALESMAN
Sql> col ename format a40
    EMPNO ENAME                                    JOB
----------   ----------------------------------------        ---------
      7369 SMITH                                   CLERK
      7499 ALLEN                                    SALESMAN
      7521 WARD                                   SALESMAN
```

4). 设置列标题的对齐方式
JUS[TIFY] {L[EFT]|C[ENTER]|C[ENTRE]|R[IGHT]}
```
SQL> col ename justify center
SQL> /
    EMPNO          ENAME                JOB
----------   ----------------------------------------       ---------
      7369 SMITH                                   CLERK
      7499 ALLEN                                    SALESMAN
7521 WARD                                  SALESMAN
```
对于NUMBER型的列，列标题缺省在右边，其它类型的列标题缺省在左边

5). 不让一个列显示在屏幕上
NOPRI[NT]|PRI[NT]
```
SQL> col job noprint
SQL> /
    EMPNO          ENAME
```

---------- --
7369 SMITH
7499 ALLEN
7521 WARD

6). 格式化NUMBER类型列的显示:
SQL> COLUMN SAL FORMAT $99,990
SQL> /
Employee
Department Name Salary Commission
---------- ---------- --------- ----------
30 ALLEN $1,600 300

7). 显示列值时，如果列值为NULL值，用text值代替NULL值
COMM NUL[L] text
SQL>COL COMM NUL[L] text

8). 设置一个列的回绕方式
WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]
COL1

HOW ARE YOU?

SQL>COL COL1 FORMAT A5
SQL>COL COL1 WRAPPED
COL1

HOW A
RE YO
U?

SQL> COL COL1 WORD_WRAPPED
COL1

HOW
ARE
YOU?

SQL> COL COL1 WORD_WRAPPED
COL1

HOW A

9). 显示列的当前的显示属性值
SQL> COLUMN column_name

10). 将所有列的显示属性设为缺省值
SQL> CLEAR COLUMNS

8. 屏蔽掉一个列中显示的相同的值
BREAK ON break_column
SQL> BREAK ON DEPTNO
SQL> Select DEPTNO, ENAME, SAL
FROM EMP
Where SAL < 2500
orDER BY DEPTNO;
DEPTNO ENAME SAL
---------- ---------- ---------
10 CLARK 2450
MILLER 1300
20 SMITH 800
ADAMS 1100

9. 在上面屏蔽掉一个列中显示的相同的值的显示中，每当列值变化时在值变化之前插入n个空行。
BREAK ON break_column SKIP n

SQL> BREAK ON DEPTNO SKIP 1
SQL> /
DEPTNO ENAME SAL
---------- ---------- ---------
10 CLARK 2450
MILLER 1300

20 SMITH 800
ADAMS 1100

10. 显示对BREAK的设置
SQL> BREAK

11. 删除6、7的设置
SQL> CLEAR BREAKS

12. Set 命令:
该命令包含许多子命令:
SET system_variable value
system_variable value 可以是如下的子句之一:
APPI[NFO]{ON|OFF|text}
ARRAY[SIZE] {15|n}
AUTO[COMMIT]{ON|OFF|IMM[EDIATE]|n}
AUTOP[RINT] {ON|OFF}
AUTORECOVERY [ON|OFF]
AUTOT[RACE] {ON|OFF|TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]
BLO[CKTERMINATOR] {.|c}
CMDS[EP] {;|c|ON|OFF}
COLSEP {_|text}
COM[PATIBILITY]{V7|V8|NATIVE}
CON[CAT] {.|c|ON|OFF}
COPYC[OMMIT] {0|n}
COPYTYPECHECK {ON|OFF}
DEF[INE] {&|c|ON|OFF}
DESCRIBE [DEPTH {1|n|ALL}][LINENUM {ON|OFF}][INDENT {ON|OFF}]
ECHO {ON|OFF}
EDITF[ILE] file_name[.ext]
EMB[EDDED] {ON|OFF}
ESC[APE] {\|c|ON|OFF}
FEED[BACK] {6|n|ON|OFF}
FLAGGER {OFF|ENTRY |INTERMED[IATE]|FULL}
FLU[SH] {ON|OFF}
HEA[DING] {ON|OFF}
HEADS[EP] {||c|ON|OFF}
INSTANCE [instance_path|LOCAL]
LIN[ESIZE] {80|n}
LOBOF[FSET] {n|1}
LOGSOURCE [pathname]
LONG {80|n}
LONGC[HUNKSIZE] {80|n}
MARK[UP] HTML [ON|OFF] [HEAD text] [BODY text] [ENTMAP {ON|OFF}] [SPOOL
{ON|OFF}] [PRE[FORMAT] {ON|OFF}]
NEWP[AGE] {1|n|NONE}
NULL text
NUMF[ORMAT] format
NUM[WIDTH] {10|n}
PAGES[IZE] {24|n}
PAU[SE] {ON|OFF|text}
RECSEP {WR[APPED]|EA[CH]|OFF}
RECSEPCHAR {_|c}
SERVEROUT[PUT] {ON|OFF} [SIZE n] [FOR[MAT] {WRA[PPED]|WOR[D_
WRAPPED]|TRU[NCATED]}]
SHIFT[INOUT] {VIS[IBLE]|INV[ISIBLE]}
SHOW[MODE] {ON|OFF}

SQLBL[ANKLINES] {ON|OFF}
SQLC[ASE] {MIX[ED]|LO[WER]|UP[PER]}
SQLCO[NTINUE] {> |text}
SQLN[UMBER] {ON|OFF}
SQLPRE[FIX] {#|c}
SQLP[ROMPT] {SQL>|text}
SQLT[ERMINATOR] {;|c|ON|OFF}
SUF[FIX] {SQL|text}
TAB {ON|OFF}
TERM[OUT] {ON|OFF}
TI[ME] {ON|OFF}
TIMI[NG] {ON|OFF}
TRIM[OUT] {ON|OFF}
TRIMS[POOL] {ON|OFF}
UND[ERLINE] {-|c|ON|OFF}
VER[IFY] {ON|OFF}
WRA[P] {ON|OFF}

1). 设置当前session是否对修改的数据进行自动提交
SQL>SET AUTO[COMMIT] {ON|OFF|IMM[EDIATE]| n}

2)．在用start命令执行一个sql脚本时，是否显示脚本中正在执行的SQL语句
SQL> SET ECHO {ON|OFF}

3).是否显示当前sql语句查询或修改的行数
SQL> SET FEED[BACK] {6|n|ON|OFF}
默认只有结果大于6行时才显示结果的行数。如果set feedback 1 ，则不管查询到多少行都返回。当为off 时，一律不显示查询的行数

4).是否显示列标题
SQL> SET HEA[DING] {ON|OFF}
当set heading off 时，在每页的上面不显示列标题，而是以空白行代替

5).设置一行可以容纳的字符数
SQL> SET LIN[ESIZE] {80|n}
如果一行的输出内容大于设置的一行可容纳的字符数，则折行显示。

6).设置页与页之间的分隔
SQL> SET NEWP[AGE] {1|n|NONE}
当set newpage 0 时，会在每页的开头有一个小的黑方框。
当set newpage n 时，会在页和页之间隔着n个空行。
当set newpage none 时，会在页和页之间没有任何间隔。

7).显示时，用text值代替NULL值
SQL> SET NULL text

8).设置一页有多少行数
SQL> SET PAGES[IZE] {24|n}
如果设为0，则所有的输出内容为一页并且不显示列标题

9).是否显示用DBMS_OUTPUT.PUT_LINE包进行输出的信息。
SQL> SET SERVEROUT[PUT] {ON|OFF}
在编写存储过程时，我们有时会用dbms_output.put_line将必要的信息输出，以便对存储过程进行调试，只有将serveroutput变量设为on后，信息才能显示在屏幕上。

10).当SQL语句的长度大于LINESIZE时，是否在显示时截取SQL语句。
SQL> SET WRA[P] {ON|OFF}
当输出的行的长度大于设置的行的长度时（用set linesize n命令设置），当set wrap on时，输出行的多于的字符会另起一行显示，否则，会将输出行的多于字符切除，不予显示。

11).是否在屏幕上显示输出的内容，主要用与SPOOL结合使用。
SQL> SET TERM[OUT] {ON|OFF}
在用spool命令将一个大表中的内容输出到一个文件中时，将内容输出在屏幕上会耗费大量的时间，设置set termspool off后，则输出的内容只会保存在输出文件中，不会显示在屏幕上，极大的提高了spool的速度。

12).将SPOOL输出中每行后面多余的空格去掉
SQL> SET TRIMS[OUT] {ON|OFF}

13)显示每个sql语句花费的执行时间
set TIMING {ON|OFF}

14). 遇到空行时不认为语句已经结束，从后续行接着读入。
SET SQLBLANKLINES ON
Sql*plus中, 不允许sql语句中间有空行, 这在从其它地方拷贝脚本到sql*plus中执行时很麻烦. 比如下面的脚本:
select deptno, empno, ename
from emp

where empno = "7788";
如果拷贝到sql*plus中执行, 就会出现错误。这个命令可以解决该问题

15).设置DBMS_OUTPUT的输出
SET SERVEROUTPUT ON BUFFER 20000
用dbms_output.put_line("strin_content");可以在存储过程中输出信息，对存储过程进行调试
如果想让dbms_output.put_line(" abc");的输出显示为:
SQL> abc，而不是SQL>abc，则在SET SERVEROUTPUT ON后加format wrapped参数。

16). 输出的数据为html格式
set markup html
在8.1.7版本(也许是816? 不太确定)以后, sql*plus中有一个set markup html的命令, 可以将sql*plus的输出以html格式展现.
注意其中的spool on, 当在屏幕上输出的时候, 我们看不出与不加spool on有什么区别, 但是当我们使用spool filename 输出到文件的时候, 会看到spool文件中出现了等tag.

14．修改sql buffer中的当前行中，第一个出现的字符串
C[HANGE] /old_value/new_value
```
SQL> l
  1* select * from dept
SQL> c/dept/emp
  1* select * from emp
```

15．编辑sql buffer中的sql语句
EDI[T]

16．显示sql buffer中的sql语句，list n显示sql buffer中的第n行，并使第n行成为当前行
L[IST] [n]

17．在sql buffer的当前行下面加一行或多行
I[NPUT]

18．将指定的文本加到sql buffer的当前行后面
A[PPEND]
```
SQL> select deptno,
  2  dname
  3  from dept;
    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES
        40 OPERATIONS

SQL> L 2
  2* dname
SQL> a ,loc
  2* dname,loc
SQL> L
  1  select deptno,
  2  dname,loc
  3* from dept
SQL> /
```

```
    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

19．将sql buffer中的sql语句保存到一个文件中
SAVE file_name

20．将一个文件中的sql语句导入到sql buffer中
GET file_name

21．再次执行刚才已经执行的sql语句
RUN
or
/

22．执行一个存储过程
EXECUTE procedure_name

23．在sql*plus中连接到指定的数据库
CONNECT user_name/passwd@db_alias

24．设置每个报表的顶部标题
TTITLE

25．设置每个报表的尾部标题
BTITLE

26．写一个注释
REMARK [text]

27．将指定的信息或一个空行输出到屏幕上
PROMPT [text]

28．将执行的过程暂停，等待用户响应后继续执行
PAUSE [text]

Sql>PAUSE Adjust paper and press RETURN to continue.

29．将一个数据库中的一些数据拷贝到另外一个数据库（如将一个表的数据拷贝到另一个数据库）
COPY {FROM database | TO database | FROM database TO database}
{APPEND|Create|Insert|REPLACE} destination_table
[(column, column, column, ...)] USING query

sql>COPY FROM SCOTT/TIGER@HQ TO JOHN/CHROME@WEST
create emp_temp
USING Select * FROM EMP

30．不退出sql*plus，在sql*plus中执行一个操作系统命令：
HOST

Sql> host hostname
该命令在windows下可能被支持。

31．在sql*plus中，切换到操作系统命令提示符下，运行操作系统命令后，可以再次切换回sql*plus：
!

sql>!
$hostname
$exit
sql>

该命令在windows下不被支持。

32．显示sql*plus命令的帮助
HELP
如何安装帮助文件：
Sql>@ ?\sqlplus\admin\help\hlpbld.sql ?\sqlplus\admin\help\helpus.sql
Sql>help index

33．显示sql*plus系统变量的值或sql*plus环境变量的值
Syntax
SHO[W] option
where option represents one of the following terms or clauses:
system_variable
ALL
BTI[TLE]
ERR[ORS] [{FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY|
TRIGGER|VIEW|TYPE|TYPE BODY} [schema.]name]
LNO
PARAMETERS [parameter_name]
PNO
REL[EASE]
REPF[OOTER]
REPH[EADER]
SGA
SPOO[L]
SQLCODE
TTI[TLE]
USER

1) . 显示当前环境变量的值：
Show all

2) . 显示当前在创建函数、存储过程、触发器、包等对象的错误信息
Show error
当创建一个函数、存储过程等出错时，变可以用该命令查看在那个地方出错及相应的出错信息，进行修改后再次进行编译。

3) . 显示初始化参数的值：
show PARAMETERS [parameter_name]

4) . 显示数据库的版本：
show REL[EASE]

5) . 显示SGA的大小
show SGA

6). 显示当前的用户名
show user

34.查询一个用户下的对象
SQL>select * from tab;
SQL>select * from user_objects;

35.查询一个用户下的所有的表
SQL>select * from user_tables;

36.查询一个用户下的所有的索引
SQL>select * from user_indexes;

37. 定义一个用户变量
方法有两个：
a. define
b. COL[UMN] [{column|expr} NEW_V[ALUE] variable [NOPRI[NT]|PRI[NT]]
OLD_V[ALUE] variable [NOPRI[NT]|PRI[NT]]

下面对每种方式给予解释:
a. Syntax
DEF[INE] [variable]|[variable = text]
定义一个用户变量并且可以分配给它一个CHAR值。

assign the value MANAGER to the variable POS, type:
SQL> DEFINE POS = MANAGER

assign the CHAR value 20 to the variable DEPTNO, type:
SQL> DEFINE DEPTNO = 20

list the definition of DEPTNO, enter
SQL> DEFINE DEPTNO
————————————————————
DEFINE DEPTNO = ”20” (CHAR)

定义了用户变量POS后，就可以在sql*plus中用&POS或&&POS来引用该变量的值，sql*plus不会再提示你给变量输入值。

b. COL[UMN] [{column|expr} NEW_V[ALUE] variable [NOPRI[NT]|PRI[NT]]
NEW_V[ALUE] variable
指定一个变量容纳查询出的列值。
例:column col_name new_value var_name noprint
  select col_name from table_name where ……..
将下面查询出的col_name列的值赋给var_name变量.

一个综合的例子:
得到一个列值的两次查询之差(此例为10秒之内共提交了多少事务):
column redo_writes new_value commit_count

select sum(stat.value) redo_writes
from v$sesstat stat, v$statname sn
where stat.statistic# = sn.statistic#
and sn.name = "user commits";

-- 等待一会儿(此处为10秒);
execute dbms_lock.sleep(10);

set veri off
select sum(stat.value) - &commit_count commits_added
from v$sesstat stat, v$statname sn
where stat.statistic# = sn.statistic#
and sn.name = "user commits";

38. 定义一个绑定变量
VAR[IABLE] [variable [NUMBER|CHAR|CHAR (n)|NCHAR|NCHAR (n) |VARCHAR2 (n)|NVARCHAR2 (n)|CLOB|NCLOB|REFCURSOR]]
定义一个绑定变量，该变量可以在pl/sql中引用。
可以用print命令显示该绑定变量的信息。
如:
column inst_num heading "Inst Num" new_value inst_num format 99999;
column inst_name heading "Instance" new_value inst_name format a12;
column db_name   heading "DB Name"   new_value db_name   format a12;
column dbid   heading "DB Id"   new_value dbid   format 9999999999 just c;

prompt
prompt Current Instance
prompt ~~~~~~~~~~~~~~~~

select d.dbid   dbid
    , d.name   db_name
    , i.instance_number inst_num
    , i.instance_name   inst_name
 from v$database d,
      v$instance i;

```
variable dbid       number;
variable inst_num   number;
begin
  :dbid     :=  &dbid;
  :inst_num  :=  &inst_num;
end;
/
```

说明:
在sql*plus中，该绑定变量可以作为一个存储过程的参数，也可以在匿名PL/SQL块中直接引用。为了显示用VARIABLE命令创建的绑定变量的值，可以用print命令

注意:
绑定变量不同于变量:
1.　　定义方法不同
2.　　引用方法不同
绑定变量：:variable_name
　　变量：&variable_name or &&variable_name
3.在sql*plus中，可以定义同名的绑定变量与用户变量，但是引用的方法不同。

39. &与&&的区别
&用来创建一个临时变量，每当遇到这个临时变量时，都会提示你输入一个值。
&&用来创建一个持久变量，就像用用define命令或带new_vlaue字句的column命令创建的持久变量一样。当用&&命令引用这个变量时，不会每次遇到该变量就提示用户键入值，而只是在第一次遇到时提示一次。

如，将下面三行语句存为一个脚本文件，运行该脚本文件，会提示三次，让输入deptnoval的值:

```
select count(*) from emp where deptno = &deptnoval;
select count(*) from emp where deptno = &deptnoval;
select count(*) from emp where deptno = &deptnoval;
```

将下面三行语句存为一个脚本文件，运行该脚本文件，则只会提示一次，让输入deptnoval的值:

```
select count(*) from emp where deptno = &&deptnoval;
select count(*) from emp where deptno = &&deptnoval;
select count(*) from emp where deptno = &&deptnoval;
```

40．在输入sql语句的过程中临时先运行一个sql*plus命令(摘自www.itpub.com)
#
有没有过这样的经历? 在sql*plus中敲了很长的命令后, 突然发现想不起某个列的名字了, 如果取消当前的命令,待查询后再重敲, 那太痛苦了. 当然你可以另开一个sql*plus窗口进行查询, 但这里提供的方法更简单.

比如说, 你想查工资大于4000的员工的信息, 输入了下面的语句:

```
SQL> select deptno, empno, ename
2 from emp
3 where
```

这时, 你发现你想不起来工资的列名是什么了.

这种情况下, 只要在下一行以#开头, 就可以执行一条sql*plus命令, 执行完后, 刚才的语句可以继续输入

```
SQL>> select deptno, empno, ename
2 from emp
3 where
6 #desc emp
Name Null? Type
----------------------------------------- -------- --------------
EMPNO NOT NULL NUMBER(4)
ENAME VARCHAR2(10)
JOB VARCHAR2(9)
MGR NUMBER(4)
HIREDATE DATE
SAL NUMBER(7,2)
COMM NUMBER(7,2)
DEPTNO NUMBER(2)

6 sal > 4000;
```

```
DEPTNO EMPNO ENAME
---------- ---------- ----------
10 7839 KING
```

41. SQLPlus中的快速复制和粘贴技巧(摘自www.cnoug.org)
1) 鼠标移至想要复制内容的开始
2) 用右手食指按下鼠标左键
3) 向想要复制内容的另一角拖动鼠标，与Word中选取内容的方法一样
4) 内容选取完毕后（所选内容全部反显），鼠标左键按住不动，用右手中指按鼠标右键
5) 这时，所选内容会自动复制到SQL*Plus环境的最后一行

点击这里进入对应文章地址 [进入主站]

oracle中导出指定记录

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

1。用pl/sql工具导出数据 里边有个where clause 输入查询条件，可以设置，比如rownow<1001，导出前一千条。

2。exp yst/yst file=d:\file.dmp tables=yst_zy query=\ "where zyid=0001\ " grants=
y

Oracle主要的配置文件解释(转贴)

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

Oracle 主要配置文件介绍:

profile文件，oratab 文件，数据库实例初始化文件 initSID.ora，监听配置文件， sqlnet.ora 文件，tnsnames.ora 文件

1.2 Oracle 主要配置文件介绍

1.2.1 /etc/profile 文件

系统级的环境变量一般在/etc/profile 文件中定义 在 CAMS 系统 与数据库相关的环境变量就定义在/etc/profile 文件中 如下所示

```
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/8.1.7
export PATH=$PATH:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib
export ORACLE_SID=cams
export ORACLE_TERM=vt100
export ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
export NLS_LANG=AMERICAN.ZHS16CGB231280
```

& 说明

1 配置上述环境变量要注意定义的先后顺序 如 定义 ORACLE_HOME 时 用 到 了 ORACLE_BASE 那 么 ORACLE_HOME 的 定 义 应 该 在 ORACLE_BASE之后

2 在使用中文版 CAMS 时 环境变量 NLS_LANG 的值应该设置为 AMERICAN.ZHS16CGB231280 如上所示 在使用英文版 CAMS时 可以不设置 NLS_LANG 即去掉 export NLS_LANG=... ... 那一行 也可以 设置 NLS_LANG 的值为 AMERICAN_AMERICA.US7ASCII

1.2.2 /etc/oratab 文件

/etc/oratab 文件描述目前系统中创建的数据库实例 以及是否通过 dbstart 和 dbshut 来控制该实例的启动与关闭 如下所示 忽略以#开头的注释部分 :

```
cams:/u01/app/oracle/product/8.1.7:Y
```

其中 cams 为实例 ID /u01/app/oracle/product/8.1.7为 ORACLE_HOME目录 Y表示允许使用 dbstart和 dbshut 启动和关闭该实例数据库 如果设置为 N 表示不通过 dbstart 和 dbshut 启动和关闭实例数据库 CAMS 系统要求在安装完 ORACLE 后要求将该参数修改为 Y 以保证 ORACLE 数据库自启动和关闭

1.2.3 数据库实例初始化文件 initSID.ora

每个数据库实例都有一个初始化 参数文件 其缺省 存放的路径为 $ORACLE_BASE/admin/<SID>/pfile 其名称为 init<SID>.ora 如 cams 实例 对 应 的 参 数 文 件 为 initcams.ora 缺 省 存 放 路 径 为 $ORACLE_BASE/admin/cams/pfile 即/u01/app/oracle/admin/cams/pfile 但 在 CAMS 应 用 中 initcams.ora 的 存 放 路 径 为 /u02/app/oracle/admin/cams/pfile 这是基于数据与应用程序分开存放更好地保护数据考虑的 尤其在 CAMS 双机应用模式下 能够保证数据的一致性 具体的修改操作可参考 Linux与 Oracle 安装手册

初始化参数文件是一个包含实例配置参数的文本文件 这些参数被设置为特定的值 用于初始化 Oracle 实例的多数内存和进程设置 以下是一些主要参数的说明

1 实例的数据库名称
db_name = "cams"
2 实例名称
instance_name = cams
3 数据库控制文件的名称和位置
control_files = ("/u02/app/oracle/oradata/cams/control01.ctl",
"/u02/app/oracle/oradata/cams/control02.ctl",
"/u02/app/oracle/oradata/cams/control03.ctl")
4 调度作业队列的 SNP 进程的数量以及 SNP 进程觉醒时间间隔 秒
JOB_QUEUE_PROCESSES=2
JOB_QUEUE_INTERVAL=60

5 存储追踪和告警文件的路径
user_dump_dest 指定记录 Oracle 用户进程产生的追踪和告警信息的文件的存放路径 background_dump_dest 指定记录 Oracle 后台进程产生的追踪和告警信息的文件的存放路径 core_dump_dest指定Oracle运行所产生的core dump 信息的文件的存放路径
background_dump_dest = /u02/app/oracle/admin/cams/bdump
core_dump_dest = /u02/app/oracle/admin/cams/cdump
user_dump_dest = /u02/app/oracle/admin/cams/udump

6 UTL_FILE_DIR 参数
UTL_FILE_DIR = *
UTL_FILE_DIR 参数指定一个或多个目录用于 Oracle 应用的文件 I/O 如备份数据到文件 在 CAMS 系统中将该值设置为 * 表示可供 Oracle 应用进行文件 I/O操作的目录为任意目录 因此 只要空间允许 可以将备份数据存放到任意目录下

1.2.4 监听配置文件
为了使得外部进程 如 CAMS后台程序 能够访问 Oracle 数据库 必须配置 Oracle 网络服务器环境 配置 Oracle 网络服务器环境是通过配置 listener.ora sqlnet.ora 和 tnsnames.ora 共三个文件来进行的 listener.ora 即监听配置文件 在本小节说明 另两个文件分别在随后的两个小节说明 监听配置文件 listener.ora 的存放路径为 $ORACLE_HOME/network/admin 以下是一个示例

```
LISTENER =                    #监听器名称
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS =
(PROTOCOL = TCP)
(HOST = localhost.localdomain)
(PORT = 1521))
      )
    )
    (DESCRIPTION =
      (PROTOCOL_STACK =
        (PRESENTATION = GIOP)
        (SESSION = RAW)
      )
      (ADDRESS =
(PROTOCOL = TCP)
(HOST = localhost.localdomain)
(PORT = 2481))
```

```
    )
  )

SID_LIST_LISTENER = #命名规则 SID_LIST_+上面定义的监听器名称
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = PLSExtProc1)
   (ORACLE_HOME = /u01/app/oracle/product/8.1.7)
   (PROGRAM = extproc)
  )
  (SID_DESC =
   (GLOBAL_DBNAME = cams)
   (ORACLE_HOME = /u01/app/oracle/product/8.1.7)
   (SID_NAME = cams)
  )
  (SID_DESC =
   (GLOBAL_DBNAME = oid)
   (ORACLE_HOME = /u01/app/oracle/product/8.1.7)
   (SID_NAME = oid)
  )
 )
```

& 说明

1 listener.ora 文件中定义一个监听器 其缺省的名称为 LISTENER，这个监听器缺省以tcp/ip为协议地址且端口号为1521运行 在CAMS应用中监听文件定义的监听器就使用这个缺省名字，并且使用缺省的协议 tcp/ip 和缺省的端口号。1521 待配置好监听文件以及随后说明的 sqlnet.ora 和tnsnames.ora 文件之后 就可以用以下命令将监听文件中定义的监听器启动起来。

```
$ lsnrctl start
```

停止监听器的命令为

```
$ lsnrctl stop
```

监测监听器当前状态的命令为

```
$ lsnrctl status
```

当 lsnrctl status 命令有如下输出结果

```
STATUS of the LISTENER
------------------------
Alias                LISTENER
Version              TNSLSNR for Linux: Version 8.1.7.4.0 - Production
Start Date           17-JAN-2004 19:00:08
Uptime               31 days 15 hr. 27 min. 59 sec
```

就说明监听器正在运行，否则说明监听器已经停止了，CAMS 系统的后台程序的正常运行不仅依赖于数据库实例的运行，还依赖于这个数据库监听器的运行。假如监听器没有启动，即使数据库已经启动，CAMS 后台程序仍然不能正常工作。

2 如(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))所示的一个IPC 协议地址的监听，是为了外部进程调用用的，在数据库安装时自动设定不需要改动。

3 在监听文件后部还有一个 SID_LIST_LISTENER 段，该段用于定义监听器的服务，即为哪些数据库实例提供监听服务，以 cams 实例为例，其对应的服务信息为:

```
 (SID_DESC =
  (GLOBAL_DBNAME = cams) #数据库名
  (ORACLE_HOME = /u01/app/oracle/product/8.1.7)
  (SID_NAME = cams) #数据库实例名
)
```

1.2.5 sqlnet.ora 文件

sqlnet.ora 文件的存放路径为 $ORACLE_HOME/network/admin 以下是一个示例

NAMES.DEFAULT_DOMAIN = localdomain
NAMES.DIRECTORY_PATH= (TNSNAMES, ONAMES, HOSTNAME)
& 说明
NAMES.DEFAULT_DOMAIN 指定网络域名，NAMES.DIRECTORY_PATH指定当解析客户端连接标识符时命名方法，naming metthods 采用的优先顺序，从左至右递减，在 CAMS 应用中，这两个参数采用上述所示的系统缺省值。

1.2.6 tnsnames.ora 文件
tnsnames.ora 文件的存放路径为 $ORACLE_HOME/network/admin 以下
是一个示例

```
OID.LOCALDOMAIN =
 (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT =
1521))
   )
   (CONNECT_DATA =
    (SERVICE_NAME = oid)
   )
 )
CAMS.LOCALDOMAIN =
 (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT =
1521))
   )
   (CONNECT_DATA =
    (SERVICE_NAME = cams)
   )
 )
INST1_HTTP.LOCALDOMAIN =
 (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT =

1521))
   )
   (CONNECT_DATA =
    (SERVER = SHARED)
    (SERVICE_NAME = MODOSE)
    (PRESENTATION = http://admin)
   )
 )
EXTPROC_CONNECTION_DATA.LOCALDOMAIN =
  (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
   )
   (CONNECT_DATA =
    (SID = PLSExtProc1)
    (PRESENTATION = RO)
   )
  )
```

& 说明
tnsnames.ora 文件中定义一个或多个网络服务 net service、cams 实例对应的网络服务为:
CAMS.LOCALDOMAIN =

```
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS =
    (PROTOCOL = TCP)
     (HOST = localhost.localdomain)
     (PORT = 1521))
   )
  (CONNECT_DATA =
   (SERVICE_NAME = cams)
  )
 )
```

注意 这里 ADDRESS项包含三个子参数

PROTOCOL： 默认协议TCP

HOST :ip地址

PORT：端口，默认1521

CAMS.LOCALDOMAIN为数据名

要确保在监听文件中也有对应的一个ADDRESS 项也包含同样的三个子参数，并且子参数的值对应都相等 另外，这里 SERVICE_NAME 的值必需确保与监听文件中某SID_DESC项下的SID_NAME参数的值相等 。

原帖http://www.4vava.cn/html/database/oracle/20070530/62859.html

防范黑客攻击Oracle系统的八大常用方法

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

Oracle的销售在向客户兜售其数据库系统一直把它吹捧为牢不可破的，要嘴皮子轻易，兑现起来可就不那么轻易了。不管什么计算机系统，人们总能够找到攻击它的方法，Oracle也不例外。本文将和大家从黑客的角度讨论是用哪些方法把黑手伸向了你原以为他们不能触及的数据，希望作为Oracle的数据库治理员能够清楚的阐明自己基础架构的哪些区域比较轻易受到攻击。同时我们也会讨论保护系统防范攻击的方法。

1.SQL注入攻击　　如今大部分的Oracle数据库都具有为某种类型网络应用服务的后端数据存储区，网页应用使数据库更轻易成为我们的攻击目标体现在三个方面。其一，这些应用界面非常复杂，具有多个组成成分，使数据库治理员难以对它们进行彻底检查。其二，阻止程序员侵入的屏障很低，即便不是C语言的编程专家，也能够对一些页面进行攻击。下面我们会简单地解释为什么这对我们这么重要。第三个原因是优先级的问题。网页应用一直处于发展的模式，所以他们在不断变化，推陈出新。这样安全问题就不是一个必须优先考虑的问题。　　SQL注入攻击是一种很简单的攻击，在页面表单里输入信息，静静地加入一些非凡代码，诱使应用程序在数据库里执行这些代码，并返回一些程序员没有料到的结果。例如，有一份用户登录表格，要求输入用户名和密码才能登录，在用户名这一栏，输入以下代码:　　cyw"); select username, passWord from all_users;--

假如数据库程序员没有聪明到能够检查出类似的信息并“清洗”掉我们的输入，该代码将在远程数据库系统执行，然后这些关于所有用户名和密码的敏感数据就会返回到我们的浏览器。　　你可能会认为这是在危言耸听，不过还有更绝的。David Litchfield在他的著作《Oracle黑客手册》(Oracle Hacker"s Handbook)中把某种非凡的pl/sql注入攻击美其名曰：圣杯(holy grail)，因为它曾通杀Oracle 8到Oracle10g的所有Oracle数据库版本。很想知道其作用原理吧。你可以利用一个被称为DBMS_EXPORT_EXTENSION的程序包，使用注入攻击获取执行一个异常处理程序的代码，该程序会赋予用户或所有相关用户数据库治理员的特权。　　这就是Oracle发布的闻名安全升级补丁Security Alert 68所针对的漏洞。不过据Litchfield称，这些漏洞是永远无法完全修补完毕的。

防范此类攻击的方法　　总而言之，虽说没有万能的防弹衣，但鉴于这个问题涉及到所有面向网络的应用软件，还是要尽力防范。目前市面上有各式各样可加以利用的SQL注入检测技术。可以参照http://www.securityfocus.com/infocus/1704 系列文章的具体介绍。　　还可以用不同的入侵检测工具在不同的水平上检测SQL注入攻击。访问专门从事Oracle安全性研究的Pete Finnigan的安全网站http://www.petefinnigan.com/orasec.htm，在该网页搜索“sql injection”，可以获得更多相关信息。Pete Finnigan曾在其博客上报告称Steven Feurstein目前正在编写一个称为SQL Guard 的pl/sql程序包，专门用来防止SQL注入攻击，详情请查看以下网页http://www.petefinnigan.com/weblog/archives/00001115.htm。　　对于软件开发人员来说，很多软件包都能够帮助你“清洗”输入信息。假如你调用对从页面表单接受的每个值都调用清洗例行程序进行处理，这样可以更加严密的保护你的系统。不过，最好使用SQL注入工具对软件进行测试和验证，以确保万无一失。

1. 默认密码　　Oracle数据库是一个庞大的系统，提供了能够创建一切的模式。绝大部分的系统自带用户登录都配备了预设的默认密码。想知道数据库治理员工作是不是够勤奋?这里有一个方法可以找到答案。看看下面这些最常用的预设用户名和密码是不是能够登录到数据库吧:　　Username　　Password　　applsys　　apps　　ctxsys　　change_on_install　　dbsnmp　　dbsnmp　　outln　　outln　　owa　　owa　　perfstat　　perfstat　　scott　　tiger　　system　　change_on_install　　system　　manager　　sys　　change_on_install　　sys　　manager

就算数据库治理员已经很勤奋的把这些默认配对都改了，有时候想猜出登录密码也不是一件困难的事情，逐个试试“oracle”、“oracle4”、“oracle8i”、“oracle11g”，看看碰巧是不是有一个能登录上去的。　　Pete Finnigan提供了一份关于缺省用户和对应密码的名单，该名单非常全面而且是最新的，并包括已经加密的密码。假如你用all_users来进行查询，可以尝试并比较一下这份名单，具体名单请参阅：http://www.petefinnigan.com/default/default_password_list.htm。

防范此类攻击的方法　　作为数据库治理员，应该定期审核所有的数据库密码，假如某些商业方面的阻力使你不能轻易更改轻易被人猜出的密码，你可以尽量心平气和地和相关人员解释，用一些直观的例子来阐明假如不

修改密码的话会有什么不好的事情发生，会有什么样的风险存在。　　Oracle也提供了密码安全profile，你可以激活该profile，在某种水平上加强数据库密码的复杂性，还可以执行定期密码失效。要注重要把这个功能设置为只对通过网络服务器或中间层应用服务器登录的事件起作用。

2. 蛮力攻击(Brute Force)　　蛮力攻击，就像其名字所暗示的，就是不停的撬，直到“锁”打开为止的方法。对于Oracle数据库来说，就是用某种自动执行的进程，通过尝试所有的字母数字组合来破解用户名和密码。Unix的治理员就可以利用一款名为John the Ripper的密码破解软件来执行这类的攻击。现在假如你下载某个补丁，你也可以利用这款软件来对Oracle进行蛮力攻击，敲开其密码。不过根据密码的复杂程度不同，这可能是个很费时的过程，假如你想加快这个进程，可以事先预备一张包含所有密码加密的表，这样的表叫做Rainbow table，你可以为每个用户名预备一张不同的rainbow table，因为这种密码加密算法把用户名作为助燃剂。在这里就不再深入介绍更多的细节问题了，大家可以查阅http://www.antsight.com/zsl/rainbowcrack/获得更多信息。Oracle服务器的默认设置是，对某个特定帐户输错密码达十次就会自动锁定该帐户。不过通常“sys as sysdba”权限没有这个限制，这可能是因为假如你锁定了治理员，那所有人都将被锁定。这样的设置为我们黑客破解软件(OraBrute)如开辟了一条生路，它们会昼夜不停地敲打你数据库的前门，直到它乖乖打开为止。

防范此类攻击的方法　　想要抵御此类攻击，可以使用之前提及的对付预设密码攻击的方法。不过好奇心过重的数据库治理员也可能下载上面提到的工具侵入自己的系统。这说明了你真正的风险来自何方。

4. 从后门偷窃数据　　在安全领域，这个概念被称为数据向外渗漏(exfiltration)，这个词来自军事术语，其反面是向敌人内部渗透(infiltration)，意思就是在不被发现的情况下偷偷潜出。对于从目标数据库获取数据的过程，可能就像从一些磁带备份中挑拣数据和还原数据库或者像从一个被毁坏的磁盘重复制一份拷贝一样简单。不过，也有可能涉及到窥探网络传输以获得相关的数据包。　　Oracle有一个名为UTL_TCP的程序包，能够使外部连接指向其他服务器。对它稍微改编一下，就可以利用它从数据库发送一套低带宽数据流到远程主机。Oracle也附带了一些有用的程序包来隐藏数据流里的信息，假如你在发动潜入行动的时候担心入侵检测系统会监测到你的不法活动，那么可以充分利用这些功能秘密嵌入，包括DBMS_OBFUSCATION_TOOLKIT和DBMS_CRYPTO。

防范此类攻击的方法　　防范此类攻击的最佳办法是安装入侵检测系统，这些系统能够检测网络中流入和流出的数据包。有一些检测系统还提供深入数据包检测，可以确实检查某些SQL，并可以通过设定规则在某种情况下触发报警器。这些工具还能够查找泄密迹象，例如添加的UNION、各种类型的short- circuiting命令、利用“--”注释进行截断等等。

5. 监听器　　计算机世界最让人觉得了不起的事情就是，不管有多么困难的事，总有办法驯服它。尤其是在安全领域，一些漏洞如此的简单，而这些漏洞的出现仅仅是因为用户(也包括我们现在扮演的角色——黑客)并没有像软件设计者(程序员或软件开发员)本来预想的那样思考和行动。　　Oracle监听器的设置是为了能够实现远程治理。那么假如攻击者把监听器的logfile设置为Unix .rhosts文件呢?这样攻击者就可以轻松的对.rhosts文件进行写操作。Unix上的这个文件设置了什么人可以不用密码而使用rsh、rlogin和rcp命令登录。你可以想想将会发生什么事情。

这其实只是围绕Oracle监听器安全问题的冰山一角而已。其他的还有缓冲区溢出等一大堆问题需要注重。事实上Litchfield的《Oracle黑客手册》里花了一整章的内容来讨论这个主题。

防范此类攻击的方法　　从预防的角度而言，Oracle已经做出了一定措施来更好的保障系统安全，前提是你能够把它实施到位。首先，为监听器设置一个治理密码。对于担负着治理不断增加的密码重担的治理员而言，这看起来像是多余的，不过在你需求其他途径来保障监听器安全之前，好好地想想上面提到的和没提到的威胁。Oracle也添加了ADMIN_RESTRICTIONS，能够阻止特定的远程控制事件。

6. 权限提升　　简单的说，“权限提升”包括使用现有的低权限帐户，利用巧取、偷窃或非法的方式获取更高的权限，甚至是数据库治理员的权限。　　下面举个使用CREATE ANY权限的例子。假设我能通过一个拥有CREATE ANY TRIGGER权限的用户CYW访问数据库，这样我就能在任意的模式里创建触发器。假如你能追踪到一个任何用户都能执行写入操作的表，你在SYSTEM里创建了一个能够在低权限的你对该可写表进行插入或更新操作时执行的触发器。你编写的触发器会调用一个存储过程(也是你自己编写的)，该存储过程会使用AUTHID CURRENT_USER为调用者授权。这就意味着，当该触发器运行“你”的存储过程时，拥有SYSTEM的权限。现在你的非法存储过程内部，包含了“EXECUTE IMMEDIATE "GRANT DBA TO CYW"”。这样我就可以在触发器运行的时候插入到我的公共表里，该触发器由SYSTEM所有，而SYSTEM会调用我的change_privileges存储过程，这个存储过程使用AUTHID CURRENT_USER为我授权。这样“我”就可以在不改变我自身权限的情况下

获得并执行SYSTEM的权限。

防范此类攻击的方法　　数据库治理员该怎么应对这个问题呢?首先，你应该审核数据库的CREATE ANY权限，删除其中不需要的那些部分。其次，看看类似于www.securityfocus.com这类的论坛，看看关于权限提升的最新漏洞。最后，激活对某些特定类型数据库活动的审计功能并没有什么坏处，这样数据库就能让你实现自我保护。当数据库自行审核类似于GRANT DBA这样的事件时，你可以通过查看审计日志知道有没有出现恶意或突发的活动

7. 操作系统**指令和安全**　　黑客并不总是通过shell命令行提示符登录到你的系统的。 不过，通过诱使Oracle数据库运行操作系统水平的指令，我们的确给黑客提供了一条运行指令的有效途径。这些指令能够删除和破坏文件、改写日志(以便隐藏他们的行踪)、创建帐户，以及其他一些能通过命令行输入指令达成的操作。他们是怎么做到的呢?尽管方法有很多，最轻易的一种就是通过Java和PL/SQL这些程序语言。通常可以利用创建外部存储过程的能力，使之执行一个具备系统调用功能的存储程序。这个系统调用指令能够以首次安装时使用的oracle帐户权限执行。

防范此类攻击的方法　　虽然Oracle在保护用户免受此类攻击上已经取得了一定进展，不过你最好还是把希望寄托在你的预防监测工作上。严密留意你的系统内部有没有出现这类活动，当有攻击者试图对你使用此类恶意攻击时，你最好能够事先把握主动权。

8. 文件系统安全　　对文件系统(filesystem)的访问是一个让你头大的棘手问题。“oracle”操作系统用户拥有所有Oracle软件和数据库数据文件的访问权限，所以假如数据库内部的某些用户利用UTL_FILE包访问filesystem上的文件时，他们就可以访问之前由于权限和角色限制而无权访问的很多数据库内部文件。

防范此类攻击的方法　　Oracle引入DirectorY对象在防止此类攻击上也有一定作用。在10g系统中，必须通过DIRECTORY对象来定义某些类型的读写操作。这意味着用户必须拥有CREATE DIRECTORY权限，而在前面介绍的权限提升问题中，我们已经看到可以通过很多方法获取这种权限。即使这些也被解决了，还是有很多方法可以通过PL/SQL或Java语言来获取对filesystem的访问权限和对文件的读写权限。

总论:　　就像上面讨论的一样，Oracle数据库产品有很多漏洞，有时候看起来就像由一些聪明透顶的工程师建造的一所豪宅，工程师固然聪明，但比那些觊觎此宅的黑客们忠厚老实多了。因此，他们没有预料到有人会利用这种种方法来偷砖窃瓦削弱豪宅的根基。黑客可以通过很多不同的方法进行攻击，侵入到目标数据库。　　不过，只要数据库治理员能够花点时间和精力来解决，其中很多问题都是可以避免的。Oracle已经针对很多漏洞在数据库内部打上了补丁，而且入侵监测系统能体构额外的安全保障。所以数据库治理员应该对每一种漏洞都铭记在心，警惕性才是防范的要害，尽量执行好自己制定的安全计划。

可预见的Oracle应用程序性能调优(转贴)

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

这篇技巧性文章是由“国际Oracle用户组”（IOUG）提供的，它是一个由用户组成的组织，这个组织通过提供高质量的信息、培训、网络和支持，来提高Oracle数据库专家和数据库开发者的水平。这篇文章摘自由David Welch所写的论文《可预见的Oracle应用程序性能调优》。点击这里成为“国际Oracle用户组”的一员，从而获得成千上万的由Oracle用户写的技巧性文章和科技文献。

引言

我们见到过很多带有巨大性能问题的Oracle应用程序和电子商务套件安装。我们得出的结论是：这些安装都可以在性能方面取得进一步的提升。换句话说，性能已经很高，几乎不能得到再得到改善的安装是很少见的。

有争议的问题

针对产品系统堆栈而言，我们的底部端对端性能调优方法总是很快产生成果，比我们认为的遵循广泛的备忘列表要快。我提出以下一些问题共讨论：

大部分性能改善的可能性都是在应用程序级上：这条结论来自Metalink上关于性能调优的一个显著的注释。这条结论和我们的经验性能调优系统堆栈没有统计意义上的关系。

平均需要两天的时间：这是书上做出的结论。但我们的经验不支持这个结论。我认为得出一个Oracle应用程序性能改善的策略最少应该需要12天。第一天早晨开会是很常见的事。最后两天主要用来完成行政方面和技术级上的有关发现、胜利和紧接着的推荐的文档工作。可以夸张地说，如果一个性能改善不被记录下来形成文档，那么以后很难再重复类似的性能改善。如果对出现的问题不记录下来形成文档，那么很可能它会再次发生。如果一个问题及其解决方法不被记录下来形成文档的话，对它的监测将非常困难。

扩展碎片：对于联机事务处理系统，这应该不是一个问题。我们听过很多有关“联机事务处理系统”对碎片严重的表（这些表完全是键值惟一的）进行事务处理不会影响性能的说法。但是，我们应该经常性地重组以消除碎片，这会带来性能上的巨大改善。Oracle存储管理改善正在向将碎片带来的影响最小化大踏步地迈进。

由于缓冲输入输出不是大问题，所以需要对磁盘输入输出进行性能调优：这里有两点需要说明。磁盘输入输出的实际开销并不是内存缓冲输入输出的一万倍。真实的比值接近70。即使你的CPU似乎正在抵销这个代价，并且不带来任何显著的性能问题，但是这个问题显然会限制你的系统的可伸缩性。随着时间的流逝，我们越来越重视过高的内存缓冲输入输出，同时找寻性能改善的机会。

OATablespace模型和迁移工具集：已发布的Metalink注释（10/03）声称“这个新模型带来了实时性能改善。”这个模型的概念是将100多个Oracle应用程序表空间合并成一个以10计数的表空间。这会带来潜在的存储空间节省么？或许。这会带来更高的操作效率么？它依赖于其他东西。我们还没有讲解这个工具集。但是我们已经理解了在白板级上的表空间合并是如何改善性能的。

对你的个人电脑客户端进行磁盘碎片整理：在这本书中有关这个问题的讨论很多。这或许是正确的，因为在写作本书时正流行“胖客户端”。但是现在，Oracle应用程序客户端是一个“瘦客户端”（从Oracle废除Jinitiator开始，我们称浏览器为瘦客户端），不要期待能从对你的个人电脑客户端硬盘驱动器进行磁盘碎片整理中得到性能提升。

载入模块补丁：这是Oracle技术支持对于性能问题经常给出的对策，其实在很多情况下，它并不合适。原因是打补丁经常会带来不稳定性。如果对于补丁的依赖性没有给予充分考虑，你可能会发现你不得不载入整个补丁包，而你根本就没打算载入它们，结果就是对你系统的堆栈稳定性产生了影响。

项目管理

项目管理是很关键的。Oracle应用程序性能实施即是技术上的也是行政上的。某个人必须出来做掌舵者，即项目管理者。必须按功能区分出不同的优先次序。如果有可能，可以按照以下方式：商业单位先计算他们选拔人才的时间延迟带来的财政开支，然后乘上用户的数量及其每分钟的收入。获得应用程序性能改善的开销之一就是要记录文档。同时，也需要记录大量的纸质文档。用户的欲望必须被管理起来，因为并不是所有的区域都会产生同样戏剧性的结果。必须有一个管理者来划分不同的优先次序，有些时候甚至需要对性能团队的访问进行

过滤。一方面，用户会频繁地提出会导致底层性能问题的主意和要求。另一方面，和用户进行交互可能会妨碍你的工作进度。成功也会导致暴露下一层性能问题的出现。

什么是用户不能告诉你的

针对某个用户的从底向上的方法揭示了一个单独的包消耗的输入输出资源占全部的25％左右。对另一个用户而言，一个单独的查询可能会引起每周4.3TB的缓冲输入输出。性能调优使得缓冲开销降至原先的0.06％。问题是它会耗尽CPU资源，同时，在那种情况下，是否对CPU进行扩充还需慎重考虑。没有人知道系统堆栈正在抵销这个代价。

关于性能调优保守最严密的一个秘密在Oracle性能调优指南中被发现的。作为一个团队，我们发现这个秘密已经多年了。对于beta级或产品系统的性能问题，你应该从系统的最底层堆栈开始诊断。不幸的是，性能诊断经常仅仅集中在系统堆栈中间的四个部分。它们是：

* 逻辑数据库结构
* 数据库操作
* 访问路径（SQL）
* 内存分配

但是，我们经常可以在Oracle底层的几个级别上发现很大的性能问题，如下所示：

* 输入输出和物理数据库结构
* 资源竞争
* 底层操作系统平台

藏宝图

在Oracle性能调优级上，藏宝图就是v$sqlarea视图。如果我是一个IT管理者，我将会记住这个视图的名字。并且，每当我在大厅遇见我的数据库管理员时，我都会问他们这周他们查询这个视图的次数。

Metalink 注释 235146.1给出了对这个视图进行查询的一些样例。例如：

```
select sql_text, executions, buffer_gets, disk_reads, rows_processed,
sorts, address, first_load_time, HASH_VALUE, module
from v$sqlarea
where executions > 0
order by reads_per desc
```

最近，越来越多的Oracle 9i版本加入了模块（MODULE）这个列，该列揭示了Oracle应用程序的模块名称。

统计包

在很多大型企业中，统计包的使用仍然被忽视。这可能是带有胁迫性的报道。不要犯试图仅仅读取输出结果，就能获取所有信息的错误，即使是第一页就足以告诉你这份报道中剩下的你应该重视的10％在哪儿。Oracle 9.2版本的统计包，现在包含CPU和消耗时间列。以前，为了将长时间运行的SQL语句排序到最顶端，我们不得不开启“追踪”，连接追踪文件，并将它们交付程序tkprof来处理。对于那些一个简单的“追踪”就要处理多达10GB数据的大型企业而言，这是不现实的。

让用户参与到性能调优中去

将这条建议（即，让用户参与到性能调优中去）写入书中的人应该因其创造性而得到赞誉。让你的用户也参与到性能诊断中去。购买一台Oracle应用程序评测个人电脑，并把它给用户使用。不要使用与个人电脑类似的配置好的笔记本，因为在同样规范的情况下，笔记本没有个人电脑的同样性能特性。配置清单如下：

* 750 MB CPU
* 256 MB 内存
* Windows 2000 企业版（第四版）
* 使用独立的逻辑磁盘
* Jinitiator- 锁定版
* 标准软件，例如Office 2003

供评测用的个人电脑不需要以下配置:

* 墙纸
* 屏幕截图
* 工具条
* 常驻程序

将评测用个人电脑送上用户的桌面，带着性能问题。将用户的电脑接入局域网，让用户工作一段时间。然后，再将用户的电脑放进计算机房间，并把它接入中间层，让用户在它上面进行更多的工作。评测用个人电脑消除了用户方对Oracle应用程序性能的主观性，同时也消除了面对用户抱怨性能问题你们的主观性。

索引计数和性能

回到70年代，开发者指南基本上说不要在一个表上建立4到5个索引。今天，开发者指南上的注释如下:

Oracle不限制在一个表上建立索引的个数。尽管如此，你需要考虑索引所带来的性能改善，以及你的数据库应用程序的实际需要，从而决定需要对哪些列建立索引。

事实是：每个Oracle应用程序表可能包含30多个索引。如果我们加入一个索引能将经常需要的SQL语句的输入输出减少，我们会不考虑高索引计数的问题而加入这个索引。

CPU

减小并发管理池的宽度，至今我们还没发现这会阻塞任务的进行。我们经常会看到的情景是：减小并发管理池的宽度实际上增加了批处理任务的吞吐量，它也使CPU不那么忙碌。有许多包含对等进程的任务必须被完成。如果一个任务的池宽度过窄，所需的任务可能永远也得不到处理，从而阻塞整体任务。

我们和Oracle应用程序安装小组、培训者打过交道，他们喜欢增加并发管理池的宽度，而无视对CPU的影响，这种设置一直保持到产品发布时仍然存在。在训练和测试环境中，安全问题的大门是开着的，并且安装者增加并发管理池的宽度以期望他们的批处理任务可以尽早完成。他们这样做或许根本没有考虑到对CPU的影响，CPU可能会因此而被完全占用。

CPU运行队列不应该比你的CPU计数的两倍还深。如果CPU在一天中被经常性完全占用，就必须放弃某些设置。寻找这个需要被放弃的设置的第一位置就应该是并发管理池。

总结:
oracle日常维护和性能调优,不是单纯的技术,指定科学严谨的管理维护计划更重要,一定要将调优,维护过程中的所有为难题记录,形成文档,在知识经验上得到积累,不至于同样的错误犯两次;
记录运行日志,什么时候系统性能差,速度慢;然后分析找出原因,指定解决的办法;
调优分两部分:
一.应用层,包括逻辑数据库结构,数据库操作,访问路径（SQL）,内存分配等.优化的方法有,分解大表,修改关键表结构,分析应用层的sql语句,优化,使之达到最优执行;配置参数,恰当地分配内存;定期分析,重建索引,移动表,消除碎片;
二.系统层,包括输入输出和物理数据库结构,资源竞争,底层操作系统平台等;根据系统应用的规模,选择恰当的文件系统,这样可以达到减少io操作的次数;操作系统是支撑大规模的吞吐量,window是微内河,linux/unix是宏内核,造成了在系统内进程间通信的速度和操作性能的差异等.

根据需求->指定运维计划->分析运行日志->更该运行计划->分析运行日志....这样一个反复的过程.

快速修复oracle参数文件的另类方法

文章整理: www.diybl.com 文章来源: 网络 去论坛 建我的blog

DBA的悲哀莫过于没有备份好文件。 eygle称之为DBA的恶梦。此言甚是！

尽管很多人认为对参数文件的备份并不重要，但你往往就因此吃亏！

参数文件，10.2.0 windows版本oracle的spfile和pfile默认在E:\oracle\product\10.2.0\db_1\database目录下，SPFILEsid.ORA和INITsid.ORA,oracle默认用spfile,若spfile损坏，则自动用pfile，如果 两个都坏了，则提示错误。如果没有备份，那怎么办呢?

当然，你可以找到oracle自带的init模板，一个个参数地设置自己系统的参数文件。那这将是一件很糟的事情，它会浪费你宝贵的时间。有什么办法吧？关键是要快速的！

正如题所示，我有个好办法……

从alert_alaska.log警告日志里着手，因为它记录着一直以来数据库运行的情况，当然也包括每次启动的参数信息啦，我们要的就是

```
processes                = 150
__shared_pool_size       = 75497472
__large_pool_size        = 4194304
__java_pool_size         = 4194304
__streams_pool_size      = 0
nls_language             = AMERICAN
nls_territory            = AMERICA
sga_target               = 167772160

control_files            = E:\ORACLE\PRODUCT\10.2.0\ORADATA\ALASKA\CONTROL01.CTL, E:\ORACLE\PRODUCT\10.2.0\ORADATA\ALAS
db_block_size            = 8192
__db_cache_size          = 79691776
compatible               = 10.2.0.1.0
db_file_multiblock_read_count= 16
db_recovery_file_dest    = e:\oracle\product\10.2.0/alash_recovery_area
db_recovery_file_dest_size= 1073741824
log_checkpoints_to_alert = TRUE
undo_management          = AUTO
undo_tablespace          = UNDOTBS1
undo_retention           = 900
remote_login_passwordfile= EXCLUSIVE
db_domain                = com.cn
dispatchers              = "(PROTOCOL=TCP) (SERVICE=alaskaXDB)"
                                   #用 " " 引起它们
job_queue_processes      = 10
audit_file_dest          = E:\ORACLE\PRODUCT\10.2.0\ADMIN\ALASKA\ADUMP
background_dump_dest     = E:\ORACLE\PRODUCT\10.2.0\ADMIN\ALASKA\BDUMP
user_dump_dest           = E:\ORACLEPRODUCT\10.2.0ADMIN\ALASKA\UDUMP
core_dump_dest           = E:\ORACLE\PRODUCT\10.2.0\ADMIN\ALASKA\CDUMP
db_name                  = alaska
open_cursors             = 300
pga_aggregate_target     = 16777216
```

复制以前成功启动的参数语句(如上代码段)到一个文本中，只需把dispatchers = (PROTOCOL=TCP) (SERVICE=alaskaXDB)的值加上""（单引号），变成dispatchers = "(PROTOCOL=TCP) (SERVICE=alaskaXDB)"，保存为c:\pfile.txt

然后运行，startup pfile="c:\pfile.txt"; 即可

是不是很快速，很另类呢！如你所愿。

——EOF——