ORACLE 数据库实用指南附录

赵元杰

2002 年 11 月 15 日

目　　录

附录 A：SQL 及 SQL*PLUS 常用命令参考

按字母顺序排列：

%FOUND

功能：判断光标属性

参见：%ISOPEN, %NOTFOUND, CURSOR, %ROWCOUNT

语法：c1%FOUND 或 SQL%FOUND

c1 为PL/SQL 光标名字

例子：

```
PL/SQL
LOOP
FETCH loan_cursor INTO customer_name, loan_amount;
IF loan_cursor%FOUND THEN
approved_loan := loan_amount * 0.80;
ELSE
EXIT;
END IF;
END LOOP;
INSERT INTO approved_loan VALUES (customer_name,
loan_amount);
IF SQL%FOUND THEN
COMMIT;
ELSE
ROLLBACK;
END IF;
```

%ISOPEN

功能：判断光标是否**已打开**

参见： %FOUND, %NOTFOUND, CURSOR, %ROWCOUNT

语法：c1%ISOPEN 或 SQL%ISOPEN

c1：为 PL/SQL 光标

例子：

```
PL/SQL
IF loan_cursor%ISOPEN THEN
FETCH loan_cursor INTO customer_name,
loan_amount;
ELSE
OPEN loan_cursor;
END IF;
```

%NOTFOUND

功能：判断光标所指是否没有记录

参见：%ISOPEN, %FOUND, CURSOR, %ROWCOUNT

语法：c1%NOTFOUND 或 SQL%NOTFOUND

变量：

c1：有效的 PL/SQL 光标

例子：

```
PL/SQL
LOOP
FETCH loan_cursor INTO customer_name,
loan_amount;
EXIT WHEN loan_cursor%NOTFOUND
END LOOP;
INSERT INTO approved_loan VALUES (customer_name,
loan_amount);
IF SQL%NOTFOUND THEN
ROLLBACK;
ELSE
COMMIT;
END IF;
```

%ROWCOUNT

功能：判断光标已经取出的记录数

参见：%ISOPEN, %FOUND, %NOTFOUND, CURSOR

语法：c1%ROWCOUNT 或 SQL%ROWCOUNT

变量：

c1：有效的 PL/SQL 光标。

例子：

```
PL/SQL
DECLARE
CURSOR c1 IS SELECT customer_name, loan_amount FROM
   Loan  ORDER BY loan_amount DESC;
cust_name VARCHAR2(50);
loan_amt NUMBER(5,2);
BEGIN
OPEN c1;
LOOP
FETCH c1 INTO cust_name, loan_amt;
EXIT WHEN (c1%ROWCOUNT > 5) OR
(c1%NOTFOUND);
INSERT INTO loan_approved VALUES
(cust_name, loan_amt);
COMMIT;
END LOOP;
CLOSE c1;
END;
UPDATE approved_loan SET approved_flag = 'Y' WHERE
loan_amount < 5000;
IF SQL%ROWCOUNT > 10 THEN
INSERT INTO approvals VALUES ('STOP MORE
APPROVAL');
END IF;
```

%ROWTYPE

功能：记录类型，用于对表的**所有列**进行选择。

参见：%TYPE, FETCH

语法： ([user.]table | cursor)%ROWTYPE

变量：

user: 用户名
table : 要访问的表或视图名
cursor: 任何有效的 PL/SQL 光标

例子：

```
DECLARE
CURSOR loan_cursor IS SELECT customer_name, loan_amount
FROM loan;
loan_record loan_cursor%ROWTYPE
BEGIN
OPEN loan_cursor;
LOOP
FETCH loan_cursor INTO loan_record;
EXIT WHEN loan_cursor%NOTFOUND;
IF loan_record.loan_amount < 5000 THEN
UPDATE loan
SET loan_approval = 'Y'
WHERE customer_name =
loan_record.customer_name;
IF SQL%FOUND THEN
COMMIT;
END IF;
END IF;
END LOOP;
CLOSE loan_cursor;
END;
```

%TYPE

功能：表的列的类型，用于对某个变量类型的定义，该变量的类型随所指定表的列类型变化而变化。

参见：

%ROWTYPE, FETCH

语法：

```
 ( [user.]table.column | variable )%TYPE
```

变量：

user: 用户名
table or view: 要访问的表名或视图名
column: 表名或视图的列名
variable: 任何有效的PL/SQL 变量

例子：

```
PL/SQL
new_balance  old_balance%TYPE;
customer_name  loan.customer_name%TYPE;
```

(+)

功能：外连接符号，用于对所连接表的具有不同记录数情况的连接操作。

参见：

Joining Tables PRIOR, CONNECT BY

语法：

```
WHERE table1.column1 = table2.column1 (+)
```

例子：

```
SQL
SELECT SALES_AGENT, MANAGER FROM SALESMEN, MANAGER
WHERE SALESMEN.MANGER = MANGER.MANGER(+);
SALES_AGENT MANAGER
---------- --------
JAMES BOND M
SADDAM HUSSAIN
MICHAEL JORDAN PHIL JACKSON
```

@ (“at” 号)

参见：

@@, CREATE DATABASE LINK

语法：

```
@ filename[.ext] [arg_1,...,arg_N]
```

例子：

SQL
```
@PRINT_LOAN_RPT
@PRINT_RESULTS.QRY
```

@@

参见：

@

语法：

```
@@ filename [.ext]
```

例子：

SQL
```
@@PRINT_LOAN_STATUS.QRY
@@PRINT_RESULTS.QRY
```

ABS

参见：

SIGN

语法：

```
ABS (n)
```

变量：

n：一个数字变量

例子：

PL/SQL
```
Var1:= ABS (-1);
```
SQL
```
SELECT ABS (-1) “Absolute Value” FROM DUAL;
Absolute Value
-------
1
```

ACCEPT

参见：

INPUT

语法：

ACCEPT variable [format] [default] [PROMPT]

例子：

SQL
```
ACCEPT loan NUMBER FORMAT ‘99999.99’ DEFAULT ‘0.00’ PROMPT
‘Enter weekly salary: ‘
ACCEPT last_name CHAR FORMAT ‘35’ PROMPT “Enter customer’s last
name: “
```

ACOS

参见：

ASIN, ATAN, and ATAN2

语法：

ACOS (n)

变量：

n：-1到1 的数字变量

例子：

PL/SQL
```
Var1:= ACOS (-1);
```
SQL
```
SELECT ACOS (-1) “Arc Cosine” FROM DUAL;
Arc Cosine
----------
3.1415927
```

ADD_MONTHS

参见：

MONTHS_BETWEEN

语法：

ADD_MONTHS (d, n)

变量：

d：有效的日期型变量
n：整型变量

例子：

PL/SQL
Date1:= ADD_MONTHS (‘26-JAN-47’, 10);
SQL
SELECT ADD_MONTHS (‘26-JAN-47’, 10) “Add Months” FROM DUAL;
Add Month

26-NOV-47

ALTER CLUSTER

参见：

CREATE CLUSTER,
DROP CLUSTER

语法：

ALTER CLUSTER [user.]cluster
[INITRANS n]
[MAXTRANS n]
[PCTFREE n]
[PCTUSED n]
[SIZE n [K | M]]
[STORAGE n]
[ALLOCATE EXTENT [SIZE n [K | M]]]
[DATAFILE ‘filename’]
[INSTANCE n]

变量：

user：建立cluster 的用户名
n：任意正整数
filename：文件名

例子：

SQL
ALTER CLUSTER customer PCTFREE 12

ALTER DATABASE

参见：

CREATE DATABASE,DROP DATABASE

语法：

```
ALTER DATABASE [database]
[ADD LOGFILE
[THREAD n] [GROUP n]
file_definition [.file_definition] ... |
ADD LOGFILE MEMBER
File [REUSE] [, file [REUSE] ...]
TO [GROUP n |
(file [, file] ...) |
file
file [REUSE] [, file [REUSE] ...]
TO [GROUP n |
(file [, file] ...) |
file) ] .. |
DROP LOGFILE
{ GROUP n | (file [,file] ... ) | file}
[GROUP n | (file [,file] ... ) | file] ... |
DROP LOGFILE MEMBER file [, file] |
RENAME FILE file to file |
NOARCHIVELOG | ARCHIVELOG |
MOUNT [EXCLUSIVE | PARALLE] |
OPEN [RESETLOGS | NORESETLOGS] |
ENABLE [PUBLIC] THREAD n |
DISABLE THREAD n |
BACKUP CONTROLFILE TO file [REUSE]
DATAFILE file [ONLINE | OFFLINE [DROP]] |
CREATE DATAFILE file[,file]
AS file_definition [.file_definition] ... |
RENAME GLOBAL_NAME TO database [.domain] |
```

```
RECOVER recover_clause |
SET [DBMAC [ON | OFF] | DBHIGH = string | DBLOW =
string ]
```

例子：

SQL

```
ALTER DATABASE NO MOUNT ;
ALTER DATABASE ADD LOGFILE GROUP 2 ‘customer.002’ size 2m;
```

ALTER FUNCTION

参见:

CREATE FUNCTION,DROP FUNCTION

语法：

```
ALTER FUNCTION [user.]function COMPILE
```

变量：

user：建立函数的用户名
function：数据库中函数名

例子：

SQL

```
ALTER FUNCTION loan_calculation COMPILE;
ALTER FUNCTION credit_points COMPILE;
```

ALTER INDEX

参见：

CREATE INDEX, DROP INDEX

语法：

```
ALTER [UNIQUE] INDEX [user.]index
[INITRANS n]
[MAXTRANS n]
[STORAGE n]
```

变量：

user：建立索引的用户名

index：数据库中的索引名

例子：

SQL
```
ALTER INDEX loan_application;
```

ALTER MATERIALIZED VIEW

参见：

CREATE MATERIALIZED VIEW

语法：

```
ALTER MATERIALIZED VIEW  [user.]mview
```

例子：

```
CREATE MATERIALIZED VIEW hq_emp
REFRESH COMPLETE
START WTIH SYSDATE NEXT SYSDATE +1/4096
AS SELECT * FROM hq_emp;

ALTER MATERIALIZED VIEW hq_emp
REFRESH FAST;
```

ALTER MATERIALIZED VIEW LOG

参见：

CREATE MATERIALIZED VIEW

语法：

```
ALTER  MATERIALIZED VIEW LOG ON [schemal.] table
```

例子：

```
ALTER MATERIALIZED VIEW LOG ON dept
STORAGE MAXEXTENTS 50;
```

ALTER PACKAGE

参见：

CREATE PACKAGE,DROP PACKAGE

语法：

```
ALTER PACKAGE [user.]package COMPILE [PACKAGE | BODY]
```

变量：

user：建立包的用户名
package：数据库中包名字

例子：

SQL
```
ALTER PACKAGE loan_processing;
```

ALTER PROCEDURE

参见：

CREATE PROCEDURE, DROP PROCEDURE

语法：

```
ALTER PROCEDURE [user.]procedure COMPILE
```

变量：

user：建立包的用户名
procedure：数据库中的存储过程名

例子：

SQL
```
ALTER PROCEDURE loan_calculation COMPILE;
ALTER PROCEDURE credit_points COMPILE;
```

ALTER PROFILE

参见：

CREATE PROFILE, DROP PROFILE

语法：

```
ALTER PROFILE profile LIMIT
[ { SESSIONS_PER_USER |
CPU_PER_SESSION |
CPU_PER_CALL |
CONNECT_TIME |
IDLE_TIME |
LOGICAL_READS_PER_SESSION |
LOGICAL_READS_PER_CALL |
COMPOSITE_LIMIT } { n | UNLIMITED |
DEFAULT } |
PRIVATE_SGA { n {K | M} | UNLIMITED |
DEFAULT } ]
```

变量：

profile：想要改变的数据库profile名字
n：任意整数值

例子：

SQL

```
ALTER PROFILE customer LIMIT SESSION_PER_USER 12 ;
ALTER PROFILE customer LIMIT CONNECT_TIME 600 ;
```

ALTER RESOURCE COST

参见：

CREATE PROFILE

语法：

```
ALTER RESOURCE COST
[ CPU_PER_SESSION n |
CONNECT_TIME n |
LOGICAL_READS_PER_SESSION n |
PRIVATE_SGA n]
```

变量：

n：any integer value

例子：

SQL

```
ALTER PROCEDURE loan_calculation COMPILE;
ALTER PROCEDURE credit_points COMPILE;
```

ALTER ROLE

参见：

CREATE ROLE, DROP ROLE

语法：

```
ALTER ROLE role
[ NOT IDENTIFIED | IDENTIFIED
[ BY PASSWORD | EXTERNALLY ] ]
```

变量：

role：需要改变的数据库中的角色

例子：

SQL

```
ALTER ROLE custom_user;
```

ALTER ROLLBACK SEGMENT

参见：

CREATE ROLLBACK SEGMENT,
DROP ROLLBACK SEGMENT

语法：

```
ALTER [PUBLIC] ROLLBACK SEGMENT rollback_segment
[STORAGE storage]
```

例子：

SQL

```
ALTER PUBLIC ROLLBACK SEGMENT humanresources;
ALTER ROLLBACK SEGMENT insurance;
```

ALTER SEQUENCE

参见：

CREATE SEQUENCE, DROP SEQUENCE

语法：

```
ALTER SEQUENCE [user.]sequence
[ INCREMENT BY n]
[ MAXVALUE n | NOMAXVALUE]
[ MINVALUE n | NOMINVALUE]
[ CYCLE | NO CYCLE]
[CACHE n | NO CACHE]
[ORDER | NO ORDER]
```

变量：

user：建立sequence 的用户名
sequence：需要改变的数据库中sequence
n：任意整数值

例子：

SQL

```
ALTER SEQUENCE customer_code_seq
INCREMENT BY 1;
ALTER SEQUENCE loan_application_seq
NOMAXVALUE ;
```

ALTER SESSION

参见：

SET ROLE, ALTER SYSTEM

语法：

```
ALTER SESSION
[ SET parameter
CLOSE DATABASE LINK link |
ADVICE [ COMMIT | ROLLBACK | NOTHING ] |
[ ENABLE | DISABLE | COMMIT IN PROCEDURE ]
```

变量：

parameter: 可以设置的下列命令：

```
 [ SQL_TRACE [TRUE | FALSE]
[ NLS_LANGUAGE = language ]
[ NLS_TERRITORY = territory ]
[ NLS_DATE_FORMAT = format ]
[ NLS_DATE_FORMAT = language ]
[ NLS_NUMERIC_CHARACTERS = string]
[ NLS_ISO_CURRENCY = territory]
[ NLS_CURRENCY = string]
[ NLS_SORT = sort]
[ LABEL = [string | DBHIGH | DBLOW | OSLABEL] ]
[ MLS_LABEL = format ] |
```

例子：

SQL

```
ALTER SESSION ENABLE COMMIT IN PROCEDURE
ALTER SESSION DISABLE COMMIT IN PROCEDURE
```

ALTER SNAPSHOT

参见：

CREATE SNAPSHOT,
DROP SNAPSHOT

语法：

```
ALTER SNAPSHOT [user.] snapshot
[PCTFREE n |
PCTUSED n |
INITRANS n |
MAXTRANS n |
STORAGE n]
[ REFRESH { FAST | COMPLETE | FORCE}
[START WITH start_date] [NEXT
next_date]
```

变量：

user：建立 snapshot的用户名
n：任意的正整数值
start_date：启动snapshot 进程的开始日期
next_date：刷新snapshot的日期
snapshot：需要改变的数据库中 snapshot 的名字

例子：

SQL
```
ALTER SNAPSHOT customer.inactive_cust_snapshot
[ PCTUSED 12]
```

ALTER SNAPSHOT LOG

参见：

CREATE SNAPSHOT LOG,
DROP SNAPSHOT LOG

语法：

```
ALTER SNAPSHOT LOG ON [user.] table
[PCTFREE n |
PCTUSED n |
INITRANS n |
MAXTRANS n |
STORAGE n]
```

变量：

n：任意正整数值
table：需要改变的与snapshot 日志有关的主表

例子：

SQL
```
ALTER SNAPSHOT LOG ON inventory
PCTUSED 12
```

ALTER SYSTEM

参见：

COMMIT, ROLLBACK,
SAVEPOINT,
SET TRANSACTION

语法：

```
ALTER SYSTEM
[ SET { RESOURCE_LIMIT = { TRUE | FALSE } |
MTS_SERVERS = n
MTS_DSIPATCHERS = ‘protocol.n’
```

```
SWITCH LOGFILE |
{ CHECKPOINT | CHECK DATAFILES } { GLOBAL | LOCAL } |
{ ENABLE | DISABLE }
{ DISTRIBUTED RECOVERY | RESTRICTED SESSION } |
ARCHIVE LOG archive_log_clause
FLUSH SHARE_POOL
KILL SESSION ‘sid, serial number’ ]
```

变量：

archive_log_clause：归档日志项
sid：希望杀掉（ KILL）的会话的sid号
serial number：希望杀掉（ KILL）的会话的序列号

例子：

SQL
```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE
```

ALTER TABLE

参见：

CREATE TABLE, DROP TABLE

语法：

```
ALTER TABLE [user.] table
{ [ ADD ( { column1 | table_constraint }
[, column2 | table_constraint} ] ... ) ]
[ MODIFY ( column1, column2) ]
[ DROP drop_constraint]
[PCTFREE n]
[PCTUSED n]
[INITRANS n]
[MAXTRANS n]
[STORAGE n]
ALLOCATE EXTENT
[ SIZE n [K | M]]
[DATAFILE file]
[INSTANCE n]
[ ENABLE | DISABLE]
```

变量：

user：建立表的用户名
table：希望改变的表名

column：列名
table_constraint：表的限制，如 NULL，NOT NULL 等
n：任意正整数值
file：物理数据文件名

例子：

SQL
```
ALTER TABLE customer
( ADD ( address VARCHAR2(50));
ALTER TABLE customer
( MODIFY ( name VARCHAR2(50) NOT NULL);
```

ALTER TABLESPACE

参见：

CREATE TABLESPACE

语法：

```
ALTER TABLESPACE tablespace
[ ADD DATAFILE file_definition
[,file_definition] |
RENAME DATAFILE file [,file] ... TO
file [, file] |
DEFAULT STORAGE storage |
ONLINE | OFFLINE [NORMAL | IMMEDIATE]
|
[ BEGIN | END] BACKUP ]
```

变量：

tablespace：希望改变的数据库表空间名
file_definition：带大小（K or M）的文件名

例子：

SQL
```
ALTER TABLESPACE customer
 ADD DATAFILE ‘customer02.dat’  size  10m;
```

ALTER TRIGGER

参见：

CREATE TRIGGER, DROP TRIGGER

语法：

```
ALTER TRIGGER [user.]trigger [ENABLE | DISABLE |
COMPILE]
```

变量：

user：建立触发器的用户名
trigger：希望改变的数据库中的触发器

例子：

SQL
```
ALTER TRIGGER loan_calculation COMPILE;
ALTER TRIGGER credit_points COMPILE;
```

ALTER TYPE

参见：

CREATE TYPE, CREATE TYPE BODY, DROP TYPE

语法：

```
ALTER TYPE schema.type_name
{ COMPILE { SPECIFICATION | BODY } |
REPLACE AS | AS TABLE | AS OBJECT }
{ VARRAY (size) | VARYING ARRAY (size) }
{ OF datatype}
{ REF object_type_name}
{ MAP | ORDER MEMBER function_specification }
{ PRAGMA RESTRICT_REFERENCES
function_specification restriction}
```

变量：

schema：建立类型的数据库用户名
type_name：希望改变的对象名
size： VARRAY 大小的上限
datatype：任意类型，如 CHAR, DATE, NUMBER等
object_type_name：对象类型引用的名字

function_specification：成员函数名
restriction：限制类型，如 WNDS, WNPS, RNDS, RNPS

例子：

SQL
```
ALTER TYPE loan_obj AS OBJECT
(customer_name CHAR(20),
loan_amount NUMBER(5),
MEMBER FUNCTION get_amount RETURN NUMBER,
MEMBER FUNCTION get_loan_approved RETURN NUMBER,
pragma RESTRICT_REFERENCES (get_amount, WNDS));
```

ALTER USER

参见：

CREATE USER, DROP USER

语法：

```
ALTER USER user [IDENTIFIED [BY password | EXTERNALLY]]
[ DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[,QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[PROFILE profile]
[DEFAULT ROLE (role1, role2, ...) | ALL EXCEPT
(role1, role2, ...) |
NONE]
```

变量：

user：希望改变的数据库用户名
tablespace：用户建立的表空间名
n：任意正整数值
profile：用户要使用的profile名
role：用户指定（或不指定）的角色

例子：

SQL
```
ALTER USER customer
DEFAULT tablespace cust_tablespace
TEMPORARY tablespace temp;
```

ALTER VIEW

参见：

CREATE VIEW, DROP VIEW

语法：

```
ALTER VIEW [user.]view COMPILE
```

变量：

user：建立视图的用户名
view：希望改变的数据库的视图名

例子：

SQL
```
ALTER VIEW loan_calculation COMPILE;
ALTER VIEW credit_points COMPILE;
```

ANALYZE

参见：

EXPLAIN PLAN, AUDIT

语法：

```
ANALYZE
{ INDEX [user.]index
{ { COMPUTE | ESTIMATE | DELETE } STATISTICS
[ SAMPLE ( n PERCENT | n ROWS } ] |
VALIDATE STRUCTURE } |
{ TABLE [user.]table | CLUSTER [user.]cluster}
{ { COMPUTE | ESTIMATE | DELETE } STATISTICS
[ SAMPLE ( n PERCENT | n ROWS } ] |
VALIDATE STRUCTURE } [CASCASE] |
LIST CHAINED ROWS [INTO [user.]table] }
```

变量：

index：希望分析的索引的名字
n：任意正整数值
table：希望分析的表名
cluster：希望分析的cluster名

例子：

SQL
```
ANALYZE INDEX cust_index ESTIMATE STATISTICS;
```

APPEND

参见：

DEL, EDIT

语法：

A[PPEND] text

变量：

text：在 SQL 缓冲区希望增加的字符串

例子：

SQL
```
A FROM TELECOM
APPEND WHERE SQLCODE = 2
```

ASCII

参见：

CHR

语法：

ASCII (‘x’)

变量：

x: 用单引号引起的字符串

例子：

PL/SQL
```
Var1:= ASCII (‘A’);
```
SQL
```
SELECT ASCII (‘Z’) FROM DUAL;
```

```
ASCII('Z')
----------
90
```

ASIN

参见：

ACOS, ATAN, ATAN2

语法：

ASIN (n)

变量：

n： -1 到 1 的数字变量

例子：

PL/SQL

```
Var1:= ASIN (-1);
```

SQL

```
SELECT ASIN (-1) "Arc Sine" FROM DUAL;
Arc Sine
---------
-1.570796
```

ATAN

参见：

ACOS, ASIN, ATAN2

语法：

ATAN (n)

变量：

n：数字变量

例子：

PL/SQL

```
Var1:= ATAN (-1);
```

SQL

```
SELECT ATAN (-1) “Arc Tangent” FROM DUAL;
Arc Tangent
-----------
-.7853982
```

ATAN2

参见：

ACOS, ASIN, ATAN

语法：

```
ATAN2 (x, y)
```

变量：

n：数字变量

例子：

PL/SQL

```
Var1:= ATAN2 (-1, 1);
```

SQL

```
SELECT ATAN2 (-1, 1) “Arc Tangent2” FROM DUAL;
Arc Tangent2
------------
-.7853982
```

AUDIT

参见：

NO AUDIT, ANALYZE

语法：

```
AUDIT command | ALL
ON [user.]object | DEFAULT
[ BY SESSION | ACCESS ]
[ WHENEVER [NOT] SUCCESSFUL ]
```

变量：

command：可以审计的命令： ALTER, AUDIT, COMMENT, DELETE, EXECUTE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT 及 UPDATE

User：建立对象的用户名
Object：对象名，可以是 table， view， synonym， sequence， procedure，function，package，或 snapshot.

例子：

SQL
```
AUDIT INSERT ON customer WHENEVER NOT SUCCESSFUL;
```

AVG

参见：

Group by SUM

语法：

```
AVG ( [DISTINCT | ALL] n)
```

变量：

DISTINCT：如果多个一样只考虑一个的选项，缺省为DISTINCT
ALL：如果多个一样都考虑一个的选项
N：数字变量

例子：

PL/SQL
```
Var1:= AVG (1, 2, 3);
```
SQL
```
SELECT AVG (DAILY_SALES) “Average Daily Sales” FROM SALES;
Average Daily Sales
--------------
655.12
```

BFILENAME

参见：

LOBs CREATE DIRECTORY

语法：

```
BFILENAME (‘directory’, ‘file’)
```

变量：

directory：服务器中的物理目录名,最长只能30个字符
file：服务中物理文件名

例子：

SQL
```
INSERT INTO lob_table VALUES (BFILENAME (‘lob_directory’,‘file1.bmp’));
```

BLOCK

参见：

LABEL

语法：

```
 [<<label>>]
DECLARE
object_declaration_1,...,object_declaration_n
[subprogram_declaration_1
[,subprogram_declaration_2,...,subprogram_declaration_n]]
BEGIN
statement_1,...,statement_n
[EXCEPTION exception_handler_1 [, exception_handler_2,...,
exception_handler_3]]
END [label];
```

变量：

label：可选，不需声明的PL/SQL 块
object_declaration_1,...,object_declaration_n：声明的对象，可以是：
constant_declaration
variable_declaration
cursor_declaration
cursor_variable_declaration
record_declaration
exception_declaration
plsql_table_declaration

subprogram_declaration_1,...,subprogram_declaration_n：子程序，可以是：
procedure_declaration
function_declaration

statement_1,...,statement_n：一系列语句
exception_handler_1,...,exception_handler_n：例外处理语句

例子：

```
PL/SQL
DECLARE
checking_acct_balance NUMBER;
BEGIN
IF checking_acct_balance = 0 THEN
INSERT INTO checking_acct VALUES (500);
DELETE FROM savings_acct VALUES (500);
END IF;
COMMIT;
END;
DECLARE
total_sales NUMBER;
bonus NUMBER;
salary NUMBER;
emp_id NUMBER;
BEGIN
IF total_sales < 1000 THEN
bonus := salary / total_sales;
ELSE
bonus := total_sales ;
END IF;
INSERT INTO employee VALUES (bonus);
COMMIT;
EXCEPTION
WHEN ZERO_DIVIDE THEN
INSERT INTO employee VALUES (0);
COMMIT;
WHEN OTHERS THEN
ROLLBACK;
END;.
```

BTITLE

参见：

TTITLE, REPHEADER, REPFOOTER

语法：

```
BTI[TLE] [ printspec [ text | variable] ] | [ OFF | ON]
```

变量：

printspec：输出说明，包括：
COL n
S[KIP] [n]
TAB n
LE[FT]
CE[NTER]
R[IGHT]
BOLD
FORMAT text
Text：标题的字符
Variable：包括下列值的变量:
SQL.LNO (当前行号)
SQL.PNO (当前页号)
SQL.RELEASE (当前Oracle版本号)
SQL.SQLCODE (当前错误代码号)
SQL.USER (当前用户)

例子：

SQL
BTITLE RIGHT ‘MARKETING REPORT’
BTITLE OFF
BTITLE ON.

CEIL

参见：

FLOOR

语法：

CEIL (n)

变量：

n：数字变量

例子：

PL/SQL
Var1:= CEIL (147.2754);
SQL
SELECT CEIL (147.2754) “Ceiling” FROM DUAL;

```
Ceiling
---------
148
```

CHANGE

参见：

EDIT

语法：

```
C[HANGE] separating_character old [separating_character [new
[separating_character]]]
```

变量：

separating_character：可分开的非字母数字字符串
old： SQL缓冲区中的旧字符串
new： 新的字符串

例子：

```
SQL
SQL>l
SELECT MARKETING,NAME from emp
SQL>C /MARKETING/SALES
```

CHARTOROWID

参见：

ROWIDTOCHAR

语法：

```
CHARTOROWID (‘x’)
```

变量：

x：单引号引起的字符 (CHAR 或 VARCHAR2 数据类型)

例子：

```
PL/SQL
Var1 := CHARTOROWID (‘ORACLE’)
```

```
SQL
SELECT CHARTOROWID (‘OOOOOOOD.1111.0022’) “ROWID_TYPE” FROM
DUAL;
ROWID_TYPE
------------------
OOOOOOOD.1111.0022
```

CHR

参见：

ASCII

语法：

```
CHR (n)
```

变量：

n: 数字变量

例子：

```
PL/SQL
Var1 := CHR (120);
SQL
SELECT CHR (74) || CHR (65) || CHR (86) || CHR (65) “Hot”
FROM DUAL;
Hot
----
JAVA
```

CLEAR

参见：

Group By BREAK, COLUMN, COMPUTE

语法：

```
CLEAR option
```

变量：

option：可以是下列之一
BUF[FER]

COL[UMNS]
COMP[UTES]
SCR[EEN]
SQL
TIM[ING]

例子：

SQL
```
CLEAR BREAKS
CLEAR BUFFER
CLEAR SCREEN
CLEAR SQL
```

CLOSE

参见：

OPEN, OPEN-FOR, FETCH

语法：

CLOSE {cursor_name | cursor_variable_name | :host_cursor_variable_name};

变量：

cursor_name：希望关闭的光标名
cursor_variable_name：希望关闭的与光标有关变量名
host_cursor_variable_name：PL/SQL中缩主变量名

例子：

PL/SQL
```
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
CLOSE loan_cur;
```

COLUMN

参见：

Titles 和Headings

语法：

```
COL[UMN] [{column|expr} [option_1 ... option_n]]
```

变量：

column：希望设置的列名
expr：有效的SQL表达式
option_1 ... option_N：下列选件之一 ：
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]

例子：

SQL

```
COLUMN SALARY FORMAT $9,999,999.99
COLUMN LAST_NAME FORMAT A35.
```

COMMENT

参见：

REMARK

语法：

```
COMMENT ON [ TABLE table | COLUMN table.column ] IS comment
```

变量：

table：希望加注释的表名
column：希望加注释的列名
comment：注释内容

例子：

SQL
```
COMMENT ON TABLE customer COLUMN cust_name IS this column
is ‘the name of the customer’;
```

COMMIT

参见：

SQL *Plus ROLLBACK, SAVEPOINT, SET
TRANSACTION, LOCK TABLE

语法：

```
COMMIT [WORK ] [COMMENT ‘comment_text’]
```

变量：

WORK：可选关键字.
COMMENT：可以允许加50个字符的注释

例子：

SQL
```
COMMIT;
COMMIT WORK;
COMMIT WORK ‘Saving loan transaction’
```

COMPUTE

参见：

Group By BREAK

语法：

```
COMP[UTE] [function [LABEL] text OF expression|column|alias ON
expression | column | alias | REPORT | ROW]
```

变量：

function：下列函数之一：
AVG, COU[NT], MAX[IMUM], MIN[IMUM], NUM[BER], STD, SUM,
VAR[IANCE]
Text：the text you will specify for the computed value

Expression：the expression you would like to use to compute the value
Column：the column(s) you would like to use to compute the value
Alias：the alias for the column(s) you would like to use to compute the value

例子：

SQL
```
COMPUTE MAX LABEL ‘MAXIMUM SALES’ OF SALES ON PRODUCTS
COMPUTE MIN LABEL ‘MINIMUM SALES’ OF SALES ON PRODUCTS
COMPUTER SUM LABEL ‘TOTAL’ OF SALES ON PRODUCTS
```

CONCAT

参见：

CONCATENATE

语法：

```
CONCAT (‘x’, ‘y’)
```

变量：

x: 字符串变量
y: 字符串变量

例子：

PL/SQL
```
Var1 := CONCAT (‘Oracle 8 ‘, ‘is now released’) ;
```
SQL
```
SELECT CONCAT (‘Oracle 8 ‘, ‘is now released’) “Press Release”
FROM DUAL;
Press Release
-----------------------
Oracle 8 is now released.
```

CONCATENATE

参见：

CONCAT

语法：

||

例子：

```
PL/SQL
Var1 := ‘Oracle 8 ‘ || ‘is now released’;
SQL
SELECT ‘Oracle 8 ‘ || ‘is now released’ “Press Release” FROM
DUAL;
Press Release
-----------------------
Oracle 8 is now released.
```

CONNECT BY

语法：

```
SELECT sql_expn FROM [user.]table WHERE where_condition
CONNECT BY [PRIOR] expn = [PRIOR] expn
START WITH expn = expn
ORDER BY expn
```

变量：

sql_expn：有效的SQL表达式
user：表的主人
table： SELECT的表
where_condition：SELECT的WHERE条件
expn：任意有效的表达式

例子：

```
SQL
SELECT employee_name, department_name FROM employee
CONNECT BY emp_no = PRIOR department_no
ORDER BY department_no;.
```

CONVERT

参见：

DECODE

语法：

CONVERT (‘x’,’desc_set’ [, ‘source_set’])

变量：

x：单引号印起的字符型
desc_set：单引号印起的字符集名
source_set：单引号印起的原来字符集名，可以选择下面字符集名：

通用字符集名	编码字符集名(为了查阅，不翻译)
US7ASCII	US 7-bit ASCII character set
WE8DEC	DEC West European 8-bit character set
WE8HP	HP West European Laserjet 8-bit character set
F7DEC	DEC French 7-bit character set
WE8EBCDIC500	IBM West European EBCDIC Code Page 500
WE8PC850	IBM PC Code Page 850
WE8ISO8859P1	ISO 8859-1 West European 8-bit character set

例子：

SQL
SELECT CONVERT (‘strutz’, ‘ WE8HP’, ‘ F7DEC ‘) “Conversion”
FROM DUAL;
Conversion

Strutz.

COPY

参见：

CREATE DATBASE LINK

语法：

COPY FROM username[/password] @database_specification TO
username[/password]@database_specification
{APPEND|CREATE|INSERT|REPLACE} destination_table
[(column_1,..., column_N)] USING query

变量：

username：用户名
password：口令

database specification: 数据库
destination_table：目的数据库
column_1,...,column_N：目标表的字段
query：查询语句

例子：

SQL
```
COPY FROM brokerage/sql TO sales/sql APPEND customers (id,
name, address) USING select id, name, address from
brokerage.customers;
```

COS

参见：

COSH, SIN, SINH, TAN, TANH

语法：

COS (n)

变量：

n：数字变量

例子：

PL/SQL
```
Var1:= COS (90);
```
SQL
```
SELECT COS (90) “Cosine of 90 degrees” FROM DUAL;
Cosine of 90 degrees
--------------------
-.4480736
```

COSH

参见：

COS, SIN, SINH, TAN, TANH

语法：

COSH (n)

变量：

n：数字变量

例子：

```
PL/SQL
Var1:= COSH (20);
SQL
SELECT COSH (20) “Hyperbolic Cosine of 20 deg” FROM DUAL;
Hyperbolic Cosine of 20 deg
---------------------------
242582598
```

COUNT

参见：

Group by SUM

语法：

```
COUNT ( * | [DISTINCT | ALL] expn)
```

变量：

*：所有列（包括重复）的个数
DISTINCT：所有不同列（不重复）的个数
ALL：所有列（包括重复）的个数
Expn：表达式

例子：

```
SQL
SELECT COUNT (*) “Total # of Sales” FROM SALES;
Total # of Sales
--------------
12
```

CREATE CLUSTER

参见：

ALTER CLUSTER, DROP CLUSTER

语法：

```
CREATE CLUSTER [user.]cluster
( column1 datatype [, column2 datatype, ....] )
[INITRANS n]
[MAXTRANS n]
[PCTFREE n]
[PCTUSED n]
[SIZE n [K | M] ]
[STORAGE n]
[TABLESPACE tablespace]
[INDEX | [HASH IS column] HASHKEYS n]
```

变量：

user：建立cluster的用户
cluster：Cluster 名
n：任何正整数

例子：

SQL
```
CREATE CLUSTER customer PCTFREE 12.
```

CREATE CONTROLFILE

参见：

CREATE DATABASE

语法：

```
CREATE CONTROLFILE [REUSE] [SET] DATABASE database
LOGFILE [GROUP n] file [,[GROUP n] file] ( RESETLOG |
NORESETLOG)
DATAFILE file [,file]
[ MAXLOGFILES n]
[ MAXLOGMEMBERS n]
[ MAXLOGHISTORY n]
[ MAXDATAFILES n]
[ MAXINSTANCE n]
[ ARCHIVELOG | NOARCHIVELOG ]
```

变量：

n：任何正整数
file：物理文件名

例子：

SQL
```
CREATE CONTROLFILE REUSE SET DATABASE ORACLE
LOGFILE ‘D:\ORAWIN95\DATABASE\LOG1ORCL.ORA’ SIZE
200K,
DATAFILE ‘D:\ORAWIN95\DATABASE\SYS1ORCL.ORA’ SIZE
20M;.
```

CREATE DATABASE

参见：

ALTER DATABASE,
CREATE CONTROLFILE

语法：

```
CREATE DATABASE [database] [CONTROLFILE REUSE] LOGFILE
[GROUP n] file [,[GROUP n] file]
( RESETLOG | NORESETLOG)
DATAFILE file [,file]
[ MAXLOGFILES n]
[ MAXLOGMEMBERS n]
[ MAXLOGHISTORY n]
[DATAFILE file_definition [,file_definition]]
[ MAXDATAFILES n]
[ MAXINSTANCE n]
[ ARCHIVELOG | NOARCHIVELOG ]
[EXCLUSIVE]
( CHARACTER SET charset)
```

变量：

n：任何正整数
file：物理文件名
charset：字符集

例子：

SQL
```
CREATE DATABASE ORACLE
CONTROLFILE REUSE
LOGFILE ‘D:\ORAWIN95\DATABASE\LOG1ORCL.ORA’ SIZE 200K
REUSE, ‘D:\ORAWIN95\DATABASE\LOG2ORCL.ORA’ SIZE 200K
REUSE
```

```
DATAFILE ‘D:\ORAWIN95\DATABASE\SYS1ORCL.ORA’ SIZE 20M
REUSE AUTOEXTEND ON
NEXT 10M MAXSIZE 200M
CHARACTER SET WE8ISO8859P1;.
```

CREATE DATABASE LINK

参见：

分布应用的 CREATE SYNONYM

语法：

```
CREATE [PUBLIC] DATABASE LINK link
CONNECT TO user IDENTIFIED
BY password USING ‘connect_string’
```

变量：

link：数据库链名
user：数据库用户
password：有效的口令
connect_string：被访问的远程数据库字串

例子：

```
SQL
CREATE DATABASE LINK international_customers connect to INTL_DB
identified by intl using ‘D:INTERNATIONAL’;
SELECT CUSTOMER_NAME FROM CUSTOMER@INTL_DB;.
```

CREATE DIRECTORY

参见：

LOBs DROP DIRECTORY, BFILENAME

语法：

```
CREATE [OR REPLACE] DIRECTORY directory AS pathname;
```

变量：

directory：建立对象的目录
pathname：单引号引起的完整路径

例子：

SQL
```
CREATE OR REPLACE DIRECTORY bfile_directory AS ‘/bfile’;
```

CREATE FUNCTION

参见：

Functions ALTER FUNCTION,
DROP FUNCTION

语法：

```
CREATE [OR REPLACE] FUNCTION [user.]function
[ (parameter [IN] datatype [,parameter [IN] datatype] ... ) ]
RETURN datatype (IS | AS) block
```

变量：

user：建立函数的用户名
function：数据库函数名
parameter：传给函数的参数
datatype：变量的数据类型
block: PL/SQL 程序块

例子：

SQL
```
CREATE FUNCTION loan_calculation (loan_amount NUMBER,
no_of_years NUMBER) RETURN NUMBER AS
Total_loan NUMBER;
BEGIN
Total_loan := loan_amount * no_of_years;
RETURN total_loan ;
END loan_calculation ;
```

CREATE INDEX

参见：

ALTER INDEX, DROP INDEX

语法：

```
CREATE INDEX [user.]index
ON [user.]table (column [ASC | DESC] [,column
[ASC | DESC] ] ... )
[CLUSTER [user.]cluster]
[INITRANS n]
[MAXTRANS n]
[PCTFREE n]
[STORAGE storage]
[TABLESPACE tablespace]
[NO SORT]
```

变量：

user：建立索引的用户名
index：索引名
table：要建索引的表名
cluster：要建索引的CLUSTER名
n：任何正整数
tablespace：存放索引的表空间

例子：

SQL

```
CREATE INDEX loan_application_indx
ON loan (loan_id ASC)
TABLESPACE temp;
CREATE INDEX customer_indx
ON loan (cust_id ASC, loan_id ASC)
TABLESPACE temp;.
```

CREATE LIBRARY

参见：

CREATE FUNCTION, CREATE
PROCEDURE, DROP LIBRARY

语法：

```
CREATE [OR REPLACE] LIBRARY library_name [IS | AS] filename;
```

变量：

library_name：SQL和PL/SQL要调用的外部（3GL）函数及存储过程
filename：外部物理文件名

例子：

SQL

```
CREATE LIBRARY ext_lib IS ‘/lib/file1.sql’;
CREATE OR REPLACE LIBRARY ext_lib2 AS ‘/lib/file2.sql’;
```

CREATE MATERIALIZED VIEW

参见：

ALTER MATERIALIZED VIEW

语法：

```
CREATE MATERIALIZED VIEW  [user.]mview
```

例子：

```
CREATE MATERIALIZED VIEW hq_emp
REFRESH COMPLETE
START WTIH SYSDATE NEXT SYSDATE +1/4096
AS SELECT * FROM hq_emp;
```

CREATE MATERIALIZED VIEW LOG

参见：

ALTER MATERIALIZED VIEW LOG

语法：

```
CREATE  MATERIALIZED VIEW LOG ON [schemal.] table
```

例子：

```
CREATE MATERIALIZED VIEW sales_by_month_by_state
TABLESPACE my_ts PARALLEL (10)
ENABLE QUERY REWRITE
BUILD IMMEDIATE
```

```
REFRESH COMPLETE
AS SELECT t.month, g.state, SUM(f.sales) AS sum_sales
FROM fact f, time t, geog g
WHERE f.cur_date = t.cur_date AND f.city_id = g.city_id
GROUP BY month, state;
```

CREATE PACKAGE

参见：

```
CREATE PACKAGE BODY,
ALTER PACKAGE,
DROP PACKAGE
```

语法：

```
CREATE [OR REPLACE] PACKAGE [user.]package { IS | AS}
{ variable_declaration |
cursor_specification |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_specification |
function_specification } ;
[{ variable_declaration |
cursor_specification |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_specification |
function_specification } ; ] ...
END [package]
```

变量：

user：建立 Package的用户名
package：建立Package 的名

例子：

SQL

```
CREATE OR REPLACE PACKAGE loan_approval AS
Type LoanRecTyp IS RECORD (customer_id INTEGER, loan_amount
REAL);
CURSOR customer_history (customer_id NUMBER) RETURN LoanRecTyp;
```

```
PROCEDURE approve_loan
( customer_id CHAR,
loan_type CHAR,
loan_amount CHAR);
PROCEDURE cumulative_loan (loan_amount REAL);
END loan_approval ;
```

CREATE PACKAGE BODY

参见：

CREATE PACKAGE,
ALTER PACKAGE BODY,
DROP PACKAGE BODY

语法：

```
CREATE [OR REPLACE] PACKAGE BODY [user.]package { IS |
AS}
{ variable_declaration |
cursor_body |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_body |
function_body } ;
[{ variable_declaration |
cursor_body |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_body |
function_body } ; ] ...
END [package]
```

变量：

user：建立Package的用户名
package：在数据库中的Package的名字

例子：

SQL

```
CREATE OR REPLACE PACKAGE BODY loan_approval AS
CURSOR customer_history (customer_id NUMBER) RETURN LoanRecTyp
IS
```

```
SELECT customer_id FROM customer ;
PROCEDURE approve_loan
( customer_id CHAR,
loan_type CHAR,
loan_amount REAL) IS
BEGIN
IF loan_amount < 10000 THEN
UPDATE customer
SET loan_type = 'A'
WHERE cust_id = customer_id;
END IF;
END approve_loan;
PROCEDURE cumulative_loan (loan_amount REAL) IS
BEGIN
UPDATE total_outstanding SET loan_amt = loan_amt +
loan_amount;
END cumulative_loan;
END approve_loan ;.
```

CREATE PROCEDURE

参见：

```
ALTER PROCEDURE,
DROP PROCEDURE
```

语法：

```
CREATE [OR REPLACE] [user.]procedure
[ (parameter [IN] datatype [,parameter [IN] datatype]
... ) ]
(IS | AS) block
```

变量：

user：建立Procedure的用户名
procedure：Procedure的名字
parameter：传参数给 Procedure的变量名
datatype：变量的数据类型
block：PL/SQL程序块

例子：

SQL

```
CREATE PROCEDURE loan_calculation (loan_amount NUMBER,
```

```
no_of_years NUMBER) AS
Total_loan NUMBER;
BEGIN
Total_loan:= loan_amount * no_of_years;
END loan_calculation ;
CREATE OR REPLACE PROCEDURE increment (emp_id INTEGER,
increment REAL) IS
current_pay REAL;
missing_pay EXCEPTION;
BEGIN
SELECT salary INTO current_pay FROM employee WHERE
empno = emp_id;
IF current_pay IS NULL THEN
RAISE missing_pay;
ELSE
UPDATE emp SET salary = salary + increment
WHERE empno = emp_id;
END IF;
EXCEPTION
WHEN missing_pay THEN
INSERT INTO emp_audit VALUES (emp_id,
'Incomplete Details');
END increment ;
```

CREATE PROFILE

参见：

```
User Profiles ALTER PROFILE, ALTER
RESOURCE COST, DROP
PROFILE
```

语法：

```
CREATE PROFILE profile LIMIT
[ { SESSIONS_PER_USER |
CPU_PER_SESSION |
CPU_PER_CALL |
CONNECT_TIME |
IDLE_TIME |
LOGICAL_READS_PER_SESSION |
LOGICAL_READS_PER_CALL |
COMPOSITE_LIMIT } { n | UNLIMITED | DEFAULT } |
PRIVATE_SGA { n {K | M} | UNLIMITED | DEFAULT } ]
```

变量：

profile：希望修改的profile文件名
n：任何正整数

例子：

SQL
```
CREATE PROFILE customer LIMIT SESSION_PER_USER 12 ;
CREATE PROFILE customer LIMIT CONNECT_TIME 600 ;.
```

CREATE ROLE

参见：

ALTER ROLE,
DROP ROLE, GRANT

语法：

```
CREATE ROLE role
[ NOT IDENTIFIED | IDENTIFIED
[ BY PASSWORD | EXTERNALLY ] ]
```

变量：

role：角色名

例子：

SQL
```
CREATE ROLE custom_user ;
```

CREATE ROLLBACK SEGMENT

参见：

ALTER ROLLBACK SEGMENT,
DROP ROLLBACK SEGMENT

语法：

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
[TABLESPACE tablespace]
```

[STORAGE storage]

变量：

rollback_segment回：滚段名
tablespace：表空间名

例子：

SQL
```
CREATE ROLLBACK SEGMENT RB_HUMANRESOURCE STORAGE (INITIAL 50K
NEXT 100K OPTIMAL 150K);
```

CREATE SCHEMA

参见：

CREATE TABLE, DROP SCHEMA

语法：

```
CREATE SCHEMA AUTHORIZATION schema
[ CREATE TABLE command |
CREATE VIEW command |
GRANT command ]
```

变量：

schema：希望建立的模式名

例子：

SQL
```
CREATE SCHEMA AUTHORIZATION cust_schema
CREATE TABLE customer
(CUSTOMER_ID NUMBER(4) NOT NULL,
CUSTOMER_NAME VARCHAR2(50) NOT NULL,
STREET_ADDRESS VARCHAR2(50) NOT NULL,
CITY VARCHAR2(50) NOT NULL,
STATE VARCHAR2(2) NOT NULL CHECK (‘FL’, ‘TX’, ‘MD’)
CUST_TYPE VARCHAR2(1) NOT NULL
LOAN_AMOUNT NUMBER(6) ) |
CREATE VIEW loan.view AS
SELECT customer_id, customer_name, loan_amount FROM customer
WHERE
customer_type = ‘L’ ;
```

CREATE SEQUENCE

参见：

ALTER SEQUENCE,
DROP SEQUENCE

语法：

```
CREATE SEQUENCE [user.]sequence
[ INCREMENT BY n]
[ START WITH n]
[ MAXVALUE n | NOMAXVALUE]
[ MINVALUE n | NOMINVALUE]
[ CYCLE | NO CYCLE]
[CACHE n | NO CACHE]
[ORDER | NO ORDER]
```

变量：

user：建立sequence的用户名
sequence：sequence名
n：任何正整数

例子：

SQL

```
CREATE SEQUENCE customer_code_seq
INCREMENT BY 1
START WITH 1;
CREATE SEQUENCE loan_application_seq
START WITH 100
INCREMENT WITH 100
NOMAXVALUE ;.
```

CREATE SNAPSHOT

参见：

ALTER SNAPSHOT,
DROP SNAPSHOT

语法：

```
CREATE SNAPSHOT [user.] snapshot
```

```
[PCTFREE n |
PCTUSED n |
INITRANS n |
MAXTRANS n |
STORAGE n |
TABLESPACE tablespace ]
[CLUSER cluster (column1 [,column2] ...) ]
[ REFRESH { FAST | COMPLETE | FORCE }
[START WITH start_date] [NEXT
next_date] ]
AS query
```

变量：

user：建立snapshot的用户名
snapshot：snapshot的名字
n：任何正整数
tablespace：表空间名
start_date：开始处理日期
next_date：刷新snapshot的日期
query：从 SELECT snapshot取数据的语句

例子：

SQL

```
CREATE SNAPSHOT customer.inactive_cust_snapshot AS
SELECT customer_id, customer_name,
customer_address
FROM customer@remote;.
```

CREATE SNAPSHOT LOG

```
CREATE SNAPSHOT,
ALTER SNAPSHOT LOG,
DROP SNAPSHOT LOG
```

语法：

```
CREATE SNAPSHOT LOG ON [user.] table
[PCTFREE n |
PCTUSED n |
INITRANS n |
MAXTRANS n |
STORAGE n |
TABLESPACE tablespace ]
```

变量：

user：建立snapshot log的用户名
table：希望snapshot log的表名
n：any positive integer value
tablespace：the Tablespace on which you would like to create the snapshot

例子：

SQL
```
CREATE SNAPSHOT LOG
ON customer.customer@remote
TABLESPACE temp ;
```

CREATE SYNONYM

参见：

CREATE DATABASE LINK,
CREATE TABLE, CREATE VIEW

语法：

```
CREATE [PUBLIC] SYNONYM [user.] synonym
FOR [user.] table [@database_link]
```

变量：

user：建立同义词的用户
table：建立同义词的表
database_link：远程数据库连接名

例子：

SQL
```
CREATE SYNONYM r_cust FOR CUSTOMER@REMOTE_SITE;
```

CREATE TABLE

参见：

ALTER TABLE,
DROP TABLE

语法：

```
CREATE TABLE [schema.]table
      [ ( { column datatype[DEFAULT expr]  [WITH ROWID]
```

```
                  [SCOPE IS [user.]scope_table_name][column_constrint]...
      | table_constraint | REF (ref_column_name)WITH ROWID
      | SCOPE FOR (ref_column_name) IS [user.]scope_table_name }

      [ ,{ column datatype[DEFAULT expr]  [WITH ROWID]
                  [SCOPE IS [user.]scope_table_name][column_constrint]...
      | table_constraint | REF (ref_column_name)WITH ROWID
      | SCOPE FOR (ref_column_name) ISuser.]scope_table_name}]...)]
[{[ORGANIZATION{HELP |INDEX}
            |PCTTHRESHOLD|INCLUDEING column_name]
   [OVERFLOW[physical_sttrbutes_clause|TABLESPACE tablespace]...]
   |hpysical_attributes_clause|TABLESPACE tablespace
                      |LOB(lob_item[,lob_item...])STORE AS
                      [lob_segname]
                     [(TABLESPACE tablespace |STORAGE storage
                        |CHUNK integer | PCTVERSION integer
                        |CACHE
                        | NOCACHE LOGGING|NOCACHE NOLOGGING
                        |INDEX[lob_index_name]
                            [(TABLESPACE tablespace
                               |STORAGE storage
                               |INITITRANS integer
                               |MAXTRANS integer )...])]
                    |NESTED TABLE nested_item STORE AS  storage_table
                    |{LOGGING|NOLOGGING}]...
              |CLUSTER cluster(column[,column]...)}]
              [PARALLEL parallel_clause ]
              [PARTITION BY RANGE (column_list)
                  (PARTITION  [partition_name]  VALUE  LESS  THAN  (value_list)
[physical_attributes-clause
                  |TABLESPACE tablespace
                  |{LOGGING|NOLOGGING}]...
        [ENABLE enable_clause|DISABLE disable_clause]...
          [AS subquery]
          [CACHE|NOCACHE]


Physical_attributes_clause ::=

        [PCTFREE integer | PCTUSED integer
         |INITRANS integer | MAXTRANS integer
         |STORAGE storage_clause]
```

变量：

user：建立表的用户名
table：表名
column：列（字段）名
expn： DEFAULT 值
column_constraint：列的完整性限制

column_constraint 可以是:

NULL   表示列可以空
NOT NULL表示列不能空
UNIQUE   列指定为唯一键
PRIMARY KEY 列指定为主键
FOREIGN KEY 列指定为外部键
REFERENCES 表示被引用的是主键
ON DELETE CASCASE 当主键表被删除时外部表自动被删除
CHECK 指定每行的条件
Cluster：   希望建立表的cluster名
N：   任意正整数
Query：   SELECT 子查询

例子：

SQL
```
CREATE TABLE customer
(CUSTOMER_ID NUMBER(4) NOT NULL,
CUSTOMER_NAME VARCHAR2(50) NOT NULL,
STREET_ADDRESS VARCHAR2(50) NOT NULL,
CITY VARCHAR2(50) NOT NULL,
STATE VARCHAR2(2) NOT NULL CHECK (‘FL’, ‘TX’, ‘MD’) ;.
```

CREATE TABLESPACE

参见：

ALTER TABLESPACE

语法：

```
CREATE TABLESPACE tablespace
DATAFILE file_definition [,file_definition] |
[DEFAULT STORAGE storage ]
[ ONLINE | OFFLINE ]
```

变量：

tablespace：表空间名
file_definition：文件大小的定义（以 K 或 M）

例子：

SQL
```
CREATE TABLESPACE customer
(DATAFILE ‘customer02.dat’);
```

CREATE TRIGGER

参见：

ALTER TRIGGER, DROP TRIGGER

语法：

```
CREATE TRIGGER [user.]trigger
{BEFORE | AFTER }
{DELETE | INSERT | UPDATE [ OF column1 [, column2] ...}
[ OR {DELETE | INSERT | UPDATE [ OF column1 [, column2]
...} ...] ON [user.]table
[ REFERENCING { OLD [AS] old | NEW [AS] new } ]
[ FOR EACH ROW]
[WHEN (when_condition) ]
block
```

变量：

user：建立触发器的用户名
trigger：触发器名
column：触发器的列名
table：定义触发器的表名
old：引用表中的旧值
new：引用表中的新值
when_condition：触发器条件
block：要执行的 PL/SQL块

例子：

SQL
```
CREATE OR REPLACE TRIGGER cumulative_loan_update
AFTER INSERT OR UPDATE OF loan_amount
ON customer
FOR EACH ROW
WHEN (new. loan_type = ‘A’)
BEGIN
UPDATE total_outstanding SET loan_amt = loan_amt +
new.loan_amount;
```

END;.

CREATE TYPE

参见：

ALTER TYPE, CREATE
TYPE BODY, DROP TYPE

语法：

CREATE [OR REPLACE] TYPE [schema.]type_name { AS | AS TABLE |
AS OBJECT }
{ VARRAY (size) | VARYING ARRAY (size) }
{ OF datatype}
{ REF object_type_name}
{ MAP | ORDER MEMBER function_specification }
{ PRAGMA RESTRICT_REFERENCES function_specification
restriction}

变量：

schema：建立对象类型的模式名
type_name：所建立的类型名
size：VARRAY大小限制
datatype：任意类型，如 CHAR, DATE, NUMBER
object_type_name：类型名
function_specification：函数名
restriction：限制的类型，如WNDS, WNPS, RNDS, RNPS

例子：

SQL
```
CREATE TYPE customer_obj AS OBJECT
(name CHAR(20),
address CHAR(50),
age NUMBER(2));
CREATE TYPE name_type AS VARRAY(100) OF CHAR(20);
CREATE TYPE loan_obj AS OBJECT
(customer_name CHAR(20),
loan_amount NUMBER(5),
MEMBER FUNCTION get_amount RETURN NUMBER,
pragma RESTRICT_REFERENCES (get_amount, WNDS));.
```

CREATE TYPE BODY

参见：

CREATE TYPE,
DROP TYPE BODY

语法：

```
CREATE [OR REPLACE] TYPE BODY [schema.]type_name { IS | AS }
{ MEMBER procedure_declaration |
function_declaration
{ MAP | ORDER MEMBER function_declaration }
END;
```

变量：

schema：建立类型体的模式名
type_name：类型名（ CREATE TYPE）
procedure_declaration： PL/SQL 过程
function_declaration： PL/SQL 函数

例子：

SQL

```
CREATE OR REPLACE TYPE BODY loan_obj IS
MAP MEMBER FUNCTION get_amount RETURN NUMBER IS
BEGIN
RETURN loan_amount ;
END;
END;
```

CREATE USER

参见：

ALTER USER, DROP USER

语法：

```
CREATE USER user [IDENTIFIED [BY password |
EXTERNALLY]]
[ DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[PROFILE profile]
```

变量：

user：用户名
tablespace：表空间名
n：任意正整数
profile：用户所挂接的profile名

例子：

SQL
```
CREATE USER supervisor identified by master
DEFAULT tablespace cust_tablespace
TEMPORARY tablespace temp;
```

CREATE VIEW

参见：

ALTER VIEW, DROP VIEW

语法：

```
CREATE [OR REPLACE] [FORCE/NO FORCE] VIEW [user.]view
[column_name1, column_name2] AS query
[WITH CHECK OPTION [CONSTRAINT constraint];
```

变量：

user：建立视图所在的用户名
view：视图名
query：SQL SELECT语句
constraint：约束名

例子：

SQL
```
CREATE OR REPLACE VIEW customer.loan.view AS
SELECT customer_id, loan_id, loan_amount
FROM customer
WHERE customer_type = ‘L’;
CREATE OR REPLACE VIEW employee.salary_view
(EMPNO, ENAME, SALARY) AS
SELECT emp_id, emp_name, grade.grade_range
FROM employee, grade
WHERE employee.grade_id = grade.grade_id;
CREATE OR REPLACE VIEW ladies_only AS
```

```
SELLECT customer_id, name, address, city, state, zip
FROM sex_type = ‘F’ WITH CHECK OPTION ;
```

CURRVAL

参见：

CREATE SEQUENCE, NEXTVAL,
PSEUDOCOLUMN

语法：

CURRVAL

变量：

user：建立sequence 所在的用户名
sequence：sequence名

例子：

SQL*Plus

```
SELECT loan_seq.CURRVAL FROM DUAL;
```

CURSOR_ALREADY_OPEN

参见：

INVALID_CURSOR

语法：

```
EXCEPTION
WHEN CURSOR_ALREADY_OPEN THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一系列语句

例子：

PL/SQL

```
BEGIN
OPEN loan_cur;
LOOP
```

```
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN CURSOR_ALREADY_OPEN THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘CURSOR_ALREADY_OPEN’ ;
WHEN OTHERS THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘OTHER ERROR’;
END;
```

DATATYPE

参见：

CREATE TABLE, ALTER TABLE

BFILE 外部大数据类型，最大GB.
BLOB 大数据类型，最大GB.
CHAR(n) n=1 到 255
CLOB大数据类型，最大4GB.
DATE 日期型 从公元前4712 1月1日 到 公元后4712 12月 31日（占7字节），还包括时分秒
FLOAT(n) p=1 到 126 二进制数
LONG 大数据类型，最大2GB
LONG RAW 大二进制数据类型，最大2GB
MLSLABEL标识二进制格式(Trusted Oracle版本)
NCHAR(n) n=1 到 2,000
NCLOB大数据类型，最大4GB
NUMBER(p,s) p=1到38, s=-84 到 127
NVARCHAR2(n) n=1 到 4,000，见 VARCHAR 和 NVARCHAR2
RAW(n) n=1到 255 原始二进制数
ROWID 伪列，十六进制格式
VARCHAR(n) n=1到 4,000
VARCHAR2(n) n=1到 4,000

DATATYPE —CHAR

参见：

VARCHAR2, LONG

语法：

CHAR (n)

变量：

n：类型大小，缺省为1,最大为 2000。

DATATYPE —DATE

参见：

MONTHS_BETWEEN,ADD_MONTHS,
LAST_DAY, NEXT_DAY

语法：

DATE

变量：

无

DATATYPE —FLOAT

参见：

NUMBER

语法：

FLOAT (n)

变量：

n：数字，最小为1，最大为126位。

DATATYPE —LONG

参见：

CHAR, VARCHAR2

语法：

LONG

变量：

无

DATATYPE —LONGRAW

参见：

RAW

语法：

LONG RAW

变量：

无

DATATYPE —MLSLABEL

参见：

Table 命名

语法：

MLSLABEL

变量：

无

DATATYPE —NUMBER

参见：

FLOAT

语法：

NUMBER (p, s)

变量：

DATATYPE —RAW

参见：

LONGRAW

语法：

RAW (n)

变量：

n：1到255

DATATYPE —ROWID

参见：

DELETE, UPDATE, SELECT, SELECT INTO

语法：

ROWID

变量：

无

DATATYPE —VARCHAR

参见：

VARCHAR2

语法：

VARCHAR (n)

变量：

n： 1 到 4000

DATATYPE —VARCHAR2

参见：

CHAR, LONG

语法：

VARCHAR2 (n)

变量：

n:：1 到 4000

DECLARE

参见：

DECLARE CURSOR, DECLARE,
DATABASE, DECLARE TABLE,
CREATE PACKAGE

语法：

DECLARE
object_declaration_1,...,object_declaration_n;

变量：

object_declaration_1,...,object_declaration_n：对象及类型声明

例子：

PL/SQL
DECLARE
total_sales NUMBER;
total_products NUMBER;
unit_price CONSTANT NUMBER:= 1.0;

DECODE

参见：

TRANSLATE

语法：

DECODE (n, if1, then1, if2, then2 ...., else)

变量：

n：编码值或列值
if1：条件1
then1：当条件成立后希望代替的值
else：如果条件都满足就用else 来代替

例子：

PL/SQL
Var1:= DECODE (Var2, ‘Sybase’, ‘Oracle’,);

SQL
SELECT DECODE (‘Microsoft’, ‘Microsoft’, ‘Macrosoft’) “Example
“ FROM DUAL;
Example

Macrosoft.

DEFINE

参见：

SET

语法：

DE[FINE] [variable]|[variable = text]

变量：

variable：希望定义的变量名
text：1分配给变量的文本

例子：

SQL
DEFINE last_name = ‘Jordan’
DEFINE department_number = 160

DEL

参见：

APPEND, CHANGE, EDIT

语法：

DEL [n|n m|n *|n LAST|*|* n|* LAST|LAST]

变量：

n：　　希望删除的缓冲区的行数
n m：　希望删除的缓冲区的n行到m行
n *：　 希望删除的缓冲区的n行到当前行
n LAST：希望删除的缓冲区的n行到最末行
*：　　希望删除的缓冲区的当前行
* n：　 希望删除的缓冲区的当前行到n行
* LAST：希望删除的缓冲区的当前行到最末行
LAST：　希望删除的缓冲区的最末行

例子：

SQL
D 1
DEL 5 LAST.

DELETE

参见：

INSERT, UPDATE, SELECT,
SELECT INTO

语法：

DELETE [FROM] {table | (sub_query)} [alias]
[WHERE {search_condition | CURRENT OF cursor_name}];

变量：

table：用户有权删除的表名
sub_query：一组子查询
alias：别名
WHERE search_condition：有效的 WHERE 条件句
CURRENT OF cursor_name：与光标有关的当前行

例子：

SQL
```
DELETE FROM employees where employee_id = ‘124561123’;
DELETE FROM employees where employee_id in
(SELECT employee_id
FROM department_head
WHERE department_active_flag = ‘N’);.
```

DEREF

参见：

REFTOHEX, MAKE_REF

语法：

```
DEREF (expn)
```

变量：

expn：引用对象的任意表达式

例子：

SQL
```
CREATE TABLE customer_tab (customer_no NUMBER, loan REF
loan_tab);
SELECT DEREF (loan) FROM customer_tab;
```

DESCRIBE

参见：

CREATE TABLE

语法：

```
DESC[RIBE] [user.]table[@database_link_name] [column]|
[user.]object[.subobject]
```

变量：

user：表所有者的用户名
table：希望列出的表名

database_link_name：指定的表或视图的名字
column：列名
object：描述的对象
subobject： 描述的子对象

例子：

SQL
```
DESCRIBE loan
DESC acct
```

DROP CLUSTER

参见：

CREATE CLUSTER

语法：

```
DROP CLUSTER [user.] cluster [INCLUDING TABLES] [CASCADE
CONSTRAINTS]
```

变量：

user：已经建立的cluster的用户名
cluster：希望从数据库中删除的cluster 名

例子：

SQL
```
DROP CLUSTER marketing;
DROP CLUSTER telecommunications INCLUDING TABLES;
DROP CLUSTER branch INCLUDING TABLES CASCADE CONSTRAINTS;
```

DROP DATABASE LINK

参见：

CREATE DATABASE LINK

语法：

```
DROP [PUBLIC] DATABASE LINK database_link
```

变量：

PUBLIC：从数据库中删除的PUBLIC

database_link：从数据库中删除的公共数据库链

例子：

SQL
```
DROP PUBLIC DATABASE LINK corporation;
DROP DATABASE LINK tracking;
```

DROP DIRECTORY

参见：

CREATE DIRECTORY,
BFILENAME

语法：

```
DROP DIRECTORY directory;
```

变量：

directory：希望删除的目录名

例子：

SQL
```
DROP DIRECTORY bfile_directory;
```

DROP FUNCTION

参见：

CREATE FUNCTION,
ALTER FUNCTION

语法：

```
DROP FUNCTION [user.]function
```

变量：

user：建立函数的用户名
function：希望删除的函数

例子：

SQL
```
DROP FUNCTION acctrpt
```

```
DROP FUNCTION loan.calint
```

DROP INDEX

参见：

CREATE INDEX, ALTER INDEX

语法：

```
DROP INDEX [user.]index
```

变量：

user：建立索引的用户名
index：希望删除的索引名

例子：

SQL
```
DROP INDEX sales;
DROP INDEX resources;
```

DROP LIBRARY

参见：

CREATE LIBRARY

语法：

```
DROP LIBRARY library_name;
```

变量：

library_name: 要删除的库名字
database.

例子：

SQL
```
DROP LIBRARY ext_procs;
```

DROP PACKAGE

参见：

CREATE PACKAGE, ALTER PACKAGE

语法：

```
DROP PACKAGE [body.] [user.] package
```

变量：

body：包体名字
user：建立包的用户名
package：希望删除的包的名字

例子：

SQL

```
DROP PACKAGE loan;
DROP PACKAGE BODY customers.newacct;
```

DROP PROCEDURE

参见：

CREATE PROCEDURE,
ALTER PROCEDURE

语法：

```
DROP PROCEDURE [user.] procedure
```

变量：

user：建立存储过程的用户名
procedure：数据库中存储过程的用户名

例子：

SQL

```
DROP PROCEDURE loanrpt;
DROP PROCEDURE acct.acctrpt;
```

DROP PROFILE

参见：

CREATE PROFILE,
ALTER PROFILE

语法：

DROP PROFILE [user.]profile

变量：

user：建立Profile的用户名
profile：Profile名字

例子：

SQL
DROP PROFILE accountant;
DROP PROFILE my_profile;

DROP ROLE

参见：

CREATE ROLE, SET ROLE, ALTER ROLE

语法：

DROP ROLE [user.] role

变量：

user：建立角色的用户名
role：希望删除的角色名

例子：

SQL
DROP ROLE manager;
DROP ROLE officer;

DROP ROLLBACK SEGMENT

参见：

CREATE ROLLBACK SEGMENT,
ALTER ROLLBACK SEGMENT

语法：

```
DROP ROLLBACK SEGMENT rollback_segment
```

变量：

rollback_segment: 希望删除的回滚段名字

例子：

SQL
```
DROP ROLLBACK SEGMENT humanresources;
DROP ROLLBACK SEGMENT insurance;
```

DROP SEQUENCE

参见：

CREATE SEQUENCE, ALTER SEQUENCE

语法：

```
DROP SEQUENCE [user.]sequence
```

变量：

user：建立序列的用户名
sequence：希望删除的序列名

例子：

SQL
```
DROP SEQUENCE customer.new_cust_seq;
DROP SEQUENCE seq1;
```

DROP SNAPSHOT

参见：

CREATE SNAPSHOT,
ALTER SNAPSHOT

语法：

```
DROP SNAPSHOT [user.] snapshot
```

变量：

user：建立快照的用户名
snapshot：希望从数据库删除的快照

例子：

SQL
```
DROP SNAPSHOT customer.inactive_cust_snapshot;
DROP SNAPSHOT snp1;
```

DROP SNAPSHOT LOG

参见：

CREATE SNAPSHOT LOG,
ALTER SNAPSHOT LOG

语法：

DROP SNAPSHOT LOG ON [user.] table

变量：

user：建立快照日志的用户名
table：希望删除的与快照日志有关的主表

例子：

SQL
```
DROP SNAPSHOT LOG ON inventory
```

DROP SYNONYM

参见：

CREATE SYNONYM

语法：

```
DROP [PUBLIC] SYNONYM [user.]synonym
```

变量：

user：建立同义词的用户名
synonym：希望删除的同义词的用户名

例子：

SQL
```
DROP SYNONYM PUBLIC SYNONYM computer;
DROP SYNONYM advertisement;
```

DROP TABLE

参见：

CREATE TABLE, ALTER TABLE

语法：

```
DROP TABLE [user.] table [CASCADE CONSTRAINTS]
```

变量：

user：建立表的用户名
table：希望从数据库中删除的表名

例子：

SQL
```
DROP TABLE loan;
DROP TABLE acct CASCADE CONSTRAINTS;
```

DROP TABLESPACE

参见：

CREATE TABLESPACE,
ALTER TABLESPACE

语法：

```
DROP TABLESPACE tablespace [INCLUDING CONTENTS]
[CASCADE CONSTRAINTS]
```

变量：

tablespace：希望删除的表空间名字

例子：

SQL
```
DROP TABLESPACE accounting;
```

```
DROP TABLESPACE airlines INCLUDING CONTENTS;
DROP TABLESPACE construction CASCADE CONSTRAINTS;
```

DROP TRIGGER

参见：

CREATE TRIGGER,
ALTER TRIGGER

语法：

```
DROP TRIGGER [user.] trigger
```

变量：

user: 建立触发器的用户名
trigger: 希望删除的数据库触发器名字

例子：

SQL*Plus
```
DROP TRIGGER accounting.acct_trigger_1;
DROP TRIGGER loan_trigger_2;
```

DROP TYPE

参见：

CREATE TYPE, ALTER TYPE,
DROP TYPE BODY

语法：

```
DROP TYPE [schema.] type_name [FORCE]
```

变量：

schema: 建立类型的模式名
type_name: 对象类型名字

例子：

SQL
```
DROP TYPE customer_obj;
DROP TYPE loan_obj FORCE;
```

DROP TYPE BODY

参见：

CREATE TYPE,
ALTER TYPE,
DROP TYPE BODY

语法：

```
DROP TYPE BODY [schema.] type_name
```

变量：

schema: 建立类型的模式名
type_name: 希望删除的对象类型名字

例子：

SQL
```
DROP TYPE BODY customer_obj;
DROP TYPE BODY loan_obj;
```

DROP USER

参见：

CREATE USER

语法：

```
DROP USER user [CASCADE]
```

变量：

user：希望删除的用户名字

例子：

SQL
```
DROP USER john;
DROP USER robert CASCADE;
```

DROP VIEW

参见：

CREATE VIEW, ALTER VIEW

语法：

DROP VIEW [user.]view

变量：

user：建立视图的用户名字
view：希望删除的视图的名字

例子：

SQL
DROP VIEW marketing.marketview;
DROP VIEW loanview;

DUMP

参见：

RAWTOHEX, HEXTORAW

语法：

DUMP (s [, fmt [, start [, length]]])

变量：

s：CHAR或VARCHAR的字符类型.
Fmt：引用的格式，缺省为 ASCII 或 EBCDIC
Start：S的开始位置
Length：长度

例子：

SQL
SELECT DUMP (SALES_AGENT) “DUMP (Sales Agent, 10, 1, 5)” FROM
SALES;
DUMP (Sales Agent)

Typ=1 Len=14: 20,4e,20,46,52.

DUP_VAL_ON_INDEX

参见：

TOO_MANY_ROWS,
VALUE_ERROR

语法：

```
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一序列语句

例子：

PL/SQL

```
BEGIN
UPDATE STATISTICS
SET ROW_COUNT = ROW_COUNT + 1;
IF SQL%ROWCOUNT = 0 THEN
INSERT INTO STATISTICS VALUES (1);
END IF;
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
UPDATE STATISTICS
SET ROW_COUNT = 0;
END;
```

EDIT

参见：

GET, SAVE, SET

语法：

```
ED[IT] [file_name[.ext]]
```

变量：

file_name：希望打开的文件名
ext：扩展类型，缺省为.SQL

例子：

SQL
```
ED LOAN_RPT
EDIT ACCT_RPT
EDIT LOAN.QRY
```

EMPTY_BLOB

参见：

BFILENAME, EMPTY_CLOB

语法：

```
EMPTY_BLOB ( )
```

变量：

无

例子：

PL/SQL
```
Var1:= EMPTY_BLOB ();
```
SQL
```
INSERT INTO lob_table VALUES (‘BLOB DEMO’, EMPTY_BLOB());
UPDATE lob
SET blob_column = EMPTY_BLOB()
WHERE name = ‘BLOB DEMO’;
```

EMPTY_CLOB

参见：

BFILENAME, EMPTY_BLOB

语法：

```
EMPTY_CLOB ( )
```

变量：

无

例子：

PL/SQL
```
Var1:= EMPTY_CLOB ();
```
SQL
```
INSERT INTO lob_table VALUES (‘CLOB DEMO’, EMPTY_CLOB());
UPDATE lob
SET clob_column = EMPTY_CLOB()
WHERE name = ‘CLOB DEMO’;
```

EXCEPTION INIT Pragma

```
DECLARE EXCEPTION,
EXCEPTION, SQLERRM
```

语法：

```
PRAGMA EXCEPTION_INIT (n);
```

变量：

n： 例外整数值

例子：

PL/SQL
```
DECLARE
exception_error_occurred exception;
pragma exception_init
(exception_error_occurred -786);
BEGIN
......
EXCEPTION
WHEN exception_error_occurred THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = EXCEPTION ERROR’ ;
END;
```

EXECUTE

参见：

PREPARE

语法：

EXEC[UTE] *statement*

变量：

statement：a PL/SQL statement

例子：

SQL
```
EXECUTE sales_calculation;
EXEC sales_input_trigger;
```

EXISTS

参见：

NOT IN, ANY,
ALL, BETWEEN

语法：

```
WHERE EXISTS (subquery)
```

变量：

subquery: 一个 SQL SELECT 子句

例子：

SQL
```
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE EXISTS (SELECT * FROM MONTHLY_SALES WHERE
DAILY_SALES.SALES_AMOUNT = MONTHLY_SALES.SALES_AMOUNT);
SALES_AMOUNT
----------
12.23
```

EXIT

参见：

LOOP, WHILE-LOOP, FOR-LOOP,
EXIT-WHEN

语法：

```
LOOP
statement_1,...statement_n;
```

```
IF condition_1 is true THEN
EXIT;
END IF;
END LOOP
```

变量：

statement_1,...,statement_n：一序列 PL/SQL 语句
condition_1：适当的 PL/SQL条件句

例子：

PL/SQL

```
LOOP
Total_Salary:= Base_Salary + Commission + Bonus;
IF Total_Salary = 0 THEN
EXIT;
END IF;
UPDATE salary SET employee_salary = Total_Salary WHERE
employee_number:= emp_id;
END LOOP;
```

EXIT

参见：

COMMIT, DISCONNECT, QUIT

语法：

```
EXIT|QUIT [SUCCESS|FAILURE|WARNING/n/variable]
[COMMIT|ROLLBACK]
```

变量：

n：返回代码正整数
variable：用户定义的变量或系统变量

例子：

SQL

```
EXIT
EXIT SQL.SQLCODE
QUIT.
```

EXIT-WHEN

参见：

LOOP, WHILE-LOOP,
FOR-LOOP, EXIT-WHEN

语法：

```
LOOP
statement_1,...statement_n;
EXIT WHEN condition_1 is true;
END LOOP
```

变量：

statement_1,...,statement_n：可多次执行的 PL/SQL 语句

condition_1：适当PL/SQL 条件句

例子：

PL/SQL
```
LOOP
Total_Salary:= Base_Salary + Commission + Bonus;
EXIT WHEN Total_Salary = 0;
UPDATE salary SET employee_salary = Total_Salary WHERE
employee_number:= emp_id;
END LOOP ;
```

EXP

参见：

SQRT, POWER

语法：

```
EXP (n)
```

变量：

n：数值变量

例子：

PL/SQL
```
Var1:= EXP (12);
```

```
SQL
SELECT EXP (12) “Exponential Value of 12” FROM DUAL;
Exponential Value of 12
----------------------
162754.79
```

EXPLAIN PLAN

参见：

ANALYZE

语法：

```
EXPLAIN PLAN
[SET STATEMENT ID = statement_name]
[ INTO [user.]table]
FOR query
```

变量：

statement_name：输出表中一个标识说明方案的名字。如果没有给定，将在输出表中存放一个空值。
User：建立表的用户名
Table：计划存储的表名
Query：计划解释的SQL 语句

例子：

```
SQL
EXPLAIN PLAN
SET STATEMENT_ID = ‘exp_plan_customer
INTO all_plan
FOR
SELECT CUSTOMER_ID, CUSTOMER_NAME FROM CUSTOMER WHERE
CUSTOMER_ID IN (SELECT CUSTOMER_ID FROM LOAN_APPROVAL );.
```

FETCH

参见：

CLOSE, OPEN, OPEN-FOR

语法：

```
FETCH {cursor_name | cursor_variable_name
|:host_cursor_variable_name}
INTO {variable_1[, variable_n]... | record_name};
```

变量：

cursor_name: 光标名
cursor_variable_name: 光标变量名
host_cursor_variable_name: 在PL/SQL宿主变量中声明的光标变量
variable_1,...,variable_n: 一序列变量名
record_name: 用户定义的，或 %ROWTYPE 取数据的名字

例子：

```
PL/SQL
OPEN loan_cur ;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
CLOSE loan_cur;
OPEN emp_cur ;
LOOP
FETCH emp_cur INTO emp_id, emp_name, emp_title,
emp_salary;
EXIT WHEN emp_cur%NOTFOUND;
END LOOP;
CLOSE emp_cur;.
```

FLOOR

参见：

CEIL

语法：

```
FLOOR (n)
```

变量：

n：一个数值变量
例子：

```
PL/SQL
Var1:= FLOOR (147.2754);
```

```
SQL
SELECT FLOOR (147.2754) “Floor” FROM DUAL;
Floor
---------
147
```

FOR-LOOP

参见：

LOOP, EXIT, WHILE-LOOP,
EXIT-WHEN

语法：

```
FOR i in 1..10 LOOP
statement_1,...,statement_n;
END LOOP
```

变量：

i：循环指示变量
statement_1,...,statement_n：一序列PL/SQL语句

例子：

```
PL/SQL
FOR i in 1..total_employees LOOP
Total_Salary:= Base_Salary + Commission + Bonus;
UPDATE salary SET employee_salary = Total_Salary WHERE
employee_number:= emp_id;
END LOOP;
```

FORMAT

说明格式的保留字

FORMAT —DATE

日期格式代码表

日期代码	格式说明	例子

AD 或 BC	AD=Anno Domini 公元,BC=Before Christ 公元前。不带点的公元或公元前	‘YYYY AD’=1999 AD
A.D. 或 B.C.	带点的公元或公元前	‘YYYY A.D.’=1999 A.D.
AM 或 PM	AM= ante meridiem 上午，PM=post meridiem 下午。不带点的上午或下午	‘HH12AM’=09AM
A.M.或 P.M.	带点的上午或下午	‘HH12A.M.’=09A.M.
DY	星期几的缩写	Mon,Tue,...
DAY	星期几的全拼	Monday,Tuesday,...
D	一周的星期几，星期天=1，星期六=7	1,2,3,4,5,6,7
DD	一月的第几天，1→31	1，2，... 31
DDD	一年的第几天，1→366	1,2,3,...366
J	公元前的第几天（从公元前 4712 起 ?）	2451514,2451515,...
W	一个月的第几周，1→ 5	1,2,3,4,5
WW,IW	一年的第几周，一年的 ISO 的第几周	1,2,3,4,... 52
MM	两为数的月	01,02,03,...12
MON	月份的缩写	Jan,Feb,Mar ,...Dec
MONTH	月份的全拼	January,February,...
RM	罗马数字的月份，I → XII	I,II,III,IV,...XII
YYYY,YYY,YY,Y	四位数的年，三位数的年	1999,999,99,9
YEAR	年的全拼	Nineteen Ninety-nine
SYYYY	如果是公元前(BC),年份前负号	-1250
RR	当前年份的后两位数字	01 代表 2001 年
HH,HH12	12 小时制，1→12	1,2,3,...12
HH24	24 小时制，0→23	0,1,2,3,...23
MI	一小时中的第几分，0→59	0,1,2,3...59
SS	一分中的第几秒，0→59	0,1,2,3,...59
SSSSS	一天中的第几秒，0→86399	0,1,2,3,...86399
../-;:	标点符号表示法	文字显示
‘text’	引号表示法	文字显示

FORMAT —NUMBER

9 指定的数字．如果为正，前用空格； 如果为负，前带 – 号
0 以 0 来填在前面
$ 美圆号在前
B 浮点数整数部分用空格
MI 带有负号的负数
S 具有给定位置的数值．正号用+；负号用 -
PR 在 <angle brackets>内的负号
D 小数后有指定位数的十进制
G 在指定位置放组分隔符
C 在指定位置的ISO货币符号
L 在指定位置的本地货币符号
, 在指定位置的豆号

. 在指定位置的小数点
V 一个被10的n次方乘的数值，n是v后面的位数
EEEE 科学记数法表示
RN 以罗马数字表示的数字
Rn 以罗马小写数字表示的数字
FM 一个不带前导和后空的数字.

GET

参见：

GET, SAVE, SET, EDIT

语法：

GET filename [.ext] [LIS[T]|NOL[IST]]

变量：

filename: 希望加载到SQL缓冲区的文件名
ext: 文件的扩展名，缺省为 SQL.

例子：

SQL
GET LOAN_RPT
GET LOAN.QRY

GLB

参见：

LUB

语法：

GLB ([DISTINCT | ALL] mls)

变量：

DISTINCT：不同的保留字，可选
ALL：包括相同的所有
Mls：MLSLABEL.类型的变量

GOTO

参见：

LOOP, WHILE-LOOP,
FOR-LOOP

语法：

```
BEGIN
...
GOTO <<calculate_loan>;
...
<<calculate_loan>
SELECT loan_amount FROM loan;
END
```

变量：

calculate_loan：标记符

例子：

PL/SQL
```
BEGIN
...
GOTO <<calculate_total_salary>>;
...
<<calculate_total_salary>>
Total_Salary:= Base_Salary + Commission + Bonus;
END
```

GRANT

参见：

REVOKE

语法：

```
GRANT system_privilege | role TO user | role | PUBLIC
[WITH ADMIN OPTION]
GRANT object_privilege | ALL column ON schema.object
FROM user | role | PUBLIC WITH GRANT OPTION
```

变量：

system_privilege：授与用户或角色的系统权限名
role：授个用户的角色名
user：用户或角色名
object_privilege：对象的权限名字，可以是：

- ALTER
- DELETE
- EXECUTE
- INDEX
- INSERT
- REFERENCES
- SELECT
- UPDATE

Column：列名
Schema：模式名
Object：对象名字

例子：

SQL
```
GRANT CREATE TABLE TO gavaskar;
GRANT team_leader TO crystal;
GRANT INSERT, UPDATE ON sales TO larry WITH GRANT
OPTION;
GRANT ALL TO PUBLIC;.
```

GREATEST

参见：

GREATEST_LB, LEAST,
LEAST_LB

语法：

```
GREATEST (expn1 [, expn2] ...)
```

变量：

expn：表达式或列名

例子：

SQL
```
SELECT GREATEST(‘JOHN’, ‘JONNY’, ‘JANARTHAN’) “GREATEST” FROM
```

```
DUAL;
GREAT
-------------
JONNY
```

GREATEST_LB

参见：

GREATEST, LEAST, LEAST_LB

语法：

```
GREATEST_LB (label1 [, label2] ...)
```

变量：

label：MLSLABEL 或 RAW MLSLABEL.类型变量

HEXTORAW

参见：

RAWTOHEX

语法：

```
HEXTORAW (‘x’)
```

变量：

x：带引号的十六进制

例子：

SQL

```
SELECT HEXTORAW(animation) “Animation” FROM animation;
Animation
----------------------------------------
3D
```

HOST

参见：

@, START

语法：

HO[ST] [command]

变量：

command：操作系统命令

例子：

SQL
```
HOST ls -d
HOST ls *.rpt
```

IF-THEN

参见：

IF-THEN-ELSE, IF-THEN-ELSEIF

语法：

```
IF condition_1 is true THEN
statement_1,...,statement_n
END IF
```

变量：

condition_1：条件1
statement_1,...,statement_n：如果条件1是真要执行的一序列PL/SQL 语句

例子：

PL/SQL
```
IF sales > quota THEN
Total_Salary:= Base_Salary + bonus;
END IF;
```

IF-THEN-ELSE

参见：

IF-THEN, IF-THEN-ELSEIF

语法：

```
IF condition_1 is true THEN
statement_1,...,statement_n
ELSE
statement_11,...,statement_n1
END IF
```

变量：

condition_1：适当的条件
statement_1,....,statement_n：如果条件1是真要执行的一序列PL/SQL 语句
statement_11,...,statement_n1：如果条件1是假要执行的一序列PL/SQL 语句

例子：

PL/SQL

```
IF sales > quota THEN
Total_Salary:= Base_Salary + Commission +
Bonus;
ELSE
Total_Salary:= Base_Salary + Commission;
END IF;
```

IF-THEN-ELSEIF

参见：

IF-THEN-ELSE, IF-THEN-ELSEIF

语法：

```
IF condition_1 is true THEN
statement_1,...,statement_n
ELSEIF condition_2 i,...,true THEN
statement_11,...,statement_n1
ELSE
statement_12,...,statement_n2
END IF
```

变量：

condition_1：第一个被判断的确定的条件
condition_2：下一个被判断的确定的条件
statement_1,...,statement_n：如果判断condition_1为真才执行的一系列PL/SQL语句
statement_11,...,statement_n1：如果判断condition_2为真才执行的一系列PL/SQL语句
statement_12,...,statement_n2：如果判断condition_1和condition_2都为假才执行的一系列PL/SQL语句

例子：

PL/SQL

```
IF sales > quota THEN
Total_Salary:= Base_Salary + Commission +
Bonus;
ELSEIF sales = quota THEN
Total_Salary:= Base_Salary + Commission;
ELSE
Total_Salary:= Base_Salary;
END IF;.
```

INITCAP

参见：

LOWER, UPPER

语法：

INITCAP (‘x’)

变量：

x：一个由圆括号所界定的字符变量

例子：

PL/SQL

```
Var1:= INITCAP (‘oracle is a good database’);
```

SQL

```
SELECT INITCAP(‘oracle is a good database.’) “Sentence” FROM
DUAL;
Sentence
-------------------------
Oracle Is A Good Database.
```

INPUT

参见：

ACCEPT

语法：

I[NPUT] text

变量：

text：你想添加到SQL缓冲区里的文字

例子：

SQL
I WHERE state = ‘FL’
INPUT ORDER BY last_name

INSERT

参见：

DELETE, UPDATE, SELECT,
SELECT INTO

语法：

```
INSERT INTO {table | (sub_query)}
[(column_1[, column_2,...column_n)]
{VALUES
(sql_expression_1[,sql_expression_2,...,sql_expression_n ])
| sub_query);
```

变量：

table：数据库中用户正在插入数据的表，用户要有INSERT权限
column_1,...,column_n：指定数据库表中将要被插入数据的列
sql_expression_1,...,sql_expression_n：一串有效的SQL表达式序列

例子：

SQL*Plus
INSERT INTO employees VALUES (‘John Doe’, ‘124561123’,
‘Manager’);
INSERT INTO employees

```
(employee_name, employee_id)
VALUES
('John Doe', '124561123');
INSERT INTO employees
(SELECT * FROM department_head) ;.
```

INSTR

参见：

函数 INSTRB

语法：

```
INSTR ('w','x' ['y','z'])
```

变量：

w：一个由单括弧界定的字符类型或 VARCHAR2 类型的变量，你将替换掉这个变量里的某些字符。
X：一个由单括弧界定的字符类型或VARCHAR2类型的变量，它是你想查找的变量。
Y：一个表示查找起始位置的数字变量。缺省值为1。这个值是可选的。
Z：一个正的数字变量，它表示查找第几位的字符。缺省为1。这个值是可选的。

例子：

```
PL/SQL
Var1:= INSTR ('Oracle Training', 'ra', 1, 2);
SQL
SELECT INSTR ('Oracle Training', 'ra', 1, 2) "Instring" FROM
DUAL;
Instring
---------
9.
```

INSTRB

参见：

INSTR

语法：

```
INSTRB ('w','x' ['y','z'])
```

变量：

w：一个由单括弧界定的字符类型或 VARCHAR2 类型的变量，你将替换掉这个变量里的某些字符。.
x：一个由单括弧界定的字符类型或VARCHAR2类型的变量，它是你想查找的变量。
Y：一个表示查找起始位置的数字变量。缺省值为1。这个值是可选的。
Z：一个正的数字变量，它表示查找第几位的字符。缺省为1。这个值是可选的。

例子：

PL/SQL
Var1:= INSTRB (‘Oracle Training’, ‘ra’, 1, 2);
SQL
SELECT INSTRB (‘Oracle Training’, ‘ra’, 1, 2) “Instring Bytes”
FROM DUAL;
Instring Bytes

9.

INSTRB 函数与INSTR函数几乎一样，除了INSTR 函数返回一个字符类型的值而INSTRB返回一个字节值。INSTRB函数在W里查找字符串x中从第y位开始的第z位字符。

如果y 是负数，oracle就会从w的末尾向前查找。如果没有找到，Oracle返回0。

INTERSECT

参见：

UNION, UNION ALL, MINUS

语法：

b1 INTERSECT b2

变量：

b1 和 b2 是选择语句

例子：

SQL
SELECT SALES_AMOUNT FROM DAILY_SALES WHERE SALES_DATE =
‘09-FEB-96’
INTERSECT
SELECT SALES_AMOUNT FROM DAILY_SALES WHERE SALES_DATE =
‘09-FEB-97’ ;
SHIPPING_T

299.95

INVALID_CURSOR

参见：

CURSOR_ALREADY_OPEN

语法：

```
EXCEPTION
WHEN INVALID_CURSOR THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一个语句序列

例子：

```
PL/SQL
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN INVALID_CURSOR THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘INVALID_CURSOR’ ;
END;
```

INVALID_NUMBER

参见：

VALUE_ERROR, ZERO_DIVIDE

语法：

```
EXCEPTION
WHEN INVALID_NUMBER THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一个语句序列

例子：

PL/SQL
DECLARE
total_salary NUMBER;
BEGIN
total_salary:= TO_NUMBER (‘ABC’);
EXCEPTION
WHEN INVALID_NUMER THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘INVALID NUMBER
CONVERSION’ ;
END;

KEYWORDS

Oracle 有一列没有保留但仍在Oracle 语法中使用的单词表。这些单词就是关键字。 你可
以使用这些关键字作为变量名或对象名，不过系统强烈建议你不要这样做，因为这样会使你的代码难于阅读和理解。

下面就是一个关键字表。

Table A-16 关键字列表

关键字	关键字	关键字	关键字
ADMIN	AFTER	ALLOCATE	ANALYZE
ARCHIVE	ARCHIVELOG	AUTHORIZATION	AVG
BACKUP	BEGIN	BECOME	BEFORE
BLOCK	BODY		
CACHE	CANCEL	CASCADE	CHANGE
CHARACTER	CHECKPOINT	CLOSE	COBOL
COMMIT	COMPILE	CONSTRAINT	CONSTRAINTS
CONTENTS	CONTINUE	CONTROLFILE	COUNT
CURSOR	CYCLE		
DATABASE	DATAFILE	DBA	DEC
DECLARE	DISABLE	DISMOUNT	DOUBLE
DUMP			
EACH	ENABLE	END	ESCAPE
EVENTS	EXCEPT	EXCEPTIONS	EXEC

EXPLAIN	EXECUTE	EXTENT	EXTERNALLY
FETCH	FLUSH	FREELIST	FREELISTS
FORCE	FOREIGN	FORTRAN	FOUND
FUNCTION			
GO	GOTO	GROUPS	
INCLUDING	INDICATOR	INITRANS	INSTANCE
INT			
KEY			
Keyword	*Keyword*	*Keyword*	*Keyword*
LANGUAGE	LAYER	LINK	LISTS
LOGFILE			
MANAGE	MANUAL	MAX	MAXDATAFILES
MAXINISTANCES	MAXLOGFILES	MAXLOGHISTORY	MAXLOGMEMBERS
MAXTRANS	MAXVALUE	MIN	MINEXTENTS
MINVALUE	MODULE	MOUNT	
NEXT	NEW	NOARCHIVELOG	NOCACHE
NOCYCLE	NOMAXVALUE	NOMINVALUE	NONE
NOORDER	NORESETLOGS	NORMAL	NOSORT
NUMERIC			
OFF	OLD	ONLY	OPTIMAL
OPEN	OWN		
PACKAGE	PARALLEL	PASCAL	PCTINCREASE
PCTUSED	PLAN	PLI	PRECISION
PRIMARY	PRIVATE	PROCEDURE	PROFILE
QUOTA			
READ	REAL	RECOVER	REFERENCES
REFERENCING	RESETLOGS	RESTRICTED	REUSE
ROLE	ROLES	ROLLBACK	
SAVEPOINT	SCHEMA	SCN	SECTION
SEGMENT	SEQUENCE	SHARED	SNAPSHOT
SOME	SORT	SQLCODE	SQLERROR
STATEMENT_ID	STATISTICS	STOP	STORAGE
SUM	SWITCH	SYSTEM	
TABLES	TABLESPACE	TEMPORARY	THREAD
TIME	TRACING	TRANSACTION	TRIGGERS
TRUNCATE			
UNDER	UNLIMITED	UNTIL	USE
USING			
WHEN	WRITE	WORK.	

LABELS

参见：

BLOCK

语法：

```
<<my_label>>
LOOP
statement_1,...,statement_n
END LOOP <<my_label>>
```

变量：

my_label：一个未声明的标识符
statement_1,...,statement_n：一个语句序列

例子：

PL/SQL

```
<<calculate_salary>>
LOOP
new_salary:= base_salary + bonus + commission;
END LOOP calculate_salary;
```

LAST_DAY

参见：

NEXT_DAY, SYSDATE,
ADD_MONTHS

语法：

```
LAST_DAY (d)
```

变量：

d：一个有效的日期型变量

例子：

PL/SQL

```
Days_Left:= LAST_DAY(SYSDATE) - SYSDATE;
```

SQL

```
SELECT LAST_DAY(SYSDATE) “Last Day” FROM DUAL;
Last Day
```

30-NOV-97

LEAST

参见：

GREATEST, GREATEST_LB,
LEAST_LB

语法：

LEAST (expn1 [, expn2] ...)

变量：

expn：任何一个有效的表达式或列。

例子：

SQL
SELECT LEAST('JOHN', 'JONNY', 'JANARTHAN') "LEAST" FROM DUAL;
LEAST

JANARTHAN

LEAST_LB

参见：

GREATEST, GREATEST_LB, LEAST

语法：

LEAST_LB (label1 [, label2] ...)

变量：

label：一个 MLSLABEL 类型或 RAW MLSLABEL.类型的变量

LENGTH

参见：

LENGTHB

语法：

LENGTH (‘x’)

变量：

x：一个由单括弧界定的字符类型或 varchar2 类型的变量。

例子：

```
PL/SQL
Var1:= LENGTH (‘Oracle’);
SQL
SELECT LENGTH (‘Oracle’) “Length” FROM DUAL;
Length
---------
6
```

LENGTHB

参见：

LENGTH

语法：

LENGTHB (‘x’)

变量：

x：一个由单括弧界定的CHAR 类型或VARCHAR2 类型的变量。

例子：

```
PL/SQL
Var1:= LENGTHB (‘Oracle’);
SQL
SELECT LENGTHB (‘Oracle’) “Length in bytes” FROM DUAL;
Length in bytes
---------------
6
```

LIKE

参见：

AND, OR, NOT

语法：

WHERE column1 LIKE ‘valuepattern’

变量：

column1：一个你要查找的CHAR 或 VARCHAR2 类型的列。
Value：那些与你所要查找的模式相似的值。
Pattern：有下面3种模式：

表A-17 LIKE 查找模式示例

模式	例子	查找对象
%	‘abc%’	以*abc开头的值*
_	‘__a’	在第三位上的是a的值
%	‘%a%a%’	含有两个a的值

%标号是通配符。
_标志是一个位置标志。
Oracle 中匹配模式对所给条件是相当敏感的。

例子：

SQL

```
SELECT NAME_OF_AGENT “AGENT NAME” FROM SALES WHERE
NAME_OF_AGENT LIKE ‘MICHAEL%’;
“AGENT NAME”
-------------------
MICHAEL GEORGE
MICHAEL JORDAN
MICHAEL MAGNUM
SELECT NAME_OF_AGENT FROM SALES WHERE NAME_OF_AGENT LIKE ‘__C’;
“AGENT NAME”
-------------------
MICKY
ARCHIE
MICHAEL
```

LIST

参见：

PAUSE

语法：

L[IST] [n|n m|n *|n LAST|*|* n|* LAST|LAST]

变量：

n：SQL*Plus 列出的SQL缓冲区里的第n行。
n m：SQL*Plus 列出的SQL缓冲区里从m行开始的n行。
n *：SQL*Plus 列出的SQL缓冲区里从当前行开始的n行。
n LAST：SQL*Plus 缓冲区里从最后一行开始向前的n行。SQL*Plus 将列出这n行。
*：SQL*Plus将列出的SQL缓冲区里的当前行。
* n：SQL*Plus 缓冲区里从当前行开始向后的n行，SQL*Plus将列出这n行。
* LAST：SQL*Plus 将列出的SQL缓冲区里从当前行开始到最后一行结束的所有行。
LAST：SQL*Plus 将列出的SQL缓冲区里的最后一行。

例子：

SQL
L 2

LN

参见：

LOG

语法：

LN (n)

变量：

n：一个大于0的数字变量。

例子：

PL/SQL
Var1:= LN (12);
SQL
SELECT LN (12) “Logarithm of 12” FROM DUAL;
Logarithm of 12

2.4849066

LOCK TABLE

参见：

COMMIT, ROLLBACK,
SAVEPOINT, SET TRANSACTION

语法：

LOCK TABLE table_1 [,table_2, ..., table_n] IN lock_mode MODE
NOWAIT

变量：

table_1,...,table_n: 一系列你想通过使用LOCK TABLE语句锁住的数据库表。
lock_mode: 对于某一数据库表你要设定的锁定模式。你可以从如下的锁定模式中任选一个。

- EXCLUSIVE
- SHARE ROW EXCLUSIVE
- SHARE
- SHARE UPDATE
- ROW SHARE
- ROW EXCLUSIVE

NOWAIT: Oracle will not wait to lock the given Table(s), if the Table(s) is(are) not available

例子：

SQL
```
LOCK TABLE loan IN SHARE MODE ;
LOCK TABLE region IN EXCLUSIVE MODE NOWAIT;
LOCK TABLE acct IN SHARE UPDATE MODE;
LOCK TABLE bank IN ROW EXCLUSIVE MODE NOWAIT;
LOCK TABLE user IN SHARE ROW EXCLUSIVE MODE;
LOCK TABLE branch IN ROW SHARE MODE NOWAIT;.
```

LOG

参见：

LN

语法：

```
LOG (m, n)
```

变量：

m：任何一个不是0 或 1的正的数字变量。.
n：任何一个正的数字变量。.

例子：

```
PL/SQL
Var1:= LOG (12, 2);
SQL
SELECT LOG (12, 2) “Log base 2 of 12” FROM DUAL;
Log base 2 of 12
---------------
.27894295
```

LOGIN_DENIED

参见：

NOT_LOGGED_ON

语法：

```
EXCEPTION
WHEN LOGIN_DENIED THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一系列的语句

例子：

```
PL/SQL
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN LOGIN_DENIED THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘ACCESS DENIED’ ;
```

```
END;
```

LOOP

参见：

EXIT, WHILE-LOOP,
FOR-LOOP, EXIT-WHEN

语法：

```
LOOP
statement_1,..., statement_n
END LOOP
```

变量：

statement_1,...,statement_n：一系列PL/SQL 将重复执行的语句

例子：

PL/SQL

```
LOOP
Total_Salary:= Base_Salary + Commission + Bonus;
UPDATE salary SET employee_salary = Total_Salary WHERE
employee_number:= emp_id;
END LOOP;
```

LOWER

参见：

INITCAP, UPPER

语法：

```
LOWER (‘x’)
```

变量：

x：一个由单括弧界定的字符类型或VARCHAR2类型的变量。

例子：

PL/SQL

```
Var1:= LOWER (‘ORACLE’);
```

SQL

```
SELECT LOWER (‘ORACLE IS A GOOD DATABASE’) “LowerCase Sentence”
FROM DUAL;
LowerCase Sentence
------------------------
oracle is a good database
```

LPAD

参见：

RPAD, REPLACE

语法：

LPAD (‘x’, n [,’y’])

变量：

x：一个由单括弧界定的字符类型或VARCHAR2类型的变量。
N：一个正的数字变量。
Y；一个由单括弧界定的字符类型或VARCHAR2l类型的变量。这个变量是可选的。如果不指定，缺省是空字符。.

例子：

```
PL/SQL
Var1:= LPAD (‘Oracle’, 1);
SQL
SELECT LPAD (‘ is a good database’, 25, ‘Oracle’) “Example of
Left Padding” FROM DUAL;
Example of Left Padding
------------------------
Oracle is a good database
```

LTRIM

参见：

RTRIM

语法：

LTRIM (‘x’ [,’y’])

变量：

x：一个由单括弧界定的字符类型或VARCHAR2类型的变量。
Y：一个由单括弧界定的字符类型或VARCHAR2l类型的变量。这个变量是可选的。如果不指定，缺省是空字符。.

例子：

```
PL/SQL
Var1:= LTRIM (‘Oracle’, ‘Or’);
SQL
SELECT LTRIM (‘The Theresa of all mothers’, ‘The’) “Example of
LTrim” FROM DUAL;
Example of LTrim
----------------------
Theresa of all mothers
```

LUB

参见：

GLB

语法：

```
LUB ( [DISTINCT | ALL] mls)
```

变量：

DISTINCT：这个选项使函数只考虑那些参数中有不同值的情况，此项是可选的。
ALL：这个选项使函数考虑所有的情况，包括参数中有拷贝值。此项是可选的。
Mls：这是一个 MLSLABEL.类型的变量。

MAKE_REF

参见：

DEREF, REFTOHEX

语法：

```
MAKE_REF(view, key [,key...])
```

变量：

view：你想为其中的行创建引用的对象视图。
Key：命令中作为主键的键。

例子：

SQL

```
CREATE TYPE loan_obj AS OBJECT
(loan_amount NUMBER, interest_rate NUMBER);
CREATE TABLE loan_table
(loan_amount NUMBER, interest_rate NUMBER);
CREATE VIEW loan_view
OF loan_table
WITH OBJECT loan_view_obj(loan_amount,
interest_rate) AS
SELECT * from loan_table;
SELECT MAKE_REF(loan_view_obj, 1, 3) FROM DUAL;
```

MAX

参见：

MIN, SUM

语法：

```
MAX ( [DISTINCT | ALL] expn)
```

变量：

DISTINCT：这个选项使函数只考虑那些参数中有不同值的情况，这项是可选的。
ALL：这个选项使函数考虑所有的情况包括参数中有相同值，这项是可选的。
Expn：这可以是一列SQL查询也可以是一列数学表达式。.

例子：

PL/SQL

```
Var1:= MAX (1, 2, 3);
```

SQL

```
SELECT MAX (DAILY_SALES) “Maximum Sales” FROM SALES;
Maximum Sales
--------------
133
```

MIN

参见：

MAX, SUM

语法：

MIN ([DISTINCT | ALL] expn)

变量：

DISTINCT：这个选项使函数只考虑那些参数中有不同值的情况，这项是可选的。
ALL：这个选项使函数考虑所有的情况包括参数中有相同值，这项是可选的。
Expn：这可以是一列SQL查询也可以是一列数学表达式。.

例子：

PL/SQL

```
Var1:= MIN (1, 2, 3);
```

SQL

```
SELECT MIN (DAILY_SALES) “Minimum Sales” FROM SALES;
Minimum Sales
--------------
12
```

MINUS

参见：

UNION, UNION ALL, INTERSECT

语法：

b1 MINUS b2

变量：

b1 and b2 是SELECT语句。

例子：

SQL

```
SELECT SALES_AMOUNT FROM DAILY_SALES WHERE SALES_DATE =
‘09-FEB-96’
MINUS
SELECT SALES_AMOUNT FROM DAILY_SALES WHERE SALES_DATE =
‘09-FEB-97’ ;
SHIPPING_T
----------
199.95
```

MOD

语法：

MOD (x , y)

变量：

x：数字变量
y：数字变量

例子：

PL/SQL

```
Var1:= MOD (12, 5);
```

SQL

```
SELECT MOD (12, 5) “Modulus Value of 12/5” FROM DUAL;
Modulus Value of 12/5
---------------------
2
```

MONTHS_BETWEEN

参见：

ADD_MONTHS

语法：

MONTHS_BETWEEN (d1, d2)

变量：

d1 & d2：有效的日期型变量

例子：

PL/SQL

```
MONTHS_INBETWEEN:= SYSDATE - SALES_DATE;
```

SQL

```
SELECT MONTHS_BETWEEN (SYSDATE, ‘26-JAN-98’) “MONTHS_INBETWEEN”
FROM DUAL;
MONTHS_INBETWEEN
----------------
-2.53143
```

NEW_TIME

参见：

SYSDATE

语法：

NEW_TIME (d, ‘tz1’, ‘tz2’)

变量：

d:：一个有效的日期型变量
tz1 & tz2:：下表中的任一时区
表A-18 NEW_TIME的时区说明

时区1	时区2	说明
AST	ADT	大西洋标准时间
BST	BDT	白令海标准时间
CST	CDT	中部标准时间
EST	EDT	东部标准时间
GMT		格林尼治标准时间
HST	HDT	阿拉斯加—夏威夷标准时间
MST	MDT	山区标准时间
NST		纽芬兰标准时间
PST	PDT	太平洋标准时间
YST	YDT	YUKON标准时间

例子：

PL/SQL

```
LONDON_TIME:= NEW_TIME (SYSDATE, ‘EST’, ‘GMT’)
```

SQL

```
SELECT NEW_TIME (SYSDATE, ‘EST’, ‘GMT’) “LONDON_TIME” FROM
DUAL;
LONDON_TI
---------
09-NOV-97.
```

NEXT_DAY

参见：

LAST_DAY, SYSDATE

语法：

NEXT_DAY (d, ‘x’)

变量：

d:：一个有效的日期型变量
x： 一个有括号界定的字符变量，它代表星期中的某一天
(MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, and SUNDAY)

例子：

PL/SQL

```
DATE_FOR_NEXT_WEEK:= NEXT_DAY(‘26-JAN-47’,’TUESDAY’);
```

SQL

```
SELECT NEXT_DAY(‘26-JAN-47’,’TUESDAY’) “DATE_AFTER_TUESDAY_26-
JAN-47” FROM DUAL;
DATE_AFTER
---------
28-JAN-47
```

NEXTVAL

参见：

CREATE SEQUENCE,
CURRVAL, PSEUDOCOLUMN

语法：

[user.]sequence.NEXTVAL

变量：

user： 创建序列的用户
sequence： 你要从中取出下一个值的序列

例子：

SQL*Plus

```
SELECT loan_seq.NEXTVAL FROM DUAL;
```

NLS_CHARSET_DECL_LEN

参见：

DELETE, UPDATE, SELECT, SELECT INTO

语法：

NLS_CHARSET_DECL_LEN (len, char_id)

变量：

len： 列的长度
char_id:：列的ID字符集合

例子：

SQL
```
SELECT NLS_CHARSET_DECL_LEN (130, nls_charset_id (‘char_cs’))
FROM DUAL;
```

NLS_CHARSET_ID

参见：

NLS_CHARSET_DECL_LEN,
NLS_CHARSET_NAME

语法：

NLS_CHARSET_ID (‘x’)

变量：

x： 一个由括号界定的字符串变量

例子：

SQL
```
SELECT NLS_CHARSET_ID (‘char_cs’)) FROM DUAL;
SELECT NLS_CHARSET_ID (‘nchar_cs’)) FROM DUAL;
```

NLS_CHARSET_NAME

参见：

NLS_CHARSET_DECL_LEN,
NLS_CHARSET_ID

语法：

NLS_CHARSET_NAME (n)

变量：

n:：一个有效的数值变量

例子：

SQL
```
SELECT NLS_CHARSET_NAME (2) FROM DUAL;
SELECT NLS_CHARSET_NAME (1001) FROM DUAL;
```

NLS_INITCAP

参见：

NLS_LOWER, NLS_UPPER

语法：

NLS_INITCAP (‘x’ [,’nlsparm’])

变量：

x:：一个由单括弧界定的字符变量
nlsparm：这个变量确定了排序规则。它可以是语言的排序顺序也可以是 BINARY 排序顺序，语言上的排序顺序主要是针对那些由于事例转换而引起的特殊语言的要求。这个参数是可选的，如果没有给定它，函数就会在此会话期间使用缺省的排序顺序。‘nlsparams’ 值可以有下述形式：
‘NLS_SORT = sort’

例子：

SQL
```
SELECT NLS_INITCAP(‘strutz’, ‘NLS_SORT = XGerman’) “Sentence”
FROM DUAL;
Senten
------
Strutz
```

NLS_LOWER

参见：

NLS_INITCAP, NLS_UPPER

语法：

NLS_LOWER (‘x’ [,’nlsparm’])

变量：

x:：一个由单括弧界定的字符变量
nlsparm：这个变量确定了排序规则。它可以是语言的排序顺序也可以是 BINARY 排序顺序，语言上的排序顺序主要是针对那些由于事例转换而引起的特殊语言的要求。这个参数是可选的，如果没有给定它，函数就会在此会话期间使用缺省的排序顺序。‘nlsparams’ 值可以有下述形式：
‘NLS_SORT = sort’

例子：

SQL
SELECT NLS_LOWER (‘strutz’, ‘NLS_SORT = XGerman’) “LowerCase
Sentence” FROM DUAL;
LowerC
------
strutz

NLS_UPPER

参见：

NLS_INITCAP, NLS_LOWER

语法：

NLS_UPPER (‘x’ [,’nlsparm’])

变量：

x:：一个由单括弧界定的字符变量
nlsparm：这个变量确定了排序规则。它可以是语言的排序顺序也可以是 BINARY 排序顺序，语言上的排序顺序主要是针对那些由于事例转换而引起的特殊语言的要求。这个参数是可选的，如果没有给定它，函数就会在此会话期间使用缺省的排序顺序。‘nlsparams’ 值可以有下述形式：
‘NLS_SORT = sort’

例子：

SQL
SELECT NLS_UPPER (‘strutz’, ‘NLS_SORT = XGerman’) “UpperCase
Sentence” FROM DUAL;
UpperC

STRUTZ

NO_DATA_FOUND

参见：

TOO_MANY_ROWS

语法：

```
EXCEPTION
WHEN NO_DATA_FOUND THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n:：一系列的语句

例子：

```
PL/SQL
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR (-20001,
‘No Data Found’);
END;
```

NOAUDIT

参见：

AUDIT

语法：

```
NOAUDIT statement | system_privilege BY user [WITH
GRANT OPTION] [WHENEVER [NOT] SUCCESSFUL]
NOAUDIT object_operating ON schema.object [WHENEVER
[NOT] SUCCESSFUL]
```

变量：

statement:：你想停止检查的语句
system_privilege:：你想停止检查的系统权限

user:：你想停止检查的用户

例子：

```
PL/SQL
NOAUDIT SELECT TABLE BY john;
NOAUDIT CREATE TABLE BY martha;
NOAUDIT SELECT ON loanschema.loan WHENEVER SUCCESSFUL;.
```

NOT_LOGGED_ON

参见：

LOGIN_DENIED

语法：

```
EXCEPTION
WHEN NOT_LOGGED_ON THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n:：一系列的语句

例子：

```
PL/SQL
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN NOT_LOGGED_ON THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘NOT LOGEED ON’ ;
END;
```

NULL

参见：

ASSIGNMENT, DECLARE

语法：

```
IF condition_1 is true THEN
statement_1,...,statement_n
ELSE
NULL
END IF
```

变量：

condition_1：一个PL/SQL 将判断的确定的条件
statement_1,...,statement_n： 一系列如果condition_1 为真 PL/SQL 就迭代执行的语句。

例子：

PL/SQL

```
IF sales > quota THEN
Total_Salary:= Base_Salary + Commission +
Bonus;
ELSE
NULL;
END IF;
```

NVL

参见：

函数 NULL

语法：

```
NVL (value, substitute)
```

变量：

value:：任何一种类型的变量，包括 CHAR，VARCHAR2，NUMBER，DATE，和其它。
substitute： 与 value相同类型的变量，如果substitute 的类型与value不一样此函数就会首先对它进行类型转换。

例子：

SQL

```
SELECT NVL(SALES_AGENT, ‘AGENT NAME NOT AVAILABLE’) “NAMES OF
SALES AGENT” FROM SALES;
NAMES OF SALES AGENT
```

```
---------------------------------
JOHN HOPKINS
MICHAEL JORDAN
AGENT NAME NOT AVAILABLE
```

OPEN

参见：

CLOSE, OPEN-FOR, FETCH

语法：

```
OPEN cursor_name [(parameter_1 [,parameter_2,...,
parameter_n]...)]
```

变量：

cursor_name： 你要打开的指定游标的名字。
parameter_1,...,parameter_n： 你要传递给游标的参数。

例子：

```
PL/SQL
OPEN loan_cur;
OPEN loan_cur (1, 'Bank Holdings, Inc.', 500000)
```

OPEN-FOR

参见：

CLOSE, OPEN, FETCH

语法：

```
OPEN {cursor_name |:host_cursor_variable_name} FOR
select_statement;
```

变量：

cursor_name：你想打开的游标名字。
host_cursor_variable_name： 在一个 PL/SQL 主环境中声明的游标变量的名字。
host environment
select_statement: 一个有效的 SELECT 语句，它返回了传送给游标的值。

例子：

PL/SQL
OPEN loan_cur FOR SELECT * FROM loan;
OPEN:emp_cur FOR SELECT * FROM employee;
OPEN salary_cur FOR SELECT * FROM salary;

运算符

0 + - PRIOR
1 * /
2 =
3 > < !=
4 NOT
5 AND
6 OR.

运算符— < >

参见：

=, <, <=, >, >=

语法：

b1 <> b2

变量：

b1, b2： 数字变量。

例子：

PL/SQL
IF DAILY_SALES <> (MONTHLY_SALES/30)
THEN
TARGET: = -1
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE <> ‘09-FEB-96’;
DISCOUNT_T

```
----------
1222.12
```

运算符—>

参见：

=, !=, <=, <, >=

语法：

```
b1 > b2
```

变量：

b1, b2:：数字变量。

例子：

```
PL/SQL
IF DAILY_SALES > (MONTHLY_SALES/30) THEN
TARGET: = -1
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE > '09-FEB-96';
DISCOUNT_T
----------
168.5
```

运算符—> =

参见：

=, !=, <=, <, >

语法：

```
b1 >= b2
```

变量：

b1, b2： 数字变量。

例子：

```
PL/SQL
IF DAILY_SALES >= (MONTHLY_SALES/30) THEN
```

```
TARGET: = -1
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE >= ‘09-FEB-96’;
DISCOUNT_T
----------
2268.5
```

运算符—！ =

参见：

=, <, <=, < >, >, >=

语法：

```
b1 != b2
```

变量：

b1, b2：数字变量。

例子：

```
PL/SQL
IF DAILY_SALES != (MONTHLY_SALES/30) THEN
TARGET: = -1
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE != ‘09-FEB-96’;
DISCOUNT_T
----------
1222.12
```

运算符—*

参见：

+, -, /

语法：

```
b1 * b2
```

变量：

b1, b2：数字变量。

例子：

```
PL/SQL
AnnualSalary:= MonthlySalary * 12;
DiscountAmount:= ProductValue *
DiscountPercent;
SQL*Plus
SELECT SALES_AMOUNT * 10 DISCOUNT_TOTAL
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-96’;
DISCOUNT_T
----------
6850
```

运算符—+

参见：

-, *, /

语法：

b1 + b2 + ... + bn

变量：

b1, b2,..., bn:：数字变量。

例子：

```
PL/SQL
NewSalary:= CurrentSalary + Bonus + Commission + Raise;
TotalPayment:= Mortgage + Interest + RealEstateTaxes +
Insurance;
SQL
SELECT SALES_AMOUNT + 4.95 SHIPPING_TOTAL
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-96’;
SHIPPING_T
----------
699.95
```

运算符—-*

参见：

-, *, /

语法：

b1 - b2

变量：

b1, b2：数字变量。

例子：

PL/SQL

```
NetProfit:= Revenue - Expenses;
NetPay:= GrossIncome - Taxes;
```

SQL

```
SELECT SALES_AMOUNT - 10.00 DISCOUNT_TOTAL
FROM DAILY_SALES
WHERE SALES_DATE = '09-FEB-96';
DISCOUNT_T
----------
685
```

运算符—/

参见：

+, -, *

语法：

b1 / b2

变量：

b1, b2：数字变量。

例子：

PL/SQL

```
MonthlySalary:= AnnualSalary / 12;
```

SQL

```
SELECT SALES_AMOUNT / 10 DISCOUNT_TOTAL
FROM DAILY_SALES
WHERE SALES_DATE = '09-FEB-96';
```

```
DISCOUNT_T
----------
68.5
```

运算符—<=

参见：

=, !=, <, >, >=

语法：

```
b1 <= b2
```

变量：

b1, b2:：数字变量。

例子：

PL/SQL

```
IF DAILY_SALES <= (MONTHLY_SALES/30) THEN
TARGET: = -1
END IF;
```

SQL

```
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE <= ‘09-FEB-96’;
DISCOUNT_T
----------
168.5
```

运算符—=

参见：

!=, <, <=, < >, >, >=

语法：

```
b1 = b2
```

变量：

b1, b2：数字变量。

例子：

PL/SQL

```
IF DAILY_SALES = (MONTHLY_SALES/30)
THEN
TARGET: = 1
END IF;
```

SQL

```
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-96’;
DISCOUNT_T
----------
68.5
```

运算符—AND

参见：

OR, NOT

语法：

b1 AND b2

变量：

b1, b2:：任何变量或结果。

例子：

PL/SQL

```
IF (DAILY_SALES = MONTHLY_SALES/30) AND
(ANNUAL_SALES > 0 )THEN
BONUS: = 100
END IF;
```

SQL

```
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE <= ‘09-FEB-96’ AND SALES_DATE >= ‘09-MAR-96’;
DISCOUNT_T
----------
368.5
```

运算符—BETWEEN

参见：

IN, NOT IN, ANY,ALL, BETWEEN

语法：

WHERE column BETWEEN n AND m

变量：

n and m：任何相同的数据类型的变量。

例子：

SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE BETWEEN ‘09-FEB-96’ AND ‘10-FEB-96’;
DISCOUNT_T

68.5
12.5

运算符—IN

参见：

NOT IN, ANY, ALL, BETWEEN

语法：

WHERE column IN

例子：

SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE IN (‘09-FEB-96’, ‘10-FEB-96’);
DISCOUNT_T

68.5
12.5

运算符—IS NOT NULL

参见：

IS NULL

语法：

```
WHERE column IS NOT NULL
```

例子：

```
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE IS NOT NULL;
DISCOUNT_T
----------
12.2
15.3
```

运算符—IS NULL

参见：

IS NOT NULL

语法：

```
WHERE column IS NULL
```

例子：

```
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE IS NULL;
DISCOUNT_T
----------
10.2
```

运算符—NOT

参见：

AND, OR

语法：

```
b1 NOT b2
```

变量：

b1, b2：结果。

例子：

```
PL/SQL
IF NOT (DAILY_SALES != MONTHLY_SALES/30) THEN
BONUS: = 100
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE NOT (SALES_DATE = ‘09-FEB-96’);
DISCOUNT_T
----------
4468.5
```

运算符—NOT BETWEEN

参见：

其中运算符，如 IN, NOT IN, ANY, ALL, BETWEEN

语法：

```
WHERE column NOT BETWEEN n AND m
```

变量：

n and m：任何相同数据类型的变量。

例子：

```
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE NOT BETWEEN ‘09-FEB-96’ AND ‘10-FEB-96’;
DISCOUNT_T
----------
13
144.12
```

运算符—NOT IN

参见：

其中运算符，如 IN，ANY，ALL，BETWEEN

语法：

```
WHERE column NOT IN
```

例子：

```
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE NOT IN (‘09-FEB-96’, ‘10-FEB-96’);
DISCOUNT_T
----------
100.12
62.5
```

运算符—OR

参见：

AND，NOT

语法：

```
b1 OR b2
```

变量：

b1，b2：任何变量或结果。

例子：

```
PL/SQL
IF (DAILY_SALES = MONTHLY_SALES/30) OR (ANNUAL_SALES >
0 )THEN
BONUS: = 100
END IF;
SQL
SELECT SALES_AMOUNT FROM DAILY_SALES
WHERE SALES_DATE <= ‘09-FEB-96’ OR SALES_DATE >= ‘09-MAR-96’;
DISCOUNT_T
----------
3368.5
```

PRIOR

参见：

CONNECT BY

语法：

```
SELECT sql_expn FROM [user.]table WHERE where_condition
CONNECT BY [PRIOR] expn = [PRIOR] expn
START WITH expn = expn
ORDER BY expn
```

变量：

sql_expn:：一个有效的 SQL 表达式。
User：表的所有者。
Table： SQL 将再其上执行SELECT的表。
where_condition：SQL SELECT 的WHERE 条件。
Expn： 任何有效的表达式。

例子：

SQL

```
SELECT employee_name, department_name FROM employee
CONNECT BY emp_no = PRIOR department_no
ORDER BY department_no;
```

PROGRAM_ERROR

参见：

NO_DATA_FOUND,
TOO_MANY_ROWS,
STORAGE_ERROR

语法：

```
EXCEPTION
WHEN PROGRAM_ERROR THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一系列的语句。

例子：

```
PL/SQL
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN PROGRAM_ERROR THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘INTERNAL PROGRAM ERROR’ ;
END;
```

PROMPT

参见：

语法：

```
PROMPT [text]
```

变量：

text：你要SQL*Plus 显示的一行文本。

例子：

```
SQL
PROMPT
PROMPT The system will be shut down within 10 minutes.
```

PSEUDOCOLUMN

参见：

```
CURRVAL, LEVEL, NEXTVAL,
NULL, ROWID, ROWNUM,
SYSDATE, UID, USER
```

例子：

See sections on specific pseudocolumns for 例子：.

RAISE

参见：

EXCEPTIONS

语法：

RAISE exception_name

变量：

exception_name：一个的或用户定义的异常。

例子：

PL/SQL
```
IF past_due > 90 THEN
RAISE past_due_exception;
END IF;
```

RAWTOHEX

参见：

HEXTORAW

语法：

RAWTOHEX (r)

变量：

r：一个 RAW.类型的变量。

例子：

SQL
```
SELECT RAWTOHEX(animation) “Animation” FROM animation;
Animation
----------------------------------------
3D
```

RECORD

参见：

TABLE

语法：

```
TYPE type_name IS RECORD OF
( field_name1 {field_type | variable%TYPE |
table.column%TYPE | table%ROWTYPE }
[NOT NULL],
field_name2 {field_type | variable%TYPE |
table.column%TYPE | table%ROWTYPE }
[NOT NULL] )
```

变量：

type_name：一个在记录的后继声明中使用的类型标识符。
field_type：任何包括RECORD和TABLE的数据类型。

例子：

PL/SQL

```
DECLARE
TYPE LoanRecTyp IS RECORD
( cname CHAR(50),
loan_amount NUMBER (5));
loan_rec LoanRecTyp;

BEGIN
SELECT cust_name, loan_amt INTO loan_rec FROM
LOAN;
END;
DECLARE
TYPE TimeTyp IS RECORD
( minute SMALLINT,
hour SMALLINT);
TYPE MeetingTyp IS RECORD
( day DATE,
time TimeTyp,
place CHAR(20));
meeting MeetingTyp;
BEGIN
meeting.day := '26-JAN-51';
meeting.time.minute := 45;
```

```
meeting.time.hour := 12;
END;.
```

REFTOHEX

参见：

DEREF, MAKE_REF

语法：

REFTOHEX (expn)

变量：

expn：任何一个返回对象引用的表达式。

例子：

SQL

```
CREATE TABLE customer_tab (customer_no NUMBER,
loan REF loan_tab);
SELECT REFTOHEX (loan) FROM customer_tab;
```

REMARK

参见：

--, /* */

语法：

REM[ARK] [comment]

变量：

comment：你要包括在命令文件中的注释。

例子：

SQL

```
REM
REM Loan account information
```

RENAME

参见：

COPY

语法：

RENAME old TO new

变量：

old:：你要命名的表、视图、序列或私有同义词。
New：表、视图、序列或私有同义词的新名字。

例子：

SQL
RENAME accounting TO loanaccts
RENAME marketing TO sales

REPFOOTER

参见：

BTITLE, TTITLE, REPHEADER

语法：

REPF[OOTER] [PAGE] [printspec [text | variable]] | [OFF|ON]

变量：

printspec：printspec包括下面这些从句，SQL*Plus用它们来放置和格式化报表脚注的文本。

- COL n
- S[KIP] [n]
- TAB n
- LE[FT]
- CE[NTER]
- R[IGHT]
- BOLD
- FORMAT text

Text：报表脚注的文本。
Variable：含有下列系统保留值的文本。
Values：

- SQL.LNO (current line number)
- SQL.PNO (current page number)

- SQL.RELEASE (current Oracle release number)
- SQL.SQLCODE (current error code)
- SQL.USER (current Username)

例子：

SQL
```
REPFOOTER PAGE RIGHT ‘END OF REPORT’
REPFOOTER OFF
REPFOOTER ON.
```

REPHEADER

参见：

BTITILE, TTITLE, REPFOOTER

语法：

REPH[EADER] [PAGE] [printspec [text|variable]] | [OFF|ON]

变量：

printspec：包括下列从句， SQL*Plus用它们来放置和格式化报表表头的文本。
- COL n
- S[KIP] [n]
- TAB n
- LE[FT]
- CE[NTER]
- R[IGHT]
- BOLD
- FORMAT text

Text：报表表头文本。
Variable：含有下列系统保留值的变量。
Values：
- SQL.LNO (current line number)
- SQL.PNO (current page number)
- SQL.RELEASE (current Oracle release number)
- SQL.SQLCODE (current error code)
- SQL.USER (current Username)

例子：

SQL
```
REPHEADER PAGE CENTER ‘TOTAL SALES BY REGION’
REPHEADER PAGE BOLD ‘TOTAL UNITS SOLD’
REPHEADER OFF
```

REPHEADER ON.

REPLACE

参见：

TRANSLATE

语法：

REPLACE (‘x’ [,’y’,’z’])

变量：

x：应该在单引号中指定的字符型或varchar2 型变量，你想在这个变量中替换字符。
Y：应该在单引号中指定的字符型或varchar2 型变量，它指向你想替换的串。它是可选的。
Z：应该在单引号中指定的字符型或varchar2 型变量，它指向你想用来替换的串。它是可选的。

例子：

PL/SQL

```
Var1:= REPLACE (‘Oracle’, ‘Or’, ‘Mir’,);
```

SQL

```
SELECT REPLACE (‘Oracle’, ‘Or’, ‘Mir’) “Example “ FROM DUAL;
Example
-------
Miracle
```

REPLACE

参见：

TRANSLATE

语法：

REPLACE (‘x’ [,’y’,’z’])

变量：

x：应该在单引号中指定的字符型或varchar2 型变量，你想在这个变量中替换字符。
Y：应该在单引号中指定的字符型或varchar2 型变量，它指向你想替换的串。它是可选的。
Z：应该在单引号中指定的字符型或varchar2 型变量，它指向你想用来替换的串。它是可选的。

例子：

```
PL/SQL
Var1:= REPLACE (‘Oracle’, ‘Or’, ‘Mir’,);
SQL
SELECT REPLACE (‘Oracle’, ‘Or’, ‘Mir’,); ‘’Example of replacing
strings’’ FROM DUAL;
Example of replacing strings
---------------------------
Miracle
```

RETURN

参见：

Functions, Procedures, RAISE

语法：

```
RETURN [expression]
```

变量：

expression：一个有效表达式。有效表达式包括变量，操作符，文字，函数调用或常量。

例子：

```
PL/SQL
RETURN loan_bal;
RETURN 5;
```

REVOKE

参见：

GRANT

语法：

```
REVOKE system_privilege | role FROM user | role | PUBLIC
REVOKE system_privilege | role FROM user | role | PUBLIC
REVOKE object_privilege | ALL ON schema.object FROM user
| role | PUBLIC CASCADE CONSTRAINTS
```

变量：

system_privilege：从用户或角色收回的系统权限。
object_privilege：从用户或角色收回的系统权限。对象权限可为如下之一：

- ALTER
- DELETE
- EXECUTE
- INDEX
- INSERT
- REFERENCES
- SELECT
- UPDATE

Role：从用户收回的角色。

例子：

SQL

```
REVOKE ALTER TABLESPACE FROM john;
REVOKE GRANT ANY ROLE FROM todd;
REVOKE manager FROM imran;
REVOKE INSERT ON sales FROM javed;
REVOKE ALL ON marketing FROM terry;.
```

ROLLBACK

参见：

COMMIT, SAVEPOINT,
SET TRANSACTION, LOCK TABLE

语法：

```
ROLLBACK [WORK ] [TO [SAVEPOINT] savepoint_work]
```

变量：

WORK：可选关键字。使用这个关键字没有实际影响。它仅仅使这条语句看起来更完整一些。.
SAVEPOINT：可选关键字。使用这个关键字没有实际影响。它仅仅使这条语句看起来更完整一些。
savepoint_work：在当前事务中的在此之前的改变可以由 ROLLBACK 语句取消。

例子：

SQL

```
ROLLBACK;
ROLLBACK WORK;
ROLLBACK TO loan_changes;
ROLLBACK TO SAVEPOINT loan_changes;
```

```
ROLLBACK WORK TO SAVEPOINT loan_changes;
```

ROUND

参见：

TRUNC

语法：

```
ROUND (d [, 'fmt'])
```

变量：

d：有效的日期型变量。
Fmt：在引号中指定的字符变量、表示返回日期的方式。.

表A-20 **ROUND** 命令的变量

方式	结果
CC, SCC	世纪
SYYYY, YYYY, YEAR, SYEAR, YYY, YY Y	年（对6月1日进行取整）
IYYY, IY, IY I	ISO 年
Q	季度（对季度的第二个月份的16号进行取整）
MONTH, MON, MM, RM	月份(对月份的16号进行取整)
WW	将某周的某一天作为一年的第一天
WN	将某周的某一天作为一年的第一天
IW	将某周的某一天作为ISO年的第一天
W	将某周的某一天作为某月的第一天
DDD, DD, J	按天向上取整
DAY, DY, D	按周的第一天取整
HH, HH12, HH24	按小时取整
MI	按分钟取整

例子：

PL/SQL

```
FISCAL_YEAR:= ROUND (SYSDATE,'YEAR');
```

SQL

```
SELECT ROUND (SYSDATE,'YEAR') "CURRENT_FISCAL_YEAR" FROM DUAL;
CURRENT_F
---------
01-JAN-98
```

PL/SQL

```
Var1:= ROUND (124.1666, 2);
```

SQL

```
SELECT ROUND (124.16666, -2) “Rounded Value” FROM DUAL;
Rounded Value
-------------
100
```

ROWIDTOCHAR

参见：

CHARTOROWID

语法：

ROWIDTOCHAR (r)

变量：

r：ROWID类型的变量。.

例子：

SQL

```
SELECT ROWIDTOCHAR (ROWID) “VARCHAR2 TYPE” FROM EMPLOYEE;
VARCHAR2 TYPE
------------------
00000545.0001.0005
00000546.0000.0005
```

ROWTYPE_MISMATCH

参见：

NO_DATA_FOUND,
TOO_MANY_ROWS

语法：

```
EXCEPTION
WHEN ROWTYPE_MISMATCH THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：语句序列。

例子：

PL/SQL

```
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO emp_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN ROWTYPE_MISMATCH THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘ ROWTYPE_MISMATCH IN
CURSOR’ ;
END;
```

RPAD

参见：

LPAD, REPLACE

语法：

RPAD (‘x’, n [,’y’])

变量：

x：应该在单引号中指定的字符或varchar2 型变量。
n：正的数字变量。
y：应该在单引号中指定的字符或varchar2 型变量。该变量为可选的。若未指定，则缺省值为空字符。

例子：

PL/SQL

```
Var1:= RPAD (‘Oracle’, 1);
```

SQL

```
SELECT RPAD (‘Oracle is a good ‘, 25, ‘database’) “Example of
Right Padding” FROM DUAL;
Example of Right Padding
-----------------------
Oracle is a good database
```

RTRIM

参见：

LTRIM

语法：

RTRIM (‘x’ [,’y’])

变量：

x：应该在单引号中指定的字符型或varchar2 型变量。
Y：应该在单引号中指定的字符型或varchar2 型变量。.
这个变量是可选的。若未指定，则缺省值为空字符。

例子：

PL/SQL
```
Var1:= RTRIM (‘Oracle’, ‘Or’);
```
SQL
```
SELECT RTRIM (‘Mother Theresa, The’, ‘The’) “Example of Right
Trimming” FROM DUAL;
Example of Right
---------------
Mother Theresa,
```

SAVE

参见：

SET, EDIT, GET

语法：

SAV[E] filename[.ext] [CRE[ATE]|REP[LACE]|APP[END]]

变量：

filename：你将把缓冲区中的内容存入的文件。
ext：若使用文件后缀，缺省的文件后缀为SQL。

例子：

SQL
```
SAVE LOANRPT
SAVE ACCT.NEW
```

SAVEPOINT

参见：

COMMIT, ROLLBACK, SET
TRANSACTION, LOCK TABLE

语法：

```
SAVEPOINT savepoint_work
```

变量：

savepoint_work：在当前事务中用于存储当前的未声明的标识符。

例子：

SQL
```
SAVEPOINT loan_changes;
SAVEPOINT update_loan;
```

SELECT

参见：

基本查询语句 SELECT INTO, DECLARE
CURSOR, FETCH

语法：

```
SELECT [DISTINCT | ALL] {* | column1[, column2]...}
FROM {table_1 | (subquery)} [alias]
[, {table_2 | (subquery)} [alias]]...
[WHERE condition]
[CONNECT BY condition [START WITH condition]
[GROUP BY expn] [HAVING expn]
[{ UNION [ALL] | INTERSECT | MINUS } SELECT . . . ]
[ ORDER BY [expn ] [ ASC | DESC]
[ FOR UPDATE [OF [user.]table | view] column ]
[NOWAIT]
```

变量：

item_1,...,item_n：SELECT 语句选择的列。
table_1,...,table_n：用于选择数据的数据库表。
sub_query：向SELECT 语句提供值和值集的 SELECT语句。
Alias：一般是指表的短名字，或者是在 SELECT语句中引用的视阈。

column_1,...,column_n：给定数据和修改表中的指定列。
where_condition: 有效的 WHERE 子句。

例子：

```
PL/SQL
SELECT employee_id, employee_name, employee_title
FROM employees
WHERE employee_id = emp_id
ORDER BY employee_id;.
```

SELECT INTO

参见：

SELECT, DECLARE
CURSOR, FETCH

语法：

```
SELECT [DISTINCT | ALL] {* | item_1[, item_2]...}
INTO {variable_1[, variable_2]... | record_1}
FROM {table_1 | (subquery)} [alias]
[, {table_2 | (subquery)} [alias]]...
rest_of_select_statement;
```

变量：

item_1,...,item_n：SELECT INTO 语句选择并存入给定变量中的项。
variable_1,...,variable_n：将保存数据的变量。
record_1：用户定义的记录或 %ROWTYPE 记录使用 SELECT INTO 语句将行的值提取出并置放在该记录中。
table_1,...,table_n：从其中选择数据的数据库表。
sub_query：向SELECT INTO 语句提供值或侄集的SELECT语句。
Alias：一般是SELECT语句中引用的表，或视图的短名字。
column_1,...,column_n：指定将要被修改表的列。
sql_expression：有效的 SQL 表达式。
WHERE search_condition：有效的WHERE 子句，用于指定s将要被修改的行。
CURRENT OF cursor_name：与cursor_name指定的游标相关的 FETCH 语句处理的当前行。
rest_of_select_statement：在SELECT 语句中FROM子句之后的合法语句。

例子：

```
PL/SQL
SELECT employee_name, employee_id,
employee_title
```

```
INTO e_name, e_id, e_title
FROM employees
WHERE employee_id = emp_id;.
```

SET

参见：

EDIT, GET, SAVE

语法：

SET variable value

变量：

value：系统变量的值。
Variable：可以用 SQL*Plus 设置值的系统变量。该变量可以取下列的系统变量。

表A_21 SQL*Plus SET 命令中的变量

变量	语法
APPI[NFO]	{ON|OFF|text}
ARRAY[SIZE]	{20|n}
AUTO[COMMIT]	{OFF|ON|IMM[EDIATE]|n}
AUTOP[RINT]	{OFF|ON}
AUTOT[RACE]	{OFF|ON|TRACE[ONLY]}
EXP[LAIN]	[STAT[ISTICS]]
BLO[CKTERMINATOR]	{.|c}
CLOSECUR[SOR]	{OFF|ON}
CMDS[EP]	{;|c|OFF|ON}
COLSEP	{_|text}
COM[PATIBILITY]	{V6|V7|NATIVE}
CON[CAT]	{.|c|OFF|ON}
COPYC[OMMIT]	{0|n}
COPYTYPECHECK	{OFF|ON}
CRT	crt
DEF[INE]	{'&'|c|OFF|ON}
ECHO	{OFF|ON}
EDITF[ILE]	file_name[.ext]
EMBEDDED	{OFF|ON}
ESC[APE]	{|c|OFF|ON}
FEED[BACK]	{6|n|OFF|ON}
FLAGGER	{OFF|ENTRY|INTERMED[IATE]|FULL}
FLU[SH]	{OFF|ON}
HEA[DING]	{OFF|ON}
HEADS[EP]	{||c|OFF|ON}
LIN[ESIZE]	{80|n}

```
LONG                            {80|n}
LONGC[HUNKSIZE]                 {80|n}
MAXD[ATA]                        n
NEWP[AGE]                       {1|n}
NULL                             text
NUMF[ORMAT]                     format
NUM[WIDTH]                      {10|n}
PAGES[IZE]                       {24|n}
PAU[SE]                          {OFF|ON|text}
RECSEP                          {WR[APPED]|EA[CH]|OFF}
RECSEPCHAR                      {_|c}
SERVEROUT[PUT]                  {OFF|ON} [SIZE n] [FOR[MAT] {WRA[PPED]|
                                   WOR[D_WRAPPED]|TRU[NCATED]}].
SHOW[MODE]                      {OFF|ON}
SQLC[ASE]                        {MIX[ED]|LO[WER]|UP[PER]}
SQLCO[NTINUE]                   {> |text}
SQLN[UMBER]                     {OFF|ON}
SQLPRE[FIX]                      {#|c}
SQLP[ROMPT]                     {SQL>|text}
SQLT[ERMINATOR]                  {;|c|OFF|ON}
SUF[FIX]                          {SQL|text}
TAB                              {OFF|ON}
TERM[OUT]                        {OFF|ON}
TI[ME]                           {OFF|ON}
TIMI[NG]                         {OFF|ON}
TRIM[OUT]                       {OFF|ON}
TRIMS[POOL]                     {ON|OFF}
UND[ERLINE]                     {-|c|ON|OFF}
VER[IFY] {OFF|ON}
WRA[P]                           {OFF|ON}
```

例子：

```
SQL
SET AUTOPRINT ON
SET AUTOCOMMIT OFF
SET APPINFO ON
SET PAUSE ‘Enter the last name of the customer’
SET TIMING OFF.
```

SET ROLE

参见：

ALTER SESSION, ALTER SYSTEM

语法：

```
SET ROLE role_name IDENTIFIED BY password
SET ROLE ALL
SET ROLE ALL EXCEPT role_name
SET ROLE NONE
```

变量：

role_name：你想在当前会话中设置的角色。
password: 你要设置的角色的口令。

例子：

SQL
```
SET ROLE manager IDENTIFIED BY magic;
SET ROLE ALL;
SET ROLE ALL EXCEPT manager;
SET ROLE NONE;
```

SET TRANSACTION

参见：

COMMIT, ROLLBACK,
SAVEPOINT, LOCK TABLE

语法：

```
SET TRANSACTION
{ READ ONLY | READ WRITE | ISOLATION LEVEL {SERIALIZABLE
| READ COMMITTED}
| USE ROLLBACK SEGMENT rollback_segment_name}
```

变量：

rollback_segment_name：回滚段的名字。

例子：

SQL
```
SET TRANSACTION READ ONLY;
SET TRANSACTION READ WRITE;
```

SHOW

参见：

SET

语法：

SHO[W] option

变量：

option：该选项可以为下列子句之一：

- ALL
- BTI[TLE]
- ERR[ORS]
- [{FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY|TRIGGER|VIEW}schema.]name]
- LABEL
- LNO
- PNO
- REL[EASE]
- REPF[OOTER]
- REPH[EADER]
- SPOO[L]
- SQLCODE
- TTI[TLE]
- USER

例子：

SQL

```
SHOW USER
SHOW LNO
SHOW BTITLE
SHOW LABEL
SHOW LINESIZE
SHOW ALL.
```

SIGN

参见：

ABS

语法：

SIGN (x)

变量：

x：数字型变量。描述SIGN函数返回值的符号。下表表示三种可能情况下的结果值。

表A_22　SIGN函数在数的范围上的结果

x的值	结果
< 0	-1
0	0
> 0	1

例子：

PL/SQL
Var1:= SIGN (3);
SQL
SELECT SIGN (-5) “SIGN OF -5” FROM DUAL;
SIGN OF -5

-1.

SIN

参见：

COS, COSH, SINH,
TAN, TANH

语法：

SIN (n)

变量：

n：数字型变量。

例子：

PL/SQL
Var1:= SIN (20);
SQL
SELECT SIN (90) “Sine of 90 degrees” FROM DUAL;
Sine of 90 degrees

.89399666

SINH

参见：

COS, COSH, SIN,
TAN, TANH

语法：

SINH (n)

变量：

n：数字型变量。

例子：

PL/SQL
Var1:= SINH (20);
SQL
SELECT SINH (20) “Hyperbolic Sine of 20 deg” FROM DUAL;
Hyperbolic Sine of 20 deg

242582598

SOUNDEX

语法：

SOUNDEX (‘x’)

变量：

x：应该在单引号中指定字符或Varchar2型变量。

例子：

SQL
SELECT DPL_NAME FROM DENIED_PARTIES_LIST WHERE
SOUNDEX(DPL_NAME) = SOUNDEX(‘Saddam Hussain’) ;
DPL_NAME

Al Husseni
Sadda Al Sada.

SPOOL

参见：

EDIT, SAVE

语法：

SPO[OL] [filename[.ext] | OFF | OUT]

变量：

filename：你想输出（spool）的文件名。
ext：文件的后缀。缺省的后缀是LST（或LIS)。

例子：

SQL

```
SPOOL REPORT
SPOOL REPORT OUT
SPOOL REPORT OFF
```

SQLERRM

参见：

DECLARE EXCEPTION,
EXCEPTION, EXCEPTION
_INIT PRAGMA

语法：

SQLERRM [n]

变量：

n：表示 SQLCODE 取整数值。这是可选项。.

例子：

PL/SQL

```
error_message := SQLERRM (1401);
```

SQLPLUS

语法：

SQLPLUS [[-S[ILENT]] [logon] [start]]|

变量：

logon：用户标识，口令和数据库信息的格式如下：
username[/password] [@database_specification]||/NOLOG
start：在 SQL*Plus 中执行的命令文件。

例子：

SQL
SQLPLUS john/cage
SQLPLUS -S john/cage
SQLPLUS -S john/cage@LOAN
SQLPLUS john/cage @LOAN_RPT

SQRT

参见：

EXP, POWER

语法：

SQRT (n)

变量：

n：正的数字变量。

例子：

PL/SQL
Var1:= SQRT (ABS(Var2));
SQL
SELECT SQRT (4) “Square Root of 4” FROM DUAL;
Square Root of 4

2

START

参见：

@, DEFINE

语法：

STA[RT] filename[.ext] [arg_1,...,arg_N]

变量：

filename：在 SQL*Plus 中执行的命令文件。
Ext：命令文件的后缀，缺省的后缀是 SQL。
arg_1,...,arg_N：如果参数可用，则为命令文件参数。

例子：

SQL
START CALCSALARY 1
START LOAN_RPT
START ACCT.QRY

STDDEV

参见：

SUM, VARIANCE

语法：

STDDEV ([DISTINCT | ALL] expn)

变量：

DISTINCT：该选项使函数只考虑参数的不同值。是可选项。.
ALL：这个选项使函数考虑参数中包括所有重复值。该项是可选项。.
Expn：可以是SQL查询的一列或是几列的数学表达式。

例子：

PL/SQL
Var1:= STDDEV (1, 2, 3);
SQL
SELECT STDDEV (DAILY_SALES) “Standard Deviation” FROM SALES;
Standard Deviation

3416.842

STORAGE

参见：

CREATE CLUSTER, CREATE
INDEX, CREATE ROLLBACK
SEGMENT, CREATE SNAPSHOT,
CREATE SNAPSHOT LOG,
CREATE TABLE,
CREATE TABLESPACE

语法：

```
STORAGE ( [INITIAL n [K | M] )
[NEXT n [K | M] ]
[PCTINCREASE n]
[MINEXTENTS n]
[MAXEXTENTS n]
[OPTIMAL n [K | M] | NULL ]
[FREELISTS n ]
[FREELIST GROUP n ]
```

变量：

n：任何有效的正整数变量。

STORAGE_ERROR

参见：

PROGRAM_ERROR, NO_DATA_
FOUND, TOO_MANY_ROWS

语法：

```
EXCEPTION
WHEN STORAGE_ERROR THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：一系列的语句。

例子：

PL/SQL

```
BEGIN
OPEN loan_cur;
LOOP
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN STORAGE_ERROR THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘STORAGE ERROR’ ;
END;
```

STORE

参见：

SAVE, GET

语法：

STORE {SET} filename[.ext] [CRE[ATE]|REP[LACE]| APP[END]]

变量：

filename：你要在其中存储SQL*Plus环境属性的文件名。
Ext：若后缀可用，则为文件后缀。

例子：

SQL
```
STORE SET MYENV
STORE SET MYENV.NEW APPEND
STORE SET MYENV CREATE
STORE SET MYENV.OLD REPLACE
```

SUBSTR

参见：

SUBSTRB

语法：

SUBSTR (‘x’ [,y, z])

变量：

x：在单引号中指定的字符或Varchar2型的变量。.
y：指出替换串起市始位置的数字型变量。是可选项。如果这个项指定为0，则作为1处理。
Z：指出需要被替换的字符总数的数字型变量。是可选项。

例子：

```
PL/SQL
Var1:= SUBSTR (‘Oracle’, 1, 3);
SQL
SELECT SUBSTR (‘Oracle’, 1, 3) “Substr” FROM DUAL;
Sub
---
Ora
```

SUBSTRB

参见：

SUBSTR

语法：

```
SUBSTRB (‘x’ [,y, z])
```

变量：

x：在单引号中指定的字符或Varchar2型的变量。.
y：指出替换串起市始位置的数字型变量。是可选项。如果这个项指定为0，则作为1处理。
Z：指出需要被替换的字符总数的数字型变量。是可选项。

例子：

```
PL/SQL
Var1:= SUBSTRB (‘Oracle’, 1, 3);
SQL
SELECT SUBSTRB (‘Oracle’, 1, 3) “Substr “ FROM DUAL;
Sub
---
Ora.
```

SUM

参见：

Group by AVG, MAX, MIN,
STDDEV, VARIANCE

语法：

```
SUM ( [DISTINCT | ALL] expn)
```

变量：

DISTINCT：该选项使函数只考虑参数的不同值。是可选项。.
ALL：这个选项使函数考虑重复值的所有参数值。该项是可选项。.
Expn：可以是SQL查询的一列或是几列的数学表达式。

例子：

```
PL/SQL
Var1:= SUM (1, 2, 3);
SQL
SELECT SUM (DAILY_SALES) “SUM” FROM SALES;
SUM
--------------
1212.1221
```

SYSDATE

参见：

PSEUDOCOLUMN

语法：

```
SYSDATE
```

例子：

```
SQL
SELECT SYSDATE “THE DATE TODAY” FROM DUAL;
THE DATE
---------
09-NOV-97
```

TABLE

参见：

RECORD

语法：

```
TYPE type_name IS TABLE OF
{ column_type | variable%TYPE | table.column%TYPE } [NOT
NULL]
INDEX BY BINARY_INTEGER ;
```

变量：

type_name：在接下来的PL/SQL 表中声明的类型标识。
column_type：诸如CHAR，DATE，或 NUMBER的数据类型。

例子：

PL/SQL

```
DECLARE
TYPE CnameTabTyp IS TABLE OF CHAR(10)
INDEX BY BINARY_INTEGER;
Cname_tab CnameTabType;
BEGIN
Cname_tab (1) := ‘LARRY’;
END;
```

TAN

参见：

COS, COSH, SIN,
SINH, TANH

语法：

```
TAN (n)
```

变量：

n：数字型变量。

例子：

PL/SQL

```
Var1:= TAN (20);
```

SQL

```
SELECT TAN (20) “Tangent of 20 degrees” FROM DUAL;
Tangent of 20 degrees
---------------------
2.2371609
```

TANH

参见：

COS, COSH, SIN,
SINH, TAN

语法：

TANH (n)

变量：

n：数字型变量。

例子：

PL/SQL
Var1:= TANH (20);
SQL
SELECT TANH (20) “Hyperbolic Tangent of 20 deg” FROM DUAL;
Hyperbolic Tangent of 20 deg

1

TIMEOUT_ON_RESOURCE

参见：

NO_DATA_FOUND,
TOO_ MANY_ROWS

语法：

EXCEPTION
WHEN TIMEOUT_ON_RESOURCE THEN
statement_1,...,statement_n

变量：

statement_1,...,statement_n：一系列语句。

例子：

PL/SQL
BEGIN
OPEN loan_cur;
LOOP

```
FETCH loan_cur INTO loan_rec;
EXIT WHEN loan_cur%NOTFOUND;
END LOOP;
EXCEPTION
WHEN TIMEOUT_ON_RESOURCE THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘ TIMEOUT ON RESOURCE ‘ ;
END;
```

TIMING

参见：

CLEAR, SET

语法：

TIMI[NG] [START text|SHOW|STOP]

变量：

text: 计时器的名字。

例子：

SQL

```
TIMING START new_timer
TIMING SHOW
TIMING STOP
```

TO_CHAR (date)

参见：

TO_DATE, FORMAT —Date

语法：

TO_CHAR (d [, ‘x’, ‘nls_parm’])

变量：

d：有效的日期变量。
X：在单引号中指定的字符型变量，它表示返回的日期类型。是可选项。
nls_parm：指定返回的年、月和日的语言。该参数是可选的。缺省的，它将采用缺省的日期语言。这个参数可以指定为：‘NLS_DATE_LANGUAGE = language’。

例子：

PL/SQL
FISCAL_YEAR:= TO_CHAR (SYSDATE,’YEAR’);

SQL
SELECT TO_CHAR (SYSDATE,’YEAR’) “CURRENT_FISCAL_YEAR” FROM
DUAL;
CURRENT_FISCAL_YEAR

NINETEEN NINETY-SEVEN.

TO_CHAR (label)

参见：

TO_LABEL

语法：

TO_CHAR (mls [, ‘x’])

变量：

mls：有效的MLSLABEL 变量。
X：在单引号中指定的字符变量，它表示返回的字符串的形式。该项是可选的。

TO_CHAR (number)

参见：

TO_NUMBER,
FORMAT —NUMBER

语法：

TO_CHAR (n [, ‘x’, ‘nls_parm’])

变量：

n：有效的数字变量。
X：在引号中指定的字符变量，它表示返回的数字形式。是可选项。
nls_parm：指定返回数字的语言。该项是可选项。这个参数可以如下指定
‘NLS_NUMERIC_CHARACTERS = ‘’,.’”
‘NLS_CURRENCY = ‘’$’”

例子：

```
PL/SQL
FISCAL_YEAR:= TO_CHAR (786);
SQL
SELECT TO_CHAR(786,'L99G999D99MI','NLS_NUMERIC_CHARACTERS =
‘’,.’’ NLS_CURRENCY = ‘’$’’ ‘) “Net Payable” FROM DUAL;
Net Payable
-------------------
$786,00.
```

TO_DATE (char)

参见：

TO_CHAR, TO_NUMBER,
FORMAT —DATE

语法：

TO_DATE (‘s’ [, ‘x’, ‘nls_parm’])

变量：

s：在引号中指定的CHAR 或 VARCHAR2 型变量。
X：在引号中指定的字符变量，它表示返回的日期形式。该项是可选的。
Nls_parm：指定返回的年、月、日的语言。该参数是可选的。缺省的，它将采用缺省的日期语言。
这个参数可以指定为‘NLS_DATE_LANGUAGE = language’。

例子：

```
PL/SQL
FISCAL_YEAR:= TO_DATE (‘26-JAN-47’);

SQL
SELECT TO_DATE (‘01/26/47’, ‘mm/dd/yy’) “REPUBLIC DAY” FROM
DUAL;
REPUBLIC
---------
26-JAN-47.
```

TO_LABEL (char)

参见：

TO_CHAR

语法：

TO_LABEL (‘s’ [, ‘x’])

变量：

s：在引号中指定的有效的CHAR 或 VARCHAR2 型变量。
X：在引号中指定的字符型变量，它表示返回的标号方式。该项为可选项。.

TO_MULTI_BYTE (char)

参见：

TO_SINGLE_BYTE

语法：

TO_MULTI_BYTE (‘s’)

变量：

s：在引号中指定的有效的 CHAR 或 VARCHAR2 型变量。

TO_NUMBER (char)

参见：

TO_CHAR, TO_DATE,
FORMAT —NUMBER

语法：

TO_NUMBER (‘s’ [, ‘x’, ‘nls_parm’])

变量：

s：在引号中指定的 CHAR 或 VARCHAR2 型变量。
X：在引号中指定的字符变量，表示返回的数字的形式，该项是可选项。
Nls_parm：指定需要返回值的字符。该参数是可选的。缺省的，它将采用缺省的数字形式。
这个参数如下指定：

‘NLS_NUMERIC_CHARACTERS = ‘’,.’’’
‘NLS_CURRENCY = ‘’$’’’

例子：

PL/SQL
FISCAL_YEAR:= TO_NUMBER (‘1947’);
SQL
SELECT TO_NUMBER (‘1947’) “FISCAL_YEAR” FROM DUAL;
FISCAL_YEAR

1947.

TO_SINGLE_BYTE (char)

参见：

TO_MULTI_BYTE

语法：

TO_SINGLE_BYTE (‘s’)

变量：

s：在引号中指定有效的 CHAR 或 VARCHAR2 型变量。

TOO_MANY_ROWS

参见：

NO_DATA_FOUND

语法：

EXCEPTION
WHEN TOO_MANY_ROWS THEN
statement_1,...,statement_n

变量：

statement_1,...,statement_n：一系列语句。

例子：

```
PL/SQL
DECLARE
total_salary NUMBER;
BEGIN
SELECT salary INTO total_salary FROM employee;
EXCEPTION
WHEN TOO_MANY_ROWS THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR =
‘ MORE THAN 1 ROW FOR THE
INTO VARIABLE’ ;
END;
```

TRANSLATE

参见：

REPLACE

语法：

TRANSLATE (‘x’ [,’y’,’z’])

变量：

x：在单引号中指定的字符或varchar2型变量，在该变量中替换字符。
Y：在单引号中指定的字符或varchar2型变量，它指向你想替换的字符。
Z：在单引号中指定的字符或varchar2型变量，它指向你想替换的字符。.

例子：

```
PL/SQL
Var1:= TRANSLATE (‘Oracle’, ‘Or’, ‘Mir’,);

SQL
SELECT TRANSLATE (‘Oracle’, ‘Or’, ‘Mir’,); ‘’Example of
translating strings’’ FROM DUAL;
Example of translating strings
----------------------------
Miracle.
```

TRUNC (date)

参见：

ROUND

语法：

TRUNC (d [, ‘x’])

变量：

d：有效的日期变量。
X：在引号中指定的字符变量，它表示返回的日期方式。

表A_24 TRUNC (date) 命令的方式

方式	结果
CC, SCC	世纪
SYYYY, YYYY, YEAR, SYEAR, YYY, YY, Y	年（对6月1日舍入）
IYYY, IY, IY, I	ISO 年
Q	季节（对季节的第二月份的16号舍入）
MONTH, MON, Month, MM, RM	月(对16号舍入)
WW	将某周的某一天作为某年的第一天
IW	将某周的某一天作为ISO年的第一天
W	将某周的某一天作为月份的第一天
DDD, DD, J	按日期取整
DAY, DY, D	按周的第一天取整
HH, HH12, HH24	按小时取整
MI	按分钟取整

例子：

PL/SQL

```
FISCAL_YEAR:= TRUNC (SYSDATE,’YEAR’);
```

SQL

```
SELECT TRUNC (SYSDATE,’YEAR’) “CURRENT_FISCAL_YEAR” FROM DUAL;
CURRENT_F
---------
01- JAN-97.
```

TRUNC (number)

参见：

ROUND

语法：

```
TRUNC (x [, y])
```

变量：

x: 数字型变量
y: 数字型变量（可选）

例子：

```
PL/SQL
Var1:= TRUNC (124.1666, 2);
SQL
SELECT TRUNC (124.16666, -2) “Rounded Value” FROM DUAL;
Rounded Value
-------------
100
SELECT TRUNC (124.16666, 2) “Rounded Value” FROM DUAL;
Rounded Value
-------------
124.16
```

TRUNCATERUNCATE

```
DELETE, DROP TABLE
```

语法：

```
TRUNCATE [TABLE | CLUSTER]
schema.[table][cluster] [DROP | REUSE STORAGE]
```

变量：

table：希望删除行的表名
cluster：希望删除行的CLUSTER名
schema：表或 cluster的schema名

例子：

```
SQL
TRUNCATE TABLE inventory;
TRUNCATE CLUSTER marketing DROP STORAGE;
TRUNCATE TABLE loan REUSE STORAGE;
```

TTITLE

参见：

BTITLE,REPHEADER, REPFOOTER

语法：

TTI[TLE] [printspec [text|variable] ...]|[OFF|ON]

变量：

printspec：printspec 包含 SQL*Plus 将用来作为题目的如下语句：

- COL n
- S[KIP] [n]
- TAB n
- LE[FT]
- CE[NTER]
- R[IGHT]
- BOLD
- FORMAT text

text:：题目的内容。

Variable：包含如下系统维护的变量：

- SQL.LNO (当前行数)
- SQL.PNO (当前页数)
- SQL.RELEASE (当前 Oracle 版本号)
- SQL.SQLCODE (当前错误代码)
- SQL.USER (当前用户名)

例子：

SQL

```
TTITLE LEFT ‘ADVENTURE REPORT’
TTITLE OFF
TTITLE ON.
```

UID

参见：

USER

语法：

UID

变量：

无

例子：

SQL
```
SELECT UID FROM DUAL;
UID
-----
33
```

UNDEFINE

参见：

DEFINE, START

语法：

UNDEF[INE] variable_1 ... variable_n

变量：

variable_1 ... variable_n：希望删除的变量

例子：

SQL
```
UNDEFINE myVar1
UNDEFINE loan_bal acct_bal
UNDEFINE myVar1 myVar2 myVar3
```

UNION

参见：

UNION ALL, INTERSECT, MINUS

语法：

b1 UNION b2

变量：

b1 和 b2 是 SELECT 语句

例子：

```
SQL
SELECT SALES_AMOUNT
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-96’
UNION
SELECT SALES_AMOUNT
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-97’ ;
SHIPPING_T
----------
699.95
122.95
```

UNION ALL

参见：

ALL UNION, INTERSECT,
MINUS, SELECT

语法：

b1 UNION ALLb2

变量：

b1 和 b2 是 SELECT 语句

例子：

```
SQL
SELECT SALES_AMOUNT
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-96’
UNION ALL
SELECT SALES_AMOUNT
FROM DAILY_SALES
WHERE SALES_DATE = ‘09-FEB-97’ ;
SHIPPING_T
----------
699.12
122.12
111.12
112.33
```

UPDATE

参见：

INSERT, DELETE, SELECT,
SELECT INTO

语法：

```
UPDATE {table | (sub_query)} [alias]
SET { column_name = {sql_expression | (sub_query)}
| (column_name[, column_name]...) = (subquery)}
[, { column_name = {sql_expression | (subquery)}
| (column_name[, column_name]...) = (subquery)}]...
[WHERE {search_condition | CURRENT OF cursor_name}];
```

变量：

table：希望更新的表名
sub_query：一个 SELECT 子查询
alias：别名
column_1,...,column_n：列名
sql_expression：有效的 SQL表达式
WHERE search_condition：有效的 WHERE 子句

例子：

SQL

```
UPDATE salary
SET employee_salary = Total_Salary
WHERE employee_number:= emp_id;
UPDATE employees
SET employee_active_flag = ‘N’
WHERE employee_id in
(SELECT employee_id
FROM department_head
WHERE department_active_flag = ‘N’);.
```

UPPER

参见：

INITCAP, LOWER

语法：

```
UPPER (‘x’)
```

变量：

x：单引号中指定的字符型或Varchar2型变量。

例子：

```
PL/SQL
Var1:= UPPER (‘oracle’);
SQL
SELECT UPPER (‘oracle is a good database’) “UpperCase Sentence”
FROM DUAL;
UpperCase Sentence
------------------------
ORACLE IS A GOOD DATABASE
```

USER

参见：

UID, PSEUDOCOLUMN

语法：

USER

变量：

无

例子：

```
SQL
SELECT USER FROM DUAL;
USER
-----
FED_REG
```

USERENV

参见：

UID, USER

语法：

USERENV (‘type’)

变量：

type：可以采用如下的选项并且必须在引号中指定：

表A_25　USERENV 命令的选项设置

选项	描述
ENTRYID	返回当前用户会话输入的 ID
SESSIONID	返回当前用户会话ID
TERMINAL	返回当前系统会话的操作系统标识
OSDBA	若当前用户会话有OSDBA 角色则返回TRUE
LABEL	返回当前用户会话标号.
LANGUAGE	返回当前用户会话的语言和区域
CLIENT_INFO	为当前用户会话返回client_info域的值。这个值由 dbms_application_info.set_client_info 过程设置
LANG	返回以ISO 省略形式的3字符表示的当前用户会话使用的语言

例子：

```
SQL
SELECT USERENV(‘SESSIONID’) “SESSION ID” FROM DUAL;
SESSION ID
---------------
21323.
```

VALUE_ERROR

参见：

NO_DATA_FOUND,
TOO_MANY_ROWS

语法：

```
EXCEPTION
WHEN VALUE_ERROR THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n: 一序列语句

例子：

```
PL/SQL
DECLARE
name VARCHAR2(10);
total_salary NUMBER;
BEGIN
```

```
total_salary:= total_salary * name;
EXCEPTION
WHEN VALUE_ERROR THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘VALUE_ERROR -
INVALID MULTIPLICATION’ ;
END;
```

VARIABLE

参见：

ACCEPT, DEFINE

语法：

```
VAR[IABLE]
[variable [NUMBER|CHAR|CHAR (n)|VARCHAR2 (n)| REFCURSOR]]
```

变量：

variable: 声明的二进制变量名

例子：

SQL

```
VARIABLE quota NUMBER
VARIABLE last_name CHAR(35)
VARIABLE r REFCURSOR
```

VARIABLE ASSIGNMENT

参见：

DECLARE, NULL

语法：

```
a:= 5
```

变量：

a: 用于赋值变量

例子：

PL/SQL

```
Total_Salary:= Base_Salary + Commission + Bonus;
```

VARIANCE

参见：

SUM, STDDEV, VARIANCE

语法：

```
VARIANCE ( [DISTINCT | ALL] expn)
```

变量：

DISTINCT: 变量的目标值
ALL: 包括重复的在内的所有值
expn: 可以是查询的列名或表达式

例子：

PL/SQL

```
Var1:= VARIANCE (1, 2, 3);
```

SQL

```
SELECT VARIANCE (DAILY_SALES) “ VARIANCE “ FROM SALES;
VARIANCE
--------------
34.12
```

VSIZE

参见：

LENGTH

语法：

```
VSIZE (v)
```

变量：

v：任何正整数

例子：

SQL

```
SELECT VSIZE (‘ORACLE’) “VSIZE” FROM DUAL;
VSIZE
```

```
--------
6
SELECT VSIZE (21.543) “VSIZE” FROM DUAL;
VSIZE
--------
4
```

RESERVED WORDS

```
ACCESS ADD ALL ALTER
AND ANY AS ASC
AUDIT
BETWEEN BY
CHAR CHECK CLUSTER COLUMN
COMMENT COMPRESS CONNECT CREATE
CURRENT
DATE DECIMAL DEFAULT DELETE
DESC DISTINCT DROP
ELSE EXCLUSIVE EXISTS
FILE FLOAT FOR FROM
GRANT GROUP
HAVING
IDENTIFIED IMMEDIATE IN INCREMENT
INDEX INITIAL INSERT INTEGER
INTERSECT INTO IS
LEVEL LIKE LOCK LONG
MAXEXTENTS MINUS MODE MODIFY
NOAUDIT NOCOMPRESS NOT NOWAIT
NULL NUMBER
OF OFFLINE ON ONLINE
OPTION OR ORDER
PCTFREE PRIOR PRIVILEGES PUBLIC
RAW RENAME RESOURCE REVOKE
ROW ROWID ROWLABEL ROWNUM
ROWS
SELECT SESSION SET SHARE
SIZE SMALLINT START SUCCESSFUL
SYNONYM SYSDATE
TABLE THEN TO TRIGGER
UID UNION UNIQUE UPDATE
USER
VALIDATE VALUES VARCHAR VARCHAR2
VIEW
```

WHENEVER WHERE WITH.

WHENEVER OSERROR

参见：

WHENEVER SQLERROR

语法：

```
WHENEVER OSERROR {EXIT [SUCCESS|FAILURE|n|variable]
[COMMIT|ROLLBACK]|CONTINUE [COMMIT|ROLLBACK|NONE]}
```

变量：

n：SQL*Plus返回的成功或失败的代码
variable：SQL*Plus返回的成功或失败的变量

例子：

SQL

```
WHENEVER OSERROR EXIT COMMIT
WHENEVER OSERROR EXIT retcode ROLLBACK.
```

WHENEVER SQLERROR

参见：

WHENEVER OSERROR

语法：

```
WHENEVER SQLERROR {EXIT [SUCCESS|FAILURE|n|variable]
[COMMIT|ROLLBACK]|CONTINUE [COMMIT|ROLLBACK|NONE]}
```

变量：

n：SQL*Plus返回的成功或失败的代码
variable：SQL*Plus返回的成功或失败的变量

例子：

SQL

```
WHENEVER SQLERROR EXIT COMMIT
WHENEVER SQLERROR EXIT FAILURE retcode ROLLBACK.
```

WHILE-LOOP

参见：

LOOP, EXIT, FOR-LOOP,
EXIT-WHEN

语法：

```
WHILE condition1 is true LOOP
statement_1,...,statement_n;
END LOOP
```

变量：

condition_1：PL/SQL 条件
statement_1,...,statement_n：一序列PL/SQL 执行语句

例子：

PL/SQL

```
WHILE Total_Salary > 0 LOOP
Total_Salary:= Base_Salary + Commission + Bonus;
UPDATE salary SET employee_salary = Total_Salary WHERE
employee_number:= emp_id;
END LOOP;
```

ZERO_DIVIDE

参见：

INVALID_NUMBER

语法：

```
EXCEPTION
WHEN ZERO_DIVIDE THEN
statement_1,...,statement_n
```

变量：

statement_1,...,statement_n：a sequence of statements

说明：

如果用0 除，PL/SQL 将隐含地启动ZERO_DIVIDE 例外。 相关的错误代码是ORA-01476

例子：

PL/SQL

```
DECLARE
total_salary NUMBER;
BEGIN
total_salary:= total_salary / 0;
EXCEPTION
WHEN ZERO_DIVIDE THEN
UPDATE APPLICATION_ERROR_TABLE
SET ERROR = ‘DIVISION BY ZERO
INVALID’ ;
END;
```

附录 B　ORACLE OFA 结构

B1 表是一般的 OFA 安装等级文件映射：

文件映射						说明
/						根安装点
	u01/					Oracle 软件安装点#1
		App/				App 软分支
			Oracle/			Oracle 软件所有者主目录
				Admin/		Oracle 管理文件的分支
					TAR/	支持日志分支
					db_name1/	数据库管理分支
					bdump/	后台进程卸出目录
					cdump/	核心卸出目录
					udump/	用户卸出目录
					create/	数据库建立 SQL 脚本
					pfile/	数据库初始化参数文件
					db_name2/	db_name2 数据库管理分支
				doc/		联机文档
				local/		本地 Oracle 软件分支
					aps6/	Oracle6 管理包
					aps7/	Oracle7 管理包
				product/		分布文件
					8.1.7/	Oracle8I 8.1.1.7 实例
			oraInventory			Oracle8I 详细目录清单分支
				logs		安装日志文件
		home				登录根目录分支
			ltb/			某用户的根目录
			sbm/			某用户的根目录
	u02/					用户数据安装点 #2
		home/				登录根目录分支
			cvm/			某用户的根目录
			vrm			某用户的根目录
		oradata/				Oracle 数据目录分支
			db_name1/			db_name1 数据库目录分支
			db_name2/			db_name2 数据库目录分支
	u03/					用户数据安装点#3
		oradata/				Oracle 数据目录分支
			db_name1/			db_name1 数据库目录分支
			db_name2/			db_name2 数据库目录分支
	u04/					用户数据安装点#4
		oradata/				Oracle 数据目录分支
			db_name1/			db_name1 数据库目录分支

			db_name2/			db_name2 数据库目录分支
	/var					
		opt/				
			oracle/			oratab 和 oraInst.loc 位置
	/usr					
		local/				
			bin/			oraenv, coraenv, 和 dbhome 脚本

B2 表是两个并行服务器的 OFA 安装等级文件映射：

文件映射				说明
u01/				
	App/oracle /admin/sab			可管理的 sab 数据库目录
	adhoc/			各种脚本目录
	arch/			所有实例的归档目录
		Redo001.arc		归档日志文件
	bdump/			后台进程卸出文件目录
		inst1/		实例 inst1 卸出目录
		Inst2/		实例 inst2 卸出目录
	cdump/			核心卸出文件目录
		inst1/		实例 inst1 核心卸出目录
		Inst2/		实例 inst2 核心 卸出目录
	create/			建立脚本目录
		1-rdbms.sql		建立 inst 数据库的 SQL 脚本
	exp/			导出目录
		19990120full.dmp		January 20, 1999 全部导出文件
		export/		导出参数文件（parfiles）目录
		import/		导入参数文件（parfiles）目录
	logbook/			Inst 运行日志目录
		inst1/		实例 inst1 报告目录
			params.lst	实例 inst1 的 v$parameter 报告
		inst2/		实例 inst2 报告目录
			params.lst	实例 inst2 的 v$parameter 报告
		user.lst		dba_users 报告
	pfile/			实例参数文件目录
		inst1/		实例 inst1 参数目录
			init	实例 inst1 参数
		Inst2/		实例 inst2 参数目录
			init	实例 inst2 参数
	udump/			用户卸出文件目录
		inst1/		实例 inst1 用户卸出文件目录
		Inst2/		实例 inst2 用户卸出文件目录

附录 C　动态性能(V$)视图

本附录介绍动态性能视图。这些视图一般作为 V$视图引用。本附录包括下列内容：

- 动态性能视图。
- 视图说明。

C.1 动态性能视图

Oracle 服务器包括一组基础视图，这些视图由服务器维护，系统管理员用户 SYS 可以访问它们。这些视图被称为动态性能视图，因为它们在数据库打开和使用时不断进行更新，而且它们的内容主要与性能有关。

虽然这些视图很像普通的数据库表，但它们不允许用户直接进行修改。这些视图提供内部磁盘结构和内存结构方面的数据。用户可以对这些视图进行查询，以便对系统进行管理与优化。

文件 CATALOG.SQL 包含这些视图的定义以及公用同义词。必须运行 CATALOG.SQL 创建这些视图及同义词。

C.1.1 V$ 视图

动态性能视图由前缀 V_$标识。这些视图的公用同义词具有前缀 V$。数据库管理员或用户应该只访问 V$对象，而不是访问 V_$对象。

动态性能视图由企业管理器和 Oracle Trace 使用，Oracle Trace 是访问系统性能信息的主要界面。

建议： 一旦实例启动，从内存读取数据的 V$视图就可以访问了。从磁盘读取数据的视图要求数据库已经安装好了。

警告：给出动态性能视图的有关信息只是为了系统的完整性和对系统进行管理。公司并不承诺以后也支持这些视图。

C.1.2 GV$ 视图

在 Oracle 中，还有一种补充类型的固定视图。即 GV$（Global V$，全局 V$）固定视图。对于本章介绍的每种V$ 视图(除V$CACHE_LOCK、V$LOCK_ACTIVITY、V$LOCKS_WITH_COLLISIONS 和 V$ROLLNAME 外)，都存在一个 GV$视图。在并行服务器环境下，可查询 GV$视图从所有限定实例中检索 V$视图的信息。除 V$信息外，每个 GV$视图拥有一个附加的名为 INST_ID 的整型列。INST_ID 列显示从其获得相关的 V$视图信息的实例号。INST_ID 列可用作一个从可得到的实例集检索 V$信息的过滤器。例如，下列查询：

SELECT * FROM GV$LOCK WHERE INST_ID = 2 OR INST_ID = 5

表示从实例 2 和 5 上的 V$ 视图中检索信息。

GV$视图可用来返回用 OPS_ADMIN_GROUP 参数定义的实例组上的信息。

GV$视图具有下列限制：

- 在安装数据库的所有实例上，PARALLEL_MAX_SERVERS 参数的值必须大于零。
- 为了成功完成查询，必须至少用一个成员来定义 OPS_ADMIN_GROUP 参数。

C.1.2 访问动态性能表

在安装之后，仅有用户名为 SYS 或具有 SYSDBA ROLE 的用户能够访问动态性能表。

C.2 视图说明

本节列出动态性能视图的列和公用同义词。

1．V$ACCESS

此视图显示数据库中当前锁定的对象及访问它们的会话。

列	数据类型	说明
SID	NUMBER	访问一个对象的会话号
OWNER	VARCHAR2(64)	对象的拥有者
OBJECT	VARCHAR2(1000)	对象号
TYPE	VARCHAR2(12)	对象的类型标识符

2．V$ACTIVE_INSTANCES

对所有当前使数据库安装的实例，此视图将实例名映射到实例号。

列	数据类型	说明
INST_NUMBER	NUMBER	实例号
INST_NAME	VARCHAR2(60)	实例名

3．V$AQ

此视图描述数据库中队列的统计数据。

列	数据类型	说明
QID	NUMBER	唯一的队列标识符
WAITING	NUMBER	队列中处于“WAITING”状态的消息号
READY	NUMBER	队列中处于“READY”状态的消息号
EXPIRED	NUMBER	队列中处于“EXPIRED”状态的消息号
TOTAL_WAIT	NUMBER	队列中处于“READY”消息的总等待时间
AVERAGE_WAIT	NUMBER	队列中处于“READY”消息的平均等待时间

4．V$ARCHIVE

此视图包含需要归档的重做日志文件的信息。每行提供一个线程的信息。这些信息在V$LOG中也可得到。Oracle建议使用V$LOG。更多信息，请参阅“V$LOG”。

列	数据类型	说明
GROUP#	NUMBER	日志文件组号
THREAD#	NUMBER	日志文件线程号
SEQUENCE#	NUMBER	日志文件序列号
CURRENT#	VARCHAR2(3)	当前在使用的归档日志
FIRST_CHANGE#	NUMBER	存储在当前日志中的第一个SCN

5． V$ARCHIVE_DEST

对于当前实例,此视图描述所有归档日志目标、它们的当前值、模式以及状态。

列	数据类型	说明
DEST_ID	NUMBER	ID(1-5)
STATUS	VARCHAR2(9)	状态：VAILID：初始化并可得到；

		INACTIVE：无目标信息； DEFERRED：用户手工禁用； ERROR：打开或拷贝中出错； DISABLED：出错后禁用； BAD PARAM：参数有错；
BINDING	VARCHAR2(9)	成功请求：MANDATORY-必须成功，否则 OPTIONAL-不需要成功（依赖于 LOG_ARCHIVE_MIS_SUCCEED_DEST）
NAME_SPACE	VARCHAR2(7)	定义范围：SYSTEM-系统定义或 SESSION-会话定义
TARGET	VARCHAR2(7)	目标：PRIMARY-拷贝到主目标或 STANDBY-拷贝到备用目标
REOPEN_SECS	VARCHAR2(7)	按秒计算的重试时间（出错之后）
DESTINATION	VARCHAR2(256)	目标文本串（转换为主位置或备用服务器名）
FAIL_DATE	DATE	最后出错的日期和时间
FAIL_SEQUENCE	NUMBER	最后出错的日志序列号
FAIL_BLOCK	NUMBER	最后出错的块号
ERROR	VARCHAR2(256)	最后出错的文本

关于归档日志目标,请参阅 LOG_ARCHIVE_DEST、LOG_ARCHIVE_DUPLEX_DEST、LOG_ARCHIVE_DEST_n、LOG_ARCHIVE_DEST_STATE_n、STANDBY_ARCHIVE_DEST 和 LOG_ARCHIVE_MIN_SUCCEED_DEST 。

6. V$ARCHIVE_LOG

此视图从包含归档日志名的控制文件中显示归档日志信息。在联机重做日志成功地归档或清除后插入的（如果日志被清除，则名称列为 NULL）后，插入一个归档日志记录。如果该日志归档两次，将有两个具有 THREAD#、SEQUENCE#、FIRST_CHANGE#但名称不同的归档日志记录。在从备份集或拷贝重新存储归档日志时，也插入一个归档日志记录。

列	数据类型	说明
RECID	NUMBER	归档日志记录 ID
STAMP	NUMBER	归档日志记录时间戳
NAME	VARCHAR2(512)	归档日志文件名。如果设置为 NULL，则日志文件在被归档前清除
THREAD#	NUMBER	重做线程号
SEQUENCE#	NUMBER	重做日志序列
RESETLOGS_CHANGE#	NUMBER	在写入此日志时重置数据库的日志更改号
RESETLOGS_TIME	DATE	在写入此日志时重置数据库的日志时间
FIRST_CHANGE#	NUMBER	归档日志中的第一个更改号
FIRST_TIME	NUMBER	第一个更改的时间戳
NEXT_CHANGE#	NUMBER	下一日志中的第一个更改
NEXT_TIME	NUMBER	下一个更改的时间戳
BLOCKS	NUMBER	以块表示的归档日志大小
BLOCK_SIZE	NUMBER	重做日志块的尺寸
ARCHIVED	VARCHAR2(3)	YES/NO
DELETED	VARCHAR2(3)	YES/NO
COMPLETION_TIME	DATE	归档完成时间

7. V$ARCHIVE_PROCESSES

此视图提供实例的各种 ARCH 进程的状态信息。

列	数据类型	说明
PROCESS	NUMBER	实例的 ARCH 进程标识符，编号从 0 至 9
STATUS	VARCHAR2(10)	ARCH 进程的状态，显示为一个关键字。可能的值为：STOPPED、SCHEDULED、STARTING、ACTIVE、STOPPING 和 TERMINATED
LOG_SEQUENCE	NUMBER	如果 STATE=“BUSY”，则这是当前归档的联机重做日志序号
STATE	VARCHAR2(4)	这是 ARCH 进程的当前状态，显示为一个关键字。可能的关键字为：IDLE 和 BUSY

8. V$BACKUP

此视图显示所有联机数据文件的备份状态。

列	数据类型	说明
FILE#	NUMBER	文件标识符
STATUS	VARCHAR2(18)	文件状态：NOT ACTIVE、ACTIVE(正在进行备份)、OFFINE NORMAL 或一个错误说明
CHANGE#	NUMBER	备份开始时的系统更改号
TIME	DATE	备份开始时间

9. V$BACKUP_ASYNC_IO

此视图显示控制文件中的备份集信息。在成功地完成备份集时，插入一个备份集记录。

列	数据类型	说明
SID	NUMBER	进行备份或恢复的会话的 Oracle SID
SERIAL	NUMBER	进行备份或恢复的 SID 的使用计数
USE_COUNT	NUMBER	可用来标识来自不同备份集的行计数器
DEVICE_TYPE	VARCHAR2(17)	放置文件的设备类型
TYPE	VARCHAR2(9)	INPUT、OUTPUT、AGGREGATE
STATUS	VARCHAR2(11)	NOT STARTED；IN PROGRESS；FINISHED
FILENAME	VARCHAR2(513)	被读取或写入的备份文件名
SET_COUNT	NUMBER	被读取或写入的备份集的集计数
SET_STAMP	NUMBER	被读取或写入的备份集的集时间戳
BUFFER_SIZE	NUMBER	用来读写这个文件的缓冲区的尺寸
BUFFER_COUNT	NUMBER	用来读写这个文件的缓冲区的数量
TOTAL_BYTES	NUMBER	如果知道，为将对这个文件进行读写的字节总数。如果不知道，此列为空
OPEN_TIME	DATE	文件打开的时间。如果 TYPE=‘AGGREGATE’，则这是聚集中第一个文件打开的时间
CLOSE_TIME	DATE	文件关闭的时间。如果 TYPE=AGGREGATE’，则这是聚集中第一个文件关闭的时间

ELAPSED_TIME	NUMBER	文件打开的时间，以百分之一秒计
MAXOPENFILES	NUMBER	同时打开的磁盘文件数。这个值仅在TYPE='AGGREGATE'的行中给出
BYTES	NUMBER	迄今为止读写的字节数
BFFECTIVE_BYTES_PER_SECOND	NUMBER	在这个备份中用这个设备归档的 I/O 率
IO_COUNT	NUMBER	对这个文件执行的 I/O 数
READY	NUMBER	缓冲区立即作好使用准备的异步请求数
SHORT_WAITS	NUMBER	缓冲区不立即可用，但缓冲区在进行 I/O 完成的非阻塞轮询后的可用次数
SHORT_WAIT_TIME_TOTAL	NUMBER	I/O 完成的非阻塞轮询所用的时间总数，以百分之一秒计
SHORT_WAIT_TIME_MAX	NUMBER	I/O 完成的非阻塞轮询所用的最大时间数，以百分之一秒计
LOG_WAITS	NUMBER	缓冲区不立即可用，缓冲区仅在进行 I/O 完成的非阻塞轮询后可用的次数
LOG_WAITS	NUMBER	缓冲区不立即可用，缓冲区仅在进行 I/O 完成的非阻塞轮询后可用的次数
LOG_WAITS_TIME_TOTAL	NUMBER	I/O 完成的阻塞等待所有的时间总数，以百分之一秒计
LOG_WAITS_TIME_MAX	NUMBER	I/O 完成的阻塞等待所有的最大时间数，以百分之一秒计

10. V$BACKUP_CORRUPTION

此视图显示来自控制文件的数据文件备份中出错的相关信息。注意，在控制文件和归档日志备份中是不容许出错的。

列	数据类型	说明
RECID	NUMBER	备份出错的记录 ID
STAMP	NUMBER	备份出错的记录时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
PIECE#	NUMBER	备份的片号
FILE#	NUMBER	数据文件号
BLOCK#	NUMBER	出错范围中的第一块
BLOCKS	NUMBER	出错范围中的邻接块数
CORRUPTION_CHANGE#	NUMBER	检查到逻辑错的更改号。设置为 0 表示介质错
MARKED_ CURRUPT	VARCHAR2(3)	YES/NO。如果设置为 YES，则在数据文件中不标记块出错，而在进行数据文件备份时检查并标记

11. V$BACKUP_DATAFILE

此视图显示来自控制文件的备份数据文件和备份控制文件。

列	数据类型	说明
RECID	NUMBER	备份数据文件记录 ID
STAMP	NUMBER	备份数据文件记录时间戳

SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
FILE#	NUMBER	数据文件号
CREATION_CHANGE#	NUMBER	数据文件的创建更改
CREATE_TIME	DATE	数据文件的创建时间戳
RESETLOGS_CHANGE#	NUMBER	数据文件备份时的重置日志更改号
RESETLOGS_TIME#	DATE	数据文件备份时的重置日志时间戳
INCREMENTAL_LEVEL	NUMBER	(0~4) 个增量备份级
INCREMENTAL_CHANGE#	NUMBER	增量更改号包含在这个备份中后更改的所有块。全备份设置为 0
CHECKPOINT_CHANGE#	NUMBER	直到检查点更改号的所有更改都包含在此备份中
CHECKPOINT_TIME	DATE	检查点时间戳
ABSOLUTE_FUZZY_CHANGE#	NUMBER	此备份中的最高更改号
MARKED_CORRUPT	NUMBER	标记出错的块数
MEDIA_CORRUPT	NUMBER	介质出错的块数
LOGICALLY_CORRUPT	NUMBER	逻辑出错的块数
DATAFILE_BLOCKS	NUMBER	备份时按块计的数据文件的尺寸。这个值也是从这个备份重新开始时数据文件占用的块数
BLOCKS	NUMBER	数据文件以块计的尺寸。未用块不拷贝到备份
BLOCKS	NUMBER	块尺寸
OLDEST_OFFLINE_RANGE	NUMBER	此备份控制文件中最旧的脱机范围记录的 RECID。对于数据文件为 0
COMPLETION_TIME	DATE	完成时间

12. V$BACKUP_DEVICE

这个视图显示支持设备的设备的有关信息。如果某种设备类型不支持指定的设备，则返回该设备的一个带设备类型和空设备的行。如果某种设备类型支持指定的设备，则为该类型的每个可用设备返回一行。这个视图不返回特殊的设备类型 DISK，因为它总是可用的。

列	数据类型	说明
DEVICE_TYPE	VARCHAR2(17)	备份设备的类型
DEVICE_NAME	VARCHAR2(512)	备份设备的名称

13. V$BACKUP_PIECE

这个视图显示来自控制文件的备份片的相关信息。每个备份集由一个或多个备份片组成。

列	数据类型	说明
RECID	NUMBER	备份片记录 ID
STAMP	NUMBER	备份片记录时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
PIECE#	NUMBER	备份片号（1~n）
COPY#	NUMBER	确定用允许双工创建的备份片的拷

		贝号。如果备份片不是双工的，为 1
DEVICE_TYPE	VARCHAR2(17)	备份片驻留的设备类型。备份组在磁盘上设置为 DISK。请参 V$BACKUP_DEVICE
HANDLE	VARCHAR2(513)	备份片句柄确定正在恢复的备份片
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注释。对于磁盘上的备份片设置为 NULL。这个信息只是提示性的；恢复时不需要
MEDIA	VARCHAR2(65)	备份片驻留其上的介质数。这个值是提示性的；恢复时不需要
MEDIA_POOL	NUMBER	备份片驻留在其中的介质池。这是一个与 Recovery Manager 的 BACKUP 命令的 POOL 操作数中输入的值相同的值
CONCUR	VARCHAR2(3)	YES/NO，确定介质上的片是否可并发地访问
TAG	VARCHAR2(32)	备份片标记。在备份集级指定这个标记，但在片级存储
DELETED	VARCHAR2(3)	如果设置为 YES，表示片被删除，否则设置为 NO
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	完成时间
ELAPSED_SECONDS	NUMBER	占用的秒数

14. V$BACKUP_REDOLOG

此视图显示来自控制文件的备份集中归档日志的信息。注意，联机重做日志不能直接备份；它们必须首先归档到磁盘，然后再备份。一个归档日志备份集可包含一个或多个归档日志。

列	数据类型	说明
RECID	NUMBER	此行的记录 ID。它是一个标识此行的整数
STAMP	NUMBER	RECID 唯一地标识此行所用的时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	标识这个备份集的 V$BACKUP_SET 表的行外部键之一
THREAD#	NUMBER	日志的线程号
SEQUENCE#	NUMBER	日志序列号
RESETLOGS_CHANGE#	NUMBER	在写入前的最后重置日志的更改号
RESETLOGS_TIME	DATE	在日志写入前的最后重置日志的更改时间。同一备份集中所有日志的这个值都是相同的
FIRST_CHANGE#	NUMBER	在将日志切换入时的 SCN。日志中的重做是在此 SCN 或更大进行的
FIRST_TIME	DATE	切换入日志时所分配的时间
NEXT_CHANGE#	NUMBER	切换出日志时的 SCN。日志中的重做低于此 SCN

NEXT_TIME	DATE	切换出日志时分配的时间
BLOCKS	NUMBER	逻辑块中的日志尺寸，包括标题块
BLOCK_SIZE	NUMBER	以字节表示的日志块尺寸

15. V$BACKUP_SET

此视图显示来自控制文件的备份集信息。在成功完成备份集后，插入一个备份集记录。

列	数据类型	说明
RECID	NUMBER	备份集记录 ID。
STAMP	NUMBER	备份集记录的时间戳
ET_STAMP	NUMBER	备份集时间戳。备份集时间戳和计数唯一标识备份集 V$BACKUP_PIECE； V$BACKUP_DATAFILE； V$BACKUP_REDOLOG； V$BACKUP_CORRUPTION；
SET_COUNT	NUMBER	备份集计数器。备份集计数每当开始一个新备份集时加 1（如果备份集永不结束则则此数会丢失）。如果重新创建控制文件，则此计数重置为 1。因此此计数必须与唯一标识一个备份集的时间戳一起使用 V$BACKUP_PIECE 表的主键，以及下列表的外部键： V$BACKUP_PIECE； V$BACKUP_DATAFILE； V$BACKUP_REDOLOG； V$BACKUP_CORRUPTION
BACKUP_TYPE	VARCHAR2(1)	此备份中的文件类型。如果此备份含有归档重做日志，则为‘L’。如果这是一个数据文件完全备份，则值为‘D’。如果这是一个增量备份，则值为‘I’。
CONTROLFILE_INCLUDED	VARCHAR2(3)	如果此备份集中含有一个控制文件，则设置为 YES，否则设置为 NO
INCREMENTAL_LEVEL	NUMBER	此备份集适合于数据库备份策略的位置。对完全的数据文件备份设置为零，对增量数据文件备份设置为非零，而对归档日志备份设置为 NULL
PIECES	NUMBER	备份集中不同备份片的数目
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	在成功完成备份时，这是备份集的完成时间。这也是由 backupEnd 返回的相同时间。如果备份正在进行中或已经失败，则设置为 NULL
ELAPSED_SECONDS	NUMBER	所占用的秒数
BLOCK_SIZE	NUMBER	备份集的块尺寸

16. V$BACKUP_SYNC_IO

此视图显示来自控制文件的备份集的信息。在备份集成功完成之后，插入一个备份集记

录。

列	数据类型	说明
SID	NUMBER	进行备份或恢复的会话的Oracle SID
SERIAL	NUMBER	进行备份或恢复的 SID 的使用计数
USE_COUNT	NUMBER	可用来标识来自不同备份集的行计数器
DEVICE_TYPE	VARCHAR2(17)	放置文件的设备类型
TYPE	VARCHAR2(9)	INPUT、OUTPUT、AGGREGATE
STATUS	VARCHAR2(11)	NOT STARTED、IN PROGRESS ;FINISHED
FILENAME	VARCHAR2(512)	被读取或写入的备份文件名
SET_COUNT	NUMBER	被读取或写入的备份集的集计数
SET_STAMP	NUMBER	被读取或写入的备份集的集时间戳
BUFFER_SIZE	NUMBER	用来读写这个文件缓冲区的尺寸
BUFFEER_COUNT	NUMBER	用来读写这个文件缓冲区的数量
TOTAL_BYTES	NUMBER	如果知道，为将对这个文件进行读写的字节总数。如果不知道，此列为空
OPEN_TIME	DATE	文件打开的时间。如果TYPE=’AGGREGATE’，则这是聚集中第一个文件打开的时间
CLOSE_TIME	DATE	文件关闭的时间。如果TYPE=’AGGREGATE’，则这是聚集中第一个文件打开的时间
ELAPSED_TIME	DATE	文件打开时间，以百分之一秒计
MAXOPENFILES	NUMVER	同时打开的磁盘文件数。这个值仅在 TYPE=’AGGREGATE’的行中同时给出
BYTES	NUMBER	迄今为止读写的字节数
BFFECTIVE_BYTES_PER_SECOND	NUMBER	在这个备份中用这个设备归档的 I/O 率
IO_COUNT	NUMBER	对这个文件执行的 I/O 数
I/O_TIME_TOTAL	NUMBER	进行此文件的 I/O 所占的时间总数，以百分之一秒计
I/O_TIME_MAX	NUMBER	单个 I/O 请求所用的最大时间
DISCRETE_BYTES PER_SECOND	NUMBER	这个文件的平均传输率

17． V$BGPROCESS

此视图描述后台进程。

列	数据类型	说明
PADDR	RAW(4)	进程状态对象的地址
NAME	VARCHAR2	后台进程的名称
DESCRIPTION	VARCHAR2	后台进程的说明
ERROR	NUMBER	所遇到的错误

18． V$BH

这是一个并行服务器视图。这个视图给出 SGA 中每个缓冲区的 ping 状态和数目。

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	FREE=当前不用 XCUR=互斥的 SCUR=当前共享 CR=一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCMx 数。此列已作废，但为了历史兼容性仍然保留
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_CLASS	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
FORCED_READS	NUMBER	锁必须从磁盘上读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
DIRTY	VARCHAR(1)	Y=修改过的块
DIRTY	VARCHAR(1)	Y=临时块
DIRTY	VARCHAR(1)	Y=ping 过的块
DIRTY	VARCHAR(1)	Y=块是陈旧的
DIRTY	VARCHAR(1)	Y=直接块
DIRTY	VARCHAR(1)	总是设置为 N。此列已废弃，但为了历史兼容而保留
OBJD	NUMBER	缓冲区代表的块的数据库对象数
TS#	NUMBER	块的表空间数

19. V$BUFFER_POOL

此视图显示实例可用的所有缓冲池的相关信息。这个“ 设施 ”适合于 LRU 栓锁组的数目。更多的信息，请参阅“ DB_BLOCK_LRU_LATCHES ”。

列	数据类型	说明
ID	NUMBER	缓冲池 ID 号
NAME	VARCHAR2(20)	缓冲池名称
LO_SETID	NUMBER	低设置 ID 号
HI_SETID	NUMBER	高设置 ID 号
SET_COUNT	NUMBER	这个缓冲池中的设置数，为 HI_SETID_LO_SETID+1
BUFFERS	NUMBER	分配给缓冲池的缓冲区数
LO_BNUM	NUMBER	本缓冲池的低缓冲区号
HI_BNUM	NUMBER	本缓冲池的高缓冲区号

20. V$BUFFER_POOL_STATISTICS

此视图显示事例可用的所有缓冲池的相关信息。这个“ 设施 ”适合于 LRU 栓锁组的数目。更多的信息，请参阅 “ DB_BLOCK_LRU_LATCHES ”。

列	数据类型	说明
ID	NUMBER	缓冲池 ID 号
NAME	VARCHAR2(20)	缓冲池名称
SET_MSIZE	NUMBER	缓冲池最大设置尺寸
CNUM_REPL	NUMBER	替换列表中的缓冲区数
CNUM_WRITE	NUMBER	写入列表中的缓冲区数
CNUM_SET	NUMBER	设置中的缓冲区数
BUF_GOT	NUMBER	设置获得的缓冲区数
SUM_WRITE	NUMBER	设置写入的缓冲区数
SUM_SCAN	NUMBER	设置扫描的缓冲区数
FREE_BUFFER_WAIT	NUMBER	可用缓冲区等待统计数据
WRITE_COMPLETE_WAIT	NUMBER	写完成等待统计数据
BUFFER_BUSY_WAIT	NUMBER	缓冲区忙等待统计数据
RFEE_BUFFER_INSPECTED	NUMBER	可用缓冲区检查统计数据
DIRTY_ BUFFER_INSPECTED	NUMBER	灰缓冲区检查统计数据
DB_BLOCK_CHANGE	NUMBER	数据块更改统计数据
DB_BLOCK_GETS	NUMBER	取得数据库块统计数据
CONSISENT_GETS	NUMBER	一致取统计数据
PHYSICAL_READS	NUMBER	物理读统计数据
PHYSICAL_WRITES	NUMBER	物理写统计数据

21. V$CACHE

这是一个并行服务器视图。此视图包含来自当前实例的 SGA 中每个块的块标题的信息，这些信息涉及特定的数据库对象。

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	FREE=当前不用 XCUR=互斥的

		SCUR=当前共享 CR=一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCMx 数。此列已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	强制读取
FORCED_WRITES	NUMBER	强制写入
NANE	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2(30)	分区名；非分区对象为 NULL
KIND	VARCHAR2930)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者
LOCK_ELEMENT_ ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区 @@@@@

表 B-1　KIND 列的值

类型号	KIND 值	类型号	KIND 值
1	index	11	PACKAGE　BODY
2	TABLE	12	TRIGGER
3	CLUSTER	13	TYPE
4	VIEW	14	TYPE　BODY
5	SYNONYM	19	TABLE　PARTITION
6	SEQUENCE	20	INDEX　PARTITION
7	PROCEDURE	21	LOB
8	FUNCTION	22	LIBRARY
9	PACKAGE	NULL	UNKNOWN
10	NON_EXISTENT	……	……

22.　V$CACHE_LOCK

这是一个并行服务器视图

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号
STATUS	VARCHAR2(1)	FREE=当前不用

		XCUR=互斥的 SCUR=当前共享 CR= 一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的并行高速缓存管理锁变换的数目
NANE	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2(30)	分区名；非分区对象为 NULL
KIND	VARCHAR2(30)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT _ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT _NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
FORCED_READS	NUMBER	锁必须从磁盘读取次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
INDX	NUMBER	平台专用的锁管理程序标识符
CLASS	NUMBER	平台专用的锁管理程序标识符

除了平台专用的锁管理标识符外，V$CASHE_LOCK 与 V$CACHE 类似。如果平台专用的锁管理程序提供了监控正在进行的 PCM 锁操作的工具，则这个信息是很有用的。例如，前一个查询利用 INDX 和 CLASS 找到了锁元素的地址，然后查询 V$BH 被锁覆盖的缓冲区。请参阅“ V$CACHE ”。

23. V$CIRCUIT

该视图包含虚电路的有关信息，虚电路是通过调度程序和服务器对数据库的用户连接。

列	数据类型	说明
CIRCUIT	RAW(4)	虚电路地址
DISPATCHER	RAW(4)	虚电路调度程序进程地址
SERVER	RAW(4)	虚电路服务器进程地址
WAITER	RAW(4)	等待（当前忙）虚电路可用的服务器进程地址
SADDR	RAW(4)	绑定到虚电路的会话地址
STATUS	RAW(4)	虚电路的状态：BREAK(当前中断)，EOF(将要被删除)，OUTBOUND(向外连接到远程数据

		库)，NORMAL(正常进入本地数据库的虚电路)
QUEUE	VARCHAR2(16)	虚电路当前所在的队列：COMMON(在公共队列上，等待被某个服务器进程选取)、DISPATCHER(等待调度程序)、SERVER(当前接受服务)、NONE(空闲虚电路)
MESSAGE0	NUMBER	以字节表示的第一个消息缓冲区中消息的大小
MESSAGE1	NUMBER	以字节表示的第二个消息缓冲区中消息的大小
MESSAGE2	NUMBER	以字节表示的第三个消息缓冲区中消息的大小
MESSAGE3	NUMBER	以字节表示的第四个消息缓冲区中消息的大小
MESSAGES	NUMBER	已通过此虚电路的消息总数
BYTES	NUMBER	已通过此虚电路的字节总数
PRESENTATION	NUMBER	此虚电路的断开（中断）数

24. V$CLASS_PING

V$CLASS_PING 显示每个块类 ping 的块数。可使用此视图在不同类中比较块的争用情况。

列	数据类型	说明
CLASS	NUMBER	表示块类别的编号
X_2_NULL	NUMBER	对指定 CLASS 的所有块，Exclusive_to_NULL 的块转换数
X_2_NULL_ FORCED_WRITE	NUMBER	对指定 CLASS 的所有块，由于 Exclusive_to_NULL 转换进行强制写的数目
X_2_NULL_ FORCED_STATE	NUMBER	CLASS 中的块，由于 Exclusive_to_NULL 转换而变陈旧的次数
X_2_S	NUMBER	指定 CLASS 的所有块的 Exclusive_to_Shared 锁转换数目
X_2_S_ FORCED_WRITE	NUMBER	对指定 CLASS 的块，由于 Exclusive_to_Shared 转换而出现的强制写的数目
X_2_SSX	NUMBER	指定 CLASS 的所有块，Exclusive_to_Sub Shared Exclusive 锁转换的数目
X_2_SSX_ FORCED _WRITE	NUMBER	对指定 CLASS 的块，由于 Exclusive_to_Sub Shared Exclusive 转换而出现的强制写的数目
S_2_NULL	NUMBER	对指定 CLASS 的所有块，Share_to_NULL 锁转换的数目
S_2_NULL_ FORCED_STATE	NUMBER	CLASS 中的块，由于 Share_to_NULL 转换而陈旧的数

		目
SS_2_NULL	NUMBER	对于指定 CLASS 的所有块 Sub Shared_to_NULL 锁转换的数目
S_2_X	NUMBER	对于指定 CLASS 的所有块，Shared_to_NULL 锁转换的数目
SSX_2_X	NUMBER	对于指定 CLASS 的所有块，Sub SharedExclusive_to_Exclusive 锁转换的数目
NULL_2_S	NUMBER	对于指定 CLASS 的所有块，NULL_to_Shared 锁转换的数目
NULL_2_SS	NUMBER	对于指定 CLASS 的所有块，NULL_to_Sub Shared 锁转换的数目

25. V$COMPATIBILITY

此视图显示数据库实例正在使用的特性，以防止性能降为以前的版本。这是此信息的动态（SGA）板，并不反映其他实例已经使用的特性，并有可能包含临时的不兼容（如 UNDO 段），这种不兼容在数据库完全关闭后就不存在了。

列	数据类型	说明
TYPE_ID	VARCHAR2(8)	内部特性标识符
RELEASE	VARCHAR2(60)	发布该特性的版本
DESCRIPTION	VARCHAR2(64)	特性描述

26. V$COMPATSEG

此视图列出数据库实例正在使用的永久特性，以防止返回到早期版本。

列	数据类型	说明
TYPE_ID	VARCHAR2(8)	内部特性标识符
RELEASE	VARCHAR2(60)	发布该特性的版本。
UPDAED	VARCHAR2(60)	首先使用此特性的版本

27. V$CONTEXT

此视图列出当前会话中设置的属性

列	数据类型	说明
NAMESPACE	VARCHAR2(30)	名称空间名
ATTRIBUTE	VARCHAR2(30)	属性名
VALUE	VARCHAR2(64)	属性值

28. V$CONTROLFILE

这个视图列出控制文件名

列	数据类型	说明
STATUS	VARCHAR2(7)	如果不能确定名称（这是不应该发生的），则为

		INVALID。如果可以确定名称，为 NULL
NAME	VARCHAR2(257)	控制文件名

29. V$CONTROLFILE_RECORD_SECTION

这个视图显示控制文件记录部分的相关信息

列	数据类型	说明
TYPE	VARCHAR2(7)	DATABASE/CKPT_PROGRESS/REDO THREAD/REDO LOG/DATAFILE/FILENAME/TABLESPACE/ LOG HISTORY/OFFLINE RANGE/ARCHIVED LOG/BACKUP SET/BACKUP PIECE/BACKUP DATAFILE/BACKUP REDOLOG/DATAFILE COPY/BACKUP CORRUPTION/COPY CORRUPTION/DELETED OBJECT
RECORD_SIZ	NUMBER	以字节表示的记录尺寸
RECORD_TOTAL	NUMBER	为该部分分配的记录数
RECORD_USED	NUMBER	该部分中使用的记录数
FIRST_INDEX	NUMBER	第一个记录的索引（位置）
LAST_INDEX	NUMBER	最后一个记录的索引
LAST_RECID	NUMBER	最后一个记录的记录 ID

30. V$COPY_CORRUPTION

这个视图显示来自控制文件的数据文件拷贝出错的相关信息。

列	数据类型	说明
RECID	NUMBER	拷贝出错的记录 ID
STAMP	NUMBER	拷贝出错的记录时间戳
COPY_ RECID	NUMBER	数据文件拷贝记录 ID
COPY_STAMP	NUMBER	数据文件拷贝记录时间戳
FILE#	NUMBER	数据文件号
BLOCK#	NUMBER	出错范围的每一个块
BLOCKS	NUMBER	出错范围中的邻接块数
CORRUPTION_CHANGE#	NUMBER	检测到逻辑错的更改号。设置为 0 表示介质出错
MARKED_CORRUPT#	VARCHAR2(3)	YES/NO。如果设置为 YES，则数据文件中不标记出错块，但在进行数据文件拷贝时进行检测和标记

31. V$DATABASE

这个视图包含来自控制文件的数据库信息

列	数据类型	说明
DBID	NUMBER	数据库 ID
NAME	VARCHAR2	数据库名
CREATED	DATE	创建日期
LOG_MODE	VARCHAR2	归档日志模式：NOARCHIVELOG 或 ARCHIVELOG
CHECKPOINT_CHANGE#	NUMBER	最后一个 SCN 检查点
ARCHIVE_CHANGE#	NUMBER	归档的最后一个 SCN
DBID	NUMBER	在所有文件标题中创建和存取数据库时，计算的数据库 ID
RESETLOGS_CHANGE#	NUMBER	打开重置日志时的更改号
RESETLOGS_TIME	DATE	打开重置日志的时间戳
PRIOR_RESETLOGS_CHANGE#	NUMBER	以前重置日志时的更改号
PRIOR_RESETLOGS_TIME	DATE	以前重置日志的时间戳
CONTROLFILE_TYPE	VARCHAR2(9)	CURRENT/STANDBY/CLONE/BACKUP/CREATED。STANDBY 表示数据库处于备用状态。CLONE 表示一个克隆数据库 BUCKUP/CREATED 表示正利用备份或创建的控制文件恢复数据库。在恢复更改类型为 CURRENT 后备用数据库启动或数据库打开
CONTROLFILE_CREATED	DATE	控制文件创建时间戳
CONTROLFILE_SEQUENCE#	NUMBER	由控制文件事务处理增加的控制文件序列号
CONTROLFILE_CHANGE#	NUMBER	备份控制文件事务中的最后更改号。如果控制文件未备份，则设置为 NULL
CONTROLFILE_TIME	DATE	备份控制文件事务中的最后时间戳。如果控制文件未备份，则设置为 NULL
OPEN_RESETLOGS	VARCHAR2(11)	NOT ALLOWED/ALLOWED/REQUIRED 指出下一次数据库打开是否允许或需要重置日志选项
VERSION_TIME	DATE	版本时间
OPEN_MODE	VARCHAR2(10)	打开模式的信息

32. V$DATAFILE

此视图含有来自控制文件的数据文件信息。还可以参阅“V$DATAFILE_HEADER”视图，该视图显示来自数据文件标题的信息。

列	数据类型	说明
FILE#	NUMBER	文件标识号

STATUS	VARCHAR2	文件（系统或用户）类型极其状态。值：OFFLINE、ONLINE、SYSTEM、RECOVER。SYSOFF（来自系统表空间的脱机文件）
ENABLED	VARCHAR2(10)	描述怎样从 SQL 访问文件。取值为表 B-1 中所示的任一值
CHECKPOINT_CHANGE#	NUMBER	最后一个检查点的 SCN
CHECKPOINT_TIME	DATE	检查点时间戳
UNRECOVERABLE_CHANGE#	NUMBER	对这个文件所做的最后一个不可恢复的更改号。此列总是在一个不可恢复的操作完成后更新
UNRECOVERABLE_TIME	DATE	最后一个不可恢复更改的时间戳
BYTES	NUMBER	以字节计的当前尺寸；如果不可访，则为 0
CREATE_BYTES	NUMBER	创建的尺寸，以字节计
NAME	VARCHAR2	文件名
CREATION_CHANGE#	NUMBER	创建数据文件的更改号
CREATION_TIME	DATE	创建数据文件的时间戳
TS#	NUMBER	表空间号
RFLE#	NUMBER	表空间的相关数据文件号
LAST_CHANGE#	NUMBER	对此数据文件所做的最后一次更改号。如果数据文件正在更改，设置为 NULL
LAST_TIME	DATE	最后一次更改的时间戳
OFFLINE_CHANGE#	NUMBER	最后脱机文件的更改号。此列仅在数据文件进入联机时更新
ONLINE_CHANGE#	NUMBER	最后脱机范围的联机更改号
ONLINE_TIME	DATE	最后脱机范围的联机时间戳
BLOCKS	NUMBER	以块计的当前数据文件尺寸；如果不可访问，为 0
BLOCK_SIZE	NUMBER	数据文件的块尺寸
NAME	VARCHAR2(512)	数据文件名
PLUGGED_IN	NUMBER	描述是否插入表空间。如果插入表空间且未进行读写，设置为 1，否则设置为 0

表 B-2ENABLED 列的值

ENABLED 列值	说明
DISABLED	不允许 SQL 访问
READ ONLY	不允许 SQL 更新
READ WRITE	允许完全访问
UNKNOWN	除非控制文件出错，否则不应该出现

33. V$DATAFILE_COPY

这个视图显示来自控制文件的数据文件拷贝信息。

列	数据类型	说明

RECID	NUMBER	数据文件拷贝记录 ID
STAMP	NUMBER	数据文件拷贝记录时间戳
NAME	VARCHAR2(512)	数据文件拷贝文件名。此名字的最大长度与 OS 有关
TAG	VARCHAR2(32)	数据文件拷贝标记
FILE#	NUMBER	绝对数据文件号
RFILE#	NUMBER	表空间的相关数据文件号
CREATION_CHANGE#	NUMBER	数据文件的创建更改号
CREATE_TIME	DATE	数据文件的创建时间戳
RESETLOGS_CHANGE#	NUMBER	数据文件拷贝时的重置日志更改号
RESETLOGS_TIME	DATE	数据文件拷贝时的重置日志时间戳
CHANGE#		
INCREMENTAL_LEVEL	NUMBER	增量级
CHECKPOINT_ CHANGE#	NUMBER	进行数据文件拷贝时的检查点更改号
CHECKPOINT_TIME	DATE	进行数据文件拷贝时的检查点时间戳
ABSOLUTE_FUZZY_	NUMBER	数据文件拷贝时看到的最高更改
CHANGE#		
RECOVERY_FUZZY_	NUMBER	介质恢复写到此文件的最高更改
CHANGE#		
RECOVERY_FUZZY_	DATE	介质恢复写到此文件的最高更改时
TIME		间戳
ONLINE_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，这是一个在崩溃或立即脱机后利用一个操作系统实用程序进行的拷贝。（或者是一个在数据库联机或打开时的无效拷贝）。恢复将需要应用所有重做直到下一个崩溃恢复标记使该文件一致
BACKUP_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，这是一个利用 BEGIN BACKUP/END BACKUP 技术进行的拷贝。恢复将需要应用所有重做直到下一个结束标记使这个拷贝一致
MARKED_CORRUPT	NUMBER	这个拷贝操作标记出错的块数。即在原数据文件中未标记出错，但在拷贝操作中检测到并标记为出错的块数
MEDIA_CORRUPT	NUMBER	介质出错块的总数。例如，校验和错的块被标记为介质错
LOGICALLY_CORRUPT	NUMBER	逻辑出错块的总数。例如，对不可恢复的操作应用重做将标记涉及的块为逻辑错
BLOCKS	NUMBER	以块计的数据文件拷贝的尺寸（也是进行拷贝时的数据文件尺寸）
BLOCK_SIZE	NUMBER	
OLDEST_OFFLINE_	NUMBER	在这个控制文件拷贝中最旧的脱机

RANGE		范围记录的 RECID。数据文件拷贝为 0
COMPLETION_TIME	DATE	拷贝完成时间
DALETED	VARCHA2(3)	YES/NO。如果设置为 YES，则该数据文件拷贝已经删除或重写

34. V$DATAFILE_HEADER

这个视图显示来自数据文件标题的数据文件信息

列	数据类型	说明
FILE#	NUMBER	数据文件号（来自控制文件）
STATUS	VARCHAR2(7)	ONLINE/OFFLINE（来自控制文件）
ERROR	VARCHAR2(18)	如果数据文件标题读取和验证成功，为 NULL 如果读失败，则其余的列为 NULL。如果验证失败，则其余的列可能显示无效数据。如果存在错误，则在数据文件可以恢复或使用前，必须利用备份进行修复
FORMAT	NUMBER	指出标题块的格式。可能的值为 6、7、8、0 6 表示 Oracle 版本 6 7 表示 Oracle 版本 7 8 表示 Oracle 版本 8 0 表示此格式不能确定（例如，标题不能读出）
RECOVER	VARCHAR2(3)	文件需要介质恢复
FUZZY	VARCHAR2(3)	文件是模糊的 YES/NO
CREATION_CHANGE#	NUMBER	数据文件创建的更改号
CREATION_TIME	DATE	数据文件创建的时间戳
TABLESPACE_NAME	VARCHAR2(30)	表空间名
TS#	NUMBER	表空间号
RFILE#	NUMBER	表空间的相关数据文件号
RESETLOGS_ CHANGE#	NUMBER	重置日志更改号
CHECKPOINT_ CHANGE#	NUMBER	数据文件检查点更改号
CHECKPOINT_ TIME	DATE	数据文件检查点时间戳
CHECKPOINT_COUNT	NUMBER	数据文件检查点计数
BYTES	NUMBER	以字节计的当前数据文件尺寸
BLOCKS	NUMBER	以块计的当前数据文件的文件尺寸
NAME	VARCHAR2(512)	数据文件名

35. V$DBFILE

此视图列出构成数据库的所有数据文件。这个视图是为了历史兼容而保存的。建议使用V$DATAFILE。更多的信息，请参阅V$DATAFILE。

列	数据类型	说明
FILE#	NUMBER	文件标识符
NAME	VARCHAR2	文件名

36. V$DBLINK

这个视图描述在V$DBLINK上发布查询的会话打开的所有数据库连接（带IN_TRANSACTION=YES的连接)。这些数据库连接在关闭前必须提交或退回。

列	数据类型	说明
DBLINK	VARCHAR2(128)	数据连接名
OWNER_ID	NUMBER	数据库连接UID的拥有者
LOGED_ON	VARCHAR2(3)	当前数据库连接是否登录
HETEROGENEOUS	VARCHAR2(3)	数据库连接是否异构
PROTOCOL	VARCHAR2(6)	数据库连接的通讯协议
OPEN_CURSORS	NUMBER	此数据库连接是否存在打开的游标
IN_TRANSACTION	NUMBER	此数据库连接当前是否处于事务处理中
UPDATE_SENT	VARCHAR2(3)	数据库连接上是否曾经有过更新
COMMIT_STRENGTH	NUMBER	数据库连接上的事务处理的提交点强度

37. V$DB_OBJECT_CACHE

这个视图显示高速缓存在数据库高速缓存中的数据库对象。这些对象包括表、索引、簇、同义词定义、PL/SQL过程和程序包、触发器。

列	数据类型	说明
OWNER	VARCHAR2	对象的拥有者
NAME	VARCHAR2	对象名
DBLINK	VARCHAR2	数据库连接名
NAMESPACE	VARCHAR2	对象的库高速缓存名称空间：TABLE/PROCEDURE、BODY、TIGGER、INDEX、CLUSTER、OBJECT
TYPE	VARCHAR2	对象类型：INDEX、TABLE、CLUSTER、VIEW、SET、SYNONYM、SEQUENCE、PROCEDURE、FUNCTION、PACKAGE、PACKAGE BODY、TIGGER、CLASS、OBJECT、USER、DBLINK
HARARLE_MEM	NUMBER	对象消耗的共享缓冲池中共享内存量

LOADS	NUMBER	对象被装载的次数。在使某个对象无效时也增加这个计数
EXECUTIONS	NUMBER	不使用。为了看到实际的执行计数，请参阅 V$SQLAREA
LOCKS	NUMBER	当前锁住这个对象的用户数
PINS	NUMBER	当前固定这个对象的用户数
KEPT	VARCHAR2	YES 或者 NO，有赖于这个对象是否已经利用 PL/SQL 过程 DBMS_SHARED_POOL.KEEP“保持”（永久固定在内存中）

38. V$DB_PIPES

这个视图显示当前位于数据库中的管道

列	数据类型	说明
OWNER_ID	NUMBER	如果这是一个私有管理，则为拥有者 ID；否则为 NULL
NAME	VARCHAR2(1000)	管道名；例如，scott.pipe
TYPE	VARCHAR2(2)	PUBLIC 或 PRIVATE
PIPE_SIZE	NUMBER	管道占用的内存量

39. V$DELETED_OBJECT

这个视图显示来自控制文件的删除归档日志、数据文件拷贝和备份片的相关信息。这个视图的唯一用途是优化恢复目录的重新同步操作。在删除归档日志、数据文件拷贝或备份片时，相应的记录标记为删除。

列	数据类型	说明
RECID	NUMBER	删除对象记录的 ID
STAMP	NUMBER	删除对象的记录时间戳
TYPE	VARCHAR2(13)	ARCHIVED LOG/DATAFILE COPY/BACKUPPIECE。删除对象的类型
OBJECT_RECID	NUMBER	删除对象的记录 ID
OBJECT_STAMP	NUMBER	删除对象的记录时间戳

40. V$DISPATCHER

这个视图提供有关调度程序进程的信息

列	数据类型	说明
NAME	VARCHAR2	调度进程名
NETWORK	VARCHAR2	调度程序的网络地址
PADDR	RAW(4)	进程地址
STATUS	VARCHAR2	调度程序状态：WAIT(空闲)、SEND(发送一个连接消息)、RECEIVE(接收一个消息)、CONNECT(建立一个连接)、DISCONNECT(处理断开请求)、BREAK(处理断开)、OUTBOUND(建立出站连接)

ACCEPT	VARCHAR2	这个调度程序是否接受新的连接：YES、NO
MESSAGES	NUMBER	此调度程序处理的消息数目
BYTES	NUMBER	此调度程序处理的消息大小、以字节计
BREAKS	NUMBER	这个连接中发生的断开数目
OWNED	NUMBER	这个调度程序拥有的虚电路数目
CREATED	NUMBER	这个调度程序创建的虚电路数目
IDLE	NUMBER	这个调度程序的总空闲时间，以百分之一秒计
BUSY	NUMBER	这个调度程序的总繁忙时间，以百分之一秒计
LISTENER	NUMBER	此调度程序从监听程序收到的最近Oracle 错误数目
CONF_INDX	NUMBER	这个调度程序使用的MTS_DISPATCHERS 配置的基于零的索引

41. V$DISPATCHER_RATE

这个视图提供调度程序进程的速率统计数据

列	数据类型	说明
NAME	VARCHAR2	调度进程名
PADDR	RAW(4)	进程地址
CUR_LOOP_RATE	NUMBER	循环事件的当前速率
CUR_EVENT_RATE	NUMBER	事件的当前速率
CUR_EVENTS._PER_LOOP	NUMBER	每个循环的当前事件
CUR_MSG_RATE	NUMBER	消息的当前速率
CUR_SVR_BYTE_RATE	NUMBER	服务器缓冲区当前速率
CUR_SVR_BYTE_PER_BUF	NUMBER	服务器当前字节速率
CUR_CLT_BUF_RATE	NUMBER	客户机缓冲区当前速率
CUR_CLT_BYTE_RATE	NUMBER	客户机缓冲区当前字节速率
CUR_CLT_BYTE_PER_BUF	NUMBER	客户机每个缓冲区当前字节
CUR_BUF_RATE	NUMBER	缓冲区当前速率
CUR_BYTE_RATE	NUMBER	当前字节速率
CUR_BYTE_PER_BUF	NUMBER	每个缓冲区当前字节
CUR_IN_CONNECT_RATE	NUMBER	当前入站连接
CUR_OUT_CONNECT_RATE	NUMBER	当前出站连接
CUR_RECONNECT_RATE	NUMBER	连接迟和复用的当前连接
MAX_LOOP_RATE	NUMBER	循环事件的最大速率
MAX_EVENT_RATE	NUMBER	事件的最大速率
MAX_EVENTS_PER_LOOP	NUMBER	每个循环的最大事件数
MAX_MSG_RATE	NUMBER	消息的最大速率
MAX_SRV_BUF_RATE	NUMBER	服务器缓冲区最大速率
MAX_SRV_BYTE_RATE	NUMBER	服务器最大字节速率
MAX_SRV_BYTE_PER_BUF	NUMBER	服务器的每个缓冲区最大字节数
MAX_CLT_BUF_RATE	NUMBER	客户机缓冲区最大速率
MAX_CLT_BYTE_RATE	NUMBER	客户机最大字节速率
MAX_CLT_BYTE_PER_BUF	NUMBER	客户机的每个缓冲区最大字节数
MAX_BUF_RATE	NUMBER	缓冲区最大速率

MAX_BYTE_RATE	NUMBER	最大字节速率
MAX_BYTE_PER_BUF	NUMBER	每个缓冲区最大字节数
MAX_IN_CONNECT_RATE	NUMBER	入站连接的最大数
MAX_OUT_CONNECT_RATE	NUMBER	出站连接的最大数
MAX_RECONNECT_RATE	NUMBER	连接池和复用的最大连接数
AVG_LOOP_RATE	NUMBER	循环事件的平均速率
AVG_EVENT_RATE	NUMBER	循环事件的平均速率
AVG_EVENTS_PER_LOOP	NUMBER	每个循环的平均事件数
AVG_MSG_RATE	NUMBER	消息的平均速率
AVG_SVR_BUF_RATE	NUMBER	服务器缓冲区平均速率
AVG_SVR_BYTE_RATE	NUMBER	服务器的平均字节速率
AVG_SVR_BYTE_PER_BUF	NUMBER	服务器每个缓冲区的平均字节数
AVG_CLT_BUF_RATE	NUMBER	客户机缓冲区平均速率
AVG_CLT_BYTE_RATE	NUMBER	客户机平均字节速率
AVG_CLT_BYTE_PER_BUF	NUMBER	客户机每个缓冲区平均字节数
AVG_BUF_RATE	NUMBER	缓冲区平均速率
AVG_BYTE_RATE	NUMBER	平均速率字节速率
AVG_BYTE_PER_BUF	NUMBER	每个缓冲区平均字节数
AVG_IN_CONNECT_RATE	NUMBER	平均入站连接数
AVG_OUT_CONNECT_RATE	NUMBER	平均出站连接数
AVG_RECONNECT_RATE	NUMBER	连接池和复用的平均重连接数
NUM_LOOPS_TRACKED	NUMBER	跟踪循环的数目
NUM_MSG_TRACKED	NUMBER	跟踪消息的数目
NUM_SRV_BUF_TRACKED	NUMBER	跟踪服务器的缓冲区数目
NUM_CLT_BUF_TRACKED	NUMBER	跟踪客户机的缓冲区数目
NUM_BUF_TRACKED	NUMBER	跟踪缓冲区的数目
NUM_IN_CONNECT_RATE	NUMBER	跟踪入站连接的数目
NUM_OUT_CONNECT_RATE	NUMBER	跟踪出站连接的数目
NUM_RECONNECT_RATE	NUMBER	跟踪重连接的数目
SCALE_LOOPS	NUMBER	循环的规模
SCALE_MSG	NUMBER	消息的规模
SCALE_SRV_BUF	NUMBER	服务器缓冲区的规模
SCALE_CLT_BUF	NUMBER	客户机缓冲区的规模
SCALE_BUF	NUMBER	缓冲区的规模
SCALE_IN_CONNECT	NUMBER	入站连接的规模
SCALE_OUT_CONNECT	NUMBER	出站连接的规模
SCALE_RECONNECT	NUMBER	重连接的规模

42. V$DLM_ALL_LOCKS

这是一个并行服务器视图。此视图列出锁管理程序当前已知的被其他锁阻塞或阻塞其他锁的所有锁的信息。

列	数据类型	说明
LOCKP	RAW(4)	锁指针
GRANT_LEVEL	VARCHAR2(9)	锁的授权级别
REQUEST_LEVEL	VARCHAR2(9)	锁的请求级别
RESOURCE_NAME1	VARCHAR2(30)	锁的资源名
RESOURCE_NAME2	VARCHAR2(30)	锁的资源名
PID	NUMBER	拥有锁的进程标识
TRANSACTION_ID0	NUMBER	锁所属的事务处理标识符的

		低 4 个字节
TRANSACTION_ID1	NUMBER	锁所属的事务处理标识符的高 4 个字节
GROUP_ID	NUMBER	锁的组标识符
OPEN_OPT_ DEADLOCK	NUMBER	如果设置 DAADLOCK 开放选项，为 1，否则为 0
OPEN_OPT_ PERSISTENT	NUMBER	如果设置 PERSISTENT 开放选项，为 1，否则为 0
OPEN_OPT_ PROCESS_OWNED	NUMBER	如果设置 PROCESS_OWNED 开放选项，为 1，否则为 0
OPEN_OPT_NO_XID	NUMBER	如果设置 NO_XID 开放选项，为 1，否则为 0
CONVERT_OPT_ GETVALUE	NUMBER	如果设置 GETVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ PUTVALUE	NUMBER	如果设置 PUTVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ NOVALUE	NUMBER	如果设置 NOVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ DUBVALUE	NUMBER	如果设置 DUBVALUE 转换选项，为否则为 0
CONVERT_OPT_ NOQUEUE	NUMBER	如果设置 NOQUEUE 转换选项，为 1，否则为 0
CONVERT_OPT_ EXPRESS	NUMBER	如果设置 EXPRESS 转换选项，为 1，否则为 0
CONVERT_OPT_ NODEADLOCKWAIT	NUMBER	如果设置 NODEADLOCKWAIT 转换选项，为 1，否则为 0
WHICH_QUEUE	NUMBER	锁当前所在的队列。NULL 队列为 0；GRANTED 队列为 1；CONVER 队列为 2
LOCKSTATE	NUMBER	拥有者所看到的锁的状态
AST_EVENT0	NUMBER	最后一个 AST 事件
OWNER_NODE	NUMBER	如果这个锁请求被其他锁请求阻塞，为否则为 0
BLOCKER	NUMBER	如果这个锁正阻塞其他锁，为 1，否则为 0

43. V$DLM_CONVERT_LOCAL

这个视图显示本地锁转换操作所用的时间。

列	数据类型	说明
INST_ID	NUMBER	实例的 ID
CONVERT_TYPE	VARCHAR2(64)	列在表 B-3 中的转换类型
AVERAGE_CONVERT _TIME	NUMBER	每种锁操作类型的平均转换时间，以百分之一秒计
CONVERT_COUNT	NUMBER	锁转换操作的数目

44. V$DLM_CONVERT_REMOTE

V$DLM_CONVERT_REMOTE 显示远程锁转换操作的时间（见表 B-3）。

列	数据类型	说明

INST_ID	NUMBER	实例的 ID
CONVERT_TYPE	VARCHAR2(64)	列在表 B-3 中的转换类型
AVERAGE_CONVERT_TIME	NUMBER	每种锁操作类型的平均转换时间，以百分之一秒计
CONVERT_COUNT	NUMBER	操作的次数

表 B-3　CONVERT_TYPE 列的值

转换类型	说　明
NULL->SS	NULL 模式到子共享模式
NULL->SX	NULL 模式到共享互斥模式
NULL->S	NULL 模式到共享模式
NULL->SSX	NULL 模式到子共享互斥模式
SS->SX	NULL 模式到互斥模式
SS->S	子共享模式到共享互斥模式
SS->SSX	子共享模式到子共享互斥模式
SS->X	子共享模式到互斥模式
SX->S	共享互斥模式到共享模式
SX-SSX	共享模式到子共享互斥模式
SX->X	共享互斥模式到互斥模式
S->SX	共享模式到共享互斥模式
S-SSX	共享模式到子共享互斥模式
S->X	共享模式到互斥模式
SSX->X	共享互斥模式到互斥模式

45.　V$DLM_LATCH

这个视图已经废弃。关于 DCM 栓锁的性能统计数据请参阅 V$LATCH。

46.　V$DLM_LOCKS

这是一个并行服务器视图。此视图列出锁管理程序当前已知的被其他锁阻塞或阻塞其他锁的信息。

列	数据类型	说明
LOCKP	RAW(4)	锁指针
GRANT_LEVEL	VARCHAR2(9)	锁的授权级别
REQUEST_LEVEL	VARCHAR2(9)	锁的请求级别
RESOURCE_NAME1	VARCHAR2(30)	锁的资源名
RESOURCE_NAME2	VARCHAR2(30)	锁的资源名
PID	NUMBER	拥有锁的进程标识符
TRANSACTION_ID0	NUMBER	锁所属的事务处理标识符的低 4 个字节
TRANSACTION_ID1	NUMBER	锁所属的事务处理标识符的高 4 个字节
GROUP_ID	NUMBER	锁的组标识符
OPEN_OPT_ DEADLOCK	NUMBER	如果设置 DAADLOCK 开放选项，为 1，否则为 0
OPEN_OPT_ PERSISTENT	NUMBER	如果设置 PERSISTENT 开放选项，为 1，否则为 0
OPEN_OPT_	NUMBER	如果设置 PROCESS_OWNED 开放

PROCESS_OWNED		选项，为 1，否则为 0
OPEN_OPT_NO_XID	NUMBER	如果设置 NO_XID 开放选项，为 1，否则为 0
CONVERT_OPT_GETVALUE	NUMBER	如果设置 GETVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_PUTVALUE	NUMBER	如果设置 PUTVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_NOVALUE	NUMBER	如果设置 NOVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_DUBVALUE	NUMBER	如果设置 DUBVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_NOQUEUE	NUMBER	如果设置 NOQUEUE 转换选项，为 1，否则为 0
CONVERT_OPT_EXPRESS	NUMBER	如果设置 EXPRESS 转换选项，为 1，否则为 0
NODEADLOCKWAIT	NUMBER	如果设置 NODEADLOCKWAIT 转换选项，为 1，否则为 0
NODEADLOCKBLOCK	NUMBER	如果设置 NODEADLOCKBLOCK 转换选项，为 1，否则为 0
WHICH_QUEUE	NUMBER	锁当前所在的队列。NULL 队列为 0；GRANTED 队列为 1；CONVER 队列为 2
LOCKSTATE	VARCHAR2(64)	拥有者所看到的锁的状态
AST_EVENT0	NUMBER	最后一个 AST 事件
OWNER_NODE	NUMBER	节点标识符
BLOCKED	NUMBER	如果这个锁请求被其他锁请求阻塞，为 1，否则为 0
BLOCKER	NUMBER	如果这个锁正阻塞其他锁，为 1，否则为 0

47. V$DLM_MISC

这个视图显示其他 DLM 统计数据

列	数据类型	说明
STATISTIC#	NUMBER	数据统计号
NAME	VARCHAR2(64)	统计数据名
VALUE	NUMBER	与统计数据有关的值

48. V$DLM_RESS

这是一个并行服务器视图。它显示锁管理器当前所知的所有资源的信息。

列	数据类型	说明
RESP	RAW(4)	资源指针
RESOURCE_NAME	VARCHAR2(30)	锁的十六进制表示的资源名
ON_CONVERT_Q	NUMBER	如果在转换队列上，为 1，否则为 0
ON_GRANT_Q	NUMBER	如果在授权队列上，为 1，

		否则为 0
PERSISTENT_RES	NUMBER	如果是一个永久资源，为 1，否则为 0
RDOMAIN_NAME	VARCHAR2(25)	恢复域名
RDOMAINP	RAW(4)	恢复指针域名
MASTER_NODE	NUMBER	主机节点 ID
NEXT_CVT_LEVEL	VARCHAR2(9)	全局转换队列上转换的下一个锁的级别
VALUE_BLK_STATE	VARCHAR2(32)	值块的状态
VALUE_BLK	VARCHAR2(64)	值块的前 64 字节

49. V$ENABLEDPRIVS

此视图显示启用的权限。这些权限可在表 SYS.SYSTEM_PRIVIELEGES_MAP 中找到。

列	数据类型	说明
PRIV_NUMBER	NUMBER	启用权限的数字标识符

50. V$ENQUEUE_LOCK

这个视图显示排队状态对象拥有的所有锁。这个视图中的列等同于 V$LOCK 视图中的列。更多的信息，请参阅 V$LOCK。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	拥有或获得此锁的会话的标识符
TYPE	VARCHAR2(2)	锁的类型。可能具有锁的用户和系统类型列表
ID1	NUMBER	锁标识符#1（依赖于类型）
ID2	NUMBER	锁标识符#2（依赖于类型）
LMODE	NUMBER	会话拥有此锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
REQUEST	NUMBER	进程请求锁的模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
CTIME	NUMBER	授予当前模式以来的时间
BLOCK	NUMBER	此锁正阻塞其他锁

51. V$EVENT_NAME

这个视图包含等待事件的有关信息

列	数据类型	说明
EVENT#	NUMBER	等待事件号
NAME	VARCHAR2(64)	等待事件名
PARAMETER1	VARCHAR2(64)	等待事件的第一个参数的说明
PARAMETER2	VARCHAR2(64)	等待事件的第二个参数的说明
PARAMETER3	VARCHAR2(64)	等待事件的第三个参数的说明

52. V$EXECUTE

这个视图并行执行的有关信息。

列	数据类型	说明
PID	NUMBER	会话 ID
DEPTH	NUMBER	深度
FUCTION	VARCHAR2(10)	会话系列号
TYPE	VARCHAR2(7)	计划表中的 OBJECT_NODE 的名称
NVALS	NUMBER	OBJECT_NODE 占用的时间
VAL1	NUMBER	编号 1 的值
VAL2	NUMBER	编号 2 的值
SEQH	NUMBER	一个序列
SEQL	NUMBER	一个序列

53. V$FALSE_PING

这个视图是一个并行服务器视图。这个视图显示可能正处于假 ping 的缓冲区。即，那些由相同的锁随其他 ping 了 10 次以上的缓冲区保护的 ping 了 10 次以上的缓冲区。标识为假 ping 的缓冲区可重新映射到 GC-FILE_TO_LOCKS 以减少锁冲突。

列	数据类型	说明
FILE#	NUMBER	文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BLOCKS#	NUMBER	块号
STATUS	VARCHAR2(1)	块状态： FREE=当前不用 XCUR=互斥的 CR=一致的读取 READ=从磁盘读 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的 PCM 锁变换数。此例已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	锁必须从磁盘重新读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并

		且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
NAME	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2	非分区对象为 NULL
KIND	VARCHAR2(12)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_CLASS	NUMBER	锁元素类

54. V$FAST_START_SERVERS

此视图提供执行并行事务处理恢复的恢复从服务器的有关信息

列	数据类型	说明
STATE	VARCHAR2(11)	服务器状态：IDLE 或 RECOVERING
UNDOBLOCKSDONE	NUMBER	至今所做的分配工作的百分比
PID	NUMBER	进程 ID

55. V$FAST_START_TRANSACTIONS

这个视图包含与 Oracle 正在恢复的事务处理的进展有关的信息。

列	数据类型	说明
USN	NUMBER	事务处理的撤消段号
SLT	NUMBER	在回退段内的位置
SEQ	NUMBER	位置的具体编号
STATE	VARCHAR2(16)	事务处理的状态可能是
UNDOBLOCKSDONE	NUMBER	这个事务处理中完成的撤消块数目 TO BE RECOVERED、RECOVERING
UNDOBLOCKSTOTAL	NUMBER	需要恢复的撤消块总数
PID	NUMBER	被分配给当前服务器的 ID
CPUTIME	NUMBER	进行恢复的已用时间，以秒计
PARENTUSN	NUMBER	PDML 中父级事务处理的撤消段号
PARENTSLT	NUMBER	PDML 中父级事务处理的位置
PARENTSEQ	NUMBER	PDML 中父级事务处理的序列号

56. V$FILE_PING

这个视图显示每个数据文件 ping 的块号。这个信息又可以用来确定对现有数据文件的访问模式和确定从数据文件块到 PCM 锁的新映射。

列	数据类型	说明

FILE_NUMBER	NUMBER	数据文件号
FREQUENCY	NUMBER	频率
X_2_NULL	NUMBER	文件中所有块从互斥到空的锁转换数
X_2_NULL_FORCED^|_WRITE	NUMBER	指定文件中的块由于互斥到空的转换而强制写的数目
X_2_NULL_FORCED^|_STALE	NUMBER	文件中的块由于互斥到空的转换而使其 STATE 的次数
X_2_S	NUMBER	文件中所有块从互斥到共享的锁转换数
X_2_S_FORCED_	NUMBER	指定文件中的块由于互斥到共享的转换而强制写的数目
X_2_SSX	NUMBER	文件中所有块从互斥到共享互斥的锁转换数
X_2_SSX_FORCED^|_WRITE	NUMBER	指定文件中的块由于互斥到子共享互斥的转换而强制写的数目
S_2_NULL	NUMBER	文件中所有块从共享到空的锁转换数
S_2_NULL_FORCED^|_STALE	NUMBER	文件中的块由于共享到空的转换而使其 STATE 的次数
SS_2_NULL	NUMBER	文件中所有块从子共享到空的锁转换数
SS_2_RLS	NUMBER	释放的 PCM 锁子共享锁的数目。在 Oracle 8.1 中为 0
WRB	NUMBER	实例接收到一个跨实例调用这个文件的写入单个缓冲区的次数
WRB_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而写入单个缓冲区导致写入的块数
RBR	NUMBER	实例接收到一个跨实例调用这个文件的重用块范围的次数
RBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而重用块范围的块数
RBR_FORECD_ STATE	NUMBER	由于跨实例调用而重用块范围而使得这个文件中的块 STATE 的数目
CBR	NUMBER	实例接收到一个跨实例调用这个文件的检查点块范围的次数
CBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件的检查点跨范围所导致的写入这个文件中块的数目
NULL_2_X	NUMBER	指定文件中所有块从空到互斥的锁转换数
S_2_X	NUMBER	指定文件中所有块从共享到互斥的锁转换数
SSX_2_X	NUMBER	指定文件中所有块从子共享互斥到的互斥的锁转换数
NULL_2_S	NUMBER	指定文件中所有块从空到共享的锁转换数转换数
NULL_2_SS	NUMBER	指定文件中所有块从空到子共享的锁
OP_2_SS	NUMBER	打开的 PCM 锁子共享锁的数目。Oracle 8.1 中为 0

57. V$FILESTAT

这个视图包含文件读写统计数据的有关信息

列	数据类型	说明
FILE#	NUMBER	文件号

PHYRDS	NUMBER	所在行的物理读的次数
PHYWRTS	NUMBER	DBWR 要求写入的次数
PHYBLKRD	NUMBER	物理块读取的次数
PHYBLKWRT	NUMBER	写入磁盘的块数目；如果所有写入都是单块，这个数与 PHYWRTS 相同
READTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成读取所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
WRITETIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成写入所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
AVGIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE ，则为完成最后 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE ，则为完成最后 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MINIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个 I/O 所用的最小时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MAXIOWTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成单个写入所用的最大时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MAXIORTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成单个读取所用的最大时间（以百分之一秒计）；如果该参数为 FALSE，则为 0

58. V$FIXED_TABLE

这个视图显示数据库中所有动态性能表、视图和导出表。某些 V$表(如 V$ROLLNAME)涉及实际的表，因此没有列出。

列	数据类型	说明
NAME	VARCHAR2(30)	对象名
OBJECT_ID	NUMBER	固定对象的标识符
TYPE	VARCHAR2(5)	对象类型：TABLE、VIEW
TABLE_NUM	NUMBER	如果动态性能表为 TABLE 类型，则为标识它的号码

59. V$FIXED_VIEW_DEFINITION

这个视图包含所有固定视图（以 V$起头的视图）的定义。应该仔细使用这些表。Oracle

试图保持这些固定视图在各个版本中都相同，但还是可能会更改某些固定视图的定义而不作通知。可通过利用动态性能表的索引列来使用这些定义以优化查询。

列	数据类型	说明
VIEW_NAME	VARCHAR2(30)	固定视图名
VIEW_DEFINTION	VARCHAR2(2000)	固定视图定义

60. V$GLOBAL_BLOCKED_LOCKS

这个视图显示全局阻塞的锁。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址（raw）
KADDR	RAW(4)	锁地址（raw）
SID	NUMBER	拥有或获得此锁的会话的标识符（number）
TYPE	VARCHAR2(2)	资源类型（number）
ID1	NUMBER	资源标识符#1（number）
ID2	NUMBER	资源标识符#2（number）
LMODE	NUMBER	拥有的锁模式（number）
REQUEST	NUMBER	请求的锁模式（number）
CTIME	NUMBER	授予当前模式以来的时间

61. V$GLOBAL_TRANSACTION

这个视图显示当前归档的全局事务处理的有关信息

列	数据类型	说明
FORMATID	NUMBER	全局事务处理的格式标识符
GLOBALID	NUMBER	全局事务处理的标识符
BRANCHID	NUMBER	全局事务处理的分支标识符
BRANCHES	NUMBER	全局事务处理的分支总数
REFCOUNT	NUMBER	这个全局事务处理的同级数，必须与分支数相同
PREPARECOUNT	NUMBER	已经预备的全局事务处理的分支数
STATE	VARCHAR2(18)	全局事务处理的分支状态
FLAGS	NUMBER	状态的数字表示
COUPLING	VARCHAR2(15)	分支是松散耦合还是紧密耦合

62. V$HS_AGENT

这个视图确定当前运行在某个主机上的 HS 代理，每个代理进程采用一行。

列	数据类型	说明
AGENT_ID	NUMBER	连接到代理使用的 Net8 会话标识符（listernet.ora SID）
MACHINE	NUMBER	操作系统机器名
PROCESS	NUMBER	代理的操作系统进程标识符
PROGRAM	NUMBER	代理的程序名
OSUSER	NUMBER	操作系统用户

STARTTIME	DATE	启动时间
AGENT_TYPE	NUMBER	代理的类型
FDS_CLASS_ID	NUMBER	外部数据存储类的 ID
FDS_INST_ID	NUMBER	外部数据存储的实例名

63. V$HS_SESSION

这个视图确定当前为 Oracle 服务器打开的 HS 会话组。

列	数据类型	说明
HS_SESSION_ID	NUMBER	唯一的 HS 会话标识符
AGENT_ID	NUMBER	V$HS_AGENT 的外部键
SID	NUMBER	用户会话标识符（V$SESSION 的外键）
DB_LINK	VARCHAR2(128)	用来访问代理的服务器数据库连接名，NULL 表示没有使用数据库连接（即，在使用外部过程时）
DB_LINK_OWNER	NUMBER	DB_LINK 中的数据库连接的拥有者
STARTTIME	DATE	连接的启动时间

64. V$INDEXED_FIXED_COLUMN

这个视图显示索引的动态性能表（X$表）中的列。X$表可以更改而不用通告。仅将此视图用于编写针对固定视图（V$视图）的查询更为有效。

列	数据类型	说明
TABLE_NAME	VARCHAR2(30)	索引的动态性能表的名称
INDEX_NUMBER	NUMBER	区分某个列属于哪个索引的编号
COLUMN_NAME	VARCHAR2(30)	被索引的列号
COLUMN_POSITION	NUMBER	索引键中列的位置（这主要与多列索引有关）

65. V$INSTANCE

这个视图显示当前实例的状态。这个版本的 V$INSTANCE 与前面版本的的 V$INSTANCE 不兼容。

列	数据类型	说明
INSTANCE_NUMBER	NUMBER	实例注册所用的实例号。对应与 INSTANCE _NUMBER 初始化参数。可参阅 INSTANCE_NUMBER
INSTANCE_NAME	VARCHAR2(16)	实例名
HOST_NAME	VARCHAR2(64)	主机名
VERSION	VARCHAR2(17)	RDBMS 版本
STARTUP_TIME	DATE	实例启动的时间
STATUS	VARCHAR2(7)	STARTED/MOUNTED/OPEN STARTED：启动安装后或数据库关闭后，OPEN：启动后或数据库打开后

PARALLEL	VARCHAR2(3)	YES/NO：是否并行服务器模式
THREAD#	NUMBER	实例打开的重做线程
ARCHIVER	VARCHAR2(7)	STOPPED/STARTED/FAILED；FAILED 表示归档程序最后一次归档某个日志失败，但在 5 分钟内将重试
LOG_SWITCH_WAIT	VARCHAR2(11)	正在等待 ARCHIVELOG/CLEAR LOG/ CHECKPOINT 事件日志切换。注意：如果 ALTER SYSTEM SWITCH LOGFILE 挂起，但在当前联机重做日志中还有空间则这个值为 NULL
LOGINS	VARCHAR2(10)	ALLOWED/RESTRICTED
SHUTDOWN_PENDING	VARCHAR2(3)	YES/NO
DATABASE_STATUS	VARCHAR2(17)	数据库的状态

66. V$INSTANCE_RECOVERY

这个视图用来监控实现恢复读取的用户特定限制的机制。

列	数据类型	说明
RECOVERY_ EXTIMATED_IOS	NUMBER	根据快速启动检查点在内存中的值，估计在恢复中将处理的块数
ACTUAL_REDO _BLOCKS	NUMBER	恢复所需的重做块的当前实际数目
TARGET_REDO _BLOCKS	NUMBER	恢复必须处理的重做块的当前目标数目
LOG_FILE_SIZE _REDO_BLKS	NUMBER	保证在检查点完成前不进行日志切换所需重做块的最大数目
LOG_CHKPT_TIME OUT_READ_BLKS	NUMBER	恢复中为满足 LOG_CHECKPOINT_TIMEOUT 需要处理的重做块数
LOG_CHKPT_ INTERVAL_READ_ BLKS	NUMBER	恢复中为满足 LOG_CHECKPOINT_INTERVAL 需要处理的重做块数
FAST_START_IO_ TARGET_REDO_ BLKS	NUMBER	恢复中为满足 FAST_START_IO_ TARGET 需要处理的重做块数

67. V$LATCH

这个视图列出非父级栓锁的统计数据和父级栓锁的汇总统计数据。即，父级栓锁的统计数据包括从其每个子级栓锁开始的数据。

说明：列 SLEEP5、SLEEP6、...SLEEP11 是为了与以前的 Oracle 班本兼容而给出的。不累加这些列的数据。

列	数据类型	说明
ADDR	RAW(4)	栓锁对象的地址
LATCH#	NUMBER	栓锁编号
LEVEL#	NUMBER	栓锁级别
NEME	VARCHAR2(64)	栓锁名
GETS	NUMBER	等待获得的次数

MISSES	NUMBER	等待获得但第一次尝试失败的次数
SLEEPS	NUMBER	在需要等待时睡眠的次数
IMMEDIATE_GETS	NUMBER	不用等待获得的次数
IMMEDIATE_MISSES	NUMBER	不等待获得失败的次数
WAITERS_WOKEN	NUMBER	等待被唤醒多少次
WAITS_HOLDING_LATCH	NMBER	拥有一个不同的栓锁时等待的次数
SPIN_GETS	NUMBER	第一次尝试失败，但在以后的轮次中成功
SLEEP1	NUMBER	睡眠 1 次的等待
SLEEP2	NUMBER	睡眠 2 次的等待
SLEEP3	NUMBER	睡眠 3 次的等待
SLEEP4	NUMBER	睡眠 4 次的等待
SLEEP5	NUMBER	睡眠 5 次的等待
SLEEP6	NUMBER	睡眠 6 次的等待
SLEEP7	NUMBER	睡眠 7 次的等待
SLEEP8	NUMBER	睡眠 8 次的等待
SLEEP9	NUMBER	睡眠 9 次的等待
SLEEP10	NUMBER	睡眠 10 次的等待
SLEEP11	NUMBER	睡眠 11 次的等待

68. V$LATCHHOLDER

这个视图包含当前栓锁拥有者的相关信息。

列	数据类型	说明
PID	NUMBER	拥有栓锁的进程标识符
SID	NUMBER	拥有栓锁的会话标识符
LADDR	LAW(4)	栓锁地址
NAME	VARCHAR2	被拥有的栓锁名称

69. V$LATCHNAME

这个视图包含确定 V$LATCH 视图中给出栓锁的栓锁名的有关信息。V$LATCHNAME 的行与 V$LATCH 的行具有一一对应的关系。更多的信息，可参阅 V$LATCH。

列	数据类型	说明
LATCH#	NUMBER	栓锁号
NAME	VARCHAR2(64)	栓锁名

70. V$LATCH_CHILDREN

这个视图包含子级栓锁的有关统计数据。这个视图包含了 V$LATCH 的所有列再加上 CHILD# 列。注意，如果子级栓锁的 LATCH# 列互相配合，则子级栓锁具有相同的父级栓锁。更多信息，可参阅 V$LATCH。

列	数据类型	说明
ADDR	RAW(4)	栓锁对象的地址
LATCH#	NUMBER	父级栓锁的栓锁号
CHILD#	NUMBER	LATCH#中给出的父级栓锁的子级栓锁编号

LEVEL#	NUMBER	栓锁级别
NAME	VARCHAR2(64)	栓锁名
GETS	NUMBER	等待获得的次数
MISSES	NUMBER	等待获得但第一次尝试失败的次数
SLEEPS	NUMBER	在需要等待时睡眠的次数
IMMEDIATE_GETS	NUMBER	不用等待获得的次数
IMMEDIATE_MISSES	NUMBER	不等待获得失败的次数
WAITERS_WORKEN	NUMBER	等待被唤醒多少次
WAITS_HOLDING _LATCH	NUMBER	拥有一个不同的栓锁时等待的次数
SPIN_GETS	NUMBER	第一次尝试失败，但在以后的轮次中成功
SLEEPn	NUMBER	睡眠 n 次的等待

71．V$LATCH_MISSES

这个视图包含试图获得一个栓锁但失败的有关信息

列	数据类型	说明
PARENT_NAME	VARCHAR2	父级栓锁名
WHERE	VARCHAR2	试图获得此栓锁的位置
NWFAIL_COUNT	NUMBER	不等待获取栓锁失败的次数
SLEEP_COUNT	NUMBER	导致睡眠的获取尝试的次数
WTR_SLP_COUNT	NUMBER	
LONGHOLD_COUNT	NUMBER	

72．V$LATCH_PARENT

这个视图包含父级栓锁的有关统计数据。此视图的列与 V$LATCH 的列相同。更多的信息请参阅 V$LATCH。

73．V$LIBRARYCACHE

这个视图包含库高速缓存性能与活动的有关统计数据。

列	数据类型	说明
NAMESPACE	VARCHAR2(15)	名称空间
GETS	NUMBER	为这个名称空间中的对象请求某个锁的次数
GETHITS	NUMBER	在内存中找到某个对象的句柄的次数
GETHITRATIO	NUMBER	GETHITS 与 GETS 的比例
PINS	NUMBER	为这个名称空间中的对象请求 PIN 的次数
PINHITS	NUMBER	在内存中找到相应库对象的所有元数据片的次数
PINHITRATIO	NUMBER	PINHITS 与 PINS 的比例
RELOADS	NUMBER	自某个对象句柄建立以来，进行该对象的非第一次 PIN 的任意 PIN，这样需要将对象从磁盘装入
INVALIDATIONS	NUMBER	此名称空间中的对象由于相关的对象被修改而无效的总次数
DLM_LOCK_ REQUESTS	NUMBER	GET 请求锁定实例锁的次数

DLM_PIN_REQUESTS	NUMBER	PIN 请求锁定实例锁的次数
DLM_PIN_RELEASES	NUMBER	发布请求 PIN 实例锁的次数
DLM_INVALIDATION_REQUESTS	NUMBER	GET 请求无效实例锁的次数
DLM_INVALIDATIONS	NUMBER	从其他实例接收到的无效 ping 的次数

74. V$LICENSE

这个视图包含有关认证限制的信息

列	数据类型	说明
SESSIONS_MAX	NUMBER	实例允许的并发用户会话的最大数目
SESSIONS_WARNING	NUMBER	实例的并发用户会话的警告数目
SESSIONS_CURRENT	NUMBER	并发用户会话的最大数目
SESSIONS_HIGHWATER	NUMBER	实例启动以来的并发用户会话的最大数目
USERS_MAX	NUMBER	数据库允许的指定用户最大数目

75. V$LOADCSTAT

这个视图包含执行一个直接装载中搜集的 SQL*Loader 的统计数据。这些统计数据应用到整个装载过程。对于这个表的任何 SELECT 都会产生 “ 无行返回 ”，因为不能同时装载数据又同时进行查询。

列	数据类型	说明
READ	NUMBER	读取记录数
REJECTED	NUMBER	拒绝的记录数
TDISCARD	NUMBER	装载中废弃的记录数
NDISCARD	NUMBER	从当前文件中废弃的记录数

76. V$LOADTSTAT

列	数据类型	说明
LOADED	NUMBER	装载的记录数
REJECTED	NUMBER	拒绝的记录数
FAILWHEN	NUMBER	不能满足任意 WHEN 子句的记录数
ALLNULL	NUMBER	全空从而不装载的记录数
LEFT2SKIP	NUMBER	在连续装载中要跳过的记录数
PTNLOADED	NUMBER	装载 PTN 的记录数

77. V$LOCK

这个视图列出 Oracle 服务器当前拥有的锁以及未完成的锁或栓锁请求。

列	数据类型	说明

ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	拥有或获得此锁的会话的标识符
TYPE	VARCHAR2(2)	锁的类型。可能具有锁的用户和系统类型列表，请参阅表 B-4 和 B-5
ID1	NUMBER	锁标识符#1（依赖于类型）
ID2	NUMBER	锁标识符#2（依赖于类型）
LMODE	NUMBER	会话拥有此锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
REQUEST	NUMBER	进程请求锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
CTME	NUMBER	授予当前模式以来的时间
BLOCK	NUMBER	此锁正阻塞其他锁

表 B-4 中用户类型上的锁可由用户应用程序获得。任何阻塞其他进程的进程都可能拥有这些锁中的某一个。

表 B-4　用户类型

用户类型	说明
TM	DML 排队
TX	事务处理排队
UL	用户提供

表 B-5 中系统类型上的锁被拥有的时间计短。

表 B-5　类型列的值：系统类型

用户类型	说明
BL	缓冲区散列表实例
CF	控制文件模式全局排队
CI	交叉实例功能调用实例
CU	游标绑定
DF	数据文件实例
DL	直接装载程序并行索引创建
DM	安装/启动数据库主/副实例
DR	分布式恢复进程
DX	分布式事务处理项
FS	文件集
HW	特定段上的空间管理
IN	实例号

IR	实例恢复串行全局队列
IS	实例状态
IV	库高速缓存无效实例
JQ	作业队列
KK	线程突跳
LA...LP	库高速缓存锁实例锁（A...P=名称空间）
MM	安装定义全局队列
MR	介质恢复
NA...NZ	库高速缓存固定实例（A...Z=名称空间）
PF	口令文件
PI，PS	并行操作
PR	进程启动
QA...QZ	行高速缓存实例（A...Z=高速缓存）
RT	重做线程全局队列
SC	系统提交编号实例
SM	SMON
SN	序列号实例
SQ	序列号队列
SS	排序段
ST	空间事务处理队列
SV	序列号值
TA	一般队列
TS	临时段队列（ID2=0）
TS	新块分配队列（ID2=1）
TT	临时表队列
UN	用户名
US	撤消段 DDL
WL	开始写重做日时局实例

表 B-5 中系统类型上的锁被拥有的时间计短。

78．V$LOCK_ACTIVITY

这是一个并行服务器视图。这个视图显示当前实例的 DLM 锁的操作。每行对应锁操作的一种类型。

列	数据类型	说明
FROM_VAL	VARCHAR2(4)	PCM 锁初始状态：NUL；S；X；SSX
TO_VAL	VARCHAR2(4)	PCM 锁初始状态：NULL；S；X；SSX
ACTION_VAL	VARCHAR2(51)	锁转换说明 读取的锁缓冲区 写入的锁缓冲区 使缓冲区 CR(非写入)读锁为写 使缓冲区 CR(写灰缓冲区) 将写锁降为读(写灰缓冲区) 写事务处理表/撤消块 事务处理表/撤消块（写灰缓区） 使事务处理表/撤消块可共享 重新配备事务处理表写机制
COUNTER	NUMBER	执行的锁操作次数

79．V$LOCK_ELEMENT

这是一个并行服务器视图。每个 PCM 锁在此视图中有一项，这项由缓冲区高速缓存使用。对应一个锁元素的 PCM 锁的名称为{‘BL’，索引，类}。

列	数据类型	说明
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素地址。如果不止一个缓冲区具有相同的地址，则这些缓冲区由相同的 PCM 覆盖。
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁的名称。
INDX	NUMBER	平台专用的锁管理程序标识符
CLASS	NUMBER	平台专用的锁管理程序标识符
MODE_HELD	NUMBER	与平台相关的拥有锁模式的值；一般：3=共享；5=互斥
BLOCK_COUNT	NUMBER	PCM 锁覆盖的块数
RELEASING	NUMBER	如果 PCM 降级，则非零
ACQUIRING	NUMBER	如果 PCM 升级，则非零
INVALID	NUMBER	如果 PCM 锁无效，则非零（系统失败后，锁可能变得无效）
FLAGS	NUMBER	LE 的进程级标志

80．V$LOCKED_OBJECT

这个视图列出系统上的每个事务处理所获得的所有锁。

列	数据类型	说明
XIDUSN	NUMBER	撤消的段号
XIDSLOT	NUMBER	位置号
XIDSQN	NUMBER	序列号
OBJECT_ID	NUMBER	被锁定的对象 ID
SESSION_ID	NUMBER	会话 ID
ORACLE_USERNAME	VARCHAR2(30)	Oracle 用户名
OS_USER_NAME	VARCHAR2(15)	OS 用户名
PROCESS	VARCHAR2(9)	OS 进程名
LOCKED_MODE	NUMBER	锁模式

81．V$LOCKS_WITH_COLLISIONS

这是一个并行服务器视图。利用这个视图来查找保护多个缓冲区的锁，其中每个曾经被强制写或强制读至少 10 次。很可能这些缓冲区由于被映射到相同的锁而经历过假的固定。

列	数据类型	说明
LOCK_ELEMENT_ADDR	NUMBER	撤消的段号

82．V$LOG

此视图包含来自控制文件的日志文件信息

列	数据类型	说明
GROUP#	NUMBER	日志组号
THREAD#	NUMBER	日志线程号
SEQUENCE#	NUMBER	日志序列号
BYTES	NUMBER	以字节表示的日志大小
MEMBERS	NUMBER	日志组中成员数
ARCHIVED	VARCHAR2	归档状态：YES、NO
STATUS	VARCHAR2(16)	日志状态。STATUS 列可具有表-6 中的值
FIRST_CHANGE#	NUMBER	日志中的最低 SCN
FIRST_TIME	DATE	日志中的第一个 SCN 的时间

表 B-6 给出日志 STATUS 列中的值。

表 B-6　STATUS 列的值

STATUS	含　义
UNUSED	表示联机重做日志文件从来没有写入过。如果不是当前重做日志，这是在刚增加的或刚好在 RESETLOGS 后的重做日志的状态
CURRENT	指出这是当前重做日志。这表示此重做日志是活动的。这个重做日志打开或关闭的
ACTIVE	表示此日志是活动的，但不是当前日志。需要进行崩溃恢复。有可能正用于块恢复。它能够或不能够归档
CLEARING	表示此日志在 ALTER DATABASE CLEAR LOGFILE 命令后正在作为空日志重建。在清除此日志后，这个状态变为 UNUSED
CLEARINGCURRENT	表示当前日志正由于关闭线程而被清除。如果在切换中存在某种故障（如写新日志标题的 I/O 错误），此日志可保持这种状态
INACTIVE	表示实例恢复不再需要这个日志。它可能在用于介质恢复。它有可能归档，也有可能不归档

83. V$LOGFILE

这个视图包含重做日志文件的有关信息

列	数据类型	说明
GROUP#	NUMBER	重做日志组标识符号
STATUS	VARCHAR2	这个日志成员的状态：INVALID(文件不可访问)、STATE(文件的内容不完整)、DELETED（文件不再使用)、或空（文件正在使用）
MEMBER	VARCHAR2	重做日志成员名

84. V$LOGHIST

这个视图包含控制文件中的日志历史信息。此视图是为历史兼容而保留的。建议使用C$LOG_HISTORY。更多的信息，请参阅 V$LOG_HISTORY。

列	数据类型	说明
THREAD#	NUMBER	日志线程号
SEQUENCE#	NUMBER	日志序列号
FIRST_CHANGE#	NUMBER	日志中的最低 SCN
FIRST_TIME	DATE	日志中的第一个 SCN 时间
SWITCH_CHANGE#	NUMBER	发生日志切换的 SCN；比日志中最高 SCN 更高的那个 SCN

85. V$LOGMNR_CONTENTS

这个视图包含日志历史信息。

列	数据类型	说明
SCN	NUMBER(15)	系统更改号
TIMESTAMP	DATE	时间戳
THREAD#	NUMBER	线程号
LOG_ID	NUMBER	日志 ID
XIDUSN	NUMBER	事务处理 ID 撤消段号
XIDSLOT	NUMBER	事务处理 ID 位置号
XIDSQN	NUMBER	事务处理 ID 日志序列号
RBASQN	NUMBER	RBA 日志序列号
RBABLK	NUMBER	RBA 块号
RBABYTE	NUMBER	RBA 字节偏移量
UBAFIL	NUMBER	UBA 文件号
UBABLK	NUMBER	UBA 块号
UBAREC	NUMBER	UBA 记录索引
UNASQN	NUMBER	UBA 撤消块序列号
ABS_FILE#	NUMBER	数据块绝对文件号
REL_FILE#	NUMBER	数据块相对文件号
DATD_BLK#	NUMBER	数据块号
DATA_OBJ#	NUMBER	数据块对象号
DATA_DOBJ#	NUMBER	数据块数据对象号
SEG_OWENR	VARCHAR2(30)	段拥有者
SEG_NAME	VARCHAR2(81)	段名
SEG_TYPE	NUMBER	段类型
TABLE_SPACE_NAME	VARCHAR2(30)	段的表空间名
ROW_ID	VARCHAR2(18)	行 ID
SESSION#	NUMBER	会话号
SERIAL#	NUMBER	系列号
USER_NAME	VARCHAR2(30)	用户名
SESSION_INFO	VARCHAR2(4000)	会话信息
ROLLBACK	NUMBER	回退请求
OPERATION	VARCHAR2(30)	操作
SQL_REDO	VARCHAR2(4000)	SQL 重做
SQL_UNDO	VARCHAR2(4000)	SQL 撤消
RS_ID	VARCHAR2(30)	记录集 ID

SSN	NUMBER	SQL 序列号
CSF	NUMBER	连续 SQL 标志
INFO	VARCHAR2(32)	通知信息
STATUS	VARCHAR2(16)	状态

86. V$LOGMNR_DICTIONARY

这个视图包含日志历史信息

列	数据类型	说明
TIMESTAMP	DATE	建立字典的时间
DB_ID	NUMBER	数据库 ID
DB_NAME	VARCHAR2(8)	数据库名
FILENAME	VARCHAR2(513)	字典文件名
DICTIONARY_SCN	NUMBER	建立字典时的系统更改号
RESET_SCN	NUMBER	建立字典时重置日志 SCN
RESET_SCN_TIME	NUMBER	获得重置日志 SCN 建立字典时的时间
ENABLED_THREAD_MAP	RAW(16)	建立字典时当前启用线程的位图
INFO	VARCHAR2(32)	信息/状态消息 BAD_DATE 表示字典文件的 SCN 与日志文件的 SCN 范围不匹配
STATUS	NUMBER	NULL 表示日志文件列表的有效字典文件。非 NULL 值表示进一步的信息作为文本串包含在 INFO 列中

87. V$LOGMNR_LOGS

此视图包含日志信息。

列	数据类型	说明
LOG_ID	NUMBER	指出日志文件。这个字段的值也在 V$LOG 的 LOG_ID 列中给出
FILENAME	VARCHAR2(512)	文件名
LOW_TIME	DATE	文件中任意记录的最早日期
HIGL_TIME	DATE	文件中任意记录的最近日期
DB_ID	NUMBER	数据库 ID
DB_NAME	VARCHAR2(8)	数据库名
RESET_SCN	NUMBER	建立日志时重置日志 SCN
RESET_SCN_TIME	NUMBER	获得重置日志 SCN 建立字典时的时间
THREAD_ID	NUMBER	线程号
THREAD_SQN	NUMBER	线程序列号
LOW_SCN	NUMBER	切换入日志时分配的 SCN
NEXT_SCN	NUMBER	此日志后的 SCN。下一日志的低 SCN
INFO	VARCHAR2(32)	通知性的消息。将 MISSING_LOGFILE 值分配给其中所需日志文件从日志文件列表中遗漏

		的行项
STATUS	NUMBER	表示日志文件的状态。NULL 值表示一个有效的日志文件；非 NULL 值表示进一步的信息为一个文本串包含在 INFO 列中。如果所有日志文件成功地加到文件列表，状态值为 NULL

88. V$LOGMNR_PARAMETERS

此视图包含日志信息。

列	数据类型	说明
START_DATE	DATE	开始搜索的日期
END_DATE	DATE	开始搜索的日期
START_SCN	NUMBER	开始搜索的系统更改号
END_SCN	NUMBER	结束搜索的系统更改号
INFO	VARCHAR2(32)	通知性的消息。
STATUS	NUMBER	状态。NULL 值表示一个参数有效。非 NULL 值表示进一步的信息为一个文本串包含在 INFO 列中。

89. V$LOG_HISTORY

这个视图包含来自控制文件的历史信息。

列	数据类型	说明
THREAD#	NUMBER	归档日志线程号
SEQUENCE#	NUMBER	归档日志序列号
FIRST_TIME	NUMBER	归档日志中的第一项的时间（最低 SCN)。此列以前名为 TIME
FIRST_CHANGE#	NUMBER	日志中最低 SCN。这个列以前名为 FIRST_CHANGE#
NEXT_CHANGE#	NUMBER	日志中最高 SCN。这个列以前名为 HOGH_CHANGE#
RECID	NUMBER	控制文件记录 ID
STAMP	NUMBER	控制文件记录时间戳

90. V$MLS_PARAMETERS

这是一个 Trusted Oracle 服务器视图，它列出 Trusted Oracle 服务器专用的安装参数。

91. V$MTS

这个视图包含优化多线程服务器的信息。

列	数据类型	说明
MAXIMUN_CONNECTIONS	NUMBER	每个调度程序可支持的连接的最大数目。这个值在启动时用 Net8 常量和其他端口专用的信息确定，或者可利用 MTS_DISPATCHERS 参数降低

SERVERS_STARTED	NUMBER	自实例启动以来启动的多线程服务器总数（但不包括启动中的启动的那些多线程服务器）
SERVERS_TERMINATED	NUMBER	自实例启动以来一次运行的服务器最大数目。如果这个值达到了 MTS_MAX_SERVERS 初始化参数设置的值，则应考虑提高 MTS_SERVERS 的值。更详细的信息，请参阅 MTS_SERVERS

92. V$MYSTAT

这个视图包含当前会话的统计数据。

列	数据类型	说明
SID	NUMBER	当前会话的 ID
STATISTIC	NUMBER	统计数据号
VALUE	NUMBER	统计数据值

93. V$NLS_PARAMETERS

这个视图包含 NLS 参数的当前值。

列	数据类型	说明
PARAMETERS	VARCHAR2	参数名： NLS_CALENDAR NLS_CHARACTERSET NLS_CURRECY NLS_DATE_LANGUAGE NLS_ISO_CURRENCY NLS_NUMEC_CHARACTERS NLS_SORT NLS_TERRITORY NLS_UNION_CURRENCY NLS_NCHAR_CHARACTERSET NLS_COMP
VALUE	VARCHAR2	NLS 参数值

94. V$NLS_VALID_VALUES

此视图列出 NLS 参数的所有有效值。

列	数据类型	说明
PARAMETERS	VARCHAR2（64）	参数名：
VALUE	VARCHAR2（64）	LANGUATE；SORT；TERRITORY CHARACTERSETNLS 参数值

95. V$OBJECT_DEPENDENCY

这个视图可用来确定当前共享池中装入的程序包、过程或游标依赖于什么样的对象。例如,它可与 V$SESSION 和 V$SQL 一道用来确定某个用户当前正在执行的 SQL 语句中所用的表。更详细的信息，请参阅 V$SESSION 和 V$SQL。

列	数据类型	说明
FROM_ADDRESS	RAW(4)	当前装入共享池中的过程、程序包或游标的地址
FROM_HASH	NUMBER	当前装入共享池中的过程、程序包或游标的散列值
TO_OWNER	VARCHAR2(64)	所依赖对象的拥有者
TO_NAME	VARCHAR2(1000)	所依赖的对象名
TO_ADDRESS	RAW(4)	所依赖对象的地址。这些地址可用来在 V$DB_OBJECT_CACHE 中查找关于对象的更多信息
TO_HASH	NUMBER	所依赖对象的散列值。这些值可用来在 V$DB_OBJECT_CACHE 中查找关于对象的更多信息
TO_TYPE	NUMBER	所依赖对象的类型

96. V$OBSOLETE_PARAMETER

此视图列出作废的参数。如果任何值为真，则应该检查为什么会这样。

列	数据类型	说明
NAME	VARCHAR2 (64)	参数名
ISSPECIFIED	VARCHAR2 (5)	在配置文件中是否给出此参数

97. V$OFFLINE_RANGE

这个视图显示控制文件中的数据文件中脱机信息。注意，每个数据文件的最后脱机范围保存在 DATAFILE 记录中。详细内容，请参阅 V$DATAFILE。

脱机范围是在数据文件的表空间 ALTER 为 OFFLINE NORMAL 或 READ ONLY，然后再 ALTER 为 ONLINE 或读写时建立的。注意，如果数据文件自身 ALTER 为 OFFLINE 或如果相应表空间 ALTER 为 OFFLINE IMMEDIATE 时，不建立脱机范围。

列	数据类型	说明
RECID	NUMBER	记录 ID
STAMP	NUMBER	记录时间戳
FILE#	NUMBER	数据文件号
OFFLINE_CHANGE#	NUMBER	脱机时的 SCN
ONLINE_CHANGE#	NUMBER	联机时的 SCN
ONLINE_TIME	NUMBER	脱机 SCN 的时间

98. V$OPEN_CURSOR

这个视图列出每个用户会话当前已经打开和分析的游标。

列	数据类型	说明
SADDR	RAW	会话地址

SID	NUMBER	会话标识符
USER_NAME	VARCHAR2(30)	登录到会话的用户
ADDRESS	RAW	与 HASH--VALUE 一道用来唯一地标识会话中执行的 SQL 语句
HASH_VALUES	NUMBER	与 ADDRESS 一道用来唯一地标识会话中执行的 SQL 语句
SQL_TEXT	VARCHAR2(60)	解析进入打开的游标的 SQL 语句的前 60 个字符

99. V$OPTION

这个视图列出与 Oracle 一道安装的选项。

列	数据类型	说明
PARAMETERS	VARCHAR2 (64)	选项名
VALUE	VARCHAR2 (64)	LANGUATE；SORT；TERRITORY CHARACTERSETNLS 参数值

100. V$PARALLEL_DEGREE_LIMIT_MTH

这个视图显示所有可用的并行度限额资源分配方法。

列	数据类型	说明
NAME	VARCHAR2 (40)	并行度限客资源分配方法的名称

101. V$PARAMETER

这个视图列出初始化参数的有关信息。

列	数据类型	说明
NUM	NUMBER	参数名
NAME	VARCHAR2(64)	参数名
		参数类型；1=布尔；2=串；3=整数
VALUE	VARCHAR2(512)	参数值
ISDEFAULT	VARCHAR2(9)	参数是否为缺省值
ISSES_MODIFIABLE	VARCHAR2(5)	TRUE=参数可用 ALTER SEEEION 更改。FALSE=参数不能用 ALTER SESSION 更改
ISSYS_MODIFIABLE	VARCHAR2(9)	IMMEDIATE=参数可用 ALTER SYSTEM 更改。DEFERRED=参数不能用 ALTER SYSTEM 更改
ISMODEIFIED	VARCHAR2(10)	指出参数怎样更改。如果执行一条 ALTER SESSION，则值将被 MODIFIED。如果执行一条 ALTER SYSTEM (它将导致所有当前登录会话的值被修改)，则参数值将为 SYS_ MODEIFIED
ISADJUSTED	VARCHAR2(5)	指出关系型数据库系统调整输入值即，参数值应该为素数，但用户输入一个非素数，因此关系型数据库系统将该值调整为下一个素数)

DESCRIPTION	VARCHAR2(64)	有关此参数的一个描述性的注释

102. V$PING

这是一个并行服务器视图。此视图等同于 V$CACHE 视图，但只显示至少 ping 了一次的块。这个视图包含来自当前实例的 SGA 中每个块的块标题的与特定数据库对象有关的信息。更多的信息，请参阅 V$CACHE。

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BLOCKS#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	块状态： FREE=当前不用 XCUR=互斥 SCUR=当前共享 READ=从磁盘读 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCM 锁转换数。此例已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	锁必须从磁盘重新读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
NAME	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2	非分区对象为 NULL
KIND	VARCHAR2(12)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁名。

103. V$PQ_SESSTAT

这个视图列出并行查询的统计数据。

注意 ：这个视图在以后的版本中将作废。

列	数据类型	说明
STATISTIC	VARCHAR2（30）	统计数据名，请参阅表 B-7

LAST_QUERY	NUMBER	最后处理的统计数据的值
SESSION_TOTAL	NUMBER	整个会话按时到达此点的统计值

已经为这个视图定义了表 B-7 中的统计数据（固定行）。在执行了查询或 DML 操作后，可利用从此视图得出的信息查看所利用的从进程，以及会话与系统的其他信息。

表 B-7　STATISTIC 列中的统计数据名

统计数据（固定行）	说明
Queries Parallelized	并行运行的查询数
DML Parallelized	并行运行的 DML 操作数
DFO Trees	执行 DFO 树的数目
Server Threads	使用的并行服务器的总数
Allocation Height	每个实例服务器请求数
Allocation Width	实例的请求数
Local Msgs Sent	发送的本地（实例内）消息数
Distr Msgs Sent	发送的远程（实例间）消息数
Local Msgs Recv'd	接收到的本地（实例内）消息数
Distr Msgs Recv'd	接收到的远程（实例间）消息数

104.　V$PQ_SLAVE

这个视图列出实例上每个活动并行执行服务器的统计数据。

说明 此视图在未来的版本中将被一个称为 V$PX_PROCESS 的视图替代或废弃。

列	数据类型	说明
SLAVE_NAME	VARCHAR2（4）	并行执行的服务器名
STATUS	VARCHAR2（4）	并行执行服务器的当前状态（BUSY 或 IDLE）
SESSIONS	NUMBER	使用过这个并行执行服务器的会话数目
IDLE_TIME_CUR	NUMBER	当前会话中处理语句时空闲所占的时间总数
BUSY_TIME_CUR	NUMBER	当前会话中处理语句时忙所占的时间总数
CPU_SECS_CUR	NUMBER	用在当前会话的 CPU 时间总数
MSGS_SENT_CUR	NUMBER	处理当前会话的语句时发送的消息数
MSGS_RCVD_CUR	NUMBER	处理当前会话的语句时接收到的消息数
IDLE_TIME_TOTAL	NUMBER	此查询服务器空闲的时间总数
BUSY_TIME_TOTAL	NUMBER	这个查询服务器激活的时间总数
CPU_SECS_TOTAL	NUMBER	这个查询服务器用来处理语句的 CPU 时间总数
MSG_SENT_TOTAL	NUMBER	这个查询服务器发送的消息总数
MSG_RCVD_TOTAL	NUMBER	这个查询服务器接收到的消息总数

105.　V$PQ_SYSSTAT

这个查询列出并行查询的系统统计数据。

说明 此视图在未来的版本中将被一个称为 V$PX_PROCESS_SYSSTAT 的视图替代或废弃。

列	数据类型	说明
STATISTIC	VARCHAR2(30)	统计数据名，请参阅表 B-8
VALUE	NUMBER	统计数据的值

为这个视图定义了表 B-8 中的统计数据（固定行）。在执行一个查询或 DML 操作后，可利用从 V$PQ_SYSSTAT 得出的信息查看所利用的从进程数，以及其他系统信息。

表 B-8　STATISTIC 列中的统计数据名

统计数据（固定行）	说明
Server Busy	这个实例上当前忙的服务器数
Server Idle	这个实例上当前空闲的服务器数
Server Highwater	这个实例上参与>=1 等操作的活动服务器的数目
Server Sessions	这个实例上所有服务器中执行的操作总数
Server Started	这个实例上启动的服务器总数
Server Shutdown	这个实例上关闭的服务器总数
Server Cleaned Up	这个实例上由于进程死亡而清除的服务器总数
Queries Initiated	这个实例上启动的并行查询总数
DML Initiated	启动的并行 DML 操作的总数
DFO Trees	这个实例上执行的 DFO 树的总数
Local Msgs Sent	这个实例上发送的本地（实例内）消息总数
Distr Msgs Sent	这个实例上发送的远程（实例间）消息总数

106.　V$PQ_TOSTAT

这个视图包含并行执行操作的统计数据。这些统计数据在查询完成后搜索，并且只在会话期间保留。此视图显示利用每个并行执行服务器在执行数的每个步骤处理的行数。它可以帮助确定查询执行中的扭曲问题。

说明 此视图在以后的版本中将被重新命名为 V$PX_TQSTAT。

列	数据类型	说明
DFO_NUMBER	NUMBER	区分查询的数据流运算符（DFO）树号
TQ_ID	NUMBER	查询内的表队列 ID，它表示查询执行树中两个 DFO 节点间的连接
SERVER_TYPE	ARCHAR2(10)	表队列中的角色：生成者/使用者/管理员
NUM_ROWS	NUMBER	生产/使用的行数
BYTES	NUMBER	生产/使用的字节数
OPEN_TIME	NUMBER	表队列保持打开的时间（秒）
AVG_LATENCY	NUMBER	表队列保持打开的时间（毫秒）
WAITS	NUMBER	退出队列时遇到的等待次数
TIMEOUTS	NUMBER	等待一个消息时超时的次数
PROCESS	VARCHAR2(10)	进程 ID

INSTANCE	NUMBER	实例 ID

107. V$PQ_PROCESS

这个视图包含当前活动进程的有关信息。在 LATCHWAIT 列表示进程等待某个栓锁时，LATCHSPIN 表示进程正在某个栓锁上循环，Oracle 进程在某个栓锁上等待前将在其上循环。

列	数据类型	说明
ADDR	RAW(4)	进程状态对象的地址
PID	NUMBER	Oracle 进程标识符
SPID	VARCHAR2	操作系统进程标识符
USERNAME	VARCHAR2	操作系统进程用户名。来自网络上的任意双任务，用户在用户名上都附加有“ -T ”
SERIAL#	NUMBER	进程系列号
TERMINAL	VARCHAR2	操作系统终端标识符
PROGRAM	VARCHAR2	正在进行中的程序
BACKGROUND	VARCHAR2	后台进程为 1；普通进程为 NULL
LATCHWAIT	VARCHAR2	进程正在等待的栓锁地址；如果没有，为 NULL
LATCHSPIN	VARCHAR2	进程正在其上循环的栓锁地址；如果没有，为 NULL

108. V$PROXY_ARCHIVEDLOG

这个视图包含归档日志备份的说明，这是与一个称为 Prpxy Copy 的新功能一道采用的。每行表示一个归档日志的备份。

列	数据类型	说明
RECID	NUMBER	代理拷贝记录 ID
STAMP	NUMBER	代理拷贝记录时间戳
DEVICE_TYPE	VARCHAR2(17)	拷贝驻留的设置类型
HANDLE	VARCHAR2(513)	代理拷贝句柄标识存储的拷贝
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注释。这个值只是通知性质的；不需要存储
MEDIA	VARCHAR2(65)	拷贝驻留的介质名。这个值只是通知性质的；不需要存储
DELETE	VARCHAR2(3)	如果设置为 YES，指出拷贝被删除，否则设置为 NO
THREAD#	NUMBER	重做线程号
SEQUENCE#	NUMBER	重做日志序列号
RESETLOGS_ CHANGE	NUMBER	写入这个日志时的数据库重置日志更改号
RESETLOGS_ TIME	DATE	写入这个日志时的数据库重置日志时间
FIRST_CHANGE#	NUMBER	归档日志中的第一个更改号
FIRST_TIME	DATE	第一个更改的时间
NEXT_CHANGE#	NUMBER	下一个日志中的第一个更改

		号
NEXT_TIME	DATE	下一个更改的时间戳
BLOCKS	NUMBER	以块表示的归档日志尺寸
BLOCK_SIZE	NUMBER	重做日志块尺寸
COMPLETION_ TIME	DATE	完成时间
ELAPSED_ SECONDS	NUMBER	占用的秒数

109. V$PROXY_ DATAFILE

这个视图包含数据文件和控制文件的描述，这是与称为 Proxy Copy 的新功能一道采用的。每行表示一个数据库文件的备份。

列	数据类型	说明
RECID	NUMBER	代理拷贝记录 ID
STAMP	NUMBER	代理拷贝记录时间戳
DEVICE_TYPE	VARCHAR2(17)	拷贝驻留的设置类型
HANDLE	VARCHAR2(513)	代理拷贝句柄标识存储的拷贝
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注释。这个值只是通知性质的；不需要存储
MEDIA	VARCHAR2(65)	拷贝驻留的介质名。这个值只是通知性质的；不需要存储
MEDIA_POOL	NUMBER	拷贝驻留的介质池。它与 RecoveryManager；不需要存储
TAG	VARCHAR2(32)	代理拷贝标志
DELETE	VARCHAR2(3)	如果设置为 YES，指出拷贝被删除，否则设置为 NO
FILE#	NUMBER	绝对数据文件号，如果这是一个控制文件备份，则设置为 0
CREATION_ CHANGE#	NUMBER	数据文件创建更改号
CREATION_ TIME	DATE	数据文件创建时间戳
RESETLOGS_ CHANGE#	NUMBER	进行这个拷贝时的数据文件的重置日志更改号
RESETLOGS_ TIME	DATE	进行这个拷贝时的数据文件的重置日志时间戳
CHECKPOINT_ CHANGE#	NUMBER	进行拷贝时的数据文件检查点更改号
CHECKPOINT_ TIME#	DATE	进行拷贝时的数据文件检查点时间戳
ABSOLUTE_ FUZZY_CHANGE	NUMBER	如果知道的话，为这个文件的任意块中最高更改
RECOVERY_ FUZZY_CHANGE	NUMBER	介质恢复写到这个文件中的最高更改
RECOVERY_ FUZZY_CHANGE	DATE	介质恢复写到这个文件中的最高更改时间戳
INCREMENTAL_ LEVEL	NUMBER	如果这个备份为增量备份策略的组成部分，则为 0，否则为 NULL
ONLINE_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，则这个拷贝是在崩溃或脱机后立即做的（或

		者是在数据库打开后不正确地进行的拷贝的拷贝)。恢复需要应用下一个恢复标记前的所有重做以便使文件一致
BACKUP_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，则这是一个利用 BEGIN BACKUP/END BACKUP 技术进行的拷贝。注意：BEGIN BACKUP/END BACKUP 技术是在创建开放文件的代理备份时内部使用的。恢复需要应用 下一个恢复标记前的所有重做以便使这个备份文件一致
BLOCKS	NUMBER	以块表示的拷贝尺寸（也是进行拷贝时的数据文件尺寸）
BLOCK_SIZE	NUMBER	数据文件块尺寸
OLDEST_OFFLINE_RANGE	NUMBER	如果文件号为 0（即，这是一个控制文件备份)，则最旧的的脱机范围的 RECID 记录在这个控制文件拷贝中。数据文件拷贝为 0
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	完成时间
ELAPSED_SECONDS	NUMBER	占用的秒数

110. V$PWFILE_USERS

这个视图列出从口令文件中导出的授予 SYSDBA 和 SYSOPER 权限的用户。

列	数据类型	说明
USERNAME	VARCHAR2(30)	包含在口令文件中的用户名
SYSDBA	VARCHAR2(5)	如果此列的值为 TURE，则该用户可利用 SYSDBA 权限进行连接
SYSOPER	VARCHAR2(5)	如果此列的值为 TURE，则该用户可利用 SYSOPER 权限进行连接

111. V$PX_PROCESS

此视图包含进行并行执行的会话的相关信息。

列	数据类型	说明
SERVER_NAME	VARCHAR2(4)	并行服务器名（P000、P001 等）
STATUS	VARCHAR2(9)	并行服务器的状态。或者为 In Use 或者为 Available
PID	NUMBER	进程标识符
SPID	VARCHAR2(9)	操作系统进程 ID
SID	NUMBER	如果使用，为从服务器的会话 ID
SERIAL#	NUMBER	如果使用，为从服务器的会话系列号

112. V$PX_PROCESS_SYSSTAT

这个视图包含进行并行执行的会话的有关信息。

列	数据类型	说明
STATISTIC	VARCHAR2(30)	统计数据名
VALUE	NUMBER	统计数据的值

STATISTIC 列的值列在表 B-9 中。

表 B-9 STATISTIC 列中的统计数据名

统计数据（固定行）	说明
Servers In Use	当前执行并行操作的 PX 服务器号
Servers Available	执行并行操作可用的 PX 服务器号
Servers Started	系统必须创建 PX 服务器进程的次数
Server Shutdown	PX 服务器进程被关闭的次数。如果 PX 服务器进程最近未使用，将被关闭。它保持 Available 的时间长度由初始化参数 PARALLEL_SERVER_IDLE_TIME 控制。如果这个值较大，应该考虑增加该参数。由于免除了 PX 服务器进程的创建等待时间，所以将会提高性能
Server HWM	并发 PX 服务器进程的最大数目。如果这个数等于初始化参数 PARALLEL_MAX_SERVERS，应该考虑增加该参数。这样能够增加吞吐量，特别是如果系统利用不充分，并且 V$SYSSTAT 统计数据“ Parallel operations downgraded to serial ”较大时更是如此
Servers Cleaned Up	PMON 必须清除某个 PX 服务器的次数。它只应该在并行操作不正常结束时进行。如果这个数较大，建议查处其原因
Sessions	所有 PX 服务器建立的会话总数
Memory Chunks Allocs	PX 服务器分配的大内存块数
Memory Chunks Freed	空闲的大内存块数
Memory Chunks Current	当前使用的大内存块数
Memory Chunks HWM	当前分配内存块的最大数目
Buffers Allocated	某个消息缓冲区被分配的次数
Buffers Freed	某个消息缓冲区被释放的次数
Buffers Current	当前使用的消息缓冲区的数目
Buffers HWM	当前分配的消息缓冲区的最大数目

113. V$PX_SESSION

这个视图包含进行并行执行的会话的相关信息。

列	数据类型	说明
SADDR	NUMBER	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号
QCSID	NUMBER	并行协调程序的会话标识符
QCSERIAL#	NUMBER	并行协调程序的会话序列号
QCINST_ID	NUMBER	并行协调程序在其上运行的实例号

SERVER_GROUP	NUMBER	此并行服务器进程所属的服务器的逻辑组
SERVER_SET	NUMBER	此并行服务器进程所属的服务器逻辑组合。单个服务器组至少有两个服务器组合
SERVER#	NUMBER	某个并行服务器进程在一个服务器集合内的逻辑号
DEGREE	NUMBER	服务器集合所用的并行度
REQ_DEGREE	NUMBER	在发布语句且优先于任意资源、多用户或负载平衡降低时，用户所请求的并行度

114. V$PX_SESSTAT

这个视图包含进行并行执行的会话的有关信息。

列	数据类型	说明
SADDR	RAW(4)	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号
QCSID	NUMBER	并行协调程序的会话标识符
QCSERIAL#	NUMBER	并行协调程序的会话序列号
QCINST_ID	NUMBER	并行协调程序在其上运行的实例号
SERVER_GROUP	NUMBER	此并行服务器进程所属的服务器的逻辑组
SERVER_SET	NUMBER	此并行服务器进程所属的服务器逻辑组合。单个服务器组至少有两个服务器组合
SERVER#	NUMBER	某个并行服务器进程在一个服务器集合内的逻辑号
DEGREE	NUMBER	服务器集合所用的并行度
REQ_DEGREE	NUMBER	在发布语句且优先于任意资源、多用户或负载平衡降低时，用户所请求的并行度
STATISTIC#	NUMBER	统计数据号（标识符）
VALUE	NUMBER	统计数据值

115. V$QUEUE

这个视图包含多线程消息队列的相关信息。

列	数据类型	说明
PADDR	RAW(4)	拥有对列的进程地址
TYPE	VARCHAR2	队列的类型：COMMON(由服务器处理)、DISPATCHER
QUEUED	NUMBER	队列中的项数
WAIT	NUMBER	此队列中所有项等待的总时间。除以 TOTAL 得每项的平均等待时间
TOTALQ	NUMBER	曾经在队列中的项数

116. V$RECOVER_FILE

这个视图显示需要介质恢复的文件状态。

列	数据类型	说明
FILE#	NUMBER	文件标识号
ONLINE	VARCHAR2	联机状态：ONLINE、OFFLINE
ERROR	VARCHAR2	为什么此文件需要恢复：如果不知道原因，则为 NULL，或者如果不需要恢复，为 OFFLINE NORMAL
CHANGE#	NUMBER	恢复必须开始的 SCN
TIME	DATE	恢复必须开始的 SCN 时间

117. V$RECOVER_FILE_STATUS

这个视图对每个 RECOVER 命令的每个数据文件包含一行。此视图仅在 Oracle 进程进行恢复时包含有用信息。在 Recovery Manager 引导服务器进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
FILENUM	NUMBER	被恢复的文件数
FILENAME	VARCHAR2(257)	被恢复的数据文件名
STATUS	VARCHAR2(13)	恢复的状态。包含下列值：IN RECOVERY；CURRENT；NOT RECOVERED

118. V$RECOVERY_LOG

这个视图列出关于需要完成介质恢复的归档日志的信息。这些信息是从日志历史视图 V$LOG_HOSTORY 中导出的。更多的信息，请参阅“V$LOG_HISTORY”。

此视图仅在 Oracle 进程进行恢复时含有有用的信息。在 Recover Manager 引导服务器进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
THREAD	NUMBER	归档日志的线程号
SEQUENCE#	NUMBER	归档日志的序列号
TIME	VARCHAR2	日志中的第一项（最低 SCN）时间
ARCHIVE_NAME	VARCHAR2	在归档时，使用由 LOG_ARCHIVE_FORMAT 指定的命名约定的文件名

119. V$RECOVERY_PROGRESS

此视图可用来跟踪数据库恢复操作以保证它们不停止，并估计完成正在进行中的操作所需的时间。

此视图是 V$SESSION_LONGOPS 的子视图。

列	数据类型	说明

TYPE	VARCHAR2(64)	正执行的恢复操作的类型
ITEM	VARCHAR2(32)	正被估计的项
SOFAR	NUMBER	迄今为止所完成的工作量
TOTAL	NUMBER	预期的总工作量

120. V$RECOVERY_STATUS

这个视图包含当前恢复进程的统计数据。此视图仅在 Oracle 进程恢复时含有有用信息。在 Recovery Manager 引导服务进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
RECOVERY_ CHECKPOINT	DATE	恢复发生的时间点。如果没有应用日志，这是恢复开始的时间点
THREAD	NUMBER	当前正处理的重做日志线程号
SEQUENCE_ NEEDED	NUMBER	恢复进程所需的日志的序列号。如果不需要日志，这个值为 0
SCN_NEEDED	VARCHAR2(16)	恢复所需的日志的低 SCN。如果不知道或不需要日志，这个值为 0
TIME_EEEDED	DATE	创建日志时间。如果不知道时间或不需要日志，这个值为 88 年 1 月 1 日午夜
PREVIOUS_LOG_ NAME	VARCHAR2(257)	日志的文件名
PREVIOUS_LOG_ STATUS	VARCHAR2(13)	以前日志的状态。包含下列某个值：RELEASY；WRONE NAME；MISSING NAME；UNNEEDED NAME；NONE
REASON	VARCHAR(13)	合理的恢复将控制返回到用户。包含下列某个值：NEED LOG；LOG REUSED；THREAD DISABLED

121. V$REQDIST

这个视图列出 MTS 调度程序请求时间除以 12 个时间块或时间范围的柱状图统计数据。时间范围作为时间块的函数按指数增长。

列	数据类型	说明
BUCKET	NUMBER	时间块数：0-11；每个块的最大时间为（4*2^N）100 秒
COUNT	NUMBER	其完成总时间（包括等待时间）落入这个范围的请求计数

122. V$RESERVED_WORDS

这个视图给出 PL/SQL 编译程序使用的所有关键字列表。这个视图帮助开发者判断某个词是否已被用作语言中的关键字。

列	数据类型	说明
KEYWORD	VARCHAR2(64)	关键字名

LENGTH	NUMBER	关键字长度

123. V$RESOURCE

此视图包含资源的名称和地址的信息。

列	数据类型	说明
ADDR	RAW(4)	资源对象的地址
TYPE	NUMBER	资源类型。资源类型列在表 B-3 中
ID1	NUMBER	资源标识符#1
ID2	NUMBER	资源标识符#2

124. V$RESOURCE_LIMIT

这个视图显示某些系统资源的全局资源应用信息。可利用这个视图监控资源的使用情况，以便有必要时能够采取正确的措施。许多资源对应于表 B-10 所列出的初始化参数。

表 B-10 RESOURCE_LIMIT 列的值

资源名	相应的初始化参数
DISTRIBUTED_TRANSACTIONS	DISTRIBUTED_TRANSACTIONS：关于这个参数的更多信息，请参阅 DISTRIBUTED_ TRANSACTIONS
DML_LOCKS	DML_LOCKS：关于这个参数的更多信息，请参阅 DML_LOCKS
ENQUEUE_LOCKS	这个值是 Oracle 计算出来的。可使用 V$ENQUEUE_LOCK 视图获得关于排队锁的更多信息
ENQUEUE_RESOURCES	ENQUEUE_RESOURCES：关于这个参数的更多信息，请参阅 ENQUEUE_RESOURCES
LM_PROCESSES	LM_PROCS：关于这个参数的更多信息，请参阅 LM_PROCS
LM_RESOURCES	LM_RESS：关于这个参数的更多信息，请参阅 LM_RESS
LM_LOCKS	LM_LOCKS：关于这个参数的更多信息，请参阅 LM_LOCKS
MTS_MAX_SERVERS	MTS_MAX_SERVERS：关于这个参数的更多信息，请参阅 MTS_MAX_SERVERS
PARALLEL_SLAVES	PARALLEL_SLAVES：关于这个参数的更多信息，请参阅 PARALLEL_SLAVES
PROCESSES	PROCESSES：关于这个参数的更多信息，请参阅 PROCESSES
ROLLBACK_SEGMENTS	MAX_ ROLLBACK_SEGMENTS：关于这个参数的更多信息，请参阅 MAX_ ROLLBACK_ SEGMENTS
SESSIONS	SESSIONS：关于这个参数的更多信息，请参阅 SESSIONTS
SORT_SEGMENT_LOCKS	这个值由 Oracle 计算
TEMPORARY_LOCKS	这个值由 Oracle 计算
TRANSACTIONS	TRANSACTIONS：关于这个参数的更多信息，请参阅 TRANSACTIONS

某些资源（如，那些 DML 所用的）有一个初始分配（软限制）和一个硬限制，从理论上说它是无限的（但实际上要受 SGA 尺寸的限制）。在 SGA 预定/初始化中，在 SGA 中为资源的 INITIAL_ALLOCATION 保留一个位置，但如果超出这个分配，可分配额外的资源，最多可达 LIMIT_VALUE 指定的值。CURRENT_UTILIZATION 列指出的是否已经超过初始分配。在超出初始分配值时，从共享池中分配额外需要的资源，在那里它们必须与其他资源竞争空间。

恰当选择 INITIAL_ALLOCATION 的值将避免空间争用。对于大多数资源，INITIAL_ALLOCATION 和 LIMIT_VALUE 的值相同。超过 LIMIT_VALUE 的值将出现错误。

列	数据类型	说明
RESOURCE_NAME	VARCHAR2(30)	资源名（见表 B-10）
CURRENT_@@@@@	NUMBER	当前在用的数目（资源、锁或进程）
UTILIZATION	NUMBER	资源类型。资源类型列在表 B-3 中
MAX_UTILIZATION	NUMBER	自最后一个实例启动以来这个资源的最大消耗
INITIAL_ ALLOCATION	VARCHAR2(10)	初始分配。这个值将等于初始化参数文件中为此资源指定的值。（无限制分配为 UNLIMITED）
LIMIT_VALUE	VARCHAR2(10)	对资源和锁无限制。它可大于初始分配值。（无限制为 UNLIMITED）

125. V$ROLLNAME

这个视图列出所有联机回退段的名称。它只有在数据库打开时访问。

列	数据类型	说明
USN	NUMBER	回退（撤消）段号
NAME	VARCHAR2	回退段名

126. V$ROLLSTAT

这个视图包含回退段统计数据。

列	数据类型	说明
USN	NUMBER	回退段号
EXTENTS	NUMBER	回退段中的区数
RSSIZE	NUMBER	回退段以字节极的尺寸
WRITES	NUMBER	写到回退段的字节数
XACTS	NUMBER	活动的事务处理数
GETS	NUMBER	标题获得的数目
WAITS	NUMBER	标题等待的数目
OPTSIZE	NUMBER	回退段的最佳尺寸
HWMSIZE	NUMBER	回退段尺寸的高水位标记
SHRINKS	NUMBER	回退段尺寸减少的倍数
WRAPS	NUMBER	回退段缠绕的倍数
EXTENDS	NUMBER	回退段段尺寸扩展的倍数
AVESHRINK	NUMBER	平均收缩尺寸
AVEACTIVE	NUMBER	活动区随时间平均的当前尺寸
STATUS	VARCHAR2(15)	回退段状态

CUREXT	NUMBER	当前区
CURBLK	NUMBER	当前块

127. V$ROWCACHE

这个视图显示数据字典活动的统计数据。每行包含一个数据字典高速缓存的统计数据。

列	数据类型	说明
CACHE#	NUMBER	行高速缓存 ID 号
TYPE	VARCHAR2	父级或子级行高速缓存类型
SUBORDINATE#	NUMBER	子级集合号
PARAMETER	NUMBER	确定数据字典高速缓存中项数的初始化参数名
COUNT	NUMBER	高速缓存中项的总数
USAGE	NUMBER	包含有效数据的高速缓存项数
FIXED	NUMBER	高速缓存中的固定项数
GETS	NUMBER	请示数据对象信息的总数
GETMISSES	NUMBER	导致高速缓存未中的数据请求数
SCANS	NUMBER	扫描请求数
SCANMISSES	NUMBER	查找高速缓存中的数据扫描失败的次数
SCANCOMPLETES	NUMBER	对于子级项的列表，完全扫描列表的次数
MODIFICATIONS	NUMBER	插入、更新与删除的次数
FLUSHES	NUMBER	对磁盘进行刷新的次数
DLM_REQUESTS	NUMBER	DLM 请求的次数
DLM_CONFILICTS	NUMBER	DLM 冲突的次数
DLM_RELEASES	NUMBER	DLM 释放的次数

128. V$ROWCACHE_PARENT

这个视图显示数据字典中父级对象的信息。每个锁拥有一行，每个对象的等待者有一行。这个行显示拥有或请求的模式。对于没有拥有者的或等待者的对象，只显示单行。

列	数据类型	说明
INDX	NUMBER	行索引
HASH	NUMBER	散列值
ADDRESS	RAW(4)	父级对象的地址
CACHE#	NUMBER	父级高速缓存 ID
CACHE_NAME	VARCHAR2(64)	父级高速缓存名
EXISTENT	VARCHAR2(1)	此对象是否一个现存的对象
LOCK_MODE	NUMBER	锁被占有的模式
LOCK_REQUEST	NUMBER	锁被请求的模式
TXN	RAW(4)	当前缩定对象的事物处理
SADDR	RAW(4)	会话的地址
INST_LOCK_ REQUEST	NUMBER	只与并行服务器有关。实例锁被请求模式
INST_LOCK_ RELEASE	NUMBER	只与并行服务器有关。是否需要释放实例锁
INST_LOCK_TYPE	VARCHAR2	只与并行服务器有关。实例锁的类型

INST_LOCK_ID1	RAW(4)	只与并行服务器有关。与实例有关的 ID
INST_LOCK_ID2	RAW(4)	只与并行服务器有关。与实例有关的 ID
KEY	RAW(100)	只与并行服务器有关。键的内容

129. V$ROWCACHE_SUBORDINATE

这个视图显示数据字典中子级对象的信息。

列	数据类型	说明
INDX	NUMBER	索引
HASH	NUMBER	散列值
ADDRESS	RAW(4)	子级对象的地址
CACHE#	NUMBER	父级高速缓存 ID
SUBCACHE_NAME	VARCHAR2(64)	子级高速缓存名
EXISTENT	VARCHAR2(1)	此对象是否一个现存的对象
PARENT	RAW(4)	父级对象的地址
KEY	RAW(100)	只与并行服务器有关。键的内容

130. V$RSRC_CONSUMER_GROUP

这个视图显示与当前活动资源消耗者组有关的数据。

列	数据类型	说明
NAME	VARCHAR2(32)	使用者组名
ACTIVE_SESSIONS	NUMBER	这个使用者组中当前活动会话数
EXECUTION_ WAITERS	NUMBER	等待执行限额的当前活动会话数
REQUESTS	NUMBER	这个使用者组中执行的请求总数
CUP_WAIT_TIME	NUMBER	会话等待 CPU 的时间总数
CPU_WAITS	NUMBER	这个使用者组中所有会话必须等待 CPU 的时间数
CONSUMED_CPU_ TIME	NUMBER	这个使用者组中所有会话使用的 CPU 的时间总数
YIELDS	NUMBER	这个使用者组中所有会话必须的 CPU 时间总数
SESSIONS_QUEUED	NUMBER	等待变成活动的当前排队的会话计数

131. V$RSRC_CONSUMER_GROUP_CPU_MTH

这个视图显示资源使用者组的所有可用资源分配方法。

列	数据类型	说明
NAME	VARCHAR2(40)	CPU 资源分配方法的名称

132. V$RSRC_PLAN

这个视图显示所有当前活动资源计划的名称。

列	数据类型	说明

NAME	VARCHAR2(32)	资源计划的名称

133. V$RSRC_PLAN_CPU_MTH

这个视图显示资源计划的所有可用 CPU 资源分配方法。

列	数据类型	说明
NAME	VARCHAR2(32)	资源分配方法的名称

134. V$SESSION

这个视图列出每个当前会话的会话信息

列	数据类型	说明
SADDR	RAW(4)	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号。用来唯一地标识绘画对象。如果该会话结束且其他会话以相同的会话 ID 开始，则保证会话级的命令被应用到正确会话对象
AUDSID	NUMBER	审计会话 ID
PADDR	RAW(4)	拥有这个会话的进程地址
USER#	NUMBER	Oracle 用户标识符
USERNAME	VARCHAR(30)	Oracle 用户名
COMMAND	NUMBER	正进行的命令（分析的最后一个语句），关于值的列表，请参阅表 B-11
OWNERID	NUMBER	如果值为 2147483644，则此列的内容无效。否则此列包含拥有可移植会话的用户标符。对于利用并行从服务器的操作，将这个值解释为一个 48 字节的值。其低位两字节表示会话号，而高位字节表示查询协调程序的实例 ID
TADDR	VARCHAR2(8)	事务处理状态对象的地址
LOCKWAIT	VARCHAR2(8)	等待锁的地址；如果没有，为 NULL
STATUS	VARCHAR2(8)	会话的状态：ACTIVE（当前执行的 SQL）、INACTIVE、KILLED(标记为终止)、CACHED（为 Oracle*XA 使用而临时高速缓存）、SNIPED（会话不活动，在客户机上等待）
SERVER	VARCHAR2(9)	服务器类型：DEDICATED、SHARED、PSEUDO、NONE
SCHEMA#	NUMBER	模式用户标识符
SCHEMANANME	VARCHAR2(30)	模式用户名
OSUSER	VARCHAR(15)	操作系统客户机用户名
PROCESS	VARCHAR2(9)	操作系统客户机进程 ID
MACHINE	VARCHAR2(64)	操作系统机器名
TERMINAL	VARCHAR2(10)	操作系统终端名
PROGRAM	VARCHAR(48)	操作系统程序名

TYPE	VARCHAR2(10)	会话类型
SQL_ADDRESS	RAW(4)	与 SQL_HASH_VALUE 一道使用标识当前正在执行的 SQL 语句
SQL_HASH_VALUE	NUMBER	与 SQL_ADDRESS 一道使用标识当前正在执行的 SQL 语句
MODULE	VARCHAR2(48)	包含当前正在执行的模块名，正如由调用 DBMS_APPLICATION_INFO.SET_MODULE 过程所设置
MODULE_HASH	NUMBER	上面 MODULE 的散列值
ACTION	VARCHAR2(32)	包含当前执行活动的名称，正如由调用 DBMS_APPLICATION_INFO.SET_ACTION 过程所设置
ACTION_HASH	NUMBER	上列活动名称的散列值
CLIENT_INFO	VARCHAR2(64)	由 DBMS_APPLICATION_INFO.SET_CLIENT_INFO 过程设置的信息
FIXED_TABLE_ SEQUENCE	NUMBER	此列包含一个数，每当会话完成一个数据库调用并且存在来自动态性能表的介入选择，它个数就增加。这个列可被性能监控程序用来监控数据库中的统计数据。每当性能监控程序查看数据库时，只需要查看当前活动的会话或在这个列中具有比上次性能监控程序所看到的最大值更大的值的会话即可。所有其他会话自上次性能监控程序查看数据库以来都是空闲的
ROW_WAIT_OBJ#	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROW# 的表的对象 ID
ROW_WAIT_FILE#	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROWID 的数据文件的标识符。此列仅在会话当前正在等待其他事务处理提交并且 ROW_ WAIT_OBJ#不为-1 时有效
ROW_WAIT_BLOCK #	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROWID 的数据文件的标识符。此列仅在会话当前正在等待其他事务处理提交并且 ROW_正在等待其他事务处理提交并且 ROW_
ROW_WAIT_ROW#	NUMBER	被锁定的当前 ROWID。此列仅在会话当前正在等待其他事务处理提交并且 ROW_ WAIT_OBJ#不为-1 时有效
LOGON_TIME	DATE	登录时间
LAST_CALL_ET	NUMBER	最后一次调用
PDML_STATUS	VARCHAR2(8)	如果 ENABLED，则会话正处于 PARALLEL DML 启用方式。如果 DISABLED，则此会话不支持 PARALLEL DML 启用方式。如果

		FORCED，则会话已经更改为强制 PARALLEL DLL
PDML_ENABLED	VARCHAR2(3)	此列已被 PDML_ENABLED 和 PDML_STATUS 所替代。请看上列内容
FAILOVER_TYPE	VARCHAR2(10)	如果这个会话禁止失败切换，则为 NONE ，如果客户机能够在断开之后失败切换其会话，则为 SESSION，如果客户机还能失败切换正在进行的选择，则为 SELECT
FAILOVER_METHOD	VARXHAR2(3)	如果这个会话禁止失败切换，则为 NONE，如果客户机能够在断开之后重新连接，为 BASIC，如果备份实例能够支持它所支持的每个实例的所有连接，则为 PRECONNECT
FAILED_OVER	VARCHAR2(13)	如果运行在失败切换方式并进行过失败切换，为 TRUE，否则为 FALSE
RESOURCE_CONSU_MER_GROUP	VARCHAR2(32)	会话的当前资源使用者组的名称

表 B-11 列出对应于会话中正在执行的命令的数字值。这些值可出现在 V$AESSION COMMAND 列中。它们还出现在数据字典视图 SYS.AUDIT_ACTIONS 中。

表 B-11 COMMAND 列的命令数字

命令数字	命令
0	正在执行的命令。在进程处于过渡状态时出现
1	CREATE TABLE
2	INSERT
3	SELECT
4	CREATE CLUSTER
5	ALTER CLUSTER
6	UPDATE
7	DELETE
8	DROP CLUSTER
9	CREATE INDEX
10	DROP INDEX
11	ALTER INDEX
12	DROP TABLE
13	CREATE SEQUENCE
14	ALTER SEQUENCE
15	ALTER TABLE
16	DROP SEQUENCE
17	GRANT
18	REVOKE
19	CREATE SYNONYM
20	DROP SYNONYM
21	CREATE VIEW

22	DROP VIEW
23	VALIDATE INDEX
24	CREATE PROCEDURE
25	ALTER PROCEDURE
26	LOCK TABLE
27	NO OPERATION
28	RENAME
29	COMMIT
30	AUDIT
31	NOAUDIT
32	CREATE DATABASE LINK
33	DROP DATABASE LINK
34	CREATE DATABASE
35	ALTER DATABASE
36	CREATE ROLLBACK SEGMENT
37	ALTER ROLLBACK SEGMENT
38	DROP ROLLBACK SEGMENT
39	CREATE TABLESPACE
40	ALTER TABLESPACE
41	DROP TABLESPACE
42	ALTER SESSION
43	ALTER USER
44	COMMIT
45	ROLLBACK
46	SAVEPOINT
47	PL/SQL EXECUTE
48	SET TRANSACTION
49	ALTER SYSTEM SWITCH LOG
50	EXPLAIN
51	CREATE USER
52	CREATE ROLE
53	DROP USER
54	DROP ROLE
55	SET ROLE
56	CREATE SCHEMA
57	CREATE CONTROL FILE
58	ALTER TRACTING
59	CREATE TRIGGER
60	ALTER TRIGGER
61	DROP TRIGGER
62	ANALYZE TABLE
63	ANALYZE INDEX
64	ANALYZE CLUSTER
65	CREATE PROFILE

66	DROP PROFILE
67	ALTER PROFILE
68	DROP PROCEDURE
69	DROP PROCRDURE
70	ALTER RESOURCE COST
71	CREATE SNAPSHOT LOG
72	ALTER SNAPSHOT
73	DROP SNAPHOST
74	CREATE SNAPHOST
75	ALTER SNAPSHOT
76	DROP SNAPSHOT
79	ALTER ROLE
85	TRUNCATE TABLE
86	TRUNCATE CLUSTE
88	ALTER VIEW
91	CREATE FUNCTION
92	ALTER FUNCTION
93	DROP FUNCTION
94	CREATE PACKAGE
95	ALTER PACKAGE
96	DROP PACKAGE
97	CREATE PACKAGE BODY
98	ALTER PACKAGE BODY
99	DROP PACKAGE BODY

135. V$SESSION_CONNECT_INFO

这个视图显示当前会话的网络连接的有关信息。

列	数据类型	说明
SID	NUMBER	会话标识符（可用来将这个视图与 V$SESSON 视图进行连接）
AUTHENTICATION_TYPE	VARCHAR2(15)	怎样验证：OS、PROTOCOL 或 NETWORK
OSUSER	VARCHAR2(30)	这个数据库用户的外部用户名
NETWORK_SERVICE_BANNER	VARCHAR2(2000)	用于这个连接的每个 Net8 服务的产品标识（每个标识一行）

136. V$SESSION_CURSOR_CACHE

这个视图显示关于当前会话的游标使用信息。

说明 V$SESSION_CURSOR_CACHE 视图不是 SESSION_CACHED_CURSORS 初始化参数的有效性度量。

列	数据类型	说明
MAXIMUM	NUMBER	高速缓存的游标最大数。一旦达到这个数，则为了打开更多的游标，

		将需要关闭某些游标。这个列中的值是从初始化参数 OPEN_ CURSORS 导出的
COUNT	NUMBER	游标的当前数目（不管是否在用）
OPENED_ONCE	NUMBER	至少打开过一次游标
OPEN	NUMBER	打开游标的当前数目
OPENS	NUMBER	游标打开的累积总数减一。这是因为本查询 当前打开且正在使用的游标并为计入 OPENS 统计数据中
HITS	NUMBER	游标打开命中的累计总数
HIT_RATIO	NUMBER	找到打开游标的次数除以查找游标的次数所得到的比例

137. V$SESSION_EVENT

此视图列出会话等待某个事件的信息。注意，TIME_WAITED 和 AVERAGE_WAIT 列在不支持快速时间机制的平台。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 TIMED_STATISTICS 为真。请注意，这样做对系统性能有轻微的负面影响。更多的信息，请参阅 TIMED_STATISTICS。

列	数据类型	说明
SID	NUMBER	会话的 ID
EVENT	VARCHAR2(64)	等待事件的名称
TOTAL_WAITS	NUMBER	此会话对这个事件的总等待次数
TOTAL_TIMEOUTS	NUMBER	此会话对这个事件的总等待超时次数
TIME_WAITED	NUMBER	此会话对这个事件的总等待时间数，以百分之一秒计
AVERAGE_WAIT	NUMBER	此会话对这个事件的平均等待时间数，以百分之一秒计
MAX_WAIT	NUMBER	此会话对这个事件的最大等待时间数，以百分之一秒计

138. V$SESSION_LONGOPS

这个视图显示某些长运操作的状态。它利用列 SOFAR 和 TOTALWORK 提供操作的进展报告。可监控下列操作状态：

- 散列值的创建
- 备份操作
- 恢复操作

列	数据类型	说明
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话系列号
OPENNAME	VARCHAR2(64)	操作名
TARGET	VARCHAR2(64)	在其上进行操作的对象
TARGET_DESC	VARCHAR2(32)	目标描述
SOFAR	NUMBER	至今为止完成的工作计数

TOTALWORK	NUMBER	总的工作量
UNITS	VARCHAR2(32)	工作度量单位
START_TIME	DATE	操作开始的时间
LAST_UPDATE_TIME	DATE	统计数据最后一次更新的时间
ELAPSED_SECOUNDS	NUMBER	操作开始以来占用的秒数
CONTEXT	NUMBER	前后关系
MESSAGE	VARCHAR2(512)	统计数据摘要消息

139. V$SESSION_OBJECT_CACHE

这个视图显示本地服务器（实例）上当前用户会话的对象高速缓存统计数据。

列	数据类型	说明
PINS	NUMBER	高速缓存中固定或查找出的对象数
HITS	NUMBER	发现对象已经在高速缓存中的对象固定的数目
TRUE_HITS	NUMBER	发现对象已经在高速缓存中并处于所需状态（从而不需要从数据库调入）的对象的数目
HIT_RATIO	NUMBER	HITS/PINS 的比例
TRUE_HIT_RATIO	NUMBER	TRUS_HITS/PINS 的比例
OBJECT_REFRESHES	NUMBER	高速缓存中利用来自数据库的新值进行刷新的对象数
CACHE_REFRESHES	NUMBER	整个高速缓存（所有对象）进行刷新的次数
OBJECT_FLUSHES	NUMBER	高速缓存中对数据库进行刷新的对象数
CACHE_FLUSHES	NUMBER	整个高速缓存（所有对象）对数据库进行刷新的次数
CACHE_SHRINKS	NUMBER	高速缓存收缩到最佳尺寸的次数
CACHE_OBJECTS	NUMBER	当前高速缓存的对象数
PINNED_OBJECTS	NUMBER	当前固定的对象数
CACHE_SIZE	NUMBER	以字节表示的高速缓存当前尺寸
OPTIMAL_SIZE	NUMBER	以字节表示的高速缓存最佳尺寸
MAXIMUM_SIZE	NUMBER	以字节表示的高速缓存最大尺寸

140. V$SESSION_WAIT

这个视图列出活动的会话等待的资源或事件。下面是优化时的考虑：

- 除显示的数值为十六进制以外，PIRAW，P2RAW，P3RAW 显示的值与 P1、P2、P3 列的值相同。
- WAIT_TIME 列在不支持快速时间机制的平台上包含值-2。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 他—STATISTICS 为真。请注意，这样做对系统性能有轻微的负面影响。更过的信息，请参阅 TIMED_STATISTICS.

在以前的版本中，WAIT_TIME 列包含一个任意大的值（而不是一个负值）以表示此平台不具备快速时间机制。

- STATE 列解释 WAIT_TIME 的值并描述当前或最近的等待状态。

列	数据类型	说明
SID	NUMBER	会话标识符
SEQ#	NUMBER	唯一标识这个等待的序列号。每次等待都增加
EVENT	VARCHAR2(64)	会话等待的资源或会话
P1TEXT	VARCHAR2	第一个附加参数的描述
P1	NUMBER	第一个附加参数
P1RAW	RAW(4)	第一个附加参数
P2TEXT	VARCHAR2	第二个附加参数的描述
P2	NUMBER	第二个附加参数
P2RAW	RAW(4)	第二个附加参数
P3TEXT	VARCHAR2	第三个附加参数的描述
P3	NUMBER	第三个附加参数
P3RAW	RAW(4)	第三个附加参数
WAIT_TIME	NUMBER	非零值为会话的上一次等待时间。零值表示会话当前正在等待
SECONDS_IN_WAIT	NUMBER	等待的秒数
STATE	VARCHAR2	等待的状态

表 B-12 定义了 V$SESSION_WAIT 的 STATE 列的值。

STATE	值	说明
WAITING	0	当前正在等待的会话
WAITED UNKNOWN TIME	-2	未知的最后一次等待持续时间
WATIED SHORT TIME	-1	最后一次等待<百分之一秒
WAITED KNOWN TIME	>0	WAIT_TIME=最后一次等待持续时间

141. V$SESSTAT

这个视图给出用户会话的统计数据。为了找到与每个统计数据号（STATISTIC#）有关的统计数据名称，请参阅 V$STATNAME。

列	数据类型	说明
SID	NUMBER	会话标识符
STATISTIC#	NUMBER	统计数据名（标识符）
VALUE	NUMBER	统计数据值

142. V$SESS_IO

这个视图列出每个用户会话的 I/O 统计数据

列	数据类型	说明
SID	NUMBER	会话标识符
BLOCK_GETS	NUMBER	这个会话的块存取
CONSISTENT_GETS	NUMBER	此会话的一致性读取
PHYSICAL_READS	NUMBER	此会话的物理读取
BLOCK_CHANGES	NUMBER	此会话的块更改
CONSISTENT_ CHANGES	NUMBER	此会话的一致性更改

143. V$SGA

这个视图包含系统全局区的摘要信息。

列	数据类型	说明
NAME	VARCHAR2	SGA 组件名
VALUE	NUMBER	以字节表示的内存尺寸

144. V$SGASTAT

这个视图包含系统全局区的详细信息

列	数据类型	说明
NAME	VARCHAR2	SGA 组件名
BYTES	NUMBER	以字节表示的内存尺寸
POOL	VARCHAR2	指出 NAME 中内存驻留的池子。其值可以是：LARGE POO——从大型池中分配的内存 SHARED POOL—从共享池中分配的内存

145. V$SHARED_POOL_RESERVED

这个固定视图列出有助于优化共享存储池中保留池和空间的统计数据。

V$SHARED_POOL_RESERVED 视图的以下列仅在初始化参数 SHARED_POOL_ RESERVED_SIZE 设置为有效值时有效。更多的信息，请参阅 SHARED_POOL_ RESERVED _SIZE。

列	数据类型	说明
FREE_SPACE	NUMBER	保留列表中可用的空间总数
AVG_FREE_SIZE	NUMBER	保留列表上可用内存的平均尺寸
FREE_COUNT	NUMBER	保留列表上可用的内存片数量
MAX_FREE_SIZE	NUMBER	保留列表上最大可用内存片的尺寸
USED_SPACE	NUMBER	保留列表上使用的内存的总数
AVG_USED_SIZE	NUMBER	保留列表上使用的内存的平均尺寸
USED_COUNT	NUMBER	保留列表上使用的内存片的数量
MAX_USED_SIZE	NUMBER	保留列表上最大使用内存片的尺寸
REQUESTS	NUMBER	搜索保留列表查找可用内存片的次数
REQUESTS_MISSED	NUMBER	保留列表没有满足请求的可用内存片，从而开始利用 LRU 列表刷新对象的次数
LAST_MISS_SIZE	NUMBER	在保留列表没有满足请求的可用内存片，从而开始利用 LRU 列表刷新对象的次数
MAX_MISS_SIZE	NUMBER	在保留列表没有满足请求的可用内存片，从而开始利用 LRU 列表刷新对象时的最大请求未命中的

	请求尺寸

V$SHARED_POOL_RESERVED 视图的以下列包含的值在即使不设置初始化参数 SHARED_POOL_ RESERVED_SIZE 时也有效。

列	数据类型	说明
REQUEST_FAILURES	NUMBER	未找到满足请求的内存次数（即，出现 ORA_4031 错误的次数）
LAST_FAILURE_SIZE	NUMBER	最后失败请求的请求尺寸（即，出现 ORA_4031 错误的请求尺寸）
ABORTED_REQUEST _THRESHOLD	NUMBER	通知出现 ORA-4031 错误的的请求的最大尺寸
ABORTED_REQUEST	NUMBER	通知出现 ORA-4031 错误而不刷新对象的请求的数目
LAST_ABORTED_SIZE	NUMBER	返回一个 ORA-4031 错误而不利用 LRU 列表刷新对象的最后请求尺寸

146. V$SHARED_SERVER

这个视图包含共享服务器进程的有关信息。

列	数据类型	说明
NAME	VARCHAR2	服务器名
PADDR	RAW(4)	服务器的进程地址
STATUS	VARCHAR2	服务器状态： EXEC(执行 SQL) WAIT (ENQ)(等待一个锁) WAIT(SEND)(等待发送数据到用户) WAIT(COMMON)(空闲；等待用户请求) WAIT(RESET)(等待中断后重新设置的虚电路) QUIT(终止)
MESSAGES	NUMBER	处理的消息数
BYTES	NUMBER	所有消息中的总字节数
BREAKS	NUMBER	中断数
CIRCUIT	RAW(4)	当前正服务的虚电路地址
IDLE	NUMBER	以百分之一秒表示的空闲总时间
BUSY	NUMBER	以百分之一秒表示的繁忙总时间
REQUESTS	NUMBER	在此服务器的生存时间内从公用队列中取现的请求总数@@@@@

147. V$SORT_SEGMENT

这个视图包含一个给定实例中每个排序段的有关信息。此视图仅在表空间为 TEMPORARY 类型时更新。

列	数据类型	说明
TABLESPACE_NAME	VARCHAR2(31)	表空间名

SEGMENT_FILE	NUMBER	第一个区的文件号
SEGMENT_BLOCK	NUMBER	第一个区的块号
EXTENT_SIZE	NUMBER	区尺寸
CURRENT_USERS	NUMBER	段的活动用户号
TOTAL_EXTENTS	NUMBER	段中区的总数
TOTAL_BLOCKS	NUMBER	段中块的总数
RELATIVE_FNO	NUMBER	排序段标题的相对文件号
USED_EXTENTS	NUMBER	分配给活动排序的区
USED_BLOCKS	NUMBER	分配给活动排序的块
FREE_EXTENTS	NUMBER	未分配给任意排序的区
FREE_BLOCKS	NUMBER	未分配给任意排序的块
ADDED_EXTENTS	NUMBER	区分配的数目
EXTENT_HITS	NUMBER	在池中找到未用区的次数
FREED_EXTENTS	NUMBER	解除分配的区的数目
FREE_REQUESTS	NUMBER	请求解除分配的数目
MAX_SIZE	NUMBER	曾经使用过的区的最大数目
MAX_BLOCKS	NUMBER	曾经使用过的块的最大数目
MAX_USED_SIZE	NUMBER	所有排序使用过的区的最大数目
MAX_USED_BLOCKS	NUMBER	所有排序使用过的块的最大数目
MAX_SORT_SIZE	NUMBER	单个排序使用过的区的最大数目
MAX_SORT_BLOCKS	NUMBER	单个排序使用过的块的最大数目

148. V$SORT_USAGE

这个视图描述排序用法。

列	数据类型	说明
USER	VARCHAR2(30)	请求临时空间的用户
SESSION_ADDR	RAW(4)	共享 SQL 游标的地址
SESSION_NUM	NUMBER	会话的系列号
SQLADDR	RAW(4)	SQL 语句的地址
SQLHASH	NUMBER	SQL 语句的散列值
TABLESPACE	VARCHAR2(31)	在其中分配空间的表空间
CONTENTS	VARCHAR2(9)	指出表空间是否是 TEMPORARY/ PERMANENT
SEGFILE#	NUMBER	初始区的文件号
SEGBLK#	NUMBER	初始区的块号
EXTENTS	NUMBER	分配给排序的区
BLOCKS	NUMBER	分配给排序的以块表示的区
SEGFNO	NUMBER	初始区的相对文件号

149. V$SQL

这个视图列出没有 GROUP BY 子句的共享 SQL 区的有关统计数据，而且对录入的原始 SQL 文本的每个孩子包含一行。

列	数据类型	说明

SQL_TEXT	VARCHAR2(1000)	当前游标的 SQL 文本的前 8 位字符
SHARABLE_MEM	NUMBER	这个子级游标使用的以字节表示的共享内存量
PERSISTENT_MEM	NUMBER	这个子级游标使用的以字节表示的持久内存量
RUNTIME_MEM	NUMBER	这个子级游标使用的临时结构尺寸
SORTS	NUMBER	为这个子级游标完成的排序数
LOADED_VERSIONS	NUMBER	如果装载了上下文堆栈，为 1，否则为 0
OPEN_VERSIONS	NUMBER	如果锁定了子级游标，为 1，否则为 0
USERS_OPENING	NUMBER	执行相应语句的用户数目
EXECUTIONS	NUMBER	自这个对象装入库高速缓存以来，在这个对象上的执行数目
USERS_EXECUTING	NUMBER	执行这个语句的用户数目
LOADS	NUMBER	对象被装入或重新装入的数目
FIRST_LOAD_TIME	VARCHAR2(19)	父级创建时间的时间戳
INVALIDATIONS	NUMBER	使子级游标无效的次数
PARSE_CALLS	NUMBER	这个子级游标的分析调用数目
DISK_READS	NUMBER	这个子级游标的磁盘读取数目
BUFFER_GETS	NUMBER	这个子级游标的缓冲区获取数目
ROWS_PROCESSED	NUMBER	分析 SQL 语句返回的总行数
COMMAND_TYPE	NUMBER	Oracle 命令类型定义
OPTIMIZER_MODE	VARCHAR2(10)	SQL 语句在其下执行的模式
OPTIMIZER_COST	NUMBER	优化程序给出这个查询的代价
PARSING_USER_ID	NUMBER	最初建立这个子游标的用户的用户 ID
PARSING_SCHMA_ID	NUMBER	最初用来建立这个子级游标的模式 ID
KEPT_VERSIONS	NUMBER	指出这个子级游标是否已经利用 DBMS_SHARED_POOL 程序包标记为固定在高速缓存中
ADDRESS	RAW(4)	这个子级游标的双亲的句柄地址
TYPE_CHK_HEAP	RAW(4)	这个子级游标的类型描述符的检查堆栈
HASH_VALUE	NUMBER	库高速缓存中的父级语句的散列值
CHILD_NUMBER	NUMBER	这个子级游标的编号
MODULE	VARCHAR2(64)	包含第一次分析 SQL 语句执行时的模块名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
MODEL_HASH	NUMBER	在 MODULE 列中指定的模块的散列值
ACTION	VARCHAR2(64)	包含第一次分析 SQL 语句时执行的动作名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
ACTION_HASH	NUMBER	在 ACTION 列中指定的动作的散列值
SERIALIZABLE_ABORT	NUMBER	每个游标的串行化事务处理失败，产生 ORA-8177 错误的次数

150. V$SQL_BIND_DATA

这个视图显示客户机为每个游标中每个明确绑定变量发送的绑定数据，前提是服务器中可得到这个数据，其中游标为查询这个视图的会话所有。

列	数据类型	说明
CURSOR_NUM	NUMBER	这个绑定的游标数
POSITION	NUMBER	绑定位置
POSITION	NUMBER	绑定位置
DATATYPE	NUMBER	绑定数据类型
SHARED_MAX_LEN	NUMBER	来自与这个绑定有关的共享游标对象的共享最大长度
PRIVATE_MAX_LEN	NUMBER	从客户机发送的这个绑定的私有最大长度
ARRAY_SIZE	NUMBER	数组元素的最大数目（仅对数组绑定）
PRECISION	NUMBER	精度（对于数值绑定）
SCALE	NUMBER	小数点位置（对于数值绑定）
SHARED_FLAG	NUMBER	共享绑定数据标志
SHARED_FLAG2	NUMBER	共享绑定数据标志(续)
BUF_ADDRESS	RAW(4)	绑定缓冲区内存地址
BUF_LENGTH	NUMBER	绑定缓冲区长度
VAL_LENGTH	NUMBER	实际的绑定值长度
BUF_FLAG	NUMBER	绑定缓冲区标志
INDICATOR	NUMBER	绑定指示器
VALUE	VARCHAR2(4000)	绑定缓冲区的内容

151. V$SQL_BIND_METADATA

这个视图显示客户机为每个游标中每个明确绑定变量提供的绑定元数据，其中的游标为查询这个视图的会话所拥有。

列	数据类型	说明
ADDRESS	RAW(4)	拥有这个绑定变量的子级游标的内存地址
POSITION	NUMBER	绑定位置
DATATYPE	NUMBER	绑定数据类型
MAX_LENGTH	NUMBER	绑定值的最大长度
ARRAY_LEN	NUMBER	数组元素的最大数目（仅对数组绑定）
BIND_NAME	VARCHAR2(30)	绑定变量名（如果使用）

152. V$SQL_CURSOR

这个视图显示与查询这个视图的会话有关的每个游标的调试信息。

列	数据类型	说明
CURNO	NUMBER	游标号
FLAG	NUMBER	游标中的标志设置
STATUS	VARCHAR2(9)	游标的状态：即，游标处于什么状态

PARENT_HANDLE	RAW(4)	指向父级游标句柄的指针
PARENT_LOCK	RAW(4)	指向父级游标锁的指针
CHILD_LOCK	RAW(4)	指向子级游标锁的指针
CHILD_PIN	RAW(4)	指向子级游标固定的指针
PERS_HEAP_MEM	NUMBER	从这个游标的永久堆栈中分配的内存总量
WORK_HEAP_MEM	NUMBER	从这个游标的工作堆栈中分配的内存总量
BIND_VARS	NUMBER	进入此游标的当前分析的查询中绑定位置的总数
DEFINE_VARS	NUMBER	进入此游标的当前分析的查询中绑定位置的总数
BIND_MEM_LOC	VARCHAR2(64)	绑定变量存入的内存堆栈；或者是 UGA 或者是 CGA
INST_FLAG	VARCHAR2(64)	安装对象标志
INST_FLAG2	VARCHAR2(64)	安装对象标志（续）

153. V$SQL_SHARED_MEMORY

该使徒显示有关内存快照共享的游标信息。在共享池中共享的每个 SQL 语句都有一个或多个与之相关的子对象。每个子对象包括几部分，其中一个是上下文堆栈，她拥有查询计划。

列	数据类型	说明
SQL_TEXT	VARCHAR2(1000)	此行正显示信息的共享游标子对象的 SQL 文本
HASH_VALUE	NUMBER	共享池的 SQL 文本上的散列值
HEAP_DESC	RAW(4)	本行所描述的子级游标的上下文堆栈的描述符地址
STRUCTURE	VARCHAR2(16)	如果本行描述的内存组块使用格式为“X:Y”的说明进行分配，则这是该说明的“X”部分
FUNCTION	VARCHAR2(16)	类似于 STRUCTURE 列，这是该说明的“Y”域
COMMENT	VARCHAR2(16)	这是在分配内存组块时提供的整个说明域
CHUNK_PTR	RAW(4)	这是分配内存组块的起始地址
CHUNK_SIZE	NUMBER	该组块所分配的内存数量
ALLOC_CLASS	VARCHAR2(8)	内存组块所属的内存类。它通常为 FREEABLE 或 PERMANENT
CHUNK_TYPE	NUMBER	进入回收函数表的一个索引，这些函数告诉服务器如何重建内存组块
CHUNK_TYPE	NUMBER	进入回收函数表的一个索引，这些函数告诉服务器如何重建内存组块
SUBHEAP_DESC	RAW(4)	如果上下文堆栈的父堆栈本身是一个子堆栈，则这是父堆栈描述符的地址

154. V$SQL AREA

本视图列出共享 SQL 区域中的统计数据，并且每个 SQL 串包含一行。它提供在内存中的、经过语法分析的、准备执行的 SQL 语句的统计数据。

列	数据类型	说明
SQL_TEXT	VARCHAR2(1000)	当前游标的 SQL 文本的前 80 个字符
SHARABLE_MEM	NUMBER	在父级游标中的所有子级游标的以字节为单位的所有共享内存的总和
PERSISTENT_MEM	NUMBER	在父级游标中的所有子级游标的以字节为单位的所有永久内存的总和
RUNTIME_MEM	NUMBER	所有子级游标的临时帧尺寸的总和
SORTS	NUMBER	所有子级游标完成的排序数量的总和
VERSION_COUNT	NUMBER	出现在父级游标高速缓存中的子级游标的数量
LOADED_VERSIONS	NUMBER	出现在高速缓存中并使用上下文堆栈（KGL 堆栈 6）被加载的子级游标的数量
OPEN_VERSIONS	NUMBER	在当前父级游标中打开的子级游标的数量
USERS_OPENING	NUMBER	所有打开的子级游标的用户数量
EXECUTIONS	NUMBER	在所有子级游标上的执行总数
USERS_EXECUTING	NUMBER	在所有子级游标上执行语句的用户总数
LOADS	NUMBER	对象被加载或重新加载的次数
FIRST_LOAD_TIME	VARCHAR2(19)	父级游标创建时间的时间戳
INVALIDATIONS	NUMBER	所有子级游标上无效的总数
PARSE_CALLS	NUMBER	对父级游标中子级游标分析调用的总数
DISK_READS	NUMBER	在所有子级游标上的磁盘读取总数
BUFFER_GETS	NUMBER	在所有子级游标上的缓冲区的总数
ROWS_PROCESSED	NUMBER	代表 SQL 语句处理的总的行数
COMMAND_TYPE	NUMBER	Oracle 命令类型定义
OPTIMIZER_MODE	VARCHAR2(10)	SQL 语句在其下执行的模式
PARSING_USER_ID	NUMBER	分析了这个父级游标下的每个第一游标的用户的用户 ID
PARSING_SCHMA_ID	NUMBER	用来分析这个子级游标的模式 ID
KEPT_VERSIONS	NUMBER	已经利用 DBMS_SHARED_POOL 程序包标记为保持的子级游标的数目
ADDRESS	RAW(4)	这个游标的父级游标的句柄地址
HASH_VALUE	NUMBER	库高速缓存中的父级语句的散列值
MODULE	VARCHAR2(64)	包含第一次分析 SQL 语句执行时的模块名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
MODEL_HASH	NUMBER	在 MODULE 列中指定的模块的散列值
ACTION	VARCHAR2(64)	包含第一次分析 SQL 语句时执行的动作名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
ACTION_HASH	NUMBER	在 ACTION 列中指定的动作的散列值
SERIALIZABLE_ABORT	NUMBER	串行化事务处理失败，产生 ORA-8177 错误的次数，所有子级游标上的总计

155. V$SQLTEXT

这个视图包含属于 SGA 共享 SQL 游标的 SQL 语句文本。

列	数据类型	说明
ADDRESS	RAW(4)	与 HASH_VALUE 一道用来唯一标识一个高速缓存游标
HASH_VALUE	NUMBER	与 ADDRESS 一道用来唯一标识一个高速缓存游标
PIECE	NUMBER	用来排序 SQL 文本片段的编号
SQL_TEXT	VARCHAR2	一列包含 SQL 文本的一个片段
COMMAND_TYPE	NUMBER	SQL 语句（SELECT、INSERT）等的类型代码

156. V$SQLTEXT_WITH_NEWLINES

这个视图除为了增加可读性不用空格替换 SQL 语句中的新行和表外，与 V$SQLTEXT 视图相同。更多的信息，请参阅 V$SQLTEXT。

列	数据类型	说明
ADDRESS	RAW(4)	与 HASH_VALUE 一道用来唯一标识一个高速缓存游标
HASH_VALUE	NUMBER	与 ADDRESS 一道用来唯一标识一个高速缓存游标
PIECE	NUMBER	用来排序 SQL 文本片段的编号
SQL_TEXT	VARCHAR2	一列包含 SQL 文本的一个片段
COMMAND_TYPE	NUMBER	SQL 语句（SELECT、INSERT）等的类型代码

157. V$STATNAME

这个视图显示列在 V$SESSTAT 和 V$SYSSTAT 表中的统计数据的解码统计数据名。详细信息，请参阅 V$SESSTAT 和 SYSSTAT。

列	数据类型	说明
STATISTIC#	NUMBER	统计数据号
NAME	VARCHAR2	统计数据名。参见表 B-13
CLASS	NUMBER	1（用户）；2（重做）；4（排队）；8（高速缓存）；16（操作系统）；32（并行服务器）；128（调试）

表 B-13 列出了 V$ATATNAME 返回的普通 Oracle 统计数据。

表 B-13 V$SESSETAT 和 V$SYSSTAT 的统计数据名

此会话使用的 CPU	调用开始时使用的 CPU
建立的 CR 块	引用的高速缓存提交 SCN
引用的提交 SCN	为 CR 转换的当前块
扫描的 DBWR 缓冲区	DBWR 检查点缓冲区写入
DBWR 检查点	DBWR 强制写入

找到的 DBWR 可用缓冲区
DBWR 构造可用空间请求
DBWR 事务处理表写入
并行化 DDL 语句
并行化 DML 语句
读写 OShars
OS Input 块
OS Karnel 页故障睡眠时间
接收到的 OS Messages
OS Minor 页故障
OS Output 块
OS Process 堆栈尺寸
OS 交换
OS 系统调用
OS 用户级 CPU 时间
OS 主动环境切换
接收到的 PX 本地消息
接收到的 PX 远程消息
串行卸载的并行操作
SQL*Net 往返数据库连接
完成的后台检查点
后台超时
固定缓冲区计数
通过 SQL*Net 从数据库连接接收到的字节
通过 SQL*Net 发送到数据库连接的字节
对 kcmgas 的调用
对 kcmgrs 的调用
清除和回退一致性读获取
簇键扫描块获取
提交清除失败；块丢失
提交清除失败；回叫失败
提交清除失败；正在进行热备份
提交清除
游标验证
数据库块获取
查到灰缓冲区
排队死锁
排队请求
排队等待
执行计数
请求的空闲缓冲区
全局高速缓存转换超时
全局高速缓存 cr 块接收时间
全局高速缓存从磁盘读取
全局高速缓存延迟
全局高速缓存空闲列表等待
全局高速缓存转换时间
全局锁同义词转换@@@@@
全局锁转换时间
全局锁转换（非异步）
全局高速缓冲散栓锁等待
DBWRLRL 扫描
访问被写入缓冲区的 DBWR
DBWR 累加扫描深度
DBWR 撤消块写入
并行化 DFO 数
OS 所有其他睡眠时间
OS 数据页故障睡眠时间
OS 强制上下文切换
OS 消息发送
OS 其他系统陷阱 CPU 时间
OS 进程堆尺寸
OS 信号接收
OS 系统调用 CPU 时间
OS 文本叶故障睡眠时间
OS 用户锁等待睡眠时间
OS 等待 CPU（延迟时间)
PX 本地消息发送
PX 远程消息发送
SQL*Net 从客户机往返
SCN 批处理的不必要的进程清除
后台检查点开始
未固定缓冲区计数
通过 SQL*Net 从客户机接收的字节
通过 SQL*Net 发送到客户机的字节
取得快照的 SCN 调用：kcmgss
调用 kcmgss
更改写入时间
只清除一致性读获取
簇键扫描
提交清除故障：缓冲区被写入
提交清除故障：不能固定
提交清除故障：写禁止
一致性获取
数据库块更改
延迟（当前）块清除应用程序
排队转换
排队释放
排队超时
交死锁
查到空闲缓冲区
全局高速缓存转换时间
全局高速缓存转换
全局高速缓存 cr 块接收
全局高速缓存 cr 超时
全局高速缓存顺利转换
全局高速缓存获取时间
全局高速缓存排队转换
全局锁异步获取
全局锁转换（异步）
全局锁获取时间
全局锁获取（异步）

全局锁获取（非异步）
全局锁释放
全局锁同义词转换@@@@@
全局锁同步获取
热缓冲区移动到 LRU 的标题
立即（CR）清除应用程序
立即（CURRENT）块清除应用程序
实例恢复数据库冻结计数
kcmccs 调用取得当前 scn
kcmccs 读取 scn 不转到 DCM
kcmccs 等待批处理
登录累计
当前登录
收到消息
消息发送
本地散列算法执行
本地散列算术失败
不用转到 DLM 取得的下一 scm
无缓冲区固定计数
非工作一致性读获取
累计打开的游标
当前打开的游标
打开替换文件
打开请求高速缓存替换
分析计数（硬）
分析计数（总计）
分析 CPU 时间
占用的分析时间
物理读取
物理写
直接物理读取
直接物理写
非检查点物理写
非检查点物理写@@@@@@@@
查到固定换冲区
处理最后非空闲时间
并行化查询
恢复数组读取时间
恢复数组读取
恢复块读取
递归调用
递归 cpu 用法
写入重做块
重做缓冲区分配项
重做项
重做日志空间请求
重做日志空间等待时间
重做日志切换中断
重做排序标记
重做尺寸
重估同步时间
重做同步写
重做消耗
重做写时间
重做写入程序闭锁时间
重做写
远程实例撤消块写入
远程实例撤消标题写入
应用回退更改撤消记录
回退段仅一致性读取获取
可串行终止
会话连接时间
会话游标高速缓存计数
会话游标高速缓存命中
会话逻辑读取
会话 pga 内存
最大会话 pga 内存
会话存储过程空间
会话 uga 内存
最大会话 uga 内存
排序（磁盘）
排序（内存
排序（行）
总计灰队列长度
按行标识符的表取数
表取数据连续行
获取的表扫描块
获得表扫描行
表扫描（高速缓存分区）
表扫描（直接读取）
表扫描（长表）
表扫描（行标识范围）
表扫描（短表）
总的文件打开
事务处理锁后台获取时间
事务处理锁后台获取
事务处理锁前台请求
事务处理锁前台等待时间
事务处理回退
事务处理表一致性读取回退
应用事务处理表一致性读取撤消记录
用户调用
用户提交
用户回退

其他信息：在某些平台上，NAME 和 CLASS 列将包含其他操作系统的专门数据。

158. V$SUBCACHE

这个视图显示当前装入库高速缓存内存的下级高速缓存的相关信息。此视图预排库高速缓存，每个库高速缓存对象的每个装入下级高速缓存印出一行。

列	数据类型	说明
OWNER_NAME	VARCHAR2(64)	包含这些高速缓存项的对象的拥有者
NAME	VARCHAR2(1000)	对象名
TYPE	NUMBER	对象类型
HEAP_NUM	NUMBER	包含这个下级高速缓存的堆栈号
CACHE_ID	NUMBER	下级高速缓存 ID
CACHE_CNT	NUMBER	这个对象中这个高速缓存的项数
HEAP_SZ	NUMBER	分配给这个堆栈的区空间量
HEAP_ALOC	NUMBER	从这个堆栈中分配的区空间量
HEAP_USED	NUMBER	从这个堆栈中利用的空间量

159. V$SYSSTAT

这个使徒列出系统统计数据。为找到与每个统计数据号（STATISTIC#）关联的统计数据名称，请参阅 V$STATNAME。

列	数据类型	说明
STATISTIC#	NUMBER	统计数据号
NAME	VARCHAR2	统计数据名。参见表 B-13
CLASS	NUMBER	统计数据类别：1（用户）；2（重做）；4（排队）；8（高速缓存）；16（操作系统）；32（并行服务器）；64（SQL）；128（调试）
VALUE	NUMBER	统计数据值
CLASS	NUMBER	统计数据类别：

160. V$SYSTEM_CURSOR_CACHE

除这个视图的信息是系统范围的外，其显示的信息与 V$SESSION_CURSOR_CACHE 的类似。更详细的信息，请参阅 V$SESSION_CURSOR_CACHE。

列	数据类型	说明
OPENS	NUMBER	游标打开的累计总数
HITS	NUMBER	游标打开命中的累计总数
HIT_RATIO	NUMBER	找到打开游标的次数除以查找游标的次数所得到的比例

161. V$SYSTEM_EVENT

这个视图包含所有等待某个事件的相关信息。注意，TIME_WAITED 和 AVERAGE_WAIT 列在那些不支持快速时间机制的平台上将包含零值。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 TIMED_STATISTICS
为 TRUE。请注意，这样做对系统性能有轻微的负面影响。更多的信息，请参阅“TIMED_STATISTICS”。

列	数据类型	说明

EVENT	VARCHAR2(64)	等待事件的名称
TOTAL_WAITS	NUMBER	这个事件的总等待次数
TOTAL_TIMEOUTS	NUMBER	这个事件的总等待超时次数
TIME_WAITED	NUMBER	这个事件的总等待时间数，以百分之一秒计
AVERAGE_WAIT	NUMBER	这个事件的平均等待时间，以百分之一秒计

162. V$SYSTEM_PARAMETER

这个视图包含关于系统参数的信息。

列	数据类型	说明
NUM	NUMBER	参数号
NAME	VARCHAR2(64)	参数名
TYPE	NUMBER	参数类型：1=布尔；2=串；3=整数
VALUE	VARCHAR2(512)	赋给此参数的值
ISDEFAULT	VARCHAR2(9)	赋给此参数的值是否为缺省值
ISSES_MODIFIABLE	VARCHAR2(5)	此参数是否可用 ALTER SESSION 更改
ISSYS_MODIFIABLE	VARCHAR2(9)	此参数是否可用 ALTER SYSTEM 更改
ISMODIFIED	VARCHAR2(8)	指出参数怎样更改。如果执行一条 ALTER SESSION，则值将被 MODIFIED。如果执行一条 ALTER SYSTEM(它将导致所有当前登录会话的值被修改)，则参数值将为 SYS_MODIFIED
ISADJUSTED	VARCHAR2(5)	指出关系型数据库系统调整输入值为一个更合适的值（即，参数值应该为素数，但用户输入一个非素数，因此关系型数据库系统将该值调整为下一个素数）
DESCRIPTION	VARCHAR2(64)	有关此参数的描述性文本

163. V$TABLESPACE

这个视图来自控制文件的信息。

列	数据类型	说明
TS#	NUMBER	表空间数
NAME	VARCHAR2(30)	表空间名

164. V$TEMPFILE

这个视图显示临时文件信息。

列	数据类型	说明
FILE#	NUMBER	数据文件号
CREATION_CHANGE#	NUMBER	创建系统更改号

CREATION_TIME	DATE	创建时间
TS#	NUMBER	表空间号
RFILE	NUMBER	表空间中的相对文件号
STATUS	VARCHAR2(7)	文件状态（脱机/联机）
ENABLED	VARCHAR2(10)	读和/或写的启用
BYTES	NUMBER	以字节表示的文件尺寸（来自文件标题）
BLOCKS	NUMBER	块中的文件尺寸（来自文件标题）
CREATE_BYTES	NUMBER	文件的创建尺寸（以字节表示）
BLOCK_SIZE	NUMBER	文件的块尺寸
NAME	VARCHAR2(513)	文件名

165. V$TEMPORARY_LOBS

这个视图显示临时 lobs。

列	数据类型	说明
SID	NUMBER	会话 ID
CACHE_LOBS	NUMBER	编号高速缓存临时 lobs
NOCACHE_LOBS	NUMBER	非高速缓存临时 lobs 的数目

166. V$TEMP_EXTENT_MAP

这个视图显示所有临时表空间的每个单元的状态。

列	数据类型	说明
TABLESPACE_NAME	NUMBER	这个单元所属的表空间名
FILE_ID	NUMBER	绝对文件号
BLOCK_ID	NUMBER	这个单元的起始块号
BYTES	NUMBER	区中的字节数
BLOCKS	NUMBER	区中的块数
OWNER	NUMBER	拥有这个单元（串）的实例
RELATIVE	FNO	相对文件号

167. V$TEMP_EXTENT_POOL

这个视图显示为某个给定实例高速缓存和使用的临时表空间的状态。注意，临时表空间高速缓存的装载是很迟钝的，该实例可以是静止的。关于所有实例的信息可使用 GV$TEMP_EXTENT_POOL。

列	数据类型	说明
TABLESPACE_NAME	NUMBER	这个单元所属的表空间名
FILE_ID	NUMBER	绝对文件号
EXENTS_CACHED	NUMBER	已经高速缓存了多少个区
EXENTS_USED	NUMBER	实际正在使用的区有多少个
BLOCKS_CACHED	NUMBER	高速缓存的块数
BLOCKS_USED	NUMBER	使用的块数

BYTES_CACHED	NUMBER	高速缓存字节数
BYTES_USED	NUMBER	使用的字节数
RELATIVE_FNO	NUMBER	相对文件号

168. V$TEMP_PING

这个视图显示每个数据文件 ping 的块数。这个信息又可用来确定现有数据文件的访问模式，决定从数据文件块到 PCM 锁的新映射。

列	数据类型	说明
FILE_NUMBER	NUMBER	数据文件号
FREQUENCY	NUMBER	频率
X_2_NULL	NUMBER	文件中所有块从互斥到空的锁转换数
X_2_NULL_FORCED_WRITE	NUMBER	指定文件中的块由于互斥到空的转换而强制写的数目
X_2_NULL_FORCED_STALE	NUMBER	文件中的块由于互斥到空的转换而使其 STATE 的次数
X_2_S	NUMBER	文件中所有块从互斥到共享的锁转换数
X_2_S_FORCED_ WRITE	NUMBER	指定文件中的块由于互斥到共享的转换而强制写的数目
X_2_SSX	NUMBER	文件中所有块从互斥到子共享互斥的锁转换数
X_2_SSX_FORCED_WRITE	NUMBER	指定文件中的块由于互斥到子共享互斥的转换而强制写的数目
S_2_NULL	NUMBER	文件中所有块从共享到空的锁转换数
S_2_NULL_FORCED_STALE	NUMBER	文件中的块由于共享到空的转换而使其 STATE 的次数
SS_2_NULL	NUMBER	文件中所有块从子共享到空的锁转换数
WRB	NUMBER	实例接收到一个跨实例调用这个文件的写入单个缓冲区的次数
WRB_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而写入单个缓冲区导致写入的块数
RBR	NUMBER	实例接收到一个跨实例调用这个文件的重用块范围的次数
RBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而重用块范围导致写入的块数
RBR_FORECD_ TATE	NUMBER	由于跨实例调用而重用块范围而使得这个文件中的块的 STATE 的数目
CBR	NUMBER	实例接收到一个跨实例调用这个文件的检查点块范围的次数
CBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件的检查点跨范围所导致的写入这个文件中块的数目
NULL_2_X	NUMBER	指定文件中所有块从空到互斥的锁转换数
S_2_X	NUMBER	指定文件中所有块从共享到互斥的锁转换数

SSX_2_X	NUMBER	指定文件中所有块从子共享互斥到的互斥的锁转换数
NULL_2_S	NUMBER	指定文件中所有块从空到共享的锁转换数
NULL_2_SS	NUMBER	指定文件中所有块从空到子共享的锁转换数
OP_2_SS	NUMBER	打开的 PCM 锁子共享锁的数目。Oracle 8.1 中为 0

169. V$TEMP_SPACE_HEADER

这个视图显示每个临时表空间的每个文件的聚集信息，即当前已使用的内存量和每个空间标题的可用空间量的信息。

列	数据类型	说明
TABLESPACE_NAME	VARCHAR2(30)	临时表空间名
FILE_ID	NUMBER	绝对文件号
BYTES_USED	NUMBER	已使用的字节数
BLOCK_USED	NUMBER	已使用的块数
BYTES_FREE	NUMBER	可用字节数
BLOCKS_FREE	NUMBER	可用块数
RELATIVE_FNO	NUMBER	文件的相对文件号

170. V$TEMPSTAT

该视图包含有关文件读/写统计的信息。

列	数据类型	说明
FILE#	NUMBER	文件号
PHYRDS	NUMBER	完成物理读取的数量
PHYWRTS	NUMBER	DBWR 需要写操作的次数
PHYBLKRD	NUMBER	读取物理块的数量
PHYBLKWRT	NUMBER	写入磁盘的块数：如果所有的写操作是针对单个块的，则 PHYBLKWRT 与 PHYWRTS 相同
READTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行读操作所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
WRITETIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行写操作所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
AVGIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为 I/O 所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行最近的 I/O 所花费的时间（以百分之一秒
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为

		TRUE，则为进行最近的 I/O 所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
MINOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个 I/O 所花费的最小时间（以百分之一秒计）；如果为 FALSE，则为 0
MAXIOWTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个写操作所花费的最大时间（以百分之一秒计）；如果为 FALSE，则为 0
MAXIORTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个读操作所花费的最大时间（以百分之一秒计）；如果为 FALSE，则为 0

171. V$THREAD

该视图包含控制文件中的线程信息。

列	数据类型	说明
THREAD#	NUMBER	线程号
STATUS	VARCHAR2	线程状态：OPEN、CLOSE
ENABLE	VARCHAR2	启用状态：DISABLED，（启用）PRIVATE，或（启用）PUBLIC
ENABLE_CHANGE#	NUMBER	启用线程的 SCN
ENABLE_TIME	DATE	启用 SCN 的时间
DISABLE_CHANGE#	NUMBER	禁用线程的 SCN
DISABLE_TIME	DATE	禁用 SCN 的时间
GROUPS	NUMBER	分配给该线程的日志组的数量
INSTANCE	VARCHAR2	实例名
OPEN_TIME	DATE	打开线程的最近时间
CURRENT_GROUP#	NUMBER	当前日志组
SEQUENCE#	NUMBER	当前日志序列号
CHECKPOINT_CHANGE#	NUMBER	最近的检查点的 SCN
CHECKPOINT_TIME	DATE	最近的检查点的时间

172. V$TIMER

此视图列出以百分之一秒为单位的消耗时间。时间自启动以来进行度量，它是操作系统专有的，当该值溢出 4 字节时（大约为 497 字节），再次返回到 0。

列	数据类型	说明
SHECS	NUMBER	占用的时间（以百分之一秒计）

173. V$TRANSACTION

该视图列出系统的活动事务处理。

列	数据类型	说明
ADDR	RAW(4)	事务处理状态对象的地址
XIDUSN	NUMBER	撤消段的号
XIDSLOT	NUMBER	插曹号
XIDSQN	NUMBER	序列号
UBAFIL	NUMBER	撤消块地址（UBA）的文件号
UBABLK	NUMBER	UBA 块号
UBASQN	NUMBER	UBA 序列号
UBAREC	NUMBER	UBA 记录号
STATUS	VARCHAR2(16)	状态号
START_TIME	VARCHAR2(20)	起始时间（挂钟）
START_SCNB	NUMBER	起始系统更改号（SCN）的基点
START_SCNW	NUMBER	起始 SCN 包
START_UEXT	NUMBER	起始区号
START_UBAFIL	NUMBER	起始 UBA 文件号
START_UBABLK	NUMBER	起始 UBA 块号
START_UBASQN	NUMBER	起始 UBA 序列号
START_UBAREC	NUMBER	起始记录号
SER_ADDR	RAW(4)	用户会话对象地址
FLAG	NUMBER	标志
SPACE	VARCHAR2(3)	如果为空间事务处理,则为 Yes
RECURSIVE	VARCHAR2(3)	如果为递归事务处理,则为 Yes
NOUNDO	VARCHAR2(3)	如果为撤消事务处理,则为 Yes
PTX	VARCHAR2(3)	如果为并行事务处理，则为 Yes，否则设为 No
PRV_XIDUSN	NUMBER	上一个事务处理的撤消段的号
PRV_XIDSLT	NUMBER	上一个事务处理的插槽号
PRV_XIDSQN	NUMBER	上一个事务处理的序列号
PTX_XIDUSN	NUMBER	父级 XID 的回退段号
PTX_XIDSLT	NUMBER	父级 XID 的插曹号
PTX_XIDSQN	NUMBER	父级 XID 的序列号
DSCN_B	NUMBER	独立的 SCN 基点
DSCN_W	NUMBER	独立的 SCN 包
USED_UBLK	NUMBER	已用的撤消块数量
USED_UREC	NUMBER	已用的撤消记录数量
LOG_IO	NUMBER	逻辑 I/O
PHY_IO	NUMBER	物理 I/O
CR_GET	NUMBER	一致性获取
CR_CHANGE	NUMBER	一致性更改

174. V$TRANSACTION_ENQUEUE

显示由事务处理状态对象拥有的锁。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	会话拥有或获取锁的标识符
TYPE	VARCHAR2(2)	锁类型。TX=事务处理队列

ID1	NUMBER	锁标识符#1（取决于类型）
ID2	NUMBER	锁标识符#2（取决于类型）
LMODE	NUMBER	会话拥有锁的方式：0，无；1，空（NULL）；2，行-S（SS）；3，行-X（SX）；4，共享（S）；5，S/行-X（SSX）；6，独占（X）
REQUEST	NUMBER	会话请求锁的方式：0，无；1，空（NULL）；2，行-S（SS）；3，行-X（SX）；4，共享（S）；5，S/行-X（SSX）；6，独占（X）
CTIME	NUMBER	自当前方式授权以来的时间
BLOCK	NUMBER	该锁正在阻塞另一个锁

175. V$TYPE_SIZE

该视图列出各种数据库组件的尺寸以估计数据块的容量。

列	数据类型	说明
COMPONENT	VARCHAR2	组件名，如段或缓冲区标题
TYPE	VARCHAR2	组件类型
DESCRIPTION	VARCHAR2	组件说明
TYPE_SIZE	NUMBER	组件大小

176. V$VERSION

Oracle 服务器的核心库组件的版本号。每个组件一行。

列	数据类型	说明
BANNER	VARCHAR2	组件名和版本号

177. V$WAITSTAT

此视图列出块争用统计信息。此表只能在启用计时统计信息时更新。

列	数据类型	说明
CLASS	VARCHAR2	块类
COUNT	NUMBER	针对块类的操作的等待次数
TIME	NUMBER	针对块类的操作的所有等待次数的总和

附录 D　DBA 常用管理命令

ALTER DATABASE

参见：

Instance Manager， CREATE DATABASE，
Database Assistant DROP DATABASE

目的：

使用以下一种方式修改已有的数据库

- 装配数据库，复制或备份数据库
- 从 ORACLE 7 到 ORACLE8 时转换数据字典
- 打开数据库
- 为日志文件组选择 ARCHIVELOG 或 NOARCHIVELOG
- 执行介质复原
- 增加或删除日志文件或日志文件组成员
- 清除或初始化连接日志文件
- 重新命名日志文件或数据文件
- 备份当前控制文件
- 将 SQL 命令备份到数据库的跟踪文件
- 将数据文件置与 online 或 offline
- 允许或静止日志文件组进程
- 改变一个或多和数据库大小
- 出于恢复的目的，创建一个新的数据文件以替换旧数据文件
- 允许或静止自动扩展数据文件大小

语法：

```
ALTER DATABASE [database]
[ADD LOGFILE
[THREAD n] [GROUP n]
file_definition [.file_definition] ... |
ADD LOGFILE MEMBER
File [REUSE] [， file [REUSE] ...]
TO [GROUP n |
(file [， file] ...) |
file
file [REUSE] [， file [REUSE] ...]
TO [GROUP n |
(file [， file] ...) |
file) ] .. |
DROP LOGFILE
{ GROUP n | (file [, file] ... ) | file}
[GROUP n | (file [, file] ... ) | file] ... |
```

```
DROP LOGFILE MEMBER file [,  file] |
RENAME FILE file to file |
NOARCHIVELOG | ARCHIVELOG |
MOUNT [EXCLUSIVE | PARALLE] |
OPEN [RESETLOGS | NORESETLOGS] |
ENABLE [PUBLIC] THREAD n |
DISABLE THREAD n |
BACKUP CONTROLFILE TO file [REUSE]
DATAFILE file [ONLINE | OFFLINE [DROP]] |
CREATE DATAFILE file[, file]
AS file_definition [.file_definition] ... |
RENAME GLOBAL_NAME TO database [.domain] |
RECOVER recover_clause |
SET [DBMAC [ON | OFF] | DBHIGH = string | DBLOW =
string ]
```

例子：

```
SQL
ALTER DATABASE NO MOUNT ;
ALTER DATABASE ADD LOGFILE GROUP 2 'customer.002' size 2m;
```

ALTER INDEX

参见：

CREATE INDEX， DROP INDEX

目的：

使用 ALTER INDEX 语句实现

- 改变存储器分配、重建或重新命名索引
- 重命名、分离、除去、标识无用，重建或改变分区索引的分区物理或登录的属性
- 改变未分区索引的物理、并行的或登录的属性
- 改变分区索引的缺省物理并行或登录的属性
- 改变索引分区的缺省物理的登录属性
- 重建索引，以相反顺序存储索引块中的字节
- 改变嵌套表的索引

语法：

```
ALTER [UNIQUE] INDEX [user.]index
[INITRANS n]
[MAXTRANS n]
[STORAGE n]
```

变量：

user:用户名

index: 索引名

例子：
SQL
ALTER INDEX loan_application;

ALTER PROFILE
参见：
CREATE PROFILE， DROP PROFILE

目的：
增加、修改或删除资源限制或环境文件中的口令管理

语法：
ALTER PROFILE profile LIMIT
[{ SESSIONS_PER_USER |
CPU_PER_SESSION |
CPU_PER_CALL |
CONNECT_TIME |
IDLE_TIME |
LOGICAL_READS_PER_SESSION |
LOGICAL_READS_PER_CALL |
COMPOSITE_LIMIT } { n | UNLIMITED |
DEFAULT } |
PRIVATE_SGA { n {K | M} | UNLIMITED |
DEFAULT }]

变量：
profile: 资源名字
n: 整数

例子：
SQL
ALTER PROFILE customer LIMIT SESSION_PER_USER 12 ;
ALTER PROFILE customer LIMIT CONNECT_TIME 600 ;

ALTER ROLE
参见：
CREATE ROLE， DROP ROLE

目的：
改变可用交色所需的权限

语法：
ALTER ROLE role
[NOT IDENTIFIED | IDENTIFIED

[BY PASSWORD | EXTERNALLY]]
变量：
role: 角色名

例子：
SQL
ALTER ROLE custom_user;

ALTER ROLLBACK SEGMENT
参见：
CREATE ROLLBACK SEGMENT,
DROP ROLLBACK SEGMENT
目的：
以下列一种方式修改已有的回混段

- 使联机
- 使脱机
- 改变存储特性
- 使收缩或最优或给出大小

语法：
ALTER [PUBLIC] ROLLBACK SEGMENT rollback_segment
[STORAGE storage]

例子：
SQL
ALTER PUBLIC ROLLBACK SEGMENT humanresources;
ALTER ROLLBACK SEGMENT insurance;

ALTER SYSTEM
参见：
COMMIT, ROLLBACK,
SAVEPOINT,
SET TRANSACTION
目的：
使用以下一种方式动态改变 ORACLE 实例

- 只对 RESTRICTED SESSION 系统特权的用户限制 ORACLE 注册
- 在系统全局区内（SGA）

. . . . . .

语法：
ALTER SYSTEM
[SET { RESOURCE_LIMIT = { TRUE | FALSE } |

```
MTS_SERVERS = n
MTS_DSIPATCHERS = 'protocol.n'
SWITCH LOGFILE |
{ CHECKPOINT | CHECK DATAFILES } { GLOBAL | LOCAL } |
{ ENABLE | DISABLE }
{ DISTRIBUTED RECOVERY | RESTRICTED SESSION } |
ARCHIVE LOG archive_log_clause
FLUSH SHARE_POOL
KILL SESSION 'sid, serial number' ]
```

变量：
archive_log_clause: 归档日志项
sid: 会话标识号
serial number: 序列号

例子：
SQL
```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE
```

ALTER TABLE
参见：
CREATE TABLE, DROP TABLE
语法：
```
ALTER TABLE [user.] table
{ [ ADD ( { column1 | table_constraint }
[, column2 | table_constraint} ] ... ) ]
[ MODIFY ( column1, column2) ]
[ DROP drop_constraint]
[PCTFREE n]
[PCTUSED n]
[INITRANS n]
[MAXTRANS n]
[STORAGE n]
ALLOCATE EXTENT
[ SIZE n [K | M]]
[DATAFILE file]
[INSTANCE n]
[ ENABLE | DISABLE]
```
变量：
user：建立表的用户名
table：希望改变的表名
column：列名
table_constraint：表的限制，如 NULL, NOT NULL 等
n：任意正整数值

file：物理数据文件名

例子：
SQL
```
ALTER TABLE customer
( ADD ( address VARCHAR2(50));
ALTER TABLE customer
( MODIFY ( name VARCHAR2(50) NOT NULL);
```

ALTER TABLESPACE
参见：
CREATE TABLESPACE
语法：
```
ALTER TABLESPACE tablespace
[ ADD DATAFILE file_definition
[, file_definition] |
RENAME DATAFILE file [, file] ... TO
file [,  file] |
DEFAULT STORAGE storage |
ONLINE | OFFLINE [NORMAL | IMMEDIATE]
|
[ BEGIN | END] BACKUP ]
```
变量：
tablespace：希望改变的数据库表空间名
file_definition：带大小（ K or M）的文件名

例子：
SQL
```
ALTER TABLESPACE customer
( ADD DATAFILE 'customer02.dat');
```

ALTER USER
参见：
CREATE USER, DROP USER
语法：
```
ALTER USER user [IDENTIFIED [BY password | EXTERNALLY]]
[ DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[, QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[PROFILE profile]
[DEFAULT ROLE (role1,  role2,  ...) | ALL EXCEPT
(role1,  role2,  ...) |
NONE]
```

变量：
user：希望改变的数据库用户名
tablespace：用户建立的表空间名
n：任意正整数值
profile：用户要使用的 profile 名
role：用户指定（或不指定）的角色

例子：
SQL
```
ALTER USER customer
DEFAULT tablespace cust_tablespace
TEMPORARY tablespace temp;
```

ANALYZE
参见：
EXPLAIN PLAN， AUDIT
语法：
```
ANALYZE
{ INDEX [user.]index
{ { COMPUTE | ESTIMATE | DELETE } STATISTICS
[ SAMPLE ( n PERCENT | n ROWS } ] |
VALIDATE STRUCTURE } |
{ TABLE [user.]table | CLUSTER [user.]cluster}
{ { COMPUTE | ESTIMATE | DELETE } STATISTICS
[ SAMPLE ( n PERCENT | n ROWS } ] |
VALIDATE STRUCTURE } [CASCASE] |
LIST CHAINED ROWS [INTO [user.]table] }
```
变量：
index：希望分析的索引的名字
n：任意正整数值
table：希望分析的表名
cluster：希望分析的 cluster 名

例子：
SQL
```
ANALYZE INDEX cust_index ESTIMATE STATISTICS;
```

AUDIT
参见：
NO AUDIT， ANALYZE

语法：
```
AUDIT command | ALL
ON [user.]object | DEFAULT
```

[BY SESSION | ACCESS]
[WHENEVER [NOT] SUCCESSFUL]
变量：
command：可以审计的命令： ALTER, AUDIT, COMMENT, DELETE, EXECUTE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT 及 UPDATE
User：建立对象的用户名
Object：对象名，可以是 table, view, synonym, sequence, procedure, function, package, 或 snapshot.

例子：
SQL
AUDIT INSERT ON customer WHENEVER NOT SUCCESSFUL;

CREATE CONTROLFILE
参见：
CREATE DATABASE

语法：
CREATE CONTROLFILE [REUSE] [SET] DATABASE database
LOGFILE [GROUP n] file [, [GROUP n] file] (RESETLOG | NORESETLOG)
DATAFILE file [, file]
[MAXLOGFILES n]
[MAXLOGMEMBERS n]
[MAXLOGHISTORY n]
[MAXDATAFILES n]
[MAXINSTANCE n]
[ARCHIVELOG | NOARCHIVELOG]
变量：
n: 任意整数
file: 物理文件名

例子：
SQL
CREATE CONTROLFILE REUSE SET DATABASE ORACLE
LOGFILE 'D:\ORAWIN95\DATABASE\LOG1ORCL.ORA' SIZE
200K,
DATAFILE 'D:\ORAWIN95\DATABASE\SYS1ORCL.ORA' SIZE
20M;.

CREATE DATABASE
参见：
ALTER DATABASE,
CREATE CONTROLFILE

语法：
CREATE DATABASE [database] [CONTROLFILE REUSE] LOGFILE
[GROUP n] file [, [GROUP n] file]
(RESETLOG | NORESETLOG)
DATAFILE file [, file]
[MAXLOGFILES n]
[MAXLOGMEMBERS n]
[MAXLOGHISTORY n]
[DATAFILE file_definition [, file_definition]]
[MAXDATAFILES n]
[MAXINSTANCE n]
[ARCHIVELOG | NOARCHIVELOG]
[EXCLUSIVE]
(CHARACTER SET charset)
变量：
n：任何正整数
file：物理文件名
charset：字符集

例子：
SQL
CREATE DATABASE ORACLE
CONTROLFILE REUSE
LOGFILE 'D:\ORAWIN95\DATABASE\LOG1ORCL.ORA' SIZE 200K
REUSE, 'D:\ORAWIN95\DATABASE\LOG2ORCL.ORA' SIZE 200K
REUSE
DATAFILE 'D:\ORAWIN95\DATABASE\SYS1ORCL.ORA' SIZE 20M
REUSE AUTOEXTEND ON
NEXT 10M MAXSIZE 200M
CHARACTER SET WE8ISO8859P1;.

CREATE DATABASE LINK
参见：
Distributed Database Applications CREATE SYNONYM
语法：
CREATE [PUBLIC] DATABASE LINK link
CONNECT TO user IDENTIFIED
BY password USING 'connect_string'
变量：
link：数据库链名
user：数据库用户
password：有效的口令
connect_string：被访问的远程数据库字串

例子：
SQL
```
CREATE DATABASE LINK international_customers connect to INTL_DB
identified by intl using 'D:INTERNATIONAL';
SELECT CUSTOMER_NAME FROM CUSTOMER@INTL_DB;.
```

CREATE PROFILE
参见：
User Profiles ALTER PROFILE， ALTER
RESOURCE COST， DROP
PROFILE
语法：
```
CREATE PROFILE profile LIMIT
[ { SESSIONS_PER_USER |
CPU_PER_SESSION |
CPU_PER_CALL |
CONNECT_TIME |
IDLE_TIME |
LOGICAL_READS_PER_SESSION |
LOGICAL_READS_PER_CALL |
COMPOSITE_LIMIT } { n | UNLIMITED | DEFAULT } |
PRIVATE_SGA { n {K | M} | UNLIMITED | DEFAULT } ]
```
变量：
profile：希望修改的 profile 文件名
n：任何正整数

例子：
SQL
```
CREATE PROFILE customer LIMIT SESSION_PER_USER 12 ;
CREATE PROFILE customer LIMIT CONNECT_TIME 600 ;.
```

CREATE DIRECTORY
参见：
LOBs DROP DIRECTORY， BFILENAME
语法：
```
CREATE [OR REPLACE] DIRECTORY directory AS pathname;
```
变量：
directory：建立对象的目录
pathname：单引号引起的完整路径
例子：
SQL
```
CREATE OR REPLACE DIRECTORY bfile_directory AS '/bfile';
```

CREATE FUNCTION

参见：
Functions ALTER FUNCTION,
DROP FUNCTION
语法：
CREATE [OR REPLACE] FUNCTION [user.]function
[(parameter [IN] datatype [, parameter [IN] datatype] ...)]
RETURN datatype (IS | AS) block
变量：
user：建立函数的用户名
function：数据库函数名
parameter：传给函数的参数
datatype：变量的数据类型
block: PL/SQL 程序块

例子：
SQL
CREATE FUNCTION loan_calculation (loan_amount NUMBER,
no_of_years NUMBER) RETURN NUMBER AS
Total_loan NUMBER;
BEGIN
Total_loan := loan_amount * no_of_years;
RETURN total_loan ;
END loan_calculation ;

CREATE INDEX
参见：
Indexes ALTER INDEX, DROP INDEX
语法：
CREATE INDEX [user.]index
ON [user.]table (column [ASC | DESC] [, column
[ASC | DESC]] ...)
[CLUSTER [user.]cluster]
[INITRANS n]
[MAXTRANS n]
[PCTFREE n]
[STORAGE storage]
[TABLESPACE tablespace]
[NO SORT]
变量：
user：建立索引的用户名
index：索引名
table：要建索引的表名
cluster：要建索引的簇名
n：任何正整数

tablespace：存放索引的表空间

例子：
SQL
```
CREATE INDEX loan_application_indx
ON loan (loan_id ASC)
TABLESPACE temp;
CREATE INDEX customer_indx
ON loan (cust_id ASC,  loan_id ASC)
TABLESPACE temp;.
```

CREATE LIBRARY
参见：
CREATE FUNCTION, CREATE
PROCEDURE, DROP LIBRARY
语法：
```
CREATE [OR REPLACE] LIBRARY library_name [IS | AS] filename;
```
变量：
library_name：SQL 和 PL/SQL 要调用的外部（3GL）函数及存储过程
filename：外部物理文件名

例子：
SQL
```
CREATE LIBRARY ext_lib IS '/lib/file1.sql';
CREATE OR REPLACE LIBRARY ext_lib2 AS '/lib/file2.sql';
```

CREATE PACKAGE
参见：
 Packages CREATE PACKAGE BODY,
ALTER PACKAGE,
DROP PACKAGE
语法：
```
CREATE [OR REPLACE] PACKAGE [user.]package { IS | AS}
{ variable_declaration |
cursor_specification |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_specification |
function_specification } ;
[{ variable_declaration |
cursor_specification |
exception_declaration |
record_declaration |
```

plsql_table_declaration |
procedure_specification |
function_specification } ;] ...
END [package]
变量：
user：建立 Package 的用户名
package：建立 Package 的名

例子：
SQL
CREATE OR REPLACE PACKAGE loan_approval AS
Type LoanRecTyp IS RECORD (customer_id INTEGER, loan_amount
REAL);
CURSOR customer_history (customer_id NUMBER) RETURN LoanRecTyp;
PROCEDURE approve_loan
(customer_id CHAR,
loan_type CHAR,
loan_amount CHAR);
PROCEDURE cumulative_loan (loan_amount REAL);
END loan_approval ;.

CREATE PACKAGE BODY
参见：
CREATE PACKAGE,
ALTER PACKAGE BODY,
DROP PACKAGE BODY
语法：
CREATE [OR REPLACE] PACKAGE BODY [user.]package { IS |
AS}
{ variable_declaration |
cursor_body |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_body |
function_body } ;
[{ variable_declaration |
cursor_body |
exception_declaration |
record_declaration |
plsql_table_declaration |
procedure_body |
function_body } ;] ...
END [package]

变量：
user：建立 Package 的用户名
package：在数据库中的 Package 的名字

例子：
SQL
```
CREATE OR REPLACE PACKAGE BODY loan_approval AS
CURSOR customer_history (customer_id NUMBER) RETURN LoanRecTyp
IS
SELECT customer_id FROM customer ;
PROCEDURE approve_loan
( customer_id CHAR,
loan_type CHAR,
loan_amount REAL) IS
BEGIN
IF loan_amount < 10000 THEN
UPDATE customer
SET loan_type = 'A'
WHERE cust_id = customer_id;
END IF;
END approve_loan;
PROCEDURE cumulative_loan (loan_amount REAL) IS
BEGIN
UPDATE total_outstanding SET loan_amt = loan_amt +
loan_amount;
END cumulative_loan;
END approve_loan ;.
```

CREATE ROLE
参见：
ALTER ROLE,
DROP ROLE， GRANT
语法：
```
CREATE ROLE role
[ NOT IDENTIFIED | IDENTIFIED
[ BY PASSWORD | EXTERNALLY ] ]
```
变量：
role: 角色名

例子：
SQL
```
CREATE ROLE custom_user ;
```

CREATE ROLLBACK SEGMENT

参见：
ALTER ROLLBACK SEGMENT,
DROP ROLLBACK SEGMENT
语法：
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
[TABLESPACE tablespace]
[STORAGE storage]
变量：
rollback_segment 回：滚段名
tablespace：表空间名
例子：
SQL
CREATE ROLLBACK SEGMENT RB_HUMANRESOURCE STORAGE (INITIAL 50K
NEXT 100K OPTIMAL 150K);

CREATE SCHEMA
参见：
CREATE TABLE， DROP SCHEMA
语法：
CREATE SCHEMA AUTHORIZATION schema
[CREATE TABLE command |
CREATE VIEW command |
GRANT command]
变量：
schema: 模式名

例子：
SQL
CREATE SCHEMA AUTHORIZATION cust_schema
CREATE TABLE customer
(CUSTOMER_ID NUMBER(4) NOT NULL,
CUSTOMER_NAME VARCHAR2(50) NOT NULL,
STREET_ADDRESS VARCHAR2(50) NOT NULL,
CITY VARCHAR2(50) NOT NULL,
STATE VARCHAR2(2) NOT NULL CHECK ('FL'， 'TX'， 'MD')
CUST_TYPE VARCHAR2(1) NOT NULL
LOAN_AMOUNT NUMBER(6)) |
CREATE VIEW loan.view AS
SELECT customer_id， customer_name， loan_amount FROM customer
WHERE
customer_type = 'L' ;

CREATE SYNONYM
参见：

CREATE DATABASE LINK,
CREATE TABLE， CREATE VIEW
语法：
CREATE [PUBLIC] SYNONYM [user.] synonym
FOR [user.] table [@database_link]
变量：
user：建立同义词的用户
table：建立同义词的表
database_link：远程数据库连接名

例子：
SQL
CREATE SYNONYM r_cust FOR CUSTOMER@REMOTE_SITE;

CREATE TABLE
参见：
ALTER TABLE,
DROP TABLE
语法：

```
CREATE TABLE [schema.]table
      [ ( { column datatype[DEFAULT expr]  [WITH ROWID]
                  [SCOPE IS [user.]scope_table_name][column_constrint]...
      | table_constraint | REF (ref_column_name)WITH ROWID
      | SCOPE FOR (ref_column_name) IS [user.]scope_table_name }

      [ , { column datatype[DEFAULT expr]  [WITH ROWID]
                  [SCOPE IS [user.]scope_table_name][column_constrint]...
      | table_constraint | REF (ref_column_name)WITH ROWID
      | SCOPE FOR (ref_column_name) ISuser.]scope_table_name}]...)]
[{[ORGANIZATION{HELP |INDEX}
            |PCTTHRESHOLD|INCLUDEING column_name]
   [OVERFLOW[physical_sttrbutes_clause|TABLESPACE tablespace]...]
   |hpysical_attributes_clause|TABLESPACE tablespace
                      |LOB(lob_item[, lob_item...])STORE AS
                      [lob_segname]
                     [(TABLESPACE tablespace |STORAGE storage
                        |CHUNK integer | PCTVERSION integer
                        |CACHE
                        | NOCACHE LOGGING|NOCACHE NOLOGGING
                        |INDEX[lob_index_name]
                            [(TABLESPACE tablespace
                               |STORAGE storage
                               |INITITRANS integer
```

```
                                |MAXTRANS integer )...])]
                        |NESTED TABLE nested_item STORE AS  storage_table
                        |{LOGGING|NOLOGGING}]...
                  |CLUSTER cluster(column[, column]...)}]
                  [PARALLEL parallel_clause ]
                  [PARTITION BY RANGE (column_list)
                      (PARTITION  [partition_name]  VALUE  LESS  THAN  (value_list)
[physical_attributes-clause
                        |TABLESPACE tablespace
                        |{LOGGING|NOLOGGING}]...
           [ENABLE enable_clause|DISABLE disable_clause]...
              [AS subquery]
              [CACHE|NOCACHE]


Physical_attributes_clause ::=

           [PCTFREE integer | PCTUSED integer
            |INITRANS integer | MAXTRANS integer
            |STORAGE storage_clause]
```

变量：
user：建立表的用户名
table：表名
column：列（字段）名
expn： DEFAULT 值
column_constraint：列的完整性限制

column_constraint 可以是:

NULL   表示列可以空
NOT NULL 表示列不能空
UNIQUE   列指定为唯一键
PRIMARY KEY 列指定为主键
FOREIGN KEY 列指定为外部键
REFERENCES 表示被引用的是主键
ON DELETE CASCASE 当主键表被删除时外部表自动被删除
CHECK 指定每行的条件
Cluster：   希望建立表的簇名
N：   任意正整数
Query：   SELECT 子查询

例子：
SQL

```
CREATE TABLE customer
(CUSTOMER_ID NUMBER(4) NOT NULL,
CUSTOMER_NAME VARCHAR2(50) NOT NULL,
STREET_ADDRESS VARCHAR2(50) NOT NULL,
CITY VARCHAR2(50) NOT NULL,
STATE VARCHAR2(2) NOT NULL CHECK ('FL',  'TX',  'MD') ;.
```

CREATE TABLESPACE

参见：

ALTER TABLESPACE

语法：

```
CREATE TABLESPACE tablespace
DATAFILE file_definition [, file_definition] |
[DEFAULT STORAGE storage ]
[ ONLINE | OFFLINE ]
```

变量：

tablespace：表空间名

file_definition：文件大小的定义（以 K 或 M）

例子：

SQL

```
CREATE TABLESPACE customer
(DATAFILE 'customer02.dat');
```

CREATE TRIGGER

参见：

ALTER TRIGGER, DROP TRIGGER

语法：

```
CREATE TRIGGER [user.]trigger
{BEFORE | AFTER }
{DELETE | INSERT | UPDATE [ OF column1 [,  column2] ...}
[ OR {DELETE | INSERT | UPDATE [ OF column1 [,  column2]
...} ...] ON [user.]table
[ REFERENCING { OLD [AS] old | NEW [AS] new } ]
[ FOR EACH ROW]
[WHEN (when_condition) ]
block
```

变量：

user：建立触发器的用户名

trigger：触发器名

column：触发器的列名

table：定义触发器的表名

old：引用表中的旧值

new：引用表中的新值

when_condition：触发器条件
block：要执行的 PL/SQL 块

例子：
SQL
```
CREATE OR REPLACE TRIGGER cumulative_loan_update
AFTER INSERT OR UPDATE OF loan_amount
ON customer
FOR EACH ROW
WHEN (new. loan_type = 'A')
BEGIN
UPDATE total_outstanding SET loan_amt = loan_amt +
new.loan_amount;
END;.
```

CREATE USER
参见：
ALTER USER， DROP USER
语法：
```
CREATE USER user [IDENTIFIED [BY password |
EXTERNALLY]]
[ DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {n [K | M] | UNLIMITED} ON tablespace]
[PROFILE profile]
```
变量：
user：用户名
tablespace：表空间名
n：任意正整数
profile：用户所挂接的 profile 名

例子：
SQL
```
CREATE USER supervisor identified by master
DEFAULT tablespace cust_tablespace
TEMPORARY tablespace temp;.
```

EXPLAIN PLAN
参见：
ANALYZE
语法：
```
EXPLAIN PLAN
[SET STATEMENT ID = statement_name]
```

[INTO [user.]table]
FOR query
变量：
statement_name：输出表中一个标识说明方案的名字。如果没有给定，将在输出表中存放一个空值。
User：建立表的用户名
Table：计划存储的表名
Query：计划解释的 SQL 语句
例子：
SQL
EXPLAIN PLAN
SET STATEMENT_ID = 'exp_plan_customer
INTO all_plan
FOR
SELECT CUSTOMER_ID, CUSTOMER_NAME FROM CUSTOMER WHERE
CUSTOMER_ID IN (SELECT CUSTOMER_ID FROM LOAN_APPROVAL);.

Filespec
参见：
CLOSE, OPEN, OPEN-FOR
目的：
定义一个文件作为数据文件或定义一个或多个文件组作为一个日志文件组

GRANT
参见：
REVOKE
语法：
GRANT system_privilege | role TO user | role | PUBLIC
[WITH ADMIN OPTION]
GRANT object_privilege | ALL column ON schema.object
FROM user | role | PUBLIC WITH GRANT OPTION
变量：
system_privilege：授与用户或角色的系统权限名
role：授个用户的角色名
user：用户或角色名
object_privilege：对象的权限名字，可以是：

ALTER
DELETE
EXECUTE
INDEX
INSERT
REFERENCES

SELECT
UPDATE

Column：列名
Schema：模式名
Object：对象名字
例子：
SQL
GRANT CREATE TABLE TO gavaskar;
GRANT team_leader TO crystal;
GRANT INSERT, UPDATE ON sales TO larry WITH GRANT
OPTION;
GRANT ALL TO PUBLIC;.

NOAUDIT
参见：
AUDIT
语法：
NOAUDIT statement | system_privilege BY user [WITH
GRANT OPTION] [WHENEVER [NOT] SUCCESSFUL]
NOAUDIT object_operating ON schema.object [WHENEVER
[NOT] SUCCESSFUL]
变量：
statement:：你想停止检查的语句
system_privilege:：你想停止检查的系统权限
user:：你想停止检查的用户
例子：
PL/SQL
NOAUDIT SELECT TABLE BY john;
NOAUDIT CREATE TABLE BY martha;
NOAUDIT SELECT ON loanschema.loan WHENEVER SUCCESSFUL;.

TRUNCATE
DELETE, DROP TABLE
语法：
TRUNCATE [TABLE | CLUSTER]
schema.[table][cluster] [DROP | REUSE STORAGE]
变量：
table： 希望删除行的表名
cluster：希望删除行的簇名
schema：表或 簇的模式（schema）名
例子：
SQL

```
TRUNCATE TABLE inventory;
TRUNCATE CLUSTER marketing DROP STORAGE;
TRUNCATE TABLE loan REUSE STORAGE;
```

附录 E　ORACLE 系统权限

下面以字母顺序给出 Oracle8i 数据库系统的系统权限。

权　　限	描　　述
CREATE SESSION	允许用户联到 ORACLE 数据库，用户可访问 ORACLF.
FORCE TRANSACTION	允许用户在本地数据库中提交或回滚分布数据库事务.一般不用设置该权限.
CREATE CLUSTER	创建属于开发者自己的表聚簇，开发者也能撤消他们拥有的聚簇
CREATE PROCEDURE	创建属于开发者的存储过程，软件包和函数.开发者也能撤消他们所拥有的这些对象.
CREATE DATABASE LINK	定义一个数据库连接，因为这是一个命名的指向其它数据库的指针，所以这个特性类一个同义词，主要差别是可以存储远程系统中确立 ORACLE id 和口令作连接的一部分.
CREATE PUBLIC SYNONYM	为了引用一个诸如表或视图的数据库对象所创建的一个替代名，实例中的任何用户都能使用这个名称调用它所代表的对象，用户要访问对象仍需要对象权限.
DROP PUBLIC SYNONYM	为了引用数据库对而删除替代名称，该数据库对象可实例中的所有用户使用.
CREATE SEQUENCE	创建一个开发者所有的序列，开发者也能撤消任何他们建立序列.
CREATE SNAPSHOTP	创建一个位于另一个 ORACLE 实例中的表的本地考备，开发者也能撤消他们拥有的快照.
CREATE SYNONYM	创建一个专用的同义词(仅供开发者使用)，开发者也能撤消他们拥有的任何同义词.
CREATE TABLE	开发者可以创建表和删除表.
CREATE TRIGGER	开发者可以创建或删除他们拥有的触发器
CREATE VIEW	开发者可以创建或删除他们拥有的视图。
UNLIMITED TABLESPACE	允许开发者在表空间中创建对象而不受表空间大小限制。
CREATE TYPE	允许开发者创建新的对象类型。
DROP TYPE	允许开发者删除对象类型。
CREATE LIBRARY	允许开发者创建新的对象库。
DROP LIBRARY	允许开发者删除对象库。
ANALYZE ANY	允许用户收集最优化统计，使结构有效或识别在数据库的任何表，表聚簇中被移动和被链接的行.
ADUIT ANY	允许用户对数据库中的任何对象进行审计.
CREATE ANY CLUSTER	允许用户创建聚簇，并给数据库中的任何用户赋予所有权.
ALTER ANY CLUSTER	允许用户改变数据库中任何用户的聚簇.

DROP ANY CLUSTER	允许用户删除数据库中任何用户的聚簇.
CREATE ANY INDEX	许用户数据库中任何表建立索引，并给数据库中任何用户授予所有权.
ALTER ANY INDEX	允许用户改变数据库中任何用户的索引.
DROP ANY INDEX	允许用户删除数据库中任何用户的索引.
GRANT ANY PRIVILEGE	允许用户将数据库中任何权限授予任何用户.这是 DBA 授予系统的基本要求.注意:本权限并不包括用户授予对象的权限，只有对象的所有者才能授予对象的权限.
CREATE ANY PROCEDURE	允许用户创建过程，软管件包或函数，并给数据库中等任何用户赋予所有权．拥有这个权限首先拥有 ALTER ANY TABLE，BACKUP ANY TABLE，DROP ANY TABLE，LOCK ANY TABLE，COMMENT ANY TABLE，SELECT ANY TABLE，DELETE ANY TABLE 或 GRANT ANY TABLE 中等某些权限，具体有那些权限取决于实际要做什么工作.
ALTER ANY PROCEDURE	允许用户修改数据库中任何用户的任何拥有的过程软件包或函数.
DROP ANY PROCEDURE	允许用户删除数据库中任何用户的任何拥有的过程软件包或函数.
EXECUTE ANY PROCEDURE	允许用户执行数据库中任何用户的任何拥有的过程软件包或函数.这个权限超越赋予所有者过程，软件包，或函数的对象权限.
ALTER ANY ROLE	允许用户修改在数据库中创建的任何角色.
DROP ANY ROLE	允许用户删除在数据库中创建的任何角色.
GRANT ANY ROLE	允许用户角色授予数据库中的另外用户.
CREATE ANY SEQUENCE	允许用户在数据库中创建的序列并给数据库中的任何用户赋予所有权.
ALTER ANY SEQUENCE	允许用户修改数据库中任何用户拥有的序列.
DROP ANY SEQUENCE	允许用户删除数据库中任何用户拥有的序列.
SELECT ANY SEQUENCE	允许用户使用数据库中任何用户拥有的序列.
CREATE ANY SNAPSHOT	允许用户创建另一个实例中的表的本地考备，并给数据库中任何用户赋予所有权.这个用户必须有 CREATE ANY TABLE 的权限。
ALTER ANY SNAPSHOT	允许用户编译数据库中任何用户拥有的快照.
DROP ANY SNAPSHOT	允许用户删除数据库中任何用户拥有的快照.
CREATE ANY SYNONYM	允许用户创建专有的同义词并给数据库中任何用户赋予所有权.
DROP ANY SYNONYM	允许用户删除数据库中任何用户拥有的同义词.
CREATE ANY TABLE	允许用户创建表并给数据库中任何用户赋予所有权.
ALTER ANY TABLE	允许用户修改数据库中任何用户拥有的表的结构.
DROP ANY TABLE	允许用户删除数据库中任何用户拥有的表.
LOCK ANY TABLE	允许用户锁住数据库中任何用户拥有的表(或表中的行).
COMMENT ANY TABLE	允许用户对数据库中任何用户拥有的表加注释.
SELECT ANY TABLE	允许用户查询数据库中任何用户拥有的表.
INSERT ANY TABLE	允许用户向数据库中任何用户拥有的表插入新行.
UPDATE ANY TABLE	允许用户更新数据库中任何用户拥有的表.
DELETE ANY TABLE	允许用户删除数据库中任何用户拥有的表的记录.
FORCE ANY TRANSACTION	允许用户提交或回滚与数据中任何用户有关的分布式数据库事务.
CREATE ANY TRIGGER	允许用户创建触发器并给数据库中任何用户赋予所有权.

ALTER ANY TRIGGER	允许用户修改(使起作用或不起作用或重新编译)数据库中任何用户拥有的触发器.
DROP ANY TRIGGER	允许用户删除数据库中任何用户拥有的触发器.
CREATE ANY VIEW	允许用户创建视图，并给数据库的任何用户赋予所有权.创建者必须拥有 ALTER ANY TABLE，BACKUP ANY TABLE，DROP ANY TABLE，LOCK ANY TABLE，COMMENT ANY TABLE，SELECT ANY TABLE，INSERT ANY TABLE，UPDATE ANY TABLE 或 DELETE ANY TABLE 中的某些权限.具体有哪些权限取决于用户要创建的视图.当用户要访问一个未被授权的表的时候，上面的权限首先要起作用.
DROP ANY VIEW	允许用户删除数据库中任何用户拥有的视图.
CREATE ANY TYPE	允许用户在数据库中创建一个用户类型.
DROP ANY TYPE	允许用户在数据库中删除一个用户类型.
CREATE ANY LIBRARY	允许用户在数据库中创建一个库.
DROP ANY LIBRARY	允许用户删除数据库中一个库.
ALTER DATABASE	发出 alter database 命令，包括安装，打开数据库，管理日志文件和控制文件以及改变归档日志状态等.
CREATE PROFILE	允许用户创建配置文件,在配置文件中可以设置某些 ORACLE 资源使用得限制(度).
ALTER PROFILE	允许用户修改配置文件.
DROP PROFILE	允许用户删除配置文件.
ALTER RESOURCE COST	当数据库跟踪资源成本时，允许用户资源成本.
CREATE PUBLIC DATABASE LINK	允许用户创建对别的 ORACLE 实例的链接，这个 ORACLE 实例能被所有的用户访问.
DROP PUBLIC DATABASE LINK	允许用户删除公共的数据库链接.
CREATE ROLE	允许用户创建角色.
CREATE ROLLBACK SEGMENT	允许用户在表空间上创建回滚段.
ALTER ROLLBACK SEGMENT	允许用户在表空间上改变回滚段的结构.
DROP ROLLBACK SEGMENT	允许用户在表空间上删除回滚段.
ALTER SYSTEM	允许用户发出 alter system 命令，改命令用来切换日志文件，检查数据文件，设置某些系统参数，切断对 ORACLE 的连接以及其它类似的功能等.
CREATE TABLESPACE	允许用户创建新的表空间.注意:该 ORACLE 用户必须有访问操作系统的权限以及有足够的盘空间.
ALTER TABLESPACE	允许用户修改已建好的表空间.包括增加数据文件等.
MANAGE TABLESPACE	允许用户对表空间执行热备份，以及将表空间在联机联机之间进行切换.
BECOME USER	切换成数据库的另一用户，这仅用于整个数据库的导入和输出.在 SQL*PLUS 提示下此命令无效.
ALTER USER	允许改变任何用户的配置(包括改变口令).
DROP USER	允许从 ORACLE 中删除用户.
ADUIT SYSTEM	允许对系统进行审计.

附录 F　ORACLE 系统常用数据字典

下面按照类别给出 Oracle8I 数据库系统的常用数据字典。

F.1　对象、表、视图、同义词、序列

DBA_OBJECTS(实例中的对象)
存放的对象包括：CLUSTER、DATABASE_LINK、FUNCTION，INDEX，PACKAGE，
　　PROCEDURE，SEQUENCE，SYNONYM，TABLE，TRIGGER，VIEW，
　　TYPE，DIRECTORY 等。

列名	说明
OWNER	VARCHAR2(30)　对象主人
OBJECT_NAME	VARCHAR2(128)　对象名
SUBOBJECT_NAME	VARCHAR2(30)　字对象名，如分区
OBJECT_ID	NUMBER　对象的标识
DATA_OBJECT_ID	NUMBER　对象的对象数
OBJECT_TYPE	VARCHAR2(18)　对象的类型，如 INDEX
CREATED	DATE　建立对象的日期及时间
LAST_DDL_TIME	DATE　最后一次执行 DDL 的时间(含 grants revokes)
TIMESTAMP	VARCHAR2(19)　对象生成的日期及时间
STATUS	VARCHAR2(7)　对象的状态： VALID， INVALID
TEMPORARY	VARCHAR2(1)　标识为临时表
GENERATED	VARCHAR2(1)　标识对象是否为系统生成
SECONDARY	VARCHAR2(1)　是否为第 2 次建立演示索引

DBA_TABLES (实例中的表)

列名	说明
OWNER	NOT NULL VARCHAR2(30) 表的所有者
TABLE_NAME	NOT NULL VARCHAR2(30) 表名
TABLESPACE_NAME	VARCHAR2(30) 包含表的表空间; NULL 为分区表
CLUSTER_NAME	VARCHAR2(30) 表所属的 CLUSTER 名
IOT_NAME	VARCHAR2(30) 表的索引结构名
PCT_FREE	NUMBER 块中自由空间百分比，NULL 为分区表
PCT_USED	NUMBER 块中使用空间百分比，NULL 为分区表
INI_TRANS	NUMBER 初始事务如口数 ，NULL 为分区表
MAX_TRANS	NUMBER 最大事务如口数 ，NULL 为分区表

INITIAL_EXTENT NUMBER 初始分配字节数，NULL 为分区表
NEXT_EXTENT NUMBER 下次扩展字节数，NULL 为分区表
MIN_EXTENTS NUMBER 最小扩展次数，NULL 为分区表
MAX_EXTENTS NUMBER 最大扩展次数，NULL 为分区表
PCT_INCREASE NUMBER 相对上次增长百分比，NULL 为分区表
FREELISTS NUMBER 段的自由分配列表数，NULL 为分区表
FREELIST_GROUPS NUMBER 组的自由分配列表数，NULL 为分区表
LOGGING VARCHAR2(3) 登录属性，NULL 为分区表
BACKED_UP VARCHAR2(1) 上次改变以来备份标记
NUM_ROWS NUMBER 表中的行数
BLOCKS NUMBER 表中的块数
EMPTY_BLOCKS NUMBER 表中分配但未使用的块数
AVG_SPACE NUMBER 表中分配但未使用的平均数
CHAIN_CNT NUMBER 表中产生行连接(从一个块到另一各块)的数
AVG_ROW_LEN NUMBER 平均行的字节数
AVG_SPACE_FREELIST_BLOCKS NUMBER 自由列表中所有块的平均自由空间
NUM_FREELIST_BLOCKS NUMBER 自由列表的块数
DEGREE VARCHAR2(10) 对于查询表的每个实例线程数
INSTANCES VARCHAR2(10) 表中事务交叉数
CACHE VARCHAR2(5) 是否为缓存表
TABLE_LOCK VARCHAR2(8) 是否使能锁
SAMPLE_SIZE NUMBER 分析的样例大小

LAST_ANALYZED DATE 最后分析的时间
PARTITIONED VARCHAR2(3) 表是否为分区表
IOT_TYPE VARCHAR2(12) 如果为索引结构表， 则 IOT_TYPE 为
IOT 或 IOT_OVERFLOW 否则为 NULL
TEMPORARY VARCHAR2(1) 仅是当前会话使用？
SECONDARY VARCHAR2(1) 是否为第 2 次建立演示索引
NESTED VARCHAR2(3) 是否嵌套表
BUFFER_POOL VARCHAR2(7) 缺省缓冲区的名字， NULL 为分区表
ROW_MOVEMENT VARCHAR2(8) 分区时行是否移动
GLOBAL_STATS VARCHAR2(3) 没有合并的统计计算？
USER_STATS VARCHAR2(3) 用户是否统计过？
DURATION VARCHAR2(15) 如果是临时表，则在 sys$session 或
sys$transaction 中的持续时间
SKIP_CORRUPT VARCHAR2(8) 是否跳过冲突块（使能或不使能）
MONITORING VARCHAR2(3) 监视
CLUSTER_OWNER VARCHAR2(30) 簇的主人

DBA_VIEWS (实例中的视图)

列名 说明

OWNER	VARCHAR2(30)	视图的主人
VIEW_NAME	VARCHAR2(30)	视图的名字
TEXT_LENGTH	NUMBER	视图的文本长度
TEXT	LONG	视图的文本
TYPE_TEXT_LENGTH	NUMBER	类型视图的子句长度
TYPE_TEXT	VARCHAR2(4000)	类型视图的子句
OID_TEXT_LENGTH	NUMBER	类型视图的 WITH OID 子句长度
OID_TEXT	VARCHAR2(4000)	类型视图的 WITH OID 子句
VIEW_TYPE_OWNER	VARCHAR2(30)	类型视图的主人
VIEW_TYPE	VARCHAR2(30)	类型视图

DBA_SYNONYMS(实例中的同义词)

列名	说明	
OWNER	VARCHAR2(30)	同义词的主人
SYNONYM_NAME	VARCHAR2(30)	同义词的名字
TABLE_OWNER	VARCHAR2(30)	表的主人
TABLE_NAME	VARCHAR2(30)	表的名字
DB_LINK	VARCHAR2(128)	数据库连接名

DBA_SEQUENCES(实例中的序列)

列名	说明	
SEQUENCE_OWNER	VARCHAR2(30)	序列的主人
SEQUENCE_NAME	VARCHAR2(30)	序列名字
MIN_VALUE	NUMBER	最小值
MAX_VALUE	NUMBER	最大值
INCREMENT_BY	NUMBER	增加步长
CYCLE_FLAG	VARCHAR2(1)	循环标记
ORDER_FLAG	VARCHAR2(1)	顺序标记
CACHE_SIZE	NUMBER	缓存大小
LAST_NUMBER	NUMBER	最后的序列值

F.2 索引、Cluster 及约束（constraints）

DBA_INDEXES (实例中索引)

列名	说明	
OWNER	VARCHAR2(30)	索引主人
INDEX_NAME	VARCHAR2(30)	索引名字
INDEX_TYPE	VARCHAR2(27)	索引类型
TABLE_OWNER	VARCHAR2(30)	表的主人
TABLE_NAME	VARCHAR2(30)	表的名字
TABLE_TYPE	VARCHAR2(11)	表的类型
UNIQUENESS	VARCHAR2(9)	是否唯一索引
COMPRESSION	VARCHAR2(8)	是否压缩
PREFIX_LENGTH	NUMBER	前缀长度
TABLESPACE_NAME	VARCHAR2(30)	表空间名
INI_TRANS	NUMBER	初始事务数
MAX_TRANS	NUMBER	最大事务数
INITIAL_EXTENT	NUMBER	初始扩展大小
NEXT_EXTENT	NUMBER	下一次扩展大小
MIN_EXTENTS	NUMBER	最小扩展次数
MAX_EXTENTS	NUMBER	最大扩展次数
PCT_INCREASE	NUMBER	相对前一次的增长百分比
PCT_THRESHOLD	NUMBER	块空间开始分配百分比
INCLUDE_COLUMN	NUMBER	包括的列数
FREELISTS	NUMBER	自由列表数
FREELIST_GROUPS	NUMBER	自由列表组数
PCT_FREE	NUMBER	块中用于更新的百分比
LOGGING	VARCHAR2(3)	日志信息
BLEVEL	NUMBER	B*树从根到枝的索引深度
LEAF_BLOCKS	NUMBER	索引中叶块的数量
DISTINCT_KEYS	NUMBER	不同键的数目
AVG_LEAF_BLOCKS_PER_KEY	NUMBER	每个键叶块的平均数
AVG_DATA_BLOCKS_PER_KEY	NUMBER	每个键数据块的平均数
CLUSTERING_FACTOR	NUMBER	基表行的排序数，如果该值： * 接近块的大小，则表非常容易排序 * 解决行的大小，则容易随机排序
STATUS	VARCHAR2(8)	状态
NUM_ROWS	NUMBER	行的数量
SAMPLE_SIZE	NUMBER	样本大小
LAST_ANALYZED	DATE	最后分析时间
DEGREE	VARCHAR2(40)	扫描索引时的实例数
INSTANCES	VARCHAR2(40)	实例数
PARTITIONED	VARCHAR2(3)	是否被分区
TEMPORARY	VARCHAR2(1)	是否存放在临时表空间
GENERATED	VARCHAR2(1)	是否是有系统产生索引名

SECONDARY	VARCHAR2(1)	是否是第 2 个对象创建
BUFFER_POOL	VARCHAR2(7)	缓冲区大小
USER_STATS	VARCHAR2(3)	是否有用户统计过
DURATION	VARCHAR2(15)	临时表的为期
PCT_DIRECT_ACCESS	NUMBER	访问百分比
ITYP_OWNER	VARCHAR2(30)	本地索引的主人
ITYP_NAME	VARCHAR2(30)	本地索引的名字
PARAMETERS	VARCHAR2(1000)	本地的参数
GLOBAL_STATS	VARCHAR2(3)	全局统计标记
DOMIDX_STATUS	VARCHAR2(12)	本地索引状态： null-非本地； valid-本地索引； idxtyp-invld:本地索引无效
DOMIDX_OPSTATUS	VARCHAR2(6)	本地索引操作状态： null-非本地； valid-本地索引操作没有错误； failed:本地索引操作有错误
FUNCIDX_STATUS	VARCHAR2(8)	本地索引函数状态： null-非基本函数索引； valid-函数索引可用； failed: 函数索引不可用。

DBA_CLUSTERS(实例中的簇)

列名	说明	
OWNER	VARCHAR2(30)	簇的主人
CLUSTER_NAME	VARCHAR2(30)	簇的名字
TABLESPACE_NAME	VARCHAR2(30)	所在表空间
PCT_FREE	NUMBER	块中用于更新的百分比
PCT_USED	NUMBER	块中释放空间的百分比
KEY_SIZE	NUMBER	键的大小
INI_TRANS	NUMBER	块中的事务初始如口数
MAX_TRANS	NUMBER	块中的事务最大如口数
INITIAL_EXTENT	NUMBER	初始扩展字节数
NEXT_EXTENT	NUMBER	下次扩展字节数
MIN_EXTENTS	NUMBER	最小扩展次数
MAX_EXTENTS	NUMBER	对大扩展次数
PCT_INCREASE	NUMBER	相对上次的增长百分比
FREELISTS	NUMBER	自由列表数（并行）
FREELIST_GROUPS	NUMBER	自由列组数（并行）

AVG_BLOCKS_PER_KEY	NUMBER	每个 Hash 键的块
CLUSTER_TYPE	VARCHAR2(5)	B*树索引或 Hash 索引
FUNCTION	VARCHAR2(15)	如果是 Hash 簇，则为 Hash 函数
HASHKEYS	NUMBER	如果书 Hash 簇，则为 Hash 键
DEGREE	VARCHAR2(10)	并行度
INSTANCES	VARCHAR2(10)	实例数
CACHE	VARCHAR2(5)	缓存数
BUFFER_POOL	VARCHAR2(7)	缓冲区数
SINGLE_TABLE	VARCHAR2(5)	是否为单表的簇

DBA_CONSTRAINTS(实例中约束)

列名		说明
OWNER	VARCHAR2(30)	限制的主人
CONSTRAINT_NAME	VARCHAR2(30)	限制的名字
CONSTRAINT_TYPE	VARCHAR2(1)	限制的类型

- C (在表上检查约束)
- P (主键)
- U (唯一键)
- R (引用完整性)
- V (视图检查)
- O (视图只读检查)

TABLE_NAME	VARCHAR2(30)	表名
SEARCH_CONDITION	LONG	搜索条件
R_OWNER	VARCHAR2(30)	表引用的主人
R_CONSTRAINT_NAME	VARCHAR2(30)	唯一限制名字
DELETE_RULE	VARCHAR2(9)	删除规则：CASCADE 或 NO ACTION
STATUS	VARCHAR2(8)	状态：ENABLED 或 DISABLED
DEFERRABLE	VARCHAR2(14)	限制是否延期
DEFERRED	VARCHAR2(9)	限制初始延期
VALIDATED	VARCHAR2(13)	是否所有数据服从限制： VALIDATED 或 NOT VALIDATED
GENERATED	VARCHAR2(14)	限制是否是系统产生
BAD	VARCHAR2(3)	yes 表示该限制在一个世纪内不理重复， 为了保证正确，要用 4 位年。
RELY	VARCHAR2(4)	是否重新启动限制
LAST_CHANGE	DATE	最后启动或静止限制的日期

F.3 触发器、过程、函数及包

DBA_TRIGGERS(实例中触发器)

列名	说明	
OWNER	VARCHAR2(30)	触发器的主人
TRIGGER_NAME	VARCHAR2(30)	触发器的名字
TRIGGER_TYPE	VARCHAR2(16)	触发器的类型
TRIGGERING_EVENT	VARCHAR2(216)	触发器的事件
TABLE_OWNER	VARCHAR2(30)	表的主人
BASE_OBJECT_TYPE	VARCHAR2(16)	基本对象类型
TABLE_NAME	VARCHAR2(30)	表的名字
COLUMN_NAME	VARCHAR2(4000)	列的名字
REFERENCING_NAMES	VARCHAR2(128)	引用名字
WHEN_CLAUSE	VARCHAR2(4000)	条件子句
STATUS	VARCHAR2(8)	状态
DESCRIPTION	VARCHAR2(4000)	描述
ACTION_TYPE	VARCHAR2(11)	措施类型
TRIGGER_BODY	LONG	触发器的主体内容

DBA_SOURCE(实例中的存储过程、函数、包及包体)：

列名	说明	
OWNER	VARCHAR2(30)	对象的主人
NAME	VARCHAR2(30)	对象的名字
TYPE	VARCHAR2(12)	对象的类型，可以是（不包括 TRIGGER)：PROCEDURE；PACKAGE；FUNCTION；PACKAGE BODY；TYPE；TYPE BODY
LINE	NUMBER	源代码的行数
TEXT	VARCHAR2(4000)	存储对象的源代码

F.4 空间分配与使用

DBA_TABLESPACES(实例中的表空间)

列名	说明

TABLESPACE_NAME	VARCHAR2(30)	表空间名字
INITIAL_EXTENT	NUMBER	初始分配大小
NEXT_EXTENT	NUMBER	下次分配大小
MIN_EXTENTS	NUMBER	最小分配次数
MAX_EXTENTS	NUMBER	最大分配次数
PCT_INCREASE	NUMBER	相对于上次增长百分比
MIN_EXTLEN	NUMBER	最小扩展长度
STATUS	VARCHAR2(9)	状态：ONLINE 或 OFFLINE 或 READ ONLY
CONTENTS	VARCHAR2(9)	内容:PERMANENT 或 TEMPORARY
LOGGING	VARCHAR2(9)	日志属性（缺省为进行日志）
EXTENT_MANAGEMENT	VARCHAR2(10)	扩展管理跟踪：DICTIONARY 或 LOCAL
ALLOCATION_TYPE	VARCHAR2(9)	扩展分配类型
PLUGGED_IN	VARCHAR2(3)	YES 表示表空间为追加

DBA_SEGMENTS(实例中的段)

列名	说明	
OWNER	VARCHAR2(30)	段的主人
SEGMENT_NAME	VARCHAR2(81)	段的名字
PARTITION_NAME	VARCHAR2(30)	分区名字
SEGMENT_TYPE	VARCHAR2(18)	段类型：
TABLESPACE_NAME	VARCHAR2(30)	表空间名字
HEADER_FILE	NUMBER	段的头部的文件 ID 号
HEADER_BLOCK	NUMBER	段的头部的块 ID 号
BYTES	NUMBER	分配的字节数
BLOCKS	NUMBER	分配的块数
EXTENTS	NUMBER	扩展的次数
INITIAL_EXTENT	NUMBER	初始分配的字节数
NEXT_EXTENT	NUMBER	下次扩展的字节数
MIN_EXTENTS	NUMBER	最小扩展的次数
MAX_EXTENTS	NUMBER	最大扩展的次数
PCT_INCREASE	NUMBER	增长的百分比
FREELISTS	NUMBER	自由列表数（并行）
FREELIST_GROUPS	NUMBER	自由列组数（并行）
RELATIVE_FNO	NUMBER	相关的文件号
BUFFER_POOL	VARCHAR2(7)	缺省底册缓冲区大小

DBA_EXTENTS(实例中段的扩展)

列名	说明	
OWNER	VARCHAR2(30)	段的主人
SEGMENT_NAME	VARCHAR2(81)	段的名字
PARTITION_NAME	VARCHAR2(30)	分区的名字
SEGMENT_TYPE	VARCHAR2(18)	段的类型
TABLESPACE_NAME	VARCHAR2(30)	表空间的名字
EXTENT_ID	NUMBER	扩展 ID 号
FILE_ID	NUMBER	文件的 ID 号
BLOCK_ID	NUMBER	块的 ID 号
BYTES	NUMBER	字节数
BLOCKS	NUMBER	块数
RELATIVE_FNO	NUMBER	相关的文件号

DBA_FREE_SPACE(实例中自由空间)

列名	说明	
TABLESPACE_NAME	VARCHAR2(30)	表空间名字
FILE_ID	NUMBER	文件号
BLOCK_ID	NUMBER	块的号
BYTES	NUMBER	字节数
BLOCKS	NUMBER	块数
RELATIVE_FNO	NUMBER	相关的文件号

F.5 用户、权限与角色

DBA_USERS(实例中用户)

列名	说明	
USERNAME	VARCHAR2(30)	用户名
USER_ID	NUMBER	用户标识(系统给的编号)
PASSWORD	VARCHAR2(30)	口令（显示进行力偶啊转换）
ACCOUNT_STATUS	VARCHAR2(32)	状态
LOCK_DATE	DATE	锁的日期

EXPIRY_DATE	DATE	过期的日期
DEFAULT_TABLESPACE	VARCHAR2(30)	缺省的表空间
TEMPORARY_TABLESPACE	VARCHAR2(30)	临时表空间
CREATED	DATE	创建日期
PROFILE	VARCHAR2(30)	资源文件名
INITIAL_RSRC_CONSUMER_GROUP	VARCHAR2(30)	初始资源消费组
EXTERNAL_NAME	VARCHAR2(4000)	外部名字

DBA_TAB_PRIVS(实例中用户可以访问表的权限)

列名	说明
GRANTEE	VARCHAR2(30) 被授权者
OWNER	VARCHAR2(30) 主人
TABLE_NAME	VARCHAR2(30) 表名
GRANTOR	VARCHAR2(30) 授权者
PRIVILEGE	VARCHAR2(40) 权限
GRANTABLE	VARCHAR2(3) 可再授权给别人

DBA_SYS_PRIVS(实例中用户被授的系统权限)

列名	说明
GRANTEE	VARCHAR2(30) 被授权者
PRIVILEGE	VARCHAR2(40) 权限名称
ADMIN_OPTION	VARCHAR2(3) 是否带 ADMIN

附录 G　ORACLE 支持的字符集

G.1　理解 Oracle NLS

Oracle 的民族语言支持（Oracle's National Language Support）允许用户以本民族语言存储、处理和检索数据。并可以在错误信息、排序、日期、货币、数字及日历进行自动转换。本地有关的操作与 ORA_NLS 环境变量有关：

Oracle 在 ORA_NLS 环境变量中指定本地运行库路径。不同的版本设置如下：

- Oracle7.2　　ORA_NLS
- Oracle7.3　　ORA_NLS32
- Oracle8,Oracle8I　　ORA_NLS33

通过统一字符编码（Unicode）--即 UTF8，Oracle 可以支持大部分的语言（见“ Language Support ”）。支持的国家和地区的语言如下表：

Oracle8I 允许用户以本民族语言存储、处理和 检索数据。可以支持的语言有：

American English *	English	Italian *	Russian *
Arabic *	Estonian	Japanese *	Simplified Chinese *
Bengali	Finnish *	Korean *	Slovak *
Brazilian Portuguese *	French *	Latin American Spanish *	Slovenian
Bulgarian	German *	Latvian	Spanish *
Canadian French	German Din	Lithuanian	Swedish *
Catalan *	Greek *	Malay	Tamil
Croatian	Hebrew *	Mexican	Spanish Thai
Czech *	Hindi	Norwegian *	Traditional Chinese *
Danish *	Hungarian *	Polish *	Turkish *
Dutch *	Icelandic	Portuguese *	Ukrainian
Egyptian	Indonesian	Romanian *	Vietnamese

表G-1 地区支持的字符集

G.2　设置 NLS 参数

可以有以下几种方法来设置 NLS 参数：

1.在 initsid.ora 文件中设置：

```
NLS_TERRITORT="CZECH REPUBLIC"
```

2.设置环境变量：

```
$setenv NLS_SORT FRENCE
```

3.在 SQL>下用 ALTER SESSION 命令设置：

```
SQL>ALTER SESSION SETR NLS_SORT=FRENCE;
```

下面是参数及设置方法：

参　数	说　明	缺　省	在init.ora	在环境变量	用ALTER SESSION

NLS_CALENDAR	日历系统	罗马教皇Gregorian	v	v	v
NLS_COMP	SQL比较二进制库	NLS_TERRITORY	v	v	v
NLS_CREDIT	信用会计符号	NLS_TERRITORY		v	
NLS_CURRENCY	本地货币符号	NLS_TERRITORY	v	v	v
NLS_DATE_FORMAT	日期格式	NLS_TERRITORY	v	v	v
NLS_DATE_LANGUAGE	天和月的名称	NLS_LANGUAGE	v	v	v
NLS_DEBIT	借方会计符号	NLS_TERRITORY		v	
NLS_ISO_CURRENCY	ISO国际货币符号	NLS_TERRITORY	v	v	v
NLS_LANG	语言、本地、字符集	American_America.US7ASCII		v	
NLS_LANGUAGE	语言	NLS_LANG	v		v
NLS_LIST_SEPARATOR	字符分隔符	NLS_TERRITORY		v	
NLS_MONETARY_CHARACTERS	美圆货币符和分币	NLS_TERRITORY		v	
NLS_NCHAR	民主字符集	NLS_LANG		v	
NLS_NUMERIC_CHARACTERS	十进制字符和组分隔符	NLS_TERRITOR	v	v	v
NLS_SORT	字符分类顺序	NLS_LANGUAGE	v	v	v
NLS_TERRITORY	本地	NLS_LANG	v		v
NLS_DUAL_CURRENCY	双重货币符	NLS_TERRITORY	v	v	v

G.3 NLS 视图

- NLS_SESSION_PARAMETERS 显示当前会话的 NLS 参数。
- NLS_INSTANCE_PARAMETERS 显示当前实例的 NLS 参数。
- NLS_DATABASE_PARAMETERS 显示当前数据库的 NLS 参数。
- V$NLS_VALID_VALUES 服务器支持的语言、本地、语言学及字符集的定义。

G.4 亚洲国家和地区字符集

字符集名字	说明	注释
BN8BSCII	Bangladesh 国家代码 8-位 BSCII	SB, ASCII
ZHT16BIG5	BIG5 16-位 传统中文	MB, ASCII
ZHS16CGB231280	CGB2312-80 16-位 简体中文	MB, ASCII
JA16EUC	EUC 24-位 日语	MB, ASCII
JA16EUCYEN	EUC 24-位 带 '\' 的日本语	MB
JA16EUCFIXED	EUC 16-位日本语. JA16EUC 固定宽度的子集	FIXED
ZHT32EUC	EUC 32-位传统中文	MB, ASCII
ZHT32EUCFIXED	EUC 32-位传统中文 (32-位固定宽度, 无符号)	FIXED
ZHS16GBK	GBK 16-位简体中文	MB, ASCII, UDC
ZHS16GBKFIXED	GBK 16-位简体中文 (16-位固定宽度, 无符号)	FIXED, UDC

ZHT16CCDC	HP CCDC 16-位传统中文	MB, ASCII
JA16DBCS	IBM EBCDIC 16-位 日本	MB, UDC
JA16EBCDIC930	IBM DBCS 日本 16 位编码	MB, UDC
JA16DBCSFIXED	IBM EBCDIC 16-位日本语 (16-位固定宽度, 无符号)	FIXED, UDC
KO16DBCS	IBM EBCDIC 16-位 朝鲜语	MB, UDC
KO16DBCSFIXED	IBM EBCDIC 16-位朝鲜语 (16-位固定宽度, 无符号)	FIXED, UDC
ZHS16DBCS	IBM EBCDIC 16-位简体中文	MB, UDC
ZHS16CGB231280 FIXED	CGB2312-80 16-位简体中文(16-位固定宽度, 无符号)	FIXED
ZHS16DBCSFIXED	IBM EBCDIC 16-位简体中文 (16-位固定宽度, 无符号)	FIXED, UDC
ZHT16DBCS	IBM EBCDIC 16-位传统中文	MB, UDC
ZHT16DBCSFIXED	IBM EBCDIC 16-位传统中文 (16-位固定宽度, 无符号)	FIXED
KO16KSC5601	KSC5601 16-位朝鲜语	MB, ASCII
KO16KSCCS	KSCCS 16-位 朝鲜语	MB, ASCII
KO16KSC5601FIXED	KSC5601 (16-位 固定宽度, 无符号)	FIXED
JA16VMS	JVMS 16-位日本语	MB, ASCII
ZHS16MACCGB231280	Mac client CGB2312-80 16-位简体中文	MB
JA16MACSJIS	Mac client Shift-JIS 16-位日本语	MB
TH8MACTHAI	Mac Client 8-位 拉丁语/泰国语	SB
TH8MACTHAIS	Mac Server 8-位 拉丁语/泰国语	SB, ASCII
TH8TISEBCDICS	泰国工业标准 620-2533-EBCDIC Server 8-位	SB
ZHT16MSWIN950	MS Windows 传统中文编码	MB, ASCII, UDC
KO16MSWIN949	MS Windows 朝鲜语编码	MB, ASCII, UDC
VN8MSWIN1258	MS Windows 越南编码	SB, ASCII, EURO
IN8ISCII	多脚本的印度标准 8-位拉丁语/印度语	SB, ASCII
JA16SJIS	变化-JIS 16-位 日本语	MB, ASCII, UDC
JA16SJISFIXED	变化-JIS 16-位日本语. JA16SJIS 固定宽度的子集	FIXED, UDC
JA16SJISYEN	变化-JIS 16-位日本语 (带 '\')	MB, UDC
ZHT32SOPS	SOPS 32-位传统中文	MB, ASCII
ZHT16DBT	中国台湾 16-位 传统中文	MB, ASCII
ZHT16BIG5FIXED	BIG5 16-位传统中文 (16-位固定宽度, 无符号)	FIXED
TH8TISASCII	泰国语工业标准 620-2533 - ASCII 8-位	SB, ASCII, EURO
TH8TISEBCDIC	泰国语工业标准 620-2533 - EBCDIC 8-位	SB
ZHT32TRIS TRIS	32-位传统中文	MB, ASCII
ZHT32TRISFIXED	TRIS 32-位固定宽度传统中文	FIXED
AL24UTFFSS	统一编码 1.1 版的 UTF8 通用字符集	MB, ASCII, EURO
UTF8	统一编码 2.1 版的 UTF8 通用字符集	MB, ASCII, EURO
UTFE	UTF-EBCDIC 编码字符集。只能在 EBCDIC 编码的机器上运行, 如 IBM 机器。	
VN8VN3	VN3 8-位 越南语	SB, ASCII

其中：
SB - 单字节编码；
MB-多字节便帽；
ASCII-是 Aerican Standard Code for Information Interchange 的缩写，即“美国信息交换标准编码”；
EBCDIC - 是扩展编码的十进制编码；
EURO-欧洲；
UDC - 是 User-defined Character Support 的缩写。即“用户定义字符集支持”编码。

附录 H　ORACLE8I/9I 初始化参数文件

H.1　Oracle8i 初始化参数文件

```
#
# Copyright (c) 1991, 2000 by Oracle Corporation
#
###########################################################################
# Example INIT.ORA file
#
# This file is provided by Oracle Corporation to help you customize
# your RDBMS installation for your site.  Important system parameters
# are discussed, and example settings given.
#
# Some parameter settings are generic to any size installation.
# For parameters that require different values in different size
# installations, three scenarios have been provided: SMALL, MEDIUM
# and LARGE.  Any parameter that needs to be tuned according to
# installation size will have three settings, each one commented
# according to installation size.
#
# Use the following table to approximate the SGA size needed for the
# three scenarious provided in this file:
#
#                     -------Installation/Database Size------
#                      SMALL           MEDIUM          LARGE
#  Block         2K    4500K            6800K          17000K
```

```
#  Size          4K    5500K            8800K           21000K
#
# To set up a database that multiple instances will be using, place
# all instance-specific parameters in one file, and then have all
# of these files point to a master file using the IFILE command.
# This way, when you change a public
# parameter, it will automatically change on all instances.  This is
# necessary, since all instances must run with the same value for many
# parameters. For example, if you choose to use private rollback' segments,
# these must be specified in different files, but since all gc_*
# parameters must be the same on all instances, they should be in one file.
#
# INSTRUCTIONS: Edit this file and the other INIT files it calls for
# your site, either by using the values provided here or by providing
# your own.  Then place an IFILE= line into each instance-specific
# INIT file that points at this file.
#
# NOTE: Parameter values suggested in this file are based on conservative
# estimates for computer memory availability. You should adjust values upward
# for modern machines.
#
###############################################################################

db_name = "s450"
instance_name = s450

service_names = s450


control_files      =      ("/home/oracle/app/oracle/oradata/s450/control01.ctl",
"/home/oracle/app/oracle/oradata/s450/control02.ctl",
"/home/oracle/app/oracle/oradata/s450/control03.ctl")

open_cursors = 300
max_enabled_roles = 30
# db_block_buffers = 2048
db_block_buffers = 8192

#shared_pool_size = 52428800
shared_pool_size=104857600

# large_pool_size = 614400
large_pool_size = 1228800
# java_pool_size = 20971520
```

```
java_pool_size = 41943040

log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

processes = 300

log_buffer = 163840

# audit_trail = false  # if you want auditing
# timed_statistics = false  # if you want timed statistics
# max_dump_file_size = 10000  # limit trace file size to 5M each

# Uncommenting the lines below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest_1 = "location=/home/oracle/app/oracle/admin/s450/arch"
# log_archive_format = arch_%t_%s.arc


#DBCA uses the default database value (30) for max_rollback_segments
#100 rollback segments (or more) may be required in the future
#Uncomment the following entry when additional rollback segments are created and made
online
#max_rollback_segments = 51
# If using private rollback segments, place lines of the following
# form in each of your instance-specific init.ora files:
rollback_segments = (  RBS1, RBS2, RBS3, RBS4, RBS5, RBS6, RBS7, RBS8, RBS9, RBS10,
RBS11, RBS12, RBS13, RBS14, RBS15, RBS16 )

# Global Naming -- enforce that a dblink has same name as the db it connects to
# global_names = false

# Uncomment the following line if you wish to enable the Oracle Trace product
# to trace server activity.  This enables scheduling of server collections
# from the Oracle Enterprise Manager Console.
# Also, if the oracle_trace_collection_name parameter is non-null,
# every session will write to the named collection, as well as enabling you
# to schedule future collections from the console.
# oracle_trace_enable = true

# define directories to store trace and alert files
background_dump_dest = /home/oracle/app/oracle/admin/s450/bdump
core_dump_dest = /home/oracle/app/oracle/admin/s450/cdump
```

```
#Uncomment this parameter to enable resource management for your database.
#The SYSTEM_PLAN is provided by default with the database.
#Change the plan name if you have created your own resource plan.#
resource_manager_plan = system_plan
user_dump_dest = /home/oracle/app/oracle/admin/s450/udump

db_block_size = 8192

remote_login_passwordfile = exclusive

os_authent_prefix = ""

# The following parameters are needed for the Advanced Replication Option
job_queue_processes = 4
job_queue_interval = 60
distributed_transactions = 10
open_links = 4

mts_dispatchers = "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"
# Uncomment the following line when your listener is configured for SSL
# (listener.ora and sqlnet.ora)
# mts_dispatchers = "(PROTOCOL=TCPS)(PRE=oracle.aurora.server.SGiopServer)"

compatible = "8.1.0"
sort_area_size = 65536
sort_area_retained_size = 65536
```

H.2 Oracle9i 初始化参数文件

```
##############################################################################
# Copyright (c) 1991, 2001 by Oracle Corporation
##############################################################################

###########################################
# MTS
###########################################
dispatchers="(PROTOCOL=TCP)(SER=MODOSE)",
```

```
"(PROTOCOL=TCP)(PRE=oracle.aurora.server.GiopServer)",
"(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"

###########################################
# 其他
###########################################
compatible=9.0.0
db_name=ora90

###########################################
# 分布式，复制和快照
###########################################
db_domain=""
remote_login_passwordfile=EXCLUSIVE

###########################################
# 排序，散列联接，位图索引
###########################################
sort_area_size=524288

###########################################
# 文件配置
###########################################
control_files=("D:\oracle\oradata\ora90\CONTROL01.CTL",
"D:\oracle\oradata\ora90\CONTROL02.CTL",
"D:\oracle\oradata\ora90\CONTROL03.CTL")

###########################################
# 池
###########################################
java_pool_size=33554432
large_pool_size=1048576
#shared_pool_size=33554432
```

```
shared_pool_size=41943040

###########################################
# 游标和库高速缓存
###########################################
open_cursors=300

###########################################
# 系统管理的撤销和回退段
###########################################
undo_management=AUTO
undo_tablespace=UNDOTBS

###########################################
# 网络注册
###########################################
instance_name=ora90

###########################################
# 诊断和统计
###########################################
background_dump_dest=D:\oracle\admin\ora90\bdump
core_dump_dest=D:\oracle\admin\ora90\cdump
timed_statistics=TRUE
user_dump_dest=D:\oracle\admin\ora90\udump

###########################################
# 进程和会话
###########################################
processes=150

###########################################
```

```
# 重做日志和恢复
##########################################
fast_start_mttr_target=300

##########################################
# 高速缓存和 I/O
##########################################
db_block_size=4096
db_cache_size=33554432
```