# 体系结构

一:物理结构

1.

数据文件（data file）：是指存储数据库数据的文件。

* 一个数据文件只能属于一个表空间。
* 数据文件创建后可改变大小。
* 创建新的表空间需创建新的数据文件。
* 数据文件一旦加入到表空间，就不能从表空间移走，也不能和其他表空间发生联系。
* 数据文件中只保留修改后的数据

重做日志文件(redo log file) ：用于记录数据库所做的全部变更（如增加、删除、修改）及由Oracle内部行为而引起的数据库变化信息。

* 日志文件中即保留修改后的数据，又保留修改前的数据。
* 一个数据库至少最少需要两个重做日志文件。

控制文件(Control file)

配置文件

重做日志文件

2.

创建重做日志组和成员

例：在ALTER DATABASE语句中使用ADD LOGFILE子句也可以创建重做日志组：

ALTER DATABASE ADD LOGFILE ('log1c.rdo', 'log2c.rdo') SIZE 5000k;

例：在使用ALTER DATABASE语句创建重做日志组时，可以使用GROUP子句定义组编号：

ALTER DATABASE ADD LOGFILE GROUP 10 ('log1a.rdo', 'log2a.rdo') SIZE 5000k;

例：在ALTER DATABASE语句中使用ADD LOGFILE MEMBER关键字，可以向已存在的重做日志组中添加成员：

ALTER DATABASE ADD LOGFILE MEMBER 'log3a.rdo' TO GROUP 10;

例：删除重做日志组log1a.rdo：

ALTER DATABASE DROP LOGFILE MEMBER 'log1a.rdo';

例：删除编号为10的重做日志组：

ALTER DATABASE DROP LOGFILE GROUP 10;

二：内存结构

* 每个运行的Oracle数据库都对应一个Oracle例程（Instance），也可以称为实例。
* SGA（系统全局区）和Oracle后台进程结合在一起，就是一个Oracle例程。
* 一个数据库可以被多个实例访问。
* SGA由数据库高速缓存区，重做日志高速缓存区，共享池组成。
* 共享池包含库高速缓存，数据字典高速缓存区，
* 库高速缓存（Library Cache）包括 共享SQL区、PL/SQL程序代码区，
* 后台进程：为了保证Oracle数据库在任意一个时刻可以处理多用户的并发请求，进行复杂的数据操作，Oracle数据库起用了一些相互独立的附加进程，称为后台进程。服务器进程在执行用户进程请求时，调用后台进程来实现对数据库的操作。
* Oracle的后台进程主要包括：
* （1）DBWR 数据库写入进程：负责将数据库缓冲区内变动过的数据块写回磁盘内的数据文件，DBWR可有多个。
* （2）LGWR 日志文件写入进程: 负责将重做日志缓冲区内变动记录循环写回磁盘内的重做日志文件，
* （3）ARCH 归档进程（archive process）: 归档进程ARCH负责在重做日志文件切换后将已经写满的重做日志文件复制到归档日志文件中，
* （4）CKPT 检查点进程：检查点是一个数据库事件,唤醒DBWR进程，
* （6）RECO 恢复进程: 在Oracle 11g分布式数据库环境中，RECO处理程序由于网络故障或系统故障挂起的分布式事务。
* （7） SMON 系统监控进程（system monitor）: SMON完成例程的恢复工作 (启动实例,装载数据库,打开数据库)
* （8） PMON 进程监控进程（process monitor）: 当用户进程失败时用于完成进程的恢复。

注：至少牢记5个

三：逻辑结构

1.逻辑结构单元类型

* + 表空间（TABLESPACE）、段（SEGMENT）、区（EXTENT）数据块（BLOCK） ；
  + ；模式对象(schema)（表、视图、序列和同义词等）

2.数据库由若干个表空间组成。

3.一个Oracle数据库可以拥有多个表空间，每个表空间可包含多个段，每个段由若干个区间组成，每个区间包含多个数据块，每个Oracle数据块由多个OS物理磁盘块组成。

4.表空间是存储模式对象的容器，一个数据库对象只能存储在一个表空间中（分区表和分区索引除外)

5.段不直接和数据文件发生关系。一个段可以属于多个数据文件。

6.段不可以跨表空间，一个段只能属于一个表空间

7.段由多个区(extents)组成。区是磁盘空间分配的最小单位,磁盘按区划分，每次至少分配一个区。

8.一个区间一定属于某个段

9.区间不可以跨数据文件，只能存在于某一个数据文件中

10.创建和管理表空间

利用CREATE TABLESPACE命令创建和管理表空间

　　语法格式：

CREATE TABLESPACE tablespace\_name

　　DATAFILE ‘path/filename’ [SIZE integer [ K∣M ]][ REUSE ]

　　　　　　[ AUTOEXTEND [ OFF∣ON

[ NEXT integer [ K∣M ]]

　　　　　　　[ MAXSIZE [ UnLIMITED∣integer [ K∣M ] ] ] ]

　　　　　　[ ONLINE∣OFFLINE ]

　　[ LOGGING∣NOLOGGING ]

　　[ PERMANENT∣TEMPORARY ]

　　[ EXTENT MANAGEMENT LOCAL [ AUTOALLOCATE∣UNIFORM SIZE integer [ K∣M ]]

[Segment space management auto|manual]

10.1.创建和管理表空间

　【例2.5】创建大小为50M的表空间TEST，禁止自动扩展数据文件。

　　　create tablespace test1

logging

datafile ‘c:\test101.dbf' size 5m reuse autoextend off;

　【例2.6】创建表空间DATA，允许自动扩展数据文件。

　　　CREATE TABLESPACE DATA2

　LOGGING

　DATAFILE 'c:\o\DATA02.DBF' SIZE 50M

　　　　REUSE AUTOEXTEND ON NEXT 10M MAXSIZE 200M

　　　　　DEFAULT STORAGE(INITIAL 16K NEXT 32K MINEXTENTS 1 )　　　;

本地管理表空间

【例】在CREATE TABLESPACE语句中使用EXTENT MANAGEMENT LOCAL子句，可以创建一个本地管理表空间，代码如下：

SQL> CREATE TABLESPACE OrclTBS01

DATAFILE 'C:\o\OrclTBS01.dbf' SIZE 5M

EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

【例】在创建表空间OrclTBS02时，指定其区间为128KB，代码如下：

SQL> CREATE TABLESPACE OrclTBS02

DATAFILE 'C:\o\OrclTBS02.dbf' SIZE 30M

EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;

【例】在CREATE TABLESPACE语句中使用BIGFILE关键词可以创建大文件表空间，代码如下：

SQL> CREATE BIGFILE TABLESPACE bigtbs

DATAFILE 'C:\o\bigtbs.dbf' SIZE 10G;

临时表空间

【例】创建临时表空间tmptbs，代码如下：

SQL> CREATE TEMPORARY TABLESPACE tmptbs

TEMPFILE 'C:\o\tmptbs.dbf'

SIZE 20M REUSE

EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;

2.撤销表空间

撤销记录可以用于完成如下功能：

1当执行ROLLBACK命令时，完成回滚操作。2.恢复数据库。3.使用闪回查询分析以前时间点的数据。4.使用闪回技术从逻辑破坏中恢复数据。

初始化参数UNDO\_TABLESPACE设置默认的撤销表空间。

【例】使用CREATE UNDO TABLESPACE语句创建撤销表空间：

SQL> CREATE UNDO TABLESPACE undotbs01

DATAFILE 'C:\o\undotbs02.dbf' SIZE 2M REUSE;

3维护表空间和数据文件

用ALTER TABLESPACE管理表空间

　语法格式：

　ALTER TABLESPACE tablespace\_name

[ ADD DATAFILE∣TEMPFILE ‘path/filename’ [ SIZE integer [ K∣M ] ] 　[ REUSE ]

　　[ AUTOEXTEND [ OFF∣ON [ NEXT integer [ K∣M ] ]

MAXSIZE [ UNLIMITED∣integer [ K∣M ] ] ]

　　　　　　　[ RENAME DATAFILE ‘path\filename’,…n TO ‘path\re\_filename’,…n ]

[ DEFAULT STORAGE storage\_clause ]

[ ONLINE∣OFFLINE [ NORMAL∣TEMPORARY∣IMMEDIATE ] ]

[ LOGGING∣NOLOGGING ]

[ READ ONLY∣WRITE ]

[ PERMANENT ]

[ TEMPORARY ]

ADD DATAFILE∣TEMPFILE：向表空间添加指定的数据文件或临时文件。

　RENAME DATAFILE：对一个或多个表空间的数据文件重命名。

　READ ONLY：表明表空间上不允许进一步写操作。

　READ WRITE：表明在先前只读表空间上允许写操作。

【例】将表空间OrclTBS02中的数据文件改名。可以使用下面的语句：

ALTER TABLESPACE mytest rename datafile 'c:\o\10.dbf' to 'c:\o\test0001.dbf'

移动mytest表空间中数据文件test001.dbf的步骤如下。

(1)修改mytest表空间的状态为OFFLINESQL> ALTER TABLESPACE mytest OFFLINE;

(2)将磁盘中的myspace02.dbf文件移动到新的目录中

(3)使用ALTER TABLESPACE语句，将mytest表空间中test001.dbf文件的原名称和路径修改为新名称和路径

(4) 修改mytest表空间的状态为ONLINE，如下：SQL> ALTER TABLESPACE mytest ONLINE;

检查文件是否移动成功需要介质恢复 recover datafile 'd:\102.dbf'

【例】在临时表空间tmptbs中添加临时文件，tmptbs01.dbf，文件大小为20MB，代码如下：

SQL> ALTER TABLESPACE tmptbs

ADD TEMPFILE ‘c:\omptbs01.dbf' SIZE 20M;

【例】将表空间OrclTBS01设置为脱机状态：

SQL> ALTER TABLESPACE OrclTBS01 OFFLINE;

【DBA\_TABLESPACES视图查看表空间的状态】

SELECT TABLESPACE\_NAME,CONTENTS,STATUS FROM DBA\_TABLESPACES;

【例】将表空间OrclTBS01设置为联机状态：

SQL> ALTER TABLESPACE OrclTBS01 ONLINE;

【例】将表空间OrclTBS01设置为只读表空间：

SQL> ALTER TABLESPACE OrclTBS01 READ ONLY;

【例】将表空间OrclTBS01设置为可读写状态：

SQL> ALTER TABLESPACE OrclTBS01 READ WRITE;

【例】删除表空间OrclTBS01：

SQL> DROP TABLESPACE OrclTBS01;

INCLUDING CONTENTS子句可以在删除表空间的同时删除其中的段，例如：

SQL> DROP TABLESPACE OrclTBS01 INCLUDING CONTENTS;

INCLUDING CONTENTS AND DATAFILE子句可以在删除表空间的同时，删除包含的段和数据文件：

SQL> DROP TABLESPACE OrclTBS03 INCLUDING CONTENTS AND DATAFILES;

例：创建多个数据文件。。。。datafile ‘数据文件’ size 10M reuse,’第二个数据文件’size 20M

例：重命名表空间

Alter tablespace 原表空间名 rename to新表空间名。

例：如何增加表空间的容量?

create tablespace 表空间名 autoextend on next 大小

alter tablespace 表空间名 add datafile ‘新的数据文件 ’

alter tablespace 数据库名 datafile ‘数据文件’resize 大小

# 对象篇

1．创建表

CREATE TABLE [schema.] table\_name

(column\_name datatype [DEFAULT expression] [column\_constraint],…n)

[PCTFREE integer]

[PCTUSED integer]

[INITRANS integer]

[MAXTRANS integer]

[TABLESPACE tablespace\_name]

[STORGE storage\_clause]

[CLUSTER cluster\_name(cluster\_column,…n)]

[ENABLE | DISABLE ]

[AS subquery]

其中：

table\_name：表名称。

schema：新表所属的用户方案。

column\_name：指定表的一个列的名字。

datatype：该列的数据类型。

PCTFREE：指定表或者分区的每一个数据块为将来更新表行所保留的空间百分比。

PCTUSED：指定Oracle维持表的每个数据块已用空间的最小百分比。

INITRANS：指定分配给表的每一数据块中的事务条目的初值。

MAXTRANS：指定可更新分配给表的数据块的最大并发事务数。

TABLESPACE：指定新表存储在由tablespace\_name指定的表空间中。

STORGE：指定表的存储特征。STORGE子句的基本格式为：

STORAGE

( INITIAL integer K | integer M

NEXT integer K | integer M

MINEXTENTS integer

MAXEXTENTS integer | UNLIMITED

PCTINCREASE integer

)

INITIAL指定为表分配的第一个区的大小；

NEXT指定第一个扩展区的大小；

MINEXTENTS为创建段时已分配的总区数；

MAXEXTENTS表示Oracle数据库可以分配给该对象的总区；

PCTINCREASE指定每个区相对于上一个区的增长百分

【例2.15】利用CREATE TABLE命令为XSCJ数据库建立表XS\_KC，

CREATE TABLE SCOTT.XS\_KC

( XH char(6) NOT NULL,

KCH char(6) NOT NULL,

CJ number(2) NULL,

XF number(2) NULL,

CONSTRAINT "PK\_XS\_KC" PRIMARY KEY("XH", "KCH") )

【例2.18】(1) 在表XS中增加2列：JXJ(奖学金等级)，DJSM(奖学金等级说明)。

CREATE TABLE SCOTT.XS

(XH CHAR(6) NOT NULL,

XM VARCHAR(8) NOT NULL,

ZYM CHAR(6),

XB CHAR(2),

CSSJ DATE,

ZXF NUMBER(2),

PRIMARY KEY (XH))

ALTER TABLE SCOTT.XS

ADD ( JXJ NUMBER(1),

DJSM VARCHAR2(40) DEFAULT '奖金1000元');

(2)在表XS中修改名为DJSM的列的默认值。

ALTER TABLE SCOTT.XS

MODIFY ( DJSM DEFAULT '奖金800元' );

(3)在表XS中删除名为JXJ列。

ALTER TABLE SCOTT.XS DROP COLUMN JXJ;

2.修改表

Alter table table\_name

[add (column\_name datatype[default expression][column\_name\_constraint])] …添加属性

[modify ([datatype][ column\_name\_constraint]])]修改

例：modify sno varchar(20)

[drop column 列名]

3.插入数据

语法格式：

INSERT INTO table\_name(column\_list)

VALUES(constant1,constant2,…)

该语句的功能是向由table\_name指定的表中加入一行，由VALUES指定的各列值。

(1) 在插入时，列值表必须与列名表顺序和数据类型一致。

(2) VALUES中描述的值可以是一个常量、变量或一个表达式。

(3) 如果列值为空，则值必须置为NULL。如果列值指定为该列的默认值，则用DEFAULT。

日期型数据常用函数

TO\_DATE(字符串, '格式’)

作用:将字符转换为日期类型

4.删除数据

. 使用 DELETE删除数据

在PL/SQL语言中，删除行可以使用DELETE语句。

语法格式：

DELETE FROM table\_name∣view\_name

[WHERE condition]

若省略该条件，表示删除所有的行。

如果确实要删除一个大表里的全部记录, 可以用TRUNCATE命令---TRUNCATE TABLE, 它可以释放占用的数据块表空间。此操作不可回退。

语法格式：

TRUNCATE TABLE table\_name

使用TRUNCATE TABLE删除了指定表中的所有行，但表的结构及其列、约束等保持不变.

Merge into 用法

语法格式：

MERGE INTO table\_name

USING table\_name ON (join\_condition)

WHEN MATCHED THEN UPDATE SET…

WHEN NOT MATCHED THEN INSERT(…) VALUES(…)

* --  USING 子句用于指定要与目标联接的数据源。
* --  ON 子句用于指定决定目标与源的匹配位置的联接条件。
* --  WHEN 子句用于根据ON 子句的结果指定要执行的操作。

【例】 检查表XS\_JSJ中的数据是否和表XS的数据相匹配，如果匹配则使用INSERT子句执行插入数据行。

搭建平台:create table xs1 as select \* from xs

* INSERT INTO XS1(XH,XM,ZYM,XB,CSSJ,ZXF)
* VALUES('121112', '王小二', '计算机',

'男',TO\_DATE('19860130','YYYYMMDD'),36)

使用MERGE语句XS\_JSJ表中新增的数据插入表XS中：

MERGE INTO XS a

USING XS1 b

ON(a.XH=b.XH)

WHEN NOT MATCHED THEN INSERT(a.xh,a.xm,a.zym,a.xb,a.cssj,a.zxf)

Values(b.xh,b.xm,b.zym,b.xb,b.cssj,b.zxf);

分区表的建立和使用

Range(范围)分区  
 是应用范围比较广的表分区方式，它是以列的值的范围来做为分区的划分条件，将记录存放到列值所在的range分区中。

* create table part\_book
* ( bid number(4),
* bookname varchar2(20),
* bookpress varchar2(30),
* booktime date)
* partition by range(booktime)
* (partition part1 values less than(to\_date('20100101','yyyymmdd')) tablespace system,
* partition part2 values less than (to\_date('20120101','yyyymmdd')) tablespace users,

partition part3 values less than (MAXVALUE) tablespace users);

* insert into part\_book values(1,'oracle','清华大学出版社',to\_date('20110102','yyyymmdd'));
* insert into part\_book values(2,‘数据结构’,‘清华大学出版社’,to\_date(‘20090102’,‘yyyymmdd'))；
* 2 list(列表) 分区

列表分区表是基于特定值对表进行分区,其分区列的值为非数值型或日期类型,并且分区列的取值范围较少, partion by list子句，列表值相同的行将被存储到同一分区中。

* create table part\_book1
* ( bid number(4),
* bookname varchar2(20),
* bookpress varchar2(30),
* booktime date)
* partition by list(bookpress)
* (partition part1 values('清华大学出版社') tablespace system,
* partition part2 values('教育出版社') tablespace users);

查询：select \* from part\_book1 partition(part1);

* Hash (散列)分区表
* create table part\_book3
* ( bid number(4),
* bookname varchar2(20),
* bookpress varchar2(30),
* booktime date)
* partition by hash(bid)
* (partition part1 tablespace system,
* partition part2 tablespace users);

同义词（synonym）

. 利用CREATE SYNONYM命令创建同义词

语法格式：

CREATE [PUBLIC] SYNONYM [schema.]synonym\_name

FOR [schema.]object [@dblink]

【例3.18】创建同义词。

(1) 为XSCJ数据库的XS\_KC表创建公用同义词XS\_KC。

CREATE PUBLIC SYNONYM XS\_KC

FOR SYSTEM.XS\_KC;

【例3.19】SYSTEM用户查询XSCJ数据库XS表中所有学生的情况。

SELECT \* FROM KC;

如果没有为XSCJ数据库XS表创建同义词XS，那么SYSTEM用户查询XS表则需指定XS表的所有者。

SELECT \* FROM SCOTT.KC;

利用DROP SYNONYM命令删除同义词

语法格式：

DROP [PUBLIC] SYNONYM [schema.]synonym\_name

说明：

PUBLIC表明删除一个公用同义词。Schema指定将要删除的同义词的用户方案。synonym\_name为将要删除的同义词名称。

序列(sequence)定义存储在数据字典中。序列通过提供唯一整数值的顺序表用来简化程序设计工作.

不管哪个用户或进程使用了序列生成器中的一个值，则下一个用户或者进程所使用的值是上一个值的后继值 。

序列(sequence)

【例10.4】创建一个降序序列。

CREATE SEQUENCE STUDENT\_SEQUENCE

START WITH 5000

INCREMENT BY -2

MAXVALUE 5000

MINVALUE 1

NOCYCLE ;

INCREMENT BY：指定序列递增或递减的间隔数值。

START WITH：序列的起始值。若不指定该值，对升序序列将使用该序列默认的最小值。对降序序列，将使用该序列默认的最大值。

Cache:设置序列占据的内存块的大小,默认值为20

2.2使用序列

搭建环境

create table student(

sno varchar(8),

sname varchar(10));

插入:Insert into student values(student\_sequence.nextval, '李明');

引用序列是通过伪列nextval完成的，它用的是序列的 下一个值，若引用当前值，则用伪列currval;

修改序列参数

alter sequence student\_sequence

maxvalue 200;

例如，要删除S\_TEST序列，可使用如下语句：

DROP sequence student\_sequence;

# Pl/sql语言篇

**---示例程序块**

**---set serveroutput on**

**Declare**

**example\_text varchar2(100);**

**begin**

**example\_text:='欢迎您!';**

**dbms\_output.put\_line(example\_text);**

**exception**

**when others then**

**dbms\_output.put\_line('出现异常了!');**

**end;**

**---示例程序块2**

**DECLARE**

**v\_xm varchar2(8):='张琼';**

**v\_zym varchar2(10):='计算机';**

**v\_zxf number(2):=45; /\*定义变量类型\*/**

**BEGIN**

**UPDATE XS**

**SET zxf=v\_zxf**

**WHERE xm=v\_xm;**

**IF SQL%NOTFOUND THEN**

**INSERT INTO XS(XH,XM,ZYM,ZXF) VALUES('001',v\_xm,v\_zym,v\_zxf);**

**END IF;**

**end;**

**/\* 说明:%属性提示符 \*/**

**SELECT INTO 赋数用法**

**select ... into  ... from ... 赋值可能有三种情况**

* **无值**
* **多值**
* **唯一值**

**%rowtype和%type类型使用举例**

DECLARE

v\_emp emp%ROWTYPE;

v\_ename emp.ename%type;

v\_sal emp.sal%type;

BEGIN

SELECT \* INTO v\_emp FROM emp WHERE ename='SMITH';

DBMS\_OUTPUT.PUT\_LINE(v\_emp.empno||' '||v\_emp.sal);

select ename,sal INTO v\_ename,v\_sal FROM emp WHERE empno=7900;

DBMS\_OUTPUT.PUT\_LINE(v\_ename||' '||v\_sal);

END;

**PL/SQL基本程序结构和语句**

**. IF逻辑结构**

**(1) IF-THEN-ENDIF结构**

**(2) IF-THEN-ELSE-ENDIF结构**

**(3) IF-THEN-ELSIF结构**

**(4) CASE结构**

**例：涉及表为scott.emp。输入一个员工号，修改该员工的工资，如果该员工为10号部门(deptno)，工资增加100；若为20号部门，工资增加150；若为30号部门，工资增加200；否则增加300。**

**分析：**

**输入一个员工号------accept**

**找出根据那个变量值的不同来做分支结构：部门号v\_deptno**

**Select into语句求此变量，部门号v\_deptno**

**分支结构**

* **DECLARE**
* **v\_deptno scott.emp.deptno%type;**
* **v\_increment NUMBER(4);**
* **v\_empno scott.emp.empno%type;**
* **BEGIN**
* **v\_empno:=&x;**
* **SELECT deptno INTO v\_deptno FROM scott.emp WHERE empno=v\_empno;**
* **IF v\_deptno=10 THEN v\_increment:=100;**
* **ELSIF v\_deptno=20 THEN v\_increment:=150;**
* **ELSIF v\_deptno=30 THEN v\_increment:=200;**
* **ELSE v\_increment:=300;**
* **END IF;**
* **UPDATE scott.emp SET sal=sal+v\_increment WHERE empno=v\_empno;**
* **END;**

循环结构

*求10的阶乘。*

DECLARE

s NUMBER:=1;

n NUMBER:=2;

BEGIN

LOOP

s:=s\*n;

n:=n+1;

IF n>10 THEN EXIT;

END IF;

END LOOP;

dbms\_output.put\_line(to\_char(s));

END;

**求10的阶乘。**

* **DECLARE**
* **s NUMBER:=1;**
* **n NUMBER:=2;**
* **BEGIN**
* **LOOP**
* **s:=s\*n;**
* **n:=n+1;**
* **exit when n>10;**
* **END LOOP;**
* **dbms\_output.put\_line(to\_char(s));**
* **END;**

**求10的阶乘。**

* **SQL> declare**
* **s number:=1;**
* **n number:=2;**
* **begin**
* **while n<=10**
* **loop**
* **s:=s\*n;**
* **n:=n+1;**
* **end loop;**
* **dbms\_output.put\_line(s);**
* **end;**

**求10的阶乘。**

* SQL> declare

s number:=1;

begin

for x in 1..10

loop

s:=s\*x;

end loop;

dbms\_output.put\_line(s);

end;

游标

四种游标遍历

SQL> declare

2 cursor my\_cursor is select xm from xs;

3 v\_xm xs.xm%type;

4 begin

5 open my\_cursor;

6 fetch my\_cursor into v\_xm;

7 while my\_cursor %found

8 loop

9 dbms\_output.put\_line('记录条数'||my\_cursor%rowcount);

10 dbms\_output.put\_line(v\_xm);

11 fetch my\_cursor into v\_xm;

12 end loop;

13 close my\_cursor;

14 end;

declare

2 cursor my\_cursor is select xm from xs;

3 v\_xm xs.xm%type;

4 begin

5 open my\_cursor;

6 loop

7 fetch my\_cursor into v\_xm;

8 exit when my\_cursor %notfound;

9 dbms\_output.put\_line('记录条数'||my\_cursor%rowcount);

10 dbms\_output.put\_line(v\_xm);

11 end loop;

12 close my\_cursor ;

13 end;

14

SQL> declare

2 cursor my\_cursor is select xm,xh from xs;

3 v\_xs xs%rowtype;

4 begin

5 for v\_xs in my\_cursor

6 loop

7 dbms\_output.put\_line(v\_xs.xm||' '||v\_xs.xh);

8 end loop;

9 end;

10 /

SQL> declare

2 v\_xm xs%rowtype;

3 begin

4 for v\_xm in (select xm,xh from xs) loop

5 dbms\_output.put\_line(v\_xm.xm);

6 end loop;

7 end;

8

9 /

带参数的游标

declare

cursor my\_cursor(v\_zym xs.zym%type) is select xm,xh from xs where zym=v\_zym;

v\_xs xs%rowtype;

begin

open my\_cursor('计算机');

loop

fetch my\_cursor into v\_xs.xm,v\_xs.xh;

exit when my\_cursor %notfound;

dbms\_output.put\_line('记录条数'||my\_cursor%rowcount);

dbms\_output.put\_line('学号'||' '||v\_xs.xh||' '||'姓名'||' '||v\_xs.xm);

end loop;

close my\_cursor ;

end;

存储过程

.概念：在Oracle中，可以在数据库中定义子程序，这种程序块称为存储过程(procedure)。它存放在数据字典中，可以在不同用户和应用程序之间共享，并可实现程序的优化和重用。

使用存储过程的优点是：

(1)过程在服务器端运行，执行速度快。

(2)过程执行一次后代码就驻留在高速缓冲存储器。

(3) 确保数据库的安全。

(4) 自动完成需要预先执行的任务。

存储过程的创建

参数模式

IN参数类型

OUT参数类型

IN OUT参数类型

IN （默认参数类型）:表示此参数接受过程外传来的值。

当过程被调用时，实参值被传递给过程。

在过程内，该参数起常数作用，可读不可写。

调用结束，实参值不变。

OUT参数：表示此参数将在过程中被赋值，并传递到过程体外。

当过程被调用时，实参事先定义,参数传递时实参值被忽略。

在过程内，该参数起未初始化的变量作用，值为NULL。

过程内，该参数可读可写。 调用结束，形参值赋给实参值。

IN OUT参数模式表示此参数同时具备IN和OUT参数型的特性。

当过程被调用时，实参值被传递给过程。

在过程内，该参数起已初始化变量作用

过程内，该参数可读可写。调用结束，形参赋给实参。

* **CREATE OR REPLACE PROCEDURE procedure\_name**
* **( parameter1\_name mode datatype,**
* **parameter2\_name mode datatype …)**
* **AS|IS**
* **/\*Declarative section is here \*/**
* **BEGIN**
* **/\*Executable section is here\*/**
* **EXCEPTION**
* **/\*Exception section is here\*/**

**END procedure\_name**

**例１：创建一存储过程update\_emp,该过程用于将emp表中empno为7876的员工姓名修改为candy**

* create or replace procedure update\_emp
* as
* begin
* update scott.emp set ename='candy' where empno=7876;
* end update\_emp;

**【例2】计算指定系总学分大于40的人数。（创建procedure count\_grade ）**

**分析: CREATE OR REPLACE PROCEDURE count\_grade**

* **( v\_zym in varchar2,person\_num out number )**
* **AS**
* **BEGIN**
* **SELECT COUNT(ZXF)**
* **INTO person\_num**
* **FROM XS**
* **WHERE ZYM=v\_zym and zxf>=40;**
* **END count\_grade;**

**注意:存储过程定义中的类型长度不需要加,即varchar2 不能写成varchar2(6),同理对number类型。**

**执行procedure count\_grade**

**declare**

**person\_n number(3);**

**begin**

**count\_grade('计算机',person\_n);**

**dbms\_output.put\_line(person\_n);**

**end;**

**注意:person\_n参数模式为out，需要先定义，IN 参数时需要赋初值**

**【例４】编写一存储过程，用于计算指定系学生的总学分。**

**分析：存储过程使用了一个输入参数和一个输出参数。**

**create or replace procedure totalcredit(v\_zym in varchar2,v\_total out number)**

**is**

**begin**

**select sum(zxf) into v\_total from xs**

**where zym=v\_zym;**

**end totalcredit;**

**执行**

**declare**

**v\_total number;**

**begin**

**totalcredit('计算机',v\_total);**

**dbms\_output.put\_line(v\_total);**

**end;**

**例 5: 创建一个存储过程，以部门号为参数，返回该部门的人数和最高工资。**

* **CREATE OR REPLACE PROCEDURE return\_deptinfo(**
* **p\_deptno IN scott.emp.deptno%TYPE,**
* **p\_avgsal OUT scott.emp.sal%TYPE,**
* **p\_count OUT scott.emp.sal%TYPE)**
* **AS**
* **BEGIN**
* **SELECT avg(sal),count(\*) INTO p\_avgsal,p\_count**
* **FROM scott.emp**
* **WHERE deptno=p\_deptno;**
* **EXCEPTION**
* **WHEN NO\_DATA\_FOUND THEN**
* **DBMS\_OUTPUT.PUT\_LINE('The department don’t exists!');**
* **END return\_deptinfo;**
* **存储过程的执行**
* **declare**
* **p\_avgsal scott.emp.sal%TYPE;**
* **p\_count scott.emp.sal%TYPE;**
* **begin**
* **return\_deptinfo(20,p\_avgsal,p\_count);**
* **dbms\_output.put\_line(p\_avgsal||**' '||**p\_count);**
* **end;**
* **补充:创建一个存储过程，以部门号为该存储过程的in类型参数，查询该部门的平均工资，并输出该部门中比平均工资高的员工号、员工名。(for 循环遍历 游标)**

**CREATE OR REPLACE PROCEDURE show\_emp(**

**p\_deptno emp.deptno%TYPE)**

**AS**

**v\_sal emp.sal%TYPE;**

**BEGIN**

**SELECT avg(sal) INTO v\_sal FROM emp WHERE deptno=p\_deptno;**

**DBMS\_OUTPUT.PUT\_LINE(p\_deptno||' '||'average salary is:'||v\_sal);**

**FOR v\_emp IN (SELECT \* FROM emp WHERE deptno=p\_deptno AND sal>v\_sal) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(v\_emp.empno||' '||v\_emp.ename);**

**END LOOP;**

**END show\_emp;**

触发器

**CREATE OR REPLACE TRIGGER trigger\_name**

**{ BEFORE∣AFTER∣INSTEAD OF }**

**DELETE OR INSERTE [R UPDATE OF column,…n**

**ON [table\_name∣view\_name**

**FOR EACH ROW [ WHEN(condition) ] ]**

**DECLARE…**

**BEGIN…**

**EXCEPTION….**

**END [trigger\_name];**

* **触发事件**

**Insert |delete|update of column**

**在行级触发,为了获得某列在修改前后的数据，---标志符:old和:new**

* 例：**为emp表创建一个触发器，当插入新员工时显示新员工的员工号、员工名；当更新员工工资时，显示修改前后员工工资；当删除员工时，显示被删除的员工号、员工名。**
* **CREATE OR REPLACE TRIGGER t3**
* **BEFORE INSERT OR UPDATE of sal OR DELETE ON scott.emp**
* **FOR EACH ROW**
* **BEGIN**
* **IF INSERTING THEN**
* **DBMS\_OUTPUT.PUT\_LINE(:new.empno||' '||**
* **:new.ename);**
* **ELSIF UPDATING THEN**
* **DBMS\_OUTPUT.PUT\_LINE(:old.sal||' '||:new.sal);**
* **ELSE**
* **DBMS\_OUTPUT.PUT\_LINE(:old.empno||' '||**
* **:old.ename);**
* **END IF;**
* **END t3;**
* **触发触发器t3**
* **Set serveroutput on**
* declare
* begin
* update scott.emp set empno=7521 where empno=7522;
* commit;
* end;

【例】**增加一新表XS\_HIS，表结构和表XS相同，用来存放从XS表中删除的记录。创建一个触发器，当XS表被删除一行，把删除的记录写到日志表XS\_HIS中。**

**分析:**

**1. 创建表 xs\_his**

**Create table xs\_his as select \* from xs;**

**2. 触发时间: before**

**3. 触发事件: delete**

**4. 触发级别: 行级 for each row**

**CREATE OR REPLACE TRIGGER del\_xs**

**BEFORE DELETE ON XS FOR EACH ROW**

**BEGIN**

**INSERT INTO XS\_HIS (XH,XM,ZYM,XB,CSSJ,ZXF) VALUES(:OLD.XH,:OLD.XH, :OLD.ZYM, :OLD.XB, :OLD.CSSJ,:OLD.ZXF);**

**END del\_xs;**

【例】**利用触发器在数据库XSCJ的XS表执行插入、更新和删除3种操作后在sql\_info中给出相应提示。**

**Create table sql\_info(info varchar(10));**

**CREATE TRIGGER cue\_xs**

**AFTER INSERT OR UPDATE OR DELETE ON XS FOR EACH ROW**

**DECLARE**

**Infor varchar2(10);**

**BEGIN**

**IF INSERTING THEN**

**Infor:='插入';**

**ELSIF UPDATING THEN**

**Infor:='更新';**

**ELSE**

**Infor:='删除';**

**END IF;**

**INSERT INTO SQL\_INFO VALUES(infor);**

**END cue\_xs;**

* **为emp表创建一个触发器，当执行插入操作时，统计操作后员工人数；当执行更新工资操作时，统计更新后员工平均工资；当执行删除操作时，统计删除后各个部门剩余的人数(游标)。 ★**
* **CREATE OR REPLACE TRIGGER t4 AFTER INSERT OR UPDATE OR DELETE ON scott.emp**
* **DECLARE**
* **v\_count NUMBER;v\_sal NUMBER(6,2);**
* **BEGIN**
* **IF INSERTING THEN**
* **SELECT count(\*) INTO v\_count FROM scott.emp;**
* **DBMS\_OUTPUT.PUT\_LINE(v\_count);**
* **ELSIF UPDATING THEN**
* **SELECT avg(sal) INTO v\_sal FROM scott.emp;**
* **DBMS\_OUTPUT.PUT\_LINE(v\_sal);**
* **ELSE**
* **FOR v\_dept IN (SELECT deptno,count(\*) num FROM scott.emp GROUP BY deptno) LOOP**
* **DBMS\_OUTPUT.PUT\_LINE(v\_dept.deptno||' '||v\_dept.num);**
* **END LOOP;**
* **END IF; END t4;**
* **创建一个触发器，在修改dept表的部门号后，同时更新emp表中相应的员工的部门号。★**

**分析:触发事件 update of deptno**

**表 scott.dept**

**触发时间 after**

**create or replace trigger tr\_update\_dept**

**after update of deptno on scott.dept**

**for each row**

**begin**

**update scott.emp set deptno=:new.deptno**

**where deptno=:old.deptno;**

**end;**

【例】**利用触发器在数据库XSCJ的XS表执行插入、更新和删除3种操作后在sql\_info中给出相应提示。**

**Create table sql\_info(info varchar(10));**

CREATE TRIGGER cue\_xs

AFTER INSERT OR UPDATE OR DELETE ON XS FOR EACH ROW

DECLARE

Infor varchar2(10);

BEGIN

IF INSERTING THEN

Infor:='XS表插入';

ELSIF UPDATING THEN

Infor:='XS表更新';

ELSE

Infor:='XS表删除';

END IF;

INSERT INTO SQL\_INFO VALUES(infor);

END cue\_xs;

* **系统触发器之logon和logoff触发器，可以将用户访问数据库的情况记录在一个日志表中。**
* **例：**
* **create table u\_log**
* **(username varchar2(50),**
* **activity varchar2(20),**
* **eventdate DATE**
* **);**

**登陆数据库**

* **Create or replace trigger st1**
* **after logon on database**
* **begin**
* **insert into u\_log values**(**user**,'LOGON',**SYSDATE**);
* **END**;

退出数据库

* **create or replace trigger st2**
* **Before logoff on database**
* **begin**
* **insert into u\_log values(user,'LOGOFF',SYSDATE);**
* **END;**

异常处理

查找“王林”同学的学号 ★

DECLARE

v\_result xs.xm%TYPE;

BEGIN

SELECT xh INTO v\_result

FROM xs

WHERE xm='王林';

DBMS\_OUTPUT.PUT\_LINE('The student number is '||v\_result);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('There has TOO\_MANY\_ROWS error');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('There has NO\_DATA\_FOUND error');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('错误情况不明');

END;

使用others异常可以借助两个函数来说明捕捉到的异常的类型-----SQLCODE SQLERRM

DECLARE

v\_result number;

BEGIN

SELECT xm INTO v\_result

FROM xs

WHERE xh='010010';

DBMS\_OUTPUT.PUT\_LINE('The student name is'||v\_result);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('the sqlcode is'||SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('the sqlERRM is'||SQLERRM);

预定义的异常



# 安全篇

数据库初始用户

* SYS，数据库中具有最高权限的DBA，可以启动、修改和关闭数据库，拥有数据字典
* SYSTEM，一个辅助的DBA，不能启动和关闭数据库，但可以进行其他一些管理工作
* SCOTT，口令tiger，用于测试网络的连接
* PUBLIC用户组，为数据库中所有用户设定必需的对象权限和系统权限

创建用户

CREATE USER user\_name IDENTIFIED BY password

DEFAULT TABLESPACE tablespace\_name

TEMPORARY TABLESPACE temp\_tablespace\_name

[ QUOTA n K|M|UNLIMITED ON tablespace\_name ]

[ PROFILE profile\_name ]

[ PASSWORD EXPIRE ]

[ ACCOUNT LOCK | UNLOCK ]

【例7】重新创建用户AUTHOR，口令为ANGEL，缺省表空间为USERS，临时表空间为TEMP。没有定额，使用缺省概要文件，登录数据库前修改口令。

DROP USER AUTHOR;

　　　　CREATE USER AUTHOR

　　　IDENTIFIED BY ANGEL

　　　DEFAULT TABLESPACE USERS

　　　TEMPORARY TABLESPACE TEMP

　　　　PASSWORD EXPIRE;

示例：为PUBLIC用户组授予CREATE SESSION系统权限

GRANT CREATE SESSION TO PUBLIC;

注:普通用户一般只具有CREATE SESSION系统权限。

示例：将scott模式下的emp表的SELECT、UPDATE、 INSERT权限授予user1用户。

GRANT SELECT,INSERT,UPDATE ON scott.emp

TO u\_1 with grant option;

对象权限的回收

语法结构

REVOKE obj\_priv\_list | ALL ON [schema.]object FROM user\_list|role\_list;

revoke SELECT,INSERT,UPDATE ON scott.emp from u\_1;

用户的锁定与解锁

* 锁定用户
* ALTER USER test ACCOUNT LOCK;
* 解锁用户

ALTER USER test ACCOUNT UNLOCK;

创建角色

语法结构

CREATE ROLE role\_name

[NOT IDENTIFIED ][ IDENTIFIED BY password ] 　[WITH ADMIN OPTION]

示例

CREATE ROLE high\_manager\_role

CREATE ROLE middle\_manager\_role

IDENTIFIED BY middlerole;

CREATE ROLE low\_manager\_role

IDENTIFIED BY lowrole;

创建一个新的角色role3，它只能创建用户，而不能执行其他DBA级命令,将role3 赋值给user2。

create role role3;

grant create session,create user to role3;

Grant role3 to user2;

概要文件的作用，限制用户对数据库和系统资源的使用以及进行用户口令管理

使用CREATE PROFILE命令创建概要文件

语法格式：

CREATE PROFILE profile\_name LIMIT

resource\_parameters∣password\_parameters

说明：

①　profile\_name：将要创建的概要文件的名称。

②　resource\_parameters：对一个用户指定资源限制的参数。

③　password\_parameters：口令参数。

* + 示例

CREATE PROFILE pf1 LIMIT  
      FAILED\_LOGIN\_ATTEMPTS 5;

【例7.7】创建一个LIMITED\_PROFILE概要文件，把它提供给用户NICK使用。

CREATE PROFILE LIMITED\_PROFILE LIMIT

FAILED\_LOGIN\_ATTEMPTS 5

/\*在锁定用户帐户之前登录用户帐户的失败次数。\*/

　　　　PASSWORD\_LOCK\_TIME 1;

/\*指定次数的登录失败而引起的帐户封锁的天数\*/

ALTER USER NICK

　　　　PROFILE LIMITED\_PROFILE; ★

将概要文件分配给用户★

* + CREATE USER user5 IDENTIFIED BY user4 PROFILE res\_profile;
  + ALTER USER user5 PROFILE pwd\_profile ;

使用CREATE PROFILE命令创建概要文件

(2) password\_parameters的表达式

[ FAILED\_LOGIN\_\_ATTEMPTS expression∣UNLIMITED∣DEFAULT ]

/\*在锁定用户帐户之前登录用户帐户的失败次数。\*/

[ PASSWORD\_LIFE\_TIME expression∣UNLIMITED∣DEFAULT ]

/\*限制同一口令可用于验证的天数\*/

[ PASSWORD\_LOCK\_TIME expression∣UNLIMITED∣DEFAULT ]

/\*指定次数的登录失败而引起的帐户封锁的天数\*/