目录

[SQL和SQLPLUS 1](#_Toc30248)

[SQLPLUS常用命令 2](#_Toc27419)

[DDL 2](#_Toc9543)

[回收站操作 2](#_Toc5283)

[DCL 2](#_Toc29218)

[子查询 2](#_Toc1890)

[DECODE函数 3](#_Toc27933)

[视图 3](#_Toc32615)

[Oracle序列 3](#_Toc8776)

[基于函数的索引 3](#_Toc32731)

[ROWID 3](#_Toc23088)

[约束 3](#_Toc24819)

[PL/SQL 4](#_Toc32403)

[PL/SQL标识符 4](#_Toc30318)

[PL/SQL程序单元 4](#_Toc10587)

[PL/SQL数据类型和变量 4](#_Toc23305)

[PL/SQL DBMS输出 4](#_Toc22732)

[PL/SQL 条件控制 4](#_Toc4241)

[PL/SQL循环语句 4](#_Toc22986)

[PL/SQL EXIT语句 4](#_Toc24950)

[PL/SQL存储过程 5](#_Toc16799)

[PL/SQL函数 5](#_Toc6158)

[存储过程、函数 5](#_Toc23214)

[PL/SQL游标 5](#_Toc241)

[PL/SQL异常 6](#_Toc17393)

# SQL和SQLPLUS

|  |  |
| --- | --- |
| SQL | SQLPLUS |
| 是操作所有关系型数据库的规则，是第四代语言  交互性强，非过程化，无需关注如何实现 | Oracle自带的一款工具所指定的命令 |
| 关键字不能简写 | 关键字可以简写 |
| 大小写不敏感，提倡大写 | 大小写不敏感，提倡大写 |
| 必须以分号结束 | 可以不用分号结束 |
|  | 提供了&进行动态操作 |

# SQLPLUS常用命令

|  |  |
| --- | --- |
| 以超级管理员角色登录 | SQLPLUS / AS sysdba |
| 以普通用户角色登录 | SQLPLUS username/password ? |
| 再连接上某个用户 | CONN username/passowrd |
| 设置显示的列宽 | COLUMN col\_name FORMAT a4 （字符类型和日期类型，这里表示显示4位） |
| COLUMN col\_name FORMAT 9999 （数字类型，这里表示显示4位） |
| 设置一页显示多少条记录 | SET PAGESIZE pagesize |
| 执行最近一次的SQL语句 | / |
|  | set timing on |
| 清屏 | HOST CLS |
| 保存SQL语句到硬盘 | SPOOL filepath; SPOOL OFF; |
| 读取SQL文件并执行SQL语句 | @ filepath; |

# DDL

|  |  |
| --- | --- |
| 创建序列/索引 | CREATE SEQUENCE seq\_name START WITH i INCREMENT BY j [MAXVALUE m] [MINVALUE n] |
| CREATE INDEX index\_name ON table\_name(column1 …) |
| 重建索引 | ALTER INDEX index\_name REBUILD |
| 创建同义词 | CREATE SYNONYM synonym\_name FOR original\_name |

# 回收站操作

|  |  |
| --- | --- |
| 删除表 | DROP TABLE table\_name; |
| 从回收站将表闪回 | FLASHBACK TABLE table\_name TO BEFORE DROP; |
| 查看回收站 | SHOW RECYCLEBIN; |
| 清空回收站 | PURGE RECYCLEBIN; |
| 彻底删除表 | DROP TABLE table\_name PURGE; |

# DCL

|  |  |  |
| --- | --- | --- |
| 授予权限 | GRANT | GRANT create view TO ljh; |
| 收回权限 | REVOKE | REVOKE select any table FROM ljh; |
| 创建用户 | CREATE USER | |

|  |  |
| --- | --- |
| 给普通用户解锁 | ALTER USER username ACCOUNT UNLOCK; |
| 给普通用户修改密码 | ALTER USER username IDENTIFIED BY password; |
| 修改自己的密码 | PASSWORD; |
| 查看当前用户是谁 | SHOW USER; |
| 查询自己的权限 | select \* from user\_sys\_privs; |

# 子查询

|  |  |
| --- | --- |
| 查询条件未知的事物 | |
| 相关子查询 | 在主查询中，每查询一条记录，需要重新做一次子查询  子查询的条件依赖于主查询的某个属性值时：EXISTS |
| 非相关子查询 | 在主查询中，子查询只需要执行一次，子查询结果不再变化，供主查询使用  单行单列：=, <, > 多行单列：IN, ALL, ANY |

# DECODE函数

|  |  |
| --- | --- |
| DECODE | CASE |
| select ename, job, sal,  DECODE(job,  'MANAGER', sal\*1.2,  'ANALYST', sal\*1.1,  sal  )bonus from emp; | select ename, job, sal,  CASE job WHEN 'MANAGER' THEN sal\*1.2  WHEN 'ANALYST' THEN sal\*1.1  ELSE sal END  bonus from emp; |

|  |
| --- |
| 查询工资高于2000，和工资高于3000，分别有多少人？ |
| select sum(case when sal>2000 then 1 else 0 end),  sum(case when sal>3000 then 1 else 0 end) from emp; |

# 视图

|  |  |  |  |
| --- | --- | --- | --- |
| 视图也被称作虚表，对应于一条SELECT语句；  视图本身并不包含任何数据，只包含映射到基表的一个查询语句，当基表数据发生变化，视图数据也随之变化  视图必须写别名； | | | |
| 简单视图 | | | 复杂视图 |
| 基于单表 | | | 基于多个表 |
| 不包含函数运算、表达式或分组函数 | | | 包含了单行函数、表达式、分组函数 |
|  | | | 必须定义列名 |
| DML操作 | 1. 简单视图能够进行DML操作（简单视图没有包含基表中非空列时，不能执行INSERT操作） 2. DELETE操作时，只能删除现有视图里能查到的记录 3. 视图定义中包含了函数、表达式、分组、DISTINCT或ROWNUM，不允许执行DML操作 | | |
| WITH CHECK OPTION | | 通过视图所做的修改，必须在视图的可见范围内 | |
| READ ONLY | | 只读 | |

# Oracle序列

|  |  |
| --- | --- |
| 一种用来生成唯一数字值的数据库对象，可以为一个表或多个表提供主键值 | |
| seq.NEXTVAL | 获取序列的下个值 |
| seq.CURRVAL | 获取序列的当前值，序列创建后必须先执行一次NEXTVAL，才能使用CURRVAL |

# 基于函数的索引

|  |
| --- |
| CREATE INDEX … ON table\_name(UPPER(name)) |

# ROWID

|  |
| --- |
| 1. 18位大小写加数字，唯一表示该条记录在DBF文件中的位置 2. 可以参与=和like比较，用单引号将rowid的值包起来 |

# 约束

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NOT NULL | | UNIQUE | PRIMARY KEY | FOREIGN KEY | CHECK |
| NN | | UK | PK | FK | *CK* |
| 允许是NULL值 | 一个表一个 |
| id NUMBER(6) CONSTRAINT emp\_id\_uk UNIQUE | | | |  | |
| id NUMBER(6), CONSTRAINT emp\_id\_uk UNIQUE(id) | | | |
| 主键选取 | 1. 主键应是对系统无意义的数据 2. 主键不应包含动态变化的数据，如时间戳、创建时间列等 3. 主键尽量建立在单列上 | | | | |

# PL/SQL

|  |  |
| --- | --- |
| Oracle数据库服务器对SQL99的规则做了符合自身的特点的扩展，叫做PL/SQL(过程化SQL)，强调过程 | |
| 1. 因为SQL是第四代命令式语言，无法显式处理过程化的业务，所以得用一个过程化程序设计语言弥补SQL的不足 2. 提高性能 | |
| 匿名PL/SQL块 | [declare] 声明（变量，游标，子程序，和其他元素）  begin 可执行命令(DML/TCL)  [exception] 异常处理  end; |

# PL/SQL标识符

|  |  |  |  |
| --- | --- | --- | --- |
| % | 属性索引 | := | 赋值运算符 |
| . | 组件选择 |  |  |

# PL/SQL程序单元

|  |
| --- |
| 1)PL/SQL块 2)函数 3)包 4)包体 5)过程 6)触发器 7)类型 8)类型体 |

# PL/SQL数据类型和变量

|  |  |  |  |
| --- | --- | --- | --- |
| emp.ename%TYPE | 与表的某字段类型相同 | emp%ROWTYPE | 与表或游标的所有字段类型相同 |
| SELECT name INTO c\_name from test where id = c\_id; | | 使用SELECT INTO语句指定SQL查询结果为PL/SQL变量 | |

# PL/SQL DBMS输出

|  |  |
| --- | --- |
| DBMS\_OUTPUT是一个内置的软件包，能够显示输出显示调试信息，并从PL/ SQL块，子程序，包和触发器发送消息 | |
| dbms\_output.put\_line(…); | 用于输出字符串自动换行 |
| set serveroutput on/off; | 打开/关闭显示PL/SQL程序的执行结果 |

# PL/SQL 条件控制

|  |  |  |
| --- | --- | --- |
| IF condition THEN  S;  END IF; | IF condition THEN  S1;  ELSE  S2;  END IF; | IF condition THEN  S1;  ELSIF condition THEN (可以多个)  S2;  ELSE  S3;  END IF; |
| CASE语句参考SQL99 | | |

# PL/SQL循环语句

|  |  |  |
| --- | --- | --- |
| LOOP  Sequence of statements;  END LOOP; | WHILE condition LOOP  sequence\_of\_statements;  END LOOP; | FOR counter IN first .. last LOOP  sequence\_of\_statements;  END LOOP; |

# PL/SQL EXIT语句

|  |  |
| --- | --- |
| EXIT; | EXIT WHEN cemp%notfound; |

# PL/SQL存储过程

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 一组为了完成特定功能的SQL语句集，经过第一次编译后再次调用不需要再次编译，用户通过存储过程的名字来执行 | | | | | |
| CREATE [OR REPLACE] PROCEDURE procedure\_name  [(parameter\_name [IN | OUT | IN OUT] type [, ...])]  {IS | AS}  BEGIN  < procedure\_body >  END procedure\_name; | | | 参数  模式 | IN | 表示该值将被从外部传递 |
| OUT | 表示该参数将被用于从过程返回一个值到外面 |
| 执行存储过程 | EXECUTE procedure\_name | | | | |
| 调用存储过程 | BEGIN procedure\_name END;/ | | | | |
| 方法传递参数  ??? | 位置表示法 | procedure\_name(a, b, c, d); | | | |
| 命名表示法 | procedure\_name(x=>a, y=>b, z=>c, m=>d); | | | |
| 混合表示法 | procedure\_name(a, b, c, m=>d); （位置表示法要先写） | | | |

# PL/SQL函数

|  |  |
| --- | --- |
| 数据库已定义的方法，它接受参数并返回某种类型的值 | |
| CREATE [OR REPLACE] FUNCTION function\_name  [(parameter\_name [IN | OUT | IN OUT] type [, ...])]  RETURN return\_datatype  {IS | AS}  BEGIN  < function\_body >  END [function\_name]; | |
| 调用函数 | function\_name() |

# 存储过程、函数

|  |  |
| --- | --- |
| 存储过程 procedure | 函数 function |
| 涉及特定表或其它对象的任务 | 不涉及特定用户表 |
| 不需要描述返回类型 | 要描述返回类型 |
| 一般作为独立的部分来执行 | 必须作为表达式的一个部分来调用 |
| 可以返回参数 | 只能返回值或者表对象 |
| 可以通过out/in out返回0个或多个值 | 通过return返回一个值。改值要与声明部分一致，也可以是通过out类型参数带出变量 |

# PL/SQL游标

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 定义 | 一种控制结构，可以帮助我们处理多条记录。不是一种数据类型 | | | | |
| 隐式游标 | Oracle在一个SQL语句的执行，当没有显式游标的语句隐式游标时自动创建。每当发出一个DML语句，隐式游标与此语句关联。 sql%notfound | | | | |
| 显式游标 | 步骤 | 声明游标 | 无参光标 | CURSOR c\_emp IS SELECT ename, sal from emp; | |
| 有参光标 | CURSOR c\_emp(c\_deptno emp.deptno&type) IS SELECT ename, sal from emp where deptno = c\_deptno; | |
| 打开游标 | 无参光标 | OPEN c\_emp; | |
| 有参光标 | OPEN c\_emp(deptno); | |
| 获取游标 | FETCH c\_emp INTO c\_ename, c\_sal; | | |
| 关闭游标 | CLOSE c\_emp; | | |
| 游标属性 | 1)%FOUND | | | | 2)%NOTFOUND |
| 3)ISOPEN 执行完SQL语句之后，隐式会自动关闭 | | | | 4)%ROWCOUNT 返回DML和DQL语句影响的行数 |

# PL/SQL异常

|  |  |  |
| --- | --- | --- |
| 语法 | [EXCEPTION]  WHEN exception\_name THEN  statement; | |
| 常见内置异常 | NO\_DATA\_FOUND | 没有找到数据 |
| TOO\_MANY\_ROWS | SELECT INTO 语句匹配多行 |
| ZERO\_DIVIDE | 被0除 |
| VALUE\_ERROR | 算术或转换错误 |
| TIMEOUT\_ON\_RESOURCE | 在等待资源时发生超时 |
| 用户自定义异常 | [DECLARE]  my-exception EXCEPTION; | |

**PL/SQL触发器**

|  |  |  |
| --- | --- | --- |
| 存储在数据库中的块，这些块一旦被构造后，就可以多次执行，当触发它的事件发生时，调用触发器。 | | |
| CREATE [OR REPLACE ] TRIGGER trigger\_name  {BEFORE | AFTER | INSTEAD OF }  {INSERT [OR] | UPDATE [OR] | DELETE}  [OF col\_name]  ON table\_name  [REFERENCING OLD AS o NEW AS n]  [FOR EACH ROW]  WHEN (condition)  …匿名PL/SQL块 | INSTEAD OF | 用于在视图中创建触发器 |
| OF col\_name | 指定被更新的列名 |
| REFERENCING OLD AS o NEW AS n | ??? |
| FOR EACH ROW | 行级触发器  :new 为一个引用最新的列值  :old 为一个引用以前的列值 |
| WHEN (condition) | 仅适用于行级触发器 |

**Oracle常用数据字典**

|  |  |
| --- | --- |
| dba\_ | 数据库实例的所有对象信息 dba\_users |
| user\_ | 用户的对象信息 user\_segments user\_part\_tables ? |
| all\_ | 用户被授权访问的对象信息 |
| v$\_ | 当前实例的动态视图 |
| gv\_ | 分布式环境下所有实例的动态视图 |