|  |
| --- |
|  |

|  |
| --- |
| GP开发文档 |

目录

[第一章 GP开发规范 2](#_Toc7066)

[1.1 对象数管控 2](#_Toc26336)

[1.2 连接数管控 2](#_Toc17258)

[1.3 建表管控 2](#_Toc1194)

[1.4 查询管控 3](#_Toc18051)

[第二章 简单的PSQL命令 5](#_Toc4506)

[2.1 psql常用参数 5](#_Toc23069)

[2.2 psql常用内部命令 5](#_Toc12336)

[第三章 语法 6](#_Toc22610)

[3.1 DDL 6](#_Toc30028)

[3.2 DML 7](#_Toc24810)

[3.3 DCL 11](#_Toc3673)

[第四章 PL/SQL和PL/PGSQL 13](#_Toc19813)

[4.1 匿名程序块 13](#_Toc12715)

[4.2 替代变量 13](#_Toc31304)

[4.3 字符串分隔符 14](#_Toc29457)

[4.4 %TYPE属性 14](#_Toc14343)

[4.5 %ROWTYPE属性 16](#_Toc9636)

[4.6 IF语句 17](#_Toc2044)

[4.7 CASE表达式和CASE语句 20](#_Toc32090)

[4.8 LOOP循环 22](#_Toc11845)

[4.9 WHILE循环 23](#_Toc30182)

[4.10 FOR循环 24](#_Toc10124)

[4.11 CONTINUE 24](#_Toc29166)

[4.12 SQL游标 25](#_Toc18941)

[4.13 异常处理 30](#_Toc23712)

[4.14 存储过程和函数 30](#_Toc21237)

[第五章 常用的数据字典表 33](#_Toc10533)

[5.1 ORACLE常用数据字典表 33](#_Toc28471)

[5.2 GP查询字段信息 34](#_Toc26299)

[5.3 GP查询主键约束 35](#_Toc1767)

[5.4 GP查询建索引语句 36](#_Toc12506)

[5.5 GP查询分布键 36](#_Toc28787)

[5.6 GP查询表大小 36](#_Toc6666)

# 第一章 GP开发规范

## 1.1 对象数管控

严格控制数据库对象的数量（包括表，视图，索引，函数等），控制在2-3万以内性能最佳，最好不要超过5万，超过10万以后GP性能下降明显；

1. ***不允许建索引（包括普通索引、唯一索引、主键等）；***
2. ***临时表对象用完即时删除；***
3. ***尽量不要建分区表，分区表会导致对象数急剧上升，建分区表考虑以几个问题：表是否足够大（上亿）？目前性能瓶颈？查询条件是否满足分区条件？是否需要滚动历史数据？数据能否被均匀分拆；***
4. ***默认使用行存，列存会急剧增加对象数；***

## 1.2 连接数管控

GP建议连接数控制在200个左右，连接数超过250个以后，GP性能明显下降，根据人数限制每个开发连接数个数，目前建议开发用户4个连接数；

需要配置后台调度程序定时每天中午13点和晚上23点，杀掉空闲超过6小时的连接。

1. ***开发人员GP数据库连接数不超过4个；***

## 1.3 建表管控

分布键会影响后续查询效率，分布键没设置好，会导致数据倾斜或者查询时数据重分布；

1. ***建表时必须手动指定分布键，优先主键或者能唯一识别记录的字段，其次经常关联的字段（前提是相对分布均匀才考虑），最后考虑随机分布***
2. ***分布键原则上为1个字段，尽量不要超过3个字段；***

GP中不加””时，不区分大小写，GP在数据字典中默认存为小写，如果需要区分大小写或者表名中存在特殊字符，请加上””;

1. ***表名尽量使用数字、字母、下划线，不建议使用特殊字符；***

GP中默认建Heap表（堆表），不能压缩，只能行存，适合数据量不大的表（维表等）；AO（Append Optimized 追加优化表），可以压缩，对批量插入做了优化，适合大表（事实表等）；

1. ***千万行级别以下的表建Heap表，千万行以上级别的表建oa表，固定zstd压缩，压缩等级3，***

***WITH (APPENDONLY=true, COMPRESSTYPE=zstd, COMPRESSLEVEL=3)；***

GP中标识符长度不超过63，ORACLE中标识符长度不超过30；

1. ***若要兼容Oracle，GP的标识符应不超过30；***

GP中char和varchar没有性能差异；

1. ***varchar类型替代char类型；***

## 1.4 查询管控

GP中通过配置的模式搜索路径对对象进行搜索，默认搜索 "$user",public;

1. ***引用其它schema下的对象时，需要追加schemaname，例如：tool.tmp;***

GP中的除法运算需特别小心，***select 1/2*** 的结果是***0***；

1. ***涉及到复合运算并且包含除法运算时，请在分子上添加\*1.0，例如：select 1/2 请改写成 select 1\*1.0/2；***

其它规范；

1. ***避免使用标量子查询，效率很低，并且容易造成表的死锁，需改成关联查询，例如：select a.id,(select b.name from tmp1 b where a.id = b.id) as name from tmp a 改成 select a.id,b.name from tmp a, tmp1 b where a.id = b.id ；***
2. ***一个sql中多表关联不要超过10张表；***
3. ***sql语句全部小写；***
4. ***清空表数据使用truncate table 替换 delete from；***
5. ***每个函数应在函数头部注释函数的创建者、功能、时间等；***

# 简单的PSQL命令

## 2.1 psql常用参数

|  |  |
| --- | --- |
| psql -d etl -E -U tool -W -h 10.75.41.115 -p 5432 | 以tool用户登录到10.75.41.115主机上5432端口下的etl数据库，并且回显\d对应的sql语句 |
| psql -d etl -U tool -W -h 10.75.41.115 -p 5432 -f xxxx.psql | 以tool用户登录到10.75.41.115主机上5432端口下的etl数据库执行xxxx.psql脚本 |
| -d | 指定数据库名 |
| -E | 回显由\d和其他反斜杠命令生成的实际查询 |
| -U | 指定用户名 |
| -W | 强制提示输入密码 |
| -h | 主机名 |
| -p | 端口 |
| -f | 执行sql脚本 |

## 2.2 psql常用内部命令

|  |  |
| --- | --- |
| \connect etl tool; | 以tool用户切换到etl数据库 |
| \q | 退出psql客户端 |
| \df | 查看所有的函数 |
| \df tool.\* | 查看tool下的函数 |
| \dn | 查看所有的schemas |
| \dt tool.\* | 查看schema为tool下的所有的表 |
| \d | 展示表的信息 |
| \d+ | 比前面的\d 多了size和despription |
| \dv gp\_toolkit.gp\_resq\* | 查看资源组的视图 |
| \du+ | 查看用户信息 |
| \set | 设置替代变量 |
| \! | 执行外部命令 |

# 语法

## 3.1 DDL

|  |  |  |
| --- | --- | --- |
| 操作 | ORACLE | GP |
| 创建非分区表 | **CREATE** **TABLE** test**(**  id NUMBER**,**  name VARCHAR2**(**40**)**  **);** | **create** **table** test(  id **int**,  **name** **varchar**(40)  ) **distributed** **by** (id); |
| 创建分区表 | **CREATE** **TABLE** test**(**  id NUMBER**,**  name VARCHAR2**(**40**),**  birth\_day DATE  **)** **PARTITION** **BY** **RANGE(**birth\_day**)(**  **PARTITION** p20090101 **VALUES** **LESS** **THAN** **(TO\_DATE(**'20090102'**,**'yyyymmdd'**)),**  **PARTITION** p20090102 **VALUES** **LESS** **THAN** **(TO\_DATE(**'20090103'**,**'yyyymmdd'**)),**  **PARTITION** p20090103 **VALUES** **LESS** **THAN** **(TO\_DATE(**'20090104'**,**'yyyymmdd'**))**  **);** | **create** **table** test(id **int**,  **name** **varchar**(40),  birth\_day **date**  ) **distributed** **by** (id)  **partition** **by** **range**(birth\_day)  (**partition** p20090101 **start**(**date** '20010101') **inclusive**,  **partition** p20090102 **start**(**date** '20090102') **inclusive**,  **partition** p20090103 **start**(**date** '20090103') **inclusive**  **end** (**date** '20090104') **exclusive**); |
| 分区表增加分区 | **ALTER** **TABLE** test **ADD** **PARTITION** p20090104 **VALUES** **LESS** **THAN** **(TO\_DATE(**'20090105'**,**'yyyymmdd'**));** | **alter** **table** test  **add** **partition** p20090104 **start** (**date** '2009-01-04') **inclusive**  **end** (**date** '2009-01-05')  **exclusive**; |
| 分区表删除分区 | **ALTER** **TABLE** test **DROP** **PARTITION** p20090104**;** | **alter** **table** test **drop** **partition** p20090104; |
| 分区表截断分区 | **ALTER** **TABLE** test **TRUNCATE** **PARTITION** p20090101**;** | **alter** **table** test  **truncate**  **partition** p20090101; |
| 分区表分裂分区 | **ALTER** **TABLE** test SPLIT **PARTITION** p200905\_mon **AT** **(TO\_DATE(**'20090515'**,**'yyyymmdd'**))**  **INTO** **(PARTITION** p200905\_01**,PARTITION** p200905\_02**);** | **alter** **table** test **split** **partition** p200905\_mon **at** ('20090515')  **into**  (**partition** p200905\_01,  **partition** p200905\_02); |
| 删除表 | **DROP** **TABLE** test**;** | **drop** **table** **if** **exists** test; |
| 重命名表 | **rename** a **to** b**;** | **alter** **table** a **rename** **to** b; |
| 截断表 | **TRUNCATE** **TABLE** test**;** | **truncate** **table** test; |
| 创建索引 | **CREATE** **INDEX** idx\_test **ON** test**(**id**);** | **create** **index** idx\_test **on** test(id); |
| 创建主键 | **ALTER** **TABLE** test **ADD** **CONSTRAINT** pk\_test **PRIMARY** **KEY** **(**id**);** | **alter** **table** test **add** **constraint** pk\_test **primary** **key** (id); |
| 删除主键 | **ALTER** **TABLE** test **DROP** **CONSTRAINT** pk\_test**;** | **alter** **table** test **drop** **constraint if** **exists** pk\_test; |
| 删除索引 | **DROP** **INDEX** IND\_TEST**;** | **drop** **index** **if** **exists** ind\_test; |
| 添加字段 | **ALTER** **TABLE** test **add** **(**  column1 number**,**  column2 varchar2**(**100**)**  **);** | **alter** **table** test  **add** column1 **numeric**,  **add** column2 **varchar**(10); |
| 修改字段 | **ALTER** **TABLE** test **modify** **(**column1 VARCHAR2**(**100**));** | **alter** **table** test **alter** **column** column1 **type** **varchar**(100);  **alter** **table** test **alter** column1 **type** **varchar**(100); |
| 删除字段 | **ALTER TABLE test DROP COLUMN column1;** | **alter** **table** test **drop** **column** **if** **exists** column1;  **alter** **table** test **drop** **if** **exists** column1; |
| 字段重命名 | **ALTER TABLE test RENAME COLUMN column2 TO column1;** | **alter** **table** test **rename** **column** column2 **to** column1;  **alter** **table** test **rename** column2 **to** column1; |
| 创建视图 | **CREATE** **VIEW** v\_test **AS** **SELECT** **\*** **FROM** test**;** | **create** **view** v\_test **as**  **select**  \*  **from**  test; |

## 3.2 DML

|  |  |  |
| --- | --- | --- |
| 操作 | ORACLE | GP |
| 插入记录 | **INSERT** **INTO** test **VALUES** **(**001**,**'SUNWG'**,TO\_DATE(**'2009-01-01'**,**'YYYY-MM-DD'**));** | **Insert into** test **values** (001,'SUNWG','2009-01-01'); |
| 删除记录 | **DELETE** **FROM** test **WHERE** id **=** 001**;** | **delete** **from** test **where** id = 001; |
| 关联更新记录 | **UPDATE** test a **SET** a**.**column1 **=** **(SELECT** b**.**column1 **FROM** test1 b **WHERE** a**.**id **=** b**.**id**);** | **update** test a **set** column1 = b.column1 **from** test1 b **where** a.id = b.id; |
| 查询表中前10条记录 | **SELECT** **\*** **FROM** test **WHERE** **ROWNUM** **<** 10**;** | **select** \* **from** test **limit** 10; |
| 两表内关联 | **SELECT** **\*** **FROM** test1 a**,** test2 b **WHERE** a**.**id **=** b**.**id**;** | **select** \* **from** test1 a **inner** **join** test2 b **on** (a.id = b.id); |
| 两表左外关联 | **SELECT** **\*** **FROM** test1 a**,** test2 b **WHERE** a**.**id **=** b**.**id(+)**;** | **select** \* **from** test1 a **left join** test2 b **on** (a.id = b.id); |
| 事务提交 | **INSERT** **INTO** test **VALUES** **(**001**,**'SUNWG'**,TO\_DATE(**'2009-01-01'**,**'YYYY-MM-DD'**));**  **COMMIT;** | **begin**;  **insert** **into** test **values** (001,'SUNWG','2009-01-01');  **commit**; |
| 事务回滚 | **INSERT** **INTO** test **VALUES** **(**001**,**'SUNWG'**,TO\_DATE(**'2009-01-01'**,**'YYYY-MM-DD'**));**  **ROLLBACK;** | **begin**;  **insert** **into** test **values** (001,'SUNWG','2009-01-01');  **rollback**; |
| 数据类型转换 | **SELECT** **to\_number(**'101'**)** **FROM** dual**;** | **select** **cast**('101' **as** **numeric**);  **select** '101' :: **numeric**; |
| 除法运算 | **SELECT** 1**/**2 **FROM** dual**;** | **select** 1.0/2; |
| 并集 | **SELECT** 1 **FROM** dual  **UNION**  **SELECT** 2 **FROM** dual**;**  **SELECT** 1 **FROM** dual  **UNION** **ALL**  **SELECT** 2 **FROM** dual**;** | **select** 1  **union**  **select** 2;  **select** 1  **union** **all**  **select** 2; |
| 交集 | **SELECT** 1 **FROM** dual  **INTERSECT**  **SELECT** 1 **FROM** dual**;** | **select** 1  **intersect**  **select** 1;  **select** 1  **intersect** **all**  **select** 1; |
| 差集 | **SELECT** 1 **FROM** dual  **MINUS**  **SELECT** 2 **FROM** dual**;** | **select** 1  **except**  **select** 2;  **select** 1  **except** **all**  **select** 2; |
| 子查询 | **SELECT** **\*** **FROM** **(SELECT** **\*** **FROM** test**);** | **select** \* **from** (**select** \* **from** test) **as** a; |
| 日期抽取 | **SELECT** **to\_char(SYSDATE,**'yyyy-mm-dd'**)** **FROM** dual**;** | **select** **extract**(century **from** **now**()); --世纪  **select** **extract**(**year** **from** **now**()); --年  **select** **extract**(dow **from** **now**()); --星期几  **select** **extract**(**quarter** **from** **now**()); --季度 |
| 时间截取 | **SELECT** **TRUNC(SYSDATE,**'hh'**)** **FROM** dual**;** | **select** **date\_trunc**('hour', **now**()); |
| null和’’ | **SELECT** '123'**||**'' **FROM** dual**;** --'123'  **SELECT** '123'**||NULL** **FROM** dual**;**--'123' | **select** '123'||''; --'123'  **select** '123'||**null**; --null |
| delete 剔重 | **DELETE** **FROM** test a **WHERE** rowid **<** **(**  SEELCT **MAX(**rowid**)** **from** test b  **WHERE** a**.**id **=** b**.**id  **);** | **delete** **from** test a **where** ctid < (  **select** **max**(ctid) **from** test b  **where** a.id = b.id  **and** a.gp\_segment\_id = b.gp\_segment\_id  ); |
| 当前的系统时间 | **SELECT** **SYSDATE,current\_timestamp** **FROM** dual**;** | **select** **current\_timestamp**,**now**(); |
| 当前时间向前推一天 | **SELECT** **SYSDATE**-1 **FROM** dual**;** | **select** **current\_timestamp** - **interval** '1 day'; |
| 当前时间向前推一个月 | **SELECT** **add\_months(SYSDATE,**-1**)** **FROM** dual**;** | **select** **current\_timestamp** - **interval** '1 month'; |
| 当前时间向前推一年 | **SELECT** **add\_months(SYSDATE,**-12**)** **FROM** dual**;** | **select** **current\_timestamp** - **interval** '1 year'; |
| 当前时间向前推一小时 | **SELECT** **SYSDATE**-1**/**24 **FROM** dual**;** | **select** **current\_timestamp** - **interval** '1 hour'; |
| 当前时间向前推一分钟 | **SELECT** **SYSDATE**-1**/**24**/**60 **FROM** dual**;** | **select** **current\_timestamp** - **interval** '1 min'; |
| 当前时间向前推30秒 | **SELECT** **SYSDATE**-1**/**24**/**60**/**60**\***30 **FROM** dual**;** | **select** **current\_timestamp** - **interval** '60 second'; |
| 正则匹配 | **SELECT** 1 **FROM** dual **WHERE** regexp\_like **(**'123'**,**'[0-9]+'**);** | **select** 1 **where** '123' ~ '[0-9]+'; |
| 正则截取 | **SELECT** regexp\_substr**(**'你好123'**,**'[0-9]+'**)** **FROM** dual**;** | **select** **substring** ('你好123' **from** '[0-9]+'); |
| 分区表查询指定分区 | **SELECT** **\*** **FROM** test **PARTITION** **(**p20090101**);** | **select** \* **from** test\_1\_prt\_p20090101; |
| 字符串截取涉及非正数位置开始截取 | **SELECT** **SUBSTR(**'12345'**,**-1**,**3**)** **FROM** dual**;** | **select** substr('12345',-1,3);  --等价于  **select** substr('12345',1,1); |
| 指定子字符串第一次出现的位置 | **SELECT** **INSTR(**'123245'**,**'24'**,**1**,**1**)** **FROM** dual**;** | **select** **strpos**('123245','24');  **select** **position** ('24' **in** '123245'); |
| 指定日期间相差的月份 | **SELECT** **months\_between(to\_date(**'202108'**,**'yyyymm'**),to\_date(**'202107'**,**'yyyymm'**))** **FROM** dual**;** | **select** **extract**(**month** **from** **age**(**to\_date**('202108','yyyymm'),**to\_date**('202107','yyyymm'))); |
| 判空 | **SELECT** **NVL(NULL,**1**)** **FROM** dual**;**  **SELECT** **NVL2(NULL,**1**,**2**)** **FROM** dual**;** | **select** **coalesce** (**null**::**integer**,1); |
| 序号 | **SELECT** **ROWNUM,**NAME **FROM** **(**  **SELECT** 'ha' NAME **FROM** dual  **UNION** **ALL**  **SELECT** 'haha' **FROM** dual  **);** | **select** **row\_number**() **over**(**order** **by** 1) xh,**name** **from** (  **select** 'ha' **name**  **union** **all**  **select** 'haha'  ) a; |
| to\_char数字转换为字符 | **SELECT** **to\_char(**123**)** **FROM** dual**;** | **select** **to\_char**(123,'fm999'); |
| 字符转换日期 | **SELECT** **to\_date(**'20210827'**,**'yyyymmdd'**)** **FROM** dual**;** | **select** **to\_date**('20210827','yyyymmdd');  **select** **to\_timestamp**('20210827','yyyymmdd');  **select** **cast** ('20210827' **as** **date**);  **select** **cast** ('20210827' **as** **timestamp**); |
| 字符转换为数字 | **SELECT** **to\_number(**'20210827'**)** **FROM** dual**;** | **select** **cast**('20210827' **as** **numeric**); |
| 字符串分割 | **SELECT**  kind**,**  listagg**(**subkind**,**';'**)**WITHIN **GROUP(ORDER** **BY** subkind**)** newkind  **FROM** **(**  **SELECT** '水果' kind**,**'苹果' subkind **FROM** dual  **UNION** **ALL**  **SELECT** '水果'**,**'梨' **FROM** dual  **UNION** **ALL**  **SELECT** '水果'**,**'香蕉' **FROM** dual  **UNION** **ALL**  **SELECT** '动物'**,**'狗' **FROM** dual  **UNION** **ALL**  **SELECT** '动物'**,**'猫' **FROM** dual  **)** a  **GROUP** **BY** kind**;** | **select**  kind,  **string\_agg**(subkind,';' **order** **by** subkind) newkind  **from** (  **select** '水果' kind,'苹果' subkind  **union** **all**  **select** '水果','梨'  **union** **all**  **select** '水果','香蕉'  **union** **all**  **select** '动物','狗'  **union** **all**  **select** '动物','猫'  ) a  **group** **by** kind; |
| 递归查询 | **start** **with…connect** **by** | **with** **recursive** |
| 合并 | **DROP** **TABLE** example**;**  **CREATE** **TABLE** example **(**  example\_id NUMBER**,**  example\_name VARCHAR2**(**200**)**  **);**  **INSERT** **INTO** example **VALUES** **(**1**,**'样例1'**);**  **INSERT** **INTO** example **VALUES** **(**2**,**'样例2'**);**  **COMMIT;**  **MERGE** **INTO** example a  **USING** **(**  **SELECT** 1 example\_id**,** 'MERGE后的样例1' example\_name **FROM** dual  **UNION**  **SELECT** 2 example\_id**,** 'MERGE后的样例2' example\_name **FROM** dual  **UNION**  **SELECT** 3 example\_id**,** 'MERGE后的样例3' example\_name **FROM** dual  **)** b  **ON** **(**a**.**example\_id **=** b**.**example\_id**)**  **WHEN** MATCHED **THEN**  **UPDATE** **SET**  a**.**example\_name **=** b**.**example\_name  --限制当 example\_id = 1 的时候才做更新  **WHERE**  a**.**example\_id **=** 1  **WHEN** **NOT** MATCHED **THEN**  **INSERT** **(**a**.**example\_id**,**a**.**example\_name**)**  **VALUES** **(**b**.**example\_id**,**b**.**example\_name**);**  **COMMIT;** | **drop** **table** example;  **create** **table** example (  example\_id **numeric**,  example\_name **varchar**(200)  );  **insert** **into** example **values** (1,'样例1');  **insert** **into** example **values** (2,'样例2');  **begin**;  **with** tmp\_a **as** (  **SELECT** 1 example\_id, 'MERGE后的样例1' example\_name  **UNION**  **SELECT** 2 example\_id, 'MERGE后的样例2' example\_name  **UNION**  **SELECT** 3 example\_id, 'MERGE后的样例3' example\_name  )  **update** example a **set** example\_name = b.example\_name  **from** tmp\_a b  **where** a.example\_id = b.example\_id  **and** a.example\_id = 1;  **with** tmp\_a **as** (  **SELECT** 1 example\_id, 'MERGE后的样例1' example\_name  **UNION**  **SELECT** 2 example\_id, 'MERGE后的样例2' example\_name  **UNION**  **SELECT** 3 example\_id, 'MERGE后的样例3' example\_name  )  **insert** **into** example  **select** a.example\_id,a.example\_name  **from** tmp\_a a  **where** **not** **exists** (  **select** 1 **from** example b  **where** a.example\_id = b.example\_id  );  **commit**; |

## 3.3 DCL

|  |  |  |
| --- | --- | --- |
| 操作 | ORACLE | GP |
| schema授权 | 无 | **grant** **usage**,**create** **on** **schema** tool **to** tool; |
| 表授权查询、插入、更新 | **GRANT** **SELECT,INSERT,UPDATE** **ON** test **TO** tool**;** | **grant** **select**,**insert**,**update** **on** **table** test **to** tool; |
| 表授权所有用户访问 | **GRANT** **SELECT** **ON** test **TO** **public;** | **grant** **select** **on** **table** test **to** public; |
| 存储过程授权 | **GRANT** **EXECUTE,**DEBUG **ON** delete\_example **TO** tool**;** | **grant** **execute** **on** **function** delete\_example() **to** tool; |

# PL/SQL和PL/PGSQL

## 4.1 匿名程序块

|  |  |  |
| --- | --- | --- |
| 数据库 | 定义 | 举例 |
| ORACLE | **DECLARE** --可选  --variables,cursors,user-defined exceptions  **BEGIN** --强制的（必须的）  --一个或多个SQL语句  --一个或多个PL/SQL语句  **EXCEPTION** --可选  --当错误发生是要进行的处理  **END;** --强制的（必须的）  **/** | **DECLARE**  **var** VARCHAR2**(**100**)** **:=** 'hello world!'**;**  **BEGIN**  dbms\_output**.**put\_line**(var);**  **END;**  **/** |
| GP | **do** **$$**  **declare** --可选  --variables,cursors,user-defined exceptions  **begin** --强制的（必须的）  --一个或多个SQL语句  --一个或多个PL/PGSQL语句  **exception** --可选  --当错误发生是要进行的处理  **end$$**; --强制的（必须的） | **do** **$$**  **begin**  **raise** **notice** 'hello world!';  **end$$**; |

## 4.2 替代变量

|  |  |  |
| --- | --- | --- |
| 数据库 | 定义 | 举例 |
| ORACLE(SQL PLUS) | define | define a **=** 1  **SELECT** &a **FROM** dual**;**  **SELECT** &&a **FROM** dual**;** |
| GP(PSQL) | \set | \**set** a 1  **select** **:a**; |

## 4.3 字符串分隔符

|  |  |  |  |
| --- | --- | --- | --- |
| 数据库 | 定义 | 描述 | 举例 |
| ORACLE | ‘’ | 单引号，字符串中有单引号，需加**’**转义 | **SELECT** 'What''s your name?' **FROM** dual**;** |
| GP | ‘’ | 单引号，字符串中有单引号，需加**’**转义 | **select** 'What''s your name?'; |
| ORACLE | q' | q’可以自定义定界符，允许嵌套 | --通过q'定义||为定界符  **SELECT** q'|What's your name?|' **FROM** dual**;**  --通过q'定义//为定界符  **SELECT** q'/What's your name?/' **FROM** dual**;**  --通过q'定义{}为定界符  **SELECT** q'{What's your name?}' **FROM** dual**;**  --通过q'定义[]为定界符  **SELECT** q'[What's your name?]' **FROM** dual**;** |
| GP | $$ $$ | 不允许嵌套 | **select** **$$**What's your name?$$; |

## 4.4 %TYPE属性

1. 使用 %type 属性按照之前已经声明过的变量或者数据库中表的列来声明一个变量；

2. 当存储在一个变量中的值来自于数据库中的表时，最好使用 %type 声明这个变量；

3. 使用 %type 后，当数据库中表的数据类型改变时，无需修改程序。

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | --建表  **CREATE** **TABLE** example **(**  example\_id NUMBER**,**  example\_name VARCHAR2**(**200**)**  **);**  --插入数据  **INSERT** **INTO** example **VALUES** **(**1**,**'样例1'**);**  **INSERT** **INTO** example **VALUES** **(**2**,**'样例2'**);**  **COMMIT;**  --测试%TYPE  **DECLARE**  --用%TYPE属性声明p\_example\_id 是与example表中example\_id相同的数据类型  p\_example\_id example**.**example\_id**%TYPE;**  --用%TYPE属性声明p\_example\_name 是与example表中example\_name相同的数据类型  p\_example\_name example**.**example\_name**%TYPE;**  **BEGIN**  --将从example表中读取的一行数据的两列分别写进p\_example\_id和p\_example\_name变量  **SELECT** **\***  **INTO** p\_example\_id**,**p\_example\_name  **FROM** example  **WHERE** **ROWNUM** **=** 1**;**  --在oracle输出区输出p\_example\_id的值  dbms\_output**.**put\_line**(**p\_example\_id**);**  --在oracle输出区输出p\_example\_name的值  dbms\_output**.**put\_line**(**p\_example\_name**);**  **END;**  **/** |
| GP | --建表  **create** **table** example (  example\_id **numeric**,  example\_name **VARCHAR**(200)  ) **distributed** **by** (example\_id);  --插入数据  **insert** **into** example **values** (1,'样例1');  **insert** **into** example **values** (2,'样例2');  --测试%TYPE  **do** **$$**  **declare**  --用%TYPE属性声明p\_example\_id 是与example表中example\_id相同的数据类型  p\_example\_id example.example\_id%**type**;  --用%TYPE属性声明p\_example\_name 是与example表中example\_name相同的数据类型  p\_example\_name example.example\_name%**type**;  **begin**  --将从example表中读取的一行数据的两列分别写进p\_example\_id和p\_example\_name变量  **select** \*  **into** p\_example\_id,p\_example\_name  **from** example  **limit** 1;  --在gp输出区输出p\_example\_id的值  **raise** **notice** '%',p\_example\_id;  --在gp输出区输出p\_example\_name的值  **raise** **notice** '%',p\_example\_name;  **end$$**; |

## 4.5 %ROWTYPE属性

利用%ROWTYPE属性声明一个能够存储一个表或视图中一整行数据的记录（变量）；

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **DECLARE**  --声明记录类型的变量  exp\_record example**%ROWTYPE;**  **BEGIN**  --将从example表中获取到的一行数据放进已经声明的记录类型的变量exp\_record中  **SELECT**  example\_id**,**  example\_name  **INTO** exp\_record  **FROM** example  **WHERE** **ROWNUM** **=** 1**;**    --输出  dbms\_output**.**put\_line**(**exp\_record**.**example\_id**);**  dbms\_output**.**put\_line**(**exp\_record**.**example\_name**);**  **END;**  **/** |
| GP | **do** **$$**  **declare**  --声明记录类型的变量  exp\_record example%rowtype;  **begin**  --将从example表中获取到的一行数据放进已经声明的记录类型的变量exp\_record中  **select**  example\_id,  example\_name  **into** exp\_record  **from** example  **limit** 1;    --输出  **raise** **notice** '%',exp\_record.example\_id;  **raise** **notice** '%',exp\_record.example\_name;  **end$$**; |

## 4.6 IF语句

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **DECLARE**  score NUMBER**;**  **BEGIN**  score **:=** 100**;**    **IF** score **>=** 90 **AND** score **<=** 100 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are very good！'**);**  **ELSIF** score **>=** 70 **AND** score **<** 90 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are good！'**);**  **ELSIF** score **>=** 60 **AND** score **<** 70 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible！'**);**  **ELSE**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible very much！'**);**  **END** **IF;**    score **:=** 80**;**    **IF** score **>=** 90 **AND** score **<=** 100 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are very good！'**);**  **ELSIF** score **>=** 70 **AND** score **<** 90 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are good！'**);**  **ELSIF** score **>=** 60 **AND** score **<** 70 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible！'**);**  **ELSE**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible very much！'**);**  **END** **IF;**    score **:=** 65**;**    **IF** score **>=** 90 **AND** score **<=** 100 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are very good！'**);**  **ELSIF** score **>=** 70 **AND** score **<** 90 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are good！'**);**  **ELSIF** score **>=** 60 **AND** score **<** 70 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible！'**);**  **ELSE**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible very much！'**);**  **END** **IF;**    score **:=** 50**;**    **IF** score **>=** 90 **AND** score **<=** 100 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are very good！'**);**  **ELSIF** score **>=** 70 **AND** score **<** 90 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are good！'**);**  **ELSIF** score **>=** 60 **AND** score **<** 70 **THEN**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible！'**);**  **ELSE**  dbms\_output**.**put\_line**(**'Your score is ' **||** **to\_char(**score**)** **||** ', you are terrible very much！'**);**  **END** **IF;**  **END;**  **/** |
| GP | **do** **$$**  **declare**  score **numeric**;  **begin**  score := 100;    **if** score >= 90 **and** score <= 100 **then**  **raise** **notice** 'Your score is %, you are very good！',score;  **elsif** score >= 70 **and** score < 90 **then**  **raise** **notice** 'Your score is %, you are good！',score;  **elsif** score >= 60 **and** score < 70 **then**  **raise** **notice** 'Your score is %, you are terrible！',score;  **else**  **raise** **notice** 'Your score is %, you are terrible very much！',score;  **end** **if**;    score := 80;    **if** score >= 90 **and** score <= 100 **then**  **raise** **notice** 'Your score is %, you are very good！',score;  **elsif** score >= 70 **and** score < 90 **then**  **raise** **notice** 'Your score is %, you are good！',score;  **elsif** score >= 60 **and** score < 70 **then**  **raise** **notice** 'Your score is %, you are terrible！',score;  **else**  **raise** **notice** 'Your score is %, you are terrible very much！',score;  **end** **if**;    score := 65;    **if** score >= 90 **and** score <= 100 **then**  **raise** **notice** 'Your score is %, you are very good！',score;  **elsif** score >= 70 **and** score < 90 **then**  **raise** **notice** 'Your score is %, you are good！',score;  **elsif** score >= 60 **and** score < 70 **then**  **raise** **notice** 'Your score is %, you are terrible！',score;  **else**  **raise** **notice** 'Your score is %, you are terrible very much！',score;  **end** **if**;    score := 50;    **if** score >= 90 **AND** score <= 100 **then**  **raise** **notice** 'Your score is %, you are very good！',score;  **elsif** score >= 70 **AND** score < 90 **then**  **raise** **notice** 'Your score is %, you are good！',score;  **elsif** score >= 60 **AND** score < 70 **then**  **raise** **notice** 'Your score is %, you are terrible！',score;  **else**  **raise** **notice** 'Your score is %, you are terrible very much！',score;  **end** **if**;  **end$$**; |

## 4.7 CASE表达式和CASE语句

CASE表达式既可以用在标准SQL中，也可以用在过程中，但是CASE语句只能用在过程中。

|  |  |  |  |
| --- | --- | --- | --- |
| 数据库 | 类型 | 样式 | 举例 |
| ORACLE | CASE表达式 | **CASE** selector  **WHEN** 表达式1 **THEN** 结果1  **WHEN** 表达式2 **THEN** 结果2  **...**  **WHEN** 表达式n **THEN** 结果n  **[ELSE** 结果n+1**]**  **END;** --必须以END结尾 | **SELECT** **CASE** 1 **WHEN** 2 **THEN** 1 **ELSE** 0 **END** **FROM** dual**;** |
| GP | CASE表达式 | **case** selector  **when** 表达式1 **then** 结果1  **when** 表达式2 **then** 结果2  **...**  **when** 表达式n **then** 结果n  **[else** 结果n+1**]**  **end;** --必须以end结尾 | **select** **case** 1 **when** 2 **then** 1 **else** 0 **end**; |
| ORACLE | CASE表达式 | **CASE**  **WHEN** 搜索条件1 **THEN** 结果1  **WHEN** 搜索条件2 **THEN** 结果2  **...**  **WHEN** 搜索条件n **THEN** 结果n  **[ELSE** 结果n+1**]**  **END;** --必须以END结尾 | **SELECT** **CASE** **WHEN** 1 **=** 2 **THEN** 1 **ELSE** 0 **END** **FROM** dual**;** |
| GP | CASE表达式 | **case**  **when** 搜索条件1 **then** 结果1  **when** 搜索条件2 **then** 结果2  **...**  **when** 搜索条件n **then** 结果n  **[else** 结果n+1**]**  **end;** --必须以end结尾 | **select** **case** **when** 1 = 2 **then** 1 **else** 0 **end**; |
| ORACLE | CASE语句 | **CASE** selector  **WHEN** 表达式1 **THEN**  statements1**;**  **WHEN** 表达式2 **THEN**  statements2**;**  **...**  **WHEN** 表达式n **THEN**  statementsn**;**  **[ELSE**  statementsn+1**]**  **END** **CASE;** --必须以END CASE;结尾 | **BEGIN**  --第一种  **CASE** 1  **WHEN** 2 **THEN**  dbms\_output**.**put\_line**(**'1 = 2'**);**  **ELSE**  dbms\_output**.**put\_line**(**'1 <> 2'**);**  **END** **CASE;**    --第二种  **CASE**  **WHEN** 1 **=** 2 **THEN**  dbms\_output**.**put\_line**(**'1 = 2'**);**  **ELSE**  dbms\_output**.**put\_line**(**'1 <> 2'**);**  **END** **CASE;**    **END;**  **/** |
| ORACLE | CASE语句 | **CASE**  **WHEN** 搜索条件1 **THEN**  statements1**;**  **WHEN** 搜索条件2 **THEN**  statements2**;**  **...**  **WHEN** 搜索条件n **THEN**  statementsn**;**  **[ELSE**  statementsn+1**]**  **END** **CASE;** --必须以END CASE;结尾 |
| GP | CASE语句 | **case** selector  **when** 表达式1 **then**  statements1**;**  **when** 表达式2 **then**  statements2**;**  **...**  **when** 表达式n **then**  statementsn**;**  **[else**  statementsn+1**]**  **end case;** --必须以end case;结尾 | **do** **$$**  **begin**  --第一种  **case** 1  **when** 2 **then**  **raise** **notice** '1 = 2';  **else**  **raise** **notice** '1 <> 2';  **end** **case**;    --第二种  **case**  **when** 1 = 2 **then**  **raise** **notice** '1 = 2';  **else**  **raise** **notice** '1 <> 2';  **end** **case**;    **end$$**; |
| GP | CASE语句 | **case**  **when** 搜索条件1 **then**  statements1**;**  **when** 搜索条件2 **then**  statements2**;**  **...**  **when** 搜索条件n **then**  statementsn**;**  **[else**  statementsn+1**]**  **end case;** --必须以end case;结尾 |

## 4.8 LOOP循环

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | /\*  循环输出1到10  \*/  **DECLARE**  --定义一个数，初始值为1  num NUMBER **:=** 1**;**  **BEGIN**  --循环开始  **LOOP**  --输出num  dbms\_output**.**put\_line**(**num**);**  --num+1  num **:=** num **+** 1**;**  --循环退出条件，当num第一次出现比10大的时候退出  **EXIT** **WHEN** num **>** 10**;**  **END** **LOOP;**  **END;**  **/** |
| GP | /\*  循环输出1到10  \*/  **do** **$$**  **declare**  --定义一个数，初始值为1  num **numeric** := 1;  **begin**  --循环开始  **loop**  --输出num  **raise** **notice** '%',num;  --num+1  num := num + 1;  --循环退出条件，当num第一次出现比10大的时候退出  **exit** **when** num > 10;  **end** **loop**;  **end$$**; |

## 4.9 WHILE循环

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | /\*  循环输出1到10  \*/  **DECLARE**  --定义一个数，初始值为1  num NUMBER **:=** 1**;**  **BEGIN**    --先判断退出循环条件，退出循环条件为 num <= 10  **WHILE** num **<=** 10  **LOOP**    --输出num  dbms\_output**.**put\_line**(**num**);**  --num++  num **:=** num **+** 1**;**    **END** **LOOP;**  **END;**  **/** |
| GP | /\*  循环输出1到10  \*/  **do** **$$**  **declare**  --定义一个数，初始值为1  num **numeric** := 1;  **begin**    --先判断退出循环条件，退出循环条件为 num <= 10  **while** num <= 10  **loop**    --输出num  **raise** **notice** '%',num;  --num++  num := num + 1;    **end** **loop**;  **end$$**; |

## 4.10 FOR循环

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | /\*  循环输出1到10  \*/  **BEGIN**    **FOR** i **IN** 1..10 **LOOP**  dbms\_output**.**put\_line**(**i**);**  **END** **LOOP;**  **END;**  **/** |
| GP | /\*  循环输出1到10  \*/  **do** **$$**  **begin**    **for** i **in** 1..10 **loop**  **raise** **notice** '%',i;  **end** **loop**;  **end$$**; |

## 4.11 CONTINUE

CONTINUE退出当前循环进入下一次循环；

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | /\*  循环输出1到4,6到10，不输出5  \*/  **BEGIN**    **FOR** i **IN** 1..10 **LOOP**  **CONTINUE** **WHEN** i **=** 5**;**  dbms\_output**.**put\_line**(**i**);**  **END** **LOOP;**    **FOR** i **IN** 1..10 **LOOP**  **IF** i **=** 5 **THEN**  **CONTINUE;**  **END** **IF;**  dbms\_output**.**put\_line**(**i**);**  **END** **LOOP;**  **END;**  **/** |
| GP | /\*  循环输出1到4,6到10，不输出5  \*/  **do** **$$**  **begin**    **for** i **in** 1..10 **loop**  **continue** **when** i = 5;  **raise** **notice** '%',i;  **end** **loop**;    **for** i **in** 1..10 **loop**  **if** i = 5 **then**  **continue**;  **end** **if**;  **raise** **notice** '%',i;  **END** **LOOP**;  **end$$**; |

## 4.12 SQL游标

1. 普通游标

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **DECLARE**  --声明游标  CURSOR exp\_cursor **IS**  **SELECT** **\*** **FROM** example**;**  --声明基于游标的记录变量  exp\_cursor\_record exp\_cursor**%ROWTYPE;**  **BEGIN**    --如果游标不是打开状态，则打开游标  **IF** **NOT** exp\_cursor**%**ISOPEN **THEN**  **OPEN** exp\_cursor**;**  **END** **IF;**    --循环获取游标的数据到基于游标的记录变量  **LOOP**  **FETCH** exp\_cursor **INTO** exp\_cursor\_record**;**  **EXIT** **WHEN** exp\_cursor**%**NOTFOUND **OR** exp\_cursor**%**NOTFOUND **IS** **NULL;**    dbms\_output**.**put\_line**(**exp\_cursor\_record**.**example\_id **||** '|' **||** exp\_cursor\_record**.**example\_name**);**    **END** **LOOP;**  **CLOSE** exp\_cursor**;**    **END;**  **/** |
| GP | **do** **$$**  **declare**  --声明游标，游标在声明时默认打开  exp\_cursor **cursor** **for** **select** \* **from** example;  --声明基于表的记录变量  exp\_cursor\_record example%rowtype;  **begin**    **open** exp\_cursor;  --循环获取游标的数据到基于游标的记录变量  **loop**  **fetch** exp\_cursor **into** exp\_cursor\_record;  **if** **found** **then**  **raise** **notice** '%|%',exp\_cursor\_record.example\_id,exp\_cursor\_record.example\_name;  **else**  **exit**;  **end** **if**;  **end** **loop**;  **close** exp\_cursor;    **end$$**; |

1. 游标的FOR循环

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **BEGIN**    **FOR** i **IN** **(**  **SELECT** **\*** **FROM** example  **)**  **LOOP**  dbms\_output**.**put\_line**(**i**.**example\_id **||** '|' **||** i**.**example\_name**);**  **END** **LOOP;**    **END;**  **/** |
| GP | **do** **$$**  **declare**  i record;  **begin**  **for** i **in** (**select** \* **from** example) **loop**  **raise** **notice** '%|%',i.example\_id,i.example\_name;  **end** **loop**;  **end$$**; |

1. 动态游标

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **CREATE** **TABLE** example\_202011 **(**  example\_id NUMBER**,**  example\_name VARCHAR2**(**200**)**  **);**  **CREATE** **TABLE** example\_202010 **(**  example\_id NUMBER**,**  example\_name VARCHAR2**(**200**)**  **);**  **INSERT** **INTO** example\_202011 **VALUES** **(**20201101**,**'样例20201101'**);**  **INSERT** **INTO** example\_202011 **VALUES** **(**20201102**,**'样例20201102'**);**  **INSERT** **INTO** example\_202010 **VALUES** **(**20201001**,**'样例20201001'**);**  **INSERT** **INTO** example\_202010 **VALUES** **(**20201002**,**'样例20201002'**);**  **COMMIT;**  **DECLARE**  --定义动态游标类型  **TYPE** cursor\_type **IS** **REF** CURSOR**;**  --声明动态游标类型的变量  v\_cursor cursor\_type**;**  --当月月份变量 &thismonth表示替代变量  thismonth VARCHAR2**(**10**)** **:=** '&thismonth'**;**  --动态sql  v\_sql VARCHAR2**(**32767**);**  --example\_id变量用来接收游标中的example\_id  example\_id NUMBER**;**  --example\_name变量用来接收游标中的example\_name  example\_name VARCHAR2**(**200**);**  **BEGIN**    --动态从指定月份的备份表中查询数据  v\_sql **:=** q'[SELECT \* FROM example\_]' **||** thismonth**;**    --打开动态游标并关联动态sql  **OPEN** v\_cursor **FOR** v\_sql**;**    --循环输出游标中的所有数据  **LOOP**  **FETCH** v\_cursor **INTO** example\_id**,**example\_name**;**  **EXIT** **WHEN** v\_cursor**%**NOTFOUND **OR** v\_cursor**%**NOTFOUND **IS** **NULL;**    dbms\_output**.**put\_line**(**example\_id **||** '|' **||** example\_name**);**    **END** **LOOP;**    --关闭游标  **CLOSE** v\_cursor**;**  **END;**  **/** |
| GP | **create** **table** example\_202011 (  example\_id **numeric**,  example\_name **varchar**(200)  );  **create** **table** example\_202010 (  example\_id **numeric**,  example\_name **varchar**(200)  );  **insert** **into** example\_202011 **values** (20201101,'样例20201101');  **insert** **into** example\_202011 **values** (20201102,'样例20201102');  **insert** **into** example\_202010 **values** (20201001,'样例20201001');  **insert** **into** example\_202010 **values** (20201002,'样例20201002');  **do** **$$**  **declare**  --定义动态游标类型  v\_cursor refcursor;  thismonth **varchar**(10) := '${thismonth}';  --动态sql  v\_sql **varchar**(32767);  --example\_id变量用来接收游标中的example\_id  example\_id **numeric**;  --example\_name变量用来接收游标中的example\_name  example\_name **varchar**(200);  **BEGIN**    --动态从指定月份的备份表中查询数据  v\_sql := 'select \* from example\_' || thismonth;    --打开动态游标并关联动态sql  **open** v\_cursor **for** **execute** v\_sql;    --循环输出游标中的所有数据  **loop**  **fetch** v\_cursor **into** example\_id,example\_name;  **exit** **when** **not** **found**;    **raise** **notice** '%|%',example\_id,example\_name;    **end** **loop**;    --关闭游标  **close** v\_cursor;  **end$$**; |

## 4.13 异常处理

|  |  |
| --- | --- |
| 数据库 | 举例 |
| ORACLE | **BEGIN**  dbms\_output**.**put\_line**(**1**/**0**);**  **EXCEPTION** **WHEN** **OTHERS** **THEN**  dbms\_output**.**put\_line**(SQLERRM);**  **END;**  **/** |
| GP | **do** **$$**  **begin**  **raise** **notice** '%',1/0;  **exception** **when** **others** **then**  **raise** **notice** '%',**sqlerrm**;  **end$$**; |

## 4.14 存储过程和函数

1. ORACLE中存储过程和函数是分开的，但是在GP中，存储过程就是函数；

2. ORACLE中的函数和存储过程，在程序包中可以重载，GP的函数可以重载；

3. ORACLE的存储过程默认是以定义者权限执行，GP的函数默认是以调用者权限执行；

4. ORACLE存储过程中可以有多个事务，GP函数中，整个函数就是一个事务；

5. ORCLE中有自治事务，子事务的提交或回滚不会影响主事务的提交或回滚，GP中不存在自治事务，但部分可以通过begin exception end;块来实现自治事务；

|  |  |  |
| --- | --- | --- |
| 数据库 | 类型 | 举例 |
| ORACLE | 函数 | /\*  --最简单的函数  输入什么，就输出什么  \*/  **CREATE** **OR** **REPLACE** **FUNCTION** **print(**  **str** **IN** VARCHAR2  **)** **RETURN** VARCHAR2  **AS**  **BEGIN**  **RETURN** **str;**  **END;**  **/**  **SELECT** **print(**'123'**)** **FROM** dual**;** |
| ORACLE | 存储过程 | **CREATE** **OR** **REPLACE** **PROCEDURE** delete\_example  **AS**  **BEGIN**  **DELETE** **FROM** example**;**  **COMMIT;**    **EXCEPTION** **WHEN** **OTHERS** **THEN**  **ROLLBACK;**  **END;**  **/**  **CALL** delete\_example**();**  **EXEC** delete\_example**();**  **BEGIN**  delete\_example**();**  **END;**  **/** |
| GP | 函数 | /\*  --最简单的函数  输入什么，就输出什么  \*/  **create** **or** **replace** **function** print(**varchar**)  **returns** **varchar**  **language** plpgsql  **as** **$function$**  **declare**  **begin**  **return** **$1**;  **end**;  **$function$**  ;  **select** print('123');  **do** **$$**  **begin**  **perform** print('123');  **end$$**;  **create** **or** **replace** **function** delete\_example()  **returns** **void**  **language** plpgsql  **as** **$procedure$**  **begin**  **delete** **from** example;  **exception** **when** **others** **then**  **rollback**;  **end**;  **$procedure$**  ;  **select** delete\_example();  **do** **$$**  **begin**  **perform** delete\_example();  **end$$**; |

# 常用的数据字典表

## 5.1 ORACLE常用数据字典表

|  |
| --- |
| --显示用户下所有表和视图  **SELECT** **\*** **FROM** cat**;**  --显示用户下的表  **SELECT** **\*** **FROM** user\_tables**;**  --显示用户下的视图  **SELECT** **\*** **FROM** user\_views**;**  --显示所有表  **SELECT** **\*** **FROM** all\_tables**;**  --显示多有视图  **SELECT** **\*** **FROM** all\_views**;**  --显示用户下的约束  **SELECT** **\*** **FROM** user\_constraints**;**  **SELECT** **\*** **FROM** user\_cons\_columns**;**  --显示所有约束  **SELECT** **\*** **FROM** all\_constraints**;**  **SELECT** **\*** **FROM** all\_cons\_columns**;**  --显示用户下的索引  **SELECT** **\*** **FROM** user\_indexes**;**  **SELECT** **\*** **FROM** user\_ind\_columns**;**  --显示用户下的对象  **SELECT** **\*** **FROM** user\_objects**;**  --显示所有对象  **SELECT** **\*** **FROM** all\_objects**;**  --显示用户下的源代码  **SELECT** **\*** **FROM** user\_source**;**  --显示所有源代码  **SELECT** **\*** **FROM** all\_source**;**  --显示用户下的触发器  **SELECT** **\*** **FROM** user\_triggers**;**  --显示多有触发器  **SELECT** **\*** **FROM** all\_triggers**;**  --显示表中列的信息  **SELECT** **\*** **FROM** user\_tab\_columns**;**  **SELECT** **\*** **FROM** all\_tab\_columns**;** |

## 5.2 GP查询字段信息

|  |
| --- |
| **select**  b.nspname 模式名,  a.relname 表名,  c.attname 字段名,  d.typname 字段类型,  **case**  **when** **decode**(d.typtype, 'd', d.typtypmod::**varchar**, c.atttypmod::**varchar**)::**int4** = -1 **then** **null**  **when** **decode**(d.typtype, 'd', d.typbasetype::**varchar**, c.atttypid::**varchar**)::**oid** **in** (1042, 1043) **then** **decode**(d.typtype, 'd', d.typtypmod::**varchar**, c.atttypmod::**varchar**)::**int4** -4  **when** **decode**(d.typtype, 'd', d.typbasetype::**varchar**, c.atttypid::**varchar**)::**oid** **in** (1560, 1562) **then** **decode**(d.typtype, 'd', d.typtypmod::**varchar**, c.atttypmod::**varchar**)::**int4**  **else** **null**  **end** 字符类型的最大精度,  c.attnotnull 是否有非空约束,  e.adsrc 默认值,  f.description 注释  **from**  pg\_class a  **inner** **join** pg\_namespace b  **on**  (a.relnamespace = b.**oid**)  **inner** **join** pg\_attribute c  **on**  (a.**oid** = c.attrelid)  **inner** **join** pg\_type d  **on**  (c.atttypid = d.**oid**)  **left** **join** pg\_attrdef e  **on**  (a.**oid** = e.adrelid  **and** c.attnum = e.adnum)  **left** **join** pg\_description f  **on**  (a.**oid** = f.objoid  **and** c.attnum = f.objsubid)  **where**  c.attnum > 0  --and a.relname = 'tmp'  --and b.nspname = 'tool'  ; |

## 5.3 GP查询主键约束

|  |
| --- |
| **with** tmp\_a **as** (  **select**  b.nspname 模式名,  a.relname 表名,  c.attname 字段名,  c.attnum 字段号,  **case**  **when** g.indkey **is** **not** **null**  **and** g.indisunique = 'false' **then** 'true'  **else** 'false'  **end**::**boolean** 是否普通索引列,  g.indisprimary 是否主键索引列,  g.indisunique 是否唯一索引列  **from**  pg\_class a  **inner** **join** pg\_namespace b  **on**  (a.relnamespace = b.**oid**)  **inner** **join** pg\_attribute c  **on**  (a.**oid** = c.attrelid)  **left** **join** pg\_index g  **on**  (a.**oid** = g.indrelid  **and** c.attnum = **any**(g.indkey))  **where**  c.attnum > 0  --and a.relname = 'tmp'  --and b.nspname = 'tool'  )  **select**  模式名,  表名,  **string\_agg**(字段名, ',' **order** **by** 字段号) 主键约束  **from**  tmp\_a  **where**  是否主键索引列 = 'true'::**boolean**  **group** **by**  模式名,  表名; |

## 5.4 GP查询建索引语句

|  |
| --- |
| **select**  \*  **from**  pg\_catalog.pg\_indexes  --where  --tablename = 'tmp'  --and schemaname = 'tool'  ; |

## 5.5 GP查询分布键

|  |
| --- |
| **select**  a.**oid**,  c.nspname 模式名,  a.relname 表名,  **decode**(b.policytype, 'p', '分区策略', 'r', '复制策略') 分布策略,  pg\_get\_table\_distributedby(a.**oid**) 分布键,  a.reloptions 压缩类型  **from**  pg\_class a  **inner** **join** gp\_distribution\_policy b  **on**  (a.**oid** = b.localoid)  **inner** **join** pg\_namespace c  **on**  (a.relnamespace = c.**oid**)  --where  -- relname = 'tmp'  -- and nspname = 'tool'  ; |

## 5.6 GP查询表大小

|  |
| --- |
| **select** relname,pg\_size\_pretty(pg\_relation\_size(relid)) **as** **size** **from** pg\_stat\_user\_tables; |