Oracle与SQL Server应用差异对比分析

|  |  |  |  |
| --- | --- | --- | --- |
| 项目 | Oracle 10g | SQLServer 2008 | 备注 |
| 存储过程格式 | CREATE OR REPLACE PROCEDURE 存储过程名(参数列表) IS  Begin  ---存储过程内容  End 存储过程名; | CREATE PROCEDURE 存储过程名 (参数列表) AS  --存储过程内容 |  |
| 存储过程参数 | 1. 指定参数类型，但不指定长度 2. 在参数及类型间要加入出标识(IN、OUT)   Eg:  task\_remark IN VARCHAR2  prm\_code OUT NUMBER | 1. 指定参数类型，并要指定长度 2. 对于出参需要在类型后面加OUTPUT(或OUT)   Eg:  @task\_remark VARCHAR(10)  @prm\_code int output |  |
| 使用存储过程 | 1. 直接使用存储过程名 2. 参数在存储过程后面的括号包内列出   Eg:  p\_wfm\_getcolumns(v\_tablename,v\_tablecolumn); | 1. 调用存储过程名前面需要加 exec关键字 2. 参数在存储过程后面逐个列出 3. 出参后面要加关键字output或out   Eg:  EXEC P\_WFM\_GETCOLUMNS @tablename,@tablecolumn OUTPUT |  |
| 自定义函数格式 | CREATE OR REPLACE FUNCTION 函数名 (参数列表)  RETURN 返回值类型 IS  BEGIN  END; | CREATE FUNCTION 函数名 (参数列表)  RETURNs 返回值类型 AS  --函数内容 |  |
| 自定义函数内容 | 1. 指定参数类型，但不指定长度 2. 在参数及类型间要加入出标识(IN、OUT)   Eg:  task\_remark IN VARCHAR2  prm\_code OUT NUMBER   1. 支持对数据增、删、改操作 2. 支持动态SQL语句 | 1. 指定参数类型，并要指定长度 2. 不支持出参   Eg:  @task\_remark VARCHAR(10)   1. 不支持对数据增、删、改操作 2. 不支持动态SQL语句 |  |
| 使用自定义函数 | 1. 直接使用函数名   Eg:  V\_result :=f\_wfm\_isandbegin(flow\_id, step\_id) | 1. 在函数名前面加上dbo.   Eg:  Set @result=  dbo.f\_wfm\_isandbegin(@flow\_id, @step\_id) |  |
| 游标 | 1. 游标声明   DECLARE cursor 游标名 is select语句   1. 使用游标过程 打开(open)->提取(fetch)->关闭(close) 2. 支持快捷使用游标，直接使用for循环，数据库会自动打开、提取及关闭游标 | 1. 游标声明   DECLARE游标名cursor for select语句   1. 使用游标过程 打开(open)->提取(fetch)->关闭(close)->销毁( |  |
| 变量 | 1. 变量前不可加@符号 2. 存储过程中变量声明不需要declare   Eg:  Code varchar2(5);   1. 变量类型可按表中字段类型动态定义   Eg:  V\_id sysc01.id%type; | 1. 变量前需要加@符号 2. 变量声明需要使用Declare关键字   Eg:  DECLARE @code varchar(5);   1. 不支持按表字段类型动态定义 |  |
| 赋值 | 1. 变量直接赋值,变量 := 表达式;   Eg:  v\_result := ‘abcd’;   1. 通过SQL语句   Select 表达式 into 变量 from 表  Eg:  Select code,name into v\_code,v\_name from sysc01  Where id = 1000 | 1. 变量直接赋值，Set 变量=表达式;   Eg:  Set @result = ‘abcd’   1. 通过SQL语句赋值   Select 变量=表达式 from 表  Eg:  Select @code=code,@name=name from sysc01 where id = 1000 |  |
| 语句结束符 | SQL语句使用分号 ; 作为语句的结束 | SQL语句不需要加分号;作为结束符（加也可） |  |
| 大小写 | Oracle对字符区分大小写 | 默认对字符不区分大小写，也可修改数据库配置支持区分大小写 |  |
| 序列 | 1. 有序列sequence对象，无自动增长列 | 1. 无序列对象，表中有自动增长列 |  |
| Select 语法 | 1. 结果集可做为表使用，使用时可不加别名   Eg:  Select \* from (select \* from tab1)   1. 虚表 dual的使用，对于select计算某些与实体表无关的表达式时，要使用虚表dual   Eg:  Select round(1/3,2) from dual ; | 1. 结果集可做为表使用，使用时必须加别名   Eg:  Select \* from (select \* from tab1) a   1. 对于Select计算某些与实体表无关的表达式时，可使用不带from的select语句   Eg:  Select round(1/3,2) ; |  |
| Update语法 | 不可关联表更新  Eg:  For cur in (select a.id,b.value from a  inner join b on a.id = b.id ) loop  Update a set a.value = cur.value  Where a.id = cur.id;  End loop;  或  Update a set a.value = (select b.value  From b where b.id = a.id ) | 可关联表更新  Eg:  Update a set a.value = isnull(b.value,’’)  From a  inner join b on a.id = b.id  或  Update a set a.value = isnull(b.value,’’)  From b where a.id = b.id | 将a表中value按ID更新成b表中的value值 |
| Delete语法 | 不可关联表删除  Eg:  Delete a where exists (select b.id from b where a.id = b.id ) | 可关联表删除  Eg:  Delate a  From a  Inner join b on a.id = b.id | 删除a表中ID值在b表中存在的记录 |
| 动态SQL语句 | 1. 普通动态SQL语句   Begin  Execute immediate ‘update tab1 set column1=5’;  End;  用变量替换SQL语句  V\_sql := ‘update tab1 set column1=5’  Execute immediate v\_sql;   1. 带出参动态SQL语句   n\_count number(10);  v\_sql varchar2(1000);  v\_sql :=’ select count(\*) from tablename’;  execute immediate v\_sql  into n\_count;   1. 动态存储过程（带入、出参）   Eg:  v\_sql := 'begin p\_test (:v1,:v2,:v3); end;'  execute immediate v\_sql  using in v\_code,in v\_name,out v\_result ;  注：p\_test为存储过程名 | 1. 普通动态SQL语句   exec('update tab1 set column1 = 5')  或  exec sp\_executesql N'select \* from tableName' -- 字符串前一定要加N  用变量替换SQL语句  Declare @sql Nvarchar(1000)  Set @sql='select \* from tableName'  exec sp\_executesql @sql   1. 带出参动态SQL语句   declare @count int  declare @sqls nvarchar(4000)  set @sqls='select @a=count(\*) from tableName '  exec sp\_executesql @sqls,N'@a int output',@count output   1. 动态存储过程（带入、出参）   Eg:  DECLARE @result VARCHAR(50);  DECLARE @sql NVARCHAR(1000);  DECLARE @para NVARCHAR(200);  SET @sql = 'p\_test @code,@name,@result output'  SET @para = '@code varchar(10),@name varchar(10),@result varchar(50) output'  EXEC sp\_executesql @sql,@para,'001','张三',@result OUTPUT  注：p\_test 为存储过程名 |  |
| TOP用法 | 1、在Oracle中采用伪列rownum 获取结果集中排在前面的部分记录  Eg: 返回结果集中前10条记录  Select \* from sysc01 where rownum <=10  Rownum可使用<、<=符号，不可使用>、>=符号，如果使用=号只可=1  2、Rownum列还常用于形成结果集的顺序号，从而可获取一定序号范围的行  Eg: 获取按code排名第10到20行  Select \* from (Select a.\*,rownum as nrow from tab01 a order by code) where nrow between 10 and 20 | 1、SQLServer中采购top方式获取结果集排在前面的部分记录  Eg:返回结果集中前10条记录  Select top 10 \* from sysc01  2、sqlserver可通过ROW\_NUMBER()排名函数实现  Eg: 获取按code排名第10到20行  SELECT a.\* FROM (SELECT a.\*,ROW\_NUMBER() OVER( ORDER BY a.code ) AS nrow FROM tab01 a) a where nrow between 10 and 20 |  |
| IF…  Else  流控制 | IF 条件表达式 then  {语句块}  Else  {语句块}  End if ; | IF 条件表达式  {语句块}  Else  {语句块}  End  如果语句块中有多于1条SQL语句，则必须要使用begin … end 构造 |  |
| Case 用法 | 1. 表达式   Case 表达式 when 匹配表达式 then 结果表达式1 else 结果表达式2 end  Eg:  Select case name when ‘张三’ then 1 when ‘李四’ then 2 else 0 end  From person  或  Case when 条件表达式 then 结果表达式1 else 结果表达式2 end  Eg:  Select case when name=‘张三’ then 1 when name=‘李四’ then 2 else 0 end  From person   1. 流控制语句   流控制与表达式结构很相似，只是结果表达式换成语句块  Case 表达式 when 匹配表达式 then {语句块1} else {语句块2} end case;  Eg:  case v\_name  when ‘张三’ then  v\_number := 1;  when ‘李四’ then  v\_number:=2;  else  v\_number:=0  end case;  或  Case when 条件表达式 then {语句块1} else {语句块2} end case;  Eg:  case  when v\_name = ‘张三’ then  v\_number := 1;  when v\_name = ‘李四’ then  v\_number:=2;  else  v\_number:=0  end case; | 1. case表达式   同Oracle  2、不支持流控制 |  |
| 临时表 | 1. 临时表结构需要象实体表一样事先定义后再在SQL脚本中使用，临时表名与实体表名规则一样   Eg:  create temporary table WFMW02  (  paraid NUMBER(9),  vvalue VARCHAR2(50),  flowid NUMBER(9),  id NUMBER(9)  )  on commit delete rows; | 1. 临时表是在执行数据库脚本会话过程中创建并使用，临时表名前面带有#号(或##全局临时表)   Eg:  create table #WFMW02  (  paraid int,  vvalue VARCHAR(50),  flowid int,  id int  )  临时表也可在使用过程中自动创建  Select paraid,vvalue,flowid,id  into #wfmw02  from wfmb01c |  |
| 常用函数 | 1. nvl(表达式,表达式) 2. sysdate 3. instr(字符表达式1,字符表达式2 [,pos][,nth])   从“表达式1”的“pos”位置搜索第“nth”个“表达式2”的位置（若未找到则返回0）   1. To\_char(),to\_date(),to\_number() | 1. isnull(表达式,表达式) 2. getdate() 3. CHARINDEX(字符表达式1,字符表达式2 [,pos])   从“表达式2”的“pos”位置搜索“表达式1”并返回“表达式1”的起始位置（若未找到则返回0）   1. convert(),cast() |  |
| 运算符 | 字符相加 ||  Eg: ‘A’||’B’ = ‘AB’ | 字符相加 +  Eg: ‘A’+’B’ = ‘AB’ |  |
| null | 对于字符型数据，空串等同于NULL | 对于字符型数据，空串 不同于null |  |
|  |  |  |  |