GBASE

Oracle 到 GBase 8s 迁移指南
存储过程函数触发器

www.gbase.cn

目 录

1. 概述

将数据库从 Oracle 迁移到 GBase 8s 主要完成三个任务： 数据库架构迁移(Schema/DDL)、数据迁移(Data)和应用迁移(Application)。数据库对象迁移的很大一部分工作量是存储过程、函数和触发器的迁移，这部分的迁移迄今为止工具（如 MTK）迁移完成的效果很不令人满意，主要还是靠人工手动将 Oracle 的语法及业务逻辑翻译成 GBase 8s 的语法及完全不改变业务逻辑本身，并且要尽量适应原应用程序的调用格式，尽量少的修改应用源码，这样既能保证少出错误，也同时能保证应用迁移的工作量尽可能的少。本文将以实际迁移中针对 Oracle 的具体语法常用的替换方式（但不一定是唯一的）来详细介绍一下过程对象的迁移过程，希望对正在或者准备进行 Oracle 到 GBase 8s 迁移项目的工作人员有一定的指导和启发。

本文基于 GBase 8s v8.7 2.0.1a2_2 版本来介绍。

2. 概念

存储过程是一个用户定义的函数，由存储过程语句（SPL）组成，以可执行代码的形式存储在数据库中，因此执行速度要比单独执行单个 SQL 快，是数据库的一种对象，用存储过程语言编写。存储过程常用于执行一个公用的应用逻辑，不用每个编程人员都去写这些重复的代码，因此可以加快开发的速度，由于代码变更时，只需要修改这个过程，因此代码的维护量也降低了。

函数的概念基本和存储过程类似，也是封装了一段业务逻辑的处理过程，要求必须有返回值。

触发器用于监听表或视图的特定事件，当发生变化时，去执行一些应用的逻辑。这些事件包括插入、删除、更新和查询四种。触发器是数据库的一种对象，常用于执行一些用户的应用逻辑、审计、级联删除、判断插入或者更新的数据是否符合要求等。

3. 存储过程迁移

3.1 定义格式

Oracle	GBase 8s
create or replace procedure proc_name is ... end proc_name;	drop procedure if exists proc_name(); create procedure proc_name() ... end procedure;

3.2 入参出参

Oracle	GBase 8s
create or replace procedure proc_name(v_1 in varchar2, v_2 in varchar2, v_3 out varchar2) is ... end proc_name;	drop procedure if exists proc_name(); create procedure proc_name(in v_1 varchar(255), in v_2 varchar(255), out v_3 varchar(255)) ... end procedure;

in/out/inout 关键字在位置上有所区别，并且对于变长字段类型，Oracle 不指定长度默认为最大长度，而 GBase 8s 默认长度为 1，所以转换时需要指定最大支持字符长度。

3.3 变量定义和赋值

Oracle	GBase 8s
v_var1 varchar2(255); v_var1 := 'Oracle' ;	define v_var1 varchar(255); let v_var1 = 'GBase 8s' ;
v_var2 tb1.name%type; select name into v_var2 from tb1;	define v_var2 like tb1.name; select name into v_var2 from tb1;
v_var3 tb1%rowtype; select * into v_var3 from tb1;	define v_var3 rt_tb1; select * into v_var3 from tb1;

%TYPE 和%ROWTYPE 在 Oracle 中称为宏定义，在 GBase 8s 中可以使用 like 表名.列名和 like 表名.*替换，但是 like 表名.*不能应用于存储过程。

在存储过程中，可以预先自定义好 row type 类型，然后再在存储过程中使用该自定义 row type 类型。如上表中的 rt_tb1 类型可以替换 Oracle 中的%ROWTYPE 宏。上例中

的表及 row type 定义格式如下：

```
create table tb1(
   id int,
   name varchar(20),
   primary key(id)
);
create row type rt_tb1(
   id int,
   name varchar(20)
);
```

3.4 SELECT INTO 语句

GBase 8s 的 SELECT INTO 语句和 Oracle 的语法格式是一致的，都是 SELECT col1,col2 INTO var1,var2 FROM tb1 WHERE 1=1; (var1 和 var2 为变量名)。

值得注意的是，Oracle 要求表中必须存在满足条件的一条记录，否则会抛出 NO_DATA_FOUND 异常，但 GBase 8s 没有这样的限制。尽管如此，同样可以使用 DBINFO 内置函数取得查询到的记录行数，如下模拟 Oracle 的 NO_DATA_FOUND 异常：

Oracle	GBase 8s

exception
when no_data_found then
rollback;
	if dbinfo('sqlca.sqlerrd2') = 0 then
	  raise exception 100;
	end if;

DBINFO('sqlca.sqlerrd2') = 0 表示上一条语句操作的记录数。GBase 8s 存储过程中的异常处理参考 3.8 节描述。

3.5 IF 判断

Oracle	GBase 8s

if 条件1 then
  语句1 ;
elsif 条件2 then
  语句2 ;
else
  语句3 ;
end if ;
	if 条件1 then
	  语句1 ;
	elif 条件2 then
	  语句2 ;
	else
	  语句3 ;
	end if ;

唯一的语法区别就是 ELSE IF 的写法，Oracle 为 ELSIF，GBase 8s 为 ELIF。

3.6 WHILE 循环

Oracle	GBase 8s

while 条件成立 loop
begin
执行语句;
end;
end loop;
	while 条件成立
	begin
	执行语句;
	end;
	end while;

区别在于：Oracle 需要 LOOP 循环，而 GBase 8s 使用 WHILE 自循环

3.7 FOR 循环

Oracle	GBase 8s
for v_index in 1..10 loop 语句; end loop;	for v_index = 1 to 10 语句; end for;
for v_index in reverse 1..10 loop 语句; end loop;	for v_index = 10 to 1 语句; end for;

区别在于：Oracle 需要 LOOP 循环，而 GBase 8s 使用 FOR 自循环

3.8 异常处理

Oracle	GBase 8s
exception when others then rollback;	on exception rollback; end exception;

err1 exception;
if v_flag = 0 then
raise err1;
exception
when err1 then
rollback;
	on exception in (10000)
	rollback;
	return 0;
	end exception;
	if v_flag = 0 then
	raise exception 10000;
	end if;

值得注意的是，GBase 8s 的 exception 异常处理语句必须写在变量定义之后，事务语句开始之前。无论是 Oracle 还是 GBase 8s 在触发 exception 之后，缺省情况都退出过程。

3.9 动态 SQL 与绑定变量

Oracle	GBase 8s
v_sql:= 'select id,name from tb1 where id=1'; execute immediate v_sql into v_id,v_name;	let v_sql = 'select id,name into '||v_id||','||v_name||' from tb1 where id =1'; execute immediate v_sql;
v_sql:= 'delete from tb1 where id = :id'; execute immediate v_sql using v_id;	let v_sql = 'delete from tb1 where id = :id'; let v_sql = replace(v_sql,':id',v_id); execute immediate v_sql;

GBase 8s 不支持动态 SQL 插入和绑定变量执行 转换的原则就是将变量代入动态 SQL 中既定的位置组成新的动态 SQL 字符串执行。

3.10 游标

Oracle	GBase 8s

type name_cursor is ref cursor;
v_cursor name_cursor;
v_sql varchar2(255);
v_id integer;
v_name varchar2(20);
v_sql:= 'select id,name from tb1';
open v_cursor for v_sql;
loop
fetch v_cursor into v_id,v_name;
EXIT WHEN v_cursor%NOTFOUND;
…
end loop;
close v_cursor;
	DEFINE v_cursor SYS_REFCURSOR;
	DEFINE v_sql VARCHAR(255);
	DEFINE v_id INT;
	DEFINE v_name VARCHAR(20);
	LET v_sql = 'select id,name from tb1';
	OPEN v_cursor FOR v_sql;
	LOOP
	FETCH v_cursor INTO v_id,v_name;
	EXIT WHEN SQLCODE = 100;
	…
	END LOOP;
	CLOSE v_cursor;

SQLCODE = 100 表示上一个 SELECT 或者 FETCH 语句无记录返回（即 NOT FOUND 或者 END OF DATA）。GBase 8s 的 sys_refcursor 支持 Oracle 的游标方式。

4. 函数迁移

函数的迁移基本和存储过程的语法格式一致，完全可以参照第 3 章存储过程的迁移去迁移函数，GBase 8s 函数必须指定返回值。函数的定义格式为：

```
drop function if exists func_name();
create function func_name(v_id int) returning varchar(20)
```

```
   define v_name varchar(20);
...
   return v_name;
end function;
```

5. 触发器迁移

触发器的迁移相比存储过程的迁移相对比较简单，不过有两点值得注意。从实际迁移项目上大体来说 Oracle 的触发器大致可以分为两大类，一类是实现字段自增长的(Oracle 字段类型没有设置 serial 自增长类型)，其他可以归类为增删改业务事件触发器。Oracle 一个触发器可以触发多个事件，但 GBase 8s 不支持，只能将 Oracle 增删改触发器迁移为三个触发器。下面分别对这两类触发器用实际的例子描述一下转换过程。

5.1 字段自增触发器

Oracle	GBase 8s

CREATE TRIGGER tri_tb1_id_serial before insert on tb1 for each row
begin
select seq_tb1_id.nextval into NEW.id from dual;
end;
	alter table tb1 modify(id serial);
	create procedure proc_tri_tb1_id_serial()
	referencing new as new for tb1
	select seq_tb1_id.nextval into new.id from dual;
	end procedure;
	
	create trigger tri_tb1_id_serial insert on tb1 for each row(execute procedure proc_tri_tb1_id_serial() with trigger references);

其中表和序列的定义分别为：

```
create table tb1(
   id int,
   name varchar(20),
   primary key(id)
);
CREATE SEQUENCE SEQ_TB1_ID
   INCREMENT BY 1
   START WITH 1
   MAXVALUE 999999999
   MINVALUE 1
```

```
NOCYCLE
CACHE 20
NOORDER;
```

针对Oracle 字段自增触发器，GBase 8s有两种迁移方法，一种是直接将业务表TB1 中的id 字段改为SERIAL/BIGSERIAL类型，另一种是保留原业务表字段类型。GBase 8s 也使用触发存储过程实现字段自增长。两种迁移方法可以根据实际业务需要按需选择。

5.2 增删改触发器

Oracle	GBase 8s

CREATE TRIGGER TRI_TB1 AFTER INSERT OR UPDATE OR DELETE ON TB1
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH ROW
declare
v_flag integer;
v_id integer;
begin
if inserting then
v_flag := 1
v_id := NEW.id;
elsif updating then
v_flag := 2;
v_id := NEW.id;
elsif deleting then
v_flag := 3;
v_id := OLD.id;
end if;
insert into tb1_idu
(seq_id, idu_type,id,idu_date)
values
(seq_tb1_idu.nextval,v_flag,v_id,sysdate);
end;
	CREATE PROCEDURE tri_tb1_proc()
	REFERENCING OLD AS OLD NEW AS NEW
	FOR tb1
	define v_flag int;
	define v_id int;
	if inserting then
	let v_flag = 1;
	let v_id = NEW._id;
	elif updating then
	let v_flag = 2;
	let v_id = NEW.id;
	elif deleting then
	let v_flag = 3;
	let v_id = OLD.id;
	end if;
	insert into tb1_idu
	(seq_id, idu_type,id,idu_date)
	values
	(seq_tb1_idu.nextval,v_flag,v_id,sysdate);
	END PROCEDURE ;
	
	CREATE TRIGGER tri_tb1_1 INSERT ON tb1
	FOR EACH ROW (
	EXECUTE PROCEDURE tri_tb1_proc()
	WITH TRIGGER REFERENCES);
	
	CREATE TRIGGER tri_tb1_2 UPDATE ON tb1
	FOR EACH ROW (
	EXECUTE PROCEDURE tri_tb1_proc() WITH
	TRIGGER REFERENCES);

	CREATE TRIGGER tri_tb1_3 DELETE ON tb1 FOR EACH ROW (EXECUTE PROCEDURE TRI_TB1_proc() WITH TRIGGER REFERENCES);

其中表 TB1 和 TB1_IDU 以及序列 seq_tb1_idu 定义为：

```
create table tb1(
   id int,
   name varchar(20),
   primary key(id)
);
create table tb1_idu(
   seq_id int,
   idu_type int,
   id int,
   idu_date datetime year to second,
   primary key(seq_id)
);
CREATE SEQUENCE SEQ_TB1_idu
   INCREMENT BY 1
   START WITH 1
   MAXVALUE 999999999
   MINVALUE 1
   NOCYCLE
   CACHE 20
   NOORDER;
```

针对 Oracle 的增删改触发器，GBase 8s 对应为三个触发器和一个存储过程组成的触发存储过程，存储过程封装了业务处理过程。

GBASE

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微博二维码　微信二维码

技术支持热线：400-817-9696