Oracle与MySQL的SQL语句区别

> 参考网址
>
> > `https://www.cnblogs.com/yanyunpiaomaio/p/10821437.html`

数据库

> **MySQL可以创建数据库**，而oracle没有这个操作，oracle只能创建实例；
>
> > **MySQL数据库操作**：database
> >
> > > **创建数据库**
> > >
> > > > 格式:create database 数据库名;
> > > >
> > > > create database 数据库名 character set 字符集;
> > > >
> > > > > 例如：
> > > > >
> > > > > > `CREATE DATABASE j0815_1; CREATE DATABASE j0815_2 CHARACTER SET utf8;`
> > >
> > > **查看数据库**
> > >
> > > > SHOW DATABASES;
> > >
> > > **查看定义的数据库**
> > >
> > > > SHOW CREATE DATABASE j0815_1;
> > >
> > > **删除数据库**
> > >
> > > > DROP DATABASE j0815_1;
> > >
> > > **切换数据库**
> > >
> > > > USE j0815_1;
> > >
> > > **查看正在使用的数据库**
> > >
> > > > SELECT database();

表

创建表（异）

> **MySQL**
>
> > mysql没有number、varchar2()类型； mysql可以声明自增长：auto_increment； mysql有double类型；
> >
> > 案例

```
create TABLE emp( eno INT PRIMARY KEY AUTO_INCREMENT, ename VARCHAR(20) NOT NULL UNIQUE, job VARCHAR(10) DEFAULT '员工', mgr INT(10), hiredate DATE, comm DOUBLE );
```

Oracle

> oracle没有double类型、有int类型但多数会用number来代替int； oracle不可以声明自增长：auto_increment，主键自带自增长； oracle小数只有float类型；

> 案例

```
create table emp( empno number(10) primary key ,--主键 ename varchar2(20) not null unique,--不能为空,唯一 job varchar2(10) default '匿名',--默认值，用单引号 mgr number(10), hiredate date,--默认格式DD-MM-YY sal number(10,2), comm float, deptno number(10) );
```

删除表（异）

Mysql

> DROP TABLE IF EXISTS 表名; 或drop table if exists 表名；

> 案例

> DROP TABLE IF EXISTS emp;

> drop table if exists emp；

Oracle

> drop table 表名；

> 注：Oracle没有if exists关键字，也没用类似if exists的SQL语法

> 案例

> drop table emp

列

添加列（异）

MySQL

> A、alter table 表名 add column 字段 数据类型;

> B、alter table 表名 add column 字段1 数据类型, add column 字段2 数据类型;

> 注：其中关键字column可有可无。

> 案例

```
ALTER TABLE emp ADD COLUMN marriage VARCHAR(2); ALTER TABLE emp ADD marriagedate DATE; ALTER TABLE emp ADD COLUMN lovename VARCHAR(50), ADD COLUMN loveage INT;
```

Oracle

A、alter table 表名 add 字段 数据类型;

B、alter table 表名 add (字段 数据类型);

C、alter table 表名 add (字段1 数据类型, 字段2 数据类型);

> 注：对于A，只有添加单列的时候才可使用，对于添加多列时需要使用C，不能像MySQL那样重复使用add column关键字。

案例

```
alter table emp add marriage varchar2(2); alter table emp add (marriagedate DATE);
alter table emp add (lovename varchar2(50), loveage INT);
```

```
desc emp;
```

删除列（异）

MySQL

A、alter table 表名 drop column 字段;

B、alter table 表名 drop column 字段, drop column 字段;

> 注：其中关键字column可有可无

案例

```
ALTER TABLE emp DROP COLUMN marriage; ALTER TABLE emp DROP COLUMN marriagedate,
DROP COLUMN lovename,DROP COLUMN loveage;
```

Oracle

A、alter table 表名 drop column 字段;

B、alter table 表名 drop (字段);

C、alter table 表名 drop (字段1,字段2);

> 注：对于A，只有删除单列的时候才可使用，对于删除多列时需要使用C，不能像MySQL那样重复使用drop column关键字

案例

```
alter table emp drop column marriage; alter table emp drop (marriagedate); alter
table emp drop (lovename,loveage);
```

修改列名（异）

MySQL

alter table 表名 change column 原来字段 新的字段 字段类型(必须);

案例

```
ALTER TABLE emp CHANGE COLUMN mgr manager VARCHAR(20);
```

Oracle:

> > alter table 表名 rename column 原来字段 to 新的字段; 注：不能有字段类型
> > 修改字段类型：alter table 表名 modify(字段 数据类型 约束条件);
>
> 案例
>
> ```
> alter table emp rename column mgr to manager;
> ```

修改列类型（说明）

> **MySQL**
>
> 无论列是否有数据都可以修改列类型
>
> alter table 表名 modify column 字段名 类型
>
> 但是当有数据时，直接修改列类型都可能对数据造成丢失等，所以一般需要结合具体的业务来对列数据做处理后，再修改列类型类型。所以修改列的类型并非使用SQL语句进行一步到位的修改，而是通过以下流程：
>
> A、添加临时列
>
> B、将需要更改的列的值经过类型转换的验证后，赋值给临时列
>
> C、删除原有列
>
> D、将临时列的列名修改为原有列列名
>
> 案例
>
> ```
> ALTER TABLE emp MODIFY COLUMN manager VARCHAR(20);
> ```
>
> **Oracle**
>
> 在列有数据的时候，无法修改列类型；没有数据时可以。 alter table 表名 modify(字段 数据类型 约束条件);
>
> 案例
>
> ```
> alter table emp modify(manager varchar2(20));
> ```

索引

> 在整个数据库内，MySQL的索引可以同名，也就是说MySQL的索引是表级别的；但是Oracle索引不可以同名，也就是说Oracle的索引是数据库级别的

创建索引（同）

> ```
> create index indexName on tableName (columnName);
> ```

删除索引（异）

> **MySQL**
>
> ```
> alter table tableName drop index indexName
> ```
>
> **Oracle**
>
> ```
> drop index indexName
> ```

查询表的索引（异）

> **MySQL**
>
> > `show index from tableName`
>
> **Oracle**
>
> > `select index_name, table_name, column_name from user_ind_columns where table_name='tableName '`

空字符串问题

> **MySQL**
>
> > 区分null和""
>
> **Oracle**
>
> > 空字符串''就是null
> >
> > `select * from table1 where user_name < ''`
> >
> > `select * from table1 where user_name > ''`
> >
> > > 查询列user_name不为空
> > >
> > > > **Oracle**
> > > >
> > > > > 查不出任何结果
> > > >
> > > > **MySQL**
> > > >
> > > > > 已经同时存在Null和''时，所有判断是否为null或者''的地方改为判断列的长度是否为0

MySQL与Oracle SQL语言差异比较

数据类型

编号	ORACLE	MYSQL	注释
1	NUMBER	int / DECIMAL	DECIMAL就是NUMBER(10,2)这样的结构，INT就是NUMBER(10)，表示整型； MYSQL有很多类int型，tinyint mediumint bigint等，不同的int宽度不一样
2	Varchar2（n）	varchar(n)	
3	Date	DATATIME	日期字段的处理 **MYSQL**日期字段分**DATE**和**TIME**两种， **ORACLE**日期字段只有**DATE**，包含**年月日时分秒**信息，用当前数据库的系统时间为 SYSDATE, 精确到秒， 或者用**字符串转换成日期型函数TO_DATE**(‘2001-08-01’,’YYYY-MM-DD’)年-月-日 24小时:分钟:秒的格式YYYY-MM-DD HH24:MI:SS TO_DATE()还有很多种日期格式, 可以参看**ORACLE DOC**. **日期型字段转换成字符串函数TO_CHAR**(‘2001-08-01’,’YYYY-MM-DD HH24:MI:SS’) 日期字段的数学运算公式有很大的不同。 **MYSQL**找到离当前时间7天用DATE_FIELD_NAME > SUBDATE（NOW（），INTERVAL 7 DAY) **ORACL**E找到离当前时间7天用 DATE_FIELD_NAME > SYSDATE - 7; **MYSQL**中插入当前时间的几个函数是：NOW()函数以`'YYYY-MM-DD HH:MM:SS'返回当前的日期时间，可以直接存到DATETIME字段中。CURDATE()以’YYYY-MM-DD’的格式返回今天的日期，可以直接存到DATE字段中。 CURTIME()以’HH:MM:SS’的格式返回当前的时间，可以直接存到TIME字段中。例：insert into tablename (fieldname) values (now()) 而**oracle**中当前时间是sysdate
4	INTEGER	int / INTEGER	Mysql中INTEGER等价于int
5	EXCEPTION	SQLEXCEPTION	详见<<2009001-eService-O2MG.doc>>中2.5 Mysql异常处理
6	CONSTANT VARCHAR2(1)	mysql中没有CONSTANT关键字	从ORACLE迁移到MYSQL,所有CONSTANT常量只能定义成变量
7	TYPE g_grp_cur IS REF CURSOR;	光标 : mysql中有替代方案	详见<<2009001-eService-O2MG.doc>>中2.2 光标处理

编号	ORACLE	MYSQL	注释
8	TYPE unpacklist_type IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;	数组: mysql中借助临时表处理或者直接写逻辑到相应的代码中， 直接对集合中每个值进行相应的处理	详见<<2009001-eService-O2MG.doc>>中2.4 数组处理
9	自动增长的序列	自动增长的数据类型	MYSQL有自动增长的数据类型，插入记录时不用操作此字段，会自动获得数据值。 ORACLE没有自动增长的数据类型，需要建立一个自动增长的序列号，插入记录时要把序列号的下一个值赋于此字段。
10	NULL	NULL	空字符的处理 MYSQL的非空字段也有空的内容， ORACLE里定义了非空字段就不容许有空的内容。 按MYSQL的NOT NULL来定义ORACLE表结构, 导数据的时候会产生错误。因此导数据时要对空字符进行判断，如果为NULL或空字符，需要把它改成一个空格的字符串。

基本语法

编号	类别	ORACLE	MYSQL	注释
1	变量的声明方式不同	li_index NUMBER := 0	DECLARE li_index INTEGER DEFAULT 0	1. mysql 使用DECLARE定义局部变量. 定义变量语法为: DECLARE var_name[,...] type [DEFAULT value] 要给变量提供一个默认值，需要包含一个DEFAULT子句。值可以被指定为一个表达式，不需要为一个常数。如果没有DEFAULT子句，初始值为NULL。
2	变量的赋值方式不同	lv_inputstr := iv_inputstr	SET lv_inputstr = iv_inputstr	1. oracle变量赋值使用:= mysql使用赋值使用set关键字. 将一个值赋给一个变量时使用"=".
3	跳出（退出）语句不同	EXIT;	LEAVE procedure name;	1. oracle: 如果exit语句在循环中就退出当前循环.如果exit语句不再循环中,就退出当前过程或方法. Mysql: 如果leave语句后面跟的是存储过程名,则退出当前存储过程. 如果leave语句后面跟的是lable名. 则退出当前lable.
while 条件 loop exit; end loop;	label_name:while 条件 do leave label_name; end while label_name;			
4	定义游标	TYPE g_grp_cur IS REF CURSOR;	DECLARE cursor_name CURSOR FOR SELECT_statement;	oracle可以先定义游标,然后给游标赋值. mysql定义游标时就需要给游标赋值. Mysql定义游标出自Mysql 5.1 参考手册20.2.11.1.声明光标.
5	定义数组	TYPE unpacklist_type IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;	可以使用临时表代替oracle数组, 也可以循环拆分字符来替代oracle数组.	目前可以使用临时表来代替oracle数组. 详见<<2009002-OTMPPS-Difficult Questions-0001.doc>>中2.4 Mysql数组处理部分
6	注释方式不同	"-- message" 或 "/** .... /" *或"/* .... */"	"-- message" 或 "/* .... */" 或 "#"	mysql注释来自 MySQL 5.1参考手册 9.5. 注释语法, 建议同oracle一样, 单行用--, 多行/* */
7	自带日期时间函数格式不同	Oracle时间格式：yyyy-MM-dd hh:mi:ss	Mysql时间格式：%Y-%m-%d %H:%i:%s	1. MYSQL日期字段分DATE和TIME两种. ORACLE日期字段只有DATE，包含年月日时分秒信息. 2. mysql中取当前系统时间为now()函数,精确到秒. oracle中取当前数据库的系统时间为SYSDATE, 精确到秒.
8	日期加减	当前时间加N天: sysdate+N 当前时间减N天: sysdate-N	日期相加: date_add(now(), INTERVAL 180 DAY) 日期相减: date_sub('1998-01-01 00:00:00', interval '1 1:1:1' day_second)	

编号	类别	ORACLE	MYSQL	注释
9	字符串连接符不同	`result := v_int1||v_int2;`	`set result =concat(v_int1,v_int2);`	1. oracle使用||连接字符串,也可以使用concat函数. 但Oracle的concat函数只能连接两个字符串. Mysql使用concat方法连接字符串. MySQL的concat函数可以连接一个或者多个字符串,如 mysql> select concat('10'); 结果为: 10. mysql> select concat('11','22','33','aa'); 结果为: 112233aa 2. "||"在Mysql是与运算
10	定义游标不同	`CURSOR l_bk_cur IS SELECT B.BK_HDR_INT_KEY, B.BK_NUM FROM ES_SR_DTL_VRB A, ES_BK_HDR B WHERE A.BK_HDR_INT_KEY = B.BK_HDR_INT_KEY AND b.BK_STATUS != ES_BK_PKG.g_status_can AND A.SR_HDR_INT_KEY = ii_sr_hdr_int_key;`	`DECLARE l_bk_cur CURSOR FOR SELECT B.BK_HDR_INT_KEY, B.BK_NUM FROM ES_SR_DTL_VRB A, ES_BK_HDR B WHERE A.BK_HDR_INT_KEY = B.BK_HDR_INT_KEY AND b.BK_STATUS != ES_BK_PKG.g_status_can AND A.SR_HDR_INT_KEY = ii_sr_hdr_int_key;`	详见<<2009002-OTMPPS-Difficult Questions-0001.doc>>中2.2 Mysql游标处理部分
11	事务回滚	ROLLBACK;	ROLLBACK;	oracle和mysql中使用方法相同
12	GOTO语句	GOTO check_date;	GOTO check_date;	oracle和mysql中使用方法相同

函数

编号	类别	ORACLE	MYSQL	注释
1	数字函数	round(1.23456,4)	round(1.23456,4)	一样： ORACLE：select round(1.23456,4) value from dual MYSQL：select round(1.23456,4) value
2	abs(-1)	abs(-1)	功能: 将当前数据取绝对值 用法: oracle和mysql用法一样 mysql: select abs(-1) value oracle: select abs(-1) value from dual	
3	ceil(-1.001))	ceiling(-1.001)	功能: 返回不小于 X 的最小整数 用法: mysqls: select ceiling(-1.001) value oracle: select ceil(-1.001) value from dual	
4	floor(-1.001)	floor(-1.001)	功能: 返回不大于 X 的最大整数值 用法: mysql: select floor(-1.001) value oracle: select floor(-1.001) value from dual	
5	Max(expr)/Min(expr)	Max(expr)/Min(expr)	功能:返回 expr 的最小或最大值。MIN() 和 MAX() 可以接受一个字符串参数；在这 种情况下，它们将返回最小或最大的字符串传下。 用法: ROACLE: select max(user_int_key) from sd_usr; MYSQL: select max(user_int_key) from sd_usr;	
6	字符串函数	ascii(str)	ascii(str)	功能:返回字符串 str 最左边的那个字符的 ASCII 码值。如果 str 是一个空字符串， 那么返回值为 0。如果 str 是一个 NULL，返回值也是 NULL. 用法: mysql:select ascii('a') value oracle:select ascii('a') value from dual
7	CHAR(N,...)	CHAR(N,...)	功能:CHAR() 以整数类型解释参数，返回这个整数所代表的 ASCII 码值给出的字符 组成的字符串。NULL 值将被忽略. 用法: mysql:select char(97) value oracle:select chr(97) value from dual	
8	REPLACE(str,from_str,to_str)	REPLACE(str,from_str,to_str)	功能: 在字符串 str 中所有出现的字符串 from_str 均被 to_str 替换，然后返回这个字符串. 用法: mysql: SELECT REPLACE('abcdef', 'bcd', 'ijklmn') value oracle: SELECT Replace('abcdef', 'bcd', 'ijklmn') value from dual	
9	INSTR('sdsq','s',2)	INSTR('sdsq','s')	参数个数不同 ORACLE: select INSTR('sdsq','s',2) value from dual（要求从位置2开始） MYSQL: select INSTR('sdsq','s') value（从默认的位置1开始）	
10	SUBSTR('abcd',2,2)	substring('abcd',2,2)	函数名称不同： ORACLE: select substr('abcd',2,2) value from dual MYSQL: select substring('abcd',2,2) value	
11	instr('abcdefg','ab')	locate('ab','abcdefg')	函数名称不同： instr -> locate（注意：locate的子串和总串的位置要互换） ORACLE: SELECT instr('abcdefg', 'ab') VALUE FROM DUAL MYSQL: SELECT locate('ab', 'abcdefg') VALUE	
12	length（str）	char_length()	函数名称不同： ORACEL: SELECT length('AAAASDF') VALUE FROM DUAL MYSQL: SELECT char_length('AAAASDF') VALUE	
13	REPLACE('abcdef', 'bcd', 'ijklmn')	REPLACE('abcdef', 'bcd', 'ijklmn')	一样： ORACLE: SELECT REPLACE('abcdef', 'bcd', 'ijklmn') value from dual MYSQL: SELECT REPLACE('abcdef', 'bcd', 'ijklmn') value	
14	LPAD('abcd',14, '0')	LPAD('abcd',14, '0')	一样： ORACLE: select LPAD('abcd',14, '0') value from dual MYSQL: select LPAD('abcd',14, '0') value from dual	
15	UPPER(iv_user_id)	UPPER(iv_user_id)	一样： ORACLE: select UPPER(user_id) from sd_usr; MYSQL: select UPPER(user_id) from sd_usr;	
16	LOWER(iv_user_id)	LOWER(iv_user_id)	一样： ORACLE: select LOWER(user_id) from sd_usr; MYSQL: select LOWER(user_id) from sd_usr;	
17	控制流函数	nvl(u.email_address, 10)	IFNULL(u.email_address, 10) 或 ISNULL(u.email_address)	函数名称不同（根据不同的作用进行选择）： ORACLE: select u.email_address, nvl(u.email_address, 10) value from sd_usr u (如果u.email_address=NULl,就在DB中用10替换其值) MYSQL: select u.email_address, IFNULL(u.email_address, 10) value from sd_usr u(如果u.email_address=NULl,显示结果中是10，而不是在DB中用10替换其值) select u.email_address, ISNULL(u.email_address) value from sd_usr u（如果u.email_address是NULL, 就显示1,否则就显示0）

编号	类别	ORACLE	MYSQL	注释
18	DECODE(iv_sr_status,g_sr_status_com, ld_sys_date, NULL)	无，请用IF或CASE语句代替. IF语句格式:(expr1,expr2,expr3)	说明: 1. decode(条件,值1,翻译值1,值2,翻译值2,...值n,翻译值n,缺省值) 该函数的含义如下： IF 条件=值1 THEN RETURN(翻译值1) ELSIF 条件=值2 THEN RETURN(翻译值2) ..... ELSIF 条件=值n THEN RETURN(翻译值n) ELSE RETURN(缺省值) END IF 2. mysql If语法说明 功能: 如果expr1 是TRUE (expr1 <> 0 and expr1 <> NULL)，则IF()的返回值为expr2; 否则返回值则为 expr3。IF() 的返回值为数字值或字符串值，具体情况视其所在 语境而定。 用法: mysql: SELECT IF(1>2,2,3);	
19	类型转换函数	TO_CHAR(SQLCODE)	date_format/ time_format	函数名称不同 SQL> select to_char(sysdate,'yyyy-mm-dd') from dual; SQL> select to_char(sysdate,'hh24-mi-ss') from dual; mysql> select date_format(now(),'%Y-%m-%d'); mysql> select time_format(now(),'%H-%i-%S');
20	to_date(str,format)	STR_TO_DATE(str,format)	函数名称不同： ORACLE:SELECT to_date('2009-3-6','yyyy-mm-dd') VAULE FROM DUAL MYSQL: SELECT STR_TO_DATE('2004-03-01', '%Y-%m-%d') VAULE	
21	trunc(-1.002)	cast(-1.002 as SIGNED)	函数名称不同： TRUNC函数为指定元素而截去的日期值。 ORACLE： select trunc(-1.002) value from dual MYSQL：select cast(-1.002 as SIGNED) value MYSQL： 字符集转换 : CONVERT(xxx USING gb2312) 类型转换和SQL Server一样,就是类型参数有点点不同 : CAST(xxx AS 类型) , CONVERT(xxx,类型)，类型必须用下列的类型： 可用的类型 二进制,同带binary前缀的效果 : BINARY 字符型,可带参数 : CHAR() 日期 : DATE 时间: TIME 日期时间型 : DATETIME 浮点数 : DECIMAL 整数 : SIGNED 无符号整数 : UNSIGNED	
22	TO_NUMBER(str)	CAST("123" AS SIGNED INTEGER)	函数名称不同 ORACLE:SELECT TO_NUMBER('123') AS VALUE FROM DUAL; MYSQL: SELECT CAST("123" AS SIGNED INTEGER) as value; SIGNED INTEGER:带符号的整形	
23	日期函数	SYSDATE	now() / SYSDATE()	写法不同： ORACLE:select SYSDATE value from dual MYSQL:select now() value select sysdate() value
24	Next_day(sysdate,7)	自定义一个函数:F_COMMON_NEXT_DAY(date,int)	函数名称不同： ORACLE: SELECT Next_day(sysdate,7) value FROM DUAL MYSQL: SELECT F_COMMON_NEXT_DAY(SYSDATE(), 3) value from DUAL; (3:指星期的索引值)返回的指定的紧接着下一个星期的日期	
25	ADD_MONTHS(sysdate, 2)	DATE_ADD(sysdate(), interval 2 month)	函数名称不同: ORACLE: SELECT ADD_MONTHS(sysdate, 2) as value from DUAL; MYSQL: SELECT DATE_ADD(sysdate(), interval 2 month) as value from DUAL;	
26	2个日期相减(D1-D2)	DATEDIFF(date1,date2)	功能: 返回两个日期之间的天数。 用法: mysql: SELECT DATEDIFF('2008-12-30','2008-12-29') AS DiffDate oracle: 直接用两个日期相减（比如d1-d2=12.3)	
27	SQL函数	SQLCODE	MYSQL中没有对应的函数，但JAVA中SQLException。getErrorCode()函数可以获取错误号	Oracle内置函数SQLCODE和SQLERRM是特别用在OTHERS处理器中，分别用来返回Oracle的错误代码和错误消息。 MYSQL: 可以从JAVA中得到错误代码，错误状态和错误消息
28	SQLERRM	MYSQL中没有对应的函数，但JAVA中SQLException。getMessage()函数可以获取错误消息	Oracle内置函数SQLCODE和SQLERRM是特别用在OTHERS处理器中，分别用来返回Oracle的错误代码和错误消息。 MYSQL: 可以从JAVA中得到错误代码，错误状态和错误消息	
29	SEQ_BK_DTL_OPT_INT_KEY.NEXTVAL	自动增长列	在MYSQL中是自动增长列. 如下方法获取最新ID: START TRANSACTION; INSERT INTO user(username,password) VALUES (username,MD5(password)); SELECT LAST_INSERT_ID() INTO id; COMMIT;	
30	SUM(enable_flag)	SUM(enable_flag)	一样： ORCALE： SELECT SUM(enable_flag) FROM SD_USR; MYSQL： SELECT SUM(enable_flag) FROM SD_USR;	
31	DBMS_OUTPUT.PUT_LINE(SQLCODE)	在MYSQL中无相应的方法，其作用是在控制台中打印，用于测试，对迁移无影响。	dbms_output.put_line每行只能显示255个字符，超过了就会报错	

循环语句

编号	类别	ORACLE	MYSQL	注释
1	IF语句使用不同	IF iv_weekly_day = 'MON'THEN ii_weekly_day := 'MON'; ELSIF iv_weekly_day = 'TUE' THEN ii_weekly_day := 'TUE'; END IF;	IF iv_weekly_day = 'MON'THEN set ii_weekly_day = 'MON'; ELSEIF iv_weekly_day = 'TUE' THEN set ii_weekly_day = 'TUE'; END IF;	1. mysql和oracle除了关键字有一个字差别外(ELSEIF/ELSIF),if语句使用起来完全相同. 2. mysql if语句语法: 摘自 MySQL 5.1 参考手册 20.2.12.1. IF语句 IF search_condition THEN statement_list [ELSEIF search_condition THEN statement_list] ... [ELSE statement_list] END IF IF实现了一个基本的条件构造。如果search_condition求值为真，相应的SQL语句列表被执行。如果没有search_condition匹配，在ELSE子句里的语句列表被执行。 statement_list可以包括一个或多个语句。
2	FOR语句不同	FOR li_cnt IN 0.. (ii_role_cnt-1) LOOP SELECT COUNT(*) INTO li_role_ik_cnt FROM SD_ROLE WHERE ROLE_CD = lo_aas_role_upl(li_cnt); IF li_role_ik_cnt = 0 THEN RETURN 'N'; END IF; li_role_ik_cnt := -3; END LOOP;	loopLable:LOOP IF i > (ii_role_cnt-1) THEN LEAVE looplable; ELSE SELECT COUNT*() INTO li_role_ik_cnt FROM SD_ROLE WHERE ROLE_CD = 'ADMIN_SUPER'; /*lo_aas_role_upl(li_cnt);*/ IF li_role_ik_cnt = 0 THEN RETURN 'N'; END IF; SET li_role_ik_cnt = -3; SET i = i+1; END IF; END LOOP loopLable;	1. oracle使用For语句实现循环. Mysql使用Loop语句实现循环. 2. oracle 使用For...loop关键字. Mysql使用loopLable:LOOP实现循环.
3	while语句不同	WHILE lv_inputstr IS NOT NULL LOOP ... END LOOP;	WHILE lv_inputstr IS NOT NULL DO ... END WHILE;	1. oracle 中使用while语句关键字为: while 表达式 loop... end loop; mysql 中使用while语句关键字为: while 表达式 do... end while;

存储过程&Function

编号	类别	ORACLE	MYSQL	注释
1	创建存储过程语句不同	create or replace procedure P_ADD_FAC(id_fac_cd IN ES_FAC_UNIT.FAC_CD%TYPE) is	DROP PROCEDURE IF EXISTS `SD_USER_P_ADD_USR`; create procedure P_ADD_FAC(id_fac_cd varchar(100))	1.在创建存储过程时如果存在同名的存储过程,会删除老的存储过程. oracle使用create or replace. mysql使用先删除老的存储过程,然后再创建新的存储过程. 2. oracle 存储过程可以定义在package中,也可以定义在Procedures中. 如果定义在包中,一个包中可以包含多个存储过程和方法.如果定义在Procedures中,存储过程中不可以定义多个存储过程. Mysql 存储过程中不可以定义多个存储过程. 3. oracle中字符串类型可以使用varchar2. Mysql 需要使用varchar 4. Oracle中参数varchar长度不是必须的, Mysql中参数varchar长度是必须的, 比如varchar(100)
2	创建函数语句不同	CREATE OR REPLACEFUNCTION F_ROLE_FACS_GRP(ii_role_int_key IN SD_ROLE.ROLE_INT_KEY%TYPE) RETURN VARCHAR2	DROP FUNCTION IF EXISTS `SD_ROLE_F_ROLE_FACS_GRP`; CREATE FUNCTION `SD_ROLE_F_ROLE_FACS_GRP` (ii_role_int_key INTEGER(10)) RETURNS varchar(1000)	1.在创建函数时如果存在同名的函数,会删除老的函数. oracle使用create or replace. mysql使用先删除老的函数,然后再创建新的函数. 2. oracle 函数可以定义在package中,也可以定义在Functions中. 如果定义在包中,一个包中可以包含多个存储过程和函数.如果定义在Functions中,每个函数只能定义一个函数. Mysql Functions不可以定义多个函数. 3. oracle返回值用return. Mysql返回值用returns.
3	传入参数写法不同	procedure P_ADD_FAC(id_fac_cd IN ES_FAC_UNIT.FAC_CD%TYPE)	create procedure P_ADD_FAC((in) id_fac_cd varchar(100))	1. oracle存储过程参数可以定义为表的字段类型. Mysql存储过程不支持这种定义方法.需要定义变量的实际类型和长度. 2. oracle 参数类型in/out/inout写在参数名后面. Mysql 参数类型in/out/inout写在参数名前面. 3. oracle 参数类型in/out/inout 都必须写. Mysql 参数类型如果是in,则可以省略. 如果是out或inout则不能省略. 注意: mysql中指定参数为IN, OUT, 或INOUT 只对PROCEDURE是合法的。（FUNCTION参数总是被认为是IN参数） RETURNS字句只能对FUNCTION做指定，对函数而言这是强制的。它用来指定函数的返回类型，而且函数体必须包含一个RETURN value语句。
function func_name(gw_id in(out) varchar2)	create function func_name(gw_id varchar（100))			
4	包的声明方式	create or replace package/package body package name	拆分成多个存储过程或函数	oracle可以创建包,包中可以包含多个存储过程和方法. mysql没有没有包这个概念,可以分别创建存储过程和方法. 每个存储过程或方法都需要放在一个文件中. 例1: 方法命名 oracle 中SD_FACILITY_PKG.F_SEARCH_FAC to mysql SD_FACILITY_F_SEARCH_FAC 例2: 过程命名 oracle 中 SD_FACILITY_PKG.P_ADD_FAC to mysql SD_FACILITY_P_ADD_FAC
5	存储过程返回语句不一样	return;	LEAVE proc; (proc 代表最外层的begin end)	oracle存储过程和方法都可以使用return退出当前过程和方法. Mysql存储过程中只能使用leave退出当前存储过程.不可以使用return. Mysql方法可以使用return退出当前方法.
6	存储过程异常处理不一样	EXCEPTION WHEN OTHERS THEN ROLLBACK ; ov_rtn_msg := c_sp_name||'('|| li_debug_pos ||'):'|| TO_CHAR(SQLCODE)||': '||SUBSTR(SQLERRM,1,100);	DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK ; set ov_rtn_msg = concat(c_sp_name,'(', li_debug_pos ,'):', TO_CHAR(SQLCODE),': ',SUBSTR(SQLERRM,1,100)); END;	oracle : 内部异常不需要定义,在存储过程或函数末尾写上EXCEPTION后,后面的部分即为异常处理的部分. oracle可以定义自定义异常,自定义异常需要使用raise关键字抛出异常后,才可以在EXCEPTION中捕获. mysql: mysql内部异常也需要先定义,在定义的同时也需要实现异常的功能. 目前mysql不支持自定义异常.
7	过程和函数的声明变量的位置不同	声明变量在begin...end体之前	声明变量在begin...end体内，begin之后其他任何内容之前	
8	NO_DATA_FOUND异常处理	EXCEPTION WHEN NO_DATA_FOUND THEN oi_rtn_cd := 1; ov_rtn_msg := SD_COMMON.P_GET_MSG('DP-CBM-01100a-016', li_sub_rtn_cd, lv_sub_rtn_msg);	使用FOUND_ROWS()代替NO_DATA_FOUND. 详见注释.	oracle中: NO_DATA_FOUND是游标的一个属性. 当select没有查到数据就会出现 no data found 的异常，程序不会向下执行. Mysql: 没有NO_DATA_FOUND这个属性.但可是使用FOUND_ROWS()方法得到select语句查询出来的数据.如果FOUND_ROWS()得到的值为0,就进入异常处理逻辑.
9	在存储过程中调用存储过程方式的不同	Procedure_Name(参数);	Call Procedure_Name(参数);	MYSQL存储过程调用存储过程，需要使用Call pro_name(参数). Oracle调用存储过程直接写存储过程名就可以了.
10	抛异常的方式不同	RAISE Exception_Name;	见备注	详见<<2009002-OTMPPS-Difficult Questions-0001.doc>>中2.5 Mysql异常处理部分

触发器

编号	类别	ORACLE	MYSQL	注释
1	创建触发器语句不同	create or replace trigger TG_ES_FAC_UNIT before insert or update or delete on ES_FAC_UNIT for each row	create trigger `hs_esbs`.`TG_INSERT_ES_FAC_UNIT` BEFORE INSERT on `hs_esbs`.`es_fac_unit` for each row	1. Oracle使用create or replace trigger语法创建触发器. Mysql使用 create trigger 创建触发器. 2. Oracle可以在一个触发器触发insert,delete,update事件. Mysql每个触发器只支持一个事件. 也就是说,目前每个trigger需要拆分成3个mysql trigger. 3. mysql trigger 不能在客户端显示或编辑.需要在服务器所在的机器上操作.
2	触发器new和old记录行的引用不同	取得新数据: :new.FAC_CD 取得老数据: :old.FAC_CD	取得新数据: NEW.FAC_CD 取得老数据: OLD.FAC_CD	1. new和old记录行的引用： mysql是NEW.col1，OLD.col1来引用。 oracle是:NEW.col1, :OLD.col1来引用。 2. NEW和OLD不区分大小写.

用户权限

编号	类别	ORACLE	MYSQL	注释
1	创建用户	Create user user_name identified by user_password default tablespace starSpace temporary tablespace temp;	CREATE USER user_name IDENTIFIED BY user_password;	1.oracle创建用户 Oracle 的默认用户有三个: sys / system / scott. 其中sys和system 是系统用户,拥有dba权限, scott用户是Oracle数据库的一个示范账户, 在数据库安装时创建, 不具备dba权限. 创建用户命令: Create user user_name identified by user_password [default tablespace tableSpace] [temporary tablespace tableSpace}; 说明: 每个用户都有一个默认表空间和一个临时表空间,如果没有指定,oracle就将system设置为默认表空间,将temp设为临时表空间. 2.mysql创建用户 创建用户命令: mysql> CREATE USER yy IDENTIFIED BY '123'; yy表示你要建立的用户名，后面的123表示密码 上面建立的用户可以在任何地方登陆。 如果要限制在固定地址登陆，比如localhost 登陆： mysql> CREATE USER yy@localhost IDENTIFIED BY '123';
2	删除用户	Drop user user_name cascade;	Drop user user_name;	1. Oracle SQL>drop user 用户名; //用户没有建任何实体 SQL> drop user 用户名 CASCADE; // 将用户及其所建实体全部删除 注: 当前正连接的用户不得删除。 2. Mysql 自4.1.1以后，删除一个MYSQL帐户，可以使用drop user 语句了。 不过在5.0.2之前的版本中，drop user语句只能删除没有任何权限的用户。 从5.0.2往后的版本中，drop user语句可以删除任何用户。(当然不能自己删自己)。示例：drop user "garfield"@"localhost"。别忘了加后面的@，不然会报错。 在4.1.1与5.0.2之间的版本中要删除一个MYSQL帐户，需要进行以下操作。 1) 使用show grants语句查看要删除的MYSQL帐户都有哪些权限，使用方法如show grants for "garfield"@"localhost"。 2) 使用revoke语句收回用户在show grants里拥有的权限。执行这个语句将删除除user表之外的其它所有权限表中的相关记录，并且收回在user表中该用户拥有的全局权限。 3) 使用drop user 语句把用户从user表中删除。

编号	类别	ORACLE	MYSQL	注释
3	修改密码	alter user user_name identified by new_password	mysqladmin -u root -p 123456 password "your password";	1.mysql修改密码 第一种方式： 1) 更改之前root没有密码的情况 c:\mysql\bin>mysqladmin -u root password "your password" 2) 更改之前root有密码的情况,假如为123456 c:\mysql\bin>mysqladmin -u root -p123456 password "your password" 注意：更改的密码不能用单引号，可用双引号或不用引号 第二种方式： 1) c:\mysql\bin>mysql -uroot -p密码 以root身份登录 2) mysql>use mysql 选择数据库 3) mysql>update user set password=password('你的密码') where User='root'; 4) mysqlflush privileges; 重新加载权限表
4	设置用户权限	Grant connect to star -- star角色允许用户连接数据库, 并创建数据库对象 Grant resource to star -- star角色允许用户使用数据库中的存储空间. Grant dba to star -- DBA权限	GRANT ALL ON picture.* TO test IDENTIFIED BY "test";	1. 详见<<oracle vs mysql 用户权限.doc>> 2.1 Oracle权限设置 2. 详见<<oracle vs mysql 用户权限.doc>> 1.4 用户权限设置

编号	类别	ORACLE	MYSQL	注释
5	回收权限	Revoke select, update on product from user02;	REVOKE privileges (columns) ON what FROM user	1. Oracle Revoke语句的基本格式如下： REVOKE 权限类型 [（字段列表）] [, 权限类型 [（字段列表）]...]ON {数据库名称.表名称}FROM 用户名@域名或IP地址 例如，管理员撤销用户admin@localhost对数据库xsxk所拥有的创建、创建数据库及表的权限，并撤销该用户可以把自己所拥有的权限授予其他用户的权限，可使用以下命令。 mysql>revoke create,drop on xsxk.* from admin@localhost; mysql>revoke grant option on xsxk.* from admin@localhost; revoke语句中的“用户名@域名或IP地址”部分必须匹配原来grant语句中的“用户名@域名或IP地址”部分，而“权限类型”部分可以是所授权的一部分权限。而且，revoke只能撤销权限，不能删除用户账户，在授权表user中仍保留该用户的记录；用户仍可以连接到数据库服务器。如果要完全删除用户，则使用前面提到的delete语句从user表中删除该用户记录。 2. Mysql 要取消一个用户的权限，使用REVOKE语句。REVOKE的语法非常类似于GRANT语句，除了TO用FROM取代并且没有INDETIFED BY和WITH GRANT OPTION子句： REVOKE privileges (columns) ON what FROM user user部分必须匹配原来GRANT语句的你想撤权的用户的user部分。privileges部分不需匹配，你可以用GRANT语句授权，然后用REVOKE语句只撤销部分权限。 REVOKE语句只删除权限，而不删除用户。即使你撤销了所有权限，在user表中的用户记录依然保留，这意味着用户仍然可以连接服务器。要完全删除一个用户，你必须用一条Delete语句明确从user表中删除用户记录

其它

编号	类别	ORACLE	MYSQL	注释
1	内连接的更改	1、 select a., b., c., d. from a, b, c, d where a.id = b.id and a.name is not null and a.id = c.id(+) and a.id = d.id(+) "(+)"所在位置的另一侧为连接的方向， 所以上面的例子1是左连接。 以下的例子2既是右连接。2、 select a., b., c., d. from a, b, c, d where a.id = b.id and a.name is not null and a.id(+) = c.id	方法一 select a., c., d.* from a left join(c, d) on (a.id = c.id and a.id = d.id), b where a.id = b.id and a.name is not null 方法二 select a., c., d.* from a left join c on a.id = c.id left join d on a.id = d.id, b where a.id = b.id and a.name is not null	oracle sql语句和mysql sql语句有一定的区别. 1. oracle左连接, 右连接可以使用(+)来实现. Mysql只能使用left join ,right join等关键字.
2	最后一句执行的sql statement所取得或影响的条数	SQL%ROWCOUNT	执行select语句后用: FOUND_ROWS() 执行update delete insert语句后用: ROW_COUNT().	oracle中: sql 表示最后一句执行的 SQL Statement, rowcount表示該 SQL 所取得或影响的条数. Mysql中: 执行select语句后查询所影响的条数用: FOUND_ROWS() 执行update delete insert语句后查询所影响的条数用: ROW_COUNT()
3	查询分页	SELECT t1.* FROM (SELECT MSG_INT_KEY, MSG_TY, MSG_CD, ROWNUM ROW_NUM FROM SD_SYS_MSG WHERE (ii_msg_int_key IS NULL OR msg_int_key = ii_msg_int_key) ORDER BY MSG_CD) t1 WHERE (in_page_no IS NULL) OR (t1.ROW_NUM > ((in_page_no -1)*li_per_page_amt) AND t1.ROW_NUM < (in_page_no*li_per_page_amt + 1));	方法:使用循环变量替换oracle中ROWNUM set @mycnt = 0; SELECT (@mycnt := @mycnt + 1) as ROW_NUM,t1.* FROM (SELECT MSG_INT_KEY, MSG_TY, MSG_CD, ROWNUM ROW_NUM FROM SD_SYS_MSG WHERE (ii_msg_int_key IS NULL OR msg_int_key = ii_msg_int_key) ORDER BY MSG_CD) t1 WHERE (in_page_no IS NULL) OR (t1.ROW_NUM> ((in_page_no - 1) * li_per_page_amt) AND t1.ROW_NUM < (in_page_no * li_per_page_amt + 1));	
4	java null值	""作为参数传入后,在oracle中将识别为null	""作为参数据传mysql还是""	现在java代码需要修改: inPara.add(MSG_TY.equals("") ? null : MSG_TY);
5	执行动态sql	lv_sql := 'SELECT ' ||' distinct ' || iv_cd_field_name || ' FIELD1 '|| ' FROM ' || iv_table_name || ' WHERE ' || NVL(iv_where_cause,' 1=1 '); OPEN l_sys_cur FOR lv_sql;	set @a = iv_cd_field_name; set @b = iv_table_name; set @c = IFNULL(iv_where_cause,' 1=1 '); SET @s = concat('SELECT distinct ', @a , ' FIELD1 FROM ' , @b , ' WHERE ' , IFNULL(@c,' 1=1 ')); PREPARE stmt3 FROM @s; EXECUTE stmt3; DEALLOCATE PREPARE stmt3;	1. oracle可以将动态sql放在游标中执行. mysql游标声明有一定的局限性: mysql游标必须在声明处理程序之前被声明，并且变量和条件必须在声明光标或处理程序之前被声明。Mysql采用Prepared Statements实现动态sql. 例子如下: INT Emp_id_var = 56 PREPARE SQLSA FROM "DELETE FROM employee WHERE emp_id=?" ; EXECUTE SQLSA USING :Emp_id_var ;
6	存储过程相互调用时传递数组	oracle使用数组步骤: 1. 将传入的字符串通过P_UNPACK_LIST方法转换为数组.(lo_holiday_jan_upl即为数组) P_UNPACK_LIST(iv_jan__str, lv_delimiter, lo_holiday_jan_upl); 2. 传数组到另一个存储过程. P_MOD_MONTH(iv_year, 1, lo_holiday_jan_upl, iv_user_cd); 3. P_MOD_MONTH中使用数组: (将数组中的各个元素取出来插入到SD_HOLIDAY表) FOR li_cnt IN 0 .. 9 LOOP IF iv_daystr(li_cnt) IS NOT NULL THEN INSERT INTO SD_HOLIDAY (HOLIDAY_INT_KEY, YEAR, MONTH, DAY, ENABLE_FLAG, CREATE_BY, CREATE_DATE, LAST_UPD_BY, LAST_UPD_DATE) VALUES (SEQ_HOLIDAY_INT_KEY.NEXTVAL, iv_year, iv_month, iv_daystr(li_cnt), 1, iv_user_cd, ld_sys_date, iv_user_cd, ld_sys_date); END IF;END LOOP;	mysql中数用数组步骤: 1. 将需要处理的字符串交给执行业务逻辑 的存储过程处理. CALL SD_HOLIDAY_P_MOD_MONTH(iv_year, 1, iv_jan__str, iv_user_cd); 2. SD_HOLIDAY_P_MOD_MONTH中处理字符串. (将字符串按自定格式分隔出来,在对每个小字符串进行业务逻辑处理.) SET lv_inputstr = iv_inputstr; loopLable:LOOP IF li_cnt > 9 THEN LEAVE looplable; ELSE SET li_pos = INSTR(lv_inputstr, iv_delimiter); IF li_pos = 0 THEN leave looplable; ELSE set temp_str = SUBSTR(lv_inputstr, 1, li_pos - 1); *//插入temp_str到SD_HOLIDAY表//* INSERT INTO SD_HOLIDAY(...) SET lv_inputstr = SUBSTRING(lv_inputstr, li_pos + LENGTH(iv_delimiter)); END IF; SET li_cnt = li_cnt+1; END IF; END LOOP loopLable;	存储过程相互调用时传递数组解决方法: oracle中传入12个字符串到存储过程,然后将这12个字符串转换为12个数组,再调用其他存储过程并将这12个数组分别传给存储过程,便利每个数组进行业务逻辑处理. mysql解决方法: 将存储过程中的数组去掉,两个存储过程调用时直接传递字符串,然后再需要处理业务逻辑的地方将字符串分解,进行业务逻辑处理. 可以参考<<2009002-OTMPPS-Difficult Questions-0001.doc>> 中 2.4.2 逐层分解字符串
7	Java无法以String来接取int	select fac_unit_key FILED1在oracle可以	select fac_unit_key FILED1在mysql中要改 select CAST(fac_unit_key AS CHAR) FILED1	CAST(intvalue AS CHAR)