从 Oracle 至 SinoDB 迁移技术指南

SINOREGAL
星瑞格软件

文件更改履历表

修改日期	版本	修改说明	修改者
2022-05-17	1.0	初稿	林质颖，林雅

目录

1. 引言

1.1 目的

为规范后续星瑞格 SinoDB 建设，特制定星瑞格 SinoDB SQL 与 ORACLE SQL 对照文档，以便达到以下几个目标：

便于数据库的管理和维护

便于后续的迁移工作

便于开发人员的开发和查阅

1.2 适用对象和范围

本规范适用于所有的设计、实施、运行和维护阶段，是开发的标准规范。适用的人员包括项目经理、数据设计人员、数据库管理员、应用开发人员和系统维护人员。

2. 创建数据库

ORACLE	星瑞格 SinoDB
Oracle 手动建库较为复杂，一般采用 dbca 的方式完成建库。	SinoDB 一般通过命令行完成建库操作。 建库语法： create database [if not exists] <db_name> [in dbs_name] [with [buffered] log | with log mode ansi] [nlscasesensitive | nlscase insensitive]; 重命名数据库： rename database <old_db_name> to <new_db_name>; 删除数据库： drop database [if exists] <db_name>; 示例：

	create database db1 in datadbs1 with buffered log; --in 指定数据库所在 dbspace 为 datadbs1 rename database db1 to db2; drop database db2;
注意：在 SinoDB 中创建数据库时，建议指定数据库所在的 dbspace,如果不指定的话会默认创建在根数据库表空间中，随着数据库数据量的增长，可能会撑爆根数据库表空间，影响数据库运行。如果要创建带日志的数据库，需要使用 with log 或者 with buffered log 来指定。	

3. 创建表

3.1 基本语法

ORACLE	星瑞格 SinoDB
CREATE TABLE accounts(acct_id NUMBER(3) NOT NULL, dept_id CHAR(3) NOT NULL, acct VARCHAR2(2000), num_projects NUMBER(1), CONSTRAINT PK_ACCT_ID PRIMARY KEY(ACCT_ID), CONSTRAINT FK_ACC_DEPT_CODE FOREIGN KEY (DEPT_CODE) REFERENCES DEPARTMENTS (DEPT_ID)) STORAGE (INITIAL 64k NEXT 128K MINEXTENTS 1 MAXEXTENTS 50) TABLESPACE user_data_tbs;	CREATE TABLE accounts(acct_id SMALLINT NOT NULL, deptid CHAR(3) NOT NULL, acct LVARCHAR(2000), num_projects SMALLINT, PRIMARY KEY(acct_id) CONSTRAINT PK_ACCT_ID, FOREIGN KEY (dept_id) REFERENCES DEPARTMENTS(DEPT_ID) CONSTRAINT FK_ACC_DEPT_CODE) IN user_data_tbs EXTENT SIZE 64 NEXT SIZE 128 LOCK MODE ROW;
对比说明：创建表时，Oracle 的建表语句需要进行转化，注意主外键约束写法的区别。需要使用 SinoDB 的 IN,EXTENT,NEXT 和 LOCK MODE 选项。 IN 指定表将存放在哪个 dbspace 中； EXTENT 指定初始分配给该表的空间大小； NEXT 指定追加给该表的空间大小； LOCK MODE 指定锁模式，SinoDB 默认为页级锁，可以在创建表时指定为行级别，	

也可以通过修改 ONCONFIG 参数文件中的 DEF_TABLE_LOCKMODE 进行指定。

Oracle 中的 STORAGE INITIAL 和 STORAGE NEXT 语句必须分别替换为SinoDB 中的 EXTENT SIZE 和 NEXT SIZE 语句。Oracle 的存储子句选项 MINEXTENTS, MAXEXTENTS 和 PCTINCREASE 必须被移除。

3.2 约束选项

ORACLE	星瑞格 SinoDB
主键约束：	
Oracle 语法： CONSTRAINT name PRIMARY KEY(column(s)) 示例： CONSTRAINT PK_ACCT_ID PRIMARY KEY(ACCT_ID)	SinoDB 语法： PRIMARY KEY (column(s)) CONSTRAINT name 示例： PRIMARY KEY(ACCT_ID) CONSTRAINT PK_ACCT_ID
外键约束：	
Oracle 语法： CONSTRAINT name FOREIGN KEY(column(s)) references_clause [ON DELETE CASCADE / ON DELETE SET NULL] 示例： CONSTRAINT FK_ACC_DEPT_CODE FOREIGN KEY (DEPT_CODE) REFERENCES DEPARTMENT (DEPT_CODE) ON DELETE CASCADE	SinoDB 语法： FOREIGN KEY(column(s)) references_clause [ON DELETE CASCADE] CONSTRAINT name 示例： FOREIGN KEY (DEPT_CODE) REFERENCES DEPARTMENT (DEPT_CODE) ON DELETE CASCADE CONSTRAINT FK_ACC_DEPT_CODE
唯一约束	
Oracle 语法： CONSTRAINT name UNIQUE(column(s)) 示例： CONSTRAINT UK_PHONE UNIQUE(PHONE)	SinoDB 语法： UNIQUE (column(s)) CONSTRAINT name 示例： UNIQUE(PHONE) CONSTRAINT UK_PHONE
检查约束	
Oracle 语法： CONSTRAINT name CHECK (condition)	SinoDB 语法： CHECK (condition) CONSTRAINT name

示例： CONSTRAINT CHK_ORD_MONTH CHECK (ORDER_MONTH BETWEEN 1 AND 12)	示例： CHECK (ORDER_MONTH BETWEEN 1 AND 12) CONSTRAINT CHK_ORD_MONTH drop table t7; create table t7(c1 int, phone char(11), ORDER_MONTH int, UNIQUE(PHONE) CONSTRAINT UK_PHONE, CHECK (ORDER_MONTH BETWEEN 1 AND 12) CONSTRAINT CHK_ORD_MONTH);

3.3 分片选项

ORACLE	星瑞格 SinoDB
SinoDB 有 round robin、expression-based、Range、List 四种分片方式， Oracle 有 Range partitioning、HASH partitioning、List partitioning 和 Composite Range-Hash partitioning 四种分片方式，对照关系如下：	
Range partitioning	Expression Based /Range Fragmentation
HASH partitioning	Round Robin Fragmentation
List partitioning	Expression Based /List Fragmentation
Composite Range-Hash partitioning	Expression Based Fragmentation
示例： CREATE TABLE accounts (acct_id NUMBER(3) NOT NULL, dept_code CHAR(3) NOT NULL, acct_desc VARCHAR2(2000), max_employees NUMBER(3), current_employees NUMBER(3), num_projects NUMBER(1)) PARTITIONING BY RANGE (acct_id)	***Expression-Based 分区示例:*** CREATE TABLE accounts(acct_id SMALLINT NOT NULL, dept_code CHAR(3) NOT NULL, acct_desc LVARCHAR(2000), max_employees SMALLINT, current_employees SMALLINT, num_projects SMALLINT,) FRAGMENT BY EXPRESSION

(PARTITION part1 VALUES LESS THAN (100) TABLESPACE part1_tbs, (PARTITION part2 VALUES LESS THAN (200) TABLESPACE part2_tbs, (PARTITION part3 VALUES LESS THAN (300) TABLESPACE part3_tbs, (PARTITION part4 VALUES LESS THAN (MAXVALUE) TABLESPACE part4_tbs);	acct_id < 100 IN dbspace1, acct_id >= 100 AND acct_id < 200 IN dbspace2, acct_id >= 200 AND acct_id < 300 IN dbspace3, REMAINDER IN dbspace4; ***Range 分区示例：*** create table sales(amount int, id int, cdatetime datetime year to second) fragment by range(cdatetime) interval(10 units day) store in (datadbs2,datadbs3,datadbs4) partition p1 values < '2011-01-01 00:00:00' in datadbs1; ***List 分区示例：*** create table customer_list(cust_id integer, name varchar(128), province varchar(30)) Fragment by list(province) PARTITION p0 values('FJ','JX') in datadbs1, PARTITION p1 values('HB','HN') in datadbs2, PARTITION p2 values('SX','SD') in datadbs3, PARTITION p3 values(NULL) in datadbs4; ***Round Robin 分区示例：*** create table t_round(c1 serial, c2 char(20)) fragment by round robin In dbspaces1,dbspaces2

	extent size 10000 next size 3000;

4. 数据类型

本节描述了星瑞格 SinoDB 数据库中使用的数据类型。

数据类型指定一个数值的特性。一个特殊值 NULL 包含在每个数据类型中表示值的缺省。

4.1 数据类型分类

星瑞格 SinoDB 数据库中内置的数据类型的特点可分为如下表：

分类	数据类型	描述
数值类型	SERIAL，SERIAL8，BIGSERIAL，INTEGER(INT)，INT8，BIGINT，SMALLINT，FLOAT，SMALLFLOAT，DOUBLE PRECISION，REAL，MONEY，DECIMAL，NUMERIC	
字符类型	BOOLEAN，CHAR，VARCHAR，LVARCHAR，NCHAR，NVARCHAR	
时间类型	（1） 一般日期时间数据类型: DATE DATETIME （2） 时间间隔数据类型 ● 年-月间隔类 INTERVAL YEAR TO MONTH INTERVAL YEAR INTERVAL MONTH ● 日-时间隔类 INTERVAL DAY INTERVAL DAY TO HOUR INTERVAL DAY TO MINUTE INTERVAL DAY TO SECOND INTERVAL HOUR INTERVAL HOUR TO MINUTE INTERVAL HOUR TO SECOND INTERVAL MINUTE	

	INTERVAL MINUTE TO SECOND INTERVAL SECOND	
多媒体数据类型	TEXT，BYTE，BLOB，CLOB	

4.2 数值类型

4.2.1 NUMERIC

ORACLE	星瑞格 SinoDB
NUMBER(p,s)	NUMERIC(p,s)/DECIMAL(p,s)
（1） 标度值域：Oracle 是[-84,127]。即 Oracle 标度可为负数，表示精确到小数点左侧多少位。如：1234.56 存到 Number(6,-2)中，表示精确到百位，最后结果为 1200。 （2） 精度表示方式不同：Oracle 精度可用*代替，表示精度为 38。 （3） 精度缺省值不同： Oracle 根据给定的值存储， 如 123.456， 则 p=3。 （4）精度和标度大小关系：Oracle 无要求，精度小于标度时， 表示小数点后有效数字最多可以是标度长度，如 0.000123 存在 Number(4,5)中，结果为：0.00012。	可存储的数字范围是 10^-130 到 10^124，只有 32 位有效位。使用 16 字节存储。 格式：decimal(p)浮点和 decimal(p,s)定点，p 指定精度，即所有数字有效位数，s 是小数位数。

4.2.2 INTEGER, INT

ORACLE	星瑞格 SinoDB
NUMBER	INTEGER(INT)
同 2.2.1 ORACLE NUMBER。	-(2^31-1) 到 (2^31-1) ，使用 4 个字节存储，10 位精度范围。

4.2.3 SMALLINT

ORACLE	星瑞格 SinoDB
NUMBER	SMALLINT
无此数据类型定义为 smallint, int 或 integer 时， 实际的数据类型为 number(22， 0)	-32767（-（2^15-1） 到 32767 (2^15-1))，2 字节，只存储整数。使用 2 个字节存储。

4.2.4 FLOAT

ORACLE	星瑞格 SinoDB

FLOAT[(n)]	FLOAT[(n)]
Oracle 的 n 表示二进制精度，范围是[1,126]，要转为 10 进制精度需要乘以因子 0.30103，所以对应的 10 进制精度范围是[1,38]，n 默认是 126。	16 个有效位双精度浮点数，使用 8 个字节存储，float(n)n 必须是 1-16 之间的整数。当迁移的数据超出范围时可以考虑使用 decimal。

4.2.5 REAL

ORACLE	星瑞格 SinoDB
REAL	SMALLFLOAT
REAL 数据类型定义一个 18 位单精度浮点数。	同 smallfloat：8 个有效数字的单精度浮点数，使用 4 个字节存储， smallfloat 转换为 decimal 值产生 9 位精度。当迁移的数据超出范围时可以考虑使用 float/decimal。

4.3 字符类型

4.3.1 CHAR/NCHAR

ORACLE	星瑞格 SinoDB
CHAR(n),NCHAR(n)	CHAR(n),NCHAR(n)
值域： CHAR=2000 byte NCHAR=2000 char| byte。	CHAR(n)，存储字符串数据，范围从 1 到 32767 字节，n 字节长度。 NCHAR(n)，存储 Unicode 字符串类型，范围从 1 到 32767 字节。

4.3.2 VARCHAR2/VARCHAR

ORACLE	星瑞格 SinoDB
VARCHAR2	VARCHAR/LVARCHAR
可变长度的字符数据，最大长度为 4000 字节；VARCHAR2 把一般情况下所有字符都占两字节处理；VARCHAR2 把空串等同于 NULL 处理 VARCHAR2 字符要用几个字节存储，要看数据库使用的字符集 值域： VARCHAR2=4000 byte， 默认长度： VARCHAR2 必须指定长度	VARCHAR(m,r)，存储范围位 1 到 255 字节，其中 m 是列的最大大小，r 是保留最小字节数。 LVARCHAR,存储范围从 1 到 32739 字节，缺省大小为 2048 个字节。 当字符串长度超出 255 字节时，使用 LVARCHAR 进行存储。

4.3.3 NVARCHAR2/NVARCHAR

ORACLE	星瑞格 SinoDB
NVARCHAR2	NVARCHAR
NVARCHAR2 用于存储可变长度的字符串，SIZE 的最大值是 2000，而最小值是 1，其值表示字符的个数，而不是字节数 值域： NVARCHAR2=2000 char|4000 byte。	存储范围位 1-255，NVARCHAR(m,r)其中 m 是列的最大大小，r 是保留最小字节数。

4.4 时间类型

4.4.1 TIMESTAMP

ORACLE	星瑞格 SinoDB
TIMESTAMP	DATETIME
可用的日期范围从 BC4712 年 1 月 1 日至 AC9999 年 12 月 31 日。 数据包含世纪，年份，月，日，时，分，秒信息，最小时间粒度是秒。 默认格式为”YYYY-MM-DD” 可显示格式：“yyyy-mm-dd hh:mm:ss”	自 1899 年 12 月 31 日以来的时间，使用 8 字节存储，时间精度为毫秒

4.4.2 DATE

ORACLE	星瑞格 SinoDB
DATE	DATE
可用的日期范围从 BC4712 年 1 月 1 日至 AC9999 年 12 月 31 日。 数据包含世纪，年份，月，日，时，分，秒信息，最小时间粒度是秒。 默认格式为”YYYY-MM-DD” 可显示格式：“yyyy-mm-dd hh:mm:ss”	DATE 存储自 1899 年 12 月 31 日以来的日期，4 字节，时间精度为日。 当迁移的字段存储具体时间时，可以考虑使用 DATETIME，存储自 1899 年 12 月 31 日以来的时间，例： datetime year to second default current year to second
对比说明： Oracle 的 Date 包括 “年月 日时分秒”,星瑞格 SinoDB 的不包括。	

4.4.3 INTERVAL YEAR TO MONTH

ORACLE	星瑞格 SinoDB
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH
功能一致	year-month 时间间隔和 day-time 时间间隔，4 字节，可精确到秒。

4.4.4 INTERVAL DAY TO SECOND

ORACLE	星瑞格 SinoDB
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND
功能一致。	year-month 时间间隔和 day-time 时间间隔，4 字节，可精确到秒。

4.4.5 TIMESTAMP WITH TIME ZONE

ORACLE	星瑞格 SinoDB
TIMESTAMP WITH TIME ZONE	无
存储时区信息。	SinoDB 是按环境变量来定义时区。 可以利用 TZ 环境变量来改变 server time zone。例如，如果将太平洋标准时间与太平洋夏令时一起使用，可以设置：export TZ=PST8PDT，设置 TZ 环境变量后需要重启数据库，DB 在 start 的时候抓 TZ。下面的例子展示相同的 UTC 在不同时区下显示的时间差异：

4.4.6 TIMESTAMP WITH LOCAL TIME ZONE

ORACLE	星瑞格 SinoDB
TIMESTAMP WITH LOCAL TIME ZONE	无
将时间数据转换为数据库时区的时间数据进行存储，但不存储时区信息；客户端检索时，oracle 会将数据库中存储的时间数据转换为客户端 session 时区的时间数据后返回给客户端	SELECT DBINFO('utc_to_datetime', -32767) AS u1, DBINFO('utc_to_datetime', +32767) AS u2, DBINFO('utc_to_datetime', 1299912999) as u3, DBINFO('utc_to_datetime', -2134567890.91234) as u4 FROM sysmaster:sysdual;

	# Server running in TZ=US/Pacific, 结果为： 1969-12-31 06:53:53 1970-01-01 01:06:07 2011-03-11 22:56:39 1902-05-12 01:28:30 # Server running in TZ=UTC0, 结果为 1969-12-31 14:53:53 1970-01-01 09:06:07 2011-03-12 06:56:39 1902-05-12 08:28:30 #从这两个查询结果可以看到，由于由于美国太平洋时区和 UTC0 时区之间有 8 小时的偏移量，因此第三个 DBINFO 结果中的日分量不同。

4.5 大对象数据类型

4.5.1 LONG

ORACLE	星瑞格 SinoDB
LONG	无
最大长度为 2G-1 byte。	可以考虑采用 byte 类型或 blob 类型进行存储。 BYTE 存储二进制字节流数据，如电子表格，强制转换可将 byte 转换为 blob。最大支持 2G。

4.5.2 BLOB

ORACLE	星瑞格 SinoDB
BLOB	BLOB
可用来存储无结构的二进制数据，最大长度是 4GB	智能二进制对象，一般用来存储数字对象，如图像、声音、视频等，最大支持 4T 大小

4.5.3 CLOB

ORACLE	星瑞格 SinoDB
CLOB	CLOB
用户存储对应于数据库定义的字符集的字符数据，最大长度为 4GB.	智能大字符对象，一般用来存储文本类信息，最大支持 4T 大小

4.5.4 (LONG) RAW

ORACLE	星瑞格 SinoDB
(LONG) RAW	无
可变长二进制数据最大长度 134217727，表中只能有一个 LONG RAW 列	SinoDB 没有该类型，可以考虑采用 byte 类型或 blob 类型进行存储。

4.5.5 NCLOB

ORACLE	星瑞格 SinoDB
NCLOB	无
存储内部定长多字节字符打对象，最大长度 4G	考虑用 CLOB 类型存储。

4.5.6 BFILE

ORACLE	星瑞格 SinoDB
BFILE	无
Oracle 支持外部的二进制文件，大小受操作系统限制，最大支持 4G	SinoDB 没有该类型，可以考虑采用 byte 类型或 blob 类型进行存储。

5. 数据类型转换

5.1 时间类型函数转换

5.1.1 TO_DATE

ORACLE	星瑞格 SinoDB
TO_DATE	TO_DATE
语法：	语法：

TO_DATE(CHAR [.fmt[,'nlsparam']]) 解释： to_date converts char of CHAR,VARCHAR2,NCHAR,OR NVARCHAR2 datatype to a value of DATE datatype. The fmt is a data format specifying the format of char. If you omit fmt, then char must be in the default data format.if fmt is 'J', for Julian,then char must be an integer。 例子： SELECT TO_DATE('20150101 22:22:22','YYYYMMDD hh24:mi:ss') FROM DUAL;	to_date(字符串|列，转换格式) 例子： select to_char(current,"%Y-%m-%d"), to_date("1978-10-07 10:23:41","%Y-%m-%d %H:%M:%S") tm from sysmaster:sysdual;

5.1.2 TO_TIMESTAMP

ORACLE	星瑞格 SinoDB
TO_TIMESTAMP	无
把字符串转换成 TIMESTAMP 型	可以使用 to_date 进行转换。 语法： to_date(字符串|列，转换格式) 例子： select to_char(current,'"%A %B %d, %Y %R"') as tc, to_date("1978-10-07 10:23:41.234","%Y-%m-%d %H:%M:%S.%F3") as tm from sysmaster:sysdual; 说明： 其中：%A：周、%B：月（英文）、%m 月（数字）、%d：日、%Y：年，%R:时间。毫秒用%Fn，其中 n 表示精确到几位，缺省为 2，有效值为 0~5。要精确到一位、两位、三位的毫秒，n 就用 1,2,3 表示。按照以上方法，可以将日期时间型按照自己喜欢的格式进行加工。

5.2 数值类型函数转换

5.2.1 TO_NUMBER

ORACLE	星瑞格 SinoDB
TO_NUMBER	TO_NUMBER
语法： TO_NUMBER(CHAR[,fmt[,'nlsparam']]) 解释： TO_NUMBER converts char,a value of char, varchar2,nchar or nvarchar2 datatype containing a number in the format specified by the optional format model fmt,to a value of NUMBER datatype 例子： select to_number('09') + to_number('10') as c1 , to_number('09') * to_number('10') as c2 From dual;	功能一致。 例子： select to_number('09') + to_number('10') as c1 , to_number('09') * to_number('10') as c2 from sysmaster:sysdual;

5.3 字符类型转换

5.3.1 TO_CHAR

ORACLE	星瑞格 SinoDB
TO_CHAR(VALUE)	TO_CHAR(VALUE)
语法： TO_CHAR(CHAR[,fmt[,'nlsparam']]) 解释： （1）TO_CHAR(character) converts NCHAR,NVARCHAR2,CLOB, OR NCLOB data to the datebase character set. （2）TO_CHAR(datetime) converts date of DATE,TIMESTAMP,TIMESTAMP WITH TIME ZONE, OR TIMESTAMP WITH LOCAL TIME ZONE datetype to a value of VARCHAR2 datatype in the formate specified by the date formate fmt	示例： SELECT to_char(current,'%Y-%-m-%d %H:%M:%S') tc from sysmaster:sysdual;

（3）TO_CHAR(number) converts n of NUMBER datatype to a value of VARCHAR2 datetype to a value of VARCHAR2 datatype,using the optional number format fmt.if you omit fmt,the n is converted to a VARCHAR2 value exactly long enough to hold its significant digits 示例： SELECT TO_CHAR(SYSDATE,'YYYY-MM-DD HH24:MI:SS') FROM DUAL; --判断当前时间处于哪个季度 SELECT TO_CHAR(SYSDATE,'Q') FROM DUAL; SELECT TO_CHAR(SYSDATE,'YYYYQ') FROM DUAL; SELECT TO_CHAR(SYSDATE,'IW') FROM DUAL; --取当前分钟数 SELECT TO_CHAR(SYSDATE,'HH24') FROM DUAL; --获取当天在本星期的星期数 SELECT TO_CHAR(SYSDATE,'DAY') FROM DUAL; --星期四 SELECT TO_CHAR(123.123) FROM DUAL;	

6. 函数类型转换

6.1 日期时间函数定义转换

6.1.1 ADD_DAYS

ORACLE	星瑞格 SinoDB
SYSDATE+N	INTERVAL(n) day to day
SELECT TO_DATE ('2009-12-05', 'YYYY-MM-DD')+30 FROM DUAL;	select current c1, current year to day + interval(9) day to day as

	c2 from sysmaster:sysdual;

6.1.2 ADD_MONTHS

ORACLE	星瑞格 SinoDB
ADD_MONTHS	ADD_MONTHS
select sysdate, add_months(sysdate,1) from dual; c1 c2 2017/8/17 1:50:42 2017/9/17 1:50:42	语法： ADD_MONTHS(日期,＜i＞) 示例： select sysdate c1, add_months(sysdate,1) c2 from sysmaster:sysdual;

6.1.3 ADD_WEEKS

ORACLE	星瑞格 SinoDB
SYSDATE+7	无
SELECT TO_DATE ('2009-12-05', 'YYYY-MM-DD')+7 FROM DUAL;	select current c1, current year to day + interval(7) day to day as c2 from sysmaster:sysdual;

6.1.4 CURRENT_DATE/ CURDATE

ORACLE	星瑞格 SinoDB
CURRENT_DATE	TODAY
返回当前日期。	select today from sysmaster:sysdual;

6.1.5 CURRENT_TIMESTAMP

ORACLE	星瑞格 SinoDB
CURRENT_TIMESTAMP	无
select CURRENT_TIMESTAMP from dual; CURRENT_TIMESTAMP --- 20-MAY-22 05.38.41.593414 PM +08:00	

6.1.6 MONTHS_BETWEEN

ORACLE	星瑞格 SinoDB

MONTHS_BETWEEN	MONTHS_BETWEEN(,)
MONTHS_BETWEEN MONTHS_BETWEEN(X,Y) 返回 X,Y 之间有几个月。如果 X 在日历中比 Y 晚，就返回正值，否则为负值 SELECT MONTHS_BETWEEN(TO_DATE('2010-01-05', 'YYYY-MM-DD'), TO_DATE ('2009-12-05', 'YYYY-MM-DD')) FROM DUAL; --1 SELECT MONTHS_BETWEEN(TO_DATE('2010-01-05', 'YYYY-MM-DD'), TO_DATE ('2009-12-06', 'YYYY-MM-DD')) FROM DUAL; --0.967741935483871	select months_between(today,add_months(today,5)) as months from sysmaster:sysdual;

6.1.7 DAY/ MONTH/ WEEKDAY/ YEAR/QUARTER

ORACLE	星瑞格 SinoDB
TO_CHAR(TO_DATE())	

SELECT TO_CHAR(TO_DATE('20110101', 'YYYYMMDD'), 'YYYYIW') AS WEEK, --ORACLE 求当年的总周
TO_CHAR(TO_DATE('20110101', 'YYYYMMDD'), 'YYYYWW') AS WEEK2, --ORACLE 求当年的第几周
TO_CHAR(TO_DATE('20110101', 'YYYYMMDD'), 'YYYY') AS YEAR, --ORACLE 求第几年
TO_CHAR(TO_DATE('20110101', 'YYYYMMDD'), 'YYYYMM') AS MONTH, --ORACLE 求当年的第几月
TO_CHAR(TO_DATE('20110101', 'YYYYMMDD'), 'YYYYDDD') AS DAY, --ORACLE 求当年的第几天道
TO_CHAR(TO_DATE('20110401', 'YYYYMMDD'), 'YYYYQ') AS QUARTER --ORACLE 求当年的第几季度
FROM DUAL
	1）DAY（DATE/DATETIME EXPRESSION）返回指定表达式中的当月几号，例：
	select day(today) from sysmaster:sysdual;
	
	2）MONTH（DATE/DATETIME EXPRESSION） 返回指定表达式中的月份，例：
	select month(today) from sysmaster:sysdual;
	
	3）YEAR（DATE/DATETIME EXPRESSION） 返回指定表达式中的年份，例：
	select year(today) from sysmaster:sysdual;
	
	4）WEEKDAY（DATE/DATETIME EXPRESSION）返回指定表达式中的当周星期几，例：
	select WEEKDAY(today) from sysmaster:sysdual;
	
	5）QUARTER（DATE/DATETIME EXPRESSION） 返回指定表达式中的季度，例：
	select quarter(today) as Q from sysmaster:sysdual;

说明：

select

to_number(substr(to_date('2022-12-31','%Y-%m-%d')-to_date('2022-01-01','%Y-%m-%d'),1,9))/7 as week1, --求当年的总周

to_number(substr(to_date('2022-12-31','%Y-%m-%d')-to_date('2022-01-01','%Y-%m-%d'),1,9))/7+1 as week2, --求当年的第几周

year(to_date('2022-01-01','%Y-%m-%d')) as cyear, --求第几年

month(to_date('2022-01-01','%Y-%m-%d')) as cmonth, --求第几月

to_number(substr(to_date('2022-12-01','%Y-%m-%d')-to_date('2022-01-01','%Y-%m-%d'),1,9))+1

```
cday, --求当年的第几天
 quarter(to_date('2022-04-01','%Y-%m-%d'))   as cquarter   --求当年的第几季度
 from sysmaster:sysdual;
```

6.1.8 EXTRACT

ORACLE	星瑞格 SinoDB
EXTRACT	无
EXTRACT()用于从 X 中提取年、月、日、时、分、秒或时区，其中 X 可以是时间戳类型或 DATE 类型。 SELECT EXTRACT(YEAR FROM DATE '2000-01-01') FROM DUAL; SELECT EXTRACT(DAY FROM DATE '2022-02-23') FROM DUAL; SELECT EXTRACT(MINUTE FROM TIME '12:00:01.35') FROM DUAL; SELECT EXTRACT(TIMEZONE_HOUR FROM TIME '12:00:01.35 +9:30') FROM DUAL; SELECT EXTRACT(TIMEZONE_MINUTE FROM TIME '12:00:01.35 +9:30') FROM DUAL; SELECT EXTRACT(SECOND FROM TIMESTAMP '2000-01-01 12:00:01.35') FROM DUAL; SELECT EXTRACT(SECOND FROM INTERVAL '-05:01:22.01' HOUR TO SECOND) FROM DUAL;	YEAR()/MONTH()/DAY() EXTENT(DATE,FORMAT) 返回指定时间段，例： select extend(current,year to day) YtoD, extend(current,hour to second) HtoM, extend(current,year to year) year , extend(current,month to month) month , extend(current,day to day) day , extend(current,second to second) second from sysmaster:sysdual;

6.1.9 NEXT_DAY

ORACLE	星瑞格 SinoDB
NEXT_DAY	NEXT_DAY
返回日期 d 后由 dow 给出的条件的第一天，dow 使用当前会话中给出的语言指定了一周中的某一天，返回的时间分量与 d 的时间分量相同。 select NEXT_DAY('01-Jan-2000','Monday') "1st Monday",NEXT_DAY('01-Nov-2004','Tuesday')+7 "2nd Tuesday") from dual;1st Monday 2nd	select today , weekday(today) cweekday, next_day(today,'TUE') cnextday, last_day(today) clastday from sysmaster:sysdual;

Tuesday03-Jan-2000 09-Nov-2004 ROUND([,]) 将日期 d 按照 fmt 指定的格式舍入，fmt 为字符串	

6.1.10 LAST_DAY

ORACLE	星瑞格 SinoDB
LAST_DAY	LAST_DAY
函数返回包含日期 d 的月份的最后一天 select sysdate,last_day(sysdate) from dual;	select today , last_day(today) clastday from sysmaster:sysdual;

6.1.11 SYSDATE

ORACLE	星瑞格 SinoDB
SYSDATE	SYSDATE
函数没有参数，返回当前日期和时间。	select sysdate from sysmaster:sysdual;

6.2 数值函数定义转换

6.2.1 ABS

ORACLE	星瑞格 SinoDB
ABS	ABS
ABS (n) 返回数字参数 n 的绝对值	ABS (n) 返回数字参数 n 的绝对值
SELECT ABS (-1) "-absolute",ABS(1) "absolute" FROM DUAL; 返回结果 1 , 1	select abs(-3) from sysmaster:sysdual;

6.2.2 ACOS

ORACLE	星瑞格 SinoDB
ACOS	ACOS
反余弦函数，返回-1 到 1 之间的数。n 表示弧度 select ACOS(-1) pi,ACOS(1) ZERO FROM dual; PI ZERO	select ACOS(-1) pi,ACOS(1) ZERO from sysmaster:sysdual;

3.14159265 0	

6.2.3 ASIN

ORACLE	星瑞格 SinoDB
ASIN	ASIN
反正弦函数，返回-1 到 1，n 表示弧度	功能一致。

6.2.4 ATAN

ORACLE	星瑞格 SinoDB
ATAN	ATAN
反正切函数，返回 n 的反正切值，n 表示弧度	功能一致。

6.2.5 ATAN2（X,Y）

ORACLE	星瑞格 SinoDB
ATAN2（X，Y）	ATAN2（X，Y）
返回坐标（X，Y）的极坐标角度组件	功能一致。

6.2.6 BITAND

ORACLE	星瑞格 SinoDB
BITAND	bitand(arg1, arg2)
SELECT bitand(3,4) FROM DUAL;	功能一致。 SELECT bitand(3,4) FROM sysmaster:sysdual;

6.2.7 CEIL/ CEILING

ORACLE	星瑞格 SinoDB
CEIL	CEIL
返回大于或等于 n 的最小整数。	功能一致。 select ceil(5.3) from sysmaster:sysdual;

6.2.8 COS

ORACLE	星瑞格 SinoDB
COS(n)	COS(n)
返回 n 的余弦值，n 为弧度	功能一致。

6.2.9 COSH

ORACLE	星瑞格 SinoDB
COSH(n）	COSH
返回 n 的双曲余弦值，n 为数字。 select COSH(1.4) FROM dual; COSH(1.4) 2.15089847	功能一致。 select COSH(1.4) c1 FROM sysmaster:sysdual; c1 2.150898465393

6.2.10 EXP

ORACLE	星瑞格 SinoDB
EXP	EXP
功能一致	功能一致。 select exp(1) c1 from sysmaster:sysdual; c1 2.718281828459

6.2.11 FLOOR

ORACLE	星瑞格 SinoDB
FLOOR(n)	FLOOR
返回小于等于 N 的最大整数。	功能一致。

6.2.12 GREATEST(exp_list)

ORACLE	星瑞格 SinoDB
GREATEST(exp_list)	GREATEST(exp_list)
exp_list 是一列表达式，返回其中最大的表达式，每个表达式都被隐含的转换第一个表达式的数据类型，如果第一个表达式是字符串数据类型中的任何一个，那么返回的结果是 varchar2 数据类型，同时使用的比较是非填充空格类型的比较。 SQL> select greatest('AA','AB','AC') from dual;	功能一致。 select greatest('AA','AB','AC') c1 from sysmaster:sysdual; c1 AC

GR -- AC	

6.2.13 LEAST(exp_list)

ORACLE	星瑞格 SinoDB
LEAST(exp_list)	LEAST(exp_list)
返回一组表达式中的最小值 SQL> select least('AA','AB','AC') from dual; LE -- AA	返回一组表达式中的最小值。 > select least('AA','AB','AC') c1 from sysmaster:sysdual; c1 AA

6.2.14 LN

ORACLE	星瑞格 SinoDB
LN	LN
返回 N 的自然对数，N 必须大于 0	功能一致。 > select ln(2.718281828459) as f_ln, logn(2.718281828459) as f_logn, log10(1000) as f_log10 from sysmaster:sysdual; f_ln f_logn f_log10 1.000000 1.000000 3.000000000000 1 row(s) retrieved.

6.2.15 LOG(n1,n2)

ORACLE	星瑞格 SinoDB
LOG(n1,n2)	ln(n2)/ln(n1)
返回以 n1 为底 n2 的对数。 SQL> select log(5,25) from dual;	select ln(25)/ln(5) c1 from sysmaster:sysdual; c1

LOG(5,25) ---------- 2	2.000000000000

6.2.16 LOG10（n）

ORACLE	星瑞格 SinoDB
LOG(10,n2)	LOG10(n)
支持 LOG(x, y)， 但不 支 持 LOG(expr)和 DLOG10(expr)。	功能一致。 Select log10(1000) as f_log10 from sysmaster:sysdual;

6.2.17 MOD

ORACLE	星瑞格 SinoDB
MOD	MOD
返回 n1 除以 n2 的余数，	功能一致。

6.2.18 POWER

ORACLE	星瑞格 SinoDB
POWER(n1,n2)	POWER(n1,n2)
返回 n1 的 n2 次方	功能一致。

6.2.19 DBMS_RANDOM

ORACLE	星瑞格 SinoDB
DBMS_RANDOM	无
Oracle 利用 DBMS_RANDOM 计算随机数， 如： 产生 N 到 M 之间的随机数 SELECT DBMS_RANDOM. VALUE(N,M) FROM DUAL; SQL> SELECT DBMS_RANDOM.VALUE(7,20) FROM DUAL; DBMS_RANDOM.VALUE(7,20) ----------	建议在程序代码中生成传入。

11.6975357	

6.2.20 ROUND(n1,n2)

ORACLE	星瑞格 SinoDB
ROUND(n1,n2)	ROUND(n1,n2)
返回舍入小数点右边 n2 位的 n1 的值，n2 的缺省值为 0，这回将小数点最接近的整数，如果 n2 为负数就舍入到小数点左边相应的位上，n2 必须是整数。 select ROUND(12345,-2),ROUND(12345.54321,2) FROM dual--ROUND(12345,-2) ROUND(12345.54321,2)12300 12345.54	功能一致。 > select round(5.1234,2) r1 from sysmaster:sysdual; r1 5.12

6.2.21 SIGN

ORACLE	星瑞格 SinoDB
SIGN	SIGN
如果 n 为负数，返回-1,如果 n 为正数，返回 1，如果 n=0 返回 0.	功能一致。 > select sign(-3) c1 from sysmaster:sysdual; c1 -1

6.2.22 SIN

ORACLE	星瑞格 SinoDB
SIN	SIN
返回 n 的正弦值,n 为弧度。	功能一致。

6.2.23 SINH

ORACLE	星瑞格 SinoDB
SINH	SINH
返回 n 的双曲正弦值,n 为弧度。	功能一致。

6.2.24 SQRT(n)

ORACLE	星瑞格 SinoDB

SQRT(n)	SQRT(n)
返回 n 的平方根,n 为弧度	功能一致。

6.2.25 TAN

ORACLE	星瑞格 SinoDB
TAN	TAN
返回 n 的正切值,n 为弧度	功能一致。

6.2.26 TANH

ORACLE	星瑞格 SinoDB
TANH	TANH
返回 n 的双曲正切值,n 为弧度	功能一致。

6.2.27 TRUNC/ TRUNCATE

ORACLE	星瑞格 SinoDB
TRUNC	TRUNC
TRUNC(n1,n2) 按照指定的精度截取一个数 SQL> select trunc(124.1666,-2) trunc1,trunc(124.16666,2) from dual; TRUNC1 TRUNC(124.16666,2) --------- ------------------ 100 124.16 返回截尾到 n2 位小数的 n1 的值，n2 缺省设置为 0，当 n2 为缺省设置时会将 n1 截尾为整数，如果 n2 为负值，就截尾在小数点左边相应的位上。	功能一致。 select trunc(124.1666,-2) tc1,trunc(124.16666,2) c2 from sysmaster:sysdual; tc1 c2 100 124.16

6.3 字符函数定义转换

6.3.1 ASCII

ORACLE	星瑞格 SinoDB
ASCII	ASCII
c1 是一字符串，返回 c1 第一个字母的 ASCII 码，他的逆函数是 CHR() SELECT ASCII('A') c1,ASCII('z') c2 FROM dual;	功能一致。 SELECT ASCII('A') c1,ASCII('z') c2 FROM sysmaster:sysdual;

BIG_A BIG_z 65 122	c1 c2 65 122

6.3.2 ASCIISTR

ORACLE	星瑞格 SinoDB
ASCIISTR	无
转换 c 为 ASCII 字符串 select asciistr('ABCDE 张三') from dual;	

6.3.3 CHR

ORACLE	星瑞格 SinoDB
CHR	CHR
CHR(＜i＞)[NCHAR_CS] i 是一个数字，函数返回十进制表示的字符 select CHR(65),CHR(122),CHR(223) FROM dual; CHR65 CHR122 CHR223 A z B	功能一致。 select chr(65) from sysmaster:sysdual;

6.3.4 CHAR_LENGTH()

	星瑞格 SinoDB
LENGTH	CHAR_LENGTH()
LENGTH() c1 为字符串，返回 c1 的长度，如果 c1 为 null，那么将返回 null 值。长度包含尾随空格。 select length('我们 ') L1,lengthb('我们') L2 from dual; L1 L2 3 6	CHAR_LENGTH() 返回字符串中逻辑字符的计数 LENGTH / LEN 返回字符列中的字节数,不包括任何尾随空格 SELECT length('我们 ') L1, OCTET_LENGTH('我们 ') L2, char_length('我们 ') L3 FROM sysmaster:sysdual; l1 l2 l3 ----- ----- ----- 6 7 3

6.3.5 CONCAT

ORACLE	星瑞格 SinoDB
CONCAT	CONCAT
CONCAT(c1,c2) c1,c2 均为字符串，函数将 c2 连接到 c1 的后面，如果 c1 为 null,将返回 c2.如果 c2 为 null,则返回 c1，如果 c1、c2 都为 null，则返回 null。他和操作符返回的结果相同 select concat('slobo','Svoboda') username from dual username sloboSyoboda	功能一致。

6.3.6 INITCAP

ORACLE	星瑞格 SinoDB
INITCAP	INITCAP
INITCAP() c1 为一字符串。函数将每个单词的第一个字母大写其它字母小写返回。单词由空格，控制字符，标点符号分隔。 select INITCAP('veni,vedi,vici') Ceasar from dual; Ceasar Veni,Vedi,Vici	功能一致。 > select INITCAP('veni,vedi,vici') c1 from sysmaster:sysdual; C1 Veni,Vedi,Vici

6.3.7 INSTR

ORACLE	星瑞格 SinoDB
INSTR	INSTR
INSTR(c1,c2,i,j) c1,c2 均为字符串，i,j 为整数。函数返回 c2 在 c1 中第 j 次出现的位置，搜索从 c1 的第 i 个字符开始。当没有发现需要的字符时返回 0,如果 i 为负数，那么搜索将从右到左进行，但是位置的计算还是从左到右，i 和 j 的缺省值为 1. select INSTR('Mississippi','i',3,3) from dual ; INSTR('MISSISSIPPI','I',3,3)	功能一致。

11	

6.3.8 INSTRB

ORACLE	星瑞格 SinoDB
INSTRB	无
与 INSTR（）函数一样，只是他返回的是字节，对于单字节 INSTRB()等于 INSTR()	

6.3.9 LCASE/ LOWER

ORACLE	星瑞格 SinoDB
LOWER	LOWER
返回 c 的小写字符，经常出现在 where 子串中 select LOWER(colorname) from itemdetail WHERE LOWER(colorname) LIKE '%white%'; Winter white	功能一致。

6.3.10 SUBSTR

ORACLE	星瑞格 SinoDB
SUBSTR	SUBSTR
SUBSTR(c1,i,j) c1 为一字符串，i,j 为整数，从 c1 的第 i 位开始返回长度为 j 的子字符串，如果 j 为空，则直到串的尾部。 select SUBSTR('Message',1,4) from dual; SUBS Mess	功能一致。

6.3.11 LENGTHB/OCTET_LENGTH ()

ORACLE	星瑞格 SinoDB
LENGTHB	OCTET_LENGTH ()
与 LENGTH()一样，返回字节。 select length('我们 ') L1,lengthb('我们') L2 from dual; L1 L2	SELECT length('我们 ') L1, OCTET_LENGTH('我们 ') L2, char_length('我们 ') L3 FROM sysmaster:sysdual;

3 6	l1 l2 l3 ----- ----- ----- 6 7 3

6.3.12 LPAD

ORACLE	星瑞格 SinoDB
LPAD	LPAD
lpad(string1, padded_length, [pad_string]) 其中 string1 是需要粘贴字符的字符串，padded_length 是返回的字符串的数量，如果这个数量比原字符串的长度要短，lpad 函数将会把字符串截取成 padded_length; pad_string 是个可选参数，这个字符串是要粘贴到 string1 的左边，如果这个参数未写，lpad 函数将会在 string1 的左边粘贴空格。 例如： lpad('tech', 7); 将返回' tech' lpad('tech', 2); 将返回'te' lpad('tech', 8, '0'); 将返回'0000tech'	功能一致。 select lpad('tech', 7) c1, lpad('tech', 2) c2, lpad('tech', 8, '0') c3 from sysmaster:sysdual; c1 c2 c3 tech te 0000tech

6.3.13 LTRIM

ORACLE	星瑞格 SinoDB
LTRIM	LTRIM
去除指定字符的左边空格或字符，可以去除多个字符 SELECT LTRIM('Hellohello world!' , 'Hello') FROM dual;	功能一致。

6.3.14 REPLACE

ORACLE	星瑞格 SinoDB
REPLACE	REPLACE
REPLACE(c1,c2,c3) c1,c2,c3 都是字符串，函数用 c3 代替出现在 c1 中的 c2 后返回。 select REPLACE('uptown','up','down') from dual;	功能一致。

6.3.15 REVERSE

ORACLE	星瑞格 SinoDB
REVERSE	REVERSE
将指定的字符串的字符排列顺序颠倒 select REVERSE('hello world!') a1 from dual;	将指定的字符串的字符排列顺序颠倒 SELECT REVERSE('hello world!') FROM systables WHERE tabid = 1; (constant) !dlrow olleh

6.3.16 RPAD

ORACLE	星瑞格 SinoDB
RPAD	RPAD
rpad(string1, padded_length, [pad_string]) 这两个函数用来格式化输出的结果。当输出结果位数少于规定的位数，使用 LPAD 函数在结果的左边添加自定义字符补齐位数，使用 RPAD 函数在结果的右边添加自定义字符补齐位数。 函数 LPAD 使用方法： LPAD (列名,位数,'添加的字符') RPAD 的使用原理是一样的。	功能一致。

6.3.17 RTRIM

ORACLE	星瑞格 SinoDB
RTTIM	RTRIM
去除指定字符右边空格或字符，可以去除多个字符。	功能一致。

6.3.18 SOUNDEX

ORACLE	星瑞格 SinoDB
SOUNDEX	无
SOUNDEX() SOUNDEX 函数返回字符串参数的语音表示形式，相对于比较一些读音相同，但是拼写不同的单词是非常有用的。	

select soundex('two'),soundex('too'),soundex('to') from dual;	

6.3.19 SUBSTR/SUBSTRING

ORACLE	星瑞格 SinoDB
SUBSTR	SUBSTR
SUBSTR(c1,i,j) c1 为一字符串，i,j 为整数，从 c1 的第 i 位开始返回长度为 j 的子字符串，如果 j 为空，则直到串的尾部。 select SUBSTR('Message',1,4) from dual; SUBS Mess	功能一致。 select substr('我们是好朋友', 1, 4) as str,substrb('我们是好朋友', 1, 4) as strb from sysmaster:sysdual; str strb 我们是好 我

6.3.20 SUBSTRB

ORACLE	星瑞格 SinoDB
SUBSTRB	SUBSTRB
SUBSTRB(c1,i,j) 与 SUBSTR 大致相同，只是 I,J 是以字节计算。 select SUBSTR('我们是好朋友',1,4) str,SUBSTRB('我们是好朋友',1,4)strb from dual; STR STRB 我们是好 我	功能一致。 select substr('我们是好朋友', 1, 4) as str,substrb('我们是好朋友', 1, 4) as strb from sysmaster:sysdual; str strb 我们是好 我

6.3.21 TRANSLATE

ORACLE	星瑞格 SinoDB
TRANSLATE	无
translate(string,from_str,to_str) 执行时,translate 依次检查 string 中的每个字符,然后查找这个字符是否在 from_str 中存在 如果不存在，那么这个 string 中的字符被保留，也就是被返回，如果存在，那么，translate 会记下这个字符在 from_str 中的位置，然后	Replace()函数多次替代。

用 to_str 的同样位置的字符代替 string 中的这个字符. select translate('abcbbaadef','ba','#@') aa from dual; AA @#c##@@def	

6.3.22 TRIM

ORACLE	星瑞格 SinoDB
TRIM	TRIM
TRIM([] from c3) 去除指定字符前后空格或去除指定字符，而且只能去除单个字符。 select TRIM(' space padded ') trim from dual; TRIM space padded	功能一致。

6.3.23 UPPER/UCASE

ORACLE	星瑞格 SinoDB
UPPER	UPPER
返回 c1 的大写，常出现 where 子串中 select name from dual where UPPER(name) LIKE 'KI%'NAMEKING	功能一致。

6.3.24 UNISTR

ORACLE	星瑞格 SinoDB
UNISTR	无
UNISTR(string) 将字符串转换为 AL16UTF16 或 UTF8 字符 select UNISTR('Hello 张三') a1 from dual;	

6.4 类型转换函数定义转换

6.4.1 CAST

ORACLE	星瑞格 SinoDB
CAST	无

CAST(c as type_name) 转换 c 为 type_name 类型[前提是能够转换才行] SELECT CAST(to_date('2013-01-01 01:01:01','YYYY-MM-DD HH24:MI:SS') AS timestamp WITH LOCAL TIME ZONE) a1 FROM DUAL; select CAST('222'as number) a2 from dual;	用 to_char()/to_number/to_date()等函数替换。

6.4.2 CONVERT

ORACLE	星瑞格 SinoDB
CONVERT	无
CONVERT(C, dset, sset) c 为字符串，dset、sset 是两个字符集，函数将字符串 c 由 sset 字符集转换为 dset 字符集，sset 的缺省设置为数据库的字符集。 SELECT CONVERT('? ¨ o ¨ a ? ? A B C DE ', 'US7ASCII', 'WE8ISO8859P1') a1 FROM DUAL;	

6.4.3 HEXTORAW

ORACLE	星瑞格 SinoDB
HEXTORAW	无
HEXTORAW(x) x 为 16 进制的字符串，函数将 16 进制的 x 转换为 RAW 数据类型。 SELECT HEXTORAW('4041424344') a1 FROM DUAL; A1 ---------- 4041424344 SELECT UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('4041424344')) a1 FROM DUAL;	

6.4.4 RAWTOHEX

ORACLE	星瑞格 SinoDB

RAWTOHEX	无
RAWTOHEX(x) x 是 RAW 数据类型字符串，函数将 RAW 数据类转换为 16 进制的数据类型。	

6.4.5 TO_BLOB

ORACLE	星瑞格 SinoDB
TO_BLOB	无
TO_BLOB(raw_val) 转换 LONG RAW 和 RAW 数据为 BLOB 类型 SELECT TO_BLOB(raw_col) blob FROM tb_name;	

6.5 杂类函数定义转换

6.5.1 DECODE

ORACLE	星瑞格 SinoDB
DECODE	DECODE
decode(expr,search1, result1, search2, result2,..., default) 检查 expr 的值 ,如果等于 search1，则返回 result1；如果等于 search2,则返回 result2，如果都没找到,则返回 default select decode(2,1,'内容为一',2,'内容为 2'),decode(2,1,'内容为一','没有条件满足') from dual;	功能一致。 select decode(2,1,'内容为一',2,'内容为 2'),decode(2,1,'内容为一','没有条件满足') from sysmaster:sysdual;

6.5.2 COALESCE

ORACLE	星瑞格 SinoDB
COALESCE	COALESCE
COALESCE(expr1, expr2...exprn) COALESCE 返回参数列表中第一个非空表达式。 必须指定最少两个参数。如果所有的参数都是 null，则返回 null。 COALESCE 函数是 NVL 函数的扩展，但	功能一致。 > select * from t1; c1 c2 1 jjj 2

是 COALESCE 函数可以使用多个值， 联合救市该函数要执行的操作； 返回参数中第一个不是 NULL 的值。如果所有参数的值都是 NULL， 则返回 NULL。	2 row(s) retrieved. > select c1,coalesce(c2,'eee','nnn') c5,coalesce(c2,null,'ppp') c6 from t1; c1 c5 c6 1 jjj jjj 2 eee ppp 2 row(s) retrieved.

6.5.3 NULLIF

ORACLE	星瑞格 SinoDB
NULLIF	NULLIF
nullif (expr1, expr2) 如果 expr1=expr2,返回 null,否则返回 expr1; 注意：该函数的第一个参数，也就是 expr1 不能是 null，如果设置为 null，会出现语法错误 SQL> select nullif(1,1) c1,nullif(1,2) c2 from dual; C1 C2 ---------- ---------- 1	功能一致。 select nullif(1,1) c1,nullif(1,2) c2 from sysmaster:sysdual; c1 c2 1

6.5.4 NVL

ORACLE	星瑞格 SinoDB
NVL	NVL
nvl(expr1, expr2) 如果 expr1 为 null,则返回 expr2,否则返回 expr1。 select	功能一致。

empno,ename,job,hiredate,(sql+nvl(comm,0))*12 年薪 from emp;	

6.6 聚合函数定义转换

注：星瑞格 SinoDB 无法嵌套聚合函数

6.6.1 AVG

ORACLE	星瑞格 SinoDB
AVG	AVG
AVG(DISTINCT|ALL) all表示对所有的值求平均值,distinct只对不同的值求平均值	功能一致。
with t as (select 1 id,20 val from dual union all select 2 id,30 val from dual union all select 3,40 from dual union all select 1,5 from dual union all select 3,15 from dual) select id,AVG(val+2) from t GROUP BY ID; ID AVG(VAL+2) 1 14.5 2 32 3 29.5	
with t as (select 1 id,20 val from dual union all select 2 id,30 val from dual union all select 3,40 from dual union all select 1,20 from dual	

union all select 3,15 from dual) select id,AVG(DISTINCT val) from t GROUP BY ID; ID AVG(DISTINCTVAL) 1 20 2 30 3 27.5	

6.6.2 COUNT

ORACLE	星瑞格 SinoDB
COUNT	COUNT
COUNT(* | [DISTINCT | ALL]表达式) 计算分区（分组）中的数据量 示例 1：查询雇员编号是 7369 的雇员姓名，职位，基本工资，部门编号，部门的人数，平均工资，最高工资，最低工资，总工资 分析：现在的程序需要进行统计查询，在学习分析函数前，这些统计查询需要在 from 子句中编写子查询后才可以使用，如果有了分析函数，则可以利用 partition 进行数据的分区，从而取得统计结果 select *from(select empno,ename,job,sal,deptno, count(empno) over (partition by deptno) count, round(avg(sal) over (partition by deptno)) avg, sum(sal) over (partition by deptno) sum, max(sal) over(partition by deptno) max, min(sal) over (partition by deptno) min from emp) temp where temp.empno=7369;	功能一致。

6.6.3 MAX

ORACLE	星瑞格 SinoDB
MAX	MAX
MAX(DISTINCT|ALL) 求最大值,ALL表示对所有的值求最大值,DISTINCT表示对不同的值求最大值,相同的只取一次 SQL> select max(distinct sal) from scott.emp; MAX(DISTINCTSAL) ---------- 5000	功能一致。
with t as (select 1 id,20 val,DATE '2014-05-03' DT from dual union all select 2 id,30 val,DATE '2014-12-03' from dual union all select 3,40,DATE '2014-11-03' from dual union all select 1,5,DATE '2014-6-03' from dual union all select 3,15,DATE '2014-10-03' from dual) select id,MAX(DT) from t GROUP BY ID; ID MAX(DT) 1 2014/6/3 2 2014/12/3 3 2014/11/3	
with t as (select 1 id,20 val,DATE '2014-05-03' DT,'b' str from dual union all	

select 2 id,30 val,DATE '2014-12-03','a' from dual union all select 3,40,DATE '2014-11-03','a' from dual union all select 1,5,DATE '2014-6-03','d' from dual union all select 3,15,DATE '2014-10-03','c' from dual) select id,MAX(str) from t GROUP BY ID; ID MAX(STR) 1 d 2 a 3 c	

6.6.4 MIN

ORACLE	星瑞格 SinoDB
MIN	MIN
求最小值,ALL表示对所有的值求最小值,DISTINCT表示对不同的值求最小值,相同的只取一次 SQL> select min(all sal) from gao.table3; MIN(ALLSAL) ---------- 1111.11	功能一致。
with t as (select 1 id,20 val,DATE '2014-05-03' DT from dual union all select 2 id,30 val,DATE '2014-12-03' from dual union all select 3,40,DATE '2014-11-03' from dual	

union all select 1,5,DATE '2014-6-03' from dual union all select 3,15,DATE '2014-10-03' from dual) select id,MIN(DT) from t GROUP BY ID; ID MIN(DT) 1 2014/5/3 2 2014/12/3 3 2014/10/3	
with t as (select 1 id,20 val,DATE '2014-05-03' DT,'b' str from dual union all select 2 id,30 val,DATE '2014-12-03','a' from dual union all select 3,40,DATE '2014-11-03','a' from dual union all select 1,5,DATE '2014-6-03','d' from dual union all select 3,15,DATE '2014-10-03','c' from dual) select id,MIN(str) from t GROUP BY ID; ID MIN(STR) 1 b 2 a 3 a	

6.6.5 STDDEV

ORACLE	星瑞格 SinoDB
STDDEV	STDDEV

STDDEV(distinct|all) 求标准差,ALL表示对所有的值求标准差,DISTINCT表示只对不同的值求标准差 SQL> select stddev(sal) from scott.emp; STDDEV(SAL) ---------- 1182.5032 SQL> select stddev(distinct sal) from scott.emp; STDDEV(DISTINCTSAL) ---------- 1229.951	功能一致。
对比说明：	

6.6.6 SUM

ORACLE	星瑞格 SinoDB
SUM	SUM
SUM([DISTINCT | ALL]表达式) 计算分区（分组）中的数据累加和	功能一致。
with t as (select 1 id,20 val from dual union all select 2 id,30 val from dual union all select 3,40 from dual union all select 1,5 from dual union all select 3,15 from dual) select id,SUM(val+2) from t GROUP BY ID; ID SUM(VAL+2) 1 29 2 32 3 59	
with t as	

(select 1 id,20 val from dual union all select 2 id,30 val from dual union all select 3,40 from dual union all select 1,20 from dual union all select 3,15 from dual) select id,SUM(DISTINCT val) from t GROUP BY ID; ID SUM(DISTINCTVAL) 1 20 2 30 3 55	

6.7 分析函数定义转换

6.7.1 RANK()

ORACLE	星瑞格 SinoDB
RANK	RANK
返回数据项在分组中的排名。RANK()在排名相等的情况下会在名次中留下空位	
--rank() 根据order by子句的排序字段，从分区（分组）查询每一行数据，按照排序生成序号，会跳跃排序，如：1，2，2，4，5 select deptno,ename,sal, rank() over (partition by deptno order by sal) rank_result, dense_rank() over(partition by deptno order by sal) dense_rank_result from emp;	功能一致。 跳跃排序，有两个第二名时接下来就是第四名 select rank() over(partition by f_deptid order by f_salary desc) as f_order, f_deptname, f_employeename, f_salary from (select a.f_employeeid, a.f_deptid, b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t;
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual	

UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,SUM(VAL) VAL,RANK() OVER (ORDER BY SUM(VAL)) RANK FROM T GROUP BY ID; ID SUM(VAL) RANK 2 25 1 1 40 2 3 160 3	
--结合NULLS FIRST /NULLS LAST WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual UNION ALL SELECT 4,null FROM dual) SELECT ID,SUM(VAL) VAL,RANK() OVER (ORDER BY SUM(VAL) nulls last) RANK FROM T GROUP BY ID; ID VAL RANK	

2 25 1 1 40 2 3 160 3 4 4	
对比说明：	

6.7.2 DENSE_RANK()

ORACLE	星瑞格 SinoDB
DENSE_RANK	DENSE _RANK
用法说明：返回数据项在分组中的排名。DENSE_RANK()在排名相等的情况下不会在名次中留下空位（这点与RANK()有明显的区别）	
dense_rank() 根据order by子句的排序字段，从分区（分组）查询每一行数据，按照排序生成序号，不会跳跃排序，如：1，2，2，3，4	功能一致。连续排序，有两个第二名时仍然跟着第三名
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,SUM(VAL) VAL,DENSE_RANK() OVER (ORDER BY SUM(VAL)) RANK FROM T GROUP BY ID; ID SUM(VAL) RANK 2 25 1 1 40 2 3 160 3	select dense_rank() over(partition by f_deptid order by f_salary desc) as f_order, f_deptname, f_employeename, f_salary from (select a.f_employeeid, a.f_deptid, b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t;
--结合NULLS FIRST /NULLS LAST	

WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual UNION ALL SELECT 4,null FROM dual) SELECT ID,SUM(VAL) VAL,DENSE_RANK() OVER (ORDER BY SUM(VAL) nulls last) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 1 1 40 2 3 160 3 4 4	
对比说明：	

6.7.3 ROW_NUMBER()

ORACLE	星瑞格 SinoDB
ROW_NUMBER	ROW_NUMBER
用法说明：返回数据项在分组中的排名。ROW_NUMBER()在排名相等的情况下,会连续编号。	
row_number() 自动生成一个行的记录号，并且不管其内容是否重复，都可以连续编号 示例1：使用此函数为行数据自动编号 SELECT deptno,ename,sal, row_number() over (partition BY deptno	ROW_NUMBER() 表示根据col1分组，在分组内部根据col2排序，而此函数计算的值就表示每组内部排序后的顺序编号（组内连续的唯一的） SELECT ROW_NUMBER() OVER(PARTITION BY

order BY sal) row_result_deptno, row_number() over (order by sal) row_result_all from emp;	pkg_type ORDER BY prod_name) AS rownum, prod_name, pkg_type FROM product;
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,SUM(VAL) VAL,ROW_NUMBER() OVER (ORDER BY SUM(VAL)) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 1 1 40 2 3 160 3	select row_number() over(partition by f_deptid order by f_salary desc) as f_order, f_deptname, f_employeename, f_salary from (select a.f_employeeid, a.f_deptid, b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t;
--结合NULLS FIRST /NULLS LAST WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL	

SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual UNION ALL SELECT 4,null FROM dual) SELECT ID,SUM(VAL) VAL,ROW_NUMBER() OVER (ORDER BY SUM(VAL) nulls last) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 1 1 40 2 3 160 3 4 4	
对比说明：星瑞格SinoDB和ORACLE都有ROW_NUMBER()函数，功能一致。	

6.7.4 PERCENT_RANK()

ORACLE	星瑞格 SinoDB
PERCENT_RANK	PERCENT_RANK
用法说明：返回某个值对于一个组的百分比排名,计算方法为 (相对位置-1)/(总行数-1),因此第一行的结果为0。返回值[0,1]。 注意对于重复行，计算时取重复行中的第一行的位置。	
--结合PARTITION BY WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,VAL,PERCNET_RANK()	PERSENT_RANK 与cume_dist的不同点在于计算分布结果的方法,计算方法为 (相对位置-1)/(总行数-1),因此第一行的结果为0。返回值(0-1) select f_deptname, f_employeename, f_salary ,PERCENT_RANK() OVER (ORDER BY f_salary) AS per_rank from (select a.f_employeeid, a.f_deptid, b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t; f_deptname Market f_employeename Kate f_salary RMB 5000.00

OVER (PARTITION BY ID ORDER BY VAL) RANK FROM T; ID VAL RANK 2 5 0 1 10 0.166666666666667 2 20 0.333333333333333 1 30 0.5 3 50 0.666666666666667 3 50 0.666666666666667 3 60 1	per_rank 0.00000000000000 f_deptname Test f_employeename Henry f_salary RMB 5000.00 per_rank 0.00000000000000 f_deptname Dev f_employeename Tom f_salary RMB 6000.00 per_rank 0.28571428571429 f_deptname Test f_employeename Bill f_salary RMB 6500.00 per_rank 0.42857142857143 .....
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,SUM(VAL) VAL,PERCENT_RANK() OVER (ORDER BY SUM(VAL)) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 0 1 40 0.5	

3 160 1	
--结合NULLS FIRST /NULLS LAST WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual UNION ALL SELECT 4,null FROM dual) SELECT ID,SUM(VAL) VAL,PERCENT_RANK() OVER (ORDER BY SUM(VAL) nulls last) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 0 1 40 0.333333333333333 3 160 0.666666666666667 4 1	

6.7.5 NTILE()

ORACLE	星瑞格 SinoDB
NTILE	NTILE
用法说明：返回N分片后的值，如三分片，四分片。	
-ntile(数字) ntile()函数对一个数据分区中的有序结果集进行划分，并为每个小组分配一个唯一的组编号 select deptno,sal, sum(sal) over(partition by deptno order	NTILE() 把记录结果集分成N部分 SELECT f_employeename, f_salary, NTILE(2) OVER (PARTITION BY f_deptid ORDER BY f_salary) as cntile FROM t_employee;

by sal) sum_result, ntile(3) over(partition by deptno order by sal) ntile_result_a, ntile(6) over(partition by deptno order by sal) ntile_result_b from emp;	f_employeename Tom f_salary RMB 6000.00 cntile 1 f_employeename Mary f_salary RMB 6600.00 cntile 1 f_employeename Jack f_salary RMB 8000.00 cntile 2 ...
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID,SUM(VAL) VAL,NTILE(2) OVER (ORDER BY SUM(VAL)) RANK FROM T GROUP BY ID; ID VAL RANK 2 25 1 1 40 1 3 160 2	
对比说明：	

6.7.6 LAG()

ORACLE	星瑞格 SinoDB
LAG	LAG

用法说明：获得位于距当前记录指定距离处的那条记录中的数据	
lag(列名称 [,行数字] [,默认值]) 访问分区（分组）中指定前N行的记录，如果没有则返回默认值，不设置默认值则返回null select deptno,empno,ename,sal, lag(sal,2,0) over(partition by deptno order by sal)lag_result, lead(sal,2,0) over(partition by deptno order by sal)lead_result from emp where deptno=20;	功能一致。 select f_month, f_quarter, f_qty, lag(f_qty) over(partition by f_quarter order by f_month) as f_lag, lead(f_qty) over(partition by f_quarter order by f_month) as f_lead from t_sale;
WITH t AS (SELECT 1 ID,'NI HAO' name FROM dual UNION ALL SELECT 2 ID,'HELLO' name FROM dual UNION ALL SELECT 2 ID,'HAPPY' name FROM dual UNION ALL SELECT 3 DI,'GOOD' name FROM dual UNION ALL SELECT 3 DI,'BAD' name FROM dual UNION ALL SELECT 3 DI,'SMILE' name FROM dual) SELECT t.*,LAG(NAME,1) OVER (ORDER BY ID,NAME) LAG FROM t; ID NAME LAG 1 NI HAO 2 HAPPY NI HAO 2 HELLO HAPPY 3 BAD HELLO 3 GOOD BAD 3 SMILE GOOD	
对比说明：	

6.7.7 LEAD()

ORACLE	星瑞格 SinoDB

LEAD	LEAD
用法说明：获得位于距当前记录指定距离处的那条记录中的数据	
lead(列名称 [,行数字] [,默认值]) 访问分区（分组）中指定后N行的记录，如果没有则返回默认值	功能一致。 select f_month, f_quarter, f_qty, lag(f_qty) over(partition by f_quarter order by f_month) as f_lag, lead(f_qty) over(partition by f_quarter order by f_month) as f_lead from t_sale;
WITH t AS (SELECT 1 ID,'NI HAO' name FROM dual UNION ALL SELECT 2 ID,'HELLO' name FROM dual UNION ALL SELECT 2 ID,'HAPPY' name FROM dual UNION ALL SELECT 3 DI,'GOOD' name FROM dual UNION ALL SELECT 3 DI,'BAD' name FROM dual UNION ALL SELECT 3 DI,'SMILE' name FROM dual) SELECT t.*,LEAD(NAME,1) OVER (ORDER BY ID,NAME) LAG FROM t; ID NAME LAG 1 NI HAO HAPPY 2 HAPPY HELLO 2 HELLO BAD 3 BAD GOOD 3 GOOD SMILE 3 SMILE	
对比说明：	

6.7.8 FIRST_VALUE 和 LAST_VALUE 函数

ORACLE	星瑞格 SinoDB

FIRST/LAST	FIRST/LAST
用法说明：获取一个分组中的第一个值或最后一个值	
WITH T AS (SELECT 1 ID,10 VAL FROM dual UNION ALL SELECT 2 ID,20 VAL FROM dual UNION ALL SELECT 1 ,30 FROM dual UNION ALL SELECT 2,5 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,50 FROM dual UNION ALL SELECT 3,60 FROM dual) SELECT ID, first_value(VAL) over (partition by ID order by VAL) first_value ,last_value(VAL) over (partition by ID order by VAL desc) last_value FROM T	功能一致。 select first_value(f_salary) over(partition by f_deptid order by f_salary desc) - f_salary as f_diff, f_deptname, f_employeename, f_salary from (select a.f_employeeid, a.f_deptid, b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t;
对比说明：星瑞格SinoDB和ORACLE功能一致。	

6.7.9 RATIO_TO_REPORT

ORACLE	星瑞格 SinoDB
RATIO_TO_REPORT	RATIO_TO_REPORT
ratio_to_report主要完成对百分比的计算，语法为：ratio_to_report(exp) over() 也就是根据over窗口函数的作用区间，求出作用区间中的单个值在整个区间的总值的比重	
ratio_to_report() 可以将需要统计的数据按照整体数据的百分比进行显示 select deptno,sum(sal), round(ratio_to_report(sum(sal)) over(),5) rate, round(ratio_to_report(sum(sal)) over(),5) * 100 || '%' precent from emp group by deptno;	RATIO_TO_REPORT() 用来计算当前记录的指标expr占开窗函数over中包含记录的所有同一指标的百分比 SELECT t.f_deptname,SUM(t.f_salary) AS SALES, RATIO_TO_REPORT(SUM(t.f_salary)) OVER() *100 AS RATIO_SALES from (select a.f_employeeid, a.f_deptid,

	b.f_deptname, a.f_employeename, a.f_salary from t_employee a, t_dept b where a.f_deptid = b.f_deptid) t group by t.f_deptname order by sales desc; f_deptname Dev sales RMB 20600.00 ratio_sales 38.4328358208955 f_deptname Test sales RMB 19000.00 ratio_sales 35.4477611940299 f_deptname Market sales RMB 14000.00 ratio_sales 26.1194029850746
对比说明：功能一致	

6.8 GROUP BY 扩展字句

- ROLLUP 是 GROUP BY 字句的一种扩展，可以为每个分组返回小计记录。
- CUBE,也是 GROUP BY 子句的一种扩展，可以返回每一个列组合的小计记录，同时在末尾加上总计记录

6.8.1 ROLLUP

ORACLE	星瑞格 SinoDB
ROLLUP	无
用法说明：ROLLUP是GROUP BY字句的一种扩展，可以为每个分组返回小计记录，并为全部分组返回总计	

create table t3(id int ,val int);
insert into t3 values(1,10);
insert into t3 values(2,20);
insert into t3 values(1,30);
insert into t3 values(2,5);
	不支持。

insert into t3 values(3,50);
insert into t3 values(3,45);
insert into t3 values(3,60);
 	
SELECT id,SUM(val) sval FROM T3
GROUP BY ROLLUP(ID);
ID SUM(VAL)
1 40
2 25
3 155
220	

对比说明：	

6.8.2 CUBE

ORACLE	星瑞格 SinoDB
CUBE	无
用法说明：返回所有列组合的小计信息，同时在最后显示汇总信息	

create table t3(id int ,val int);
insert into t3 values(1,10);
insert into t3 values(2,20);
insert into t3 values(1,30);
insert into t3 values(2,5);
insert into t3 values(3,50);
insert into t3 values(3,45);
insert into t3 values(3,60);
SELECT id,SUM(val) FROM T3 GROUP
BY CUBE(ID);
	不支持。

create table t4(id int,name varchar(30),val
int);
insert into t4 values(1,'fuzhou',30);
insert into t4 values(2,'xiamen,30);
insert into t4 values(2,'fuzhou',20);
insert into t4 values(2,'xiamen',20);
insert into t4 values(3,'fuzhou',50);
insert into t4 values(3,'fuzhou',45);
insert into t4 values(3,'xiamen',60);
SELECT
nvl(id,DECODE(name,NULL,9999,9998))	

id,nvl(name,decode(id,null,'heji','xiaoji')) name,SUM(val) FROM T4 GROUP BY CUBE(ID,name) ORDER BY id;	

6.8.3 GROUPING()

ORACLE	星瑞格 SinoDB
GROUPING	无
用法说明：可以返回一个列，返回0或1.如果列值为空，那么GROUPING()返回1，如果列值非空，则返回0.GROUPING只能在使用ROLLUP或CUBE的查询中使用，当需要在某个空值的地方限制某个值时，GORUPING是非常有用的	
WITH T AS (SELECT 1 ID,'fuzhou' name,10 VAL FROM dual UNION ALL SELECT 1 ,'fuzhou',30 FROM dual UNION ALL SELECT 1 ,'xiamen',30 FROM dual UNION ALL SELECT 2,'fuzhou',5 FROM dual UNION ALL SELECT 2 ID,'fuzhou',20 VAL FROM dual UNION ALL SELECT 2 ID,'xiamen',20 VAL FROM dual UNION ALL SELECT 3,'fuzhou',50 FROM dual UNION ALL SELECT 3,'fuzhou',45 FROM dual UNION ALL SELECT 3,'xiamen',60 FROM dual) SELECT GROUPING(ID) GROUPING,ID,SUM(VAL) VAL FROM T GROUP BY ROLLUP(ID); GROUPING ID VAL 0 1 70 0 2 45	不支持。

0 3 155 1 270	
WITH T AS (SELECT 1 ID,'fuzhou' name,10 VAL FROM dual UNION ALL SELECT 1 ,'fuzhou',30 FROM dual UNION ALL SELECT 1 ,'xiamen',30 FROM dual UNION ALL SELECT 2,'fuzhou',5 FROM dual UNION ALL SELECT 2 ID,'fuzhou',20 VAL FROM dual UNION ALL SELECT 2 ID,'xiamen',20 VAL FROM dual UNION ALL SELECT 3,'fuzhou',50 FROM dual UNION ALL SELECT 3,'fuzhou',45 FROM dual UNION ALL SELECT 3,'xiamen',60 FROM dual) SELECT GROUPING(ID) GROUPING_ID,GROUPING(NAME) GROUP_NAME,ID,NAME,SUM(VAL) VAL FROM T GROUP BY ROLLUP(ID,NAME); GROUPING_ID GROUP_NAME ID NAME VAL 0 0 1 fuzhou 40 0 0 1 xiamen 30 0 1 1 70 0 0 2 fuzhou 25 0 0 2 xiamen 20 0 1 2 45 0 0 3 fuzhou 95 0 0 3 xiamen 60	

0 1 3 155 1 1 270	
对比说明：	

6.8.4 GROUPING SET()

ORACLE	星瑞格 SinoDB
GROUPING_SET	无
用法说明：返回小计记录,不包括总计记录	
WITH T AS (SELECT 1 ID,'fuzhou' name,10 VAL FROM dual UNION ALL SELECT 1 ,'fuzhou',30 FROM dual UNION ALL SELECT 1 ,'xiamen',30 FROM dual UNION ALL SELECT 2,'fuzhou',5 FROM dual UNION ALL SELECT 2 ID,'fuzhou',20 VAL FROM dual UNION ALL SELECT 2 ID,'xiamen',20 VAL FROM dual UNION ALL SELECT 3,'fuzhou',50 FROM dual UNION ALL SELECT 3,'fuzhou',45 FROM dual UNION ALL SELECT 3,'xiamen',60 FROM dual) SELECT ID,NAME,SUM(VAL) VAL FROM T GROUP BY GROUPING SETS(ID,NAME); ID NAME VAL 1 70 2 45 3 155 xiamen 110 fuzhou 160	不支持。
对比说明：	

6.8.5 GROUPING_ID

ORACLE	星瑞格 SinoDB
GROUPING_ID	无
用法说明：可以借助HAVING字句对记录进行过滤，返回分组记录中的小计和总计部分，GROUPING_ID函数可以接受一列或多列，返回GROUPING位向量的十进制值，	
WITH T AS (SELECT 1 ID,'fuzhou' name,10 VAL FROM dual UNION ALL SELECT 1 ,'fuzhou',30 FROM dual UNION ALL SELECT 1 ,'xiamen',30 FROM dual UNION ALL SELECT 2,'fuzhou',5 FROM dual UNION ALL SELECT 2 ID,'fuzhou',20 VAL FROM dual UNION ALL SELECT 2 ID,'xiamen',20 VAL FROM dual UNION ALL SELECT 3,'fuzhou',50 FROM dual UNION ALL SELECT 3,'fuzhou',45 FROM dual UNION ALL SELECT 3,'xiamen',60 FROM dual) SELECT GROUPING_ID(ID,NAME) GROUPING_ID,ID,NAME,SUM(VAL) VAL FROM T GROUP BY CUBE(ID,NAME) HAVING GROUPING_ID(ID,NAME) > 0;	不支持。
对比说明：	

7. 其他对象转换

7.1 ROWID 伪列

ORACLE	星瑞格 SinoDB
ROWID	ROWID
Oracle的ROWID用来唯一标识表中的一条	也有rowid。可以根据每张表对应的partnum

记录，是这条数据在数据库中存放的物理地址. Select rowid,t1.* from test_rowid t1; ROWID ID AAAc4AAAZAAEjJ2AAA 1 AAAc4AAAZAAEjJ2AAB 2 AAAc4AAAZAAEjJ2AAC 3 AAAc4AAAZAAEjJ2AAD 4 AAAc4AAAZAAEjJ2AAE 5 AAAc4AAAZAAEjJ2AAF 6	（相当于物理地址），再加上rowid（相当于行偏移量）定位到某一行数据。fragment分片表会有rowid重复的情况，但其实分片表会有多个partnum，其中某一个partnum加上对应的rowid，还是可以定位到某一行的数据。

7.2 ROWNUM

ORACLE	星瑞格 SinoDB
ROWNUM	无

rownum是根据sql查询出的结果给每行分配一个逻辑编号,SQL的不同会导致你的rownum不同.
Drop table test_rowid;
Create table test_rowid
(rowid_id number(10)
);
insert into test_rowid values(6);
insert into test_rowid values(2);
insert into test_rowid values(3);
insert into test_rowid values(4);
insert into test_rowid values(5);
insert into test_rowid values(1);
select rownum,t1.* from test_rowid t1;
ROWNUM ROWID_ID
1 6
2 1
3 2
4 3
	自定义函数实现。
	CREATE FUNCTION rownum () returning int as rownum;
	define global counter int default 0;
	let counter = counter + 1;
	return counter;
	end function;
	
	CREATE PROCEDURE init_rownum ();
	define global counter int default 0;
	let counter = 0;
	end procedure;
	
	call init_rownum (); --每次执行rownum()前，需要对其进行初始化。
	
	select rownum() from sysmaster:sysdual

5 4
6 5
	或者考虑用row_num()函数实现
	drop table t1;
	create table t1(c1 int,c2 int);
	insert into t1 values(1,5);
	insert into t1 values(2,4);
	insert into t1 values(2,3);
	insert into t1 values(3,3);
	insert into t1 values(4,2);
	insert into t1 values(5,1);
	
	select row_number() over(order by c1 asc) as r1,c1,c2,row_number() over(order by c2 asc) as r2 from t1;

7.3 CONNECT BY 递归查询

ORACLE	星瑞格 SinoDB
CONNECT BY	CONNECT BY
SQL 用法：递归查询用于查询父子借点结构的数据表，形成树状结构的数据集，常用于菜单数据集、报表结构数据。	
DROP TABLE test_table; CREATE TABLE test_table(root_id number, sub_id number, name varchar(5), description varchar(10)); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(0,1,'a','parent_a'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(1,2,'a1','sub_a2'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(1,3,'a2','sub_a3'); INSERT INTO test_table(root_id,sub_id,name,description)	DROP TABLE test_table; CREATE TABLE test_table(root_id int, sub_id int, name varchar(5), description varchar(10)); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(0,1,'a','parent_a'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(1,2,'a1','sub_a2'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(1,3,'a2','sub_a3'); INSERT INTO test_table(root_id,sub_id,name,description)

VALUES(3,9,'a2','sub_a9'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(0,4,'b','parent_b'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(4,5,'b1','sub_b1'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(4,6,'b2','sub_b2'); select * from test_table start with root_id = 0 connect by prior sub_id = root_id; ROOT_ID SUB_ID NAME DESCRIPTIO ---------- ---------- ----- ---------- 0 1 a parent_a 1 2 a1 sub_a2 1 3 a2 sub_a3 3 9 a2 sub_a9 0 4 b parent_b 4 5 b1 sub_b1 4 6 b2 sub_b2	VALUES(3,9,'a2','sub_a9'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(0,4,'b','parent_b'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(4,5,'b1','sub_b1'); INSERT INTO test_table(root_id,sub_id,name,description) VALUES(4,6,'b2','sub_b2'); select * from test_table start with root_id = 0 connect by prior sub_id = root_id; root_id sub_id name description 0 4 b parent_b 4 6 b2 sub_b2 4 5 b1 sub_b1 0 1 a parent_a 1 3 a2 sub_a3 3 9 a2 sub_a9 1 2 a1 sub_a2

7.4 UNION ALL

ORACLE	星瑞格 SinoDB
UNION ALL	UNION ALL
用法说明：返回各个查询检索出来的所有行，包含重复行	

drop table t5;
create table t5(id int,name varchar(30));
insert into t5 values(1,'lily');
insert into t5 values(1,'pepper');
 	
drop table t6;
create table t6(id int,name varchar(30));
insert into t6 values(3,'mary');
insert into t6 values(1,'lily');
	功能一致。
	drop table t5;
	create table t5(id int,name varchar(30));
	insert into t5 values(1,'lily');
	insert into t5 values(1,'pepper');
	
	drop table t6;
	create table t6(id int,name varchar(30));
	insert into t6 values(3,'mary');

select id,name FROM t5 Union all select id,name FROM t6; ID NAME ---------- ------------------------------ 1 lily 1 pepper 3 mary 1 lily	insert into t6 values(1,'lily'); select id,name FROM t5 Union all select id,name FROM t6; id name 1 lily 1 pepper 3 mary 1 lily
对比说明：星瑞格SinoDB和ORACLE 功能一致,	

7.5 UNION

ORACLE	星瑞格 SinoDB
UNION	UNION
用法说明：返回各个查询检索出来的所有行，不包含重复行	

drop table t5;
create table t5(id int,name varchar(30));
insert into t5 values(1,'lily');
insert into t5 values(1,'pepper');
 	
drop table t6;
create table t6(id int,name varchar(30));
insert into t6 values(3,'mary');
insert into t6 values(1,'lily');
 	
select id,name FROM t5
Union
select id,name FROM t6;
 	
ID NAME
---------- ------------------------------
1 lily
1 pepper
3 mary
	功能一致。
	drop table t5;
	create table t5(id int,name varchar(30));
	insert into t5 values(1,'lily');
	insert into t5 values(1,'pepper');
	
	drop table t6;
	create table t6(id int,name varchar(30));
	insert into t6 values(3,'mary');
	insert into t6 values(1,'lily');
	
	select id,name FROM t5
	Union all
	select id,name FROM t6;
	id name
	1 lily
	1 pepper
	3 mary

对比说明：星瑞格SinoDB和ORACLE 功能一致,	

7.6 INTERSECT

ORACLE	星瑞格 SinoDB
INTERSECT	INTERSECT
用法说明：返回各个查询检索出来的共有行	

drop table t5;
create table t5(id int,name varchar(30));
insert into t5 values(1,'lily');
insert into t5 values(1,'pepper');
 	
drop table t6;
create table t6(id int,name varchar(30));
insert into t6 values(3,'mary');
insert into t6 values(1,'lily');
 	
select id,name FROM t5
INTERSECT
select id,name FROM t6;
 	
ID NAME
---------- ------------------------------
1 lily
	功能一致。
	drop table t5;
	create table t5(id int,name varchar(30));
	insert into t5 values(1,'lily');
	insert into t5 values(1,'pepper');
	
	drop table t6;
	create table t6(id int,name varchar(30));
	insert into t6 values(3,'mary');
	insert into t6 values(1,'lily');
	
	select id,name FROM t5
	INTERSECT
	select id,name FROM t6;
	
	id name
	1 lily

对比说明：星瑞格SinoDB和ORACLE 功能一致.	

7.7 MINUS

ORACLE	星瑞格 SinoDB
MINUS	MINUS
用法说明：返回将第二个查询出来的行从第一个查询检索出来的行中减去之后剩余的行	

drop table t5;
create table t5(id int,name varchar(30));
insert into t5 values(1,'lily');
insert into t5 values(1,'pepper');
	功能一致。
	drop table t5;
	create table t5(id int,name varchar(30));
	insert into t5 values(1,'lily');

drop table t6;
create table t6(id int,name varchar(30));
insert into t6 values(3,'mary');
insert into t6 values(1,'lily');
 	
select id,name FROM t5
Minus
select id,name FROM t6;
 	
ID NAME
---------- ----------
1 pepper
	insert into t5 values(1,'pepper');
	
	drop table t6;
	create table t6(id int,name varchar(30));
	insert into t6 values(3,'mary');
	insert into t6 values(1,'lily');
	
	select id,name FROM t5
	Minus
	select id,name FROM t6;
	
	id name
	1 pepper

对比说明：星瑞格SinoDB和ORACLE 功能一致。	

7.8 PIVOT

ORACLE	星瑞格 SinoDB
PIVOT	无
用法说明：使用交叉表报	

drop table t1;
create table t1(dt varchar(12),name varchar(20),val int);
insert into t1 values('2018-01-01' ,'chen' , 11);
insert into t1 values('2018-01-01' ,'zhen' , 22);
insert into t1 values('2018-01-01' ,'li' , 44);
insert into t1 values('2018-01-01' ,'wang' , 55);
insert into t1 values('2020-01-01' ,'chen' , 3442);
insert into t1 values('2020-01-01' ,'zhen' , 1231);
insert into t1 values('2020-01-01' ,'li' , 4441);
insert into t1 values('2020-01-01' ,'wang' , 1233);
	create table test_rowcols(
	name char(10),
	course char(10),
	score int,
	primary key(name,course)
	);
	
	insert into test_rowcols(name , course,score)
	values('张三','语文',74);
	insert into test_rowcols(name , course,score)
	values('张三','数学',99);
	insert into test_rowcols(name , course,score)
	values('张三','物理',85);
	insert into test_rowcols(name , course,score)
	values('李四','语文',78);
	insert into test_rowcols(name , course,score)
	values('李四','数学',94);
	insert into test_rowcols(name , course,score)

select * from t1 pivot (sum(val) for name in ('chen','li','zhen','wang')) ; DT 'chen' 'li' 'zhen' 'wang' ---------- ---------- ---------- ---------- ---------- 2018-01-01 11 44 22 55 2020-01-01 3442 4441 1231 1233	values('李四','物理',86); select name,max(case when course='语文' then score else 0 end) as yuwen , max(case when course='数学' then score else 0 end) as shuxue , max(case when course='物理' then score else 0 end) as wuli from test_rowcols group by name; name yuwen shuxue wuli 张三 74 99 85 李四 78 94 86
对比说明：星瑞格SinoDB无此函数。	

7.9 分页

ORACLE	星瑞格 SinoDB
select * from customer where rownum <= 100;	select first 100 * from customer;
select * from customer where rownum > 100 and rownum <= 200;	select skip 100 first 100 * from customer;
SELECT * FROM (SELECT A.*,ROWNUM AS RN FROM (SELECT * FROM customer order by col) A WHERE ROWNUM <= 200) T WHERE T.RN > 100	select skip 100 first 100 * from customer order by col; 说明：FIRST n 取结果集中的前n行； SKIP n 忽略结果集中的前n行，为（第几页－1）＊每页记录数。skip,first后面只能跟数字，不能数字计算和函数。

8. 查询对照

8.1 子查询

8.1.1 单行子查询

8.1.1.1 在 WHERE 子句中使用子查询

ORACLE	星瑞格 SinoDB
用法说明：子查询可以在另外一个查询的WHERE子句中 WHERE 字段 IN/=/>/</>=/<= (子查询)	

drop table t1;
create table t1(id int ,name varchar(30));
insert into t1 values(1,'a');
insert into t1 values(2,'b');
insert into t1 values(3,'c');
 	
drop table t2;
create table t2(id int ,name varchar(30));
insert into t2 values(1,'a');
 	
SELECT * FROM t1 WHERE id = (SELECT id FROM T2);
	功能一致。

SELECT * FROM t1 WHERE id >(SELECT id FROM T2);	
SELECT * FROM t1 WHERE id <(SELECT id FROM T2);	
对比说明：功能一致	

8.1.1.2 在 HAVING 子句中使用子查询

ORACLE	星瑞格 SinoDB
用法说明：HAVING 字段 {>/</>=/<=/=}（子查询）	

drop table t3;
create table t3(id int ,val int);
insert into t3 values(1,10);
	SELECT ID,AVG(VAL) VAL
	FROM t3
	GROUP BY id

insert into t3 values(2,20);
insert into t3 values(1,30);
insert into t3 values(2,5);
insert into t3 values(3,50);
insert into t3 values(3,45);
insert into t3 values(3,60);
 	
SELECT ID,AVG(VAL) VAL
FROM t3
GROUP BY id
HAVING AVG(VAL)<(SELECT
MAX(AVG(VAL)) FROM t3 group by ID);
	HAVING AVG(VAL)<(SELECT
	MAX(AVG(VAL)) FROM t3 group by ID);
	--SinoDB的聚集函数不能嵌套，会报错
	544: Cannot have aggregates within
	aggregates.
	--需要把语句改成以下
	SELECT ID,AVG(VAL) VAL
	FROM t3
	GROUP BY id
	HAVING AVG(VAL)<(select max(ta.c1)
	from (SELECT AVG(VAL) c1 FROM t3
	group by ID) ta);

8.1.1.3 在 FROM 语句中使用子查询

ORACLE	星瑞格 SinoDB
用法说明：FROM（子查询）	

drop table t3;
create table t3(id int ,val int);
insert into t3 values(1,10);
insert into t3 values(2,20);
insert into t3 values(1,30);
insert into t3 values(2,5);
insert into t3 values(3,50);
insert into t3 values(3,45);
insert into t3 values(3,60);
SELECT ID FROM (
SELECT * FROM T3 WHERE VAL > 20);
	功能一致。

对比说明：功能一致	

8.1.2 编写多行子查询

ORACLE	星瑞格 SinoDB
用法说明：FROM　字段 IN（子查询）	

drop table t1;
create table t1(id int ,name varchar(30));
insert into t1 values(1,'a');
insert into t1 values(2,'b');
insert into t1 values(3,'c');
	功能一致。

drop table t2;
create table t2(id int ,name varchar(30));
insert into t2 values(1,'a');
 	
SELECT * FROM t1 WHERE id IN
(SELECT id FROM T2);	

对比说明：功能一致	

8.1.3 编写多列子查询

ORACLE	星瑞格 SinoDB

drop table t1;
create table t1(id int ,name varchar(30));
insert into t1 values(1,'a');
insert into t1 values(2,'b');
insert into t1 values(3,'c');
 	
drop table t2;
create table t2(id int ,name varchar(30));
insert into t2 values(1,'a');
 	
SELECT * FROM t1 WHERE (id,name) IN
(SELECT id,name FROM T2);
	不支持。得把列拆分写。
	SELECT * FROM t1 WHERE id IN
	(SELECT id FROM T2) and
	name IN (SELECT name FROM T2);

8.1.4 编写嵌套子查询

ORACLE	星瑞格 SinoDB

drop table t3;
create table t3(id int ,val int);
insert into t3 values(1,10);
insert into t3 values(2,20);
insert into t3 values(1,30);
insert into t3 values(2,5);
insert into t3 values(3,50);
insert into t3 values(3,45);
insert into t3 values(3,60);
 	
SELECT ID,AVG(VAL)
	drop table t3;
	create table t3(id int ,val int);
	insert into t3 values(1,10);
	insert into t3 values(2,20);
	insert into t3 values(1,30);
	insert into t3 values(2,5);
	insert into t3 values(3,50);
	insert into t3 values(3,45);
	insert into t3 values(3,60);
	
	SELECT ID,AVG(VAL)

FROM T3 GROUP BY ID HAVING AVG(VAL) < (SELECT AVG(VAL) FROM T3 WHERE ID IN (SELECT ID FROM T3 WHERE VAL >= 30));	FROM T3 GROUP BY ID HAVING AVG(VAL) < (SELECT AVG(VAL) FROM T3 WHERE ID IN (SELECT ID FROM T3 WHERE VAL >= 30));
对比说明：功能一致	

8.1.5 使用标量子查询

ORACLE	星瑞格 SinoDB

drop table a;
create table a (id int,name varchar2(10));
create table b (id int,name varchar2(10));
insert into a values (1,'a1');
insert into a values (2,'a2');
insert into b values (1,'b1');
insert into b values (2,'b2');
 	
select a.*,(select name from b where b.id=a.id) from a;
	drop table a;
	create table a (id int,name varchar(10));
	create table b (id int,name varchar(10));
	insert into a values (1,'a1');
	insert into a values (2,'a2');
	insert into b values (1,'b1');
	insert into b values (2,'b2');
	
	select a.*,(select name from b where b.id=a.id) from a;

对比说明：功能一致	

8.1.6 编写包含子查询的 UPDATE

ORACLE	星瑞格 SinoDB

drop table t3;
create table t3(id int ,val int);
insert into t3 values(1,20);
insert into t3 values(2,30);
insert into t3 values(3,40);
insert into t3 values(1,50);
insert into t3 values(3,15);
 	
UPDATE T3 SET VAL =(SELECT AVG(VAL) FROM T3 WHERE ID = 1)
WHERE ID = 1 AND VAL = 20;
 	
SELECT * FROM T3;
	支持。
	drop table t3;
	create table t3(id int ,val int);
	insert into t3 values(1,20);
	insert into t3 values(2,30);
	insert into t3 values(3,40);
	insert into t3 values(1,50);
	insert into t3 values(3,15);
	
	UPDATE T3 SET VAL =(SELECT AVG(VAL) FROM T3 WHERE ID = 1)
	WHERE ID = 1 AND VAL = 20;

	SELECT * FROM T3;
对比说明：功能一致	

8.1.7 编写包含子查询的 DELETE

ORACLE	星瑞格 SinoDB

drop table t3;
create table t3(id int ,val int);
insert into t3 values(1,20);
insert into t3 values(2,30);
insert into t3 values(3,40);
insert into t3 values(1,50);
insert into t3 values(3,15);
 	
DELETE FROM T3 WHERE VAL <
(SELECT AVG(VAL) FROM T3);
 	
SELECT * FROM t3;
	支持。
	drop table t3;
	create table t3(id int ,val int);
	insert into t3 values(1,20);
	insert into t3 values(2,30);
	insert into t3 values(3,40);
	insert into t3 values(1,50);
	insert into t3 values(3,15);
	
	DELETE FROM T3 WHERE VAL <
	(SELECT AVG(VAL) FROM T3);
	
	SELECT * FROM t3;

对比说明：功能一致	

8.2 MERGE

ORACLE	星瑞格 SinoDB
SQL用法：对于已经存在的记录进行更新，对于新的记录插入	

DROP TABLE test_table;
CREATE TABLE test_table(
root_id number(10) PRIMARY KEY,
sub_id number(10)
);
INSERT INTO test_table(root_id,sub_id)
VALUES(0,1);
 	
MERGE INTO test_table t1
USING (SELECT 1 ROOT_ID,2 SUB_ID
FROM DUAL) t2
on(t1.root_id = t2.root_id)
WHEN matched THEN
update set sub_id = 5
	一致。
	drop table if exists t_user1;
	create table t_user1(f_userid int, f_username
	varchar(20), f_age int);
	insert into t_user1 values(1, 'Tom', 28);
	insert into t_user1 values(2, 'Jack', 26);
	insert into t_user1 values(4, 'Rose', 18);
	
	drop table if exists t_user2;
	create table t_user2(f_userid int, f_username
	varchar(20), f_age int);
	insert into t_user2 values(3, 'Jim', 25);
	insert into t_user2 values(4, 'Rose', 23);
	insert into t_user2 values(5, 'Mike', 21);

WHEN not matched THEN insert values(1,2)；	insert into t_user2 values(6, 'Bill', 19); merge into t_user1 a using t_user2 b on a.f_userid = b.f_userid when matched then update set a.f_age = b.f_age when not matched then insert (a.f_userid, a.f_username, a.f_age) values(b.f_userid, b.f_username, b.f_age);

8.3 WITH

ORACLE	星瑞格 SinoDB
SQL用法：with 表名 as ( )Select * from 表名 用于定义虚拟表，方便多层嵌套语句的可读性	

With t as
(SELECT 1 id,'des' name FROM DUAL
UNION ALL
SELECT 2 id,'des1' FROM DUAL
)
SELECT * FROM t;
	更新版本后支持。
	Drop table t1;
	create table t1(c1 int, c2 int);
	
	insert into t1 values(1,1);
	insert into t1 values(2,2);
	insert into t1 values(3,3);
	
	with a1 as (select * from t1)
	select * from a1;

8.4 表连接

8.4.1 自连接

ORACLE	星瑞格 SinoDB
用法说明：表自身的连接	
drop table if exists t_dept; create table t_dept(f_deptid int, f_deptname	drop table if exists t_dept; create table t_dept(f_deptid int, f_deptname

varchar(10), f_parentid int);
insert into t_dept values(0, 'MS', -1);
insert into t_dept values(1, 'Dev', 0);
insert into t_dept values(2, 'Test', 1);
insert into t_dept values(3, 'Market', 0);
insert into t_dept values(4, 'HR', 0);
 	
select a.*, b.f_deptname as f_parentname from t_dept a, t_dept b where a.f_parentid = b.f_deptid;
	varchar(10), f_parentid int);
	insert into t_dept values(0, 'MS', -1);
	insert into t_dept values(1, 'Dev', 0);
	insert into t_dept values(2, 'Test', 1);
	insert into t_dept values(3, 'Market', 0);
	insert into t_dept values(4, 'HR', 0);
	
	select a.*, b.f_deptname as f_parentname from t_dept a, t_dept b where a.f_parentid = b.f_deptid;

对比说明：功能一致	

8.4.2 内连接

ORACLE	星瑞格 SinoDB
用法说明：内连接只返回两个表中连接字段相等的行。	

drop table t_dept;
create table t_dept(f_deptid int, f_deptname varchar(10), f_parentid int);
insert into t_dept values(0, 'MS', -1);
insert into t_dept values(1, 'Dev', 0);
insert into t_dept values(2, 'Test', 1);
insert into t_dept values(3, 'Market', 0);
 	
drop table t_employee;
create table t_employee(f_employeeid int, f_deptid int, f_employeename varchar2(10), f_salary number);
insert into t_employee values(1, 1, 'Tom', 6000.00);
insert into t_employee values(2, 1, 'Jack', 8000.00);
insert into t_employee values(3, 1, 'Mary', 6600.00);
insert into t_employee values(4, 2, 'Henry', 5000.00);
insert into t_employee values(6, 5, 'Bill',
	drop table if exists t_dept;
	create table t_dept(f_deptid int, f_deptname varchar(10), f_parentid int);
	insert into t_dept values(0, 'MS', -1);
	insert into t_dept values(1, 'Dev', 0);
	insert into t_dept values(2, 'Test', 1);
	insert into t_dept values(3, 'Market', 0);
	
	drop table if exists t_employee;
	create table t_employee(f_employeeid int, f_deptid int, f_employeename varchar(10), f_salary money);
	insert into t_employee values(1, 1, 'Tom', 6000.00);
	insert into t_employee values(2, 1, 'Jack', 8000.00);
	insert into t_employee values(3, 1, 'Mary', 6600.00);
	insert into t_employee values(4, 2, 'Henry', 5000.00);

6500.00); select a.f_employeeid, a.f_employeename, b.f_deptname from t_employee a inner join t_dept b on a.f_deptid = b.f_deptid;	select a.f_employeeid, a.f_employeename, b.f_deptname from t_employee a inner join t_dept b on a.f_deptid = b.f_deptid;
对比说明：功能一致	

8.4.3 左连接

ORACLE	星瑞格 SinoDB
用法说明：左连接返回包括左表中的所有记录和右表中连接字段相等的记录	

select a.f_employeeid, a.f_employeename,
b.f_deptname
from t_employee a
left outer join t_dept b
on a.f_deptid = b.f_deptid;
 	
select a.f_employeeid, a.f_employeename,
b.f_deptname
from t_employee a, t_dept b
where a.f_deptid = b.f_deptid(+);
 	
F_EMPLOYEEID F_EMPLOYEE
F_DEPTNAME
---------- ---------- ----------
1 Tom Dev
2 Jack Dev
3 Mary Dev
4 Henry Test
6 Bill
	select a.f_employeeid, a.f_employeename,
	b.f_deptname
	from t_employee a
	left outer join t_dept b
	on a.f_deptid = b.f_deptid;
	
	f_employeeid f_employeename f_deptname
	
	1 Tom Dev
	2 Jack Dev
	3 Mary Dev
	4 Henry Test
	6 Bill

对比说明：SinoDB不支持（+）写法。	

8.4.4 右连接

ORACLE	星瑞格 SinoDB
用法说明：右连接返回包括右表中的所有记录和左表中连接字段相等的记录；	
select a.f_employeeid, a.f_employeename,	select a.f_employeeid, a.f_employeename,

b.f_deptname
from t_employee a
right outer join t_dept b
on a.f_deptid = b.f_deptid;
 	
select a.f_employeeid, a.f_employeename,
b.f_deptname
from t_employee a,t_dept b
where a.f_deptid(+) = b.f_deptid;
 	
F_EMPLOYEEID F_EMPLOYEE
F_DEPTNAME
---------- ---------- ----------
1 Tom Dev
2 Jack Dev
3 Mary Dev
4 Henry Test
Market
MS
	b.f_deptname
	from t_employee a
	right outer join t_dept b
	on a.f_deptid = b.f_deptid;
	
	f_employeeid f_employeename f_deptname
	
	MS
	1 Tom Dev
	2 Jack Dev
	3 Mary Dev
	4 Henry Test
	Market

对比说明：SinoDB不支持（+）写法。	

8.4.5 全连接

ORACLE	星瑞格 SinoDB
用法说明：右连接返回包括右表中的所有记录和左表中连接字段相等的记录；	
select a.f_employeeid, a.f_employeename, b.f_deptname from t_employee a full outer join t_dept b on a.f_deptid = b.f_deptid; F_EMPLOYEEID F_EMPLOYEE F_DEPTNAME ---------- ---------- ---------- 1 Tom Dev 2 Jack Dev 3 Mary Dev	select a.f_employeeid, a.f_employeename, b.f_deptname from t_employee a full outer join t_dept b on a.f_deptid = b.f_deptid; f_employeeid f_employeename f_deptname 1 Tom Dev 2 Jack Dev 3 Mary Dev 4 Henry Test

4 Henry Test 6 Bill Market MS	6 Bill MS Market
对比说明：功能一致	

9. 自定义类型

ORACLE	星瑞格 SinoDB
CREATE TYPE name_type AS OBJECT (first_name VARCHAR2(25), middle_initial CHAR(1), last_name VARCHAR2(30));	CREATE ROW TYPE name_type (first_name VARCHAR(25), middle_initial CHAR(1), last_name VARCHAR(30));

10. 序列，索引，同义词和视图

10.1 序列

10.1.1 创建序列

ORACLE	星瑞格 SinoDB
语法： CREATE SEQUENCE sequence_name [START WITH num] [INCREMENT BY increment] [MAXVALUE num|NOMAXVALUE] [MINVALUE num|NOMINVALUE] [CYCLE|NOCYCLE] [CACHE num|NOCACHE] [ORDER|NOORDER] [KEEP|NOKEEP]	Oracle的maxvalue支持29个9，Sinodb支持18个。 drop sequence seq1; create sequence seq1 minvalue 1 maxvalue 999999999999999999 start with 1 increment by 1 nocache ; 改变序列：

[SESSION|GLOBAL]
 	
● sequence_name指明序列名。
● START WITH定义起始序列值。如果未定义START WITH 子句值，递增序列将使用MINVALUE，递减序列将使用MAXVALUE，如果没有定义MINVALUE,MAXVALUE,则默认为1。
定义序列可生成的最大数值，必须为0 至4611686018427387903 之间。
● INCREMENT BY <increment_value>
<increment_value> ::= <signed_integer>
定义上个分配值递增到下一个序列值的量，默认值为1。指定一个负的值来生成一个递减的序列。INCREMENT BY值为0，将返回错误。
● MINVALUE <min_value>序列能够生成的最小整数，min_value必须在-4611686018427387904 and 4611686018427387902之间，且min_value必须小于或等于start_value,而且min_value必须小于max_value
● NO MINVALUE 当使用no minvalue指令的时候，递增序列的最小值为1，递减序列的最小值为-4611686018427387903.
● MAXVALUE <max_value>
<max_value> ::= <signed_integer>
序列生成的最大值，<max_value>必须在-4611686018427387903 and 4611686018427387902之间.
max_value必须大于等于start_value,
	Alter sequence seq1 restart with 5 increment by 2 maxvalue 500;
	
	重命名序列：
	Rename sequence seq1 to seq2;
	
	删除序列：
	Drop sequence seq2;
	
	使用序列：
	drop table t1;
	create table t1(c1 int,c2 int);
	insert into t1 values(seq1.nextval,seq1.nextval);
	insert into t1 values(seq1.currval,2);
	select * from t1;
	
	要注意的是：
	①第一次nextval返回的是初始值；随后的nextval会自动增加你定义的increment by值，然后返回增加后的值。
	curval总是返回当前sequence的值，但是在第一次nextval初始化之后才能使用currval，否则会出错。
	一次nextval会增加一次sequence的值。但是如果你在同一个条SQL语句里面针对同一个sequence使用多次nextval，其值都是一样的。
	②如果指定cache值，就可以预先在内存里面放置一些sequence，这样存取的快些。cache里面的取完后，自动再取一组到cache。使用cache或许会跳号，比如数据库突然不正常down掉（shutdown abort），cache中的sequence就会丢失.所以可以在create sequence的时候用nocache防止这种情况。

且max_value必须大于min_value ● NO MAXVALUE 使用NO MAXVALUE 指令时，递增序列的最大值将为4611686018427387903，递减序列的最大值为-1。 ● CYCLE 指定该序列如果已经达到设置的最大值或最小值的时候，仍然能够生成整数。当升序序列达到最大值时，下一个生成的序列值为最小值；当降序序列达到最小值时，下一个生成的序列值为最大值。 ● NO CYCLE 跟CYCLE是相对的，指如果序列已经达到最大值或者最小值时，将不再生成整数了,即不重新开始。NOCYCLE为默认选线 ● CACHE <cache_size>指定要保留在内存中的整数的个数，cache_size的类型必须为unsigned integer ● NO CACHE 当NO CACHE指令被使用个，序列将不在被缓存，为默认选项 ● RESET BY <subquery> 数据库重启期间，，系统自动执行RESET BY 语句，并且将用RESET BY 子查询确定的值重启序列。如果未指定RESET BY，序列值将持久地存储在数据库中。在数据库重启过程中，序列的下一个值将由已保存的序列值生成。	
DROP SEQUENCE seq_text; create sequence seq_text minvalue 1 maxvalue 99999999999 start with 1 increment by 1 cache 20;	

10.1.2 使用序列

ORACLE	星瑞格 SinoDB
直接调用CURRVAL,看ORACLE的以下运行结果: SELECT SEQ_TEST.CURRVAL FROM DUAL; 都会产生报错	直接调用CURRVAL，星瑞格SinoDB也会产生报错。

SELECT SEQ_TEST.NEXTVAL FROM DUAL;
SELECT SEQ_TEST.CURRVAL FROM DUAL;
SELECT SEQ_TEST.NEXTVAL,SEQ_TEST.CURRVAL FROM DUAL;
	drop table t1;
	create table t1(c1 int,c2 int);
	insert into t1 values(seq1.nextval,seq1.nextval);
	insert into t1 values(seq1.currval,2);
	select * from t1;

10.1.3 修改序列

ORACLE	星瑞格 SinoDB
语法： ALTER SEQUENCE sequence_name [START WITH num] [INCREMENT BY increment] [MAXVALUE num|NOMAXVALUE] [MINVALUE num|NOMINVALUE] [CYCLE|NOCYCLE] [CACHE num|NOCACHE] [ORDER|NOORDER] [KEEP|NOKEEP] [SESSION|GLOBAL]	Alter sequence seq1 restart with 5 increment by 2 maxvalue 500;

10.1.4 删除序列

ORACLE	星瑞格 SinoDB
语法： DROP SEQUENCE sequence_name;	DROP SEQUENCE sequence_name;

10.2 索引

星瑞格 SinoDB 支持 Oracle 的大多数索引特性。 这些特性如 UNIQUE 索引、 函数索引、 全局和分区索引、 B-树索引、 升序/降序索引方式。 但是， 星瑞格 SinoDB 不支持 Oracle 的 bitmap 和 Domain 索引。

下面分别给出 星瑞格 SinoDB 和 Oracle 在索引实现方面的主要相同点和异同点。

10.2.1 聚簇索引

ORACLE	星瑞格 SinoDB
Oracle 也支持类似的功能，语法格式如下所示： CREATE CLUSTER …	聚集索引表示表中存储的数据按照索引的顺序存储，检索效率比非聚集索引高，但对数据更新影响较大。 create cluster index idx_emp_id on t_employee(f_employeeid);

10.2.2 函数索引

ORACLE	星瑞格 SinoDB
create index idx_emp_name on t_employee(upper(f_employeename));	create index idx_emp_name on t_employee(upper(f_employeename)); 9844: Invalid function (upper) used in a functional key. --如果使用内置函数定义函数索引会报错，必须进行修改，如下所示： CREATE FUNCTION toUpper(name VARCHAR(100)) RETURNS VARCHAR(100) WITH (NOT VARIANT); RETURN upper(name); END FUNCTION; create index idx_emp_name on t_employee

	(toUpper(f_employeename)); SELECT * FROM t_employee WHERE toUpper(f_employeename) LIKE 'BI%L';
对比说明：SinoDB 的函数索引不能是内置的代数、指数、对数或十六进制函数。如果需要使用内置函数定义函数索引，那么必须从 SQL 或外部语言函数中调用该函数。不能针对返回大对象的 UDR 创建函数索引。不允许使用大对象作为索引键，因为一般情况下不能够对大对象进行比较和排序。然而，需要注意，可以将大对象作为参数传递给 UDR。如果将某个 UDR 用于函数索引，则该 UDR 不能使用集合数据类型作为参数类型。集合数据类型包括 SET、MULTISET 和 LIST。	

10.2.3 索引禁用

ORACLE	星瑞格 SinoDB
ALTER INDEX name UNUSABLE;	set indexes idx_name disabled; --禁用 set indexes idx_name enabled; --启用

10.2.4 重命名索引

ORACLE	星瑞格 SinoDB
ALTER INDEX name RENAME TO new_name;	语法： rename index <old_index_name> to <new_index_name>; 示例： rename index idx_emp_name to idx_e1;

10.2.5 全局索引和分区索引

全局索引和分区索引都是有关水平分区的索引。 其中， 全局索引是指在所有分区表上创建的索引。

本质上， 它和非分区表上的普通索引没有区别。 而分区索引是指在单个分区上创建的索引。

ORACLE	星瑞格 SinoDB
CREATE TABLE testgla (id INT, name CHAR(10), birthdate DATE,age INT)	drop table tuser; create table tuser(

PARTITION BY LIST (id) (PARTITION tgal1 VALUES (10), PARTITION tgal2 VALUES (20, 30), PARTITION tgal3 VALUES (40)); --全局索引 CREATE INDEX i_testgl_name_g_1 on testgla(name) GLOBAL; --分区索引 CREATE INDEX i_testgl_birthdate_l_1 on testgla(birthdate) LOCAL	id int, fdeptid int, fusername varchar(10), falary money) partition by expression partition part1 mod(id,3)=0 in datadbs1, partition part2 mod(id,3)=1 in datadbs2, partition part3 mod(id,3)=2 in datadbs3; --全局索引 Drop index idx_tuser_id; Create index idx_tuser_id on tuser(id) in datadbs1; --分区索引 Create index idx_tuser_id on tuser(id) partition by expression partition part1 mod(id,3)=0 in datadbs1, partition part2 mod(id,3)=1 in datadbs2, partition part3 mod(id,3)=2 in datadbs3;
对比说明：在 SinoDB 数据库中如果不想对索引分区，需要指定希望整个索引位于的那个 dbspace。	

10.3 同义词

在实际应用中， 用户通常会为远程对象创建一个简单易记的同义词标识，

然后使用该标识， 从本地数据库中透明地访问远程数据库对象。 这种方式既可保证访问安全， 又可简化冗长的远程对象标识。

依据所属特性， Oracle 的同义词可分为：

公有同义词： 当创建同义词使用 PUBLIC 关键字时， 则创建的同义词是公有同义词， 其他用

户都可访问。

私有同义词： 如果不指定 PUBLIC 关键字， 则同义词会创建到"$user"模式下， 此时的同义词

是私有同义词， 其他用户应该使用“模式名.同义词名”来引用这个同义词。

ORACLE	星瑞格 SinoDB

CREATE TABLE emp(empno INT, ename VARCHAR(20));
INSERT INTO emp VALUES(1, 'Tom');
CREATE OR REPLACE SYNONYM my_syn FOR emp;
SELECT * FROM my_syn;
TEST=# SELECT * FROM my_syn;
EMPNO | ENAME
-------+-------
1 | Tom
(1 行)
	drop table if exists t_user;
	create table t_user(f_userid int, f_username varchar(20));
	drop synonym if exists syn_user;
	create synonym syn_user for t_user;
	insert into syn_user values(1,'sinodb');
	select * from t_user;
	select * from syn_user; drop synonym syn_user;

10.4 视图

10.4.1 常规视图

在视图上， 星瑞格 SinoDB 和 Oracle 是非常相似的。 Oracle 的大多数 CREATE VIEW 语句都无需改动就可在星瑞格 SinoDB 上运行， 并支持对视图的更新 update 操作。

ORACLE	星瑞格 SinoDB
语法： create or replace view_name as select a.col1,null as col2 from table_name a where 条件	功能一致。 create view v_tuser as select * from tuser; select * from v_tuser; drop view v_tuser;
创建具有 CHECK OPTION 约束的视图 CREATE TABLE VIEW_TABLE (A INT PRIMARY KEY, B INT); CREATE OR REPLACE VIEW TEST_VIEW1 as select * from VIEW_TABLE where A >5 with check option constraint CHECK_A;	不支持
创建只读试图： DROP TABLE VIEW_TABLE; DROP VIEW TEST_VIEW;	不支持

CREATE TABLE VIEW_TABLE (A INT PRIMARY KEY, B INT); CREATE VIEW TEST_VIEW AS SELECT * FROM VIEW_TABLE with read only	

10.4.2 物化视图

物化视图（Materialized View）在 ORACLE9i 以前的版本叫做快照（SNAPSHOT)，从 9i 开始改名叫做物化视图。它是用于预先计算并保存表连接或聚集等耗时较多的操作的结果，这样，在执行查询时，就可以避免进行这些耗时的操作，从而快速的得到结果。物化视图有很多方面和索引很相似：使用物化视图的目的是为了提高查询性能；物化视图对应用透明，增加和删除物化视图不会影响应用程序中 SQL 语句的正确性和有效性；物化视图需要占用存储空间；当基表发生变化时，物化视图也应当刷新。

其中物化视图有三种：聚集物化视图、包含连接物化视图、嵌套物化视图。但三种物化视图的快速刷新的限制条件有很大区别，而其他方面则区别不大。

ORACLE	星瑞格 SinoDB
create materialized view MVKPI1 refresh force on demand as select to_date(Ayear * 100 + Amonth, 'YYYYMM') whenDate, Ayear, Amonth, Sho_ID, Bra_ID, Sum(FEEDMONEY) FEEDMONEY, Sum(ORDERMONEY) ORDERMONEY, Sum(INTIMEFEEDMONEY) INTIMEFEEDMONEY, Sum(INTIMEORDERMONEY) INTIMEORDERMONEY, Sum(DELIVERMONEY) DELIVERMONEY, Sum(SALEMONEY) SALEMONEY, Sum(PROFITMONEY) PROFITMONEY,	不支持。

Sum(NEWSTOCKMONEY) NEWSTOCKMONEY, Sum(STOCKMONEY) STOCKMONEY, Sum(SALE7MONEY) SALE7MONEY, Sum(NEWGOODSMONEY) NEWGOODSMONEY, Sum(BESTGOODSMONEY) BESTGOODSMONEY, Sum(DeliverDays * DeliverMoney) DeliverDaysMoney from KPI group by Ayear, Amonth, Sho_ID, Bra_ID;	

11. 匿名块

ORACLE	星瑞格 SinoDB

匿名块是能够动态地创建和执行过程代码的 PL/SQL 结构，而不需要以持久化的方式将代码作为数据库对象储存在系统目录中。
DECLARE
TYPE r_cursor IS REF CURSOR;
c_emp r_cursor;
TYPE rec_emp is RECORD
(ename VARCHAR2(20),
sal NUMBER(6));
er rec_emp;
BEGIN
OPEN c_emp FOR SELECT ename,sal FROM emps;
LOOP
FETCH c_emp INTO er;
EXIT WHEN c_emp%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(er.ename || ' - ' || er.sal);
END LOOP;
CLOSE c_emp;
END
	不支持。

set serveroutput on;
declare
v_name varchar2(20);
v_hiredate date:=sysdate;
begin
v_name:='Mark';
dbms_output.put_line('name:'||v_name||
',hiredate:'||v_hiredate);
end;
/	

12. 存储过程

12.1 创建存储过程

ORACLE	星瑞格 SinoDB

create or replace procedure proc_test is
v_id number(3);
v_divided number(3);
v_result number(6,2);
begin
v_id := 11;
v_divided := 2;
v_result := v_id / v_divided;
end;
	语法：
	create procedure proc_name(param1 data_type1, param2 data_type2, ...)
	spl_code;
	end procedure;
	示例：
	create procedure Pro_loop1()
	define i int;
	let i = 1;
	loop
	if i > 10 then
	exit;
	else
	insert into t_user values(i, concat('user_loop1_', to_char(i)));
	end if;
	
	let i = i + 1;
	end loop;
	end procedure;

12.2 修改存储过程

ORACLE	星瑞格 SinoDB
ALTER PROCEDURE [schema.] procedure COMPILE [DEBUG] [compiler_parameters_clause ... [REUSE SETTINGS];	先 drop procedure ，再重新创建 create procedure。

12.3 调用存储过程

ORACLE	星瑞格 SinoDB
[CALL] procName [(parameters)]; [EXEC] procName [(parameters)]; BEGIN procName [(parameters)] END;	call pro_area_add(1, 'Beijing'); execute procedure pro_area_add(1, 'Beijing');

13. 函数

ORACLE	星瑞格 SinoDB
在调用函数时， Oracle 在无参数时， 对函数后面的括号可加可不加。 在传入参数时不能定义其类型的长度，其传入与传出参数的长度由系统自动处理。	语法： create function [if not exists] <function_name>(param1 data_type1, param2 data_type2, ...) returning data_type1 [as var1] [, data_type2 [as var2], ...] <spl code> return var1[, var2, ...]; end function;

//传入和传出字符型参数不定义长度
TEST-# create or replace function p_fun(name1
varchar) return varchar
TEST-# is
TEST-# begin
TEST-# return name1;
TEST-# end;
	示例：
	create function fn_get_ymd(dt date)
	returning int as year, int as month, int as day
	define y,m,d int;
	let y = year(dt);
	let m = month(dt);
	let d = day(dt);
	return y,m,d;

TEST-# / CREATE FUNCTION TEST-# call p_fun('abcd'); TEST-# / ERROR: 类型 VARCHAR(1)的值过长 CONTEXT: PL/SQL 函数"p_fun" 保存调用参数到本地变量	end function; execute function fn_get_ymd(today); drop function fn_get_ymd;

//传入字符型参数定义长度
TEST=# create or replace function
p_fun(name1 varchar(20)) return varchar
TEST-# is
TEST-# begin
TEST-# return name1;
TEST-# end;
TEST-# /
CREATE FUNCTION
TEST=# call p_fun('abcdf');
TEST-# /
P_
FUN
------- 	
abcdf
(1 行)	

//传入和传出数值型
TEST=# create or replace function p_fun(id
int,sal number) return number
TEST-# is
TEST-# begin
TEST-# return id*sal;
TEST-# end;
TEST-# /
CREATE FUNCTION
TEST=# call p_fun(10,11.99);
TEST-# /
P_
FUN
------- 		

120 (1 行	

14. 包

Oracle 中可以通过 Package Body 来对多个 procedure 进行整合，并且共用 package head，但是在 SinoDB 中并不支持 Package Body 的方式，所以需要对其中所有包含的 procedure/function 配合对应的 head 变量声明拆写成独立的 procedure 和 function。

Oracle 中的 Package 是一系列方法、过程、变量的集合。SinoDB 中没有与之直接对应的概念，但是有一些方法可以实现近似的功能。

在 Oracle 中,程序可以通过语法 package_name.procedure_name 访问一个包。如果在 SinoDB 创建一个用户或 SCHEMA 与 Oracle 的 Package 名一样,那么这将是更容易迁移过程的调用。因为语法实际上是一样的，虽然意思有所改变。使用这种做法,调用一个迁移过程将是 owner.procedure_name （owner 将和 Package 的名字是一样的）。将 Package 名字转换为 owner 的名字是一个维持 Package 的方法。

15. 触发器

触发器是一种特殊类型的存储过程， 用户不能直接调用。 但是， 当插入、删除或修改表或视图数据等特定事件发生时， 它将被自动触发执行。

SinoDB 的触发器与 Oracle 的最大不同之处在于，SinoDB 的一个触发器只能触发一个事件，而 Oracle 的一个触发器可以触发多个事件。所以在迁移过程中，Oracle 的一个触发器常会被拆分成多个 SinoDB 触发器。

Oracle 允许 SQL 语句,逻辑语句和存储过程,在触发器中调用。SinoDB 只允许 SQL 语句和存储过程在触发器内调用。所以，Oracle 触发器包含的逻辑语句必须转换为 SinoDB 的存储过程，然后存储过程由触发执行。

	ORACLE	星瑞格 SinoDB
变量	:NEW :OLD	NEW OLD

INSTEAD OF 触发器	支持	支持 CREATE TRIGGER update_manager_info_update INSTEAD OF UPDATE ON manager_info_view REFERENCING NEW AS n FOR EACH ROW (EXECUTE PROCEDURE update_m_p (n.empno, n.empname, n.deptno)); CREATE PROCEDURE instab (dno INT, eno INT) INSERT INTO dept(deptno, manager_num) VALUES(dno, eno); INSERT INTO emp (empno, deptno) VALUES (eno, dno); END PROCEDURE;
WHEN 触发器	支持	支持
WHERE 触发器	支持	支持
AS 关键字	不需要 AS 关键字	需要 AS 关键字
编译	支持 COMPILE	支持 COMPILE
INSERTING UPDATING DELETING	支持 INSERTING、 UPDATING、 DELETING	支持 INSERTING、 UPDATING、 DELETING
触发事件	支持 DDL 事件和 DATABASE 事件	支持 DML/DQL 事件。
声明新老值的别名	REFERENCING OLD AS "OLD" NEW AS "NEW"	referencing old as old_item referencing new as new_item
触发器名称	创建 trigger 时触发器名称前可以添加模式名 CREATE OR REPLACE TRIGGER	创建 trigger 时触发器名称前可以添加所属用户名称模 Create trigger testdb:informix.trg_sale_insert

	"GJXFJ08"."XFSXXX_PINDEX"	

	CREATE TRIGGER
	insert_salary_changes
	BEFORE UPDATE ON tab1
	FOR EACH ROW
	DECLARE
	valuediff INT;
	BEGIN
	valuediff := :new.col - :old.col;
	INSERT INTO tab2
	VALUES(valuediff);
	END insert_salary_changes;
		CREATE TRIGGER
		update_unit_price
		UPDATE OF unit_price
		ON stock
		REFERENCING OLD AS
		pre NEW AS post
		FOR EACH ROW WHEN
		(post.unit_price > pre.unit_price
		* 2)
		(INSERT INTO
		warn_table
		VALUES(pre.stock_num,
		pre.manu_code,
		pre.unit_price,
		post.unit_price, CURRENT));
		这个示例在一个名为
		update_unit_price 的触发器，在
		stock 表上创建了一个 update
		trigger，当列 unit_price 更新时
		触发器将执行动作。
		当这个触发器执行动作时，触
		发器将为每一个限定行执行动
		作(影响触发器事件及执行的
		条
		件是：“ post.unit_price >
		pre.unit_price * 2 ”为 true)。

	CREATE OR REPLACE
	TRIGGER
	"GJXFJ08"."XFSXXX_PINDEX"
	BEFORE INSERT ON
	"GJXFJ08"."XFSXXX"
	REFERENCING OLD AS "OLD"
	NEW AS "NEW" FOR EACH
	ROW
	ENABLE WHEN (old.pindex is
	null)
	begin
		触发器存储过程示例：
		CREATE PROCEDURE
		items_trigger_proc()
		REFERENCING
		OLD AS pre NEW AS post
		FOR items;
		if (SELECTING) then
		insert
		into log_records values (1,
		pre.quantity, -1);
		end if

	select xfsxxx_pindex_seq.nextva
	linto :new.pindex
	from dual;
	end;
		if (INSERTING) then
		insert
		into log_records values (2, -1,
		post.quantity);
		end if
		if (DELETING) then
		insert
		into log_records values (3,
		pre.quantity, -1);
		end if
		if (UPDATING) then
		insert
		into log_records values (4,
		pre.quantity, post.quantity);
		end if
		END PROCEDURE;
		
		CREATE TRIGGER
		update_quantity
		UPDATE OF quantity ON
		items
		FOR EACH ROW
		(EXECUTE
		PROCEDURE
		items_trigger_proc()
		WITH TRIGGER
		REFERENCES);

16. DBLINK

ORACLE	星瑞格 SinoDB
CREATE DATABASE LINK dbtarget CONNECT TO targetuser IDENTIFIED BY targetpwd USING ‘targetentry’; SELECT * FROM my_schema.my_table@dbtarget;	--另一个db在同一台机器时 SELECT name, number FROM dbname:tablename; --另一个db在其他server时 SELECT name, number FROM dbname@distantservenamer:tablename;

	--ANSI模式下时 SELECT name, number FROM dbname@aservername:ownername.tablename;
对比说明：Oracle 中，可以通过 DBLINK 来访问其他库的对象，SinoDB并不支持 DBLINK 方式， 但可以通过外部数据库的书写方式进行访问。在 SinoDB中访问其他库对象时，需要确保拥有足够的权限，并且两个库的locale 必须相同。	

附录 1：分析函数相关表

```
drop table if exists t_dept;
create table t_dept(f_deptid int, f_deptname varchar(50));
insert into t_dept values(1, 'Dev');
insert into t_dept values(2, 'Test');
insert into t_dept values(3, 'Market');

drop table if exists t_employee;
create table t_employee(f_employeeid int, f_deptid int, f_employeename varchar(50), f_salary money);
insert into t_employee values(1, 1, 'Tom', 6600.00);
insert into t_employee values(2, 1, 'Jack', 8000.00);
insert into t_employee values(3, 1, 'Mary', 6600.00);
insert into t_employee values(4, 2, 'Henry', 6500.00);
insert into t_employee values(5, 2, 'Rose', 7500.00);
insert into t_employee values(6, 2, 'Bill', 6500.00);
insert into t_employee values(7, 3, 'Kate', 5000.00);
insert into t_employee values(8, 3, 'Bob', 9000.00);
insert into t_employee values(9, 1, 'Will', 5000.00);
insert into t_employee values(10, 2, 'Judy', 5000.00);
```