**HANA SQL参考及Oracle对照**

**By hujue**

在本部分中，主要参考了《SAP\_HANA数据库SQL参考手册》，以及《ORACLE sql参考手册》，针对我们日常使用的sql方式、内置函数、关键字，进行了对比

1. 常用SQL写法
   1. 递归查询

|  |  |
| --- | --- |
| SQL用法：  递归查询用于查询父子节点结构的数据表，形成树状结构的数据集，常用于菜单数据集、报表结构数据 | |
| ORACLE | HANA |
| **select** \* **from**  tf\_rpt\_hn\_stat\_bshzqk t  **where** t.dept\_id='10262200'  start **with** t.dept\_id='10000000'  **connect** **by** **prior** t.dept\_id=t.dept\_mng\_id  **order** siblings **by** t.detp\_id | **select** \* **from** "tf\_rpt\_hn\_stat\_bshzqk" t  **where** t.dept\_id='10262200'  start **with** t.dept\_id='10000000'  **connect** **by** **prior** t.dept\_id=t.dept\_mng\_id  **order** **by** t.detp\_id |
| 对比说明：  Oracle与hana的start with…connect by prior语句是一致的，条件语句也同样是先递归查询，后where筛选  除了以上，HANA中没有level、siblings关键字，无法像oracle有额外扩展功能 | |

* 1. 竖转横查询

|  |  |
| --- | --- |
| SQL用法：  竖转横查询用于将数据表中竖列的数据转换为横式进行显示 | |
| ORACLE | HANA |
| **select** t.issue,  t.dept\_id,  t.obj\_add\_id,  **max**(decode(t.zbdm,'FDCYXQK\_0001',t.zbz,**null**)) A,  **max**(decode(t.zbdm,'FDCYXQK\_0002',t.zbz,**null**)) B,  **max**(decode(t.zbdm,'FDCYXQK\_0003',t.zbz,**null**)) C  **from** tf\_rpt\_item\_data\_fdsc t  **group** **by** t.issue,t.dept\_id,t.obj\_add\_id | **select** t.issue,  t.dept\_id,  t.obj\_add\_id,  **max**(**map**(t.zbdm,'FDCYXQK\_0001',t.zbz,**null**)) A,  **max**(**map**(t.zbdm,'FDCYXQK\_0002',t.zbz,**null**)) B,  **max**(**map**(t.zbdm,'FDCYXQK\_0003',t.zbz,**null**)) C  **from** "tf\_rpt\_item\_data\_fdsc" t  **group** **by** t.issue,t.dept\_id,t.obj\_add\_id |
| 对比说明：  在HANA中，没有decode函数，与之功能相同的是map函数 | |

* 1. nullif

|  |  |
| --- | --- |
| SQL用法：  一般用于设置被除数条件，当被除数为0时，设置结果为null | |
| ORACLE | HANA |
| **select** 100/**nullif**(0,0)  **from** dual t | **select** 100/**nullif**(0,0)  **from** DUMMY |
| 对比说明：  在ORACLE与HANA中，nullif功能相同，不同的是HANA没有dual表，只有DUMMY表 | |

* 1. nvl（oracle），ifnull（hana）

|  |  |
| --- | --- |
| SQL用法：  当参数值为null时，返回指定的value | |
| ORACLE | HANA |
| **select** nvl(aaa,0)  **from** tab t | **select** nvl(aaa,0)  **from** tab t |
| 对比说明：  hana中为ifnull；oracle中为nvl | |

* 1. delete

|  |  |
| --- | --- |
| SQL用法：  delete数据 | |
| ORACLE | HANA |
| delete from t1 a where exists (select 1 from t2 b where a.id = b.id ) | delete from t1 a where exists (select 1 from t2 b where a.id = b.id ) |
| 对比说明：  HANA的DELETE语句与ORACLE类似，FROM子句只支持1个表(即需要从中删除记录的表)。 如果某个表需要删除的记录集合与其他表相关，那么需要在WHERE条件中实现关联。 | |

* 1. with子句

|  |  |
| --- | --- |
| SQL用法：  用于定义虚拟表，方便多层嵌套语句的可读性 | |
| ORACLE | HANA |
| with bb as(  select t.\* from a t)  select \* from bb | **with**子句写法语法有误  替代写法  **select** \* **from** (**select** \* **from** "STS"."TF\_RBAC\_DEPARTMENT") bb |
| 对比说明：  在HANA中，不支持with子句功能 | |

* 1. update表

|  |  |
| --- | --- |
| SQL用法：  update数据表中的数据 | |
| ORACLE | HANA |
| **update** "STS"."TF\_RBAC\_DEPARTMENT" t  **set** (addr,sort)=('11111',100) | **update** "STS"."TF\_RBAC\_DEPARTMENT" t  **set** (addr,sort)=('11111',100) |
| 对比说明：  update数据表语句一致 | |

* 1. 多表关联update

|  |  |
| --- | --- |
| SQL用法：  update数据表中的数据 | |
| ORACLE | HANA |
| update TF\_RPT\_AJ\_M\_JZJCSJB t  set (t.tcrq,t.zjycdxsj)=(select t1.tbbssj,t1.tjsj  from TF\_RPT\_HN\_STAT\_BSHZQK t1  where t.issue=t1.issue and t.dept\_id=t1.dept\_id and t1.rpt\_id='HAY01')  where exists (select 1  from TF\_RPT\_HN\_STAT\_BSHZQK t1  where t.issue=t1.issue and t.dept\_id=t1.dept\_id and t1.rpt\_id='HAY01'  ) | **update** "TB"."TF\_RPT\_AJ\_M\_JZJCSJB" t  **set** (t.tcrq,t.zjycdxsj)=(t1.tbbssj,t1.tjsj)  **from** "TB"."TF\_RPT\_HN\_STAT\_BSHZQK" t1  **where** t.issue=t1.issue  **and** t.dept\_id=t1.dept\_id  **and** t1.rpt\_id='HAY01' |
| 对比说明：  在hana中，如果某个表需要更新的记录集合与其他表相关，那么需要在FROM子句中指定相关的表，在WHERE子句中加入关联条件；  而对于oracle，多表关联update复杂一些，oracle不支持from子句，即不支持update from语法 | |

* 1. 插入更新

|  |  |
| --- | --- |
| SQL用法：  对于已经存在的记录进行更新，对于新的记录插入 | |
| ORACLE | HANA |
| merge into | **UPSERT** T(**key**,val) **VALUES** (1, 9) **WHERE** **KEY** = 1;  **UPSERT** T(**key**,val) **VALUES** (1, 8) **WITH** **PRIMARY** **KEY**; |
| 对比说明：  Oracle的merge into从9i就存在，可以对存在的记录进行更新，对于新的记录插入，避免proc中条件判断的低效率，相关信息可以自行baidu；  Hana的upsert可以对存在的记录进行更新，对于新的记录插入，在列出的字段中，必须包含所有的主键字段，需要注意的是，upsert不是标准的sql语句，在各个数据库中不同 | |

* 1. 多表关联插入更新

|  |  |
| --- | --- |
| SQL用法：  关联其他数据表数据，对于已经存在的记录进行更新，对于新的记录插入 | |
| ORACLE | HANA |
| merge into | **UPSERT** "TB"."TF\_RPT\_AJ\_M\_JZJCSJB" (issue,dept\_id,jz\_id,tcrq,zjycdxsj)  **select** t1.issue,t1.dept\_id,t.jz\_id,t1.tbbssj,t1.tjsj  **from** "TB"."TF\_RPT\_HN\_STAT\_BSHZQK" t1,TF\_RPT\_AJ\_M\_JZJCSJB t  **where** t.issue=t1.issue  **and** t.dept\_id=t1.dept\_id  **and** t1.rpt\_id='HAY01' |
| 对比说明：  Oracle的merge into从9i就存在，可以对存在的记录进行更新，对于新的记录插入，避免proc中条件判断的低效率，相关信息可以自行baidu；  Hana的upsert可以对存在的记录进行更新，对于新的记录插入，在列出的字段中，必须包含所有的主键字段，多表关联时，比较类似insert语法  需要注意的是，upsert不是标准的sql语句，在各个数据库中不同 | |

* 1. like搜索

|  |  |
| --- | --- |
| SQL用法：  用like进行模糊搜索 | |
| ORACLE | HANA |
| **select** \* **from** "STS"."tf\_rpt\_hn\_stat\_bshzqk" t  **where** t.dept\_id **like** '100\_' | **select** \* **from** "STS"."tf\_rpt\_hn\_stat\_bshzqk" t  **where** t.dept\_id **like** '100\_' |
| 对比说明：  like数据表语句一致，可以用%或是\_进行匹配 | |

* 1. case when

|  |  |
| --- | --- |
| SQL用法：  在sql中使用if…then…else逻辑 | |
| ORACLE | HANA |
| select t.jzlb,  case when t.jzlb='火电' then 'HD'  when t.jzlb='水电' then 'SD'  ELSE 'UNKOWN' END  from tf\_rpt\_dept\_jzxx t | **select** t.jzlb,  **case** **when** t.jzlb='火电' **then** 'HD'  **when** t.jzlb='水电' **then** 'SD'  **ELSE** 'UNKOWN' **END**  **from** "STS"."tf\_rpt\_dept\_jzxx" t |
| 对比说明：  case when使用同oracle一样， | |

* 1. 多表关联

|  |  |
| --- | --- |
| SQL用法：  2个及以上数据表关联查询 | |
| ORACLE | HANA |
| select \*  from tf\_rpt\_dept\_jzxx t,tf\_rpt\_hn\_stat\_bshzqk t1  where t.dcdm=t1.dept\_id | **select** \*  **from** "tf\_rpt\_dept\_jzxx" t,"tf\_rpt\_hn\_stat\_bshzqk" t1  **where** t.dcdm=t1.dept\_id |
| 对比说明：  写法一样 | |

* 1. 外关联

|  |  |
| --- | --- |
| SQL用法：  外连接查询 | |
| ORACLE | HANA |
| select \*  from tf\_rpt\_dept\_jzxx t  left outer join tf\_rpt\_hn\_stat\_bshzqk t1  on t.dcdm=t1.dept\_id  where t1.issue='320131100' | **select** \*  **from** "tf\_rpt\_dept\_jzxx" t  **left** **outer** **join** "tf\_rpt\_hn\_stat\_bshzqk" t1  **on** t.dcdm=t1.dept\_id  **where** t1.issue='320131100' |
| 对比说明：  写法一样，另外说明Oracle有一种(+)的外关联写法，不推荐这种方式 | |

* 1. 当前时间1天后

|  |  |
| --- | --- |
| SQL用法：  外连接查询 | |
| ORACLE | HANA |
| select sysdate+1  from dual t | **select** ADD\_DAYS(**CURRENT\_TIMESTAMP**,1) **from** dummy |
| 对比说明：  在hana中，日期时间不能加减number，只能通过日期时间函数 | |

当前时间1个月后

|  |  |
| --- | --- |
| SQL用法：  外连接查询 | |
| ORACLE | HANA |
| select add\_months(sysdate,1)  from dual t | **select** ADD\_MONTHS(**CURRENT\_TIMESTAMP**,1) **from** dummy |
| 对比说明：  在hana中，日期时间不能加减number，只能通过日期时间函数 | |

* 1. 返回数据集的前几个记录

|  |  |
| --- | --- |
| SQL用法：  用于返回sql查询数据集的前几个记录 | |
| ORACLE | HANA |
| select \* from dual t  where rownum<=10 and rownum>=5 | **select** \* **from** "TB"."TF\_RPT\_HN\_STAT\_BSHZQK" t  **where** t.issue='2013-12'  **limit** 10 offset 5 |
| 对比说明：  在oracle中，是通过其内置的rownum字段进行记录集的截取；在hana中，是通过limit子句进行 | |

* 1. 全球唯一标识符

|  |  |
| --- | --- |
| SQL用法：  返回全球唯一的标识符，常用于写主键值 | |
| ORACLE | HANA |
| select sys\_guid() from dual; | **select** SYSUUID **from** dummy; |
| 对比说明：  Hana与oracle中都是通过函数实现返回全球唯一的标识符，生成一个16字节的原始值，每个字节2个16进制显示位，即可认为是32位长度的字符串，  该标识符主要是用于数据表主键字段的取值，避免seq的序列值，同时对数据表数据的合并时很有用。 | |

* 1. 计算累计partition by

|  |  |
| --- | --- |
| SQL用法：  计算累计 | |
| ORACLE | HANA |
| select t.issue,t.zbz,sum(zbz) over (partition by substr(t.issue,1,5) order by t.issue)  from tf\_rpt\_item\_data\_fdsc t  where substr(t.issue,1,5)='32013'  and t.zbdm='FDCYXQK\_0004'  and t.dept\_id='10272700'  and t.obj\_add\_id='1' | **select** t.issue,t.zbz,**sum**(zbz) over (**partition** **by** **substr**(t.issue,1,5) **order** **by** t.issue)  **from** tf\_rpt\_item\_data\_fdsc t  **where** **substr**(t.issue,1,5)='32013'  **and** t.zbdm='FDCYXQK\_0004'  **and** t.dept\_id='10272700'  **and** t.obj\_add\_id='1' |
| 对比说明：  Hana与oracle的计算累计sql是一致的，同时其他的聚合函数，如max、min、avg等都支持partition by | |

1. 数据类型对比
   1. 日期时间类型
      1. DATE

|  |  |
| --- | --- |
| ORACLE | HANA |
| **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” | 时间值的范围从0001-01-01 至9999-12-31  DATE 数据类型由年、月、日信息组成，表示一个日期值,最小时间粒度为日  DATA 类型的默认格式为‘YYYY-MM-DD’。YYYY 表示年，MM 表示月而DD 表示日。 |
| 对比说明：  虽然在Oracle和Hana中都有DATE数据类型，但是需要注意的是在HANA中，date只表示日期，即数据精度为日，这一点与Oracle中的date有很大不同 | |

* + 1. TIME

|  |  |
| --- | --- |
| ORACLE | HANA |
| **无** | **TIME** |
|  | TIME 数据类型由小时、分钟、秒信息组成，表示一个时间值。  TIME 类型的默认格式为‘HH24:MI:SS’。HH24 表示从0 至24 的小时数，MI 代表0 至59 的分钟值而SS 表示0 至59的秒 |
| 对比说明：  HANA中的time类型表示时间，而Oracle中没有对应的类型 | |

* + 1. SECONDDATE

|  |  |
| --- | --- |
| ORACLE | HANA |
| **DATE** | **SECONDDATE** |
|  | TIME 数据类型由小时、分钟、秒信息组成，表示一个时间值。  TIME 类型的默认格式为‘HH24:MI:SS’。HH24 表示从0 至24 的小时数，MI 代表0 至59 的分钟值而SS 表示0 至59的秒 |
| 对比说明：  HANA中的SECONDDATE表示日期时间，与oracle中的date相同，只是可用时间范围不同 | |

* + 1. Timestamp

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TIMESTAMP** | **TIMESTAMP** |
| TIMESTAMP 数据类型由日期和时间信息组成。其默认格式为‘YYYY-MM-DD HH24:MI:SS.FF6’  其中精度位为0至9 | TIMESTAMP 数据类型由日期和时间信息组成。其默认格式为‘YYYY-MM-DD HH24:MI:SS.FF7’。FFn 代表含有小数的秒，其中n 表示小数部分的数字位数。时间戳的范围从0001-01-01 00:00:00.0000000 至9999-12-31 23:59:59.9999999  其中精度位为1至7 |
| 对比说明：  ORACLE与HANA中的TIMESTAMP表示日期时间，以及表示精度的小数位秒，Oracle默认精度为0.000001秒，HANA默认精度为0.0000001秒 | |

* 1. 数字类型
     1. DECIMAL

|  |  |
| --- | --- |
| ORACLE | HANA |
| **NUMBER** | **DECIMAL** |
| Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from -84 to 127 | DECIMAL (p, s) 数据类型指定了一个精度为p 小数位数为s 的定点小数。精度是有效位数的总数，范围从1 至34。  小数位数是从小数点到最小有效数字的数字个数，范围从-6,111 到6,176，这表示位数指定了十进制小数的指数范围从10-6111 至106176。如果没有指定小数位数，则默认值为0。  当数字的有效数字在小数点的右侧时，小数位数为正；有效数字在小数点左侧时，小数位数为负。 |
| 对比说明：  ORACLE中的number与HANA中的decimal是基本一致的，在精度和小数位数的范围上有所不同 | |

* 1. 字符类型
     1. VARCHAR2

|  |  |
| --- | --- |
| ORACLE | HANA |
| **VARCHAR2** | **VARCHAR** |
| varchar2 存放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 | VARCHAR (n) 数据类型定义了一个可变长度的ASCII 字符串，n 表示最大长度，是一个1 至5000 的整数值 |
| 对比说明：  ORACLE中的varchar2与HANA中的varchar是基本一致的，在最大长度上有所区别 | |

* + 1. NVARCHAR

|  |  |
| --- | --- |
| ORACLE | HANA |
| **NVARCHAR2** | **NVARCHAR** |
| nvarchar2用于存储可变长度的字符串，size 的最大值是 4000，而最小值是 1，其值表示字符的个数，而不是字节数 | NVARCHAR (n) 数据类型定义了一个可变长度的Unicode 字符串，n 表示最大长度，是一个1 至5000 的整数值 |
| 对比说明：  ORACLE中的nvarchar2与HANA中的nvarchar是基本一致的，在最大长度上有所区别 | |

* 1. 大对象类型
     1. BLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **CLOB** | **CLOB** |
| 可用来存储无结构的二进制数据  最大长度是4GB | BLOB 数据类型用来存储大二进制数据  最大大小为2GB |
| 对比说明：  ORACLE和HANA中的BLOB是基本一致的，在最大大小上有所区别 | |

* + 1. CLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **CLOB** | **CLOB** |
| 用于存储对应于数据库定义的字符集的字符数据  最大长度是4GB | CLOB 数据类型用来存储大ASCII 字符数据  最大大小为2GB |
| 对比说明：  ORACLE和HANA中的CLOB是基本一致的，在最大大小上有所区别 | |

* + 1. NCLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **NCLOB** | **NCLOB** |
| 用来存储定宽多字节字符数据，保存本地语言字符集数据最大长度是4GB | NCLOB 数据类型用来存储大Unicode 字符对象  最大大小为2GB |
| 对比说明：  ORACLE和HANA中的NCLOB是基本一致的，在最大大小上有所区别 | |

* + 1. TEXT

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TEXT** |
|  | TEXT 数据类型指定支持文本搜索功能，这不是一个独立的SQL 类型。选择一列TEXT 列会生成一个NCLOB 类型的列 |
| 对比说明：  TEXT是HANA所特有的数据类型，是为了满足其非结构化数据分析的需要，Oracle中没有相应的数据类型 | |

1. SQL函数
   1. 数据类型转换函数
      1. CAST

|  |  |
| --- | --- |
| ORACLE | HANA |
| **CAST** | **CAST** |
| CAST (expression AS data\_type) | CAST (expression AS data\_type) |
| CAST converts one built-in datatype or collection-typed value into another built-in datatype or collection-typed value. | 用来把参数从一个数据类型转换为另一个数据类型，或者测试转换是否可行 |
| **select** **CAST**(111 **AS** **DATE**) **FROM** DUAL  执行出错 | **select** **CAST**(111 **AS** **DATE**) **FROM** DUMMY |
| 对比说明：  CAST在oracle和hana中功能是一致的，只是在转换类型方面有所差距，比如，oracle中number不能转换成date，而hana中可以 | |

* + 1. TO\_ALPHANUM

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_ALPHANUM** |
|  | TO\_ALPHANUM (value) |
|  | 将给定的value 转换为ALPHANUM 数据类型。 |
|  | **SELECT** TO\_ALPHANUM ('10') "to alphanum" **FROM** DUMMY; |
| 对比说明：  ALPHANUM是可变长度的包含字母数字的字符串，在实际应用中很少使用，故该转换函数TO\_ALPHANUM也很少使用 | |

* + 1. TO\_BIGINT

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_BIGINT** |
|  | TO\_BIGINT (value) |
|  | 将value 转换为BIGINT 类型 |
|  | **SELECT** TO\_BIGINT ('10') "to bigint" **FROM** DUMMY; |
| 对比说明：  BIGINT是64位有符号整数，Oracle中一般使用number类型，没有对应转换函数 | |

* + 1. TO\_BINARY

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_BINARY** |
|  | TO\_BINARY (value) |
|  | 将value 转换为BINARY 类型 |
|  | **SELECT** TO\_BINARY ('abc') "to binary" **FROM** DUMMY; |
| 对比说明：  BINARY是二进制数据，Oracle没有对应转换函数 | |

* + 1. TO\_BLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_BLOB** | **TO\_BLOB** |
| TO\_BLOB (value) | TO\_BLOB (value) |
| 将value 转换为BLOB 类型。参数值LONG RAW 或 RAW 类型的数据，输出的是 BLOB 值，LONG RAW、RAW类型相当于其他编程语音中的bin类型数据，即二进制数据 | 将value 转换为BLOB 类型。参数值必须是二进制字符串，即VARBINARY类型 |
| **SELECT** TO\_BLOB (**CAST**('abcde' **AS** RAW)) "to blob" **FROM** DUMMY; | **SELECT** TO\_BLOB (TO\_BINARY('abcde')) "to blob" **FROM** DUMMY; |
| 对比说明：  在oracle和hana中都有TO\_BLOB函数，转换功能基本一样，但是Oracle中的TO\_BLOB函数在11g才有，之前8i到10g，只有TO\_LOB函数 | |

* + 1. TO\_CHAR

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_CHAR** | **TO\_CHAR** |
| TO\_CHAR (value) | TO\_CHAR (value) |
| TO\_CHAR (character) converts NCHAR, NVARCHAR2, CLOB, or NCLOB data to the database character set.  TO\_CHAR (datetime) converts *date* of DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, or TIMESTAMP WITH LOCAL TIME ZONE datatype to a value of VARCHAR2 datatype in the format specified by the date format *fmt*  TO\_CHAR (number) converts *n* of NUMBER datatype to a value of VARCHAR2 datatype, using the optional number format *fmt*. If you omit *fmt*, then *n* is converted to a VARCHAR2 value exactly long enough to hold its significant digits | 将value 转换为CHAR 类型。如果省略format 关键字，转换将会使用Date Formats 中说明的日期格式模型 |
|  | **SELECT** TO\_CHAR (TO\_DATE('2009-12-31'), 'YYYY/MM/DD') "to char" **FROM** DUMMY; |
| 对比说明：  在oracle和hana中都有TO\_CHAR函数，转换功能基本一样。不同之处在于，hana的to\_char是转化为char类型，而oracle的to\_char则是转化为varchar2类型，并且字符编码是基于数据库字符集 | |

* + 1. TO\_CLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_CLOB** | **TO\_CLOB** |
| TO\_CLOB (value) | TO\_CLOB (value) |
| TO\_CLOB converts NCLOB values in a LOB column or other character strings to CLOB values. *char* can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. Oracle executes this function by converting the underlying LOB data from the national character set to the database character set. | 将value 转换为CLOB 类型 |
| **SELECT** TO\_BLOB (**CAST**('abcde' **AS** RAW)) "to blob" **FROM** DUMMY; | **SELECT** TO\_CLOB ('TO\_CLOB converts the value to a CLOB data type') "to clob" **FROM** DUMMY; |
| 对比说明：  在oracle和hana中都有TO\_CLOB函数，转换功能基本一样，但是Oracle中的TO\_BLOB函数在10g才有，之前8i到9i，只有TO\_LOB函数 | |

* + 1. TO\_DATE

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_DATE** | **TO\_DATE** |
| TO\_DATE ( char [, fmt [, 'nlsparam']] ) | TO\_DATE (d [, format]) |
| TO\_DATE converts *char* of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 datatype to a value of DATE datatype. The *fmt* is a date format specifying the format of *char*. If you omit *fmt*, then *char* must be in the default date format. If *fmt* is 'J', for Julian, then *char* must be an integer | 将日期字符串d 转换为DATE 数据类型。如果省略format 关键字，转换将会使用Date Formats 中说明的日期格式模型。 |
| **SELECT** TO\_DATE(  'January 15, 1989, 11:00 A.M.',  'Month dd, YYYY, HH:MI A.M.',  'NLS\_DATE\_LANGUAGE = American')  **FROM** DUAL; | **SELECT** TO\_DATE('2010-01-12', 'YYYY-MM-DD') "to date" **FROM** DUMMY; |
| 对比说明：  在oracle和hana中都有TO\_DATE函数，但是转换功能有一点不同，hana的to\_date只能转换成日期，即hana的date类型，只包含年月日，而oracle的to\_date是转换为oracle的date类型，包含年月日时分秒 | |

* + 1. TO\_DATS

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_DATS** |
|  | TO\_DATS (d) |
|  | 将字符串d 转换为ABAP 日期字符串，格式为”YYYYMMDD” |
|  | **SELECT** TO\_DATS ('2010-01-12') "abap date" **FROM** DUMMY; |
| 对比说明：  该转换函数是HANA所特有的，用于转换为ABAP日期字符串 | |

* + 1. TO\_DECIMAL

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_NUMBER** | **TO\_DECIMAL** |
| TO\_NUMBER ( char [, fmt [, 'nlsparam']] ) | TO\_DECIMAL (value [, precision, scale]) |
| 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 | 将value 转换为DECIMAL 类型。 |
| **SELECT** TO\_NUMBER('-AusDollars100','L9G999D99', ' NLS\_NUMERIC\_CHARACTERS = '',.''  NLS\_CURRENCY = ''AusDollars'' ') "Amount" **FROM** DUAL; | **SELECT** **TO\_DECIMAL**(7654321.888888, 10, 3) "to decimal" **FROM** DUMMY; |
| 对比说明：  HANA的to\_decimal与ORACLE的to\_number在功能上基本一致，都能将字符串转换为数字类型；在参数上有所区别，hana的to\_decimal的参数还包括数字的精度和小数位，而Oracle的to\_number的参数包括数字格式，以及nlsparam参数 | |

* + 1. TO\_DOUBLE

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_DOUBLE** |
|  | TO\_DOUBLE (value) |
|  | 将value 转换为DOUBLE（双精度）数据类型。 |
|  | **SELECT** 3\*TO\_DOUBLE ('15.12') "to double" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_INT

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_INT** |
|  | TO\_INTEGER (value) |
|  | 将value 转换为INTEGER 类型。 |
|  | **SELECT** TO\_INT('10') "to int" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_INTEGER

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_INTEGER** |
|  | TO\_INTEGER (value) |
|  | 将value 转换为INTEGER 类型。 |
|  | **SELECT** TO\_INTEGER('10') "to integer" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_NCHAR

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_NCHAR** |
|  | TO\_NCHAR (value [, format]) |
|  | 将value 转换为NCHAR Unicode 字符类型。如果省略format 关键字，转换将会使用Date Formats中说明的日期格式模型。 |
|  | **SELECT** TO\_NCHAR (TO\_DATE('2009-12-31'), 'YYYY/MM/DD') "to nchar" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_NCLOB

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_NCLOB** | **TO\_NCLOB** |
| TO\_NCLOB ( lob\_column | char ) | TO\_NCLOB (value) |
| TO\_NCLOB converts CLOB values in a LOB column or other character strings to NCLOB values. *char* can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. Oracle implements this function by converting the character set of the LOB column from the database character set to the national character set. | 将value 转换为NCLOB 数据类型。 |
| **INSERT** **INTO** print\_media (product\_id, ad\_id, ad\_fltextn)  **VALUES** (3502, 31001,  TO\_NCLOB('Placeholder for new product description')); | **SELECT** TO\_NCLOB ('TO\_NCLOB converts the value to a NCLOB data type') "to nclob" **FROM** DUMMY; |
| 对比说明：  功能基本一致，其中Oracle的TO\_NCLOB还会将value的字符集转换为本地字符集 | |

* + 1. TO\_NVARCHAR

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_NVARCHAR** |
|  | TO\_NVARCHAR (value [,format]) |
|  | 将value 转换为NVARCHAR Unicode 字符类型。如果省略format 关键字，转换将会使用Date Formats 中说明的日期格式模型。 |
|  | **SELECT** TO\_NVARCHAR(TO\_DATE('2009/12/31'), 'YY-MM-DD') "to nchar" **FROM** DUMMY; |
| 对比说明：  Hana的中to\_nvarchar是将value转化为unicode字符的NVARCHAR类型，功能都是转化为字符串，只是字符集不同，而Oracle中to\_char函数统一实现该功能，通过nlsparam参数实现字符集的定义 | |

* + 1. TO\_REAL

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_REAL** |
|  | TO\_REAL (value) |
|  | 将value 转换为实数（单精度）数据类型。 |
|  | SELECT 3\*TO\_REAL ('15.12') "to real" FROM DUMMY; |
| 对比说明： | |

* + 1. TO\_SECONDDATE

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_DATE** | **TO\_SECONDDATE** |
| TO\_DATE ( char [, fmt [, 'nlsparam']] ) | TO\_REAL (value) |
| TO\_DATE converts *char* of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 datatype to a value of DATE datatype. The *fmt* is a date format specifying the format of *char*. If you omit *fmt*, then *char* must be in the default date format. If *fmt* is 'J', for Julian, then *char* must be an integer | 将value 转换为SECONDDATE 类型。如果省略format 关键字，转换将会使用Date Formats 中说明的日期格式模型 |
| **SELECT** TO\_DATE(  'January 15, 1989, 11:00 A.M.',  'Month dd, YYYY, HH:MI A.M.',  'NLS\_DATE\_LANGUAGE = American')  **FROM** DUAL; | **SELECT** TO\_SECONDDATE ('2010-01-11 13:30:00', 'YYYY-MM-DD HH24:MI:SS') "to seconddate" **FROM** DUMMY; |
| 对比说明：  HANA中有几种日期时间类型，也分别对应3种转换函数，该转换函数to\_seconddate是最接近Oracle的to\_date，在具体使用时，注意甄别 | |

* + 1. TO\_SMALLDECIMAL

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_SMALLDECIMAL** |
|  | TO\_SMALLDECIMAL (value) |
|  | 将value 转换为SMALLDECIMAL 类型。 |
|  | **SELECT** TO\_SMALLDECIMAL(7654321.89) "to smalldecimal" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_SMALLINT

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_SMALLINT** |
|  | TO\_SMALLINT (value) |
|  | 将value 转换为SMALLINT 类型。 |
|  | **SELECT** TO\_SMALLINT ('10') "to smallint" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_TIME

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_TIME** |
|  | TO\_TIME (t [, format]) |
|  | 将时间字符串t 转换为TIME 类型。如果省略format 关键字，转换将会使用Date Formats 中说明的日期格式模型 |
|  | **SELECT** TO\_TIME ('08:30 AM', 'HH:MI AM') "to time" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_TIMESTAMP

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_TIMESTAMP** |
|  | TO\_TIMESTAMP (d [, format]) |
|  | 将时间字符串t 转换为TIMESTAMP 类型。如果省略format 关键字，转换将会使用Date Formats中说明的日期格式模型。 |
|  | **SELECT** TO\_TIMESTAMP ('2010-01-11 13:30:00', 'YYYY-MM-DD HH24:MI:SS') "to timestamp" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_TINYINT

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **TO\_TINYINT** |
|  | TO\_TINYINT (value) |
|  | 将value 转换为TINYINT 类型。 |
|  | **SELECT** TO\_TINYINT ('10') "to tinyint" **FROM** DUMMY; |
| 对比说明： | |

* + 1. TO\_VARCHAR

|  |  |
| --- | --- |
| ORACLE | HANA |
| **TO\_CHAR** | **TO\_VARCHAR** |
| TO\_CHAR (value) | TO\_VARCHAR (value [, format]) |
|  | 将给定value 转换为VARCHAR 字符串类型。如果省略format 关键字，转换将会使用Date Formats中说明的日期格式模型。 |
|  | **SELECT** TO\_VARCHAR (TO\_DATE('2009-12-31'), 'YYYY/MM/DD') "to char" **FROM** DUMMY; |
| 对比说明：  HANA中的to\_varchar与Oracle的to\_char基本相关，都是将数字型、时间日期型数据转换为varchar（varchar2）类型 | |

* 1. 日期时间函数
     1. ADD\_DAYS

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **ADD\_DAYS** |
|  | ADD\_DAYS (d, n) |
|  | 计算日期d 后n 天的值 |
|  | **SELECT** ADD\_DAYS (TO\_DATE ('2009-12-05', 'YYYY-MM-DD'), 30) "add days" **FROM** DUMMY; |
| 对比说明：  Oracle中没有对应的函数，只是在oracle中，date类型可以直接+-number，实现同样功能；HANA中该函数可以用于date、seconddate类型，并且date、seconddate不能加减number | |

* + 1. ADD\_MONTHS

|  |  |
| --- | --- |
| ORACLE | HANA |
| **ADD\_MONTHS** | **ADD\_MONTHS** |
| ADD\_MONTHS (d, n) | ADD\_MONTHS (d, n) |
| 返回日期d后n个月的日期，如果日期d是月末、或是结果日期所在月天数较少，少于日期d的日期数，结果将返回结果月份的月末日期。 | 计算日期d 后n 月的值。 |
| **SELECT** ADD\_MONTHS (TO\_DATE ('2009-12-05', 'YYYY-MM-DD'), 1) "add months" **FROM** DUAL | **SELECT** ADD\_MONTHS (TO\_DATE ('2009-12-05', 'YYYY-MM-DD'), 1) "add months" **FROM** DUMMY |
| 对比说明：  Hana与oracle中的该函数功能一样，HANA中该函数可以用于date、seconddate类型 | |

* + 1. ADD\_SECONDS

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **ADD\_SECONDS** |
|  | ADD\_SECONDS (t, n) |
|  | 计算时间t 后n 秒的值。 |
|  | **SELECT** ADD\_SECONDS (TO\_TIMESTAMP ('2012-01-01 23:30:45'), 60\*30) "add seconds" **FROM** DUMMY; |
| 对比说明：  Oracle中没有对应的函数，只是在oracle中，date类型可以直接+-number，实现同样功能；HANA中该函数可以用于time、timestamp、seconddate类型，并且time、timestamp、seconddate不能加减number | |

* + 1. ADD\_YEARS

|  |  |
| --- | --- |
| ORACLE | HANA |
|  | **ADD\_YEARS** |
|  | ADD\_YEARS (d, n) |
|  | 计算日期d 后n 年的值。 |
|  | **SELECT** ADD\_YEARS (TO\_DATE ('2009-12-05', 'YYYY-MM-DD'), 1) "add years" **FROM** DUMMY; |
| 对比说明： | |

* + 1. CURRENT\_DATE

|  |  |
| --- | --- |
| ORACLE | HANA |
| **SYSDATE** | **CURRENT\_DATE** |
| Sysdate | CURRENT\_DATE |
| 返回当前本地系统日期时间。 | 返回当前本地系统日期。 |
| **SELECT** **SYSDATE** **FROM** DUMMY; | **SELECT** **CURRENT\_DATE** "current date" **FROM** DUMMY; |
| 对比说明：  HANA的CURRENT\_TIME与oracle的SYSDATE功能基本一致，但是hana的该函数只是返回日期 | |

* + 1. CURRENT\_TIME
    2. CURRENT\_TIMESTAMP
    3. CURRENT\_UTCDATE
    4. CURRENT\_UTCTIME
    5. CURRENT\_UTCTIMESTAMP
    6. DATNAME
    7. DAYOFMONTH
    8. DAYOFYEAR
    9. DAYS\_BETWEEN
    10. EXTRACT
    11. HOUR
    12. ISOWEEK
    13. LAST\_DAY
    14. LOCALTOUTC
    15. MINUTE
    16. MONTH
    17. MONTHNAME
    18. NEXT\_DAY
    19. NOW
    20. QUARTER
    21. SECOND
    22. SECONDS\_BETWEEN
    23. UTCTOLOCAL
    24. WEEK
    25. WEEKDAY
    26. YEAR
  1. 数字函数

ABS

ACOS

ASIN

ATAN

ATAN2

BINTOHEX

BITAND

CEIL

COS

COSH

COT

EXP

FLOOR

GREATEST

HEXTOBIN

LEAST

LN

LOG

MOD

POWER

ROUND

SIGN

SIN

SINH

SQRT

TAN

TANH

UMINUS

* 1. 字符串函数
     1. ASCII
     2. CHAR
     3. CONCAT
     4. LCASE
     5. LEFT
     6. LENGTH
     7. LOCATE
     8. LOWER
     9. LPAD
     10. LTRIM
     11. NCHAR
     12. REPLACE
     13. RIGHT
     14. RPAD
     15. RTRIM
     16. SUBSTR\_AFTER
     17. SUBSTR\_BEFORE
     18. SUBSTRING
     19. TRIM
     20. UCASE
     21. UNICODE
     22. UPPER
  2. 杂项函数
     1. COALESCE
     2. CURRENT\_CONNECTION
     3. CURRENT\_SCHEMA
     4. CURRENT\_USER
     5. GROUPING\_ID
     6. IFNULL
     7. MAP
     8. NULLIF
     9. SESSION\_CONTEXT
     10. SESSION\_USER
     11. SYSUUID