DB2 9 开发应用手册

作者：王亚南（db2china论坛id：leo）

SQL Cookbook

目 录

1 数据类型 4

1.1 内置数据类型 4

1.2 数字类型 (Number) 5

1.2.1 整型 (Binary Integer) 5

1.2.2 浮点型 (Floating Point) 5

1.2.3 小数型 (Decimal) 5

1.2.4 字符类型 (Character strings) 6

1.2.5 定长字符 (Fixed-length Character String) 6

1.2.6 变长字符 (Varying-length Character String) 6

1.3 日期时间类型 (Datatime Values) 6

1.4 特殊寄存器 (Special Registers) 6

2 函数(Functions) 7

2.1 标量函数(Scalar Functions) 7

2.1.1 ABS or ABSVAL 7

2.1.2 ASCII 8

2.1.3 BIGINT 8

2.1.4 CEIL(CEILING) 8

2.1.5 FLOOR 9

2.1.6 CHAR 9

2.1.7 CHR 10

2.1.8 COALESCE 10

2.1.9 CONCAT 11

2.1.10 DATE 11

2.1.11 DAY 12

2.1.12 DAYOFWEEK\_ISO 12

2.1.13 DAYOFWEEK 13

2.1.14 DAYNAME 13

2.1.15 DAYOFYEAR 14

2.1.16 DAYS 14

2.1.17 DEC / DECIMAL 14

2.1.18 DIGITS 15

2.1.19 DOUBLE/DOUBLE\_PRECISION 16

2.1.20 ENCRYPT 16

2.1.21 EXP 17

2.1.22 GENERATE\_UNIQUE 17

2.1.23 HEX 18

2.1.24 HOUR 18

2.1.25 INSERT 18

2.1.26 INTEGER/INT 19

2.1.27 JULIAN\_DAY 19

2.1.28 LCASE(LOWER) / UCASE(UPPER) 20

2.1.29 LEFT 20

2.1.30 LENGTH 20

3 SQL 语句(SQL Statements) 21

3.1 Values子句 (Values Clause) 21

3.2 CASE 22

3.2.1 类型1 22

3.2.2 类型2 22

3.3 Union, Intersect and Except 22

3.4 Order By 23

3.5 Group By, and Having 24

3.5.1 GROUPING SETS 24

3.5.2 ROLLUP( ) 25

3.5.3 CUBE( ) 25

3.6 Flow Control Statements 26

3.6.1 IF 26

3.6.2 FOR 26

3.6.3 WHILE 26

3.6.4 ITERATE 26

3.6.5 LEAVE 27

3.6.6 SIGNAL 27

3.6.7 GET DIAGNOSTICS 27

3.7 JOIN 27

3.8 Query Sequence (查询检索顺序) 28

3.9 OLAP Functions (分析函数) 28

3.9.1 Main Olap Funcation Summary(主要Olap函数概要) 30

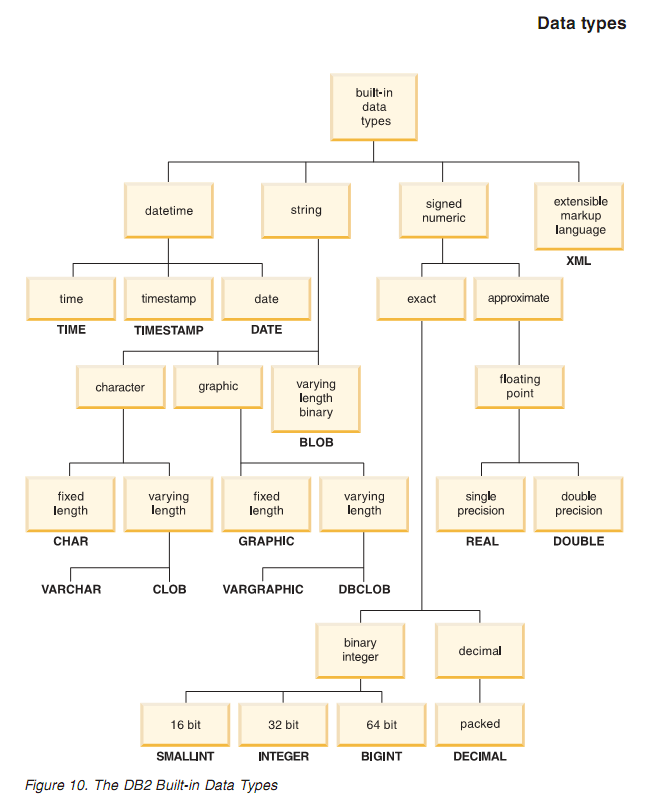
3.10 GLOBAL TEMPORARY TABLE (全局临时表) 31

4 附录: 33

4.1 参考文档 33

# 数据类型

## 内置数据类型



## 数字类型 (Number)

### 整型 (Binary Integer)

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Precision** | **Range** |
| SMALLINT | 5 | [-32768, 32767] |
| INTEGER | 10 | [−2147483648, 214483647] |
| BIGINT | 19 | [−9223372036854775808, 9223372036854775807] |

/\*\* 提示: 使用下面SQL进行相关信息查询:

select

name**,** schema**,** length**(**rtrim**(**char**(**power**(**bigint**(-**2**),** length**\***8**-**1**))))-**1 precision**,**  
power**(**bigint**(-**2**),** length**\***8**-**1**)** minvalue**,** abs**(**power**(**bigint**(-**2**),** length**\***8**-**1**)+**1**)** maxvalue  
from sysibm**.**sysdatatypes  
where name in **(**'SMALLINT'**,** 'INTEGER'**,** 'BIGINT'**)**

**\*\*/**

### 浮点型 (Floating Point)

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Precision** | **Range** |
| REAL | -- | [-3.4028234663852886e+38,1.1754943508222875e-38]  [1.1754943508222875e-38,3.4028234663852886e+38] |
| FLOAT DOUBLE | -- | [-1.7976931348623158e+308,-2.2250738585072014e-308]  [2.2250738585072014e-308,1.7976931348623158e+308] |

### 小数型 (Decimal)

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Precision** | **Range** |
| DECIMAL  NUMERIC | <=31 | [-10\*\*31+1,10\*\*31-1] |

### 字符类型 (Character strings)

### 定长字符 (Fixed-length Character String)

|  |  |
| --- | --- |
| **Data Type** | **Length (byte)** |
| CHAR | [1, 254] |

### 变长字符 (Varying-length Character String)

|  |  |
| --- | --- |
| **Data Type** | **Length (byte)** |
| VARCHAR | <=32672 |
| LONG VARCHAR | <=32700 |
| CLOB | <=2147483647 (2G) |

## 日期时间类型 (Datatime Values)

|  |  |
| --- | --- |
| **Data Type** | **Length (byte)** |
| DATE | 10 |
| TIME | 8 |
| DATETIME | 26 |

## 特殊寄存器 (Special Registers)

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial** | **Special Register** | **Updatable** | **Data Type** |
|  | CURRENT CLIENT\_ACCTNG | No | VARCHAR(255) |
|  | CURRENT CLIENT\_APPLNAME | No | VARCHAR(255) |
|  | CURRENT CLIENT\_APPLNAME | No | VARCHAR(255) |
|  | CURRENT CLIENT\_USERID | No | VARCHAR(255) |
|  | CURRENT CLIENT\_WRKSTNNAME | No | VARCHAR(255) |
|  | CURRENT DATE | No | DATE |
|  | CURRENT DBPARTITIONNUM | No | INTEGER |
|  | CURRENT DEFAULT TRANSFORM GROUP | Yes | VARCHAR(18) |
|  | CURRENT DEGREE | Yes | CHAR(5) |
|  | CURRENT EXPLAIN MODE | Yes | VARCHAR(254) |
|  | CURRENT EXPLAIN SNAPSHOT | Yes | CHAR(8) |
|  | CURRENT ISOLATION | Yes | CHAR(2) |
|  | CURRENT LOCK TIMEOUT | Yes | INTEGER |
|  | CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION | Yes | VARCHAR(254) |
|  | CURRENT PACKAGE PATH | Yes | VARCHAR(4096) |
|  | CURRENT PATH | Yes | VARCHAR(254) |
|  | CURRENT QUERY OPTIMIZATION | Yes | INTEGER |
|  | CURRENT REFRESH AGE | Yes | DECIMAL(20,6) |
|  | CURRENT SCHEMA | Yes | VARCHAR(128) |
|  | CURRENT SERVER | No | VARCHAR(18) |
|  | CURRENT TIME | No | TIME |
|  | CURRENT TIMESTAMP | No | TIMESTAMP |
|  | CURRENT TIMEZONE | No | DECIMAL(6,0) |
|  | CURRENT USER | No | VARCHAR(128) |
|  | SESSION\_USER | Yes | VARCHAR(128) |
|  | SYSTEM\_USER | No | VARCHAR(128) |
|  | USER | Yes | VARCHAR(128) |

/\*\* 说明: Updatable=Yes可使用set 语句进行更新 \*\*/

# 函数(Functions)

## 标量函数(Scalar Functions)

概要：可被用于任何表达式能够使用的地方, 每次只能返回单个值, 而不是一组值。

### ABS or ABSVAL

|  |  |
| --- | --- |
| Format | ABS / ABSVAL (expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的绝对值 |
| Input Argument | 任何内置的数字类型表达式 |
| Returns | 返回值精度、数据类型与输入参数相同 |
| Example | select d1, abs(d1) as ABS\_d1, f1, abs(f1) as ABS\_f1 from scalar |
| Note | 若参数为SMALLINT, INT, BIGINT数据类型负数最大值, 则SQL0802N:算术溢出或其他算术异常, 如:  Select ABS(SMALLINT(-32768)) d1 From scalar |

### ASCII

|  |  |
| --- | --- |
| Format | ABCII (expression) |
| Schema | SYSFUN |
| Description | 返回输入参数的ASCII码值 |
| Input Argument | * CHAR * VARCHAR(4000) * CLOB(1M) |
| Returns | INTEGER |
| Example | select ascii(c1) from scalar |
| Note | 仅返回输入参数"最左边"字符ASCII码值 |

### BIGINT

|  |  |
| --- | --- |
| Format | BIGINT (numeric-expression)  BIGINT (character-expression)  BIGINT (datetime-expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的64 bit 整数 |
| Input Argument | * numeric-type * VARCHAR |
| Returns | BIGINT |
| Example | Select d1, bigint(d1) BIGINT\_d1, f1, bigint(f1) BIGINT\_f1, ts1, bigint(ts1) BIGINT\_ts1, bigint(dt1) BIGINT\_dt1, bigint(ts1) BIGINT\_ts1 From db2admin.scalar |
| Note | datetime-expression 数据类型:  date 返回值为yyyymmdd  time 返回值为hhmmss  timestamp 返回值为 yyyymmddhhmmss (microseconds忽略) |

### CEIL(CEILING)

|  |  |
| --- | --- |
| Format | CEIL (numeric-expression) |
| Schema | SYSIBM |
| Description | 返回>=输入参数的最小整数 |
| Input Argument | * SMALLINT * INTEGER * BIGINT * DOUBLE |
| Returns | 与输入参数类型相同 |
| Example | values ceil(1.1), ceil(0), ceil(-1.1) |
| Note | 与FLOOR近似 |

### FLOOR

|  |  |
| --- | --- |
| Format | FLOOR (numeric-expression) |
| Schema | SYSIBM |
| Description | 返回<=输入参数的最大整数 |
| Input Argument | * SMALLINT * INTEGER * BIGINT * DOUBLE |
| Returns | 与输入参数类型相同 |
| Example | values floor(1.1), floor(0), floor(-1.1) |
| Note | 与CEIL近似 |

### CHAR

|  |  |
| --- | --- |
| Format | CHAR(character-expression, length)  CHAR(datetime-expression, format)  CHAR(integer-expression)  CHAR(float-point-expression, decimal-character)  CHAR(decimal-expression, decimal-charact) |
| Schema | SYSIBM |
| Description | 将输入参数转换为字符串 |
| Input Argument | * character-type * character-type, INTEGER * datetime-type * datetime-type, key * SMALLINT * INTEGER * BIGINT * DECIMAL * DECIMAL, VARCHAR |
| Returns | * CHAR * CHAR(integer) * CHAR * CHAR * CHAR(6) * CHAR(11) * CHAR(20) * CHAR(2+precision) * CHAR(2+precision) |
| Example | /\* Integer / float-point / decimal expression \*/  values (char(1), char(0), char(-1)) ,(char(double(1.0)), char(double(0.0)), char(double(-1.0))) ,(char(dec(1.0, 5, 2), ‘\*’), char(dec(0.0, 5, 2), ’\*’), char(dec(-1.0, 5, 2)))  /\* datetime-expression format: ISO / USA / EUR / JIS / LOCAL\*/  values char(current date, iso)   ,char(current date, eur)   ,char(current date, jis)   ,char(current date, usa) |
| Note | 1. 输入参数character-expression length范围[0,254] 2. 输入参数small / integer / bigint expression, 返回值lengh in (6 , 11, 20), 若位数不足, 应用”右对齐”, 向右补” ” 3. 输入参数double / float expression, 返回值lengh=24 , 若位数不足, 应用”左对齐”, 向左补”0” 4. 输入参数decimal expression, 返回值lengh=p+2, p为输入参数精度, 若位数不足, 应用”左对齐”, 向左补”0” 5. 输入参数timstamp expression不支持任何format参数 |

### CHR

|  |  |
| --- | --- |
| Format | CHR(expression) |
| Schema | SYSFUN |
| Description | 返回输入参数对应的ASCII |
| Input Argument | INTEGER, |
| Returns | CHAR(1) |
| Example | values chr(0), chr(-1), chr(255), chr(32768) |
| Note | 1、与ASCII()互为反函数  2、参数值范围 [0, 255], 若参数值>255, 结果仍为CHR(255) |

### COALESCE

|  |  |
| --- | --- |
| Format | COALESCE(expression, expression, …) |
| Schema | SYSIBM |
| Description | 按照输入参数顺序, 返回列表中第一个非null参数值 |
| Input Argument | * any-type, * any-union-compatible-type,... |
| Returns | 与输入参数类型相同 |
| Example | with temp1(c1,c2,c3) as ( values (cast(null as smallint)   ,cast(null as smallint)   ,cast(10 as smallint)))  select coalesce(c1,c2,c3) as cc1   ,case   when c1 is not null then c1   when c2 is not null then c2   when c3 is not null then c3   end as cc2  from temp1 |
| Note | 1. 输入参数类型必须兼容 2. CASE表达式的简化版 |

### CONCAT

|  |  |
| --- | --- |
| Format | CONCAT(expression1, expression2) |
| Schema | SYSIBM |
| Description | 返回2个输入参数的合并字符串 |
| Input Argument | * string-type, * compatible-string-type |
| Returns | 与输入参数类型相同 |
| Example | values 'A' || 'B', 'A' concat 'B', concat('A','B') |
| Note | 1. 输入参数类型必须兼容 2. ‘||’ 为CONCAT同义词(synonym) |

### DATE

|  |  |
| --- | --- |
| Format | DATE(expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的日期表示 |
| Input Argument | * DATE * TIMESTAMP * DOUBLE * VARCHAR |
| Returns | DATE |
| Example | values date('2009-01-01'), date('2009001'), date(2009\*365) |
| Note | 输入参数类型及格式说明:   1. Date 或Timestamp 类型: 时间部分忽略 2. 字符类型: 3. Char或Varchar有效的Date /Timestamp 字符表达式; 4. 字符串长度=7 Byte, 且非CLOB, LONG VARCHAR, DBCLOB, LONG VARGRAPHIC, 格式：yyyynnn, 表示yyyy年第nnn天, nnn 取值范围 [1, 366]; 注意nnn的取值, 最大值应为yyyy年最后一天   3、数字类型: 表示从0001-01-01始n-1天, n>0 且小数部分忽略 |

### DAY

|  |  |
| --- | --- |
| Format | DAY(expression) |
| Schema | SYSIBM |
| Description | 返回日期/时间戳表达式所在天数 |
| Input Argument | * VARCHAR * DATE * TIMESTAMP * DECIMAL |
| Returns | INTEGER |
| Example | values day(date('2009-12-31')) |
| Note | 输入参数类型及格式说明:   1. Date 或Timestamp 类型: 时间部分忽略 2. 字符类型:Char或Varchar有效的Date /Timestamp 字符表达 3. Date /Timestamp 区间段, 结果[-30, 30] 4. 若参数为Date /Timestamp 区间段, 只比较2个日期的天数部分,忽略年\月   with tmp (duration) as ( values date('2008-11-01')-date('2009-12-31'))  select duration, day(duration) dd from tmp |

### DAYOFWEEK\_ISO

|  |  |
| --- | --- |
| Format | DAYOFWEEK\_ISO (expression) |
| Schema | SYSFUN |
| Description | 返回日期时间表达式所属星期的第几天 |
| Input Argument | * VARCHAR(26) * DATE * TIMESTAMP |
| Returns | INTEGER |
| Example | with tmp (num) as (  values 1   union all   select num+1 from tmp where num <7 ) select dayofweek(current date + num days) from tmp |
| Note | 返回值范围 [1, 7], 1=Monday (星期一) |

### DAYOFWEEK

|  |  |
| --- | --- |
| Format | DAYOFWEEK (expression) |
| Schema | SYSFUN |
| Description | 返回日期时间表达式所属星期的第几天 |
| Input Argument | * VARCHAR(26) * DATE * TIMESTAMP |
| Returns | INTEGER |
| Example | with tmp (num) as (  values 1   union all   select num+1 from tmp where num <7 ) select dayofweek(current date + num days) from tmp |
| Note | 返回值范围 [1, 7], 1=Sunday(星期日) |

### DAYNAME

|  |  |
| --- | --- |
| Format | DAYNANE(expression) |
| Schema | SYSFUN |
| Description | 返回日期时间表达式所属星期名 如: Sunday, Monday |
| Input Argument | * VARCHAR(26) * DATE * TIMESTAMP |
| Returns | * VARCHAR(100) * VARCHAR(100) * VARCHAR(100) |
| Example | with tmp (num) as (  values 1   union all   select num+1 from tmp where num <7 ) select dayname(current date + num days) from tmp |
| Note | 返回值基于本地的日期时间格式设置 |

### DAYOFYEAR

|  |  |
| --- | --- |
| Format | DAYOFYEAR (expression) |
| Schema | SYSFUN |
| Description | 返回日期表达式所属年的第几天 |
| Input Argument | * VARCHAR(26) * DATE * TIMESTAMP |
| Returns | INTEGER |
| Example | values dayofyear(current date) , dayofyear(current timestamp) , dayofyear(date('2010-01-01')) |
| Note | 返回值范围 [1, 366] |

### DAYS

|  |  |
| --- | --- |
| Format | DAYS(expression) |
| Schema | SYSIBM |
| Description | 返回从0001-01-01到日期表达式所经过的天数 |
| Input Argument | * VARCHAR * TIMESTAMP * DATE |
| Returns | INTEGER |
| Example | values days(current date) , days(current timestamp) , days(date('0001-01-01')) |
| Note |  |

### DEC / DECIMAL

|  |  |
| --- | --- |
| Format | DEC (numeric-expression, precision-integer, scale-integer)  DEC (character-expression, precision-integer, scale-integer,  decimal-character)  DEC(datetime-expression, precision-integer, scale-integer) |
| Schema | SYSIBM |
| Description | 返回输入参数的DECIMAL类型表示 |
| Input Argument | * numeric-type * numeric-type, INTEGER * numeric-type , INTEGER, INTEGER * VARCHAR * VARCHAR, INTEGER * VARCHAR, INTEGER, INTEGER * VARCHAR, INTEGER, INTEGER, VARCHAR |
| Returns | DECIMAL |
| Example | values (  dec(1, 5, 2)  ,dec(1.2345678e2, 5, 2)  ,dec('-1,00', 5, 2, ',')  ,dec(current date)  ,dec(current time)  ,dec(current timestamp)) |
| Note | 1. numeric-expression 参数说明: precision-integer 精度 范围[1,31],缺省的数字表达式精度:   (float / decimal=15位, bigint=19位, integer=11位, smallint=5位) scale-integer 小数位数, 范围[0, p], 默认值=0   1. character-expression 参数说明: precision-integer 精度 范围[1,31],缺省的字符表达式精度=15位, scale-integer 小数位数, 范围[0, p], 默认值=0, decimal-character 小数点定界符 2. datetime-expression参数说明: 3. DATE类型: DECIMAL(8,0) Format=yyyymmdd 4. TIME类型: DECIMAL(6,0) Format =hhmmss 5. TIMESTAMP类型: DECIMAL(20,6)   Format = yyyymmddhhmmss.nnnnnn |

### DIGITS

|  |  |
| --- | --- |
| Format | DIGITS(expression) |
| Schema | SYSIBM |
| Description | 返回数字输入参数的字符表达式 |
| Input Argument | * SMALLINT * INTEGER * BIGINTEGER * DECIMAL |
| Returns | CHAR |
| Example | values (  digits(smallint(1))  ,digits(int(1))  ,digits(bigint(1))  ,digits(dec(1.1, 5, 2))  ,digits(dec(-1.1, 5, 2))  ,digits(dec(0, 5, 2)) ) |
| Note | 1. 返回值仅包含数字, 不包含+/-和小数点 2. 根据输入参数类型, 返回值长度固定, 缺省长度: 3. SMALLINT 5位 4. INTEGER 10位 5. BIGINT 19 位 6. DECIMAL P 位(P为输入参数精度) 7. 若返回值长度位数不足, 应用“左对齐”, 向“左”补0 |

### DOUBLE/DOUBLE\_PRECISION

|  |  |
| --- | --- |
| Format | DOUBLE / DOUBLE\_PRECISION (numeric-expression)  DOUBLE / DOUBLE\_PRECISION(string-expression) |
| Schema | SYSIBM  SYSFUN(输入参数为string-expression) |
| Description | 返回输入参数的对应的浮点值 |
| Input Argument | * numeric-type * VARCHAR |
| Returns | DOUBLE |
| Example | values  double(cast(null as varchar(10)))  ,double(-1)  ,double(0)  ,double(dec(1.23456, 7, 6)) |
| Note |  |

### ENCRYPT

|  |  |
| --- | --- |
| Format | ENCRYPT (data-string-expression, password-string-expression, hint-string-expression) |
| Schema | SYSIBM |
| Description | 返回输入字符表达式加密值 |
| Input Argument | * VARCHAR |
| Returns | * VARCHAR FOR BIT DATA |
| Example | with tmp as ( select id, name,  encrypt(name,'that idiot','my brother') encrypt\_hint\_name, encrypt(name,'that idiot') encrypt\_name from leo.staff  where id < 30  ) select id, name, encrypt\_name,  gethint(encrypt\_hint\_name) hint, gethint(encrypt\_name) no\_hint from tmp  /\* set encryption password 指定当前session加密密码 \*/  Create table emp (ssn varchar(24) for bit data);  Set encryption password=’Leo20101129’;  Insert into emp (ssn) values encrypt(‘289-46-8832’); |
| Note | * Data-string-expression:不使用hint length=(0,32663);使用hint length=(0,32631) * password-string-expression: length=[6,127] bit, 若null, 则set encryption password 指定当前session加密密码 * hint-string-expression:length=[0,32] bit, 使用GETHINE() 函数查看 |

### EXP

|  |  |
| --- | --- |
| Format | EXP(expression) |
| Schema | SYSFUN |
| Description | 返回e的n次幂 (n 输入参数) |
| Input Argument | DOUBLE |
| Returns | DOUBLE |
| Example | values   ln(exp(-1.0))  ,ln(exp(0.0))  ,ln(exp(1.0)) |
| Note | 与LN() 互为反函数 |

### GENERATE\_UNIQUE

|  |  |
| --- | --- |
| Format | GENERATE\_UNIQUE () |
| Schema | SYSIBM |
| Description | 返回根据时间戳和节点号(多分区数据库)产生的唯一键值 |
| Input Argument | -- |
| Returns | CHAR(13) FOR BIT DATA |
| Example | values (  generate\_unique()  ,timestamp(generate\_unique())   ,timestamp(generate\_unique())+current timezone ) |
| Note | 当多事务并发时, 函数有可能返回非唯一值, 不建议用于主键生成 |

### HEX

|  |  |
| --- | --- |
| Format | HEX (expression) |
| Schema | SYSIBM |
| Description | 返回表达式16进制字符串表示 |
| Input Argument | * any-builtin-type |
| Returns | * VARCHAR |
| Example | values  hex('A')  ,hex(1)  ,hex(float(1.0))  ,hex(dec(1.0))  ,hex(current date)  ,hex(current time)  ,hex(current timestamp) |
| Note | datetime-expression 数据类型:  date 返回值为yyyymmdd  time 返回值为hhmmss  timestamp 返回值为 yyyymmddhhmmss (microseconds忽略) |

### HOUR

|  |  |
| --- | --- |
| Format | HOUR (expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的小时部分 |
| Input Argument | * VARCHAR * TIME * TIMESTAMP |
| Returns | INTEGER |
| Example | values  hour(current time) ,hour(time('23:59:59')) ,hour(current time)-hour(time('23:59:59')) |
| Note | * 当输入参数为TIME/TIMESTAMP, 返回值范围[0, 24] * 当输入参数为TIME/TIMESTAMP DURATION, 返回值范围[-24, 24] |

### INSERT

|  |  |
| --- | --- |
| Format | INSERT (source, startPos, delBytes, newValue) |
| Schema | SYSFUN |
| Description | 在source的startPos位置开始替换 delBytes个字符并插入newValue |
| Input Argument | * VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000) * CLOB(1M), INTEGER, INTEGER, CLOB(1M) * BLOB(1M), INTEGER, INTEGER, BLOB(1M) |
| Returns | * VARCHAR(4000) * CLOB(1M) * BLOB(1M) |
| Example | select name   ,insert(name, 3, 2, 'a')   ,insert(name, 3, 2, 'ab')   ,insert(name, 3, 2, 'abc') from leo.staff  where id < 40 |
| Note | 如果插入newValue, 设置delBytes=0 |

### INTEGER/INT

|  |  |
| --- | --- |
| Format | INTEGER / INT(expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的整数表达式 |
| Input Argument | * VARCHAR * numeric-type * DATE * TIME |
| Returns | INTEGER |
| Example | values int(current date), int(current time) |
| Note | * 当输入参数为DATE类型, 返回格式为yyyyymmdd * 当输入参数为TIME类型, 返回格式为hhmmss |

### JULIAN\_DAY

|  |  |
| --- | --- |
| Format | JULIAN\_DAY (expression) |
| Schema | SYSFUN |
| Description | 从4713BC-01-01开始到输入参数经过的天数 |
| Input Argument | * VARCHAR(26) * DATE * TIMESTAMP |
| Returns | INTEGER |
| Example | values julian\_day('0001-01-01') |
| Note |  |

### LCASE(LOWER) / UCASE(UPPER)

|  |  |
| --- | --- |
| Format | LCASE/LOWER(expression)  UCASE/UPPER(expression) |
| Schema | SYSIBM |
| Description | 返回输入参数的大/小写表达式 |
| Input Argument | * CHAR * VARCHAR |
| Returns | * CHAR * VARCHAR |
| Example | Values lower(‘A’), upper(‘a’) |
| Note |  |

### LEFT

|  |  |
| --- | --- |
| Format | LEFT (expression1, expression2) |
| Schema | SYSFUN |
| Description | 返回从expression1最左边起向右截取expression2个字符 |
| Input Argument | * VARCHAR(4000), INTEGER * CLOB(1M), INTEGER * BLOB(1M), INTEGER |
| Returns | * VARCHAR(4000) * CLOB(1M) * BLOB(1M) |
| Example | values left('ABCD', 2) |
| Note |  |

### LENGTH

|  |  |
| --- | --- |
| Format | LENGTH (expression, CODEUNITS16)  LENGTH (expression, CODEUNITS32)  LENGTH (expression,OCTETS) |
| Schema | SYSIBM |
| Description | 返回输入参数的长度 |
| Input Argument | * any-builtin-type * any-builtin-type, string-unit   CODEUNITS16 -- 16-bit UTF-16 code units  CODEUNITS32 -- 32-bit UTF-32 code units  OCTETS -- 字节数 |
| Returns | INTEGER |
| Example | values  length(current date) ,length(current time) ,length(current timestamp) ,length('ABCD') ,length(1.2345E4) ,length(dec(12.34, 5, 2))  values  length(x'F09D849E', OCTETS) ,length(x'F09D849E', CODEUNITS16) ,length(x'F09D849E', CODEUNITS32) |
| Note |  |

# SQL 语句(SQL Statements)

## Values子句 (Values Clause)

概述: 一般用于临时表或视图的定义，不能使用group by , order by，且不能被更新。

|  |  |  |
| --- | --- | --- |
| **Usage** | **Example** | **Remark** |
| 一般用法 | VALUES  VALUES (1)  VALUES 1, 2, 3  VALUES (1), (2), (3)  VALUES(1, 1), (2, 2), (3, null) | 1 row, 1 col  1 row, 1 col  1 row, 3 col  3 row, 1 col  3 row, 2 col |
| 使用NULL | VALUES (1, null), (2, null), (3, null)  VALUES(1, null),(2, null),(3, cast(null as int)) | 语法错误  语法正确 |
| 定义临时表 | WITH temp1 (col1, col2) AS (  VALUES (0, cast ('AA' AS char(1)))  ,(1, cast ('BB' AS char (1)))  ,(2, cast (null AS char (1)))  ) |  |
| 定义临时表并命名列 | Select c1, c2, c3  From table (  VALUES(1, 1, 1), (2, 2, 2), (3, 3, 3)  ) AS T(c1, c2, c3) |  |

/\*\* 注意:

1、"( )"内的","作为"列"分割符, " ( ) "间的","作为"行"分割符 ；

2、不支持列值全部为NULL, 否则需用CAST函数进行显示类型转；

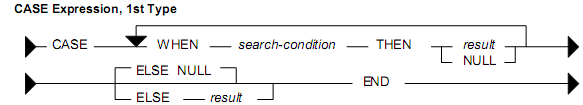
3、与”select expression from sysibm.sysdummy1” 功能近似。 \*\*/

## CASE

概述: case表达式能够在SQL Statements中实现if-then-else功能。

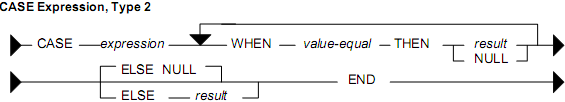
### 类型1

每个when子句单独进行条件匹配，语法:



### 类型2

每个when 子句仅与条件表达式作相等检查，语法:



/\*\* 注意:

1、若存在多个when子句匹配, 按表示式顺序仅应用第一个子句匹配；

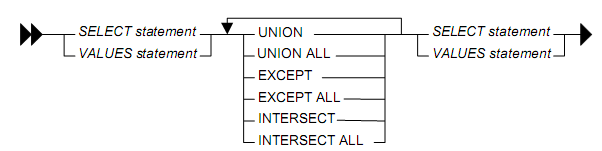
2、若不存在when 子句匹配, 应用else子句；若else 子句不存在, 返回null；

3、所有when或else子句定义的返回值类型相同；

4、不能应用在TSQL符合语句中。\*\*/

## Union, Intersect and Except

概述：在列方向上组合多个结果集为一个结果集，区别于JOIN，操作顺序“从上到下”，语法:

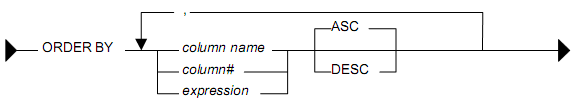


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| R1 | R2 | R1  Union  R2 | R1  Union all  R2 | R1  Intersect  R2 | R1  Intersect all  R2 | R1  Except  R2 | R1  Except all  R2 |
| A | A | A | A | A | A | E | A |
| A | A | B | A | B | A |  | C |
| A | B | C | A | C | B |  | C |
| B | B | D | A |  | B |  | E |
| B | B | E | A |  | C |  |  |
| C | C |  | B |  |  |  |  |
| C | D |  | B |  |  |  |  |
| C |  |  | B |  |  |  |  |
| E |  |  | B |  |  |  |  |
|  |  |  | B |  |  |  |  |
|  |  |  | C |  |  |  |  |
|  |  |  | C |  |  |  |  |
|  |  |  | C |  |  |  |  |
|  |  |  | C |  |  |  |  |
|  |  |  | D |  |  |  |  |
|  |  |  | E |  |  |  |  |

|  |  |
| --- | --- |
| **Usage** | **Remark** |
| R1 Union R2 | 包含R1与R2并集非重复行, 需要排序 |
| R2 Union All R2 | 包含R1与R2并集所有行 |
| R1 Intersect R2 | 包含R1与R2交集非重复行, 需要排序 |
| R1 Intersect All R2 | 包含R1与R2交集所有行 |
| R1 Except R2 | 包含存在R1但不存在R2的补集非重复行, 需要排序 |
| R1 Except All R2 | 包含R1存在但R2不存在的补集所有行 |

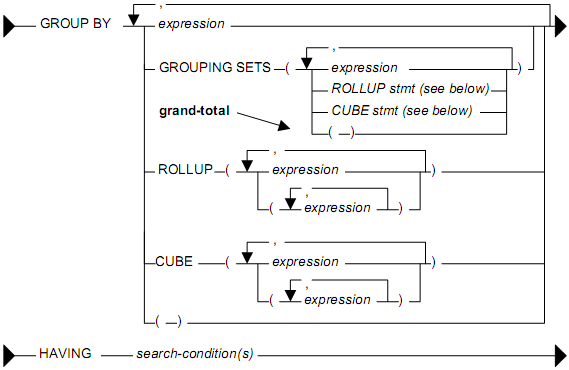
## Order By

概要: 对结果集进行按一或多个字段、表达式或序号进行排序，语法:



## Group By, and Having

概要: 对结果集进行分类汇总，分组字段值每种组合仅显示一行，where 谓词 (perdict) 应用在Group By之前， having谓词 (perdict) 应用Group By之后，语法:



### GROUPING SETS

|  |  |
| --- | --- |
| **GROUPING SETS ( )** | **Corresponding GROUP BY** |
| ***GROUPING SETS((), (), …)*** | ***GROUP BY ( ) UNION ALL***  ***GROUP BY ( ) UNION ALL …*** |
| GROUPING SETS((A, B, C)) | GROUP BY A, B, C |
| GROUPING SETS(A, (B, C)) | GROUP BY A UNION ALL  GROUP BY B, C |
| GROUPING SETS(A, B, C) | GROUP BY A UNION ALL  GROUP BY B UNION ALL  GROUP BY C |
| ***GROUPING SETS(),***  ***GROUPING SETS(), …*** | ***GROUP BY (), (), …*** |
| GROUPINGSETS(A),  GROUPING SETS(B),  GROUPING SETS(C) | GROUP BY A, B, C |
| GROUPING SETS(A),  GROUPING SET((B, C)) | GROUP BY A, B, C |
| GROUPING SETS(A),  GROUPING SETS(B, C) | GROUP BY A, B UNION ALL  GROUP BY A, C |
| ***GROUP BY (), GROUPING SETS()*** | ***GROUP BY (), ()*** |
| GROUP BY A, B,  GROUPING SETS((B, C)) | GROUP BY A, B, B, C |
| GROUP BY A, B, C |
| GROUP BY A, B,  GROUPING SETS(B, C) | GROUP BY A, B, B UNION ALL  GROUP BY A, B, C |
| GROUP BY A, B UNION ALL  GROUP BY A, B, C |

### ROLLUP( )

|  |  |
| --- | --- |
| **ROLLUP ( )** | **Corresponding GROUPING SETS ( )** |
| ROLLUP(A, B, C) | GROUPING SETS((A, B, C),(A, B),(A),()) |
| ROLLUP(A, (B, C)) | GROUPING SETS((A, B, C),(A),()) |
| ROLLUP(A), ROLLUP(B, C) | GROUPING SETS(A, ())\*  GROUPING SETS((B, C),(B),()) |
| GROUPING SETS((A, B, C),(A, B),(A),(B, C),(B),()) |

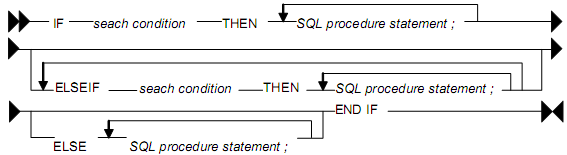
### CUBE( )

|  |  |
| --- | --- |
| **CUBE ( )** | **Corresponding GROUPING SETS ( )** |
| GROUP BY CUBE(A, B, C) | GROUPING SETS((A,B,C),(A,B),(A,C),(A),(B,C),(B),(C),()) |
| GROUP BY CUBE(A, (B, C)) | GROUPING SETS((A,B,C),(A),(B,C),()) |
| GROUP BY CUBE(A, B), CUBE(B, C) | GROUPING SETS (  (A, B, B, C), (A, B, B),(A, B, C), (A, B),  (A, B, C), (A, B), (A, C), (A),  (B, B, C), (B, B), (B, C) ,(B),  (B, C), (B), (C), ()  ) |
| GROUPING SETS (  (A, ,C), (A,B), (A,B,C), (A,B),  (A, B, C), (A, B), (A, C), (A),  (B, C), (B), (B, C), (B),  (B, C), (B), (C), ()  ) |

## Flow Control Statements

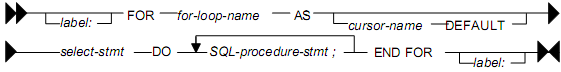
### IF

概述: 用于实现标准的if-then-else分支逻辑，语法:



### FOR

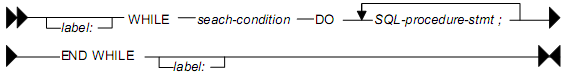
概述: 顺序遍历(处理)查询中的每一行，语法:



/\*\* 注意: 在Function中不能使用游标, 但可以使用FOR来替代 \*\*/

### WHILE

概述: 当条件满足时，循环执行一或多个语句，语法:



### ITERATE

概述: 返回循环标签的开始处，一般用于循环语句，如FOR, WHILE，语法:



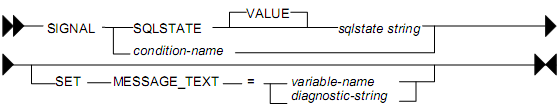
### LEAVE

概述: 退出标签对应的循环，执行循环体外的语句，语法:



### SIGNAL

概述: 抛出错误或警告，语法:



### GET DIAGNOSTICS

概述: 返回最近运行SQL Statements信息，包括:

1. DML语句影响行数
2. 调用过程返回状态值

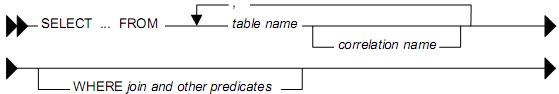
语法:



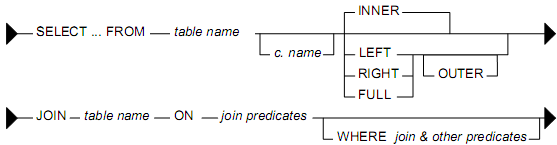
## JOIN

概述: “横向”连接多个结果集，而UNION ALL“纵向”连接结果集。

1. 仅适用于内连接(自然连接)，语法:

****

2、标准内连接或外连接(包括: 左外连接 / 右外连接 / 全连接)，语法:



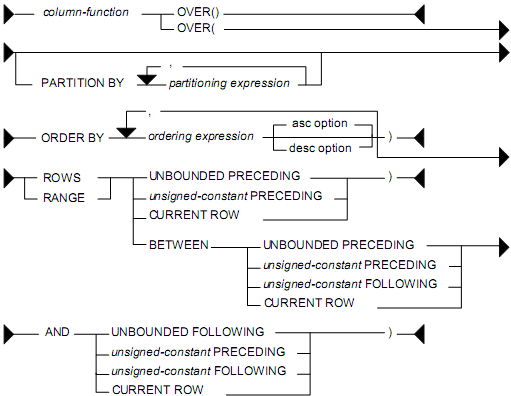
/\*\* 注意: 在外连接中ON谓词优先于WHERE谓词运行, 内连接中忽略顺序 \*\*/

## Query Sequence (查询检索顺序)

|  |  |  |
| --- | --- | --- |
| **Keywords** | **Seq** | **Function Description** |
| **FROM** Clause | First  End |  |
| **JOIN ON** Clause |  |
| **WHERE** Clause | 内连接(自然连接), WHERE与 JOIN 功能相同 |
| **GROUP BY** And Aggregate |  |
| **HAVING** Clause |  |
| **SELECT** List |  |
| **OLAP** Functions | 作为SELECT List 中的一部分 |
| **ORDER BY** Clause |  |
| **FETCH FIRST** |  |

## OLAP Functions (分析函数)

概述: OLAP (Online Analytical Processing) 函数用于计算基于组的某种聚合值，与聚合函数的不同之处是对于每个组返回多行，而聚合函数对于每个组只返回一行。语法:



|  |  |  |
| --- | --- | --- |
| **Keywords** | **Summary** | **Function Description** |
| Over | 关键字(必选) | 用于区别聚合函数与分析函数 |
| Partition By | 分析函数应用分区(可选) | 1. 若省略, 默认分析函数应用范围:   **Range** Between Ubounded Preceding  And Ubounded Following  2、分析函数在新分区操作前复位  3、功能类似Group By |
| Order By  [Asc / Desc]  [Nulls  First / Last] | 分析函数应用顺序(可选, 影响开窗函数应用范围) | 1. 若省略, 默认分析函数应用范围:每个分区中   **Range** Between Ubounded Preceding  And Ubounded Following   1. 若不指定Range或Rows，默认分析函数应用范围:   每个分区中**Range** Between Ubounded Preceding  And Current Row(包含与当前Range值相等的多行数据) |
| Windowing Clause | 分析函数应用范围(可选), 分为Range / Rows | 1. 按照分区(Partition By)、顺序(Order By)和分析语句(Range / Rows)，定义分析函数应用范围 2. Range仅适用于数字或日期字段(即可进行加减的字段), 相同Range包含多行 3. Rows可用于多种字段类型, 可为多列, 物理行序 |
| 例子: 得到本年、明年、后年对应的指标值  select year, idx, val,  min(year) over (partition by idx order by year nulls last range between 1 following and 1 following) as next1year,  min(val) over (partition by idx order by year nulls last range between 1 following and 1 following) as next1val,  min(year) over (partition by idx order by year nulls last range between 2 following and 2 following) as next2year,  min(val) over (partition by idx order by year nulls last range between 2 following and 2 following) as next2val,  min(year) over (partition by idx order by year nulls last range between 3 following and 3 following) as next3year,  min(val) over (partition by idx order by year nulls last range between 3 following and 3 following) as next3val  from table(values  (2001, '[1]', 1), (2001, '[2]', 1), (2001, '[3]', 1),  (2002, '[1]', 2), (2002, '[2]', 2),  (2003, '[1]', 3), (2003, '[3]', 3),  (2004, '[1]', 4), (2004, '[2]', 4)  ) as t(year, idx, val) | | |

### Main Olap Funcation Summary(主要Olap函数概要)

|  |  |  |
| --- | --- | --- |
| **OLAP Function** | **Description** | **Example** |
| AVG | 返回查询范围内非NULL表达式的平均值(等同于SUM() / COUNT()) | select  name, dept, salary, avg(salary) over (partition by dept order by salary) as avg\_salary, avg(salary) over (partition by dept) as total\_avg\_salary  from leo.staff |
| COUNT | 返回查询范围内非NULL表达式的计数值,可使用DISTINCT来去除范围内完全相同的数据出现的行数 | select  name, dept, salary, count(name) over (partition by dept order by salary) as num, count(name) over (partition by dept) as cnt  from leo.staff |
| DENSE\_RANK | 根据ORDER BY子句中表达式,对查询返回的每一行，计算它们与其它行的相对位置，组内的数据按ORDER BY子句排序后给每一行赋一个序号，从而形成一个序列。该序列从1开始，每次ORDER BY表达式的值发生变化，该序列随之增加。有同样值的行得到同样的序号(NULL都相等)，返回无间隔的序列，但同一序号可能存在多行, partition by / order by 不能同时省略 | select  name, dept, salary, rownumber() over (order by dept) as row\_num, dense\_rank() over (order by dept) as denserank\_num, rank() over (order by dept) as rank\_num  from leo.staff |
| RANK | 同DENSE\_RANK,但存在序号跳跃，如两行序号同为1，则没有序号2，序列将给组中的下一行分配序号3，partition by / order by 不能同时省略 |
| ROWNUMBER | 返回有序组中一行的偏移量，从而得到按特定标准排序的连续且不重复序(行)号 |
| MAX | 返回查询范围内非NULL表达式的最大值 | select  name, dept, salary, min(salary) over (partition by dept) as min\_salary, max(salary) over (partition by dept) as max\_salary  from leo.staff |
| MIN | 返回查询范围内非NULL表达式的最小值 |
| SUM | 返回查询范围内非NULL表达式的累积和 | select  name, dept, salary, sum(salary) over (partition by dept) as total\_salary, sum(salary) over (partition by dept order by salary) as acc\_salary  from leo.staff |

## GLOBAL TEMPORARY TABLE (全局临时表)

概述: 驻留在内存、非持久性、特定于会话的表中存储数据的表, 可在SQL 过程或任何可连接 DB2 的应用程序中创建和使用。在同一个session中一次“声明”多次“引用”, 建议使用过程来初始化和删除临时表。

**封装临时表声明的过程:**

CREATE PROCEDURE INIT\_TEMP()

SPECIFIC INIT\_TEMP

BEGIN

DECLARE GLOBAL TEMPORARY TABLE

SESSION.TEMP (id INT, data VARCHAR(10))

ON COMMIT PRESERVE ROWS;

END@

**封装删除临时表的过程:**

CREATE PROCEDURE CLOSE\_TEMP()

SPECIFIC CLOSE\_TEMP

BEGIN

DROP TABLE SESSION.TEMP;

END@

使用过程来初始化和删除临时表的主要优点:

1、过程开发人员不必为了得到临时表的 DDL 而找遍应用程序代码(可能是由其他人来维护的);

2、如果在应用程序代码中的多个位置实例化相同的临时表, 该表的定义可以集中在一个地方, 假如要求改变表结构, 就不必搜索所有的声明, 而只需在一个地方改变其定义;

3、通过存储过程声明完成临时表的封装。

/\*\* 注意:

1、要使用临时表, 数据库中必须存在一个用户临时表空间(默认情况下没有, 例:

create user temporary tablespace usertemp1 managed by system using ('usertemp1')

2、删除临时表并非是绝对必要的, 当应用程序断开连接时它们会被自动删除, 然而, 如果应用程序使用连接池, 在返回到连接池的连接前务必要删除临时表;

3、全局临时表在 DB2 UDB v8.1 中得到了显著的加强, 可创建"索引和生成统计结果";

4、临时表存储在用户临时表空间中, 如果使用大型数据集,可考虑专用的缓冲池来代替表空间,

通过表空间快照来决定是否将数据真正写入磁盘。\*\*/

# 附录:

## 参考文档

* 《DB2 v9 SQL Reference Volume 1》
* 《DB2 v9 SQL Reference Volume 2》
* 《DB2 v9 SQL Cookbook》 Author By Graeme Birchall