### extract，date\_format函数无效值问题：

**场景：**

select extract(week from ‘2021-0-1’);

 select date\_format(‘9999-0-12’, ‘%U %u %V %v %X %x %j’);

openGauss：



mysql:



由无效日期计算出的week或者dayofyear都是没有意义的。

**原因：**

openGauss的有关week函数运算的内部参数类型范围设置的比mysql的大：mysql是unsigned int，openGauss是long。将部分数据类型按照mysql修改之后，对于’2021-0-1’或者’2021-1-0’形式的无效日期，计算出的无效值可以和mysql一致。

![C:\Users\Wat\AppData\Roaming\Tencent\Users\793822091\QQ\WinTemp\RichOle\(LO@0FZOZ1OGMG]@ZRWPEHC.png]()

但对于’0000-0-1’这种年份和类型的日期，由于openGauss的底层日期编码函数date2j以及j2date和mysql有区别，所以计算出的无效值也不一致。

### time类型在函数中对字符串和浮点数的接收区别

**问题描述：**

time\_to\_sec或extract等函数在MySQL中参数为’8385959.1’时认为不合法，参数为8385959.1时可以正常执行。

**现象分析：**

MySQL中time\_to\_sec或extract等函数，将参数解析为time类型时，函数接收到字符串形式’8385959.1’会认为是不合法的；接收到数值形式的8385959.1会认为是合法的。

插表时，’8385959.1’和8385959.1都是不合法的，即只能插入[-838:59:59, 838:59:59]之间的值。

查看MySQL代码，发现字符串形式或者数值形式（带小数秒部分）输入的time，都应该经过范围[-838:59:59, 838:59:59]的校验。





MySQL表现

|  |  |
| --- | --- |
| time\_to\_sec(‘8385959.1’) | 认为非法（严格模式插表报错，非严格模式转为对应边界值，报warning） |
| time\_to\_sec(8385959.1) | 认为正常 |
| insert(‘8385959.1’)  | 认为非法（严格模式报错，非严格模式转为对应边界值，报warning） |
| insert(8385959.1) | 认为非法（严格模式报错，非严格模式转为对应边界值，报warning） |

openGauss中time类型判断范围都是一致的，即超过[-838:59:59, 8389:59:59]就为非法时间值。此处输入8385959.1或者’8385959.1’经过time类型解析函数解析后，均被认为是超范围的。在严格模式下报错，非严格模式下转为对应边界值，报warning。

openGauss表现

|  |  |
| --- | --- |
| time\_to\_sec(‘8385959.1’) | 认为非法（严格模式插表报错，非严格模式转为对应边界值，报warning） |
| time\_to\_sec(8385959.1) | 认为非法（严格模式插表报错，非严格模式转为对应边界值，报warning） |
| insert(‘8385959.1’) | 认为非法（严格模式报错，非严格模式转为对应边界值，报warning） |
| insert(8385959.1) | 认为非法（严格模式报错，非严格模式转为对应边界值，报warning） |

结论：

openGauss中time\_to\_sec或者extract等涉及时间解析的日期处理函数，解析时将‘8385959.1’和8385959.1都认为是越界的，严格模式报错，非严格模式报warning并将结果置为对应边界值。

### time解析函数在非严格模式的解析

MySQL中time解析函数在非严格模式下可以将类似’2021-11-12’的输入解析为’00:20:21’，目前有类似PR存在。



预计后续在此PR的基础上，针对一些函数非严格模式下time解析类似’2021-11-12’输入问题进行修复。

### 部分日期处理函数的结果不支持运算符。

**如date函数：**

**MySQL:**



**openGauss：**

 **严格模式：**

 ****

 **非严格模式：**

 **![C:\Users\Wat\AppData\Roaming\Tencent\Users\793822091\QQ\WinTemp\RichOle\Y7$]UUOL{(I@PP[U_WA@RFE.png]()**

**原因：**

openGauss中相关函数采用了text作为函数返回值类型，因此函数返回值进行运算，与日期或者时间类型不一致。以上演示的openGauss中的date函数后跟运算操作符，即为text类型运算符表现。

采用text作为函数返回值类型，原因有两个：

1. **为了使一些函数能够返回不正常日期。**

以date函数为例，若采用dolphin插件中的date类型作为返回值类型，则date函数在返回如’2021-0-1’或者’2021-1-0’形式的日期时，date类型底层值由date2j和j2date函数进行编码解码造成结果与预期不一致，但此时对正常的日期可以正常返回，且可以正常支持运算符。其表现如下：

月份或者天为0：



正常日期：



1. **为了使具有多种返回值类型的函数返回正确格式的内容。**

以str\_to\_date(str, format)函数为例,其返回值有date、datetime、time三种。其返回值类型由format字符串中包含的符号决定。

如果定义三个函数:

date str\_to\_date(str, format)

datetime str\_to\_date(str, format)

time str\_to\_date(str, format)

则这三个函数因为只有返回值类型不一样，所以不能被区分，函数注册时会失败：



因此注册一个返回值为text类型的str\_to\_date函数，在函数内部通过解析format得知返回类型type，根据返回类型type来选择不同类型的输出函数：

EncodeDateOnlyForBDatabase

EncodeDateTimeForBDatabase

EncodeTimeOnly

这样可以得到格式正确的输出内容。但是返回值依然是text类型，进行运算时依然是text类型的操作符表现。

### INTERVAL后不能跟表达式问题

目前进展：

按照专家建议以及功能要求，在AexprConst候选产生式处添加如下语法：



此时会有关键字、操作符等移进规约冲突，冲突的种类和数量都较多。



![C:\Users\Wat\AppData\Roaming\Tencent\Users\793822091\QQ\WinTemp\RichOle\1L$XS[]WS9T`I}~E~)Y4HC9.png]()

1. INTERVAL后跟表达式，其表达式计算过程需要建立一种新的NODE来实现。
2. INTERVAL与其他SQL操作也有联系。

如和各种时间类型的运算操作：



1. 与分区的一些内容也有关系。

![C:\Users\Wat\AppData\Roaming\Tencent\Users\793822091\QQ\WinTemp\RichOle\3MICXZF]Q0O$JO)D3HZQ5RH.png]()

INTERVAL后不能跟表达式问题较复杂，涉及语法重新构建，INTERVAL类型在openGauss中的各种使用场景等方面，此问题建议作为openGauss的INTERVAL类型的独立问题解决。