openGauss SQL兼容性需求check-in评审报告

评审纪要

时间：
特性名称：openGauss兼容mysql时间数据类型
特性责任人：梁宏达
评审人：

设计checklist

模块	类别	检查项	检查项详细说明	是否涉及	是否完成检查
GUC参数	功能	新增GUC参数适配项	①guc.cpp里需要新增guc参数的相应代码，包括参数的校验/赋值等功能函数； ②如果要加入postgres.conf，需要修改postgresql_single.conf.sample 路径：src/common/backend/utils/misc/ 一般以注释方式加入，如果非常确认使用某个默认值，也可以以非注释方式加入 ③如果要允许gs_guc工具设置，请修改cluster_guc.conf 路径：src/bin/gs_guc/cluster_guc.conf	否	是
升级	兼容性	版本号变更	src/common/backend/utils/init/globals.cpp：GRAND_VERSION_NUM	否	是
升级	兼容性	前向兼容性	1）涉及系统表修改的特性，必须提供系统表升级脚本和回滚脚本，修改版本号文件 2）对于涉及修改持久化数据（如日志）的特性，必须考虑新老版本共存时的兼容性场景，必要情况下，需要增加版本号以进行识别 3）对于涉及修改执行态数据格式（如syscache结构）的特性，必须考虑新老版本共存时的兼容性场景，必要情况下，需要增加版本号以进行识别	否	是
升级	兼容性	验证升级正常	涉及到升级时需要验证如下场景： 1. 集群环境安装：该步骤会验证集群环境安装是否正常 2. 集群环境升级：从老的版本（1.1.0版本）升级到最新版本是否正常	否	是
公共数据结构	功能	hashtable	hashtable使用注意事项 ①在使用hash表时，entry必须封装成一个结构体，第一个成员必须是hash key 否则会造成查找失败！！ ②封装的结构体必须只有两个成员：key+value，即value需要封装起来 否则hash表在search的时候会core！！ ③hash表在创建的时候，entry 的size一定要正确 否则在enter的时候会造成写越界，不会报错但是其它变量内存会被踩，不易发现！！	否	是
公共数据结构	功能	禁止使用commandTag判断Query的类型	需要对SELECT\INSERT\UPDATE\DELETE\MERGE INTO等query类型进行识别判断时，应该使用CmdType进行判断或新增标识。 禁止使用commandTag作为判断标识，因为commandTag在带returning*语句执行中变化。	否	是
系统函数	兼容性	系统表升级是否正确	通过对比initdb和升级后的db中pg_proc每一列的值，避免出现升级和非升级不一致的情况	否	是

模块	类别	检查项	检查项详细说明	是否涉及	是否完成检查
系统函数	兼容性	新增数据类型适配FENCED函数	新增数据类型如果是FENCED函数使用，需要修改UDFArgumentHandler函数，把数据类型添加到对应收发分支。	否	是
系统函数	功能	NULL值处理	对于非strict函数允许接收NULL入参，在函数逻辑中应对NULL值进行细致的处理	否	是
前向兼容性	兼容性	Node相关数据结构删减	禁止对Node相关数据结构（如Query、RangeTblEntry）数据成员删减	否	是
兼容性	兼容性	兼容性设计原则	目前支持兼容性PG,A,B,C;对于每一个兼容点要求：1、如果该兼容点为非冲突兼容点，应该是各个兼容模式下均支持，不需要开关控制 2、如果该兼容性点为冲突兼容点，应该由兼容开关控制。	否	是
查询解析	兼容性	添加关键字场景	为了尽可能的防止移进规约冲突，最大化的用关键字作为对象名，关键字分有4类：非保留关键字、列名关键字、类型函数关键字、保留关键字。他们的约束逐渐加强，比如保留关键字不可以用做表名。添加关键字需要考虑如下： 1. 添加新的关键字最好先作为非保留关键字添加 2. 如果要加其他关键字类型那么需要考虑是否有前向兼容问题。比如排查客户场景是否有用该关键字作为对象名。 3. 关键字还需要加入kwlist.h文件中。	否	是
查询解析	兼容性	Gram.y中处理语法冲突	1、合理使用a_expr/b_expr/c_expr使用场景 为了尽可能防止冲突，划分了a_expr/b_expr/c_expr，比如BETWEEN a_expr AND b_expr中 AND属于a_expr范畴，b_expr中不含AND。否则就无法区分BETWEEN之后是Bool判断还是范围。c_expr是a_expr和b_expr的交集。所以 尽量用a_expr这样支持的范围最广。如果用b_expr/c_expr 那需要考虑潜在的用户问题场景（参考BETWEEN...AND...例子）并在手册中写清楚约束。	否	是
其他	功能	新增已有的数据结构成员适配序列化与反序列化	注意适配： src/common/backend/nodes/copyfuncs.cpp src/common/backend/nodes/equalfuncs.cpp src/common/backend/nodes/outfuncs.cpp src/common/backend/nodes/readfuncs.cpp	否	是

模块	类别	检查项	检查项详细说明	是否涉及	是否完成检查
公共子系统	内存定位定界	局部变量内存释放，非程序结尾位置，释放进行置空；全局内存变量释放，无论什么位置均需要置空	防止内存使用出现use-after-free模式，内存越界	否	是
公共子系统	内存定位定界	对于使用数组、下标访问内存操作场景，是否提前判断下标符合内存空间要求	防止出现heap overflow场景	是	是
公共子系统	内存定位定界	内存分配报错日志中，是否需要新增信息，便于分析报错原因	内存报错场景下，快速定位内存报错原因	否	是
资源负载管理	兼容性	只能新增系统表参数，不可以删除已有的参数	前向兼容	否	是
资源负载管理	功能	新增系统表参数的默认初始化数值要合理，不影响已有的功能	前向兼容	否	是
HA用例	功能	ha check	1.HA相关设计实现需要严格系统的测试。 2.需要保证最简单的功能验证，包括搭建主备环境，确认主备可以正常同步，以及跑ha check用例，确认所有用例可以跑过。 3.需要完成各种故障场景的验证，设计详细的故障库，确保覆盖各个场景。	否	是
安全	未公开接口	是否新增或修改函数、视图、系统表	新增或修改函数、视图、系统表需刷新产品文档，考虑权限控制	是	是

模块	类别	检查项	检查项详细说明	是否涉及	是否完成检查
安全	未公开接口	是否新增SQL语法	新增SQL语法需刷新产品文档，支持记录审计日志	否	是

特性LLT测试方案及验证结论

基本测试

date 类型测试

```
-- 'YYYY-MM-DD' 'YY-MM-DD'
SELECT date'2001-01-01';
SELECT date'999-01-01';
SELECT date'1-01-01';
SELECT date'10120-01-01';
SELECT date'2000-2-29';
SELECT date'2001-2-29';
SELECT date'2100-2-29';
SELECT date'0000-2-29';
SELECT date'01-01';
SELECT date'9-01-01';
SELECT date'99-01-01';
SELECT date'999-01-01';
SELECT date'1000-01-1';

-- 'YYYYMMDD' or 'YYMMDD'
SELECT date'99991231';
SELECT date'100000101';
SELECT date'10000101';
SELECT date'9990101';
SELECT date'701010';
SELECT date'691010';
SELECT date'691310';
SELECT date'691131';
SELECT date'10101';
SELECT date'10130';

-- YYYYMMDD or YYYYMMDD
SELECT 99990101::date;
SELECT 990101::date;
SELECT 100::date;
SELECT 101::date;
SELECT 10228::date;
SELECT 10229::date;
SELECT 991231::date;
```

```
SELECT 99991231::date;
SELECT 21000228::date;
SELECT 21000229::date;
SELECT 100000000::date;
```

time 类型测试

```
SELECT time(7)'12:12:12.123456';
SELECT time(6)'12:12:12.123456';
SELECT time(5)'12:12:12.123456';
SELECT time(4)'12:12:12.123456';
SELECT time(3)'12:12:12.123456';
SELECT time(2)'12:12:12.123456';
SELECT time(1)'12:12:12.123456';
SELECT time(0)'12:12:12.123456';
SELECT time'12:12:12.123456';
SELECT time'34 22:59:59.999999';
SELECT time'12:60:12.123456';
SELECT time'12:12:60.123456';
SELECT time'34 23:00:00';
SELECT time'-34 23:00:00';
SELECT time'34 22:59:59';
SELECT time'-34 22:59:59';
SELECT time'34 22:59:59.999999';
SELECT time'34 22:59:59.999999999999999999999';
SELECT time'-34 22:59:59.999999999999999999999';
SELECT time'-34 22:59:59.999999999999999999999';
SELECT time'838:59:59';
SELECT time'839:00:00';
SELECT time'59:59';
SELECT time'59:59.123';
SELECT time'59';
SELECT time'59.12';

-- 'hhmmss.fsec'
SELECT time(1)'1.123456';
SELECT time(2)'12.123456';
SELECT time(3)'123.123456';
SELECT time(4)'1234.123456';
SELECT time(5)'12345.123456';
SELECT time'123456.123456';

-- hhmmss.fsec
SELECT 2::time;
SELECT 20::time;
SELECT 201::time;
SELECT 2011::time;
SELECT 2115.12::time;
SELECT 20115.23::time;
SELECT 233445.123::time;
SELECT 233445.123::time;
SELECT 8385959.999999::time;
SELECT 8385959.9999999::time;
```

datetime 类型测试

```
-- 'YYYY-MM-DD hh:mm:ss.fsec' or 'YY-MM-DD hh:mm:ss.fsec'
SELECT datetime(2)'2001-01-01  23:59:59.999999';
SELECT datetime(3)'2001-01-01  23:59:59.9999999';
SELECT datetime(4)'9999-12-31  23:59:59.9999999';
SELECT datetime(5)'999-12-31  23:59:59.9999999';
SELECT datetime'99-01-01  23:59:59.123';
SELECT datetime'2001-2-28  23:59:59.123';
SELECT datetime'2000-2-28  23:59:59.123';

-- 'YYYYMMDDhhmmss.fsec' or 'YYMMDDhhmmss.fsec'
SELECT datetime(1)'20010101010101.1278';
SELECT datetime(1)'010101010101.1234';
SELECT datetime(2)'0101010101';
SELECT datetime(1)'01010101';
SELECT datetime(3)'0101017';
SELECT datetime(1)'010101';
SELECT datetime(1)'0101';

-- YYYYMMDDhhmmss or YYMMDDhhmmss
SELECT 11::datetime;
SELECT 111::datetime;
SELECT 1111::datetime;
SELECT 11111::datetime;
SELECT 111111::datetime;
SELECT 1111111::datetime;
SELECT 11111111::datetime;
SELECT 111111111::datetime;
SELECT 1111111111::datetime;
SELECT 11111111111::datetime;
SELECT 111111111111::datetime;
SELECT 1111111111111::datetime;
SELECT 11111111111111::datetime;

-- test datetime operation
create table test_dt(dt datetime);
insert into test_dt values('1997-11-10');
insert into test_dt values('2000-11-10');
insert into test_dt values('1997-11-10');
insert into test_dt values('2012-11-10');
insert into test_dt values('2008-11-10');
insert into test_dt values('1994-11-10');
select max(dt) from test_dt;
select min(dt) from test_dt;
select distinct(dt) from test_dt;
select * from test_dt;
create index on test_dt(dt);
select * from test_dt where dt > '2007-01-01';
select * from test_dt where dt = '1997-11-10';
select * from test_dt where dt < '1997-11-10';
```

timestamp 类型测试

```
set time zone 'PRC';
CREATE TABLE timestamp_tbl (v timestamp(2));

-- 'YYYY-MM-DD hh:mm:ss.fsec' or 'YY-MM-DD hh:mm:ss.fsec'
INSERT INTO timestamp_tbl VALUES ('2001-01-01  23:59:59.999999');
INSERT INTO timestamp_tbl VALUES ('2001-01-01  23:59:59.9999999');
INSERT INTO timestamp_tbl VALUES ('9999-12-31  23:59:59.9999999');
INSERT INTO timestamp_tbl VALUES ('999-12-31  23:59:59.9999999');
INSERT INTO timestamp_tbl VALUES ('99-01-01  23:59:59.123');
INSERT INTO timestamp_tbl VALUES ('2001-2-28  23:59:59.123');
INSERT INTO timestamp_tbl VALUES ('2000-2-28  23:59:59.123');

-- 'YYYYMMDDhhmmss.fsec' or 'YYMMDDhhmmss.fsec'
INSERT INTO timestamp_tbl VALUES ('20010101010101.1278');
INSERT INTO timestamp_tbl VALUES ('010101010101.1234');
INSERT INTO timestamp_tbl VALUES ('0101010101');
INSERT INTO timestamp_tbl VALUES ('01010101');
INSERT INTO timestamp_tbl VALUES ('0101017');
INSERT INTO timestamp_tbl VALUES ('010101');
INSERT INTO timestamp_tbl VALUES ('0101');

-- YYYYMMDDhhmmss or YYMMDDhhmmss
INSERT INTO timestamp_tbl VALUES (11::timestamp);
INSERT INTO timestamp_tbl VALUES (111::timestamp);
INSERT INTO timestamp_tbl VALUES (1111::timestamp);
INSERT INTO timestamp_tbl VALUES (11111::timestamp);
INSERT INTO timestamp_tbl VALUES (111111::timestamp);
INSERT INTO timestamp_tbl VALUES (1111111::timestamp);
INSERT INTO timestamp_tbl VALUES (11111111::timestamp);
INSERT INTO timestamp_tbl VALUES (111111111::timestamp);
INSERT INTO timestamp_tbl VALUES (1111111111::timestamp);
INSERT INTO timestamp_tbl VALUES (11111111111::timestamp);
INSERT INTO timestamp_tbl VALUES (111111111111::timestamp);
INSERT INTO timestamp_tbl VALUES (1111111111111::timestamp);
INSERT INTO timestamp_tbl VALUES (11111111111111::timestamp);

SELECT * FROM timestamp_tbl;
set time zone 'UTC';
SELECT * FROM timestamp_tbl;
set time zone 'PRC';

DROP TABLE timestamp_tbl;
```

year 类型

```
DROP TABLE IF EXISTS year_tbl;

-- test invalid display width
CREATE TABLE year_tbl (y YEAR(-1));
CREATE TABLE year_tbl (y YEAR(0));
CREATE TABLE year_tbl (y YEAR(1));
CREATE TABLE year_tbl (y YEAR(3));
```

```
CREATE TABLE year_tbl (y YEAR(5));
CREATE TABLE year_tbl (y YEAR(4294967297));

-- test YEAR YEAR(2) YEAR(4)
CREATE TABLE year_tbl (y YEAR, y4 YEAR(4), y2 YEAR(2));
INSERT INTO year_tbl VALUES (1900, 1900, 1900);
INSERT INTO year_tbl VALUES ('1900', '1900', '1900');
INSERT INTO year_tbl VALUES (2156, 2156, 2156);
INSERT INTO year_tbl VALUES ('2156', '2156', '2156');

INSERT INTO year_tbl VALUES (1901, 1901, 1901);
INSERT INTO year_tbl VALUES ('1901', '1901', '1901');
INSERT INTO year_tbl VALUES (2155, 2155, 2155);
INSERT INTO year_tbl VALUES ('2155', '2155', '2155');

INSERT INTO year_tbl VALUES (70, 70, 70);
INSERT INTO year_tbl VALUES ('70', '70', '70');
INSERT INTO year_tbl VALUES (69, 69, 69);
INSERT INTO year_tbl VALUES ('69', '69', '69');
INSERT INTO year_tbl VALUES (2156, 2156, 2156);
INSERT INTO year_tbl VALUES ('2156', '2156', '2156');
INSERT INTO year_tbl VALUES (-1, -1, -1);
INSERT INTO year_tbl VALUES ('-1', '-1', '-1');

INSERT INTO year_tbl VALUES (0, 0, 0);
INSERT INTO year_tbl VALUES ('0', '0', '0');
INSERT INTO year_tbl VALUES ('00', '00', '00');
INSERT INTO year_tbl VALUES ('000', '000', '000');
INSERT INTO year_tbl VALUES ('0000', '0000', '0000');
INSERT INTO year_tbl VALUES (1, 1, 1);
INSERT INTO year_tbl VALUES ('1', '1', '1');

SELECT * FROM year_tbl;
-- test order by
SELECT * FROM year_tbl ORDER BY y ASC;
-- test max, min, distinct
SELECT max(y), min(y4) FROM year_tbl;
SELECT distinct(y) FROM year_tbl;
-- test index
CREATE INDEX test_b_tree_idx ON year_tbl(y);
DROP INDEX test_b_tree_idx;

DROP TABLE year_tbl;

-- test operation of YEAR and YEAR(2)
SELECT ('2010')::YEAR(2)<(2001)::YEAR;
SELECT ('2010')::YEAR(2)>(2001)::YEAR;
SELECT ('2010')::YEAR(2)>(2001);
SELECT (2010)>(2001)::YEAR;
SELECT (2001)=(2001)::YEAR;
SELECT (2001)=(2001)::YEAR(2);

-- test operation of year and YEAR(2)
SELECT (2010)::YEAR-(2001)::YEAR;
SELECT (2010)::YEAR-(2111)::YEAR;
SELECT (2010)::YEAR-(2111)::YEAR;

SELECT (2010)::YEAR-(interval '2' year);
```

```
SELECT (2010)::YEAR+(interval '2' year);

SELECT (1970)::YEAR-(interval '69' year);
SELECT (1970)::YEAR-(interval '70' year);
SELECT (2069)::YEAR+(interval '86' year);
SELECT (2069)::YEAR+(interval '87' year);

SELECT (2069)::YEAR+(interval'366day');

SELECT (2010)::YEAR(2)-(2001)::YEAR(2);
SELECT (2010)::YEAR(2)-(2111)::YEAR;
SELECT (2010)::YEAR-(2111)::YEAR(2);

SELECT (2010)::YEAR(2)-(interval '2' year);
SELECT (2010)::YEAR(2)+(interval '2' year);

SELECT (1970)::YEAR(2)-(interval '69' year);
SELECT (1970)::YEAR(2)-(interval '70' year);
SELECT (2069)::YEAR(2)+(interval '86' year);
SELECT (2069)::YEAR(2)+(interval '87' year);

SELECT (2069)::YEAR(2)+(interval'366day');
```

回归测试验证

测试集	结论
fastcheck	通过
memcheck	通过
hacheck	无

Fastcheck

```
============== running regression test queries        ==============
test builtin_funcs/conv       .... ok
test builtin_funcs/crc32      .... ok
test builtin_funcs/db_b_format .... ok
test builtin_funcs/db_b_hex   .... ok
test builtin_funcs/db_b_if    .... ok
test db_b_new_gram_test       .... ok
test db_b_parser1             .... ok
test db_b_parser3             .... ok
test db_b_parser4             .... ok
test db_b_plpgsql_test        .... ok
test empty_value_support_value .... ok
test join_without_on          .... ok
test nvarchar                 .... ok
test regexp                   .... ok
test test_binary              .... ok
test test_blob                .... ok
test test_datatype            .... ok
test test_fixed               .... ok
test ast                      .... ok
test float_numeric_test/db_b_log_test .... ok
test float_numeric_test/db_b_sqrt_test .... ok
test mysqlmode_fullgroup      .... ok
test mysqlmode_strict         .... ok
test string_func_test/db_b_ascii_test .... ok
test string_func_test/db_b_left_right_test .... ok
test string_func_test/db_b_quote_test .... ok
test string_func_test/db_b_string_length_test .... ok
test string_func_test/db_b_substr_test .... ok
test string_func_test/db_b_trim_test .... ok
test b_compatibility_time_type .... ok
test empty_value_list         .... ok
test empty_value_lists        .... ok
test db_b_parser2             .... ok
```

Memcheck

```
> hongda > memchk > asan >  runlog.15502
  ------------------------------------------------------
Suppressions used:
  count      bytes template
    135       7055 regression_main
  ------------------------------------------------------
```

```
home > hongda > memchk > asan >  runlog.13580
1   ------------------------------------------------------
2   Suppressions used:
3     count      bytes template
4       3         44 regression_main
5   ------------------------------------------------------
6
```

hacheck

无

代码检视结论

编码规范检查

编程规范

已按照 pr 中提出的修改意见修改

PclintPlus告警

内存使用排查

覆盖率

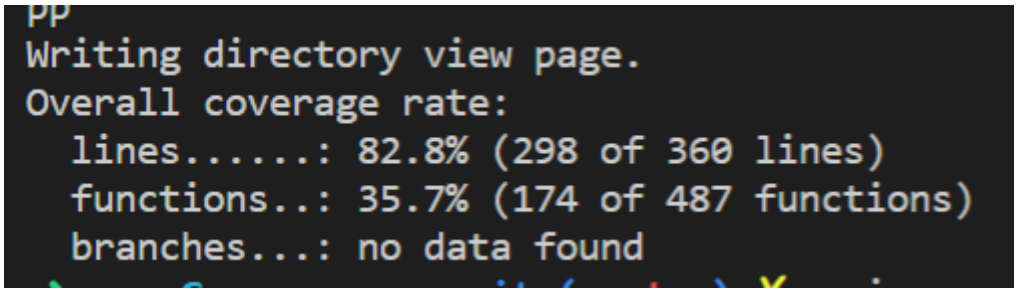
增量代码测试用例覆盖率80%+

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	Lines:	298	360	82.8 %
Test:	increment.info	Functions:	174	487	35.7 %
Date:	2022-06-14 15:49:14	Branches:	0	0	-

Directory	Line Coverage		Functions		Branches	
/opt/software/openGauss/openGauss-server/contrib/dolphin/plugin_parser	100.0 %	2 / 2	44.9 %	31 / 69	-	0 / 0
/opt/software/openGauss/openGauss-server/contrib/dolphin/plugin_utils/adt	82.7 %	296 / 358	34.2 %	143 / 418	-	0 / 0

Generated by: LCOV version 1.13



鲲鹏平台乱序排查

编号	排查依据	排查结果
1	**是否为多线程并发场景?**	否
2	**是否涉及线程间共享数据?**	否
3	**是否未对共享数据加锁保护?**	否
4	线程间共享数据是否存在无锁同步模式?	否
5	识别是否为典型使用模式?	否
6	是否 **未** 在正确的位置插入合适的内存屏障	否

代码检视意见

提出人	意见	答复详情
zhangzhixian	此处是更改了原有内置函数的调用方法的名称吗？如果是，builtin_funcs.ini是否也需要做相应修改？	应该是不需要的：原来的函数 timestamp_in 在内核代码中已存在，实际上插件中的 timestamp_in 函数是不必要的，这里直接将插件的 timestamp_in 函数进行修改，变成 datetime 类型的输出处理函数 datetime_in
仲夏十三	为什么要修改以前的测试用例 timestamp 不支持原来的使用方式了么	兼容前，timestamp 类型为 PG 中不带时区属性的时间戳；兼容后，datetime 类型为不带时区属性的时间戳，timestamp 类型为带时区属性的时间戳，因此原本以前的样例 timestamp 应该改为使用 datetime
仲夏十三	请删掉create extension dolphin现在插件是自动加载	在新提交的 pr 已经更新
熊小军	1、考虑各时间类型的默认值 2、考虑各类型带小数时的精度范围是否和M*保持一致 3、timestamp类型请在文档中说明会影响之前用户应用的行为，时区。。。	1.mysql 时间类型的默认空值为 NULL(select 语句对应的列输出为 NULL)，mysql 的默认空值也是 NULL(select 语句对应得列输出为空白，含义应该也是 NULL) 2.所有兼容后的时间数据类型精度范围和mysql一样 3.在新提交的 pr 中已经增加用户文档，同时说明 timestamp 兼容问题的情况
仲夏十三	++cp 保持格式一致 不要一个先加后赋值，另一个先赋值后加	在新提交的 pr 已经更新
pengjiong	为了确保宏的正确使用， h/m/s/fsec 这些参数在使用时应该加上括号，避免副作用	已修改
zhangzhixian	请添加相应的描述信息，或关联issue，方便检视者理解背景	已关联相关 issue，其中包含项目目标

遗留问题

测试建议