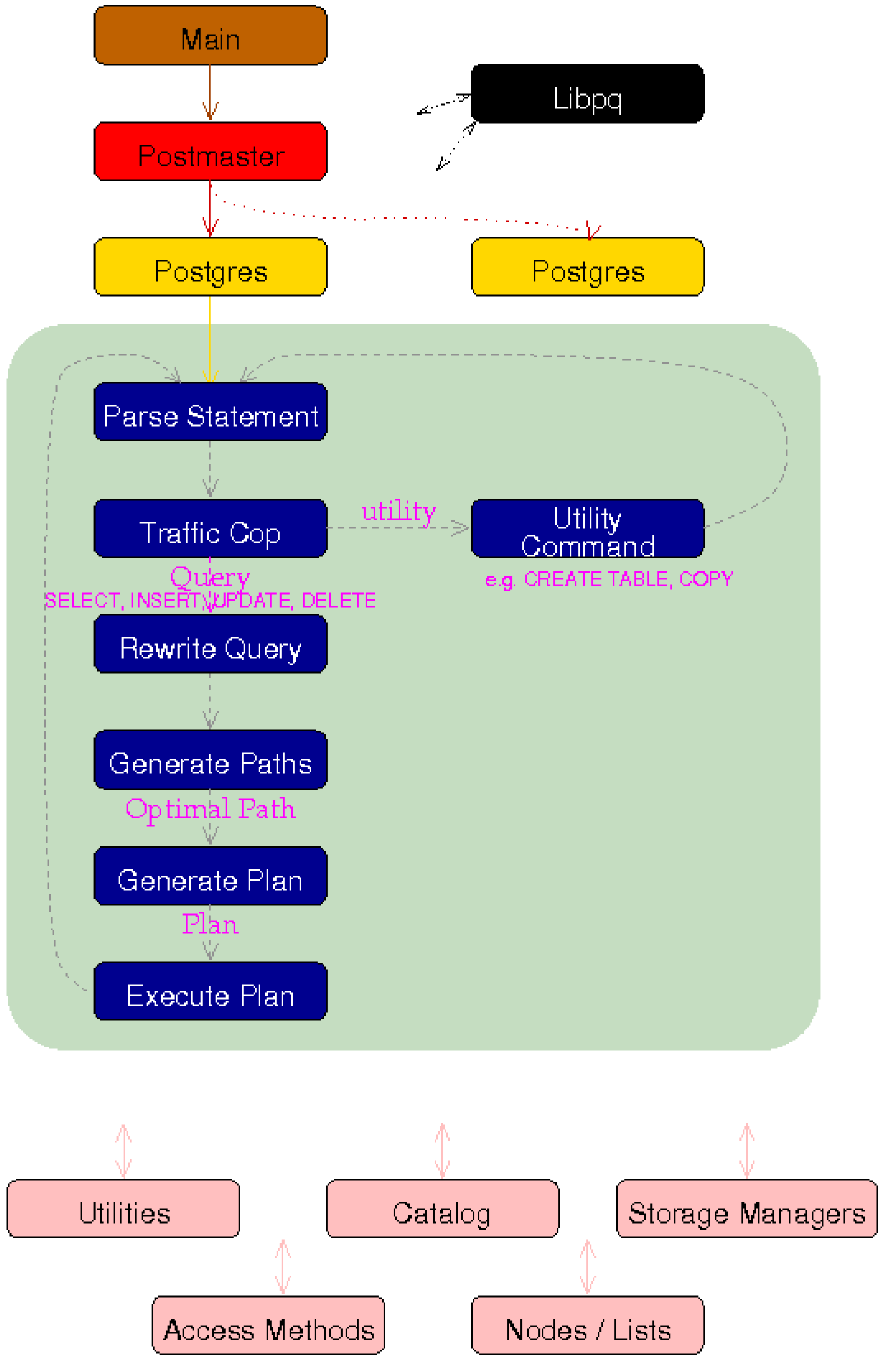
PostgreSQL 内核扩展入门培训

digoal

目录

PostgreSQL flow chart

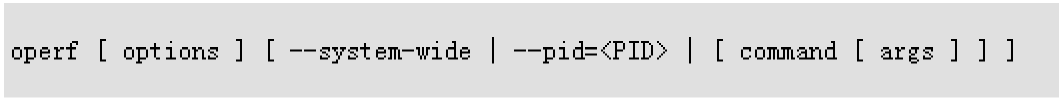
- https://wiki.postgresql.org/wiki/Backend_flowchart
- https://www.postgresql.org/developer/backend/



bootstrap : 初始化数据库集群
main : 程序入口
postmaster : 监听， fork
libpq : 通信库
tcop : 解包，分发请求到适当的模块， backend process入口
parser : 词法分析(输出优化器或执行器需要的结构)
rewrite : view, rule
optimizer : 基于执行计划优化算法，生成执行树
executor : 执行QUERY
commands : DDL, DCL相关
catalog : 元数据
access : 索引、堆表、事务相关
(common, gin, gist, hash, heap, index, nbtree, spgist, transam)
(公共代码, 索引, 堆表, 事务)
storage : 存储接口
(buffer, file, freespace, ipc, large_object, lmgr, page, smgr)
(缓存, 文件, FSM, 内部进程通信, 大对象, 锁, 页, 磁盘)
utils : 工具包
(adt build-in数据类型相关, cache 元数据, 函数, 类型等缓存)
(error, fmgr, hash, init, mb, misc, mmgr, resowner, sort, time)
(错误处理, 内部/外部自定义函数接口,内部公用的hash算法库
如扫描cache, 初始化数据库, 多字节字符, context内存管理,
资源owner跟踪, 内部公用排序算法库,
MVCC相关row可见性管理)
include, lib, snowball, tsearch :
port : 平台兼容性相关
regex : 正则
replication : 流复制相关

如何分析代码瓶颈

- OProfile
  - http://oprofile.sourceforge.net/

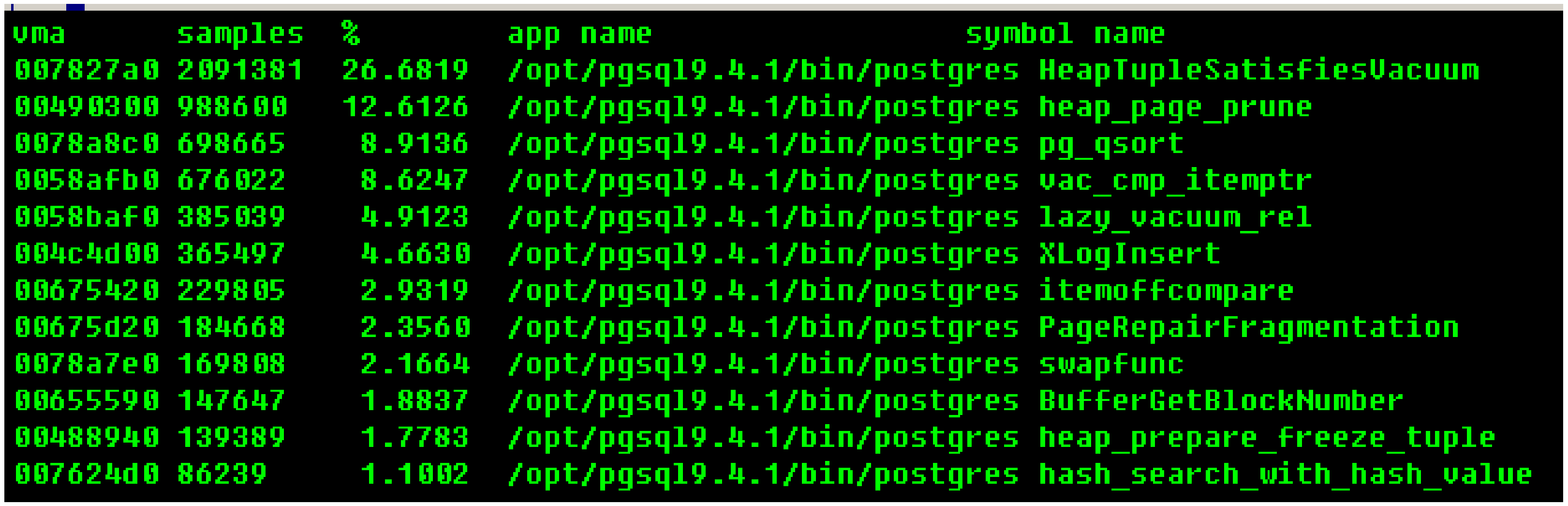


```
operf [ options ] [ --system-wide | --pid=<PID> | [ command [ args ] ] ]
```

  - mkdir /tmp/optest ; cd /tmp/optest ; operf -l postgres -B 2GB -c port=$i -c listen_addresses='0.0.0.0' -c synchronous_commit=off -c full_page_writes=off -c wal_buffers=1900MB -c wal_writer_delay=10ms -c max_connections=100 -c max_wal_size=4GB -c log_destination='csvlog' -c logging_collector=on -D $PGDATA -k $PGDATA
  - 进行一些压测
- perf
  - perf top

如何分析代码瓶颈

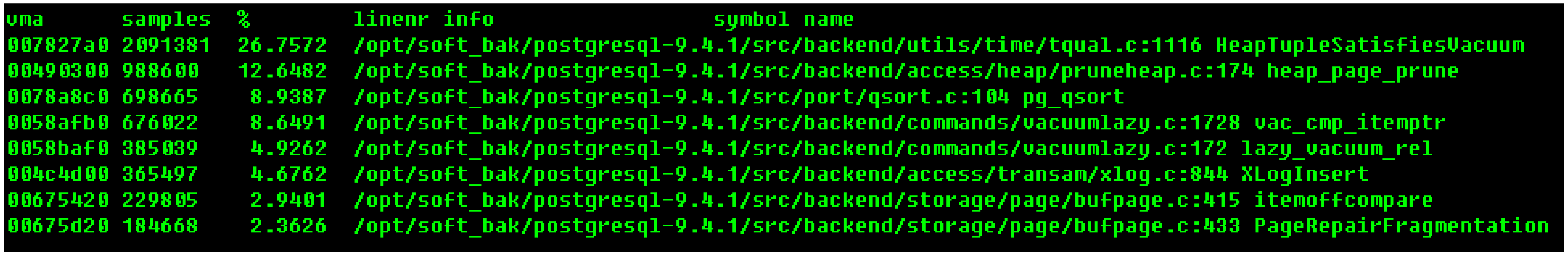
- OProfile
- cd /tmp/optest; opreport -l -f -w -x -t 1



```
vma      samples  %        app name                        symbol name
007827a0 2091381  26.6819  /opt/pgsql9.4.1/bin/postgres HeapTupleSatisfiesVacuum
00490300 988600   12.6126  /opt/pgsql9.4.1/bin/postgres heap_page_prune
0078a8c0 698665    8.9136  /opt/pgsql9.4.1/bin/postgres pg_qsort
0058afb0 676022    8.6247  /opt/pgsql9.4.1/bin/postgres vac_cmp_itemptr
0058baf0 385039    4.9123  /opt/pgsql9.4.1/bin/postgres lazy_vacuum_rel
004c4d00 365497    4.6630  /opt/pgsql9.4.1/bin/postgres XLogInsert
00675420 229805    2.9319  /opt/pgsql9.4.1/bin/postgres itemoffcompare
00675d20 184668    2.3560  /opt/pgsql9.4.1/bin/postgres PageRepairFragmentation
0078a7e0 169808    2.1664  /opt/pgsql9.4.1/bin/postgres swapfunc
00655590 147647    1.8837  /opt/pgsql9.4.1/bin/postgres BufferGetBlockNumber
00488940 139389    1.7783  /opt/pgsql9.4.1/bin/postgres heap_prepare_freeze_tuple
007624d0 86239     1.1002  /opt/pgsql9.4.1/bin/postgres hash_search_with_hash_value
```

如何分析代码瓶颈

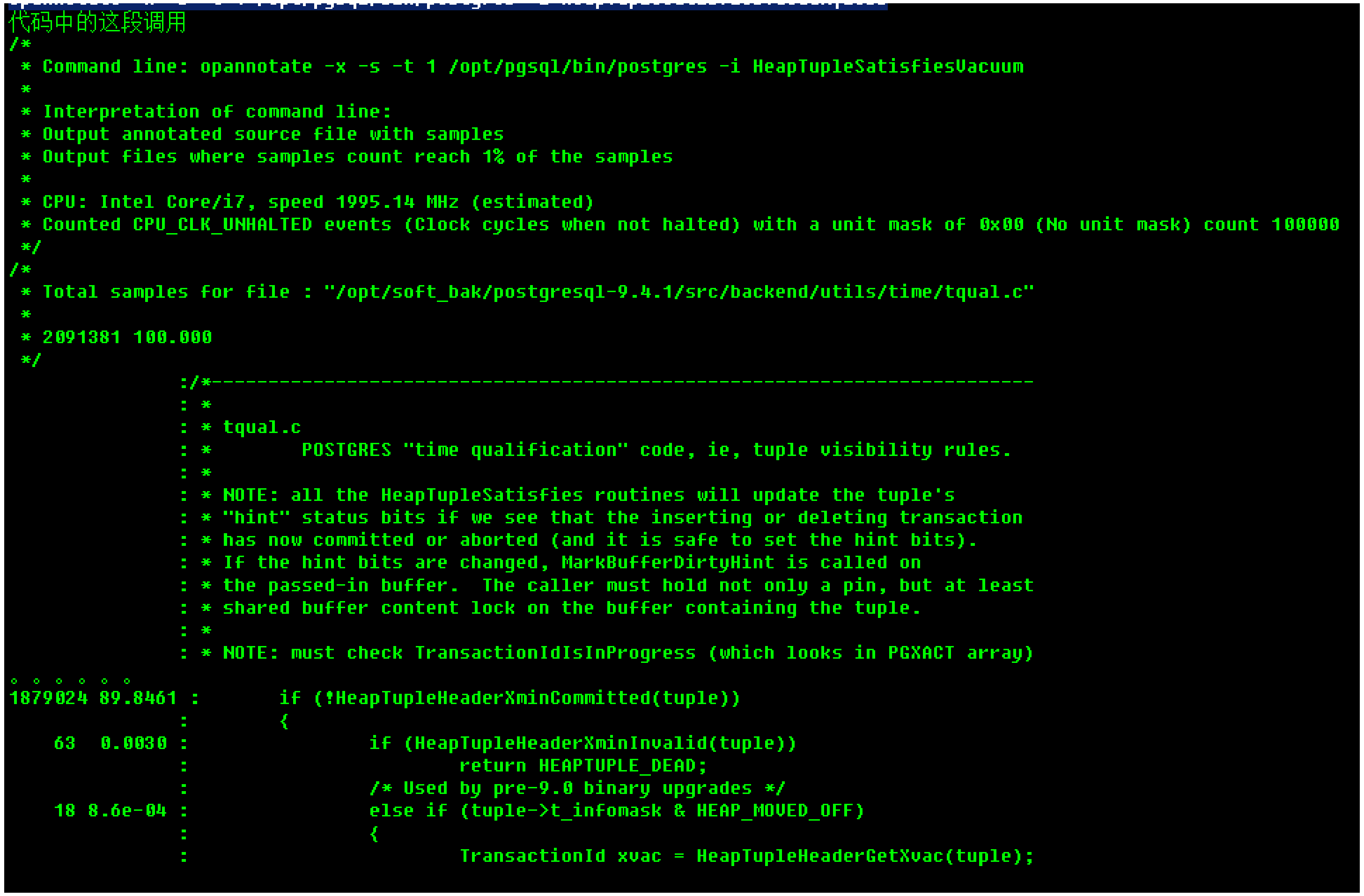
- OProfile
- opreport -l -f -g -w -x -t 1 /opt/pgsql/bin/postgres



```
vma      samples  %        linenr info                symbol name
007827a0 2091381  26.7572  /opt/soft_bak/postgresql-9.4.1/src/backend/utils/time/tqual.c:1116 HeapTupleSatisfiesVacuum
00490300 988600   12.6482  /opt/soft_bak/postgresql-9.4.1/src/backend/access/heap/pruneheap.c:174 heap_page_prune
0078a8c0 698665    8.9387  /opt/soft_bak/postgresql-9.4.1/src/port/qsort.c:104 pg_qsort
0058afb0 676022    8.6491  /opt/soft_bak/postgresql-9.4.1/src/backend/commands/vacuumlazy.c:1728 vac_cmp_itemptr
0058baf0 385039    4.9262  /opt/soft_bak/postgresql-9.4.1/src/backend/commands/vacuumlazy.c:172 lazy_vacuum_rel
004c4d00 365497    4.6762  /opt/soft_bak/postgresql-9.4.1/src/backend/access/transam/xlog.c:844 XLogInsert
00675420 229805    2.9401  /opt/soft_bak/postgresql-9.4.1/src/backend/storage/page/bufpage.c:415 itemoffcompare
00675d20 184668    2.3626  /opt/soft_bak/postgresql-9.4.1/src/backend/storage/page/bufpage.c:433 PageRepairFragmentation
```

如何分析代码瓶颈

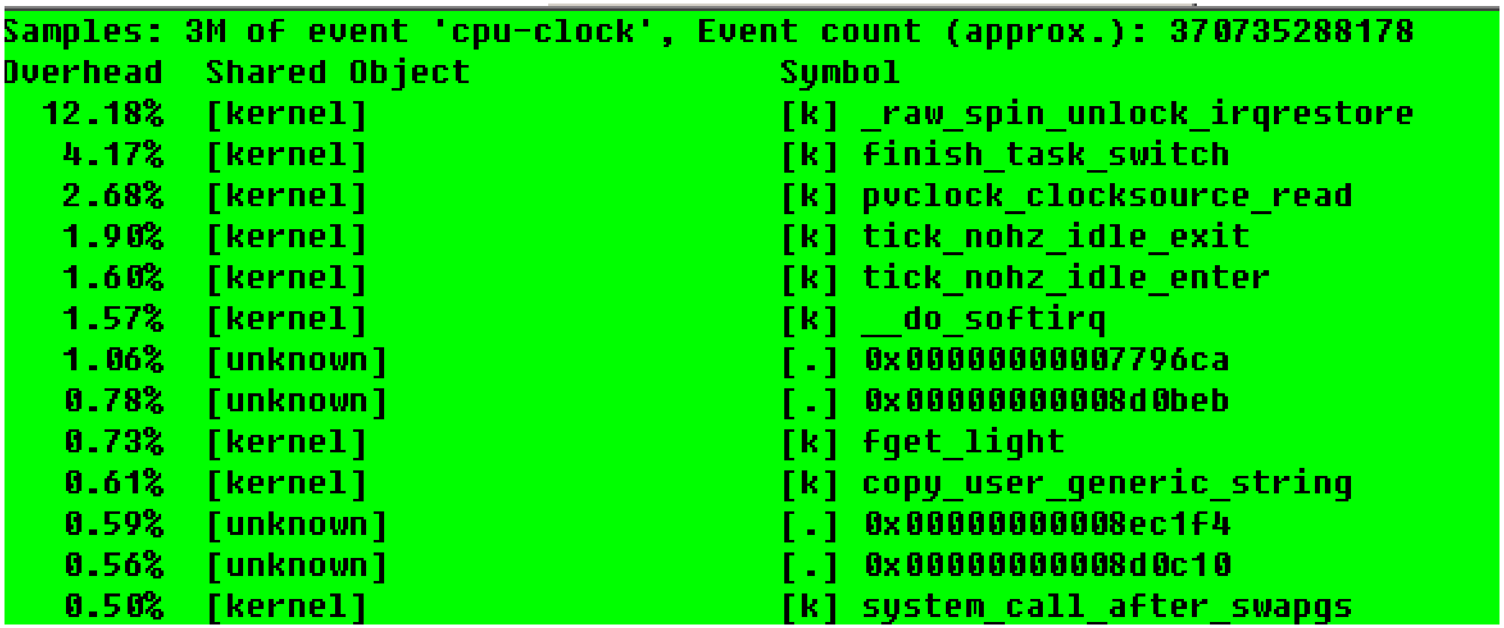
- OProfile
- opannotate -x -s -t 1 /opt/pgsql/bin/postgres -i HeapTupleSatisfiesVacuum|less



```
代码中的这段调用
/*
 * Command line: opannotate -x -s -t 1 /opt/pgsql/bin/postgres -i HeapTupleSatisfiesVacuum
 *
 * Interpretation of command line:
 * Output annotated source file with samples
 * Output files where samples count reach 1% of the samples
 *
 * CPU: Intel Core/i7, speed 1995.14 MHz (estimated)
 * Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00 (No unit mask) count 100000
 */
/*
 * Total samples for file : "/opt/soft_bak/postgresql-9.4.1/src/backend/utils/time/tqual.c"
 *
 * 2091381 100.000
 */
               :/*-------------------------------------------------------------------------
               : *
               : * tqual.c
               : *       POSTGRES "time qualification" code, ie, tuple visibility rules.
               : *
               : * NOTE: all the HeapTupleSatisfies routines will update the tuple's
               : * "hint" status bits if we see that the inserting or deleting transaction
               : * has now committed or aborted (and it is safe to set the hint bits).
               : * If the hint bits are changed, MarkBufferDirtyHint is called on
               : * the passed-in buffer.  The caller must hold not only a pin, but at least
               : * shared buffer content lock on the buffer containing the tuple.
               : *
               : * NOTE: must check TransactionIdIsInProgress (which looks in PGXACT array)
。。。。。。。
1879024 89.8461 :       if (!HeapTupleHeaderXminCommitted(tuple))
               :       {
    63  0.0030 :               if (HeapTupleHeaderXminInvalid(tuple))
               :                       return HEAPTUPLE_DEAD;
               :               /* Used by pre-9.0 binary upgrades */
    18 8.6e-04 :               else if (tuple->t_infomask & HEAP_MOVED_OFF)
               :               {
               :                       TransactionId xvac = HeapTupleHeaderGetXvac(tuple);
```

如何分析代码瓶颈

- perf top



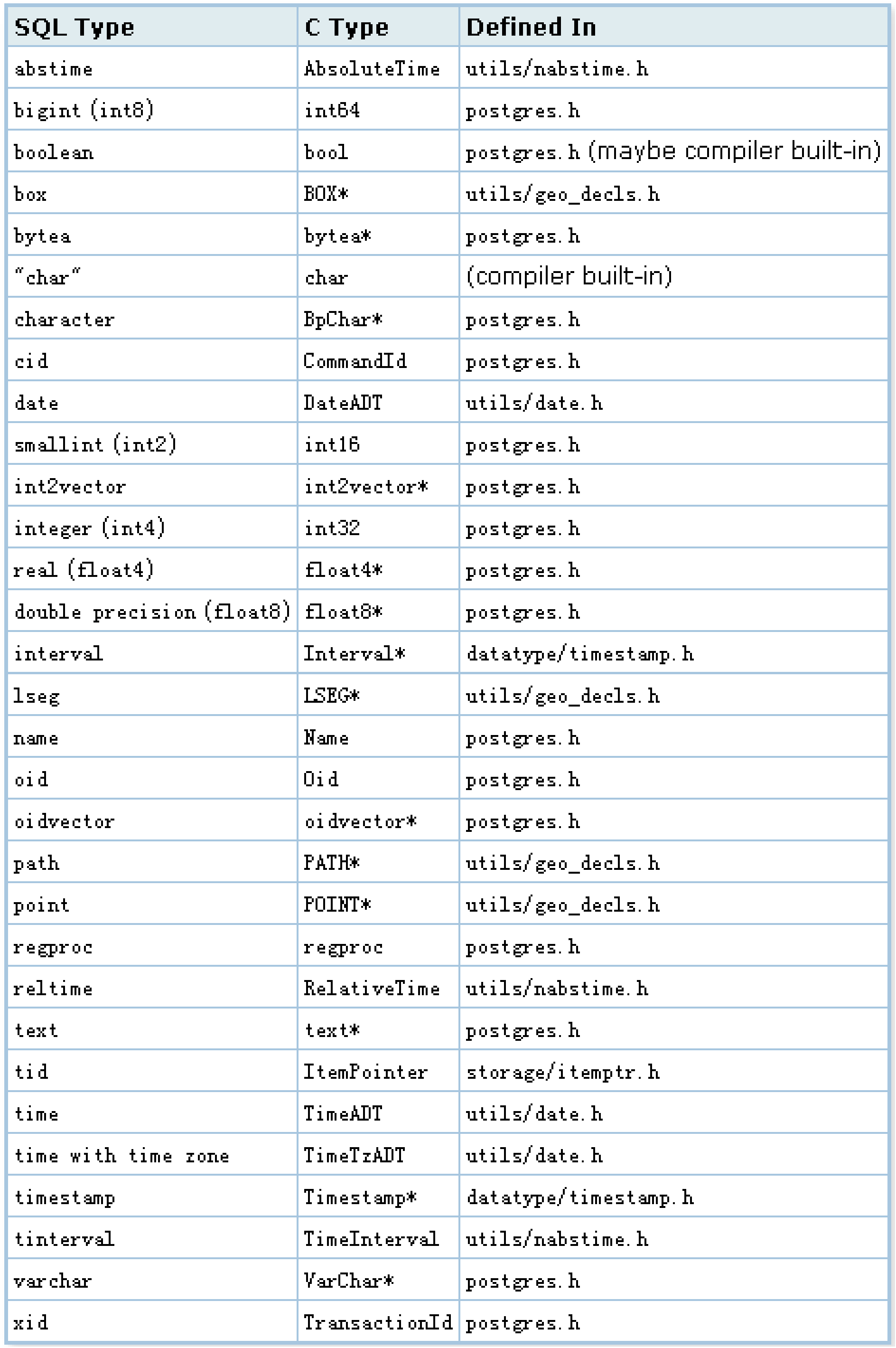
```
Samples: 3M of event 'cpu-clock', Event count (approx.): 370735288178
Overhead  Shared Object                         Symbol
  12.18%  [kernel]                              [k] _raw_spin_unlock_irqrestore
   4.17%  [kernel]                              [k] finish_task_switch
   2.68%  [kernel]                              [k] pvclock_clocksource_read
   1.90%  [kernel]                              [k] tick_nohz_idle_exit
   1.60%  [kernel]                              [k] tick_nohz_idle_enter
   1.57%  [kernel]                              [k] __do_softirq
   1.06%  [unknown]                             [.] 0x00000000007796ca
   0.78%  [unknown]                             [.] 0x00000000008d0beb
   0.73%  [kernel]                              [k] fget_light
   0.61%  [kernel]                              [k] copy_user_generic_string
   0.59%  [unknown]                             [.] 0x00000000008ec1f4
   0.56%  [unknown]                             [.] 0x00000000008d0c10
   0.50%  [kernel]                              [k] system_call_after_swapgs
```

如何自定义C UDF

- C与SQL的类型对应关系
- 如何获取SQL函数的args
- 如何从C函数返回结果给SQL函数
- 示例
- 自定义普通UDF
- 自定义聚合函数
- 自定义窗口函数

- SQL
- -
- C
- 类型
- 对应
- 关系

- 头文件
- 不一定准确



SQL Type	C Type	Defined In
abstime	AbsoluteTime	utils/nabstime.h
bigint (int8)	int64	postgres.h
boolean	bool	postgres.h (maybe compiler built-in)
box	BOX*	utils/geo_decls.h
bytea	bytea*	postgres.h
"char"	char	(compiler built-in)
character	BpChar*	postgres.h
cid	CommandId	postgres.h
date	DateADT	utils/date.h
smallint (int2)	int16	postgres.h
int2vector	int2vector*	postgres.h
integer (int4)	int32	postgres.h
real (float4)	float4*	postgres.h
double precision (float8)	float8*	postgres.h
interval	Interval*	datatype/timestamp.h
lseg	LSEG*	utils/geo_decls.h
name	Name	postgres.h
oid	Oid	postgres.h
oidvector	oidvector*	postgres.h
path	PATH*	utils/geo_decls.h
point	POINT*	utils/geo_decls.h
regproc	regproc	postgres.h
reltime	RelativeTime	utils/nabstime.h
text	text*	postgres.h
tid	ItemPointer	storage/itemptr.h
time	TimeADT	utils/date.h
time with time zone	TimeTzADT	utils/date.h
timestamp	Timestamp*	datatype/timestamp.h
tinterval	TimeInterval	utils/nabstime.h
varchar	VarChar*	postgres.h
xid	TransactionId	postgres.h

获取args macro(Datum -> c type)

- contrib/*/*.h 插件新建类型相关 arg MACRO
  - contrib/hstore/hstore.h:#define PG_GETARG_HS(x) DatumGetHStoreP(PG_GETARG_DATUM(x))
  - contrib/cube/cubedata.h:#define PG_GETARG_NDBOX(x) DatumGetNDBOX(PG_GETARG_DATUM(x))
  - contrib/ltree/ltree.h:#define PG_GETARG_LTREE(x) ((ltree*)DatumGetPointer(PG_DETOAST_DATUM(PG_GETARG_DATUM(x))))
  - contrib/ltree/ltree.h:#define PG_GETARG_LTREE_COPY(x) ((ltree*)DatumGetPointer(PG_DETOAST_DATUM_COPY(PG_GETARG_DATUM(x))))
  - contrib/ltree/ltree.h:#define PG_GETARG_LQUERY(x) ((lquery*)DatumGetPointer(PG_DETOAST_DATUM(PG_GETARG_DATUM(x))))
  - contrib/ltree/ltree.h:#define PG_GETARG_LQUERY_COPY(x) ((lquery*)DatumGetPointer(PG_DETOAST_DATUM_COPY(PG_GETARG_DATUM(x))))
  - contrib/ltree/ltree.h:#define PG_GETARG_LTXTQUERY(x) ((ltxtquery*)DatumGetPointer(PG_DETOAST_DATUM(PG_GETARG_DATUM(x))))
  - contrib/ltree/ltree.h:#define PG_GETARG_LTXTQUERY_COPY(x) ((ltxtquery*)DatumGetPointer(PG_DETOAST_DATUM_COPY(PG_GETARG_DATUM(x))))
  - contrib/isn/isn.h:#define PG_GETARG_EAN13(n) PG_GETARG_INT64(n)
  - contrib/intarray/_int.h:#define PG_GETARG_QUERYTYPE_P(n) DatumGetQueryTypeP(PG_GETARG_DATUM(n))
  - contrib/intarray/_int.h:#define PG_GETARG_QUERYTYPE_P_COPY(n) DatumGetQueryTypePCopy(PG_GETARG_DATUM(n))

获取args macro(Datum -> c type)

- 全文检索相关类型 arg MACRO
  - src/include/tsearch/ts_utils.h:#define PG_GETARG_TSQUERYSIGN(n) DatumGetTSQuerySign(PG_GETARG_DATUM(n))
  - src/include/tsearch/ts_type.h:#define PG_GETARG_TSVECTOR(n) DatumGetTSVector(PG_GETARG_DATUM(n))
  - src/include/tsearch/ts_type.h:#define PG_GETARG_TSVECTOR_COPY(n) DatumGetTSVectorCopy(PG_GETARG_DATUM(n))
  - src/include/tsearch/ts_type.h:#define PG_GETARG_TSQUERY(n) DatumGetTSQuery(PG_GETARG_DATUM(n))
  - src/include/tsearch/ts_type.h:#define PG_GETARG_TSQUERY_COPY(n) DatumGetTSQueryCopy(PG_GETARG_DATUM(n))

获取args macro(Datum -> c type)

- 相关类型 arg MACRO
  - src/include/utils/timestamp.h:#define PG_GETARG_TIMESTAMP(n) DatumGetTimestamp(PG_GETARG_DATUM(n))
  - src/include/utils/timestamp.h:#define PG_GETARG_TIMESTAMPTZ(n) DatumGetTimestampTz(PG_GETARG_DATUM(n))
  - src/include/utils/timestamp.h:#define PG_GETARG_INTERVAL_P(n) DatumGetIntervalP(PG_GETARG_DATUM(n))
  - src/include/utils/timestamp.h:#define PG_GETARG_TIMESTAMP(n) DatumGetTimestamp(PG_GETARG_DATUM(n))
  - src/include/utils/timestamp.h:#define PG_GETARG_TIMESTAMPTZ(n) DatumGetTimestampTz(PG_GETARG_DATUM(n))
  - src/include/utils/timestamp.h:#define PG_GETARG_INTERVAL_P(n) DatumGetIntervalP(PG_GETARG_DATUM(n))
  - src/include/utils/nabstime.h:#define PG_GETARG_ABSOLUTETIME(n) DatumGetAbsoluteTime(PG_GETARG_DATUM(n))
  - src/include/utils/nabstime.h:#define PG_GETARG_RELATIVETIME(n) DatumGetRelativeTime(PG_GETARG_DATUM(n))
  - src/include/utils/nabstime.h:#define PG_GETARG_TIMEINTERVAL(n) DatumGetTimeInterval(PG_GETARG_DATUM(n))
  - src/include/utils/date.h:#define PG_GETARG_DATEADT(n)   DatumGetDateADT(PG_GETARG_DATUM(n))
  - src/include/utils/date.h:#define PG_GETARG_TIMEADT(n)   DatumGetTimeADT(PG_GETARG_DATUM(n))
  - src/include/utils/date.h:#define PG_GETARG_TIMETZADT_P(n) DatumGetTimeTzADTP(PG_GETARG_DATUM(n))
  - src/include/utils/xml.h:#define PG_GETARG_XML_P(n)   DatumGetXmlP(PG_GETARG_DATUM(n))
  - src/include/utils/varbit.h:#define PG_GETARG_VARBIT_P(n)   DatumGetVarBitP(PG_GETARG_DATUM(n))
  - src/include/utils/varbit.h:#define PG_GETARG_VARBIT_P_COPY(n) DatumGetVarBitPCopy(PG_GETARG_DATUM(n))
  - src/include/utils/uuid.h:#define PG_GETARG_UUID_P(X)   DatumGetUUIDP(PG_GETARG_DATUM(X))
  - src/include/utils/numeric.h:#define PG_GETARG_NUMERIC(n)   DatumGetNumeric(PG_GETARG_DATUM(n))
  - src/include/utils/numeric.h:#define PG_GETARG_NUMERIC_COPY(n) DatumGetNumericCopy(PG_GETARG_DATUM(n))
  - src/include/utils/acl.h:#define PG_GETARG_ACLITEM_P(n)   DatumGetAclItemP(PG_GETARG_DATUM(n))
  - src/include/utils/acl.h:#define PG_GETARG_ACL_P(n)   DatumGetAclP(PG_GETARG_DATUM(n))
  - src/include/utils/acl.h:#define PG_GETARG_ACL_P_COPY(n)   DatumGetAclPCopy(PG_GETARG_DATUM(n))

- 相关类型 arg MACRO
  - src/include/utils/geo_decls.h:#define PG_GETARG_POINT_P(n) DatumGetPointP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_LSEG_P(n) DatumGetLsegP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_PATH_P(n)   DatumGetPathP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_PATH_P_COPY(n) DatumGetPathPCopy(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_LINE_P(n) DatumGetLineP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_BOX_P(n) DatumGetBoxP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_POLYGON_P(n)   DatumGetPolygonP(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_POLYGON_P_COPY(n) DatumGetPolygonPCopy(PG_GETARG_DATUM(n))
  - src/include/utils/geo_decls.h:#define PG_GETARG_CIRCLE_P(n) DatumGetCircleP(PG_GETARG_DATUM(n))
  - src/include/utils/inet.h:#define PG_GETARG_INET_P(n) DatumGetInetP(PG_GETARG_DATUM(n))
  - src/include/utils/inet.h:#define PG_GETARG_INET_PP(n) DatumGetInetPP(PG_GETARG_DATUM(n))
  - src/include/utils/inet.h:#define PG_GETARG_MACADDR_P(n) DatumGetMacaddrP(PG_GETARG_DATUM(n))
  - src/include/utils/array.h:#define PG_GETARG_ARRAYTYPE_P(n)   DatumGetArrayTypeP(PG_GETARG_DATUM(n))
  - src/include/utils/array.h:#define PG_GETARG_ARRAYTYPE_P_COPY(n) DatumGetArrayTypePCopy(PG_GETARG_DATUM(n))
  - src/include/utils/array.h:#define PG_GETARG_EXPANDED_ARRAY(n) DatumGetExpandedArray(PG_GETARG_DATUM(n))
  - src/include/utils/array.h:#define PG_GETARG_EXPANDED_ARRAYX(n, metacache) \
  - src/include/utils/array.h: DatumGetExpandedArrayX(PG_GETARG_DATUM(n), metacache)
  - src/include/utils/array.h:#define PG_GETARG_ANY_ARRAY(n)   DatumGetAnyArray(PG_GETARG_DATUM(n))
  - src/include/utils/pg_lsn.h:#define PG_GETARG_LSN(n)   DatumGetLSN(PG_GETARG_DATUM(n))
  - src/include/utils/cash.h:#define PG_GETARG_CASH(n)   DatumGetCash(PG_GETARG_DATUM(n))
  - src/include/utils/rangetypes.h:#define PG_GETARG_RANGE(n) DatumGetRangeType(PG_GETARG_DATUM(n))
  - src/include/utils/rangetypes.h:#define PG_GETARG_RANGE_COPY(n) DatumGetRangeTypeCopy(PG_GETARG_DATUM(n))
  - src/include/utils/jsonb.h:#define PG_GETARG_JSONB(x)   DatumGetJsonb(PG_GETARG_DATUM(x))

获取args macro(Datum -> c type)

- 相关类型 arg MACRO
  - src/include/fmgr.h:#define PG_GETARG_DATUM(n)   (fcinfo->arg[n])
  - src/include/fmgr.h:#define PG_GETARG_INT32(n)   DatumGetInt32(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_UINT32(n) DatumGetUInt32(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_INT16(n)   DatumGetInt16(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_UINT16(n) DatumGetUInt16(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_CHAR(n)   DatumGetChar(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BOOL(n)   DatumGetBool(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_OID(n)   DatumGetObjectId(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_POINTER(n) DatumGetPointer(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_CSTRING(n) DatumGetCString(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_NAME(n)   DatumGetName(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_FLOAT4(n) DatumGetFloat4(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_FLOAT8(n) DatumGetFloat8(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_INT64(n)   DatumGetInt64(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_RAW_VARLENA_P(n)   ((struct varlena *) PG_GETARG_POINTER(n))
  - src/include/fmgr.h:#define PG_GETARG_VARLENA_P(n) PG_DETOAST_DATUM(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_VARLENA_PP(n) PG_DETOAST_DATUM_PACKED(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BYTEA_P(n)   DatumGetByteaP(PG_GETARG_DATUM(n))

- 相关类型 arg MACRO
  - src/include/fmgr.h:#define PG_GETARG_BYTEA_PP(n) DatumGetByteaPP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_TEXT_P(n) DatumGetTextP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_TEXT_PP(n) DatumGetTextPP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BPCHAR_P(n) DatumGetBpCharP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BPCHAR_PP(n) DatumGetBpCharPP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_VARCHAR_P(n) DatumGetVarCharP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_VARCHAR_PP(n) DatumGetVarCharPP(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_HEAPTUPLEHEADER(n) DatumGetHeapTupleHeader(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BYTEA_P_COPY(n) DatumGetByteaPCopy(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_TEXT_P_COPY(n) DatumGetTextPCopy(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BPCHAR_P_COPY(n) DatumGetBpCharPCopy(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_VARCHAR_P_COPY(n) DatumGetVarCharPCopy(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_HEAPTUPLEHEADER_COPY(n) DatumGetHeapTupleHeaderCopy(PG_GETARG_DATUM(n))
  - src/include/fmgr.h:#define PG_GETARG_BYTEA_P_SLICE(n,a,b) DatumGetByteaPSlice(PG_GETARG_DATUM(n),a,b)
  - src/include/fmgr.h:#define PG_GETARG_TEXT_P_SLICE(n,a,b) DatumGetTextPSlice(PG_GETARG_DATUM(n),a,b)
  - src/include/fmgr.h:#define PG_GETARG_BPCHAR_P_SLICE(n,a,b) DatumGetBpCharPSlice(PG_GETARG_DATUM(n),a,b)
  - src/include/fmgr.h:#define PG_GETARG_VARCHAR_P_SLICE(n,a,b) DatumGetVarCharPSlice(PG_GETARG_DATUM(n),a,b)

返回结果 macro(c type -> Datum)

- contrib/cube/cubedata.h:#define PG_RETURN_NDBOX(x)   PG_RETURN_POINTER(x)
- contrib/isn/isn.h:#define PG_RETURN_EAN13(x) PG_RETURN_INT64(x)
- src/include/access/gin.h:#define PG_RETURN_GIN_TERNARY_VALUE(x) return GinTernaryValueGetDatum(x)
- src/include/tsearch/ts_utils.h:#define PG_RETURN_TSQUERYSIGN(X) return TSQuerySignGetDatum(X)
- src/include/tsearch/ts_type.h:#define PG_RETURN_TSVECTOR(x)   return TSVectorGetDatum(x)
- src/include/tsearch/ts_type.h:#define PG_RETURN_TSQUERY(x)   return TSQueryGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_TIMESTAMP(x) return TimestampGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_TIMESTAMPTZ(x) return TimestampTzGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_INTERVAL_P(x) return IntervalPGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_TIMESTAMP(x) return TimestampGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_TIMESTAMPTZ(x) return TimestampTzGetDatum(x)
- src/include/utils/timestamp.h:#define PG_RETURN_INTERVAL_P(x) return IntervalPGetDatum(x)

返回结果 macro(c type -> Datum)

- src/include/utils/nabstime.h:#define PG_RETURN_ABSOLUTETIME(x) return AbsoluteTimeGetDatum(x)
- src/include/utils/nabstime.h:#define PG_RETURN_RELATIVETIME(x) return RelativeTimeGetDatum(x)
- src/include/utils/nabstime.h:#define PG_RETURN_TIMEINTERVAL(x) return TimeIntervalGetDatum(x)
- src/include/utils/date.h:#define PG_RETURN_DATEADT(x)   return DateADTGetDatum(x)
- src/include/utils/date.h:#define PG_RETURN_TIMEADT(x)   return TimeADTGetDatum(x)
- src/include/utils/date.h:#define PG_RETURN_TIMETZADT_P(x) return TimeTzADTPGetDatum(x)
- src/include/utils/xml.h:#define PG_RETURN_XML_P(x)   PG_RETURN_POINTER(x)
- src/include/utils/varbit.h:#define PG_RETURN_VARBIT_P(x)   return VarBitPGetDatum(x)
- src/include/utils/uuid.h:#define PG_RETURN_UUID_P(X)   return UUIDPGetDatum(X)
- src/include/utils/numeric.h:#define PG_RETURN_NUMERIC(x)   return NumericGetDatum(x)
- src/include/utils/acl.h:#define PG_RETURN_ACLITEM_P(x)   PG_RETURN_POINTER(x)
- src/include/utils/acl.h:#define PG_RETURN_ACL_P(x)   PG_RETURN_POINTER(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_POINT_P(x) return PointPGetDatum(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_LSEG_P(x) return LsegPGetDatum(x)

返回结果 macro(c type -> Datum)

- src/include/utils/geo_decls.h:#define PG_RETURN_PATH_P(x)		return PathPGetDatum(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_LINE_P(x) return LinePGetDatum(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_BOX_P(x) return BoxPGetDatum(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_POLYGON_P(x)		return PolygonPGetDatum(x)
- src/include/utils/geo_decls.h:#define PG_RETURN_CIRCLE_P(x) return CirclePGetDatum(x)
- src/include/utils/inet.h:#define PG_RETURN_INET_P(x) return InetPGetDatum(x)
- src/include/utils/inet.h:#define PG_RETURN_MACADDR_P(x) return MacaddrPGetDatum(x)
- src/include/utils/array.h:#define PG_RETURN_ARRAYTYPE_P(x)	PG_RETURN_POINTER(x)
- src/include/utils/array.h:#define PG_RETURN_EXPANDED_ARRAY(x) PG_RETURN_DATUM(EOHPGetRWDatum(&(x)->hdr))
- src/include/utils/pg_lsn.h:#define PG_RETURN_LSN(x)	return LSNGetDatum(x)
- src/include/utils/cash.h:#define PG_RETURN_CASH(x)	return CashGetDatum(x)
- src/include/utils/rangetypes.h:#define PG_RETURN_RANGE(x)		return RangeTypeGetDatum(x)
- src/include/utils/jsonb.h:#define PG_RETURN_JSONB(x)	PG_RETURN_POINTER(x)

返回结果 macro(c type -> Datum)

- /* To return a NULL do this: */
- #define PG_RETURN_NULL() \
-         do { fcinfo->isnull = true; return (Datum) 0; } while (0)
- src/include/fmgr.h:#define PG_RETURN_VOID()   return (Datum) 0
- src/include/fmgr.h:#define PG_RETURN_DATUM(x)   return (x)
- src/include/fmgr.h:#define PG_RETURN_INT32(x)   return Int32GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_UINT32(x) return UInt32GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_INT16(x)   return Int16GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_UINT16(x) return UInt16GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_CHAR(x)   return CharGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_BOOL(x)   return BoolGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_OID(x)   return ObjectIdGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_POINTER(x) return PointerGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_CSTRING(x) return CStringGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_NAME(x)   return NameGetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_FLOAT4(x) return Float4GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_FLOAT8(x) return Float8GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_INT64(x)   return Int64GetDatum(x)
- src/include/fmgr.h:#define PG_RETURN_BYTEA_P(x)   PG_RETURN_POINTER(x)
- src/include/fmgr.h:#define PG_RETURN_TEXT_P(x)   PG_RETURN_POINTER(x)
- src/include/fmgr.h:#define PG_RETURN_BPCHAR_P(x) PG_RETURN_POINTER(x)
- src/include/fmgr.h:#define PG_RETURN_VARCHAR_P(x) PG_RETURN_POINTER(x)
- src/include/fmgr.h:#define PG_RETURN_HEAPTUPLEHEADER(x) return HeapTupleHeaderGetDatum(x)

composite type arg 例子

```
•    #include "postgres.h"
•    #include "fmgr.h"
•    #include "executor/executor.h"      /* for GetAttributeByName()   根据composite 元素名读取Datum  */

•    #ifdef PG_MODULE_MAGIC
•    PG_MODULE_MAGIC;
•    #endif

•    PG_FUNCTION_INFO_V1(c_overpaid);

•    Datum
•    c_overpaid(PG_FUNCTION_ARGS)
•    {
•        HeapTupleHeader  t = PG_GETARG_HEAPTUPLEHEADER(0);  // 获取参数1 composite Datum转成复合类型对应 c type
•        int32            limit = PG_GETARG_INT32(1);  //  获取参数2
•        bool isnull;
•        Datum salary;

•        salary = GetAttributeByName(t, "salary", &isnull);  // 获取composite中的某个元素的Datum
•        if (isnull)
•            PG_RETURN_BOOL(false);
•        /* Alternatively, we might prefer to do PG_RETURN_NULL() for null salary. */

•        PG_RETURN_BOOL(DatumGetInt32(salary) > limit);    // Datum 转ctype进行比较, 通过PG_RETURN*输出Datum
•    }
```

返回record 例子

- 将参数反转输出fun(复合(a,b,c))   输出 c, b, a, c*b+a
- Datum
- c_reverse_tuple(PG_FUNCTION_ARGS)
- {
-   HeapTupleHeader th; // 复合类型对应的C type
-   int32 a,b,c; // 复合类型对应的三个子类型
-   bool aisnull, bisnull, cisnull;   // 是否为空
-   TupleDesc resultTupleDesc;   // 返回record值的描述类型
-   Oid resultTypeId; // 返回值的OID（如果有的话）
-   Datum retvals[4]; // 返回值子类
-   bool retnulls[4]; // 返回值子类是否为空
-   HeapTuple rettuple; // 返回tuple
-   // get the tuple header of 1st argument
-   th = PG_GETARG_HEAPTUPLEHEADER(0);
-   // get argument Datum's and convert them to int32
-   a = DatumGetInt32(GetAttributeByName(th, "a", &aisnull));   // 从composite c type获取子类 Datum并转换为C type
-   b = DatumGetInt32(GetAttributeByName(th, "b", &bisnull));
-   c = DatumGetInt32(GetAttributeByName(th, "c", &cisnull));

返回record 例子

- // debug: report the extracted field values
- ereport(INFO,
-   (errmsg("arg: (a: %d,b: %d, c: %d)", a, b, c)));
- // set up tuple descriptor for result info
- get_call_result_type(fcinfo, &resultTypeId, &resultTupleDesc);
- // check that SQL function definition is set up to return arecord
- Assert(resultTypeId == TYPEFUNC_COMPOSITE);
- // make the tuple descriptor known to postgres as valid return type
- BlessTupleDesc(resultTupleDesc);
- retvals[0] = Int32GetDatum(c); // 构造返回值子集
- retvals[1] = Int32GetDatum(b);
- retvals[2] = Int32GetDatum(a);
- retvals[3] = Int32GetDatum(retvals[0]*retvals[1]+retvals[2]);
- retnulls[0] = aisnull;
- retnulls[1] = bisnull;
- retnulls[2] = cisnull;
- retnulls[3] = aisnull || bisnull || cisnull;
- rettuple = heap_form_tuple(resultTupleDesc, retvals, retnulls); // 构造tuple
- PG_RETURN_DATUM(HeapTupleGetDatum(rettuple)); // 返回Datum
- }

返回表(SRF) 例子

- 伪代码
- Datum
- my_set_returning_function(PG_FUNCTION_ARGS)
- {
-    FuncCallContext *funcctx;
-    Datum   result;
-    further declarations as needed

-    if (SRF_IS_FIRSTCALL()) // 判断该函数是否在该会话第一次被调用
-    {
-       MemoryContext oldcontext;

-       funcctx = SRF_FIRSTCALL_INIT(); // 初始化FuncCallContext
-       oldcontext = MemoryContextSwitchTo(funcctx->multi_call_memory_ctx);
-       /* One-time setup code appears here: */
-       user code
-       if returning composite
-          build TupleDesc, and perhaps AttInMetadata
-       endif returning composite
-       user code
-       MemoryContextSwitchTo(oldcontext);
-    }

返回表(SRF) 例子

```
•    /* Each-time setup code appears here: */
•      user code
•      funcctx = SRF_PERCALL_SETUP();  // 清除之前调用产生的结果
•      user code

•      /* this is just one way we might test whether we are done: */
•      if (funcctx->call_cntr < funcctx->max_calls)
•      {
•         /* Here we want to return another item: */
•         user code
•         obtain result Datum
•         SRF_RETURN_NEXT(funcctx, result);  // 返回一条记录，循环往复
•      }
•      else
•      {
•         /* Here we are done returning items and just need to clean up: */
•         user code
•         SRF_RETURN_DONE(funcctx);  // 调用结束，返回
•      }
•   }
```

示例

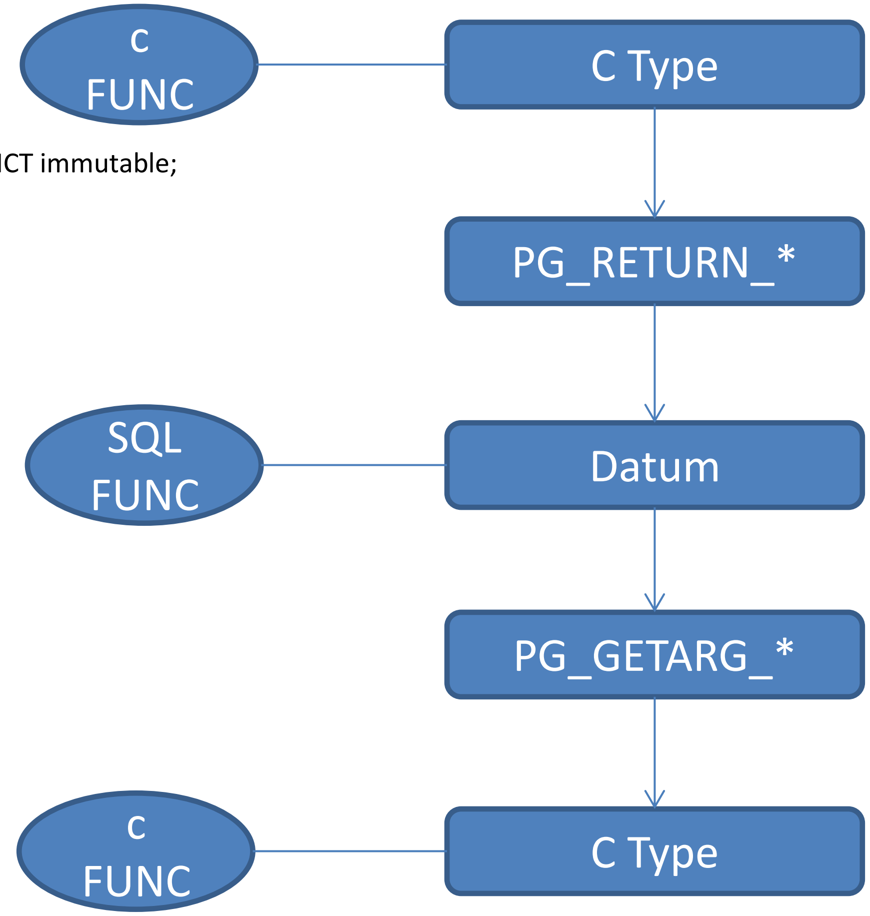
```
•	#include <string.h>
•	#include "postgres.h"
•	#include "fmgr.h"

•	PG_MODULE_MAGIC;    // V1 C UDF

•	PG_FUNCTION_INFO_V1(text_reverse);    // 声明

•	/*
•	 * Return reversed string
•	 */
•	Datum
•	text_reverse(PG_FUNCTION_ARGS)
•	{
•	    text       *str = PG_GETARG_TEXT_PP(0);
•	    const char *p = VARDATA_ANY(str);
•	    int                 len = VARSIZE_ANY_EXHDR(str);
•	    const char *endp = p + len;
•	    text       *result;
•	    char        *dst;

•	    result = palloc(len + VARHDRSZ);
•	    dst = (char *) VARDATA(result) + len;
•	    SET_VARSIZE(result, len + VARHDRSZ);
```

示例

```
•           if (pg_database_encoding_max_length() > 1)
•           {
•               /* multibyte version */
•               while (p < endp)
•               {
•                   int         sz;

•                   sz = pg_mblen(p);
•                   dst -= sz;
•                   memcpy(dst, p, sz);
•                   p += sz;
•               }
•           }
•           else
•           {
•               /* single byte version */
•               while (p < endp)
•                   *(--dst) = *p++;
•           }

•           PG_RETURN_TEXT_P(result);
•       }
```

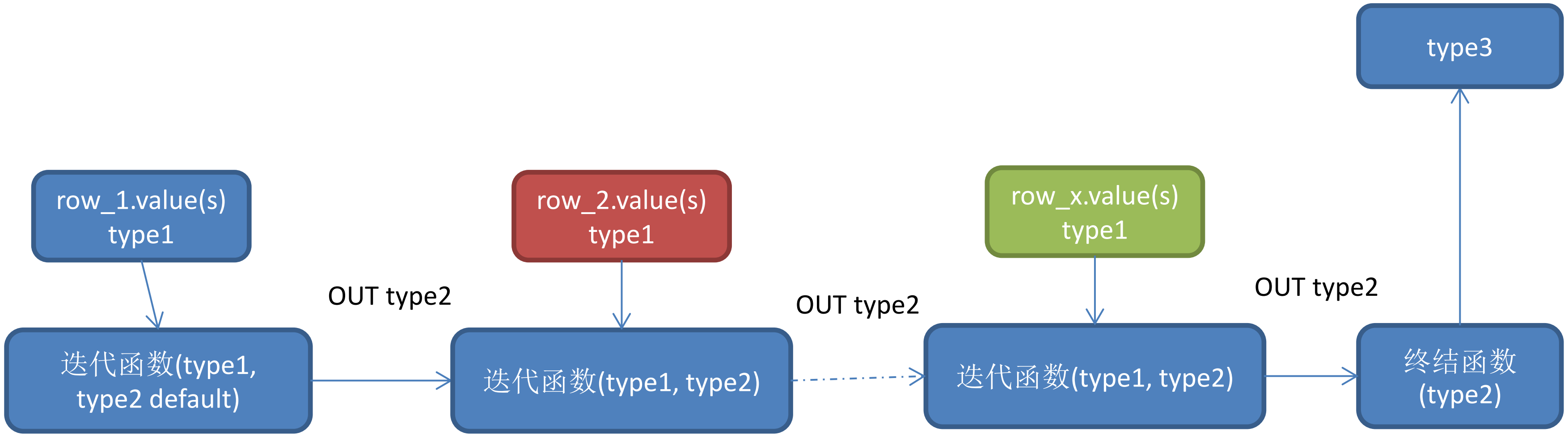
示例

```
gcc -O3 -Wall -Wextra -Werror -I /home/digoal/pgsrc/src/include -g -fPIC -c ./reverse.c -o reverse.o
gcc -O3 -Wall -Wextra -Werror -I /home/digoal/pgsrc/src/include -g -shared  reverse.o -o libreverse.so

cp libreverse.so /home/digoal/pghome/lib

postgres=# create or replace function reverse(text) returns text as
'/home/digoal/pghome/lib/libreverse.so', 'text_reverse' language C STRICT immutable;
CREATE FUNCTION
postgres=# select reverse('abc');
 reverse
---------
 cba
(1 row)

postgres=# select reverse('a f d12');
 reverse
---------
 21d f a
(1 row)

postgres=# select reverse(null);
 reverse
---------

(1 row)
```



聚合函数原理

- 简单的聚合伪代码
  - 迭代函数(传入type1、输出type2(可以设置默认值))
  - 迭代结果(与输出类型2一致)
  - 终结函数(可选，输入type2, 输出type3)
  - 聚合函数返回类型 type3(有终结函数), 或type2(无终结函数)



自定义聚合函数

- http://blog.163.com/digoal@126/blog/static/163877040201211181125334 10/

- digoal=# create aggregate agg_append (text) (
- sfunc = textcat,
- stype = text);

- digoal=# create aggregate array_agg (anyelement) (
-   sfunc = array_append,
-   stype = anyarray);

- digoal=# create aggregate agg_append (text) (
- sfunc = textcat,
- stype = text,
- FINALFUNC = final_array_agg);

窗口函数原理

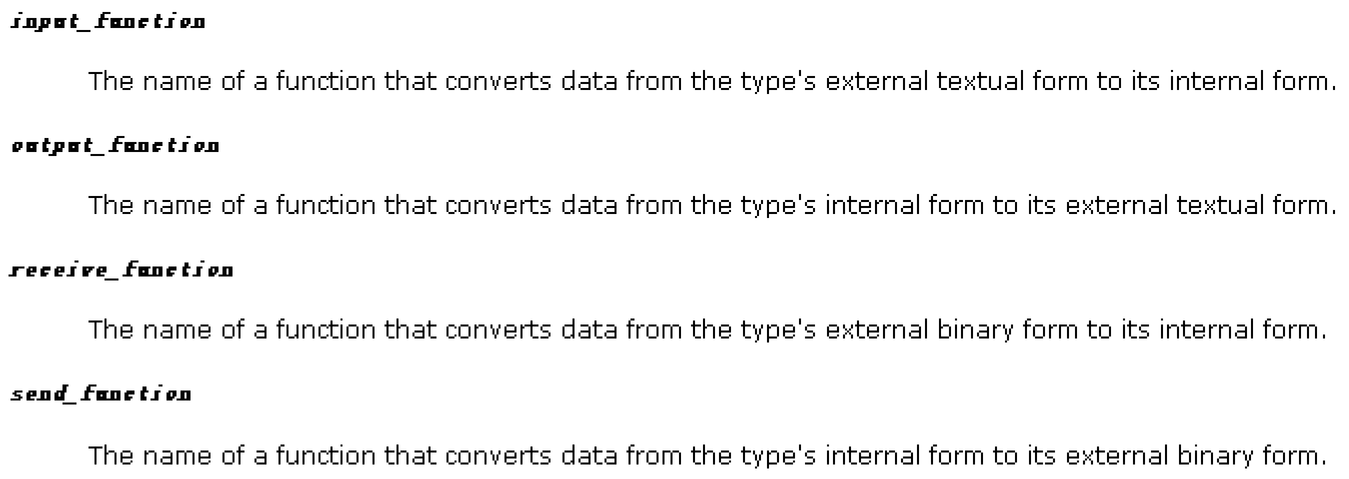
自定义窗口函数

- src/include/utils/builtins.h
- src/backend/utils/adt/windowfuncs.c

```
/*
 * first_value
 * return the value of VE evaluated on the first row of the
 * window frame, per spec.
 */
Datum
window_first_value(PG_FUNCTION_ARGS)
{
	WindowObject winobj = PG_WINDOW_OBJECT();
	Datum		result;
	bool		isnull;

	result = WinGetFuncArgInFrame(winobj, 0,
								  0, WINDOW_SEEK_HEAD, true,
								  &isnull, NULL);
	if (isnull)
		PG_RETURN_NULL();

	PG_RETURN_DATUM(result);
}
```

自定义数据类型

- 语法
- https://www.postgresql.org/docs/9.6/static/sql-createtype.html
- 例子
- https://www.postgresql.org/docs/9.6/static/xtypes.html



input_function

The name of a function that converts data from the type's external textual form to its internal form.

output_function

The name of a function that converts data from the type's internal form to its external textual form.

receive_function

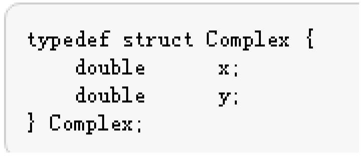
The name of a function that converts data from the type's external binary form to its internal form.

send_function

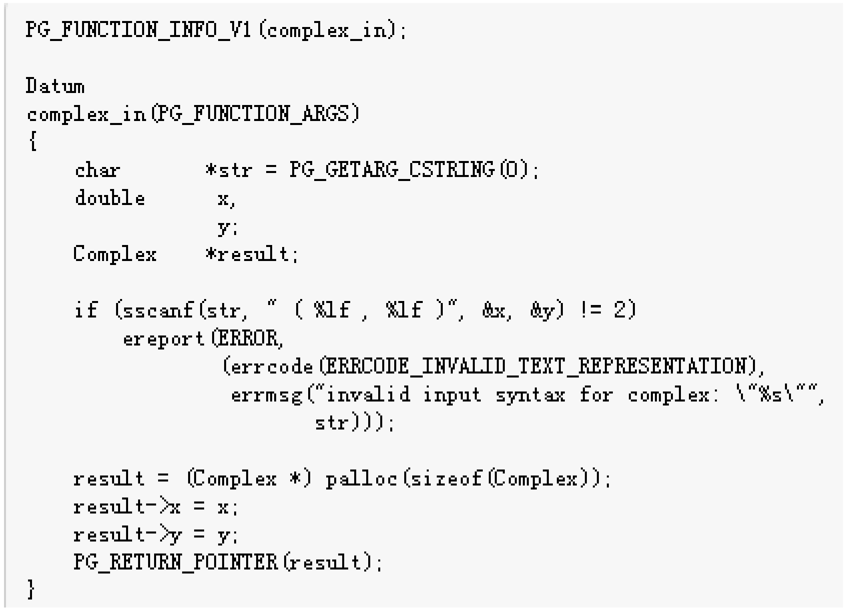
The name of a function that converts data from the type's internal form to its external binary form.

自定义数据类型

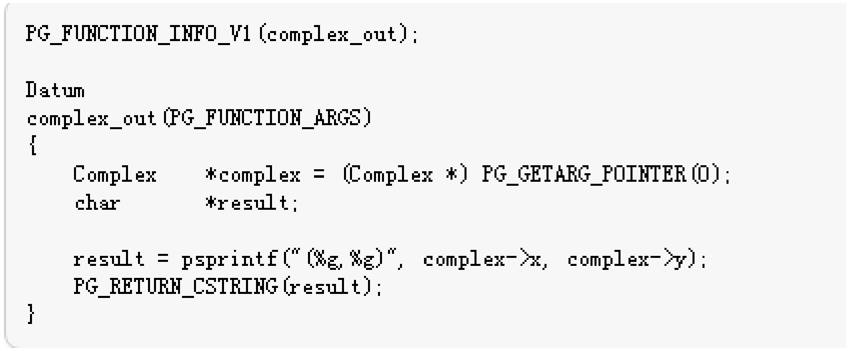
- 语法
- https://www.postgresql.org/docs/9.6/static/sql-createtype.html
- 例子
- https://www.postgresql.org/docs/9.6/static/xtypes.html



```
typedef struct Complex {
    double      x;
    double      y;
} Complex;
```



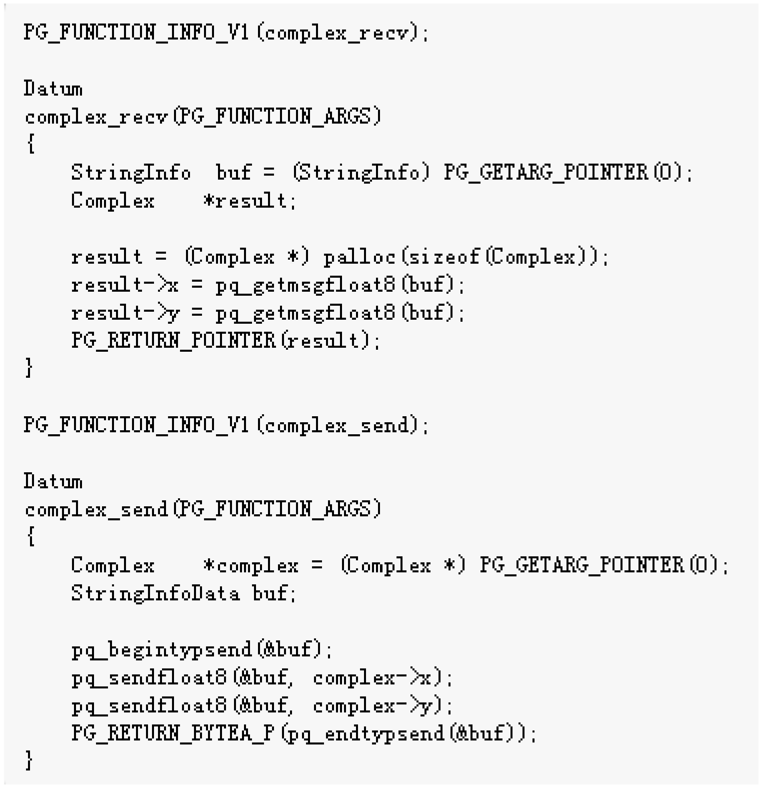
```
PG_FUNCTION_INFO_V1(complex_in);

Datum
complex_in(PG_FUNCTION_ARGS)
{
    char       *str = PG_GETARG_CSTRING(0);
    double      x,
                y;
    Complex    *result;

    if (sscanf(str, " ( %lf , %lf )", &x, &y) != 2)
        ereport(ERROR,
                (errcode(ERRCODE_INVALID_TEXT_REPRESENTATION),
                 errmsg("invalid input syntax for complex: \"%s\"",
                        str)));

    result = (Complex *) palloc(sizeof(Complex));
    result->x = x;
    result->y = y;
    PG_RETURN_POINTER(result);
}
```



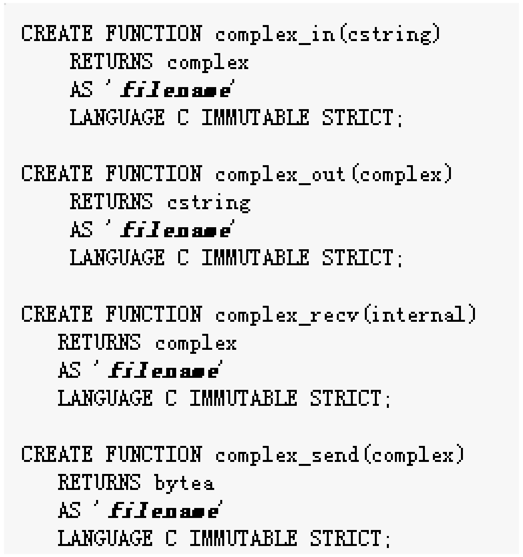
```
PG_FUNCTION_INFO_V1(complex_out);

Datum
complex_out(PG_FUNCTION_ARGS)
{
    Complex    *complex = (Complex *) PG_GETARG_POINTER(0);
    char       *result;

    result = psprintf("(%g,%g)", complex->x, complex->y);
    PG_RETURN_CSTRING(result);
}
```

自定义数据类型

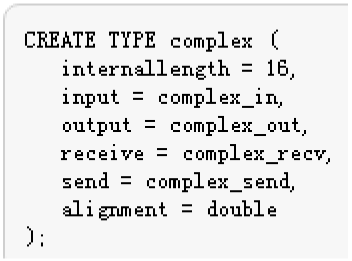
- 例子
- https://www.postgresql.org/docs/9.6/static/xtypes.html
- internallength
  - A numeric constant that specifies the length in bytes of the new type's internal representation. The default assumption is that it is variable-length.
- alignment
  - The storage alignment requirement of the data type. If specified, it must be char, int2, int4, or double; the default is int4.



```
PG_FUNCTION_INFO_V1(complex_recv);

Datum
complex_recv(PG_FUNCTION_ARGS)
{
    StringInfo  buf = (StringInfo) PG_GETARG_POINTER(0);
    Complex    *result;

    result = (Complex *) palloc(sizeof(Complex));
    result->x = pq_getmsgfloat8(buf);
    result->y = pq_getmsgfloat8(buf);
    PG_RETURN_POINTER(result);
}

PG_FUNCTION_INFO_V1(complex_send);

Datum
complex_send(PG_FUNCTION_ARGS)
{
    Complex    *complex = (Complex *) PG_GETARG_POINTER(0);
    StringInfoData buf;

    pq_begintypsend(&buf);
    pq_sendfloat8(&buf, complex->x);
    pq_sendfloat8(&buf, complex->y);
    PG_RETURN_BYTEA_P(pq_endtypsend(&buf));
}
```



```
CREATE FUNCTION complex_in(cstring)
    RETURNS complex
    AS 'filename'
    LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION complex_out(complex)
    RETURNS cstring
    AS 'filename'
    LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION complex_recv(internal)
   RETURNS complex
   AS 'filename'
   LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION complex_send(complex)
   RETURNS bytea
   AS 'filename'
   LANGUAGE C IMMUTABLE STRICT;
```



```
CREATE TYPE complex (
   internallength = 16,
   input = complex_in,
   output = complex_out,
   receive = complex_recv,
   send = complex_send,
   alignment = double
);
```

如何自定义操作符

- https://www.postgresql.org/docs/9.5/static/sql-createoperator.html
- http://blog.163.com/digoal@126/blog/static/16387704020156158447718/

- CREATE OPERATOR name (
-     PROCEDURE = function_name
-     [, LEFTARG = left_type] [, RIGHTARG = right_type]
-     [, COMMUTATOR = com_op] [, NEGATOR = neg_op]
-     [, RESTRICT = res_proc] [, JOIN = join_proc]
-     [, HASHES] [, MERGES]
-)

如何自定义操作符

- 1. commutator，指明**x op1 y等效于y op2 x**，即操作数调换，返回的值一样。例如2>1 和1<2结果是一致的。那么>就是<的commutator或者反之。又例如1+2和2+1是等价的，那么+就是+的commutator。commutator只需要在创建其中一个操作符时指定，创建另一个对应的操作符时可以不需要指定，PostgreSQL会自动建立这个关系。例如创建>操作符时指定了它的commutator是<，那么在创建<操作符时可以不需要指定>是它的commutator。
- 另外需要注意，有commutator操作符的操作符的**左右两侧的参数类型必须一致**，这样才能满足x op1 y等价于y op2 x。
- 优化器如何利用commutator呢？例如**索引扫描，列必须在操作符的左侧才能使用索引。1 > tbl.c这个条件，如果>没有commutator的话，是不能使用索引的。**

如何自定义操作符

- 2. negator， 指**x op1 y 等价于 not(y op2 x)**， 或者x op1等价于not(y op2)， 或者op1 x 等价于not(op2 y)， 因此negator支持一元和二元操作符。
- 例子:
- 如果=和<>是一对negator操作符， NOT (x = y)可以替换为 x <> y。
- 同样， 操作符两侧参数x,y的**类型必须一致**。并且仅适用于**返回布尔逻辑类型**的操作符。

如何自定义操作符

- 3. restrict，是用于**评估选择性**的函数，仅适用于二元操作符，例如where col>100，这个查询条件，如何评估选择性呢？是通过操作符的restrict来指定的，**选择性乘以pg_class.reltuples**就可以评估得到这个**查询条件的行数**。

如何自定义操作符

- 4. join，是joinsel即join的选择性计算函数。
- 对应pg_operator.oprjoin

- 5. hashes
- 6. merges
- hashes和merges表示该操作符是否允许hash join和merge join, 只有**返回布尔逻辑值**的**二元操作符**满足这个要求。

自定义操作符例子

```
Datum
citext_ne(PG_FUNCTION_ARGS)
{
    text       *left = PG_GETARG_TEXT_PP(0);
    text       *right = PG_GETARG_TEXT_PP(1);
    char       *lcstr,
               *rcstr;
    bool        result;

    /* We can't compare lengths in advance of downcasing ... */

    lcstr = str_tolower(VARDATA_ANY(left), VARSIZE_ANY_EXHDR(left), DEFAULT_COLLATION_OID);
    rcstr = str_tolower(VARDATA_ANY(right), VARSIZE_ANY_EXHDR(right), DEFAULT_COLLATION_OID);

    /*
     * Since we only care about equality or not-equality, we can avoid all the
     * expense of strcoll() here, and just do bitwise comparison.
     */
    result = (strcmp(lcstr, rcstr) != 0);

    pfree(lcstr);
    pfree(rcstr);
    PG_FREE_IF_COPY(left, 0);
    PG_FREE_IF_COPY(right, 1);

    PG_RETURN_BOOL(result);
}
```

自定义操作符例子

- CREATE FUNCTION citext_ne(citext, citext)
- RETURNS bool
- AS 'MODULE_PATHNAME'
- LANGUAGE C IMMUTABLE STRICT;

- CREATE OPERATOR <> (
-     LEFTARG   = CITEXT,
-     RIGHTARG   = CITEXT,
-     NEGATOR   = =,
-     COMMUTATOR = <>,
-     PROCEDURE = citext_ne,
-     RESTRICT   = neqsel,
-     JOIN   = neqjoinsel
-);

x <> y
等价于
not(x = y)

- 等效优化
- https://yq.aliyun.com/articles/51131

自定义索引语法

- 扩展索引语法
- CREATE OPERATOR CLASS name [DEFAULT] FOR TYPE data_type
- USING index_method [FAMILY family_name] AS
- { OPERATOR strategy_number operator_name [(op_type, op_type)] [FOR SEARCH | FOR ORDER BY sort_family_name]
- | FUNCTION support_number [(op_type [, op_type])] function_name (argument_type [, ...])
- | STORAGE storage_type
- } [, ...]

- 操作符strategy_number、函数support_number
- https://www.postgresql.org/docs/9.5/static/xindex.html

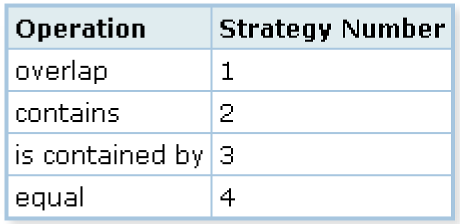
开放索引接口介绍

- GIN
  - 索引结构 value : (ctid1, ctid2, ...)
  - https://www.postgresql.org/docs/9.5/static/gin.html
  - 开发接口
  - https://www.postgresql.org/docs/9.5/static/gin-extensibility.html
  - int compare(Datum a, Datum b)
    - 比较两个element
  - Datum *extractValue(Datum itemValue, int32 *nkeys, bool **nullFlags)
    - 输入ctid返回对应行所在列存储的elements
  - Datum *extractQuery(Datum query, int32 *nkeys, StrategyNumber n, bool **pmatch, Pointer **extra_data, bool **nullFlags, int32 *searchMode)
    - column op query 返回对应行(s)所在列存储的elements
  - bool consistent(bool check[], StrategyNumber n, Datum query, int32 nkeys, Pointer extra_data[], bool *recheck, Datum queryKeys[], bool nullFlags[])
    - column op query 返回 true or false
  - GinTernaryValue triConsistent(GinTernaryValue check[], StrategyNumber n, Datum query, int32 nkeys, Pointer extra_data[], Datum queryKeys[], bool nullFlags[])
    - GIN_TRUE, GIN_FALSE and GIN_MAYBE(需要recheck, lossy部分).
- 例子
  - array, ts, hstore

开放索引接口介绍

- operator strategy number
- 不固定策略号、视数据类型

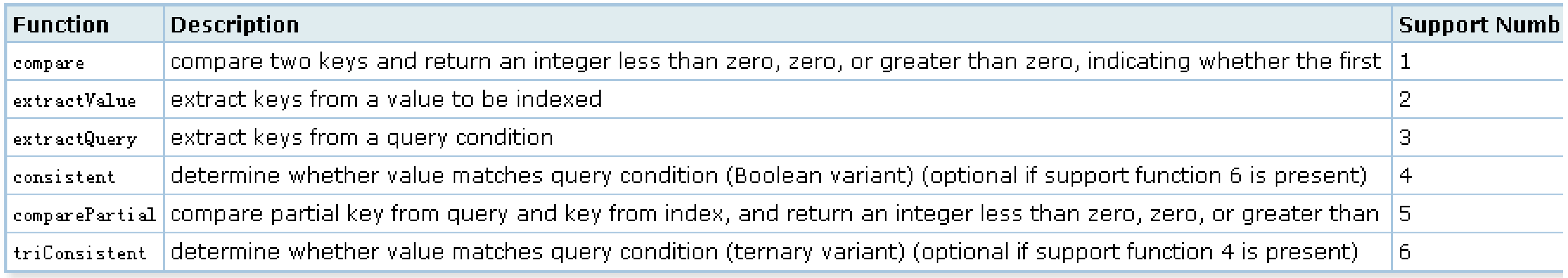
Table 35-6. GIN Array Strategies



Operation	Strategy Number
overlap	1
contains	2
is contained by	3
equal	4

- **Index Method Support Routines**

Table 35-12. GIN Support Functions



Function	Description	Support Numb
`compare`	compare two keys and return an integer less than zero, zero, or greater than zero, indicating whether the first	1
`extractValue`	extract keys from a value to be indexed	2
`extractQuery`	extract keys from a query condition	3
`consistent`	determine whether value matches query condition (Boolean variant) (optional if support function 6 is present)	4
`comparePartial`	compare partial key from query and key from index, and return an integer less than zero, zero, or greater than	5
`triConsistent`	determine whether value matches query condition (ternary variant) (optional if support function 4 is present)	6

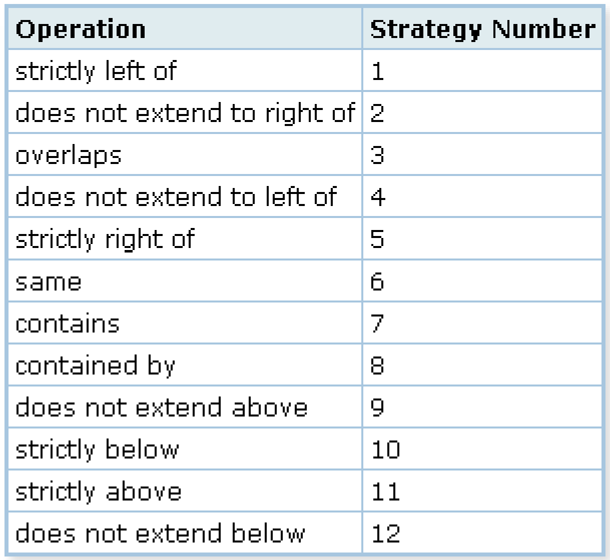
开放索引接口介绍

- GiST (balanced, tree结构索引)
  - https://www.postgresql.org/docs/9.5/static/gist-intro.html
  - 开发接口
  - https://www.postgresql.org/docs/9.5/static/gist-extensibility.html
  - consistent
    - column op query 返回 true or false (recheck 表示是否为lossy)
  - union
    - 输入一批entry转换成一个entry, 基于结果entry创建索引
  - compress
    - 将被索引的entry压缩成适合在index page中存储的Datum
  - decompress
    - 解压
  - penalty
    - 计算并返回entry插入索引branch的cost
  - picksplit
    - 当索引页需要分裂时，决定哪些entry需要保留在原地page，哪些entry需要移到新的page。
  - same
    - 比较两个entry是否相等
  - distance
    - column op query 返回"距离"，需要排序的话必须实现distance接口函数
  - fetch
    - 获取索引entry对应的column value
- 例子
  - range, point, box

开放索引接口介绍

- operator strategy number
- 不固定策略号、视数据类型

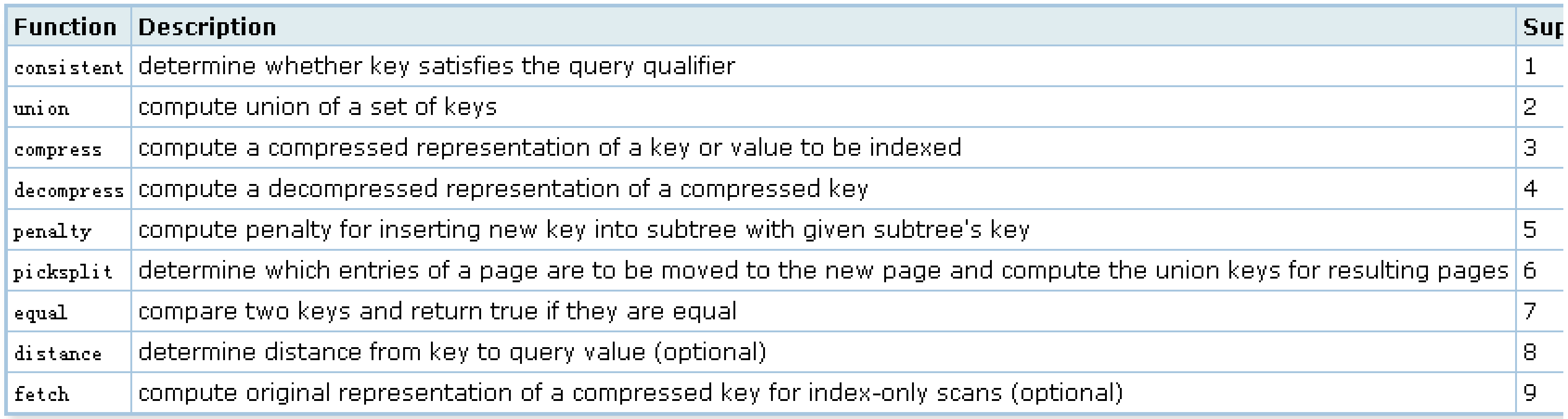
Table 35-4. GiST Two-Dimensional "R-tree" Strategies



Operation	Strategy Number
strictly left of	1
does not extend to right of	2
overlaps	3
does not extend to left of	4
strictly right of	5
same	6
contains	7
contained by	8
does not extend above	9
strictly below	10
strictly above	11
does not extend below	12

- **Index Method Support Routines**

Table 35-10. GiST Support Functions



Function	Description	Sup
`consistent`	determine whether key satisfies the query qualifier	1
`union`	compute union of a set of keys	2
`compress`	compute a compressed representation of a key or value to be indexed	3
`decompress`	compute a decompressed representation of a compressed key	4
`penalty`	compute penalty for inserting new key into subtree with given subtree's key	5
`picksplit`	determine which entries of a page are to be moved to the new page and compute the union keys for resulting pages	6
`equal`	compare two keys and return true if they are equal	7
`distance`	determine distance from key to query value (optional)	8
`fetch`	compute original representation of a compressed key for index-only scans (optional)	9

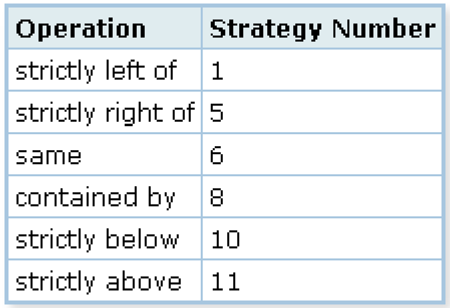
开放索引接口介绍

- SP-GiST (non-balanced 数据结构, quad-trees, k-d trees, and radix trees (tries))
- repeatedly divide search space into partitions that need not be of equal size
  - https://www.postgresql.org/docs/9.5/static/spgist.html
  - 开发接口
  - https://www.postgresql.org/docs/9.5/static/spgist-extensibility.html
- 例子
  - range, point

开放索引接口介绍

- operator strategy number
- 不固定策略号、视数据类型

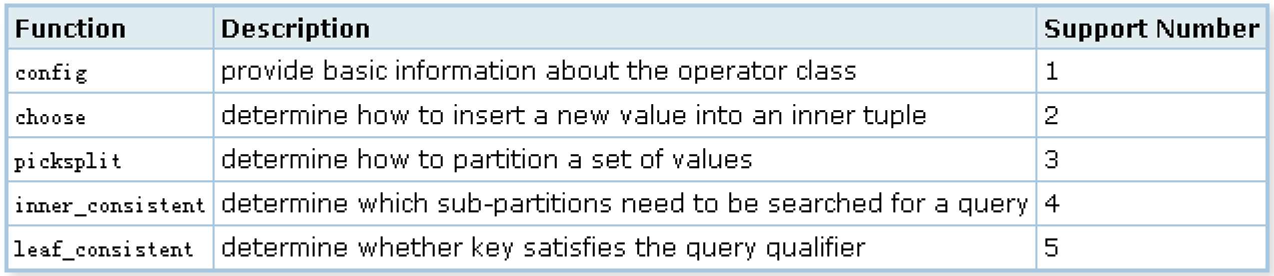
Table 35-5. SP-GiST Point Strategies



Operation	Strategy Number
strictly left of	1
strictly right of	5
same	6
contained by	8
strictly below	10
strictly above	11

- **Index Method Support Routines**

Table 35-11. SP-GiST Support Functions



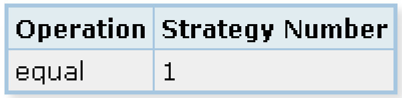
Function	Description	Support Number
`config`	provide basic information about the operator class	1
`choose`	determine how to insert a new value into an inner tuple	2
`picksplit`	determine how to partition a set of values	3
`inner_consistent`	determine which sub-partitions need to be searched for a query	4
`leaf_consistent`	determine whether key satisfies the query qualifier	5

开放索引接口介绍

- Operator strategy number
- btree
- hash
- 固定策略号

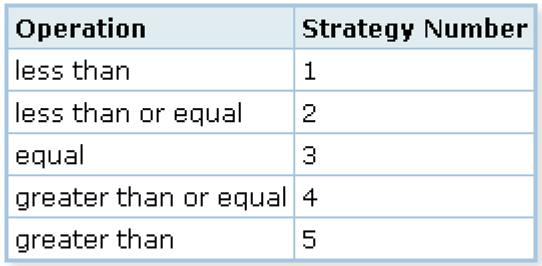
- brin
- 不固定策略号、视数据类型

Table 35-3. Hash Strategies



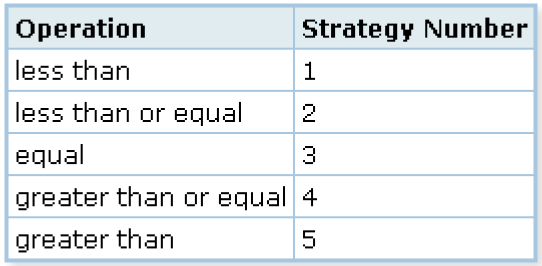
Operation	Strategy Number
equal	1

Table 35-2. B-tree Strategies



Operation	Strategy Number
less than	1
less than or equal	2
equal	3
greater than or equal	4
greater than	5

Table 35-7. BRIN Minmax Strategies

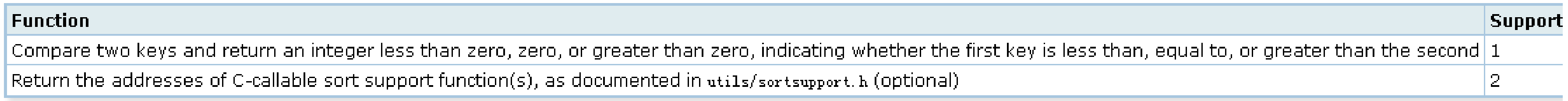


Operation	Strategy Number
less than	1
less than or equal	2
equal	3
greater than or equal	4
greater than	5

开放索引接口介绍

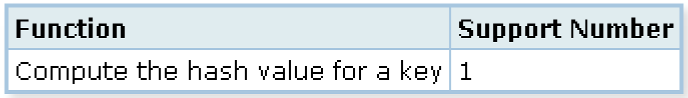
- **Index Method Support Routines**
  - btree
  - hash
  - brin

Table 35-8. B-tree Support Functions



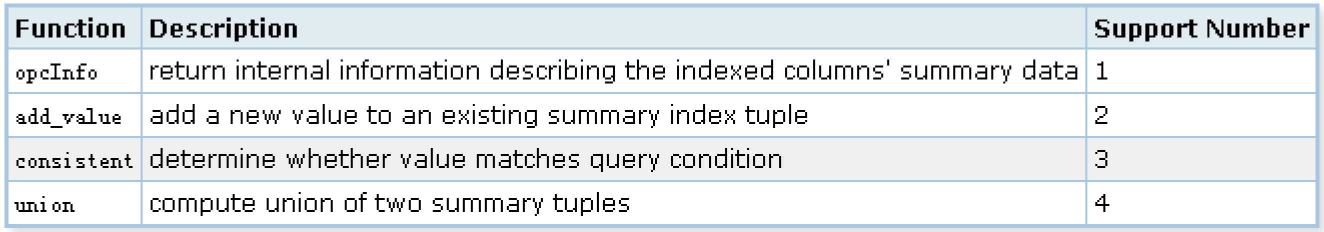
Function	Support
Compare two keys and return an integer less than zero, zero, or greater than zero, indicating whether the first key is less than, equal to, or greater than the second	1
Return the addresses of C-callable sort support function(s), as documented in `utils/sortsupport.h` (optional)	2

Table 35-9. Hash Support Functions



Function	Support Number
Compute the hash value for a key	1

Table 35-13. BRIN Support Functions



Function	Description	Support Number
`opcInfo`	return internal information describing the indexed columns' summary data	1
`add_value`	add a new value to an existing summary index tuple	2
`consistent`	determine whether value matches query condition	3
`union`	compute union of two summary tuples	4

自定义 GIN 索引 例子

- Operator(s) function

```
•    PG_FUNCTION_INFO_V1(hstore_contains);
•    Datum
•    hstore_contains(PG_FUNCTION_ARGS)
•    {
•        HStore     *val = PG_GETARG_HS(0);
•        HStore     *tmpl = PG_GETARG_HS(1);
•        bool        res = true;
•        HEntry     *te = ARRPTR(tmpl);
•        char       *tstr = STRPTR(tmpl);
•        HEntry     *ve = ARRPTR(val);
•        char       *vstr = STRPTR(val);
•        int         tcount = HS_COUNT(tmpl);
•        int         lastidx = 0;
•        int         i;

•        /*
•         * we exploit the fact that keys in "tmpl" are in strictly increasing
•         * order to narrow the hstoreFindKey search; each search can start one
•         * entry past the previous "found" entry, or at the lower bound of the
•         * search
•         */
```

自定义 GIN 索引 例子

```
•       for (i = 0; res && i < tcount; ++i)
•           {
•               int             idx = hstoreFindKey(val, &lastidx,
•                                                   HSTORE_KEY(te, tstr, i),
•                                                   HSTORE_KEYLEN(te, i));

•               if (idx >= 0)
•               {
•                   bool        nullval = HSTORE_VALISNULL(te, i);
•                   int             vallen = HSTORE_VALLEN(te, i);

•                   if (nullval != HSTORE_VALISNULL(ve, idx) ||
•                       (!nullval && (vallen != HSTORE_VALLEN(ve, idx) ||
•                                     memcmp(HSTORE_VAL(te, tstr, i),
•                                            HSTORE_VAL(ve, vstr, idx),
•                                            vallen) != 0)))
•                       res = false;
•               }
•               else
•                   res = false;
•           }

•           PG_RETURN_BOOL(res);
•       }
```

自定义 GIN 索引 例子

- CREATE FUNCTION hs_contains(hstore,hstore)
- RETURNS bool
- AS 'MODULE_PATHNAME','hstore_contains'
- LANGUAGE C STRICT IMMUTABLE;

- CREATE OPERATOR @> (
-         LEFTARG = hstore,
-         RIGHTARG = hstore,
-         PROCEDURE = hs_contains,
-         COMMUTATOR = '<@',
-         RESTRICT = contsel,
-         JOIN = contjoinsel
-);

- ... ...

自定义 GIN 索引 例子

- -- GIN support
- **index method support functions (C code)**

```
Datum
gin_extract_hstore(PG_FUNCTION_ARGS)
{
    HStore     *hs = PG_GETARG_HS(0);
    int32      *nentries = (int32 *) PG_GETARG_POINTER(1);
    Datum      *entries = NULL;
    HEntry     *hsent = ARRPTR(hs);
    char       *ptr = STRPTR(hs);
    int         count = HS_COUNT(hs);
    int         i;

    *nentries = 2 * count;
    if (count)
        entries = (Datum *) palloc(sizeof(Datum) * 2 * count);
```

自定义 GIN 索引 例子

```
•   for (i = 0; i < count; ++i)
•       {
•           text        *item;

•           item = makeitem(HSTORE_KEY(hsent, ptr, i),
•                           HSTORE_KEYLEN(hsent, i),
•                           KEYFLAG);
•           entries[2 * i] = PointerGetDatum(item);

•           if (HSTORE_VALISNULL(hsent, i))
•               item = makeitem(NULL, 0, NULLFLAG);
•           else
•               item = makeitem(HSTORE_VAL(hsent, ptr, i),
•                               HSTORE_VALLEN(hsent, i),
•                               VALFLAG);
•           entries[2 * i + 1] = PointerGetDatum(item);
•       }

•       PG_RETURN_POINTER(entries);
•   }

•   ... ...
```

自定义 GIN 索引 例子

```
-- GIN support
index method support functions (C code 其他 略 )
CREATE FUNCTION gin_extract_hstore(internal, internal)
RETURNS internal
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION gin_extract_hstore_query(internal, internal, int2, internal, internal)
RETURNS internal
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION gin_consistent_hstore(internal, int2, internal, int4, internal, internal)
RETURNS bool
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT;

CREATE OPERATOR CLASS gin_hstore_ops
DEFAULT FOR TYPE hstore USING gin
AS
    OPERATOR    7    @>,
    OPERATOR    9    ?(hstore,text),
    OPERATOR    10   ?|(hstore,text[]),
    OPERATOR    11   ?&(hstore,text[]),
    FUNCTION    1    bttextcmp(text,text),
    FUNCTION    2    gin_extract_hstore(internal, internal),
    FUNCTION    3    gin_extract_hstore_query(internal, internal, int2, internal, internal),
    FUNCTION    4    gin_consistent_hstore(internal, int2, internal, int4, internal, internal),
    STORAGE     text;
```

自定义 GiST 索引 例子

- 略

PostgreSQL 内核扩展接口总结

- PostgreSQL有哪些开放接口
  - UDF（包括聚合、窗口以及普通的函数）
    - https://www.postgresql.org/docs/9.5/static/xfunc-c.html
  - GiST, SP-GiST, GIN, BRIN 自定义索引接口
    - https://www.postgresql.org/docs/9.5/static/gist.html ... ...
  - 扩展索引接口(bloom例子)
    - https://www.postgresql.org/docs/9.6/static/bloom.html
    - https://www.postgresql.org/docs/9.6/static/xindex.html
  - 操作符
    - https://www.postgresql.org/docs/9.5/static/sql-createoperator.html
  - 数据类型
    - https://www.postgresql.org/docs/9.5/static/sql-createtype.html
  - FDW
    - https://www.postgresql.org/docs/9.5/static/fdwhandler.html
  - 函数语言 handler
    - https://www.postgresql.org/docs/9.5/static/plhandler.html
  - SPI
    - https://www.postgresql.org/docs/9.5/static/spi.html
  - 动态fork 进程，动态创建共享内存段
    - https://www.postgresql.org/docs/9.5/static/bgworker.html
  - table sampling method
    - https://www.postgresql.org/docs/9.5/static/tablesample-method.html
  - custom scan provider
    - https://www.postgresql.org/docs/9.5/static/custom-scan.html
  - 自定义REDO日志encode,decode接口
    - https://www.postgresql.org/docs/9.6/static/generic-wal.html

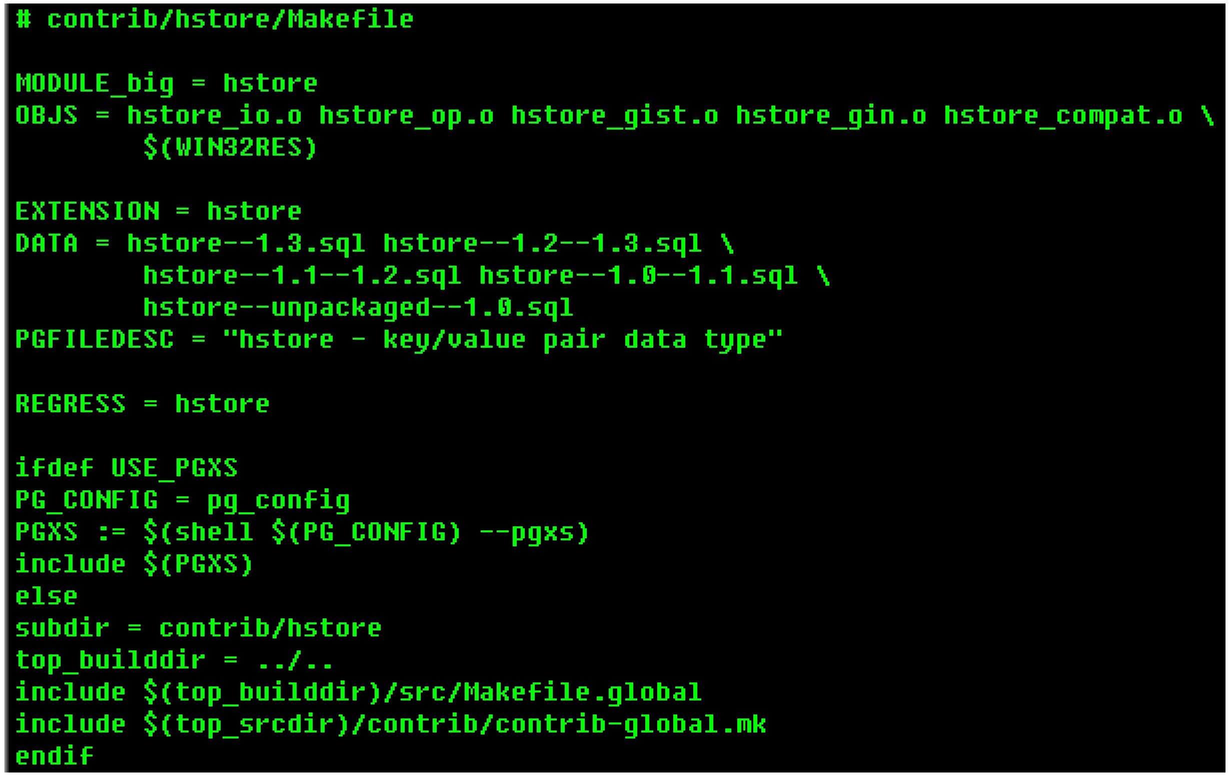
PostgreSQL 插件打包、发布

- https://www.postgresql.org/docs/9.6/static/extend-extensions.html
- https://www.postgresql.org/docs/9.6/static/extend-pgxs.html

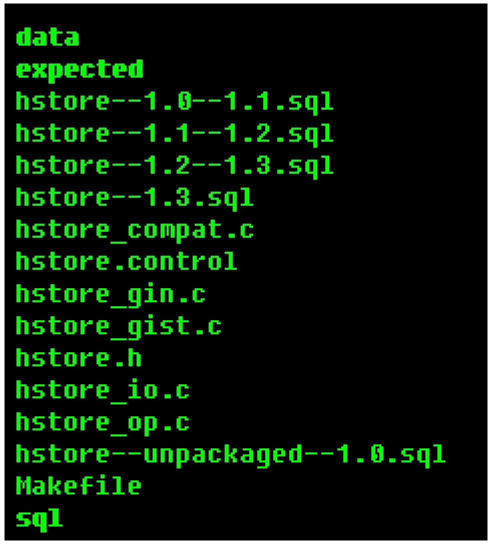
- http://pgxn.org/about/

PostgreSQL 插件打包、发布

- Makefile



```
# contrib/hstore/Makefile

MODULE_big = hstore
OBJS = hstore_io.o hstore_op.o hstore_gist.o hstore_gin.o hstore_compat.o \
        $(WIN32RES)

EXTENSION = hstore
DATA = hstore--1.3.sql hstore--1.2--1.3.sql \
        hstore--1.1--1.2.sql hstore--1.0--1.1.sql \
        hstore--unpackaged--1.0.sql
PGFILEDESC = "hstore - key/value pair data type"

REGRESS = hstore

ifdef USE_PGXS
PG_CONFIG = pg_config
PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
else
subdir = contrib/hstore
top_builddir = ../..
include $(top_builddir)/src/Makefile.global
include $(top_srcdir)/contrib/contrib-global.mk
endif
```



```
data
expected
hstore--1.0--1.1.sql
hstore--1.1--1.2.sql
hstore--1.2--1.3.sql
hstore--1.3.sql
hstore_compat.c
hstore.control
hstore_gin.c
hstore_gist.c
hstore.h
hstore_io.c
hstore_op.c
hstore--unpackaged--1.0.sql
Makefile
sql
```

PostgreSQL 插件打包、发布

- control file
  - # hstore extension
  - comment = 'data type for storing sets of (key, value) pairs'
  - default_version = '1.3'
  - module_pathname = '$libdir/hstore'
  - relocatable = true

GPU，FPGA 如何与 PostGIS深度整合

- custom scan provider API
- https://www.postgresql.org/docs/9.5/static/custom-scan.html

pg_strom介绍

- https://github.com/pg-strom/devel
- https://wiki.postgresql.org/wiki/PGStrom

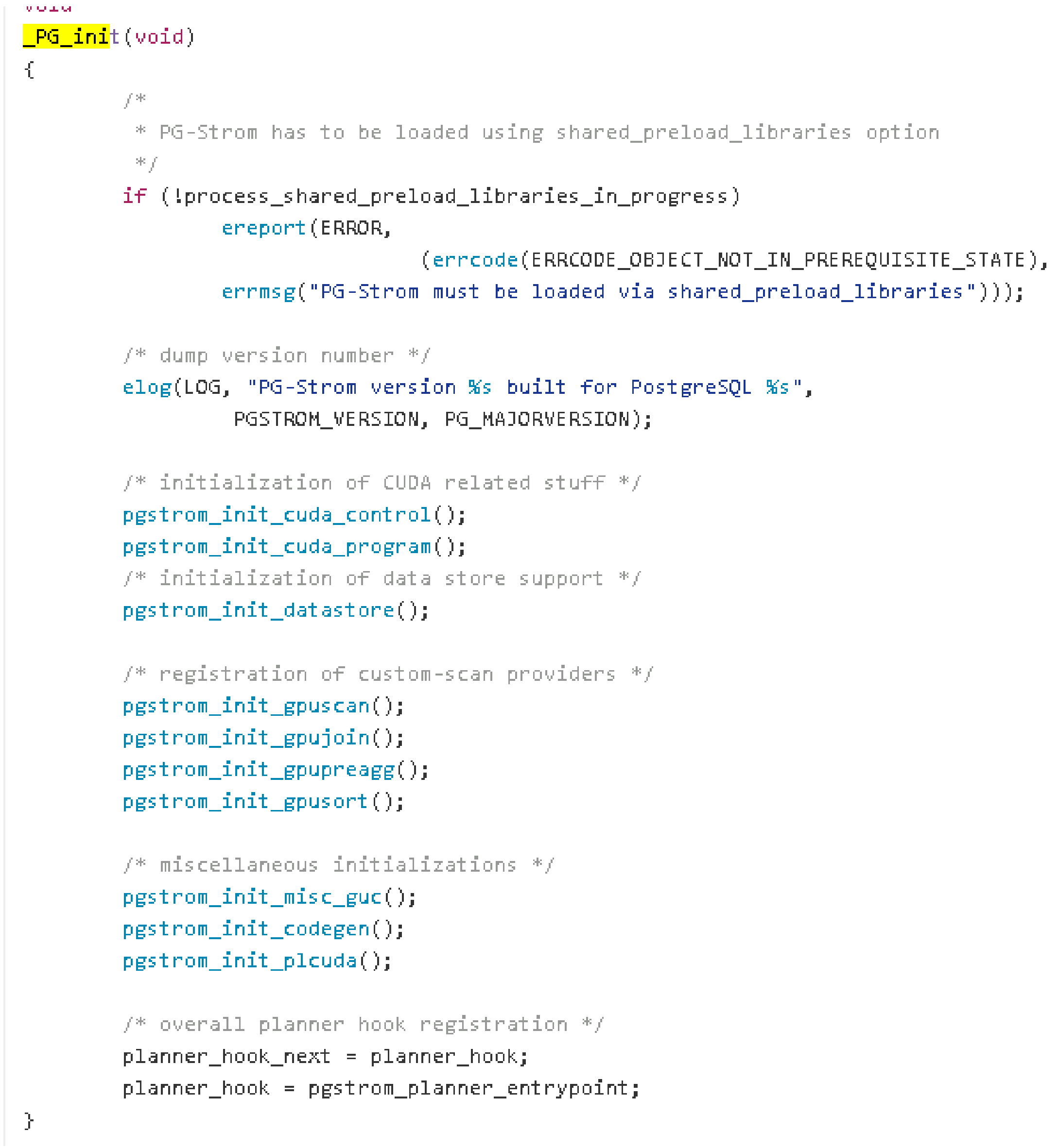
pg_strom 加速例子

- src/backend/optimizer/plan/planner.c

```
• PlannedStmt *
• planner(Query *parse, int cursorOptions, ParamListInfo boundParams)
• {
•         PlannedStmt *result;

•         if (planner_hook)
•                 result = (*planner_hook) (parse, cursorOptions, boundParams);
•         else
•                 result = standard_planner(parse, cursorOptions, boundParams);
•         return result;
• }
```

pg_strom 加速例子

- _PG_init

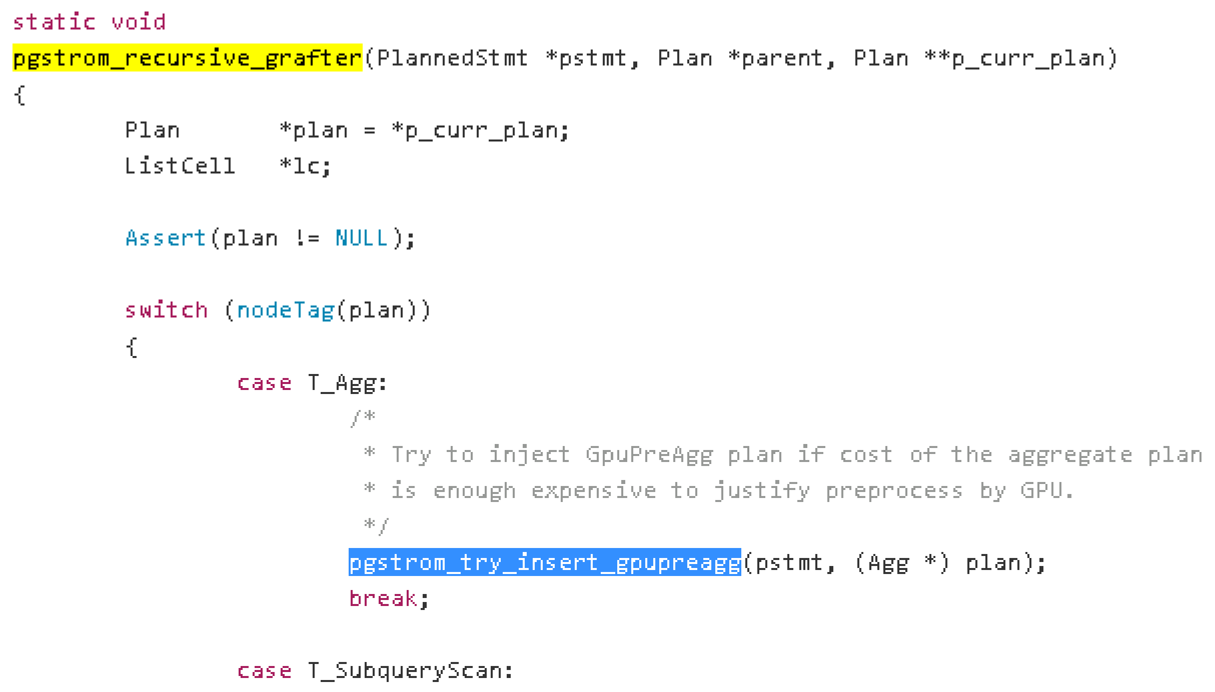


```
_PG_init(void)
{
        /*
         * PG-Strom has to be loaded using shared_preload_libraries option
         */
        if (!process_shared_preload_libraries_in_progress)
                ereport(ERROR,
                                (errcode(ERRCODE_OBJECT_NOT_IN_PREREQUISITE_STATE),
                errmsg("PG-Strom must be loaded via shared_preload_libraries")));

        /* dump version number */
        elog(LOG, "PG-Strom version %s built for PostgreSQL %s",
                  PGSTROM_VERSION, PG_MAJORVERSION);

        /* initialization of CUDA related stuff */
        pgstrom_init_cuda_control();
        pgstrom_init_cuda_program();
        /* initialization of data store support */
        pgstrom_init_datastore();

        /* registration of custom-scan providers */
        pgstrom_init_gpuscan();
        pgstrom_init_gpujoin();
        pgstrom_init_gpupreagg();
        pgstrom_init_gpusort();

        /* miscellaneous initializations */
        pgstrom_init_misc_guc();
        pgstrom_init_codegen();
        pgstrom_init_plcuda();

        /* overall planner hook registration */
        planner_hook_next = planner_hook;
        planner_hook = pgstrom_planner_entrypoint;
}
```

pg_strom 加速例子

```
static PlannedStmt *
pgstrom_planner_entrypoint(Query *parse,
                                                  int cursorOptions,
                                                  ParamListInfo boundParams)
{
        PlannedStmt     *result;

        if (planner_hook_next)
                result = planner_hook_next(parse, cursorOptions, boundParams);
        else
                result = standard_planner(parse, cursorOptions, boundParams);

        if (pgstrom_enabled)
        {
                ListCell   *cell;

                Assert(result->planTree != NULL);
                pgstrom_recursive_grafter(result, NULL, &result->planTree);

                foreach (cell, result->subplans)
                {
                        Plan   **p_subplan = (Plan **) &cell->data.ptr_value;
                        pgstrom_recursive_grafter(result, NULL, p_subplan);
                }
        }
        return result;
}
```

pg_strom 加速例子



```
static void
pgstrom_recursive_grafter(PlannedStmt *pstmt, Plan *parent, Plan **p_curr_plan)
{
        Plan        *plan = *p_curr_plan;
        ListCell    *lc;

        Assert(plan != NULL);

        switch (nodeTag(plan))
        {
                case T_Agg:
                        /*
                         * Try to inject GpuPreAgg plan if cost of the aggregate plan
                         * is enough expensive to justify preprocess by GPU.
                         */
                        pgstrom_try_insert_gpupreagg(pstmt, (Agg *) plan);
                        break;

                case T_SubqueryScan:
```

pg_strom 加速例子

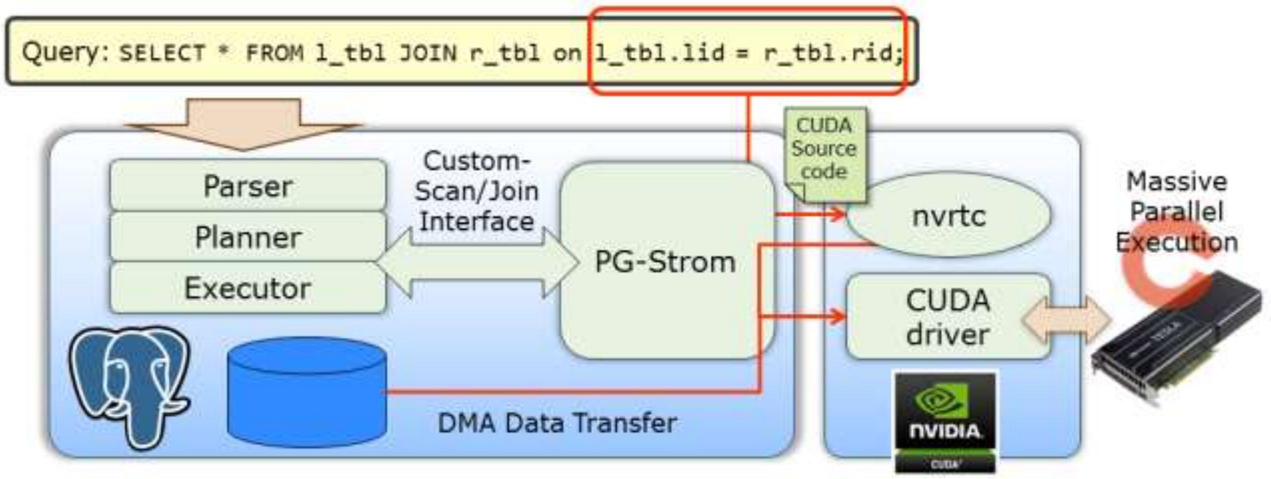
```
case T_SubqueryScan:
        {
                SubqueryScan   *subquery = (SubqueryScan *) plan;
                Plan                 **p_subplan = &subquery->subplan;
                pgstrom_recursive_grafter(pstmt, plan, p_subplan);
        }
        break;
case T_ModifyTable:
        {
                ModifyTable *mtplan = (ModifyTable *) plan;

                foreach (lc, mtplan->plans)
                {
                        Plan  **p_subplan = (Plan **) &lfirst(lc);
                        pgstrom_recursive_grafter(pstmt, plan, p_subplan);
                }
        }
        break;
case T_Append:
        {
                Append *aplan = (Append *) plan;

                foreach (lc, aplan->appendplans)
                {
                        Plan  **p_subplan = (Plan **) &lfirst(lc);
                        pgstrom_recursive_grafter(pstmt, plan, p_subplan);
                }
        }
        break;
case T_MergeAppend:
```

pg_strom 加速例子

```
case T_MergeAppend:
        {
                MergeAppend *maplan = (MergeAppend *) plan;

                foreach (lc, maplan->mergeplans)
                {
                        Plan  **p_subplan = (Plan **) &lfirst(lc);
                        pgstrom_recursive_grafter(pstmt, plan, p_subplan);
                }
        }
        break;
case T_BitmapAnd:
        {
                BitmapAnd  *baplan = (BitmapAnd *) plan;

                foreach (lc, baplan->bitmapplans)
                {
                        Plan  **p_subplan = (Plan **) &lfirst(lc);
                        pgstrom_recursive_grafter(pstmt, plan, p_subplan);
                }
        }
        break;
case T_BitmapOr:
        {
                BitmapOr   *boplan = (BitmapOr *) plan;

                foreach (lc, boplan->bitmapplans)
                {
                        Plan  **p_subplan = (Plan **) &lfirst(lc);
                        pgstrom_recursive_grafter(pstmt, plan, p_subplan);
                }
        }
        break;
```

pg_strom 加速例子

- planner hook，将plan的工作旁路到用户定制的分支处理。
- 用户定制的 planner分支 将生成基于GPU的plan tree (例如dma的数据访问，基于CUDA库的并行计算等)
- executor 执行这个plan tree


Query: SELECT * FROM l_tbl JOIN r_tbl on l_tbl.lid = r_tbl.rid;
Parser
Planner
Executor
Custom-Scan/Join Interface
PG-Strom
CUDA Source code
nvrtc
CUDA driver
Massive Parallel Execution
DMA Data Transfer
NVIDIA
CUDA

路径动态规划

- pgrouting
- http://pgrouting.org/
- http://workshop.pgrouting.org/

bit逻辑运算

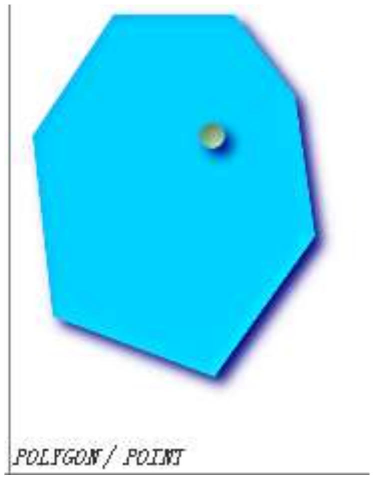
- bit运算
- /*
- * bit_and
- * perform a logical AND on two bit strings.
- */
- Datum
- bit_and(PG_FUNCTION_ARGS)
- {
-         VarBit   *arg1 = PG_GETARG_VARBIT_P(0);
-         VarBit   *arg2 = PG_GETARG_VARBIT_P(1);
-         VarBit   *result;
-         int   len,
-                                 bitlen1,
-                                 bitlen2,
-                                 i;
-         bits8   *p1,
-                            *p2,
-                            *r;

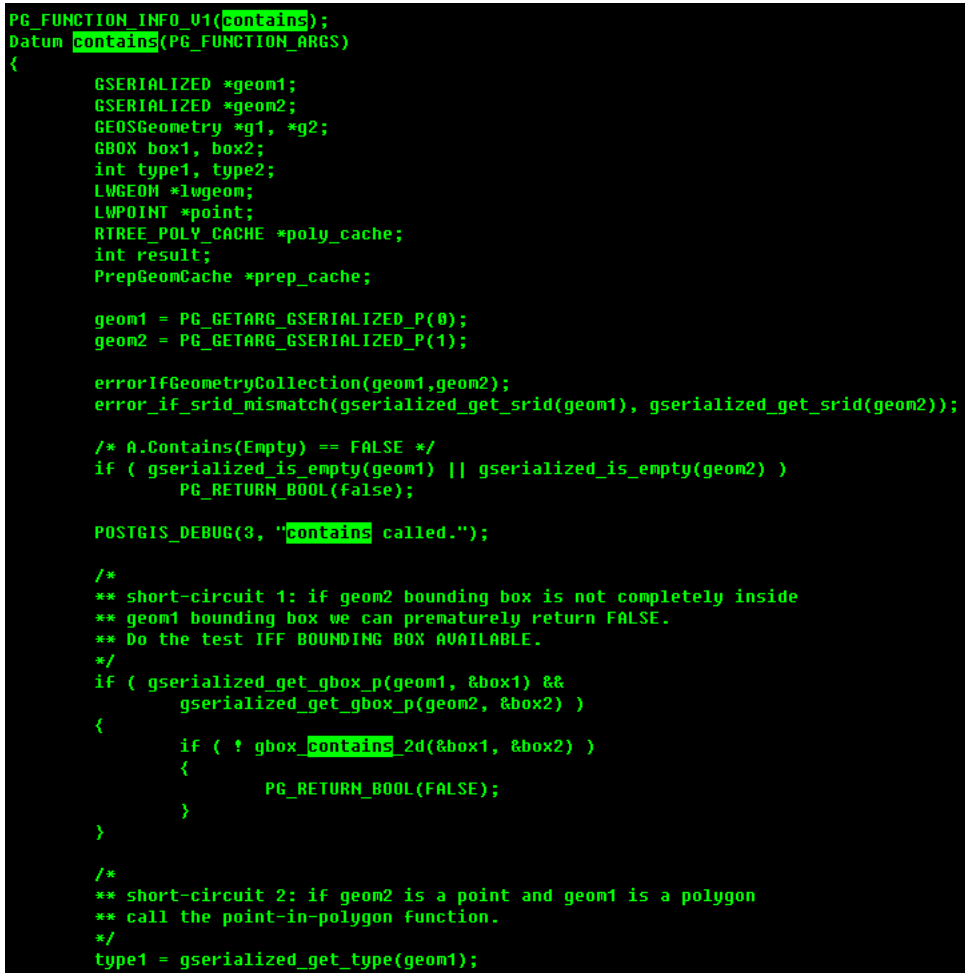
bit逻辑运算

```
•           bitlen1 = VARBITLEN(arg1);
•           bitlen2 = VARBITLEN(arg2);
•           if (bitlen1 != bitlen2)
•               ereport(ERROR,
•                       (errcode(ERRCODE_STRING_DATA_LENGTH_MISMATCH),
•                        errmsg("cannot AND bit strings of different sizes")));

•           len = VARSIZE(arg1);
•           result = (VarBit *) palloc(len);
•           SET_VARSIZE(result, len);
•           VARBITLEN(result) = bitlen1;

•           p1 = VARBITS(arg1);
•           p2 = VARBITS(arg2);
•           r = VARBITS(result);
•           for (i = 0; i < VARBITBYTES(arg1); i++)
•               *r++ = *p1++ & *p2++;

•           /* Padding is not needed as & of 0 pad is 0 */

•           PG_RETURN_VARBIT_P(result);
•       }
```

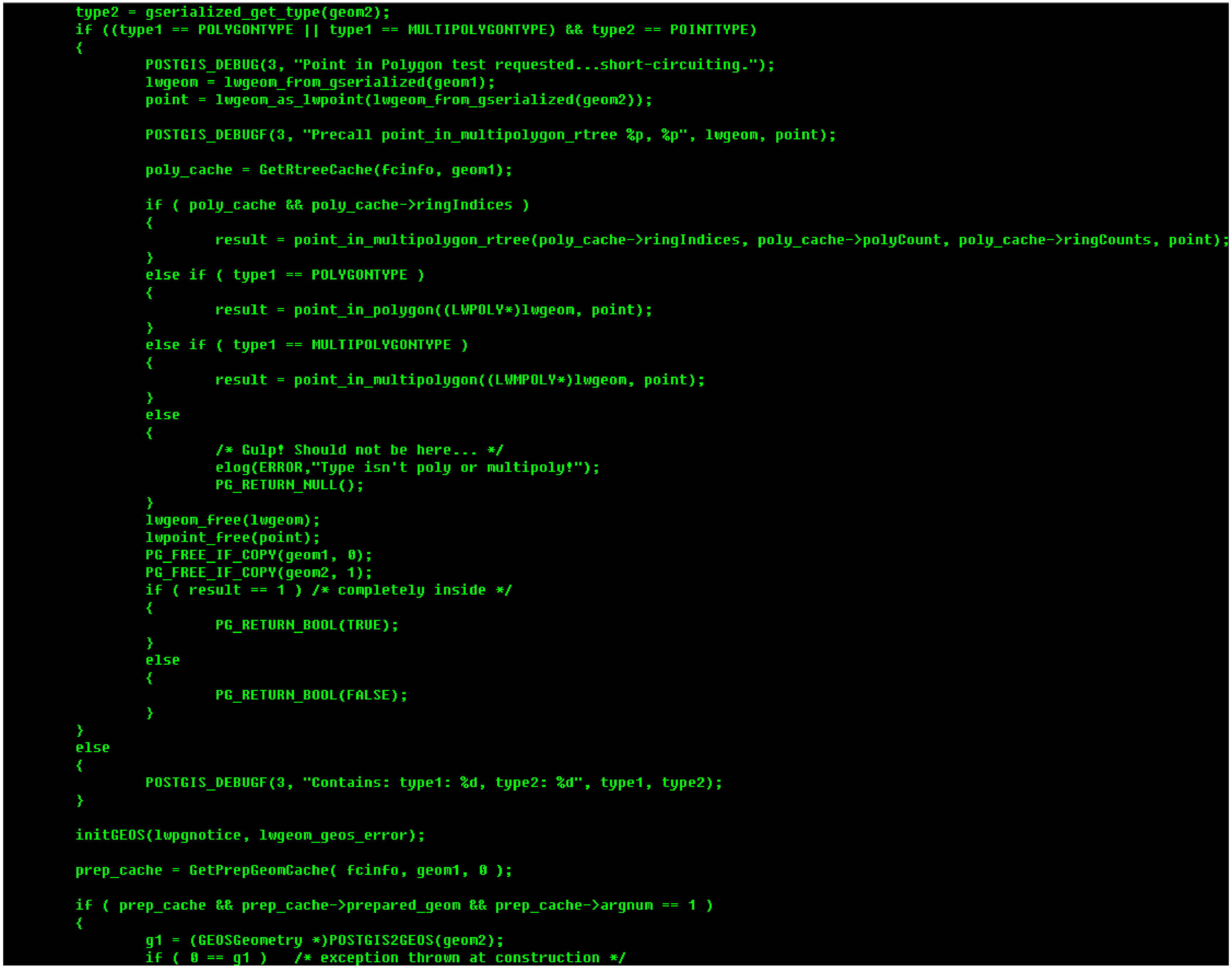
点面判断

- PostGIS
  - http://postgis.net/docs/manual-2.2/ST_Contains.html
  - boolean **ST_Contains**(geometry *geomA*, geometry *geomB*);
  - lwgeom_geos.c

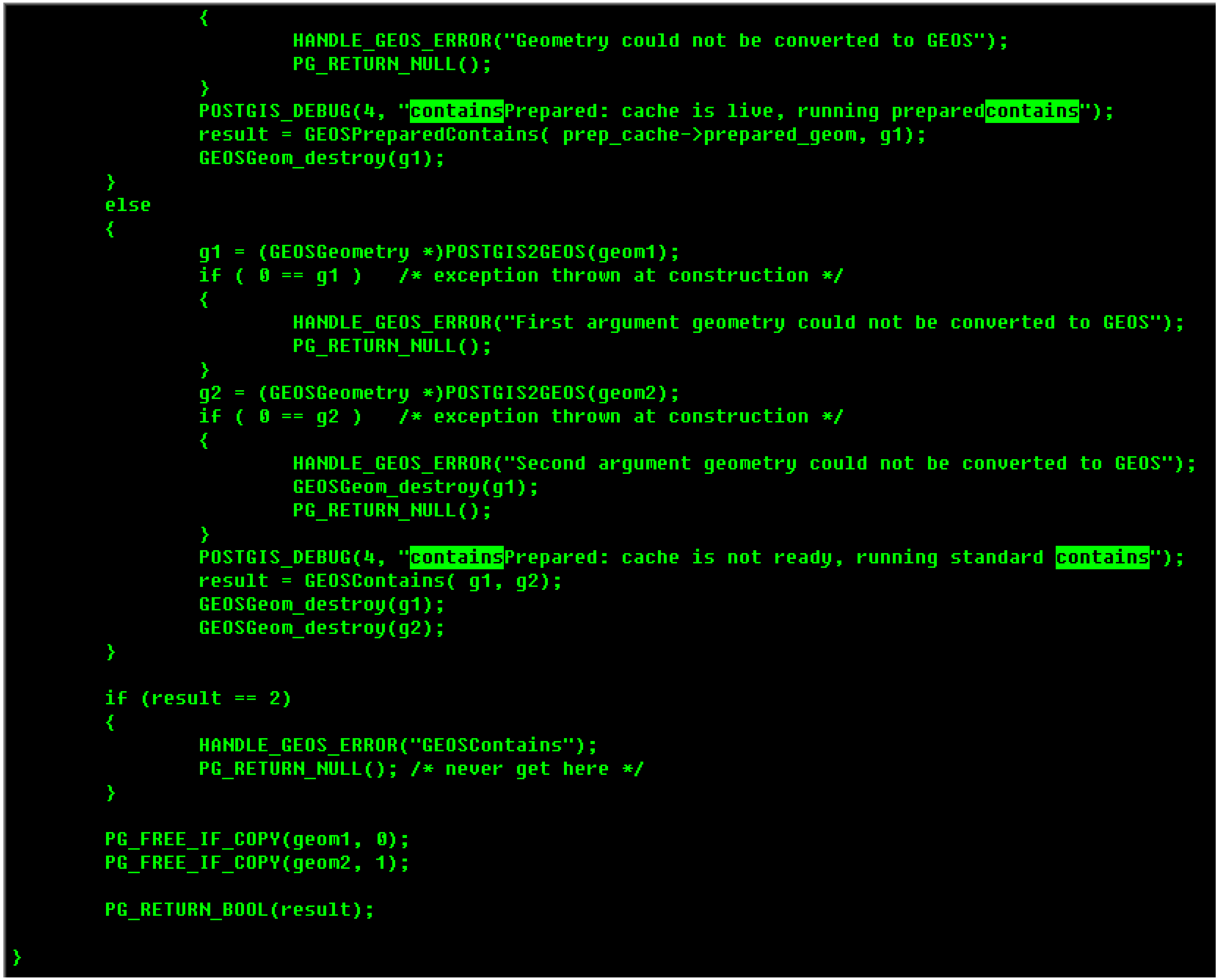




```
PG_FUNCTION_INFO_V1(contains);
Datum contains(PG_FUNCTION_ARGS)
{
        GSERIALIZED *geom1;
        GSERIALIZED *geom2;
        GEOSGeometry *g1, *g2;
        GBOX box1, box2;
        int type1, type2;
        LWGEOM *lwgeom;
        LWPOINT *point;
        RTREE_POLY_CACHE *poly_cache;
        int result;
        PrepGeomCache *prep_cache;

        geom1 = PG_GETARG_GSERIALIZED_P(0);
        geom2 = PG_GETARG_GSERIALIZED_P(1);

        errorIfGeometryCollection(geom1,geom2);
        error_if_srid_mismatch(gserialized_get_srid(geom1), gserialized_get_srid(geom2));

        /* A.Contains(Empty) == FALSE */
        if ( gserialized_is_empty(geom1) || gserialized_is_empty(geom2) )
                PG_RETURN_BOOL(false);

        POSTGIS_DEBUG(3, "contains called.");

        /*
        ** short-circuit 1: if geom2 bounding box is not completely inside
        ** geom1 bounding box we can prematurely return FALSE.
        ** Do the test IFF BOUNDING BOX AVAILABLE.
        */
        if ( gserialized_get_gbox_p(geom1, &box1) &&
                gserialized_get_gbox_p(geom2, &box2) )
        {
                if ( ! gbox_contains_2d(&box1, &box2) )
                {
                        PG_RETURN_BOOL(FALSE);
                }
        }

        /*
        ** short-circuit 2: if geom2 is a point and geom1 is a polygon
        ** call the point-in-polygon function.
        */
        type1 = gserialized_get_type(geom1);
```



```
	type2 = gserialized_get_type(geom2);
	if ((type1 == POLYGONTYPE || type1 == MULTIPOLYGONTYPE) && type2 == POINTTYPE)
	{
		POSTGIS_DEBUG(3, "Point in Polygon test requested...short-circuiting.");
		lwgeom = lwgeom_from_gserialized(geom1);
		point = lwgeom_as_lwpoint(lwgeom_from_gserialized(geom2));

		POSTGIS_DEBUGF(3, "Precall point_in_multipolygon_rtree %p, %p", lwgeom, point);

		poly_cache = GetRtreeCache(fcinfo, geom1);

		if ( poly_cache && poly_cache->ringIndices )
		{
			result = point_in_multipolygon_rtree(poly_cache->ringIndices, poly_cache->polyCount, poly_cache->ringCounts, point);
		}
		else if ( type1 == POLYGONTYPE )
		{
			result = point_in_polygon((LWPOLY*)lwgeom, point);
		}
		else if ( type1 == MULTIPOLYGONTYPE )
		{
			result = point_in_multipolygon((LWMPOLY*)lwgeom, point);
		}
		else
		{
			/* Gulp! Should not be here... */
			elog(ERROR,"Type isn't poly or multipoly!");
			PG_RETURN_NULL();
		}
		lwgeom_free(lwgeom);
		lwpoint_free(point);
		PG_FREE_IF_COPY(geom1, 0);
		PG_FREE_IF_COPY(geom2, 1);
		if ( result == 1 ) /* completely inside */
		{
			PG_RETURN_BOOL(TRUE);
		}
		else
		{
			PG_RETURN_BOOL(FALSE);
		}
	}
	else
	{
		POSTGIS_DEBUGF(3, "Contains: type1: %d, type2: %d", type1, type2);
	}

	initGEOS(lwpgnotice, lwgeom_geos_error);

	prep_cache = GetPrepGeomCache( fcinfo, geom1, 0 );

	if ( prep_cache && prep_cache->prepared_geom && prep_cache->argnum == 1 )
	{
		g1 = (GEOSGeometry *)POSTGIS2GEOS(geom2);
		if ( 0 == g1 )   /* exception thrown at construction */
```



```
                {
                        HANDLE_GEOS_ERROR("Geometry could not be converted to GEOS");
                        PG_RETURN_NULL();
                }
                POSTGIS_DEBUG(4, "containsPrepared: cache is live, running preparedcontains");
                result = GEOSPreparedContains( prep_cache->prepared_geom, g1);
                GEOSGeom_destroy(g1);
        }
        else
        {
                g1 = (GEOSGeometry *)POSTGIS2GEOS(geom1);
                if ( 0 == g1 )   /* exception thrown at construction */
                {
                        HANDLE_GEOS_ERROR("First argument geometry could not be converted to GEOS");
                        PG_RETURN_NULL();
                }
                g2 = (GEOSGeometry *)POSTGIS2GEOS(geom2);
                if ( 0 == g2 )   /* exception thrown at construction */
                {
                        HANDLE_GEOS_ERROR("Second argument geometry could not be converted to GEOS");
                        GEOSGeom_destroy(g1);
                        PG_RETURN_NULL();
                }
                POSTGIS_DEBUG(4, "containsPrepared: cache is not ready, running standard contains");
                result = GEOSContains( g1, g2);
                GEOSGeom_destroy(g1);
                GEOSGeom_destroy(g2);
        }

        if (result == 2)
        {
                HANDLE_GEOS_ERROR("GEOSContains");
                PG_RETURN_NULL(); /* never get here */
        }

        PG_FREE_IF_COPY(geom1, 0);
        PG_FREE_IF_COPY(geom2, 1);

        PG_RETURN_BOOL(result);

}
```

参考资料

- 范例
  - contrib , ...
  - pgxn, github
- 书籍
  - PostgreSQL数据库内核分析
  - PostgreSQL数据库服务端编程
- 网站资料
  - http://blog.163.com/digoal@126/blog/static/163877040201172183022203/