Алгоритмы и структуры данных
PostgreSQL edition

Андрей Бородин,
пишу PostgreSQL для Yandex.Cloud

Core algorithms deployed

320

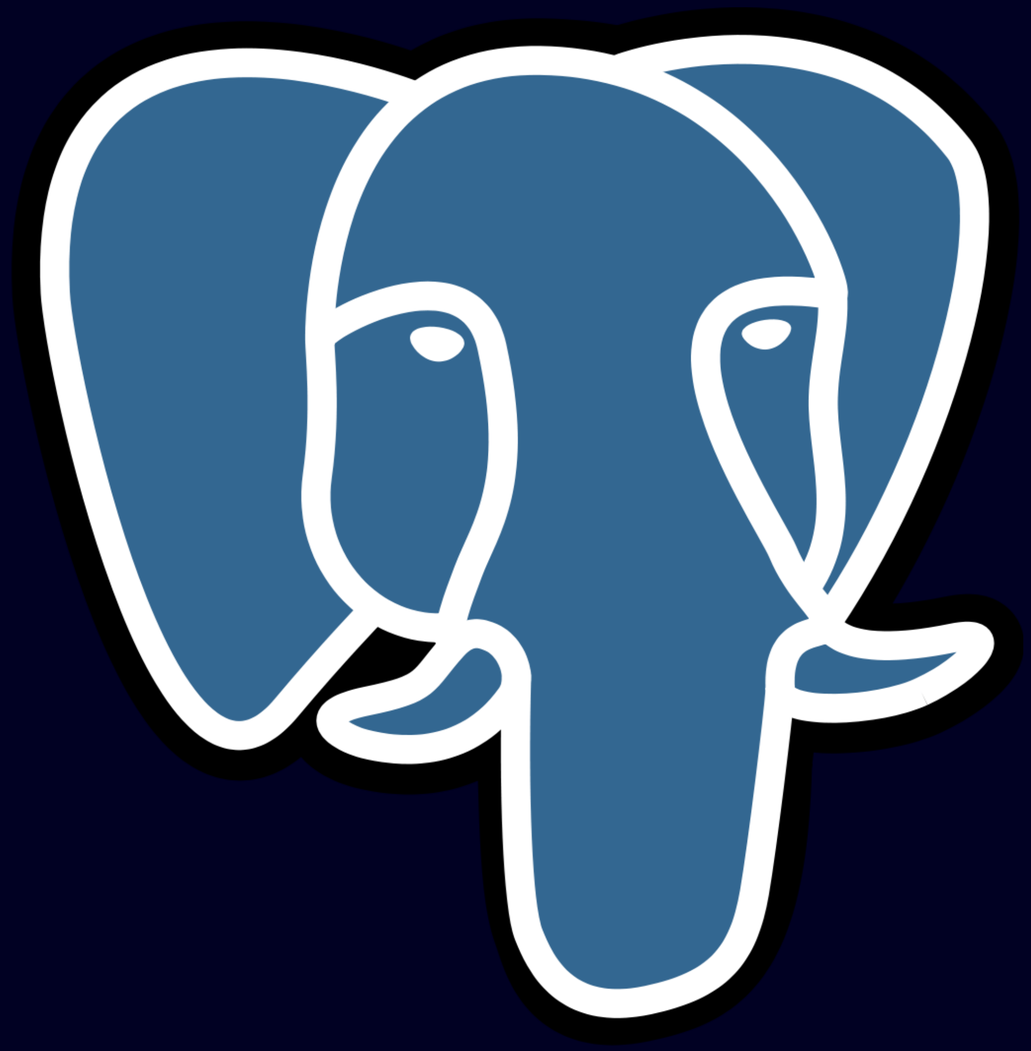
838

To demonstrate the importance of algorithms (e.g. to students and professors who don't do theory or are even from entirely different fields) it is sometimes useful to have ready at hand a list of examples where core algorithms have been deployed in commercial, governmental, or widely-used software/hardware.

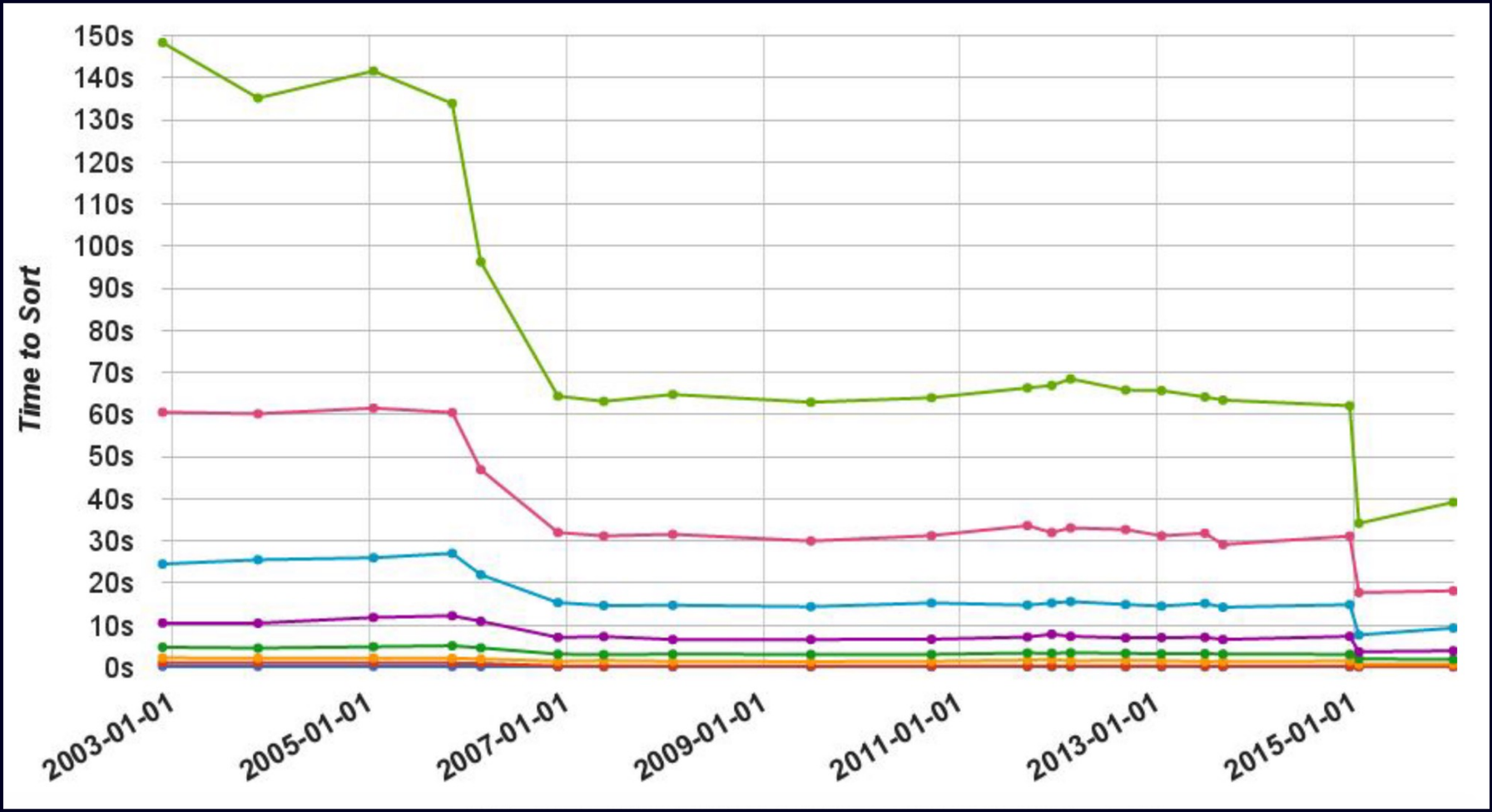
I am looking for such examples that satisfy the following criteria:

1. The software/hardware using the algorithm should be in wide use right now.

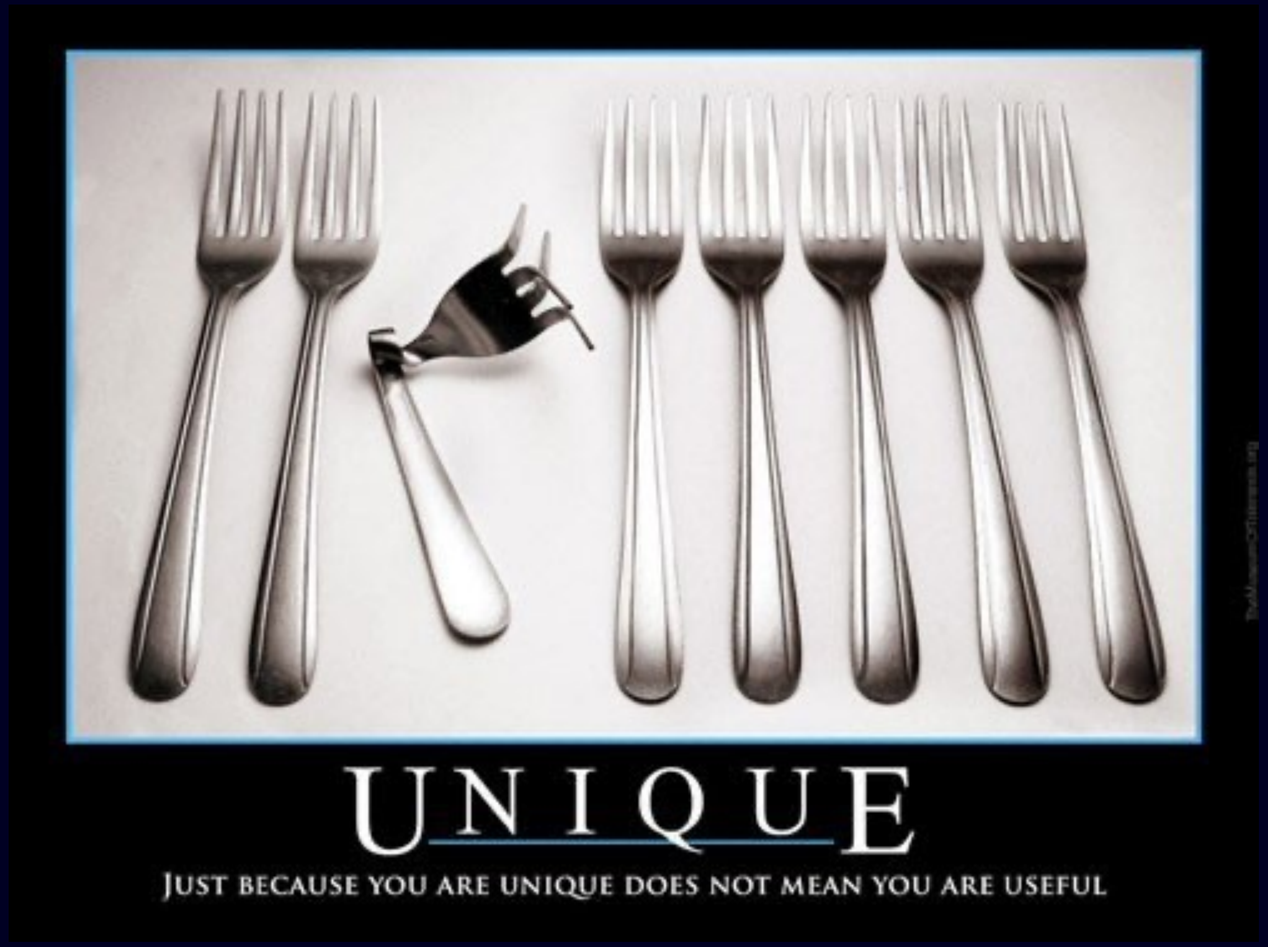
https://cstheory.stackexchange.com/questions/19759/core-algorithms-deployed#answer-19773



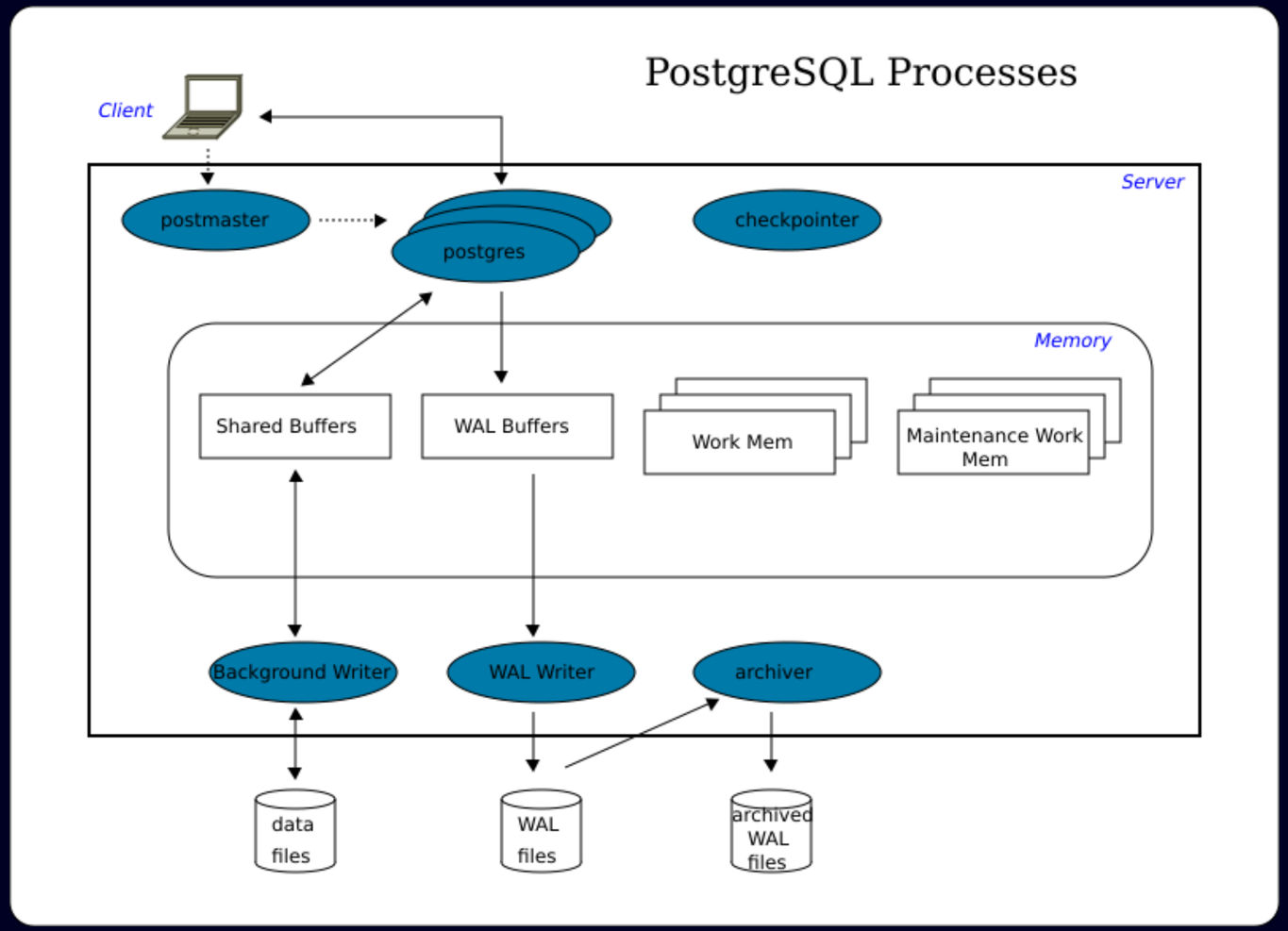
Sorting through ages



https://wiki.postgresql.org/images/5/59/Sorting_through_the_ages.pdf


UNIQUE
JUST BECAUSE YOU ARE UNIQUE DOES NOT MEAN YOU ARE USEFUL

Архитектура



https://en.wikibooks.org/wiki/PostgreSQL/Architecture

Процессная модель

```
597  void function1()
598  {
599      int someVar = 42;
600      function2(someVar);
601  }
602
603  void function2(int someVar)
604  {
605      elog(NOTICE, "%d", someVar);
606  }
607
```

Процессная модель

```
598  int someVar;
599  void function1()
600  {
601      someVar = 42;
602      function2();
603  }
604
605  void function2()
606  {
607      elog(NOTICE, "%d", someVar);
608  }
```

Ошибки

```
374     ereport(FATAL,
375             (errcode(ERRCODE_TOO_MANY_CONNECTIONS),
376              errmsg("too many connections for database \"%s\"",
377                     name)));
```

https://github.com/postgres/postgres/blob/e1025044cd4e7f33f7304aed54d5778b8a82cd5d/src/backend/utils/init/postinit.c#L375

Ошибки

```
26327   #: utils/init/postinit.c:387
26328   #, c-format
26329   msgid "too many connections for database \"%s\""
26330   msgstr "слишком много подключений к БД \"%s\""
```

Ошибки

```
3233    ereport(FATAL,
3234            (errcode(ERRCODE_T_R_SERIALIZATION_FAILURE),
3235             errmsg("terminating connection due to conflict with recovery"),
3236             errdetail_recovery_conflict(),
3237             errhint("In a moment you should be able to reconnect to the"
3238                     " database and repeat your command.")));
```

https://github.com/postgres/postgres/blob/cafde58b337e007cb6a719f5ab4dd6459d932a39/src/backend/tcop/postgres.c#L3172

Ошибки

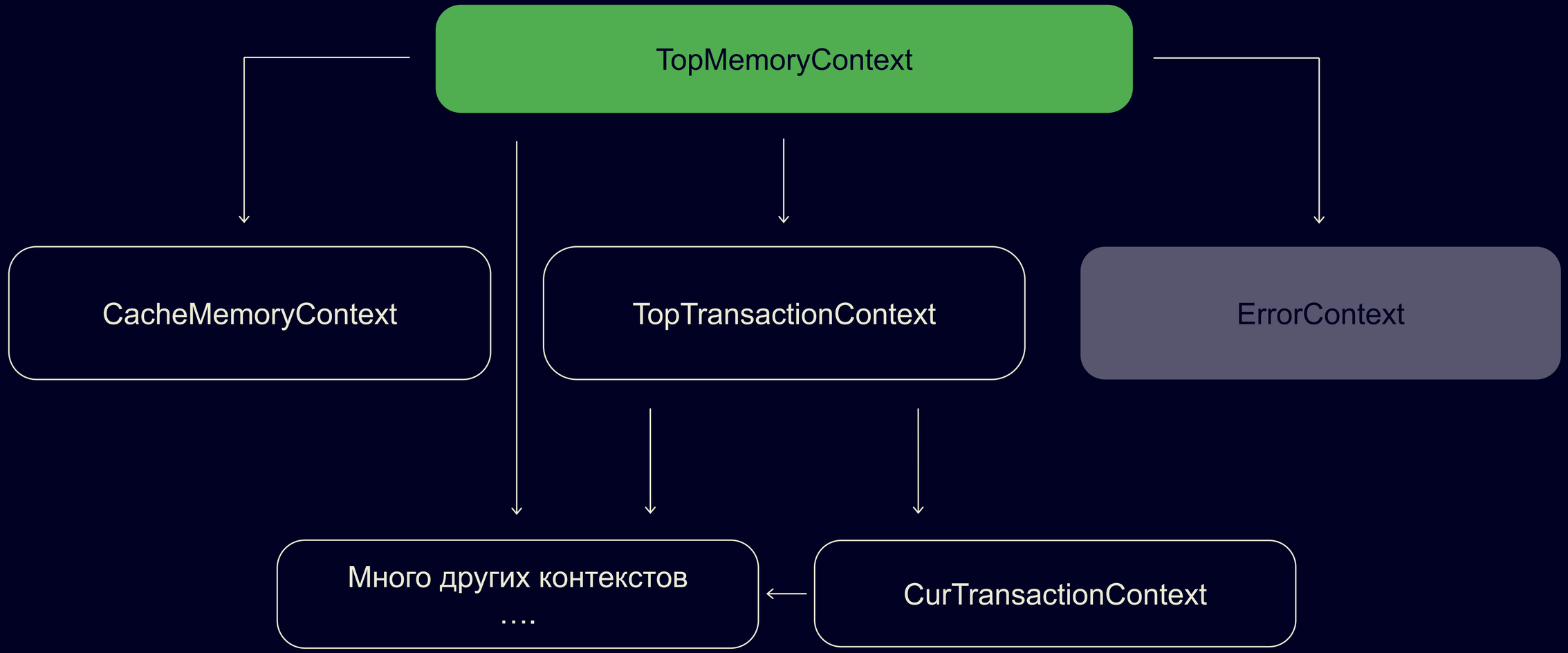
```
788     /* Run the triggers. */
789     PG_TRY();
790     {
791         EventTriggerInvoke(runlist, &trigdata);
792     }
793     PG_FINALLY();
794     {
795         currentEventTriggerState->in_sql_drop = false;
796     }
797     PG_END_TRY();
```

Ошибки

```
313  #define PG_TRY()  \
314      do { \
315          sigjmp_buf *_save_exception_stack = PG_exception_stack; \
316          ErrorContextCallback *_save_context_stack = error_context_stack; \
317          sigjmp_buf _local_sigjmp_buf; \
318          bool _do_rethrow = false; \
319          if (sigsetjmp(_local_sigjmp_buf, 0) == 0) \
320          { \
321              PG_exception_stack = &_local_sigjmp_buf
322
323  #define PG_CATCH()  \
324          } \
325          else \
326          { \
327              PG_exception_stack = _save_exception_stack; \
328              error_context_stack = _save_context_stack
329
```

https://github.com/postgres/postgres/blob/d7cff12c4c035b7cf12bb8454824f48f13018730/src/include/utils/elog.h#L313

Контексты памяти



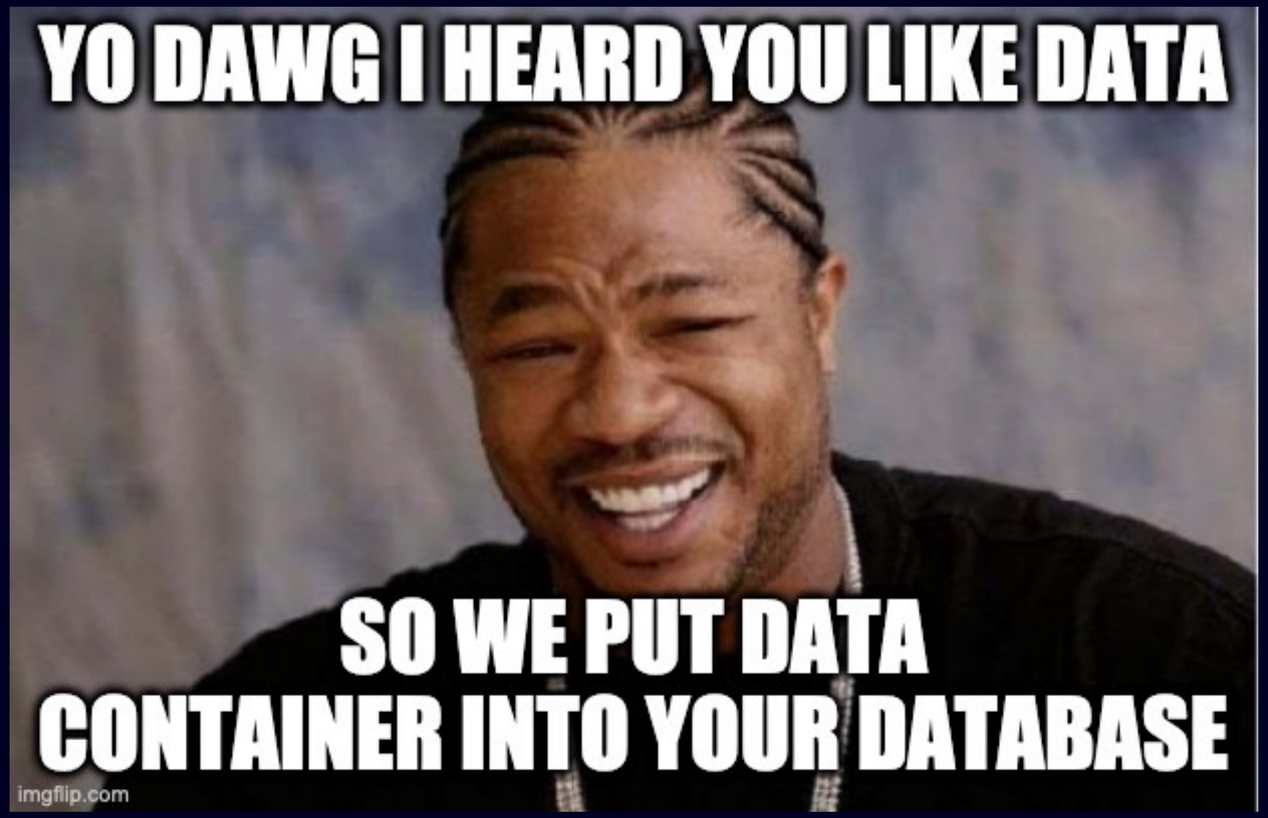
Работа с памятью

```
100     else if (strategy == HStoreExistsStrategyNumber)
101     {
102         text       *query = PG_GETARG_TEXT_PP(0);
103         text       *item;
104
105         *nentries = 1;
106         entries = (Datum *) palloc(sizeof(Datum));
107         item = makeitem(VARDATA_ANY(query), VARSIZE_ANY_EXHDR(query), KEYFLAG);
108         entries[0] = PointerGetDatum(item);
109     }
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/contrib/hstore/hstore_gin.c#L106

Разделяемая память

- shmem
- shmem_toc
- shmem_mq


YO DAWG I HEARD YOU LIKE DATA
SO WE PUT DATA
CONTAINER INTO YOUR DATABASE
imgflip.com

Список

- pg_list.h
- simple_list.h

Список

```
18  ·*·We·support·three·types·of·lists:
19  ·*
20  ·*→ T_List:·lists·of·pointers
21  ·*→ →   (in·practice·usually·pointers·to·Nodes,·but·not·always;
22  ·*→ →   declared·as·"void·*"·to·minimize·casting·annoyances)
23  ·*→ T_IntList:·lists·of·integers
24  ·*→ T_OidList:·lists·of·Oids
```

Список

```
206   #ifndef DEBUG_LIST_MEMORY_USAGE
207       /* Normally, let repalloc deal with enlargement */
208       list->elements = (ListCell *) repalloc(list->elements,
209                                              new_max_len * sizeof(ListCell));
210   #else
211       /*
212        * repalloc() might enlarge the space in-place, which we don't want
213        * for debugging purposes, so forcibly move the data somewhere else.
214        */
215       ListCell   *newelements;
216
217       newelements = (ListCell *)
218           MemoryContextAlloc(GetMemoryChunkContext(list),
219                              new_max_len * sizeof(ListCell));
220       memcpy(newelements, list->elements,
221              list->length * sizeof(ListCell));
222       pfree(list->elements);
223       list->elements = newelements;
224   #endif
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/nodes/list.c#L210

Dynamic chained hash tables

```
68  * Dynamic hashing, after CACM April 1988 pp 446-457, by Per-Ake Larson.
69  * Coded into C, with minor code improvements, and with hsearch(3) interface,
70  * by ejp@ausmelb.oz, Jul 26, 1988: 13:16;
71  * also, hcreate/hdestroy routines added to simulate hsearch(3).
```

```
89  * Modified margo@postgres.berkeley.edu February 1990
90  *→ →   added multiple table interface
91  * Modified by sullivan@postgres.berkeley.edu April 1990
92  *→ →   changed ctl structure for shared memory
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/utils/hash/dynahash.c#L4

Dynamic chained hash tables

- Local or shared
- Partitioned
- Stable pointers

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/utils/hash/dynahash.c#L4

Simplehash

```
241  /* define hashtable mapping block numbers to PagetableEntry's */
242  #define SH_USE_NONDEFAULT_ALLOCATOR
243  #define SH_PREFIX pagetable
244  #define SH_ELEMENT_TYPE PagetableEntry
245  #define SH_KEY_TYPE BlockNumber
246  #define SH_KEY blockno
247  #define SH_HASH_KEY(tb, key) murmurhash32(key)
248  #define SH_EQUAL(tb, a, b) a == b
249  #define SH_SCOPE static inline
250  #define SH_DEFINE
251  #define SH_DECLARE
252  #include "lib/simplehash.h"
```

PerfectHash

```
372  PG_KEYWORD("security", SECURITY, UNRESERVED_KEYWORD, BARE_LABEL)
373  PG_KEYWORD("select", SELECT, RESERVED_KEYWORD, BARE_LABEL)
374  PG_KEYWORD("sequence", SEQUENCE, UNRESERVED_KEYWORD, BARE_LABEL)
375  PG_KEYWORD("sequences", SEQUENCES, UNRESERVED_KEYWORD, BARE_LABEL)
376  PG_KEYWORD("serializable", SERIALIZABLE, UNRESERVED_KEYWORD, BARE_LABEL)
377  PG_KEYWORD("server", SERVER, UNRESERVED_KEYWORD, BARE_LABEL)
378  PG_KEYWORD("session", SESSION, UNRESERVED_KEYWORD, BARE_LABEL)
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/include/parser/kwlist.h

PerfectHash

```
40  /* Perfect hash function for decomposition */
41  static int
42  Decomp_hash_func(const void *key)
43  {
44      static const int16 h[13209] = {
45          0,    1515,  4744,  4745,  0,    0,    0,    0,
46          0,    0,    0,    0,    3890,  3890,  0,    0,
47          3891,  3891,  -2046,  2800,  3890,  3890,  3890,  -4396,
48          4361,  4362,  -4441,  -4441,  -4396,  1773,  1773,  1773,
49          4372,  4373,  -4438,  -4438,  -4393,  -4393,  2619,  17,
50          -4347,  -4393,  -4393,  -4393,  -4393,  -4393,  2619,  2619,
51          1560,  4346,  4347,  4348,  1917,  1873,  1874,  1875,
52          -7856,  4358,  17619,  2622,  2622,  2622,  6357,  6358,
53          6359,  6360,  6361,  6362,  6363,  2622,  -4390,  -4390,
```

Bitmapset

```
29  #define WORDNUM(x)  ((x) / BITS_PER_BITMAPWORD)
30  #define BITNUM(x)   ((x) % BITS_PER_BITMAPWORD)
31
32  #define BITMAPSET_SIZE(nwords)  \
33      (offsetof(Bitmapset, words) + (nwords) * sizeof(bitmapword))
```

Bitmapset

```
1023  /*
1024   * bms_next_member - find next member of a set
1025   *
1026   * Returns smallest member greater than "prevbit", or -2 if there is none.
1027   * "prevbit" must NOT be less than -1, or the behavior is unpredictable.
1028   *
1029   * This is intended as support for iterating through the members of a set.
1030   * The typical pattern is
1031   *
1032   *→ →   →   x = -1;
1033   *→ →   →   while ((x = bms_next_member(inputset, x)) >= 0)
1034   *→ →   →   →   process member x;
1035   *
1036   * Notice that when there are no more members, we return -2, not -1 as you
1037   * might expect.  The rationale for that is to allow distinguishing the
1038   * loop-not-started state (x == -1) from the loop-completed state (x == -2).
1039   * It makes no difference in simple loop usage, but complex iteration logic
1040   * might need such an ability.
1041   */
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/nodes/bitmapset.c#L1043

IntegerSet

Дерево из двух видов узлов

intset_internal_node

intset_leaf_node

```
161  /* Leaf node */
162  typedef struct
163  {
164      uint64      first;      /* first integer in this item */
165      uint64      codeword;   /* simple8b encoded differences from 'first' */
166  } leaf_item;
167
168  #define MAX_VALUES_PER_LEAF_ITEM    (1 + SIMPLE8B_MAX_VALUES_PER_CODEWORD)
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/lib/integerset.c

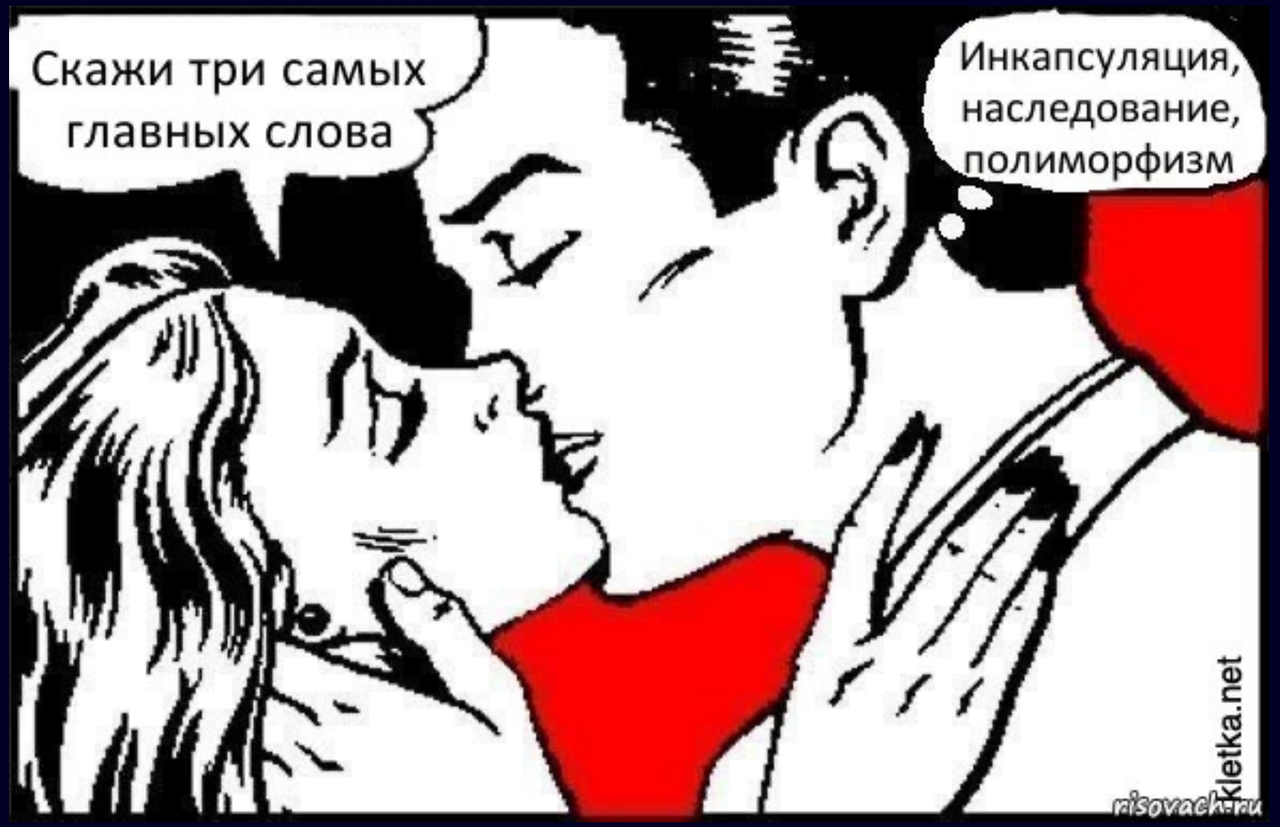
IntegerSet

```
824  static const struct simple8b_mode
825  {
826      uint8       bits_per_int;
827      uint8       num_ints;
828  }           simple8b_modes[17] =
829
830  {
831      {0, 240},           /* mode  0: 240 zeroes */
832      {0, 120},           /* mode  1: 120 zeroes */
833      {1, 60},            /* mode  2: sixty 1-bit integers */
834      {2, 30},            /* mode  3: thirty 2-bit integers */
835      {3, 20},            /* mode  4: twenty 3-bit integers */
836      {4, 15},            /* mode  5: fifteen 4-bit integers */
837      {5, 12},            /* mode  6: twelve 5-bit integers */
838      {6, 10},            /* mode  7: ten 6-bit integers */
839      {7, 8},             /* mode  8: eight 7-bit integers (four bits
840                           * are wasted) */
841      {8, 7},             /* mode  9: seven 8-bit integers (four bits
842                           * are wasted) */
843      {10, 6},            /* mode 10: six 10-bit integers */
844      {12, 5},            /* mode 11: five 12-bit integers */
845      {15, 4},            /* mode 12: four 15-bit integers */
846      {20, 3},            /* mode 13: three 20-bit integers */
847      {30, 2},            /* mode 14: two 30-bit integers */
848      {60, 1},            /* mode 15: one 60-bit integer */
849
850      {0, 0}              /* sentinel value */
851  };
```

https://github.com/postgres/postgres/blob/c30f54ad732ca5c8762bb68bbe0f51de9137dd72/src/backend/lib/integerset.c

SLRU

```
/*-------------------------------------------------------------------------
 *
 * slru.h
 *		Simple LRU buffering for transaction status logfiles
 *
 * Portions Copyright (c) 1996-2021, PostgreSQL Global Development Group
 * Portions Copyright (c) 1994, Regents of the University of California
 *
 * src/include/access/slru.h
 *
 *-------------------------------------------------------------------------
 */
```


Скажи три самых главных слова
Инкапсуляция, наследование, полиморфизм
risovach.ru
Kletka.net


Видишь ООП?
- Нет.
- И я не вижу. А он там есть!

Object oriented programming

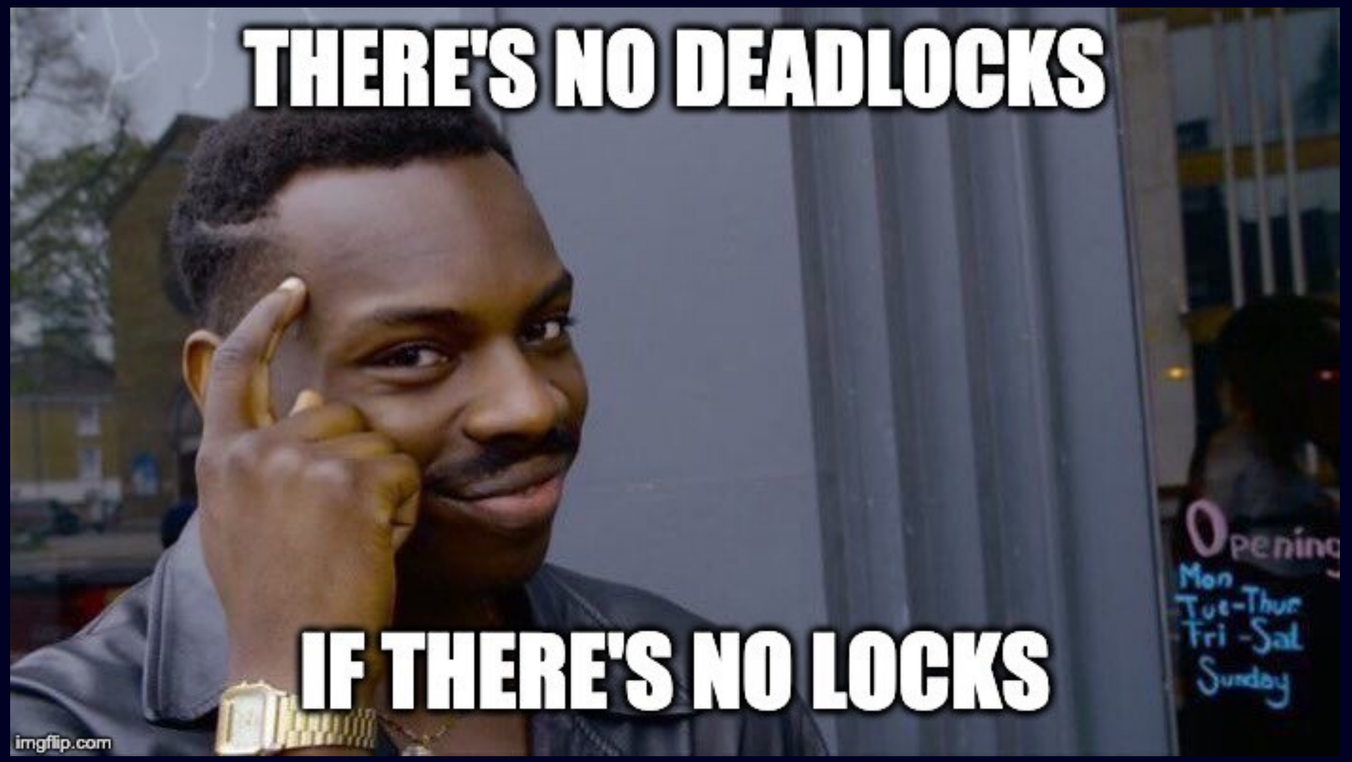
```
2091    else if (IsA(node, NextValueExpr))
2092    {
2093        NextValueExpr *nve = (NextValueExpr *) node;
2094
2095        add_object_address(OCLASS_CLASS, nve->seqid, 0,
2096                           context->addrs);
2097    }
2098    else if (IsA(node, OnConflictExpr))
2099    {
2100        OnConflictExpr *onconflict = (OnConflictExpr *) node;
2101
2102        if (OidIsValid(onconflict->constraint))
2103            add_object_address(OCLASS_CONSTRAINT, onconflict->constraint, 0,
2104                               context->addrs);
2105        /* fall through to examine arguments */
2106    }
```

Object oriented programming

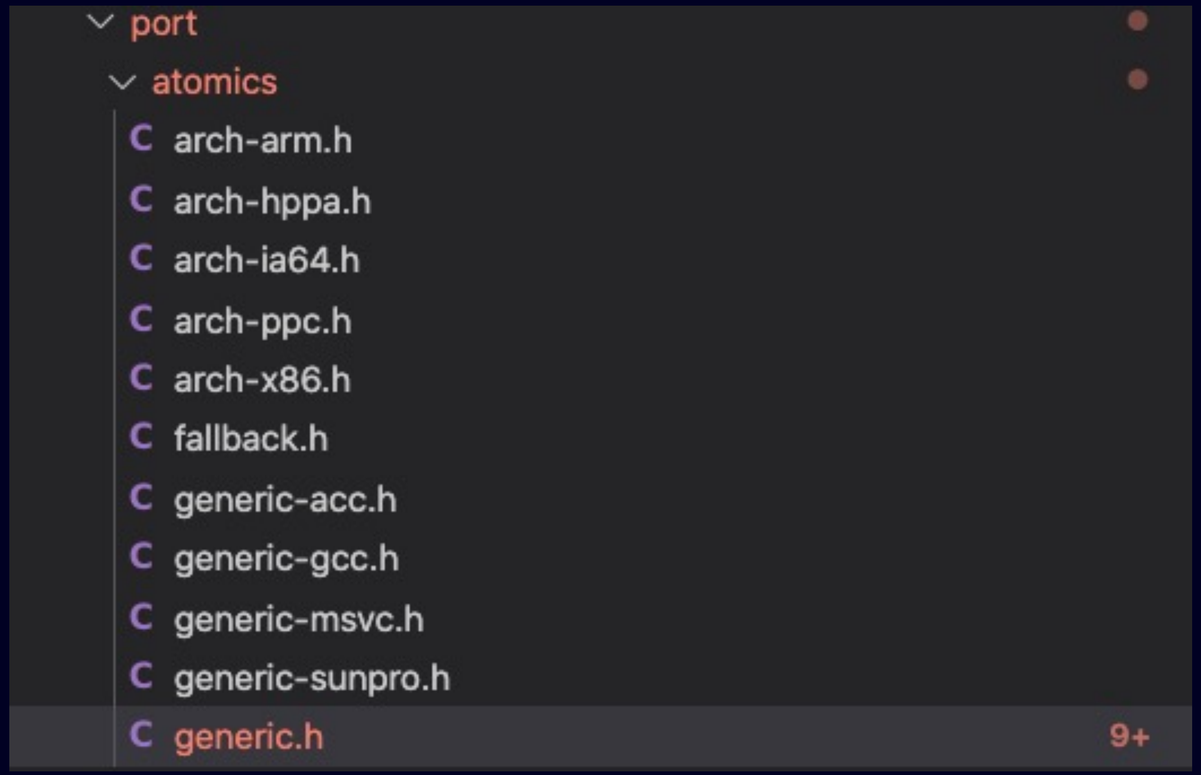
```
590    #define IsA(nodeptr,_type_)    (nodeTag(nodeptr) == T_##_type_)
```

Object oriented programming

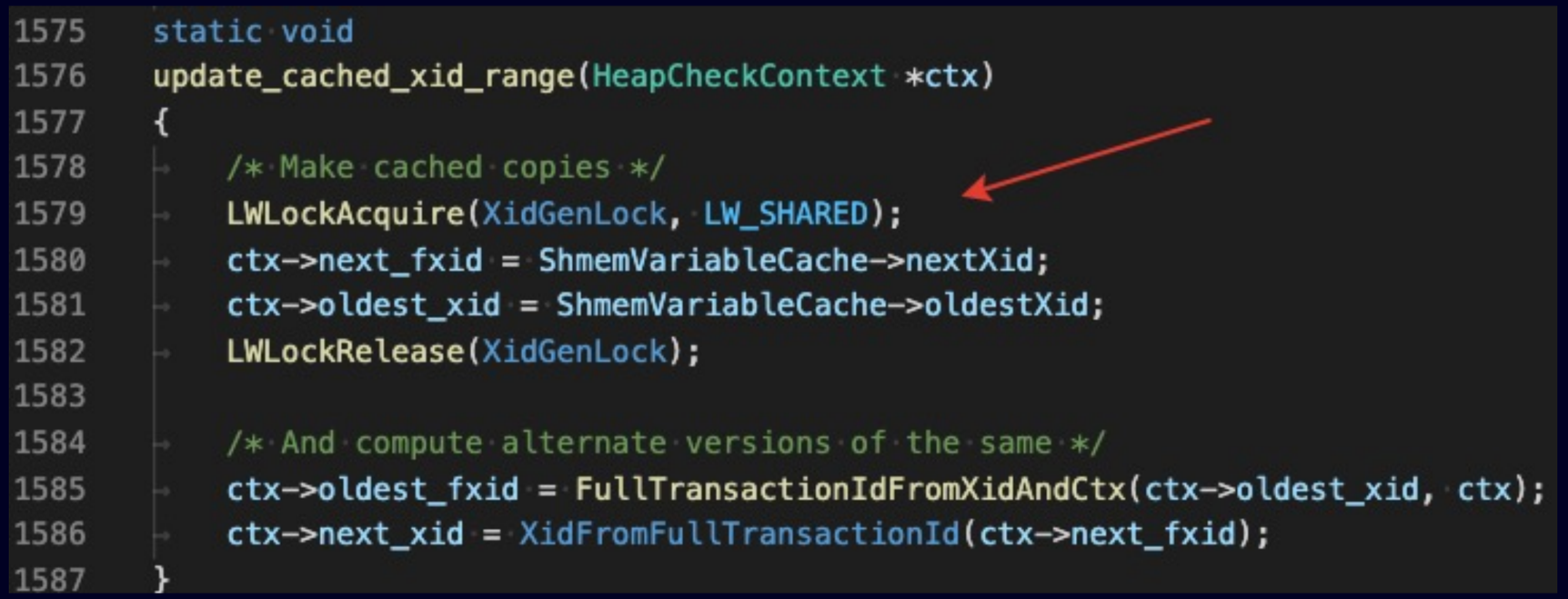
```
26  typedef enum NodeTag
27  {
28      T_Invalid = 0,
29
30      /*
31       * TAGS FOR EXECUTOR NODES (execnodes.h)
32       */
33      T_IndexInfo,
34      T_ExprContext,
35      T_ProjectionInfo,
36      T_JunkFilter,
37      T_OnConflictSetState,
38      T_ResultRelInfo,
39      T_EState,
40      T_TupleTableSlot,
41
42      /*
43       * TAGS FOR PLAN NODES (plannodes.h)
44       */
45      T_Plan,
```


THERE'S NO DEADLOCKS
IF THERE'S NO LOCKS
Opening
Mon
Tue-Thur
Fri -Sat
Sunday
imgflip.com

```
759  static void
760  test_atomic_uint64(void)
761  {
762      pg_atomic_uint64 var;
763      uint64      expected;
764      int         i;
765
766      pg_atomic_init_u64(&var, 0);
767      EXPECT_EQ_U64(pg_atomic_read_u64(&var), 0);
768      pg_atomic_write_u64(&var, 3);
769      EXPECT_EQ_U64(pg_atomic_read_u64(&var), 3);
770      EXPECT_EQ_U64(pg_atomic_fetch_add_u64(&var, pg_atomic_read_u64(&var) - 2),
771                    3);
772      EXPECT_EQ_U64(pg_atomic_fetch_sub_u64(&var, 1), 4);
773      EXPECT_EQ_U64(pg_atomic_sub_fetch_u64(&var, 3), 0);
774      EXPECT_EQ_U64(pg_atomic_add_fetch_u64(&var, 10), 10);
775      EXPECT_EQ_U64(pg_atomic_exchange_u64(&var, 5), 10);
776      EXPECT_EQ_U64(pg_atomic_exchange_u64(&var, 0), 5);
```


port
atomics
arch-arm.h
arch-hppa.h
arch-ia64.h
arch-ppc.h
arch-x86.h
fallback.h
generic-acc.h
generic-gcc.h
generic-msvc.h
generic-sunpro.h
generic.h
9+

Lightweight locks



```
1575  static void
1576  update_cached_xid_range(HeapCheckContext *ctx)
1577  {
1578      /* Make cached copies */
1579      LWLockAcquire(XidGenLock, LW_SHARED);
1580      ctx->next_fxid = ShmemVariableCache->nextXid;
1581      ctx->oldest_xid = ShmemVariableCache->oldestXid;
1582      LWLockRelease(XidGenLock);
1583
1584      /* And compute alternate versions of the same */
1585      ctx->oldest_fxid = FullTransactionIdFromXidAndCtx(ctx->oldest_xid, ctx);
1586      ctx->next_xid = XidFromFullTransactionId(ctx->next_fxid);
1587  }
```

Wait interface

```
                                        среда,  6 мая 2020 г. 11:42:49 (every 1s)

  pid  |         wait_event          | wait_event_type | state  |                                                  query
-------+-----------------------------+-----------------+--------+-----------------------------------------------------------
 33418 | ClientRead                  | Client          | idle   |
 54946 | ClientRead                  | Client          | idle   | insert into t1 select generate_series(1,10000,1)
 59980 | ClientRead                  | Client          | active | select count(*) from (select * from t1 where i > $1 and
 59984 | ClientRead                  | Client          | active | select count(*) from (select * from t1 where i > $1 and
 59987 | ClientRead                  | Client          | active | select count(*) from (select * from t1 where i > $1 and
 59991 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 59994 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 59995 | MultiXactMemberControlLock  | LWLock          | active | select count(*) from (select * from t1 where i > $1 and
 59996 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 59997 | MultiXactOffsetControlLock  | LWLock          | active | select count(*) from (select * from t1 where i > $1 and
 59998 | MultiXactGenLock            | LWLock          | active | select count(*) from (select * from t1 where i > $1 and
 59999 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 60000 | MultiXactGenLock            | LWLock          | active | select count(*) from (select * from t1 where i > $1 and
 60001 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 60003 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 60005 | MultiXactMemberControlLock  | LWLock          | active | select count(*) from (select * from t1 where i > $1 and
 60007 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
 60009 |                             |                 | active | select count(*) from (select * from t1 where i > $1 and
(18 rows)
```

Wait interface

Table 28.10. Wait Events of Type IPC

IPC Wait Event	Description
AppendReady	Waiting for subplan nodes of an Append plan node to be ready.
BackendTermination	Waiting for the termination of another backend.
BackupWaitWalArchive	Waiting for WAL files required for a backup to be successfully archived.
BgWorkerShutdown	Waiting for background worker to shut down.
BgWorkerStartup	Waiting for background worker to start up.
BtreePage	Waiting for the page number needed to continue a parallel B-tree scan to become available.
BufferIO	Waiting for buffer I/O to complete.
CheckpointDone	Waiting for a checkpoint to complete.
CheckpointStart	Waiting for a checkpoint to start.
ExecuteGather	Waiting for activity from a child process while executing a Gather plan node.
HashBatchAllocate	Waiting for an elected Parallel Hash participant to allocate a hash table.
HashBatchElect	Waiting to elect a Parallel Hash participant to allocate a hash table.


Wait Event
Ед. в 5 секунд
500
450
400
350
300
250
200
150
100
50
22:05
22:10
22:15
22:20
22:25
22:30
22:35
22:40
22:45
22:50
22:55
push-postgres_wait_event_CPU_CPU_tmmx
push-postgres_wait_event_Client_ClientRead_tmmx
push-postgres_wait_event_Client_ClientWrite_tmmx
push-postgres_wait_event_IO_DataFileRead_tmmx
push-postgres_wait_event_IO_SLRURead_tmmx
push-postgres_wait_event_IO_WALWrite_tmmx
push-postgres_wait_event_IPC_SyncRep_tmmx
push-postgres_wait_event_LWLock_MultiXactGenLock_tmmx
push-postgres_wait_event_LWLock_MultiXactMemberControlLock_tmmx
push-postgres_wait_event_LWLock_MultiXactOffsetControlLock_tmmx
push-postgres_wait_event_LWLock_WALWriteLock_tmmx
push-postgres_wait_event_LWLock_buffer_content_tmmx
push-postgres_wait_event_LWLock_buffer_io_tmmx
push-postgres_wait_event_LWLock_buffer_mapping_tmmx
push-postgres_wait_event_LWLock_lock_manager_tmmx
push-postgres_wait_event_LWLock_multixact_member_tmmx
push-postgres_wait_event_LWLock_multixact_offset_tmmx
push-postgres_wait_event_Lock_advisory_tmmx
push-postgres_wait_event_Lock_transactionid_tmmx
push-postgres_wait_event_Lock_tuple_tmmx
hosts=
hosts=
hosts=
hosts=
hosts=

Heavyweight lock

- Локи на строки
- Локи на Relation

Heavyweight lock

- Локи на строки
- Локи на Relation

- Deadlock detection
- Partitioned

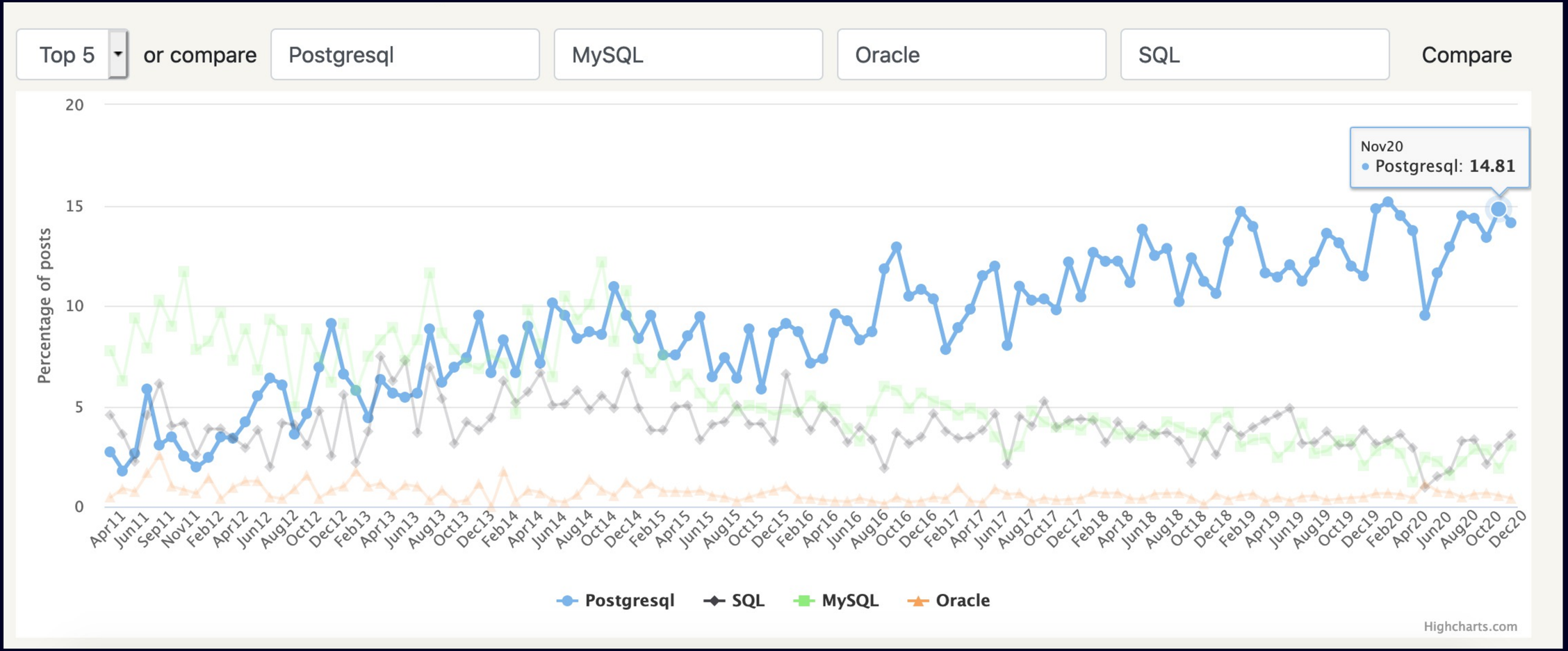
Heavyweight lock

- Локи на строки
- Локи на Relation

Посмотрите в докладе
Unlocking the Postgres Lock Manager. Брюса Момжиана

https://habr.com/ru/post/510322/

Hacker News Hiring Trends



https://www.hntrends.com/2020/dec-year-unlike-any-other-tech-tools-didnt-change-much.html?compare=Postgresql&compare=MySQL&compare=Oracle&compare=SQL

PostgreSQL в Yandex.Cloud

Во внутренней инсталляции

- 6+ Пб данных (июнь 2021)
- 3 миллиона запросов в секунду

Спасибо

Андрей Бородин,
PostgreSQL hacker

x4mmm@yandex-team.ru

@x4mmm