Intro to Postgres Hacking

Tomas Vondra, Microsoft

vondratomas@microsoft.com / tomas@vondra.me

https://vondra.me

OpenAlt 2024, Brno

Microsoft

why?

Resources

Microsoft

- https://www.postgresql.org/developer/coding/
- https://wiki.postgresql.org/wiki/Development_information
  - https://wiki.postgresql.org/wiki/Developer_FAQ
  - https://wiki.postgresql.org/wiki/So,_you_want_to_be_a_developer%3F
- other talks / workshops
  - https://www.neilconway.org/talks/hacking/
  - https://momjian.us/main/presentations/internals.html
  - http://www.interdb.jp/pg/
- multiple books

PostgreSQL Hacking Workshop/Discord

Microsoft

- Mentoring
  - http://rhaas.blogspot.com/2024/07/mentoring-program-updates.html
- Hacking Workshop
  - http://rhaas.blogspot.com/2024/07/postgresql-hacking-workshop-august-2024.html
- Discord channel
  - https://discord.com/invite/yMbds24D8n

Resources

- pgconf.dev 2024 (Vancouver, May)
  - https://www.pgevents.ca/events/pgconfdev2024/schedule/
  - https://www.youtube.com/@pgconfdev
- pgconf.eu 2024 (Athens, October)
  - https://www.postgresql.eu/events/pgconfeu2024/schedule/
  - https://www.youtube.com/@pgeu

Prague PostgreSQL Developer Day 2025 (P2D2)

Microsoft

https://p2d2.cz/call-for-papers / https://p2d2.cz/call-for-sponsors

Agenda

- (very) brief intro to dev process
  - basic development (clone, build, test)
- advanced
  - github CI
  - ccache, gdb, valgrind, address sanitizer, rr, ...
- reviewing patches
  - testing, meld
- specific infrastructure / idioms
  - Datum, varlena
  - memory management (palloc, contexts)
  - elog/ereport

intro to dev process

Microsoft

- cloning repository
- configure & make
- running tests
  - basic regression tests (make check)
  - full tests (make check-world)
- github CI setup
- cfbot

IDE or no IDE?

- geany
- git grep
- terminal
- VSCode, CLion, emacs

Microsoft

cloning repository

Microsoft

- fork and clone repository on github
  - https://github.com/postgres/postgres
  - git clone
- `./configure --help`
- ```
  ./configure --prefix=/home/user/builds/pg \
      --enable-debug --enable-depend --enable-cassert \
      --enable-tap-tests CPPFLAGS="-O0 -ggdb3"
  ```
- `make -s -j4`
  ```

running tests

- `make check`
- `make installcheck`
- `make check-world`

Microsoft

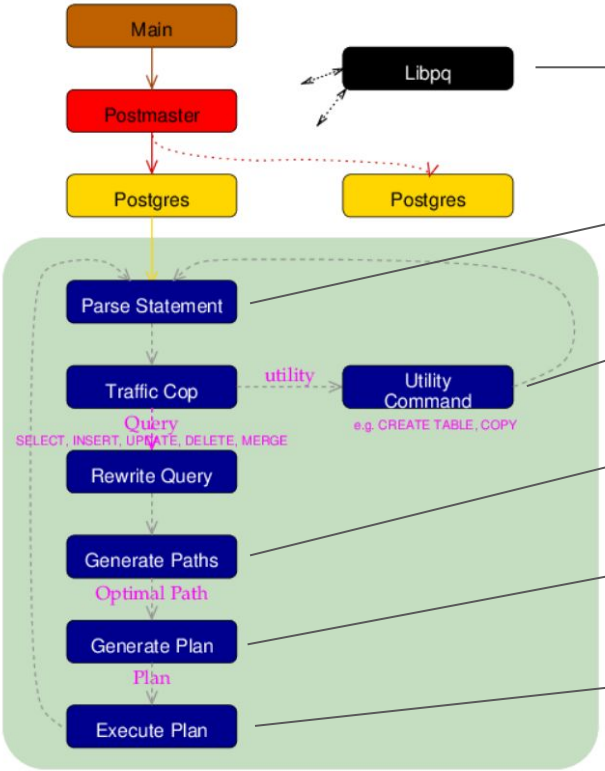
repository structure

```
-rw-r--r--.  1 user user   1192 Oct 15 23:58 COPYRIGHT
-rw-r--r--.  1 user user   4288 Nov  1 18:11 GNUmakefile
-rw-r--r--.  1 user user   4288 Nov  1 18:10 GNUmakefile.in
-rw-r--r--.  1 user user    277 Oct 15 23:58 HISTORY
-rw-r--r--.  1 user user   1875 Nov  1 18:10 Makefile
-rw-r--r--.  1 user user   1213 Nov  1 18:10 README
-rw-r--r--.  1 user user    721 Nov  1 18:10 README.git
-rw-r--r--.  1 user user    365 Nov  1 00:43 aclocal.m4
drwxr-xr-x.  2 user user   4096 Nov  1 18:10 config
-rw-r--r--.  1 user user 369370 Nov  1 18:11 config.log
-rwxr-xr-x.  1 user user  39574 Nov  1 18:11 config.status
-rwxr-xr-x.  1 user user 584716 Nov  1 18:10 configure
-rw-r--r--.  1 user user  87336 Nov  1 18:10 configure.ac
drwxr-xr-x. 61 user user   4096 Nov  1 18:10 contrib
drwxr-xr-x.  3 user user   4096 Nov  1 18:10 doc
-rw-r--r--.  1 user user 107061 Nov  1 18:11 meson.build
-rw-r--r--.  1 user user   6266 Nov  1 18:10 meson_options.txt
drwxr-xr-x. 16 user user   4096 Nov  1 18:11 src
```

repository structure ./src

```
-rw-r--r--.  1 user user   173 Oct 10 16:32 DEVELOPERS
-rw-r--r--.  1 user user  1858 Nov  1 18:10 Makefile
-rw-r--r--.  1 user user 35795 Nov  1 18:11 Makefile.global
-rw-r--r--.  1 user user 35880 Nov  1 18:10 Makefile.global.in
lrwxrwxrwx.  1 user user    31 Nov  1 18:11 Makefile.port -> ../src/makefiles/Makefile.linux
-rw-r--r--.  1 user user 15048 Nov  1 18:10 Makefile.shlib
drwxr-xr-x. 30 user user  4096 Nov  1 18:12 backend
drwxr-xr-x. 24 user user  4096 Nov  1 18:10 bin
drwxr-xr-x.  4 user user 12288 Nov  1 18:11 common
drwxr-xr-x.  3 user user  4096 Nov  1 18:12 fe_utils
drwxr-xr-x. 34 user user  4096 Nov  1 18:11 include
drwxr-xr-x.  4 user user  4096 Nov  1 18:10 interfaces
drwxr-xr-x.  2 user user  4096 Nov  1 18:10 makefiles
-rw-r--r--.  1 user user  1295 Nov  1 18:10 meson.build
-rw-r--r--.  1 user user  6885 Nov  1 18:10 nls-global.mk
drwxr-xr-x.  6 user user  4096 Nov  1 18:10 pl
drwxr-xr-x.  3 user user  4096 Nov  1 18:11 port
drwxr-xr-x.  2 user user  4096 Nov  1 18:10 template
drwxr-xr-x. 16 user user  4096 Nov  1 18:10 test
drwxr-xr-x.  5 user user  4096 Nov  1 18:11 timezone
drwxr-xr-x. 10 user user  4096 Nov  1 18:11 tools
drwxr-xr-x.  2 user user  4096 Nov  1 18:10 tutorial
```

backend flowchart [https://www.postgresql.org/developer/backend/]



src/backend/libpq [src/bin/psql]

src/backend/parser

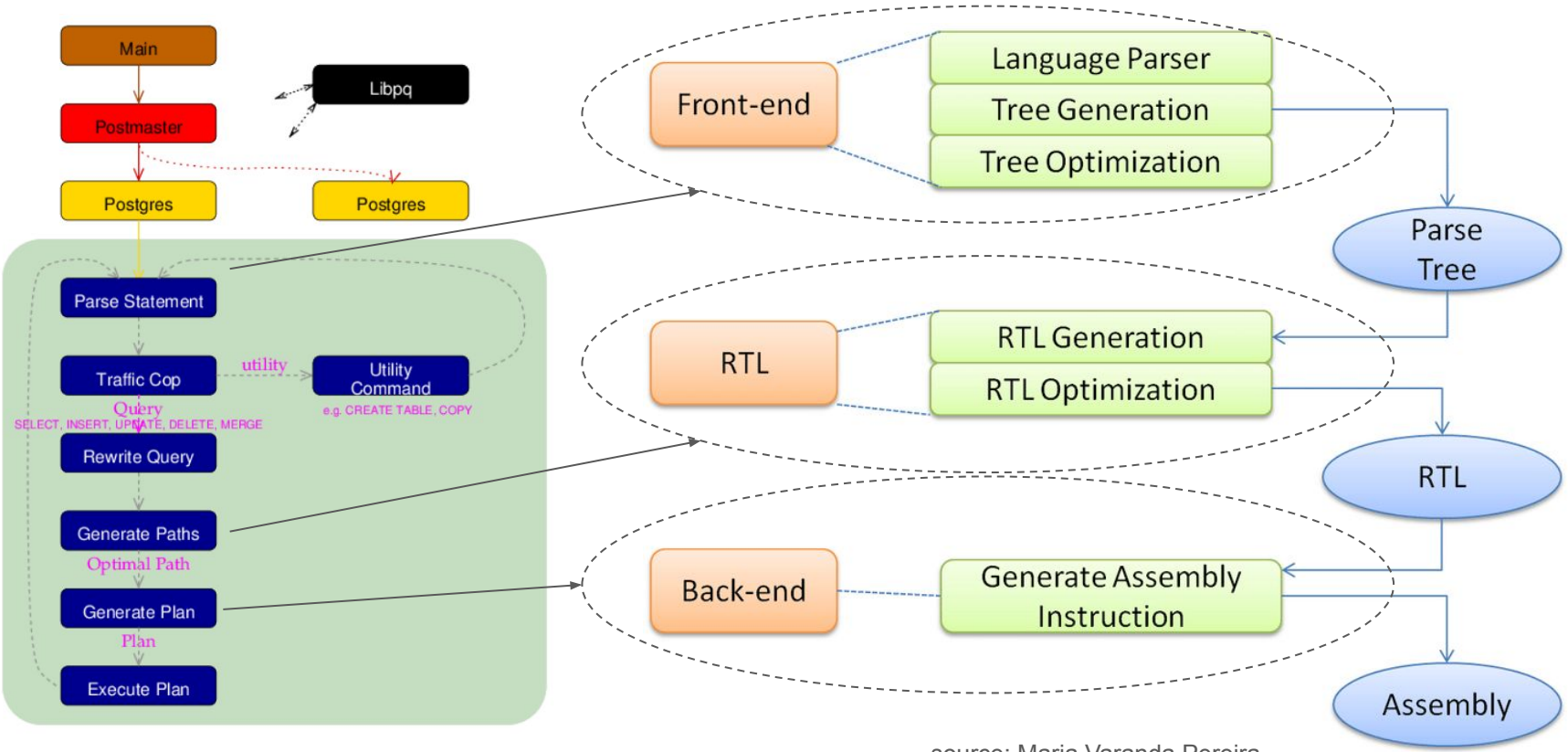
src/backend/command

src/backend/optimiser/path

src/backend/optimiser/plan

src/backend/executor

backend flowchart vs. GCC architektura



source: Maria Varanda Pereira

Microsoft

hello world

hello_world()

Microsoft

- add new "built-in" function
- no arguments, returns "text"
- C function + catalog definition
  - same result as CREATE FUNCTION, but during initdb
- src/include/catalog/
  - all catalogs have stuff here, 1:1 name matching
- pg_proc -> pg_catalog.pg_proc
  - pg_proc.h - essentially "struct" definition (FormData_pg_proc)
  - pg_proc_d.h - generated automatically, attribute numbers (for parsing rows)
  - pg_proc.dat - contents (all predefined procedures)
- src/backend/catalog/system_{constraints,functions,views}.sql

hello_world()

Microsoft

- find a definition for "similar" function, copy and modify

```
{ oid => '877', descr => 'extract portion of string',
  proname => 'substr', prorettype => 'text', proargtypes => 'text int4 int4',
  prosrc => 'text_substr' },

{ oid => '878', descr => 'hello world function',
  proname => 'hello_world', prorettype => 'text', proargtypes => '',
  prosrc => 'hello_world_1' },
```

- try building
  - make -s

hello_world()

```
$ make -s
Duplicate OIDs detected:
878
found 1 duplicate OID(s) in catalog data
make[2]: *** [Makefile:141: bki-stamp] Error 2
make[1]: *** [Makefile:121: submake-catalog-headers] Error 2
make: *** [src/Makefile.global:385: submake-generated-headers] Error 2
```

- ./src/include/catalog/duplicate_oids
- ./src/include/catalog/unused_oids
  - try changing OID to 816 (or whatever unused_oids suggests)

hello_world()

```
$ make -s
/usr/bin/ld: utils/fmgrtab.o:(.rodata+0x77a0): undefined reference to
`hello_world_1'
collect2: error: ld returned 1 exit status
make[2]: *** [Makefile:67: postgres] Error 1
make[1]: *** [Makefile:42: all-backend-recurse] Error 2
make: *** [GNUmakefile:11: all-src-recurse] Error 2
```

- git grep text_substr
- src/backend/utils/adt/varlena.c

hello_world()

```
Datum
hello_world_1(PG_FUNCTION_ARGS)
{
    PG_RETURN_TEXT_P(cstring_to_text("Hello World!"));
}
```

- make -s install
- pg_ctl -D /tmp/data init
- pg_ctl -D /tmp/data -l pg.log start
- createdb test
- psql test
  - SELECT hello_world();

hello_world()

Microsoft

- git grep substr -- *.sql
- src/test/regress/sql/strings.sql
- add new test at the end
- make check
- compare
  - meld src/test/regress/expected/strings.out src/test/regress/results/strings.out
- fix expected output
- make check

Microsoft

github CI

src/tools/ci/README

```
Enabling cirrus-ci in a github repository
=========================================

To enable cirrus-ci on a repository, go to
https://github.com/marketplace/cirrus-ci and select "Public
Repositories". Then "Install it for free" and "Complete order". The next page
allows to configure which repositories cirrus-ci has access to. Choose the
relevant repository and "Install".

See also https://cirrus-ci.org/guide/quick-start/

Once enabled on a repository, future commits and pull-requests in that
repository will automatically trigger CI builds. These are visible from the
commit history / PRs, and can also be viewed in the cirrus-ci UI at
https://cirrus-ci.com/github/<username>/<reponame>/

Hint: all build log files are uploaded to cirrus-ci and can be downloaded
from the "Artifacts" section from the cirrus-ci UI after clicking into a
specific task on a build's summary page.
```

Microsoft

http://cfbot.cputube.org
https://commitfest.postgresql.org

Microsoft

coding idioms

Datum

Microsoft

- C and SQL have completely different data types
- Datum = "arbitrary data type"
- sizeof(Datum) == sizeof(void *)
- requires context
  - pg_type -> typbyval, typlen
- typbyval=true (typlen > 0, typlen <= sizeof(Datum))
  - integers, float, ...
- typlen > 0
  - uuid, macaddr, point, ...
- typlen < 0
  - varlena (-1), cstring (-2)

int8, int16, int32, int64, uint...

Microsoft

- C: https://en.wikipedia.org/wiki/C_data_types

  https://en.wikipedia.org/wiki/C_data_types

  e.g. "int" is "minimum 16 bits"
- problem for storage, 32/64 bit platforms, ...
- types with explicit length for portability / clarity
- don't confuse with SQL types
  - int2, int4, int8
- INT64_FORMAT, UINT64_FORMAT

DatumGet* macros

```
src/include/postgres.h:#define DatumGetBool(X) ((bool) ((X) != 0))
src/include/postgres.h:#define DatumGetChar(X) ((char) (X))
src/include/postgres.h:#define DatumGetUInt8(X) ((uint8) (X))
src/include/postgres.h:#define DatumGetInt16(X) ((int16) (X))
src/include/postgres.h:#define DatumGetUInt16(X) ((uint16) (X))
src/include/postgres.h:#define DatumGetInt32(X) ((int32) (X))
src/include/postgres.h:#define DatumGetUInt32(X) ((uint32) (X))
src/include/postgres.h:#define DatumGetObjectId(X) ((Oid) (X))
src/include/postgres.h:#define DatumGetTransactionId(X) ((TransactionId) (X))
src/include/postgres.h:#define DatumGetCommandId(X) ((CommandId) (X))
src/include/postgres.h:#define DatumGetPointer(X) ((Pointer) (X))
src/include/postgres.h:#define DatumGetCString(X) ((char *) DatumGetPointer(X))
src/include/postgres.h:#define DatumGetName(X) ((Name) DatumGetPointer(X))
src/include/postgres.h:#define DatumGetInt64(X) ((int64) (X))
src/include/postgres.h:#define DatumGetInt64(X) (* ((int64 *) DatumGetPointer(X)))
src/include/postgres.h:#define DatumGetUInt64(X) ((uint64) (X))
src/include/postgres.h:#define DatumGetUInt64(X) (* ((uint64 *) DatumGetPointer(X)))
```

????

PG_FUNCTION_ARGS

Microsoft

- C and SQL have completely different calling conventions
  - overloading
  - default values
  - null values (C "null" != SQL "NULL")
- infrastructure to map these conventions
  - PG_FUNCTION_ARGS (fcinfo)
  - PG_GETARG_* macros
  - PG_ARGISNULL

PG_GETARG_* macros

```
src/include/fmgr.h:#define PG_GETARG_DATUM(n)   (fcinfo->args[n].value)
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)
```

varlena

```
/*
 * cstring_to_text_with_len
 *
 * Same as cstring_to_text except the caller specifies the string length;
 * the string need not be null_terminated.
 */
text *
cstring_to_text_with_len(const char *s, int len)
{
    text    *result = (text *) palloc(len + VARHDRSZ);

    SET_VARSIZE(result, len + VARHDRSZ);
    memcpy(VARDATA(result), s, len);

    return result;
}
```

varlena

- used to store all variable-length types (typlen = -1)
  - text, bytea, record, json, jsonb, ...
- essentially buffer with length at the beginning
- basics
  - calculate needed space, account for header (VARHDRSZ)
  - allocate the buffer (palloc)
  - store the length at the beginning (SET_VARSIZE)
  - copy the data in (VARDATA is pointer right after the header)
- advanced: shorter headers, detoasting

elog / ereport

Microsoft

- elog - simple internal logging
  - not meant for users (can't happen cases)
  - no translations
  - `elog(WARNING, "hello world");`
  - `elog(LOG, "hello world");`
- ereport - better user-facing messages
  - translations (gettext), hints for users, details, ...
  - ```
    ereport(ERROR,
            (errcode(ERRCODE_NULL_VALUE_NOT_ALLOWED),
             errmsg("JSON value must not be null"),
             errdetail("Exception was raised because null_value_treatment is ..."),
             errhint("To avoid, either change the null_value_treatment ...")));
    ```
    ```

--enable-casserts

- runtime checks
- Assert(condition);
- fails (with ABORT) if not true
- super useful
- enforcing (complex) invariants
- expensive, debug builds only
- `sudo sysctl -w kernel.core_pattern=/home/user/cores/core.%p`

gdb config

- don't optimize stuff too much
  - `CPPFLAGS="-O0 -ggdb3"`
- allow long strings
  - `set print elements number-of-elements`
- disable USER1 signal
  - `handle SIGUSR1 noprint nostop`
- get PID for the backend, attach gdb to it
  - `SELECT pg_backend_pid()`
- breakpoints

memory management

Microsoft

```
MemoryContext ctx;
MemoryContext oldctx;

ctx = AllocSetContextCreate(CurrentMemoryContext,
                            "my temporary context",
                            ALLOCSET_DEFAULT_SIZES);

oldcxt = MemoryContextSwitchTo(ctx);

ptr = palloc(...);
ptr = palloc(...);
ptr = palloc(...);

MemoryContextSwitchTo(oldcxt);
MemoryContextDelete(ctx);
```

memory management

Microsoft

- hierarchy of memory contexts
  - different life spans: TopMemoryContext, query, row, plan operation, ...
- CurrentMemoryContext, MemoryContextSwitchTo
  - palloc()
- MemoryContextStats()
  - can be called from gdb if needed
  - sometimes called in OOM situations
- problem with libraries (call malloc)

Node *

- C doesn't have classes / polymorphism
  - but it's a convenient / useful abstraction
- e.g. all operations in relation algebra are "similar"
  - consume 1+ relations, produce relation
  - may be composed into execution plan (tree of nodes)
- but classes can be simulated using C structs
  - src/include/nodes/{parsenodes,pathnodes,plannodes,...}
- nodeToString()

gram.y

Microsoft

- grammar definition
- Bison generates parser
- expressions (queries, ...) mapped to parse nodes
  - src/include/nodes/parsenodes.h
- usually not very difficult if you know what you want
  - figuring out a "good" syntax is hard
  - also hard - fixing shift/reduce conflicts (ambiguities)
- What to do after parsing?
  - https://www.postgresql.org/developer/backend/
  - utility commands are fairly simple (no planning)

table AM

- src/include/access/table.h
- src/include/access/tableam.h
- https://www.postgresql.org/docs/17/tableam.html

memory management

```
static int
table_count(Oid relid)
{
        int             cnt = 0;
        Relation rel;
        TableScanDesc scan;
        TupleTableSlot *slot;

        rel = table_open(relid, RowExclusiveLock);
        scan = table_beginscan(rel, GetActiveSnapshot(), 0, (ScanKey) NULL);

        slot = table_slot_create(rel, NULL);

        while (table_scan_getnextslot(scan, ForwardScanDirection, slot))
                cnt++;

        table_endscan(scan);
        table_close(rel, NoLock);

        return cnt;
}
```

SPI

Microsoft

- Server Programming Interface
- pretend you're a client, connect and run queries
- e.g. ALTER TABLE ... ADD FOREIGN KEY
  - doesn't do the heavy lifting "on your own"
  - generates a JOIN query and runs it through SPI
  - simple, automatically benefits from all optimizations

https://www.postgresql.org/docs/17/spi.html

The Server Programming Interface (SPI) gives writers of user-defined C functions the ability to run SQL commands inside their functions or procedures. SPI is a set of interface functions to simplify access to the parser, planner, and executor. SPI also does some memory management.

SPI

Microsoft

- table_count - returns numer of rows in a table
- get a cstring argument with name (PG_GETARG_CSTRING)
- build a SELECT COUNT(*) query
- connect to SPI
- run query, get result (1 row, 1 column)
- parse result from HeapTuple
- disconnect from SPI
- return count (PG_RETURN_INT64)

Microsoft

patch reviews

reviewing patches

- git apply / patch
- meld
-

?

- ?

Microsoft