



Pluggable Storage in PostgreSQL

Andres Freund

PostgreSQL Developer & Committer

Email: andres@anarazel.de

Email: andres.freund@enterprisedb.com

Twitter: [@AndresFreundTec](https://twitter.com/AndresFreundTec)

anarazel.de/talks/2018-10-25-pgconfeu-pluggable-storage/pluggable.pdf



Pluggable Storage

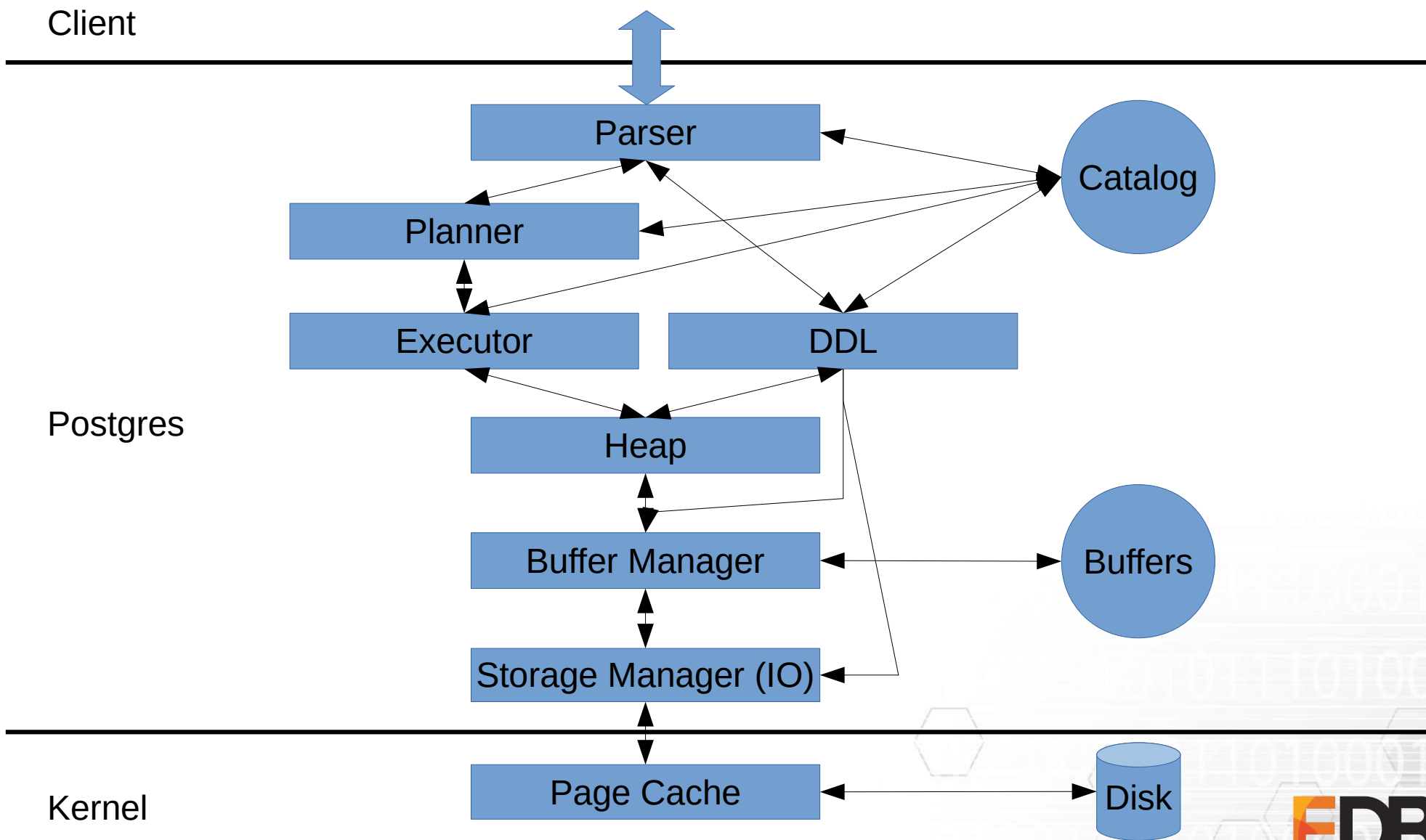
```
CREATE TABLE ...(...) USING heap;
```

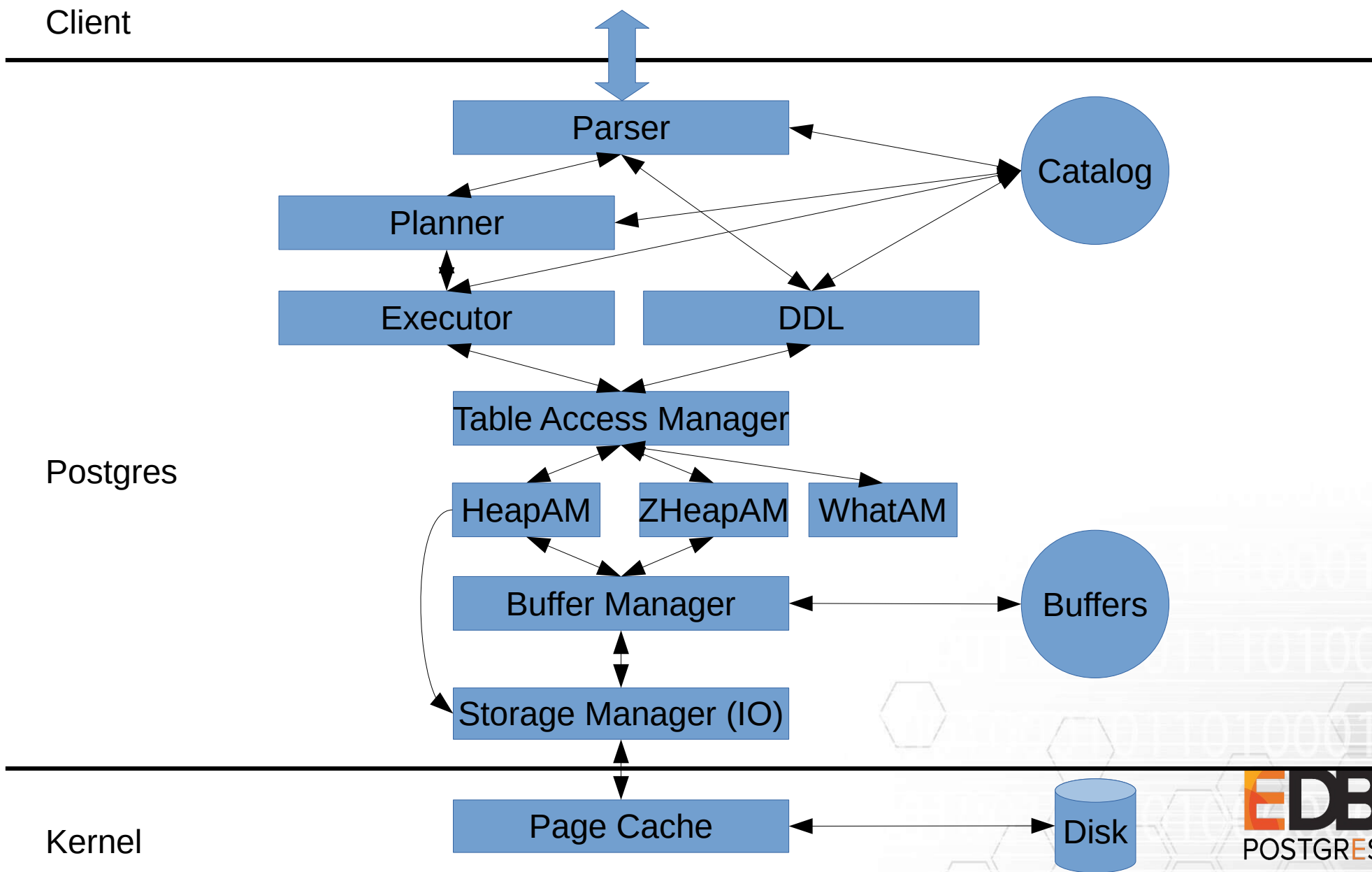
Work by:

- Haribabu Kommi
- Alvaro Herrera
- Alexander Korotkov
- Ashutosh Bapat
- Amit Khandekar

What do you mean: storage

- Contents of a TABLE / MATERIALIZED VIEW
- **NOT** contents of indexes
- Not purely a change of IO layer
- => Code name: 'tableam' - table access method





What do you mean: pluggable

```
CREATE EXTENSION magic_storage;  
CREATE TABLE something (...) USING magic_storage;  
SET default_table_access_method = 'magic_storage';  
CREATE TABLE else (...); -- still uses magic_storage
```

Why?

- ZHeap – UNDO based storage, address bloat and write amplification problems
- Columnar Storage
- Experiments

Why not?

- Proliferation of half-baked storage engines, rather than one good one
- Proliferation of closed & commercial storage engines
- Architectural impact

What?

- Multiple table AMs should be able to exist at compile time
- new table AMs can be added at runtime (i.e. `CREATE EXTENSION new_am;`)
- Indexes: Should work across different table AMs
- Planner: Should work largely unmodified against different AMs
- NOT: non-heap catalog tables
- NOT: Fully extensible WAL logging
- NOT: Executor/Planner magic to make every storage method superfast

Contrast to Foreign Data Wrapper API

- FDWs basically hook in at the planner level
- FDWs not intended to locally store data
- DDL not really supported
- Foreign Keys not supported (and it doesn't really make sense to support)
- Different goals, but some overlap exists

TupleTableSlots

- historic name, “Tuple Holder” more accurate
- Holds various forms of tuples, makes accesses to columns cheap(er)
- Made extensible:

```
typedef struct TupleTableSlotOps
{
    size_t      base_slot_size;
    void        (*init)(TupleTableSlot *slot);
    void        (*getsomeattrs)(TupleTableSlot *slot, int natts);
    HeapTuple   (*copy_heap_tuple)(TupleTableSlot *slot);
    ...
}
struct TupleTableSlot
{
    NodeTag     type;
    uint16      tts_flags;
    AttrNumber  tts_nvalid;
    const TupleTableSlotOps *const tts_cb;
    Datum       *tts_values;          /* current per-attribute values */
    bool        *tts_isnull;         /* current per-attribute isnull flags */
}
```

Table AM Handlers

```
postgres[28850][1]=# SELECT * FROM pg_am WHERE amtype = 't';
```

amname	amhandler	amtype
heap	heap_tableam_handler	t

(1 row)

```
postgres[28850][1]=# \df heap_tableam_handler
List of functions
```

Schema	Name	Result data type	Argument data types	Type
pg_catalog	heap_tableam_handler	table_am_handler	internal	func

(1 row)

Table AM Handlers

```
Datum  
heap_tableam_handler(PG_FUNCTION_ARGS)  
{  
    PG_RETURN_POINTER(&heapam_methods);  
}  
  
static const TableAmRoutine heapam_methods = {  
    .type = T_TableAmRoutine,  
  
    .slot_callbacks = heapam_slot_callbacks,  
  
    ...  
};
```

TupleTableSlots #2

```
extern PGDLLIMPORT const TupleTableSlotOps TTSOpsVirtual;  
extern PGDLLIMPORT const TupleTableSlotOps TTSOpsHeapTuple;  
extern PGDLLIMPORT const TupleTableSlotOps TTSOpsMinimalTuple;  
extern PGDLLIMPORT const TupleTableSlotOps TTSOpsBufferTuple;  
...  
extern TupleTableSlot *MakeTupleTableSlot(TupleDesc tupleDesc,  
                                           const TupleTableSlotOps *tts_cb);  
extern TupleTableSlot *ExecAllocTableSlot(List **tupleTable, TupleDesc desc,  
                                           const TupleTableSlotOps *tts_cb);  
extern TupleTableSlot *MakeSingleTupleTableSlot(TupleDesc tupdesc,  
                                                  const TupleTableSlotOps *tts_cb);  
...
```

Table AM API – DML & DDL

```
/*
 * API struct for a table AM. Note instances of this this must be
 * allocated in a server-lifetime manner, typically as a static const struct.
 */
typedef struct TableAmRoutine
{
...
    TupleInsert_function tuple_insert;
    TupleInsertSpeculative_function tuple_insert_speculative;
    TupleCompleteSpeculative_function tuple_complete_speculative;
    TupleUpdate_function tuple_update;
    TupleDelete_function tuple_delete;
    MultiInsert_function multi_insert;
    TupleLock_function tuple_lock;
...
    RelationVacuum_function relation_vacuum;
    RelationScanAnalyzeNextBlock_function scan_analyze_next_block;
    RelationScanAnalyzeNextTuple_function scan_analyze_next_tuple;
    RelationCopyForCluster_function relation_copy_for_cluster;
    RelationSync_function relation_sync;
...
    IndexBuildRangeScan_function index_build_range_scan;
    IndexValidateScan_function index_validate_scan;
...
} TableAmRoutine;
```

Table AM API – Scans

```
typedef struct TableAmRoutine
{
...
    ScanBegin_function scan_begin;
    ScanRescan_function scan_rescan;
    ScanEnd_function scan_end;
    ScanGetnextSlot_function scan_getnextslot;
...
    BeginIndexFetchTable_function begin_index_fetch;
    EndIndexFetchTable_function reset_index_fetch;
    EndIndexFetchTable_function end_index_fetch;

    TupleFetchFollow_function tuple_fetch_follow;
...
    BitmapPagescan_function scan_bitmap_pagescan;
    BitmapPagescanNext_function scan_bitmap_pagescan_next;
...
}           TableAmRoutine;
```


Infrastructure Changes

- Lots of rote changes to using slots & new scan APIs
 - DDL
 - Many executor nodes
- More complex slot changes:
 - Triggers
 - EvalPlanQual
 - Fix discrepancies between “declared” type of slot, and actually returned slot types
 - Analyze
 - COPY
- More complex executor changes:
 - Bitmap Scan
 - Sample Scan
- Other changes
 - error checks in extensions like pageinspect



Problems

- Indexes only have space for 6byte tuple-identifier
 - good enough for now, probably needs to be generalized
- Planner / Executor improvements needed for efficiency for some storage types (columnar)
 - can be addressed via planner hooks + custom executor nodes
- Unnecessary conversions to/from HeapTuple
- WAL logging not as extensible as desirable