Update on Goals

- Write a framework that determines whether extensions are likely incompatible with each other and fixes some incompatibilities (partially ✔️)
- Create an extension that manages extensions, get this to work for pg_stat_statement and pg_hint_plan; only need to #include “pgextmgr.h” without significant code change. (✔️)
- Postgres Extension Manager as an Extension

Extension Manager is also an extension!

Extension Manager
- Hooks Registry
- Extension 1
- Extension 2

Postgres Source Tree
- Parser Hooks
- Planner Hooks
- Optimizer Hooks
pgextmgrext in compatibility mode

- pg_stat_statements
- planner_hook
- Postgres
pgextmgrext in compatibility mode

```
pg_stat_statements
  ^-- planner_hook
    |    Postgres
    |    prev_planner
    |    pg_hint_plan
```

pg_stat_statements

prev_planner

pg_hint_plan

planner_hook

Postgres
pgextmgrext in compatibility mode

pgextmgrext

planner_hook

Postgres
By including the pgext header, we will do some work before / after the actual init process.
By including the pgext header, we will do some work before / after the actual init process.
By including the pgext header, we will do some work before / after the actual init process.
pgextmgrext in compatibility mode

We are able to reorder/disable/enable hooks and detect missing calls within the manager.

1. pgextmgrext calls pg_stat_statements
2. pg_stat_statements calls back to pgextmgrext
3. pgextmgrext calls pg_hint_plan
4. pg_hint_plan calls back to pgextmgrext
Easy to integrate – include a single header to use existing extensions with pgextmgr.

Easy to adapt – no need to build Postgres from source code, it is an extmgr as an extension!

Extensibility – adding new APIs and functionalities over existing Postgres hooks so that developers can build extensions easier and write less code.
Demo Video: here
pg_poop

- Toy extension that uses our extension as a manager to replace all VARCHAR/TEXT output with poop emojis
- 200 LoCs -> 70 LoCs by using pgextmgrext!
- Using pgextmgr’s output rewriter APIs (which is build upon existing ExecutorRun hook)
Postgres Extension Compatibility Analysis

- Spent time understanding the extension environment (hook case studies, extension case studies)
- Tested general compatibility (installation, basic tests) of 38 extensions with each other
  - Determined that only 57 of these pairs failed (8% incompatibility)
# Extension Compatibility Table

![Table Image]
- TLDR: 8% is a low result—we’re aiming for a higher incompatibility rate!
- Potentially interesting findings
  - Some extensions have checks to ensure that proper ordering (e.g. pg_queryid)
  - Some heavyweight extensions tend to clash with a lot of other extensions (Citus, plprofiler)
  - These are pretty good leads!
Testing + Code Quality

- General extension testing framework is well-tested (interface works for a lot of different extensions)
- `pgextmgrext` has a proof of concept implementation (`pg_poop`) but is otherwise not tested much (not production ready!)
- Code quality:
  - Ran cargo fmt + cargo clippy to ensure decent code style
  - We have validation of arguments in our software
  - Code is not documented very well 😞
Future Work

- Focus extension compatibility testing to extensions that share the same hooks, and extend testing to try to catch more conflicts
- Redesigning PostgreSQL extensions API using findings from pgextmgr/ext and related work (e.g. MySQL server, Redis modules, eBPF)
Thank you for listening!