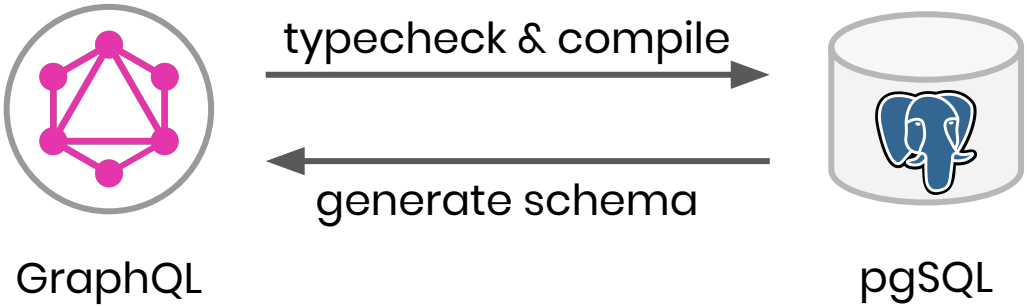
Haskell at Hasura

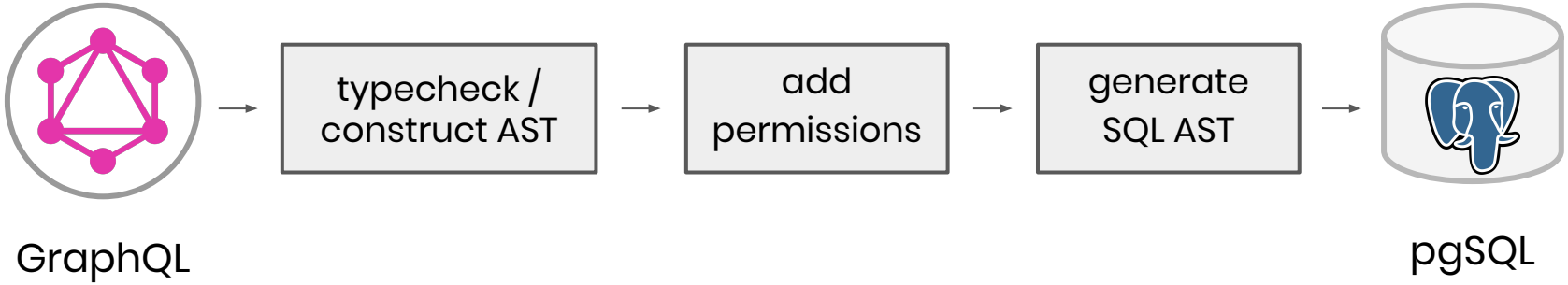
GraphQL Engine as a compiler



query compilation is just-in-time

schema generation is cached

The compilation pipeline



Postgres query planner does optimization for us!

Logic is mostly pure

```
convertSelect
  :: ( MonadReusability m, MonadError QErr m, MonadReader r m
     , Has FieldMap r , Has OrderByCtx r, Has SQLGenCtx r )
  ⇒ SelectOpCtx → Field → m QueryRootFieldResolved

toPGQuery :: QueryRootFieldResolved → SQL.Query
```

Odds and ends

- Postgres `LISTEN`/`NOTIFY` to synchronize schema changes across `graphql-engine` instances
- Also use `LISTEN`/`NOTIFY` with SQL triggers to dispatch event triggers
- Small bit of logic for stitching GraphQL schemas together from remote data sources and dispatching queries to multiple backends

Tools and libraries

- Recently switched from `stack` to `cabal-install`
- `spock` for HTTP
- Currently no effect system, just plain `mtl`/`transformers` code
- Small wrapper around `postgresql-libpq`
- Simple, homegrown `graphql-parser` package

Language extensions

```
default-extensions:
  ApplicativeDo BangPatterns BlockArguments ConstraintKinds
  DefaultSignatures DeriveDataTypeable DeriveFoldable DeriveFunctor
  DeriveGeneric DeriveLift DeriveTraversable DerivingVia EmptyCase
  FlexibleContexts FlexibleInstances FunctionalDependencies
  GeneralizedNewtypeDeriving InstanceSigs LambdaCase
  MultiParamTypeClasses MultiWayIf NamedFieldPuns NoImplicitPrelude
  OverloadedStrings QuantifiedConstraints QuasiQuotes RankNTypes
  ScopedTypeVariables StandaloneDeriving TemplateHaskell TupleSections
  TypeApplications TypeFamilies TypeOperators
```

34 language extensions enabled by default!

Data modeling with Haskell

```
data GBoolExp a
  = BoolAnd ![GBoolExp a]
  | BoolOr  ![GBoolExp a]
  | BoolNot !(GBoolExp a)
  | BoolExists !(GExists a)
  | BoolFld !a
```

"Parse, don't validate"

https://lexi-lambda.github.io/blog/2019/11/05/parse-don-t-validate/

Abstracting with Haskell

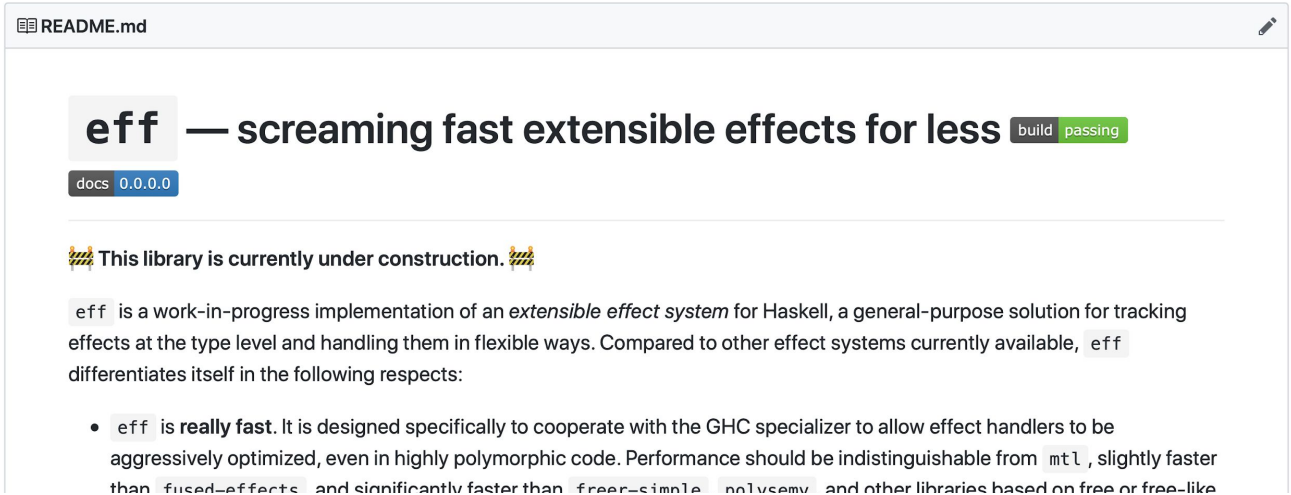
```
buildSchemaCacheRule
  :: ( ArrowChoice arr, Inc.ArrowDistribute arr, Inc.ArrowCache m arr
     , MonadIO m, MonadTx m, MonadReader BuildReason m, HasHttpManager m, HasSQLGenCtx m )
  ⇒ (CatalogMetadata, InvalidationMap) `arr` SchemaCache
buildSchemaCacheRule = proc inputs → do
  (outputs, collectedInfo) ← runWriterA buildAndCollectInfo -≺ inputs
  let (inconsistentObjects, unresolvedDependencies) = partitionCollectedInfo collectedInfo
  (resolvedOutputs, extraInconsistentObjects, resolvedDependencies) ←
    resolveDependencies -≺ (outputs, unresolvedDependencies)
  ...
```

monads, arrows, `lens`, generic programming

Six months experience report

- Yes, it's true: refactoring Haskell code really is that easy
- Lack of real structurally typed records is sometimes very painful
- Dealing with monad transformers can be a real chore

Building a better effect system



README.md

eff — screaming fast extensible effects for less build passing

docs 0.0.0.0

🚧 **This library is currently under construction.** 🚧

`eff` is a work-in-progress implementation of an *extensible effect system* for Haskell, a general-purpose solution for tracking effects at the type level and handling them in flexible ways. Compared to other effect systems currently available, `eff` differentiates itself in the following respects:

- `eff` is **really fast**. It is designed specifically to cooperate with the GHC specializer to allow effect handlers to be aggressively optimized, even in highly polymorphic code. Performance should be indistinguishable from `mtl`, slightly faster than `fused-effects` and significantly faster than `freer-simple`, `polysemy` and other libraries based on free or free-like

`eff`: an efficient implementation of delimited control

https://github.com/hasura/eff

Other random notes

- Extensive end-to-end test suite is invaluable
- Haskell's support for concurrency and parallelism is understatedly fantastic
- Strict datatypes are a good default
- Ongoing effort to make contributing easier: more documentation, adopting GHC's "notes"

Questions?

graphql-engine

github: https://github.com/hasura/graphql-engine

discord server: https://discordapp.com/invite/hasura

me

blog: https://lexi-lambda.github.io

twitter: @lexi_lambda