main Go to file Code

tobilg Add examples to README a8fa037 · yesterday

.cargo	Initial implementation	5 days ago
.github/workflows	Update CI	4 days ago
crates	Fixes	yesterday
docs	Cleanup	yesterday
examples	Add examples	yesterday
licenses	Prepare v0.1.1	3 days ago
packages	Prepare v0.1.2 release	yesterday
tools	Add playground etc.	4 days ago
.gitignore	Many updates, prep...	3 days ago
Cargo.lock	Prepare v0.1.2 release	yesterday
Cargo.toml	Add examples	yesterday
LICENSE	Initial implementation	5 days ago
Makefile	Fixes	yesterday
README.md	Add examples to RE...	yesterday
pnpm-lock.yaml	Add playground etc.	4 days ago
pnpm-workspace.y...	Add playground etc.	4 days ago

README MIT license

Polyglot

Rust/Wasm-powered SQL transpiler for 32 dialects, inspired by sqlglot.

Polyglot parses, generates, transpiles, and formats SQL across 32 database dialects. It ships as a Rust crate (polyglot-sql) and a TypeScript/WASM SDK (@polyglot-sql/sdk on npm).

There's also a playground where you can try it out in the browser, as well as the Rust API Docs and TypeScript API Docs.

Features

- **Transpile** SQL between any pair of 32 dialects
- **Parse** SQL into a fully-typed AST
- **Generate** SQL back from AST nodes
- **Format** / pretty-print SQL
- **Fluent builder API** for constructing queries programmatically
- **Validation** with syntax, semantic, and schema-aware checks
- **AST visitor** utilities for walking, transforming, and analyzing queries

Supported Dialects (32)

Athena	BigQuery	ClickHouse	Cockroach
Doris	Dremio	Drill	Druid
Dune	Exasol	Fabric	Hive
MySQL	Oracle	PostgreSQL	Presto
RisingWave	SingleStore	Snowflake	Solr
SQLite	StarRocks	Tableau	Teradata
Trino	TSQL		

Quick Start

Rust

```rust
use polyglot_sql::{transpile, DialectType};

// Transpile MySQL to PostgreSQL
let result = transpile(
    "SELECT IFNULL(a, b) FROM t",
    DialectType::MySQL,
    DialectType::Postgres,
).unwrap();
assert_eq!(result[0], "SELECT COALESCE(a, b) FR


use polyglot_sql::builder::*;

// Fluent query builder
let query = select(["id", "name"])
    .from("users")
    .where_col("age").gt(lit(18))
    .order_by(["name"])
    .limit(10)
    .build();
```

See the full Rust crate README for more examples.

TypeScript

```
npm install @polyglot-sql/sdk
```

```
import { transpile, Dialect } from "@polyglot-


// Transpile MySQL to PostgreSQL
const result = transpile(
  "SELECT IFNULL(a, b) FROM t",
  Dialect.MySQL,
  Dialect.PostgreSQL,
);
console.log(result.sql[0]); // SELECT COALESCE


import { select, col, lit } from "@polyglot-sq


// Fluent query builder
const sql = select('id', 'name')
  .from('users')
  .where(col('age').gt(lit(18)))
  .orderBy(col('name').asc())
  .limit(10)
  .toSql('postgresql');
```

See the full TypeScript SDK README for more examples.

Project Structure

```
polyglot/
├── crates/
│   ├── polyglot-sql/          # Core Rust lil
│   └── polyglot-sql-wasm/     # WASM binding
├── packages/
│   ├── sdk/                   # TypeScript SI
│   └── playground/            # Playground fi
└── tools/
    ├── sqlglot-compare/       # Test extract
    └── bench-compare/         # Performance
```

Examples

Standalone example projects are available in the `examples/` directory. Each one pulls the latest published package and can be run independently.

Rust

```
cargo run --manifest-path examples/rust/Cargo.
```

TypeScript

```
cd examples/typescript
pnpm install --ignore-workspace && pnpm start
```

Building from Source

```
# Build Rust core
cargo build -p polyglot-sql

# Build WASM + TypeScript SDK
make build-all

# Or step by step:
cd crates/polyglot-sql-wasm && wasm-pack build
cd packages/sdk && npm run build
```

Testing

Polyglot maintains compatibility with sqlglot through **8,455 fixture tests** extracted from the Python reference implementation. All test suites pass at **100%**.

Category	Count	Pass Rate
Generic identity	955	100%
Dialect identity	3,461	100%
Transpilation	4,015	100%
Pretty-print	24	100%
Lib unit tests	704	100%
Total	**9,159**	**100%**

```
# Setup fixtures (required once)
make setup-fixtures

# Run all tests
make test-rust-all            # All 8,455 fixtur
make test-rust-lib            # 704 lib unit tes
make test-rust-verify         # Full verificatio

# Individual test suites
make test-rust-identity       # 955 generic iden
make test-rust-dialect        # 3,461 dialect id
make test-rust-transpile      # 4,015 transpilat
make test-rust-pretty         # 24 pretty-print

# Additional tests
make test-rust-roundtrip      # Organized roundtr
make test-rust-matrix         # Dialect matrix te
make test-rust-compat         # SQLGlot compatib
make test-rust-errors         # Error handling te
make test-rust-functions      # Function normali

# TypeScript SDK tests
cd packages/sdk && npm test

# Full comparison against Python SQLGlot
make test-compare
```

Benchmarks

```
make bench-compare            # Compare polyglot
make bench-rust               # Rust benchmarks
make bench-python             # Python sqlglot be
cargo bench -p polyglot-sql   # Criterion benche
```

Fuzzing

```
cargo +nightly fuzz run fuzz_parser
cargo +nightly fuzz run fuzz_roundtrip
cargo +nightly fuzz run fuzz_transpile
```

Makefile Targets

Target	Description
`make help`	Show all available commands
`make build-all`	Build WASM + Rust (release)
`make build-wasm`	Build WASM package + TypeScript SDK
`make test-rust`	Run all sqlglot compatibility tests
`make test-rust-all`	Run all 6,284 fixture tests
`make test-rust-lib`	Run 693 lib unit tests
`make test-rust-verify`	Full verification suite
`make test-compare`	Compare against Python sqlglot
`make bench-compare`	Performance comparison

About

Rust/Wasm-powered SQL transpiler for more than 30 SQL dialects

polyglot.gh.tobilg.com

rust sql wasm transpiler sqlglot

Readme
MIT license
Activity
93 stars
1 watching
5 forks
Report repository

Releases

3 tags

Languages

- Rust 86.8%
- TypeScript 12.5%
- Python 0.3%
- JavaScript 0.2%
- Makefile 0.1%
- CSS 0.1%

`make extract-fixtures`	Regenerate JSON fixtures from Python
`make setup-fixtures`	Create fixture symlink for Rust tests
`make generate-bindings`	Generate TypeScript type bindings
`make clean`	Remove all build artifacts

Licenses

MIT solalgot MIT