Rust-Postgres

An idiomatic, native Postgres driver

Steven Fackler - sfackler

July 31, 2014

Outline

What's PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness.

Connecting

Connect with a standard psql-style URI:

```
use postgres::{PostgresConnection, NoSsl};
let url = "postgresql://sfackler@localhost:15410/mydb";
let conn = try!(
        PostgresConnection::connect(url, &NoSsl));
```

Connecting

Unix sockets are supported as well:

```
use postgres::{PostgresConnection, NoSsl};
let url = "postgresql://sfackler@%2Frun%2Fpostgres/mydb";
let conn = try!(
        PostgresConnection::connect(url, &NoSsl));
```

Connecting

Alternatively, pass a 'PostgresConnectParams' struct:

```
use postgres::{PostgresConnection, NoSsl,
               PostgresConnectParams, TargetUnix};
let params = PostgresConnectParams {
    target: TargetUnix(Path::new("/run/postgres")),
    port: Some(1234),
    .....
};
let conn = try!(
        PostgresConnection::connect(params, &NoSsl));
```

Statement Preparation

Queries must first be *prepared* before they can be executed. They may be parameterized. Parameters are denoted by `$n`, and are 1-indexed.

```
let query = "SELECT name, height
             FROM people
             WHERE age < $1";
let stmt = try!(conn.prepare(query));
```

Execution

The `execute` method takes a slice of values to bind to the query parameters and returns the number of rows modified.

```
let query = "UPDATE users SET name = $1
             WHERE age = $2";
let stmt = try!(conn.prepare(query));
let rows_updated = try!(stmt.execute(
        [&"Steven", &Some(24i32)]));
```

Querying

query is similar to execute but it returns an iterator over the rows returned by a query. Columns may be accessed by index or name.

```
let query = "SELECT name, age FROM users
             WHERE age < $1";
let stmt = try!(conn.prepare(query));
for row in try!(stmt.query([&18i32])) {
    let name: String = row.get(0u);
    let age: Option<i32> = row.get("age");
    println!("{} is {}", name, age);
}
```

Parameterization

Use it. Seriously.

Parameterization

```
fn update_grade(conn: &PostgresConnection,
                name: &str, grade: f32)
                -> PostgresResult<()> {
    let query = format!("UPDATE students SET grade = {}
                         WHERE name = '{}'",
                        grade, name);
    try!(stmt.batch_execute(query.as_slice()));
    Ok(())
}
```

Parameterization

```
let name = "Robert'); DROP TABLE Students;--";
let grade = 100f32;
update_grade(&conn, name, grade);
```

Parameterization

```
fn update_grade(conn: &PostgresConnection,
                name: &str, grade: f32)
                -> PostgresResult<Student> {
    let query = "UPDATE students SET grade = $1
                 WHERE name = $1";
    let stmt = try!(conn.prepare(query));
    try!(stmt.update([&grade, &name]));
    Ok(())
}
```

Transactions

Transactions are managed by the `PostgresTransaction` object:

```
let trans = try!(conn.transaction());
let stmt = try!(trans.prepare(...));
....

if the_coast_is_clear {
    trans.set_commit();
}

try!(trans.finish()); // COMMIT / ROLLBACK here
```

RAII and Lifetimes are Awesome

Strict Typing

1. Rust-Postgres differs from many drivers in

Extensibility

Postgres allows for extensions which define new types and operations.
Rust-Postgres

Extensibility

All conversions are done through two traits

```
pub trait ToSql {
    fn to_sql(&self, ty: &PostgresType)
              -> PostgresResult<(Format,
                                 Option<Vec<u8>>)>;
}

pub trait FromSql {
    fn from_sql(y: &PostgresType,
                raw: &Option<Vec<u8>>)
                -> PostgresResult<Self>;
}
```

What's Missing

Rust-Postgres defines the low basics, but much of the infrastructure on top is still missing.

- Connection Pool - There is a pool in Rust-Postgres but it's no where near sufficient.
- ORM - Syntax extensions could allow for a very nice ORM system.

That's It!

Questions?