A Question of Ownership

Roland Bock

http://ppro.com
rbock at eudoxos dot de

https://github.com/rbock/sqlpp11
https://github.com/rbock/sqlpp17

Munich, 2018-01-25

Two query types

Prepared statement

```
auto ps = db.prepare(select(all_of(tabPerson))
                     .from(tabPerson)
                     .where(tabPerson.id == sqlpp::parameter<int>(alias::id)));
```

Two query types

Prepared statement

```
auto ps = db.prepare(select(all_of(tabPerson))
                     .from(tabPerson)
                     .where(tabPerson.id == sqlpp::parameter<int>(alias::id)));

ps.id = 42;
```

Two query types

Prepared statement

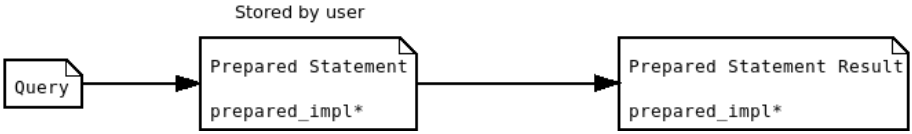
```
auto ps = db.prepare(select(all_of(tabPerson))
                     .from(tabPerson)
                     .where(tabPerson.id == sqlpp::parameter<int>(alias::id)));

ps.id = 42;

for (const auto& row : execute(ps))
{
  std::cout << row.id << " "
            << row.name << " "
            << row.yearOfBirth << "\n";
}
```

Two query types

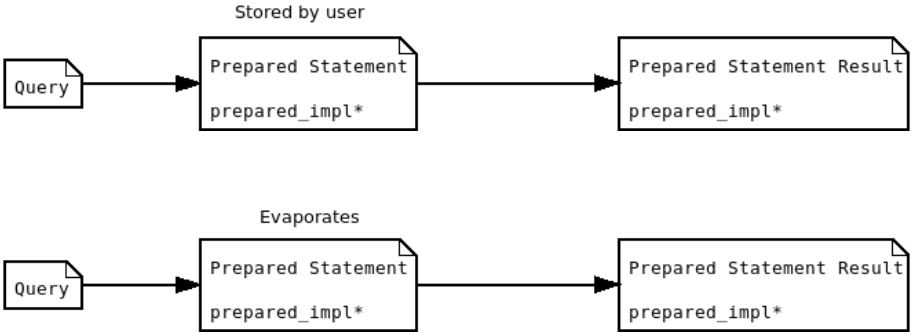
Direct execution

```
for (const auto& row : db(select(all_of(tabPerson))
                          .from(tabPerson)
                          .where(tabPerson.id == 42)))
{
  std::cout << row.id << " "
            << row.name << " "
            << row.yearOfBirth << "\n";
}
```

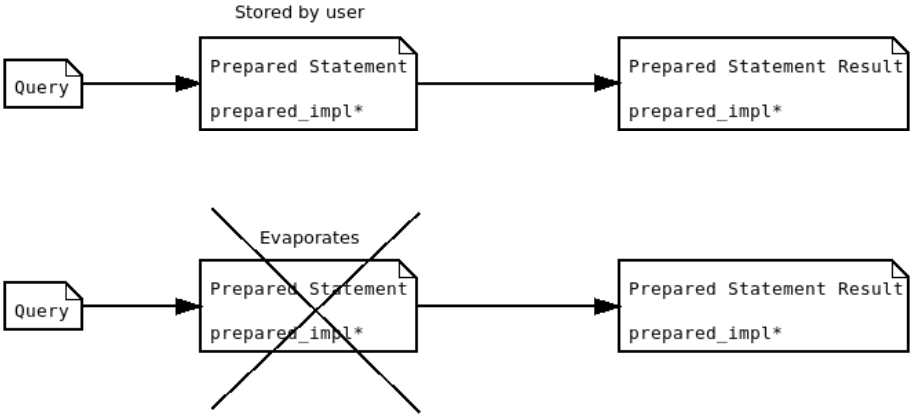
Implementation for sqlite



Implementation for sqlite



Implementation for sqlite



A question of ownership

Seems like in one case result is merely an observer, while in the other case it has to be the owner of that pointer.

How do we model that?

shared_ptr

```
class result
{
  std::shared_ptr<prepared_impl> _impl;

  // ...
};
```

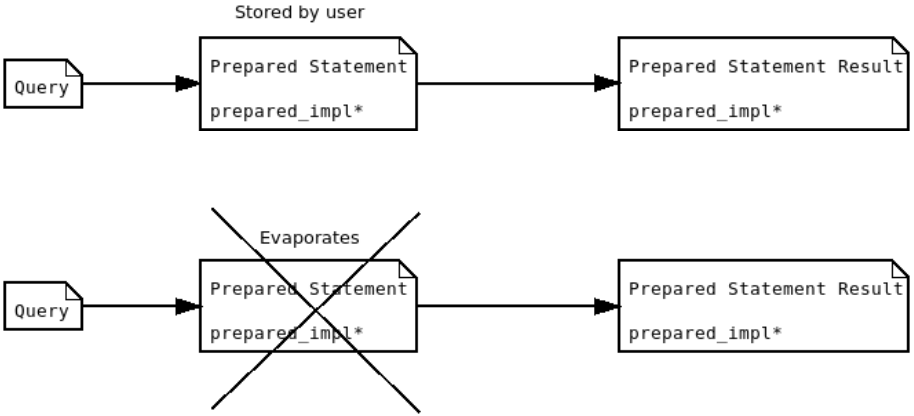
unique_ptr

```
class result
{
  std::unique_ptr<prepared_impl, maybe_deleter> _impl;

  // ...
};
```

variant

```
class result
{
  std::variant<prepared_impl*,
               std::unique_ptr<prepared_impl>> _impl;

  // ...
};
```

Two classes?


Stored by user
Query
Prepared Statement
prepared_impl*
Prepared Statement Result
prepared_impl*
Evaporates
Query
Prepared Statement
prepared_impl*
Prepared Statement Result
prepared_impl*

Two classes

```
class result_view
{
  prepared_impl*_impl;
  // ...
};
```

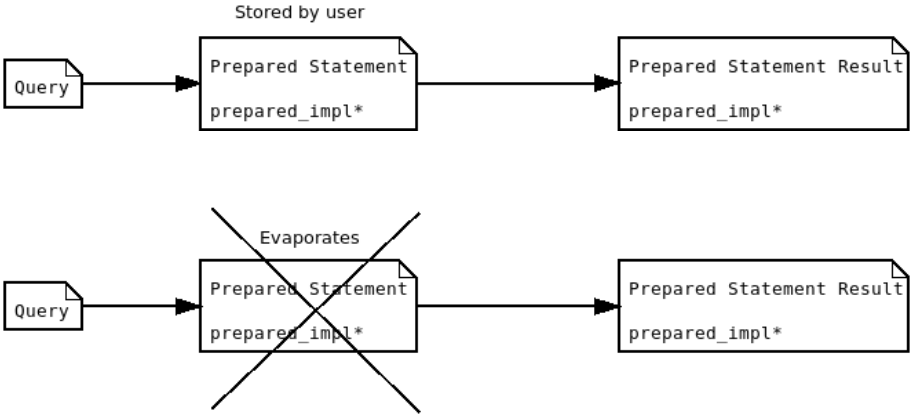
Two classes

```
class result_view
{
  prepared_impl*_impl;
  // ...
};

class result : public result_view
{
  std::unique_ptr<prepared_impl>_impl;
  // ...
};
```

A template class

```
template<typename OwnershipPolicy>
class result
{
  typename OwnershipPolicy::storage _impl;
  // ...
};
```

Borrowed ownership



Borrowed ownership

```
class result
{
  std::unique_ptr<prepared_impl> _impl;
  statement _owner;
public:
  result(std::unique_ptr<prepared_impl>, statement* owner);

  ~result()
  {
    if (_owner)
      owner.return_ownership(_impl);
  }
  // ...
};
```

What would you do?

Let me know.