Making optional optional

Roland Bock

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

https://github.com/rbock/sqlpp11

CppCon, 2017-09-26

```
template <typename K, typename V>
auto version_book(const K& key, const std::map<K, V>& data)
{
  auto opt = std::optional<V>{};

  if (data.count(key))
  {
    opt = data.at(key);
  }

  test(opt);
}
```

std::make_optional?

```
template <typename Factory>
decltype(auto) make_really_optional(bool condition,
                                    const Factory& factory)
{
  auto opt = std::optional<std::decay_t<decltype(factory())>>{};

  if (condition)
  {
    opt = factory();
  }

  return opt;
}
```

```
template <typename K, typename V>
auto version_lambda(const K& key, const std::map<K, V>& data)
{
  test(
     make_really_optional(
        data.count(key),
        [&](){ return data.at(key); }));
}
```

```
struct naught
{
  template <typename T>
  operator std::optional<T>() const
  {
    return std::optional<T>{};
  }
};
```

```
template <typename K, typename V>
auto version_implicit_conversion(const K& key, const std::map<K, V>& data)
{
  test(data.count(key) ? std::optional{data.at(key)} : naught{});
}
```

```
template <typename K, typename V>
auto version_standard(const K& key, const std::map<K, V>& data)
{
  test(data.count(key) ? std::optional{data.at(key)} : std::nullopt);
}
```