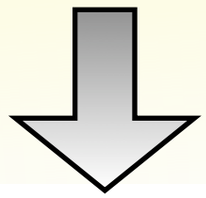
An Introduction to DBIx::Class

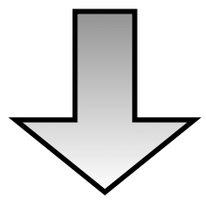
Tom Hukins

Maps Database Structures to Object Oriented Structures

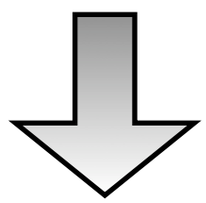
Schema



Result Source



Result Set



Row

DBIx::Class::Schema

```
CREATE DATABASE example;
```

Your Database Schema:
All tables, their
relationships and contents

DBIx::Class::ResultSource

```
CREATE TABLE foo (
    id PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    favourite_colour INT REFERENCES colour.id,
    pointless BIT,
    skillz CHAR(3) DEFAULT 'lol'
);
```

Your Database Tables and their relationships with each other.

DBIx::Class::ResultSet

id	name	colour
1	Sky	Blue
2	Grass	Green
3	Clouds	Monotonous

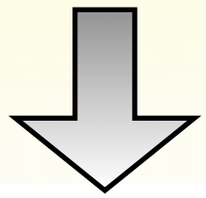
Zero or more records
within a table

DBIx::Class::Row

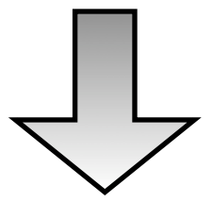
id	name	colour
1	Sky	Blue

All the fields within a row

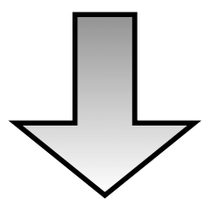
Schema



Result Source



Result Set



Row

Schema

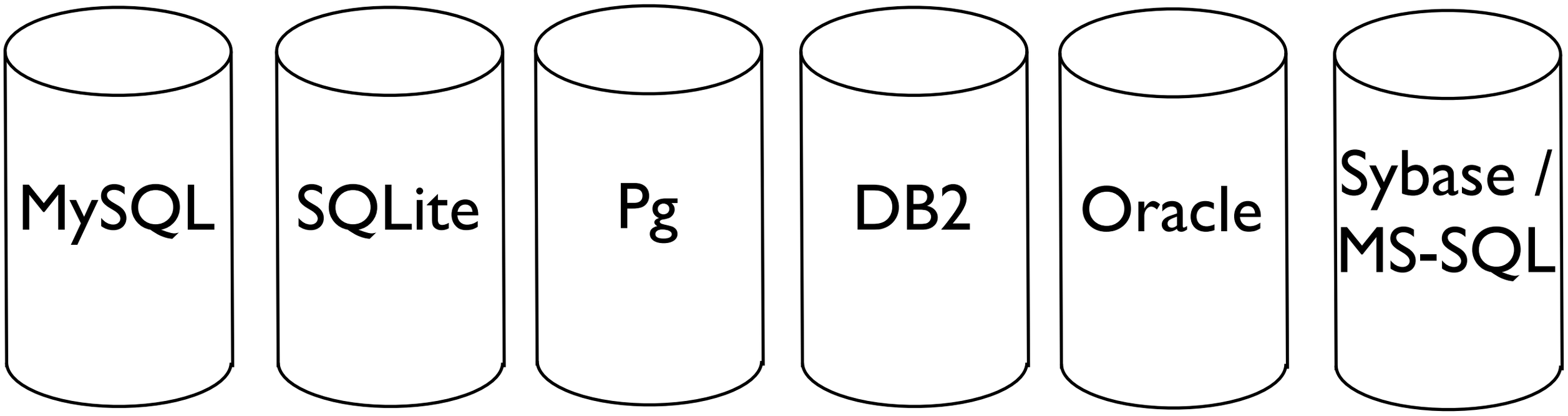
Storage

Result Source

Result Set

Row

DBIx::Class::Storage



Connects DBIx::Class and
DBD:: Database Drivers

Defining a Schema

```
1 package Drink;
2
3 use base qw/DBIx::Class::Schema/;
4
5 __PACKAGE__->load_classes();
6
7 1;
```

Defining Result Sources

```
1 package Drink::Cocktails;
2
3 use base qw/DBIx::Class/;
4
5 __PACKAGE__->load_components(qw/Core/);
6 __PACKAGE__->table('cocktails');
7 __PACKAGE__->add_columns(qw/id name abv/);
8 __PACKAGE__->set_primary_key('id');
9 __PACKAGE__->has_many(
10     ingredients => 'Drink::Ingredients', 'cocktail'
11 );
12
13 1;
```

Defining Result Sources

```
1 package Drink::Ingredients;
2
3 use base qw/DBIx::Class/;
4
5 __PACKAGE__->load_components(qw/Core/);
6 __PACKAGE__->table('ingredients');
7 __PACKAGE__->add_columns(qw/id cocktail name amount/);
8 __PACKAGE__->set_primary_key('id');
9 __PACKAGE__->belongs_to(cocktail => 'Drink::Cocktails');
10
11 1;
```

Defining a SQLite Schema

```
1 CREATE TABLE cocktails (
2   id              INTEGER PRIMARY KEY,
3   name            TEXT,
4   abv             NUMERIC
5 );
6
7 CREATE TABLE ingredients (
8   id              INTEGER PRIMARY KEY,
9   cocktail        INTEGER,
10  name            TEXT,
11  amount          TEXT
12 );
```

Creating Records

```
1 my $drink = Drink->connect(
2     'dbi:SQLite:dbname=cocktails.db', '', '');
3
4 my $cocktails = $drink->resultset('Cocktails');
5 my $gin_and_french = $cocktails->create({
6     name => 'Gin and French',
7     abv => 12.7,
8 });
9
10 my $ingredients = $drink->resultset('Ingredients');
11 $ingredients->create({
12     name => 'Gin',
13     amount => '1.5 Shots',
14     cocktail => $gin_and_french,
15 });
```

Retrieving a Record

```
1 use Drink;
2
3 my $drink = Drink->connect(
4     'dbi:SQLite:dbname=cocktails.db', '', '');
5
6 my $cocktails = $drink->resultset('Cocktails');
7 my $cocktail_by_id = $cocktails->find(2);
8 print $cocktail_by_id->name, " contains: \n";
9 foreach ($cocktail_by_id->ingredients) {
10     print " - ", $_->amount, " of ", $_->name, "\n";
11 }
```

Retrieving a Record

```
Martini (Dirty) contains:
 - 2.5 Shots of Gin
 - 1 Large Dash of Noilly Prat Dry
 - 0.5 Shot of Brine from Olives
```

Tracing Your Queries

- `$ENV{DBIC_TRACE} = 1`
- `$storage->debug(1);`

Tracing Your Queries

```
SELECT me.id, me.name, me.abv
    FROM cocktails me WHERE ( ( me.id = ? ) ): '2'

SELECT me.id, me.cocktail, me.name, me.amount
    FROM ingredients me WHERE ( me.cocktail = ? ): '2'

Martini (Dirty) contains:
 - 2.5 Shots of Gin
 - 1 Large Dash of Noilly Prat Dry
 - 0.5 Shot of Brine from Olives
```

Improving the SQL

```
1 my $ingredients = $drink->resultset('Ingredients');
2 my $our_ingredients = $ingredients->search(
3     { cocktail => 2 },
4     {
5         join => 'cocktail',
6         prefetch => 'cocktail',
7     },
8 );
```

Improving the SQL

```
 9 my $first_row = 1;
10 while (my $ingredient = $our_ingredients->next) {
11     if ($first_row) {
12         my $name = $ingredient->cocktail->name;
13         print "$name contains:\n";
14     }
15     print " - ", $ingredient->amount, " of ",
16         $ingredient->name, "\n";
17     $first_row = 0;
18 }
```

Tracing the Improved SQL

```
SELECT me.id, me.cocktail, me.name, me.amount, cocktail.name
FROM ingredients me
JOIN cocktails cocktail ON ( cocktail.id = me.cocktail )
WHERE ( cocktail = ? ): '2'

Martini (Dirty) contains:
 - 2.5 Shots of Gin
 - 1 Large Dash of Noilly Prat Dry
 - 0.5 Shot of Brine from Olives
```

ResultSet::Column

```
1 use Drink;
2
3 my $drink = Drink->connect(
4     'dbi:SQLite:dbname=cocktails.db', '', '');
5
6 my $cocktails = $drink->resultset('Cocktails');
7 print $cocktails->get_column('abv')->max, "\n";
```

ResultSet::Column

```
SELECT MAX( abv ) FROM cocktails me:
32.6
```

Advanced Searches

```
use Drink;

my $drink = Drink->connect(
    'dbi:SQLite:dbname=cocktails.db', '', '');

my $cocktails = $drink->resultset('Cocktails');
my $max_abv = $cocktails->search(
    { abv => \'= (SELECT MAX(abv) FROM cocktails)' }
)->single;

print $max_abv->name," is ", $max_abv->abv, "%.\n";
```

Advanced Searches

```
SELECT me.id, me.name, me.abv FROM cocktails me
WHERE ( abv = (SELECT MAX(abv) FROM cocktails) ):
```

```
Martini (Dirty) is 32.6%.
```

DBIx::Class::Schema::Loader

```
1 package My::Schema;
2 use base qw/DBIx::Class::Schema::Loader/;
3
4 __PACKAGE__->loader_options(
5     relationships           => 1,
6 );
7
8 my $drinks = $connect(
9     'dbi:SQLite:dbname=cocktails.db', '', '');
```

DBIx::Class::InflateColumns

```
1 __PACKAGE__->load_components(qw/
2     InflateColumn::DateTime
3     Core
4 /);
5
6 __PACKAGE__->add_columns(
7     starts_when => { data_type => 'datetime' }
8 );
```

Other Useful Things

- Custom Result Sources
- Paged Results (uses Data::Page)
- DBIx::Class::Schema's deploy()
- populate() in DBIx::Class::Schema and ::ResultClass::HashRefInflator