

DBIx::Class

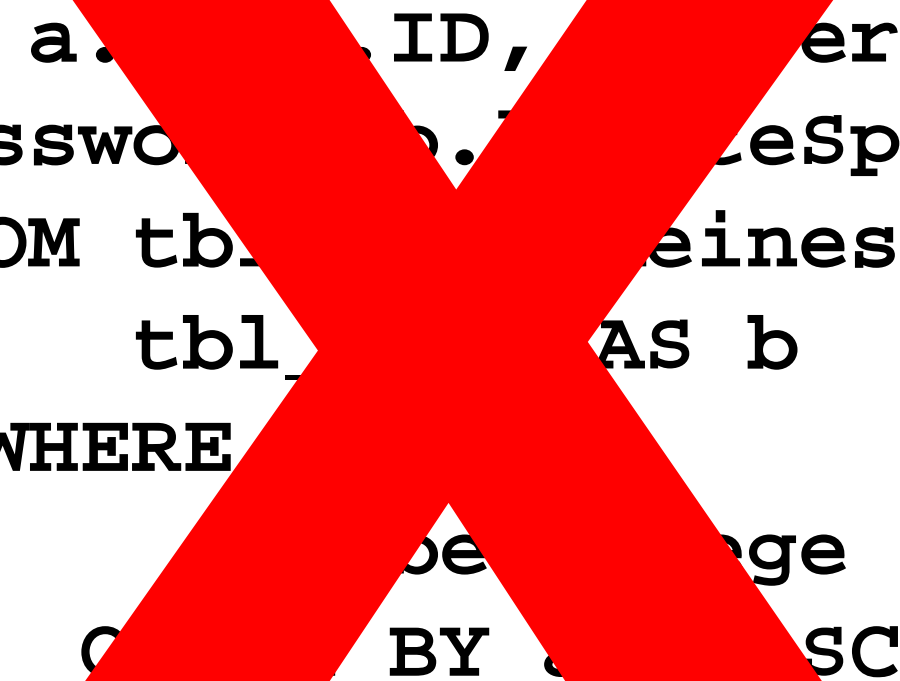
## Was ist DBIx::Class?

```
SELECT a.*, b.ID, b.Username,  
       b.Password, b.letzteSpalte  
FROM tbl_allgemeines AS a,  
     tbl_User AS b  
WHERE a.ID           = b.ID AND  
       a.beitraege = 199  
ORDER BY a.IDASC
```



## Was ist DBIx::Class?

```
SELECT a.ID, a.Username,  
       b.Password, b.ZeSpalte  
FROM tbl_eines AS a,  
     tbl_zwei AS b  
WHERE a.ID = b.ID AND  
       a.ZeSpalte = 199  
ORDER BY a.ID ASC
```



## Was ist DBIx::Class?

```
my $results = $obj->resultset( 'Allgemeines' )  
    ->search(  
        #...  
    );
```

Was ist DBIx::Class





**Doch von vorne...**

## Was ist DBIx::Class

<http://de.wikipedia.com/wiki/ORM>:

**„Objektrelationale Abbildung** (englisch *object-relational mapping*, ORM) ist eine Technik der Softwareentwicklung, mit der ein in einer objektorientierten Programmiersprache geschriebenes Anwendungsprogramm seine Objekte in einer relationalen Datenbank ablegen kann. Dem Programm erscheint die Datenbank dann als objektorientierte Datenbank, was die Programmierung erleichtert. Implementiert wird diese Technik normalerweise über Klassenbibliotheken, wie beispielsweise [...]“  
DBIx::Class für die Programmiersprache Perl.

## Eine Tabelle

| BundesligaTabelle ▼  |                 |
|--|-----------------|
|   | Platz: INTEGER  |
|   | Verein: VARCHAR |
|   | Punkte: INTEGER |
|  | Tore: VARCHAR   |

# Eine Tabelle

```
package My::DB::Bundesliga;
```

```
use base qw(DBIx::Class);
```

```
__PACKAGE__->load_components(  
    qw/PK::Auto Core/  
);
```

```
__PACKAGE__->table(  
    'BundesligaTabelle',  
);
```

```
__PACKAGE__->add_columns(qw/
```

```
    Platz
```

```
    Punkte
```

```
    Tore
```

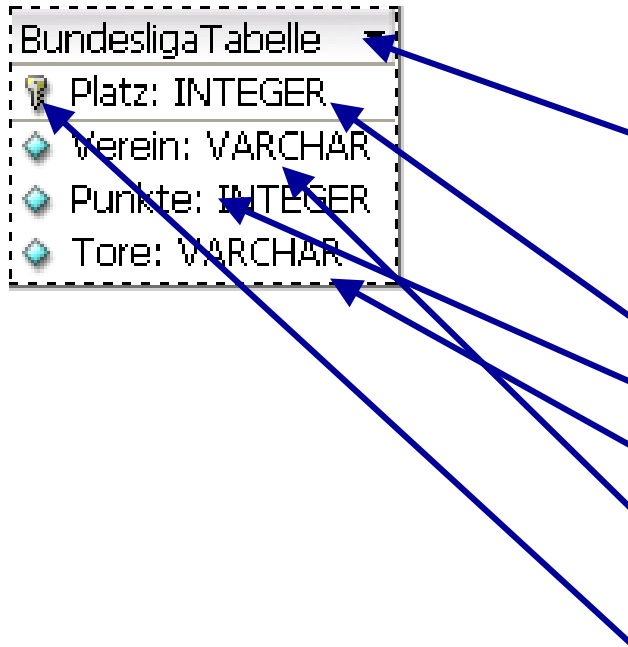
```
    VereinID
```

```
;/);
```

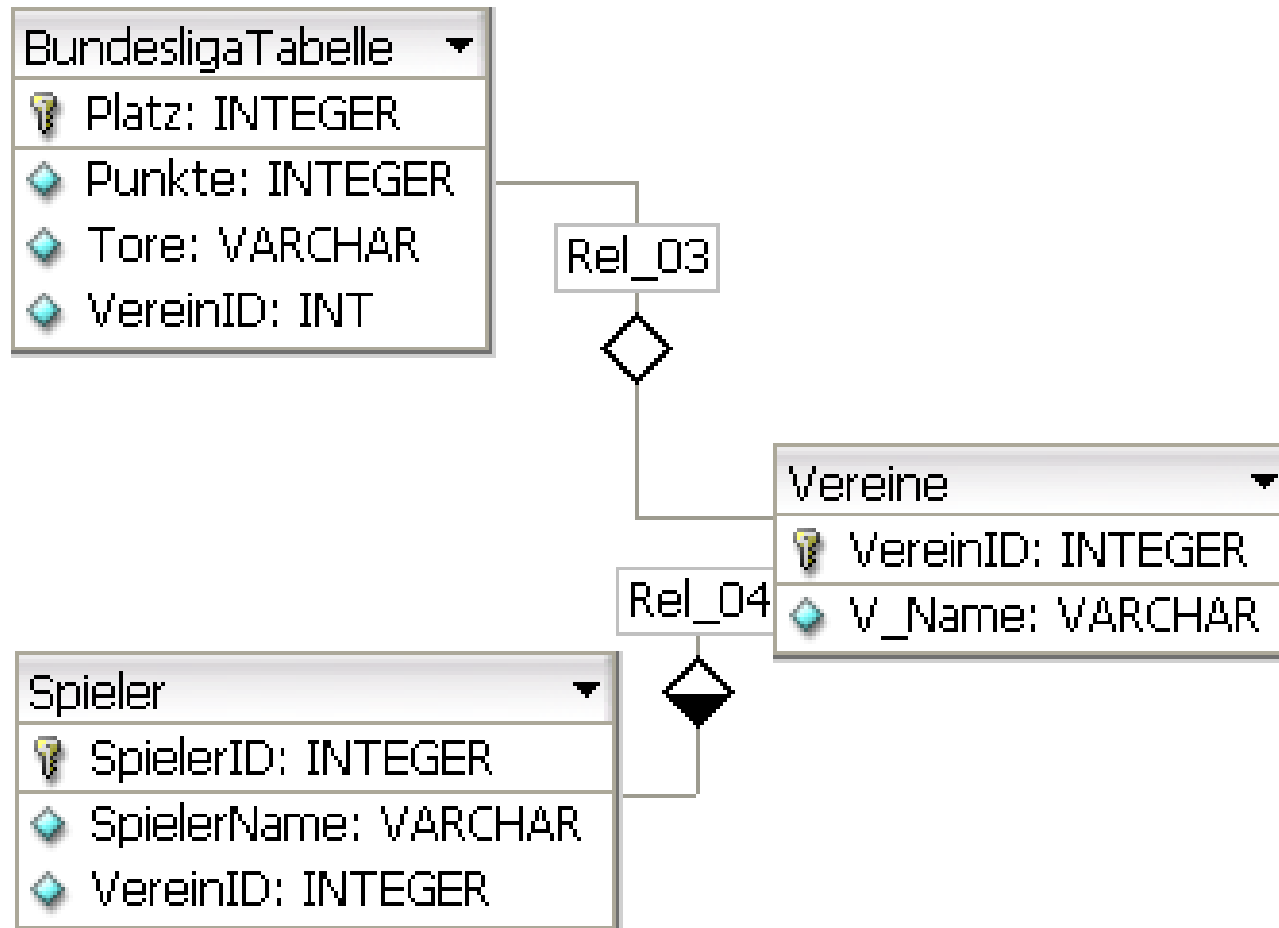
```
__PACKAGE__->set_primary_key('Platz');
```

```
1;
```

| BundesligaTabelle |                 |
|-------------------|-----------------|
| 🏆                 | Platz: INTEGER  |
| 👤                 | Verein: VARCHAR |
| 📊                 | Punkte: INTEGER |
| ⚽                 | Tore: VARCHAR   |



# Beziehungen



# Beziehungen

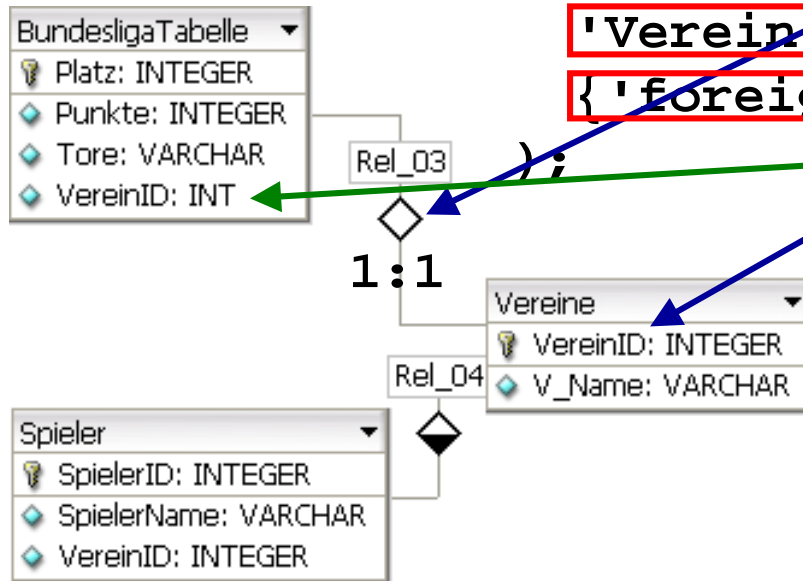
```
package My::DB::Bundesliga;
```

```
# ...
```

```
__PACKAGE__->has_one(
```

```
'Verein' => 'My::DB::Verein',
```

```
{'foreign.VereinID' => 'self.VereinID'}
```



# Beziehungen

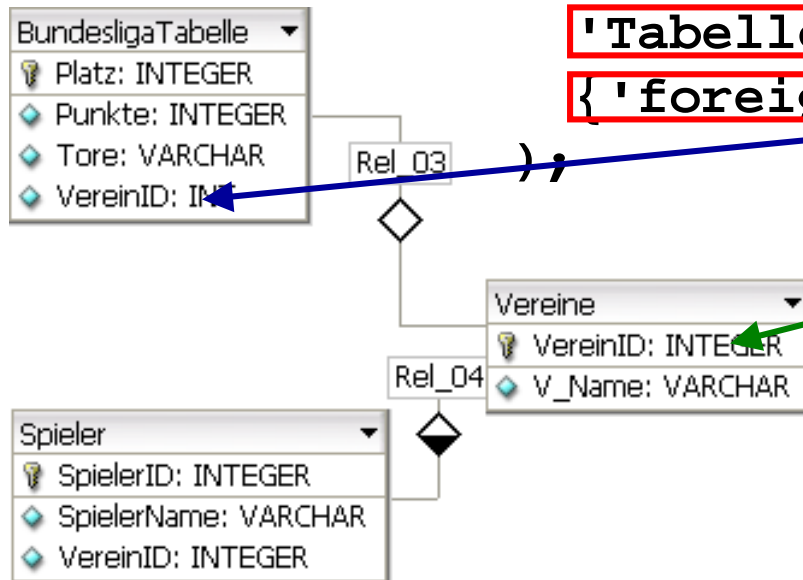
```
package My::DB::Verein;
```

```
# ...
```

```
__PACKAGE__->has_one(
```

```
  'Tabelle' => 'My::DB::Bundesliga',
```

```
  {'foreign.VereinID' => 'self.VereinID'});
```



# Beziehungen

```
package My::DB::Verein;
```

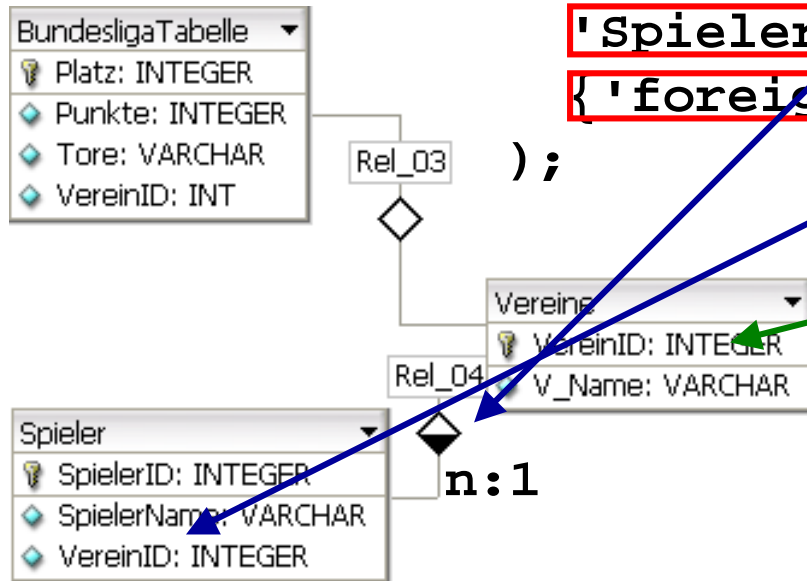
```
# ...
```

```
__PACKAGE__->has_many(
```

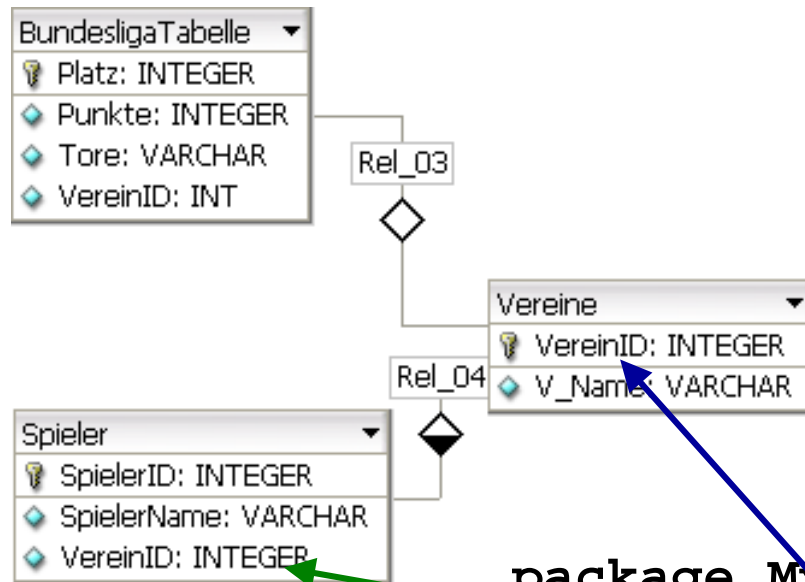
```
'Spieler' => 'My::DB::Spieler',
```

```
{'foreign.VereinID' => 'self.VereinID'}
```

```
);
```

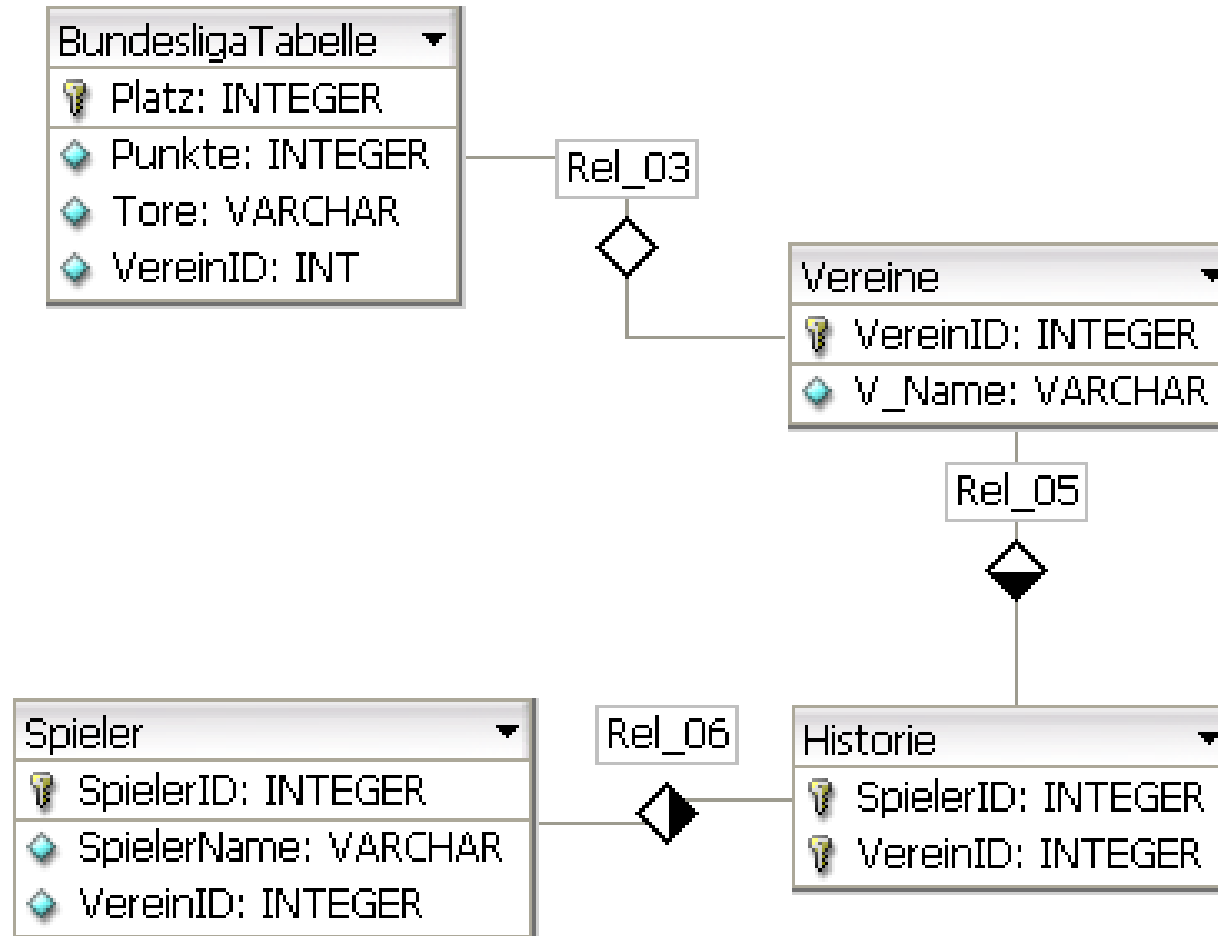


# Beziehungen

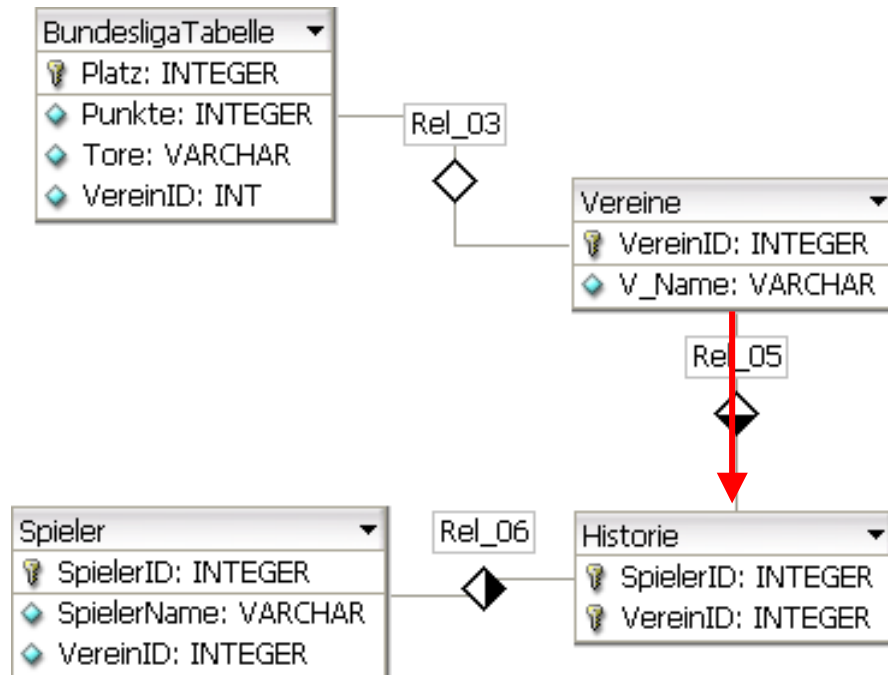


```
package My::DB::Spieler;  
# ...  
__PACKAGE__->belongs_to(  
    'Verein' => 'My::DB::Verein',  
    {'foreign.VereinID' => 'self.VereinID'}  
);
```

# Beziehungen



# Beziehungen

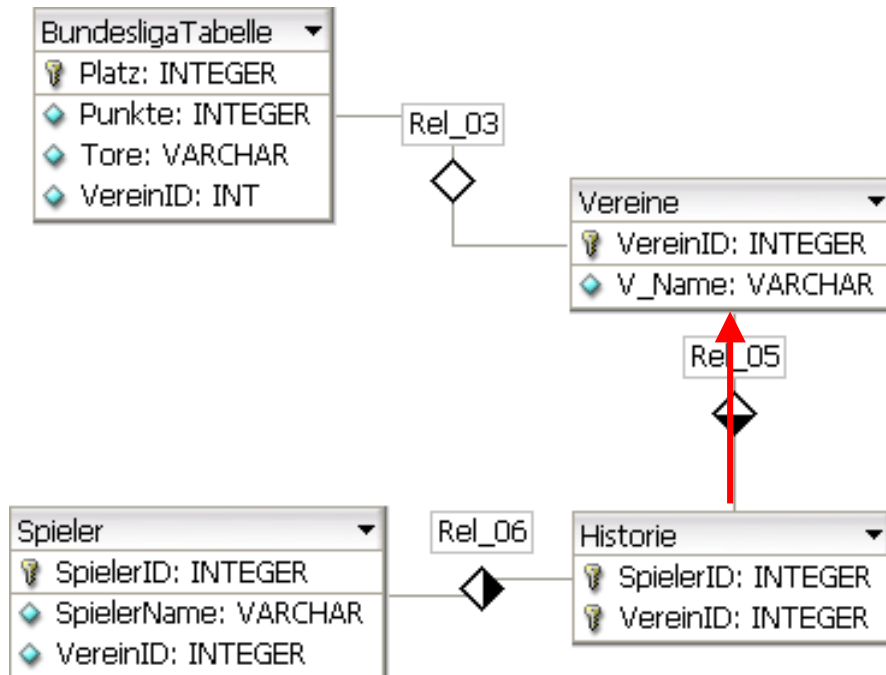


```
package My::DB::Verein;
```

```
#...
```

```
__PACKAGE__->has_many(  
    'History'=>'My::DB::History',  
    {'foreign.VereinID' =>  
     'self.VereinID'});
```

# Beziehungen

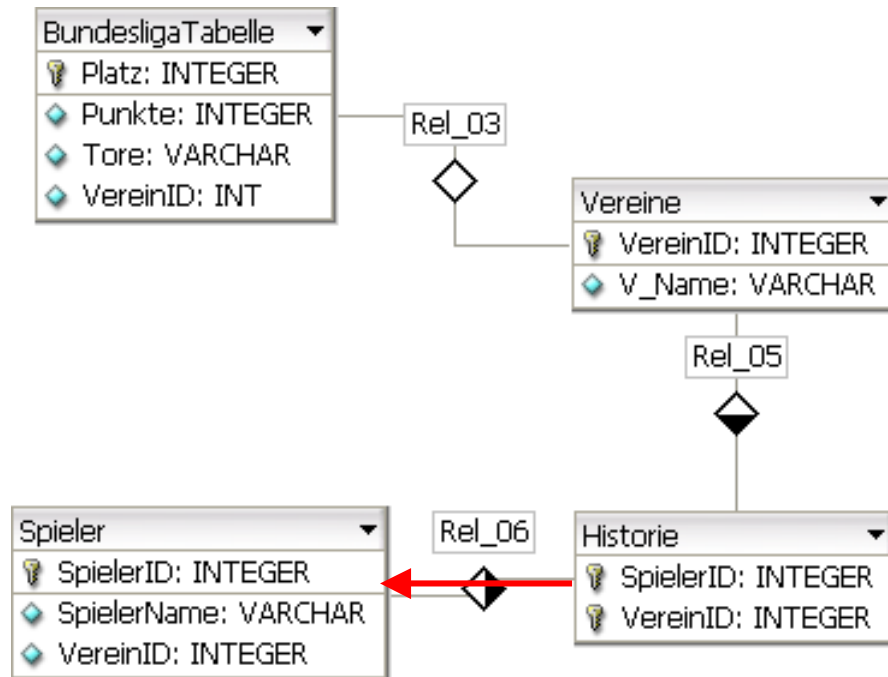


```
package My::DB::History;
```

```
# ...
```

```
__PACKAGE__->belongs_to(  
    'Vereine' => 'My::DB::Verein',  
    {'foreign.VereinID' =>  
        'self.VereinID'});
```

# Beziehungen

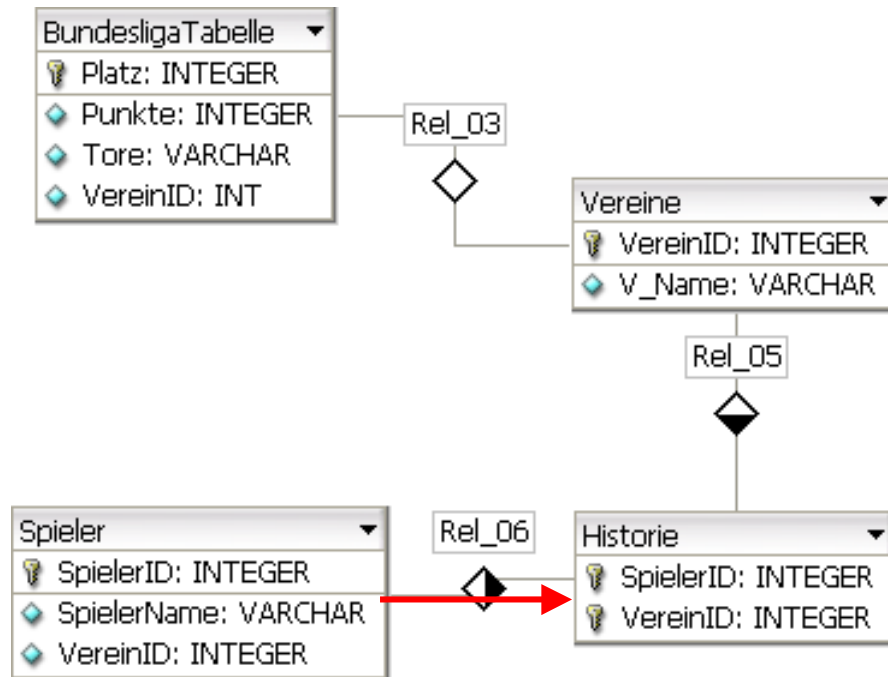


```
package My::DB::History;
```

```
# ...
```

```
__PACKAGE__->belongs_to(  
  'Spieler' => 'My::DB::Spieler',  
  {'foreign.SpielerID' =>  
    'self.SpielerID'});
```

# Beziehungen

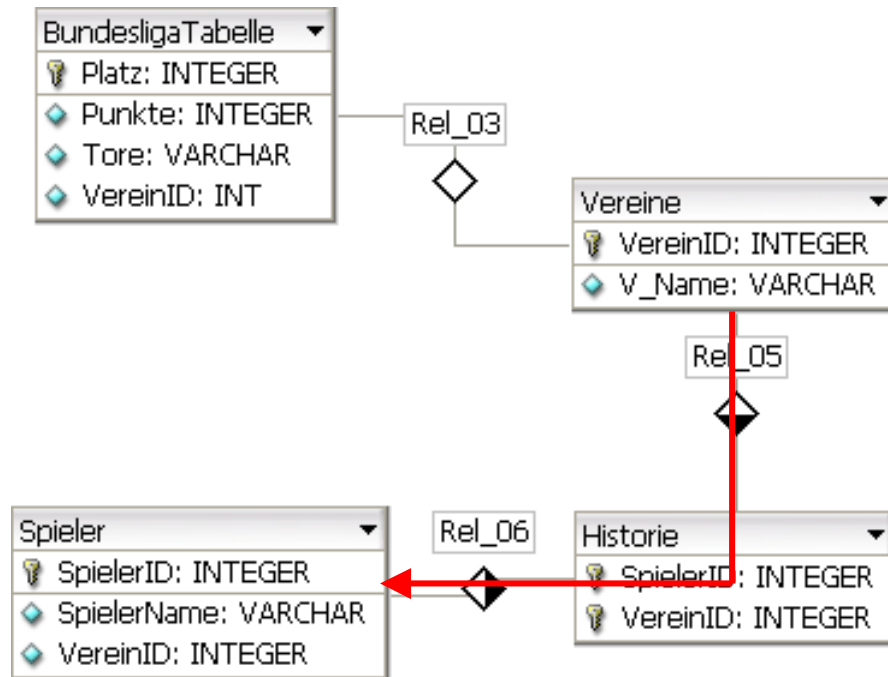


```
package My::DB::Spieler;
```

```
# ...
```

```
__PACKAGE__->has_many(  
    'Vereine' => 'My::DB::History',  
    {'foreign.SpielerID' =>  
        'self.SpielerID'});
```

# Beziehungen

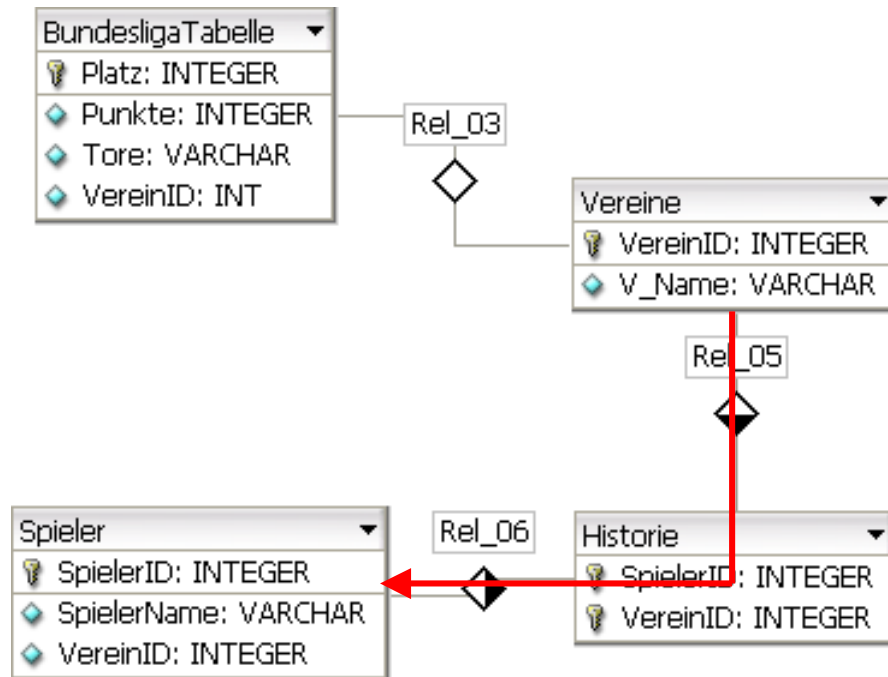


```
package My::DB::Verein;
```

```
# ...
```

```
__PACKAGE__->many_to_many(  
  'Spieler' => 'History',  
  'Spieler' );
```

# Beziehungen

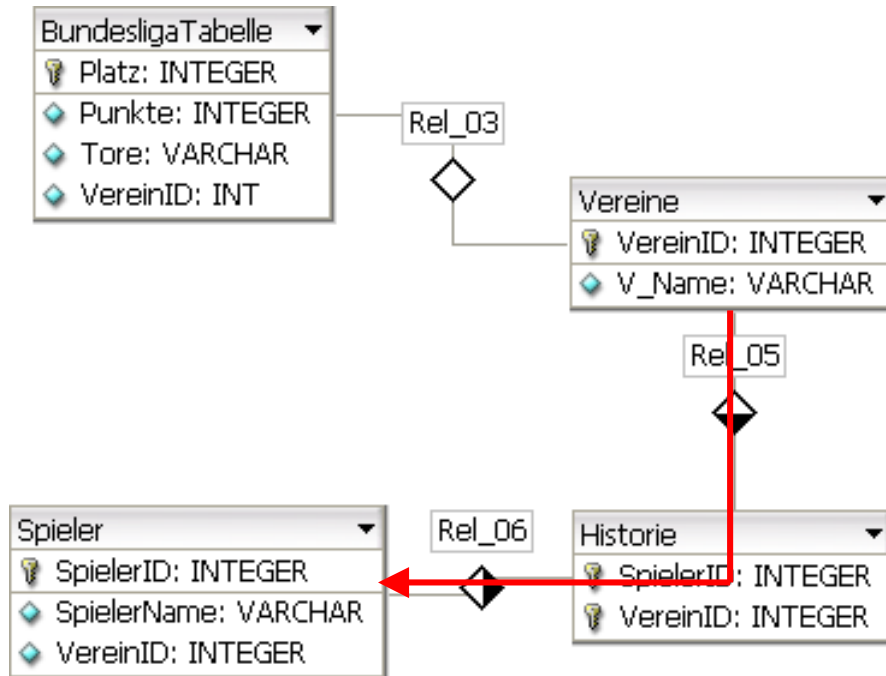


```
package My::DB::Verein;
```

```
# ...
```

```
__PACKAGE__->many_to_many(  
  'Spieler' => 'History',  
  'Spieler' );
```

# Beziehungen



```
package My::DB::Verein;
```

```
#...
```

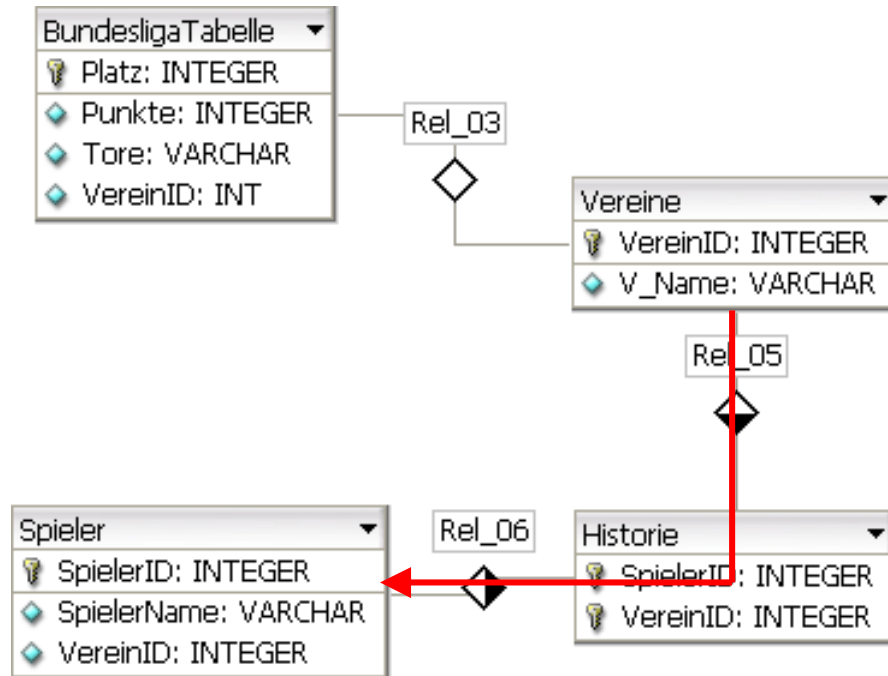
```
__PACKAGE__->has_many(  
  'History'=>'My::DB::History',  
  {'foreign.VereinID' =>  
   'self.VereinID'});
```

```
package My::DB::Verein;
```

```
# ...
```

```
__PACKAGE__->many_to_many(  
  'V_Spieler' => 'History',  
  'Spieler' );
```

# Beziehungen



```
package My::DB::History;
```

```
# ...
```

```
__PACKAGE__->belongs_to(  
    'Spieler' => 'My::DB::Spieler',  
    {'foreign.SpielerID' =>  
        'self.SpielerID'});
```

```
package My::DB::Verein;
```

```
# ...
```

```
__PACKAGE__->many_to_many(  
    'V_Spieler' => 'History',  
    'Spieler');
```

## Und wie benutze ich das?

```
package My::DB;
```

```
use strict;
```

```
use warnings;
```

```
→ use base qw(DBIx::Class::Schema);
```

```
→ __PACKAGE__->load_classes(qw/  
    Bundesliga  
    Verein  
    Spieler  
    History  
/);
```

```
1;
```

# Das Skript...

```
#!/usr/bin/perl
```

```
use strict;
```

```
use warnings;
```

```
→ use My::DB;
```

```
→ my $schema = My::DB->connect(...);
```

```
→ my @entries = $schema->resultset('Bundesliga')->all;
```

```
for my $entry(@entries){
```

```
    print sprintf "%2d %-12s (ID: %2d) %2d %6s\n",
```

```
→         $entry->Platz,
```

```
→         $entry->Verein->V_Name,
```

```
         $entry->VereinID,
```

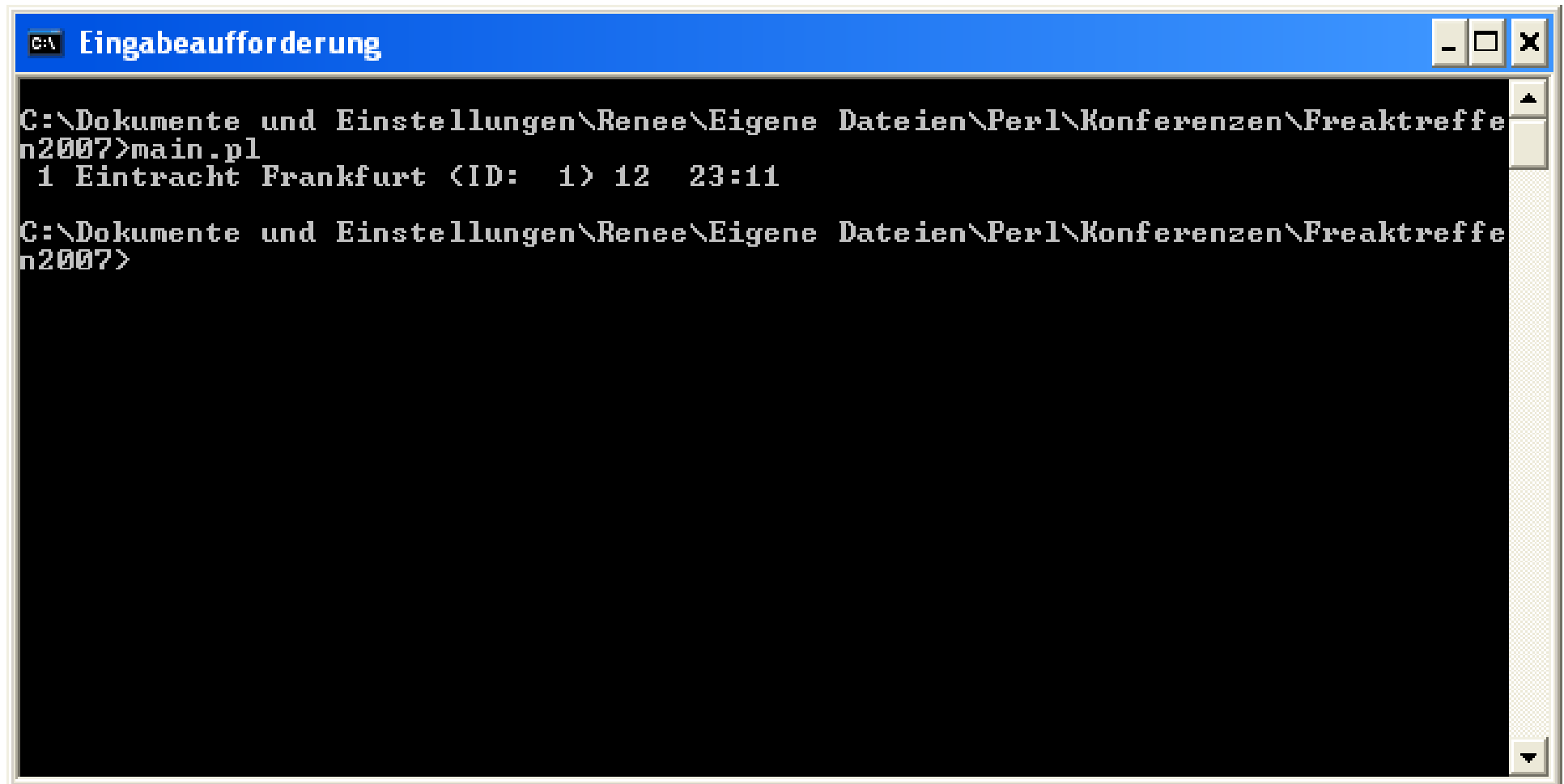
```
         $entry->Punkte,
```

```
         $entry->Tore);
```

```
}
```

```
'Verein' => 'My::DB::Verein',
```

# Live-Demo



```
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\Konferenzen\Freaktreff  
n2007>main.pl  
1 Eintracht Frankfurt <ID: 1> 12 23:11  
  
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\Konferenzen\Freaktreff  
n2007>
```

## Einträge suchen

```
my ($verein) = $schema->resultset( 'Bundesliga' )->search({  
    VereinID => 1,  
});
```

```
print_info( $verein );
```

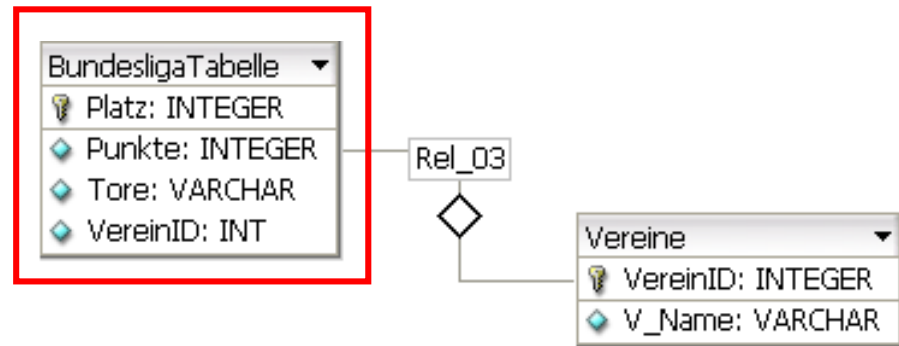
```
sub print_info{  
    my ($verein) = @_;  
    my $obj = $verein->Verein;
```

```
    print "Name:\n  ", $obj->V_Name, "\n\n";
```

```
    print "Tabellensituation:\n", $verein->Platz,  
          " ", $verein->Punkte, " ", $verein->Tore, "\n\n";
```

```
    print "Spieler:\n";  
    my ($spieler) = $verein->Verein->Spieler;  
    print $spieler->SpielerName;
```

```
}
```



## Einträge suchen

```
my ($verein) = $schema->resultset( 'Bundesliga' )->search({
    VereinID => 1,
});
```

```
print_info( $verein );
```

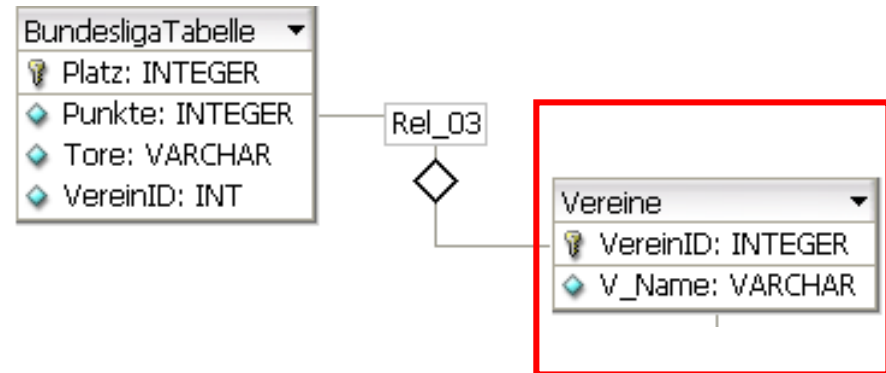
```
sub print_info{
    my ($verein) = @_;
    my $obj = $verein->Verein;
```

```
    print "Name:\n  ", $obj->V_Name, "\n\n";
```

```
    print "Tabellensituation:\n", $verein->Platz,
          " ", $verein->Punkte, " ", $verein->Tore, "\n\n";
```

```
    print "Spieler:\n";
    my ($spieler) = $verein->Verein->Spieler;
    print $spieler->SpielerName;
```

```
}
```



## Einträge suchen

```
my ($verein) = $schema->resultset( 'Bundesliga' )->search({  
    VereinID => 1,  
});
```

```
print_info( $verein );
```

```
sub print_info{  
    my ($verein) = @_;  
    my $obj = $verein->Verein;
```

```
    print "Name:\n  ", $obj->V
```

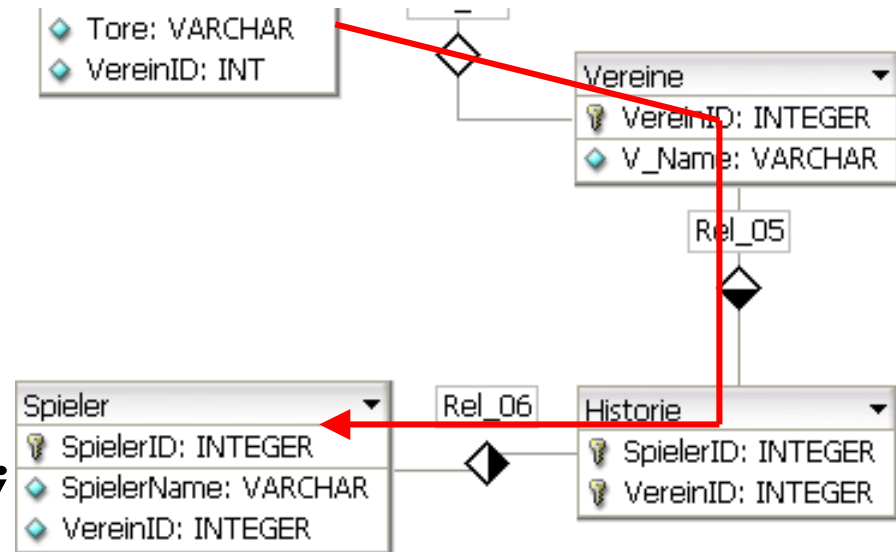
```
    print "Tabellensituation:\n", $verein->Platz,  
          " ", $verein->Punkte, " ", $verein->Tore, "\n\n";
```

```
    print "Spieler:\n";
```

```
    my ($spieler) = $verein->Verein->Spieler;
```

```
    print $spieler->SpielerName;
```

```
}
```



## Alte Einträge aktualisieren

```
my ($spieler) = $verein->Verein->Spieler;  
$spieler->set_columns({  
    SpielerName => 'Takahara',  
});  
$spieler->update;
```

VS.

```
my ($spieler) = $verein->Verein->Spieler;  
$spieler->update({  
    SpielerName => 'Takahara',  
});
```

## Neue Einträge in der Datenbank

```
my $neuer_spieler =  
  $schema->resultset( 'Spieler' )  
    ->create({  
      SpielerID => 2,  
      SpielerName => 'Kyrgiakos',  
    });  
  
$neuer_spieler->update;
```

## Einträge löschen

```
my ($spieler) = $schema->resultset( 'Spieler' )
    ->search({
    SpielerID => 3,
});

print $spieler->SpielerName;

$spieler->delete;
```

# Komplexere Abfragen

```
#SELECT cd.title FROM cd JOIN map ON map.cd = cd.id
#         JOIN genre ON map.genre = genre.id
#         WHERE cd.year = 1995
#         AND genre.name = 'Techno';
```

```
my @all_titles = $schema->resultset( 'CD' )
                    ->search( {
                        'genre.name' => 'Techno',
                        'me.year' => '1995',
                    },
                    {
                        join => [qw/map genre/],
                    } )->all;
```

# Komplexere Abfragen

```
my @albums = $schema->resultset('Album')->search({
  -or => [
    -and => [
      artist => { 'like', '%Smashing Pumpkins%' },
      title => 'Siamese Dream',
    ],
    artist => 'Starchildren',
  ],
});
```

```
# WHERE ( artist LIKE '%Smashing Pumpkins%'
#         AND title = 'Siamese Dream' )
#       OR artist = 'Starchildren'
```

## Fazit

- DBIx::Class ist sehr gut geeignet wenn man SQL nicht sonderlich mag.
- Durch die Relationships ist „Navigation“ durch Tabellen möglich...
- Komplexe SQL-Befehle können auch bei DBIx::Class ziemlich komplex werden

## FabForce::DBDesigner4::DBIC

- Unterstützt beim Erstellen von Klassen
- Noch am Anfang der Entwicklung

```
#!/usr/bin/perl

use strict;
use warnings;
use FabForce::DBDesigner4::DBIC;

my $dbic = FabForce::DBDesigner4::DBIC->new(
    inputfile => 'test.xml',
    namespace => 'Freaktreffen2007',
);
$dbic->create_scheme;
```

## Referenzen

- <http://search.cpan.org/dist/DBIx-Class/>
- <http://faq.perl-community.de/bin/view/Wissensbasis/DBIxClassLiteralSQL>
- <http://perlcast.com/2007/06/12/matt-trout-on-dbixclass/>