

Работа с базами данных

SQL

```
SELECT name, surname  
FROM users  
WHERE age > 18;
```

```
SELECT balance  
FROM account  
WHERE user_id = 81858
```

```
SELECT *  
FROM users u JOIN accounts a  
    ON u.id = a.user_id  
WHERE account.balance > 0
```

DBI

```
$dbh = DBI->connect(  
    $dsn, $user, $password,  
    {RaiseError => 1, AutoCommit => 0}  
);  
  
$dbh->do($sql);
```

connect

```
$dbh = DBI->connect($data_source,  
    user, $password, {...});  
  
# DBD::SQLite  
$dbh = DBI->connect("dbi:SQLite:dbname=dbfile",  
    "", "");  
  
# DBD::mysql  
$dbh = DBI->connect(  
    "DBI:mysql:database=$database;" .  
    "host=$hostname;port=$port",  
    $user, $password  
);
```

dbi:DriverName:database_name

dbi:DriverName:database_name@hostname:port

dbi:DriverName:database=DBNAME;host=HOSTNAME;port=PORT

do

```
my $number_of_rows = $dbh->do(  
    'DELETE FROM user WHERE age < 18'  
);  
  
my $name = <>;  
$dbh->do("DELETE FROM user WHERE name = '$name'");
```

SQL injections

```
my $name = q{' OR (DELETE FROM log) AND '' = '};  
$dbh->do("DELETE FROM user WHERE name = '$name'");
```

```
DELETE FROM user WHERE name = ''  
OR (DELETE FROM log) AND '' = ''
```

```
$name = $dbh->quote($name);
```

prepare, execute

```
my $sth = $dbh->prepare(  
    'DELETE FROM user WHERE name = ?'  
);  
  
$sth->execute( 'Vadim' );
```

fetchrow

```
my $ary_ref = $sth->fetchrow_arrayref();  
my @ary     = $sth->fetchrow_array();  
my $hash    = $sth->fetchrow_hashref();  
  
while (@row = $sth->fetchrow_array()) {  
    print "@row\n";  
}
```


fetchall_arrayref

```
my $ary = $sth->fetchall_arrayref;
# [ [...], [...], [...] ]

my $ary = $sth->fetchall_arrayref({});
# [ {...}, {...}, {...} ]

$tbl_ary_ref = $sth->fetchall_arrayref(
    [0]
);

$tbl_ary_ref = $sth->fetchall_arrayref(
    [-2,-1]
);

$tbl_ary_ref = $sth->fetchall_arrayref({
    foo => 1,
    BAR => 1,
});
```

fetchall_hashref

```
$sth->fetchall_hashref('id');  
# { 1 => {...}, 2 => {...} }  
  
$sth->fetchall_hashref([ qw(foo bar) ]);  
  
{  
  1 => { a => {...}, b => {...} },  
  2 => { a => {...}, b => {...} },  
}
```

selectrow

```
$dbh->selectrow_array(  
    $statement, \%attr, @bind_values  
);  
  
$dbh->selectrow_arrayref(  
    $statement, \%attr, @bind_values  
);  
  
$dbh->selectrow_hashref(  
    $statement, \%attr, @bind_values  
);
```

selectall

```
$dbh->selectall_arrayref(  
    $statement, \%attr, @bind_values);  
  
$dbh->selectall_hashref(  
    $statement, $key_field, \%attr, @bind_values);  
  
$dbh->selectall_arrayref(  
    "SELECT ename FROM emp ORDER BY ename",  
    { Slice => {} }  
);
```

Errors

```
$dbh = DBI->connect(  
    "dbi:DriverName:db_name", $user, $password,  
    { RaiseError => 1 }  
);
```

```
$dbh->err;  
$dbh->errstr;
```

Transactions

```
$dbh = DBI->connect(  
    "dbi:DriverName:db_name", $user, $password,  
    { AutoCommit => 1 }  
);  
  
$dbh->begin_work;  
$dbh->rollback;  
$dbh->commit;
```

last_insert_id

```
$dbh->do('INSERT INTO user VALUES(...)');  
  
my $user_id = $dbh->last_insert_id(  
    $catalog, $schema, $table, $field, \%attr  
);
```

DBIx::Class

```
package Local::Schema::User;
use base qw(DBIx::Class::Core);

__PACKAGE__->table('user');
__PACKAGE__->add_columns(
    id => {
        data_type => 'integer',
        is_auto_increment => 1,
    },
    name => {
        data_type => 'varchar',
        size      => '100',
    },
    superuser => {
        data_type => 'bool',
    },
);
```


DBIx::Class

```
--PACKAGE__->set_primary_key('id');  
--PACKAGE__->has_many(  
    visits => 'Local::Schema::Visit',  
    'user_id'  
);  
--PACKAGE__->many_to_many(  
    visited_cities => 'visits',  
    'city'  
);
```

Files

```
package Local::Schema;  
use base qw/DBIx::Class::Schema/;  
  
__PACKAGE__->load_namespaces();  
  
1;
```

```
Local::Schema::Result::*;  
Local::Schema::ResultSet::*;
```

resultset, result

```
my $resultset = $schema->resultset('User');  
my $resultset2 = $resultset->search({age => 25});  
  
while (my $user = $resultset->next) {  
    print $user->name . "\n";  
}  
  
print join "\n", $resultset2->all();
```

search

```
$rs = $rs->search({  
    age => {'>=' => 18},  
    parent_id => undef,  
});
```

```
@results = $rs->all();  
@results = $rs->search(...);  
$rs = $rs->search(...);  
$rs = $rs->search_rs(...);
```

search — attributes

```
$rs = $rs->search(  
  { page => { '>=' => 18 } },  
  { order_by => { -desc => [qw(a b c)] } },  
);  
  
$rs = $rs->search(undef, {rows => 100});
```

search — duplicate key

```
# :-(
$rs = $rs->search({
  age => {'>=' => 18},
  age => {'<'  => 60},
});

# :-)
$rs = $rs->search([
  { age => {'>=' => 18} },
  { age => {'<'  => 60} },
]);
```

find, single

```
my $rs = $schema->resultset('User');  
  
$user = $rs->find({id => 81858});  
$user = $rs->find(81858);  
  
$user = $rs->search({id => 81858})->single();
```

count

```
my $count = $schema->resultset('User')->search({  
  name => 'name',  
  age => 18,  
})->count();
```


select — advanced

```
$resultset->search({  
  date => { '>' => \'NOW()\' },  
});  
  
$rs->search(  
  \[ 'YEAR(date_of_birth) = ?', 1979 ]  
);  
  
my @albums = $schema->resultset('Album')->search({  
  -or => [  
    -and => [  
      artist => { 'like', '%Smashing Pumpkins%' },  
      title  => 'Siamese Dream',  
    ],  
    artist => 'Starchildren',  
  ],  
});
```

Relations

```
package Local::Schema::User;
use base qw(DBIx::Class::Core);

__PACKAGE__->table('user');
__PACKAGE__->has_many(
    dogs => 'Local::Schema::Dog',
    'user_id'
);

package Local::Schema::Dog;
use base qw(DBIx::Class::Core);

__PACKAGE__->table('dog');
__PACKAGE__->belongs_to(
    user => 'Local::Schema::User',
    'user_id'
);
```

Relations — usage

```
$user = $schema->resultset('User')->find(81858);  
  
foreach my $dog ($user->dogs) {  
    print join(' ', $dog->id, $dog->user->id);  
}
```

join

```
$rs = $schema->resultset('Dog')->search({  
    'me.name' => 'Sharik',  
    'user.name' => 'Vadim',  
}, {  
    join => 'user',  
});
```

prefetch

```
foreach my $user ($schema->resultset('User')) {  
    foreach my $dog ($user->dogs) {  
        # ...  
    }  
}
```

```
$rs = $schema->resultset('User')->search({}, {  
    prefetch => 'dogs', # implies join  
});
```

Custom resultset methods

```
my @women = $schema->resultset('User')->
    search_women()->all();
```

```
package Local::Schema::ResultSet::User;

sub search_women {
    my ($self) = @_;

    return $self->search({
        gender => 'f',
    });
}
```

Custom result methods

```
foreach my $woman (@women) {  
    $woman->log('was selected');  
}
```

```
package Local::Schema::Result::User;  
  
sub log {  
    print {$log} @_;  
}
```

new_result, create

```
my $user = $schema->resultset('User')->new_result({  
  name => 'Vadim',  
  superuser => 1,  
});  
  
$user->insert();
```

```
my $artist = $artist_rs->create(  
  { artistid => 4, name => 'Blah-blah', cds => [  
    { title => 'My First CD', year => 2006 },  
    { title => 'e.t.c', year => 2007 },  
  ],  
  },  
);
```


update, delete

```
$result->last_modified(\ 'NOW()' )->update();  
# OR  
$result->update({ last_modified => \ 'NOW()' }));
```

```
$user->delete();
```

many_to_many

```
package Local::Schema::User;

__PACKAGE__->has_many(
    visits => 'Local::Schema::Visit', 'user_id');
__PACKAGE__->many_to_many(
    visited_cities => 'visits', 'city');

package Local::Schema::City;

__PACKAGE__->has_many(
    visits => 'Local::Schema::Visit', 'city_id');
__PACKAGE__->many_to_many(
    visited_by => 'visits', 'user');
```

many_to_many

```
package Local::Schema::Visit;  
  
__PACKAGE__->belongs_to(  
    user => 'Local::Schema::User', 'user_id');  
__PACKAGE__->belongs_to(  
    city => 'Local::Schema::City', 'city_id');
```

```
my @cities = $schema->resultset('User')->  
    find(81858)->visited_cities;
```

storage

```
$schema->storage->debug(1);  
$schema->storage->dbh();
```

DBIx::Class::Schema::Loader

```
use DBIx::Class::Schema::Loader qw(
    make_schema_at
);

make_schema_at(
    'My::Schema',
    { debug => 1,
      dump_directory => './lib',
    },
    [ 'dbi:Pg:dbname="foo"', 'user', 'pw' ]
);
```

```
dbicdump -o dump_directory=./lib \
-o debug=1 \
My::Schema \
'dbi:Pg:dbname=foo' \
myuser \
mypassword
```

SQL::Translator

```
$schema->deploy();
```

Memcached

```
use Cache::Memcached::Fast;

my $memd = Cache::Memcached::Fast->new({
    servers => [
        {address => 'localhost:11211', weight => 2.5},
        '192.168.254.2:11211',
        '/path/to/unix.sock'
    ],
    namespace => 'my:',
    connect_timeout => 0.2,
    # ...
});
```

Memached — operations

```
$memd->add( 'skey' , 'text' );  
$memd->set( 'nkey' , 5, 60 );  
$memd->incr( 'nkey' );  
$memd->get( 'skey' );
```


Д3

<https://github.com/Nikolo/Technosfera-perl/>

`/homeworks/grades`

