

<http://www.stardata.it>



Giuseppe Maxia

MySQL

Perl / DBI idioms



g.maxia@stardata.it

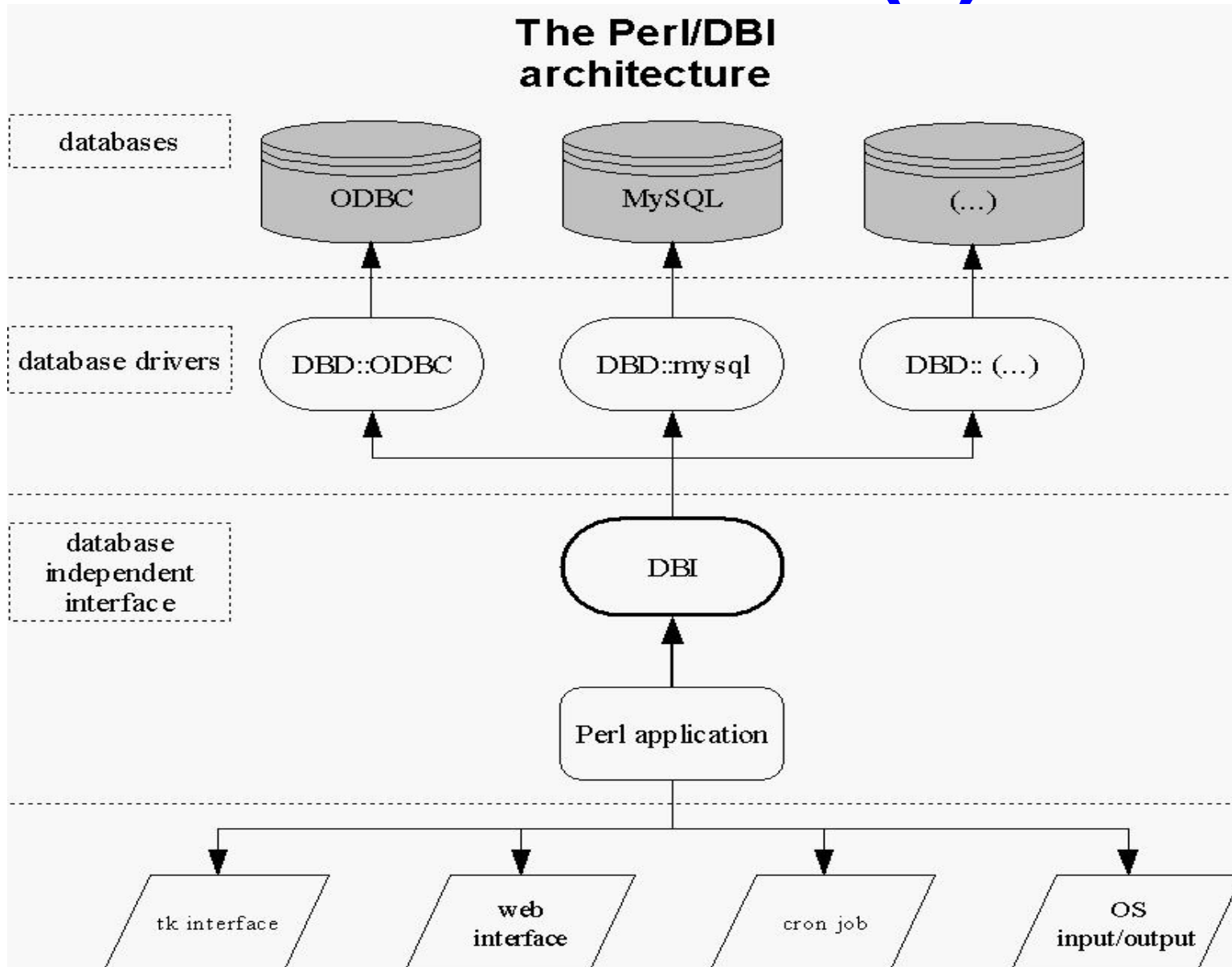


Agenda

- **Recall of Perl / DBI architecture and principles**
- **Connection idioms**
- **Simple inserts**
- **Multiple inserts**
- **Inserting lists**
- **Fetch into complex data**
- **Peculiar multiple fetch operations**
- **Brevity**

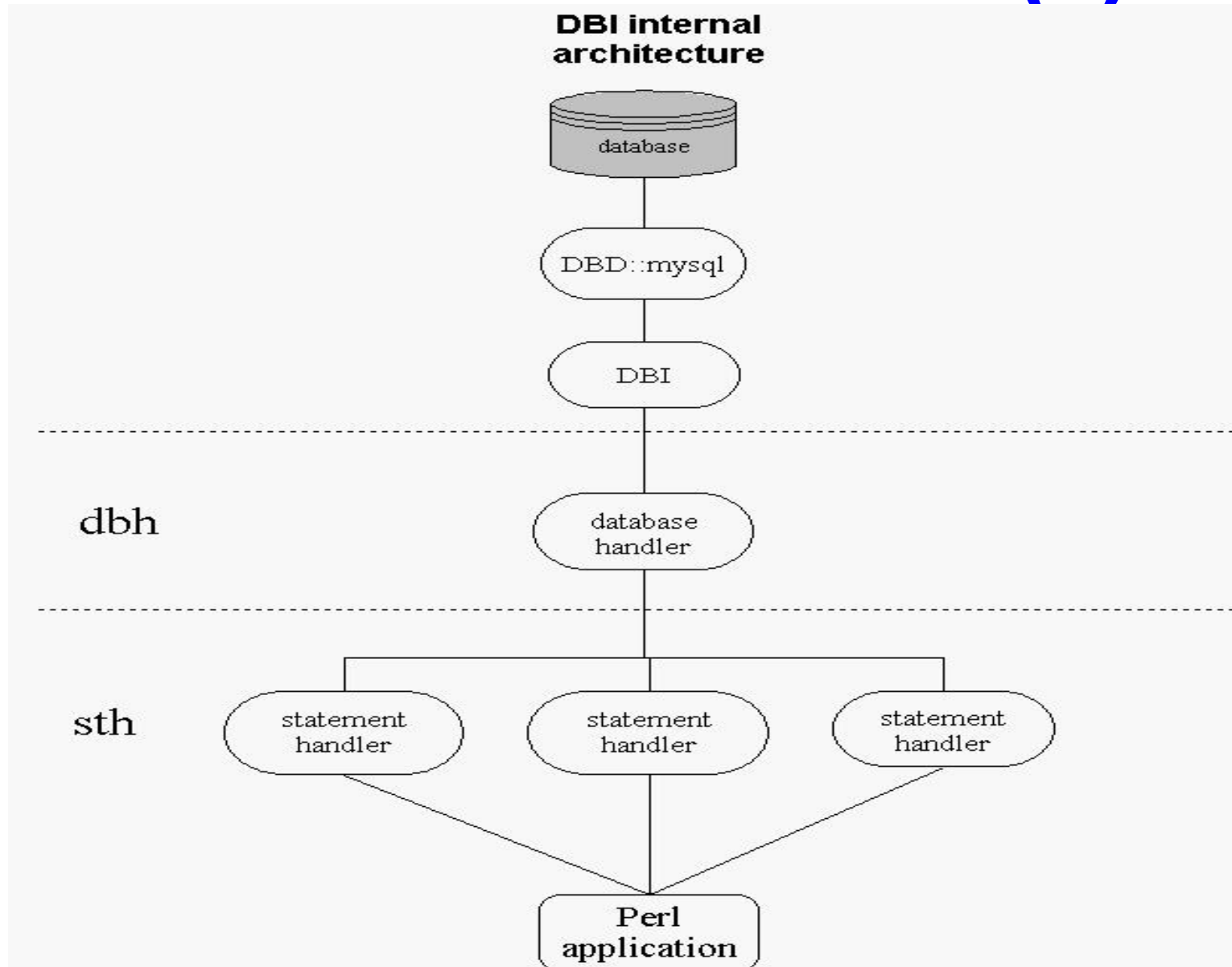


DBI architecture (1)



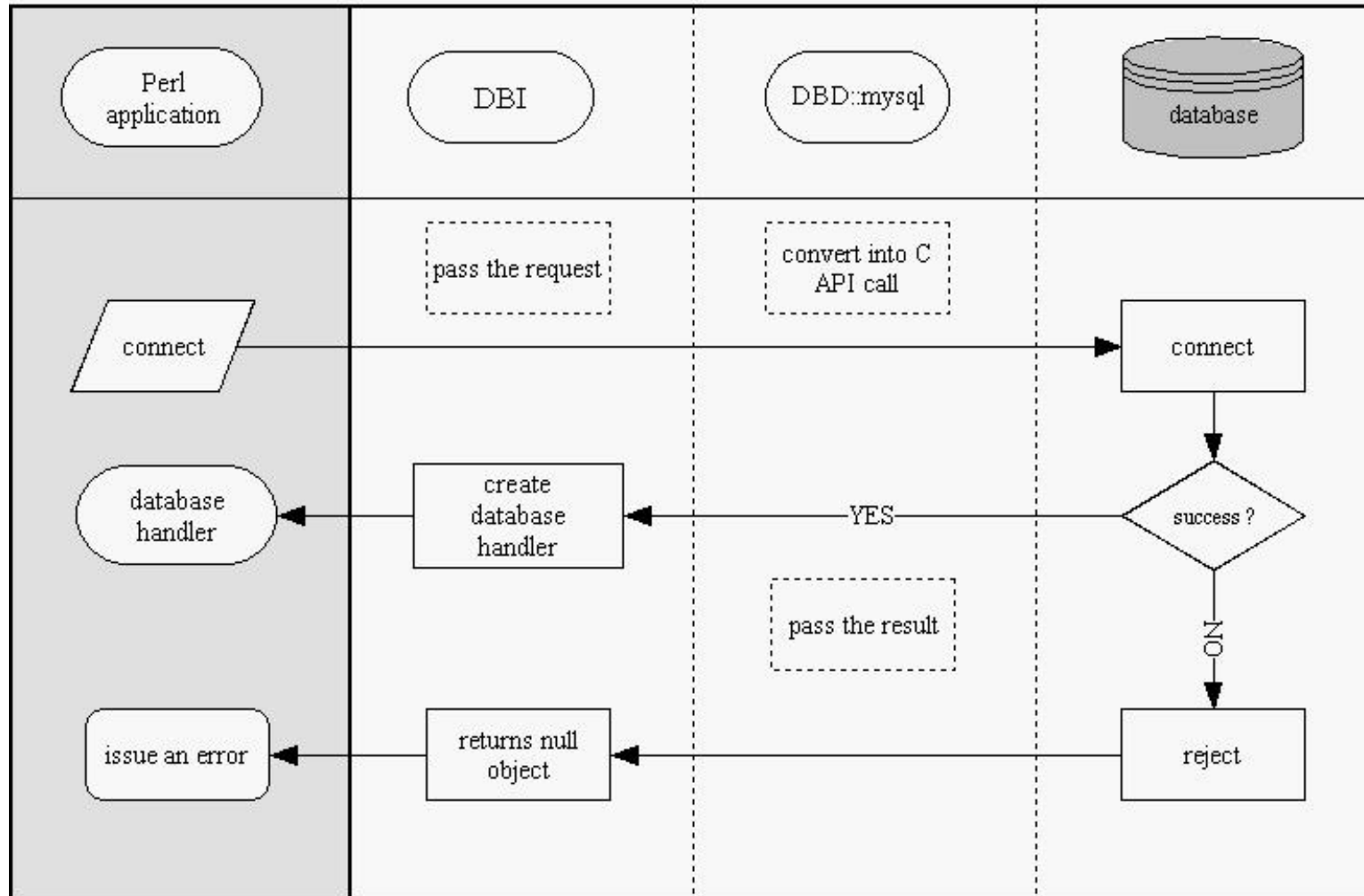


DBI architecture (2)

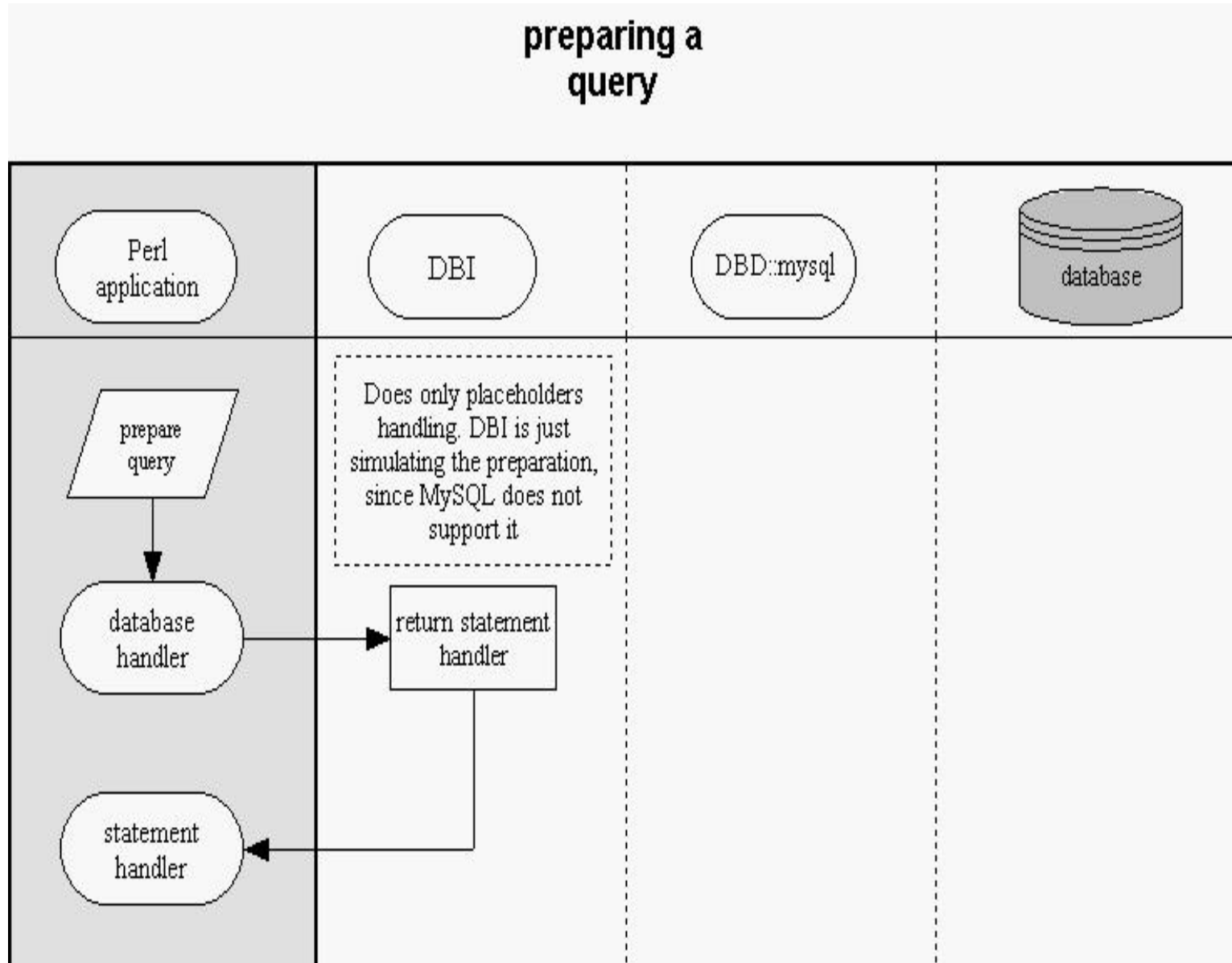


DBI operations (1)

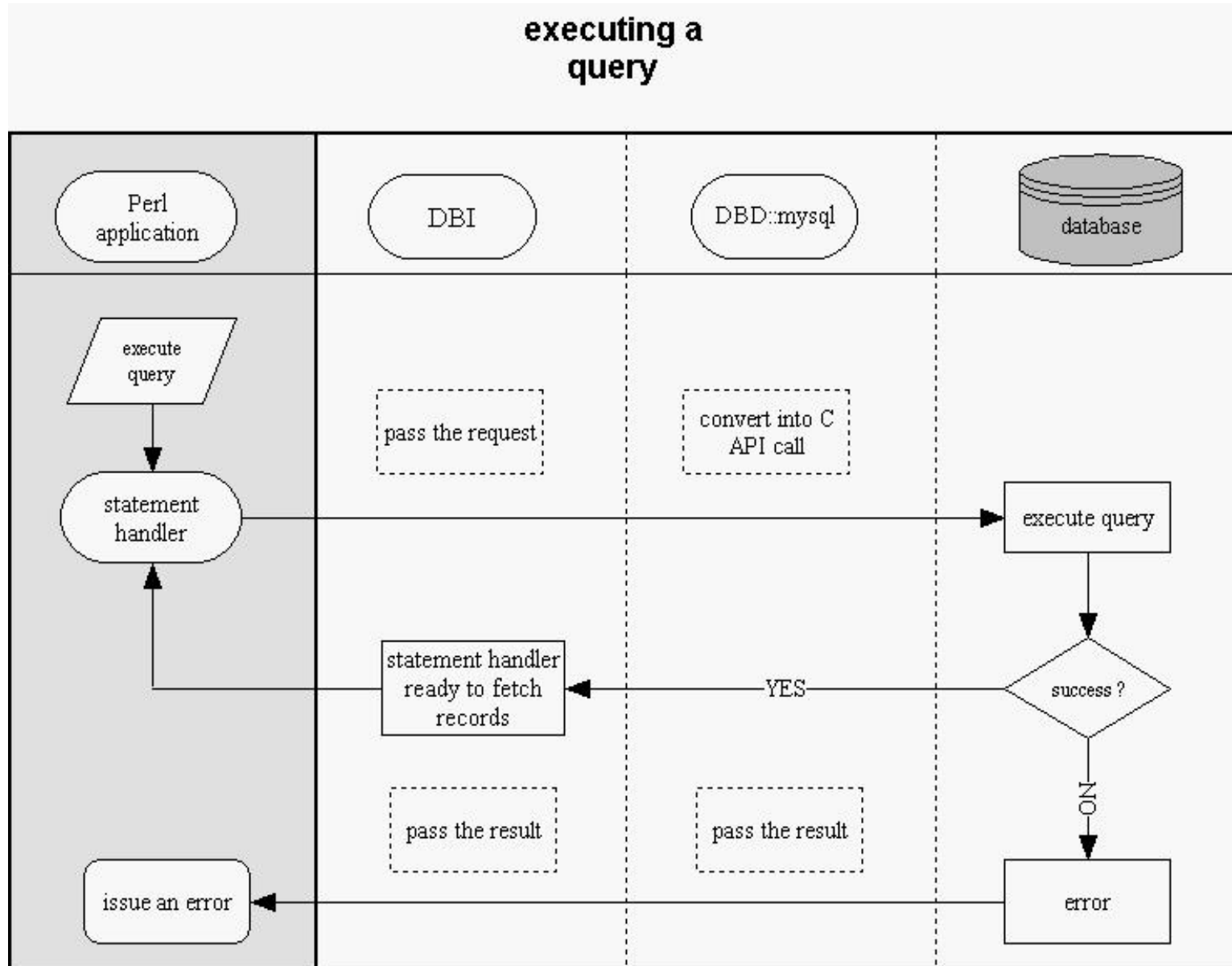
connecting to a database using DBI



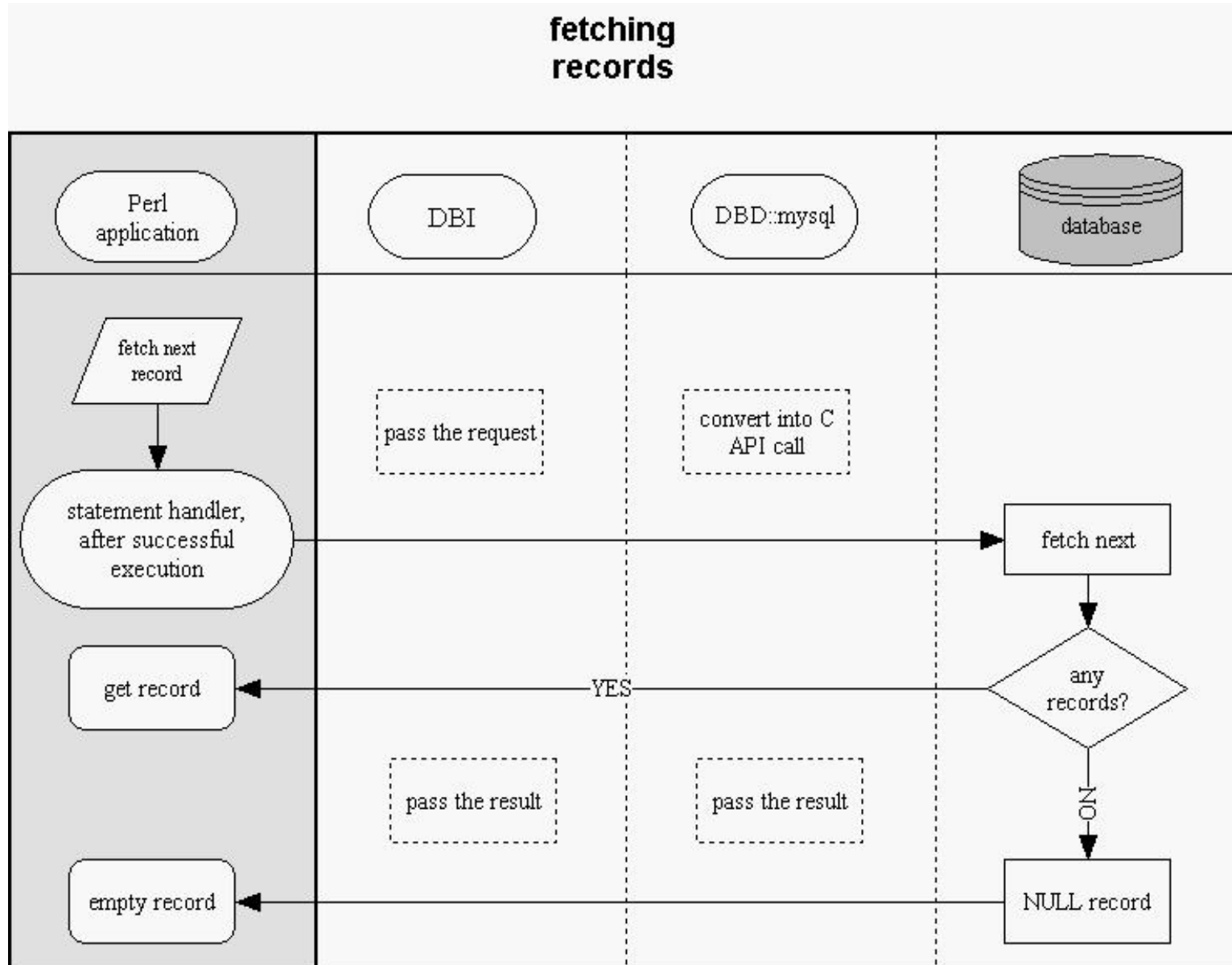
DBI operations (2)



DBI operations (3)

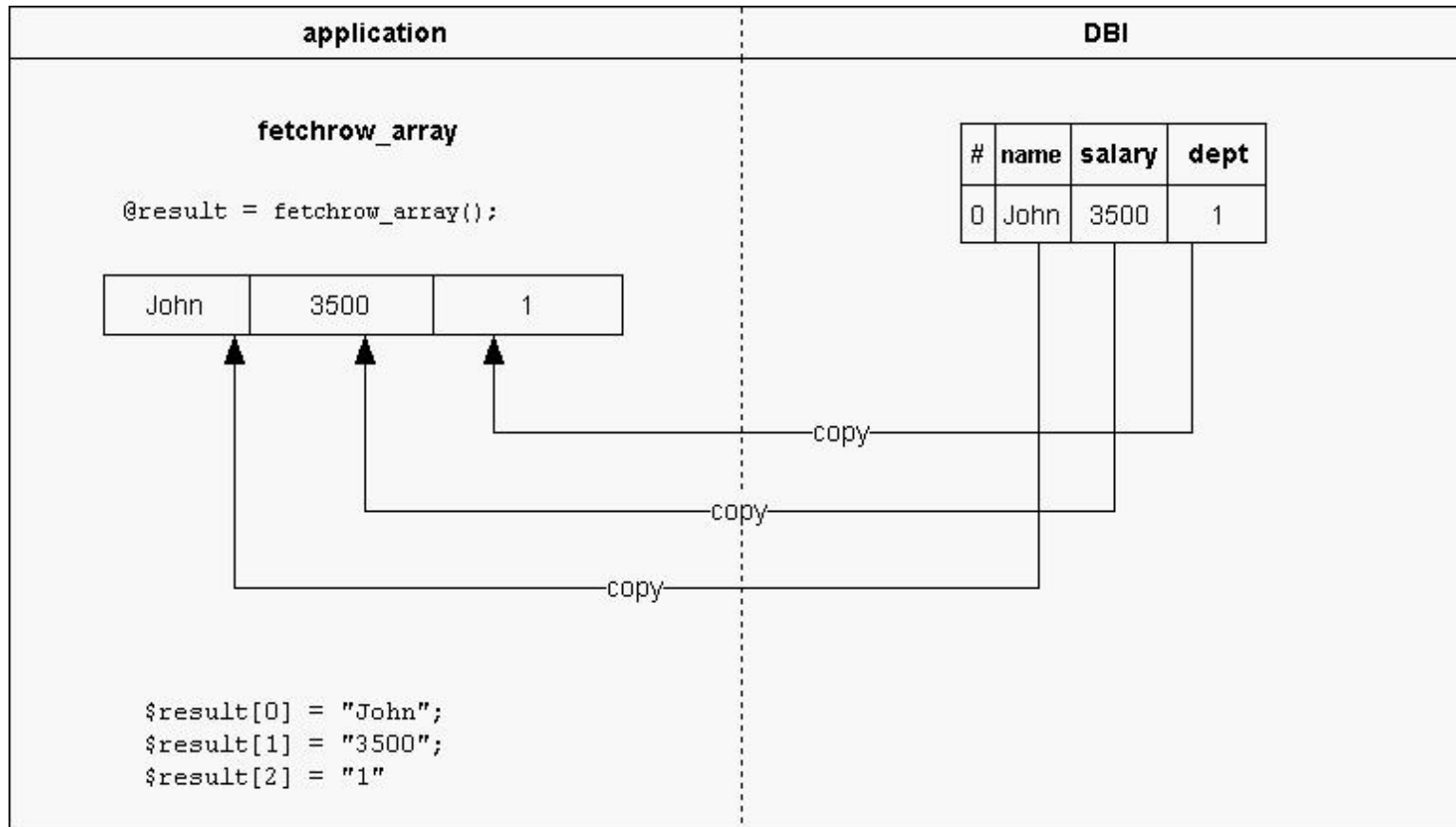


DBI operations (4)



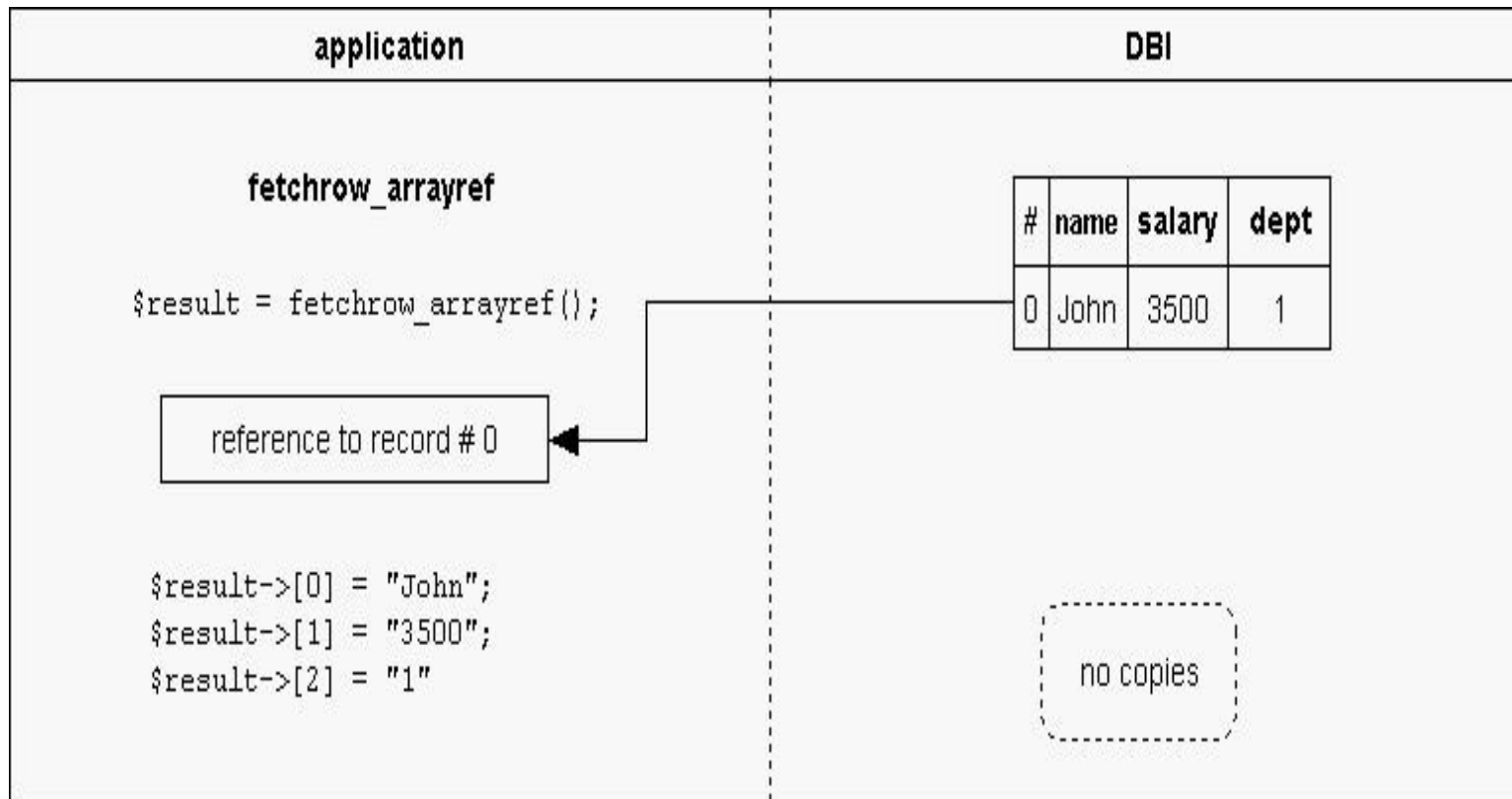


DBI results (1)



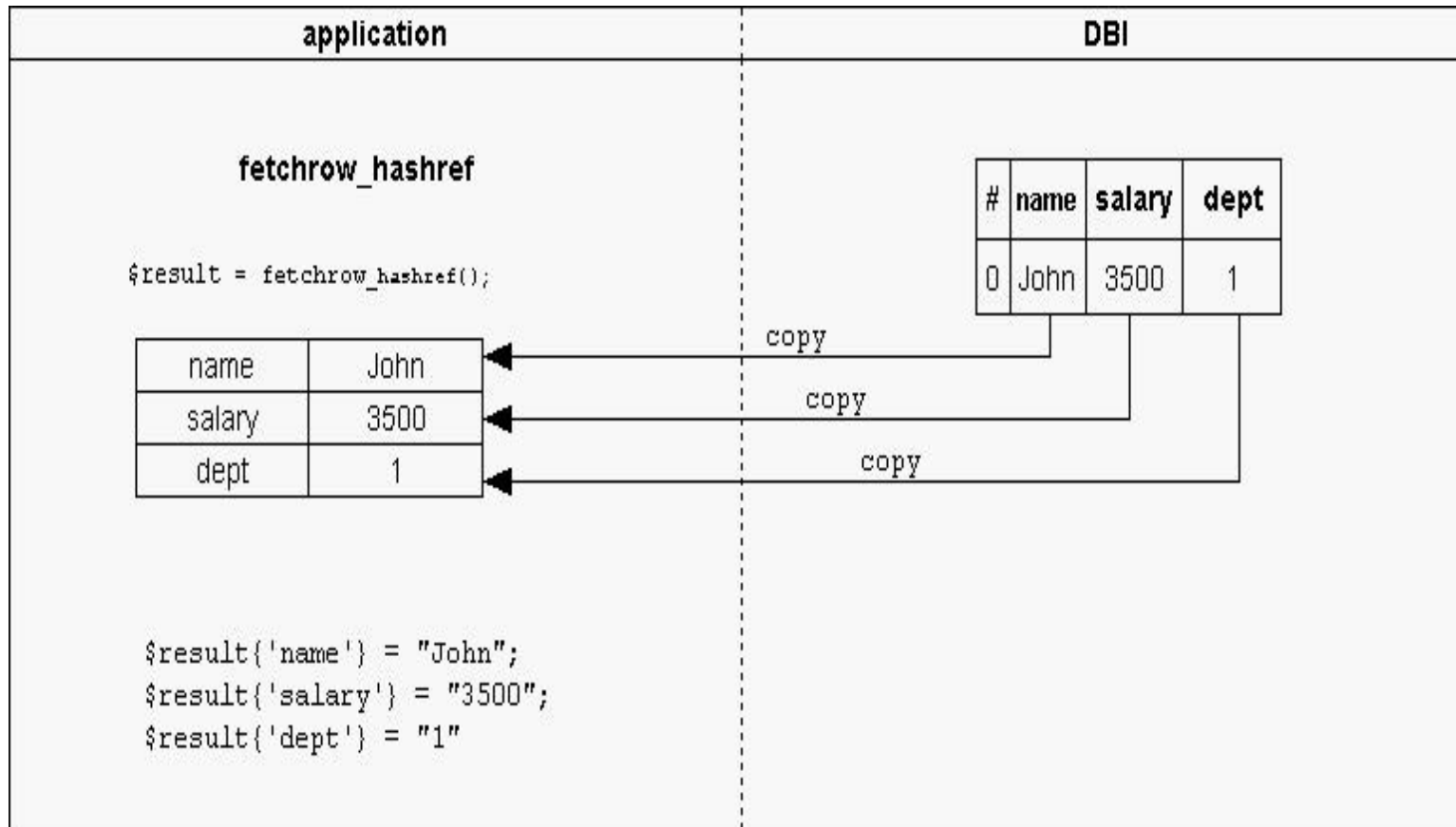


DBI results (2)



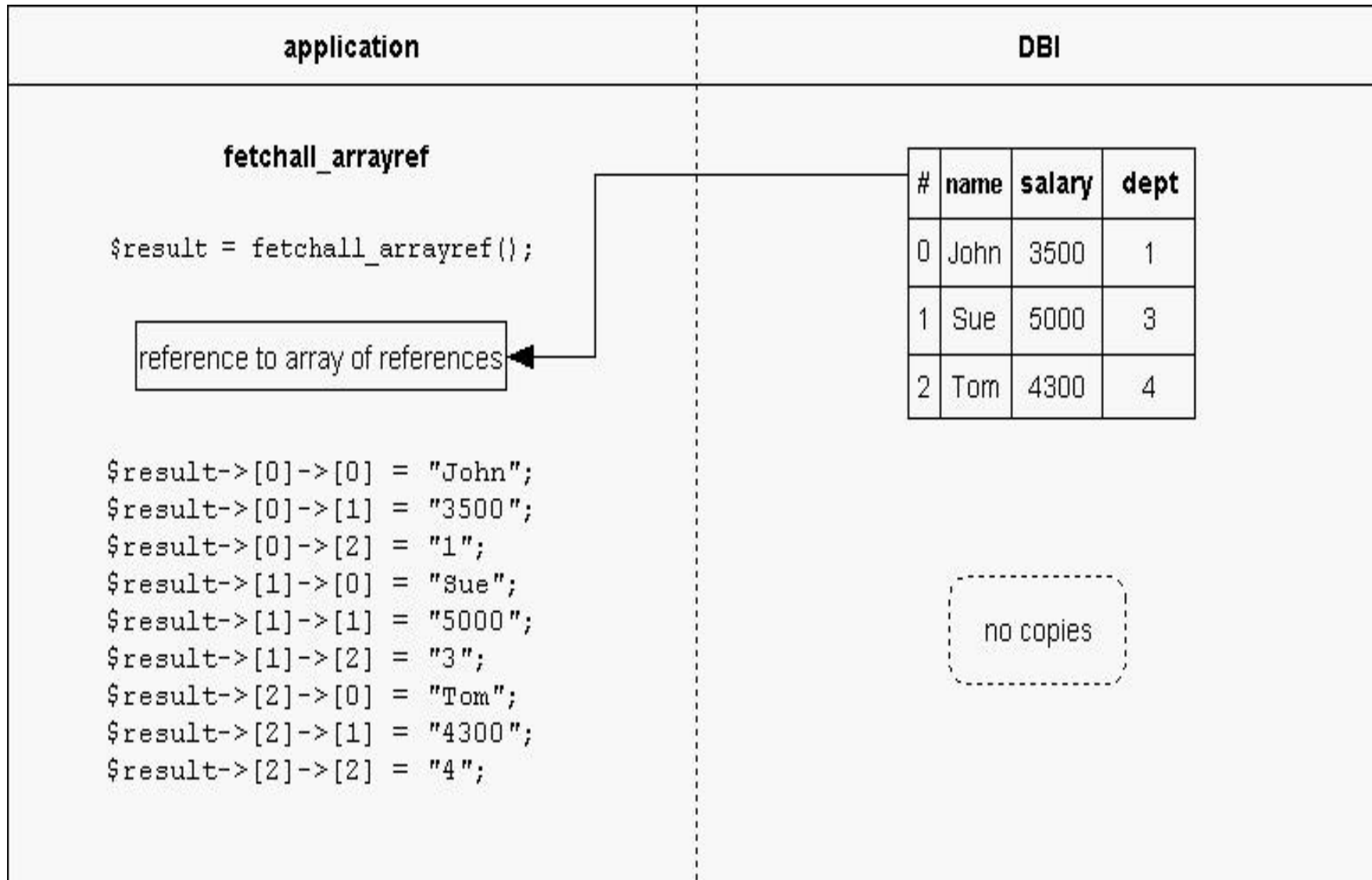


DBI results (3)





DBI results (4)





CAVEAT

- **Since this presentation is focused on idioms, I have omitted error checking code from most of the examples.**
- **Error checking with the DBI can be performed in various ways, either testing the return value or each function call, or using eval.**
- **General purpose error checking can be implemented with the HandleError function.**
- **See the DBI docs for details.**



Connection (1)

- Showing connection parameters

```
my $dbh = DBI->connect(  
    "dbi:mysql:dbname",  
    "gmax",           # username  
    "mysecret",      # password  
    {RaiseError => 1} # options  
) or die "can't connect\n";
```



Connection (2)

- Hiding connection parameters

```
my $dbh = DBI->connect(  
    "dbi:mysql:dbname"  
    . ";mysql_read_default_file="  
    . "$ENV{HOME}/.my.cnf",  
    undef,          # username  
    undef,          # password  
    {RaiseError => 1} # options  
) or die "can't connect\n";
```



Connection (3)

- Hiding connection parameters ...

```
$ ls -l ~/.my.cnf
```

```
-rw----- 1 gmax gmax 39 [...] .my.cnf
```

```
$ cat .my.cnf
```

```
[client]
```

```
user=gmax
```

```
password=mysecret
```



Connection (4)

- Using more parameters

```
my $dbh = DBI->connect(  
    "dbi:mysql:dbname"  
    . ";mysql_read_default_file="  
    . "$ENV{HOME}/.my.cnf"  
    . ";mysql_read_default_group=myapp",  
    undef,          # username  
    undef,          # password  
    {RaiseError => 1} # options  
    ) or die "can't connect\n";
```



Connection (5)

- Using more parameters

```
$ cat .my.cnf  
[client]  
user=gmax  
password=mysecret
```

```
[myapp]  
host=192.168.2.33  
user=gmax33  
password=myother
```



Insertion (1)

- **Creating a simple insertion query**

```
my @columns = qw(id name surname gender salary
dept_id);
```

```
my $insert_query =
    "INSERT INTO employees ("
    . join(", ", @columns)
    . ") VALUES ("
    . join(", ", map {"?"} @columns )
    . ")";
```

```
INSERT INTO employees (id, name, surname,
gender, salary, dept_id)
VALUES (?, ?, ?, ?, ?, ?)
```



Insertion (1a)

- Inserting a list of lists

```
my @values = (  
  [ '1', 'Mary', 'Smith', 'M', '3500', '1' ],  
  [ '2', 'Susan', 'Doe', 'F', '4500', '2' ],  
  [ '3', 'Jerry', 'Murgo', 'M', '4000', '1' ],  
  [ '4', 'Carol', 'Jones', 'F', '4000', '2' ],  
  [ '5', 'Joe', 'Smith', 'M', '3800', '2' ],  
  [ '6', 'Fred', 'O\'Hara', 'M', '3900', '1' ] );
```

```
my $sth = $dbh->prepare($insert_query);  
$sth->execute(@$_) for @values;
```



Insertion (2)

- **Creating a multiple insertion query**

```
my @values = ();  
while (<DATA>) {  
    chomp;  
    push @values, [ split /\s+/, $_ ]  
}
```

___DATA___

```
1 Mary Smith M 3500 1  
2 Susan Doe F 4500 2  
3 Jerry Murgio M 4000 1  
4 Carol Jones F 4000 2  
5 Joe Smith M 3800 2  
6 Fred O'Hara M 3900 1
```



Insertion (3a)

- **Creating a multiple insertion query**

```
$insert_query =  
    "INSERT INTO employees ("  
    . join(", ", @columns)  
    . ") VALUES\n";  
my $start=0;  
for (@values) {  
    $query .= ',' if $start++;  
    $query .= '('  
        . (join (",", map { $dbh->quote($_)} @$_))  
        . ')';  
}
```



Insertion (3b)

- **Creating a multiple insertion query**

```
my $insert_query =  
    "INSERT INTO employees ("  
    . join(", ", @columns)  
    . ") VALUES\n"  
    . join(", \n",  
        map { "(" .  
            join ( ", ",  
                map { $dbh->quote($_) } @$_  
            ) . ")"  
        } @values  
    );
```



Insertion (4)

- **Creating a multiple insertion query**

```
INSERT INTO employees (id, name, surname,
gender, salary, dept_id) VALUES
('1', 'Mary', 'Smith', 'M', '3500', '1'),
('2', 'Susan', 'Doe', 'F', '4500', '2'),
('3', 'Jerry', 'Murgo', 'M', '4000', '1'),
('4', 'Carol', 'Jones', 'F', '4000', '2'),
('5', 'Joe', 'Smith', 'M', '3800', '2'),
('6', 'Fred', 'O\'Hara', 'M', '3900', '1')
```



Insertion (5)

- Inserting a list of hashes

```
my @values = (  
  { id => 1, name => 'Mary', gender => 'F' },  
  { id => 2, name => 'Susan', gender => 'F' },  
  { id => 3, name => 'Jerry', gender => 'M' },  
  { id => 4, name => 'Carol', gender => 'F' },  
  { id => 5, name => 'Joe', gender => 'M' },  
  { id => 6, name => 'Fred', gender => 'M' } );
```

```
my $sth = $dbh->prepare($insert_query);
```

```
$sth->execute( @{$_}{@columns} ) for @values;
```



Fetching records (1)

- **Fetching one record at a time**

```
while (my @row = $sth->fetchrow_array()) {  
    print $row[0], " ", $row[1] # and so on  
}
```

```
while (my $row = $sth->fetchrow_arrayref()) {  
    print $row->[0], " ", $row->[1] # and so on  
}
```

```
while (my $row = $sth->fetchrow_hashref()) {  
    print $row->{id}, " ", $row->{name}  
}
```



Fetching records (2)

- Fetching a list of lists

```
my $rows = $sth->fetchall_arrayref();  
for (@$rows) {  
    print $_->[0], " ", $_->[1]; # and so on  
}
```



Fetching records (3)

- **Creating an array from a column (1)**

```
my $query =  
    "select id, name, salary from employees";  
  
my @names = map {$_->[1]}  
    @{$dbh->selectall_arrayref($query)};  
__END__  
@names = (qw(Fred Joshua Kim Dave Sal));
```



Fetching records (4)

- **Creating an array from a column (2)**

```
my $query =  
    "select name, salary from employees";
```

```
my @names =  
    @{$dbh->selectcol_arrayref($query)};  
__END__  
@names = (qw(Fred Joshua Kim Dave Sal));
```



Fetching records (5)

- Fetching a list of hashes (1)

```
my $query =
    "select id, name, gender from employees";
my $sth = $dbh->prepare($query);
$sth->execute();
my $employees_loh = $sth->fetchall_arrayref({});
__END__
$employees_loh = [
    { id => 1, name => 'Mary', gender => 'F' },
    { id => 2, name => 'Susan', gender => 'F' },
    { id => 3, name => 'Jerry', gender => 'M' },
    { id => 4, name => 'Carol', gender => 'F' },
    { id => 5, name => 'Joe', gender => 'M' },
    { id => 6, name => 'Fred', gender => 'M' }];
```



Fetching records (6)

- **Fetching a list of partial hashes (2)**

```
$sth->execute();  
my $employees_loh =  
    $sth->fetchall_arrayref({id =>1, gender => 1});
```

___END___

```
$employees_loh = [  
    { id => 1, gender => 'F' },  
    { id => 2, gender => 'F' },  
    { id => 3, gender => 'M' },  
    { id => 4, gender => 'F' },  
    { id => 5, gender => 'M' },  
    { id => 6, gender => 'M' }];
```



Fetching records (7)

- Fetching into a hash (1)

```
my %employees_h =  
    map { $_->[1], $_->[2] }  
        @{$dbh->selectall_arrayref($query)};  
__END__  
%employees_h = (  
    'Mary'    => 'F',  
    'Susan'  => 'F',  
    'Jerry'  => 'M',  
    'Carol'  => 'F',  
    'Joe'    => 'M',  
    'Fred'   => 'M'  
);
```



Fetching records (8)

• Fetching into a hash (2)

```
my %employees_h =  
    map { $_->[0], [ $_->[1], $_->[2] ] }  
        @{$dbh->selectall_arrayref($query)};  
__END__  
%employees_h = (  
    1 => [ 'Mary', 'F' ],  
    2 => [ 'Susan', 'F' ],  
    3 => [ 'Jerry', 'M' ],  
    4 => [ 'Carol', 'F' ],  
    5 => [ 'Joe', 'M' ],  
    6 => [ 'Fred', 'M' ]  
);
```



Fetching records (9)

- **Fetching into a hash (3)**

```
my %employees_h =  
    map { shift @$_ , [ @$_ ]  
        @{$dbh->selectall_arrayref($query) } ;  
__END__  
%employees_h = (  
    1 => [ 'Mary' , 'F' ],  
    2 => [ 'Susan' , 'F' ],  
    3 => [ 'Jerry' , 'M' ],  
    4 => [ 'Carol' , 'F' ],  
    5 => [ 'Joe' , 'M' ],  
    6 => [ 'Fred' , 'M' ]  
);
```



Fetching records (10)

- **Fetching into a hash of hashes**

```
my $employees_h =  
    $dbh->selectall_hashref($query, 1);  
__END__  
$employees_h = {  
    1 => { id => 1, name => 'Mary', gender => 'F' },  
    2 => { id => 2, name => 'Susan', gender => 'F' },  
    3 => { id => 3, name => 'Jerry', gender => 'M' },  
    4 => { id => 4, name => 'Carol', gender => 'F' },  
    5 => { id => 5, name => 'Joe', gender => 'M' },  
    6 => { id => 6, name => 'Fred', gender => 'M' }  
};
```



Fetching records (11)

- **Fetching into a hash of hashes (pitfall)**

```
my $employees_h =
    $dbh->selectall_hashref($query, 3);
__END__
$employees_h = {
    'F' => {id => 4, name => 'Carol', gender => 'F'},
    'M' => {id => 6, name => 'Fred', gender => 'M'}
};
```

If the chosen key is not unique, the resulting hash will **lose records** (multiple records will overwrite the previous value for each key)



Fetching records (12)

- **Fetching hashes without speed penalty**

```
$sth->execute;  
my %rec = ();
```

```
$sth->bind_columns(  
    map { \$_rec{$_} } @{$sth->{NAME}});
```

```
print "$rec{id}\t",  
      "$rec{name}\t",  
      "$rec{gender}\n"  
while $sth->fetchrow_arrayref;
```



Special features (1)

- **Copying from a DBMS to another**

```
my $dbh1=DBI->connect( 'dbi:mysql:db1;host=host1' );  
my $dbh2=DBI->connect( 'dbi:mysql:db2;host=host2' );
```

```
my $query1 = "SELECT col1,col2,col3 FROM table1";  
my $recs = $dbh1->selectall_arrayref($query1);
```

```
my $insert = "INSERT INTO table1 (col1,col2,col3)  
VALUES (?, ?, ?)";  
my $sth2 = $dbh2->prepare($insert);  
for (@$recs) {  
    $sth2->execute(@$_);  
}
```



Special features (2)

- **Getting all data from a DBMS in SQL (1)**

```
my $dbh=DBI->connect('dbi:mysql:db1;host=host1');

for my $db (@{$dbh->selectcol_arrayref('show
databases')}) {
    for my $table (@{$dbh->selectcol_arrayref(
        "show tables from $db")}) {
my $sth = $dbh->prepare(
    "show create table $db.$table");
$sth->execute();
my ($null, $creation) = $sth->fetchall_array();
$sth->finish();
# ...
}
}
```



Special features (2)

- Getting all data from a DBMS in SQL (2)

```
for my $table (@{$dbh->selectcol_arrayref(
    "show tables from $db"}) {
    # ...
    my $query = "select * from $db.$table";
    my $recs = $dbh->selectall_arrayref($query);
    for(@$recs) {
        print "insert into $db.$table VALUES ("
            . (join ", ", map {$dbh->quote($_)} @$_)
            . " );\n";
    }
}
```



Further reading

- **A full article about DBI idioms**

<http://gmax.oltrelinux.com/dbirecipes.html>

- **An article about DBI efficiency**

http://www.perlmonks.org/index.pl?node_id=273952

Thanks for your attention

Any questions?



g.maxia@stardata.it