

Giuseppe Maxia
a.k.a. **gmax**



**idiomi del
Perl/DBI**

<http://www.stardata.it>

Italian Perl Workshop
Pisa 19- 20 giugno 2004



idiomi del Perl / DBI



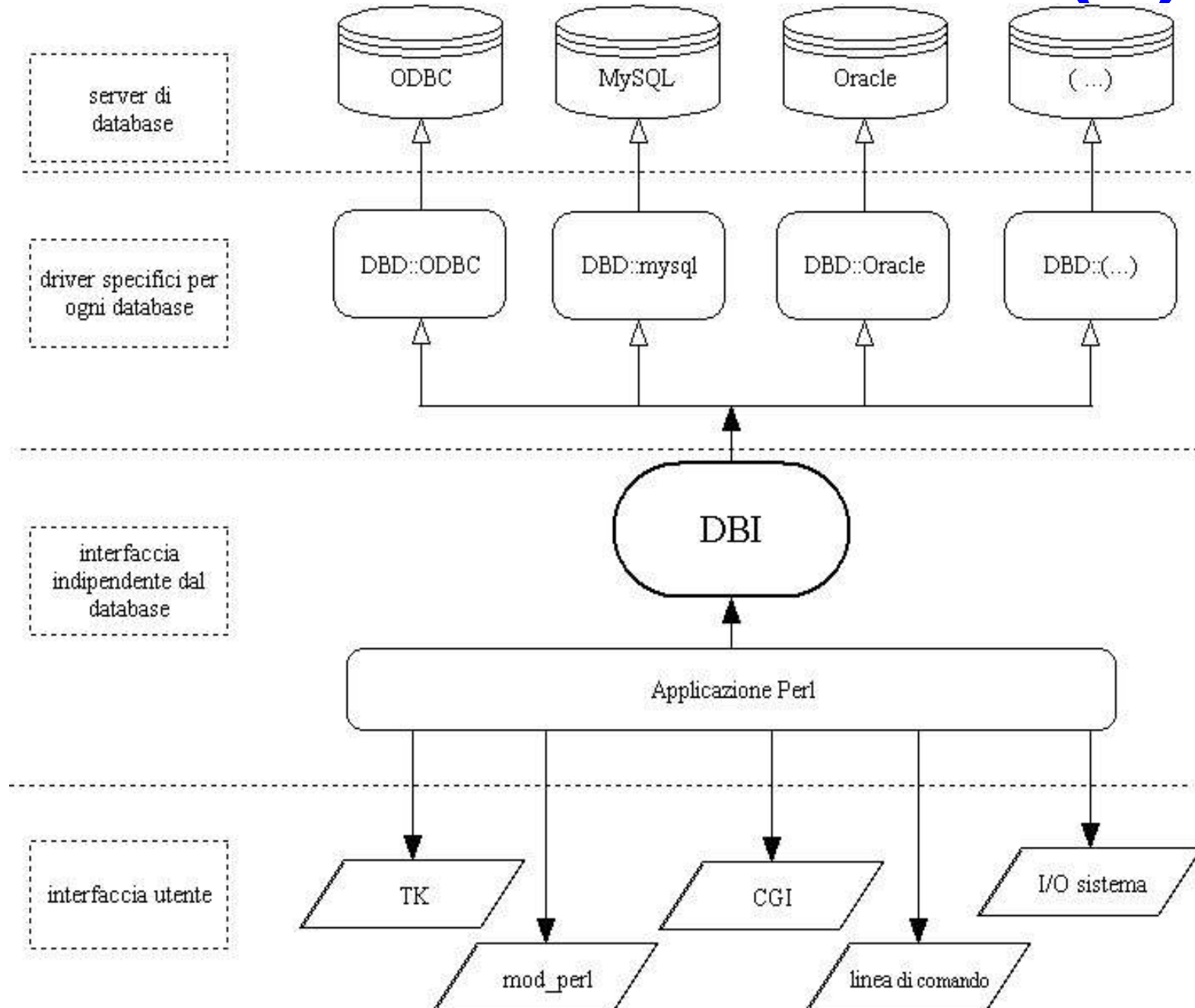
Giuseppe Maxia
g.maxia@stardata.it



Agenda

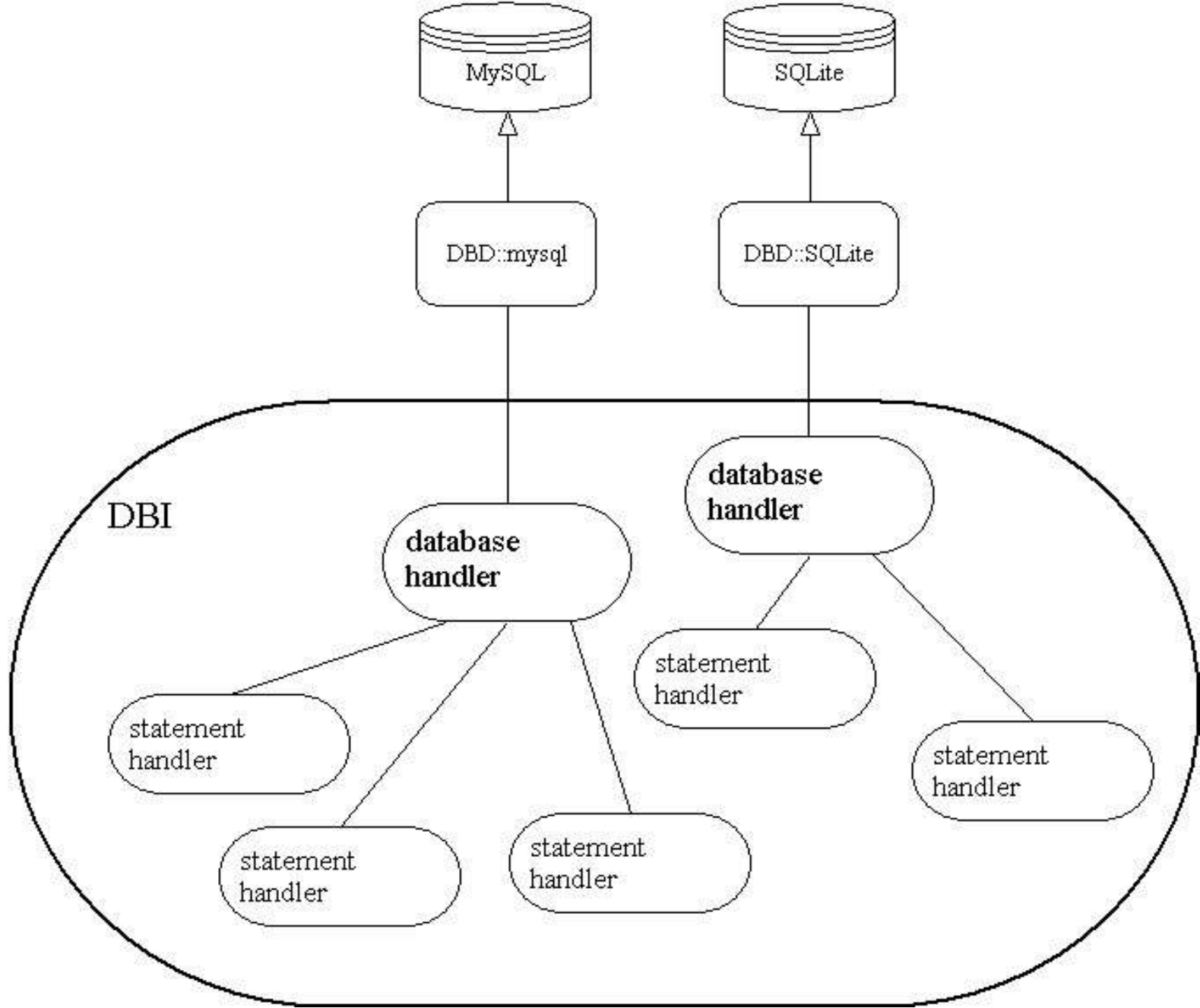
- **Richiami di architettura del DBI**
- **Idiomi di connessione**
- **inserimenti semplici**
- **Inserimenti multipli**
- **Inserire liste**
- **Leggere dati complessi**
- **peculiarità delle letture multiple**
- **Brevità**

architettura del DBI (1)



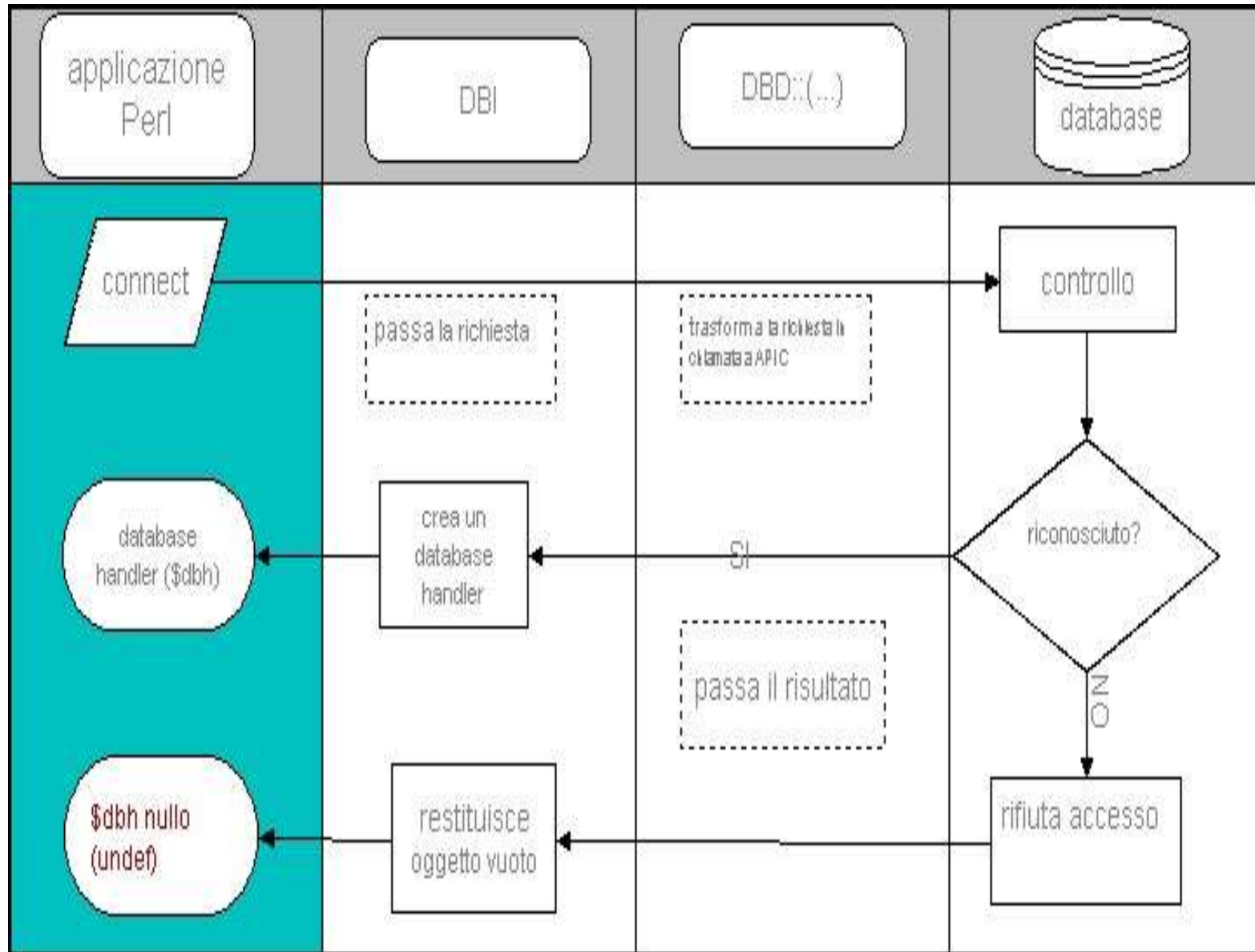


architettura DBI (2)



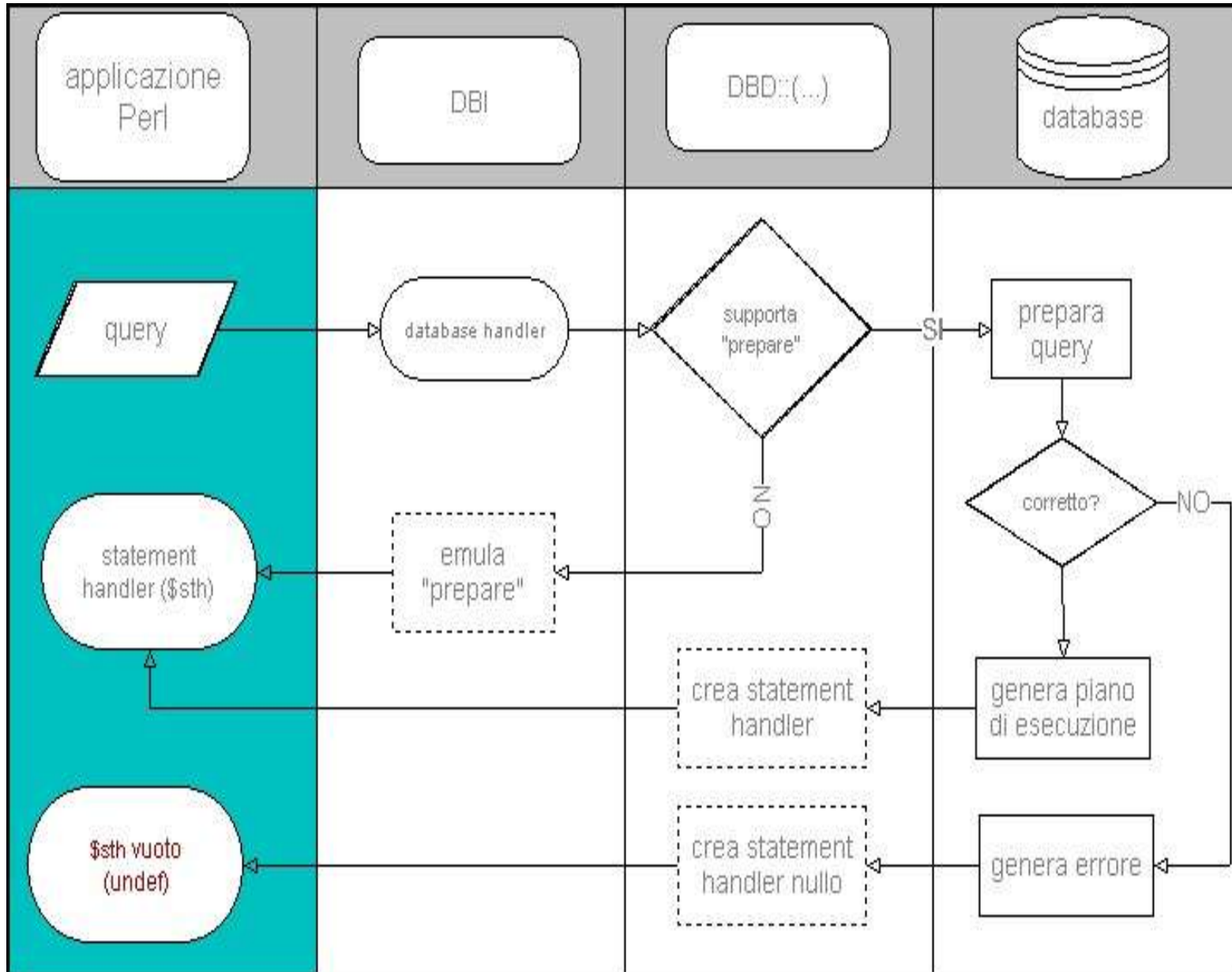
operazioni del DBI

connessione(1)



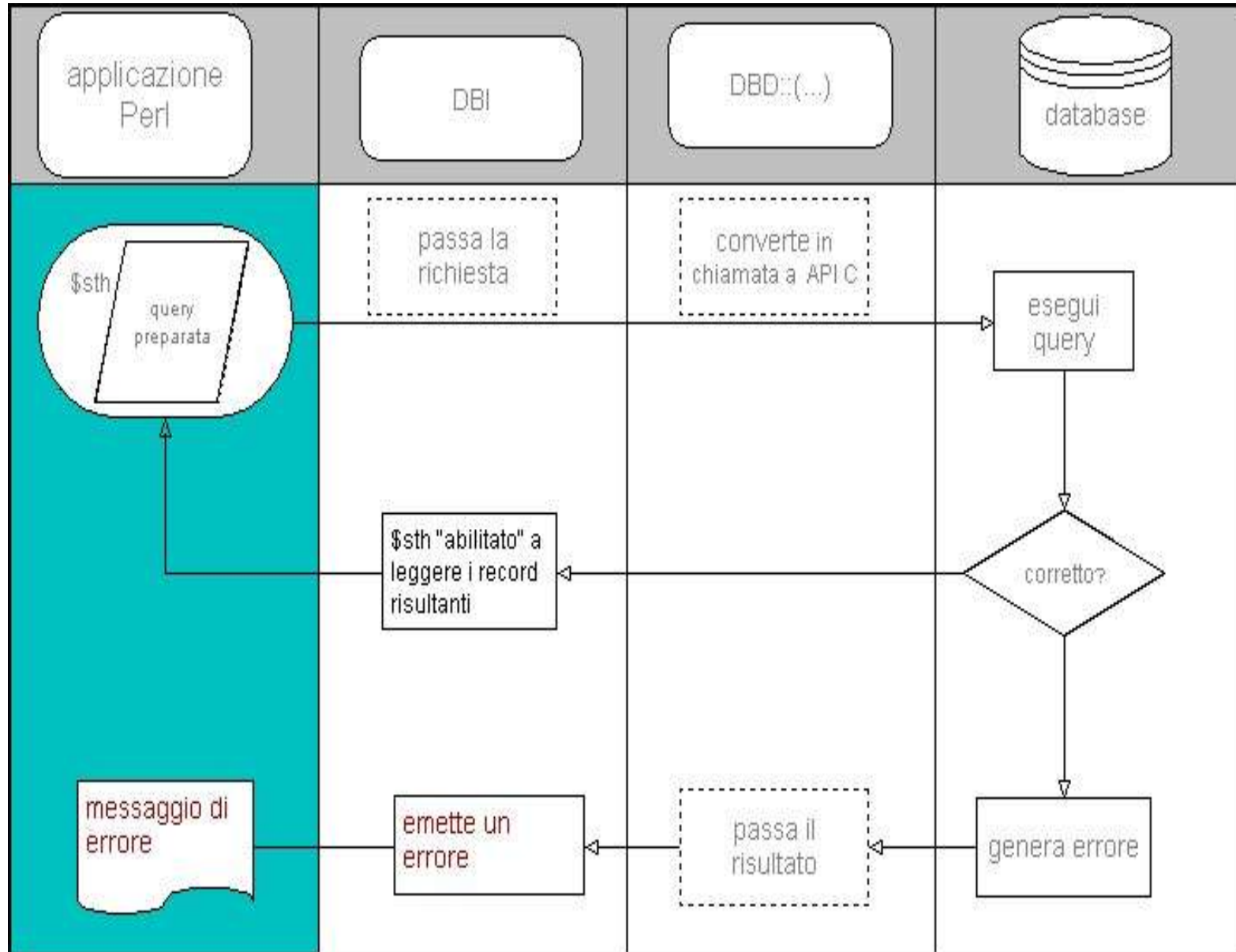
operazioni del DBI

preparare una query(2)



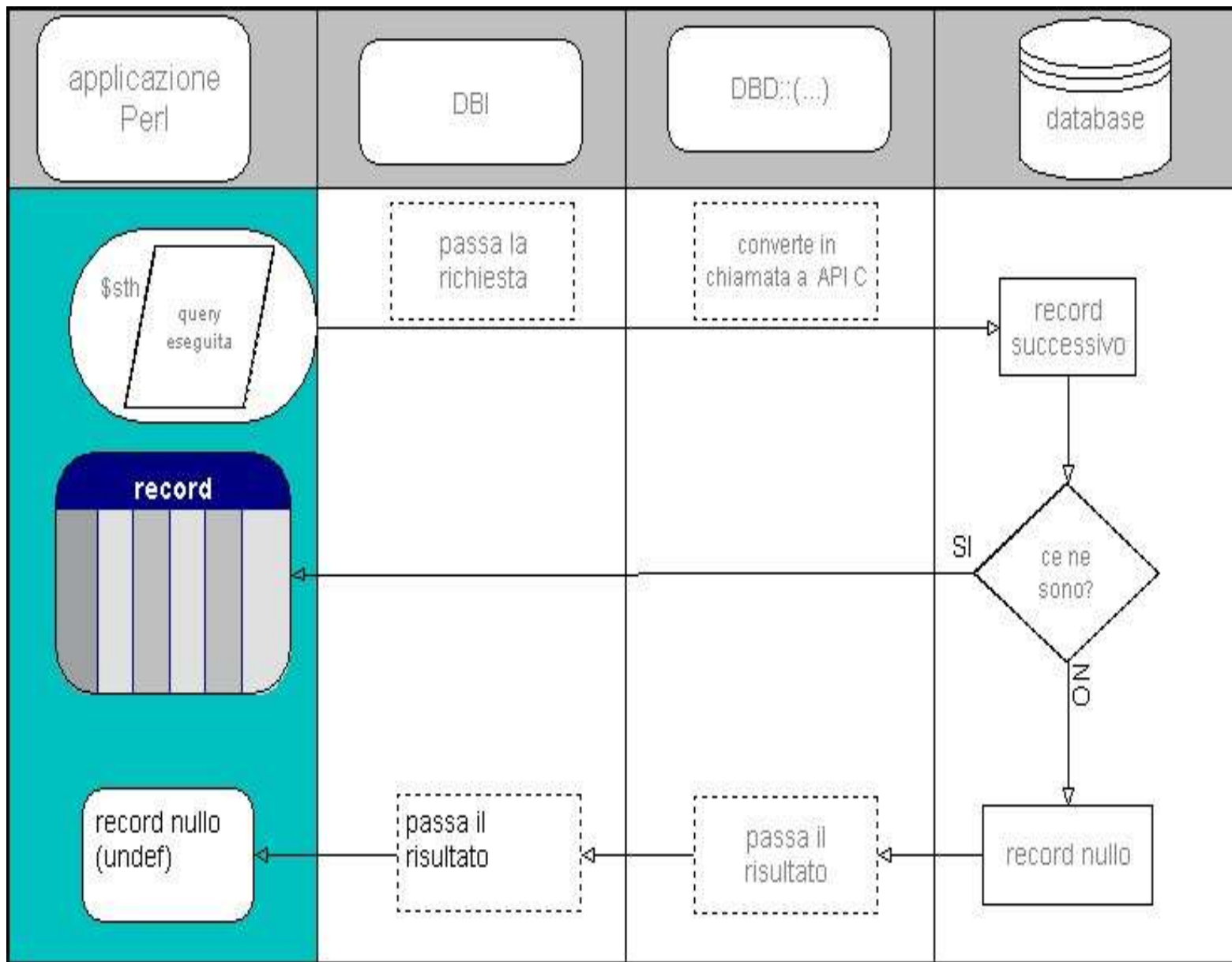
operazioni del DBI

eseguire una query(3)

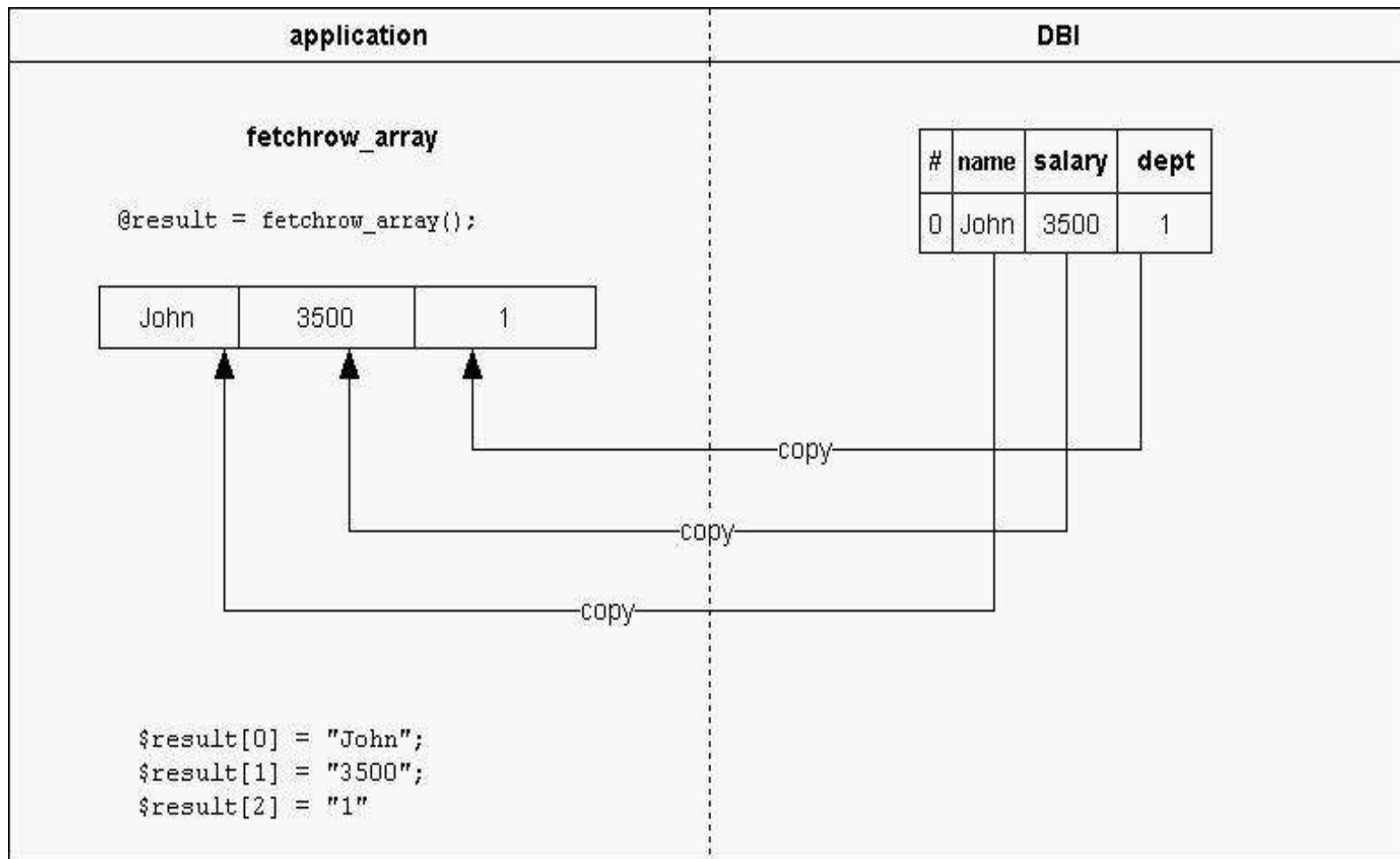


operazioni del DBI

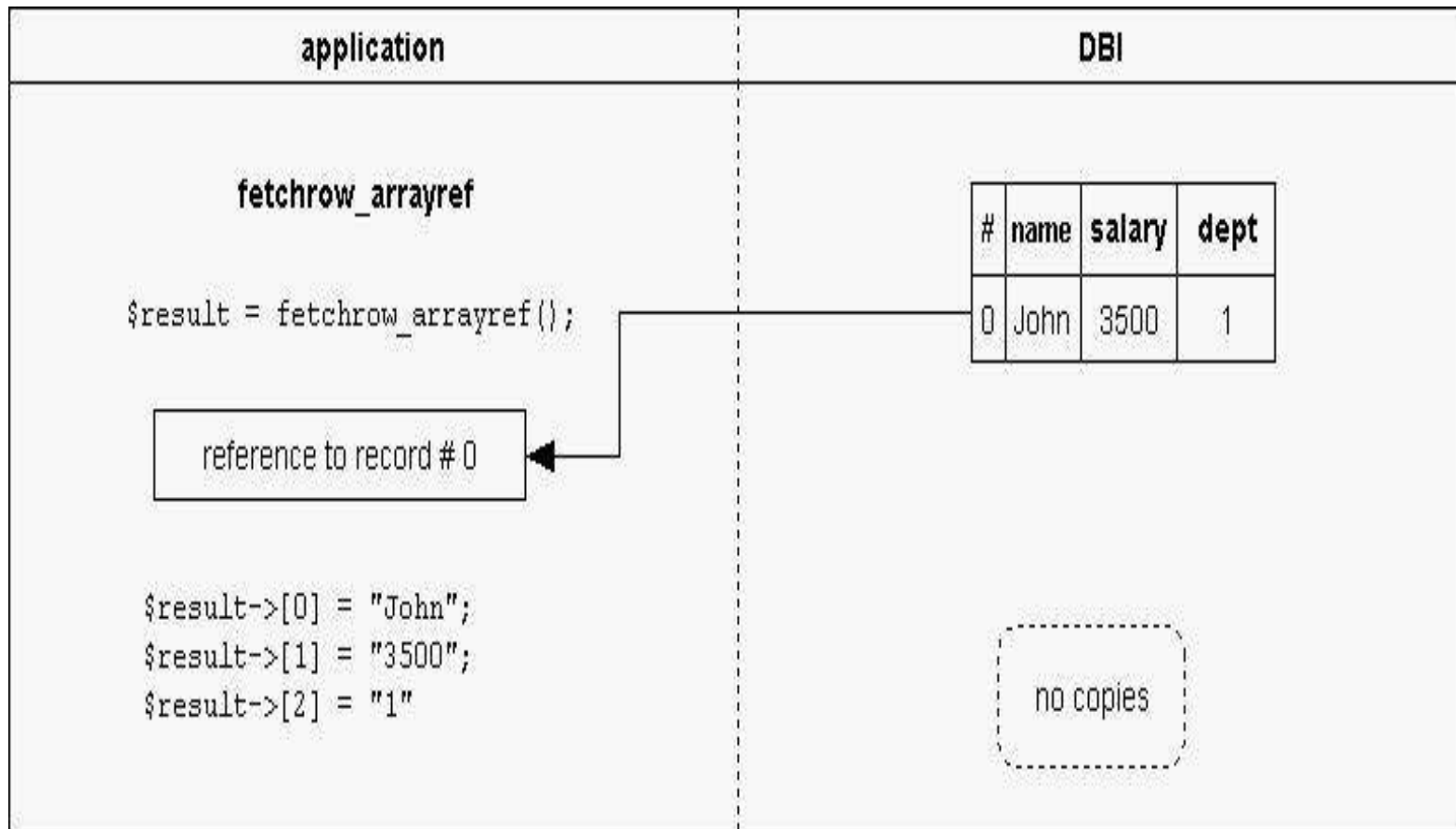
lettura record (4)



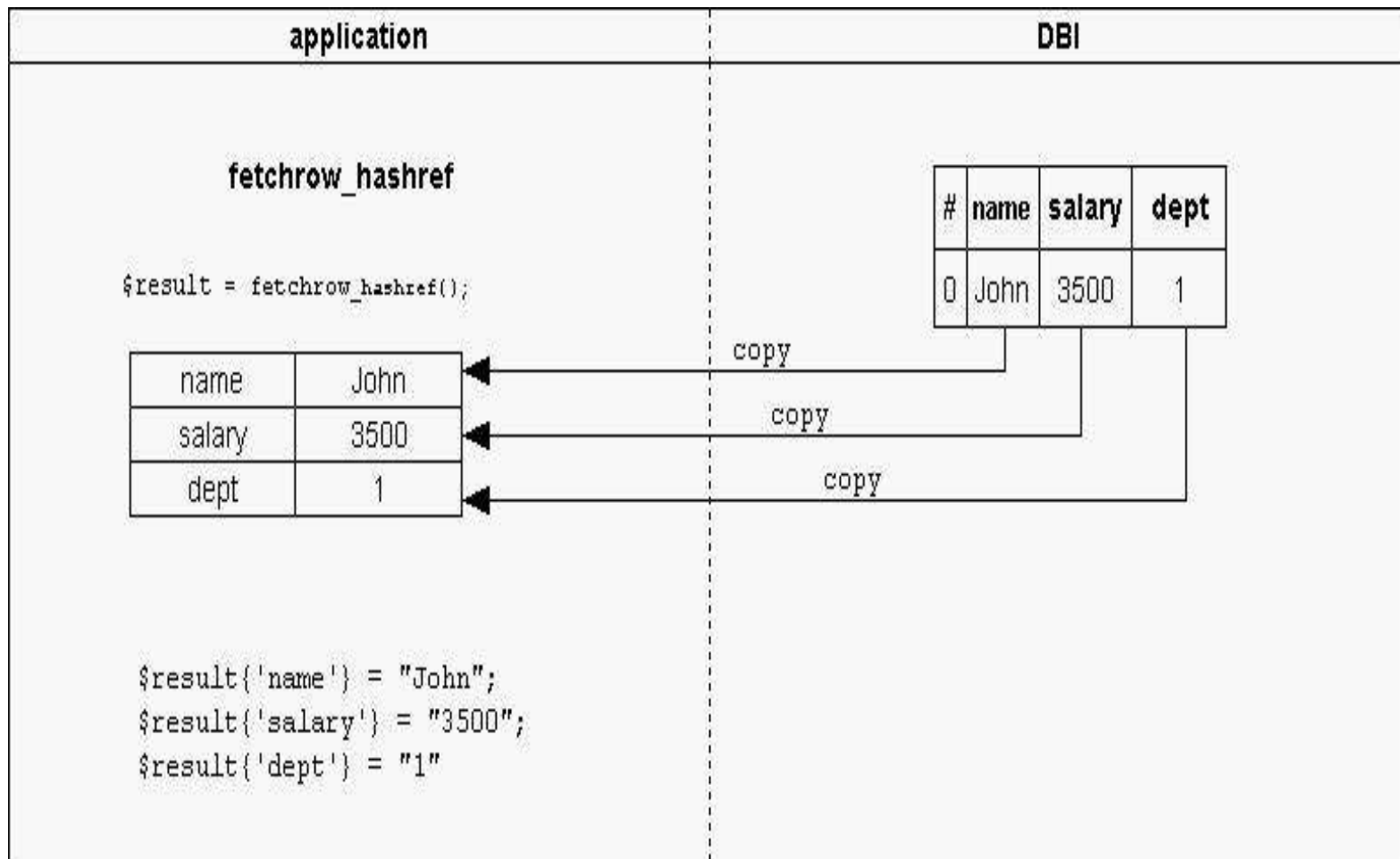
risultati DBI (1)



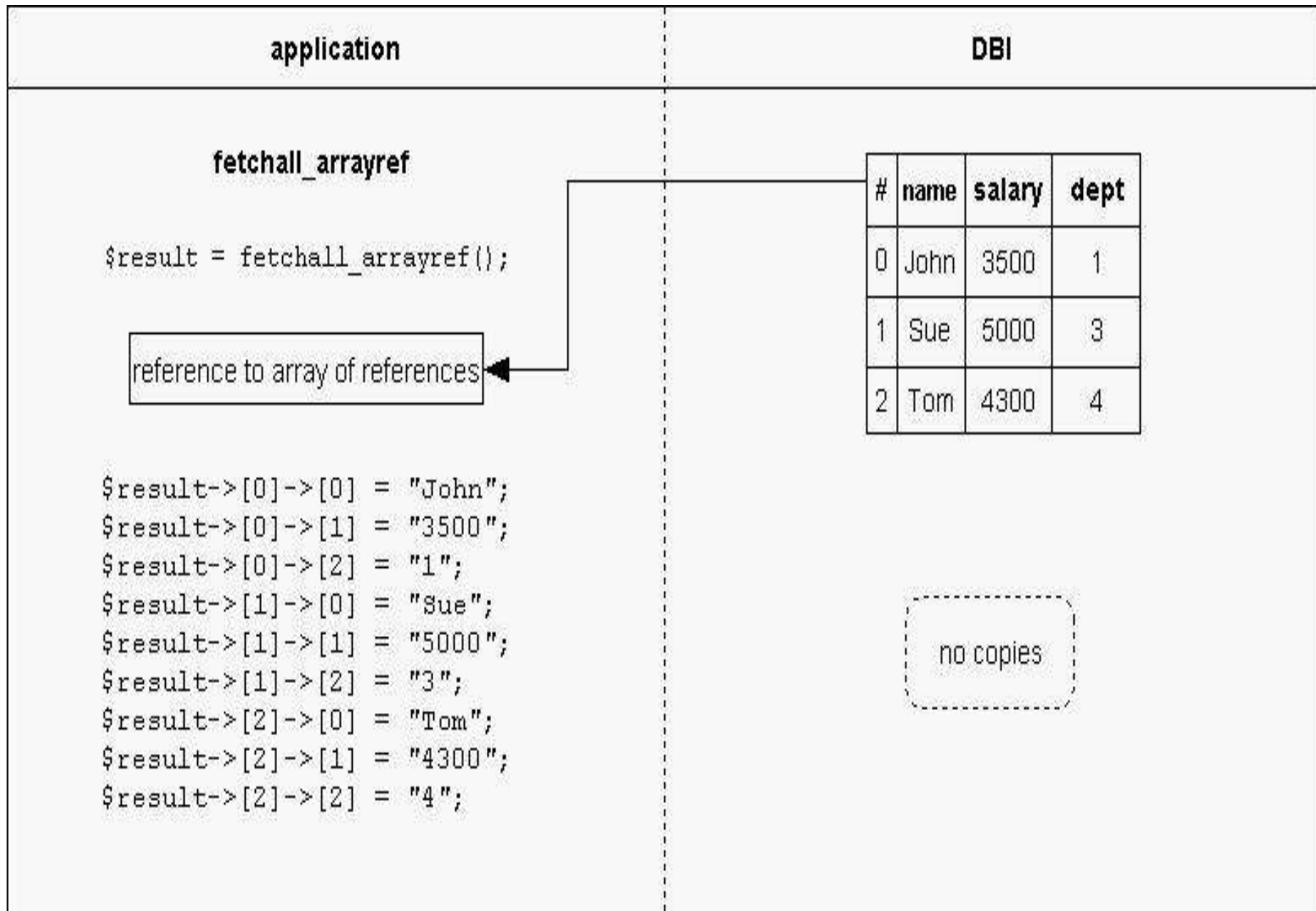
risultati DBI (2)



risultati DBI (3)



risultati DBI (4)





AVVERTENZA

- **Poiché questa presentazione è dedicata agli idiomi, ho omesso molti controlli degli errori.**
- **Con il DBI, gli errori si possono controllare verificando il valore restituito dai metodi o racchiudendoli in un blocco eval.**
- **La funzione HandleError consente di implementare controlli personalizzati**
- **Vedere la documentazione del DBI per dettagli.**



Connessione (1)

Mostrare i parametri di connessione

```
my $dbh = DBI->connect(  
    "dbi:mysql:dbname",  
    "gmax",           # utente  
    "mysecret",      # password  
    {RaiseError => 1} # opzioni  
    ) or die "non posso connettermi\n";
```



Connessione (2)

nascondere i parametri di connessione (idioma del DBD::mysql)

```
my $dbh = DBI->connect(  
    "dbi:mysql:dbname"  
    . ";mysql_read_default_file="  
    . "$ENV{HOME}/.my.cnf",  
    undef,          # utente  
    undef,          # password  
    {RaiseError => 1} # opzioni  
    ) or die "non posso connettermi\n";
```



Connessione (3)

nascondere i parametri di connessione ...

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

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

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

```
[client]
```

```
user=gmax
```

```
password=mysecret
```



Connessione (4)

Più parametri

```
my $dbh = DBI->connect(
    "dbi:mysql:dbname"
    . ";mysql_read_default_file="
    . "$ENV{HOME}/.my.cnf"
    . ";mysql_read_default_group=myapp",
    undef,          # utente
    undef,          # password
    {RaiseError => 1} # opzioni
) or die "non posso connettermi\n";
```



Connessione (5)

Più parametri

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

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



Inserimenti (1)

Creare una query di inserimento

```
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 (?, ?, ?, ?, ?, ?)
```



Inserimenti (1 a)

- **Inserire una lista di liste**

```
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;
```



Inserimenti (2)

Creare una query con inserimenti multipli (MySQL e DB2)

```
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
```



Inserimenti (3a)

Creare una query con inserimenti multipli (MySQL e DB2)

```

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

```



Inserimenti (3b)

Creare una query con inserimenti multipli (MySQL e DB2)

```

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

```



Inserimenti (4)

Creare una query con inserimenti multipli (MySQL e DB2)

```
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')
```



Inserimenti (5)

- **Inserire una lista di hash**

```
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;
```



Lettura record (1)

- Leggere un record alla volta

```
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}  
}
```



Letture record (2)

- leggere una lista di liste

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



lettura record (3)

Creare un array da una colonna (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));
```



Lettura record (4)

- Creare un array da una colonna (2)

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



Lettura record (5)

Leggere una lista di hash (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' }];
```



Lettura record (6)

- leggere una lista di hash parziali (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' }];
```



Lettura record (7)

Leggere un 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'
);

```



Leggere record (8)

Leggere un 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' ]  
);
```



Leggere record (9)

leggere un 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' ]
);
```



Lettura record (10)

- leggere un hash di hash

```

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' }
};

```



Letture record (1 1)

Tranello nella lettura di un hash di hash

```
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'}
};
```

Se la chiave scelta non è unica, l'hash risultante **perderà record** (record multipli sovrascriveranno i valori precedenti di ogni chiave)



Letture record (12)

Leggere hash senza perdere efficienza

```
$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;
```



Lecture di approfondimento

(in inglese)

- **Un articolo sugli idiomi del DBI**

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

- **Un articolo sull'efficienza del DBI**

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

Grazie per l'attenzione



Domande?

g.maxia@stardata.it