DBIx::DataModel

Classes and UML-style associations on top of DBI

(just an appetizer...)

laurent.dami@justice.ge.ch

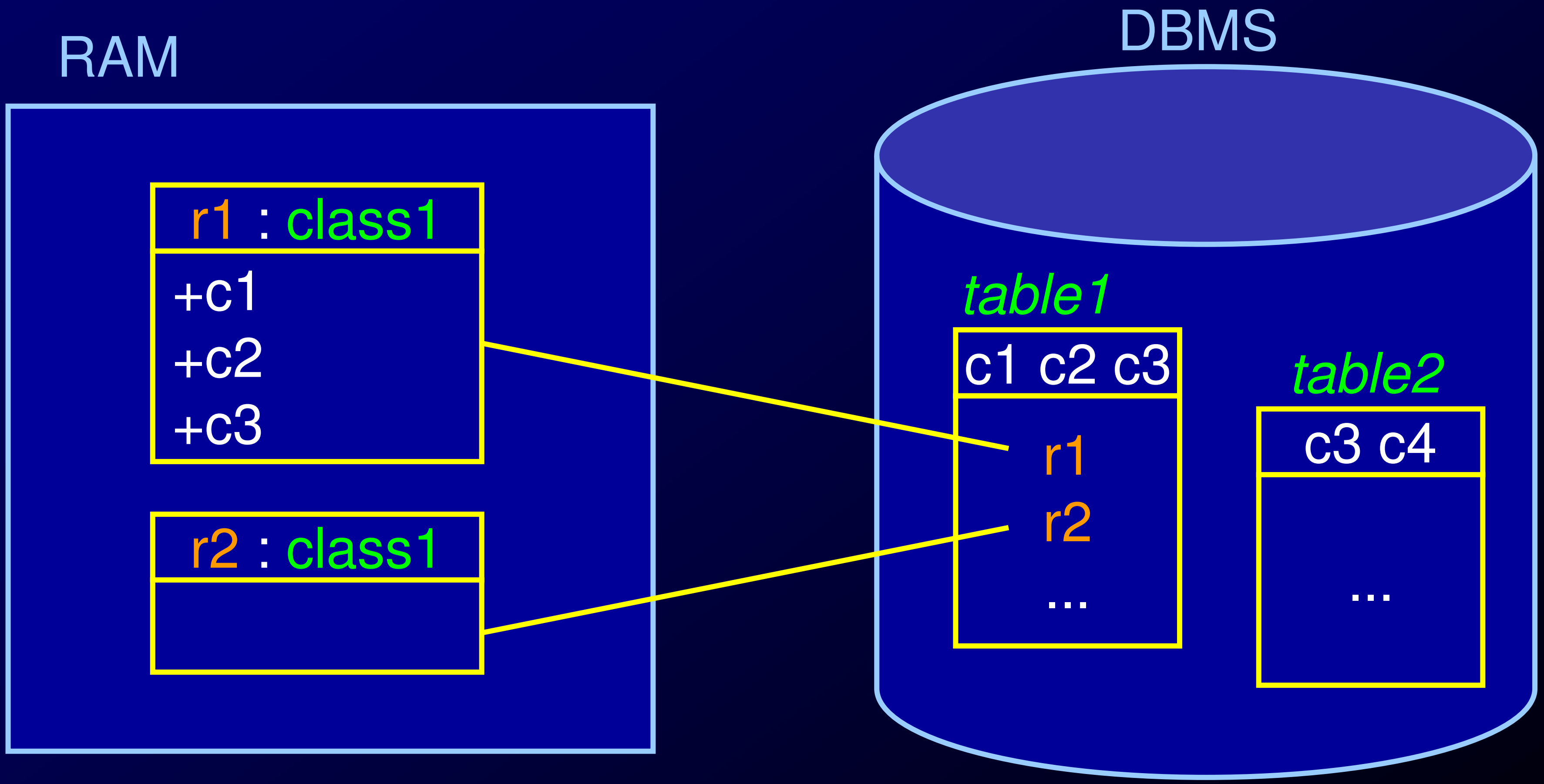
Agenda

- Introduction
  - ORMs
  - Design issues
- Unified Modelling Language (UML)
- Tables
  - Declaration
  - Usage
- Associations
  - Declaration
  - Usage

Introduction

- ORMs
- Design issues

ORM : Object-Relational Mapping



What for ?

[catalyst list] On Thu, 2006-06-08, Steve wrote:

- Not intending to start any sort of rancorous discussion,
- but I was wondering whether someone could illuminate
- me a little?

- I'm comfortable with SQL, and with DBI. I write basic
- SQL that runs just fine on all databases, or more
- complex SQL when I want to target a single database
- (ususally postgresql).

- What value does an ORM add for a user like me?

ORM useful for ...

- navigation between tables
- data conversions
- generate complex SQL queries from Perl datastructures
- expansion of tree data structures coded in the relational model
- transaction encapsulation
- data validation
- auto-filling some columns at update

➔ See Also : http://lists.rawmode.org/pipermail/catalyst/2006-June

ORMs in CPAN : TIMTOWTDI !

...

...

DBIx::SQLEngine

DBIx::RecordSet

Class::DBI

Tangram

Data::ObjectDriver

Rose::DB::Object

DBIx::Class

Alzabo

...

ORM

...

Class::PObject

SPOPS

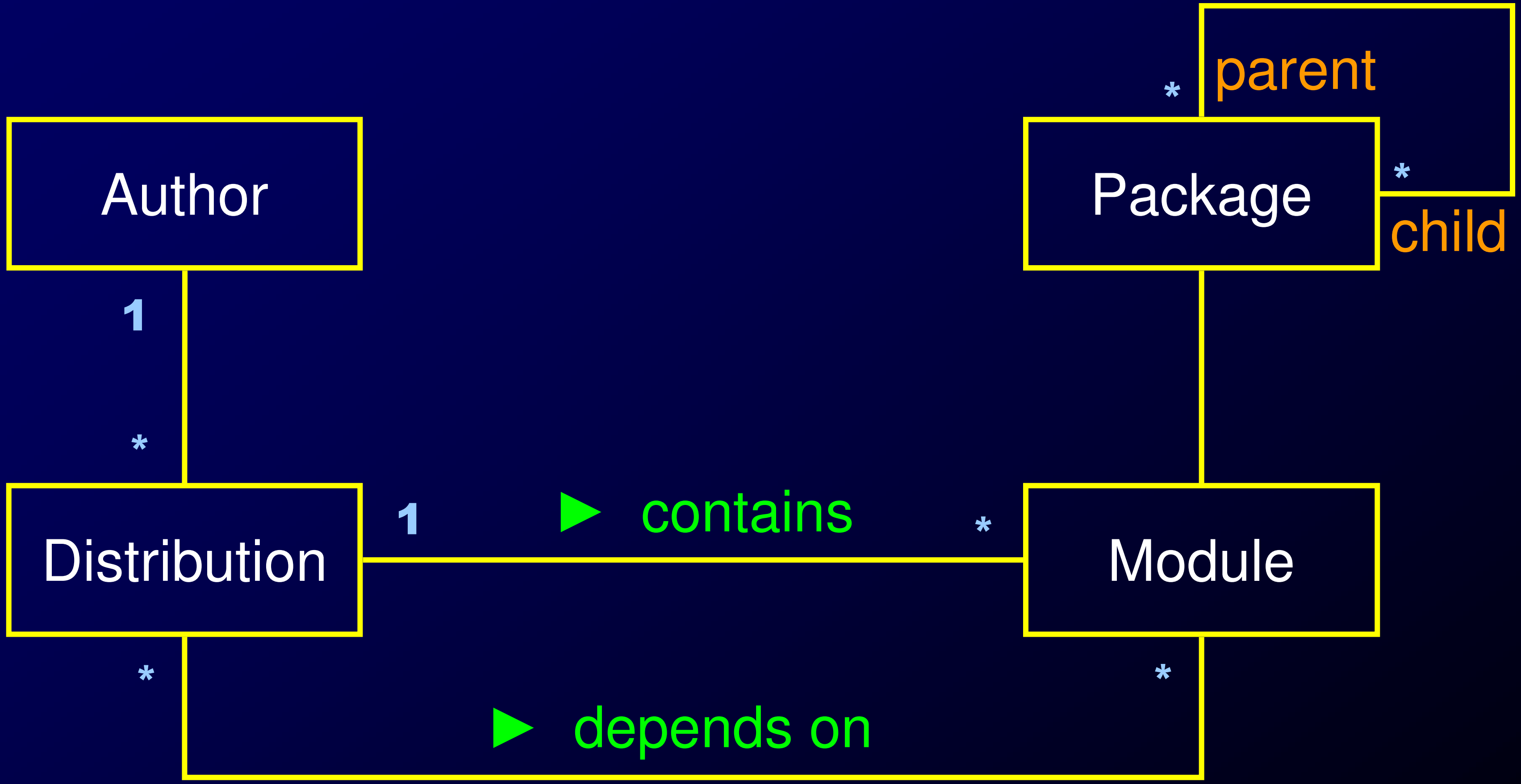
DBIx::DataModel

Some design issues

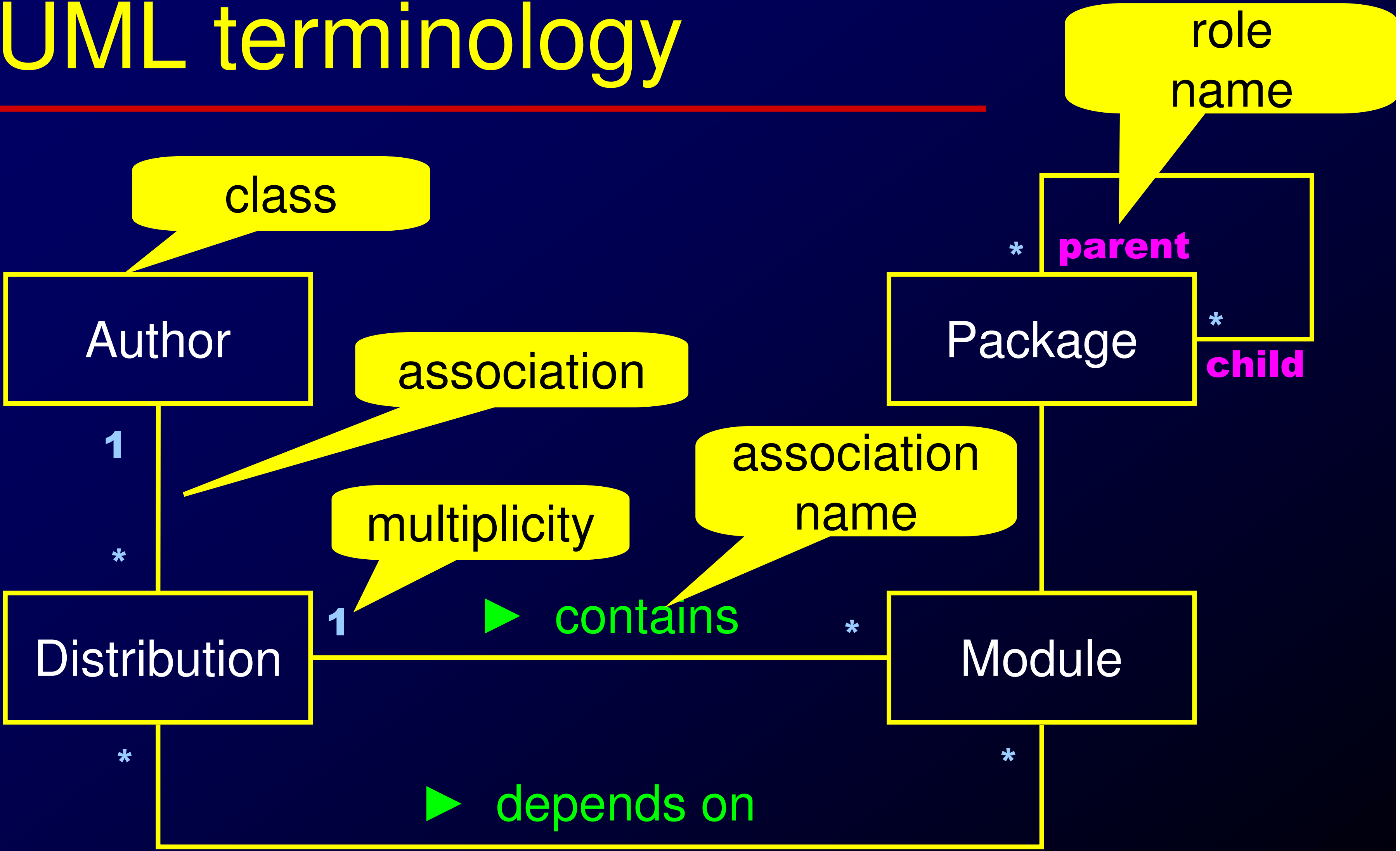
- DRY : Don't Repeat Yourself
- When to fetch column values, and which
- Pure objects versus Perl datastructures
- Encapsulation / collaboration with lower-level layers
- Fine SQL tuning
- Caching

Unified Modeling Language

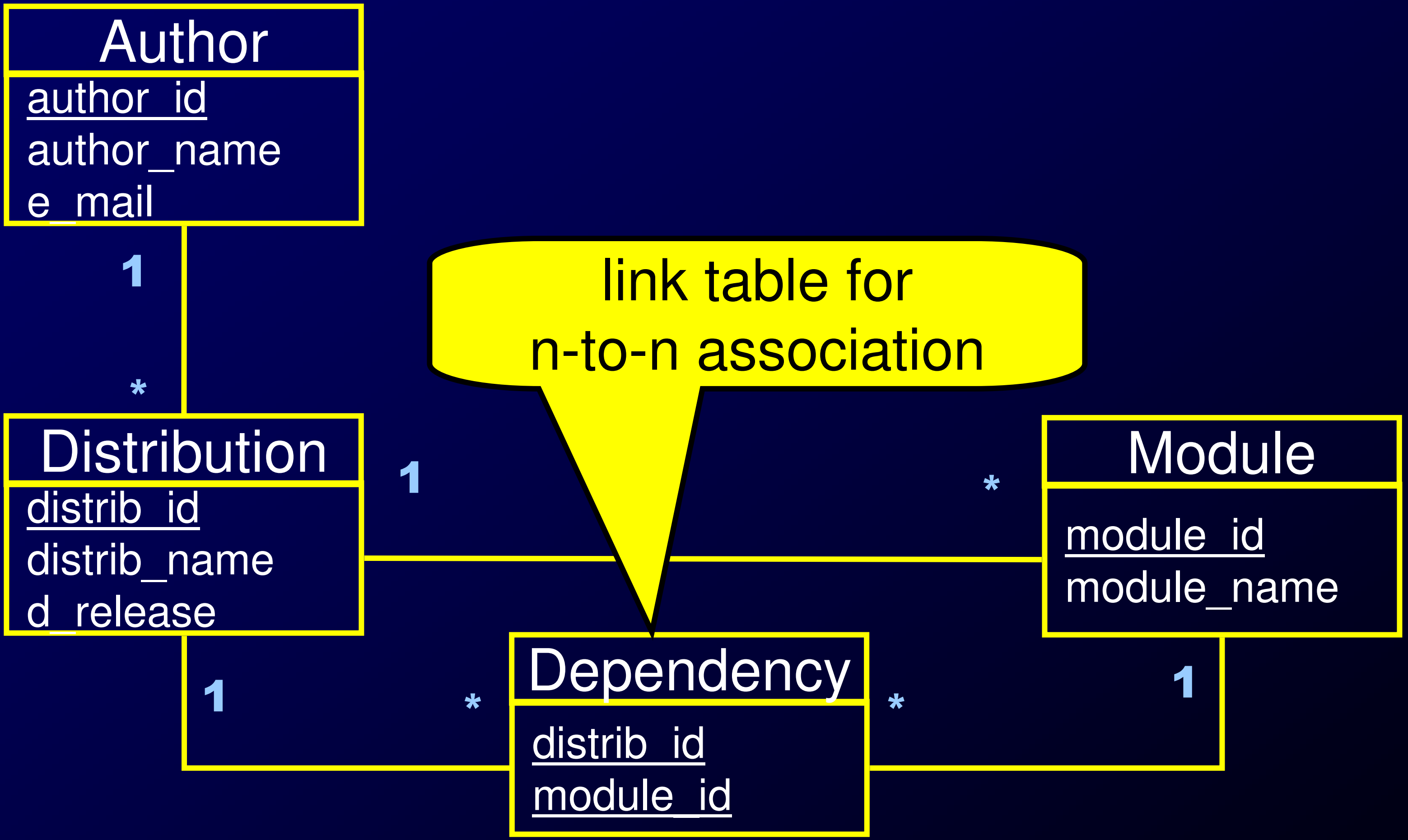
UML conceptual Model : example



UML terminology



Model implementation



Tables [and Views]

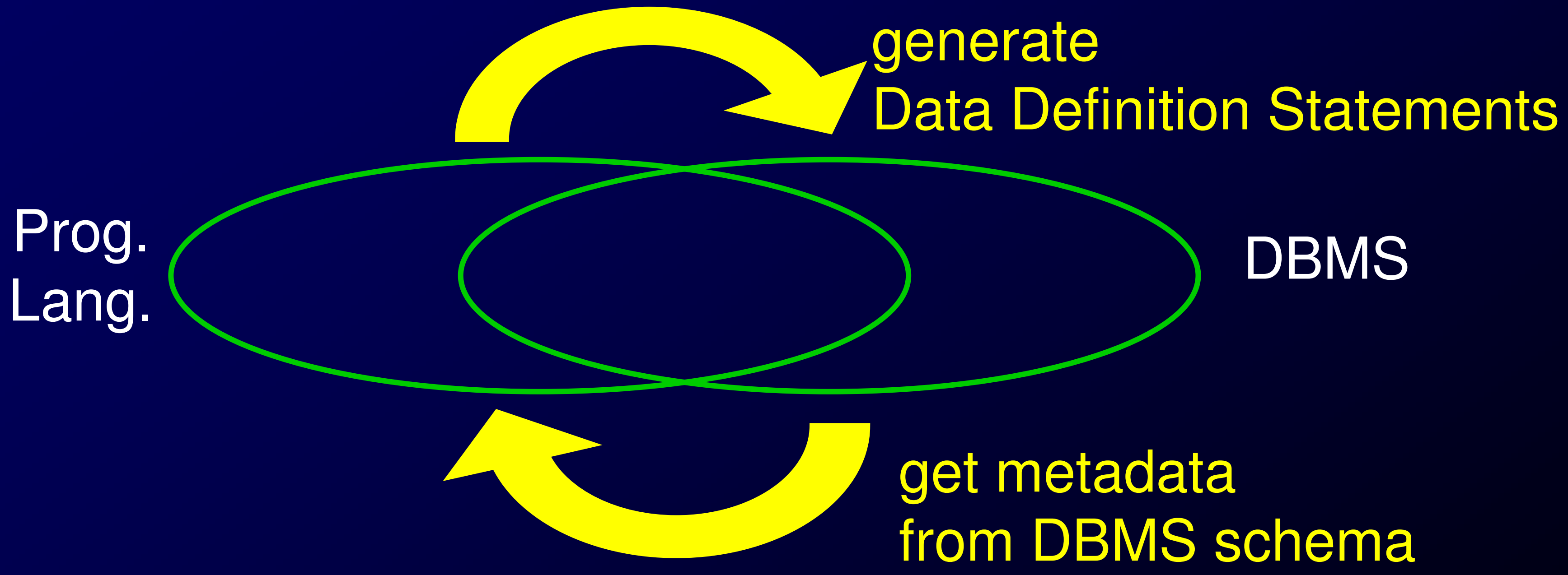
- declaration
- usage

Schema and table declarations

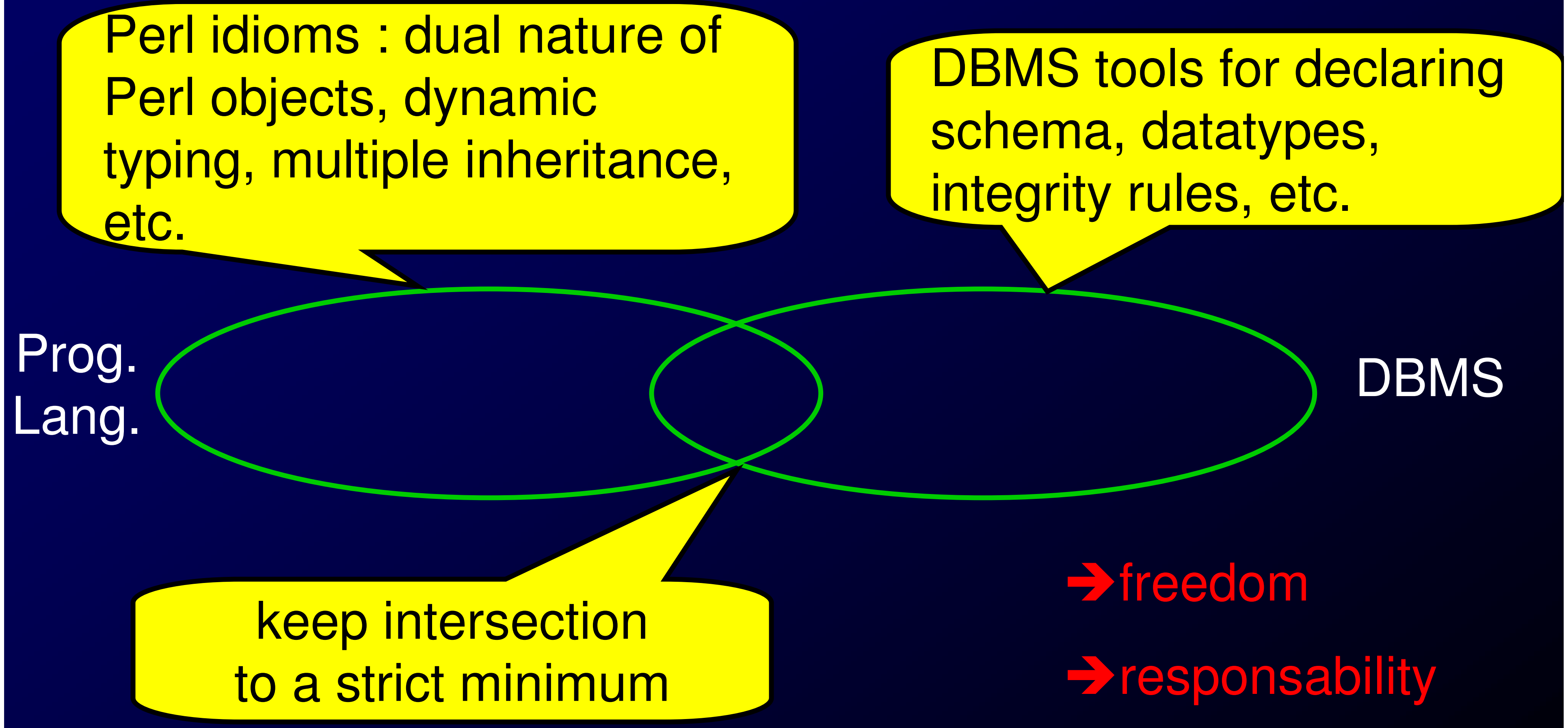
```
# declare schema (Module-Author-Distribution)
DBIx::DataModel->Schema('MAD');

# declare tables in schema
#                Class    DB_table   Primary_key
MAD->Table([qw/Author   t_author   author_id /]);
MAD->Table([qw/Distrib t_distrib distrib_id/]);
MAD->Table([qw/Module   t_module   module_id /]);
```

Don't Repeat Yourself



Design philosophy


Perl idioms : dual nature of Perl objects, dynamic typing, multiple inheritance, etc.
DBMS tools for declaring schema, datatypes, integrity rules, etc.
Prog. Lang.
DBMS
keep intersection to a strict minimum

→freedom

→responsability

Fetching data : example 1

```
# fetch from primary key
# by default, retrieves all columns ('*')
my $author = Author->fetch('dami');

# reach columns through the hashref API ...
while (my ($k, $v) = each %$author) {
  print "$k : $v\n";
}
# ... or use object-oriented methods
MAD->Autoload(1); # Autoload is off by default
print $author->e_mail();
```

Fetching data : example 2

```
# select multiple records
my $recent_distribs = Distrib->select(
 -columns => [qw/distrib_name d_release/]      ,
 -where   => {d_release => {'>' => $some_date}},
 -orderBy => 'd_release DESC'                  ,
);

foreach my $distrib (@$recent_distribs) {...}
```

Select API

```
TableOrView->select(
  -columns   => \@columns,
    # OR : -distinct => \@columns,
  -where     => \%where,
  -groupBy   => \@groupings,
  -having    => \%criteria,
  -orderBy   => \@order,
  -for       => 'read only',
  -preExec   => \&preExec_callback,
  -postExec  => \&preExec_callback,
  -resultAs  => 'rows' || 'sth' || 'sql'
                       || 'iterator');
```

➔ See Also : **SQL::Abstract**

Retrieving columns

- programmer decides which

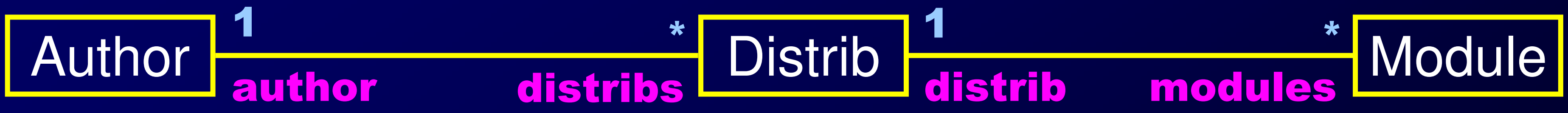
```
    -columns  => \@columns       # arrayref
    -columns  => "col1, col2" # string
    -columns  => "*"             # default
```

- no delayed fetching
- objects have variable size !
- runtime error if missing keys
  - for following joins
  - for updates and deletes

Associations

- declaration
- usage

Association declarations



```
#    Class    role_name  multipl. join_key(s)
MAD->Association(
 [qw/Author   author     1       author_id/],
 [qw/Distrib  distribs   *       author_id/]);

MAD->Association(
 [qw/Distrib  distrib    1       distrib_id/],
 [qw/Module   modules    *       distrib_id/]);
```

Associations

```
MAD->Association(
  [qw/Author    author       1     author_id/],
  [qw/Distrib   distribs     *     author_id/]);
```

creates method **Distrib::author**

which returns a single object

creates method **Author::distribs**

which returns an arrayref

creates method **Author::insert_into_distribs**

will generate LEFT OUTER JOINS

Role methods

```
foreach my $distrib (@$recent_distribs) {
  my $id     = $distrib->{distrib_id};
  my $author = $distrib->author();
  my $other_distribs = $author->distribs(
     -where    => {distrib_id => {'<>' => $id}},
     -resultAs => 'iterator'
     );
}
```

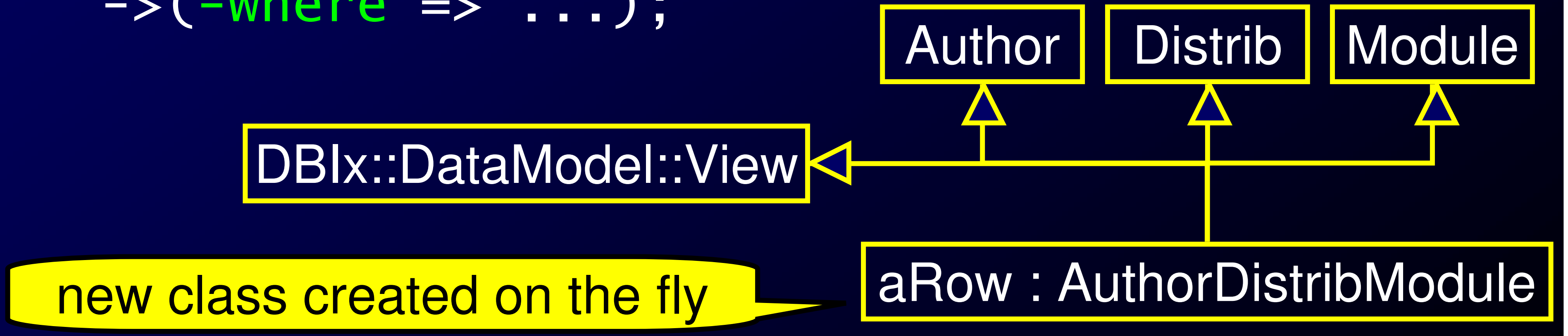
role methods

same API as TableOrView::select()

Follow several roles at once

```
$rows = MAD
  ->ViewFromRoles(qw/Author distribs modules/)
  ->select(-where => ...);

$rows = $author
  ->selectFromRoles(qw/distribs modules/)
  ->(-where => ...);
```



Generated SQL

```
$rows = $author
  ->selectFromRoles(qw/distribs modules/)
  ->(-columns => [qw/distrib_name module_name/],
     -where   => {d_release => {'<' => $date});
```

```
SELECT         distrib_name, module_name
FROM           Distrib
  LEFT OUTER JOIN Module
  ON  Distrib.distrib_id = Module.distrib_id
WHERE distrib.author_id = $author->{author_id}
  AND d_release < $date
```

n-to-n Associations

```
MAD->Association(# from table1 to the link table
  [qw/Distrib    distrib      1  distrib_id/],
  [qw/Dependency dependencies *  distrib_id/]);

MAD->Association(# from table2 to the link table
  [qw/Module     module       1  module_id/],
  [qw/Dependency dependencies *  module_id/]);

MAD->Association(# n-to-n assoc with role names
  [qw/Distrib distribs * dependencies distrib/],
  [qw/Module  modules  * dependencies module /]);
```

Not covered here

- updates and transactions
- tree expansions and exports (XML, Json)
- column handlers for
  - data conversions (scalar or object)
  - data validation
- adding ad hoc methods
- criteria combinations : preselectWhere
- Views
- ...

→ See Also : **DBIx::DataModel** manual