MySQL4OCL: Un compilador de OCL a MySQL

Carolina Inés Dania

Director: Dr. Manuel García Clavel
Colaboradora externa: Dra. Marina Soledad Egea Gonzalez

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

28 de septiembre de 2011

Organización

Organización

- Introducción

  Desarrollo dirigido por modelos

  UML - OCL

  Base de datos - SQL y MySQL

Organización

- Introducción

  Desarrollo dirigido por modelos

  UML - OCL

  Base de datos - SQL y MySQL

- MySQL4OCL: Descripción general

  De diagramas UML a tablas MySQL

  De expresiones OCL a procedimientos almacenados

Organización

- Introducción

  Desarrollo dirigido por modelos

  UML - OCL

  Base de datos - SQL y MySQL

- MySQL4OCL: Descripción general

  De diagramas UML a tablas MySQL

  De expresiones OCL a procedimientos almacenados

- MySQL4OCL: Arquitectura

  Optimizaciones y caso de estudio

Organización

- Introducción

  Desarrollo dirigido por modelos

  UML - OCL

  Base de datos - SQL y MySQL

- MySQL4OCL: Descripción general

  De diagramas UML a tablas MySQL

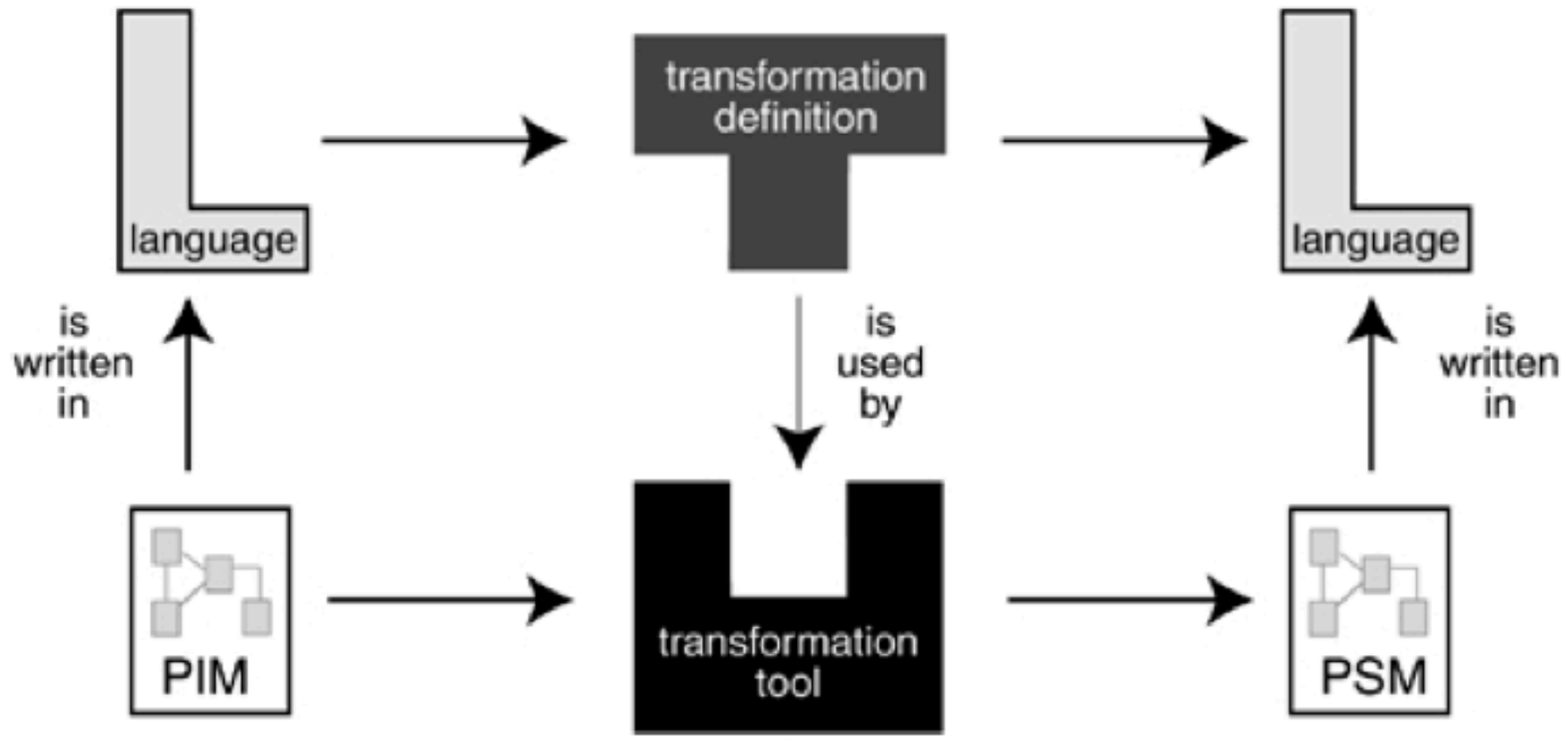
  De expresiones OCL a procedimientos almacenados

- MySQL4OCL: Arquitectura

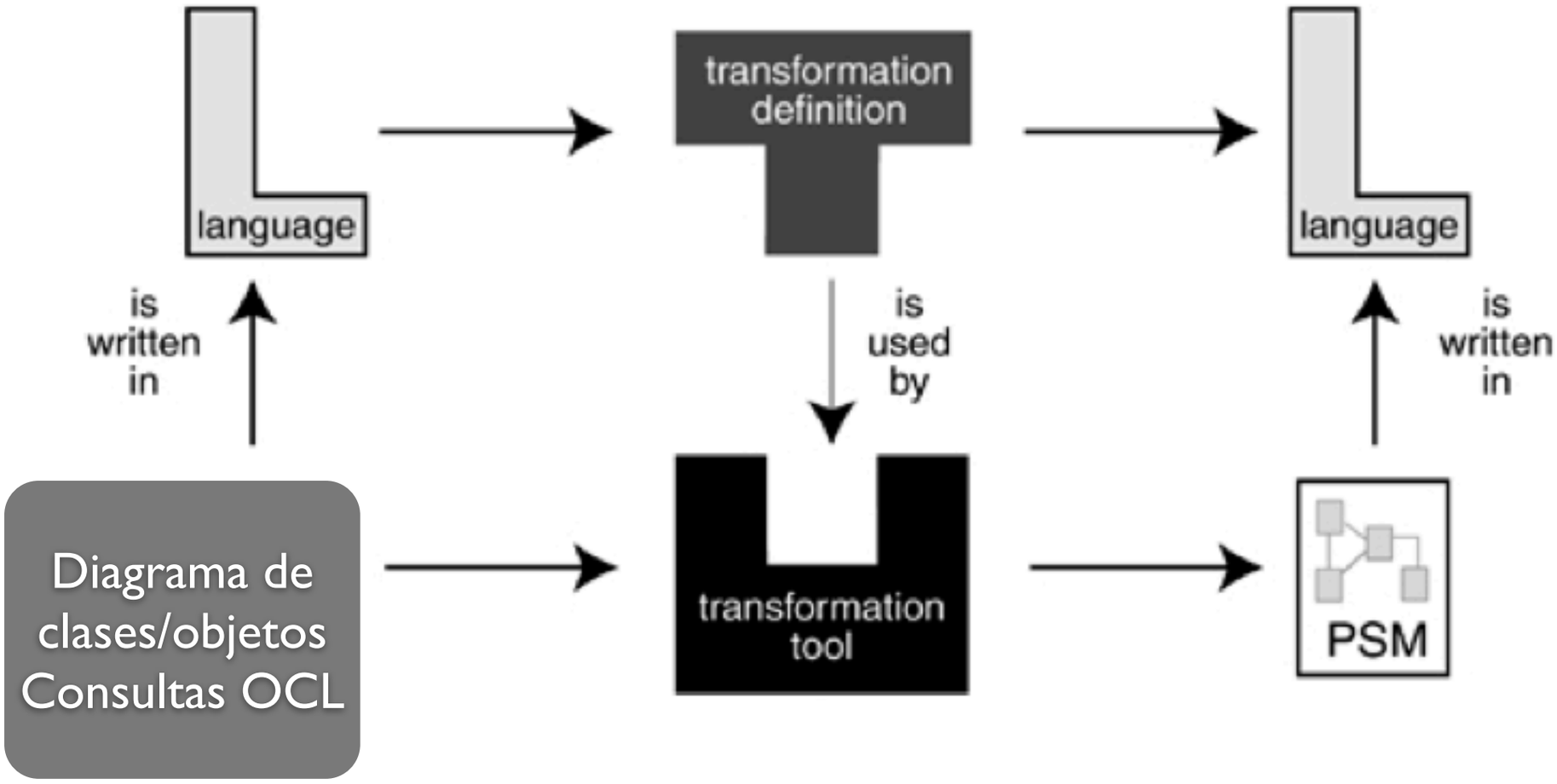
  Optimizaciones y caso de estudio

- Conclusiones y trabajo futuro

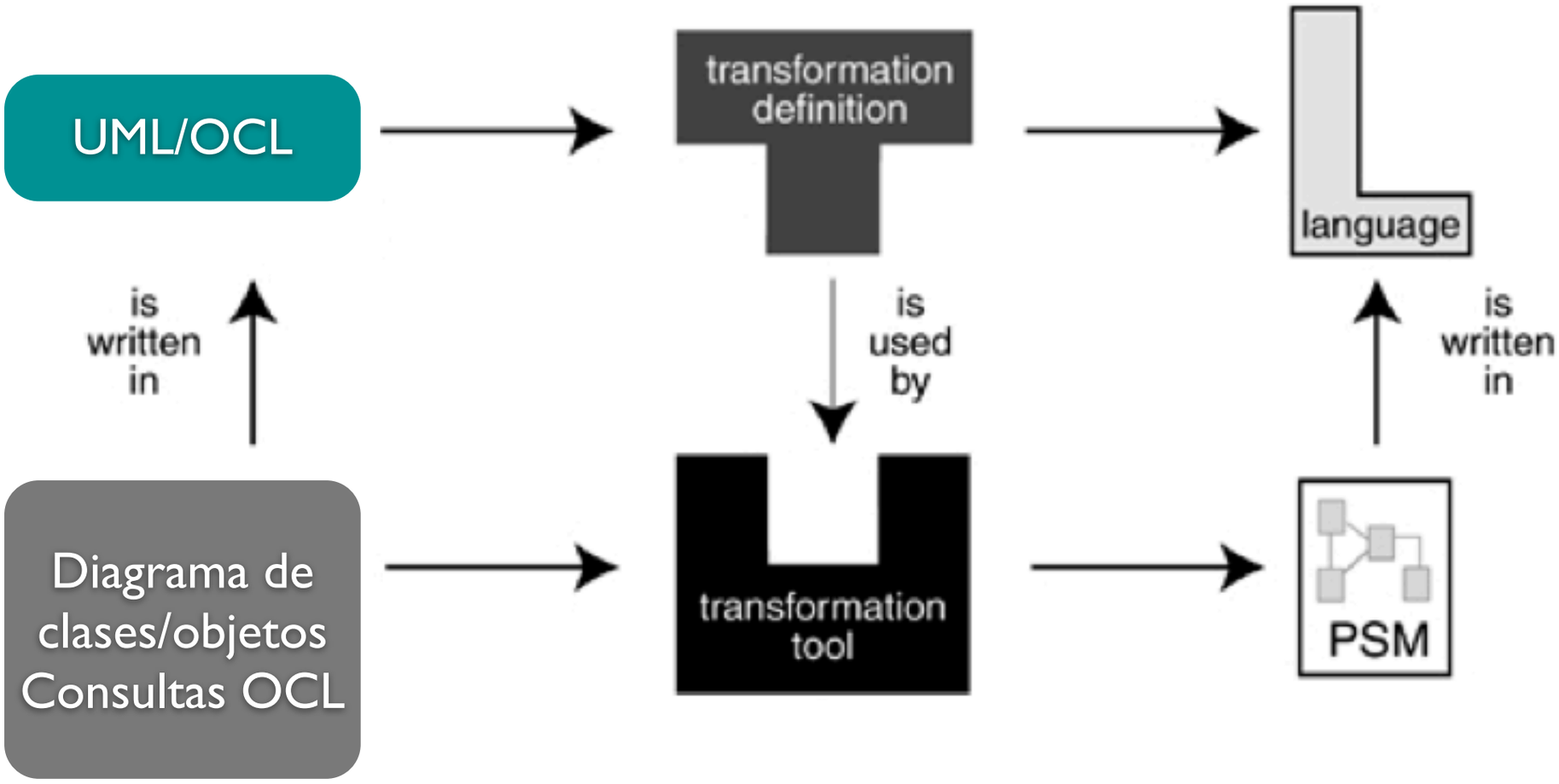
MDA: Desarrollo de software dirigido por modelos



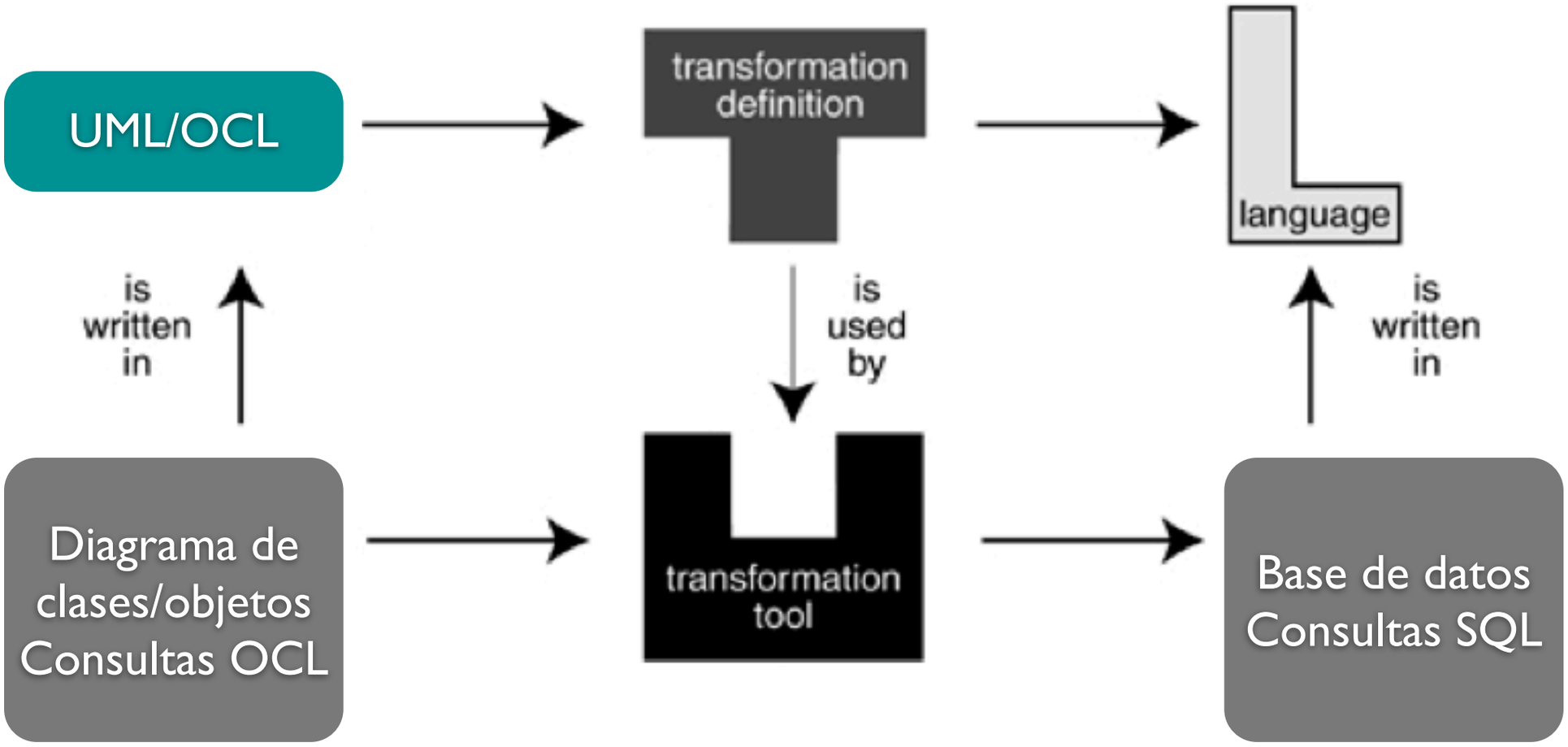
MDA: Desarrollo de software dirigido por modelos



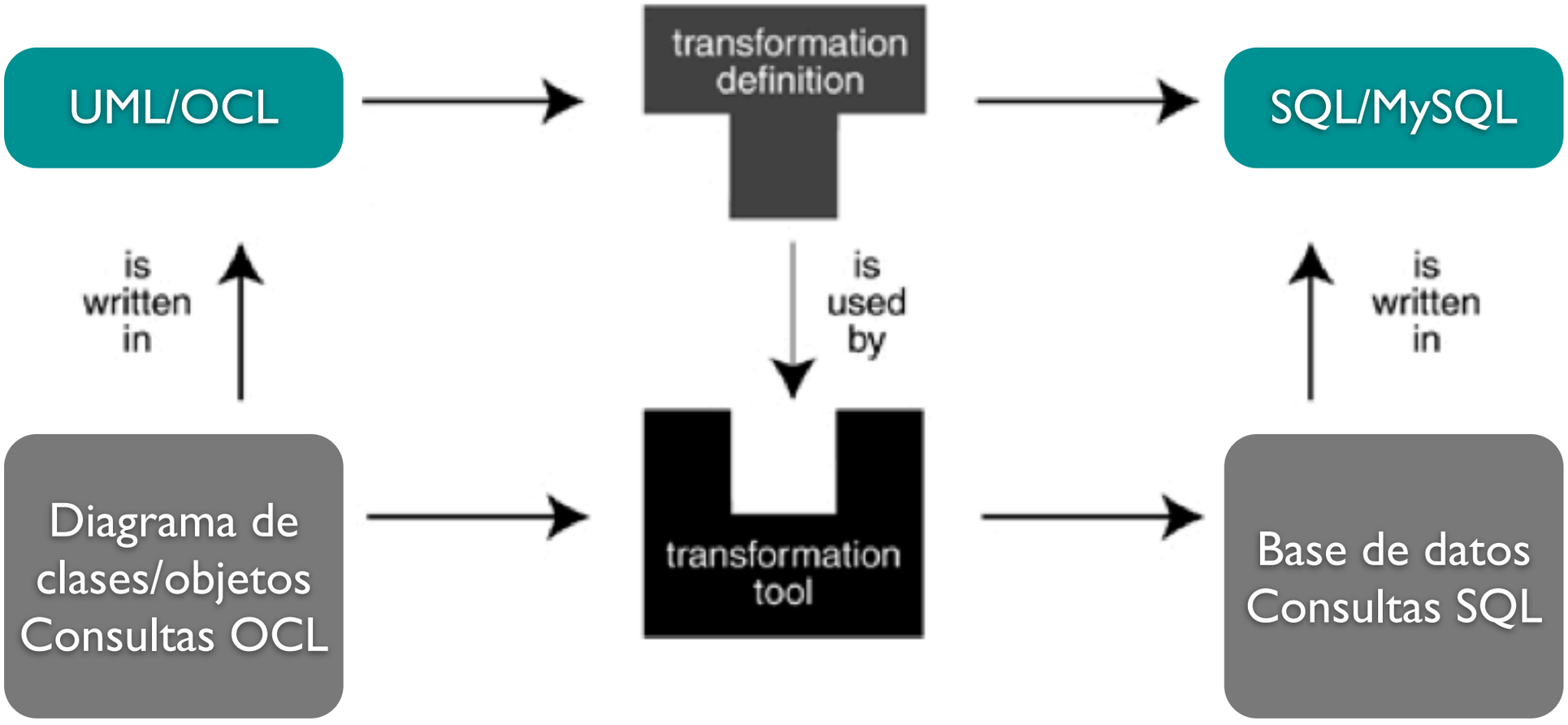
MDA: Desarrollo de software dirigido por modelos



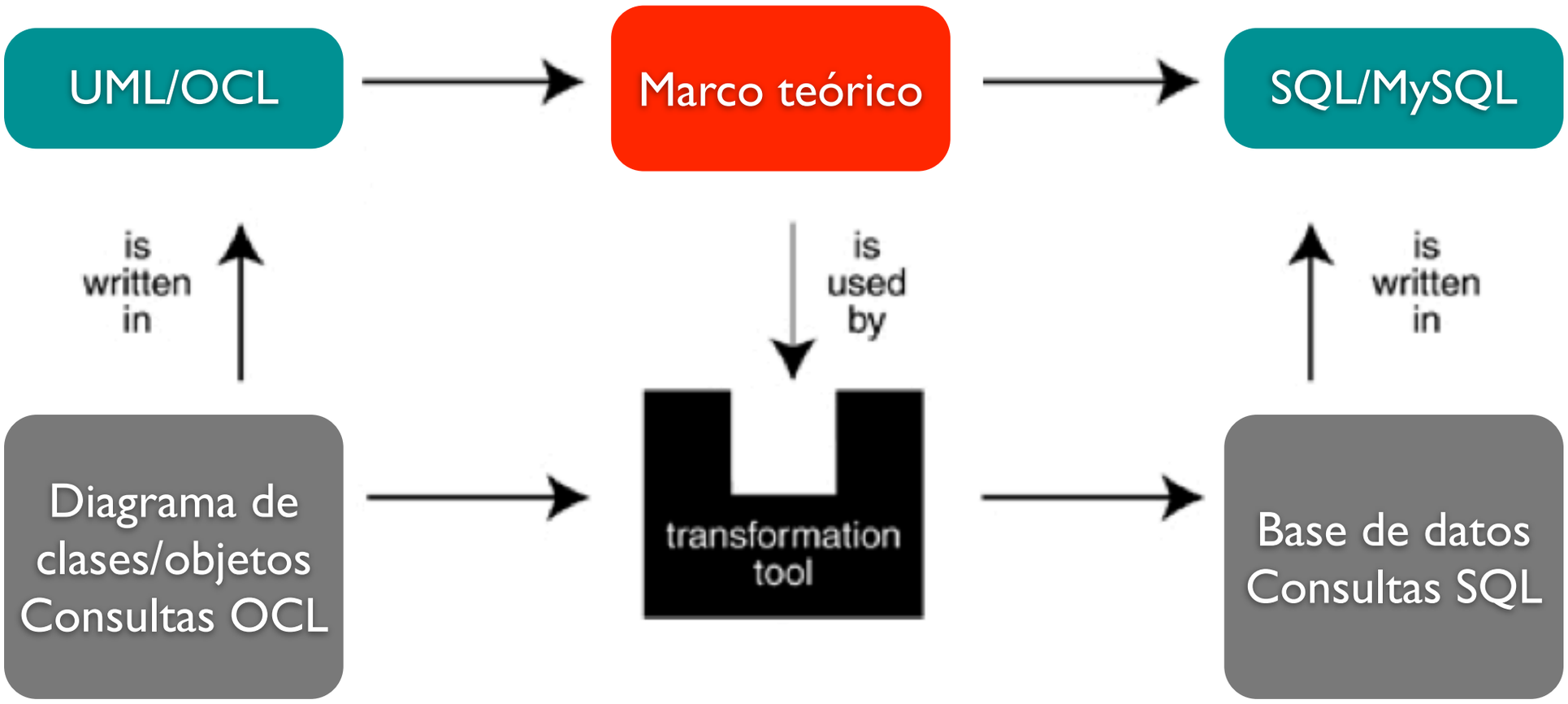
MDA: Desarrollo de software dirigido por modelos



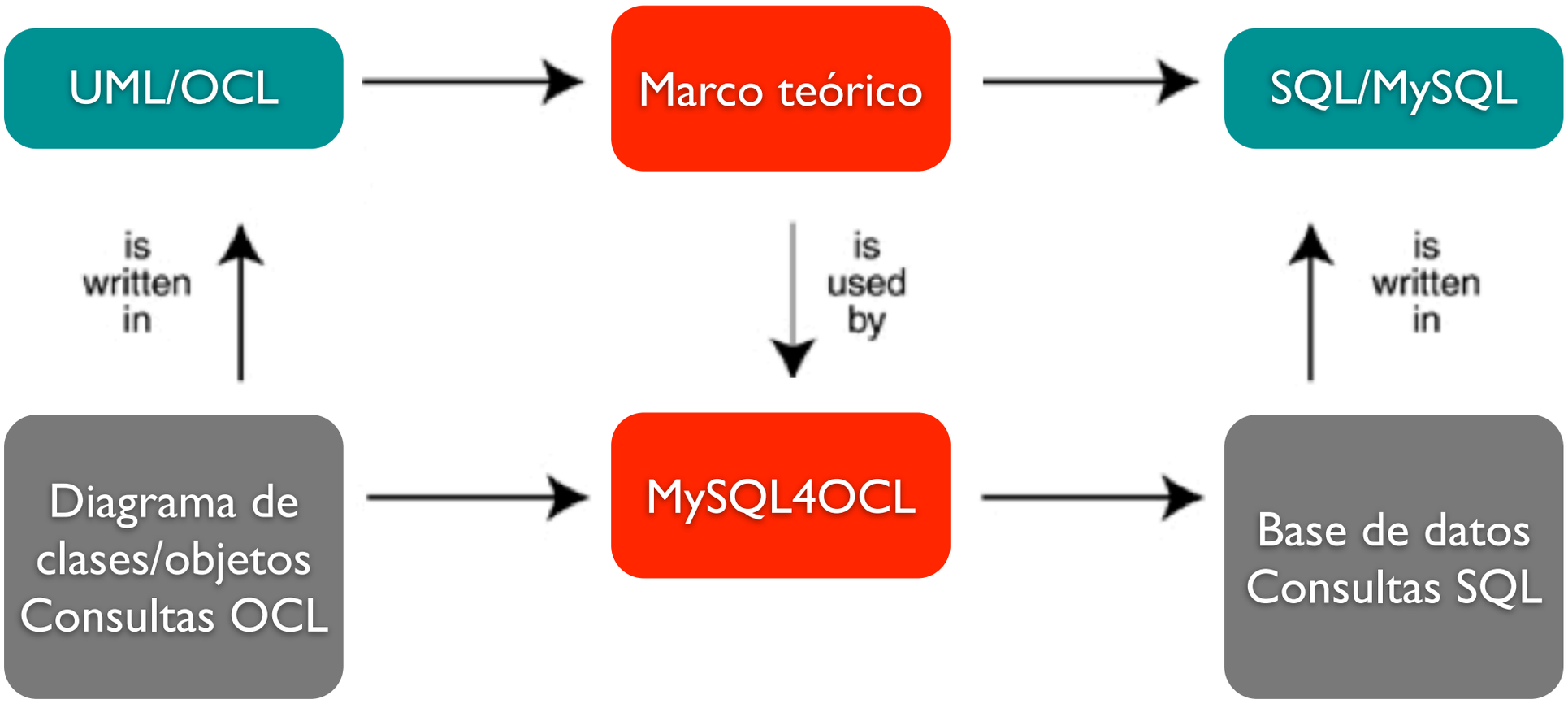
MDA: Desarrollo de software dirigido por modelos



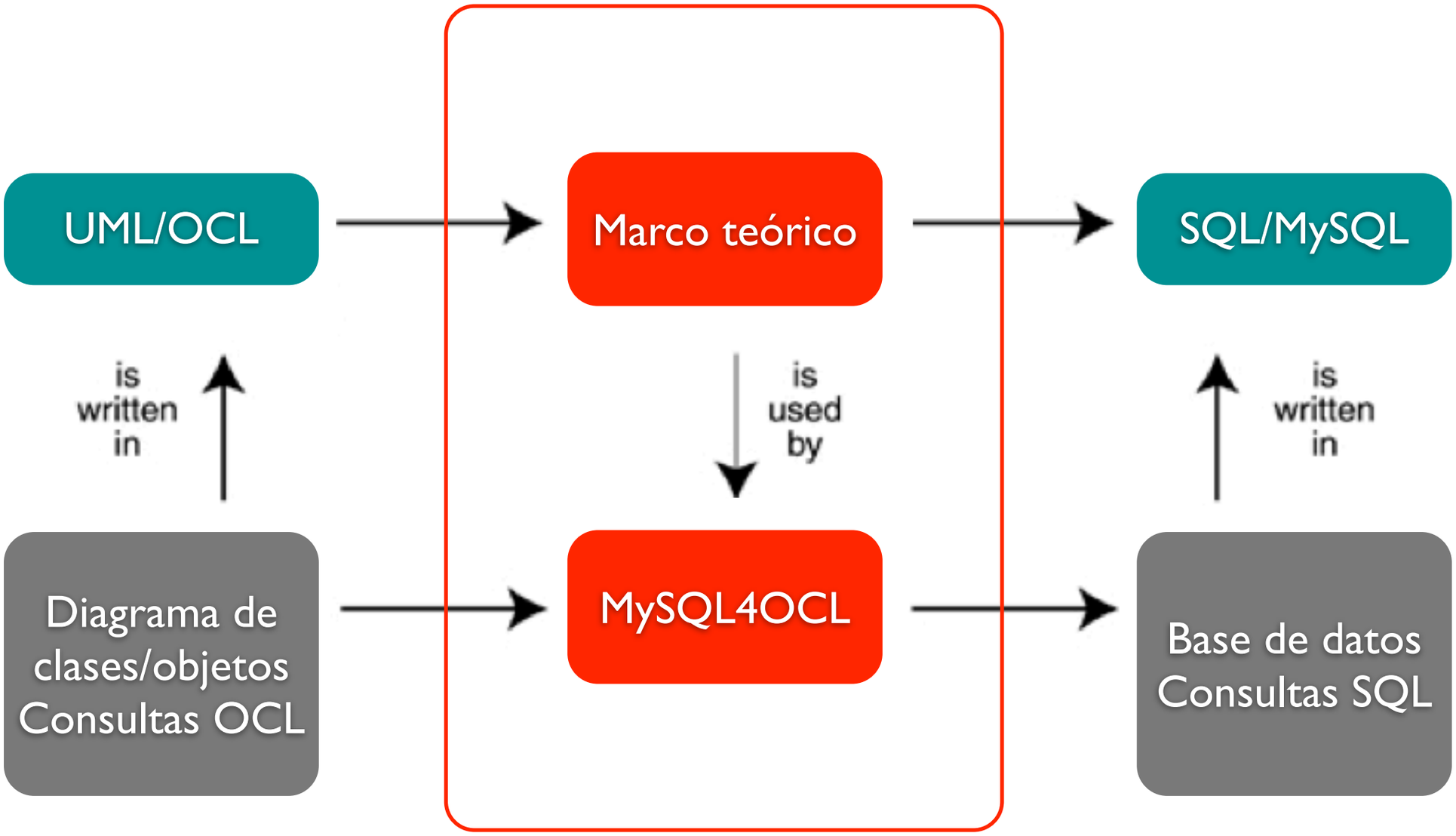
MDA: Desarrollo de software dirigido por modelos



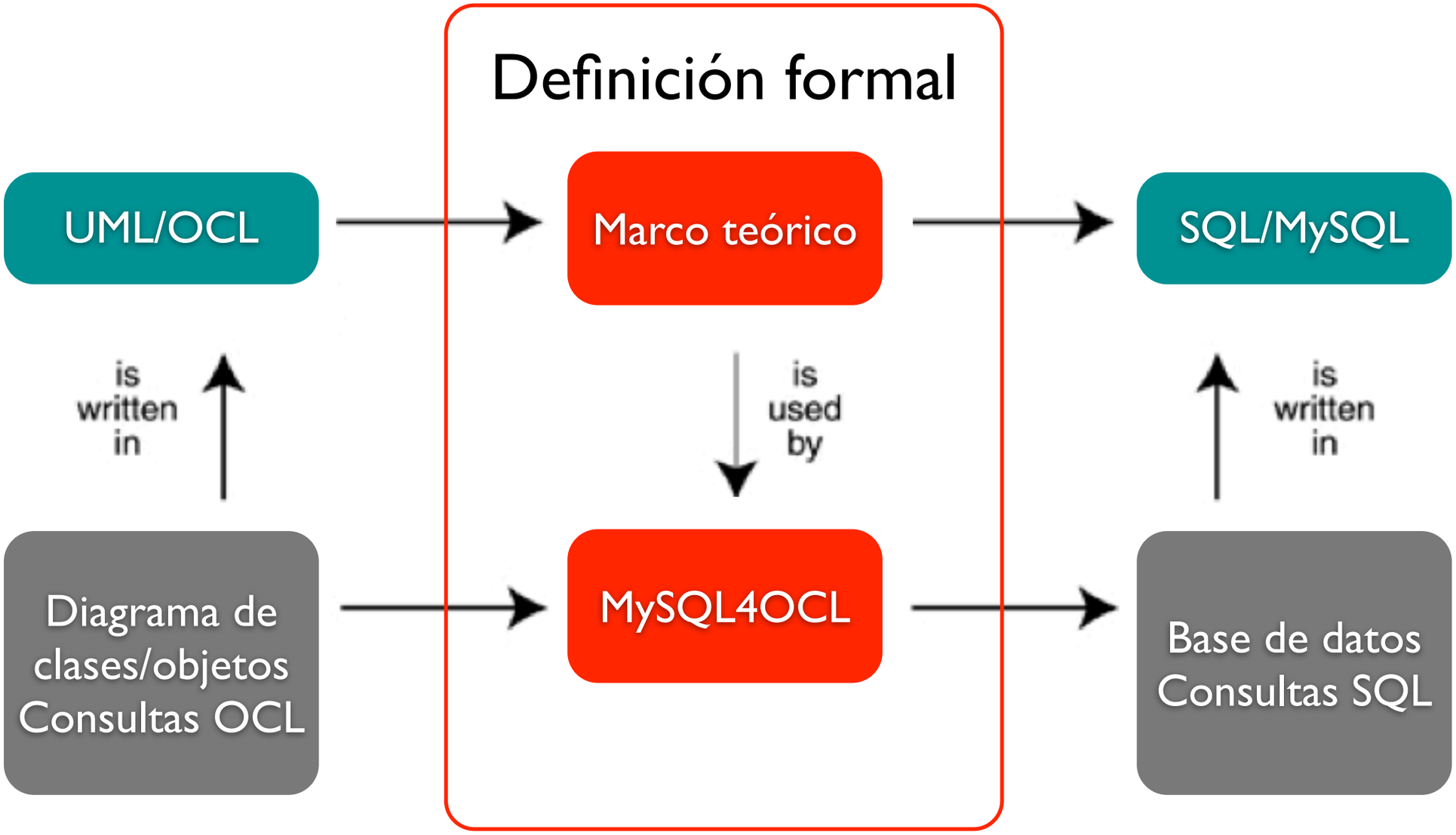
MDA: Desarrollo de software dirigido por modelos



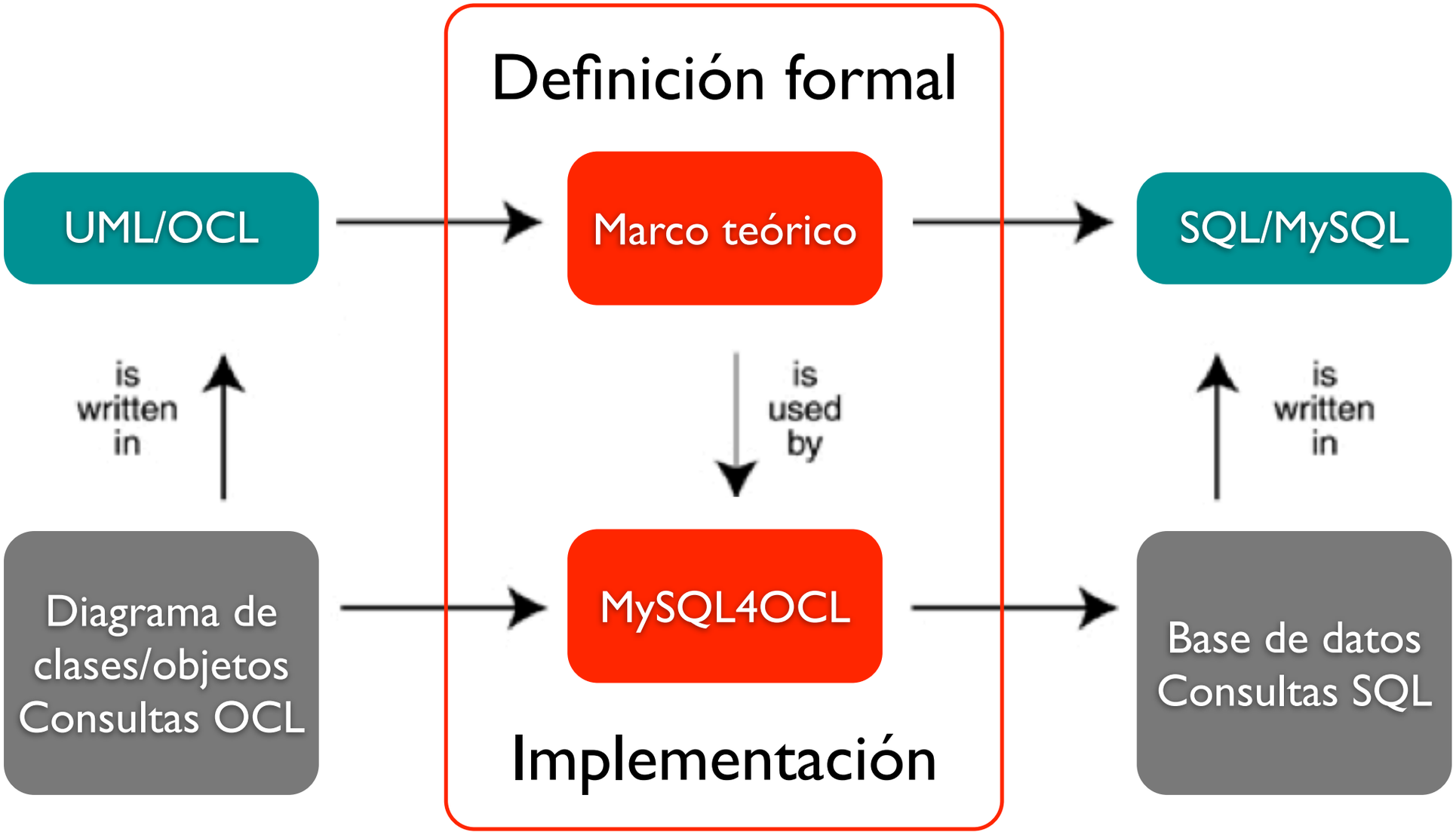
MDA: Desarrollo de software dirigido por modelos



MDA: Desarrollo de software dirigido por modelos



MDA: Desarrollo de software dirigido por modelos



UML: Lenguaje de modelado unificado

UML: Lenguaje de modelado unificado

- Diagrama de clases

UML: Lenguaje de modelado unificado

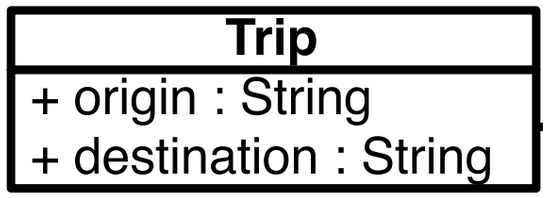
- Diagrama de clases
- Diagrama de objetos

UML: Lenguaje de modelado unificado

- Diagrama de clases
- Diagrama de objetos

UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
- Diagrama de objetos

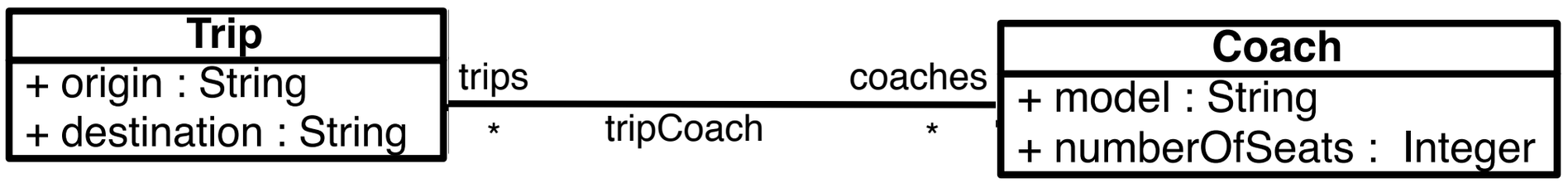


UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
- Diagrama de objetos

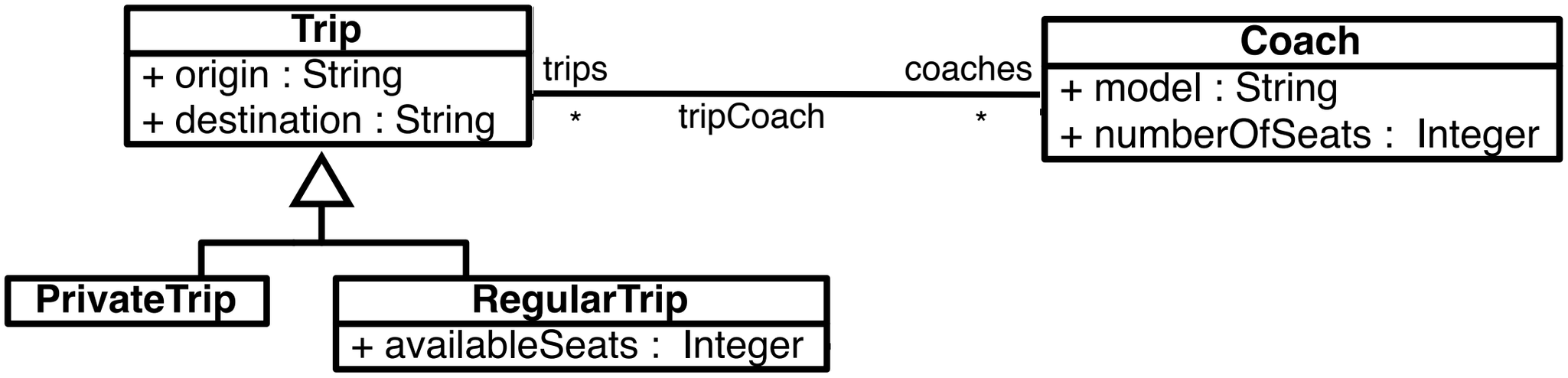
UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
- Diagrama de objetos



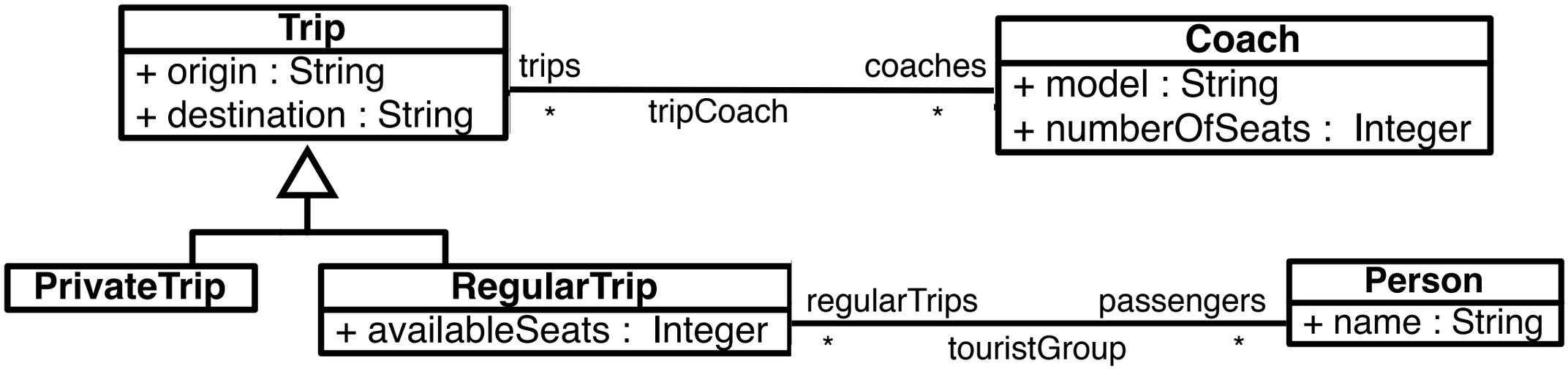
UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos



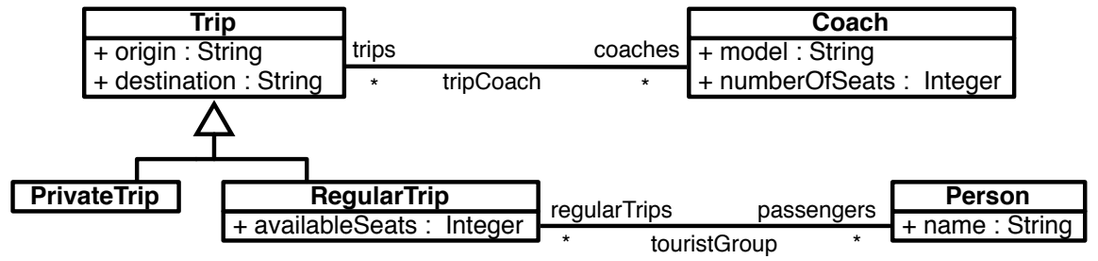
UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos



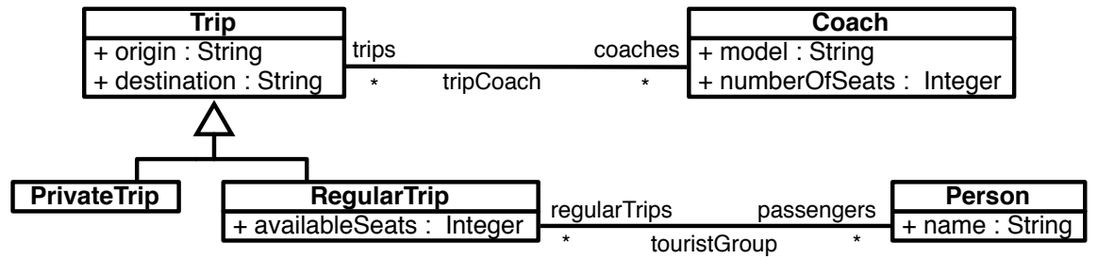
UML: Lenguaje de modelado unificado

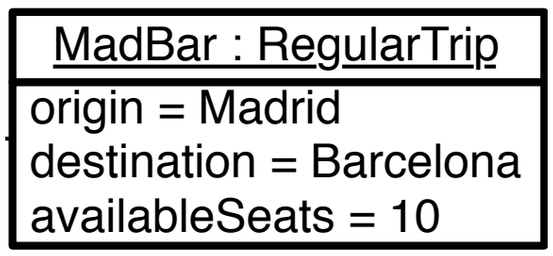
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos



UML: Lenguaje de modelado unificado

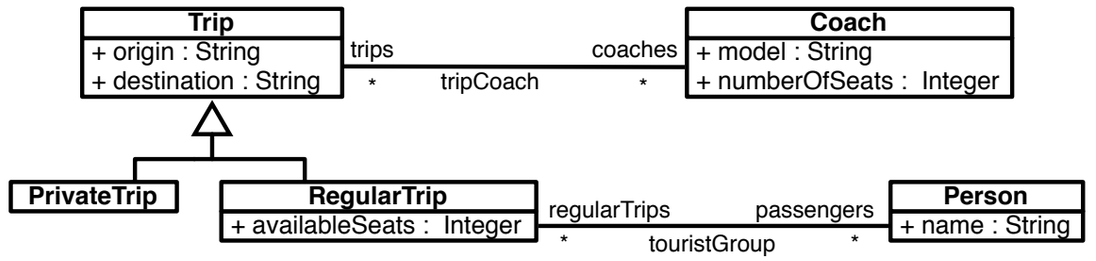
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores

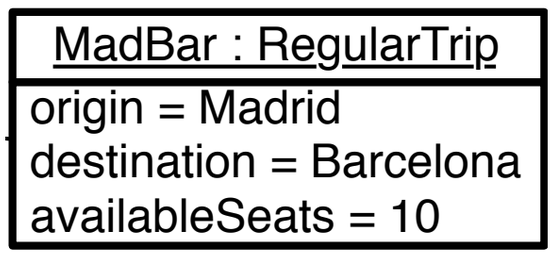


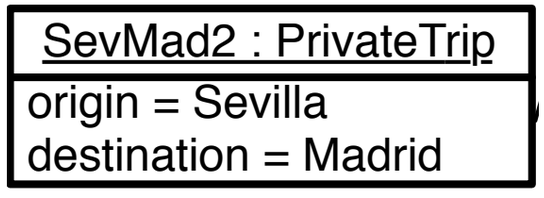


UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores

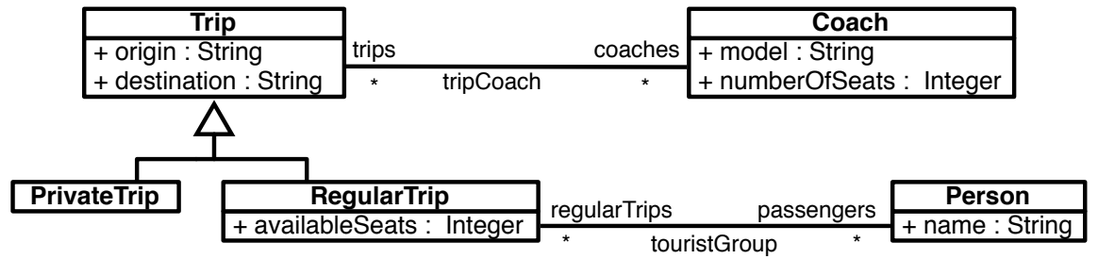


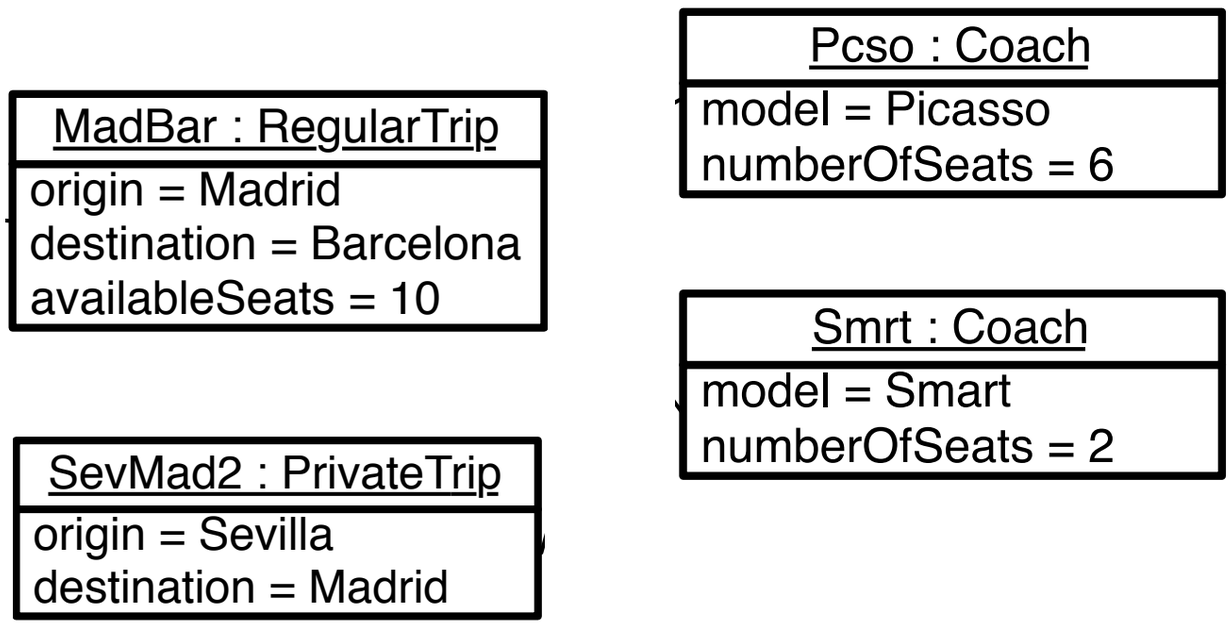




UML: Lenguaje de modelado unificado

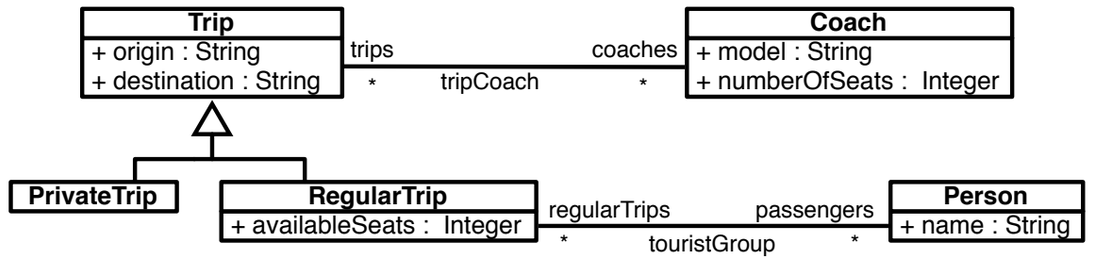
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores

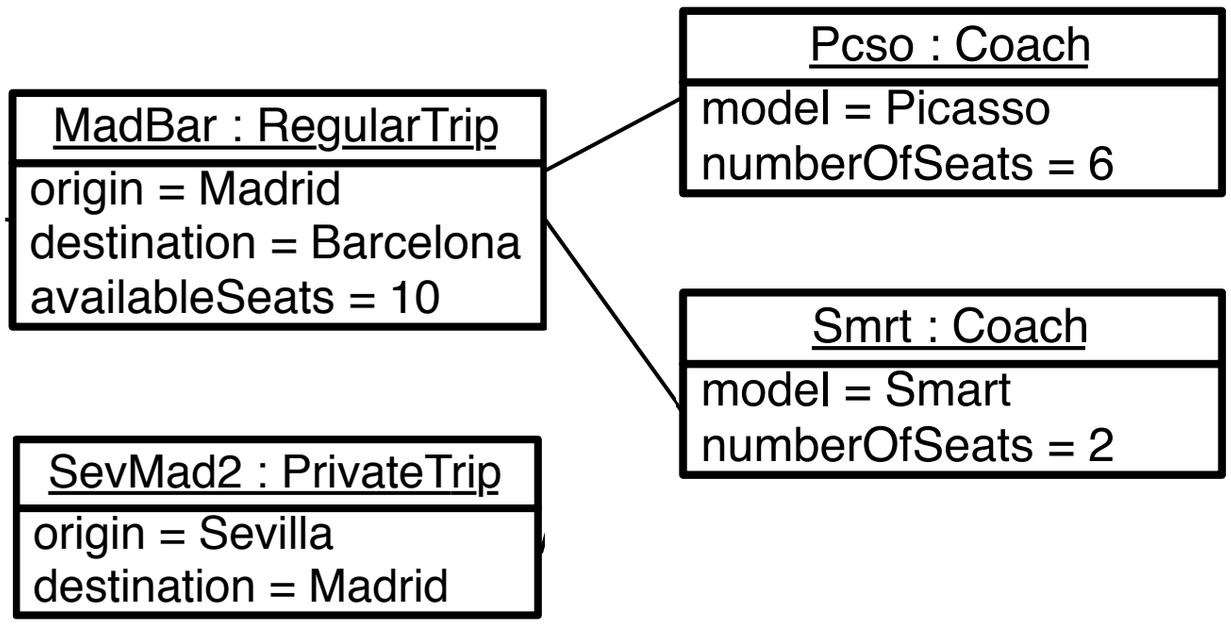




UML: Lenguaje de modelado unificado

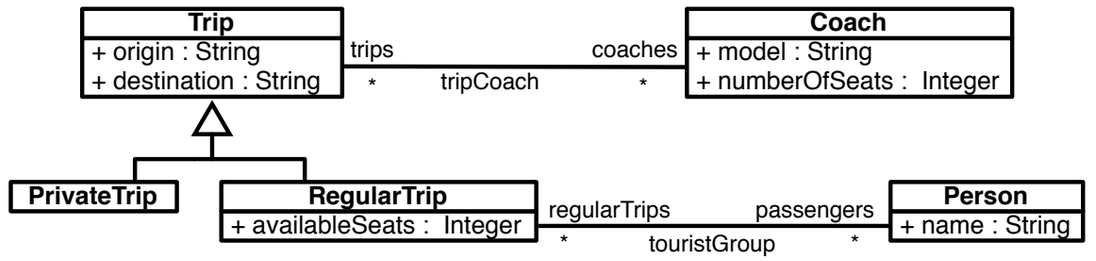
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores
  - enlaces

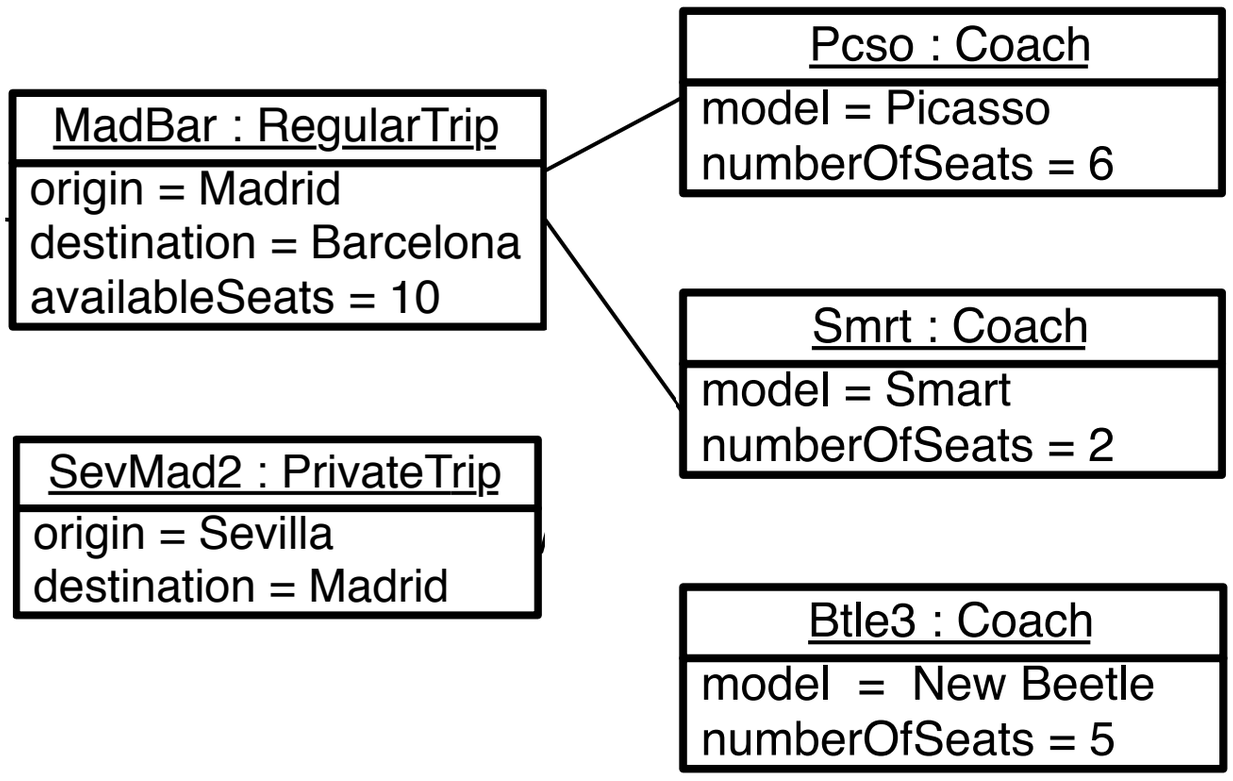




UML: Lenguaje de modelado unificado

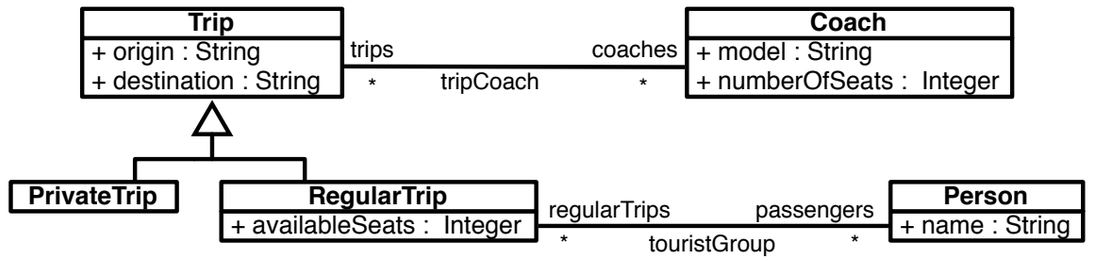
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores
  - enlaces

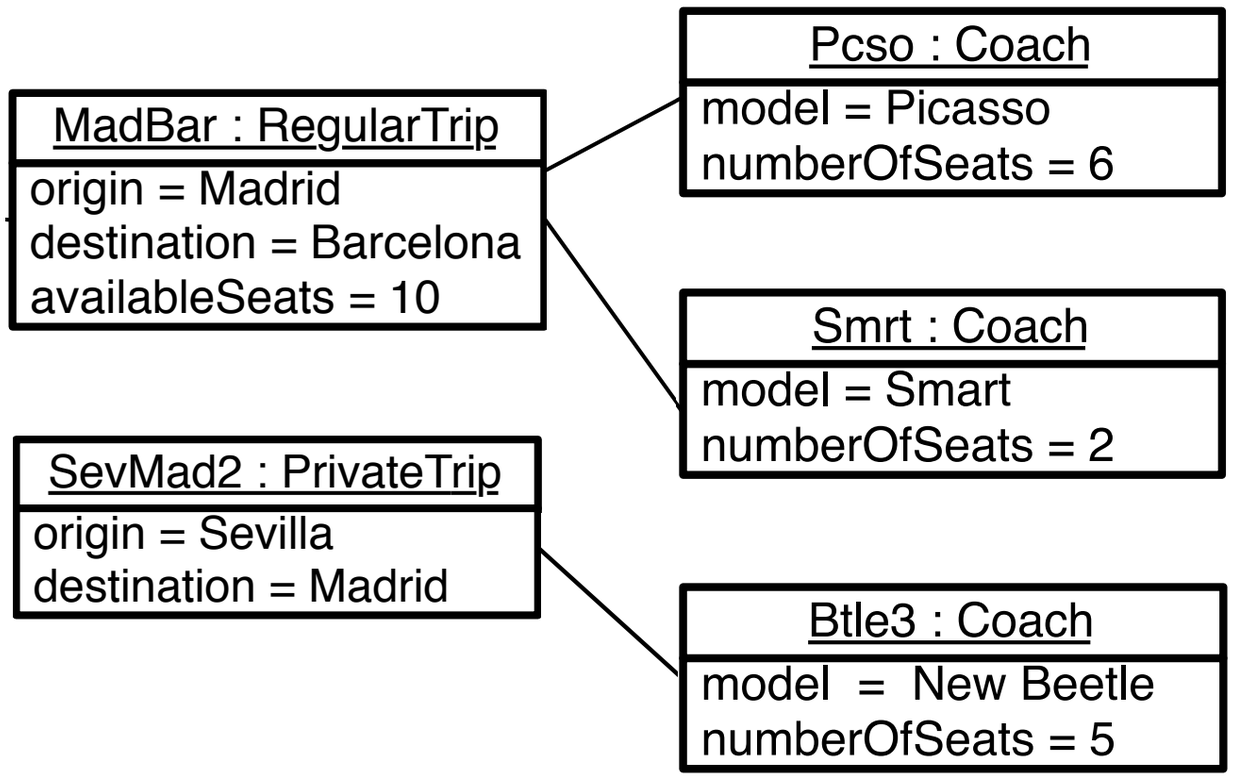




UML: Lenguaje de modelado unificado

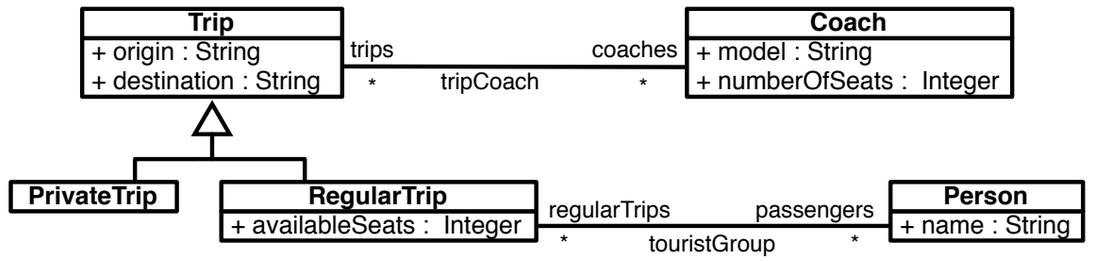
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores
  - enlaces

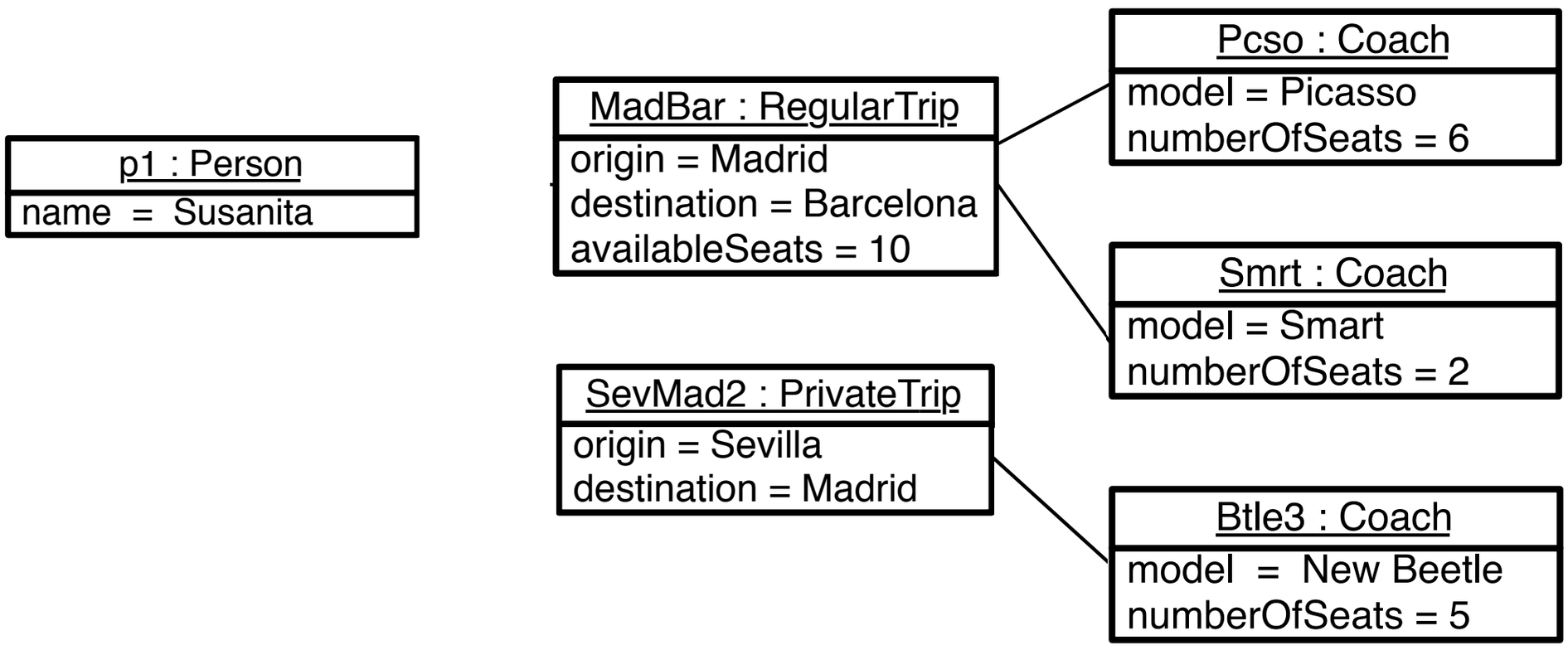




UML: Lenguaje de modelado unificado

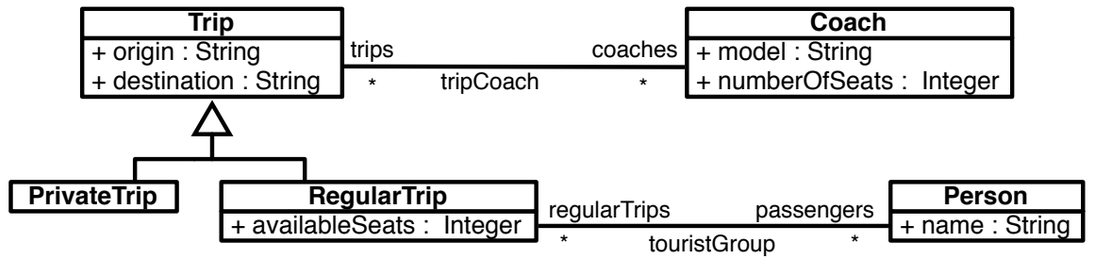
- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores
  - enlaces

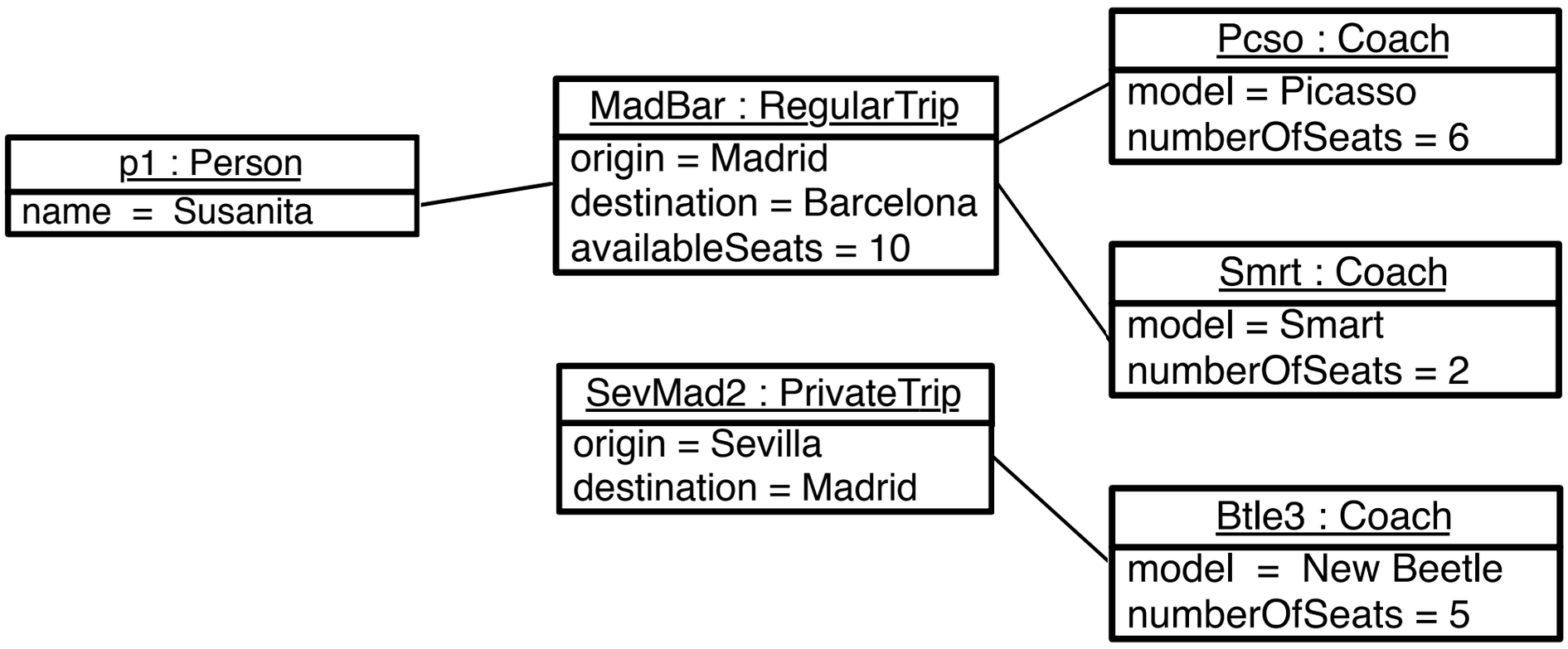




UML: Lenguaje de modelado unificado

- Diagrama de clases
  - clases
  - atributos
  - asociaciones (extremo de asociación)
  - herencia
- Diagrama de objetos
  - objetos
  - valores
  - enlaces





OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  `Madbar`

- Todos los viajes

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

- Todos los viajes

  Trip.allInstances()

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

- Todos los viajes

  Trip.allInstances()

- La suma de los asientos que hay en Pcso y Smrt

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

- Todos los viajes

  Trip.allInstances()

- La suma de los asientos que hay en Pcso y Smrt

  Pcso.numberOfSeats + Smrt.numberOfSeats

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

- Todos los viajes

  Trip.allInstances()

- La suma de los asientos que hay en Pcso y Smrt

  Pcso.numberOfSeats + Smrt.numberOfSeats

- Los viajes que tienen como origen Madrid

OCL: Lenguaje de restricciones y consultas a objetos (OMG - 2000)

- El viaje MadBar de Madrid a Barcelona

  Madbar

- Todos los viajes

  Trip.allInstances()

- La suma de los asientos que hay en Pcso y Smrt

  Pcso.numberOfSeats + Smrt.numberOfSeats

- Los viajes que tienen como origen Madrid

  Trip.allInstances()->select(t|t.origin='Madrid')

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - tablas estáticas y derivadas

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - tablas estáticas y derivadas

```
create table nameTbl(val Int);
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - tablas estáticas y derivadas

```
create table nameTbl(val Int);

select * from nameTbl;
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias

- sentencias

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural

  - tablas
  - sentencias

- sentencias

```
select * from nameTbl;
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias

- sentencias

```
select * from nameTbl;

insert into nameTbl(val) (select pk from Trip);
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural

  - tablas
  - sentencias
  - consultas

- consultas

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas

- consultas

```
select * from nameTbl
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas

- consultas

```
select * from nameTbl

select *
 from (select * from nameTbl) as t
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas
  - procedimientos almacenados
    - cursores
    - condicionales
    - bucles
- procedimientos almacenados

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas
  - procedimientos almacenados
    - cursores
    - condicionales
    - bucles
- procedimientos almacenados

```
declare procedure nameProc
begin
  ...
end;
```

Base de datos

- SQL: lenguaje de consulta estructurado
- MySQL: gestor de base de datos relacional que soporta SQL y lenguaje procedural
  - tablas
  - sentencias
  - consultas
  - procedimientos almacenados
    - cursores
    - condicionales
    - bucles
- procedimientos almacenados

```
declare procedure nameProc
begin
  ...
end;

call nameProc;
```

MySQL4OCL: Descripción general

MySQL4OCL: Descripción general

- MySQL4OCL compila cada expresión OCL *exp* como un procedimiento proc(*exp*) que típicamente invocará a los procedimientos que compilan las subexpresiones de la expresión dada.

MySQL4OCL: Descripción general

- MySQL4OCL compila cada expresión OCL *exp* como un procedimiento proc(*exp*) que típicamente invocará a los procedimientos que compilan las subexpresiones de la expresión dada.

- El resultado de cada procedimiento proc(*exp*) se almacena en una tabla table(proc(*exp*)).

MySQL4OCL: Descripción general

- MySQL4OCL compila cada expresión OCL *exp* como un procedimiento proc(*exp*) que típicamente invocará a los procedimientos que compilan las subexpresiones de la expresión dada.

- El resultado de cada procedimiento proc(*exp*) se almacena en una tabla table(proc(*exp*)).

```
call proc(exp);
select * from table(proc(exp))
```

Descripción general

Descripción general

Para toda expresión *exp*, en el contexto de un diagrama de clases *dcl*, MySQL4OCL utiliza los siguientes conjuntos de reglas:

Descripción general

Para toda expresión *exp*, en el contexto de un diagrama de clases *dcl*, MySQL4OCL utiliza los siguientes conjuntos de reglas:

- toRDB: define el modo de almacenar los diagramas de objetos del diagrama *dcl*. [cap. 3]

Descripción general

Para toda expresión *exp*, en el contexto de un diagrama de clases *dcl*, MySQL4OCL utiliza los siguientes conjuntos de reglas:

- toRDB: define el modo de almacenar los diagramas de objetos del diagrama *dcl*. [cap. 3]

- toProc: define el modo de generar el procedimiento proc(*exp*). [cap. 4,5 y anexo]

Descripción general

Para toda expresión *exp*, en el contexto de un diagrama de clases *dcl*, MySQL4OCL utiliza los siguientes conjuntos de reglas:

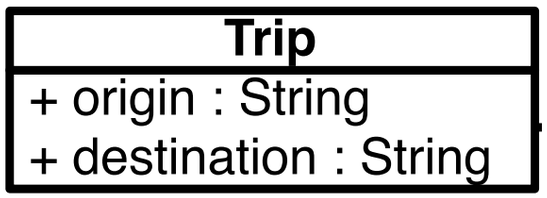
- toRDB: define el modo de almacenar los diagramas de objetos del diagrama *dcl*. [cap. 3]

- toProc: define el modo de generar el procedimiento proc(*exp*). [cap. 4,5 y anexo]

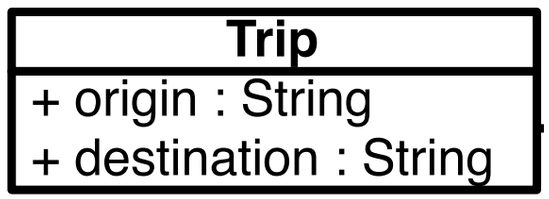
En las aplicaciones de MySQL4OCL, las bases de datos sobre las que se ejecutan las expresiones compiladas deben satisfacer las reglas toRDB.

toRDB:
De diagramas UML a tablas MySQL

toRDB:
De diagramas UML a tablas MySQL



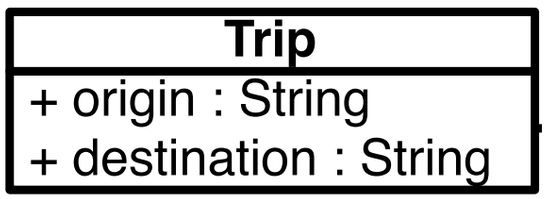
toRDB:
De diagramas UML a tablas MySQL

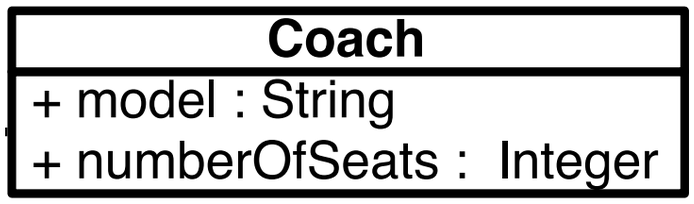


Trip

pk	origin	destination

toRDB:
De diagramas UML a tablas MySQL

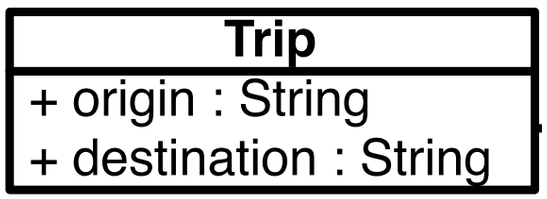


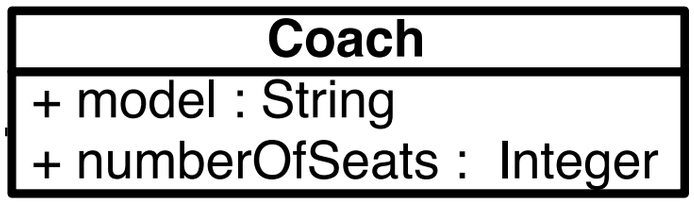


Trip

pk	origin	destination

toRDB:
De diagramas UML a tablas MySQL





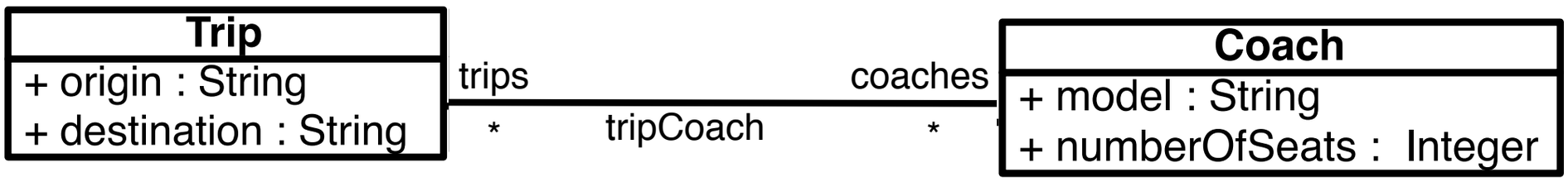
Trip

pk	origin	destination

Coach

pk	model	numberOfSeats

toRDB:
De diagramas UML a tablas MySQL



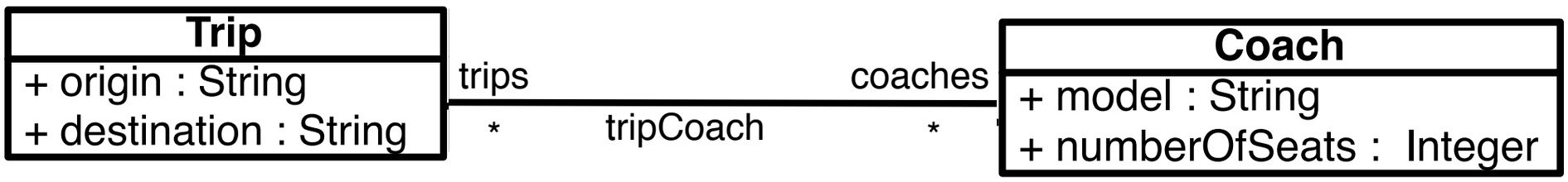
Trip

pk	origin	destination

Coach

pk	model	numberOfSeats

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination

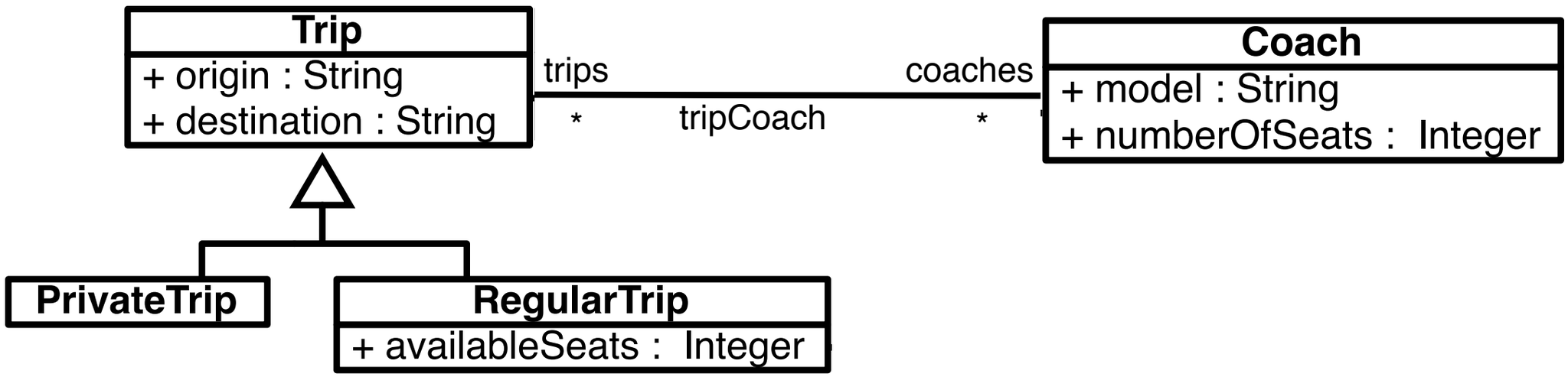
TripCoach

trips	coaches

Coach

pk	model	numberOfSeats

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination

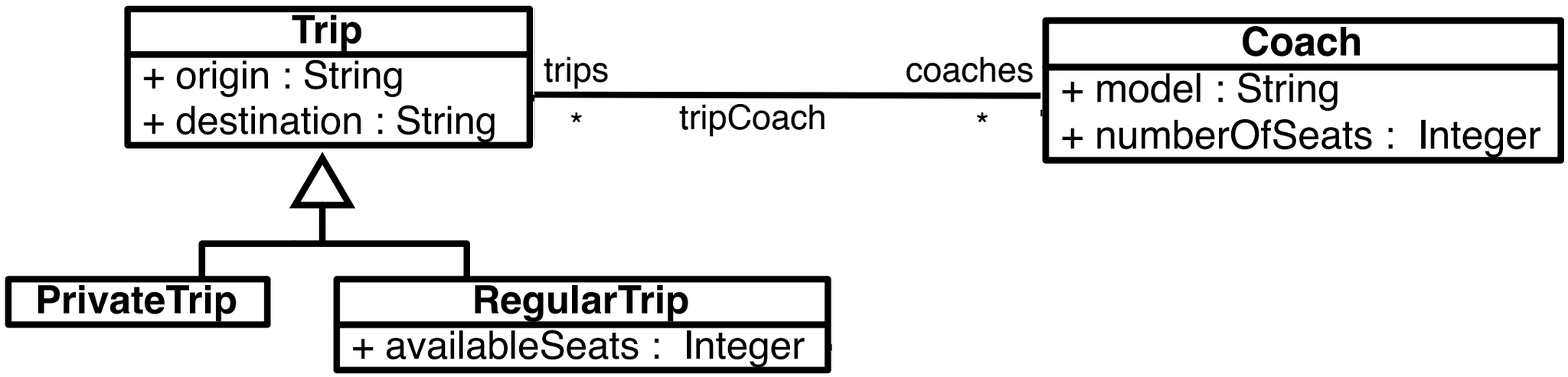
TripCoach

trips	coaches

Coach

pk	model	numberOfSeats

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination

TripCoach

trips	coaches

Coach

pk	model	numberOfSeats

PrivateTrip

pk

RegularTrip

pk	availableSeats

toRDB:
De diagramas UML a tablas MySQL

toRDB:
De diagramas UML a tablas MySQL

toRDB:
De diagramas UML a tablas MySQL

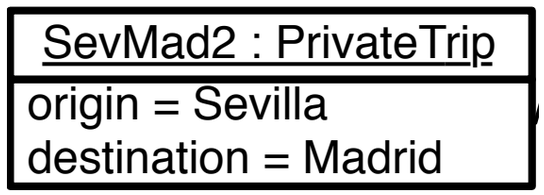
Trip

pk	origin	destination
1	Madrid	Zaragoza

RegularTrip

pk	availableSeats
1	10

toRDB:
De diagramas UML a tablas MySQL



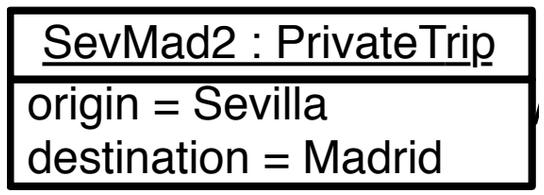
Trip

pk	origin	destination
1	Madrid	Zaragoza

RegularTrip

pk	availableSeats
1	10

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination
1	Madrid	Zaragoza
2	Sevilla	Madrid

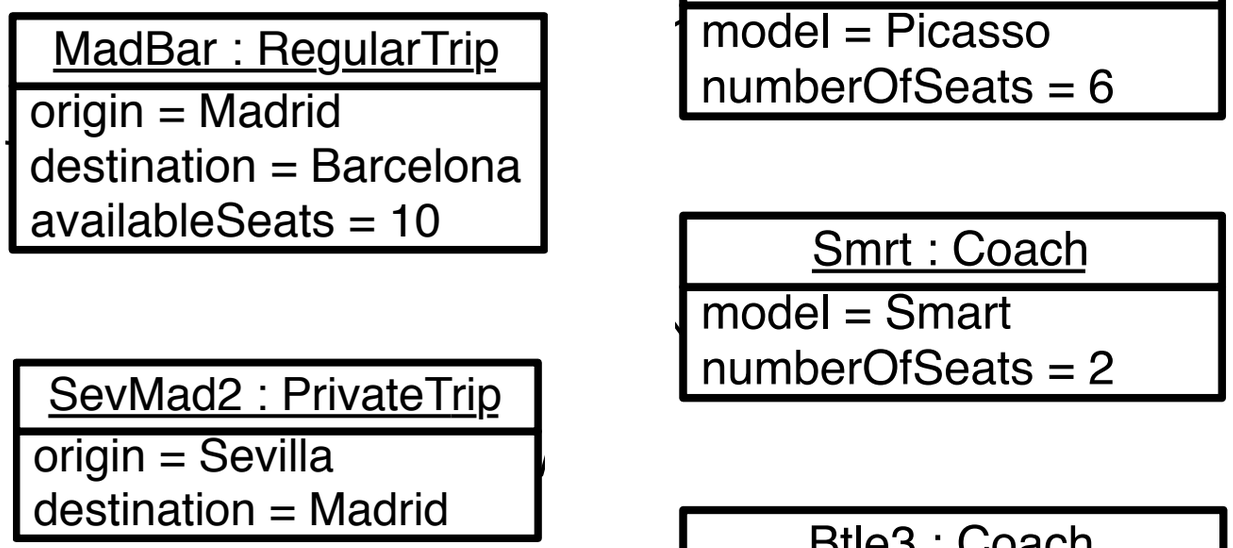
RegularTrip

pk	availableSeats
1	10

PrivateTrip

pk
2

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination
1	Madrid	Zaragoza
2	Sevilla	Madrid

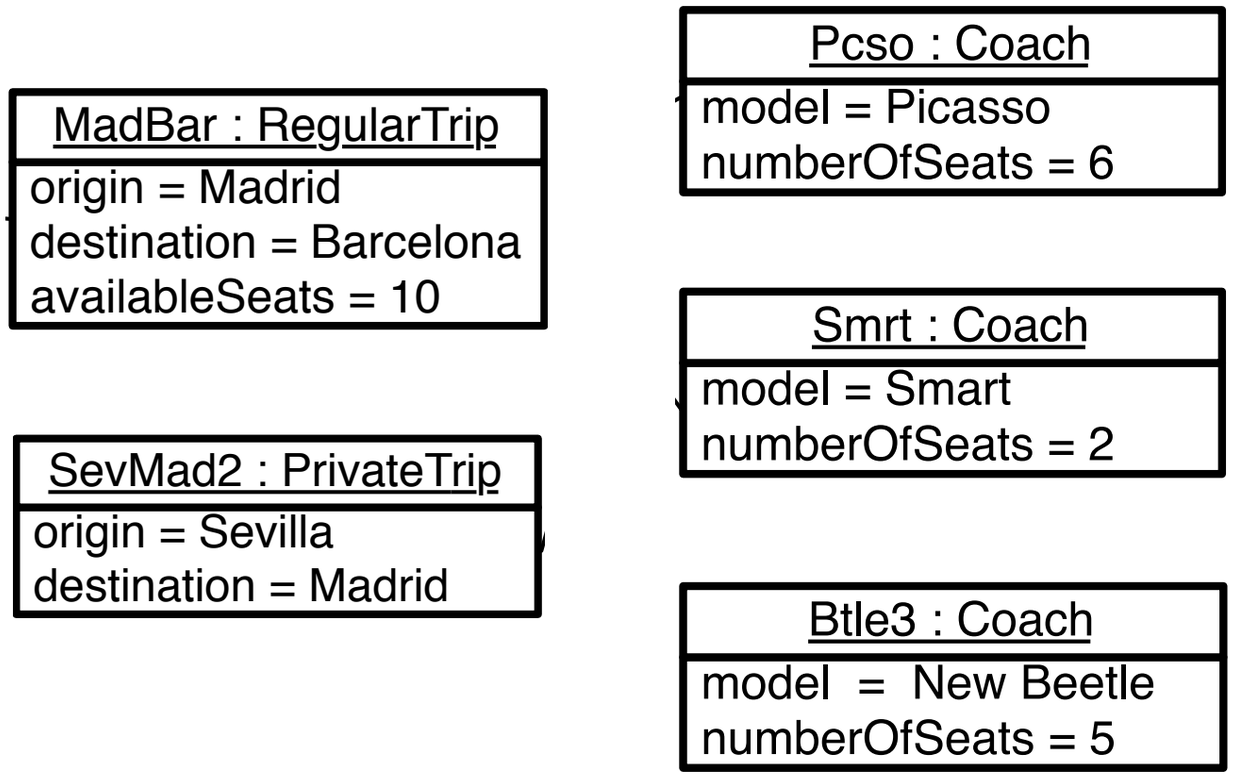
RegularTrip

pk	availableSeats
1	10

PrivateTrip

pk
2

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination
1	Madrid	Zaragoza
2	Sevilla	Madrid

RegularTrip

pk	availableSeats
1	10

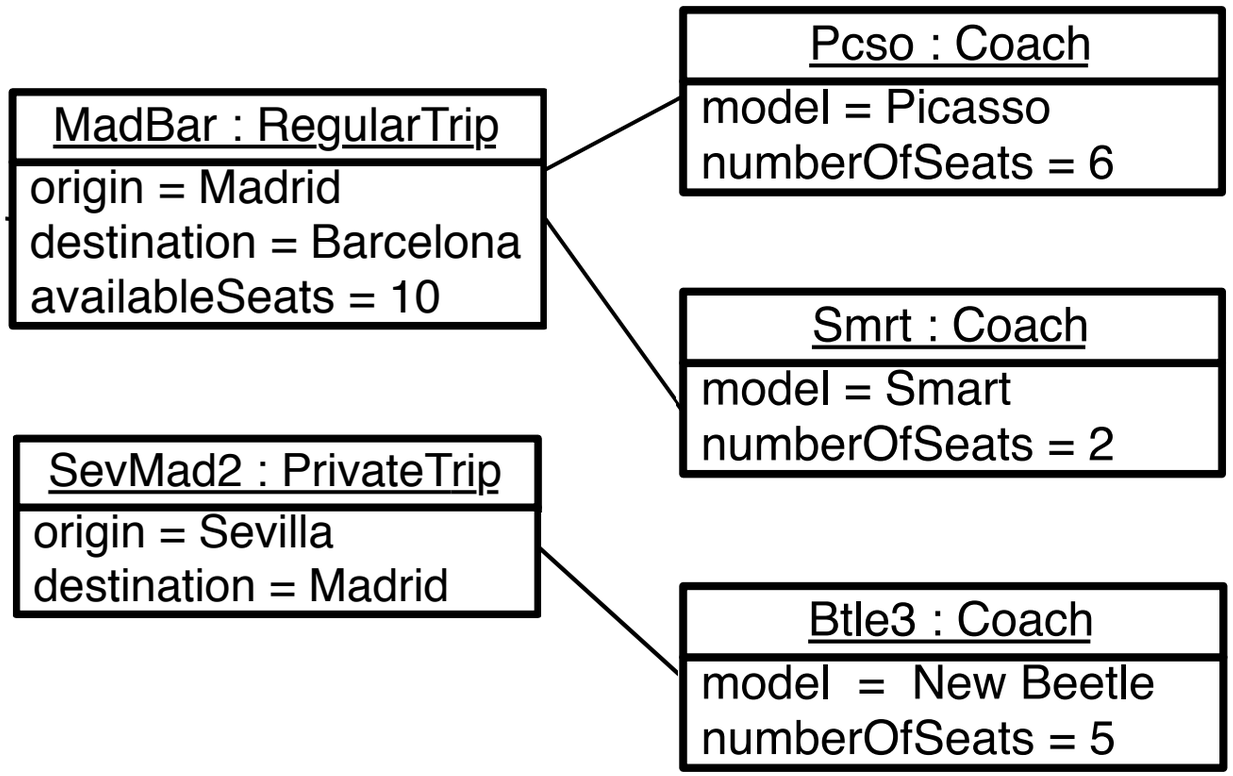
PrivateTrip

pk
2

Coach

pk	model	numberOfSeats
1	Picasso	6
2	Smart	2
3	New Beetle	5

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination
1	Madrid	Zaragoza
2	Sevilla	Madrid

Coach

pk	model	numberOfSeats
1	Picasso	6
2	Smart	2
3	New Beetle	5

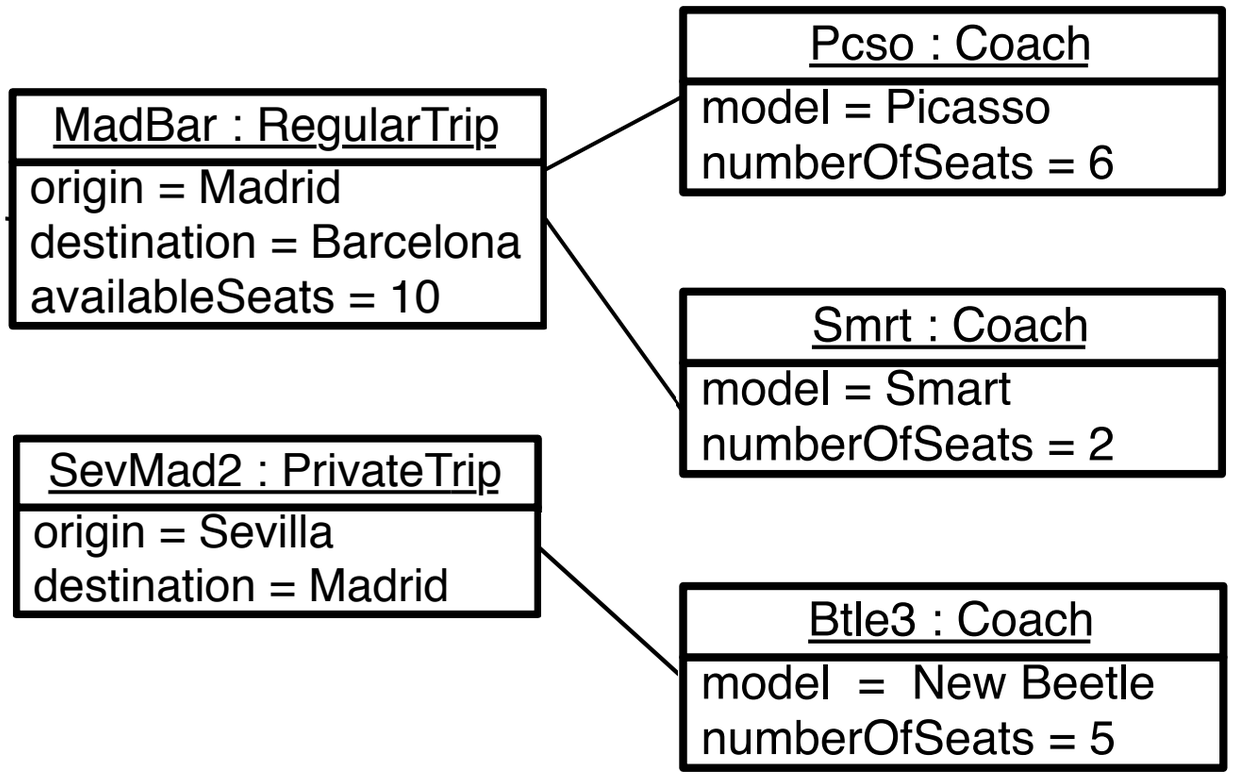
RegularTrip

pk	availableSeats
1	10

PrivateTrip

pk
2

toRDB:
De diagramas UML a tablas MySQL



Trip

pk	origin	destination
1	Madrid	Zaragoza
2	Sevilla	Madrid

TripCoach

trips	coaches
1	1
1	2
2	3

Coach

pk	model	numberOfSeats
1	Picasso	6
2	Smart	2
3	New Beetle	5

RegularTrip

pk	availableSeats
1	10

PrivateTrip

pk
2

toProc: De expresiones OCL a procedimientos almacenados

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: 1.+(2)

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: 1.+(2)

```
create procedure sum1()
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: 1.+(2)

```
create procedure sum1()
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: 1.+(2)

```
create procedure sum1()
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

```
create procedure const1()
begin
create table
 const1Tbl(val int);
insert into const1Tbl(val)
 (select 1 as val);end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: 1.+(2)

```
create procedure sum1()
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: op(*exp1*,...,*expn*)

```
create procedure sum1()
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: op(*exp1,...,expn*)

```
create procedure proc(op(exp1,...expn))
begin
 call const1();

 call const2();
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: op(*exp1,...,expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 create table sum1Tbl(val int);
 insert into sum1Tbl(val)
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: op(*exp1,...,expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 create table nameTable(columns(exp));
 insert into nameTable(columns(exp))
  select t0.val + t1.val
  from
   (select * from const1Tbl) as t0,
   (select * from const2Tbl) as t1;
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea exp la expresión: op(*exp1,...,expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 create table nameTable(columns(exp));
 insert into nameTable(columns(exp))
   query(op(exp1,...,expn));



end;
```

toProc: De expresiones OCL a procedimientos almacenados

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: 'hola'.characters()

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: 'hola'.characters()

```
create procedure characters1
begin
 call constH;
 call characters1Shp;
 create table characters1Tbl(pos int, val varchar...);
 insert into characters1Tbl(val)
  (select val
  from characters1ShpTbl)));


end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure characters1
begin
 call constH;
 call characters1Shp;
 create table characters1Tbl(pos int, val varchar...);
 insert into characters1Tbl(val)
  (select val
  from characters1ShpTbl)));


end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call constH;
 call characters1Shp;
 create table characters1Tbl(pos int, val varchar..);
 insert into characters1Tbl(val)
  (select val
  from characters1ShpTbl)));


end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 call characters1Shp;
 create table characters1(pos int, val varchar...);
 insert into characters1(val)
  (select val
  from characters1ShpTbl)));
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 call proc#(op(exp1,...,expn));
 create table characters1(pos int, val varchar...);
 insert into characters1(val)
  (select val
  from characters1ShpTbl)));
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 call proc#(op(exp1,...,expn));
 create table nameTable(columns(proc(exp)));
 insert into nameTable(columns(proc(exp)))
  (select val
  from characters1ShpTbl));
end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: op(*exp1*,...,*expn*)

```
create procedure proc(op(exp1,...,expn))
begin
 call proc(exp1);
 ...
 call proc(expn);
 call proc#(op(exp1,...,expn));
 create table nameTable(columns(proc(exp)));
 insert into nameTable(columns(proc(exp)))
  (select val
  from table(proc#(op(exp1,...,expn))));
end;
```

toProc: De expresiones OCL a procedimientos almacenados

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión

```
Trip.allInstances().origin->exists(o|o='Madrid')
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión

```
Trip.allInstances().origin->exists(o|o='Madrid')
```

```
create procedure existsIt
begin
 call origin;
 call existsItShp;
 create table existsItTbl(val Boolean);
 insert into existsItTbl(val)
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure existsIt
begin
 call origin;
 call existsItShp;
 create table existsItTbl(val Boolean);
 insert into existsItTbl(val)
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure proc(iter(fuente,cuerpo))
begin
 call origin;
 call existsItShp;
 create table existsItTbl(val Boolean);
 insert into existsItTbl(val)
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure proc(iter(fuente,cuerpo))
begin
 call proc(fuente);
 call existsItShp;
 create table existsItTbl(val Boolean);
 insert into existsItTbl(val)
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure proc(iter(fuente,cuerpo))
begin
 call proc(fuente);
 call proc#(iter(fuente,cuerpo));
 create table existsItTbl(val Boolean);
 insert into existsItTbl(val)
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure proc(iter(fuente,cuerpo))
begin
 call proc(fuente);
 call proc#(iter(fuente,cuerpo));
 create table nameTable(columns(exp));
 insert into nameTable(columns(exp))
   (select *
   from existsItShpTbl);



end;
```

toProc: De expresiones OCL a procedimientos almacenados

Sea *exp* la expresión: fuente->iter(o|cuerpo)

```
create procedure proc(iter(fuente,cuerpo))
begin
 call proc(fuente);
 call proc#(iter(fuente,cuerpo));
 create table nameTable(columns(exp));
 insert into nameTable(columns(exp))
   (select *
   from table(proc#(iter(fuente,cuerpo))));



end;
```

toProc: Procedimientos almacenados para iteradores

toProc: Procedimientos almacenados para iteradores

Sea *exp* la expresión: iter(*fuente*,cuerpo)

toProc: Procedimientos almacenados para iteradores

Sea *exp* la expresión: iter(*fuente*,cuerpo)

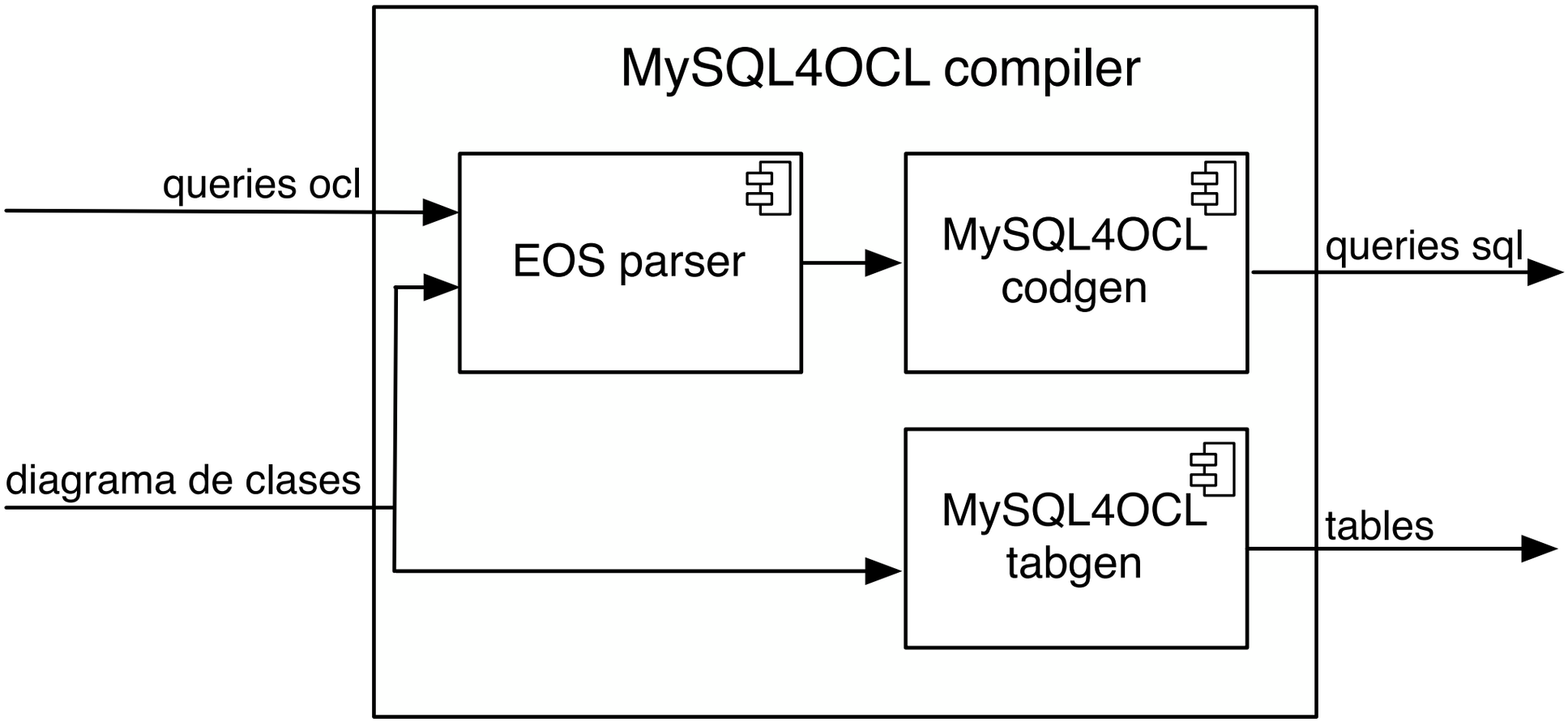
```
create procedure proc#(iter(fuente,cuerpo))
begin
 declare crs cursor for
    (select val from table(proc(fuente)));
 declare var, done ...;
 create table nameTable(columns(exp));
 open crs;
 repeat
   fetch crs into var;
   call proc(cuerpo);
   insert into nameTable(columns(exp)) ...;
 until done end repeat; close crs;
end;
```

toProc: Procedimientos almacenados para iteradores

Sea *exp* la expresión: iter(*fuente*,cuerpo)

```
create procedure proc#(iter(fuente,cuerpo))
begin
 declare crs cursor for
    (select val from table(proc(fuente)));
 declare var, done ...;
 create table nameTable(columns(exp));
 open crs;
 repeat
   fetch crs into var;
   call proc(cuerpo);
   insert into nameTable(columns(exp)) ...;
 until done end repeat; close crs;
end;
```

MySQL4OCL: Arquitectura



Optimizaciones

Optimizaciones

La creación de tablas tiene un coste significativo.

Optimizaciones

La creación de tablas tiene un coste significativo.

Creamos un algoritmo bottom up que elimina las llamadas a procedimientos (o las reemplaza por consultas), evitando la creación de tablas auxiliares siempre que esto es posible: o sea, cuando el cuerpo del procedimiento:

Optimizaciones

La creación de tablas tiene un coste significativo.

Creamos un algoritmo bottom up que elimina las llamadas a procedimientos (o las reemplaza por consultas), evitando la creación de tablas auxiliares siempre que esto es posible: o sea, cuando el cuerpo del procedimiento:

- no incluye ninguna llamada a procedimiento, sino que él mismo es una consulta, o
- incluye llamadas a procedimientos, pero éstas pueden realizarse "fuera" del mismo.

Ejemplo

Ejemplo

```
exp = 'hola'.characters()->size()
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;

create procedure characters1()
begin
 call proc(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin
 call proc(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select * from constHTbl) as t;
 end while;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin
 call proc(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select * from constHTbl) as t;
 end while;
end;
```

```
create procedure constH()
begin
 create table constHTbl(val int);
 insert into constHTbl(val)
   (select 'hola' as val);
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin
 call proc(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

```
create procedure constH()
begin
 create table constHTbl(val int);
 insert into constHTbl(val)
   (select 'hola' as val);
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin
 call proc(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

```
create procedure constH()
begin
 create table constHTbl(val int);
 insert into constHTbl(val)
   (select 'hola' as val);
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin
 call proc#(constH);
 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc(characters1);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1Tbl) as t;
end;
```

```
create procedure characters1()
begin

 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc#(characters1Shp);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1ShpTbl) as t;
end;
```

```
create procedure characters1()
begin

 call proc#(characters1Shp);
 create table
  characters1Tbl(val varchar(50));
 insert into characters1Tbl(val)
  (select * from characters1ShpTbl);
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc#(characters1Shp);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1ShpTbl) as t;
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

Ejemplo

exp = 'hola'.characters()->size()

```
create procedure size1()
begin
 call proc#(characters1Shp);
 create table size1Tbl(val int);
 insert into size1Tbl(val)
  select count(*)
  from (select *
    from characteres1ShpTbl) as t;
end;
```

```
create procedure characters1Shp()
begin
 declare ...
 create table characters1ShpTbl(pos
int, val varchar(250));
 ...
 while ... do
  insert into characters1ShpTbl(val)
   select substring(val,i,1) as val
   from (select 'hola' as val) as t;
 end while;
end;
```

MySQL4OCL en ActionGUI

MySQL4OCL en ActionGUI

- MySQL4OCL está integrado en la herramienta ActionGUI de desarrollo de aplicaciones web basadas en modelos.

MySQL4OCL en ActionGUI

- MySQL4OCL está integrado en la herramienta ActionGUI de desarrollo de aplicaciones web basadas en modelos.

- ActionGUI se ha utilizado para el desarrollo de un sistema de voluntariado para un hospital

Sistema de voluntariado en hospital

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Un voluntario puede leer la información de un compromiso (self) si es suyo y el voluntario está activo

self.tieneCompromiso.comoUsuario = caller and self.tieneCompromiso.estado = 1

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Un voluntario puede leer la información de un compromiso (self) si es suyo y el voluntario está activo

self.tieneCompromiso.comoUsuario = caller and self.tieneCompromiso.estado = 1

Modelo de GUI:

- 92 ventanas, 263 botones, 657 etiquetas, 161 campos de datos, 108 tablas.
- 2697 expresiones OCL (condiciones y argumentos de acciones, valores para mostrar).

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Un voluntario puede leer la información de un compromiso (self) si es suyo y el voluntario está activo

self.tieneCompromiso.comoUsuario = caller and self.tieneCompromiso.estado = 1

Modelo de GUI:

- 92 ventanas, 263 botones, 657 etiquetas, 161 campos de datos, 108 tablas.
- 2697 expresiones OCL (condiciones y argumentos de acciones, valores para mostrar).

Sesiones de cursos que comienzan en una fecha específica

[cursos.row].sesiones->select(s|s.fechaInicio.date() = [agendaGrupo.fecha].date())

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Un voluntario puede leer la información de un compromiso (self) si es suyo y el voluntario está activo

self.tieneCompromiso.comoUsuario = caller and self.tieneCompromiso.estado = 1

Modelo de GUI:

- 92 ventanas, 263 botones, 657 etiquetas, 161 campos de datos, 108 tablas.
- 2697 expresiones OCL (condiciones y argumentos de acciones, valores para mostrar).

Sesiones de cursos que comienzan en una fecha específica

[cursos.row].sesiones->select(s|s.fechaInicio.date() = [agendaGrupo.fecha].date())

- MySQL4OCL generó

Sistema de voluntariado en hospital

Modelo de datos: 22 clases y 34 asociaciones

Modelo de seguridad: 46 restricciones de acceso (en OCL)

Un voluntario puede leer la información de un compromiso (self) si es suyo y el voluntario está activo

self.tieneCompromiso.comoUsuario = caller and self.tieneCompromiso.estado = 1

Modelo de GUI:

- 92 ventanas, 263 botones, 657 etiquetas, 161 campos de datos, 108 tablas.
- 2697 expresiones OCL (condiciones y argumentos de acciones, valores para mostrar).

Sesiones de cursos que comienzan en una fecha específica

[cursos.row].sesiones->select(s|s.fechaInicio.date() = [agendaGrupo.fecha].date())

- MySQL4OCL generó

4434 procedimientos almacenados (46647 líneas de código MySQL).

Conclusiones

Conclusiones

- Presentamos un compilador de OCL a MySQL que permite la evaluación de expresiones OCL en base de datos.

  superando la limitación actual de los evaluadores OCL sobre escenarios grandes

Conclusiones

- Presentamos un compilador de OCL a MySQL que permite la evaluación de expresiones OCL en base de datos.

  superando la limitación actual de los evaluadores OCL sobre escenarios grandes

- Se define de manera recursiva sobre las expresiones

  superando las limitaciones de las propuestas anteriores basadas en patrones. (OCL2SQL de DresdenOCL)

Conclusiones

- Presentamos un compilador de OCL a MySQL que permite la evaluación de expresiones OCL en base de datos.

  superando la limitación actual de los evaluadores OCL sobre escenarios grandes

- Se define de manera recursiva sobre las expresiones

  superando las limitaciones de las propuestas anteriores basadas en patrones. (OCL2SQL de DresdenOCL)

- Las construcciones utilizadas son soportadas por otros gestores de base de datos.

Trabajo futuro

Trabajo futuro

- Demostrar formalmente la corrección de nuestro compilador y del algoritmo de optimización.

Trabajo futuro

- Demostrar formalmente la corrección de nuestro compilador y del algoritmo de optimización.
- Extender el compilador MySQL4OCL para dar soporte a operaciones sobre colección de colecciones, tuplas y funciones definidas por el usuario.

Trabajo futuro

- Demostrar formalmente la corrección de nuestro compilador y del algoritmo de optimización.
- Extender el compilador MySQL4OCL para dar soporte a operaciones sobre colección de colecciones, tuplas y funciones definidas por el usuario.
- Optimizar el código generado con un análisis de pre-compilación y uno de post-compilación.

Trabajo futuro

- Demostrar formalmente la corrección de nuestro compilador y del algoritmo de optimización.
- Extender el compilador MySQL4OCL para dar soporte a operaciones sobre colección de colecciones, tuplas y funciones definidas por el usuario.
- Optimizar el código generado con un análisis de pre-compilación y uno de post-compilación.
- Integrar el compilador con una política de seguridad de forma que la consulta/procedimiento resultado tenga en cuenta esta política.

Preguntas?

Publicaciones

- Marina Egea, Carolina Dania, Manuel Clavel: MySQL4OCL: A Stored Procedure-Based MySQL Code Generator for OCL. Workshop OCL. ECEASST 36: (2010)
- Miguel Angel García de Dios, Carolina Dania, Michael Schläpfer, David A. Basin, Manuel Clavel, Marina Egea: SSG: A Model-Based Development Environment for Smart, Security-Aware GUIs. ICSE (2) 2010: 311-312
- David A. Basin, Manuel Clavel, Marina Egea, Miguel Angel García de Dios, Carolina Dania, Gonzalo Ortiz, Javier Valdazo: Model-Driven Development of Security-Aware GUIs for Data-Centric Applications. FOSAD 2011: 101-124
- Disponible en: http://www.bm1software.com/mysql-ocl/