SQL-PL4OCL: an automatic code generator from OCL to SQL procedural language

Marina Egea and Carolina Dania

Sep. 19, 2017

MoDELS. Austin, Texas

Outline

OCL as a query language
Motivation

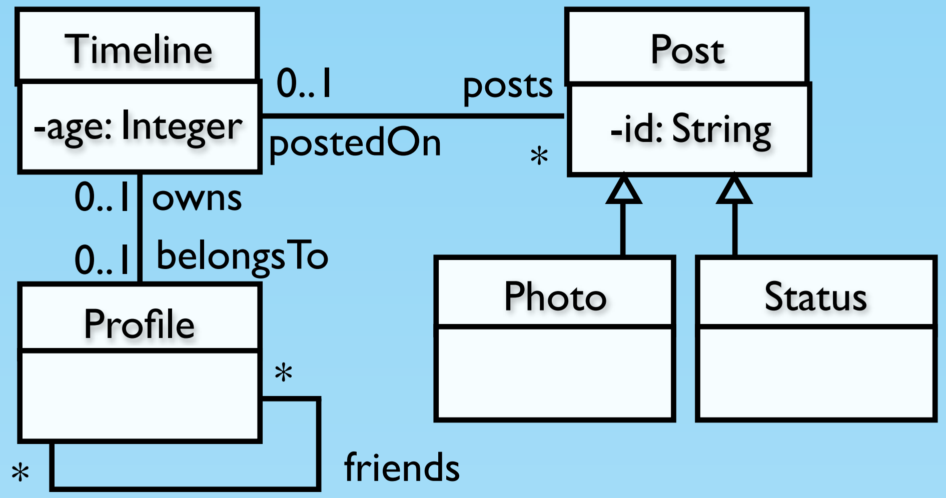
- Evaluation of OCL expression on medium/large scenarios.
- Integration of OCL expressions (invariants/queries) into an automated code generation process where the persistent layer are SQL/PL databases

Background

UML (Unified Modeling Language)
Ex. Social Network

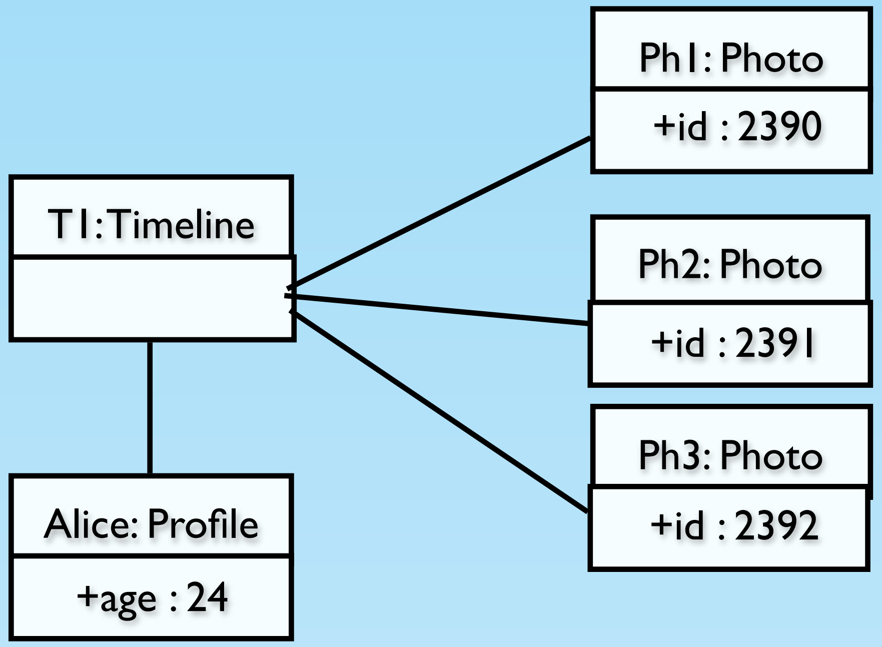
Class diagram

- classes
- attributes
- associations (association-ends)
- inheritance



Object diagram

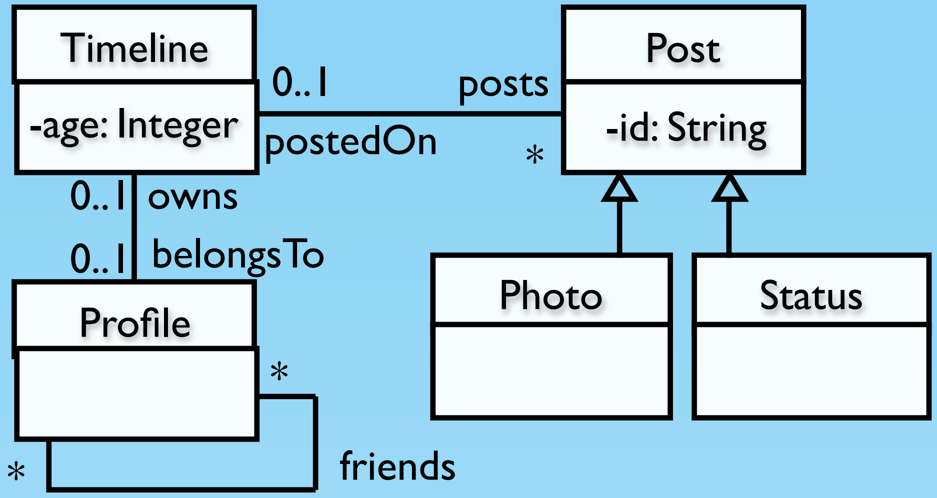
- objects
- values
- links



OCL (Object Constraint Language)

- It is a general-purpose (textual) formal language that allows:
  - retrieve objects and their values
  - navigate through related objects
- It supports a set of types with a set of operations over them, and
  - primitive types (Integer, String, Boolean), and
  - collection types (Set, Bag, OrderedSet, and Sequence), and
  - operators like: +, -, >, <, size, isEmpty, notEmpty, characters, and
  - iterators like: forAll, exists, collect

OCL (Object Constraint Language)



- All instances of Timeline

  Timeline.allInstances()

- Number of instances

  Timeline.allInstances()->size()

- Every profile is older than 18 years old

  Profile.allInstances()->forAll(p|p.age > 18)

- There isn't any profile older than 18

  Profile.allInstances()->select(p|p.age > 18)->isEmpty()

- Convert the string 'hi' in a sequence of characters

  'hi'.characters()

Databases

- Structured Query language (SQL)
- RBMS: MySQL, MariaDB, PostgreSQL, and MS SQL.

1.queries

```
select * from Photo

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

Databases

- Structured Query language (SQL)
- RBMS: MySQL, MariaDB, PostgreSQL, and MS SQL.

1.queries 2.sentences

```
create temporary table Photo(pk Int);

insert into Photo(val) (select pk from Photo);
```

Databases

- Structured Query language (SQL)
- RBMS: MySQL, MariaDB, PostgreSQL, and MS SQL.

1.queries

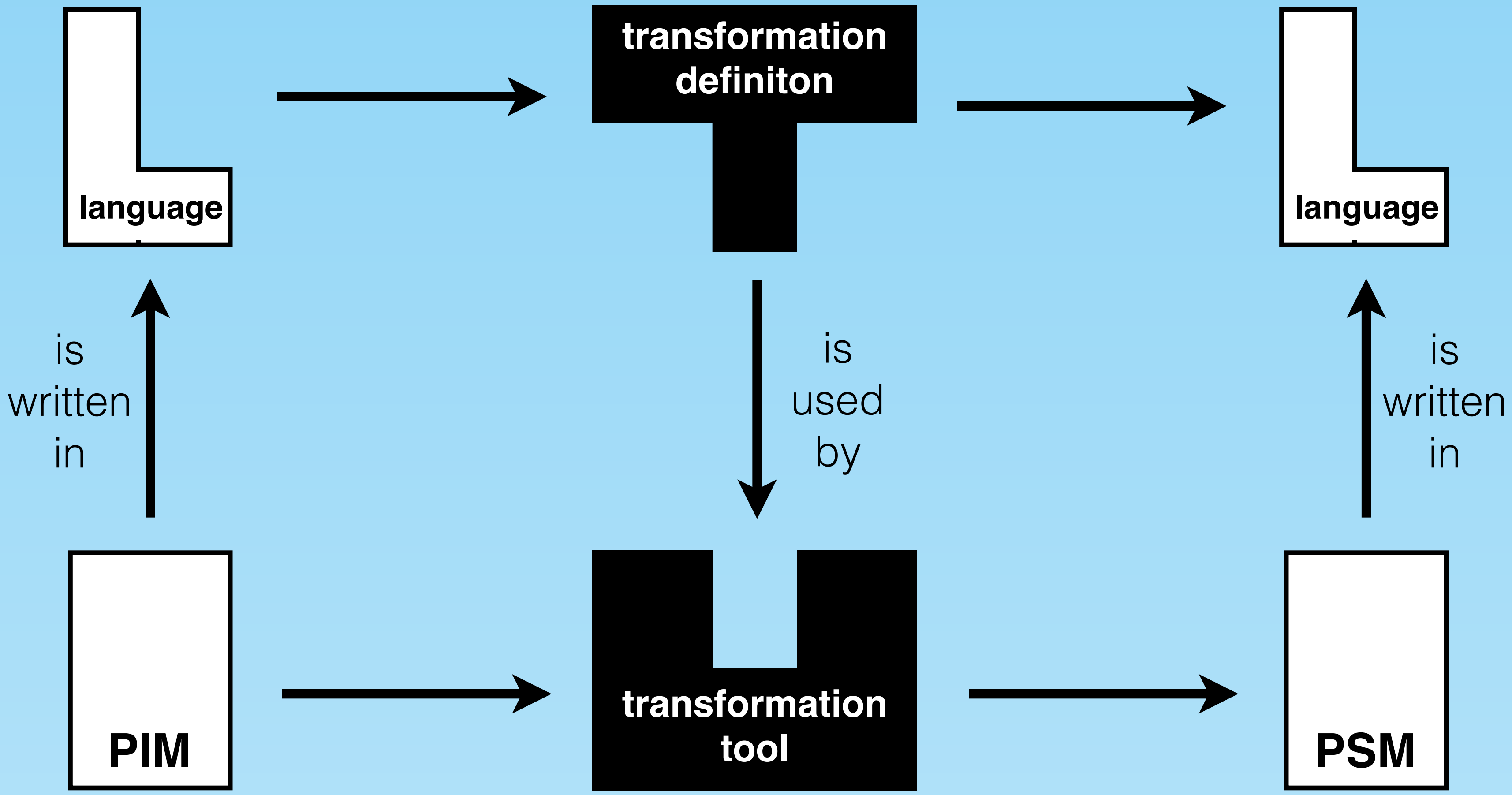
2.sentences

3.store procedures
- cursors
- conditionals
- loops

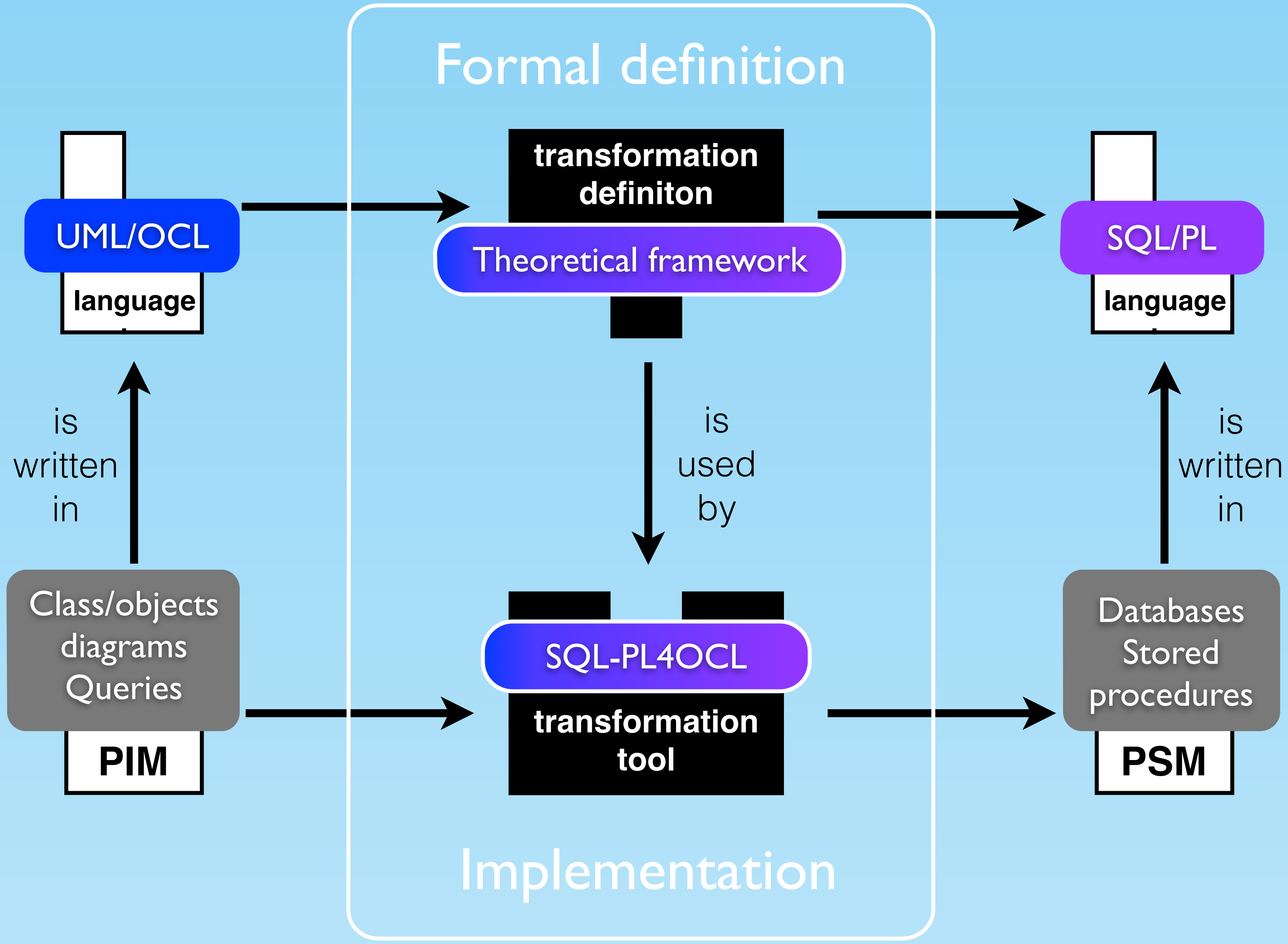
```
declare procedure nameProc
begin
 ...
end;
call nameProc;
```

Mapping OCL to SQL-PL

Mapping OCL to SQL-PL

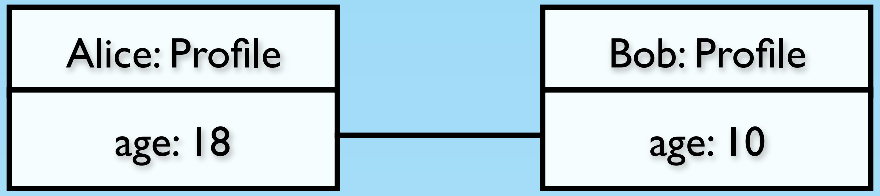


Mapping OCL to SQL-PL



From OCL to SQL-PL
Mapping data/object models

- a table with a column for each class
- a column for each attribute
- a table with two columns for each association



Object model

- a row for each object in the table associated with the class
- a row for each link in the corresponding table

table: Profile

pk	age
1	18
2	10

table: friendship

myFriends	friendsOf
1	2

From OCL to SQL-PL
Mapping OCL expressions

Every expression is mapped into a stored procedure

```
create procedure name
begin

    OCL to SQL-PL expression

end;//
call name()//
```

The mapping is recursive over the expression.

Depending on the complexity of the OCL expressions, they are mapped:

- into a SQL query
- into a SQL query and need an auxiliary block definition

From OCL to SQL-PL
Mapping OCL expressions (cont.)

- Expressions that are mapping into a SQL query

```
Timeline.allInstances()

select Timeline.pk as val
from Timeline
```

```
create procedure name
begin

                                   ;

end; //
call name(); //
```

From OCL to SQL-PL
Mapping OCL expressions (cont.)

- Expressions that are mapping into a SQL query

```
Timeline.allInstances()
```

```
create procedure name
begin



    select Timeline.pk as val
    from Timeline                ;
end; //
call name(); //
```

From OCL to SQL-PL

Mapping OCL expressions (cont.)

- Expressions that are mapping into a SQL query

Timeline.allInstances()

```
select Timeline.pk as val
from Timeline
```

Timeline.allInstances()->size()

```
select count(t1.val) as val
from
  (
                         ) as t1
```

```
create procedure name
begin

                                             ;

end; //
call name(); //
```

From OCL to SQL-PL

Mapping OCL expressions (cont.)

- Expressions that are mapping into a SQL query

```
Timeline.allInstances()
```

```
create procedure name
begin
```

```
Timeline.allInstances()->size()
```

```
select count(t1.val) as val
from
  ( select Timeline.pk as val
    from Timeline ) as t1
```

```
;
end; //
call name(); //
```

From OCL to SQL-PL
Mapping OCL expressions (cont.)

- Expressions that are mapping into a SQL query

```
Timeline.allInstances()

Timeline.allInstances()->size()
```

```
create procedure name
begin
 select count(t1.val) as val
 from
   (select Timeline.pk as val
    from Timeline  ) as t1     ;
end; //
call name(); //
```

From OCL to SQL-PL

Mapping OCL expressions (cont.)

- Expressions that are mapped into a SQL query and need an auxiliary block definition

’hi’.characters()

pos	val
1	h
2	i

```
create procedure name
begin
   begin
      drop table if exists wchars;
      create temporary table wchars (pos int not null auto increment,
         val varchar(250), primary key(pos));
      insert into wchars(val) (select ’h’ as val);
      insert into wchars(val) (select ’i’ as val);
   end;
   select val from wchars order by pos;
end;//
```

From OCL to SQL-PL

Intermediate tables and queries

	Primitive types, sets, and bags	OrderedSets and sequences
Tables	**create temporary table** *name* (val ***type*****);**	**create temporary table** *name* (val ***type*****,** pos **int not null auto increment, primary key(**pos**));**
Queries	**select** val **from** *name;*	**select** val **from** *name* **order by** pos;

From OCL to SQL-PL
Mapping OCL expressions (cont.)

'hi'.characters()−>union('ho'.characters())

```
create procedure name
begin
 begin
    begin
      drop table if exists wchars1;
      create temporary table wchars1 (pos int not null auto increment, val varchar(250), primary key(pos));
      insert into wchars1(val) (select 'h' as val);
      insert into wchars1(val) (select 'i' as val);
    end;
    begin
     drop table if exists wchars2;
     create temporary table wchars2 (pos int not null auto increment, val varchar(250), primary key(pos));
     insert into wchars2(val) (select 'h' as val);
     insert into wchars2(val) (select 'o' as val);
   end;
   create temporary table union(pos int NOT NULL auto_increment, val varchar(250), primary key (pos));
   insert into union(val)
     (select t1.val as val  from (select val from wchars1 order by pos asc) as t1);
   insert into union(val)
     (select t1.val as val from (select val from wchars2 order by pos asc) as t1);
 end;
 select val from union order by pos;
end;//
call name();//
```

From OCL to SQL-PL

Mapping OCL expressions (cont.)

‘hi‘.characters()−>union(‘ho‘.characters())

‘hi‘.characters()

‘ho‘.characters()

```
create procedure name
begin
 begin
   begin
    drop table if exists wchars1;
    create temporary table wchars1 (pos int not null auto increment, val varchar(250), primary key(pos));
    insert into wchars1(val) (select ’h’ as val);
    insert into wchars1(val) (select ’i’ as val);
   end;
   begin
    drop table if exists wchars2;
    create temporary table wchars2 (pos int not null auto increment, val varchar(250), primary key(pos));
    insert into wchars2(val) (select ’h’ as val);
    insert into wchars2(val) (select ’o’ as val);
  end;
  create temporary table union(pos int NOT NULL auto_increment, val varchar(250), primary key (pos));
  insert into union(val)
    (select t1.val as val  from (select val from wchars1 order by pos asc) as t1);
  insert into union(val)
    (select t1.val as val from (select val from wchars2 order by pos asc) as t1);
 end;
 select val from union order by pos;
end;//
call name();//
```

From OCL to SQL-PL

Structures in Store Procedures

```
create procedure name
begin
  begin
    begin
      ...
      ...
     ...
    end; //
  end; //
end; //
call name(); //
```

Nested blocks structure

```
create procedure name
begin
  begin
    ...
  end; //
  begin
    ...
  end; /
  ...
end; //
call name(); //
```

Sequencial blocks structure

From OCL to SQL-PL

Iterators

src—>it(body)

```
begin
  declare done int default 0;
  declare var;
  declare crs cursor for (cursor-specific type - src );
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists blq_name;
  create temporary table blq_name (value-specif type)
  open crs;
  repeat
   fetch crs into var;
  if not done then
   Iterator-specific body query
   Iterator-specific processing code
   end if;
 until done end repeat;
 close crs;
end;//
```

From OCL to SQL-PL

Iterators (cont.)

Profile.allInstances()–>forAll(p|p.age > 18)

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
    select val into tempResult from (select tbl2.val > tbl3.val as val
    from (select Person.age as val from Person, (select var1 as val) as tbl1
    where pk = tbl1.val) as tbl2,
    (select 18 as val) as tbl3) as tbl5;
    if not tempResult or tempResult is null then
      set done = 1;
      set result = 0;
    end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

From OCL to SQL-PL

Iterators (cont.)

$\text{Profile.allInstances()} \rightarrow \text{forAll}(p | p.age > 18)$

variables

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
    select val into tempResult from (select tbl2.val > tbl3.val as val
    from (select Person.age as val from Person, (select var1 as val) as tbl1
    where pk = tbl1.val) as tbl2,
    (select 18 as val) as tbl3) as tbl5;
    if not tempResult or tempResult is null then
      set done = 1;
      set result = 0;
    end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

From OCL to SQL-PL

Iterators (cont.)

Profile.allInstances()−>forAll(p|p.age > 18)

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;  cursor-specific type - src
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
    select val into tempResult from (select tbl2.val > tbl3.val as val
    from (select Person.age as val from Person, (select var1 as val) as tbl1
    where pk = tbl1.val) as tbl2,
    (select 18 as val) as tbl3) as tbl5;
    if not tempResult or tempResult is null then
       set done = 1;
       set result = 0;
    end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

From OCL to SQL-PL

Iterators (cont.)

Profile.allInstances()–>forAll(p|p.age > 18)

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
   select val into tempResult from (select tbl2.val > tbl3.val as val
   from (select Person.age as val from Person, (select var1 as val) as tbl1
   where pk = tbl1.val) as tbl2,
   (select 18 as val) as tbl3) as tbl5;
   if not tempResult or tempResult is null then
     set done = 1;
     set result = 0;
   end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

temporary table

From OCL to SQL-PL

Iterators (cont.)

$\text{Profile.allInstances()}{-}{>}\text{forAll}(p|p.\text{age} > 18)$

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool); value specific-type
  open crs;
  repeat
  fetch crs into var1;
  if not done then
   select val into tempResult from (select tbl2.val > tbl3.val as val
   from (select Person.age as val from Person, (select var1 as val) as tbl1
   where pk = tbl1.val) as tbl2,
   (select 18 as val) as tbl3) as tbl5;
   if not tempResult or tempResult is null then
     set done = 1;
     set result = 0;
   end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

From OCL to SQL-PL

Iterators (cont.)

Profile.allInstances()–>forAll(p|p.age > 18)

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
   select val into tempResult from (select tbl2.val > tbl3.val as val
   from (select Person.age as val from Person, (select var1 as val) as tbl1
   where pk = tbl1.val) as tbl2,
   (select 18 as val) as tbl3) as tbl5;
   if not tempResult or tempResult is null then
     set done = 1;
     set result = 0;
   end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

Iterator-specific body query

From OCL to SQL-PL

Iterators (cont.)

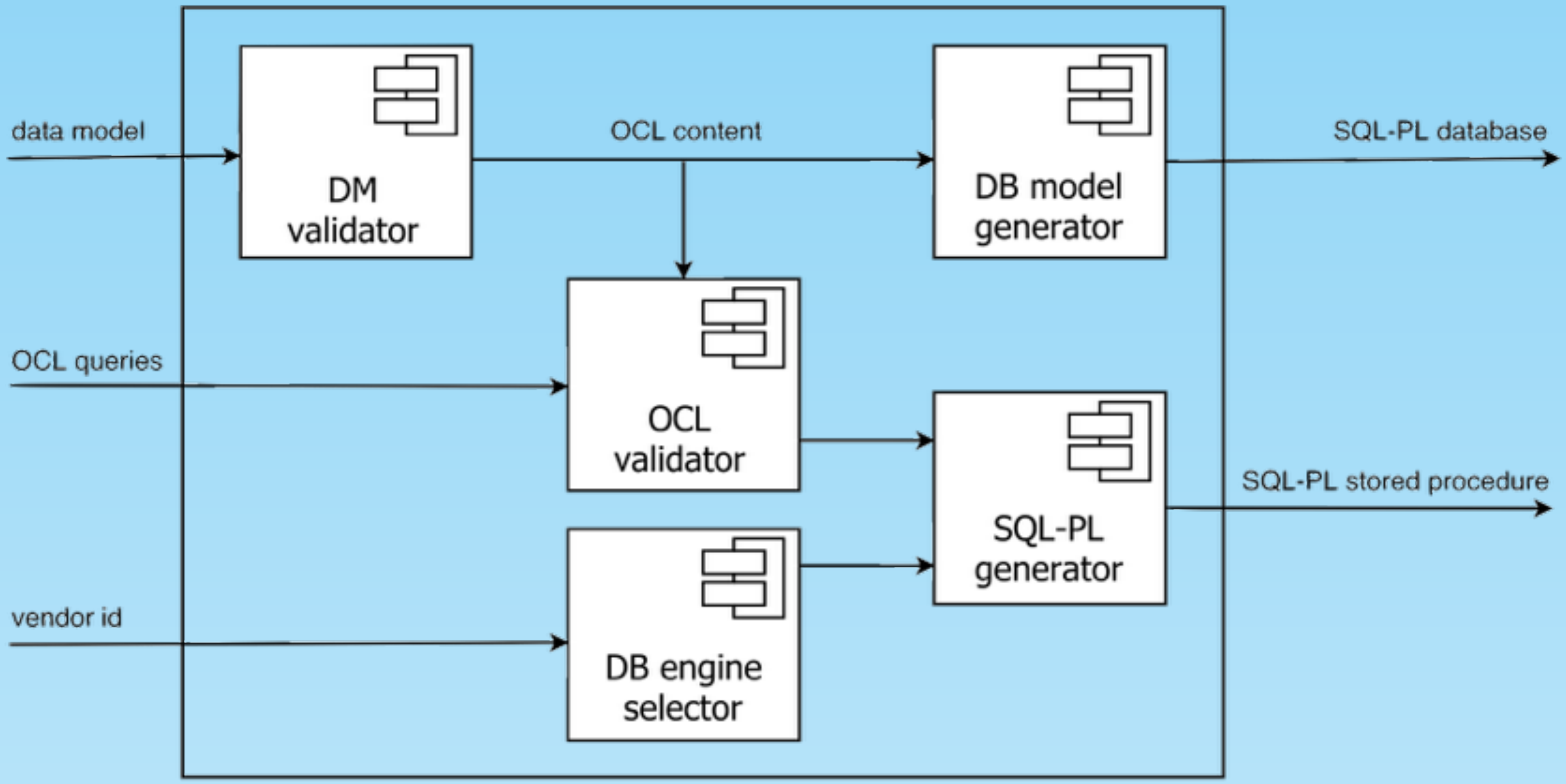
Profile.allInstances()–>forAll(p|p.age > 18)

```
create procedure forAll()
begin
 begin
  declare done int default 0 ;
  declare result boolean default true;
  declare tempResult int default 0;
  declare var1 int;
  declare crs cursor for select pk as val from Person;
  declare continue handler for sqlstate '02000' set done = 1;
  drop table if exists forAll;
  create temporary table forAll(val bool);
  open crs;
  repeat
  fetch crs into var1;
  if not done then
   select val into tempResult from (select tbl2.val > tbl3.val as val
   from (select Person.age as val from Person, (select var1 as val) as tbl1
   where pk = tbl1.val) as tbl2,
   (select 18 as val) as tbl3) as tbl5;
   if not tempResult or tempResult is null then
     set done = 1;
     set result = 0;
   end if;
  end if;
  until done end repeat;
  insert into forAll(val) (select result as val);
  close crs;
 end;
 select val from forAll;
end;//
```

Iterator-specific processing

SQL-PL4OCL

tool component architecture



SQL-PL4OCL

Benchmark

- Vendor specific supported: MySQL/MariaDB, PostgreSQL, SQL Server DBMS
- MariaBD works faster in most of the cases

	MySQL	MariaDB	PostgreSQL	MSSQL
Q1	0.19s	0.13s	**0.10s**	0.12s
Q2	0.25s	**0.20s**	0.33s	0.28s
Q3	0.36s	0.35s	0.27s	**0.26s**
Q4	**0.04s**	**0.04s**	**0.04s**	0.05s
Q5	0.55s	**0.40s**	0.40s	0.42s
Q6	1.05s	**0.55s**	1.06s	1.03s
Q7	2.07s	**1.56s**	1.99s	2.08s
Q8	50.02s	**43.08s**	57.04s	53.47s
Q9	9.14s	**8.00s**	8.18s	8.89s
Q10	0.05s	**0.04s**	0.07s	0.05s
Q11	49.56s	**40.02s**	40.10s	43.46s
Q12	59.58s	**51.23s**	51.25s	54.82s
Q13	**1.67s**	1.98s	2.35s	1.90s
Q14	59.52s	**54.33s**	63.35s	58.33s

Related work
(comparison with OCL2SQL-DresdenOCL)

OCL pattern

```
context: Class
inv: OCL boolean expression
```

MySQL pattern

```
select *
from Class
where not OCL2SQL(OCL boolean expression)
```

OCL2SQL mapping is based on patterns and it does not support iterators.

Conclusions

- Code-generator from OCL queries to the procedural language extensions of SQL (SQL-PL)
  - each OCL expression is mapped to a single stored procedure
  - temporary tables are used
  - the three-valued evaluation semantics of OCL is considered

Future work

- Look for the integration of developed tools into CASE tools
- Empirical validation of the usefulness of the approach for a software engineering team.

Questions?

http://software.imdea.org/~dania/