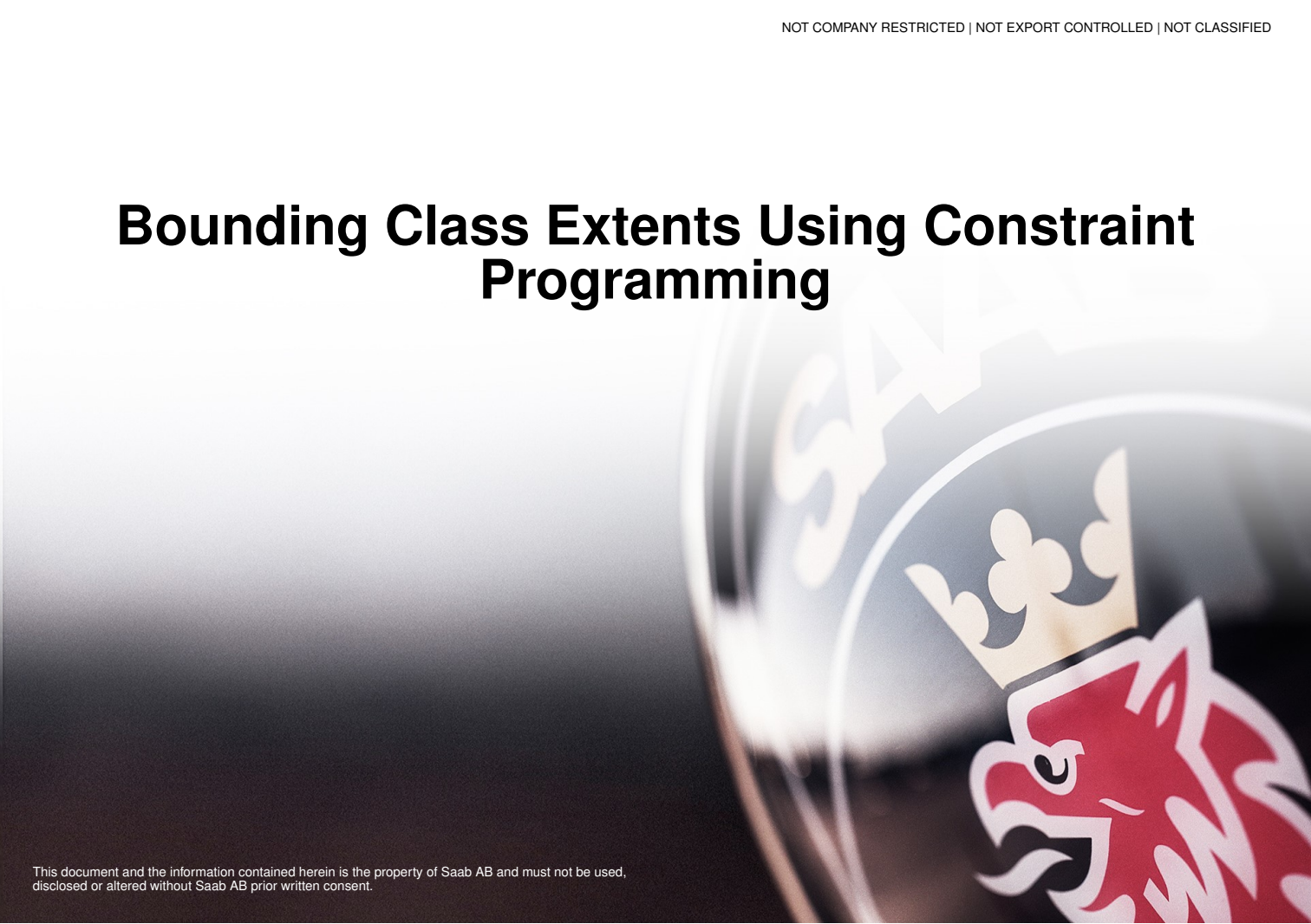
Bounding Class Extents Using Constraint Programming



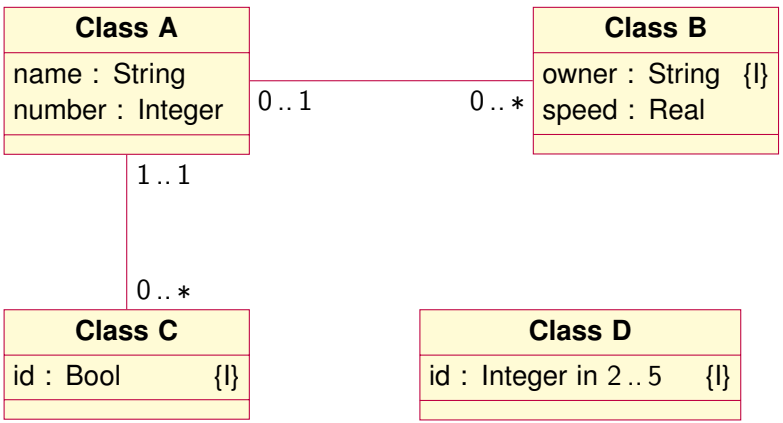
OUTLINE

What?

How?

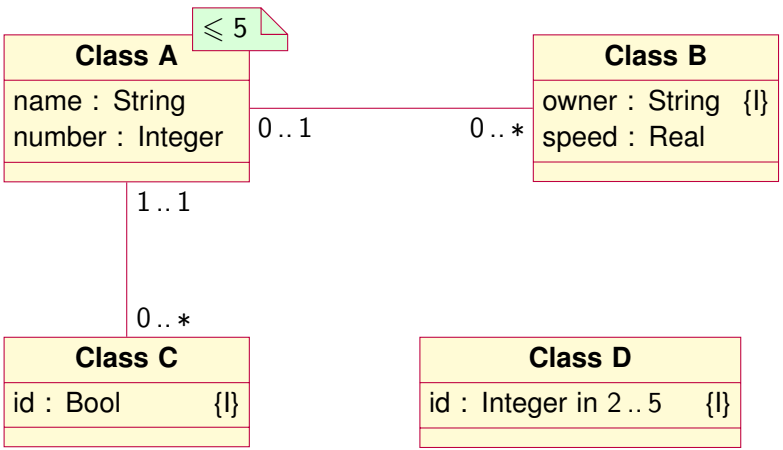
Doing better

BOUNDING RESOURCES



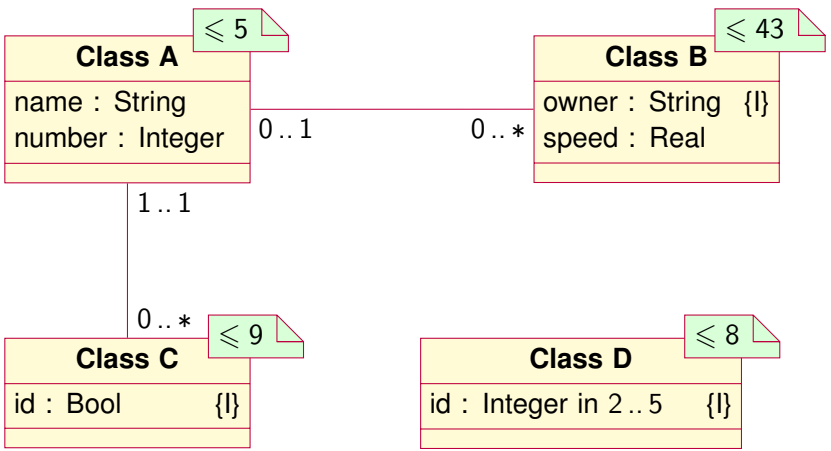
For reasons (predictability, reliability, safety, etc.) one might want to allocate memory on initialization, and not during execution.
To do this we need to know how much memory we are going to need.

BOUNDING RESOURCES



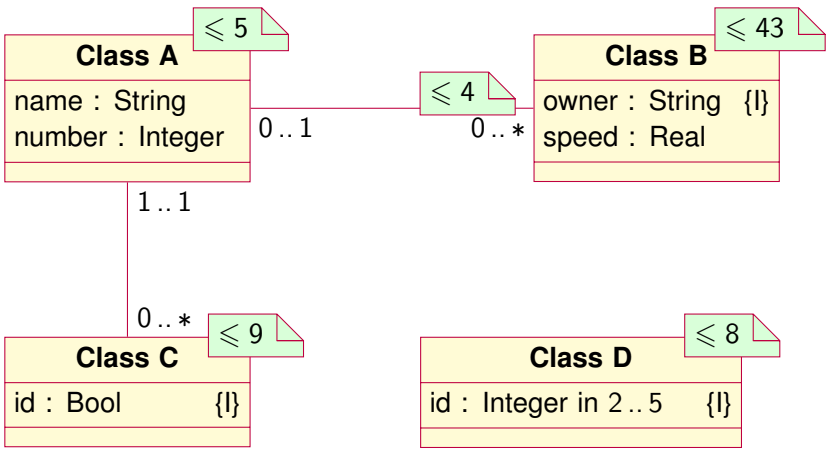
One way to obtain a bound is to annotate the model with information about maximum number of instances and links.

BOUNDING RESOURCES



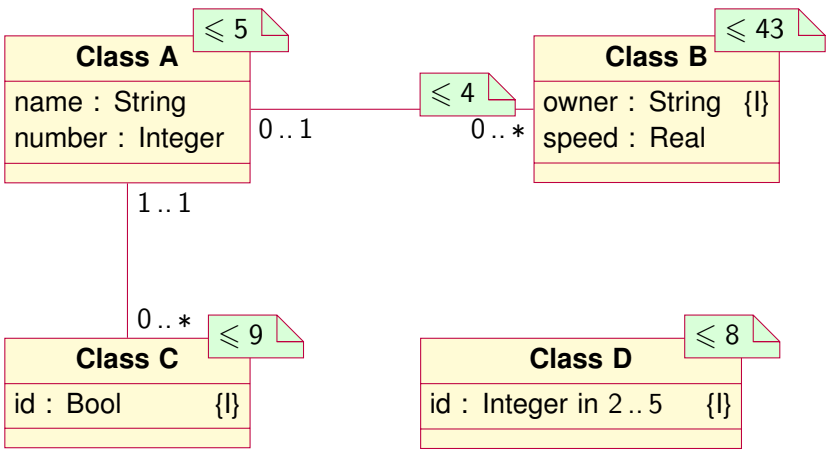
One way to obtain a bound is to annotate the model with information about maximum number of instances and links.

BOUNDING RESOURCES



One way to obtain a bound is to annotate the model with information about maximum number of instances and links.

BOUNDING RESOURCES



One way to obtain a bound is to annotate the model with information about maximum number of instances and links.

Want to use information in the model to assist with this marking.

STRATEGY

Extract constraints

Solve optimization problem

STRATEGY

Extract constraints

Solve optimization problem

This appears to be an approach that nicely fit many operations on models.

It offers a very good decoupling of the information retrieval part
(which can be expressed in the model in a very natural way),
and the computation part (for which external general-purpose tools can be used).

STRATEGY

Our constraint problem

We create a constraint problem with variables representing *upper bounds* for the quantities we are interested in.

The variables are:

c_X
Maximum number of objects of class X

r_R
Maximum number of object pairs associated over relation R

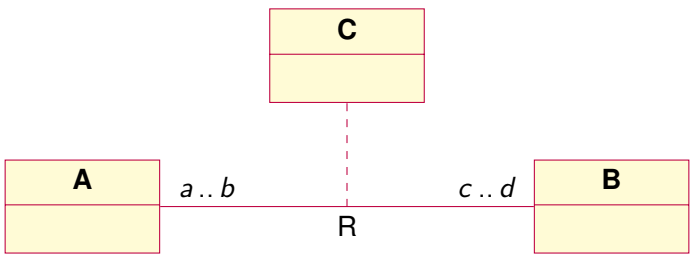
$s_{A,R,B}$
Maximum number of objects of class B that can be associated to an object of class A over relation R

CONSTRAINTS

$$d \cdot c_A \geqslant r_R \quad \text{if } d \neq \infty$$
$$c \cdot c_A \leqslant r_R \quad \text{if } c \neq 0$$
$$a \cdot c_B \leqslant r_R \quad \text{if } a \neq 0$$
$$b \cdot c_B \geqslant r_R \quad \text{if } b \neq \infty$$
$$s_{A,R,B} \cdot c_A \geqslant r_R$$
$$s_{B,R,A} \cdot c_B \geqslant r_R$$

$$s_{A,R,B} \leqslant d \quad \text{if } d \neq \infty$$
$$s_{A,R,B} \cdot c_A \geqslant c \cdot c_A \quad \text{if } c \neq 0$$
$$s_{B,R,A} \cdot c_B \geqslant a \cdot c_B \quad \text{if } a \neq 0$$
$$s_{B,R,A} \leqslant b \quad \text{if } b \neq \infty$$
$$s_{A,R,B} \leqslant c_B$$
$$s_{B,R,A} \leqslant c_A$$

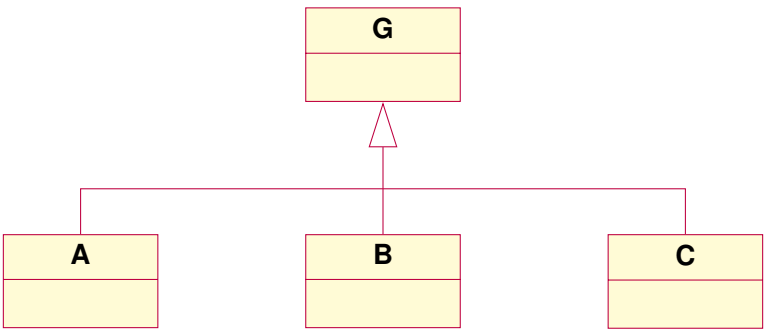
CONSTRAINTS



$$c_C \leqslant r_R$$
$$c_C \geqslant r_R$$
$$s_{A,R,C} \leqslant s_{A,R,B}$$
$$s_{A,R,C} \geqslant s_{A,R,B}$$
$$s_{A,R,C} \cdot c_A \geqslant r_R$$
$$s_{B,R,C} \cdot c_B \geqslant r_R$$

$$s_{C,R,A} \leqslant 1$$
$$s_{C,R,B} \leqslant 1$$
$$s_{B,R,C} \leqslant s_{B,R,A}$$
$$s_{B,R,C} \geqslant s_{B,R,A}$$
$$s_{A,R,C} \leqslant c_C$$
$$s_{B,R,C} \leqslant c_C$$

CONSTRAINTS



$$c_A \leqslant c_G$$
$$c_B \leqslant c_G$$
$$c_C \leqslant c_G$$
$$c_G \leqslant c_A + c_B + c_C$$

SOLVER

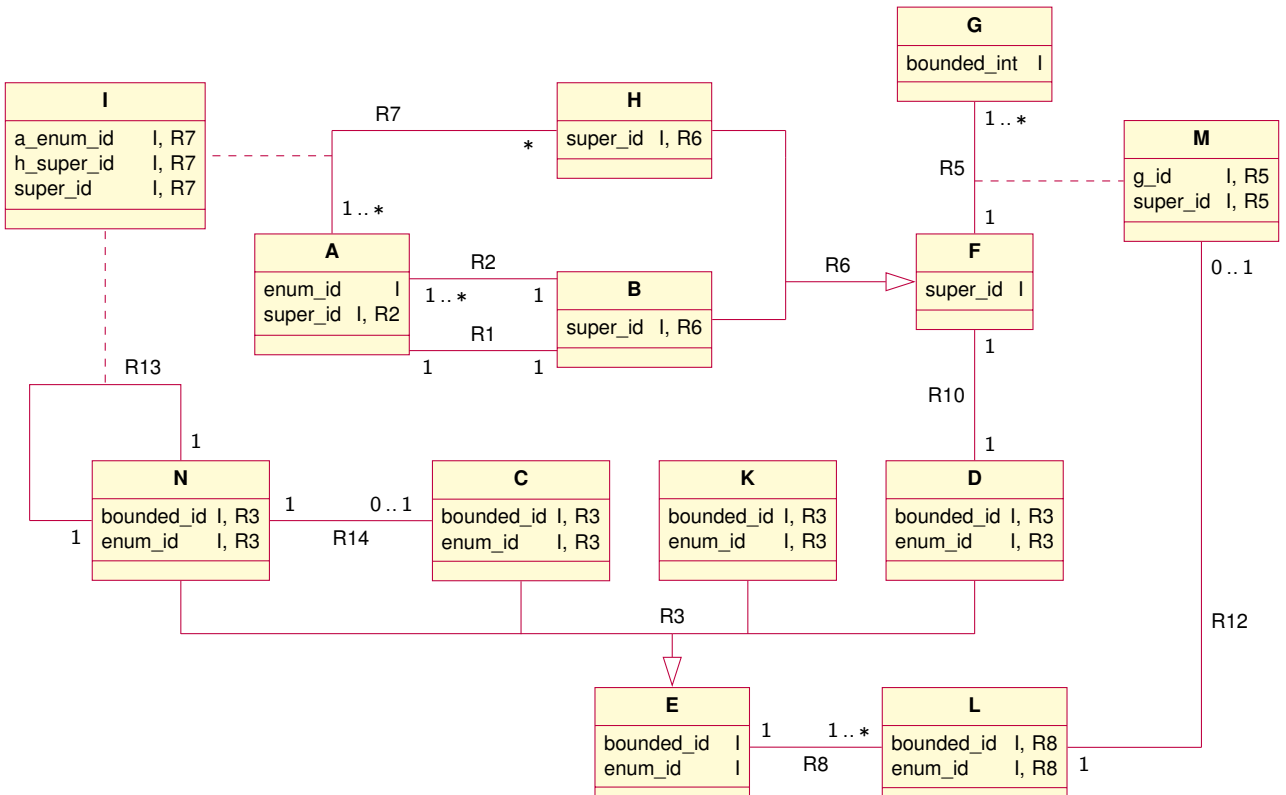
We could easily use an external solver

However, our problem is very simple

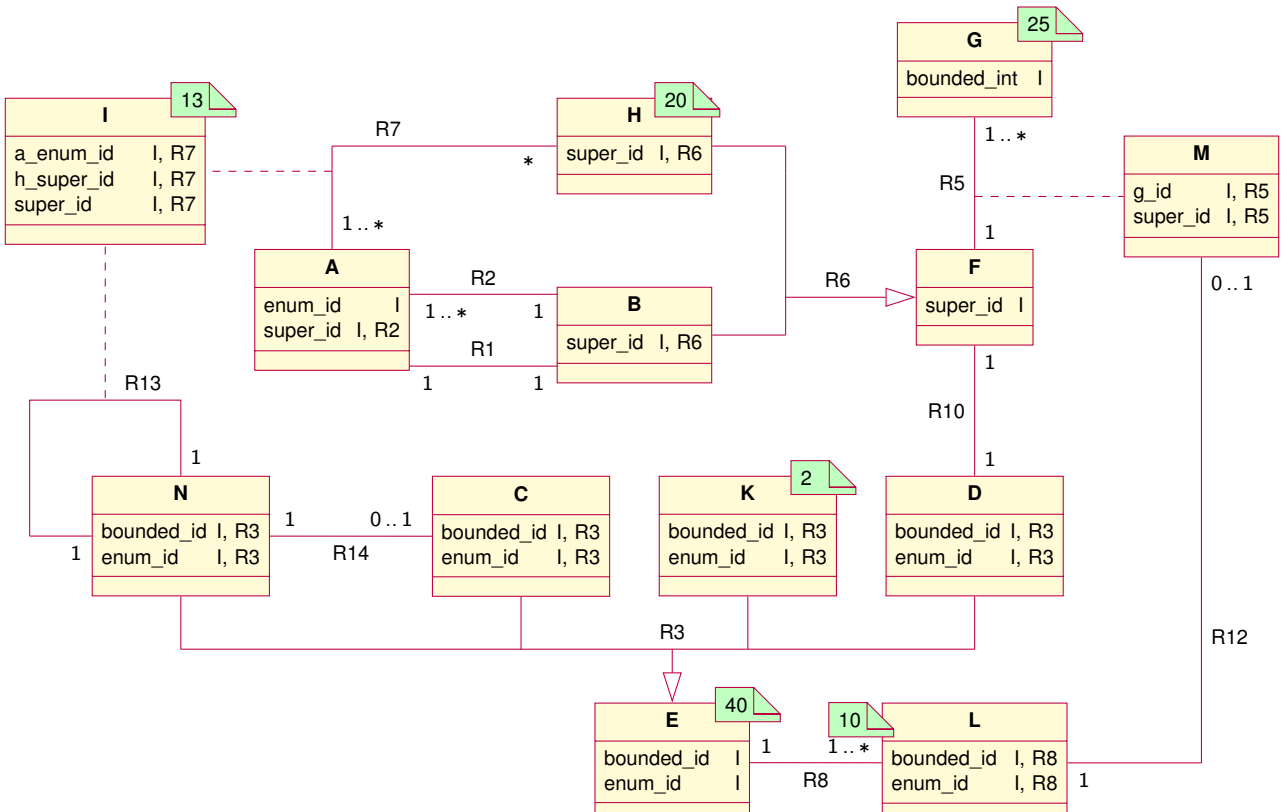
Can be solved with the following method:

Initialize all variables to infinity,
Find a violated constraint,
Satisfy the constraint by lowering the value of one variable,
Repeat the above two steps until all constraints satisfied

EXAMPLE

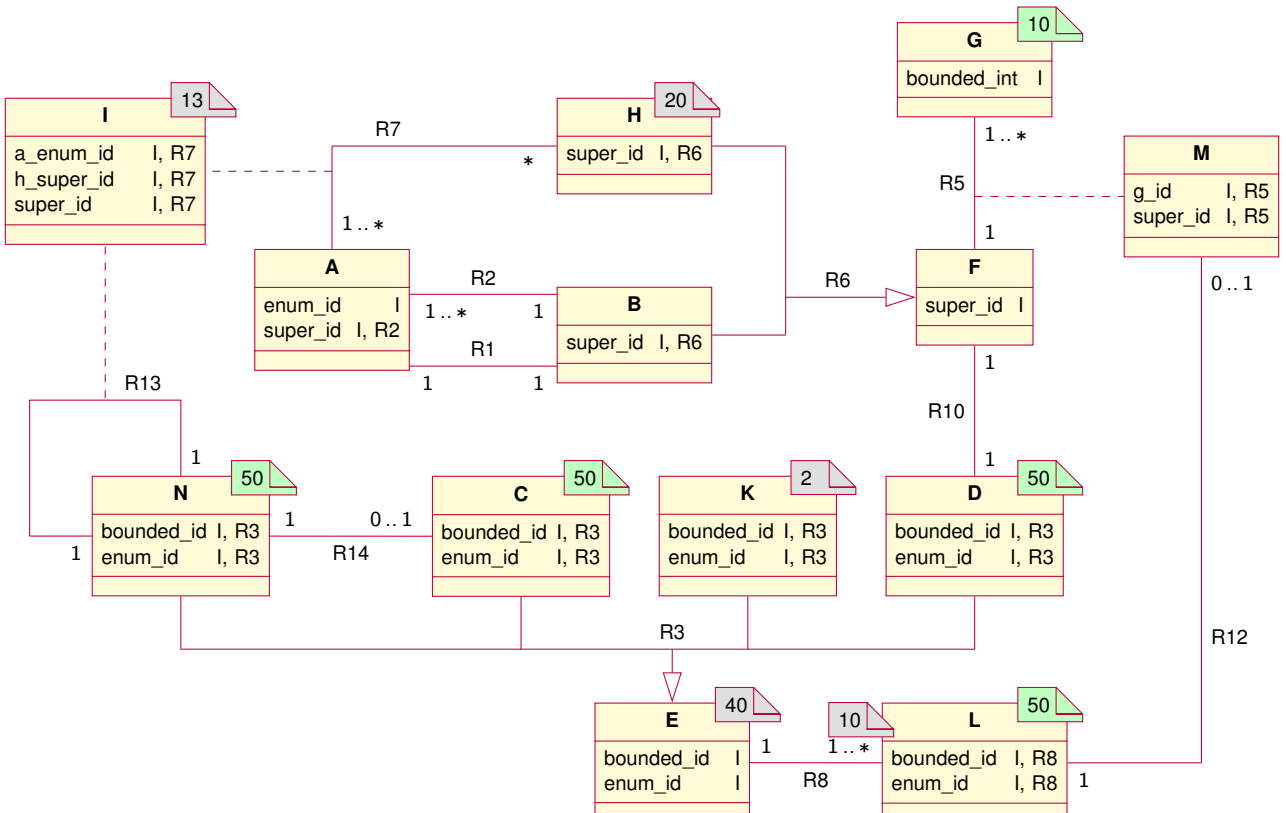


EXAMPLE



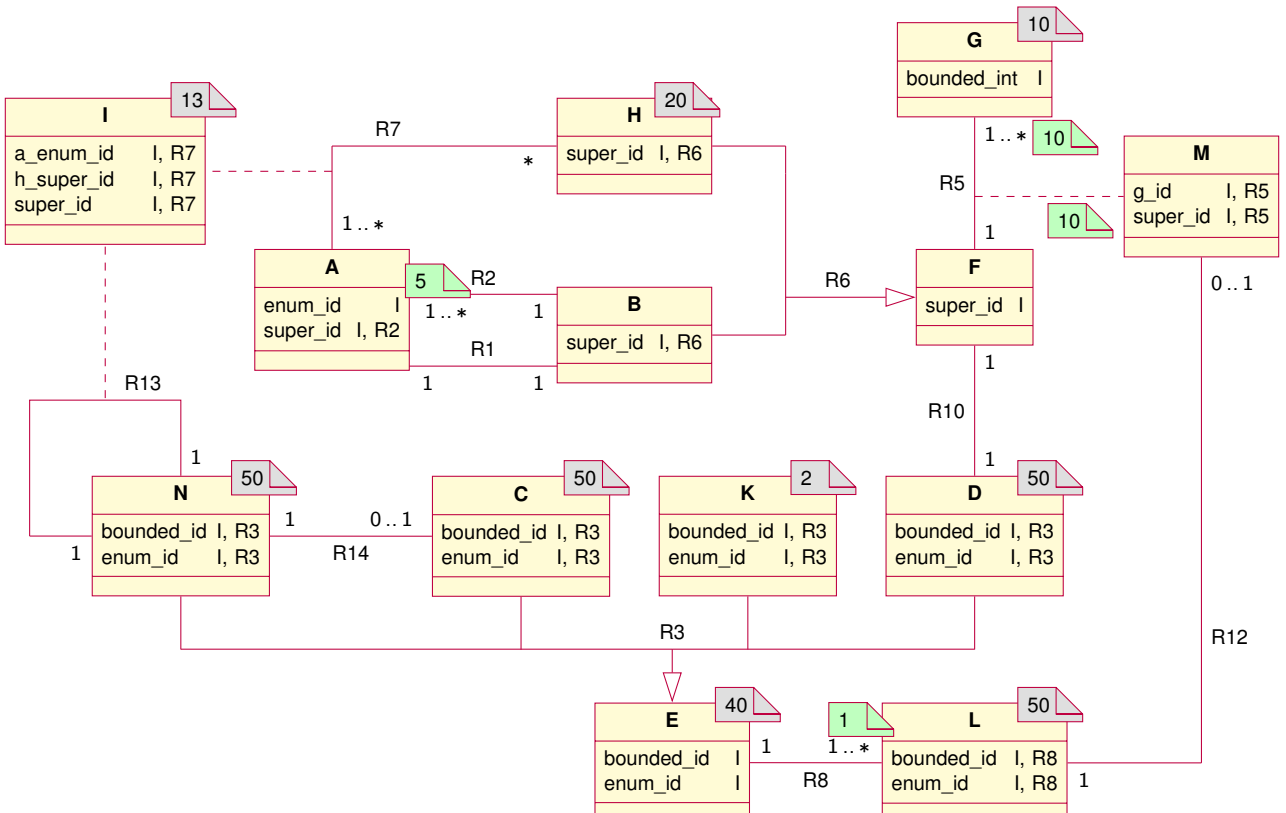
User marking

EXAMPLE



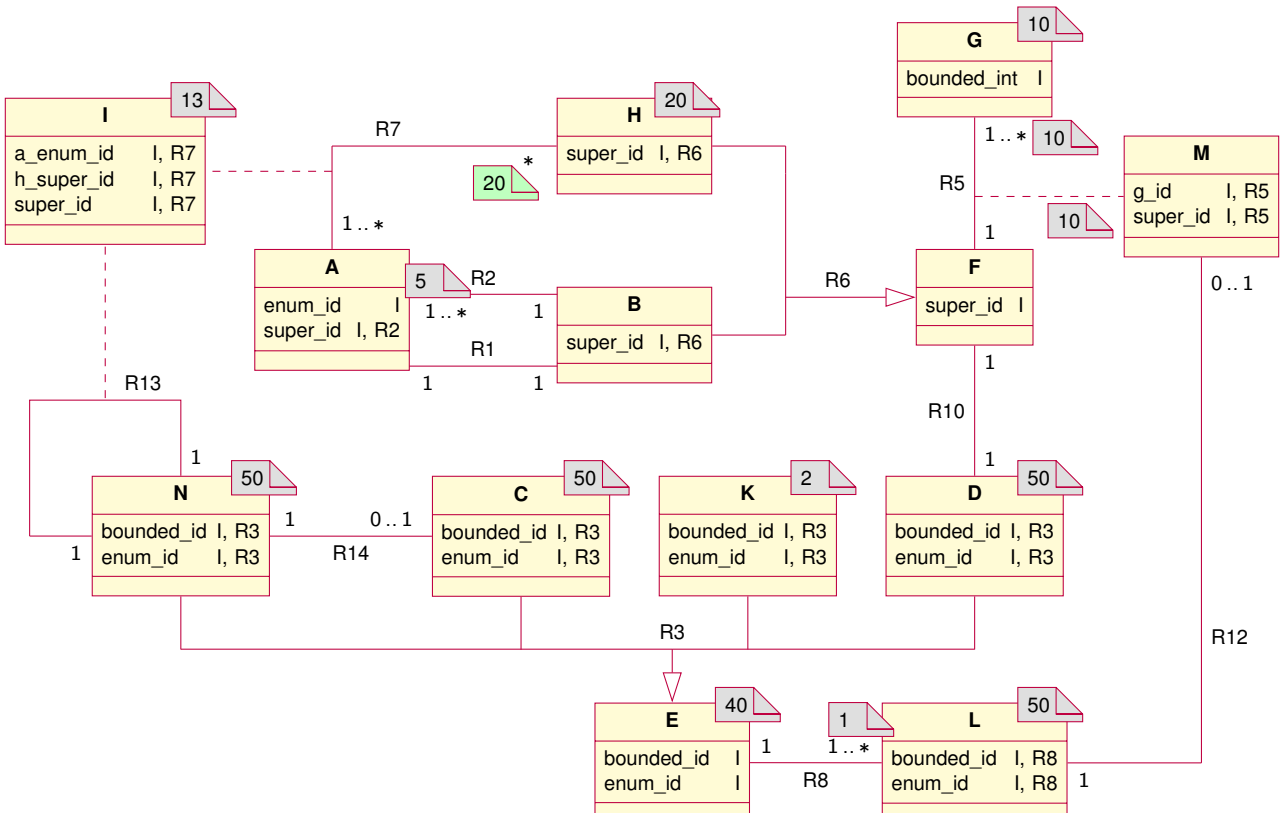
Class bounds from identifiers

EXAMPLE



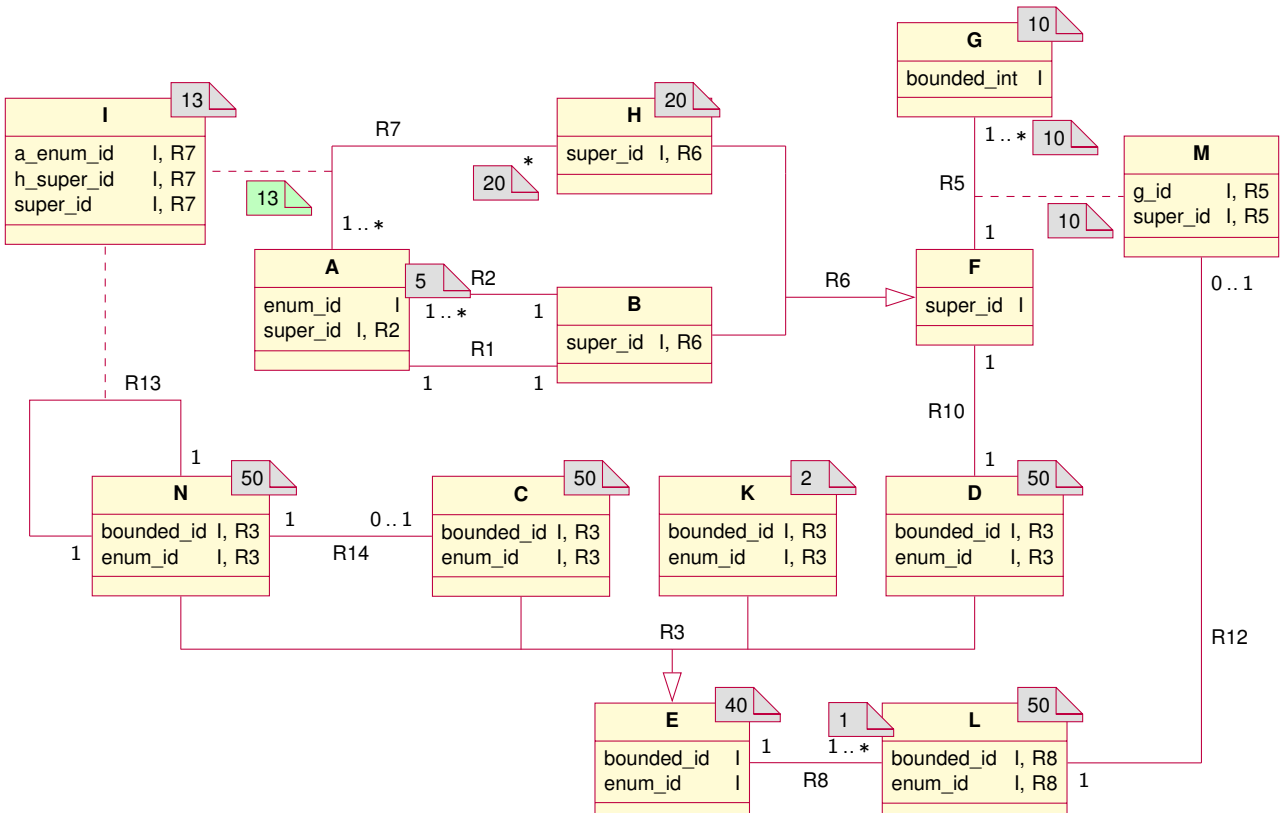
Link bounds from identifiers on target class

EXAMPLE



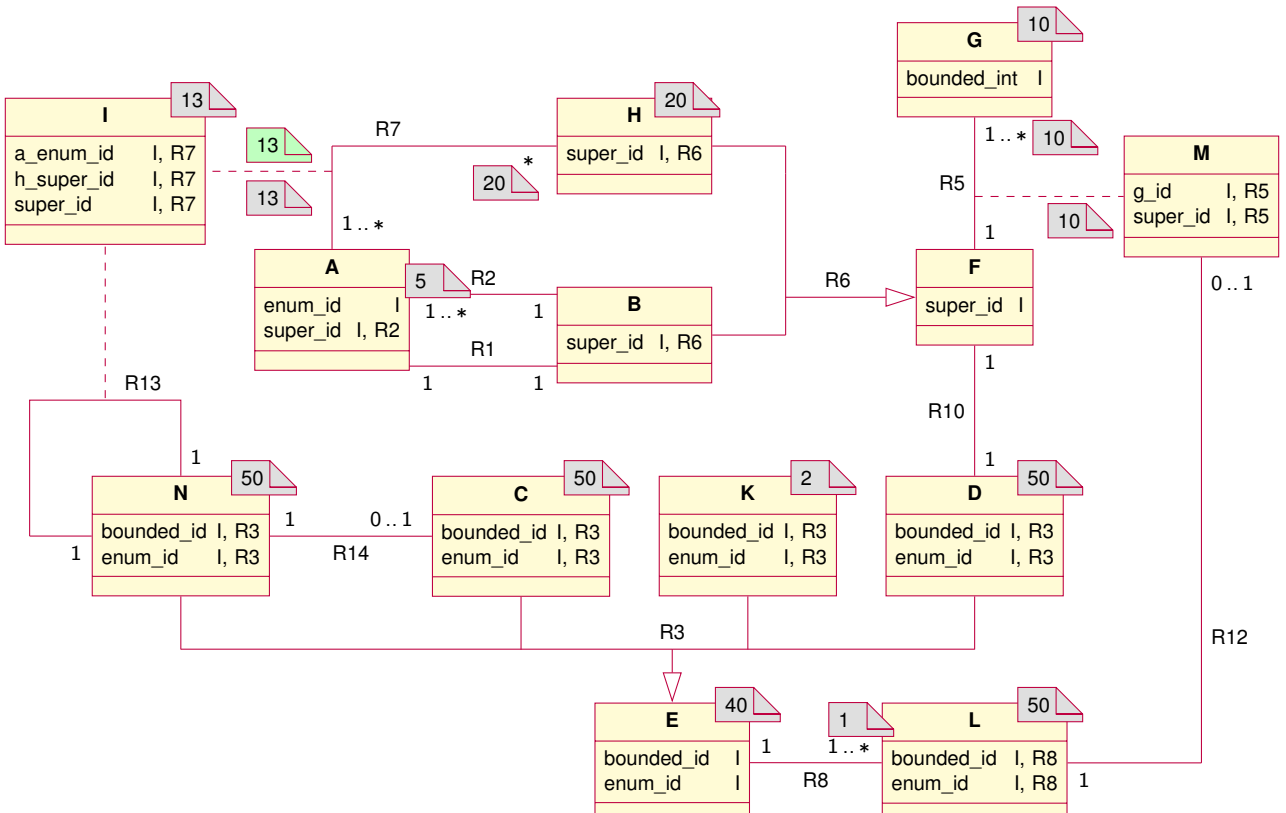
$s_{A,R7,H} \leqslant c_H$

EXAMPLE



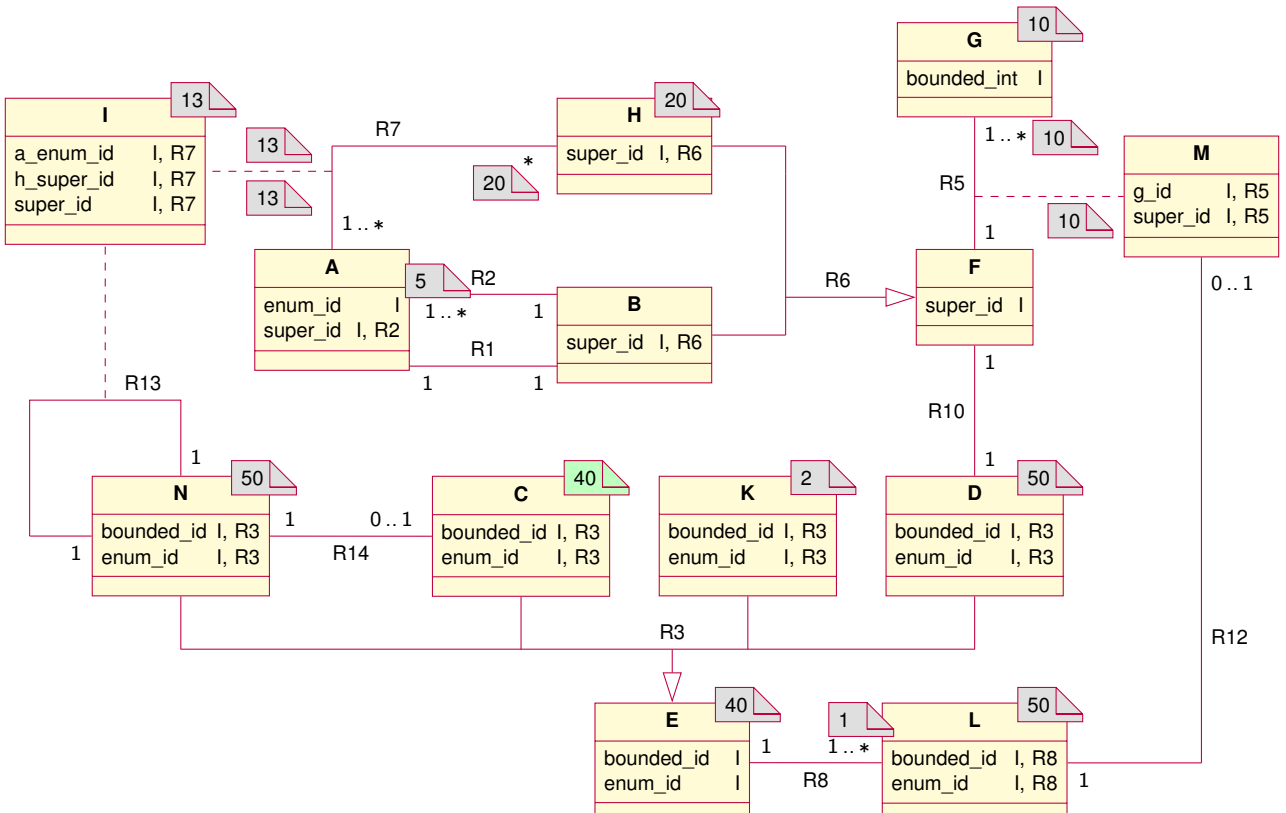
$s_{A,R7,I} \leqslant c_I$

EXAMPLE



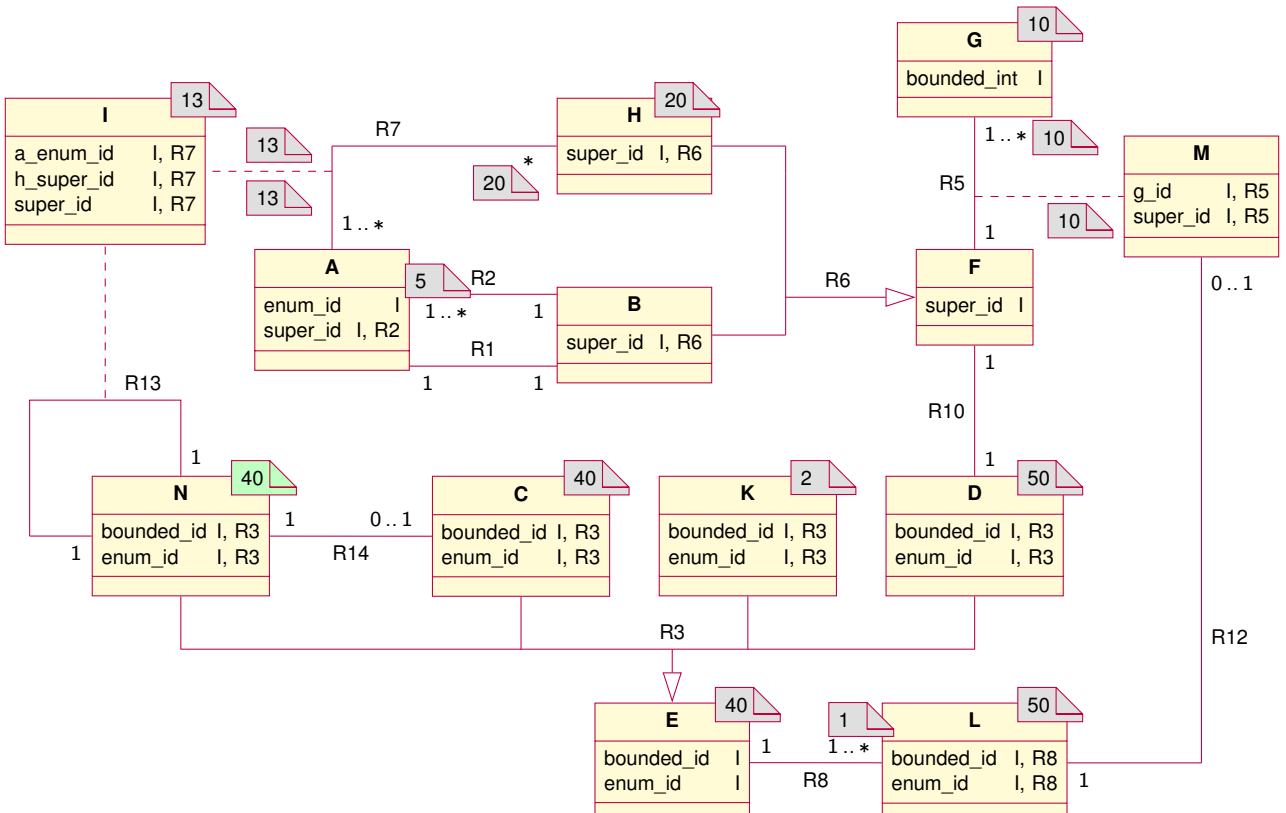
$s_{H,R7,I} \leqslant c_I$

EXAMPLE



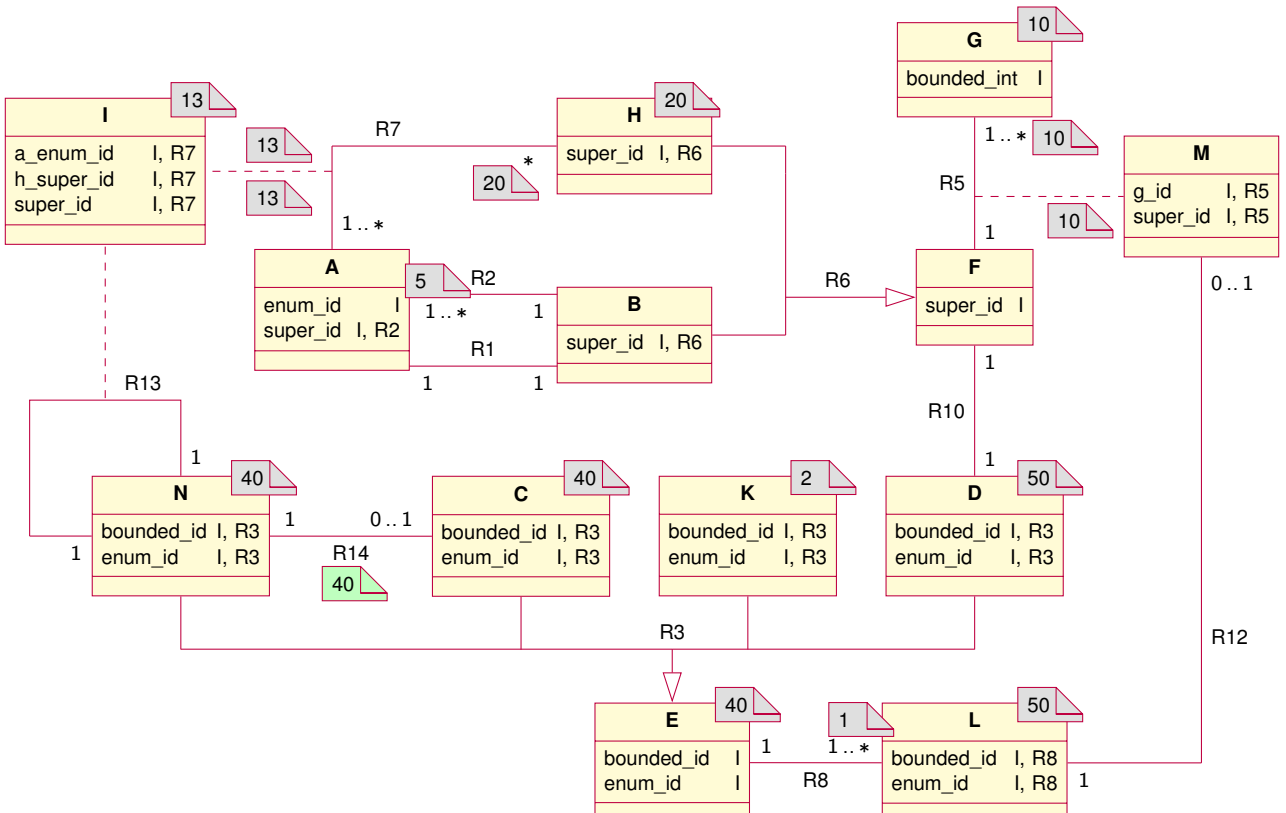
$c_C \leqslant c_E$

EXAMPLE



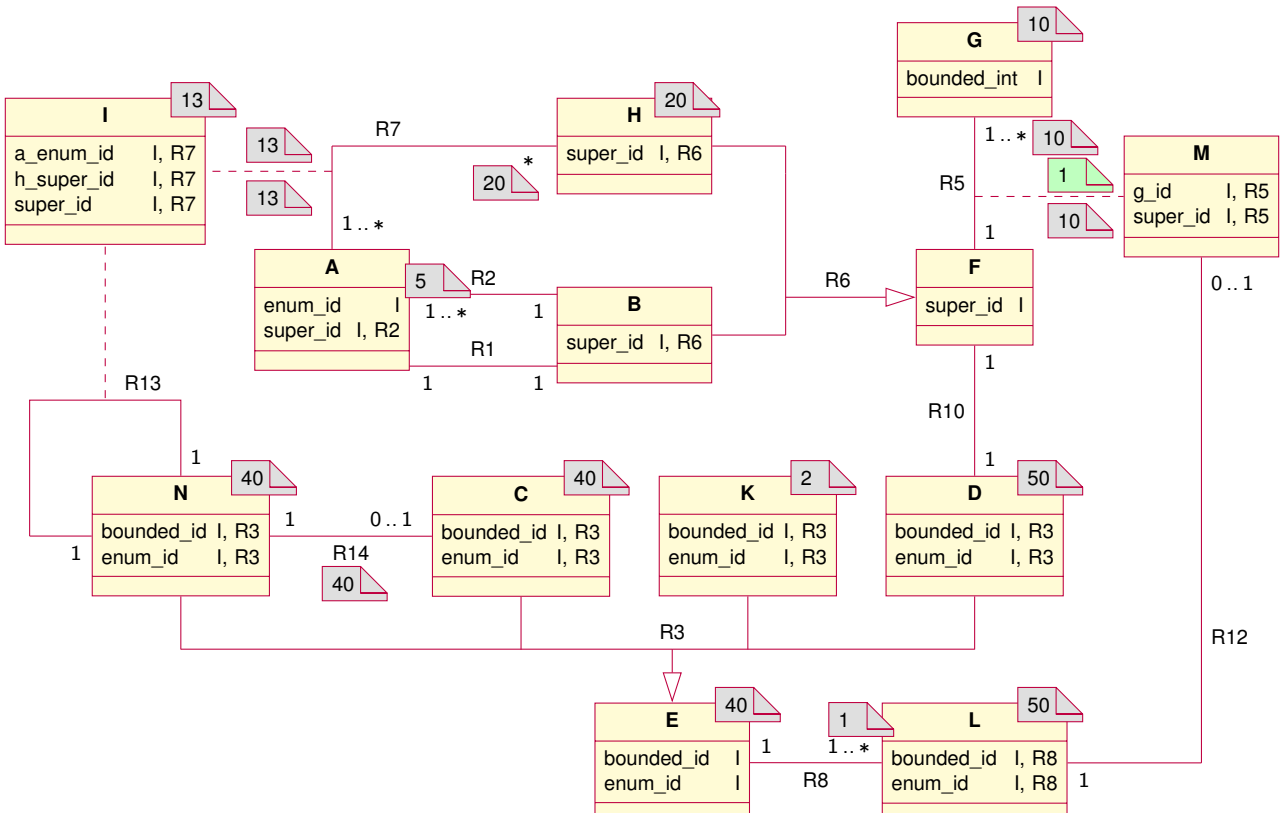
$c_N \leqslant c_E$

EXAMPLE



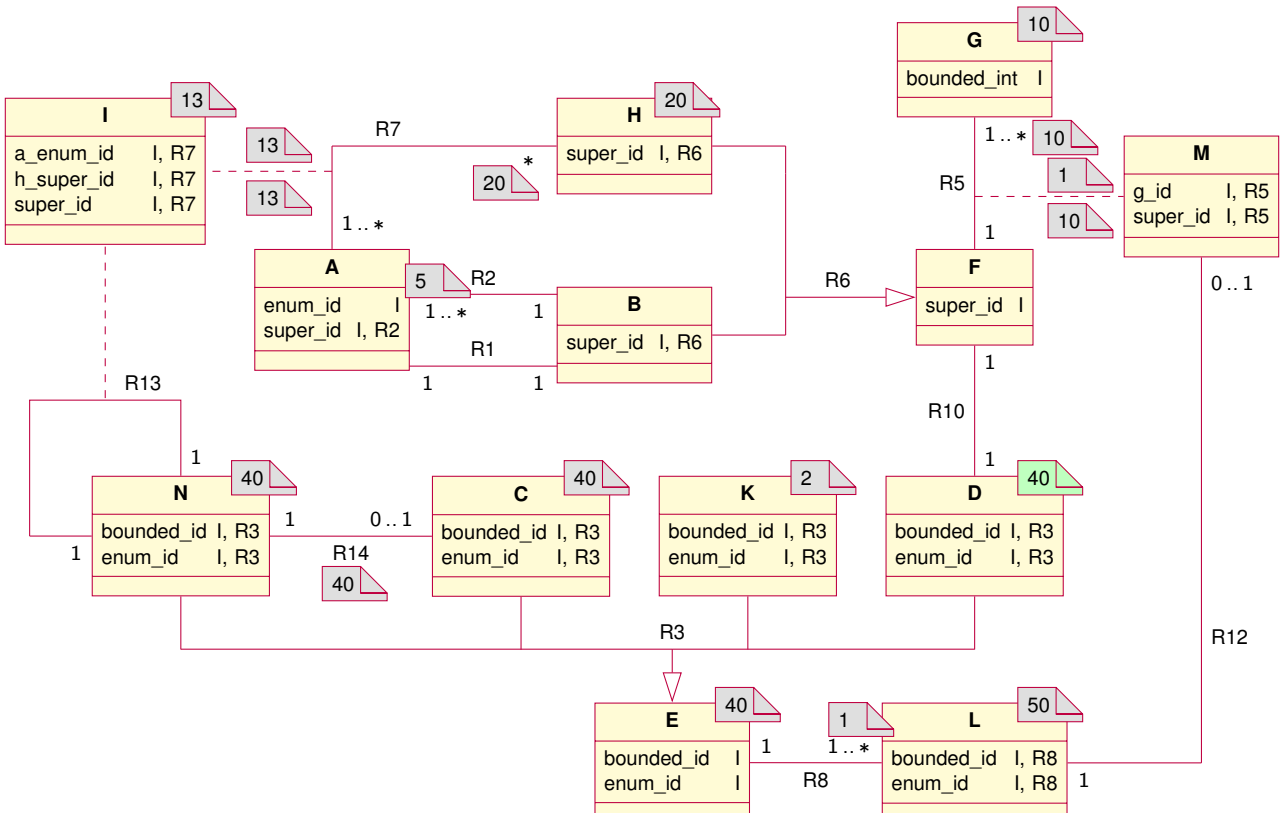
$r_{R14} \leqslant c_C \cdot s_{C,R14,N}$

EXAMPLE



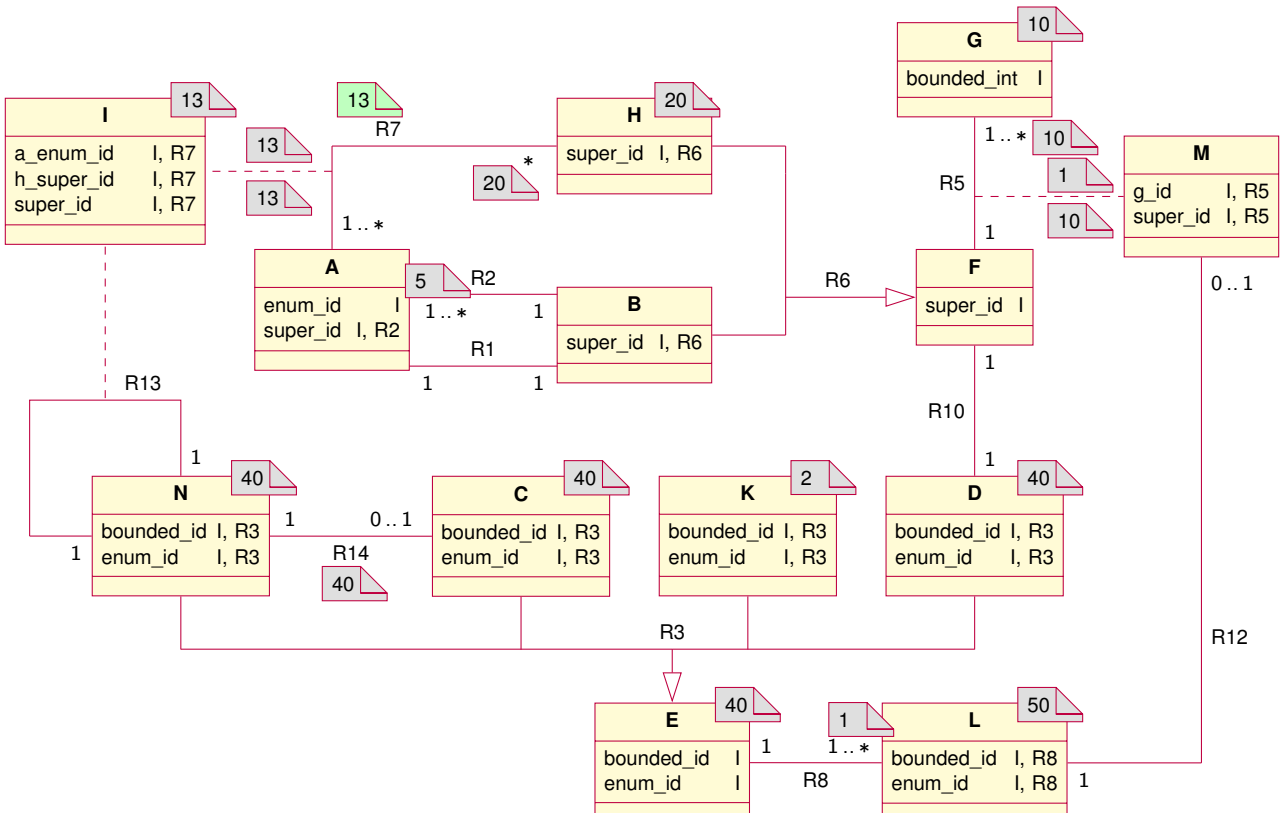
$s_{G,R5,M} \leqslant s_{G,R5,F}$

EXAMPLE



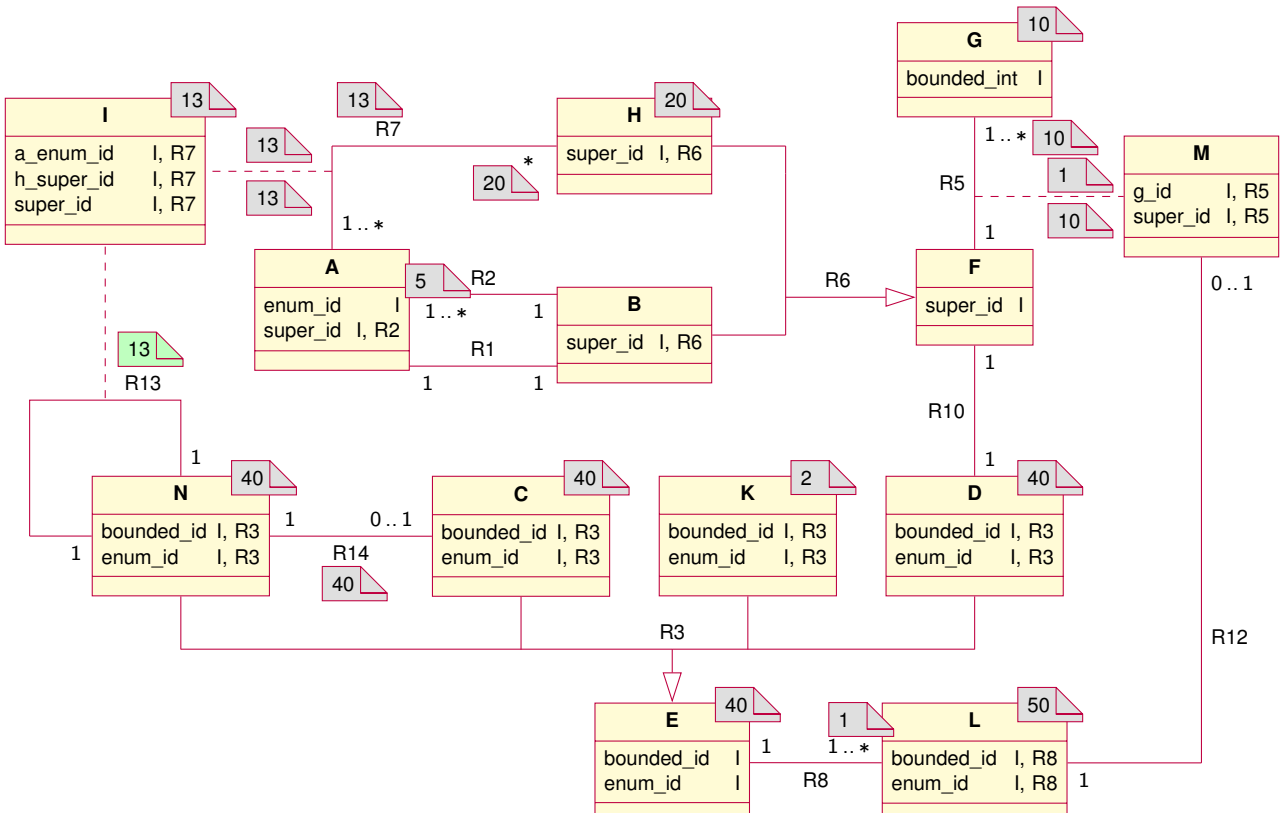
$c_D \leqslant c_E$

EXAMPLE



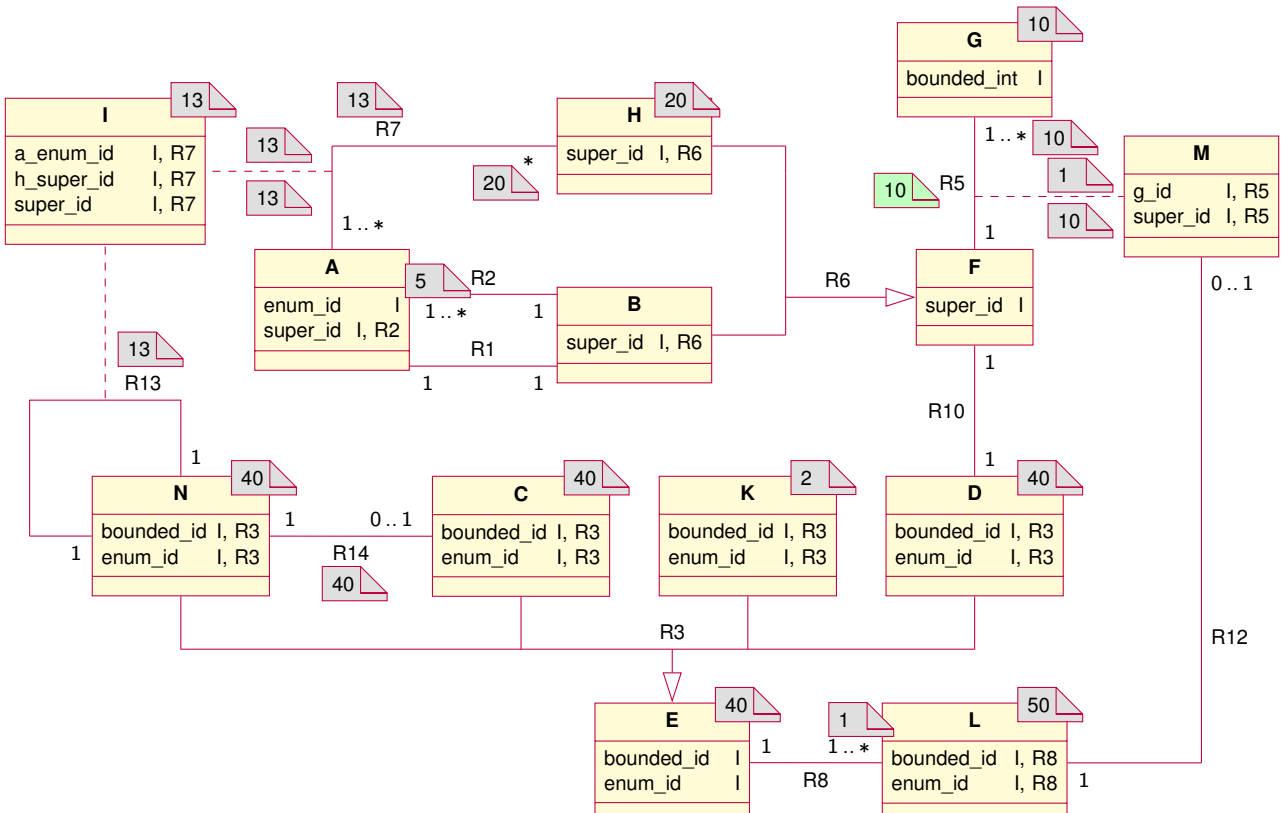
$r_{R7} \leqslant c_I$

EXAMPLE



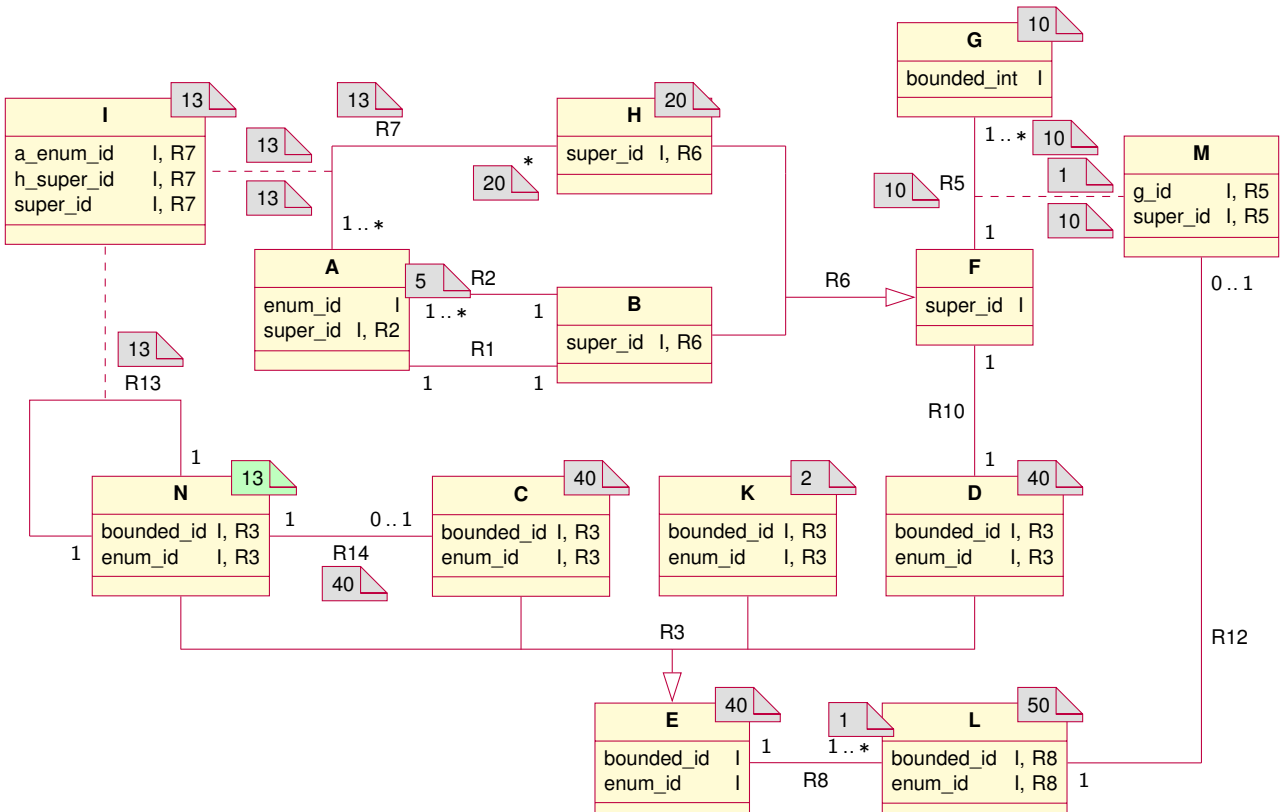
$r_{R13} \leqslant c_I$

EXAMPLE



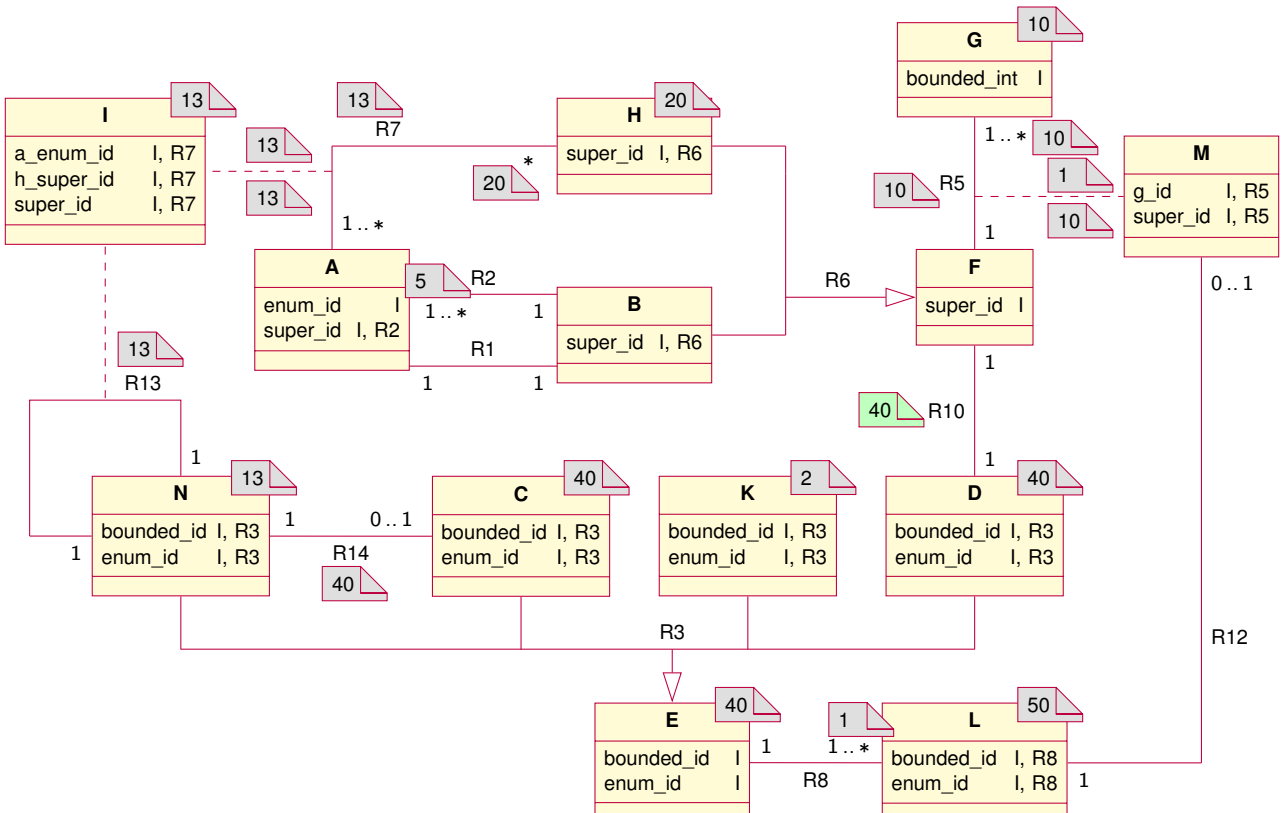
$r_{R5} \leqslant c_G \cdot s_{G,R5,M}$

EXAMPLE



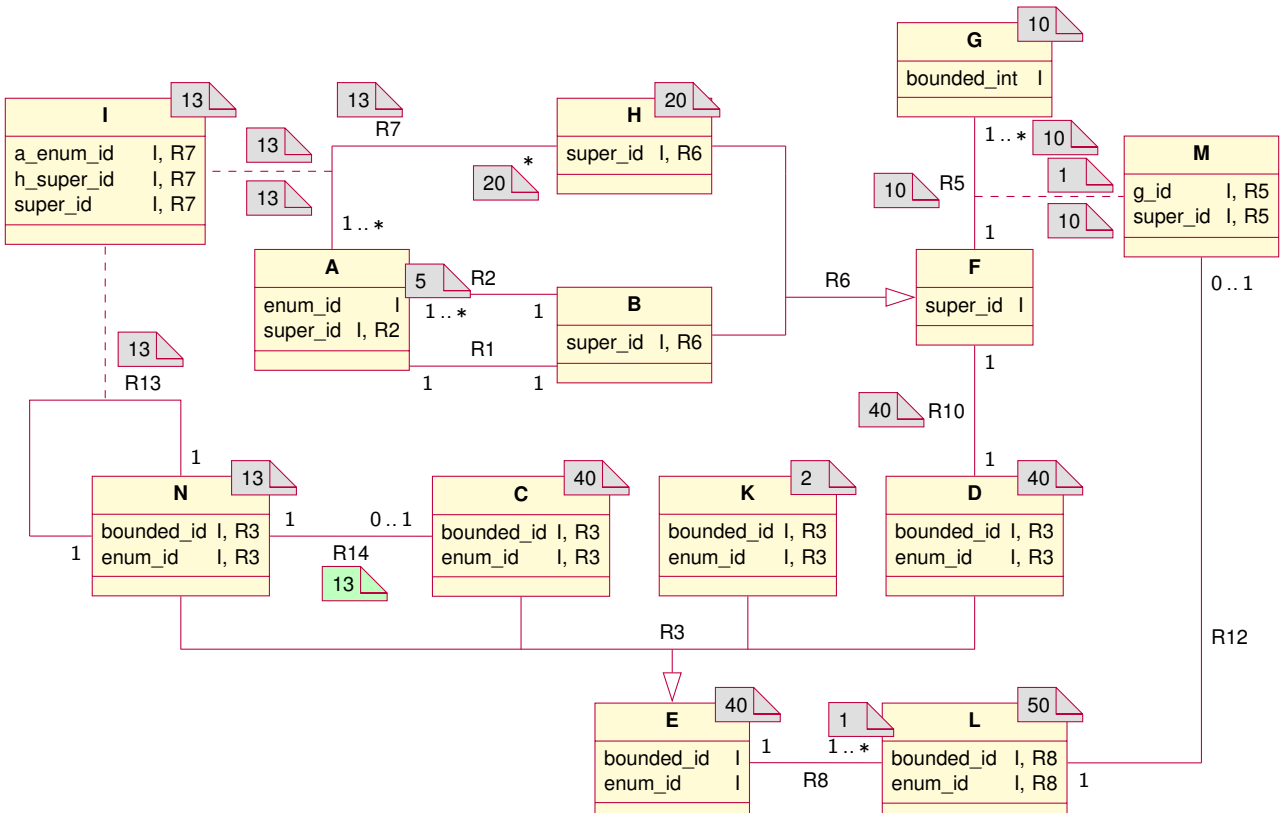
$c_N \leqslant r_{R13}$

EXAMPLE



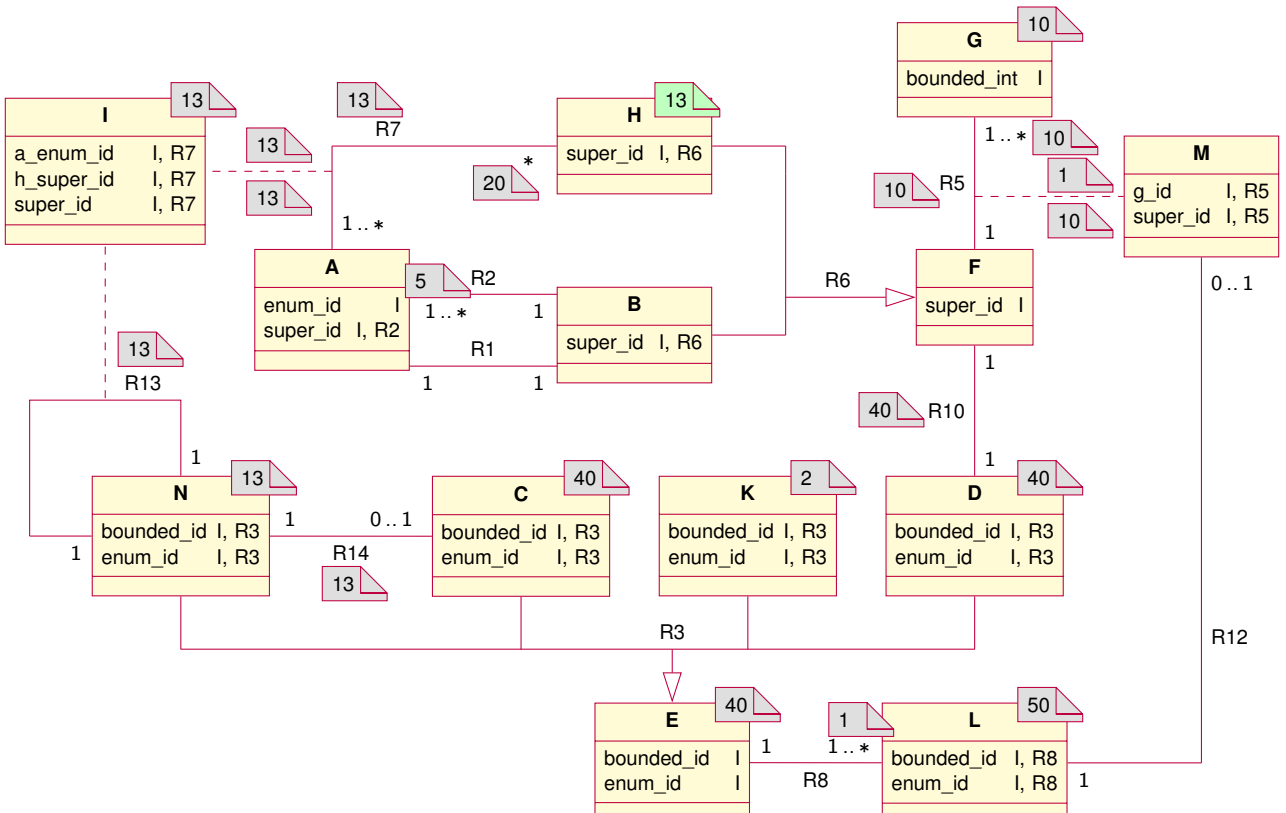
$r_{R10} \leqslant c_D$

EXAMPLE



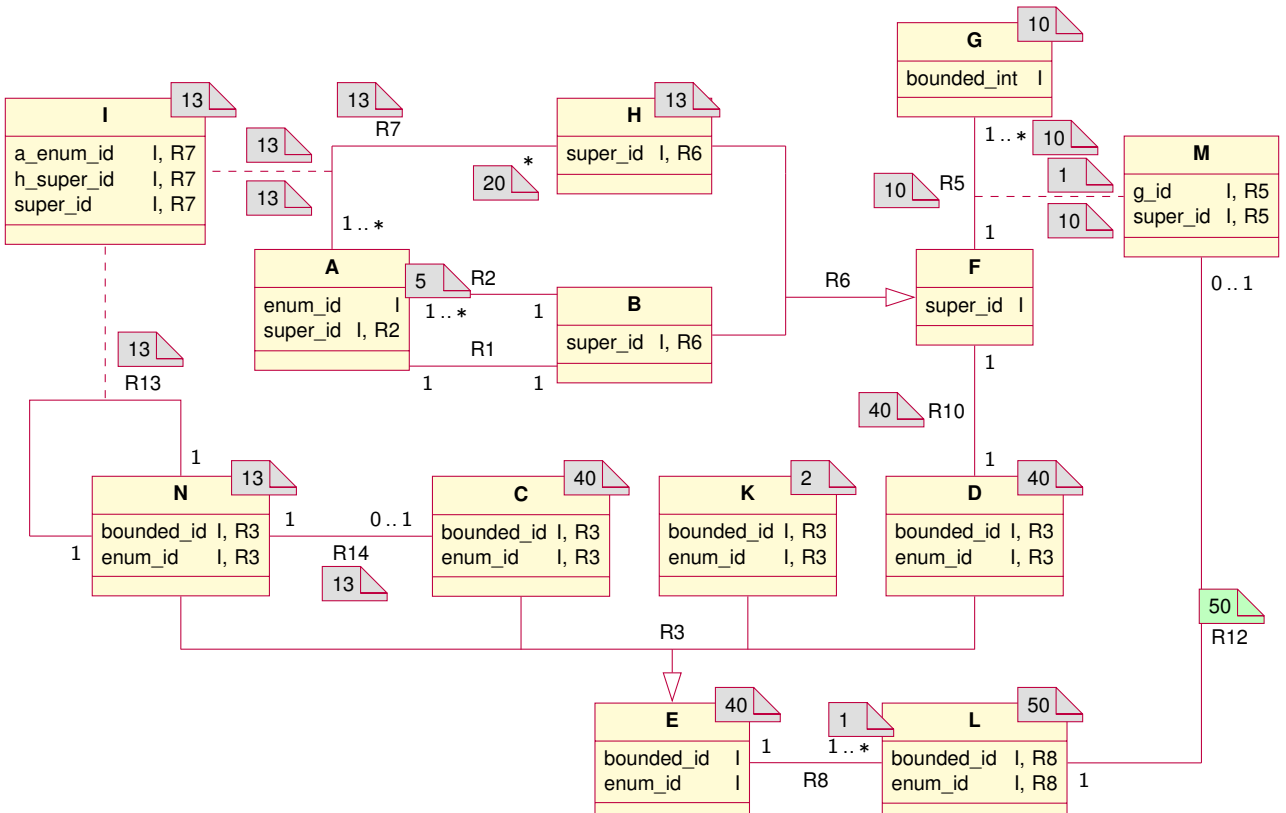
$r_{R14} \leqslant c_N \cdot s_{N,R14,C}$

EXAMPLE



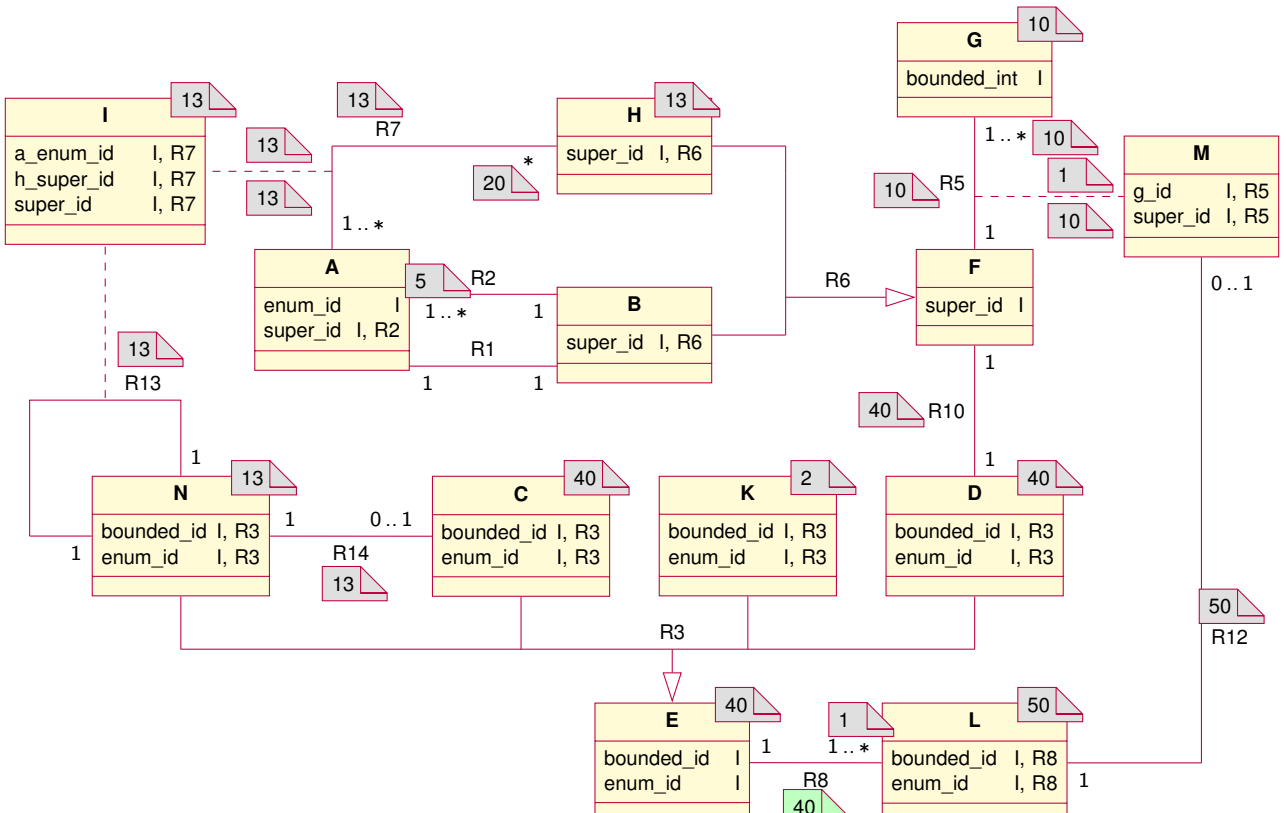
$c_H \leqslant r_{R7}$

EXAMPLE



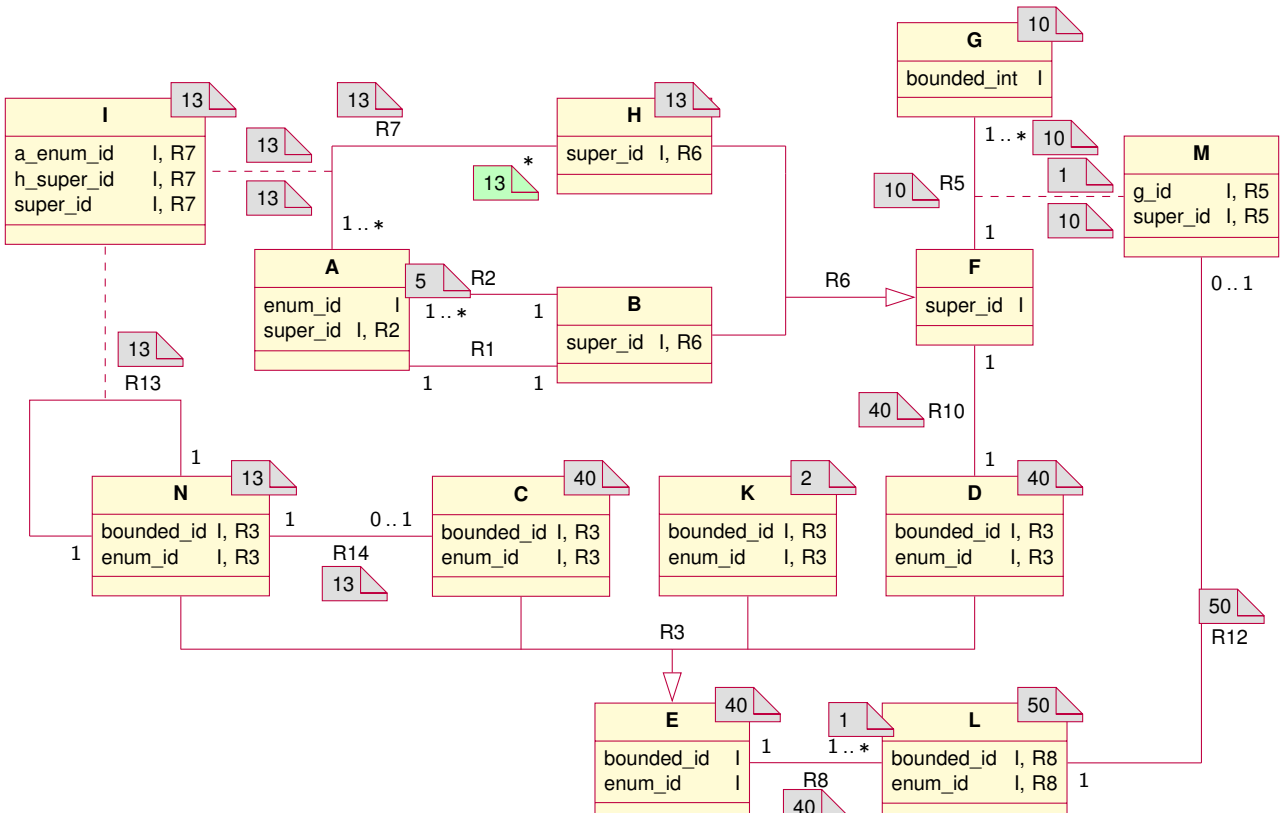
$r_{R12} \leqslant c_L$

EXAMPLE



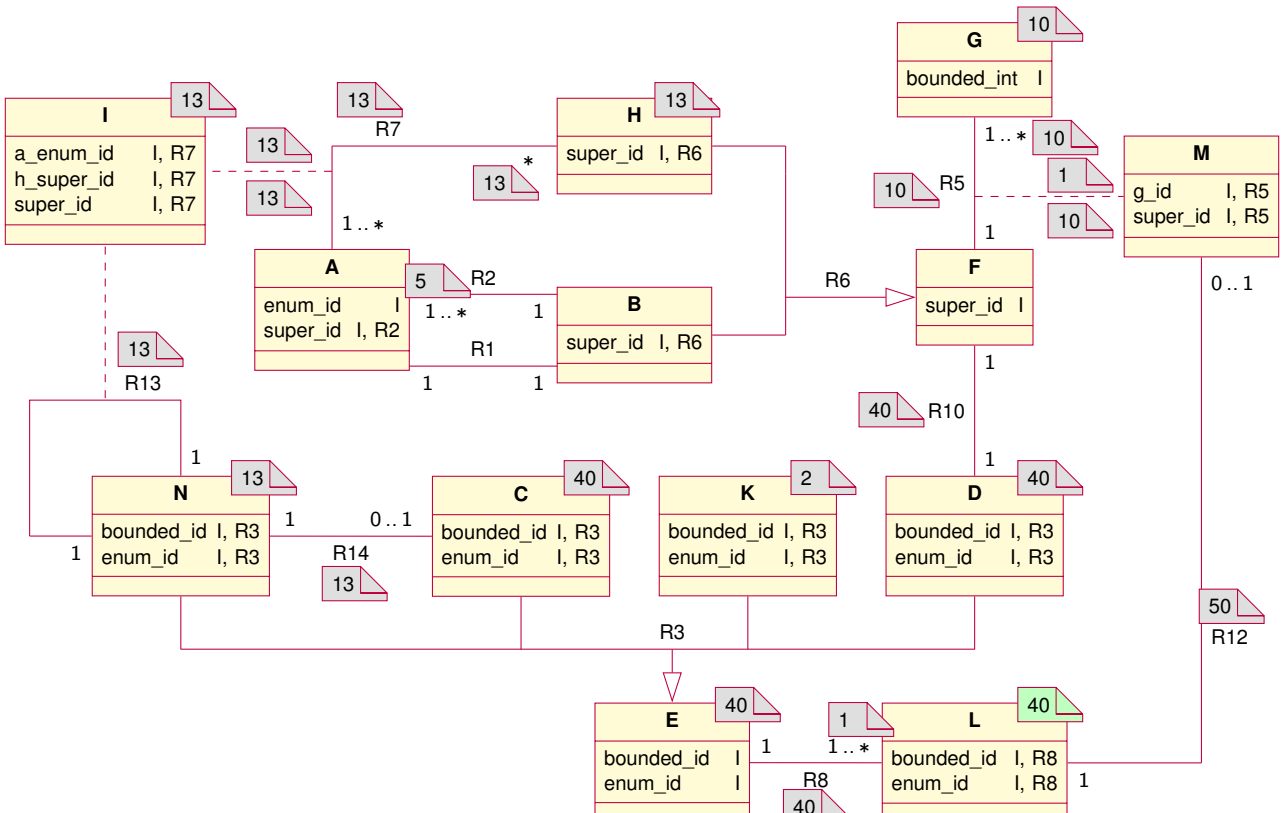
$r_{R8} \leqslant c_E \cdot s_{E,R8,L}$

EXAMPLE



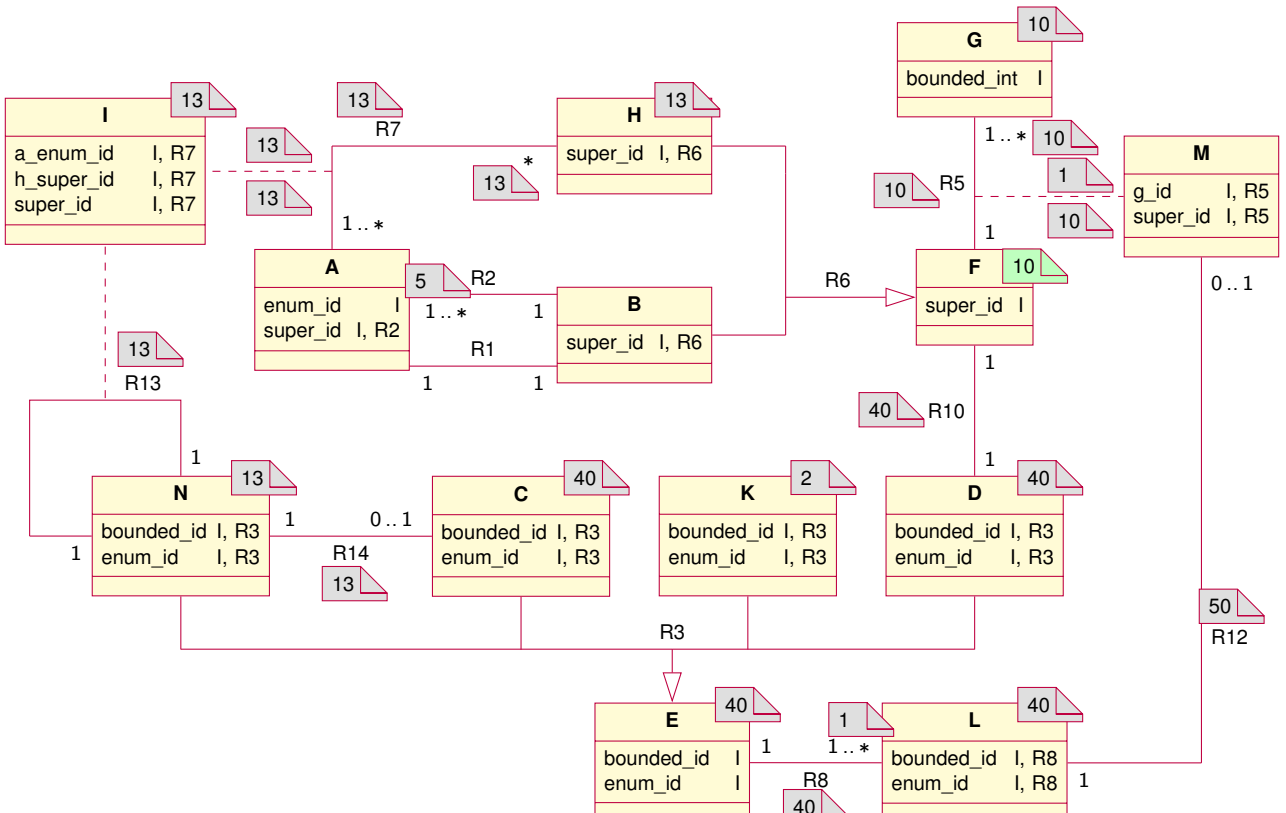
$s_{A,R7,H} \leqslant c_H$

EXAMPLE



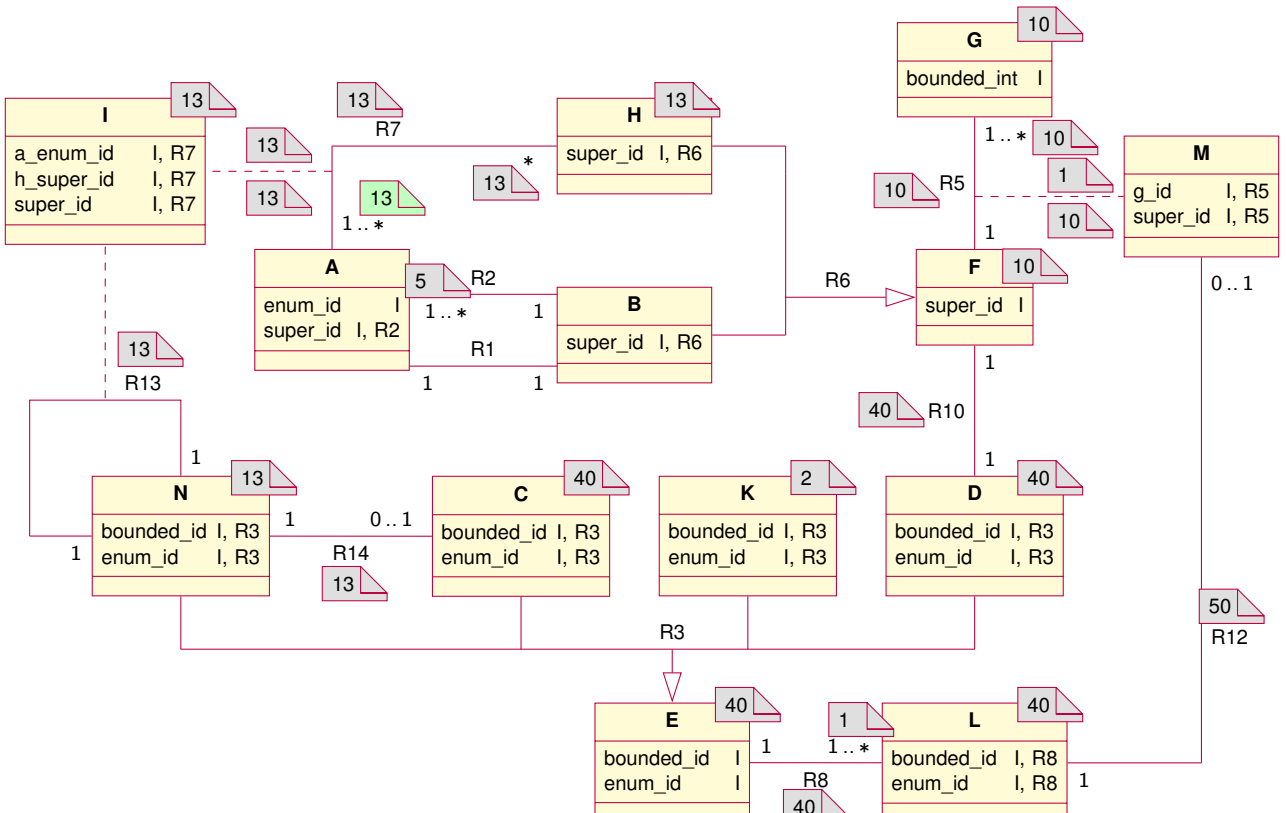
$c_L \leqslant r_{R8}$

EXAMPLE



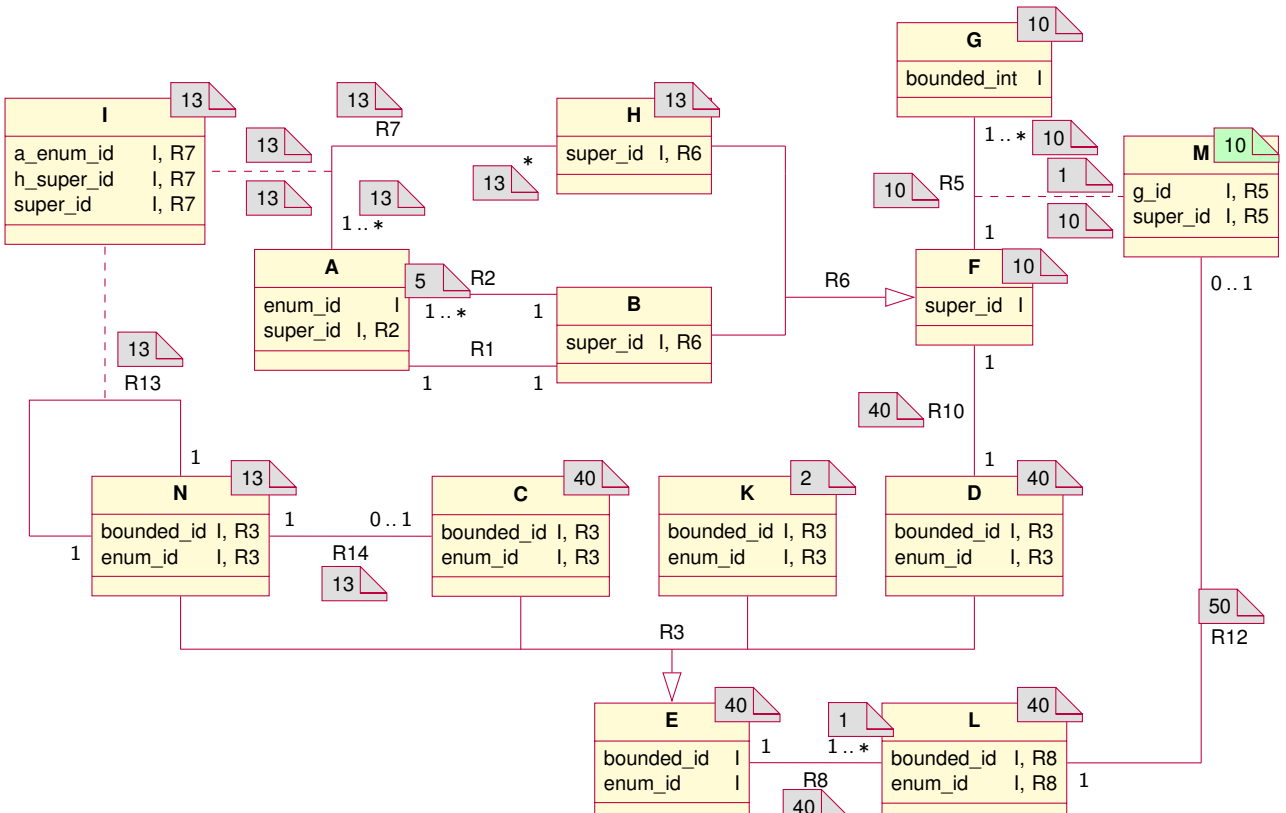
$c_F \leqslant r_{R5}$

EXAMPLE



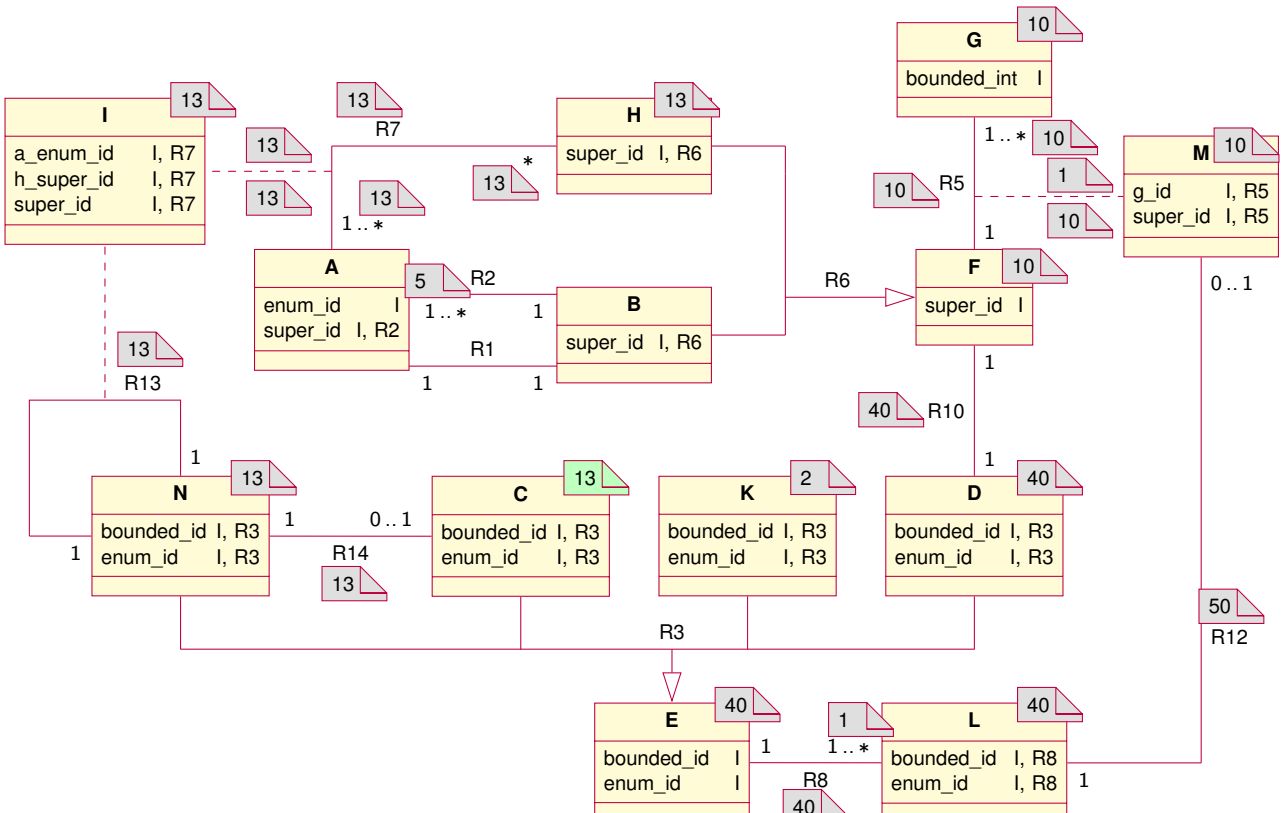
$s_{H,R7,A} \leqslant s_{H,R7,I}$

EXAMPLE



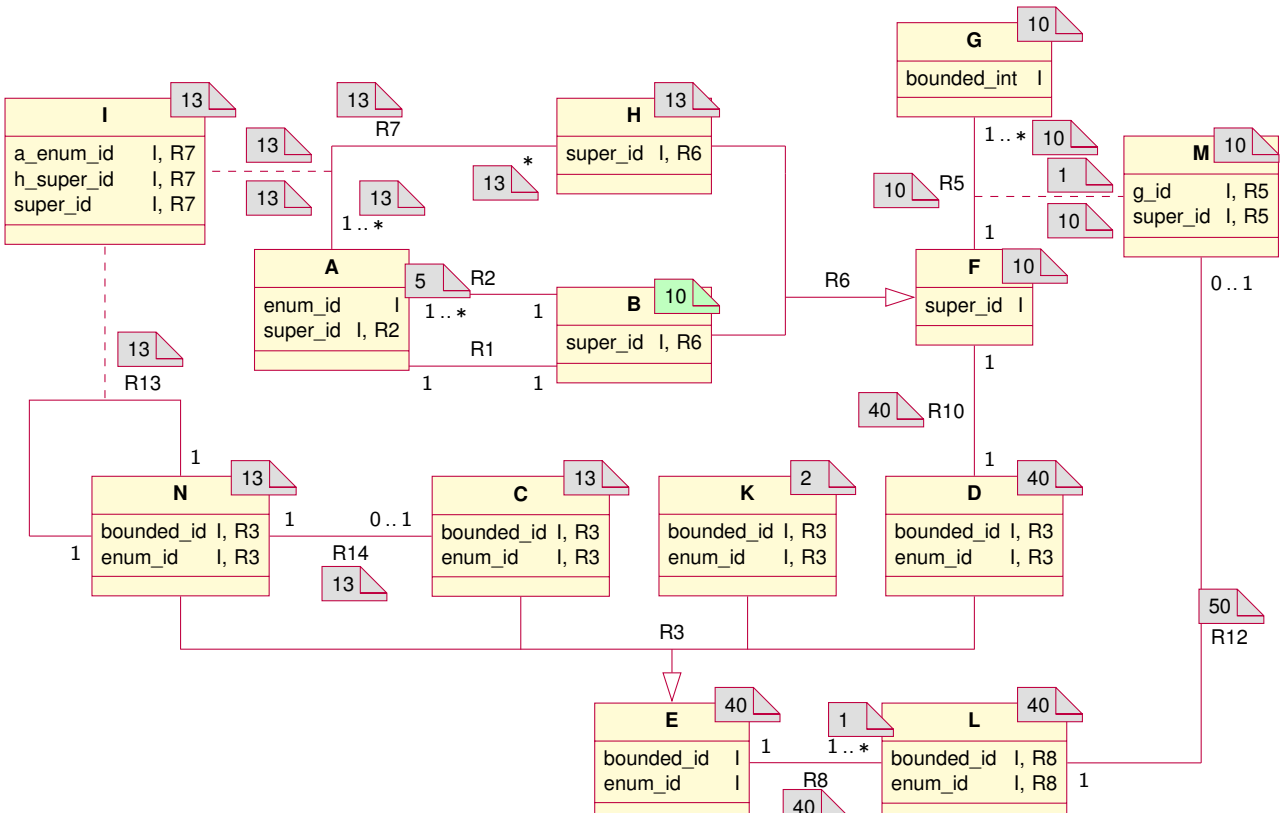
$c_M \leqslant r_{R5}$

EXAMPLE



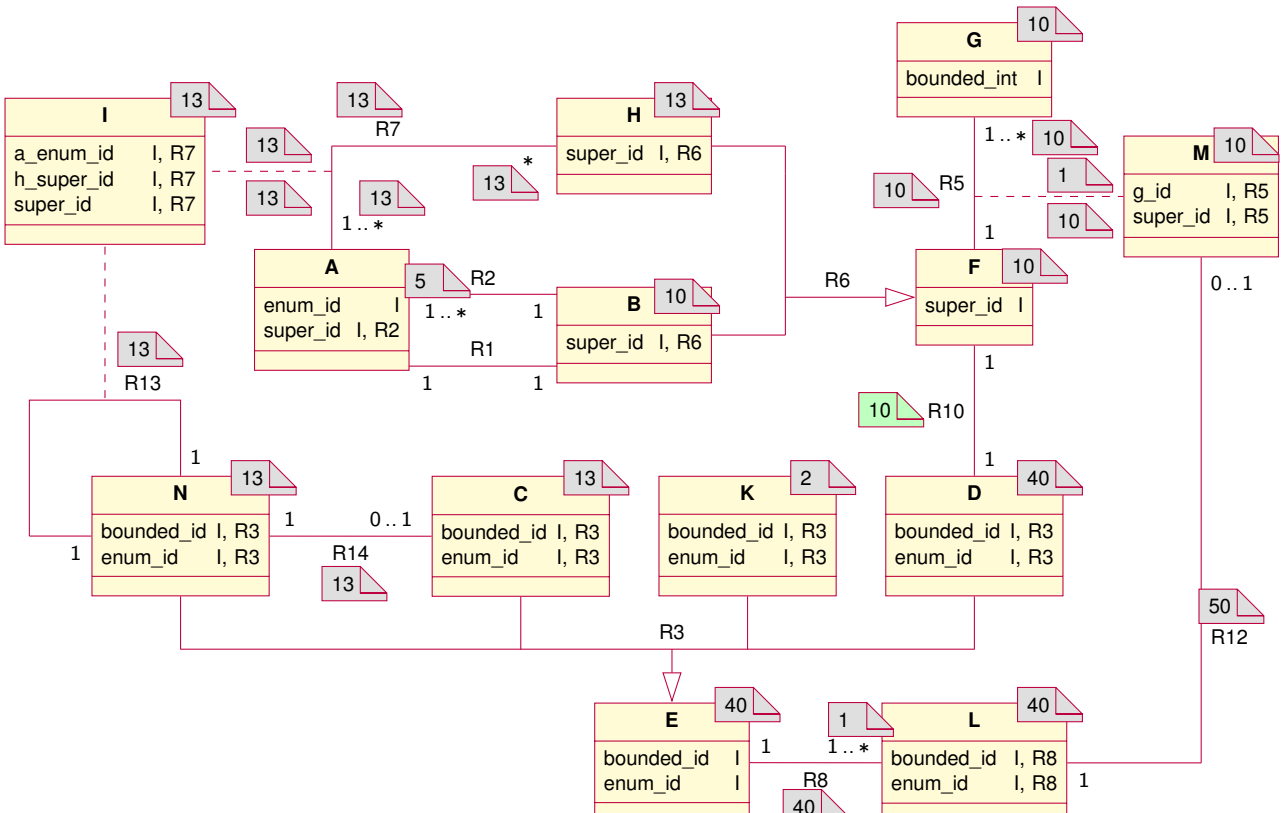
$c_C \leqslant r_{R14}$

EXAMPLE



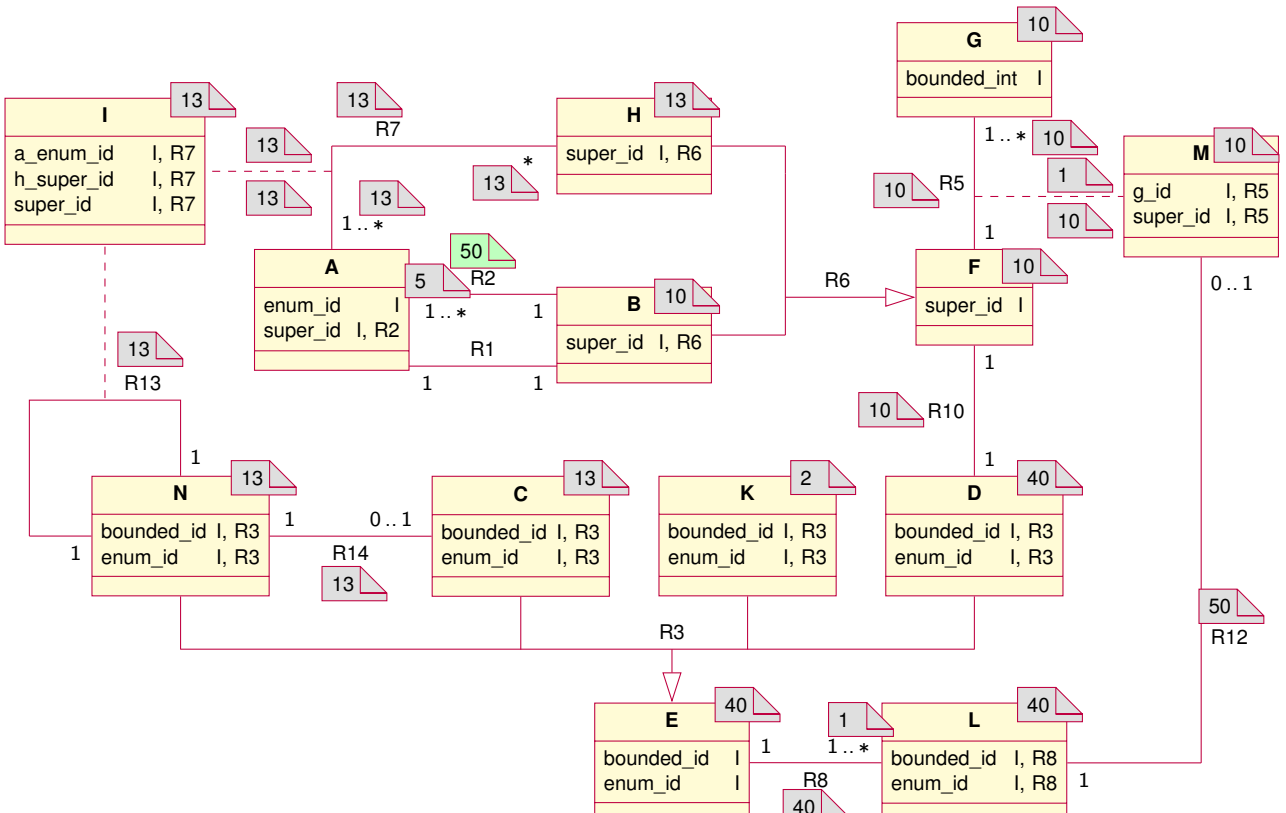
$c_B \leqslant c_F$

EXAMPLE



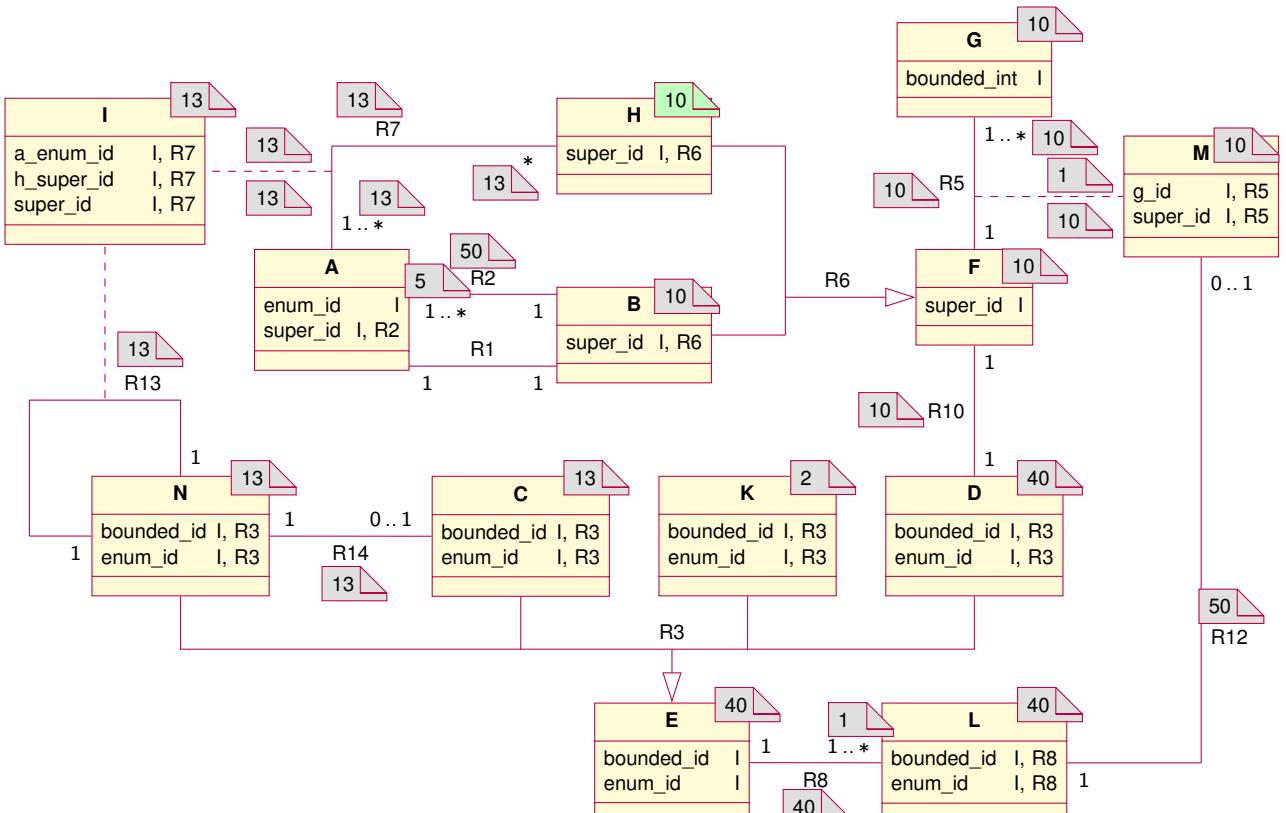
$r_{R10} \leqslant c_F \cdot s_{F,R10,D}$

EXAMPLE



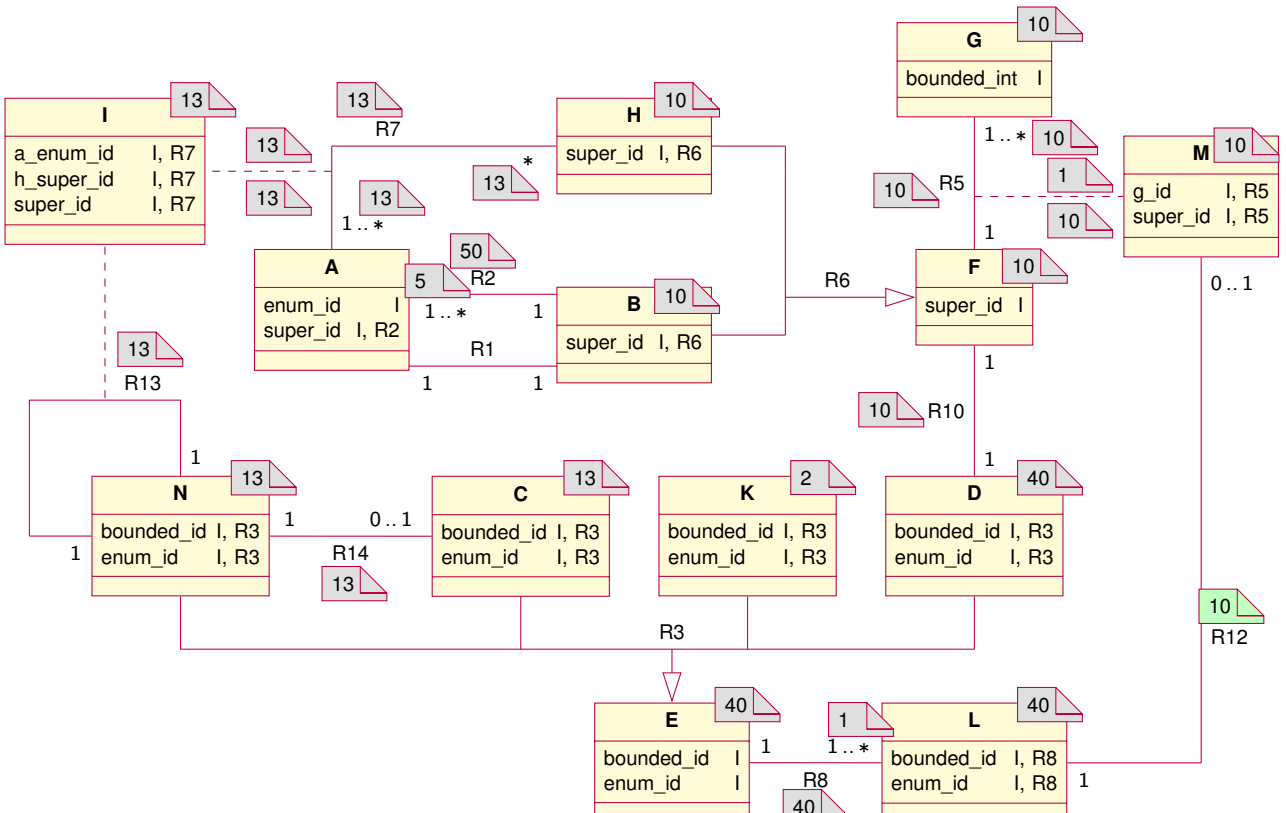
$r_{R2} \leqslant c_B \cdot s_{B,R2,A}$

EXAMPLE



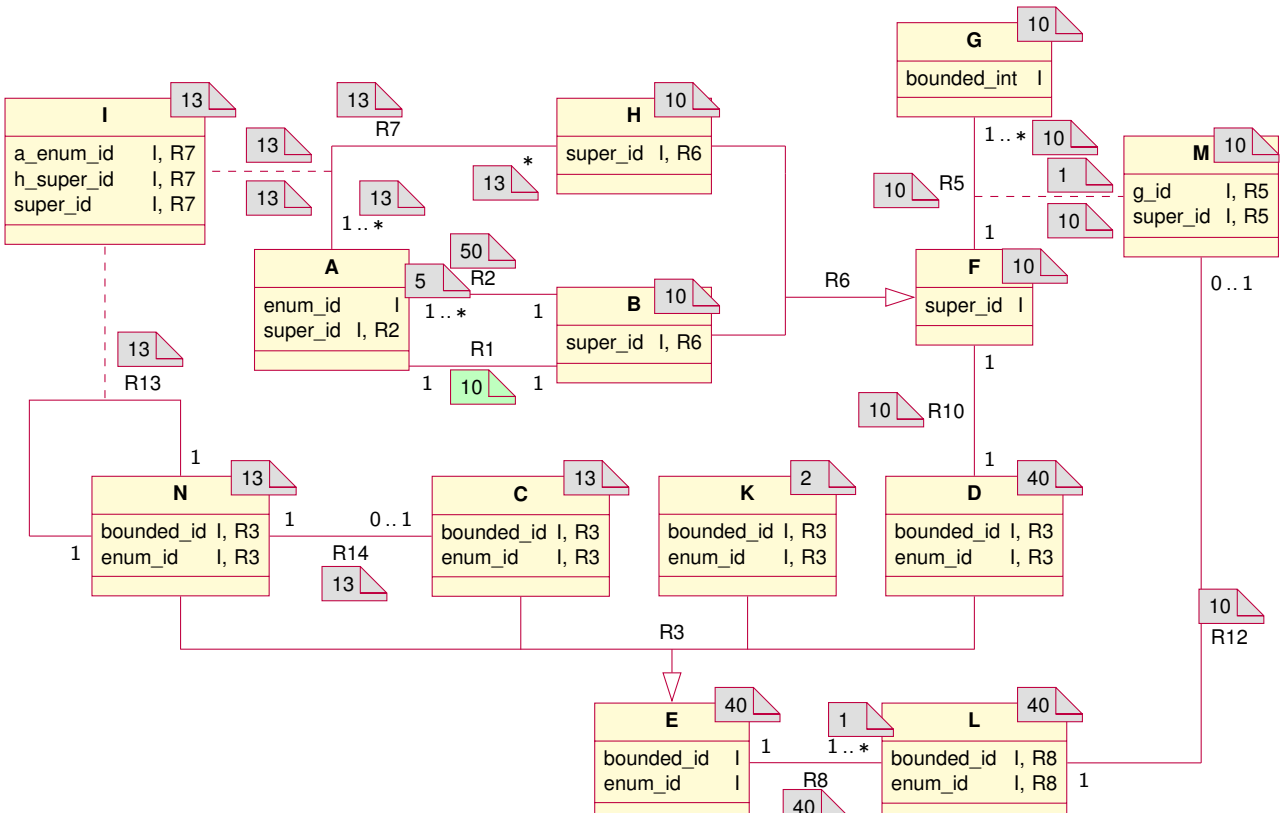
$c_H \leqslant c_F$

EXAMPLE



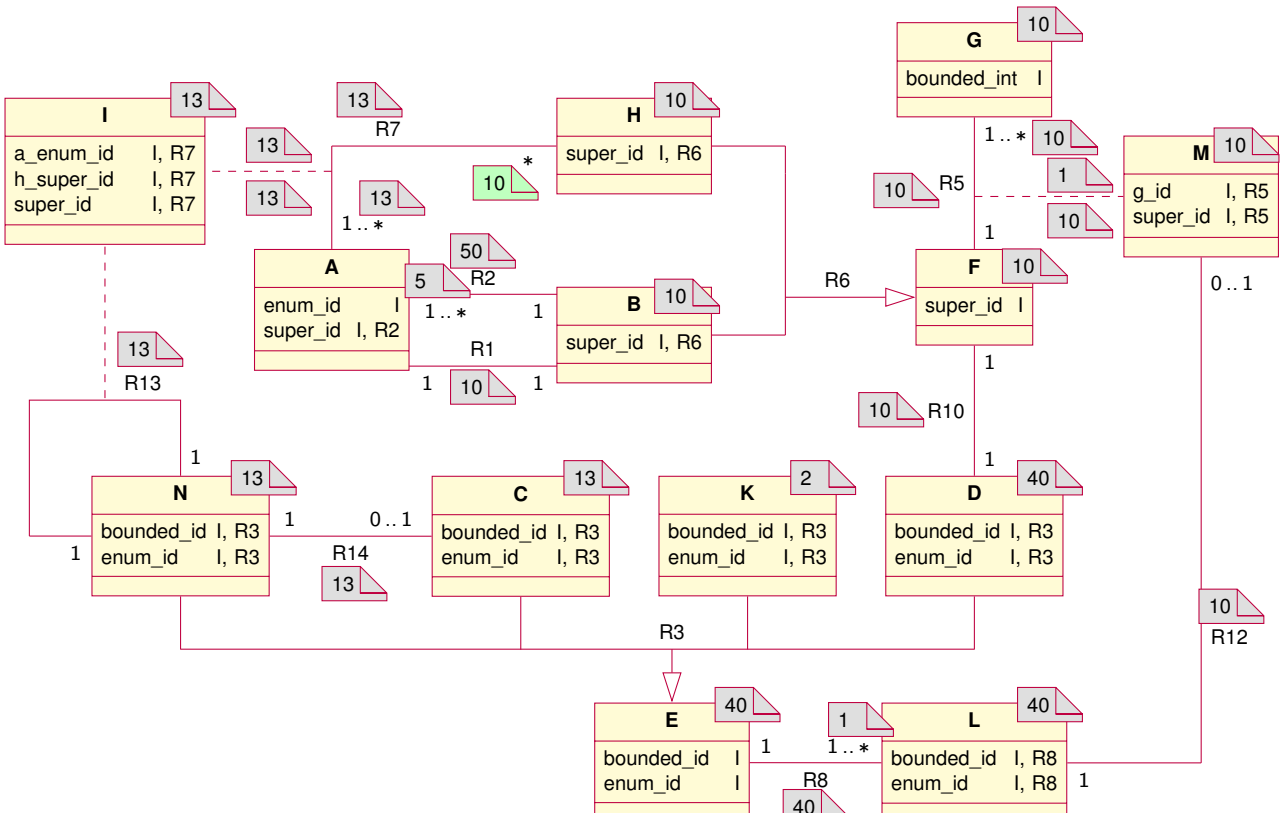
$r_{R12} \leqslant c_M$

EXAMPLE



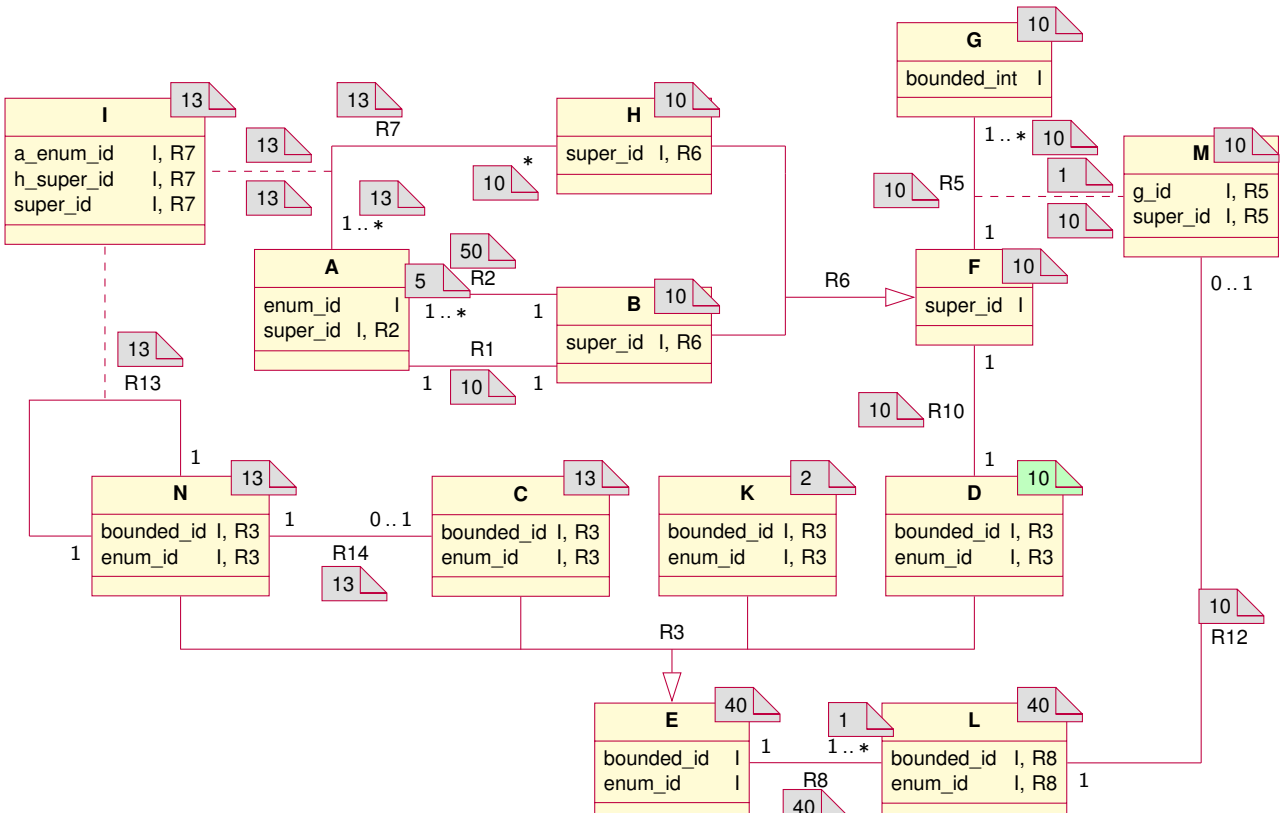
$r_{R1} \leqslant c_B$

EXAMPLE



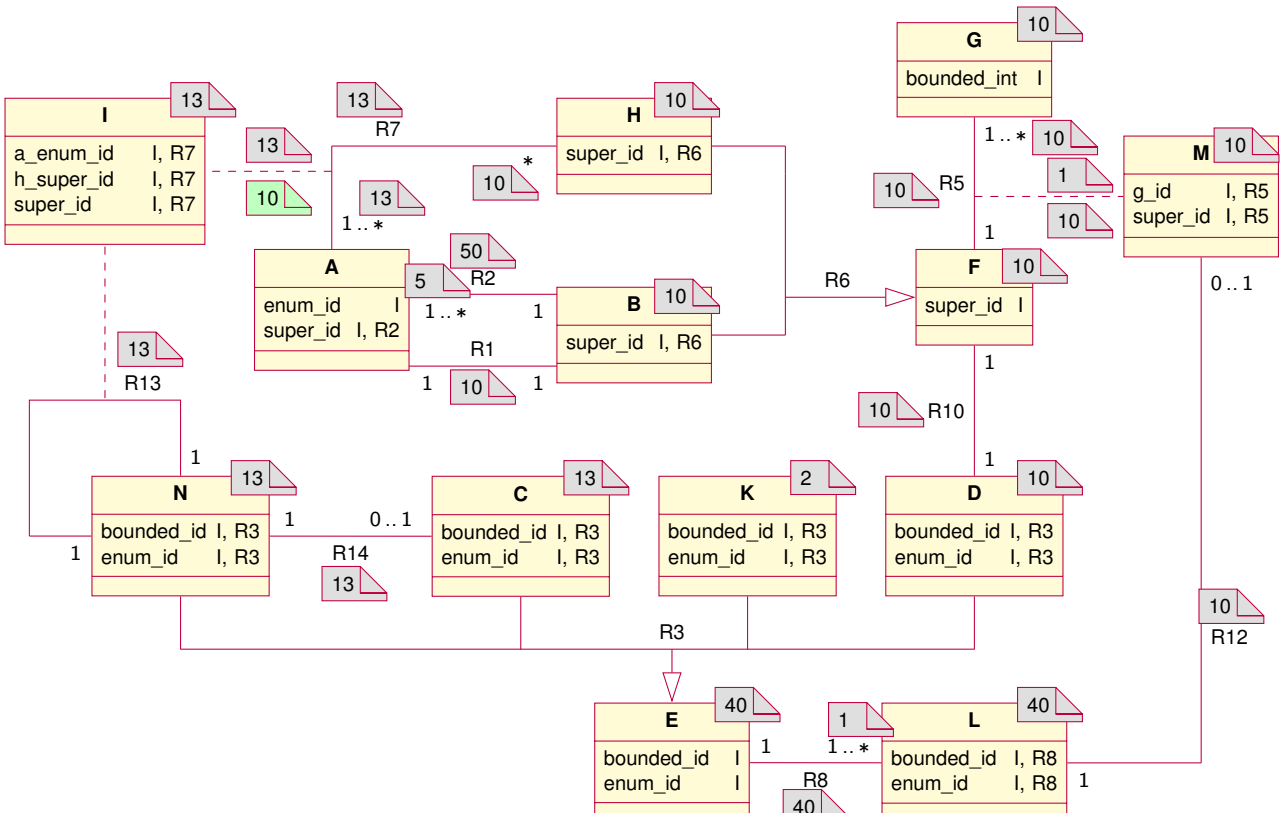
$s_{A,R7,H} \leqslant c_H$

EXAMPLE



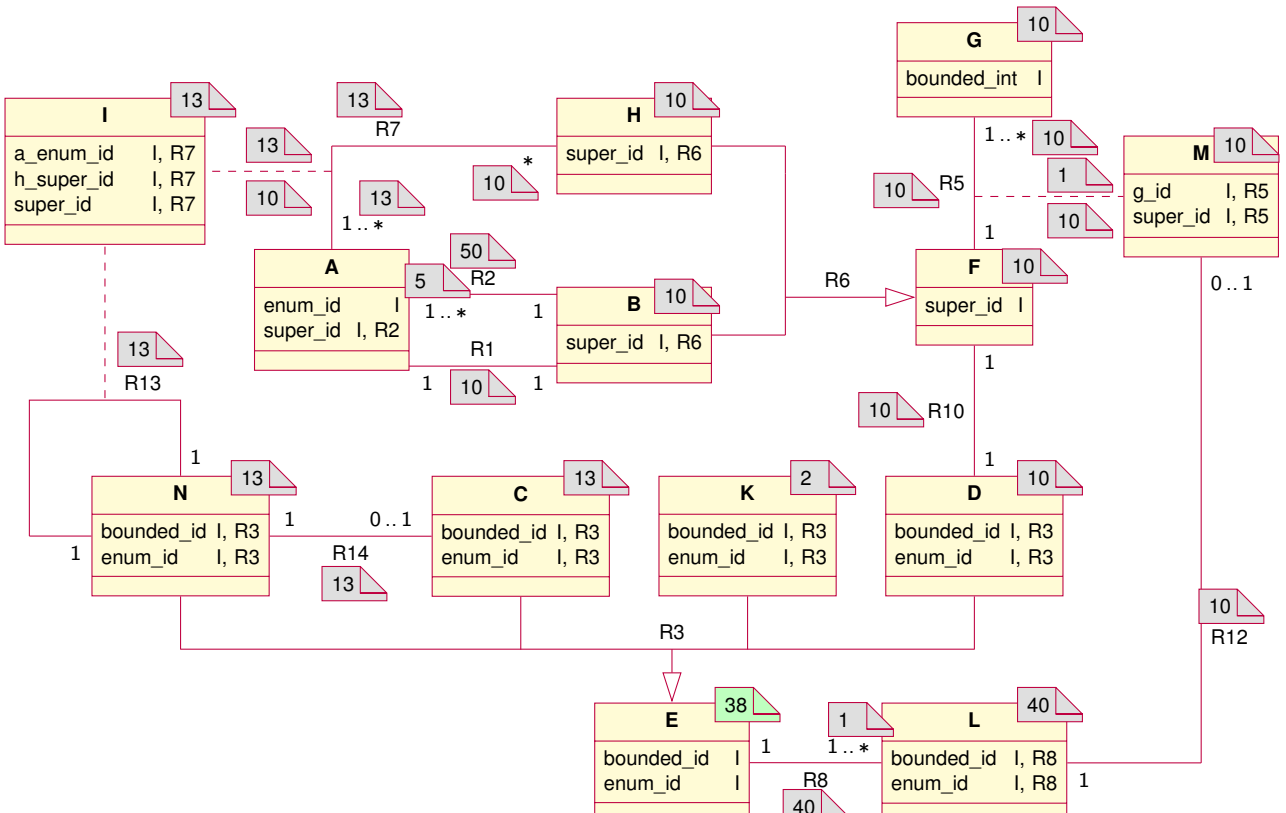
$c_D \leqslant r_{R10}$

EXAMPLE



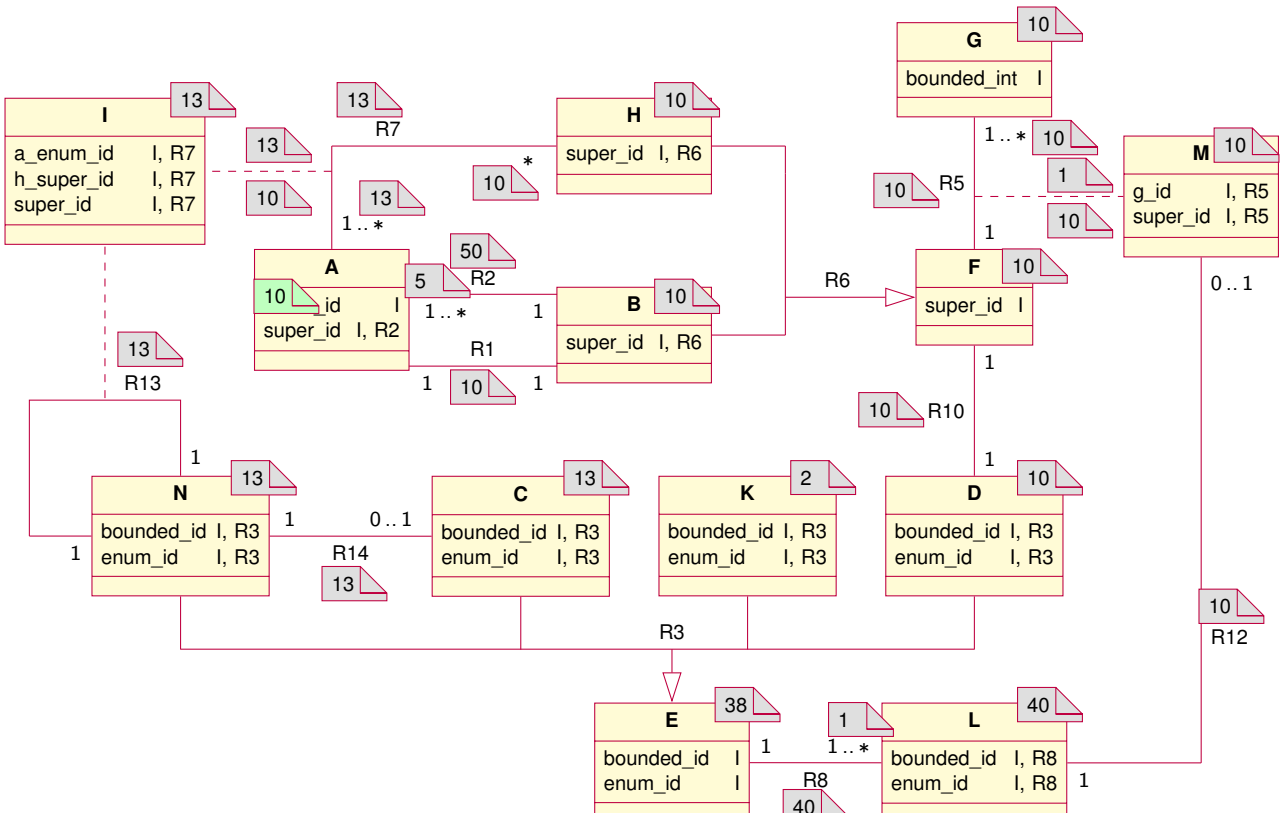
$s_{A,R7,I} \leqslant s_{A,R7,H}$

EXAMPLE



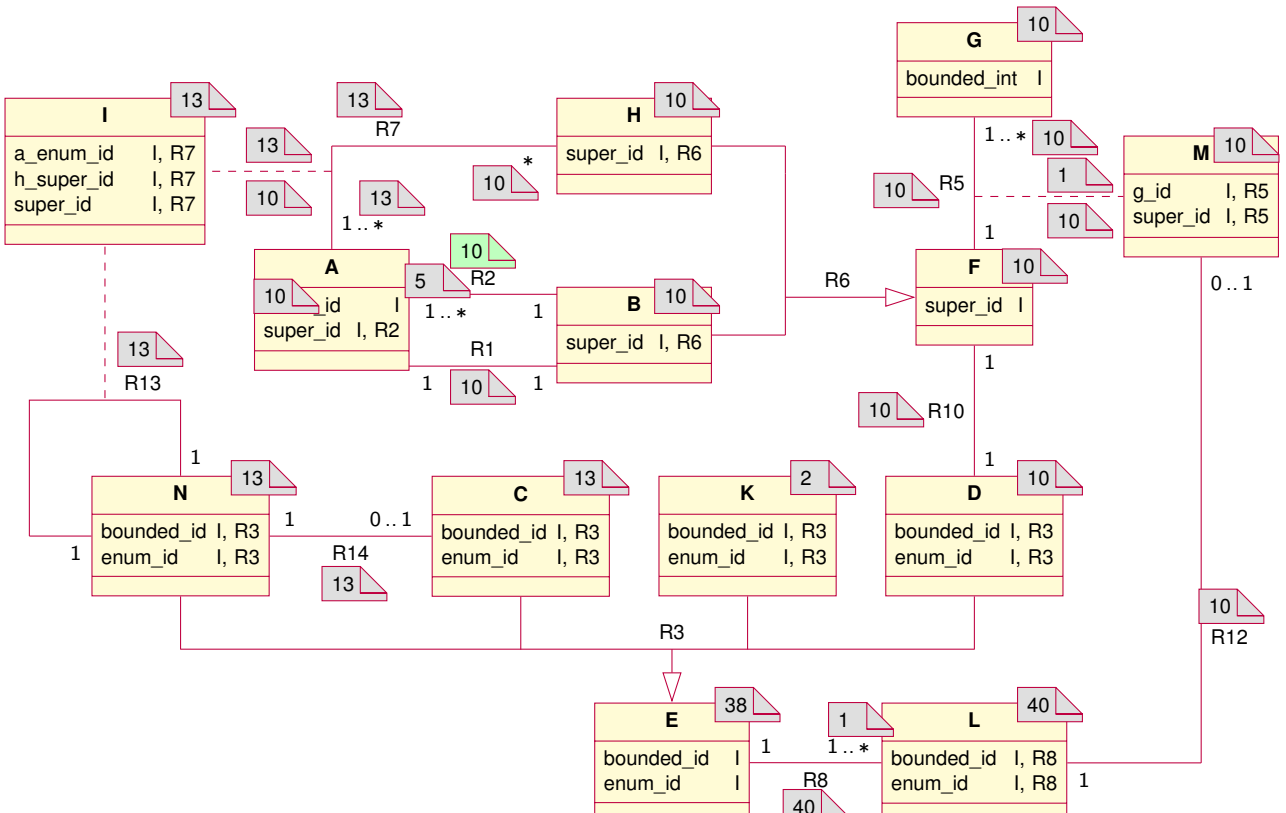
$$c_E \leqslant c_C + c_D + c_K + c_N$$

EXAMPLE



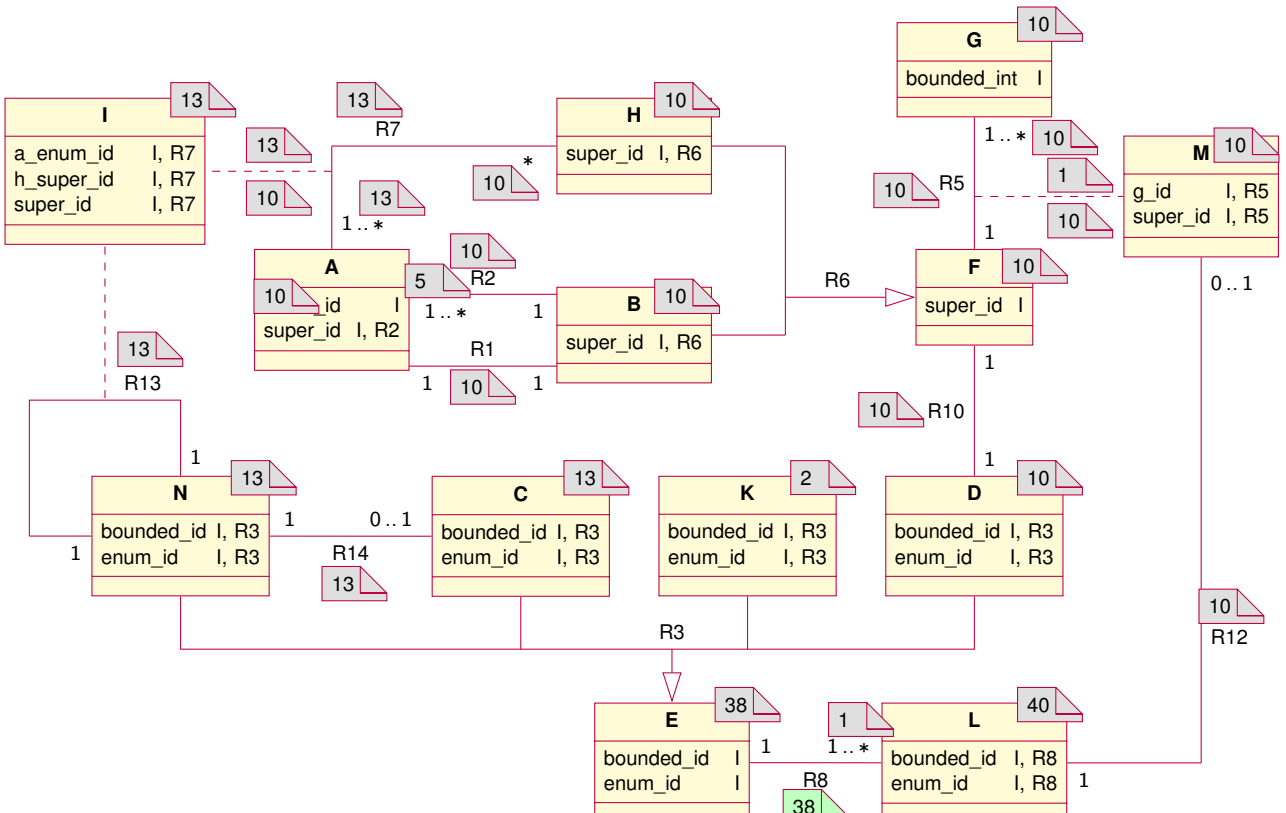
$c_A \leqslant r_{R1}$

EXAMPLE



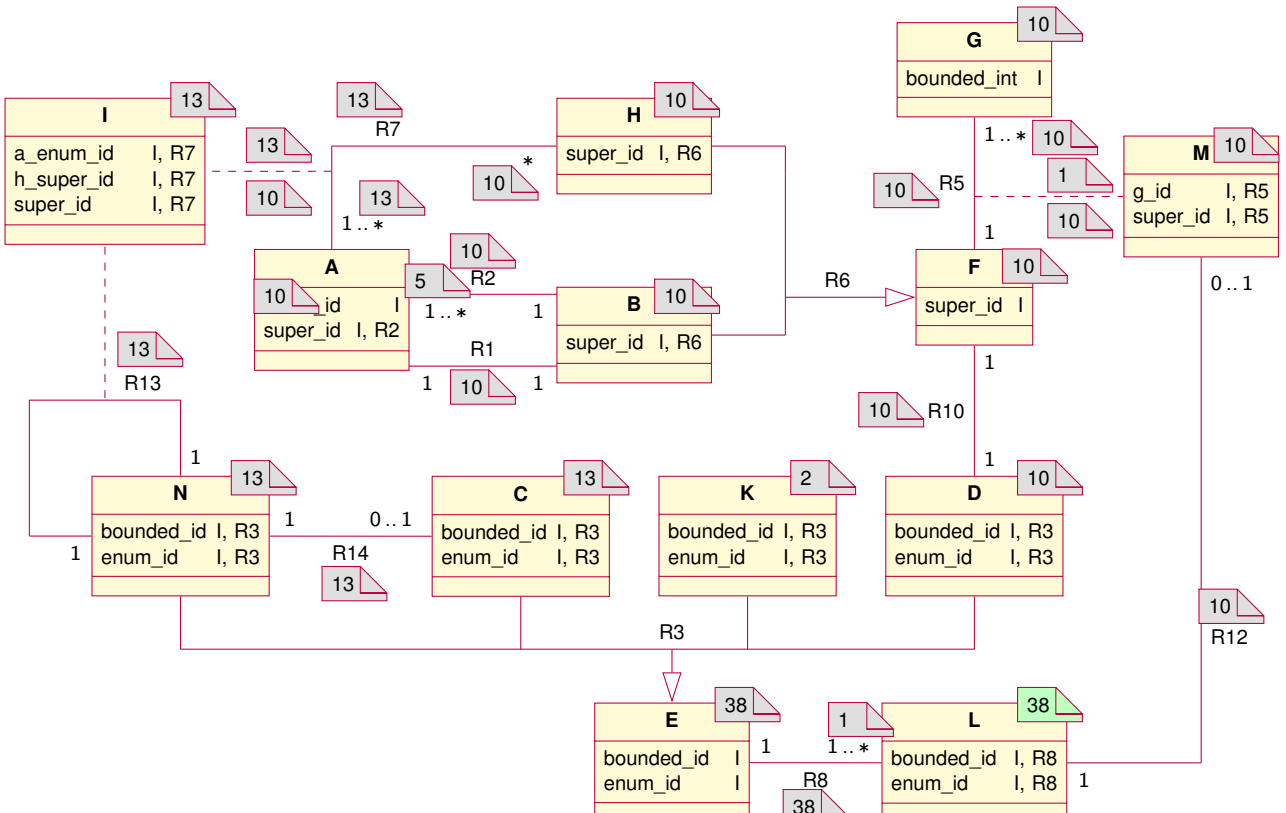
$r_{R2} \leqslant c_A$

EXAMPLE



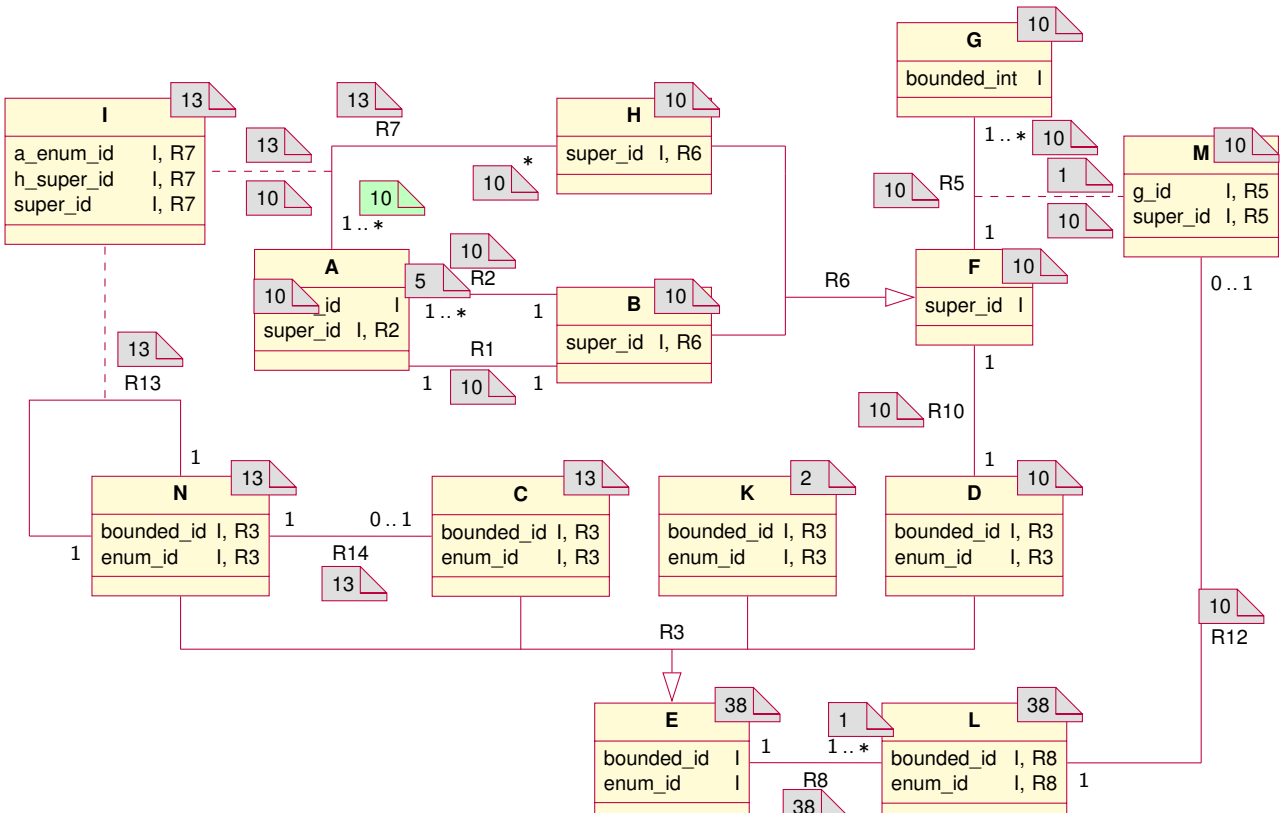
$r_{R8} \leqslant c_E \cdot s_{E,R8,L}$

EXAMPLE



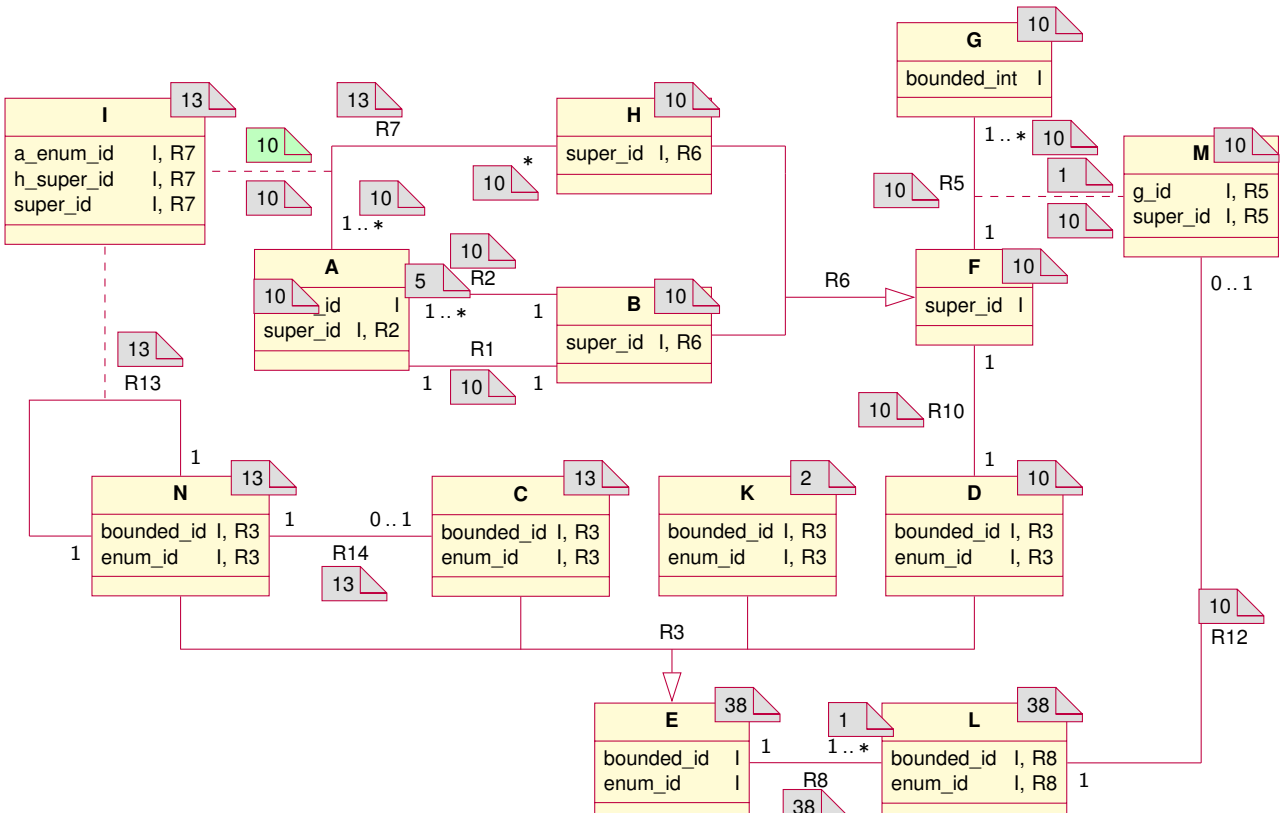
$c_L \leqslant r_{R8}$

EXAMPLE



$s_{H,R7,A} \leqslant c_A$

EXAMPLE



$s_{H,R7,I} \leqslant s_{H,R7,A}$

DIFFICULTIES

In example solution

Note that if there is an instance of class H,
then there is at least one instance of class A,
which means that there is at least one instance of class B.

Since there are at most 10 instance of class F,
this means that there are at most 9 instances of class H.

Our program fails to realize this.

DIFFICULTIES

In example solution

Note that if there is an instance of class H,
then there is at least one instance of class A,
which means that there is at least one instance of class B.

Since there are at most 10 instance of class F,
this means that there are at most 9 instances of class H.

Our program fails to realize this.

To overcome this we could reason not about upper bounds,
but about number of instances in a particular population of the model.
(This was what we did in the informal argument above.)

In this setting the generalization constraint $c_G \leqslant c_A + c_B + c_C$
would change to $c_G = c_A + c_B + c_C$.
This would allow us to get better bounds.

DIFFICULTIES

However, with this new approach we would most likely need to solve
one optimization problem for each variable in the constraint problem:
First find a solution that maximizes the first variable,
then find a solution that maximizes second variable, etc.

Also, solving the individual optimization problems
could become more challenging . . .

SAT

. . . In fact, this new optimization problem is no easier
than the Boolean satisfiability problem (SAT).

Given any Boolean formula, we can construct a model with a class that
can be instantiated if and only the formula is satisfiable.

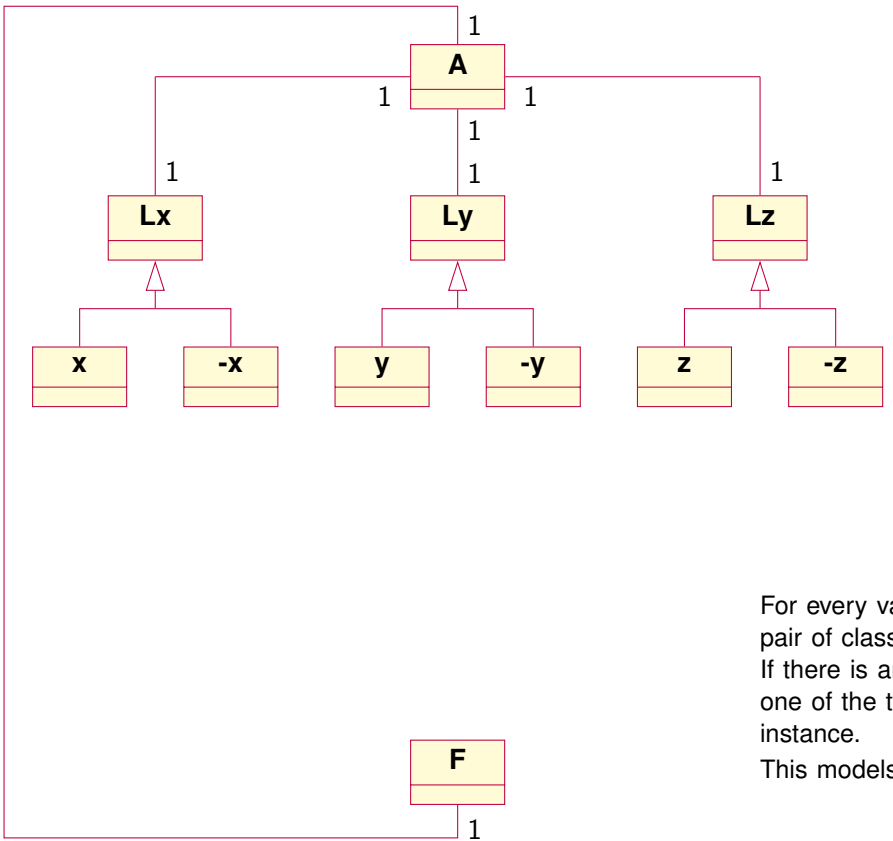
We can show this using the example

$$(x \vee y \vee \neg z) \wedge (\neg y \vee z).$$

Our model will have a class A that can have an instance
if and only if the formula is satisfiable.

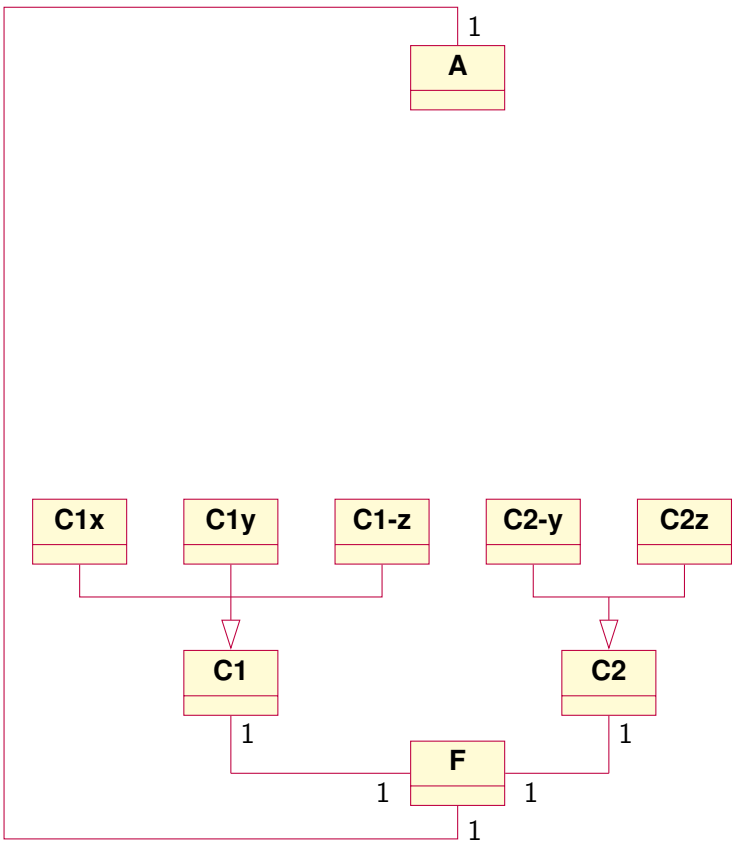
We assume that class A carries a single attribute identifier with
an enum data type that allows only a single value.
This ensures that class A never can have more than one instance.

SAT



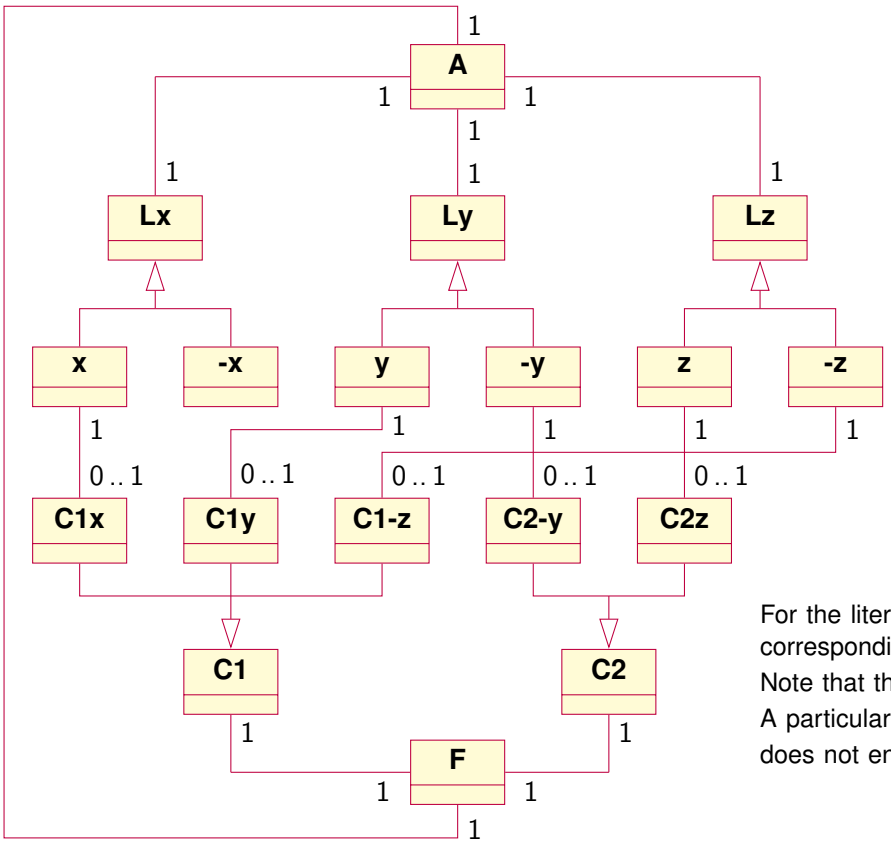
For every variable in the formula we add a pair of classes in the model.
If there is an instance of class A, then exactly one of the two classes for a variable has an instance.
This models a truth value for the variable.

SAT



For every clause in the formula we use the same idea to model the fact that the clause must be satisfied by a literal.
Of course, the clause might be satisfied by more than one literal, but we can always choose exactly one of them as a certificate.

SAT



For the literal certificate that is choosen, the corresponding variable must be true.
Note that the other direction is unconstrained: A particular truth assignment to variables does not enforce a particular literal choice.

SAT

So, given a population of the model where there is an instance of class A,
we can extract a satisfying assignment of truth values for the variables in the Boolean formula.

For the other direction:
Note that if the formula is satisfiable, then from the satisfying assignment of truth values
to the variables we get a way in which we can populate the model (with an instance of class A).

PROOFS

In addition to the more difficult constraint optimization problems
we also have the following issue:

Not obvious how to get proofs for bounds obtained with a brute-force solver

From the simple “upper bounds” solver we get, every time a bound is improved
a reason for why it could be done (a constraint originating from a particular
model construction).

This information is useful for validation of the obtained bounds.

For the more general constraint problems we probably would not get similar proofs.

Suggestions?

THANK YOU