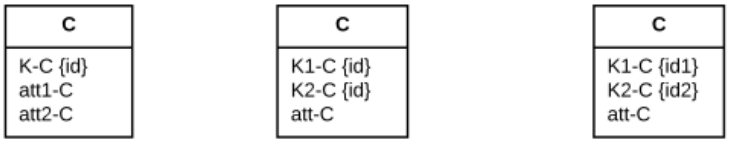
UML to RELATIONAL with rule names

Classes — R_Class



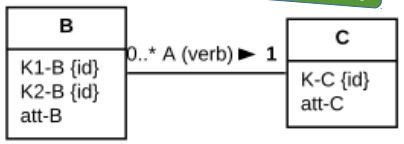
C(K-C, att1-C, att2-C);

C(K1-C, K2-C, att-C);
(Composed key)

C(K1-C, K2-C, att-C);
(Two candidate keys)

Associations & Association Classes

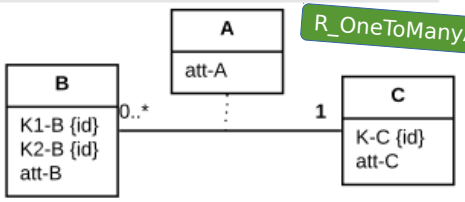
1 to Many — R_OneToMany



B(K1-B, K2-B, att-B, **K-C**); **C**(K-C, att-C);

+Constraints

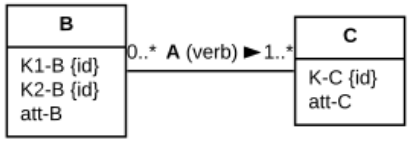
R_OneToManyAC



B(K1-B, K2-B, att-B, att-A, **K-C**); **C**(K-C, att-C);

+Constraints

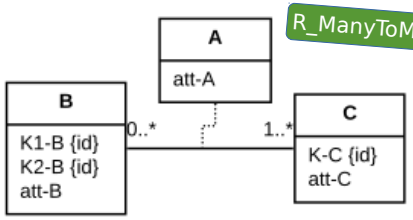
Many to Many — R_ManyToMany



B(K1-B, K2-B, att-B); **C**(K-C, att-C);
A (K1-B, K2-B, K-C)

+Constraints

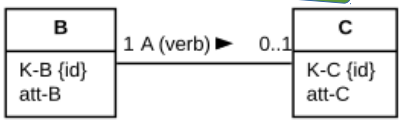
R_ManyToManyAC



B(K1-B, K2-B, att-B); **C**(K-C, att-C);
A (K1-B, K2-B, K-C, att-A)

+Constraints

1 to 1 — R_OneToOpt



B(K-B, att-B); **C**(K-C, att-C, K-B)

⚠ Not composed key but two candidate keys

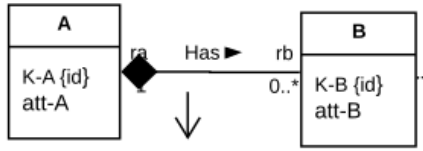
+Constraints

R_OneToOne

B(K-B, att-B); **C**(K-C, att-C, K-B)
OR
B(K-B, att-B, K-C); **C**(K-C, att-C);

+Constraints

Composition — R_Comp

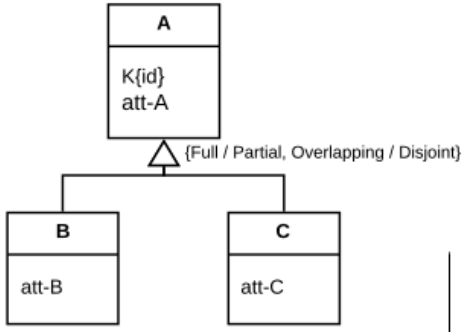


We will also use the notion of lid (local id) in order to specify that it is an id within a context (composition)

A(K-A, att-A); **B**(K-A, K-B, att-B)

+Constraints

Inheritance



R_PushUp

Unification (push-up)

Interesting if heavy overlapping

A(K, att-A, att-B, att-C, type_)

+Constraints

- ⊕ easy insertion
- ⊕ no incoherence when update
- ⊕ immediate access to subclasses
- ⚠ null values to be managed

R_PushDown

Duplication (push-down)

Interesting if full, disjoint (A not necessary)

A(K, att-A);
B(K, att-A, att-B); **C**(K, att-A, att-C)

+Constraints

- ⊕ immediate access to subclasses
- ⚠ need to control unity of the key
- ⚠ more space used
- ⚠ if double insertion -> redondance

R_Reference

Reference

Interesting if full, disjoint (A not necessary)

A(K, att-A); **B**(K, att-B); **C**(K, att-C)

+Constraints

- ⊕ less space used
- ⊕ no incoherence when update
- ⚠ two insertions needed for subclasses
- ⚠ join needed for accesing subclasses