Relationship	ERD	Rails / AR	Example	Comments
One to One	Side A: line (may have "1" above line)	child: belongs_to	Address belongs_to Customer	For a one-to-one, you need to pick a side that is the child and a side that is the parent. It might be somewhat arbitrary but consider the order of creation and deletion of each side.
	Side B: line (may have "1" above line)	parent: has_one	Customer has_one Address	
One to Many	Side A: line (may have "1" above line)	parent: has_many	Order belongs_to Customer	Side A, the one side (aka the parent) is created first. Side B, the many side (aka the child) is created last and depends on the parent.
	Side B: line with crows feet (or have astrick above line)	child: belongs_to	Customer has_many orders	
Many to Many (no Association Model)	Side A: line with crows feet (or have astrick above line)	Side A: has_and_belongs_to_many	Movie has_and_belongs_to_many actors	In a many-to-may we need a place to store the foreign keys between Side A and Side B. Thus an association table is created for us to manage all of the associations.
	Side B: line with crows feet (or have astrick above line)	Side B: has_and_belongs_to_many	Actor has_and_belongs_to_many movies	
Many to Many (with Association Model = AM)	AM: one-to-many to A	AM belongs_to A	Customer has_many reservations	In this many-to-many, the assocation model contains data that has business meaning (i.e. it isn't just for storing the foreign keys). In our example the AM would have the start_date and the end_date for the reservation. Therefore we need to have a dedicated Model class for the association. In our example this would be the Reservation class.
	AM: one-to-many to B	AM belongs_to B	Car has_many reservations	
		A has_many AM	Reservation belongs_to a customer	
		B has_many AM	Reservation belongs_to a car	
		A has_many B through AM	Customer has_many cars through Reservation	
		B has_many A through AM	Car has_many customers through Reservation	