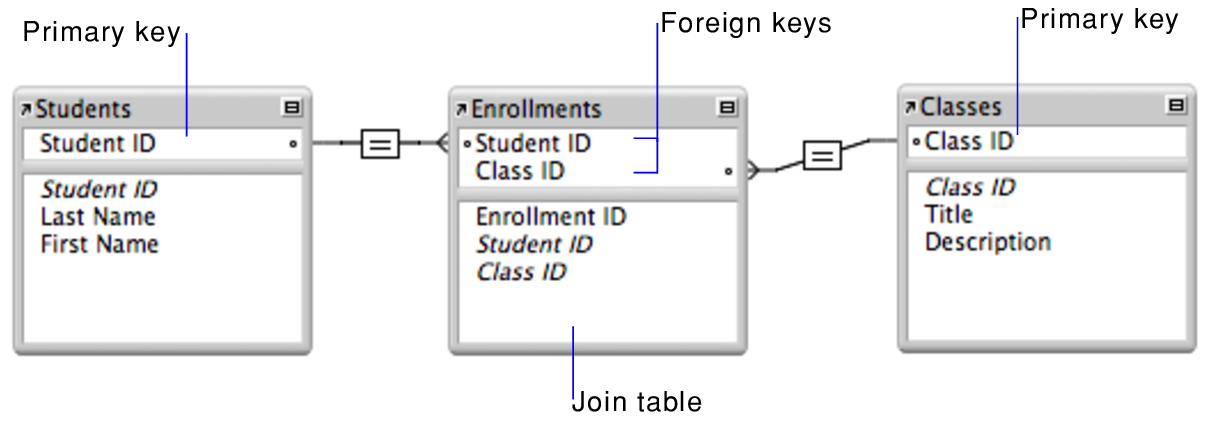
Model and Database Relationships Documentation

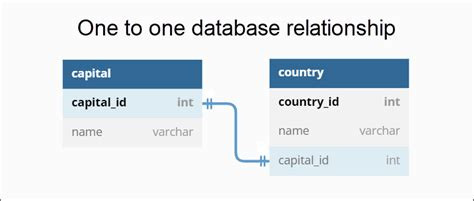
**Database Table Relationships**

Primary Key: In Rails, every table(or model) will have a primary key that is unique for every record.

Foreign Key: Foreign key is used to create relationships between two tables by associating row of one table with that of another.



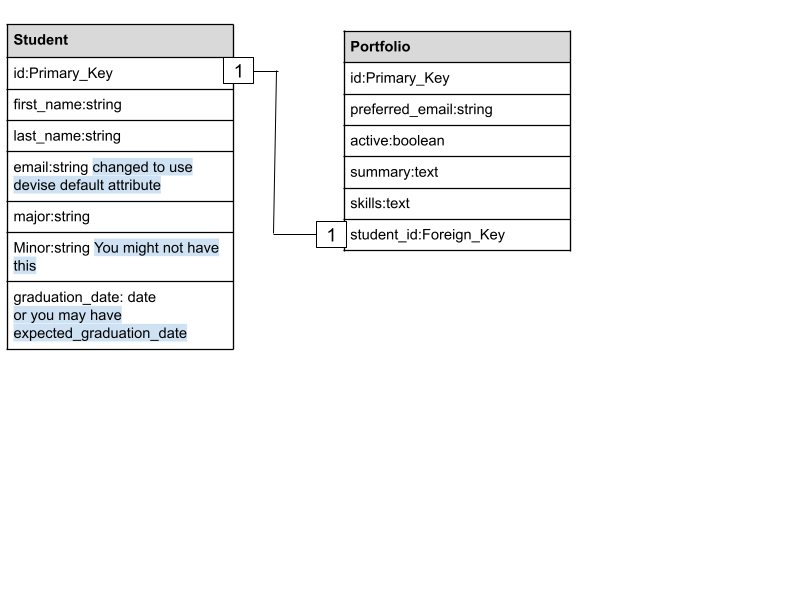
One to One:



Definition: A one to one relationship connects one entity to one other entity

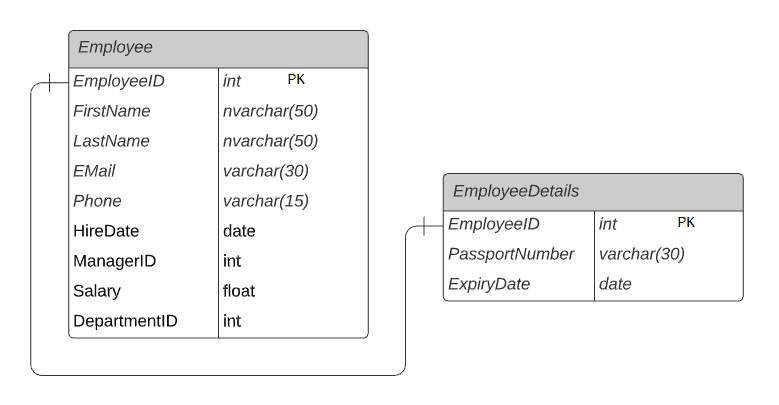
Examples:

In our app we have been working on in class our one to one relationship would be a student has\_one portfolio.

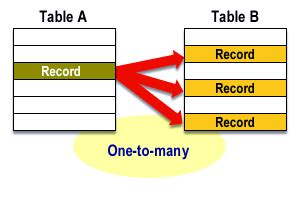


<https://www.tutorialsteacher.com/sqlserver/tables-relations>

Another example could be an employee having one employee detail, where the employee detail could include the employee ID, passport number, and expiry date.



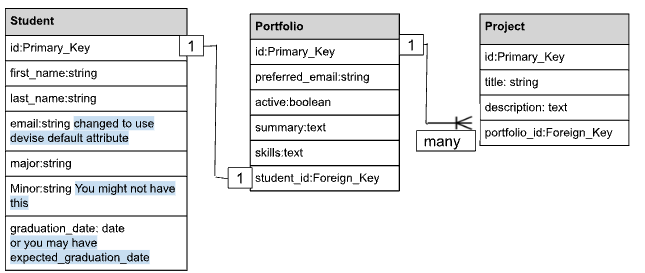
One to Many:



Definition: A single record from one table can be linked to zero or more rows in another table

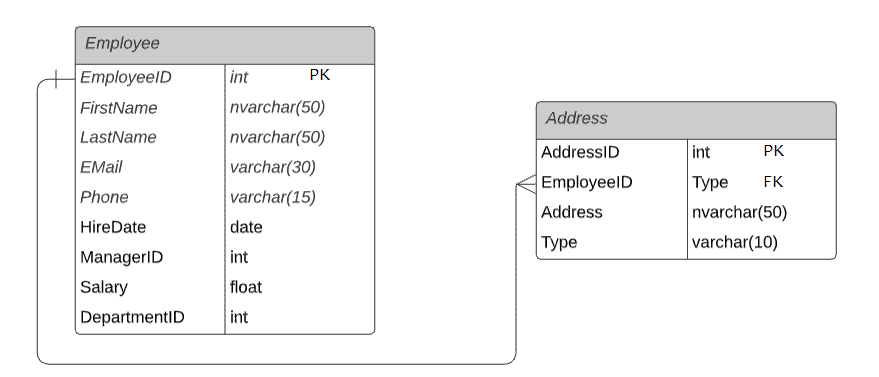
Examples:

In our app we have been working on in class our one to many relationship would be one portfolio to many projects.

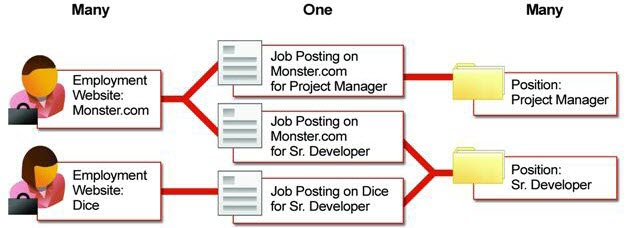


<https://www.tutorialsteacher.com/sqlserver/tables-relations>

Using that same example above an employee ID can have multiple addresses



Many to Many:



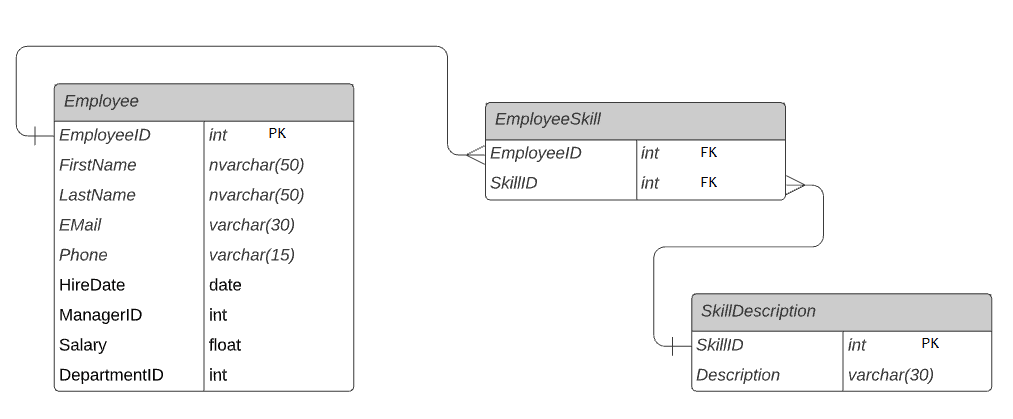
Definition: Many to many relationship lets you relate each row in one table to many rows in another table and vice versa

Examples

Although we didn't do a many to many relationship in our portfolio app, a different good example could be a group can have many memberships and a group can have many users through those memberships.

<https://www.tutorialsteacher.com/sqlserver/tables-relations>

Using that same example an employee table can have many skills from the employee skill table and also one skill can be associated with one or more employees



Useful Keywords:

belongs\_to :is a one to one relationship, it establishes a foreign key in the current model's table pointing to the associated model.

has\_one :one to one, indicating that this model owns or has a single instance of another model.

has\_many :one to many, indicating that this model can have multiple instances of another model.

has\_many :through :many to many, enabling more detailed control over the relationship. Useful when you want additional data on the relationship itself.

has\_one :through :one to one, often used when you want to retrieve a single associated object indirectly.

has\_and\_belongs\_to\_many :many to many, suitable for simpler many-to-many relationships where no extra information about the association is needed.

Helpful Resources:

<https://www.tutorialsteacher.com/sqlserver/tables-relations>

<https://blog.dennisokeeffe.com/blog/2022-03-16-understanding-rails-associations>

<https://guides.rubyonrails.org/v7.1/association_basics.html#the-has-one-association>

<https://guides.rubyonrails.org/v7.1/association_basics.html>

<https://medium.com/@natekontny/handling-slow-cascading-deletes-in-rails-f2581f34c186>

<https://ashton.codes/the-difference-between-on_delete-restrict-nullify-and-cascade/>