



Curso completo de Python!





Instrutor: Vitor Mazuco

<http://facebook.com/vitormazuco>

Email: vitor.mazuco@gmail.com

WebSite:

<http://vmzsolutions.com.br>



N-N no SQLAlchemy

No conceito de muitos para muitos, temos um exemplo de muitos analistas poder acessar diversos servidores.

```
association_table =  
Table('analistas_servidores', Base.metadata,  
      Column('analistas_id', Integer, ForeignKey('analistas.id')),  
      Column('servidores_id', Integer, ForeignKey('servidores.id')))
```



N-N no SQLAlchemy

Essa tabela foi do tipo associativa, juntando assim os analistas aos seus servidores. Nesse caso, não precisa de uma classe, pois nunca ela será acessada de forma direta. Ou seja, sempre será necessário entrar pela tabela principal para assim buscar informações de servidores em uma tabela secundária.



N-N no SQLAlchemy

Em nosso script, vamos criar duas tabelas para assim criar o nosso BD. O seu relacionamento agora é por uma tabela secundária que fará a organização pelo analista x servidor.

* Veja mais no arquivo N-N_ALC.py



N-N no SQLAlchemy

Apoós executar o script, vejamos os resultados dessa aplicação:

```
$ sqlite3 banco2.db
```

```
> .tables
```

```
> select * from analistas;
```

```
> select * from analistas_servidores;
```

```
> select * from servidores;
```



N-N no SQLAlchemy

Observe que foram criados 3 tabelas diferentes, mas a tabela `analista_servidores` tem apenas os id's dos analistas e seus servidores, porque seria inviável usar um relacionamento 1 > N. Mas ainda os tratamos como se fossem listas.