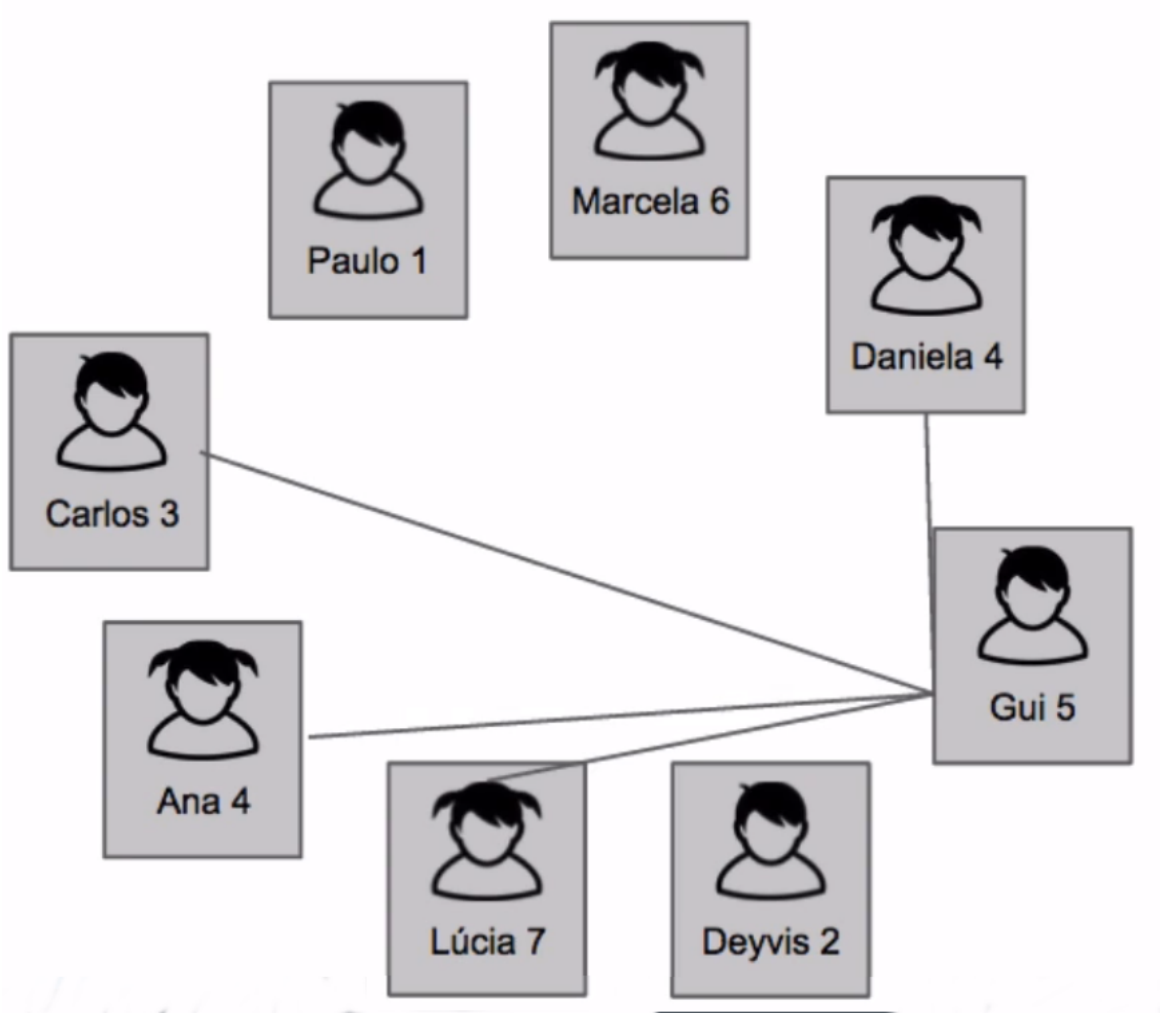
05

Rede social e relacionamento

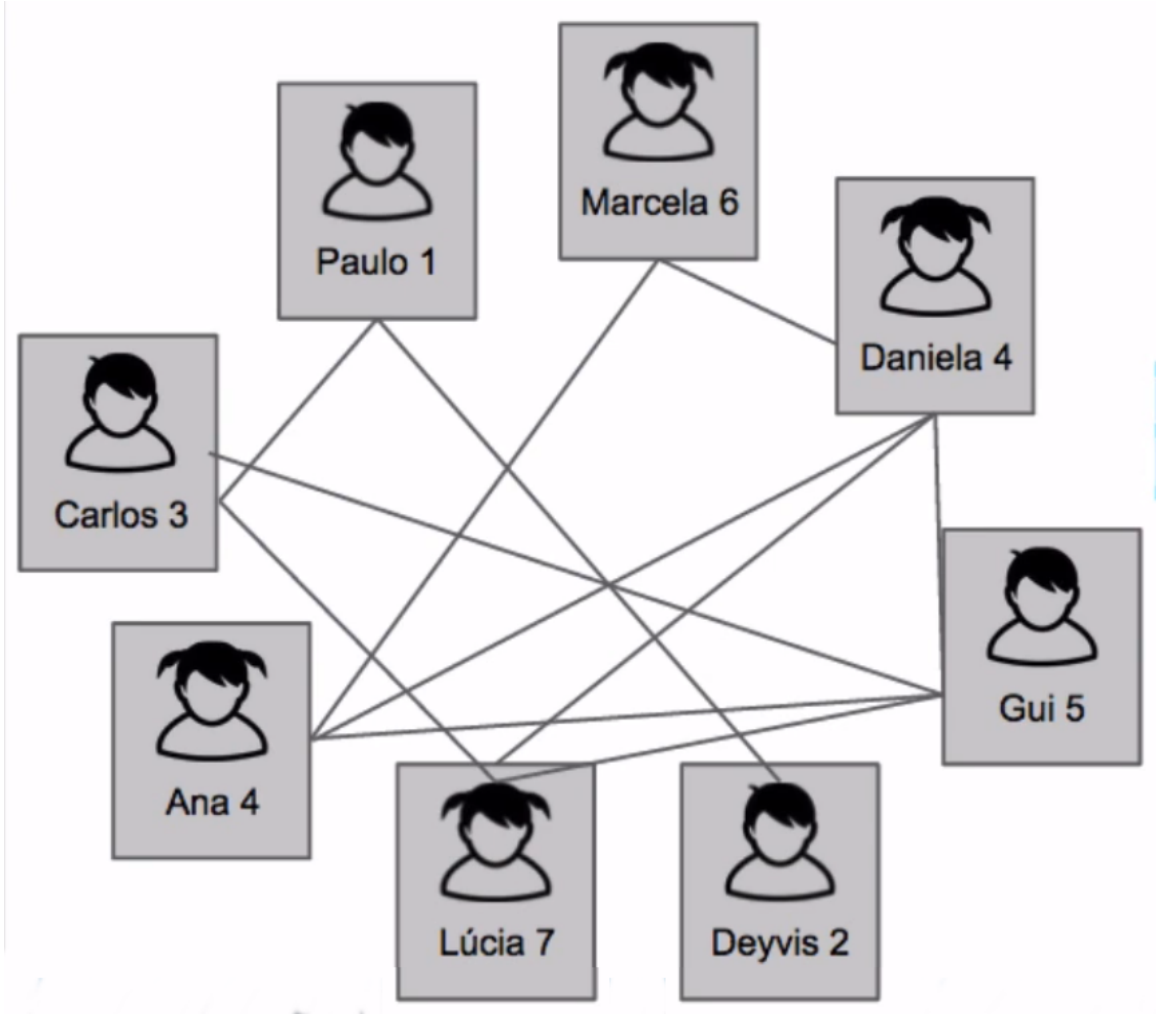
Capítulo 4 - Rede social e relacionamento

Um exemplo de conjunto de informações que poderíamos querer utilizar no dia-a-dia é o de amigos em uma rede social.



Aqui mostramos o Guilherme, que possui quatro amigos: Daniela, Carlos, Ana e Lúcia. Poderíamos fazer isso para todos desse grupo. Cada linha representa uma conexão. Isso se aplica para contatos telefônicos ou qualquer tipo de relacionamento entre elementos. Então um *Conjunto* de elementos são esses elementos juntos por uma característica em comum.

Não podemos falar de lista aqui, pois esta traz a ideia de ordem. Se não existe a enumeração dos elementos, podemos fazer diversas operações entre esses conjuntos como, por exemplo, dizer quem são os amigos em comum entre o Guilherme e o Carlos.



Os amigos do Guilherme são:

- Daniela
- Carlos
- Ana
- Lúcia

Os amigo do Carlos são:

- Paulo
- Lúcia
- Guilherme

Os amigos em comum entre Guilherme e Carlos são:

- Lúcia

Vimos a intersecção de dois conjuntos. Veremos que serão possíveis outras operações como união, diferença, etc e faremos tudo isso no *Redis*.

Para representar um conjunto (set) no *Redis*, criamos uma chave (relacionamentos do Guilherme) e adicionaremos *strings* (amigos) a ela:

```
SADD "relacionamentos:guilherme" "daniela" "carlos" "ana" "lucia"
(integer) 4
```

Se tentarmos adicionar elementos que já pertencem ao conjunto, teremos:

```
SADD "relacionamentos:guilherme" "ana" "lucia"
(integer) 0
```

Não foram adicionados, pois uma das características dos conjuntos é que não existem elementos repetidos, diferentemente da lista.

Para visualizarmos a quantidade de elementos (cardinalidade) em um conjunto fazemos:

```
SCARD "relacionamentos:guilherme"
(integer) 4
```

E para discriminar os "membros" do conjunto:

```
SMEMBERS "relacionamentos:guilherme"
1) "lucia"
2) "ana"
3) "carlos"
4) "daniela"
```

Para descobrirmos se um elemento pertence ao conjunto:

```
SISMEMBER "relacionamentos:guilherme" "carlos"
(integer) 1
SISMEMBER "relacionamentos:guilherme" "joao"
(integer) 0
```

O que significa que o elemento "carlos" está dentro do conjunto e o "joao" não.

Para remover um membro:

```
SREM "relacionamentos:guilherme" "lucia"
(integer) 1
```

Voltando agora à intersecção de conjuntos, vimos que, no caso dos amigos do Guilherme e do Carlos, a Lúcia é a amiga em comum entre eles. Veremos como implementar tal operação no *Redis*. Primeiro, criemos o conjunto de relacionamentos do Carlos:

```
SADD "relacionamentos:carlos" "paulo" "lucia" "guilherme"
(integer) 3
```

Agora basta fazer

```
SINTER "relacionamentos:guilherme" "relacionamentos:carlos"
1) "lucia"
```

para saber o elemento em comum.

Uma outra pergunta que pode ser feita em relação aos conjuntos de amigos é: quem o Guilherme conhece que a marcela não conhece? Isso nada mais é do que a diferença entre dois conjuntos.

```
amigos do guilherme - amigos da marcela

daniela, carlos, ana, lucia -
daniela, ana =
carlos, lucia
```

Implementando no *Redis*, vamos criar o conjunto de amigos da Marcela:

```
SADD "relacionamentos:marcela" "daniela" "ana"
(integer) 1
```

E fazer a diferença:

```
SDIFF "relacionamentos:guilherme" "relacionamentos:marcela"
1) "carlos"
2) "lucia"
```

Se invertêssemos os conjuntos:

```
SDIFF "relacionamentos:marcela" "relacionamentos:guilherme"
(empty list or set)
```

De fato, não existe membro do conjunto da Marcela que não está no conjunto do Guilherme.

Represente o que representar os conjuntos, a ***intersecção*** e a ***diferença*** falam muito sobre eles.