Mapeamento Estrutural e de Operações

Mapeamento OO - Relacional

Mapeamento Estrutural

- Classes / Atributos
- Associações
- Herança

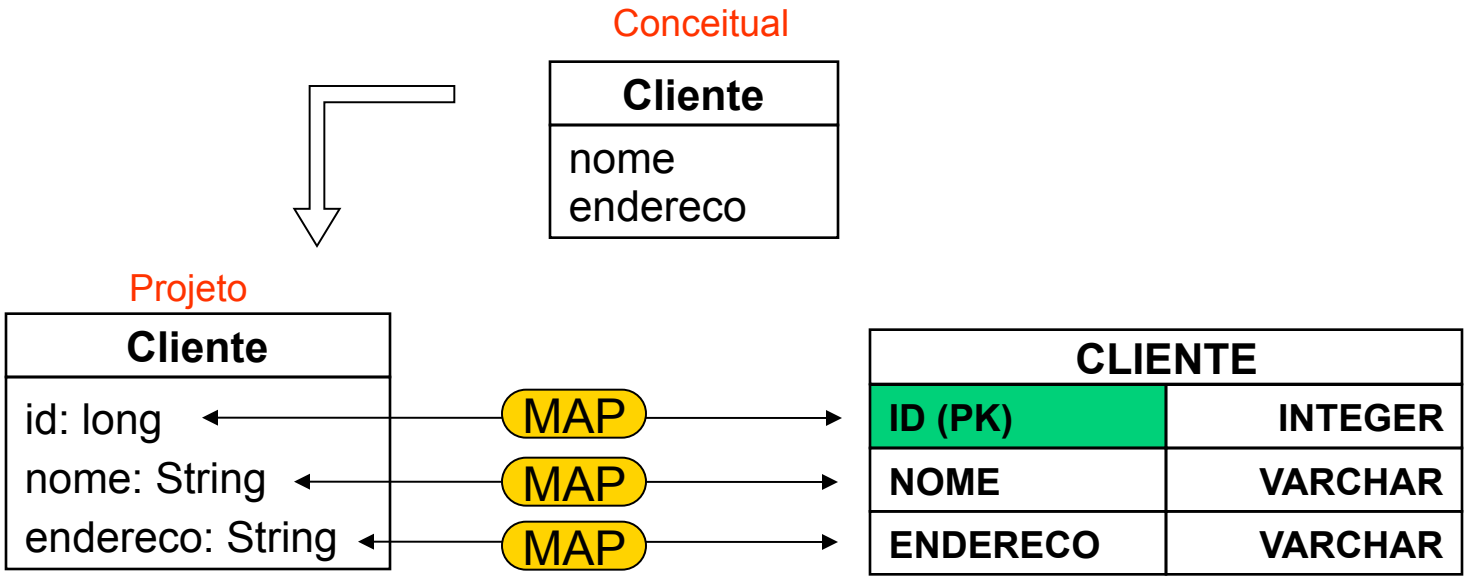
Mapeamento de Operações na Base de Dados

- Recuperação ou Instanciação
- Inserção
- Atualização
- Remoção

Classes - Mapeamento Estrutural

- Para cada atributo mapeado deve existir uma coluna na tabela correspondente cujo domínio é compatível com o tipo do atributo.

- O atributo identificador da classe deve estar mapeado para a coluna correspondente à chave primária. Caso não exista um atributo identificador na classe original, é necessário criá-lo.



Classes - Mapeamento Operações

- Recuperação:

cli := new Cliente()

cli.id,cli.nome,cli.endereco:= SELECT ID, NOME, ENDERECO FROM CLIENTE WHERE ID = oid

- Inserção:

INSERT INTO CLIENTE(ID,NOME,ENDERECO) VALUES (cli.id, cli.nome, cli.endereco)

- Atualização:

UPDATE CLIENTE SET NOME = cli.nome, ENDERECO = cli.endereco WHERE ID = cli.id

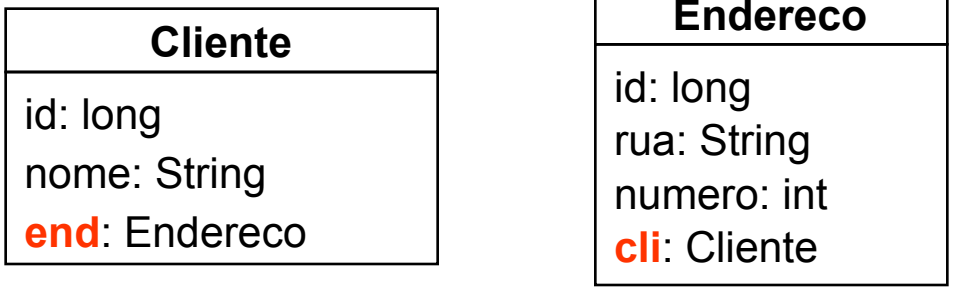
- Remoção:

DELETE FROM CLIENTE WHERE ID = cli.id

Associações 1:1 - Mapeam. Estrutural

➡ As classes associadas podem ser mapeadas para uma mesma tabela ou tabelas separadas.

Projeto

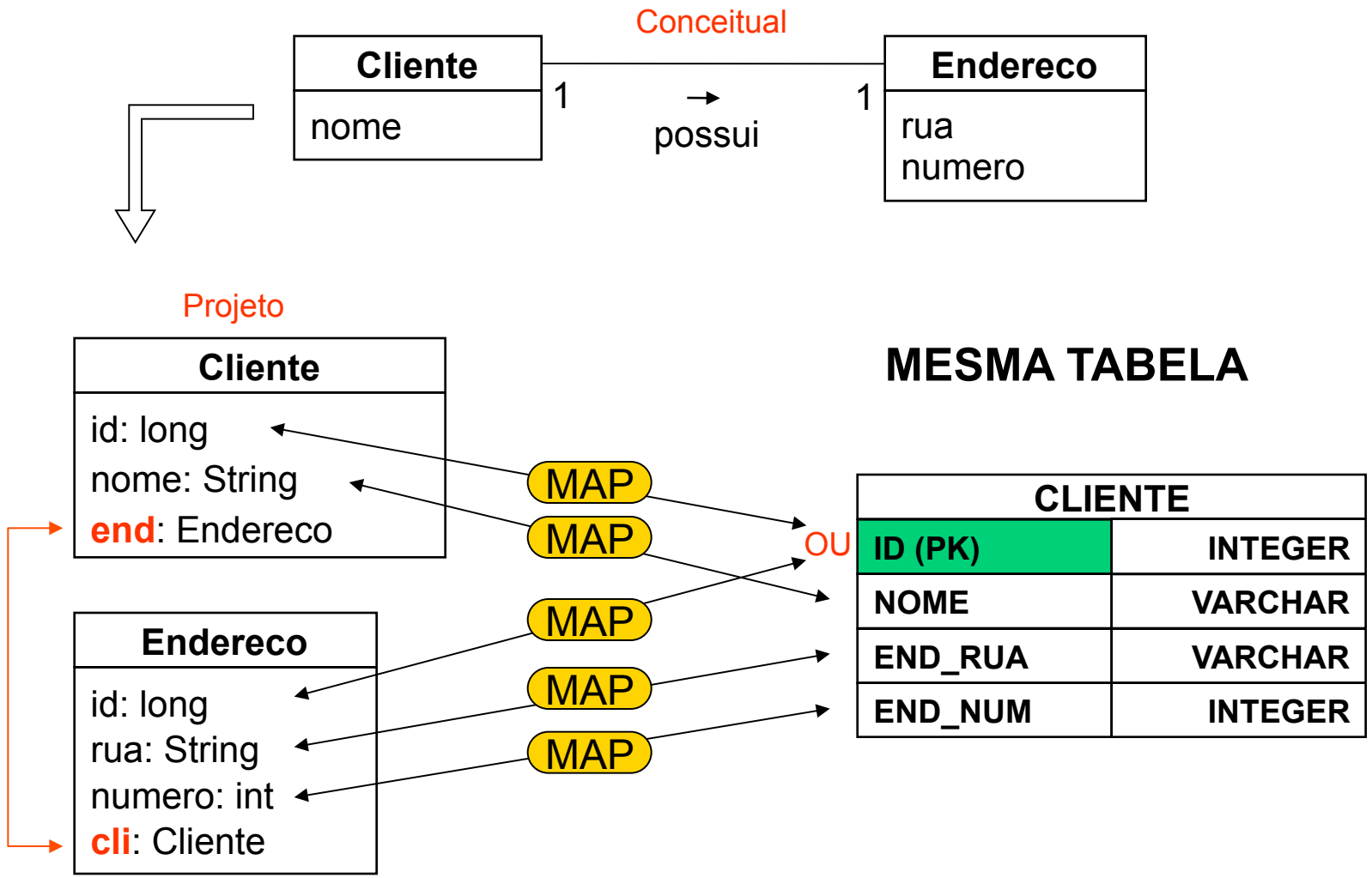


➔ Se as classes são mapeadas para uma mesma tabela, o mapeamento é direto.

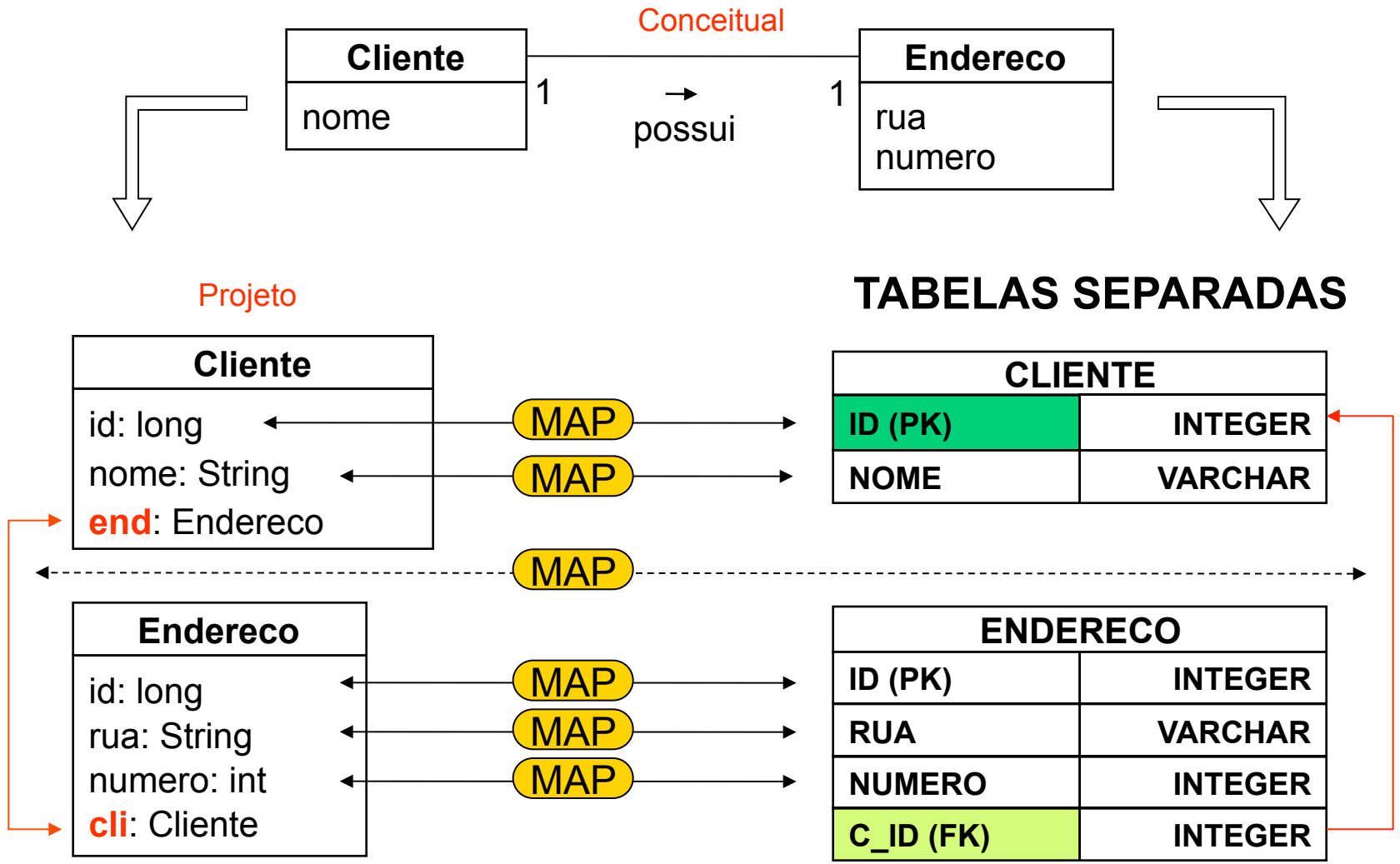
➔ Se as classes mapeiam para tabelas separadas, uma chave estrangeira precisa estar definida em uma das tabelas.

Obs: É conveniente que as classes associadas refiram-se mutuamente através de referências inversas.

Associações 1:1 - Mapeam. Estrutural



Associações 1:1 - Mapeam. Estrutural

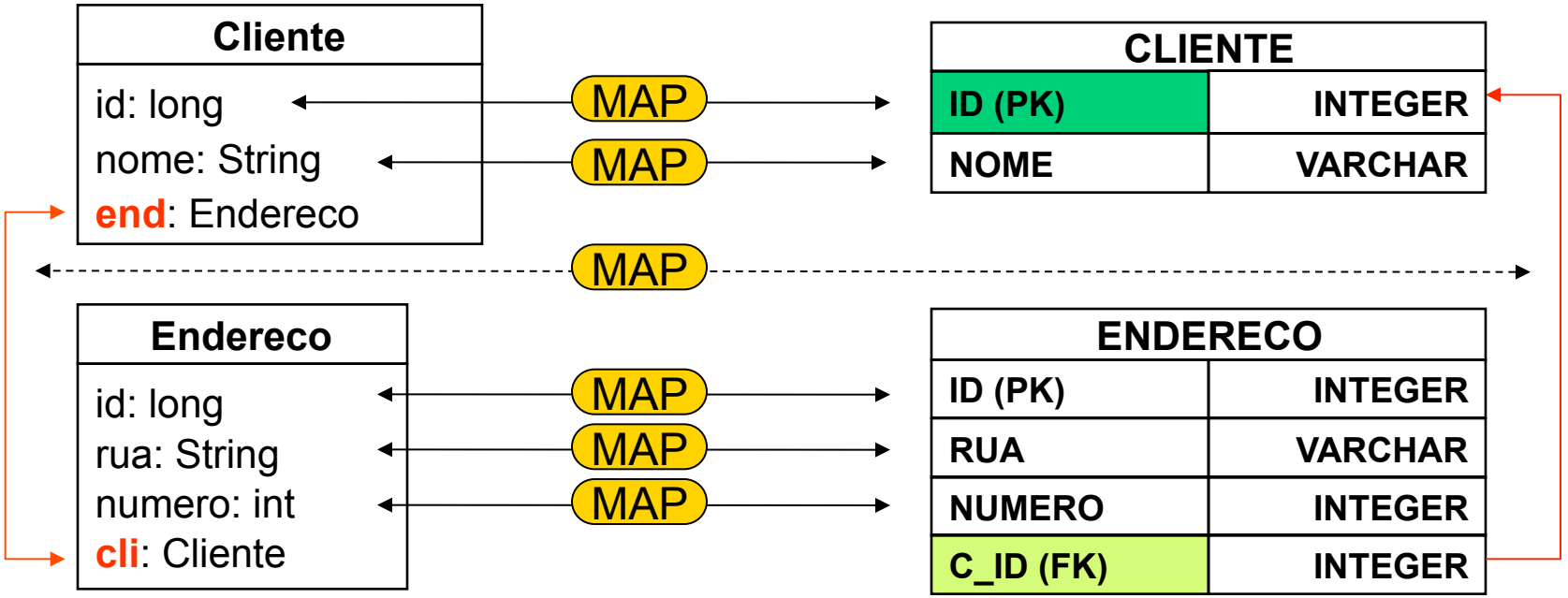


Associações 1:1 - Mapeam. Operações

Caso 1: Cliente(1) –> Endereco(1)

- Recuperação:a partir do oid de um cliente

```
cli := new Cliente()
cli.id,cli.nome := SELECT ID, NOME FROM CLIENTE WHERE  ID = oid
end := new Endereco()
end.id,end.rua,end.numero := SELECT ID, RUA, NUMERO FROM ENDERECO
                                WHERE C_ID = cli.id
cli.end := end
end.cli := cli
```



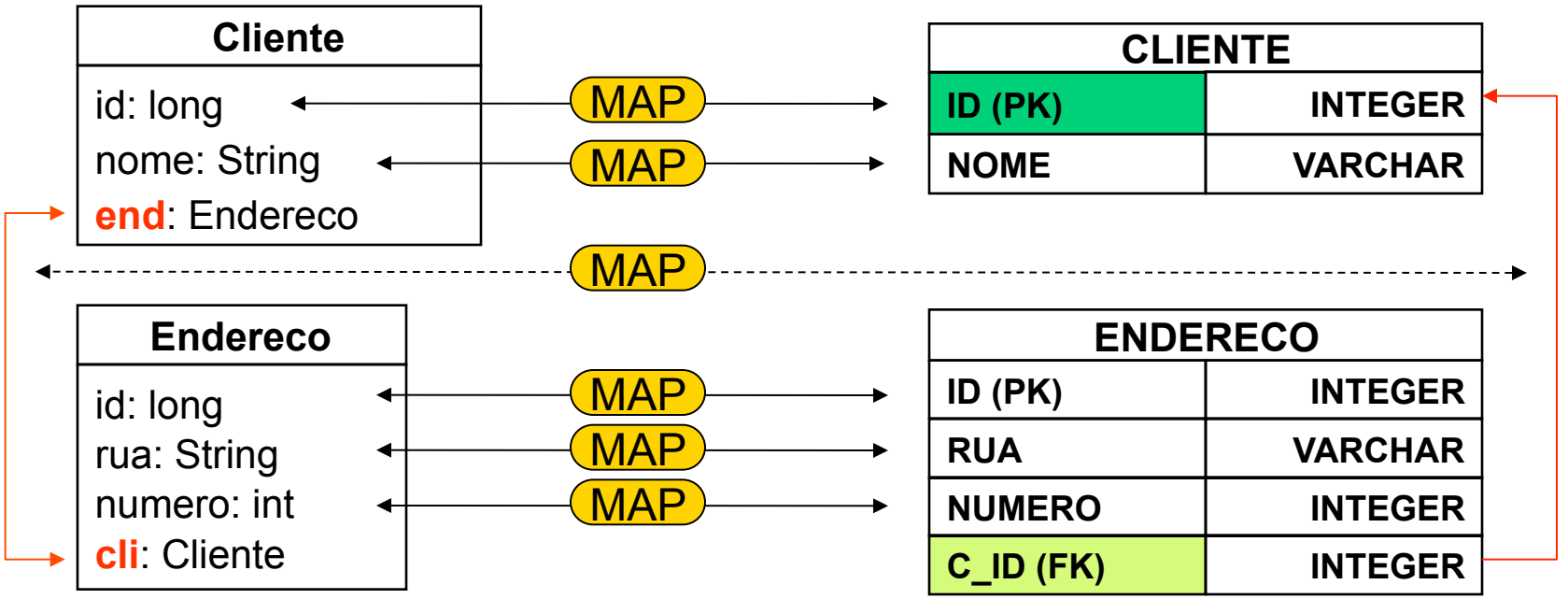
Associações 1:1 - Mapeam. Operações

Caso 1: Cliente(1) –> Endereco(1)

- Inserção: a partir de um cliente

INSERT INTO CLIENTE(ID,NOME) VALUES (cli.id,cli.nome)

INSERT INTO ENDERECO(ID,RUA,NUMERO,C_ID) VALUES (cli.end.id, cli.end.rua,
cli.end.numero, cli.id)

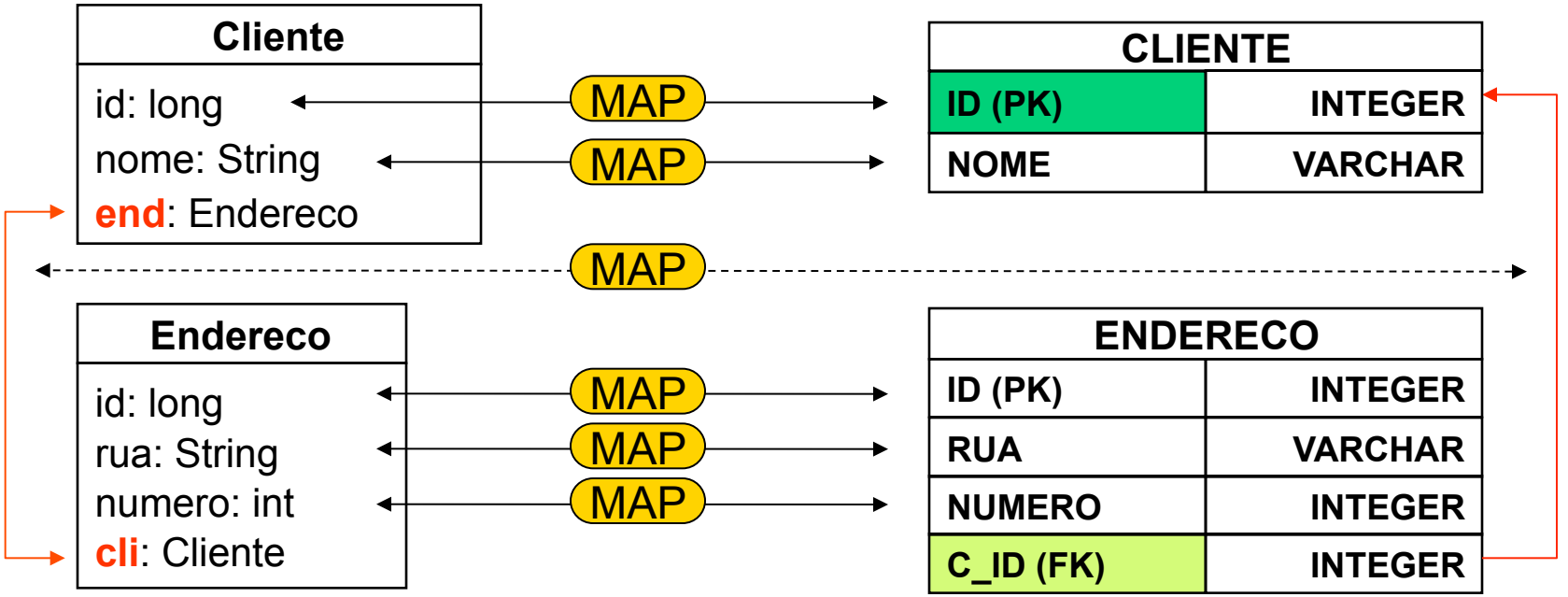


Associações 1:1 - Mapeam. Operações

Caso 1: Cliente(1) –> Endereco(1)

- Atualização: a partir de um cliente

```
UPDATE CLIENTE SET NOME = cli.nome  WHERE ID = cli.id
UPDATE ENDERECO SET RUA = cli.end.rua, NUMERO = cli.end.numero
                    WHERE ID = cli.end.id
```



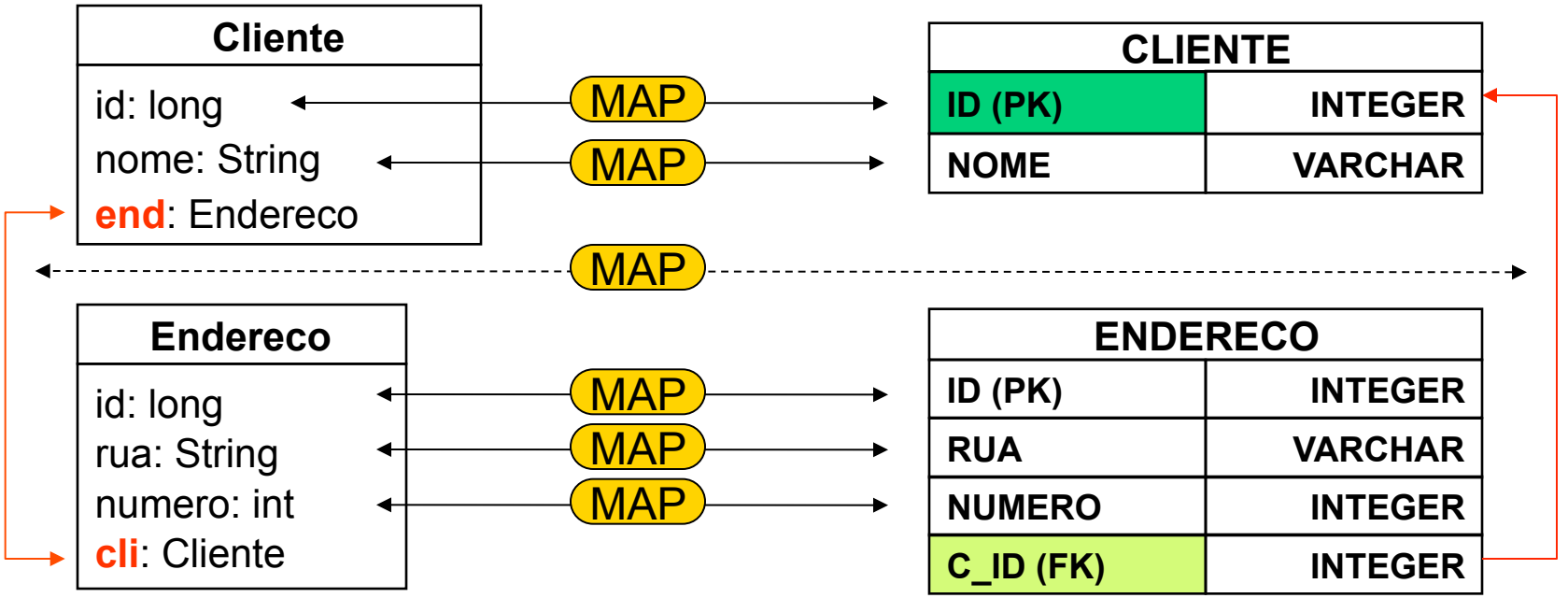
Associações 1:1 - Mapeam. Operações

Caso 1: Cliente(1) –> Endereco(1)

- Remoção: a partir de um cliente

DELETE FROM ENDERECO WHERE C_ID = cli.id * OU
UPDATE ENDERECO SET C_ID = NULL WHERE C_ID = :cli.id **

DELETE FROM CLIENTE WHERE ID = cli.id



*cardinalidade mínima de Cliente em Endereço é 1 (Endereço tem que ter um Cliente)

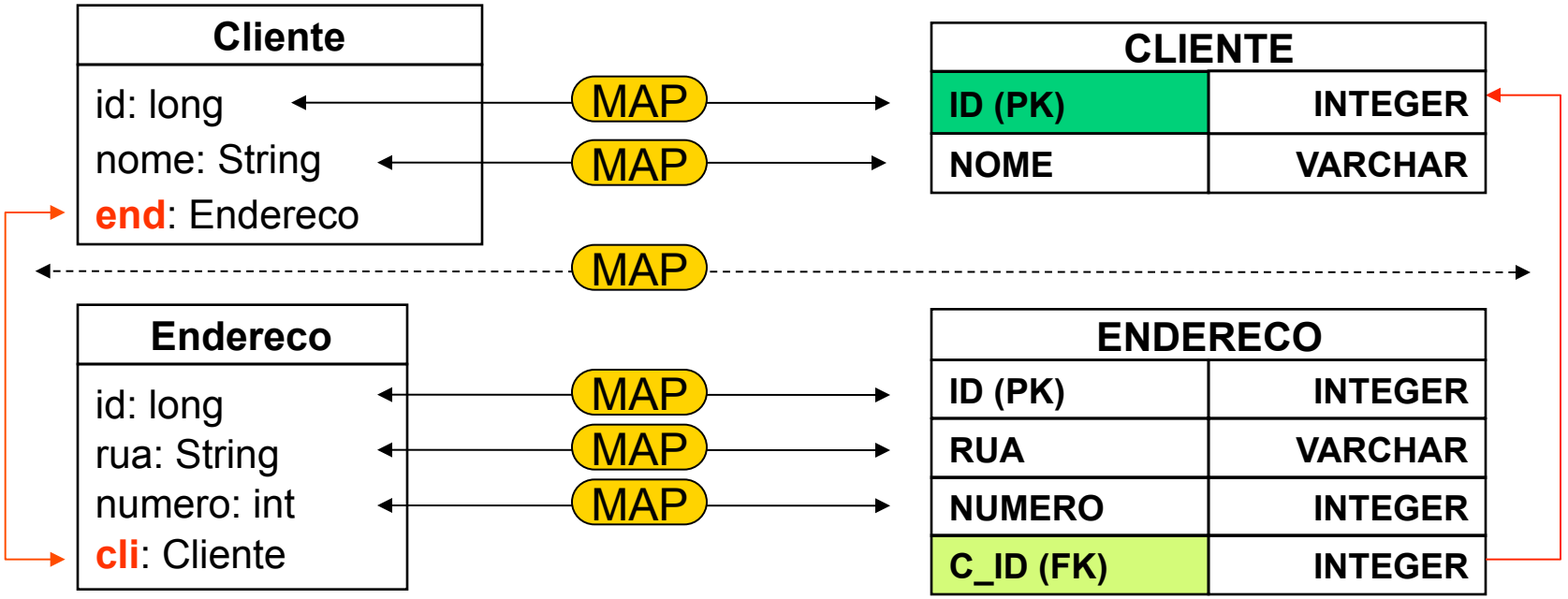
** cardinalidade mínima de Cliente em Endereco é 0 (Endereço pode não ter um Cliente)

Associações 1:1 - Mapeam. Operações

Caso 2: Endereco(1) –> Cliente(1)

- Recuperação: a partir do oid de um endereço

```
end := new Endereco()
end.id,end.rua,end.numero, var_cid := SELECT ID, RUA, NUMERO, C_ID
                                      FROM ENDERECO WHERE ID = oid
cli := new Cliente()
cli.id,cli.nome := SELECT ID, NOME FROM CLIENTE WHERE ID = var_cid
end.cli := cli
cli.end := end
```



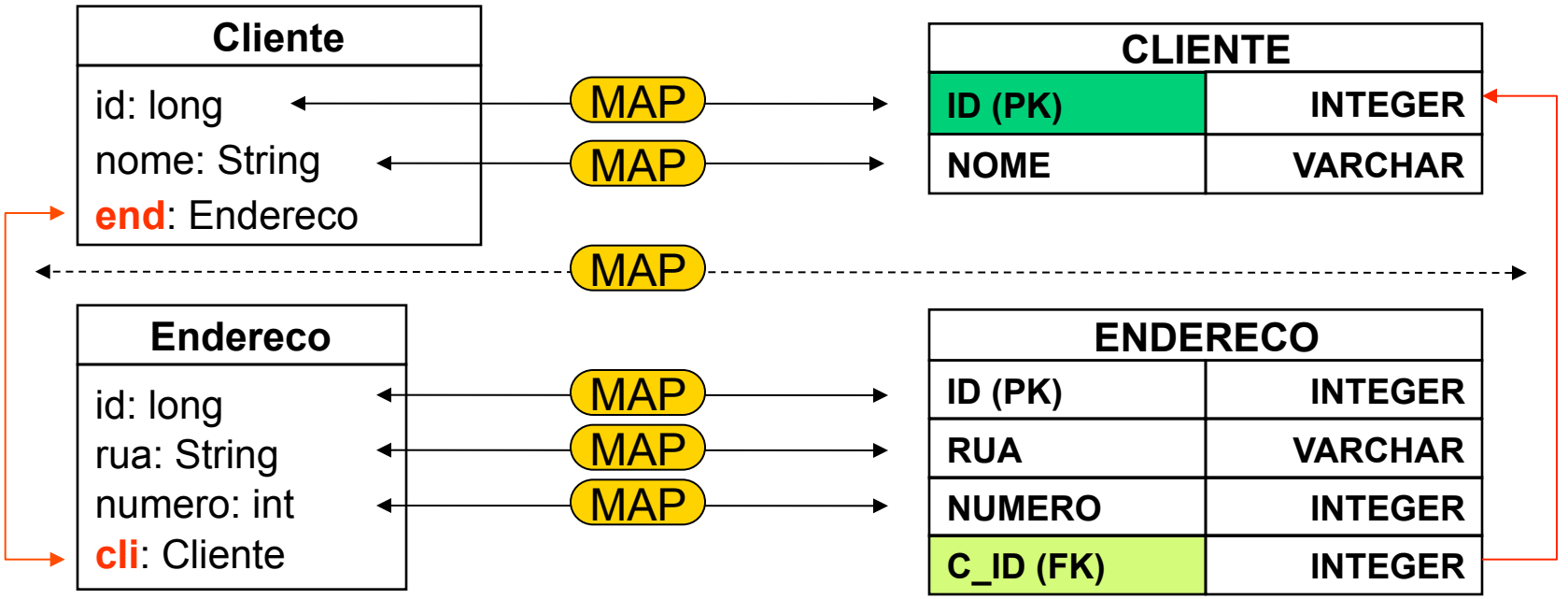
Associações 1:1 - Mapeam. Operações

Caso 2: Endereco(1) –> Cliente(1)

- Inserção: a partir de um endereço

INSERT INTO CLIENTE(ID,NOME) VALUES (end.cli.id, end.cli.nome)

INSERT INTO ENDERECO(ID,RUA,NUMERO,C_ID) VALUES (end.id, end.rua,
end.numero, end.cli.id)



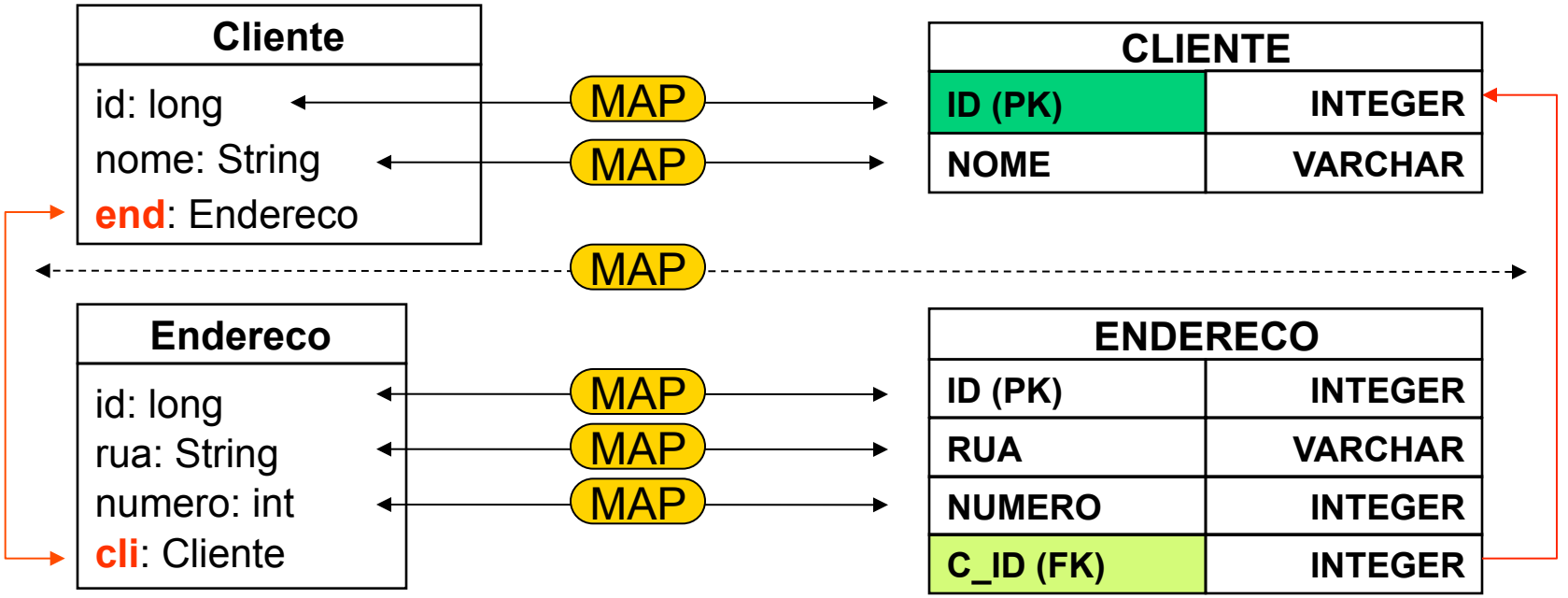
Associações 1:1 - Mapeam. Operações

Caso 2: Endereco(1) –> Cliente(1)

- Atualização: a partir de um endereço

UPDATE ENDERECO SET RUA = end.rua, NUMERO = end.numero WHERE ID = end.id

UPDATE CLIENTE SET NOME = end.cli.nome WHERE ID = end.cli.id



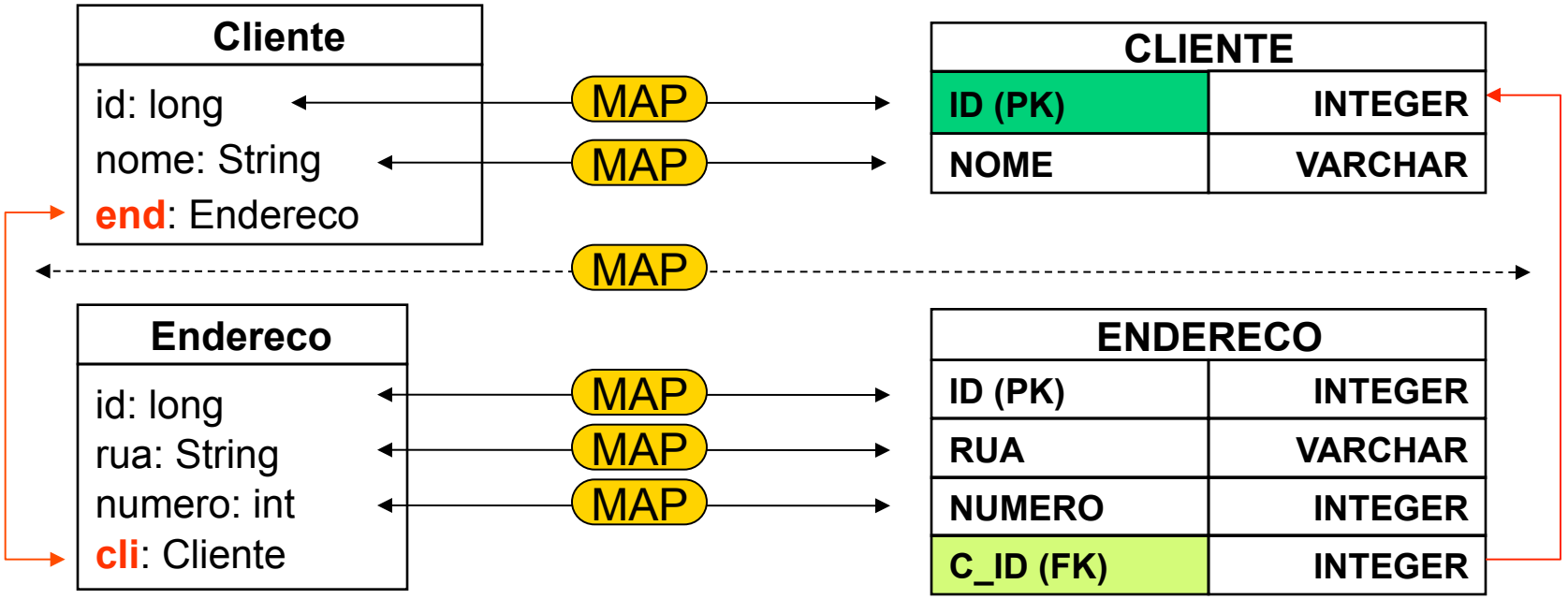
Associações 1:1 - Mapeam. Operações

Caso 2: Endereco(1) –> Cliente(1)

- Remoção: a partir de um endereço

DELETE FROM ENDERECO WHERE ID = end.id

DELETE FROM CLIENTE WHERE ID = end.cli.id *



*cardinalidade mínima de Endereço em Cliente é 1 (Cliente tem que ter um Endereço).Caso contrário, não faz nada.

Associações 1:N - Mapeam. Estrutural

- As classes associadas devem estar mapeadas em tabelas separadas

- A tabela correspondente à classe do lado N da associação deve conter uma chave estrangeira para a tabela correspondente à classe do lado 1

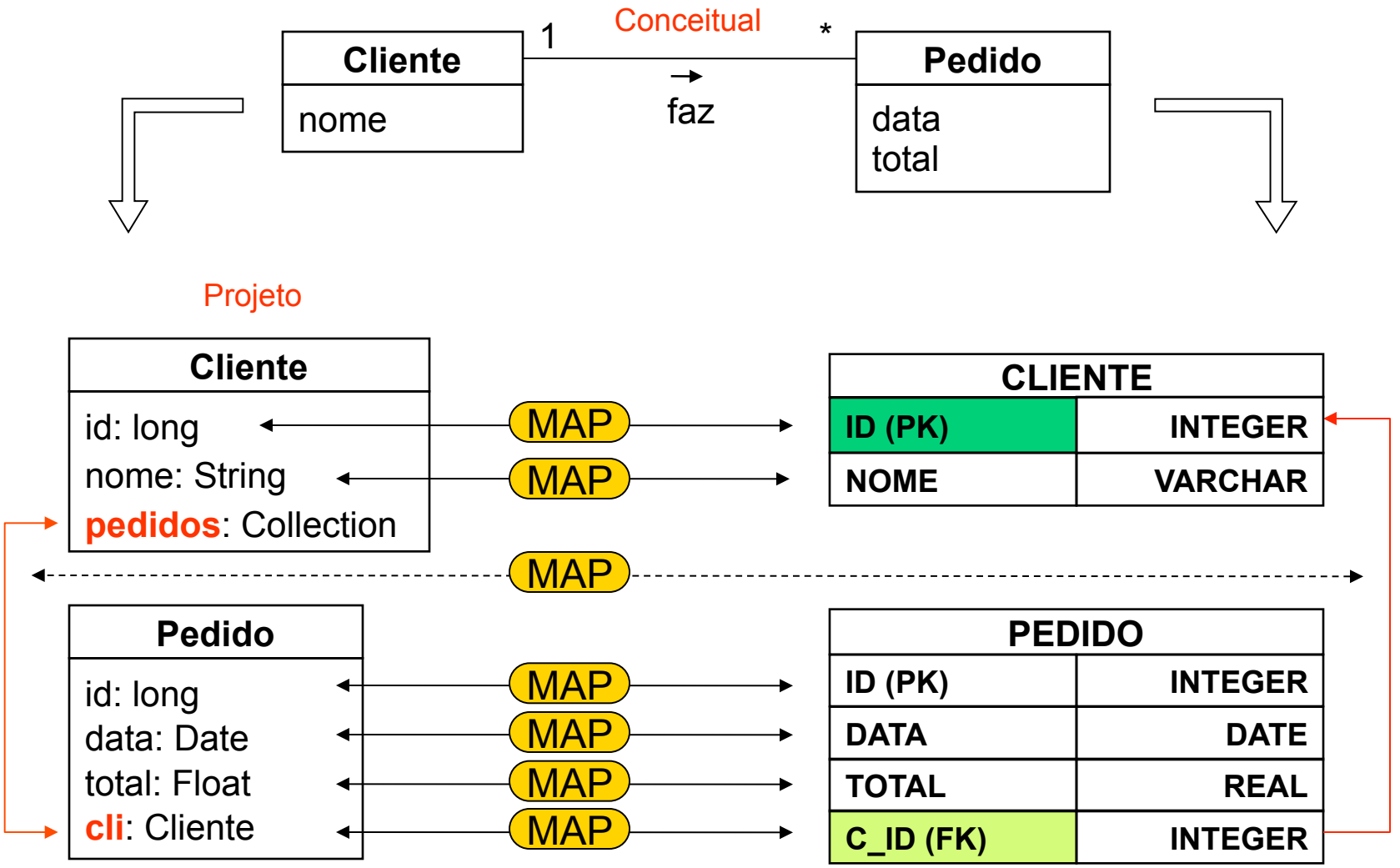
Projeto

Cliente
id: long nome: String **pedidos**: Collection

Pedido
id: long data: Date total: Float **cli**: Cliente

Obs: É conveniente que as classes associadas refiram-se mutuamente através de referências inversas

Associações 1:N - Mapeam. Estrutural

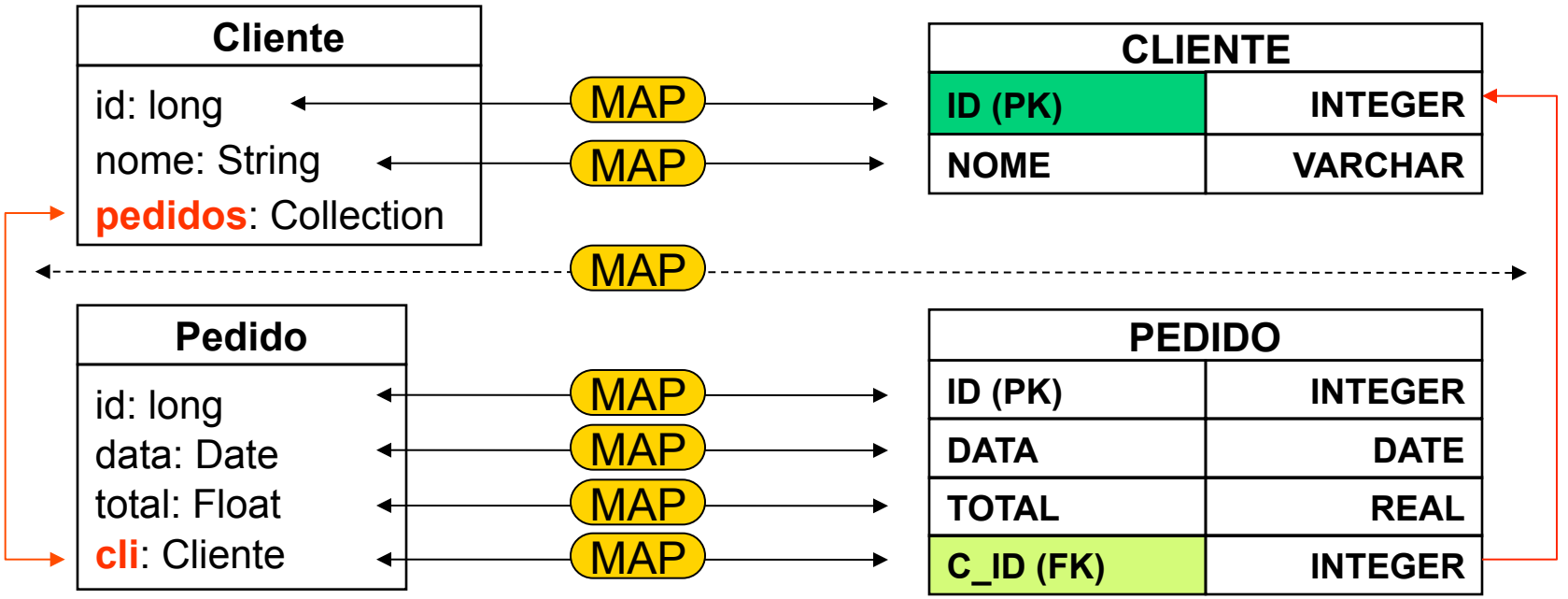


Associações 1:N - Mapeam. Operações

Caso 1: Cliente(1) –> Pedido(N)

- Recuperação: a partir do oid do cliente

```
cli := new Cliente()
cli.id, cli.nome := SELECT ID, NOME FROM CLIENTE WHERE  ID = oid
pedidos(id,data,total)* := SELECT ID, DATA, TOTAL FROM PEDIDO
                           WHERE C_ID = cli.id (instancia coleção)
cli.pedidos := pedidos
      p/ cada :ped em cli.pedidos
  ped.cli := cli
```



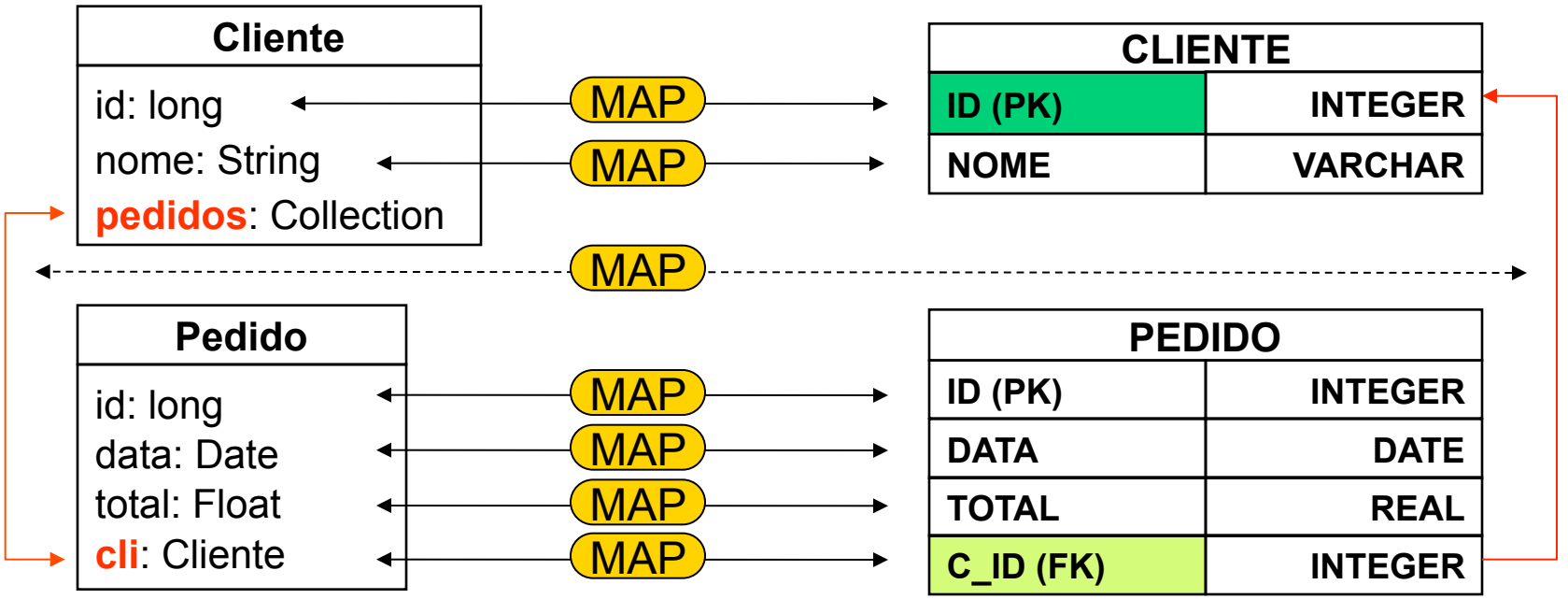
Associações 1:N - Mapeam. Operações

Caso 1: Cliente(1) –> Pedido(N)

- Inserção: a partir de um cliente

```
INSERT INTO CLIENTE(ID,NOME) VALUES (cli.id, cli.nome)

        p/ cada pedido :ped em cli.pedidos
INSERT INTO PEDIDO(ID,DATA,TOTAL,C_ID) VALUES (ped.id, ped.data, ped.total,
                                                cli.id)
```



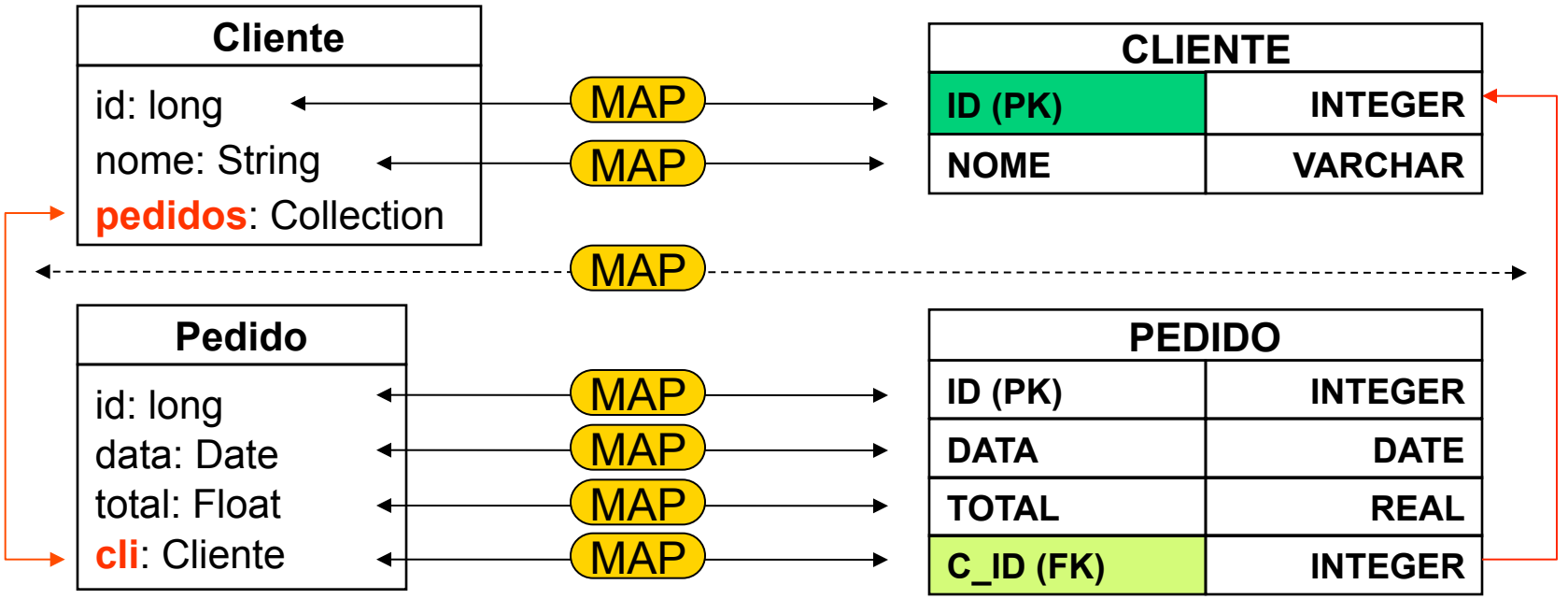
Associações 1:N - Mapeam. Operações

Caso 1: Cliente(1) –> Pedido(N)

- Atualização: a partir de um cliente

```
UPDATE CLIENTE SET NOME = cli.nome  WHERE ID = cli.id

      p/ cada pedido :ped em cli.pedidos
UPDATE PEDIDO SET DATA = ped.data, TOTAL = ped.total  WHERE ID = ped.id
```



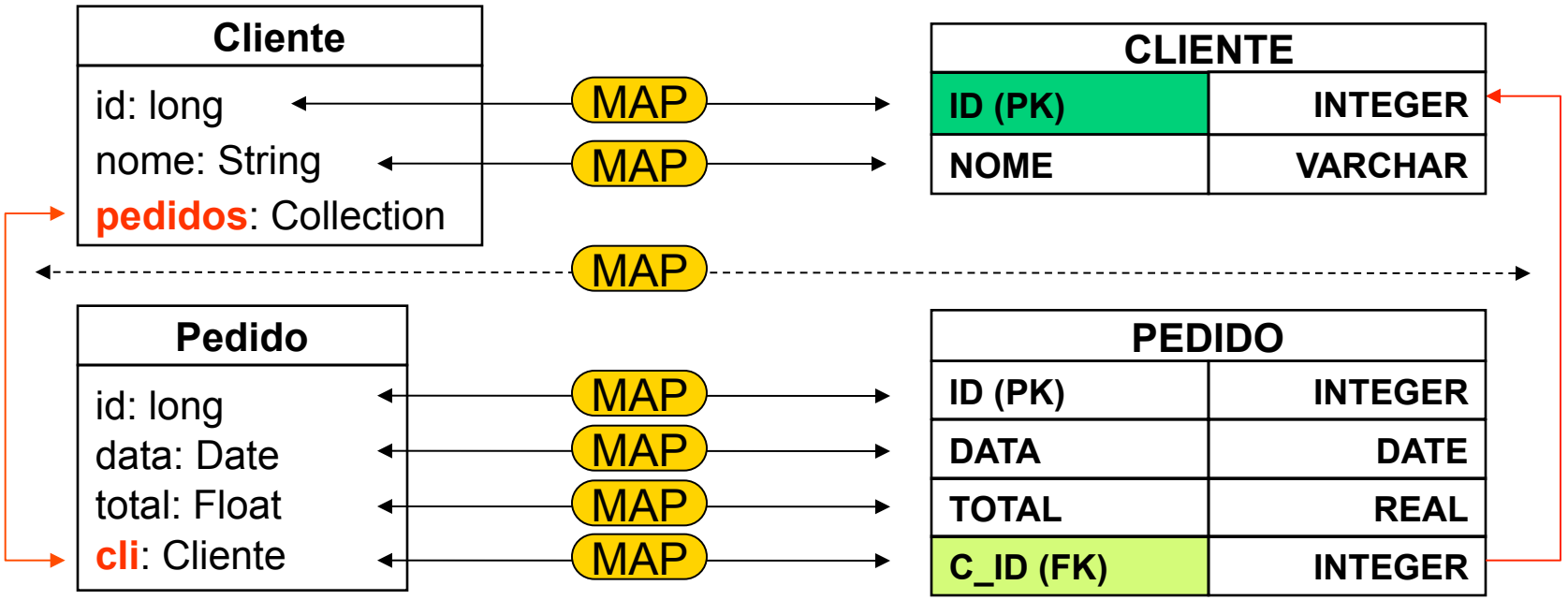
Associações 1:N - Mapeam. Operações

Caso 1: Cliente(1) –> Pedido(N)

- Remoção: a partir de um cliente

DELETE FROM PEDIDO WHERE C_ID = cli.id * OU
UPDATE PEDIDO SET C_ID = NULL WHERE C_ID = cli.id **

DELETE FROM CLIENTE WHERE ID = cli.id



*cardinalidade mínima de Cliente em Pedido é 1 (Pedido tem que ter um Cliente)

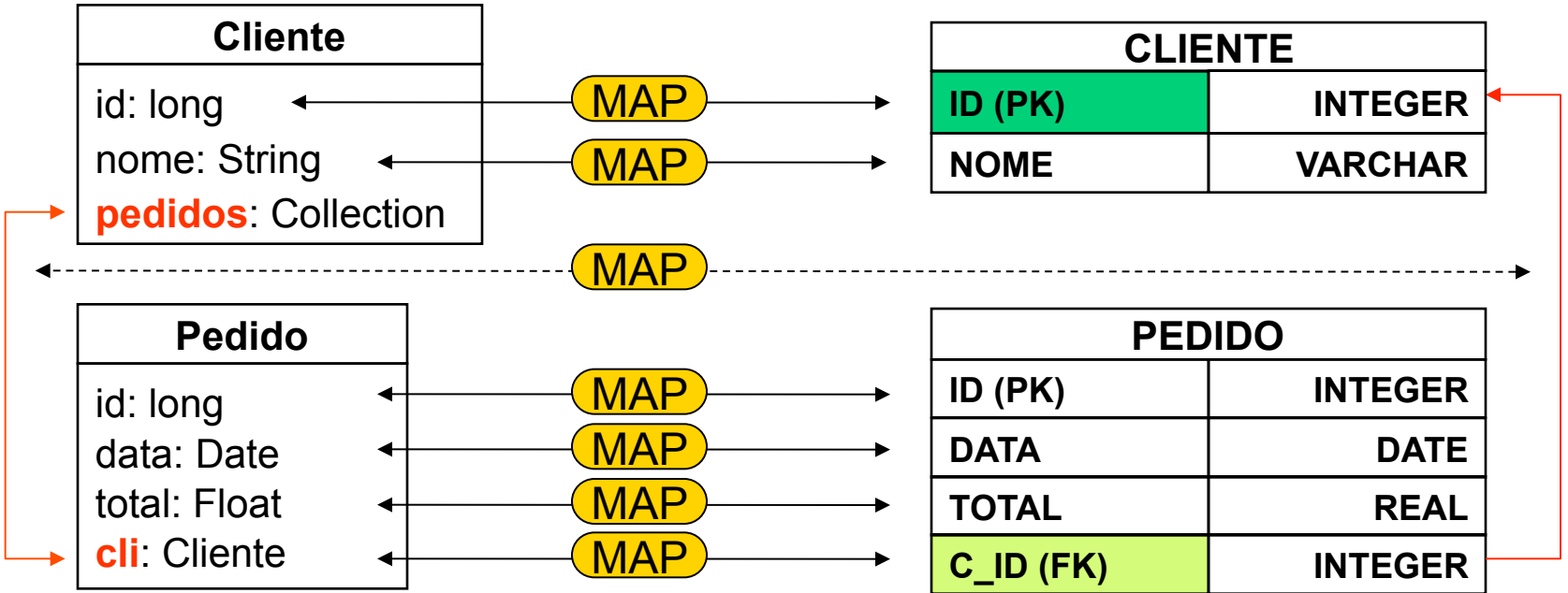
** cardinalidade mínima de Cliente em Pedido é 0 (Pedido pode não ter um Cliente)

Associações 1:N - Mapeam. Operações

Caso 2: Pedido(N) –> Cliente(1)

- Recuperação: a partir do oid de um pedido

```
ped := new Pedido()
ped.id,ped.data,ped.total,var_cid := SELECT ID, DATA, TOTAL, C_ID
                                     FROM PEDIDO WHERE ID = oid
cli := new Cliente()
cli.id,cli.nome := SELECT ID, NOME FROM CLIENTE WHERE ID = var_cid
ped.cli := cli
cli.pedidos.add(ped)
```



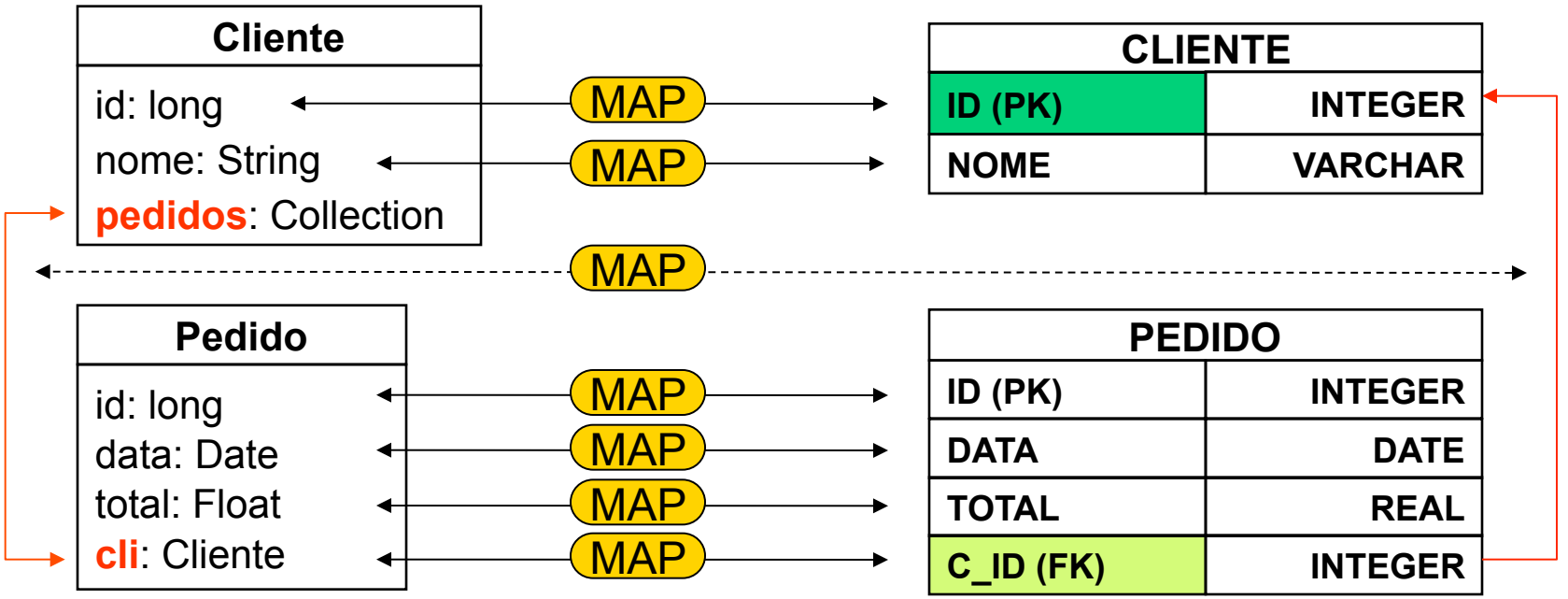
Associações 1:N - Mapeam. Operações

Caso 2: Pedido(N) –> Cliente(1)

- Inserção: a partir de um pedido

INSERT INTO CLIENTE(ID,NOME) VALUES (ped.cli.id, ped.cli.nome) // se cliente ainda não existia

INSERT INTO PEDIDO(ID,DATA,TOTAL,C_ID) VALUES (ped.id, ped.data, ped.total,
ped.cli.id)



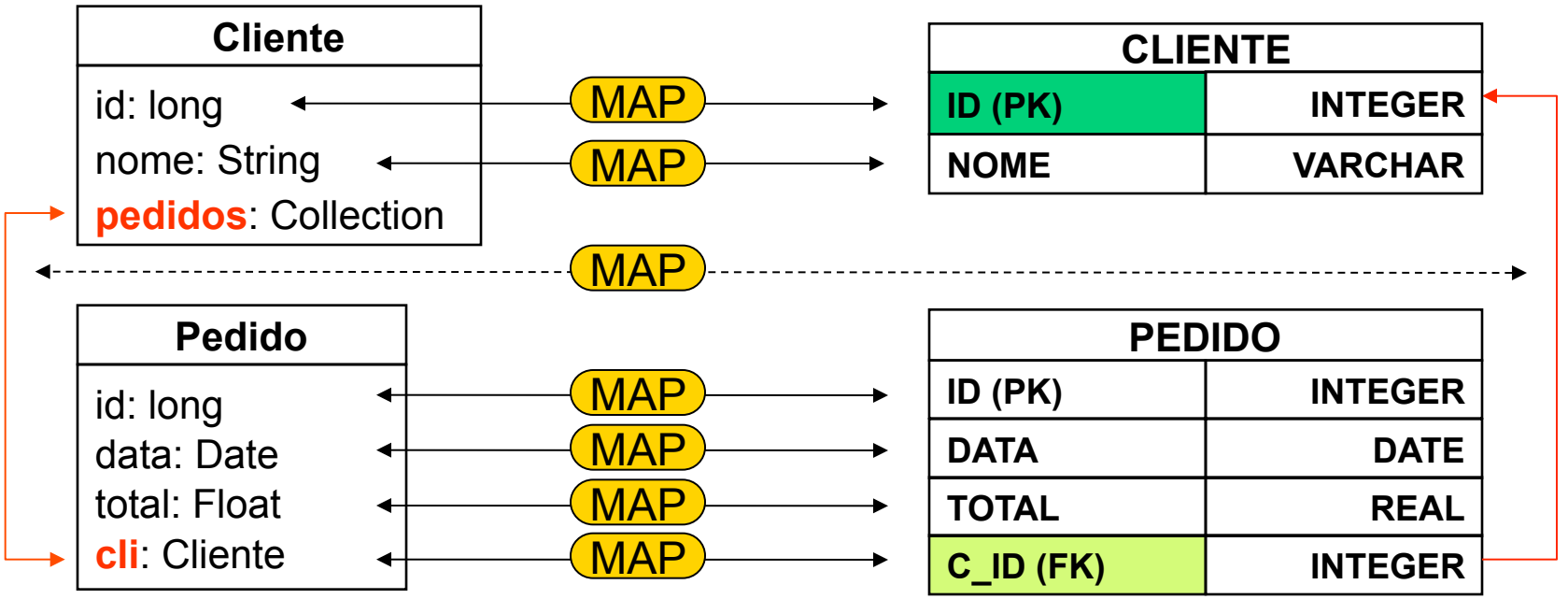
Associações 1:N - Mapeam. Operações

Caso 2: Pedido(N) –> Cliente(1)

- Atualização: a partir de um pedido

UPDATE PEDIDO SET DATA = ped.data, TOTAL = ped.total WHERE ID = ped.id

UPDATE CLIENTE SET NOME = ped.cli.nome WHERE ID = ped.cli.id



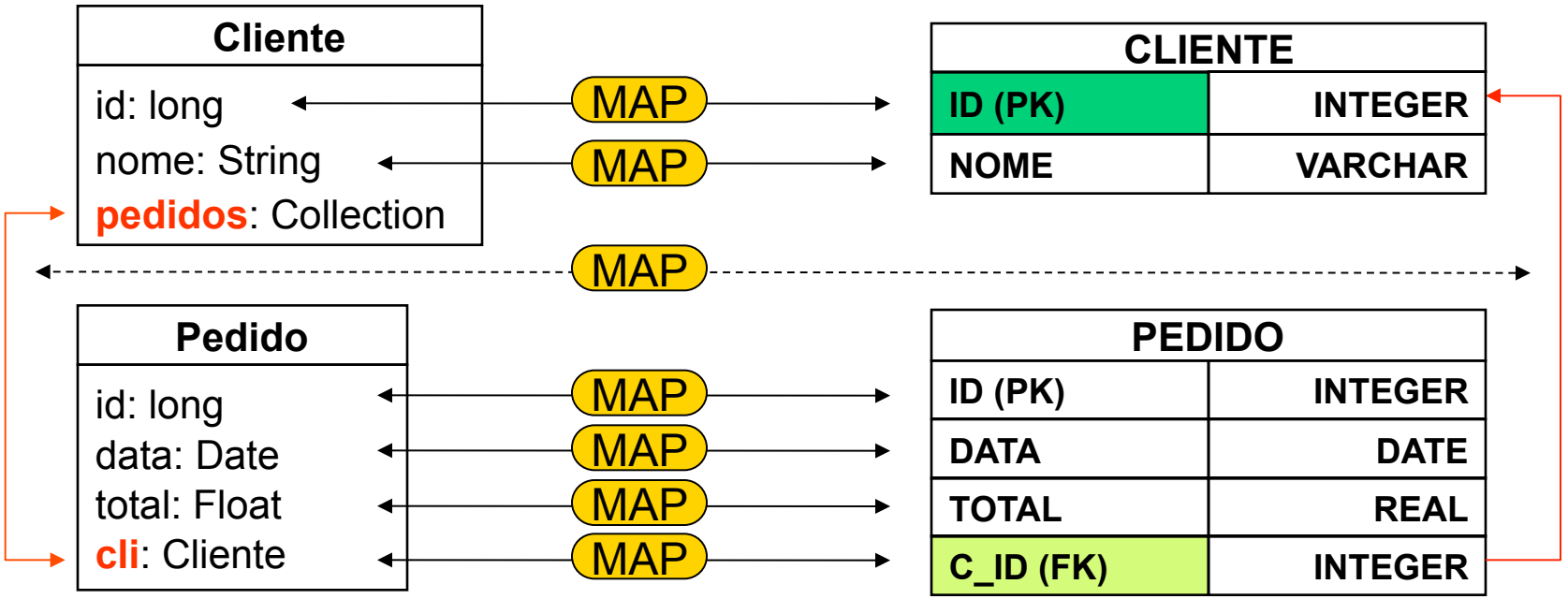
Associações 1:N - Mapeam. Operações

Caso 2: Pedido(N) –> Cliente(1)

- Remoção: a partir de um pedido

DELETE FROM PEDIDO WHERE ID = ped.id

DELETE FROM CLIENTE WHERE ID = ped.cli.id *



* Se cardinalidade mínima de Pedido em Cliente é 1, remoção só deve ocorrer se ped.cli.pedidos.size() = 1

Associações N:N - Mapeam. Estrutural

- As classes associadas devem estar mapeadas em tabelas separadas

- Uma terceira tabela deve ser definida para representar a associação, com chaves estrangeiras que se referem às tabelas das classes associadas

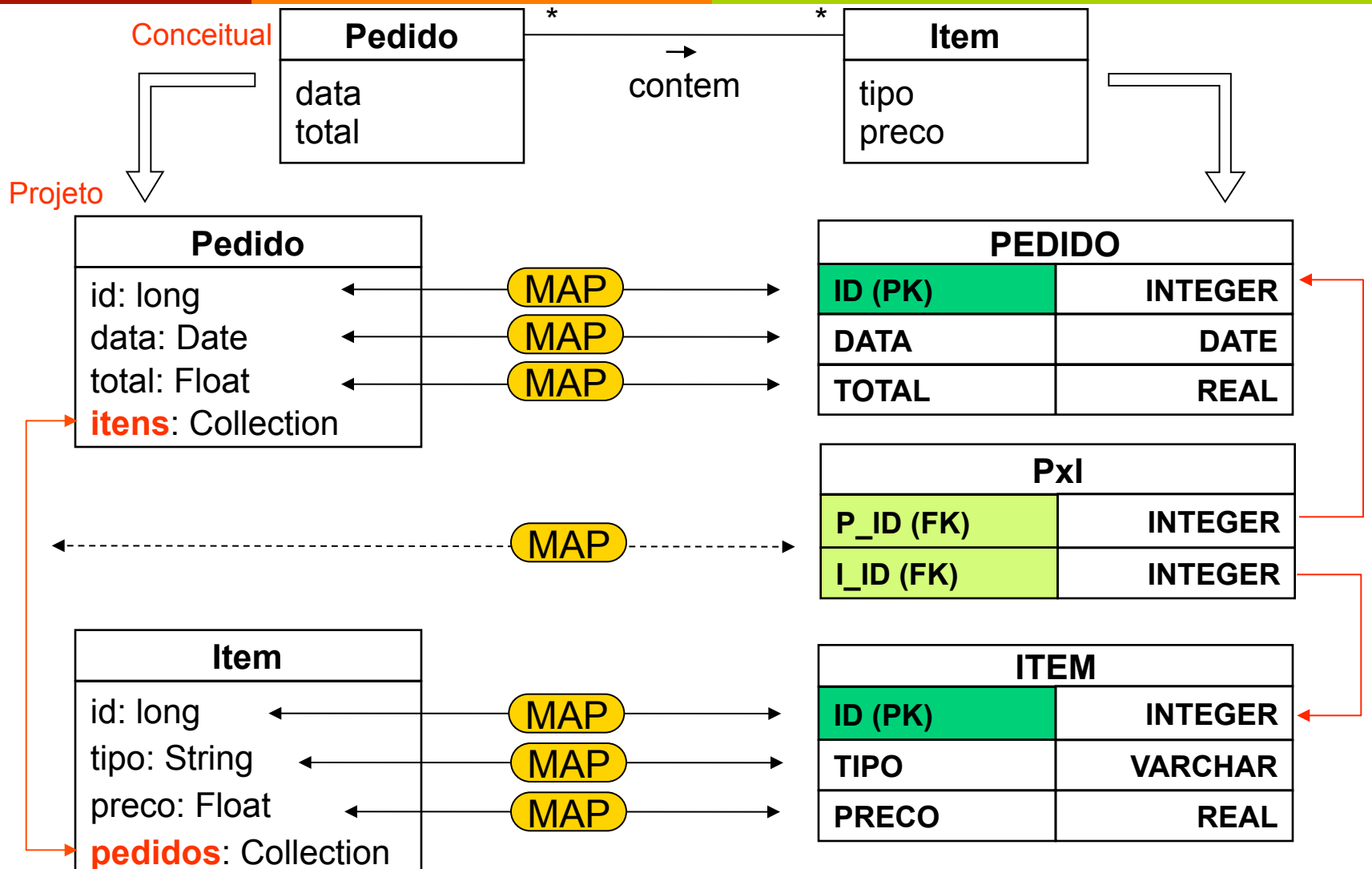
Projeto

Pedido
id: long data: Date total: Float **itens**: Collection

Item
id: long tipo: String preco: Float **pedidos**: Collection

Obs: É conveniente que as classes associadas refiram-se mutuamente através de referências inversas.

Associações N:N - Mapeam. Estrutural

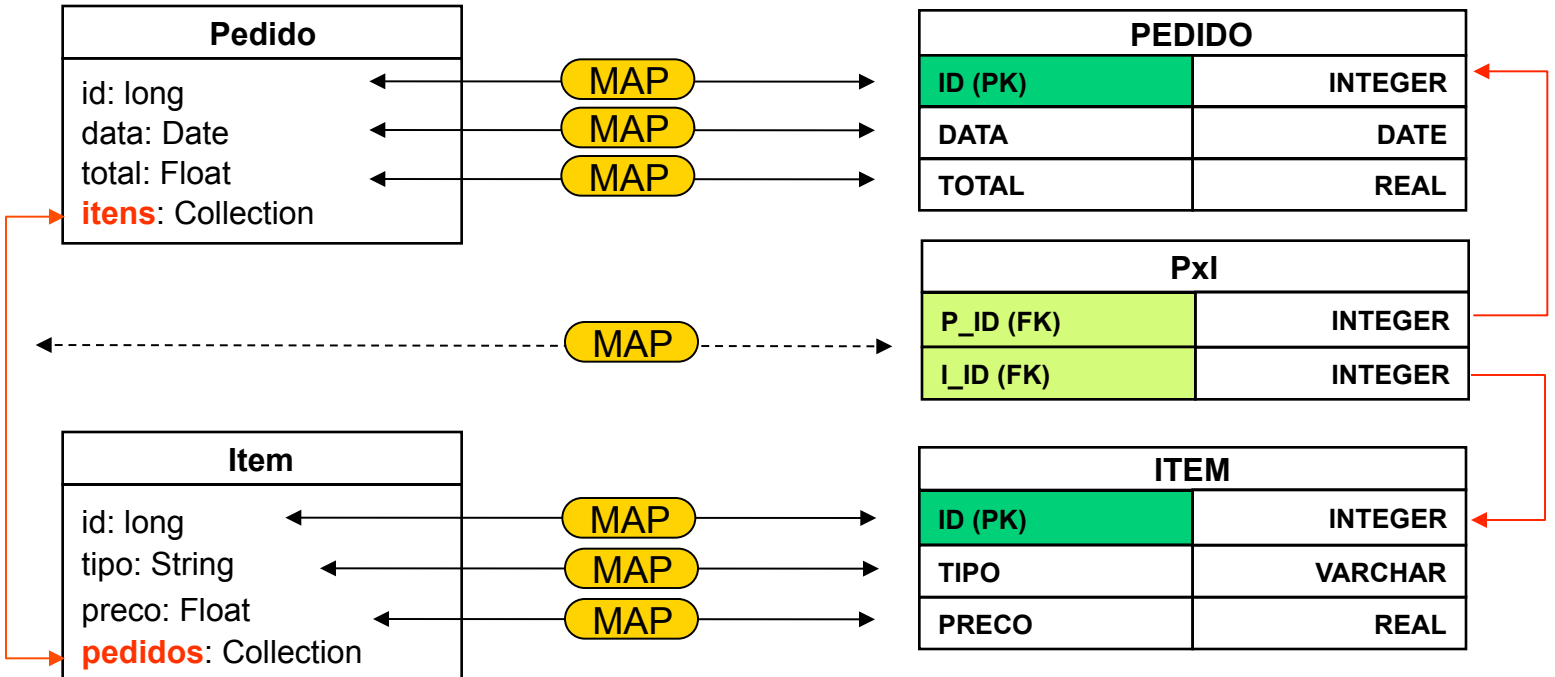


Associações N:N - Mapeam. Operações

Pedido(N) –> Item(N)

- Recuperação: a partir do oid de um pedido

```
ped := new Pedido()
ped.id,ped.data,ped.total := SELECT ID, DATA, TOTAL FROM PEDIDO WHERE  ID = oid
itens(id,tipo,preco)* := SELECT ID, TIPO, PRECO FROM PxI JOIN ITEM ON PxI.I_ID = ITEM.ID
                         WHERE PxI.P_ID = ped.id (instancia coleção)
ped.itens := itens
        p/ cada :item em ped.itens
item.pedidos.add(ped)
```



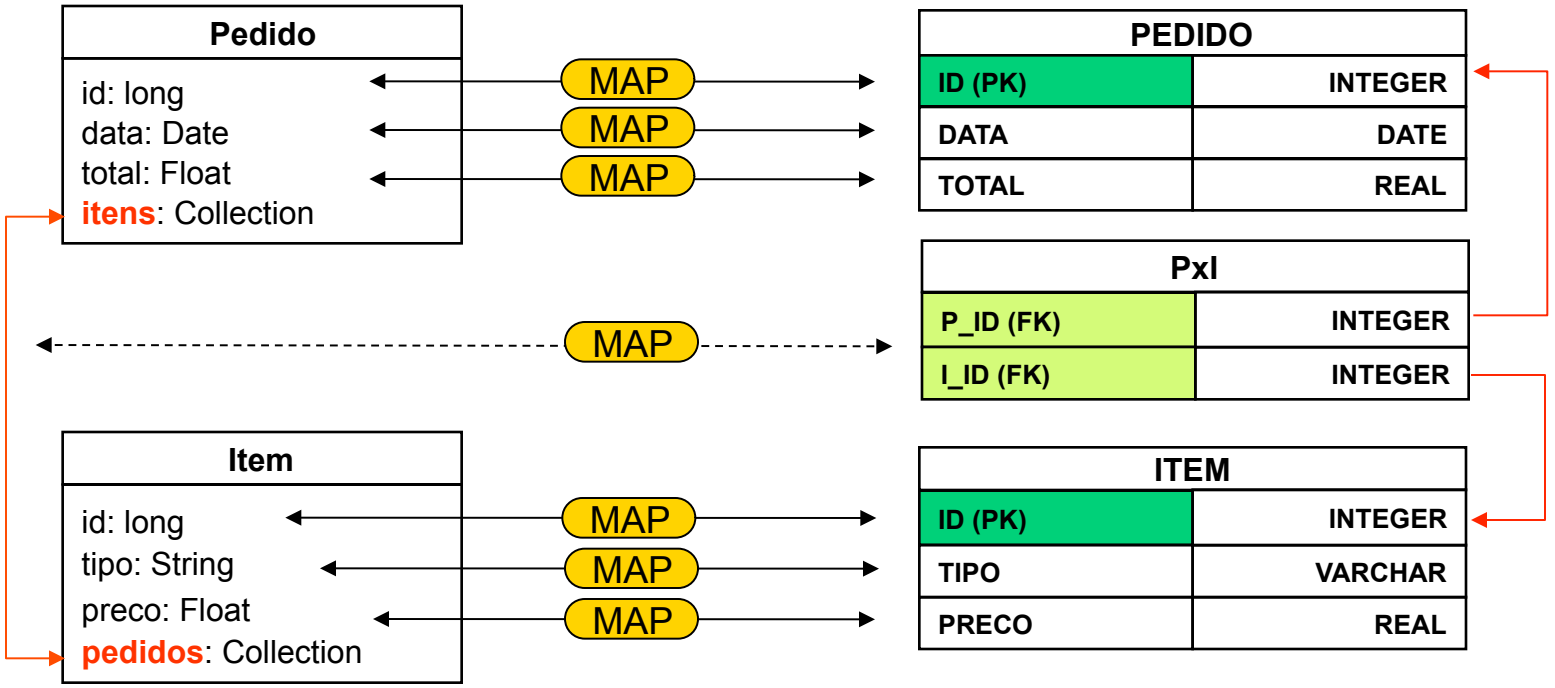
Associações N:N - Mapeam. Operações

Pedido(N) –> Item(N)

- Inserção: a partir de um pedido

INSERT INTO PEDIDO(ID,DATA,TOTAL) VALUES (ped.id, ped.data, ped.total)

p/ cada novo item :item em ped.itens
INSERT INTO ITEM(ID,TIPO,PRECO) VALUES (item.id, item.tipo, item.preco)
INSERT INTO PxI(P_ID,I_ID) VALUES (ped.id, item.id)



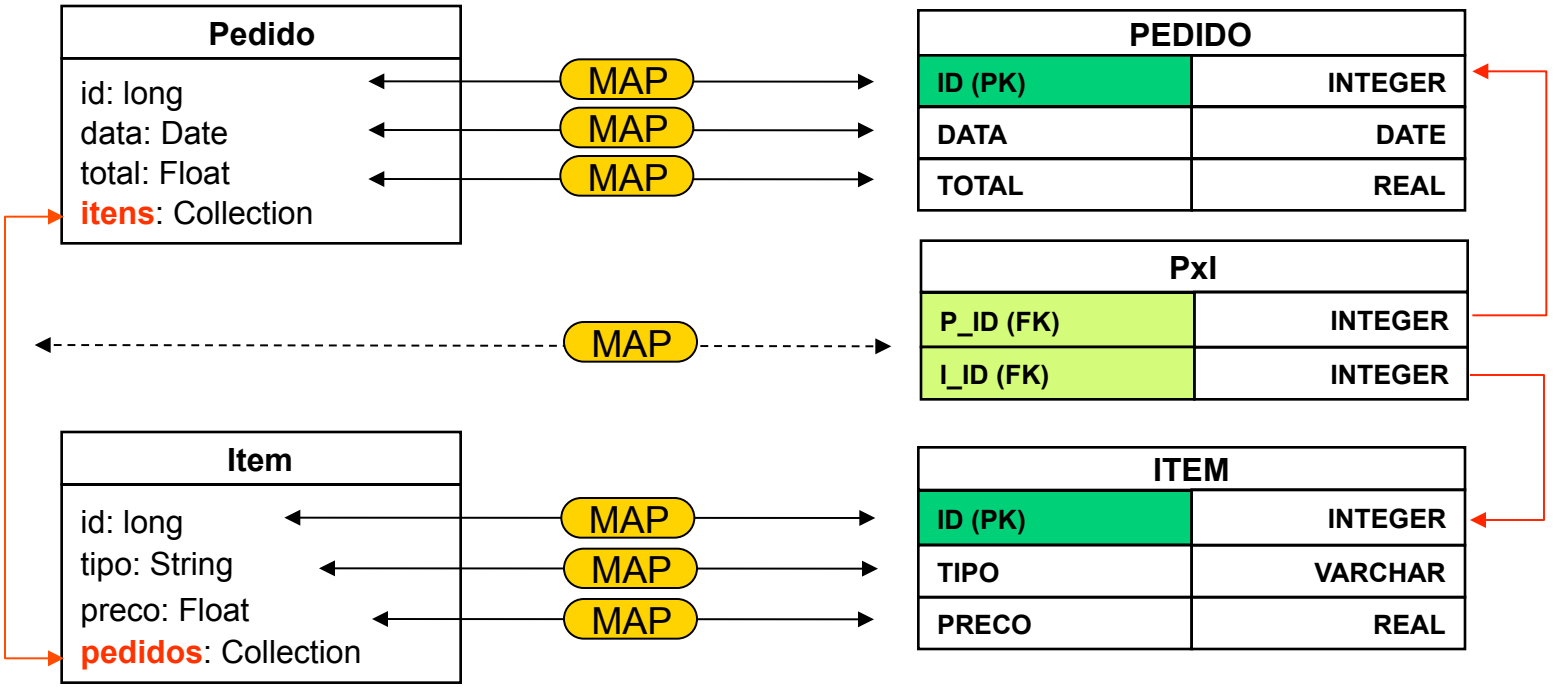
Associações N:N - Mapeam. Operações

Pedido(N) –> Item(N)

- Atualização: a partir de um pedido

UPDATE PEDIDO SET DATA = ped.data, TOTAL = ped.total WHERE ID = ped.id

p/ cada item modificado :item em ped.itens:
UPDATE ITEM SET TIPO = item.tipo, PRECO = item.preco WHERE ID = item.id

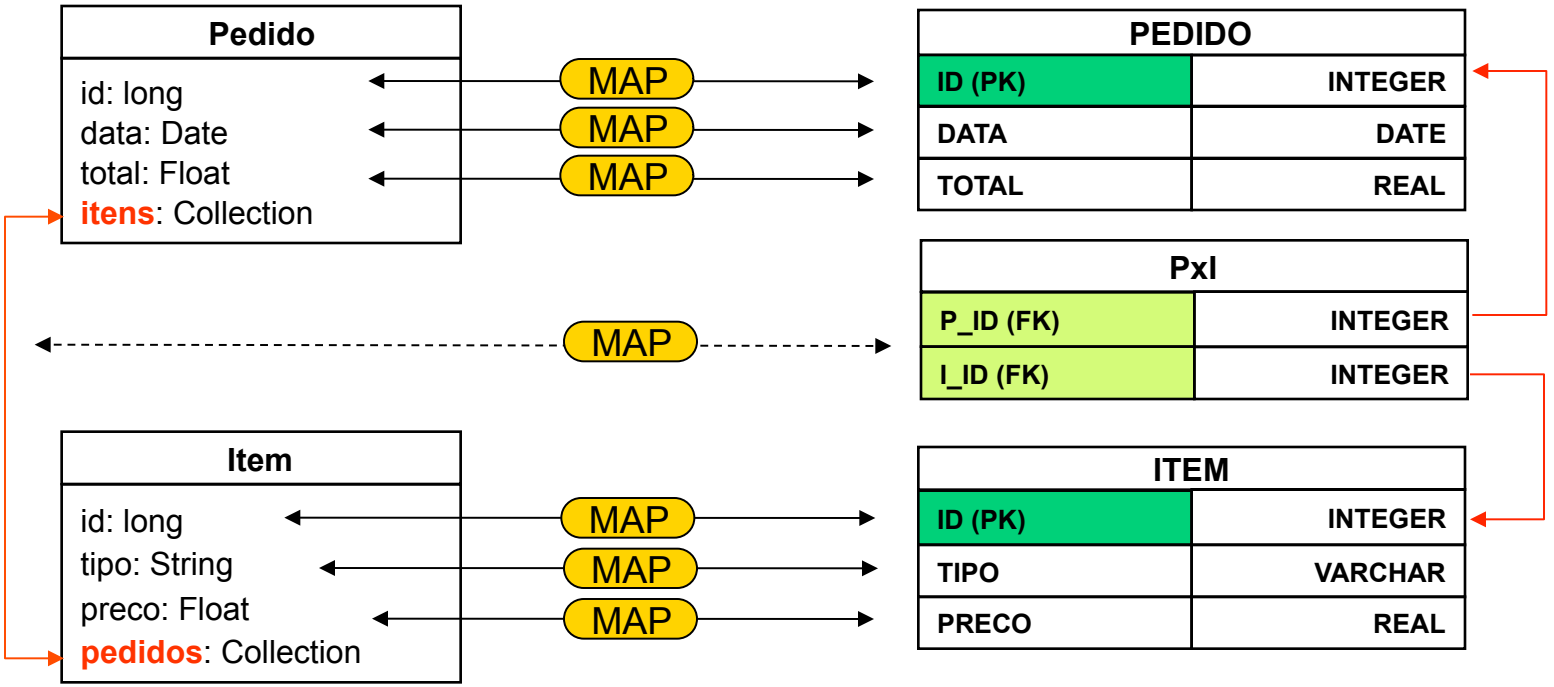


Associações N:N - Mapeam. Operações

Pedido(N) –> Item(N)

- Remoção: a partir de um pedido

DELETE FROM PxI WHERE P_ID = ped.id
DELETE FROM PEDIDO WHERE ID = ped.id *



* Se o Item tem que ter no mínimo 1 Pedido, a remoção só deve ocorrer se p/ cada :item em ped:itens, :item.pedidos.size() > 1 ou remove o Item

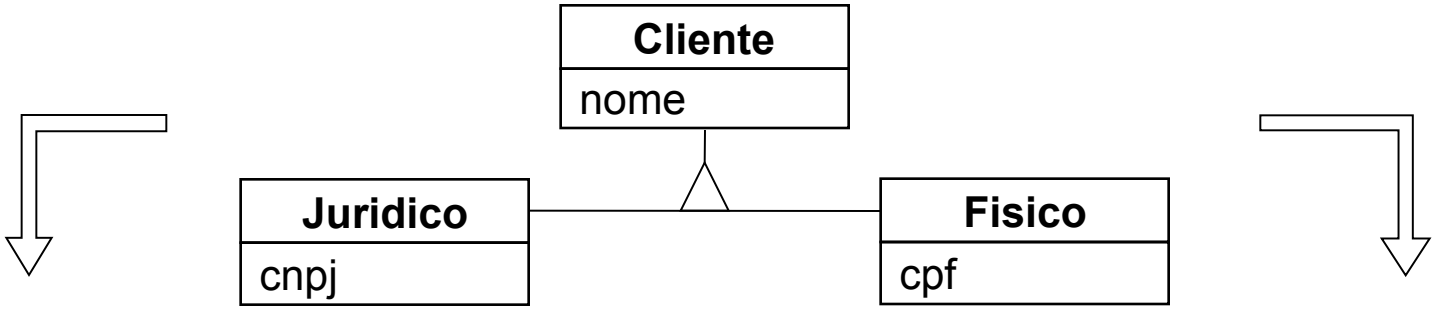
Herança - Mapeamento Estrutural

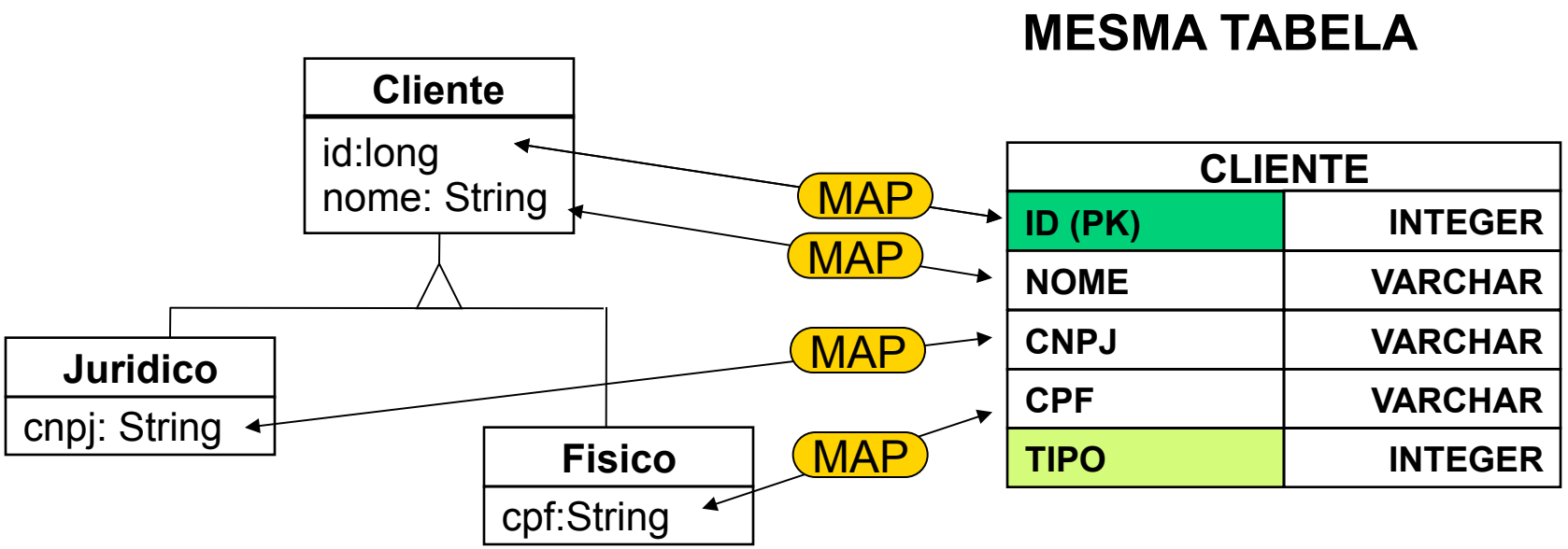
➡ As classes da hierarquia podem estar mapeadas para uma mesma tabela, tabelas separadas, ou cada classe em uma tabela diferente*.

➔ Mesma tabela: se as classes são mapeadas para uma mesma tabela, esta tabela deve conter colunas para todos os atributos da superclasse e respectivas subclasses, além de um atributo discriminador que indica o tipo do objeto.

* Usado em sistemas legados.

Herança - Mapeamento Estrutural



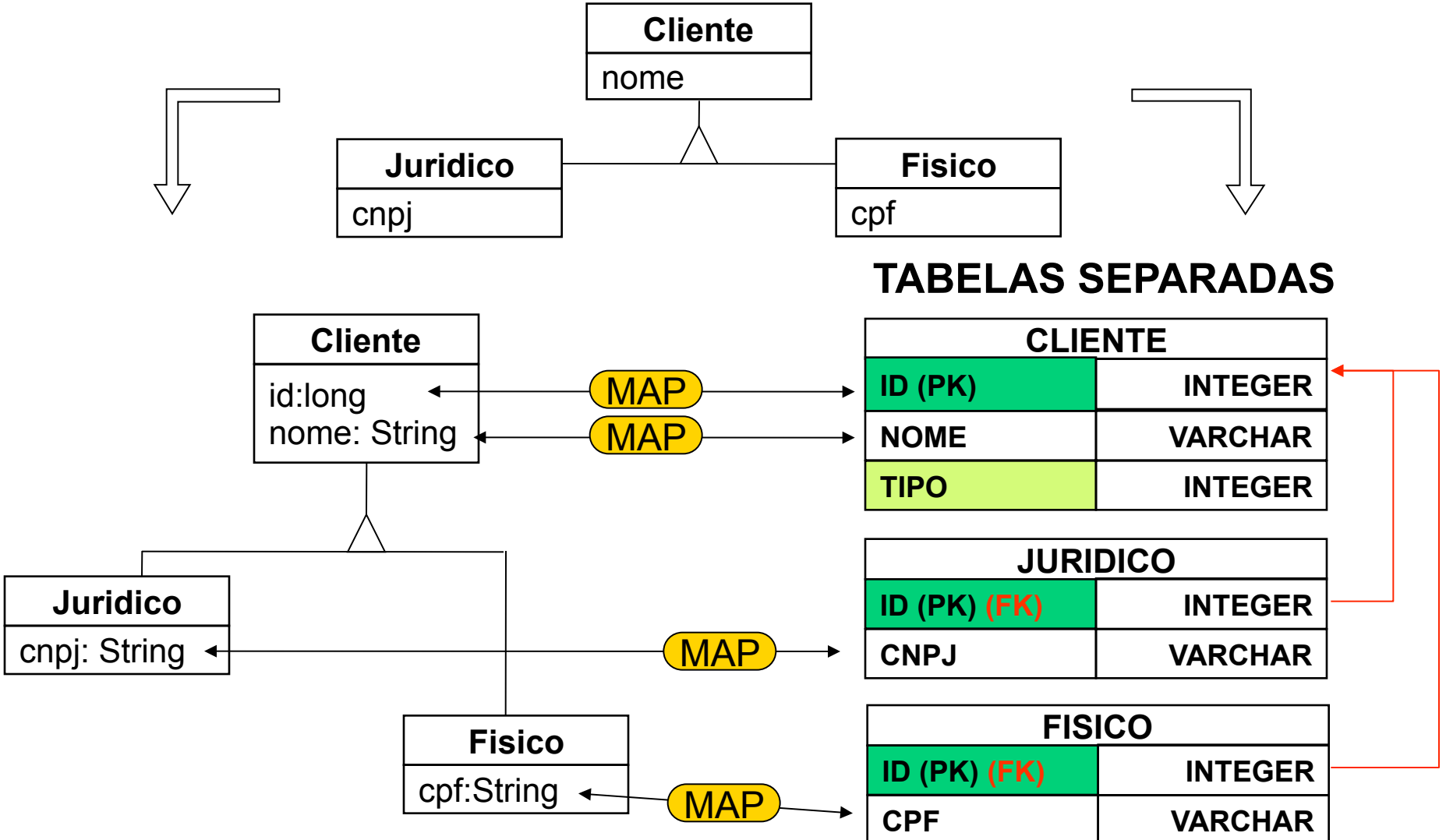


Herança - Mapeamento Estrutural

➡ As classes da hierarquia podem estar mapeadas para uma mesma tabela ou tabelas separadas, ou cada classe em uma tabela diferente*.

➔ Tabelas separadas: se as classes são mapeadas para tabelas separadas, cada tabela deve conter apenas colunas para os atributos da classe correspondente. Além disso, as tabelas das subclasses devem se referir à tabela da superclasse através de uma chave estrangeira. Neste caso, o atributo discriminador fica na tabela da superclasse raiz.

Herança - Mapeamento Estrutural



Herança - Mapeamento Operações

- Recuperação a partir da seleção na superclasse a partir de um oid (usando vários select's):

cid, cnome, tipo := SELECT ID, NOME, TIPO FROM CLIENTE WHERE ID = oid

Se tipo = "Fisico"

cli := new Fisico()
cli.id := cid
cli.nome := cnome
cli.cpf := SELECT CPF FROM FISICO WHERE ID = cli.id

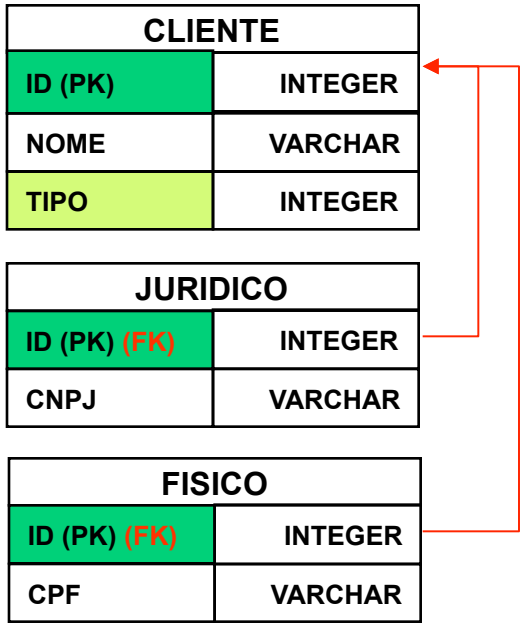
Se tipo = "Juridico"

cli := new Juridico()
cli.id := cid
cli.nome := cnome
cli.cnpj := SELECT CNPJ FROM JURIDICO WHERE ID = cli.id

Se tipo = "Cliente" **

cli := new Cliente()
cli.id := cid
cli.nome := cnome

** a menos que cliente seja uma classe abstrata



Herança - Mapeamento Operações

- Recuperação a partir da seleção na superclasse (usando outer join):

```
cid, cnome, ccnpj, ccpf, tipo := SELECT CLIENTE.ID, NOME, CNPJ, CPF, TIPO
                                 FROM CLIENTE LEFT JOIN JURIDICO ON CLIENTE.ID = JURIDICO.ID
                                              LEFT JOIN FISICO ON CLIENTE.ID = FISICO.ID
                                 WHERE CLIENTE.ID = oid
```

Se tipo = "Fisico"

```
cli := new Fisico()
cli.id := cid
cli.nome := cnome
cli.cpf := ccpf
```

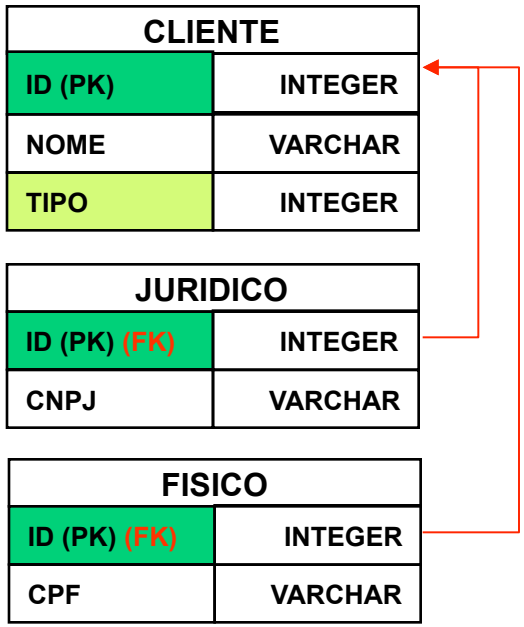
Se tipo = "Juridico"

```
cli := new Juridico()
cli.id := cid
cli.nome := cnome
cli.cnpj := ccnpj
```

Se tipo = "Cliente" **

```
cli := new Cliente()
cli.id := cid
cli.nome := cnome
```

** a menos que cliente seja uma classe abstrata



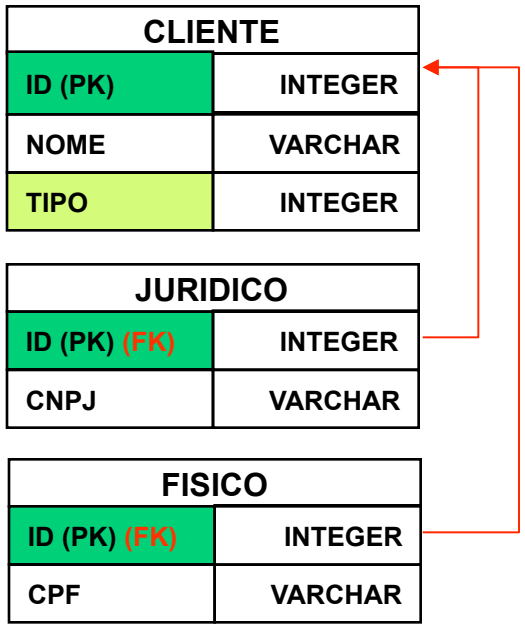
Herança - Mapeamento Operações

- Recuperação a partir da seleção na subclasse a partir do oid de um físico (usando vários select's)

```
fis := new Fisico()
fis.id, fis.cpf := SELECT ID, CPF FROM FISICO WHERE ID = oid
fis.nome = SELECT nome FROM CLIENTE WHERE ID = fis.id
```

- Recuperação a partir da seleção na subclasse a partir do oid de um físico (usando junção)

```
fis := new Fisico()
fis.id, fis.nome, fis.cpf := SELECT CLIENTE.ID, NOME, CPF
                             FROM CLIENTE JOIN FISICO
                                  ON CLIENTE.ID = FISICO.ID
                             WHERE CLIENTE.ID = oid
```



Herança - Mapeamento Operações

- Inserção de uma instância na superclasse: a partir de um cliente

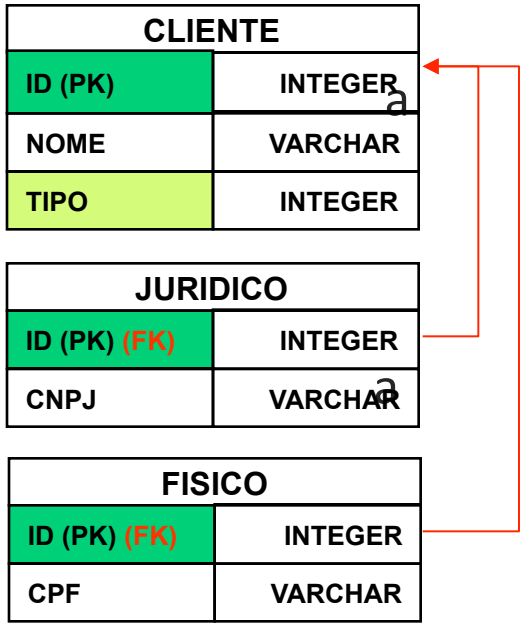
INSERT INTO CLIENTE(ID,NOME,TIPO) VALUES (cli.id,cli.nome,"Cliente")

- Atualização de uma instância na superclasse: partir de um cliente

UPDATE CLIENTE SET NOME = cli.nome WHERE ID = cli.id

- Remoção de uma instância na superclasse: partir de um cliente

DELETE FROM CLIENTE WHERE ID = cli.id



Herança - Mapeamento Operações

- Inserção na subclasse: a partir de um físico

INSERT INTO CLIENTE(ID,NOME,TIPO) VALUES (fis.id, fis.nome,"Fisico")
INSERT INTO FISICO(ID,CPF) VALUES (fis.id, fis.cpf)

- Atualização na subclasse: a partir de um físico

UPDATE CLIENTE NOME = fis.nome WHERE ID = fis.id
UPDATE FISICO CPF = fis.cpf WHERE ID = fis.id

- Remoção na subclasse: a partir de um físico

DELETE FROM FISICO WHERE ID = fis.id
DELETE FROM CLIENTE WHERE ID = fis.id