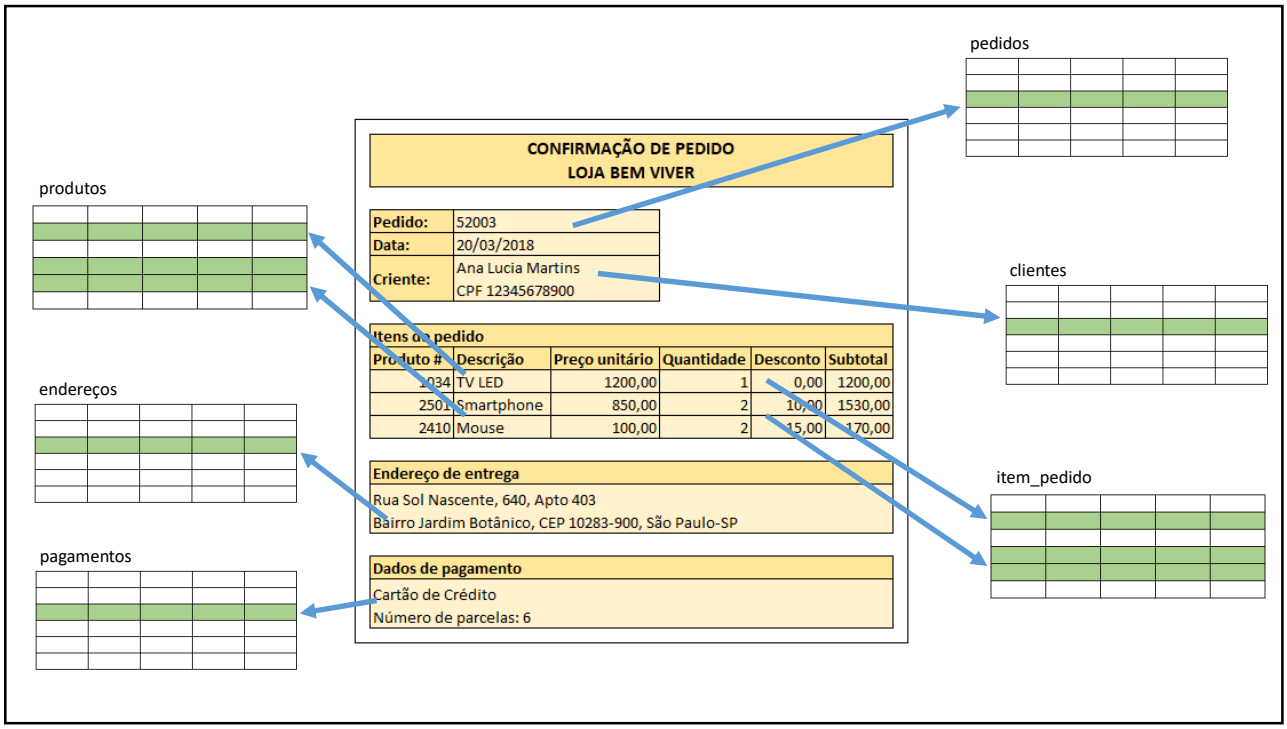
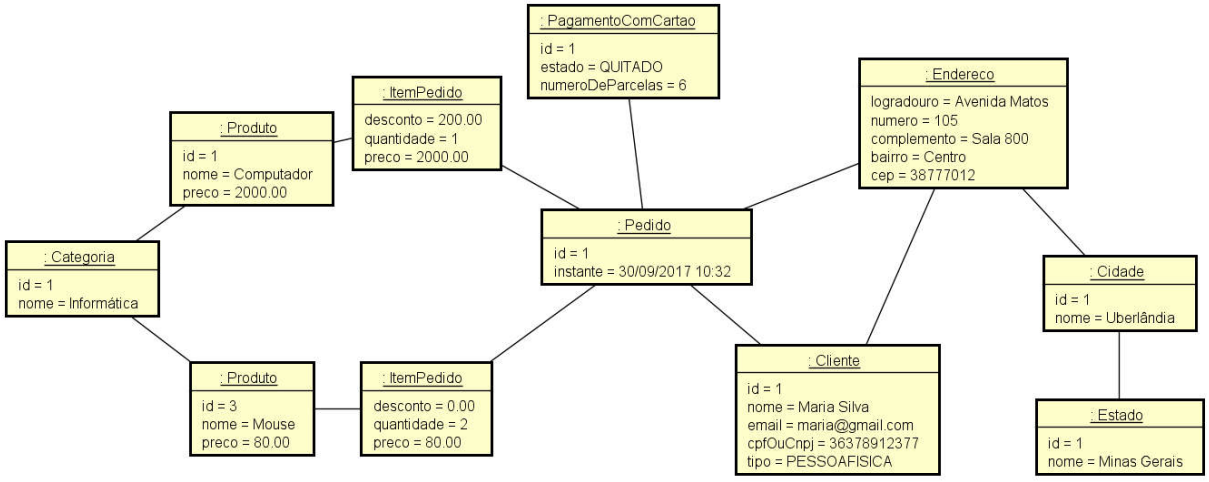
# Nivelamento sobre NoSQL e MongoDB

# Problema 1: incompatibilidade de impedância

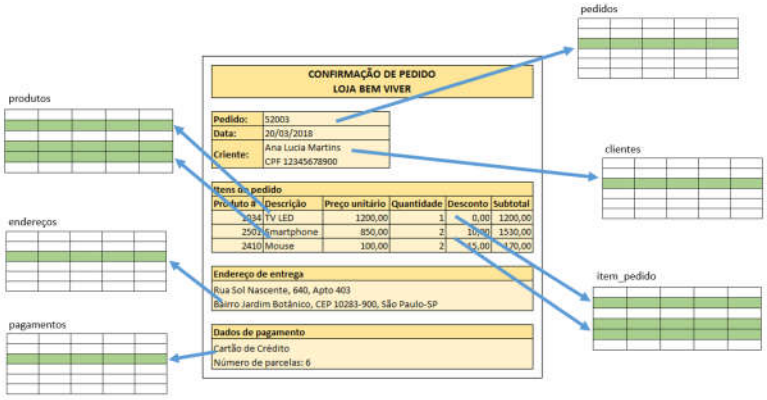



Solução (para os programadores): ORM

# Com ou sem ORM

- Transações e junções degradam performance!




```
(...) from
       pedido pedido0_
   left outer join
       cliente cliente1_
           on pedido0_.cliente_id=cliente1_.id
   left outer join
       perfis perfis2_
           on cliente1_.id=perfis2_.cliente_id
   left outer join
       endereco endereco3_
           on pedido0_.endereco_de_entrega_id=endereco3_.id
   left outer join
       cidade cidade4_
           on endereco3_.cidade_id=cidade4_.id
   left outer join
       estado estado5_
           on cidade4_.estado_id=estado5_.id
   left outer join
       cliente cliente6_
           on endereco3_.cliente_id=cliente6_.id
   left outer join
       pagamento pagamento7_
           on pedido0_.id=pagamento7_.pedido_id
   left outer join
       pagamento_com_cartao pagamento7_1_
           on pagamento7_.pedido_id=pagamento7_1_.pedido_id
   left outer join
       pagamento_com_boleto pagamento7_2_
           on pagamento7_.pedido_id=pagamento7_2_.pedido_id
   where (...)
```

# Problema 2: grande volume de dados e acessos

## Primeira decisão (infra): escala vertical ou horizontal?

- Custo
- Resiliência (alta confiabilidade)
- Crescimento menos limitado
- Virtualização

## BD relacional vs. cluster

## NoSQL

- Primeiras influências: Google (BigTable) e Amazon (Dynamo)
- O nome NoSQL é acidental
- Características mais comuns:
  - Não utilizam modelo relacional
  - Tem uma boa execução em clusters
  - Código aberto
  - Século XXI
  - Não tem esquema

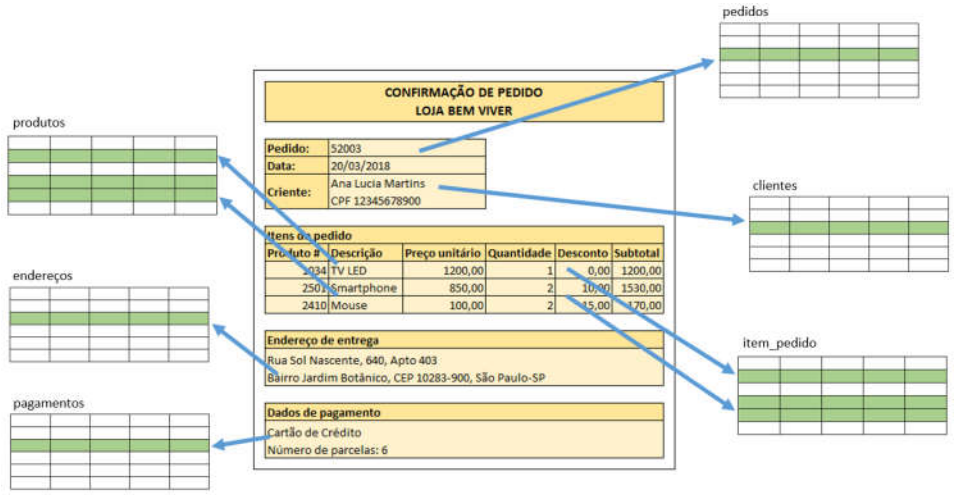
## Duas classes principais de bancos de dados

- Banco de dado orientado a agregados
  - Modelo chave-valor (Riak, Redis)
  - Modelo de documentos (MongoDB, CouchDB)
  - Modelo família de colunas (Cassandra, Apache HBase)
- Banco de dados de grafos (Neo4j)
  *(dados com relacionamentos complexos)*

## Agregado

É um conjunto de objetos relacionados que desejamos tratar como uma unidade.

# Relembrando. Modelo relacional:




```
{
    "id": 1,
    "instante": "30/09/2017 01:32",
    "pagamento": {
        "@type": "pagamentoComCartao",
        "id": 1,
        "estado": "QUITADO",
        "numeroDeParcelas": 6
    },
    "cliente": {
        "id": 1,
        "nome": "Maria Silva"
    },
    "enderecoDeEntrega": {
        "id": 1,
        "logradouro": "Rua Flores",
        "numero": "300"
    },
    "itens": [
        {
            "quantidade": 1,
            "preco": 2000,
            "produto": {"id": 1, "nome": "Computador"}
        },
        {
            "quantidade": 2,
            "preco": 80,
            "produto": {"id": 3, "nome": "Mouse" }
        }
    ]
}
```

# Agregado: pedido

- Conjunto de objetos relacionados, tratados como uma unidade
- Não normalizado

Dados frequentemente acessados juntos > Normalização

# Por que o uso de agregados?

- **Problema 1**: incompatibilidade de impedância
- **Problema 2**: grande volume de dados e acessos (em cluster)

- **Ele já possui a estrutura de objetos associados**

- **É uma unidade natural de replicação e fragmentação**
  - Todos os dados de um agregado estão armazenados **JUNTOS** e no **MESMO NODO** do cluster

- *Nota: não suportam todo suporte ACID como bancos relacionais, mas garantem atomicidade no agregado.*

```
{
    "id": 1,
    "instante": "30/09/2017 01:32",
    "pagamento": {
        "@type": "pagamentoComCartao",
        "id": 1,
        "estado": "QUITADO",
        "numeroDeParcelas": 6
    },
    "cliente": {
        "id": 1,
        "nome": "Maria Silva"
    },
    "enderecoDeEntrega": {
        "id": 1,
        "logradouro": "Rua Flores",
        "numero": "300"
    },
    "itens": [
        {
            "quantidade": 1,
            "preco": 2000,
            "produto": {"id": 1, "nome": "Computador"}
        },
        {
            "quantidade": 2,
            "preco": 80,
            "produto": {"id": 3, "nome": "Mouse" }
        }
    ]
}
```