

# **PostgreSQL**

## **Aula 07**



# Presença

- Linktree: Presente na bio do nosso instagram
- Presença ficará disponível até 1 hora antes da próxima aula
- É necessário 70% de presença para obter o certificado



# Presença





# **Introdução à modelagem de dados**

# Introdução à modelagem

- Etapa fundamental em qualquer projeto de banco de dados
- Objetivo: representar informações de forma organizada e lógica
- Permite armazenar, consultar e analisar dados com eficiência
- Cria uma ponte entre mundo conceitual e físico

# Definição e Objetivo da Modelagem

- Processo de estruturar e organizar dados
- Representa entidades, atributos e relacionamentos
- Facilita uso transacional e analítico
- Base para sistemas consistentes e bem planejados

# Modelagem Transacional (OLTP)

- Voltada ao registro contínuo de transações operacionais
- Foco: desempenho em inserções, atualizações e exclusões
- Exemplos: vendas, cadastros, estoque
- Características:
  - Estrutura altamente normalizada
  - Tabelas pequenas e interligadas
  - Consultas rápidas e simples

# Modelagem Analítica (OLAP)

- Foco em análises e consultas complexas
- Trabalha com grandes volumes de dados históricos
- Suporte à tomada de decisão
- Características:
  - Estruturas desnormalizadas (esquema estrela/floco de neve)
  - Uso de funções agregadas (SUM, AVG, COUNT)
  - Consultas multidimensionais



# Papel do Modelo de Dados

- Serve como mapa estrutural do sistema
- Define relações e permite gerar insights
- Garante:
  - Integridade e consistência
  - Facilidade de manutenção
  - Desempenho otimizado

# Exemplo Prático — OLTP x OLAP

## Exemplo OLTP (Sistema de Vendas):

```
CREATE TABLE vendas (
    id_venda SERIAL PRIMARY KEY,
    id_cliente INT,
    data_venda DATE,
    valor NUMERIC(10,2)
);
```

- Diferença:

- OLTP → operação diária
- OLAP → análise de desempenho

## Exemplo OLAP (Fato de Vendas e Dimensões):

```
-- Tabela fato (dados quantitativos)
CREATE TABLE fato_vendas (
    id_cliente INT,
    id_produto INT,
    id_tempo INT,
    valor_venda NUMERIC(10,2)
);
```



# Boas Práticas de Modelagem

- Comece pelo modelo conceitual (entidades e relacionamentos)
- Normalize até a 3<sup>a</sup> forma normal (modelo transacional)
- Para análise → desnortealize (melhor desempenho)
- Documente chaves e relacionamentos
- Mantenha flexibilidade para crescimento futuro

# Arquitetura Star



# Arquitetura Star

Um dos modelos mais simples e utilizados

- Modelo **OLAP**

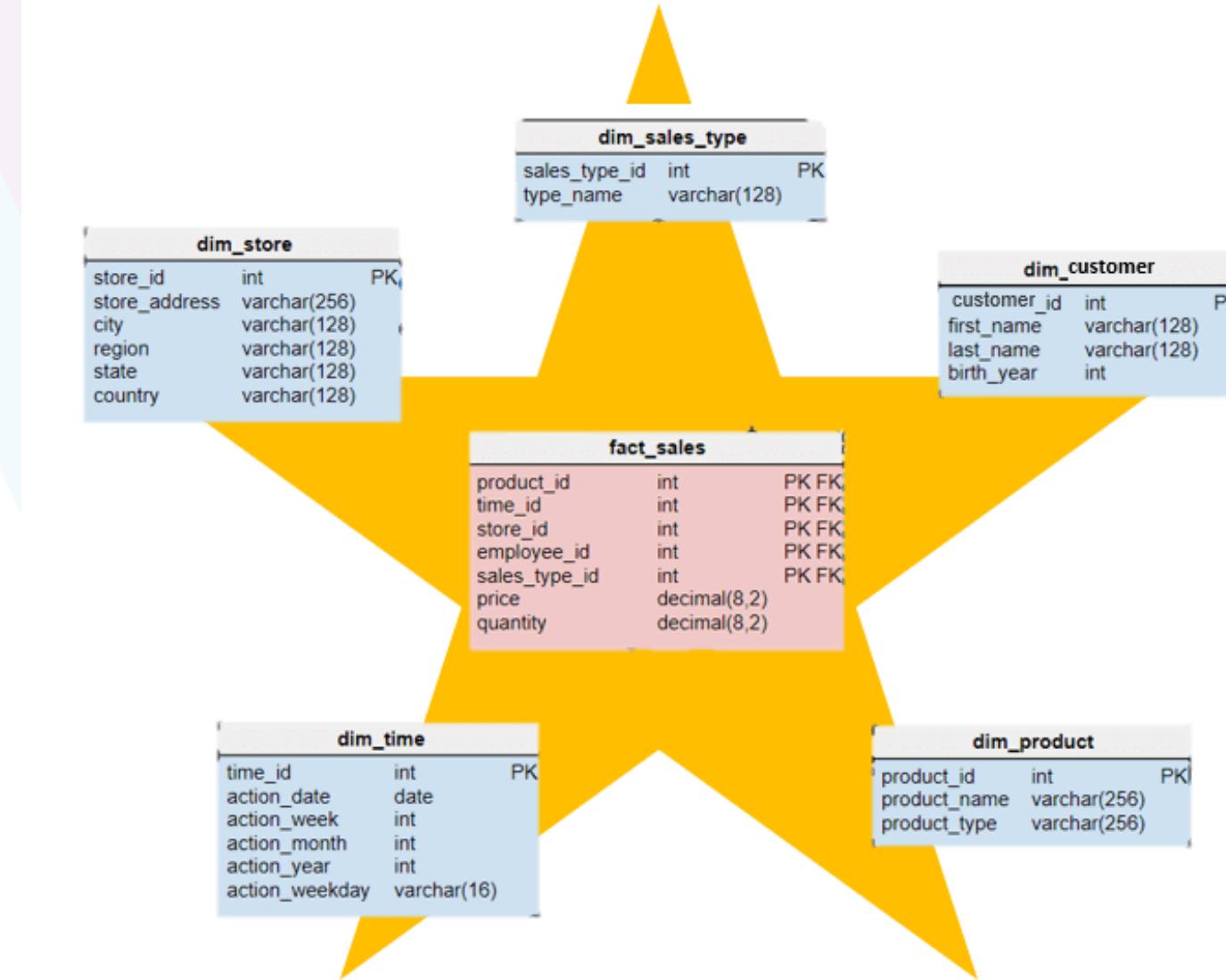
# Arquitetura Star

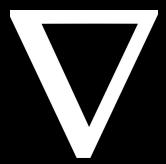
Um dos modelos mais simples e utilizados

- Modelo **OLAP**

## Estrutura Geral

- Tabela Fato
- Tabela Dimensão

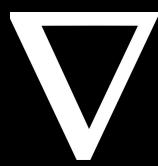




# Arquitetura Star

## Tabela Fato

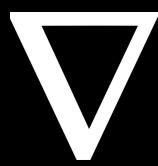
- Tabela Central com chaves estrangeiras e dados numéricos/categóricos
- Aplicação de somas, contagem, média, etc. sobre os dados
- Foco maior em número de linhas



# Arquitetura Star

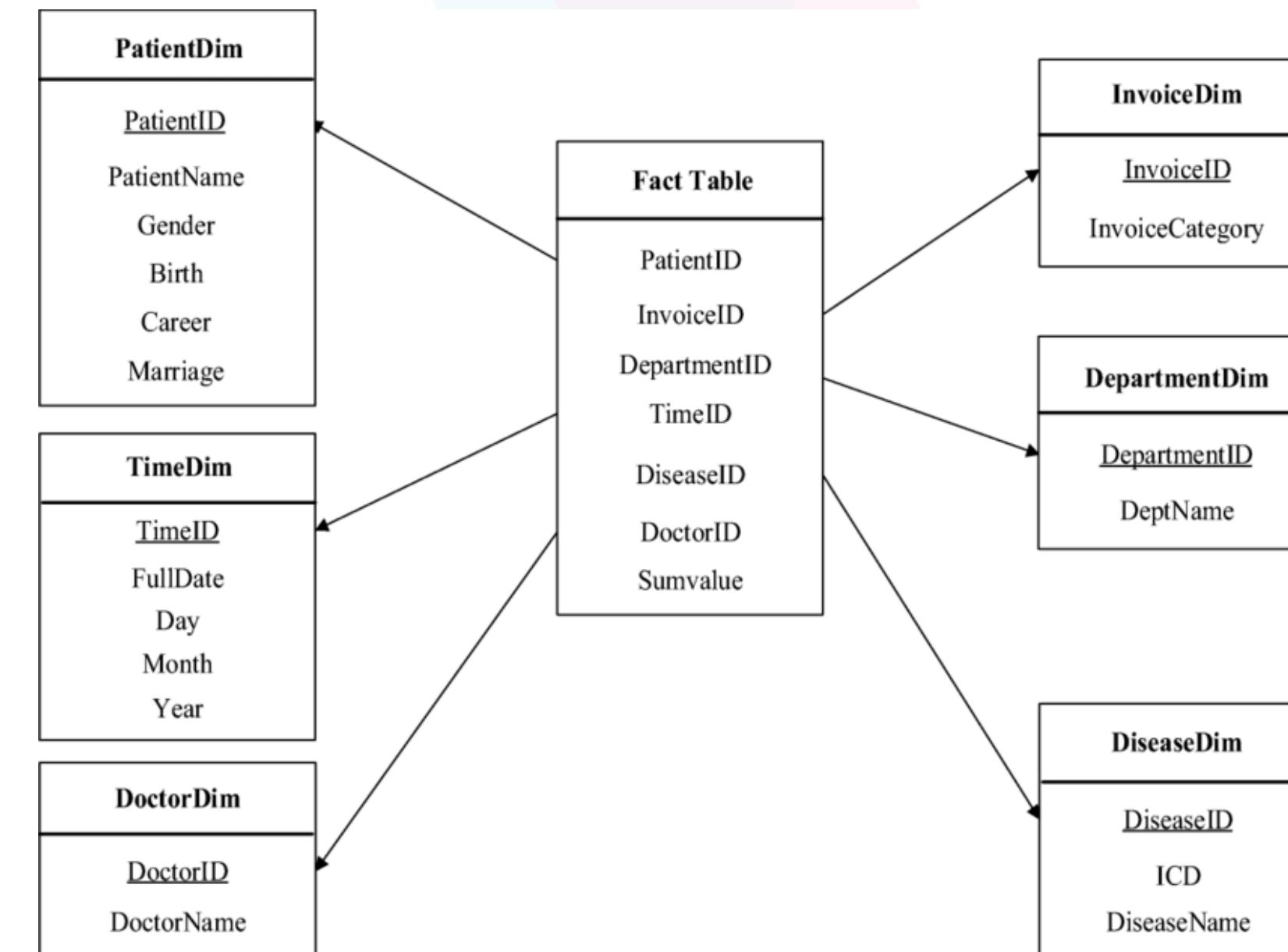
## Tabela Dimensão

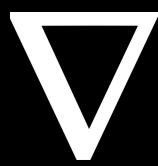
- Tabelas com dados descritivos dispostas em camadas ao redor da tabela fato
- Aplicação de filtragem, agregação, etc.
- Foco maior em número de colunas
- **Não normalizada**



# Arquitetura Star

## Exemplo: Rede Hospitalar





# Arquitetura Star

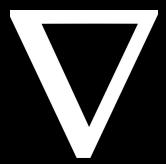
## Granulação

- Nível de detalhamento dos registros da tabela fato
- Maior granulação → Maior detalhamento

# Arquitetura Star

## Granulação

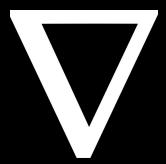
- Nível de detalhamento dos registros da tabela fato
- Maior granulação → Maior detalhamento
- Exemplo: companhia de vendas de produtos
  - Alta Granularidade: cada linha representa uma compra de um produto específico por um cliente
  - Baixa Granularidade: cada linha representa o fechamento de uma compra de um cliente
- Explicabilidade e uso de memória/tempo de execução



# Arquitetura Star

## Vantagens

- Consultas otimizadas e simples
- Entendimento fácil da estrutura e hierarquia



# Arquitetura Star

## Vantagens

- Consultas otimizadas e simples
- Entendimento fácil da estrutura e hierarquia

## Desvantagens

- Simplicidade
- Falta de normalização
- Mais uso de memória



# **Tipos de Tabelas Fato**



# Tabelas Fato - Conceito Geral

- Componentes centrais de um modelo dimensional de banco de dados
- Armazemam medidas quantitativas de eventos de negócio
- Permitem análise e monitoramento de processos
- Classificam-se conforme o tipo de evento ou processo representado

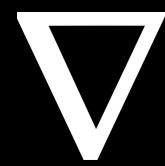


# Fato Transacional

- Registra eventos individuais e detalhados (ex.: vendas, transações bancárias)
  - Cada linha representa um evento específico ocorrido em um momento no tempo
  - Ideal para análises operacionais e relatórios de atividades diárias

Tabela 1: Exemplo de Tabela Fato Transacional — Vendas

ID Venda	Data	Produto	Quantidade	Valor Total (R\$)
001	10/03/2025	Notebook	2	8.000,00
002	10/03/2025	Teclado	5	750,00
003	11/03/2025	Monitor	3	2.700,00
004	12/03/2025	Mouse	4	400,00



# Fato Snapshot Periódico

- Captura o estado de um processo em intervalos regulares
- **Exemplos:** estoque diário, saldo mensal, número de funcionários ativos
- Útil para análises históricas e acompanhamento de desempenho



# Fato Snapshot Acumulativo

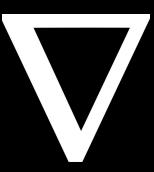
- Representa o ciclo de vida de um processo com início e fim definidos
- Exemplos: acompanhamento de pedidos, projetos ou funil de vendas
- Permite observar a progressão e o tempo de cada etapa do processo

# Exemplo Prático – Fato Transacional

- Tabela de vendas com colunas: ID, Data, Produto, Quantidade e Valor Total
- Cada linha = uma transação de venda
- Mostra atributos descritivos (produto, data) e medidas quantitativas (valor, quantidade)
- Permite análises como total de vendas por período ou por categoria

Tabela 1: Exemplo de Tabela Fato Transacional — Vendas

ID Venda	Data	Produto	Quantidade	Valor Total (R\$)
001	10/03/2025	Notebook	2	8.000,00
002	10/03/2025	Teclado	5	750,00
003	11/03/2025	Monitor	3	2.700,00
004	12/03/2025	Mouse	4	400,00



# Arquitetura Galaxy

# Arquitetura Galaxy

## Modelo Star

- Dificuldade de representação de sistemas complexos

## Empresa de varejo

- Vendas online a clientes finais
- Vendas a revendedores
- Metas a serem batidas pelos vendedores

# Arquitetura Galaxy

## Modelo Star

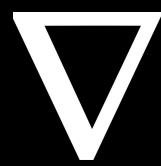
- Dificuldade de representação de sistemas complexos

## Empresa de varejo

- Vendas online a clientes finais
- Vendas a revendedores
- Metas a serem batidas pelos vendedores

Problema: todas os dados estão em tabelas que se comportam como tabela fato

- Necessidade de mais do que 1 tabela fato
- **Arquitetura Galaxy**



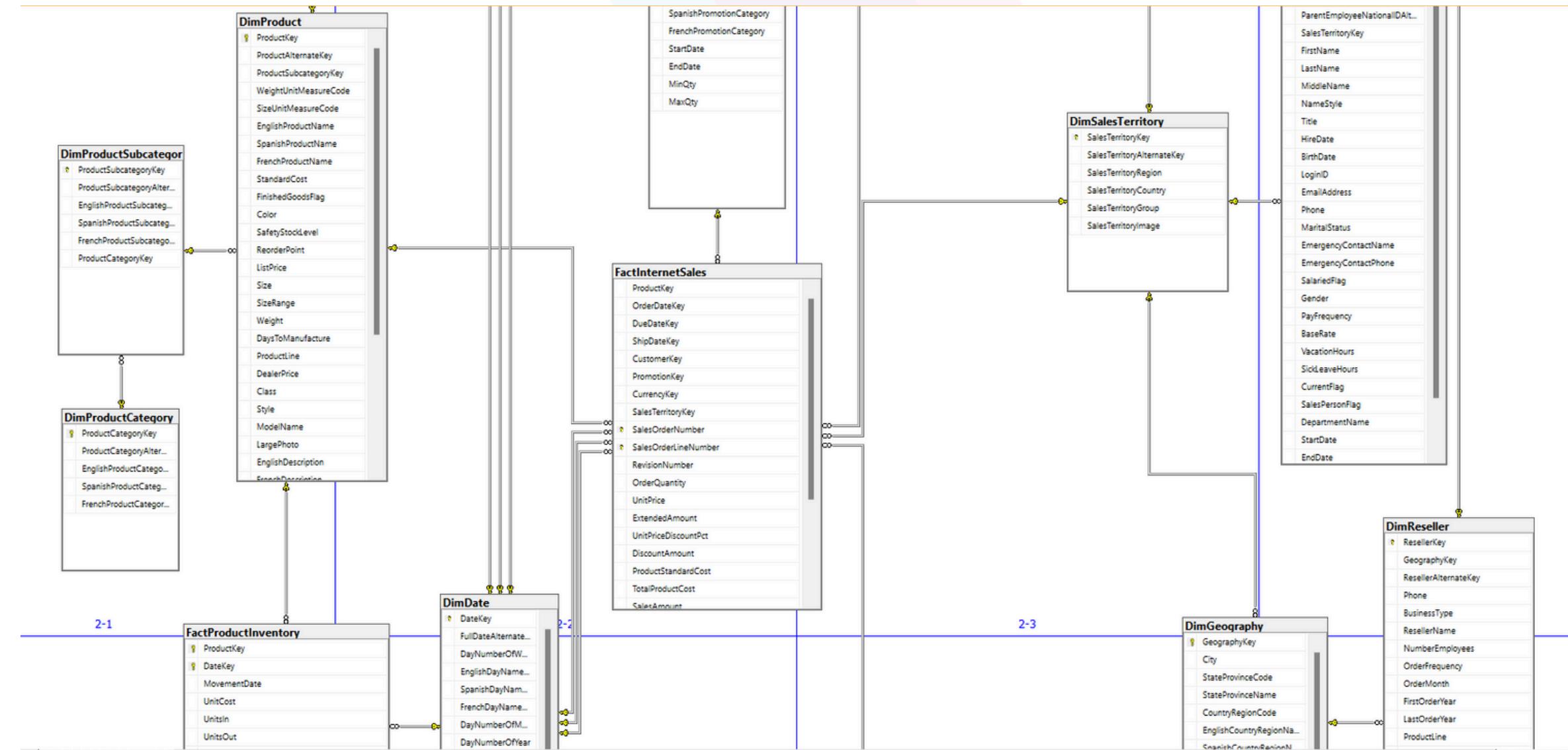
# Arquitetura Galaxy

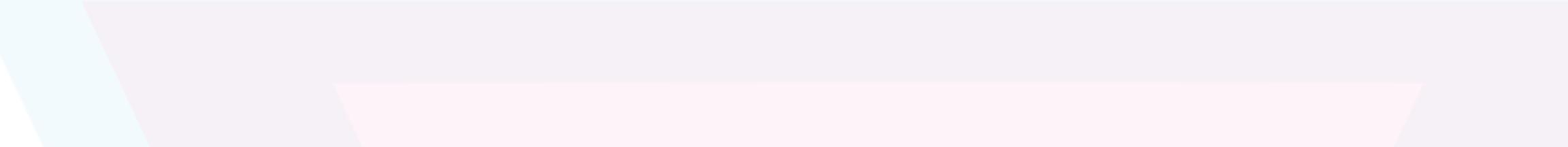
Arquitetura Star com 2 ou mais tabelas fato

- Aplicações do mundo real
- Cada tabela fato tem sua granularidade
- Tabelas de dimensão podem ser compartilhadas entre tabelas fato

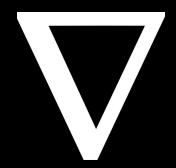
# Arquitetura Galaxy

## AdventureWorks - Microsoft





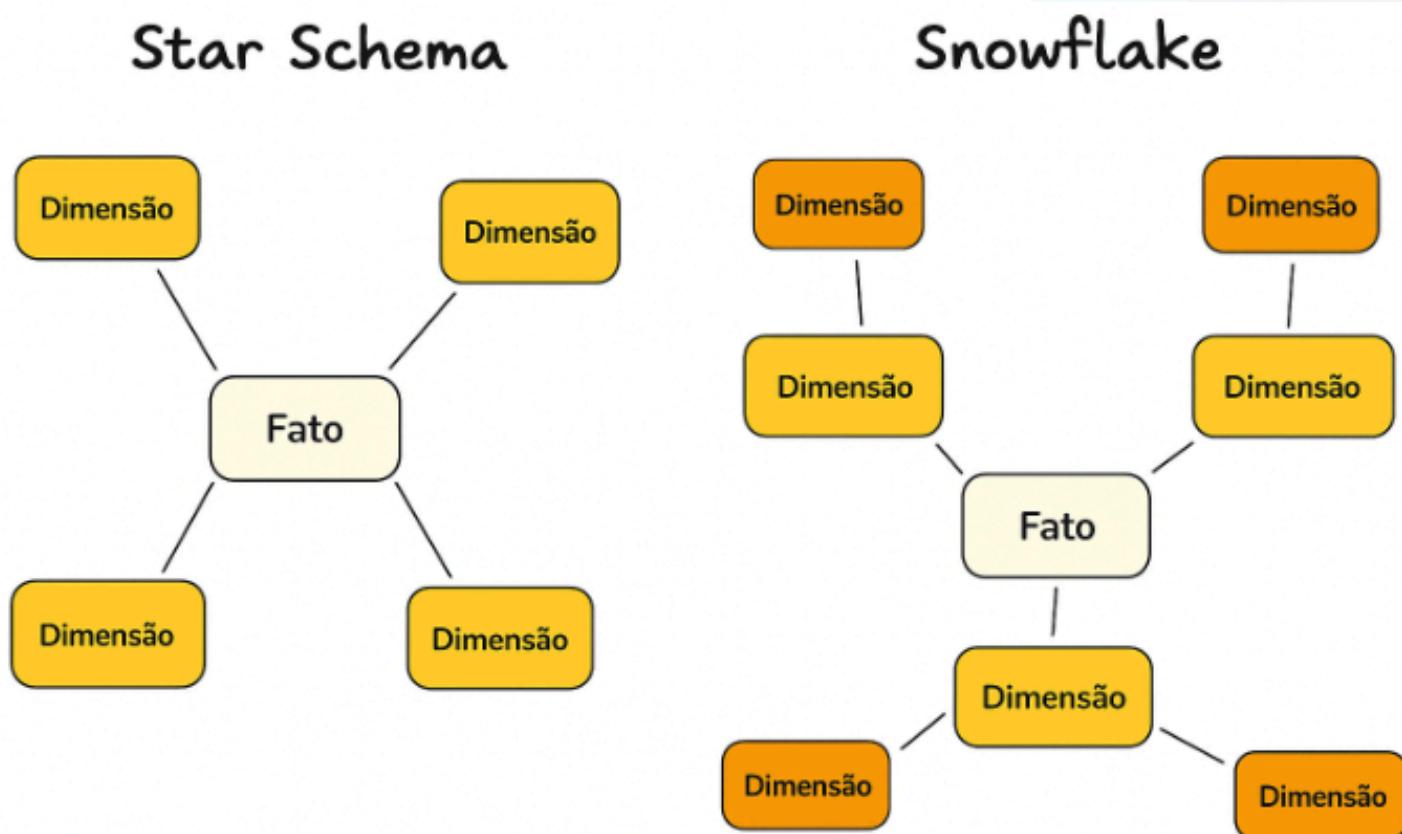
# **Arquitetura SnowFlake**



# Arquitetura SnowFlake

- Variação do modelo em estrela
- Busca reduzir redundâncias por meio da normalização das dimensões.
- Divide dimensões em múltiplas tabelas inter-relacionadas

# Arquitetura SnowFlake



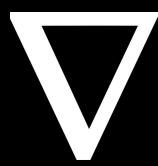
- **Star Schema:**

- Estrutura simples e de fácil compreensão.
- Consultas mais rápidas, pois requerem menos junções.
- Redundância de dados nas dimensões.

- **Snowflake Schema:**

- Estrutura mais normalizada, com menor redundância.
- Consultas mais complexas e potencialmente mais lentas devido ao número maior de junções.
- Facilita manutenção e atualização de dados dimensionais.

# **Slowly Changing Dimensions (SCD)**



# Slowly Changing Dimensions (SCD)

- SCD é um conjunto de técnicas para rastrear e gerenciar mudanças em atributos de tabelas de dimensão (que mudam de forma lenta e imprevisível) ao longo do tempo.
- Exemplo: Lidar com a promoção de um funcionário ou a mudança de endereço de um cliente.

# SCD - Tipo 1

- O valor antigo é sobreescrito pelo novo. É o método mais simples (comando UPDATE em SQL).
- O histórico é permanentemente perdido, mas é o mais eficiente de todos.

ID_funcionario	nome	cargo
123	Murilo Leandro	Desenvolvedor



ID_funcionario	nome	cargo
123	Murilo Leandro	Gerente

# SCD - Tipo 2

- Cria uma nova linha/registro na tabela de dimensão para cada mudança de atributo.
- Preserva o histórico, aumenta o tamanho da tabela, torna operações de query mais complexas.

ID_funcionario	nome	cargo	inicio	fim
123	Murilo Leandro	Desenvolvedor	01-01-2025	31-05-2025
123	Murilo Leandro	Gerente	01-06-2025	31-12-2025



## SCD - Outros

Os tipos 1 e 2 são particularmente comuns, mas existem diversos outros:

- Tipo 3 - Cria uma coluna nova para cada mudança no dado (ex: cargo\_antigo, cargo\_novo)
- Tipo 4 - Similar ao tipo 1, mas cria uma tabela com o histórico do dado.



# **Transformação de um Modelo Genérico**

# Transformação de um Modelo Genérico

Para entender na prática as diferenças entre os modelos analíticos, vamos tomar como ponto de partida um modelo de dados genérico, tipicamente encontrado em sistemas transacionais (OLTP), e transformá-lo em um Star Schema e em um Snowflake Schema.

# ▽

# O Modelo Genérico (Transacional - OLTP)

Vamos imaginar um modelo OLTP simplificado para um sistema de vendas:

- **Cientes** (id\_cliente, nome, cidade, estado, pais)
- **Categorias** (id\_categoria, nome\_categoria)
- **Subcategorias** (id\_subcategoria, nome\_subcategoria, id\_categoria)
- **Produtos** (id\_produto, nome\_produto, id\_subcategoria)
- **Vendas** (id\_venda, data\_venda, id\_cliente)
- **ItensVenda** (id\_item, id\_venda, id\_produto, quantidade, preco\_unitario)

# Problemas deste modelo para Análise (OLAP)

Embora perfeito para o dia a dia, esse modelo é muito ineficiente para relatórios gerenciais. Uma simples pergunta de negócio como:

"Qual a receita total por categoria de produto e por estado do cliente no último trimestre?"

... exigiria uma consulta SQL extremamente complexa e lenta:

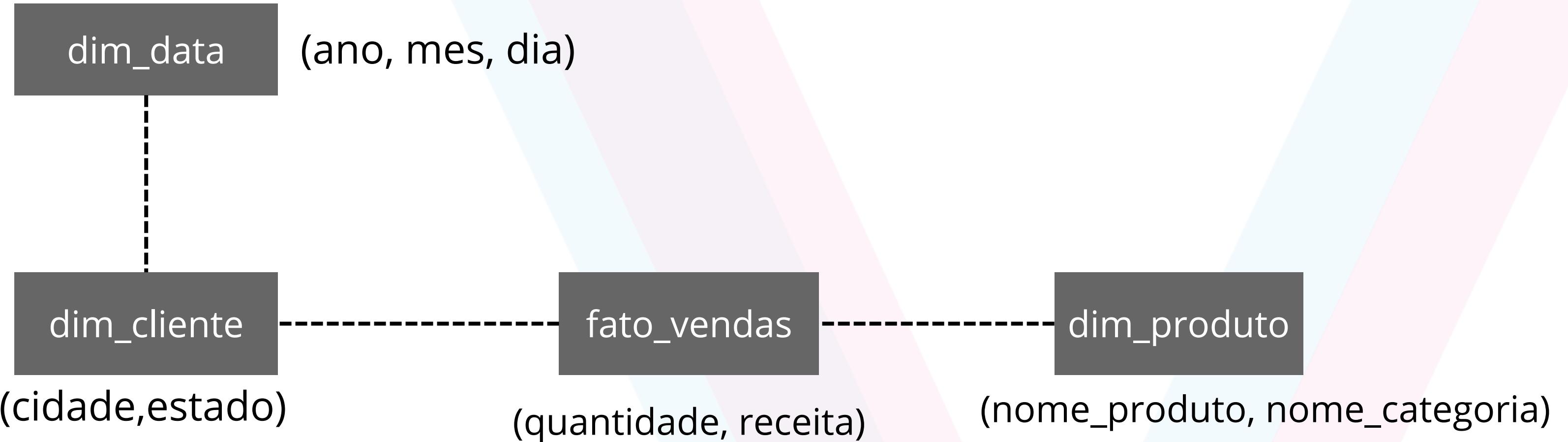
- 1. Vendas JOIN Clientes** (para pegar o estado)
- 2. Vendas JOIN ItensVenda** (para pegar os produtos e valores)
- 3. ItensVenda JOIN Produtos** (para pegar a subcategoria)
- 4. Produtos JOIN Subcategorias** (para pegar a categoria)
- 5. Subcategorias JOIN Categorias** (para pegar o nome da categoria)
- 6. Filtrar por data\_venda**
- 7. Agrupar (GROUP BY) por estado e nome\_categoria**

# ▼

## Transformação para Star Schema (Foco em Desempenho)

- **Tabela Fato (Métricas):** O evento central é o item vendido.
  - `fato_vendas (id_data, id_cliente, id_produto, quantidade_vendida, receita_total)`
- **Tabelas Dimensão (Contexto):**
  - `dim_data (id_data, data_completa, ano, mes, dia, trimestre, dia_da_semana)`
  - `dim_cliente (id_cliente, nome, cidade, estado, pais)`
  - `dim_produto (id_produto, nome_produto, nome_subcategoria, nome_categoria)`

# Transformação para Star Schema (Foco em Desempenho)



**Mudança principal:** Observe a dim\_produto. Nós "achatamos" a hierarquia. As tabelas Categorias e Subcategorias foram fundidas diretamente na dimensão de produto. Há redundância (o nome "Eletrônicos" vai se repetir para cada produto dessa categoria), mas isso é intencional para eliminar JOINs.

# ▽

## O que mudou?

**Vantagem:** A mesma consulta ("Receita por categoria e estado") agora é trivial e muito rápida:

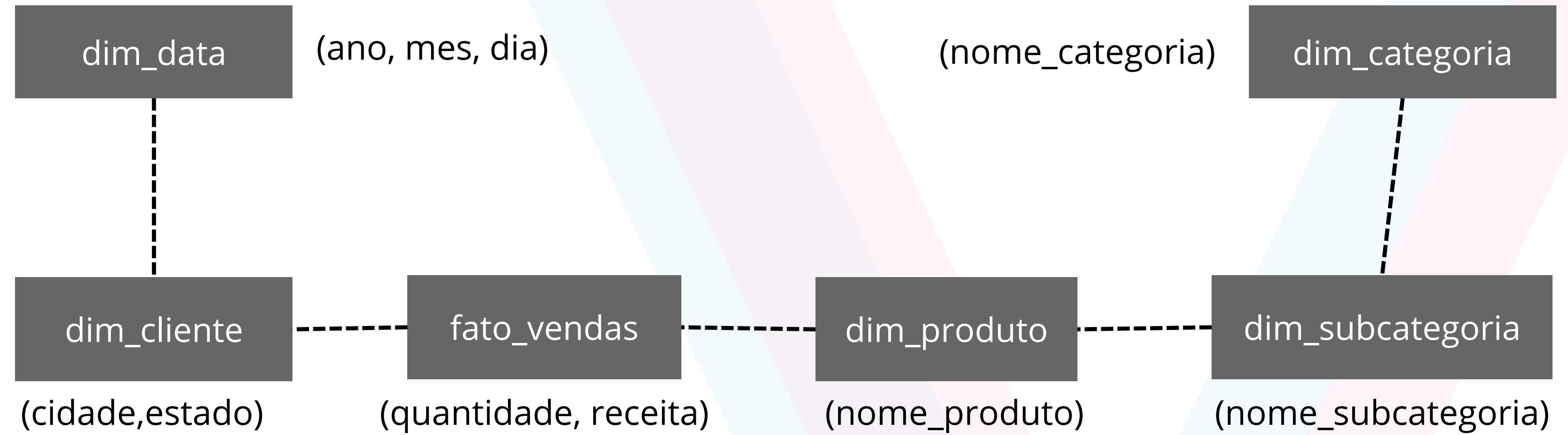
1. `fato_vendas JOIN dim_cliente`
2. `fato_vendas JOIN dim_produto`
3. Agrupar por estado e nome\_categoria

Reduzimos de 5 JOINs para apenas 2.

# Transformação para Snowflake Schema

- **Tabela Fato (Métricas):** Permanece idêntica à do Star Schema.
  - Fato\_vendas (id\_data, id\_cliente, id\_produto, quantidade\_vendida, receita\_total)
- **Tabelas Dimensão (Contexto):**
  - dim\_data (Idêntica)
  - dim\_cliente (Idêntica)
  - dim\_produto (id\_produto, nome\_produto, id\_subcategoria)
  - dim\_subcategoria (id\_subcategoria, nome\_subcategoria, id\_categoria)
  - dim\_categoria (id\_categoria, nome\_categoria)

# Transformação para Snowflake Schema



**Mudança principal:** A dimensão Produto foi "quebrada"(normalizada). As informações de subcategoria e categoria foram movidas para suas próprias tabelas, criando as ramificações que dão nome ao modelo ("floco de neve").



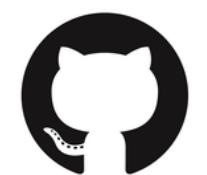
[data@icmc.usp.br](mailto:data@icmc.usp.br)



[@data.icmc](https://www.instagram.com/@data.icmc)



[/c/DataICMC](https://www.youtube.com/c/DataICMC)



[/icmc-data](https://github.com/icmc-data)



[data.icmc.usp.br](http://data.icmc.usp.br)

|| obrigado!