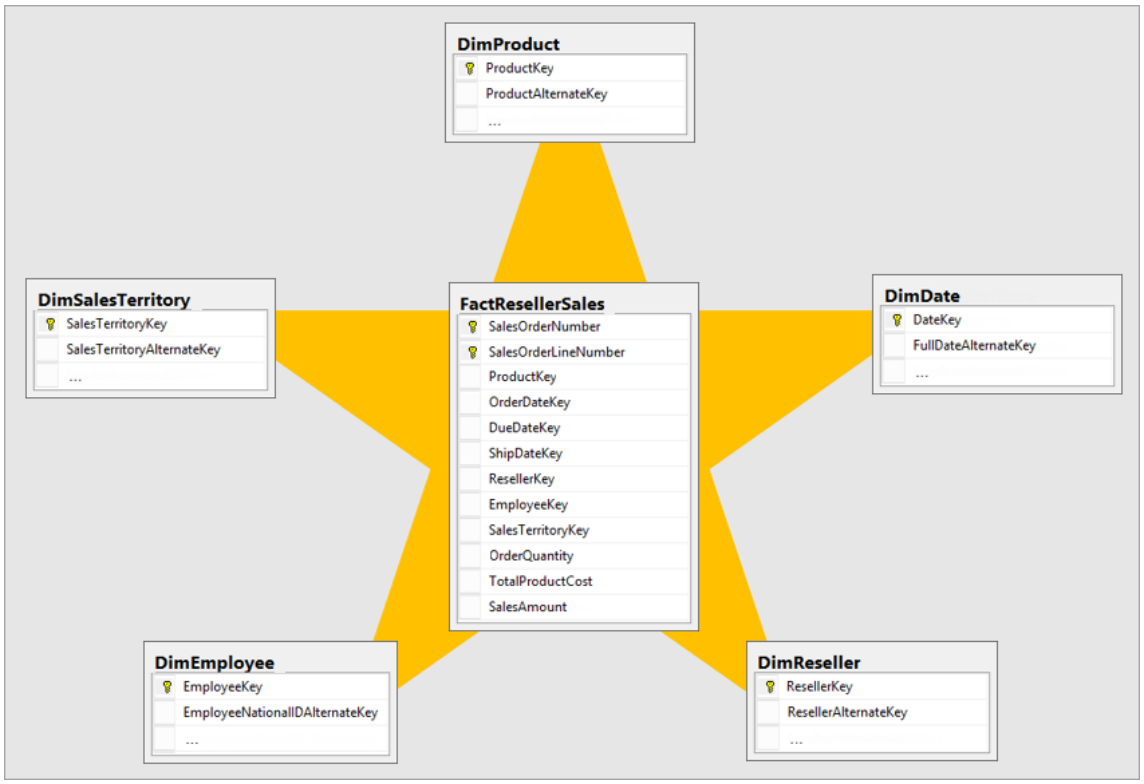
Descrição geral do esquema de estrela

O **esquema de estrela** é uma abordagem de modelação avançada adotada amplamente pelos armazéns de dados relacionais. Requer que os modeladores classifiquem as respetivas tabelas de modelos como *dimensão* ou *facto*.

As **tabelas de dimensão** descrevem entidades empresariais, os *conteúdos* que modela. As entidades podem incluir produtos, pessoas, locais e conceitos, incluindo o próprio tempo. A tabela mais consistente que irá encontrar num esquema de estrela é uma tabela de dimensão de data. Uma tabela de dimensão contém uma coluna de chave (ou colunas) que atua como um identificador exclusivo e colunas descritivas.

Tabela de factos armazenam observações ou eventos e podem ser notas de vendas, quantidade de stock, taxas de câmbio, temperaturas, etc. Uma tabela de factos contém colunas de chave de dimensão relacionadas com tabelas de dimensão e colunas de medidas numéricas. As colunas de chave de dimensão determinam a *dimensionalidade* de uma tabela de factos, enquanto os valores de chave de dimensão determinam a *granularidade* de uma tabela de factos. Por exemplo, considere uma tabela de factos concebida para armazenar objetivos de vendas que têm duas colunas de chave de dimensão **Data** e **ProductKey**. É fácil compreender que a tabela tem duas dimensões. No entanto, a granularidade não pode ser determinada sem considerar os valores de chave de dimensão. Neste exemplo, considere que os valores armazenados na coluna **Data** são o primeiro dia de cada mês. Neste caso, a granularidade está ao nível do mês-produto.

Geralmente, as tabelas de dimensão contêm um número relativamente pequeno de linhas. Por outro lado, as tabelas de factos podem conter um grande número de linhas e continuar a crescer ao longo do tempo.



Normalização vs. desnormalização

Para compreender alguns conceitos de esquema estelar descritos neste artigo, é importante saber dois termos: normalização e desnormalização.

Normalização é o termo usado para descrever dados que são armazenados de uma forma que reduz os dados repetitivos. Considere uma tabela de produtos que tem uma coluna de valor chave única, como a chave do produto, e colunas adicionais que descrevem as características do produto, incluindo nome do produto, categoria, cor e tamanho. Uma tabela de vendas é considerada normalizada quando armazena apenas chaves, como a chave do produto. Na imagem seguinte, note que apenas a coluna **ProductKey** regista o produto.

	SalesOrderNumber	OrderDate	ProductKey	ResellerKey	SalesAmount
1	SO69561	2020-05-31	594	546	226.00
2	SO69560	2020-05-30	513	100	218.45
3	SO69560	2020-05-30	594	100	113.00
4	SO69539	2020-05-28	243	529	858.90
5	SO69539	2020-05-28	378	529	1466.01
6	SO69541	2020-05-28	594	661	113.00
7	SO69542	2020-05-28	243	317	1717.80
8	SO69544	2020-05-28	243	666	3435.60
9	SO69545	2020-05-28	378	436	5864.04
10	SO69532	2020-05-27	594	312	113.00
11	SO69532	2020-05-27	513	312	436.90
12	SO69533	2020-05-27	594	476	226.00

Se, no entanto, a mesa de vendas armazena detalhes do produto para além da chave, é considerado *desnormalizado*. Na imagem seguinte, note que o **ProductKey** e outras colunas relacionadas com o produto registam o produto.

	SalesOrderNumber	OrderDate	ProductKey	Product	Category	Color	Size	ResellerKey	SalesAmount
1	SO69561	2020-05-31	594	Mountain-500 Silver, 48	Bikes	Silver	48	546	226.00
2	SO69560	2020-05-30	513	ML Mountain Frame-W - Silver, 46	Components	Silver	46	100	218.45
3	SO69560	2020-05-30	594	Mountain-500 Silver, 48	Bikes	Silver	48	100	113.00
4	SO69539	2020-05-28	243	HL Road Frame - Red, 44	Components	Red	44	529	858.90
5	SO69539	2020-05-28	378	Road-250 Black, 52	Bikes	Black	52	529	1466.01
6	SO69541	2020-05-28	594	Mountain-500 Silver, 48	Bikes	Silver	48	661	113.00
7	SO69542	2020-05-28	243	HL Road Frame - Red, 44	Components	Red	44	317	1717.80
8	SO69544	2020-05-28	243	HL Road Frame - Red, 44	Components	Red	44	666	3435.60
9	SO69545	2020-05-28	378	Road-250 Black, 52	Bikes	Black	52	436	5864.04
10	SO69532	2020-05-27	594	Mountain-500 Silver, 48	Bikes	Silver	48	312	113.00
11	SO69532	2020-05-27	513	ML Mountain Frame-W - Silver, 46	Components	Silver	46	312	436.90
12	SO69533	2020-05-27	594	Mountain-500 Silver, 48	Bikes	Silver	48	476	226.00

Quando se fornece dados a partir de um ficheiro de exportação ou extrato de dados, é provável que represente um conjunto de dados desnormalizado.

Como descrito neste artigo, deve esforçar-se para desenvolver modelos de dados power bi otimizados com tabelas que representem dados de fato e dimensão normalizados. No entanto, há uma exceção em que uma dimensão do floco de neve deve ser desnormalizada para produzir uma única tabela modelo.

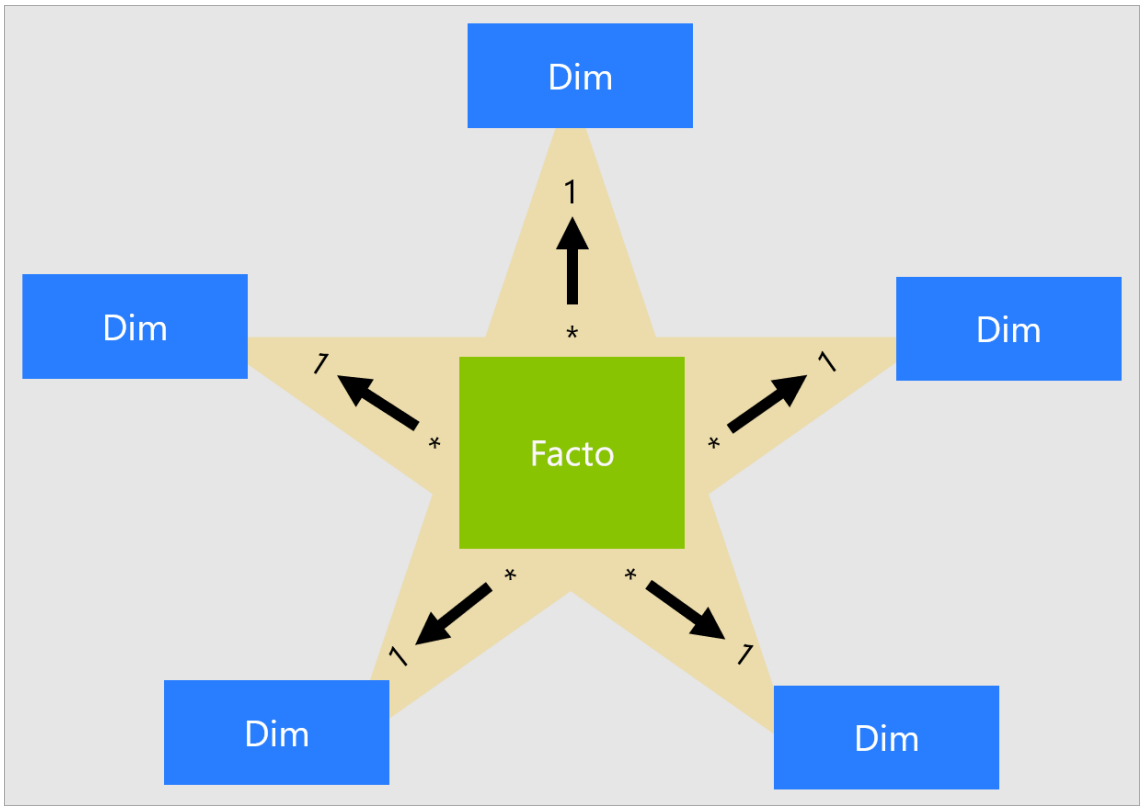
Relevância do esquema de estrela

O design do esquema de estrela e vários conceitos relacionados apresentados neste artigo são altamente relevantes para desenvolver modelos otimizados para desempenho e utilização.

Considere que cada elemento visual do relatório que gera uma consulta que é enviada para análise. Estas consultas são utilizadas para filtrar, agrupar e resumir dados do modelo. Um modelo bem concebido é um modelo que fornece tabelas para filtragem e agrupamento e tabelas para resumos. Este design ajusta-se bem aos princípios do esquema de estrela:

- As tabelas de dimensões suportam *filtragem* e *agrupamento*
- As tabelas de factos suportam *resumos*

Não existe nenhuma propriedade de tabela que os modeladores definam para configurar o tipo de tabela como dimensão ou facto. Na verdade, esta situação é determinada pelas relações de modelos. Uma relação de modelo estabelece um caminho de propagação de filtro entre duas tabelas e é a propriedade **Cardinalidade** da relação que determina o tipo de tabela. Uma cardinalidade de relação comum é *um-para-muitos* ou o inverso *muitos-para-um*. O lado "um" é sempre uma tabela de dimensão, enquanto o lado "muitos" é sempre uma tabela de factos. Para obter mais informações sobre relações.



Um design de modelo bem estruturado deve incluir tabelas de dimensões ou tabelas de factos. Evite misturar os dois tipos numa única tabela. Recomendamos que forneça o número certo de tabelas com as relações certas em vigor. Também é importante que as tabelas de factos carreguem sempre dados a um nível consistente.

Por fim, é importante compreender que o design de modelo ideal é composto por ciência e arte. Por vezes, pode interromper com uma boa orientação quando fizer sentido.

O esquema estrela

A implementação do modelo dimensional em um banco de dados relacional é chamada de esquema estrela (do inglês, star schema). O centro de um esquema estrela é composto por uma tabela fato que agrupa medidas de um processo de negócio específico (ex. pedidos). A tabela fato é relacionada por diferentes tabelas dimensão que agrupam dimensões relacionadas entre si (ex. clientes, locais, produtos). Diferente do banco de dados normalizado do capítulo 4, os esquemas estrela são em geral desnormalizados e apresentam redundância. No mundo de analytics, a facilidade de consulta é mais importante que ganhos marginais de desempenho do banco de dados.

Os esquemas estrela são desenhados para responder perguntas de negócios de áreas específicas da empresa. Ao conjunto de esquemas estrela de áreas específicas é dado o nome de Data Mart. Uma empresa pode, por exemplo, possuir um data mart dos processos da área comercial, outro da produção, marketing, financeiro etc. O grau de compatibilidade entre diferentes marts

depende da forma como o data warehouse foi desenvolvido e também pode variar entre as diferentes arquiteturas de data warehouses.

Para cada tabela dimensão é gerada uma chave única, em geral sem significado de negócio, chamada de surrogate key, ou em geral, SK. Nas tabelas fato, essas chaves são tratadas como chaves estrangeiras para as tabelas dimensão e sua combinação define o grão, ou granularidade, da tabela fato. Em geral, as tabelas dimensão não possuem chaves estrangeiras de modo que um esquema estrela só possui um nível de relacionamentos ou JOINS, facilitando as análises de negócio ad-hoc e por ferramentas de BI.

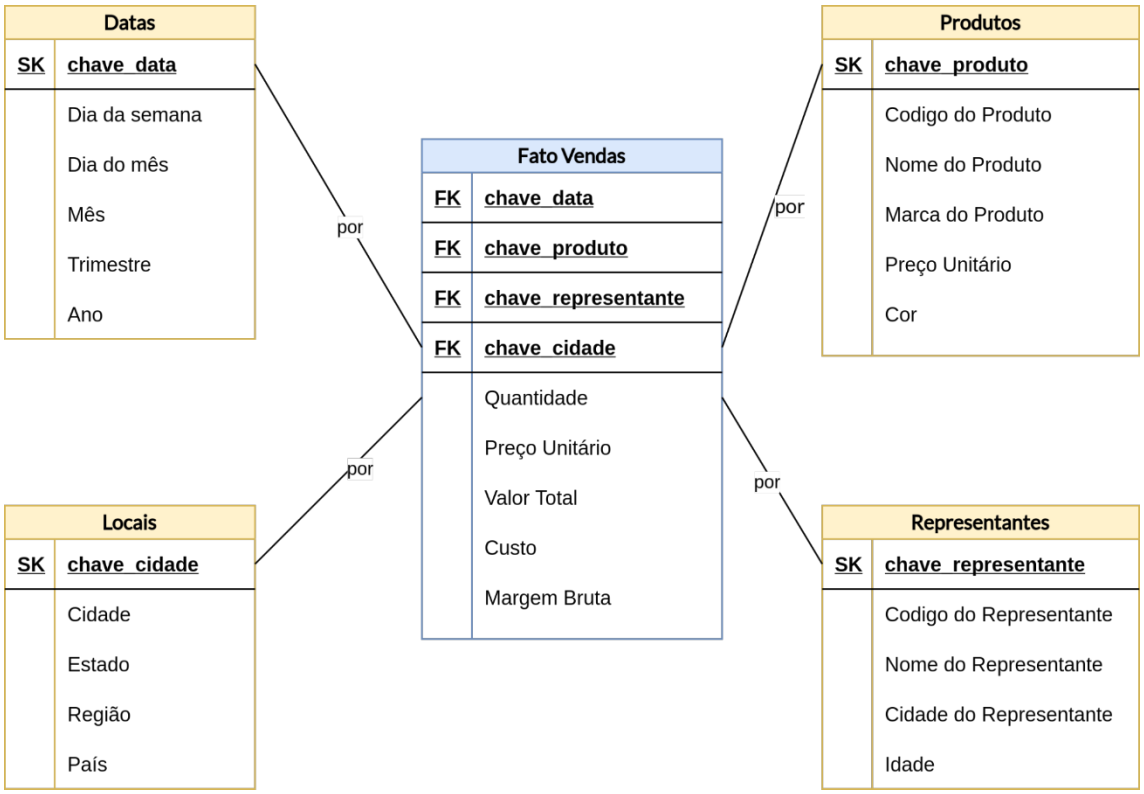


Fig. 30 Um esquema estrela para vendas contém fatos e dimensões

Note que o uso do esquema estrela evita o uso de subqueries para a grande maioria dos casos analíticos. No exemplo abaixo, a consulta de vendas por mês, cidade, representante e produto é facilmente construída em SQL (nomes de colunas foram alterados para replicar o padrão de banco de dados):

```
SELECT
        datas.mes,
        locais.cidade,
        produto.nome_do_produto,
        representante.nome_do_representante,
        SUM(fato_vendas.valor_total)
FROM
        datas,
        locais,
```

```
        produto,
        representante,
        fato_vendas
WHERE
        datas.chave_data = fato_vendas.chave_data
        AND locais.chave_local = fato_vendas.chave_local
        AND produto.chave_produto = fato_vendas.chave_produto
        AND representante.chave_representante
 = fato_vendas.chave_representante
Group by datas.mes,locais.cidade, produto.nome_do_produto,
        representante.nome_do_representante
```

Em alguns casos pode ser necessária a inclusão de tabelas normalizadas em um data mart. Quando isso ocorre, o esquema resultante é chamado de modelo *snowflake*. Existem ainda outros tipos de tabelas como bridge e hierarquias que respondem a problemas práticos que não são completamente satisfeitos pelo esquema estrela tradicional. Em geral, a omissão ou utilização de outras estruturas no design do data warehouse possuem prós e contras e precisam ser avaliadas caso a caso pelo analista.

Estudo de Caso

Elaborar um DataWareHouse com o objetivo de analisar a quantidade de funcionários e a somatória salarial agrupados por Sexo, Cidade e Estado.

REFERÊNCIA

https://www.engdeanalytics.com.br/chapters/08/03/esquema_estrela.html

https://learn.microsoft.com/pt-pt/power-bi/guidance/star-schema

COMANDOS

```
--PESQUISA GERAL
--USANDO O INNER JOIN
SELECT NM_FUNCIONARIO,DS_SEXO,DT_ADMISSAO,VL_SALARIO,NM_BAIRRO,
NM_DEPTO,NM_CIDADE,DS_SIGLA,NM_ESTADO
FROM FUNCIONARIO AS F
INNER JOIN BAIRRO AS B ON B.CD_BAIRRO=F.CD_BAIRRO
INNER JOIN DEPTO AS D ON D.CD_DEPTO=F.CD_DEPTO
INNER JOIN CIDADE AS C ON C.CD_CIDADE=F.CD_CIDADE
INNER JOIN ESTADO AS E ON E.CD_ESTADO=C.CD_ESTADO
ORDER BY NM_FUNCIONARIO
--CRIAR UMA VIEW MELHORAR A PERFORMANCE
CREATE VIEW VW_GERAL
AS
SELECT NM_FUNCIONARIO,DS_SEXO,DT_ADMISSAO,VL_SALARIO,NM_BAIRRO,
NM_DEPTO,NM_CIDADE,DS_SIGLA,NM_ESTADO
FROM FUNCIONARIO AS F
```

```
INNER JOIN BAIRRO AS B ON B.CD_BAIRRO=F.CD_BAIRRO
INNER JOIN DEPTO AS D ON D.CD_DEPTO=F.CD_DEPTO
INNER JOIN CIDADE AS C ON C.CD_CIDADE=F.CD_CIDADE
INNER JOIN ESTADO AS E ON E.CD_ESTADO=C.CD_ESTADO
--PESQUISAR A VIEW
SELECT * FROM VW_GERAL ORDER BY NM_FUNCIONARIO
```

COMANDOS DAS PROCEDURES

```
CREATE PROCEDURE SP_CARGA_BAIRRO
AS
IF NOT EXISTS(SELECT * FROM SYSOBJECTS WHERE NAME = 'DIM_BAIRRO')
   BEGIN
      CREATE TABLE DIM_BAIRRO
         (CD_BAIRRO INT IDENTITY(1,1) NOT NULL,
         NM_BAIRRO VARCHAR(30) NULL,
         CONSTRAINT PK_BAIRRO PRIMARY KEY(CD_BAIRRO))
   END
ELSE
   BEGIN
      TRUNCATE TABLE DIM_BAIRRO
   END
INSERT INTO DIM_BAIRRO
SELECT NM_BAIRRO FROM RH_ADEMIR..BAIRRO
--
CREATE PROCEDURE SP_CARGA_DEPTO
AS
IF NOT EXISTS(SELECT * FROM SYSOBJECTS WHERE NAME = 'DIM_DEPTO')
   BEGIN
      CREATE TABLE DIM_DEPTO
         (CD_DEPTO INT IDENTITY(1,1) NOT NULL,
         NM_DEPTO VARCHAR(30) NULL,
         CONSTRAINT PK_DEPTO PRIMARY KEY(CD_DEPTO))
   END
ELSE
   BEGIN
      TRUNCATE TABLE DIM_DEPTO
   END
INSERT INTO DIM_DEPTO
SELECT NM_DEPTO FROM RH_ADEMIR..DEPTO
--
CREATE PROCEDURE SP_CARGA_CIDADE
AS
IF NOT EXISTS(SELECT * FROM SYSOBJECTS WHERE NAME = 'DIM_CIDADE')
   BEGIN
      CREATE TABLE DIM_CIDADE
         (CD_CIDADE INT IDENTITY(1,1) NOT NULL,
         NM_CIDADE VARCHAR(50) NULL,
         CONSTRAINT PK_CIDADE PRIMARY KEY(CD_CIDADE))
   END
ELSE
   BEGIN
      TRUNCATE TABLE DIM_CIDADE
   END
INSERT INTO DIM_CIDADE
SELECT NM_CIDADE FROM RH_ADEMIR..CIDADE
--
CREATE PROCEDURE SP_CARGA_ESTADO
AS
IF NOT EXISTS(SELECT * FROM SYSOBJECTS WHERE NAME = 'DIM_ESTADO')
   BEGIN
      CREATE TABLE DIM_ESTADO
```

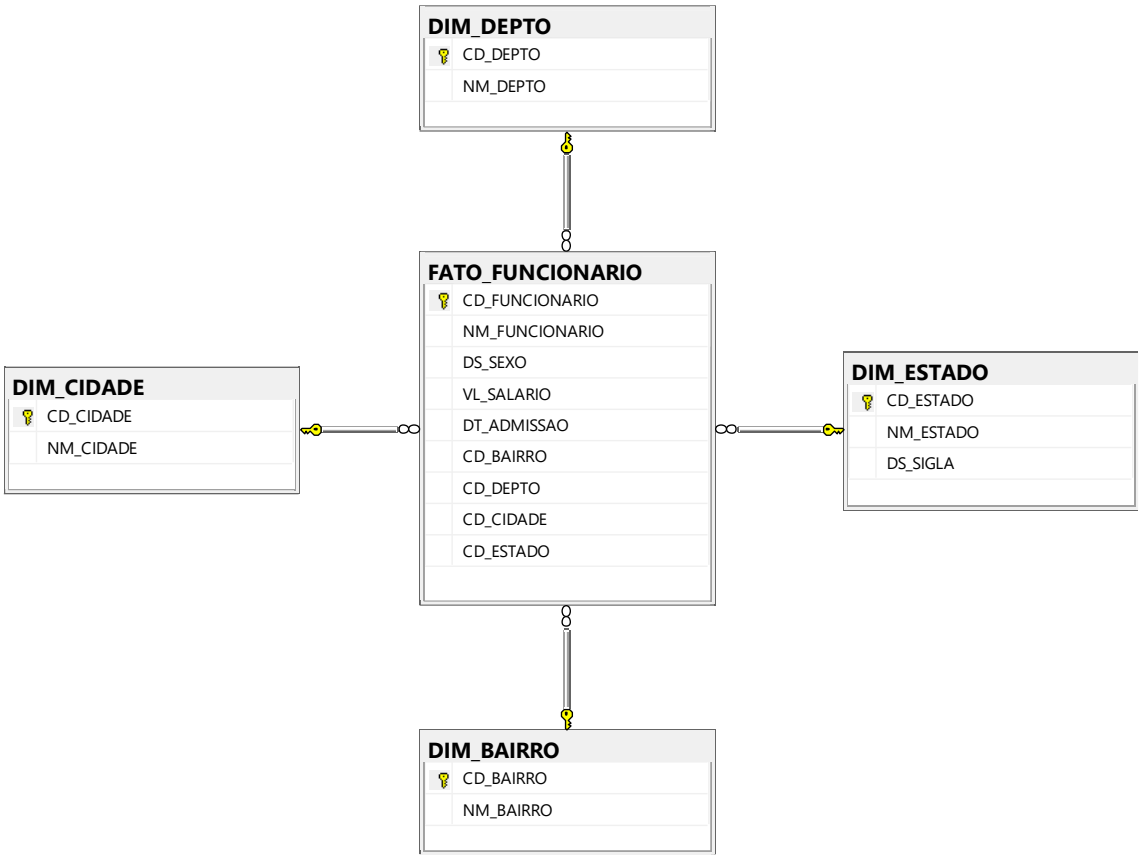
```
        (CD_ESTADO INT IDENTITY(1,1) NOT NULL,
        NM_ESTADO VARCHAR(30) NULL,
        DS_SIGLA VARCHAR(2) NULL,
        CONSTRAINT PK_ESTADO PRIMARY KEY(CD_ESTADO))
   END
ELSE
   BEGIN
      TRUNCATE TABLE DIM_ESTADO
   END
INSERT INTO DIM_ESTADO
SELECT NM_ESTADO,DS_SIGLA FROM RH_ADEMIR..ESTADO

--CRIAR A TABELA FATO CHAMADA FUNCIONÁRIO
--USANDO O RECURSO DE PROCEDURE
CREATE PROCEDURE SP_CRIACAO_GERAL
AS
EXECUTE SP_CARGA_BAIRRO
EXECUTE SP_CARGA_DEPTO
EXECUTE SP_CARGA_CIDADE
EXECUTE SP_CARGA_ESTADO

IF NOT EXISTS(SELECT * FROM SYSOBJECTS WHERE NAME = 'FATO_FUNCIONARIO')
   BEGIN
      CREATE TABLE FATO_FUNCIONARIO
        (CD_FUNCIONARIO INT IDENTITY(1,1) NOT NULL,
        NM_FUNCIONARIO VARCHAR(50) NULL,
        DS_SEXO VARCHAR(1) NULL,
        VL_SALARIO NUMERIC(18,2) NULL,
        DT_ADMISSAO DATE NULL,
        CD_BAIRRO INT NOT NULL,
        CD_DEPTO INT NOT NULL,
        CD_CIDADE INT NOT NULL,
        CD_ESTADO INT NOT NULL,
        CONSTRAINT PK_FUNCIONARIO PRIMARY KEY(CD_FUNCIONARIO),
        CONSTRAINT FK_BAIRRO FOREIGN KEY(CD_BAIRRO) REFERENCES
        DIM_BAIRRO(CD_BAIRRO),
        CONSTRAINT FK_DEPTO FOREIGN KEY(CD_DEPTO) REFERENCES
        DIM_DEPTO(CD_DEPTO),
        CONSTRAINT FK_CIDADE FOREIGN KEY(CD_CIDADE) REFERENCES
        DIM_CIDADE(CD_CIDADE),
        CONSTRAINT FK_ESTADO FOREIGN KEY(CD_ESTADO) REFERENCES
        DIM_ESTADO(CD_ESTADO))
   END
ELSE
   BEGIN
      TRUNCATE TABLE FATO_FUNCIONARIO
   END
INSERT INTO FATO_FUNCIONARIO
SELECT NM_FUNCIONARIO,DS_SEXO,VL_SALARIO,DT_ADMISSAO,
B.CD_BAIRRO,D.CD_DEPTO,C.CD_CIDADE,E.CD_ESTADO
FROM RH_ADEMIR..FUNCIONARIO AS F
INNER JOIN RH_ADEMIR..BAIRRO AS B ON B.CD_BAIRRO=F.CD_BAIRRO
INNER JOIN RH_ADEMIR..DEPTO AS D ON D.CD_DEPTO=F.CD_DEPTO
INNER JOIN RH_ADEMIR..CIDADE AS C ON C.CD_CIDADE=F.CD_CIDADE
INNER JOIN RH_ADEMIR..ESTADO AS E ON E.CD_ESTADO=C.CD_ESTADO

SELECT * FROM FATO_FUNCIONARIO
```

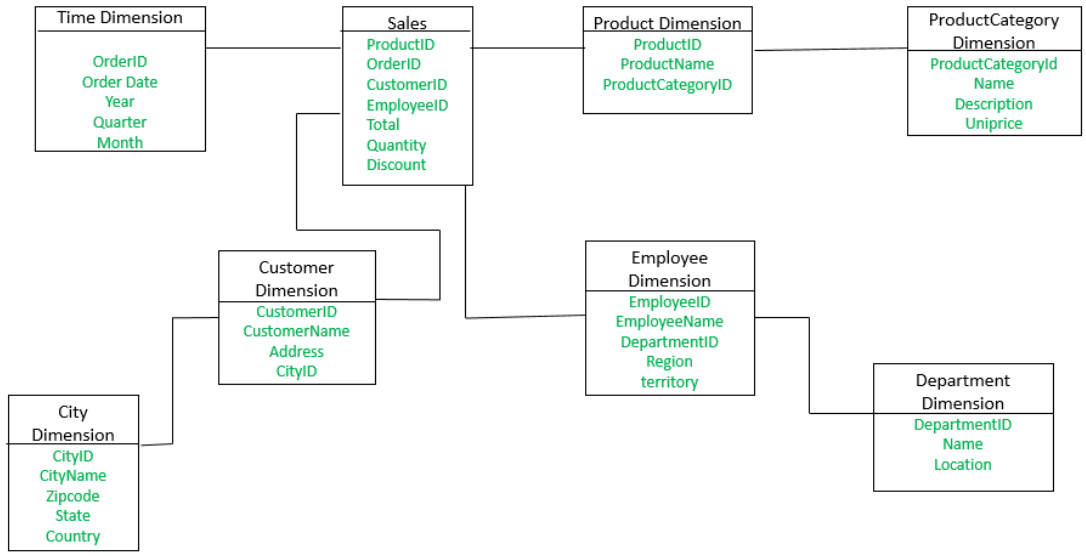
MODELO ESTRELA



ESQUEMA DO FLOCO DE NEVE NO MODELO DE DATA WAREHOUSE

Introdução: o esquema do floco de neve é uma variante do esquema em estrela. Aqui, a tabela de fatos centralizada é conectada a várias dimensões. No esquema em floco de neve, as dimensões estão presentes em uma forma normalizada em várias tabelas relacionadas. A estrutura em floco de neve se materializou quando as dimensões de um esquema em estrela são detalhadas e altamente estruturadas, tendo vários níveis de relacionamento, e as tabelas filho têm várias tabelas pai. O efeito floco de neve afeta apenas as tabelas de dimensão e não afeta as tabelas de fatos.

Exemplo:



A tabela de dimensão **Employee** agora contém os atributos: EmployeeID, EmployeeName, DepartmentID, Region, Territory. O atributo DepartmentID é vinculado à tabela **Employee** com a tabela de dimensão **Department**. A dimensão **Departamento** é usada para fornecer detalhes sobre cada departamento, como o Nome e a Localização do departamento. A tabela de dimensões do **cliente** agora contém os atributos: CustomerID, CustomerName, Address, CityID. Os atributos CityID vinculam a tabela de dimensão do **cliente** com a tabela de dimensão da **cidade**. A tabela de dimensão **City** possui detalhes sobre cada cidade, como CityName, Zipcode, State e Country.

A principal diferença entre o esquema em estrela e o esquema em floco de neve é que a tabela de dimensões do esquema em floco de neve é mantida na forma normalizada para reduzir a redundância. A vantagem aqui é que essas tabelas (normalizadas) são fáceis de manter e economizam espaço de armazenamento. No entanto, também significa que mais junções serão necessárias para executar a consulta. Isso terá um impacto adverso no desempenho do sistema.

O que é snowflaking?

O design do floco de neve é o resultado de uma expansão adicional e normalizado da tabela de dimensão. Em outras palavras, uma tabela de dimensão é considerada flocos de neve se o atributo de baixa cardinalidade das dimensões tiver sido dividido em tabelas normalizadas separadas. Essas tabelas são então unidas à tabela

de dimensão original com restrições referenciais (restrição de chave estrangeira).

Geralmente, snowflaking não é recomendado na tabela de dimensão, pois isso dificulta a compreensão e o desempenho do modelo de dimensão, pois mais tabelas seriam necessárias para serem unidas para satisfazer as consultas.

Características do esquema do floco de neve: o modelo de dimensão de um floco de neve nas seguintes condições:

- O esquema em floco de neve usa pouco espaço em disco.
- É fácil implementar a dimensão que é adicionada ao esquema.
- Existem várias tabelas, portanto, o desempenho é reduzido.
- A tabela de dimensão consiste em dois ou mais conjuntos de atributos que definem informações em diferentes grãos.
- Os conjuntos de atributos da mesma tabela de dimensão estão sendo preenchidos por diferentes sistemas de origem.

Vantagens: Existem duas vantagens principais do esquema em floco de neve, fornecidas abaixo:

- Ele fornece dados estruturados que reduzem o problema de integridade dos dados.
- Ele usa pouco espaço em disco porque os dados são altamente estruturados.

Desvantagens:

- Snowflaking reduz o espaço consumido pelas tabelas de dimensão, mas em comparação com todo o data warehouse, a economia geralmente é insignificante.
- Evite flocos de neve ou normalização de uma tabela de dimensão, a menos que seja necessário e apropriado.
- Não flua as hierarquias de uma tabela de dimensão em tabelas separadas. As hierarquias devem pertencer à tabela de dimensão apenas e nunca devem ser flocos de neve.
- Várias hierarquias que podem pertencer à mesma dimensão foram projetadas com o mínimo de detalhes possível.