

RAFAEL PITON

DATA WAREHOUSE PASSO A PASSO

O GUIA
PRÁTICO DE
COMO CONSTRUIR
UM DATA WAREHOUSE
DO ZERO

RAFAEL PITON

DATA WAREHOUSE

PASSO A PASSO

**O GUIA PRÁTICO DE COMO CONSTRUIR UM DATA
WAREHOUSE DO ZERO**

Porto Alegre
Raizzer
2018

SUMÁRIO

SUMÁRIO.....	3
O QUE É ESTE LIVRO.....	7
FUNDAMENTOS DO BI	8
O QUE É BUSINESS INTELLIGENCE?	10
POR QUE BI?	16
QUE TIPO DE PERGUNTA O BI RESPONDE?	20
AS ETAPAS PARA TOMADA DE DECISÃO	26
ARQUITETURA DE BUSINESS INTELLIGENCE.....	35
DATA SOURCE	35
DATA INTEGRATION	37
DATA WAREHOUSE	40
DATA ANALYSIS.....	41
DATA VISUALIZATION.....	48
FUNDAMENTOS DO DATA WAREHOUSE.....	55
FERRAMENTAS DE DATA WAREHOUSE.....	57
DADOS NORMALIZADOS VS DADOS DESNORMALIZADOS.....	83
OBJETIVOS DO DATA WAREHOUSE	88

ARQUITETURA DE DATA WAREHOUSE	92
FUNDAMENTOS DA MODELAGEM DIMENSIONAL	97
STAR SCHEMA.....	99
TABELA FATO	101
TABELA DIMENSÃO	103
TIPOS DE CHAVES.....	106
STAR SCHEMA COOKBOOK.....	108
CRIANDO MEU PRIMEIRO PROJETO NO SQL POWER ARCHITECT	117
PASSO 1: SALVAR O PROJETO	118
PASSO 2: CRIAR DIMENSÕES	118
PASSO 3: CRIAR FATOS E MÉTRICAS	138
PASSO 4: FAZER JOINS ENTRE FATOS E DIMENSÕES	142
PASSO 5: CARREGAR A MODELAGEM NO BANCO DE DADOS..	147
DESIGN PATTERNS	154
TIPOS DE MÉTRICAS	156
MÉTRICA ADITIVA	156
MÉTRICA DERIVADA.....	158
MÉTRICA SEMIADITIVA	161
MÉTRICA NÃO ADITIVA	162
PASSO A PASSO: COMO FAZER MÉTRICAS ADITIVAS E DERIVADAS	163
TIPOS DE DIMENSÕES	169
DIMENSÃO HIERÁRQUICA: PAI-FILHO.....	169
DIMENSÃO ESPECIAL: DIMENSÃO DE TEMPO	175
DIMENSÃO DEGENERADA	177

SLOWLY CHANGING DIMENSION.....	178
PASSO A PASSO: COMO CRIAR HIERARQUIA PAI-FILHO	181
PASSO A PASSO: COMO CRIAR DIMENSÃO DE TEMPO.....	188
PASSO A PASSO: COMO CRIAR DIMENSÃO DEGENERADA	197
PASSO A PASSO: COMO CRIAR SLOWLY CHANGING DIMENSION	199
FATO TRANSACIONAL	205
PASSO A PASSO: COMO CRIAR FATO TRANSACIONAL.....	209
AGILE DATA WAREHOUSE DESIGN	217
PASSO 1 - ENTREVISTAR OS KEY USERS.....	218
PASSO 2 - IDENTIFICAR A FATO	224
PASSO 3 - IDENTIFICAR AS MÉTRICAS	225
PASSO 4 - IDENTIFICAR AS DIMENSÕES	225
PASSO 5 - DEFINIR A HIERARQUIA	227
PASSO 6 - IDENTIFICAR OS ATRIBUTOS	227
BUS MATRIX	230
BUS MATRIX POR PROCESSO DE NEGÓCIO	231
BUS MATRIX POR MÉTRICAS E DIMENSÕES	233
BUS MATRIX POR FATO E DIMENSÕES.....	234
FUNDAMENTOS DO ETL	236
O QUE É ETL?	238
O QUE É STAGING AREA?	242
VARIACÕES DO ETL	246
PROCESSO DE ETL	250
PASSO 0 - CRIAR A ORIGEM.....	255

PASSO 1 - CRIAR A STAGING AREA.....	272
PASSO 2 – CARREGAR A STAGING AREA	276
PASSO 3 – CRIAR AS DIMENSÕES.....	280
PASSO 4 – CARREGAR AS DIMENSÕES	285
PASSO 5 – CRIAR A FATO	293
PASSO 6 – CARREGAR A FATO	295
PASSO 7 – VALIDAR A CARGA	306
MODELAGEM DIMENSIONAL AVANÇADA.....	308
DIMENSÃO NÃO QUALIFICADA	310
ROLE-PLAYING DIMENSION.....	314
CONFORMED DIMENSION.....	316
JUNK DIMENSION	319
FATO AGREGADA.....	322
FATO CONSOLIDADA.....	326
FATO SNAPSHOT PERIÓDICO.....	330
FATO SNAPSHOT ACUMULADO.....	333
FATO SEM FATO.....	337

O QUE É ESTE LIVRO

Você pode considerar este livro como um download da parte mais relevante de todo o conteúdo de Data Warehouse que está no meu cérebro.

Depois de participar de projetos como Lojas Renner, Emissora SBT, Cooxupé, C.Vale, M. Dias Branco, Oxiteno, Sodexo, entre outros gigantes, eu reuni todas minhas experiências para escrever este livro.

Como seria inviável colocar todo meu conhecimento nestas páginas, deixei aqui tudo aquilo que você vai, sem dúvida, usar para sobreviver no mercado de trabalho. O que eu estou te entregando aqui é o atalho que você precisa para se tornar um especialista de Data Warehouse agora mesmo. Neste livro você vai encontrar:

- muita prática;
- muito conceito;
- muitos exemplos do que eu já experienciei na vida real;
- nada de teoria inútil e encheção de linguiça.

Quando você terminar de ler este livro, terá todo o conhecimento que precisa para no dia seguinte desenvolver projetos de Data Warehouse onde quiser.

FUNDAMENTOS DO BI

Antes de eu te ensinar o exato passo a passo de como construir um Data Warehouse do zero, eu preciso que você entenda primeiro o que é Business Intelligence e porque o Data Warehouse existe, onde que ele está inserido nesse ecossistema.

O Data Warehouse é o coração das soluções de BI, é onde tudo fica armazenado. E você precisa entender como funcionam essas soluções para entender porque você precisa do Data Warehouse e para que fim ele vai ser utilizado.

Se você só entender como criar e não souber porque deve fazer isso, vai acabar se tornando só mais um ferramenteiro. Você primeiro precisa entender os processos e como as coisas funcionam.

O QUE É BUSINESS INTELLIGENCE?

Para muita gente vai parecer meio óbvio, para outras nem tão óbvio assim, e para boa parte das pessoas, Business Intelligence é só um gerador de relatórios.

E para que você fique preparado para os próximos capítulos, você precisa entender (de verdade) o que é BI.

Eu defino Business Intelligence como:

**BI É UM CONJUNTO DE TÉCNICAS E
PROCESSOS, NÃO UMA
FERRAMENTA**

Durante muito tempo as pessoas passaram a entender BI de uma forma um pouco diferente do que ele realmente é. Se você analisar o nome Business Intelligence, o que significa? Inteligência de Negócio.

E é para isso que o BI serve, para ajudar a tomar decisões inteligentes com base em dados, não em achismo ou Mãe Diná. Indiscutivelmente, BI é um conjunto de técnicas e processos e não uma

ferramenta. E esse conjunto de técnicas e processos é uma coisa que você precisa urgentemente dominar.

A boa notícia é que é exatamente isso que você vai aprender neste livro.

Quando eu estava começando em BI, eu só queria saber de aprender diversos tipos de ferramentas.

E isso não está errado, mas se você não dominar os processos essenciais, duas coisas vão acontecer:

- você não vai conseguir desenvolver um projeto de BI de ponta a ponta;
- você vai ter muita dor de cabeça, porque vai achar que BI é só uma ferramentinha de gerar relatório.

O que eu vou te mostrar aqui é como você cria projetos de ponta a ponta e evita a dor de cabeça.

E a primeira coisa que você precisa fazer é entender as técnicas e os processos. Depois que você entender isso, não importa mais a ferramenta que venha. Todas as ferramentas hoje no mercado fazem quase a mesma coisa, umas melhores, umas mais fáceis, umas pagas e outras gratuitas. Mas todas implementam BI, e você vai conseguir usar qualquer uma delas se entender o conceito.

Você provavelmente já está acostumado a trabalhar com sistemas transacionais, que são os sistemas que fazem toda a entrada de dados, como cadastrar um cliente ou uma venda, ele também faz toda a saída e atualização de dados. A arquitetura do sistema é projetada para fazer essas transações, para inserir, remover e atualizar dados, ou seja, ela é basicamente para entrada e saída rápida de dados.

O BI não, o BI é projetado para consulta de dados em alta performance de uma grande variedade de fontes de dados e sem que isso demore.

BI é um processo muito mais complexo do que a gente imagina que ele é, ele tem o poder de fechar uma empresa, mandar pessoas embora, aumentar faturamento e garantir uma tomada de decisão mais certeira.

No momento que você entender o que é esse conceito, você vai estar apto a criar, construir e desenvolver qualquer solução de BI, independente se quiser fazer para a empresa que você trabalha, começar uma nova carreira dentro de BI ou vender projetos.

Mas aí alguém vai te perguntar:

“Certo, mas como que eu gero um relatório sem ter uma ferramenta?”

“Como que eu faço uma integração sem ter a ferramenta?”

Ok, a gente entende isso, mas a mensagem principal aqui é:

O ETL, o KPI, a tomada de decisão com base no indicador XYZ, sempre vão ser os mesmos, não importa a ferramenta que você estiver utilizando. Até porque se for pensar dessa forma, você pode fazer BI com uma planilha de Excel, porque ele nada mais é do que um cubo se você usar a planilha dinâmica. Mas antes você vai precisar saber o que é um cubo.

Além disso, muitas vezes a escolha da ferramenta nem está relacionada com o que ela vai ajudar no projeto, mas sim com alguma parceria comercial.

Deixa eu te contar uma história...

No projeto de BI do SBT, eles usaram por muito tempo um projeto piloto em Qlikview. Mas para o que eles queriam fazer, o Qlikview começou a travar e ficar lento.

E isso quer dizer que o Qlikview é ruim? Não. Quer dizer que cada ferramenta tem uma melhor aplicabilidade do processo. E no caso deles, o Qlikview não era a escolha certa.

Eles queriam utilizar as técnicas de balanced scorecard e KPI, vendo isso em tempo real. Quem conhece essa técnica, entenderia o que eles estavam pedindo, mas a pessoa que começou esse projeto era um ferramenteiro, ele entendia de Qlikview, não de BI. E o que estavam fazendo, eram métricas com gráficos, não KPIs.

Por não entender os fundamentos do BI, escolheram uma ferramenta que não dava conta, porque ela não foi projetada para aquele tipo de situação. Naquele momento, ninguém fez uma análise para entender se realmente aquela ferramenta atingiria os objetivos.

Depois, como o SBT já trabalhava com banco de dados Oracle, a Oracle fez uma promoção para o SBT entrar no mundo de BI com eles.

Quando eu cheguei no projeto, implementei o OBIEE (Oracle Business Intelligence Enterprise Edition), utilizei as documentações que já tinham sido feitas e então seguimos o desenvolvimento.

Qual o objetivo dessa história toda?

**BI É UM CONJUNTO DE TÉCNICAS E
PROCESSOS, NÃO FERRAMENTA**

O cara lá sabia usar o Qlikview, só não sabia os fundamentos de KPI, e por isso colocou todo o projeto em risco.

BI é toda a técnica e processo para a coleta, organização, análise, compartilhamento e monitoramento dos dados para suportar a tomada de decisão.

E como todo bom processo, ele segue uma sequência de passos, que são:

- passo 1: ele coleta os dados de uma fonte de dados, como um banco de dados legado, uma planilha, e outras fontes de dados;
- passo 2: depois organiza todos esses dados coletados em um lugar, que nesse caso nós chamamos de Data Warehouse;
- passo 3: depois disso, a gente pode analisar e compartilhar esses dados;
- passo 4: e por fim se toma uma decisão com base nos dados que foram coletados, organizados e analisados;
- passo 5: depois a gente monitora todo esse processo para entender se a nossa tomada de decisão foi coerente ou se piorou a situação.

Atualmente, o interesse pelo BI vem crescendo à medida que o seu emprego possibilita às empresas realizar uma série de análises e projeções, de forma que a gente possa agilizar o processo da tomada de decisão.

Vou te dar um exemplo...

Uma vez eu fiz um trabalho para um cliente que não posso divulgar o nome, porque assinei um termo de confidencialidade para esse

projeto. Nele, eu precisava identificar os 20% das lojas que equivaliam a 80% do faturamento da empresa.

Nós fizemos essa análise utilizando as técnicas e processos de BI, nós aplicamos o princípio de Pareto para que ele pudesse tomar essa decisão.

A gente usou as ferramentas de BI disponíveis no mercado, que nesse caso foi o OBIEE. Então fizemos algumas análises e chegamos à conclusão de que tinha uma loja que na verdade estava dando prejuízo para o resto do grupo inteiro.

Depois, a gente fez uma outra análise em cima dos dados já coletados para entender o seguinte: qual vai ser o impacto se essa loja fechar?

E o resultado foi o seguinte: a empresa ia aumentar a margem de lucro dela se aquela loja fechasse. Isso foi apresentado para a diretoria, e com base nesses dados que eu fiz lá no BI, o diretor tomou a decisão: “pode fechar.”

Naquele momento, por volta de 500 pessoas foram mandadas embora, porque a loja simplesmente fechou.

Você deve estar pensando: “mas aí aumentou o desemprego.”

Lembra, a nossa missão aqui é suportar a tomada de decisão. Quem aperta o botão para lançar o foguete é o Presidente, mas o foguete é você quem faz, e deixa pronto para destruir ou para salvar, dependendo do lado que você estiver.

Nós fornecemos todas as análises necessárias e o diretor, com base na expertise do negócio, cruzando com várias outras análises, como de marketing, de projeção do futuro da companhia e várias outras coisas, ele toma a decisão.

POR QUE BI?

Para explicar essa parte, quero usar como referência um livro: A Arte da Guerra, de Sun Tzu.

E qual o motivo de eu indicar esse livro para você? Apesar do livro ser um clichê nos cursos de administração, áreas de negócios e afins, se você der uma chance e tentar interpretar o que ele quer dizer, vai ver que a gente pode implementar essas dicas do Sun Tzu em BI.

Sun Tzu foi um general chinês, um grande estrategista. Ganhou diversas batalhas, diversas guerras e tem muita coisa que trazemos até hoje dos seus ensinamentos.

Eu tirei um ensinamento desse livro, e acredito que este trecho representa bem o que eu quero mostrar para você:

“Conheça a si mesmo e ao inimigo e, em cem batalhas, você nunca correrá perigo.

Conheça a si mesmo, mas desconheça seu inimigo, e suas chances de ganhar e perder são iguais.

Desconheça a si mesmo e ao inimigo e você sempre correrá perigo.”

Naquela época ele estava falando sobre guerra, mas hoje a nova guerra é a das empresas tentando conquistar os clientes. O inimigo aqui nesse caso são seus concorrentes, ou os concorrentes do seu cliente. E fazendo uma tradução disso para o nosso mundo de BI, vamos fazer isso analisando os dados. É dessa forma que você consegue entender no que é bom e no que não é: analisando seus dados internos. Os dados, por exemplo, de ERP, CRM, frente de caixa, independentemente de onde ele esteja, o importante é extrair as informações e entender suas forças e fraquezas.

A falta desses conhecimentos hoje em dia pode resultar na derrota. Se você não conhecer seu concorrente, ou o concorrente do seu cliente, a empresa ainda continua viva, ela não ganha a guerra, mas não perde a batalha.

Então você precisa ter conhecimento dos seus dados, do que pode gerar e do que não pode gerar com eles, com ponto de vista de produto, vendas, economia, equipe, etc.

Essa é reflexão que eu quero deixar para você.

Também quero deixar uma dica sobre o livro: leia um trecho de cada vez. Tente entender o que ele está dizendo e como você pode trazer isso para o nosso mundo moderno, como podemos aplicar isso com as nossas técnicas e estratégias em cima da análise de dados.

Esse é um pensamento mais aprofundado de porque nós usamos o BI. Por isso eu estou trazendo para você um dos maiores generais de guerra, que já aplicava as técnicas de BI lá em 500 a.C.

E também temos um cara chamado Piton, e ele já dizia:

“Já se foi o tempo em que se tomava decisões no achismo ou feeling.”

Ninguém mais toma decisão porque acordou feliz ou triste. Hoje, em qualquer empresa, as decisões são tomadas com base em dados e evidências. Nós produzimos uma massa de dados gigante, e esses dados são uma mina de ouro, eles precisam ser lapidados.

Se você tomar decisão em cima de 10% desses dados, vai tomar uma decisão muito melhor do que usando técnicas não científicas.

Eu estou explicando isso para você aplicar na sua empresa ou para você aplicar no negócio das outras pessoas, como consultor ou analista.

“Seu concorrente quer destroçar você! ”

Quando você está fazendo um projeto de BI para uma empresa, você tem que entender que o concorrente quer destroçar ela. O concorrente não quer que a empresa fique mais no mapa. Às vezes nem é por questões comerciais, mas porque ele não quer nenhum tipo de concorrente. Você como consultor ou analista de BI, tem a capacidade de ajudar a evitar esse tipo de problema.

Para resumir essa parte, quero que você entenda uma coisa:

“Ou você toma uma decisão de forma rápida, objetiva e certeira, ou... caixão e vela preta para você.”

Quero que você se coloque no lugar do tomador de decisão. Se você quer trabalhar com BI, você precisa entender que para tomar uma decisão rápida, objetiva e certeira é necessário o BI.

BI NÃO É LUXO, BI É SOBREVIVÊNCIA

Antigamente, o mundo inteiro fazia a gestão de empresas em bloquinho e máquina de escrever. Depois vieram os sistemas ERPs para gerenciar os processos da empresa. Hoje, não tem mais como uma empresa andar sem o BI e continuar viva. E quando eu digo BI, falo das técnicas e processos que ele implementa.

BI é obrigação em todas as empresas. Isso é, se essa empresa quiser se manter viva no mercado. Então ainda tem muito mercado pela frente para você explorar e muita oportunidade profissional para você.

A sua missão como profissional de BI é poder ajudar esses empreendedores, empresários e todos os tomadores de decisão em geral.

QUE TIPO DE PERGUNTA O BI RESPONDE?

Antes de a gente começar a falar em modelar o Data Warehouse, é importante que você entenda que as técnicas e processos de BI focam no suporte a tomada de decisão, e é para chegar nessa tomada de decisão que você precisa de um Data Warehouse, para reunir e organizar todos os dados que antes estavam espalhados pela empresa.

O BI utiliza métricas e KPIs para medir o desempenho passado e poder orientar o planejamento futuro. Ou seja, olhamos o que já aconteceu, que pode ser um dia atrás, uma semana, no mês passado, no ano passado, 10 anos ou um minuto atrás.

E existem 3 perguntas que o BI vai responder para suportar a tomada de decisão

#1: O que aconteceu?

Por exemplo:

- aconteceu uma venda;
- um funcionário foi demitido;
- novos itens chegaram no estoque;
- as contas foram pagas.

#2: Quantas vezes isso aconteceu?

Já sabemos o que aconteceu. Agora, quantas vezes isso aconteceu?

Por exemplo:

Aconteceu uma venda. E quantas vezes isso aconteceu?

Se foram feitas 30 vendas daquele produto, essa venda aconteceu 30 vezes.

#3: Com que frequência isso acontece ou aconteceu?

Aqui tem relação com o tempo, com a frequência que aquele evento acontece.

Por exemplo:

O que aconteceu? Contratação de colaborador.

Quantas vezes isso aconteceu? Uma vez.

Com que frequência? Em janeiro.

Você obviamente não tem como saber tudo que vai acontecer no futuro, mas o BI ajuda a gente a diminuir o erro.

Tenho um exemplo para você:

Lembra aquela vez que a seleção brasileira estava jogando com a Alemanha na Copa de 2014?

Se o técnico tivesse um BI, ele ia conseguir ver que 2 gols a cada 5 minutos não era normal, e naquele momento já poderia ter identificado que tem algo errado.

É aí que entra a magia do BI. Se a gente seguir essas exatas 3 perguntas, poderia fazer uma análise dos jogos anteriores.

O que aconteceu? O brasil tomou gol nos últimos 30 jogos.

Quantas vezes isso aconteceu nos últimos 30 jogos? 7 vezes.

E com que frequência esses gols acontecem? Geralmente o Brasil toma gol no primeiro tempo.

No jogo da Alemanha, o que teria acontecido?

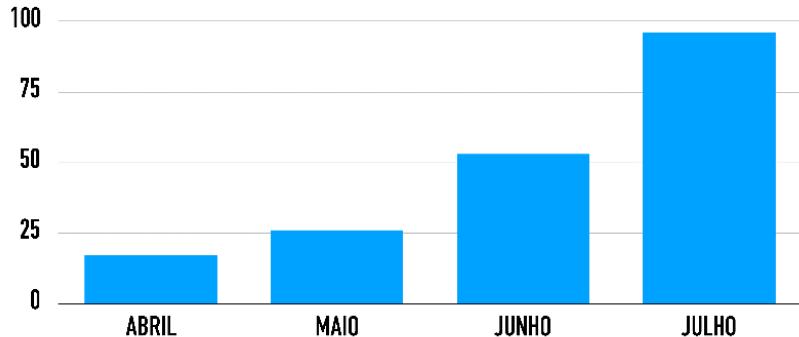
A Alemanha fez um gol nos primeiros 5 minutos. Está dentro do padrão...

Mas vamos supor que isso aconteça 3 vezes. Nesse momento, se a comissão técnica do Brasil tivesse uma análise em cima disso, facilmente veria que saiu da curva dos grandes números. Seguindo a lei dos grandes números, a gente sempre vai chegar mais ou menos naquilo que já estamos acostumados, ou seja, se o time faz 2 gols por partida, não espere que ele vai fazer 7 gols em um dia. Acontece? Sim, mas é exceção à regra.

Eu trouxe aqui um gráfico de vendas de Produtos por Região para exemplificar um pouco melhor:

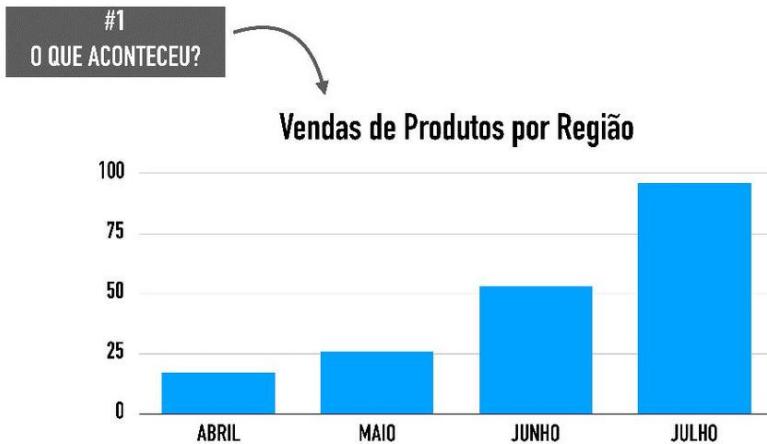
É um gráfico de barras no tempo.

Vendas de Produtos por Região



Nele, a gente consegue identificar a respostas para as 3 perguntas.

#1: O que aconteceu?



Nesse exemplo aqui, aconteceu venda de produtos em uma determinada região. Vamos considerar que o produto seja café. E que a região seja São Paulo.

#2: Quantas vezes isso aconteceu?



Aqui a gente pode ver pela altura das barras, que em abril aconteceram 17 vendas, em maio aconteceram foram 26, 53 em junho e 96 em julho.

#3: Com que frequência isso aconteceu?



Nesse exemplo, a frequência é mês a mês. Podia ser ano a ano, semana a semana, hora a hora, de 5 em 5 minutos.

Agora você sabe que em abril, aconteceram 17 vendas de café em São Paulo.

É dessa forma que a gente analisa o passado, e o importante é você entender essas técnicas. No momento que você entende os processos, a ferramenta é somente a furadeira que vai fazer o buraco na parede, e não o buraco em si.

Lembra sempre disto:

As ferramentas de BI servem para nos auxiliar em uma determinada tarefa.

Por exemplo:

Se eu tenho que fazer um buraco na parede, eu posso:

- usar uma furadeira;
- abrir com uma martelada;
- abrir com uma faquinha que nem presidiário.

Todas são ferramentas para atingir um determinado objetivo. No BI, esse objetivo é responder essas 3 perguntas. Se você entendeu isso, eu fico muito satisfeito, quase que digo para você: missão cumprida.

Essas são as perguntas básicas que você tem que responder, e aqui eu estou mostrando um exemplo bem prático do mercado, que é como você vai projetar um gráfico.

Com essas 3 informações, você tem uma arma muito poderosa para tomar a decisão. E muitas vezes, nem os key users sabem dessas perguntas, então se você mostrar isso para uma pessoa de negócio, vai ganhar o respeito e admiração do time de negócio. Você vai ser um profissional muito mais completo entendendo esses 3 pontos base.

AS ETAPAS PARA TOMADA DE DECISÃO

Por que diabos você precisa entender essas 4 etapas para construir um Data Warehouse? Porque você só vai conseguir criar soluções profissionais no momento que entender onde se encaixam essas etapas.

Não posso entrar na parte mais técnica com você sem eu ter certeza de que você sabe do que estou falando.

Você já deve ter escutado sobre:

- dados;
- informação;
- conhecimento;
- sabedoria.

Esse é o ciclo de vida da tomada de decisão, ou ciclo de vida do conhecimento.

Dados



Os dados são uma parte pequena da informação, que sozinhos não fazem sentido.

São os ativos mais importantes de qualquer organização. Na Raizzer, a minha empresa, eu considero que os ativos mais importantes são as pessoas e os dados. Nós podemos perder tudo que temos, se tivermos o nosso time e os nossos dados, conseguimos reconstruir tudo.

Tem até uma frase muito utilizada, que diz que os dados são o novo petróleo. Pode até ser, mas se a gente não souber e o que fazer com eles, não servem para nada.

Informação



Com base nesses dados, nós geramos a informação. São dados agrupados, organizados e lapidados. E é aqui que seu Data Warehouse vai entrar. Nele você vai fazer esse agrupamento e organização dos dados.

Há muitos dados pelo mundo para gerar informação, é aqui que entra o conceito de Big Data, dados estruturados e não estruturados. Os dados são diversos, e para gerar informação, você precisa agrupar, organizar e limpar eles, deixar redondinho.

Conhecimento



Interpretando e aprendendo com as informações geradas, você vai gerar novos conhecimentos.

Depois, você combina seus conhecimentos atuais com os novos, para gerar um novo leque de conhecimento.

Por exemplo:

Antes eu sabia ler e escrever. Agora eu sei ler, escrever e fazer conta de matemática. Meu conhecimento está maior.

E com base nesse leque de conhecimentos, vem a parte da sabedoria.

Sabedoria



Você toma decisões com base no seu leque de conhecimentos. Você tem um determinado problema, e tem esse tipo de conhecimento, e com ele você toma a decisão que achar adequada no momento.

Então vem a sabedoria, ou seja, o momento da tomada de decisão. Com base no conhecimento que eu gerei, eu tenho a sabedoria de tomar uma decisão.

Se eu e você tivermos o mesmo problema, nós vamos tomar decisões diferentes se esse ciclo da tomada de decisão for diferente para você e para mim. No momento que você organiza as suas informações, você pode entender e interpretar elas de um jeito, enquanto eu posso entender e interpretar de outro jeito.

Fiz um exemplo para explicar isso:

DADOS	INFORMAÇÃO	CONHECIMENTO	SABEDORIA
40°			
90°			
20°			

Aqui eu tenho 3 dados:

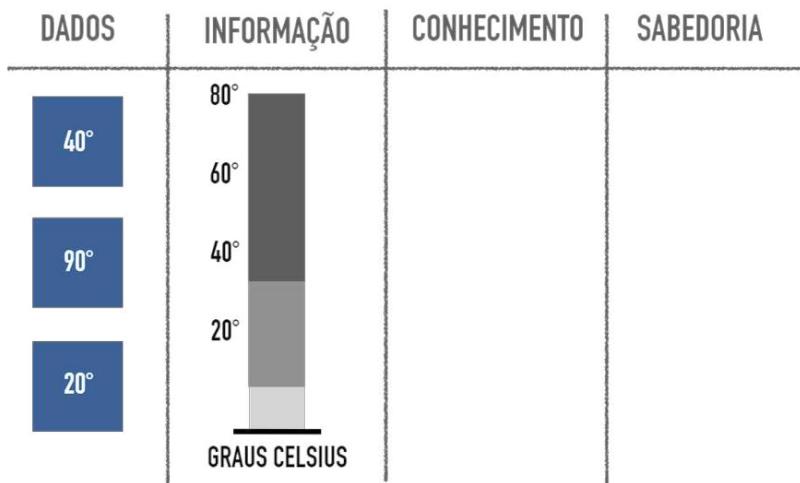
- 40°;
- 90°;
- 20°.

O que é isso para você? Pode ser qualquer coisa que trabalhe com graus. Por exemplo:

- coordenadas geográficas;
- temperatura;
- unidade de medida ótica;
- ângulo;
- etc.

De repente nem é grau, é só um número com uma bolinha do lado, nós que estamos interpretando assim. Esses dados estão soltos aqui, aleatórios.

Agora, se eu fizer um trabalho para gerar uma informação em cima deles, podemos entender o que eles querem dizer.



Nesse exemplo aqui é um termômetro, são graus Celsius. Agora a gente consegue entender, eu gerei a informação para você.

Agora imagina que eu falo para você: “hoje está 40°.”

Você vai pensar que está quente demais para sair de casa. E como que você sabe que está quente demais? Para uma pessoa que vive no Rio de Janeiro, 40° é uma temperatura normal.

Percebe que a diferença da interpretação dessas informações depende do seu conhecimento? Depende de como que você interpretou essa informação.

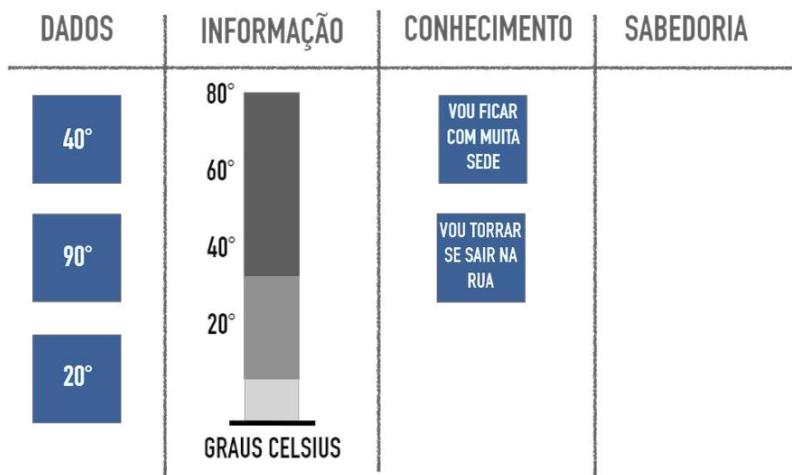
Então a próxima etapa desse ciclo da tomada de decisão é gerar o conhecimento.

Eu tenho os conhecimentos que eu já adquiri ao longo da vida, e tenho os novos conhecimentos que eu vou adquirir com base nas informações que me foram fornecidas.

Hoje está 40°C. Qual o conhecimento que eu já tenho sobre isso aqui?

Como eu sou gaúcho, 40°C definitivamente não é nada normal para mim. Então eu já tenho o conhecimento de que:

- eu vou ficar com muita sede;
- eu vou torrar se sair na rua.

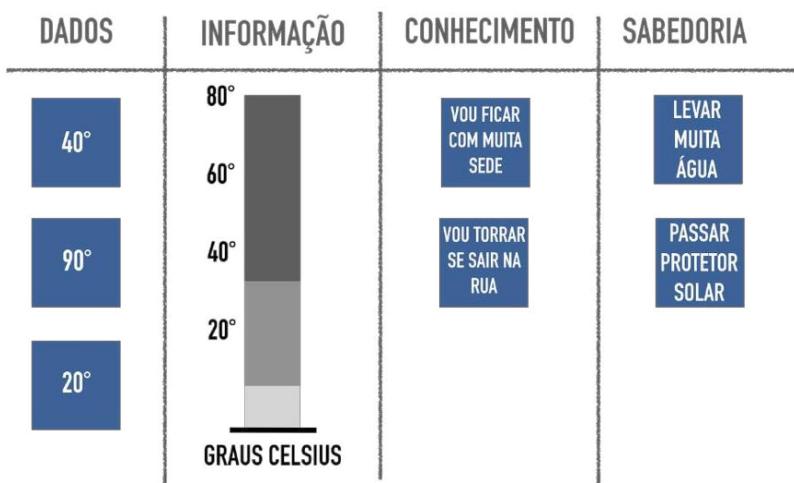


Essa é a grande sacada de ter o conhecimento. E nós, como profissionais de BI, temos que entender isso para que a gente possa ajudar quem vai tomar a decisão a saber o que fazer.

A gente ajuda o tomador de decisão fornecendo as informações. Nós pegamos os dados dos sistemas de origem, geramos essas informações e depois geramos o conhecimento.

Depois disso vem a sabedoria, que é a tomada de decisão. Muita gente para na parte do conhecimento porque acredita que o conhecimento é tomada de decisão, mas o conhecimento nada mais é que: você sabe de alguma coisa.

Eu tenho que saber o que eu vou fazer com aquele conhecimento, e isso é a sabedoria.



Por exemplo:

Se eu sei que eu vou ficar com muita sede, qual é a tomada de decisão? Vou levar muita água.

Se eu sei que vou me queimar, qual é a tomada de decisão? Vou passar um protetor solar.

Esse processo que nós seguimos para a tomada de decisão é o mesmo no BI, no Data Warehouse, no Analytics, no Data Science, em qualquer coisa, porque é como nós, seres humanos, tomamos decisões.

Você não consegue tomar uma decisão inteligente com o dado puro “40°” e você não consegue tomar uma decisão só olhando para o termômetro.

O que eu vou te mostrar nos próximos capítulos é como a gente vai aplicar esse processo, essa técnica para tomada de decisão na prática.

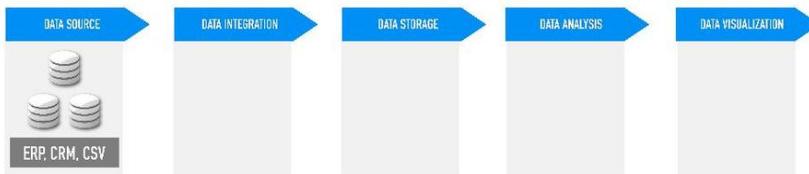
ARQUITETURA DE BUSINESS INTELLIGENCE

Agora que você já sabe o que é o BI, todo o contexto em que ele é aplicado e como nós podemos auxiliar gestores na tomada de decisão, posso entrar na parte técnica do conteúdo com a certeza de que você está entendendo tudo que estou falando.

E para começar, vou te explicar a arquitetura de Business Intelligence, como funciona esse processo de ponta a ponta e tudo que acontece em uma solução de BI.

Eu quero que primeiro você entenda as 4 etapas do processo de BI. Depois, vou mostrar as possíveis ferramentas para implementar cada uma delas. E então vamos nos aprofundar no Data Warehouse e no ETL, que você vai aprender a implementar neste livro.

DATA SOURCE



Embora o Data Source não seja uma das etapas de uma solução de BI, gosto de representar ele como uma torre também porque é de onde os dados vão sair, é a origem deles.

Geralmente, o processo de BI começa no Data Source. Aqui é onde nós temos os dados que não viraram informação ainda. E eu falo *geralmente* de uma forma meio irônica, porque às vezes, tem pessoas que não sabem como funciona a arquitetura e acham que ligar uma ferramenta direto no Data Source e gerar relatório é BI.

Indiscutivelmente, na minha opinião, isso não é BI. Isso não tem nada a ver com BI. Isso é um gerador de relatório; e BI é muito mais que isso.

Então, o que nós temos no Data Source? Temos o ERP, CRM, arquivos CSV, arquivos TXT, etc. Importante: o BI trabalha com dados estruturados e semiestruturados. Ou seja, bancos de dados relacionais, algo que tenha uma estrutura implícita para a organização dos dados.

Ele trabalha 100% dos dados estruturados e alguma coisa de dados semiestruturados. E o que isso quer dizer?

Por exemplo:

O dado estruturado está em um banco de dados, geralmente relacional. Lá tem colunas, linhas, tabelas, relacionamento, integridade referencial. Ele está modelado na terceira forma normal, as tabelas se correlacionam para não ter duplicidade, e várias regras assim para garantir a integridade dos dados.

Já o semiestruturado é um e-mail, por exemplo. O que temos no e-mail?

- assunto;

- remetente;
- quem viu;
- quem recebeu;
- e o corpo do e-mail, que não está estruturado.

Isso é o semiestruturado.

Um CSV também uma fonte de dados semiestruturada. No CSV você pode colocar linhas e usar ponto e vírgula para representar a colunas. Aí você vai ter, por exemplo: “nome, e-mail, CPF”. Depois, vai colocar os dados: “Rafael, rafael@bol.com, 010...”.

Nesse momento, os dados são semiestruturados porque ele não garante toda a integridade, mas está um pouco estruturado.

Quando você vai falar no conceito de Big Data, são dados não estruturados, geralmente. É lógico, o Big Data consegue trabalhar com dados estruturados também, mas o grande foco são dados não estruturados. Por exemplo, metadados de um vídeo, imagem ou até comentários nas redes sociais.

DATA INTEGRATION



Depois de você ter identificado o Data Source, a primeira fase na arquitetura de BI é o Data Integration, que é onde ficam os robozinhos, é a parte do ETL ou EL-T.

Nessa etapa, é onde nós transformamos os dados em informação.

Aqui eu vou te explicar como funciona o ETL:

- E = extract (extrair);
- T = transform (transformar);
- L = load (carregar).

Extract

O "E" vai extrair os dados de todos os Data Sources e colocar eles na staging area, que eu vou explicar mais para frente o que é e como você cria uma.

Transform

Depois que os dados de todos os bancos foram tirados de lá e eu tenho todos juntos da staging area, o "T" vai unificar os dados, limpar eles e deixar redondinho.

Por exemplo:

Esses são os dados que eu tenho nos Data Sources:

ERP:

ID	Nome do Cliente	Sexo do Cliente
143	Rafael Piton	m

Sistema Financeiro:

ID	Nome do Cliente	Cidade do Cliente
1435	Rafael Piton	Porto Alegre

CRM:

ID	Nome do Cliente	Sexo do Cliente
343	Rafael Piton	masculino

Esse é um exemplo bem simples para você entender o que eu estou querendo dizer.

O que a transformação vai fazer aqui é pegar todas as informações do mesmo cliente:

As informações que temos sobre o cliente Rafael Piton nesses 3 sistemas é:

- sexo: m;
- cidade: Porto Alegre;
- sexo: masculino.

Depois, ele vai tirar as sujeiras e unificar tudo em uma informação só. Aqui ele tem duas informações sobre "sexo", o que ele vai fazer?

Rafael Piton:

- sexo: masculino;
- cidade: Porto Alegre.

Nesse momento, a gente está unificando os dados.

Nessa etapa você já pode aplicar alguma regra simples de negócio, quando for algo que vai servir para o projeto como o todo, para sempre, aí não tem problema de isso ficar no ETL.

Load

O "L" vai fazer a carga efetivamente. Então você extraiu de algum lugar, fez a transformação, e depois vai pegar esses carinhos todos e vai carregar. Ou seja, vai inserir no Data Warehouse.

Esse processo do ETL vai acontecer com uma periodicidade, por exemplo, uma vez por dia, todo o dia a meia noite ele vai lá, pega os dados, faz a transformação e coloca no Data Warehouse. A periodicidade vai depender da necessidade do negócio.

Geralmente a gente faz todo o dia a meia noite porque é um processamento bem pesado de dados, mas nada impede de você dividir isso em pequenas cargas. A gente vai falar mais de carga na parte de Data Integration.

Existem diversas ferramentas que você pode utilizar para implementar isso, como Pentaho Data Integration, Power Center, Talend, SSIS (SQL Server Integration Services) e ODI (Oracle Data Integrator).

DATA WAREHOUSE



Nessa torre do Data Storage, a gente trabalha muito com Data Warehouse, mas em alguns projetos você vai ouvir falar de Data Mart.

Então aqui a gente tem o Data Warehouse, que é um banco de dados relacional modelado dimensionalmente. Diferente do Data Source, que tinha os dados modelados de forma transacional e normalizados, naquelas várias regras onde não pode haver redundância de dados.

Uma das formas de implementar a modelagem dimensional é com modelo Star Schema, concebido pelo Ralph Kimball, que é em quem eu baseio a minha experiência em Data Warehouse. Nesse modelo, os dados ficam desnormalizados, ou seja, há redundância nos dados, eles são duplicados, dependendo do formato que a gente vai trabalhar. Mas a consulta fica extremamente veloz e a gente consegue tirar dele toneladas de dados em um segundo. Esse é o objetivo.

O ETL extrai os dados do Data Source, faz uma transformação e depois carrega no Data Warehouse. O Data Warehouse é o repositório de dados centrais, onde fica a informação.

Para a criação do Data Warehouse, nós podemos usar bancos de dados relacionais, como PostgreSQL, Oracle Database, SQL Server e Teradata.

DATA ANALYSIS



Nessa etapa é onde nós vamos gerar o conhecimento, onde começamos a cruzar as informações disponibilizadas e começamos a analisar e identificar padrões passados e problemas futuros.

Aqui é onde vai entrar a parte de análise do negócio. E tem algumas técnicas que você vai utilizar para fazer isso.

Métricas

A métrica é uma medida, ela serve para você metrificar algo, então não é nada incomum você tentar relacionar a uma fita métrica, daquelas de pedreiro, que começa a puxar e medir.

Métrica é também chamada de quantificador ou medida. Elas são utilizadas para metrificar algo e sempre são números, porque precisam ser contáveis. Esses números são provenientes de transações da empresa.

Tudo que a empresa for mensurar é uma métrica. Na maioria das vezes, a métrica vai ser o que o usuário quer medir.

Pergunta para o usuário o que ele quer medir, que informação quer ver na tela?

Normalmente vão responder: quero ver as vendas, os seguidores no Twitter, o fluxo de caixa etc....

Você identifica muito fácil quais são as métricas só com essas perguntas.

Por exemplo:

Hoje eu tomei 5 cafés.

Métrica: quantidade de cafés bebidos (5 nesse caso).

Dá para fazer uma análise com ela, por exemplo, no tempo: nos últimos 30 dias, que dias eu tomei mais café?

Alguns exemplos de métricas são:

- valor de vendas;
- quantidade de vendas;
- quantidade de colaboradores;
- valor de contas a pagar.

Key Performance Indicator (KPI)

Também conhecido como indicador chave de desempenho, os famosos indicadores.

E o indicador é um índice que a gente usa para mensurar percentualmente as variações que ocorrem na empresa, no produto, no setor, etc. Geralmente, um KPI tem uma meta.

Por exemplo:

Nossa meta de faturamento este ano é R\$200.000, e estamos em R\$100.000 de faturamento, então o KPI está em 50% do batimento da meta, é assim que se analisa.

Ele costuma utilizar operações matemáticas, por exemplo:

KPI de Turnover (Rotatividade de Colaboradores)

Objetivo: Monitorar o giro de entradas e saídas de pessoal na empresa. Ou seja, todo funcionário que entrou e que saiu.

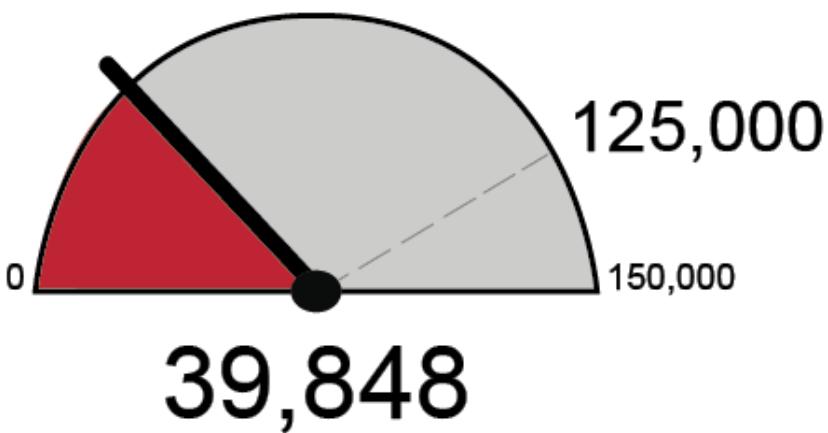
Fórmula: (nº de desligamentos x 100) / média de empregados = % de Turnover.

$$(50 \times 100) / 100 = 50\% \text{ de Turnover}$$

Leitura: 50% dos colaboradores afastaram-se da empresa naquele determinado período.

Se isso é bom ou é ruim, depende da empresa. Eu acho 50% de turnover alto, mas para empresas de telemarketing, pode ser que não seja, ou para uma empresa que não tem sistema de retenção de talentos.

Aqui tem um desenho de como a gente geralmente utiliza gráficos para ver os KPIs.



Geralmente nós utilizamos o gráfico de gauge, que parece um velocímetro, onde nós temos a meta e um ponteirinho que representa o realizado.

O que acontece é que se em um determinado momento aquele ponteirinho deveria estar na linha pontilhada e no momento que a gente está analisando ele não está, ele vai estar vermelho.

Se estivesse no meio, estaria amarelo, e se estivesse no pontilhado estaria verde, porque estamos onde deveríamos para alcançar a meta a tempo.

Geralmente um KPI se comporta dessa forma.

Analysis OLAP (Cubos)

De uma forma bem resumida, Analysis OLAP são os famosos cubos. OLAP significa online analytical processing, ou seja, processamento analítico em tempo real.

Mas para que alguém precisa de uma solução dessas? Com ele, a gente pode combinar dados, pivotear e fazer diversos cruzamentos em tempo de execução, ou seja, com os cubos, a gente pode manipular e analisar um grande volume de dados em tempo real, por isso é online, e o mais importante, com múltiplas perspectivas e hipóteses.

Por exemplo:

Eu tenho uma pizzaria e fiz a venda de um produto. Sei que vendi um produto, mas quando que isso aconteceu? Em janeiro. E que pizza foi essa? Uma pizza de quatro queijos. Mas para que cliente foi? Então você começa a combinar esses dados para a melhor tomada de decisão, e isso acontece tudo em tempo real.

E como isso vai acontecer? O key user vai estar usando o que nós chamamos de OLAP Client, que é onde ele vai poder arrastar a informação que ele quer ver. Esse OLAP Client, vai enviar uma consulta para o OLAP Server, que é o cubo em si. Esse cubo, vai consultar o Data Warehouse e retornar para o OLAP Client a informação que o usuário pediu, e na mesma hora ele já vai carregar na tela o resultado da consulta.



Geralmente essa tecnologia OLAP é projetada para suportar tomadas de decisões dos usuários de negócios, que pode fazer o conceito que nós chamamos de ad hoc.

Por exemplo:

Ele arrasta a coluna do valor da venda. Depois, ele quer ver essas vendas cruzadas com produto, depois com a forma de pagamento, com região, com mês, e com o que mais ele precisar.

O usuário vai montando a análise dele para entender o que está acontecendo com o negócio.

Então como eu falei, essas Analysis OLAP são uma tela em branco para que o business user possa analisar o negócio de várias perspectivas e em tempo real, esse é o grande segredo, a grande cereja de bolo.

O BI na sua essência tem alguma solução que implementa esse conceito de cubos, e uma das tecnologias para fazer isso é o OLAP.

O OLAP vai fornecer para o seu cliente a possibilidade de construir gráficos, filtros, e tudo mais que for necessário para analisar o negócio, inclusive cálculos, e entregar essa informação o mais rápido possível.

Por exemplo:

Você tem o valor vendido e a quantidade vendida salvas no Data Warehouse. Mas não tem o percentual de vendas por um determinado

período, então o usuário de negócio pode calcular isso em tempo de execução quando ele precisar.

Esse é um grande problema que os gestores tinham, sempre que precisavam de uma informação, precisavam pedir para a TI fazer essa análise, e depois que tinham o resultado, se precisassem de mais alguma informação, precisava pedir de novo para a TI uma nova análise. E isso demorava muito, mas hoje a gente não pode demorar com esse tipo de coisa, tomada de decisão às vezes é feita em 15 minutos e às vezes é no próprio minuto, então esse tipo de solução veio para resolver esse problema. Com esse tipo de solução a gente consegue identificar novos gaps, novos problemas, novos produtos e padrões do passado.

Além disso, os cubos permitem uma coisa bem interessante que nós chamamos de drill down, drill up ou drill through, que é perfurar os dados.

Imagina que eu vendi R\$200.000 no mês de dezembro no Brasil. Estou vendo isso em um gráfico de pizza, ele me mostra que 85% dessas vendas aconteceram no Rio Grande do Sul e as outras em São Paulo. Se você quiser mais informações sobre essas vendas, qual a coisa mais intuitiva do ser humano? Clicar ali para saber o que tem dentro, e isso é o que nós chamamos de drill.

Então no momento que eu clico naquela parte da pizza de 85%, ele vai fazer o drill e vai me mostrar que daquelas vendas, na verdade 70% vem da cidade de Porto Alegre e que o resto é das outras cidades da grande Porto Alegre.

Esse tipo de análise é muito interessante porque se a pessoa olhar no ponto de vista do estado só, ele vai pensar que o Rio Grande do Sul está vendendo muito, e vai pensar em abrir filial em tudo quanto é

canto do estado, mas na verdade o que está vendendo é só Porto Alegre e um percentual menor nas outras cidades.

Então com esse tipo de solução, o usuário de negócio consegue explorar os dados sozinho e rápido, sem depender de setor de TI ou ter que esperar meses para tomar uma decisão.

Tudo isso que eu mostrei aqui para você são conceitos que se você entender, não interessa a ferramenta em que você vai implementar, porque elas se baseiam todas em cima de um conceito.

Algumas ferramentas que você pode utilizar para criar os cubos são Qlikview, Microsoft BI, Microstrategy, OBIEE, Pentaho Mondrian etc

DATA VISUALIZATION



Depois de ter feito as análises, é hora de preparar a visualização delas. Como essa informação vai ser entregue para o tomador de decisão?

Existem algumas técnicas que você pode utilizar para fazer isso.

Reporting

Uma delas são os relatórios de BI. Quase todas as ferramentas de BI implementam esses relatórios. E qual a grande diferença deles para os demais relatórios?

As técnicas de BI reporting conseguem transformar os dados em informações comprehensíveis, flexíveis e fáceis de serem analisadas pelo usuário final em relatórios pré-formatados.

E aqui eu já te dou uma dica: o que é fácil para você, não necessariamente vai ser fácil para o tomador de decisão. Você está estudando para ser especialista de BI, ~~um analista de negócios~~, então a sua obrigação é saber tornar o relatório fácil para o cliente analisar.

Como eu já falei no início do livro, muitas vezes as pessoas acham que o BI é somente um gerador de relatório. Digamos que poder fazer um relatório seria o primeiro passo de um BI.

Você pode conectar direto no Data Warehouse e criar um relatório em PDF, PowerPoint e diversos formatos. Esse é o grande objetivo dele, os reports podem ser entregues automaticamente para o usuário final em qualquer tipo de formato.

Outra coisa interessante é que os dados do relatório são carregados automaticamente dentro da solução, ou seja, acabou aquele tempo que a requisição era enviada para a TI fazer aquelas consultas gigantes. Aqui nós já desenhamos o relatório, por isso que ele é pré-formatado, porque desenhamos junto com as pessoas que vão utilizar.

Tenho aqui um exemplo de relatório:

STEELWHEELS

500 International Speedway, Daytona Beach FL 32114
(123) 456-7890 <http://www.steelwheels.com>
Qua Nov 27 17:26:56 GMT 2013

TO:	Australian Gift Network, Co 31 Duncan St. West End, South Brisbane, Queensland 4101 Australia	INVOICE
Attn: Tony Calaghan		Invoice #: 10152
Sales Rep: 1611		Account Number: 333
Terms: Net 30 days		Date: Setembro 25, 2003
SKU	Product Description	Price/Unit Qty Ordered Total Price
S18_4027	1970 Triumph Spitfire	\$4.524,10 35 \$163,45
S32_3207	1950's Chicago Surface Lines Streetcar	\$1.681,35 33 \$56,77
S24_4048	1992 Porsche Cayenne Turbo Silver	\$2.802,09 23 \$64,64
S24_1444	1970 Dodge Coronet	\$1.632,75 25 \$40,82
		\$10.640,29

Payment History		
Date	Check#	Amount
11-15-03	HL209210	\$ 27.098,80
10-17-03	JL479662	\$ 10.640,29
03-01-05	NF959653	\$ 21.730,03

Send Payment and Remittance Slip to:
Steel Wheels
500 International Speedway
Daytona Beach FL 32114

Thank you for your business!

Fonte: Pentaho.

É um relatório bem simples, mas perceba que ele está em um formato para ser impresso. Então não necessariamente você precisa ter gráficos e coisas que pulam na tela.

Também trouxe alguns cases com reporting de projetos que eu já participei para você:

Eu peguei um projeto com o BI Publisher da Oracle, que foi um relatório todo automatizado e pré-formatado para ser impresso em uma folha de ofício A4, com todos os campos preenchidos e tudo mais. Ou seja, nós cortamos a etapa da menina que fazia a impressão desse relatório todos os dias e deixamos ele automatizado para ser impresso todo dia às 8h.

O scheduler do Oracle BI Publisher já consultava as informações nos bancos, organizava, exportava em PDF, e enviava para a impressora.

Então quando o pessoal chegava para a reunião, o relatório já estava impresso com o que precisavam analisar na pauta daquele dia.

Isso é uma economia de tempo absurda, e quando você mostra isso para diretores, gestores, empresários, empreendedores, todo mundo que entende que tempo dentro da empresa é muito escasso, eles vão com certeza querer isso.

Teve um outro projeto em que eu trabalhei que fiz um reporting de avaliação de desempenho de funcionários, eles tinham um sisteminha onde os gestores lançavam as avaliações do colaborador, só que essa pessoa tinha o gestor do departamento, o gestor de um projeto, de outro projeto, então tinha várias pessoas que avaliavam várias coisas.

O que eu fiz foi pegar o BI Publisher e construir um relatório, como se fosse um livrinho com o nome da pessoa, as avaliações, as características, as habilidades que ela aumentou, as que ela diminuiu, e coloquei alguns gráficos.

E o legal é que o gestor, na hora de avaliar, simplesmente apertava um botão e já saía impressa a avaliação daquele determinado funcionário.

Por mais que a gente queira colocar inovação, ainda existe muita gente que gosta de tomar decisão pegando no papel e fazendo anotações em cima.

Você também pode automatizar esse relatório para ser enviado por email para a pessoa que vai utilizar ele. Ele é customizado e personalizado para aquela pessoa que está recendo o e-mail, ou seja, aqui está o BI na sua essência máxima, que é entregar a informação certa para a pessoa certa no momento certo.

No momento que você entende isso aqui, você pode ajudar verdadeiramente o tomador de decisão a fazer um trabalho muito melhor do que ele vinha fazendo.

Eu fiz diversos jobs desse estilo com grandes diretores financeiros, o pessoal de finanças adora esses reportings.

É um negócio simples, se você aprende qualquer ferramenta que implementa esse tipo de técnica, você aprende uma vez e acabou. Mas isso tem tanta importância para os tomadores de decisão que você não faz ideia do quanto eles ficam felizes com isso e o quanto isso economiza de grana para esse pessoal.

Aqui a gente não está falando de análises multidimensionais com Data Warehouse, de Data Science, de Big Data, a gente está falando do simples e efetivo relatório de BI, e às vezes, por causa dos hypes que vão aparecendo no mercado, a gente acaba esquecendo do básico. Você precisa começar dos fundamentos, e se você está lendo isso agora, já começou certo, você está indo pela base.

É uma coisa que eu sempre digo: o Messi não nasceu jogando no time profissional do Barcelona, ele começou a jogar com 7 anos no Newell's Old Boys. Não nasceu da noite para o dia um astro, e esse é o segredo, um passo de cada vez, vai implementando e vai entregando valor.

Acho que a grande mensagem aqui é: Saiba entregar valor para o seu cliente, saiba entregar algo que ele realmente, verdadeiramente vá utilizar.

Dashboards

O dashboard, também conhecido como painel, é uma técnica para visualização de dados, o famoso painel de controle da empresa.

Em uma única tela, você vai organizar análises, gráficos, métricas, KPIs e reports de toda a empresa ou de um ponto de vista do negócio ou departamento.

Trouxe aqui um exemplo:



Fonte: Pentaho.

O objetivo dele é tomar uma decisão de alto nível, com a possibilidade de fazer drill, ou seja, poder descer no nível da análise e ver mais informações.

A ideia de um dashboard é que a gente consiga encaixar várias peças do nosso quebra-cabeça em um local só.

Ainda são poucas as empresas que sabem o benefício que isso tem. Elas podem trabalhar com painéis gerenciais, táticos e inclusive operacionais, não tem problema.

O dashboard apresenta visualmente as informações mais importantes e mais necessárias para a tomada de decisão. Não adianta você pegar um dashboard e achar que é só socar tudo lá dentro e botar todas as informações que está pronto.

Ele ajuda o usuário de negócio, o tomador de decisão, a visualizar graficamente toda a empresa em uma única tela, então como eu falei, não adianta você criar 700 dashboards e dizer que é dashboard estratégico da empresa, isso não tem nada de estratégico. A pessoa não pode ter que olhar 29.000 gráficos para tomar uma decisão.

Você pode ter um dashboard para venda, um para prospecção de clientes, um para suporte e um para qualidade no atendimento, isso não tem problema. O que não pode é encher de coisa desnecessária que não vai ajudar na tomada de decisão.

E importante: dashboard bom não é o que tem gráficos bonitos, é o que entrega as informações necessárias para a tomada de decisão.

Várias ferramentas que implementam a parte de Data Analysis também implementam a de Visualization, como Power BI e OBIEE. Além delas, também tem Datazen, Tableau, CTools, Qlikview.

FUNDAMENTOS DO DATA WAREHOUSE

Depois de você ter entendido do que se trata o BI como um todo, eu posso começar a me aprofundar no Data Warehouse. A primeira coisa você vai fazer agora é instalar as ferramentas necessárias, criar o banco de dados e fazer as conexões que precisar, para a gente poder começar a modelar o Data Warehouse.

Eu vou te mostrar detalhadamente a criação do Data Warehouse, vamos passar por todas as etapas juntos passo a passo, vou te mostrar os prints das ferramentas para você não ficar com dúvida em nenhuma parte e conseguir implementar tudo tranquilamente e sem erros. E caso você tenha dificuldade com as instalações das ferramentas, pode mandar um email para suporte@raizzer.com que meu time vai te auxiliar nessa etapa.

FERRAMENTAS DE DATA WAREHOUSE

A primeira coisa que você vai fazer é instalar as ferramentas que vamos utilizar para a criação do Data Warehouse.

Ferramenta 1: PostgreSQL

O banco de dados relacional que eu gosto de usar para criar o Data Warehouse é o PostgreSQL, que é o banco que vou utilizar para os exemplos deste livro.

Este passo a passo de instalação é feito considerando que você está instalando o PostgreSQL pela primeira vez no computador. Se você já tiver instalado ele antes, algumas configurações já vão estar prontas, então pode ter diferenças na instalação.

Para baixar, acesse o site do PostgreSQL em <https://www.postgresql.org/download/> e selecione a opção de download para o seu sistema operacional.

No caso do Windows, ele vai redirecionar para uma página para você baixar o instalador.

Nela, você vai clicar em *Download the Installer*.

The screenshot shows the PostgreSQL website's download page. On the left, there's a sidebar with links like 'Downloads', 'Binary', 'Source', 'Software Catalogue', and 'File Browser'. A red arrow points from the text 'baixe a versão 9.6.x.' in the previous slide to the 'Downloads' link in the sidebar. The main content area is titled 'Windows installers' and features a section for 'Interactive installer by EnterpriseDB'. It includes a link to download the installer, a note about its contents, and instructions for running it. Below this, there's a note for advanced users about a zip archive.

PostgreSQL

The world's most advanced open source database.

Home About Download Documentation Community Developers Support Your account

Windows installers

Interactive installer by EnterpriseDB

[Download the installer](#) certified by EnterpriseDB for all supported PostgreSQL versions.

This installer includes the PostgreSQL server, pgAdmin, a graphical tool for managing and developing your databases, and StackBuilder, a package manager that can be used to download and install additional PostgreSQL tools and drivers. StackBuilder includes management, integration, migration, replication, geospatial, connectors and other tools.

This installer can run in graphical or silent install modes.

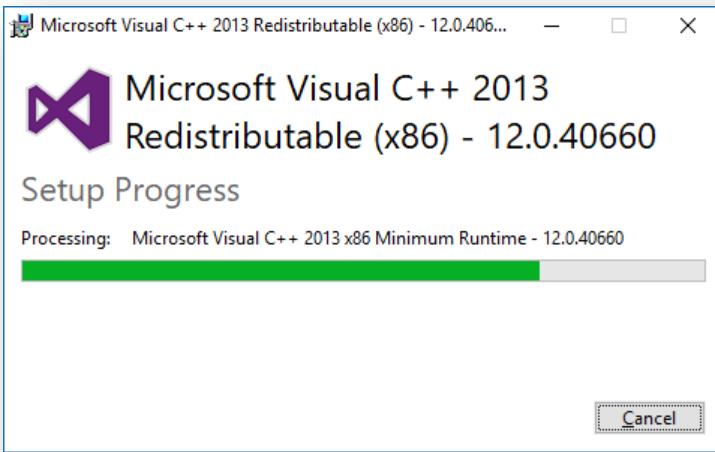
The installer is designed to be a straightforward, fast way to get up and running with PostgreSQL on Windows.

Advanced users can also download a [zip archive](#) of the binaries, without the installer. This download is intended for users who wish to include PostgreSQL as part of another application installer.

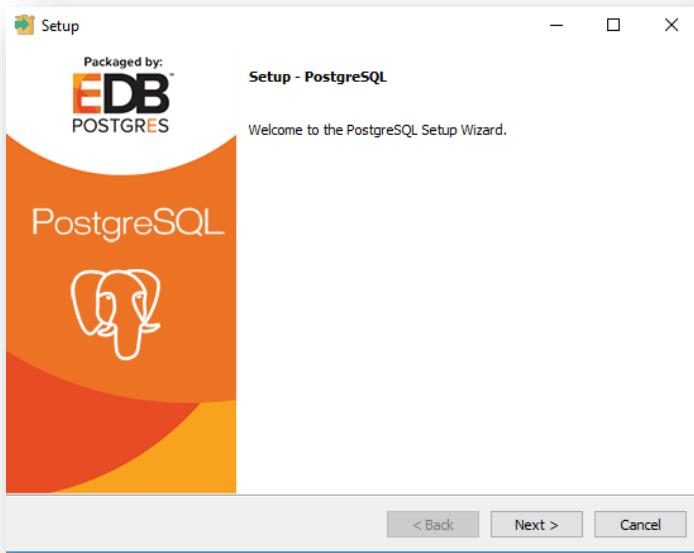
Na próxima página, na hora de escolher a versão do PostgreSQL, baixe a versão 9.6.x.

Depois de baixar, é só você abrir o instalador e seguir os passos abaixo.

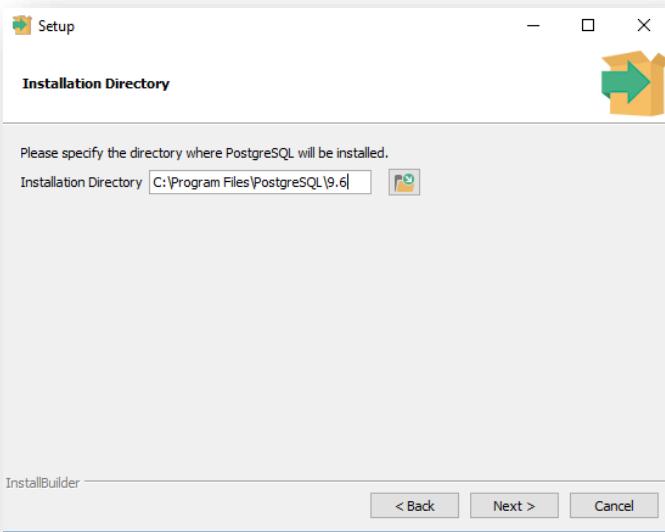
Se você estiver usando Windows, pode precisar instalar ou atualizar o Visual Studio. Ele vai fazer a instalação sozinho, mas pode demorar um pouco.



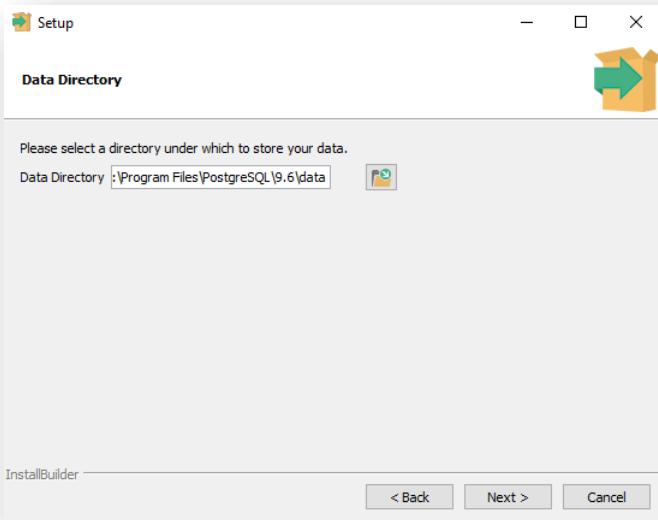
No instalador do PostgreSQL, na primeira tela você não precisa fazer nada, só está iniciando o instalador.



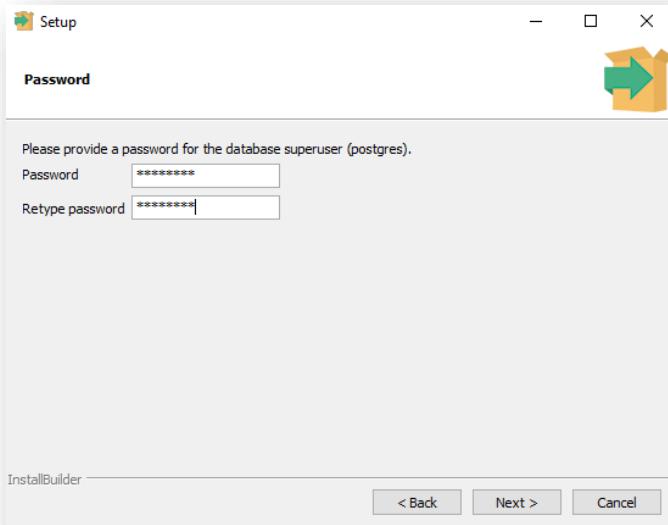
Aqui você vai escolher o local para instalar o PostgreSQL.



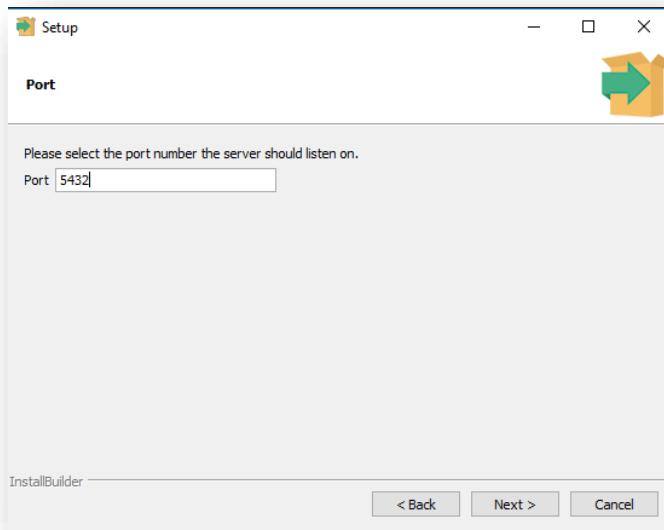
E depois vai selecionar a pasta onde vai armazenar os dados. Mas ele já vai preencher isso para você por padrão, não precisa se preocupar.



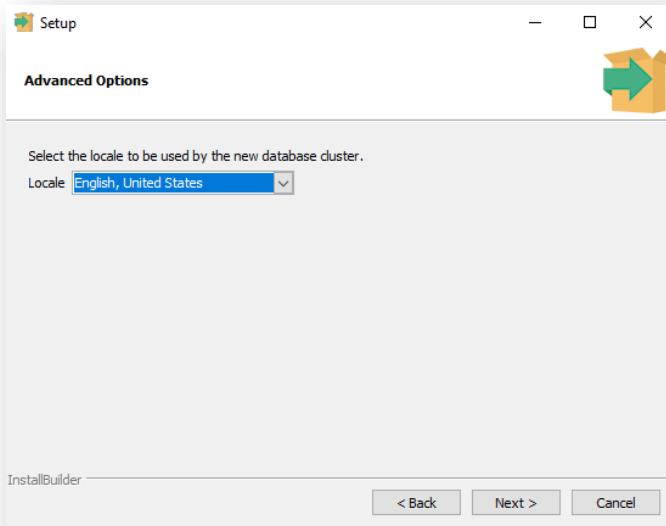
Depois, você terá que definir a senha para o usuário *postgres*, que é o administrador. É importante que você salve essa senha, porque vai precisar dela para acessar o banco depois.



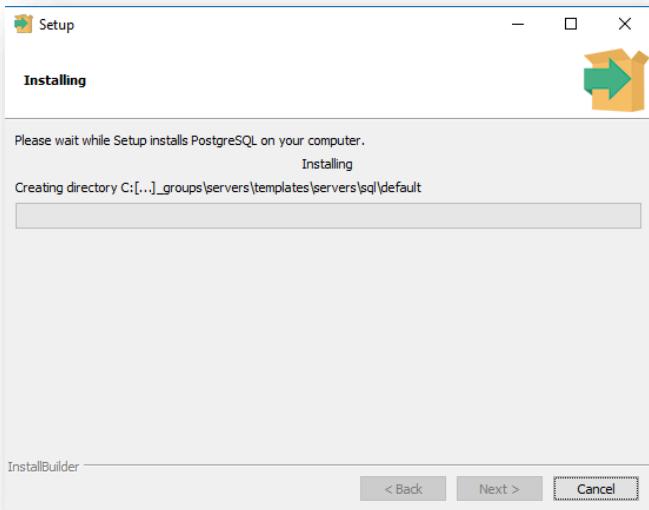
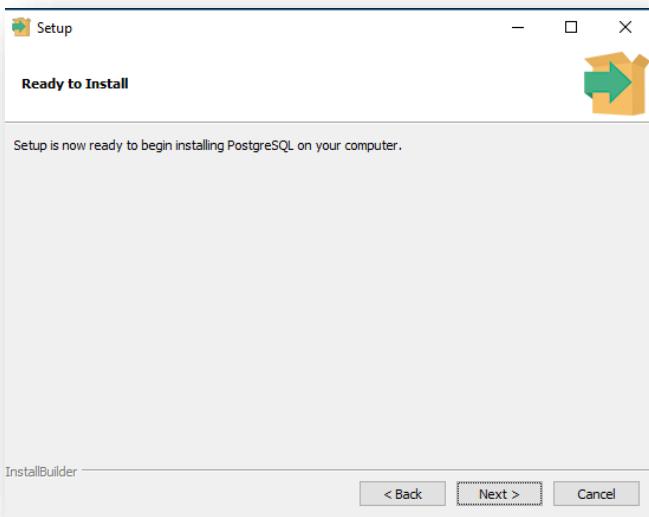
Ele vai pedir para você definir a porta que o servidor deve escutar. Por padrão ele vai preencher com 5432. Você pode deixar essa mesma se não tiver colocado algum outro software para usar ela.



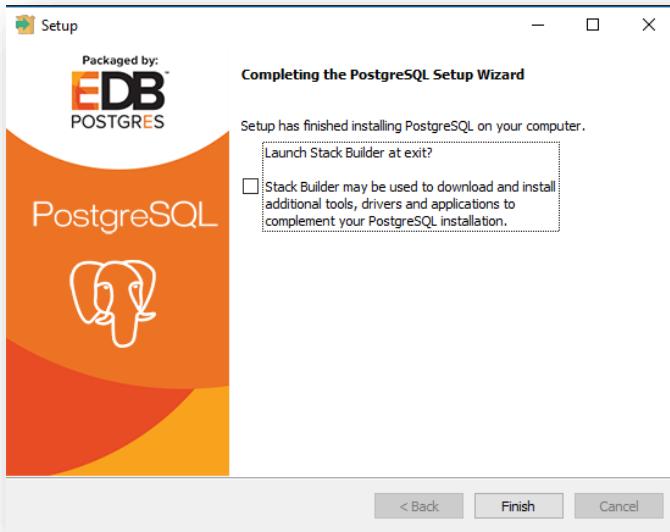
E a última configuração é você definir a *locale* que o banco deve utilizar. Por padrão, eu utilizei *English, United States*.



E agora ele vai finalizar a instalação.

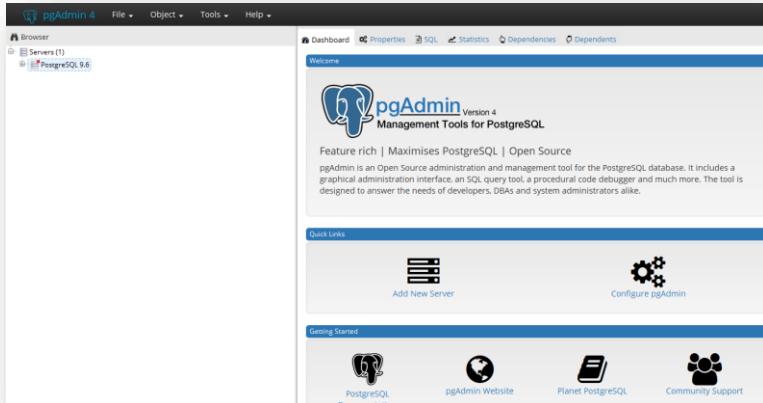


No final, ele vai perguntar se você quer abrir o *Stack Builder* quando sair do instalador, mas não tem necessidade.

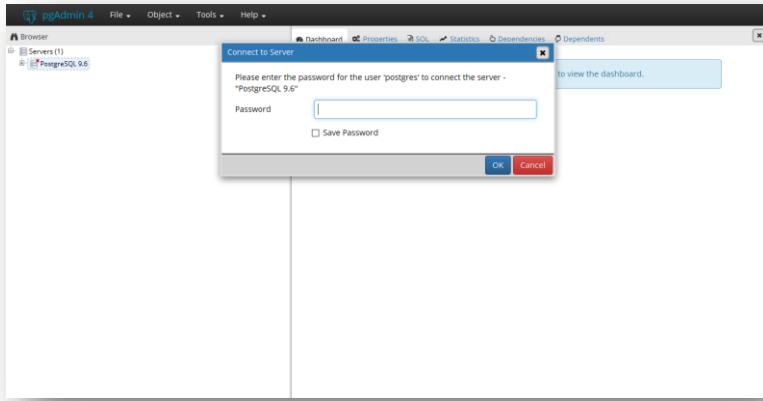


Com o banco de dados instalado, abra o *pgAdmin 4*.

No *pgAdmin 4*, na parte da esquerda, você vai abrir o *Servers* e depois clicar duas vezes em *PostgreSQL 9.6*.

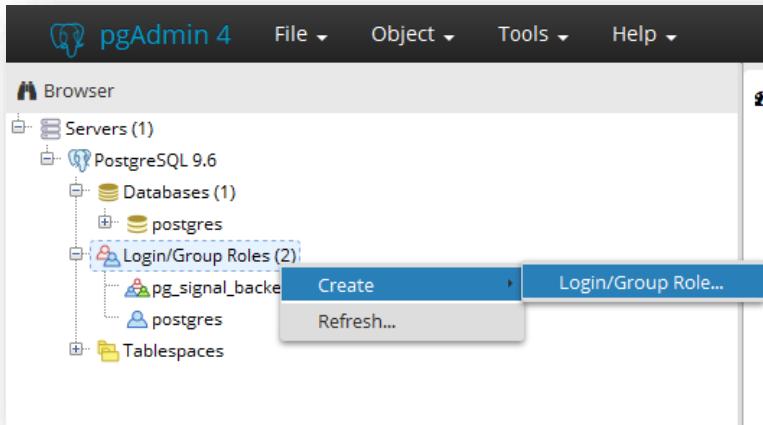


Ele vai pedir para você digitar a senha do usuário *postgres*, que é aquela senha que você definiu durante a instalação.

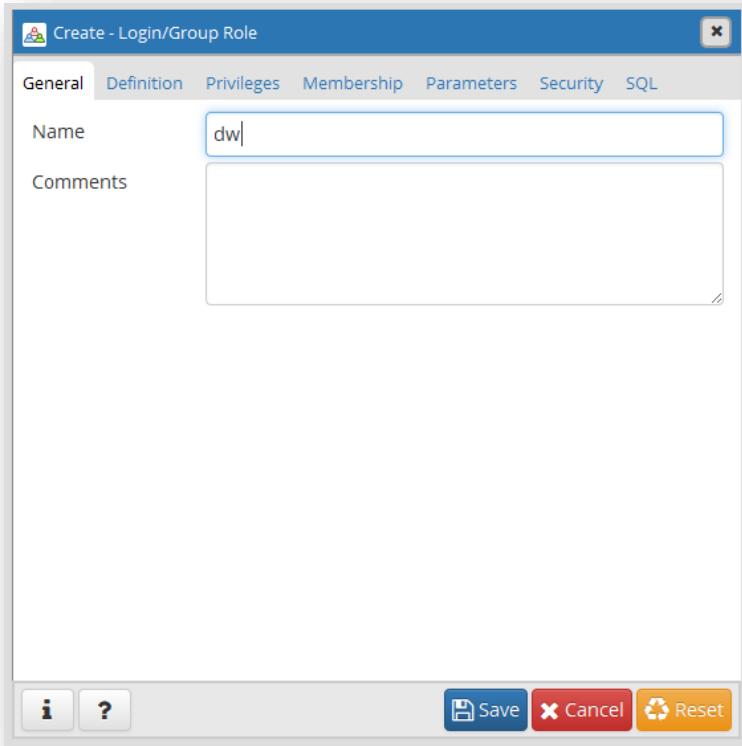


Depois de se conectar, você precisa criar uma nova Role (usuário) para gerenciar seu Data Warehouse.

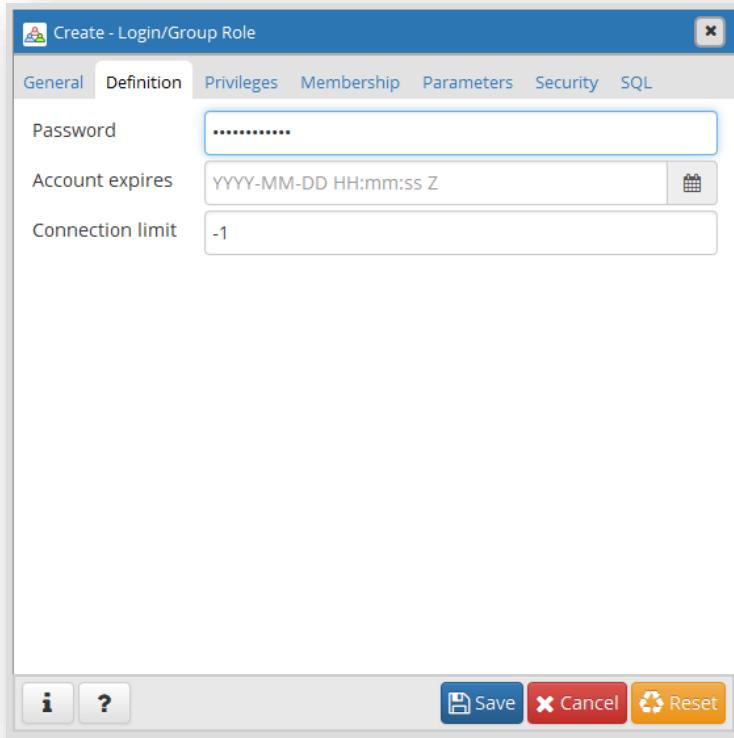
Para isso, clique com o botão direito em *Login/Group Roles*, em *Create* e então em *Login/Group Role...*.



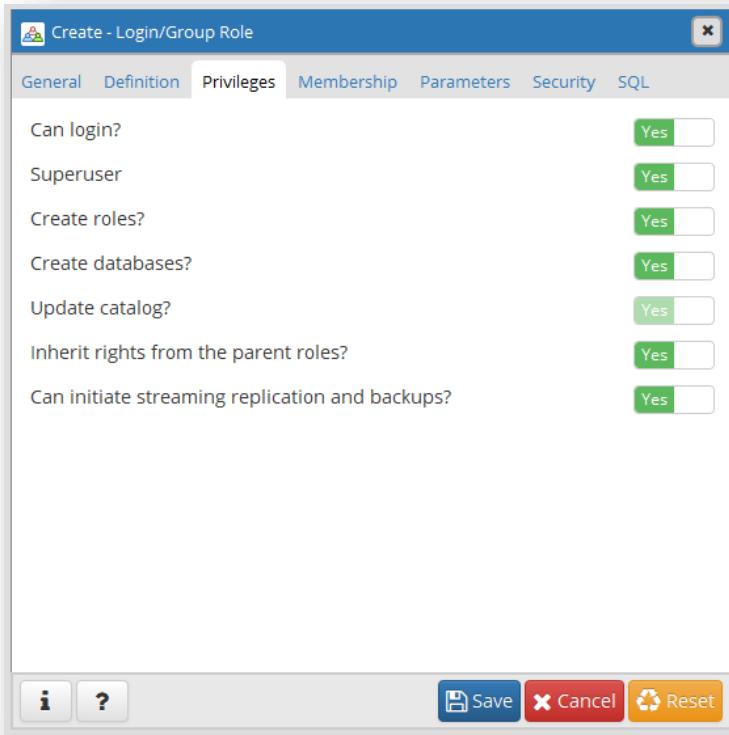
Na aba General, você vai definir o nome da role, que eu vou chamar de *dw*.



E na aba *Definition* você vai definir a senha. Novamente, você vai precisar saber essa senha depois.

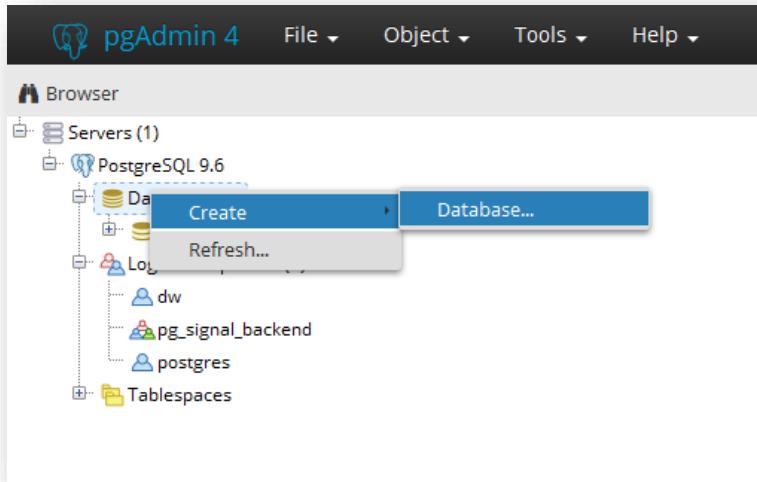


Na aba *Privileges* você vai definir que permissões esse usuário terá. Para fins didáticos, vamos colocar todas as permissões para não ter nenhum problema. Na vida real, isso vai depender das diretrizes da empresa, e muitas vezes nem é o profissional de BI que faz essa parte, ele já recebe o usuário pronto.

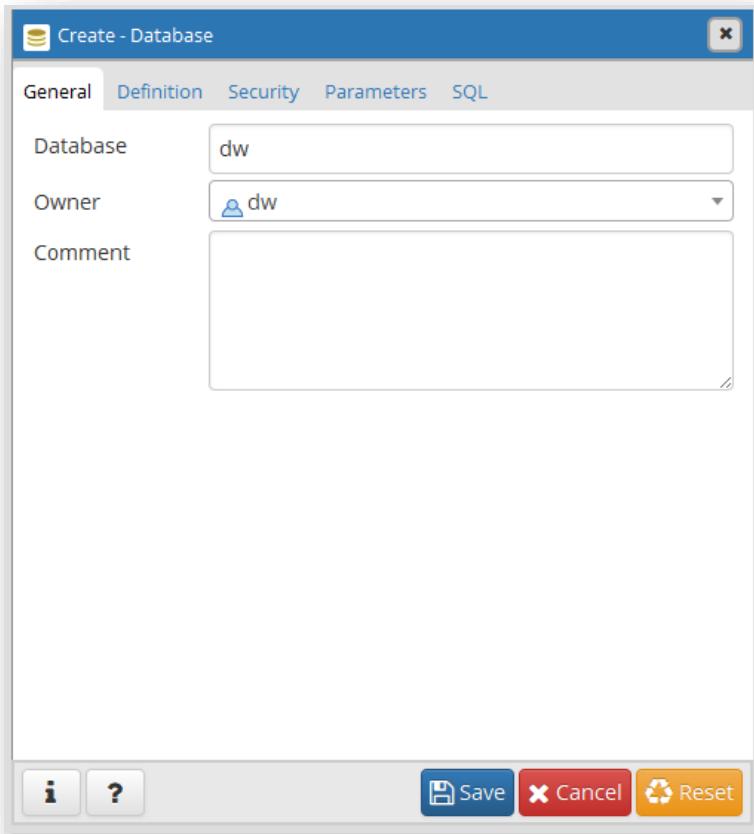


Com a Role criada, você já pode criar o banco de dados para o Data Warehouse.

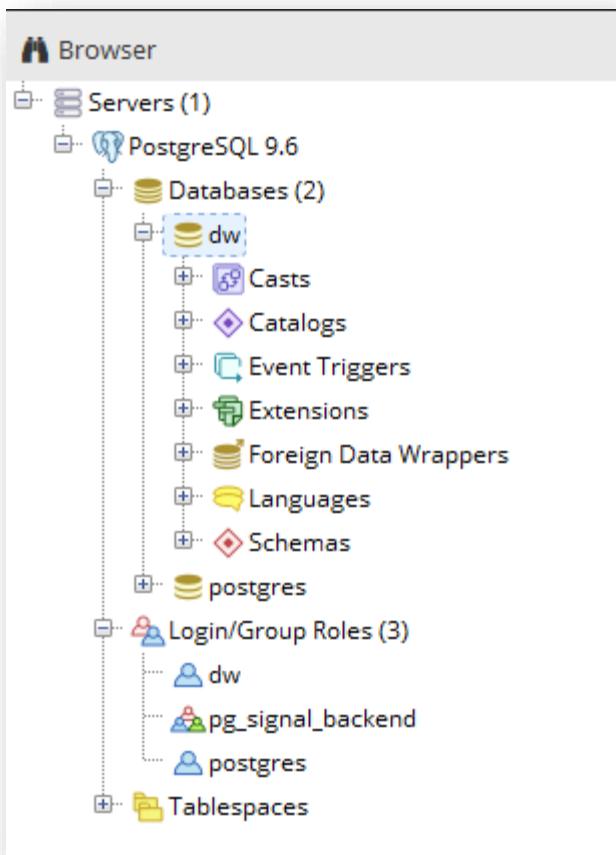
Clique com o botão direito em *Database*, selecione *Create* e então *Database...*



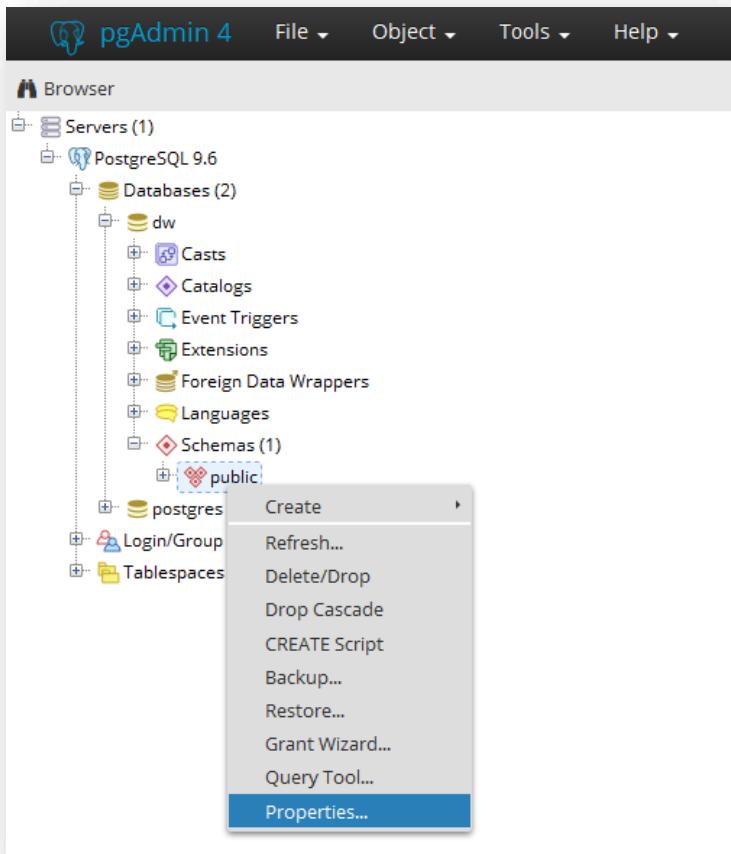
Aqui, na aba *General*, você vai definir o nome do banco, que eu vou chamar de *dw*, e o owner (dono do banco), que deve ser a role que você acabou de criar.



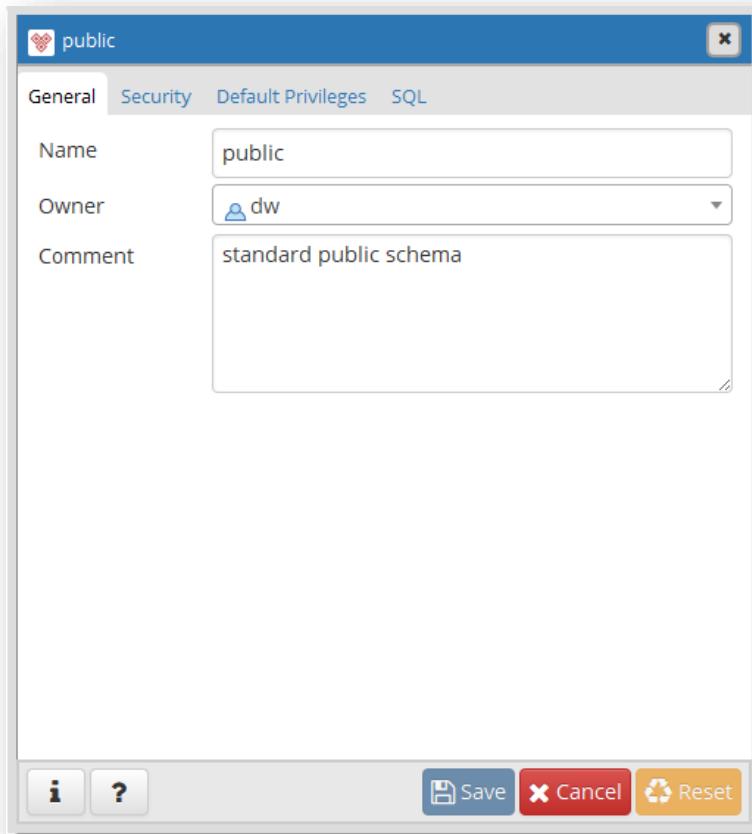
Feito isso, você está com o banco de dados criado.



Você só precisa agora alterar o dono do Schema public desse banco. Para fazer isso, você vai clicar com o botão direito em *public*, que vai estar dentro de *Schemas* e em *Properties...*



Na aba *General*, troque o *Owner* e coloque a role que você criou.



E com isso, o banco de dados está pronto para ser usado.

Ferramenta 2: SQL Power Architect

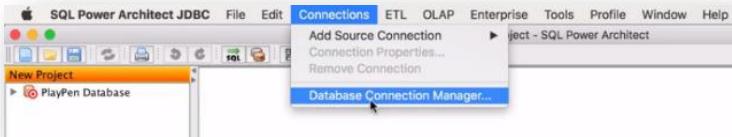
O SQL Power Architect é a ferramenta que a gente vai usar para desenhar o Data Warehouse e criar o que precisar dentro do banco.

Lógico, há muitas outras ferramentas para desenhar o Data Warehouse, mas eu gosto muito dela porque:

- é uma das mais usadas no mercado;
- sua curva de aprendizado é baixíssima;
- é open source e gratuita. Se você quiser usar algumas opções mais avançadas, precisa pagar, mas para o normal do dia a dia, o SQL Power Architect atende muito bem a demanda.

Você pode baixar o SQL Power Architect e instalar ele. A instalação do SQL Power Architect não tem mistério. No Windows pode ser que você precise rodar o instalador como administrador, mas para isso é só você clicar com o botão direito em cima do arquivo e clicar em Executar como administrador.

Para conectar o SQL Power Architect com o banco de dados, clique em Connections na barra de menu superior e em Database Connection Manager...



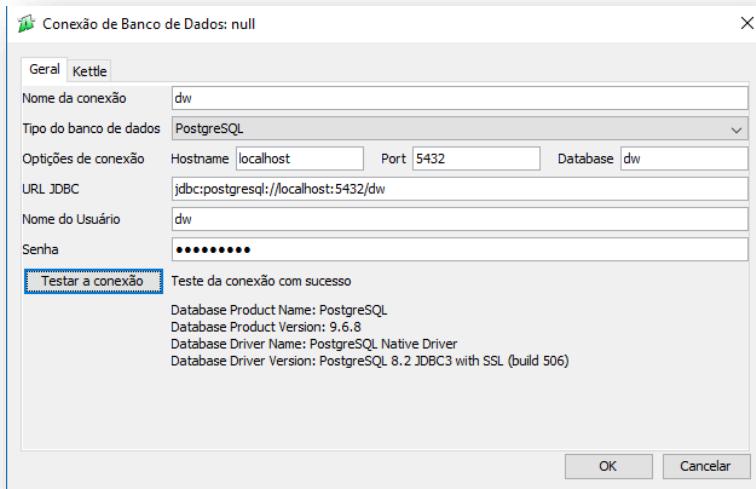
Na janela que vai abrir, você clica em Add...



Aí você vai preencher as configurações:

- **Connection Name** - aqui você pode colocar o nome que você quiser, é só o nome da conexão;
- **Database Type** - aqui vai ser PostgreSQL;
- **Hostname** - como o banco de dados está no seu computador, o hostname vai ser localhost;
- **Port** - a porta é a que você configurou na instalação do PostgreSQL, que no meu caso foi a 5432;
- **Database** - coloca o nome do banco de dados, que nesse caso é dw;

- **JDBC URL** - essa parte ele já vai preencher automaticamente conforme você for preenchendo as opções de cima;
- **Username** - aqui é o nome da role que você criou para acessar o banco do Data Warehouse;
- **Password** - e aqui vai ser a senha da role.



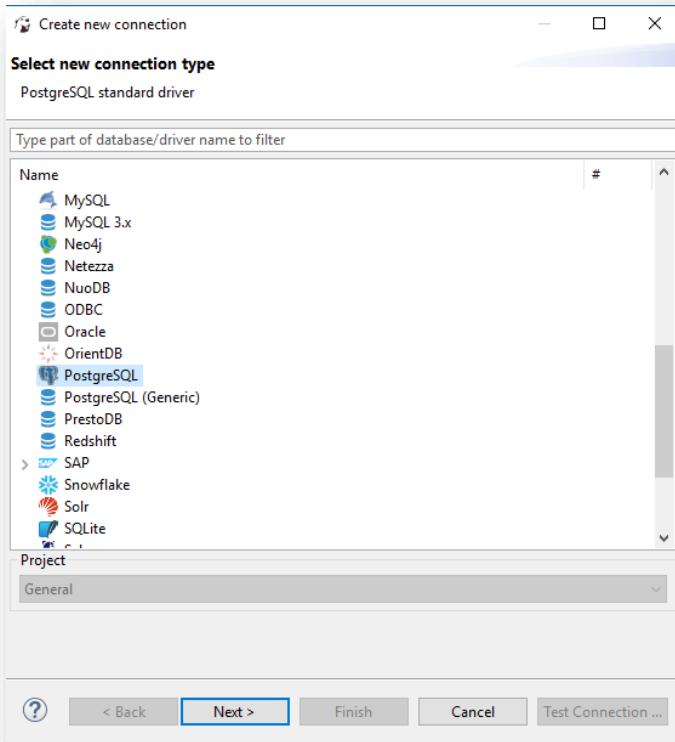
Depois clica em *Test Connection*. Se a *mensagem Connection test successful* aparecer está tudo certo. Se não, verifique se você não esqueceu de fazer alguma coisa.

Esses são os procedimentos que você precisa fazer para iniciar a construção do seu Data Warehouse.

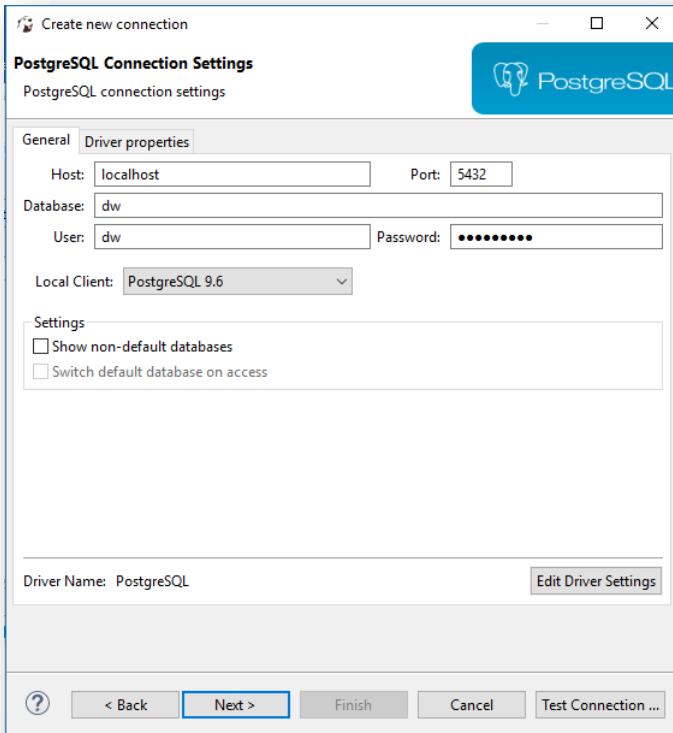
Ferramenta 3: DBeaver

O DBeaver é uma ferramenta open source de acesso ao banco de dados, ela não é muito bonita visualmente, mas é muito robusta e conecta em vários bancos de dados.

Você pode [baixar o DBeaver](#) e instalar ele. Depois de instalado, você vai precisar conectar o DBeaver ao banco de dados também.



Abrindo o DBeaver, você vai cair nessa tela aqui, onde nós vamos escolher o tipo de banco de dados, que no nosso caso vai ser PostgreSQL.

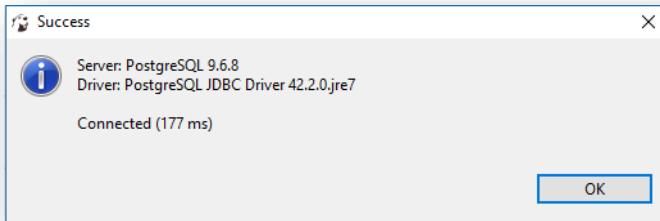


Depois, você vai colocar as configurações da conexão com o banco, igual você fez no SQL Power Architect.

- **Host** - aqui vai ser *localhost*, porque você está com o banco no computador local, e não em um servidor;
- **Port** - a porta que você configurou na instalação do PostgreSQL, que nesse caso é *5432*;

- **Database** - o banco de dados que você criou para o Data Warehouse, que foi o *dw*;
- **User** - o usuário que você criou para gerenciar o banco de dados do Data Warehouse, que também se chamava *dw*;
- **Password** - a senha que você definiu para o usuário *dw*.

Depois você pode clicar em *Test Connection*, e ele vai avisar que precisa baixar e instalar o driver do PostgreSQL. É só você autorizar e esperar ele dar a mensagem confirmando que a conexão está funcionando.



Depois é só você apertar em *Finish* e a conexão está pronta.

DADOS NORMALIZADOS VS DADOS DESNORMALIZADOS

Os dados normalizados permitem um armazenamento consistente e acesso eficiente aos dados. Essa forma reduz a redundância de dados e as chances de inconsistência. Esse estilo de banco tem como foco inserir, alterar e deletar os dados. É utilizado em ERPs, CRMs, enfim, no sistema transacional, na origem dos dados.

Se você fez alguma faculdade em Tecnologia da Informação, provavelmente aprendeu sobre esse tipo de modelagem, que é o clássico modelo ER (Entidade Relacionamento).

Já os dados desnormalizados, que é com o que nós trabalhamos, têm foco na consulta.

As consultas feitas em bancos de dados totalmente normalizados têm um mau desempenho, e para isso se usa a desnormalização: para melhorar o desempenho das consultas.

Mas claro, com um custo: com ela não temos a garantia de consistências dos dados, além de termos um banco muito maior.

O objetivo dos dados desnormalizados é entregar informação, é um sistema informacional, feito para ter uma consulta rápida. Vai ter redundância de dados, mas vai ser mais rápido.

Como isso funciona?

Por exemplo:

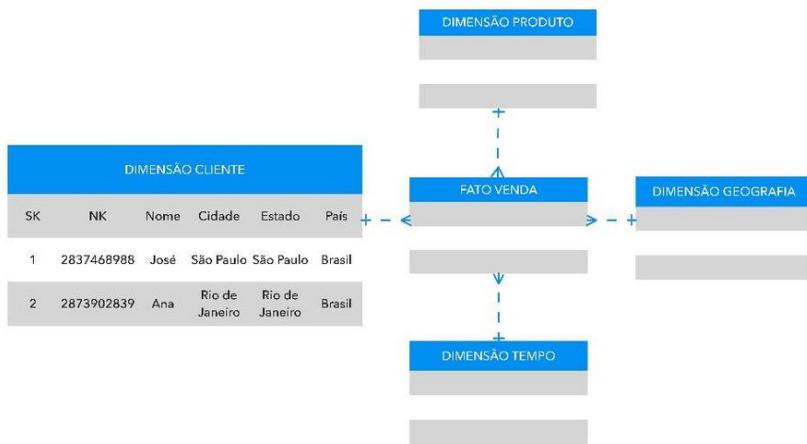
Em um banco normalizado, ou relacional, temos diversas tabelas, cada uma com sua chave primária e informações.

Temos uma tabela “cidade” com o ID da cidade e o nome dela, e outra tabela “cliente” com o ID do cliente, o nome dele e a cidade onde mora. Mas nesse último item, ao invés de termos o nome da cidade registrado, temos o ID referente àquela cidade na tabela “cidade”.



Isso serve para evitar a redundância, evitando que o mesmo nome seja armazenado diversas vezes e garantindo que ele estará escrito sempre da mesma forma para todos os clientes.

No caso dos dados desnormalizados, nós temos o nome da cidade registrado na tabela de clientes uma vez para cada cliente que morar naquela mesma cidade, o que aumenta muito o tamanho do banco e não garante tanta consistência dos dados.



Em um modelo transacional, ou sistema operacional, você precisa ter integridade dos dados, eles não podem se repetir. Já no modelo dimensional, eles podem. Na verdade, eles devem.

Muitos bancos de dados não têm separação o suficiente entre projeto lógico e a base de dados, você não consegue ver uma implementação física e lógica do banco de dados. E isso acontece porque a normalização do banco foi malfeita. Sabe aquele codigozinho extra, aquele puxadinho que se faz lá no transacional? Se você já foi programador ou DBA, sabe que sempre acontece um puxadinho no código, e a gambiarra que funcionou vira produção. O banco não volta

mais ao estado natural dele, e isso é o que a gente encontra pelo mercado, então tem muita, muita bagunça.

Então geralmente nos projetos, a gente cai nessas buchas aí, qual é a malandragem aqui? Saber resolver isso de forma que a gente se livre desse problema, garanta que o nosso trabalho vai ser impecável, vai ser classe A, e se o sistema transacional estiver todo bugado e cheio de furos, o problema é de quem cuida do sistema transacional.

Nesse tipo de situação, por vezes, a desnormalização ajuda a melhorar o desempenho de consultas porque não vai passar por todos aqueles buracos que tem no transacional.

E como se faz isso?

Os dados desnormalizados são utilizados na modelagem dimensional, uma das técnicas e conhecimentos mais importantes que você precisa ter para modelar o Data Warehouse, Data Mart ou o que for.

O modelo dimensional existe há muito mais tempo que você imagina, e provavelmente, você que não conhece modelo dimensional, já até usou e não sabe.

Até para utilizar ferramentas, na parte de modelar os metadados ou cubos OLAP, você vai precisar entender de modelagem dimensional, a não ser que você utilize outro tipo de arquitetura de modelo de dados.

Existem dois tipos de metodologias de modelagem de dados usadas no Data Warehouse: a Snowflake e a Star Schema, que é a mais utilizada.

Basicamente, o que eu quero que você entenda é: normalização foi feita para entrada e saída de dados com foco na transação.

Desnormalização é o modelo que o Data Warehouse utiliza para melhorar o desempenho da consulta e tem um custo de redundância de dados, mas ele consegue responder muito rápido uma pilha enorme de dados.

OBJETIVOS DO DATA WAREHOUSE

Aqui eu quero falar um pouco sobre os objetivos do Data Warehouse.

Para resumir o que é um Data Warehouse: ele tem a ideia de otimizar o processo de consultas e análises de dados estruturados, ou seja, ele pega os dados de um ERP, CRM, geralmente de todos os dados estruturados que a gente tem dentro da organização e alguma coisa semiestruturada, e faz uma modelagem para que ele consiga suportar o processo de consultas em grande volume de dados.

Às vezes a gente se confunde um pouco entre Data Warehouse e BI, porque eles têm muitas coisas em comum. Antigamente só se falava em projetos de Data Warehouse, e hoje em dia se fala em projetos de Business Intelligence e o Data Warehouse está incorporado dentro do BI.

Então, os objetivos do Data Warehouse são:

- permitir que a gente analise os eventos do passado e tome uma decisão melhor com base no que aconteceu;
- permitir um tempo de resposta em minutos e garantir um alto índice de desempenho. Esse é um dos grandes objetivos dele, a gente não pode ter um Data Warehouse que demora muito

para responder. Geralmente quando isso acontece, o problema está na modelagem dos dados. Quando você instalar uma ferramenta de OLAP para fazer os cubos, ela vai conectar no Data Warehouse, e aquela ferramenta é super-rápida, mas se o Data Warehouse estiver uma carroça e não responder a tempo, é trabalho em vão;

- garantir a consistência e confiabilidade dos dados. Em quase todos os lugares que utilizam técnicas de Data Warehouse de verdade, ele é o ponto central para a tomada de decisão, ou seja, os dados que estão lá são confiáveis, são consistentes. O que está lá é real;
- garantir que você tem 100% de controle dos dados. Tem que ter uma forma de ver quem que está vendo que dado, ou seja, se você precisar tirar o acesso de uma pessoa, isso tem que ser fácil.
- possibilitar a fácil interpretação e acesso dos dados;
- garantir a unicidade dos conceitos e regras de confiabilidade dos dados. A mesma informação divulgada por áreas diferentes dentro da empresa deve ser exatamente igual. Isso aqui é onde o cara da planilha, o planilheiro de Excel, fica nervoso, porque para isso acontecer, precisa morrer as milhares de planilhas, a gente só consegue fazer com que as mesmas informações divulgadas por áreas distintas fique igual no momento que você elimina aquele ser que fica entre a planilha e a cadeira;
- ser um repositório único e centralizado com a verdade única da empresa. Você quer saber qual a margem de contribuição? O Data Warehouse te diz disso, ou pelo menos ele vai te dar

os dados, as métricas, para que você possa fazer seus cálculos.;

- o que mais acontece quando a empresa começa a crescer é ela virar um barco viking desgovernado, vai cada um remando para um lado e ninguém sabe mais para onde está indo. Então o Data Warehouse é o Ragnar Lothbrok dos vikings, mostrando o caminho que devem seguir. E quem não aceitar, atira para fora do barco;
- ser adaptável e flexível às várias fontes de dados. Você não vai fazer ele uma vez só e acabou, ele vive em constante alteração. Mas importante, você nunca, nunca, nunca, nunca vai desenhar um Data Warehouse com base nos dados do legado, nos dados do sistema transacional. O Data Warehouse é desenhado com base no negócio, na tomada de decisão. Chega a ser triste ver equipes, e às vezes até com gente senior, modelando Data Warehouse com base nos dados que a pessoa tem. Se você faz algo que já existe no transacional e que você não pode cruzar e fazer nada, como é que você vai tomar decisão? Baseada em quê? Assim você não está tomando decisão no negócio;
- permitir o controle do acesso aos dados, onde a gente pode também ter a visibilidade global de onde os dados estão sendo utilizados. Eu perdi a conta da péssima segurança dos dados que tem nas empresas. Vou te dizer que eu acho que até hoje, se eu pegar os acessos que me passaram, eu consigo acessar o Data Warehouse dos clientes que eu já passei, só para você ter uma ideia. Não estou dizendo que eu vou vender os dados, mas no momento que o consultor saiu, precisa tirar o acesso dele. Também perdi a conta de quantas vezes, dentro de empresas, alguém estava vendendo um dado que não era para ver.

Mas como é que faz tudo isso? É uma ferramenta? É simplesmente a ferramenta que eu disponibilizei para modelagem e um banco de dados relacional. O resto é técnica de modelagem de Data Warehouse.

Por isso que eu estou mostrando o que você precisa entender, estou deixando aqui o que acontece na vida real.

E depois, na hora que a gente entrar na parte de modelagem, eu vou mostrar para você como funciona isso na ferramenta, a gente vai ver tudo funcionando.

Para fechar, o objetivo do Data Warehouse qual é? De tudo que eu falei agora, entregar a informação certa para a pessoa certa no tempo certo. O Data Warehouse é desenhado com base no negócio, é para suportar a tomada de decisão.

ARQUITETURA DE DATA WAREHOUSE

Você deve estar pensando: “para que eu preciso saber disso? Quero colocar logo a mão na massa.” Mas calma que a gente já vai chegar lá.

É o seguinte:

Antes de a gente efetivamente colocar a mão na ferramenta e criar nosso primeiro modelo dimensional, você precisa entender porque diabos a gente precisa de um, acho que é o fundamental.

Eu sempre falo da importância da arquitetura, de entender como que isso funciona. Nada do que você aprendeu até aqui foi em vão, é para que você possa entender onde a gente quer chegar.

Aqui só estou mostrando para você a realidade, a prática. Às vezes a gente abre a ferramenta, às vezes não, eu estou mostrando aqui o que a gente usa no mercado.

Vou dividir essa arquitetura em três locais pelos quais os dados vão passar.

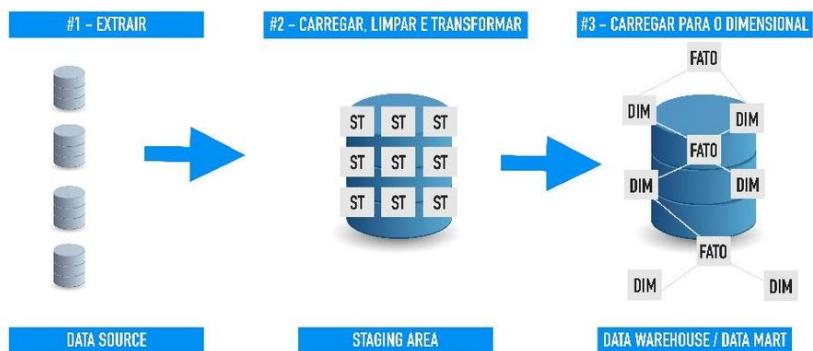
O primeiro é o Data Source, nossas fontes de dados, o transacional da vida.

O segundo é a staging area, que é um banco de dados relacional que vai receber todos os dados que a gente tem no Data Source, mas já vou dar mais detalhes sobre isso.

O terceiro é propriamente o nosso Data Warehouse ou Data Mart, ele é um outro banco de dados relacional como a gente conhece, com linha, coluna e tabela.

Tem 3 passos para você seguir que vão deixar o seu Data Warehouse pronto:

- **1º passo:** extrair os dados das fontes de dados;
- **2º passo:** carregar, limpar e transformar os dados na staging area;
- **3º passo:** carregar os dados para o modelo dimensional.



Primeira coisa, você precisa tirar os dados do transnacional para limpar e transformar. Você não pode alterar os dados que estão em produção e nem deixar o sistema lento, porque as pessoas estão utilizando ele.

No segundo passo, você carrega esses dados na staging area. Eu vou entrar em mais detalhes sobre ela na parte do ETL, mas por

enquanto, é mais um banco de dados, que a gente vai utilizar para colocar esses dados que retiramos do transacional. E na staging area você vai transformar esses dados, fazer aquelas alterações que a gente viu na parte do Data Integration.

E depois disso, como a gente fala no dia a dia, você vai carregar para o dimensional.

E o que é esse dimensional? É o nosso modelo Star Schema, com aquele monte de fatos e dimensões, que eu vou explicar melhor no Passo 3.

O Data Warehouse ficou mundialmente conhecido pela utilização de fatos e dimensões, mas se você voltar 10 ou 15 anos, muitas dessas etapas, como a staging area e a parte de integração, ficava dentro do Data Warehouse, como se fosse a mesma pessoa que fizesse tudo isso.

O tempo passou e as coisas foram mudando, então hoje a integração é muito mais complexa, e por isso foi separada. Quem trabalha com integração de dados às vezes trabalha só com isso, e quem trabalha com modelagem de Data Warehouse, trabalha só com isso.

Mas antes de entrarmos nesse assunto, quero falar sobre a diferença de um Data Warehouse e um Data Mart, porque isso sempre gera muita confusão.

O Data Mart tem uma fato que armazena os dados que foram organizados na staging, e tem as dimensões que se relacionam com essa fato.

Essa modelagem do Data Warehouse é feita para que a consulta seja rápida. Conceitualmente, esse modelo que parece uma estrela é chamado de Star Schema. Essa estrela, na sua essência, é um Data Mart.

No momento que a gente começa a juntar os Star Schemas, se torna um Data Warehouse.

O Data Mart é isolado por área de assunto ou departamento, e quando é feita a junção de vários assuntos, temos um Data Warehouse.

Hoje em dia se fala também em Enterprise Data Warehouse, que é mais um nome novo para algo que já existia. Quando você vê a diferença entre o Data Warehouse e o Enterprise Data Warehouse, estão chamando o Data Mart de Data Warehouse e o Data Warehouse de Enterprise Data Warehouse. Essa questão do EDW surgiu porque ele atende a empresa toda.

Antigamente, quando se tinha um Data Warehouse pronto, se conectava nele para extrair o Data Mart necessário para aquela tomada de decisão.

Era criado outro banco de dados só com esse Data Mart para ter mais performance e um resultado melhor das informações. Mas isso porque antigamente a memória RAM era muito cara.

Durante 6 meses ou 1 ano, era feita a modelagem e os testes para chegar em um super Data Warehouse com tudo.

Depois disso, se criava um outro banco de dados para criar o Data Mart, e aí vinham as ferramentas de análise e conectavam ali.

O projeto do SBT que eu participei, foi vendido tipo Big Ben, a gente tinha que implementar o projeto até uma certa data e foi desse jeito, criamos tudo para só depois entregar. Eu e time do projeto acabamos tendo retrabalho porque quando tem dimensões que são compartilhadas, você tem que sentar com os dois setores para entender o que os dois precisam, e isso é sempre difícil de acontecer.

É por isso eu vou apresentar o método que eu uso, que é o Agile Data Warehouse. Vou mostrar como você pode desenvolver um Data Warehouse de forma rápida, simples e com baixo custo.

O que a gente faz hoje para ter mais rapidez e agilidade na implementação? A gente pensa grande e começa pequeno.

No primeiro projeto, você deve começar com apenas um assunto e fazer todo esse processo até carregar os dados. No segundo projeto, ataca o próximo departamento, e assim vai até estar com todo o Data Warehouse completo.

Assim, você entrega pequenos pedaços já funcionais e mantém o Data Warehouse vivo e recebendo sempre atualizações e melhorias.

FUNDAMENTOS DA MODELAGEM DIMENSIONAL

Agora você já entendeu todos os fundamentos do Data Warehouse e já está com o ambiente todo preparado para construir seu Data Warehouse.

No Passo 3, a gente vai entrar na modelagem dimensional, vamos entender o que é um Star Schema e como funciona cada um dos seus componentes, também vamos modelar todos eles e persistir no banco de dados.

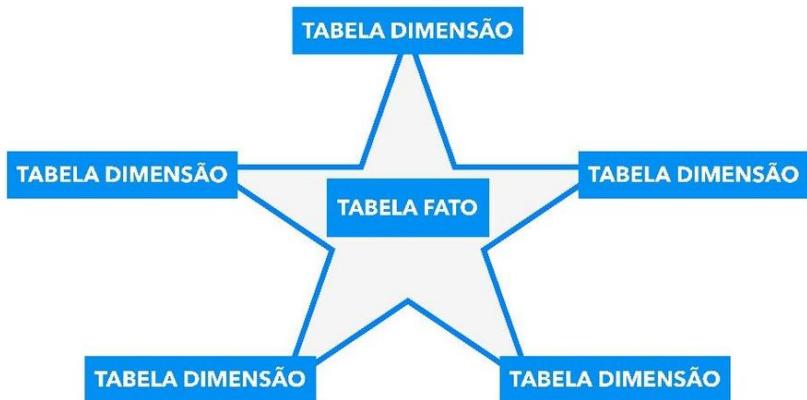
Você vai terminar essa parte já tendo um Data Warehouse em mãos.

Mas pode ficar tranquilo que vou te mostrar cada etapa por partes e vamos fazendo essa modelagem juntos.

STAR SCHEMA

O conceito de Star Schema, ou modelo estrela, foi idealizado por Ralph Kimball, um dos precursores do modelo Star Schema. A ideia dele é propor uma visão para modelagem de dados que suporte a tomada de decisão orientada a dados, e a modelagem dimensional é uma das técnicas mais utilizadas para fazer isso, sendo o Star Schema o coração dessa modelagem dimensional.

O Star Schema é composto no centro por uma tabela fato que é rodeada por tabelas de dimensão, ficando parecido com a forma de uma estrela (que é de onde vem o nome).



A modelagem Snowflake, defendida pelo Bill Inmon, também é projetada para suportar tomada de decisão, mas economizando espaço em disco. Para o Star Schema, a Snowflake é apenas mais um tipo de dimensão.

Embora o Star Schema ocupe mais espaço em disco, ele é mais fácil de implementar e acabou sendo mais utilizado porque além de hoje em dia o espaço em disco ser muito barato, ele permite entregar projetos por pedacinhos. Você pode criar um assunto ou departamento em uma estrela para só depois projetar outro, e assim por diante.

Na maioria esmagadora dos projetos, Star Schema é uma excelente escolha.

Antes de criar o Star Schema, você deve conhecer suas características e componentes.

Então agora o que você precisa entender é como funciona:

- a tabela fato;
- a tabela dimensão.

TABELA FATO

A tabela fato armazena o que ocorreu, é o fato propriamente dito, por isso ela tem esse nome, porque é o fato ocorrido. A tabela fato está sempre ligada a duas ou mais dimensões, não existe tabela fato com menos de duas dimensões.

Essa tabela armazena 2 coisas:

- as métricas;
- as foreign keys.

As métricas são o dado bruto, um valor numérico, que a gente já viu na parte de Data Analysis.

A foreign key é uma combinação de uma ou mais colunas e serve para você relacionar os dados entre duas tabelas. Você vai usar ela para conectar a fato com a dimensão, que vai ter as informações que descrevem as métricas.

TABELA FATO

FK DIM 1	FK DIM 2	FK DIM 3	FK DIM 4	MÉTRICAS
FK da foto conecta com a PK da dimensão				Dado numérico

Alguns exemplos de fato seriam:

- vendas;
- contas a pagar;
- contas a receber;
- estoque;
- compras.

Existem 6 tipos de fatos:

- fato transacional;
- fato agregada;
- fato consolidada;
- fato snapshot periódico;
- fato snapshot acumulado;
- fato sem fato.

A gente vai se aprofundar em cada um desses tipos mais para frente.

TABELA DIMENSÃO

A dimensão, também chamada de qualificador, descreve o fato ocorrido, ela contém as características do evento.

Por exemplo, quando eu faço uma venda, quero saber por onde a venda foi feita, que produto foi vendido ou para quem.

Elas vão qualificar, classificar ou descrever os dados que estão nas fatos, ou seja, as métricas.

Para identificar as dimensões, normalmente perguntamos pelo que o usuário quer mensurar uma informação. Elas vão ser toda e qualquer informação que qualifique uma métrica ou medida.

Por exemplo:

Preciso medir as vendas.

Ótimo, mas pelo quê?

Pela empresa / pelo produto / pelas filiais / por dia.

Você também pode usar a pergunta de quais filtros o usuário gostaria de ter naquela visão.

Por que filtros?

Imagina que você tem um dashboard com um menuzinho e o usuário pode escolher: eu quero um relatório por produto.

Aí ele clica naquela setinha e tem a lista de todos os produtos. Ele seleciona um deles e mostra um gráfico com as vendas só de café, por exemplo.

Esses filtros, na maioria dos casos, vão ser as dimensões. E embora nem todo filtro seja uma dimensão, podem ser dimensões candidatas, que ainda não foram colocadas no modelo, mas podem vir a ser colocadas.

As dimensões armazenam 3 coisas:

- a surrogate key;
- a natural key;
- os atributos.

TABELA DIMENSÃO

Surrogate Key (PK)	Natural Key	Atributo 1	Atributo 2
Chave artificial autoincremental	PK da origem/ legado	Qualificadores da dimensão	

Alguns exemplos de dimensão seriam:

- loja;
- cliente;
- produto;
- data/tempo;
- região/geografia;
- funcionário;
- fornecedor.

As chaves, a surrogate key e a natural key, eu vou explicar o que são e como funcionam no próximo capítulo. E os atributos são as características do fato, os qualificadores dele. Em uma dimensão Loja, por exemplo, os atributos seriam:

- nome;
- endereço;
- bairro;
- cidade;
- estado;
- país.

TIPOS DE CHAVES

Aqui eu vou explicar todas as chaves que você vai utilizar na criação do seu Data Warehouse. Separei um capítulo só para isso porque é algo que costuma causar bastante confusão, mas é fundamental que você entenda para que cada uma delas serve.

Mas antes de eu falar sobre as chaves que o Data Warehouse utiliza, preciso que você entenda o que é uma primary key. Se você já trabalha com TI, provavelmente já está acostumado com ela.

A primary key, ou chave primária, é uma coluna ou uma combinação de colunas em uma tabela que serve para identificar cada linha de forma única. Ou seja, quando eu defino uma coluna para ser primary key, eu garanto que a integridade dos dados não vai ser corrompida, porque ela não pode ser repetida, cada linha da tabela vai ter uma primary key diferente.

Quanto às chaves do Data Warehouse, a surrogate key é uma primary key da dimensão. É uma chave artificial e autoincremental.

Ela é artificial porque não existe em lugar nenhum, não está no transacional, ela é criada no Data Warehouse.

E é autoincremental porque toda vez que é chamada, troca de número, então ela começa com 1 e vai indo para 2, 3, 4, e assim por diante.

E ela não é reutilizável. Se toda vez que você carregasse a dimensão fosse reutilizar a chave, viraria uma zona e você não saberia mais o que está acontecendo.

Uma surrogate key nada mais é que um campo com as características de uma primary key, e é gerada automaticamente na hora da carga, quando você carrega a dimensão no ETL.

A foreign key, ou chave estrangeira, serve para você relacionar os dados entre duas tabelas. Então quando eu conectar a dimensão na fato, a surrogate key da dimensão vai ser referenciada na fato como uma foreign key, simples assim.

Já a natural key, ou business key, que é usada nas dimensões, é a primary key daquele dado na origem ou legado, é o que você vai usar para identificar de onde ele veio.

No Data Warehouse, ela vira uma simples natural key, não é esse código que vai controlar tudo e fazer o relacionamento, como era no transacional, aqui quem faz isso é a surrogate key.

STAR SCHEMA COOKBOOK

Vou te explicar o passo a passo de como funciona todo o processo de modelagem do Data Warehouse em Star Schema. Você segue esse passo a passo e não tem erro.

Tem gente que paga fortunas, rios de dinheiro para aprender isso em um MBA que leva dois anos para ensinar o que eu vou ensinar agora nessa aula.

Aqui vou te dar uma visão geral de todos os passos dessa construção e depois te explico as técnicas que a gente utiliza para fazer isso.

Na construção de um Star Schema, a gente primeiro define a tabela fato, que possui:

- **foreign keys** - servem para conectar com as primary keys das dimensões;
- **métricas** - são dados numéricos.

TABELA FATO

FK DIM 1	FK DIM 2	FK DIM 3	FK DIM 4	MÉTRICA
FK da fato conecta com a PK da dimensão				Dado numérico

Depois que a tabela fato foi criada e as métricas definidas, é hora de partir para as chaves estrangeiras. E para isso, se cria a primeira dimensão.

O que tem em todas as dimensões que é padrão, não importando qual projeto está sendo desenvolvido:

- **surrogate key** - uma primary key nas dimensões, uma chave artificial e autoincremental;
- **natural key** - a primary key da origem ou legado;
- **atributos** - qualificam os dados da dimensão.

Depois que a dimensão estiver definida, a gente faz um join dela com a tabela fato.

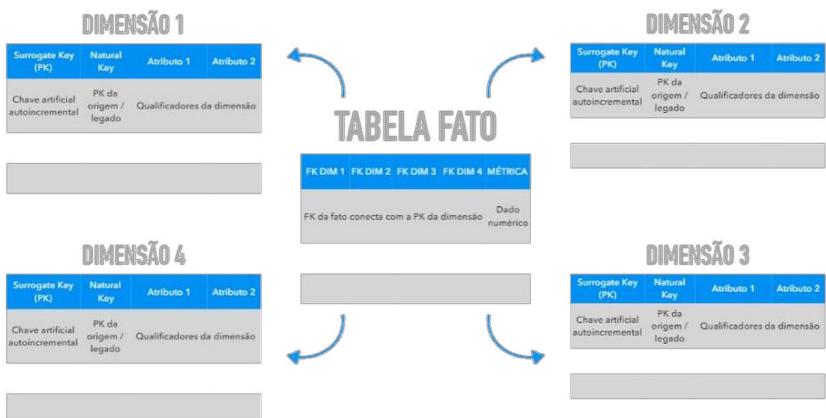
é nesse momento que a foreign key da fato vai e conecta lá na primary key da dimensão, ou seja, na surrogate key.



Depois são definidas as demais dimensões e é feito o join de todas elas com a fato. O procedimento é o mesmo para todas as dimensões.

Aqui só estou fazendo uma modelagem, não tem dado nenhum aqui dentro.

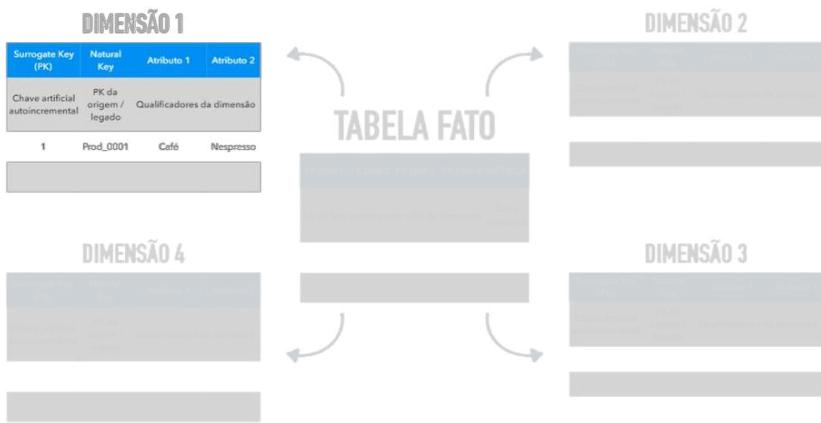
E com isso, a modelagem do Data Warehouse com modelo Star Schema está pronto para receber os dados do ETL.



Entendendo isso aqui, você está com 80% do projeto entregue. E ao longo do livro eu vou te mostrando mais detalhes de como você deixa essa modelagem ainda melhor.

Agora eu quero que você entenda como que isso vai se comportar quando você for jogar os dados dentro dessas tabelas. Eu estou trazendo aqui como que isso deve acontecer conceitualmente, independente da ferramenta.

Então, na hora de fazer o ETL, qual tabela que carrega primeiro? A dimensão.

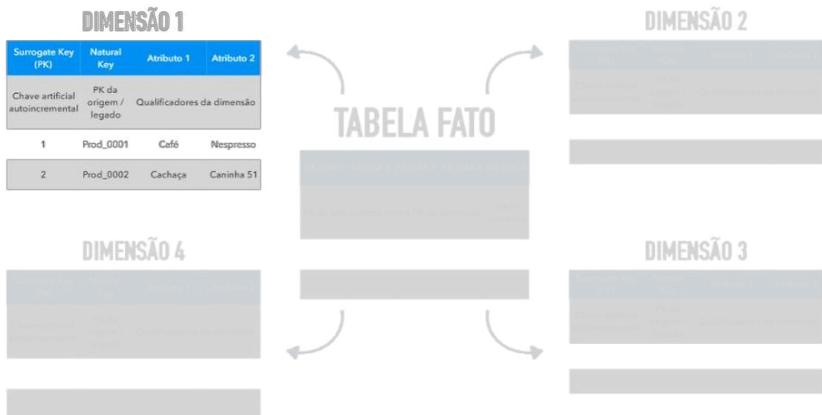


A primeira dimensão é uma dimensão de produto.

A chave autoincremental (a surrogate key) começou a trabalhar, foi lá e definiu a primeira linha como 1.

A natural key é o mesmo código que está na origem.

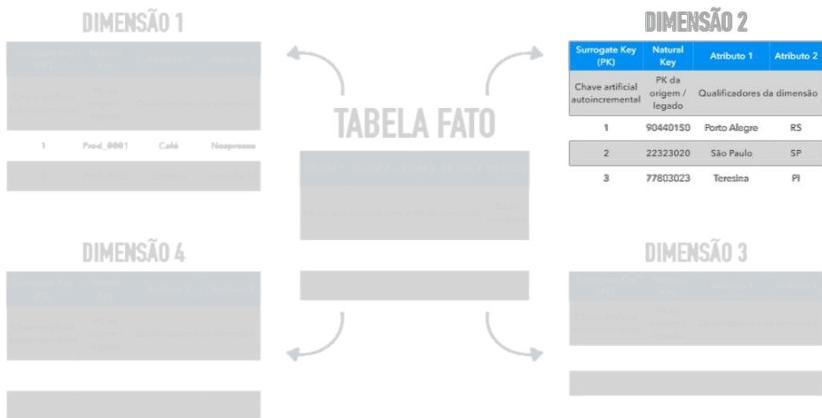
Os outros atributos nessa tabela são o nome e a marca do produto.



Depois adiciona uma nova linha na dimensão produto.

A surrogate key segue a sequência e coloca 2 nessa linha.

E com isso termina a carga da primeira dimensão.



Depois a gente vai lá e carrega a segunda dimensão.

Essa é uma dimensão de geografia, ou localidade.

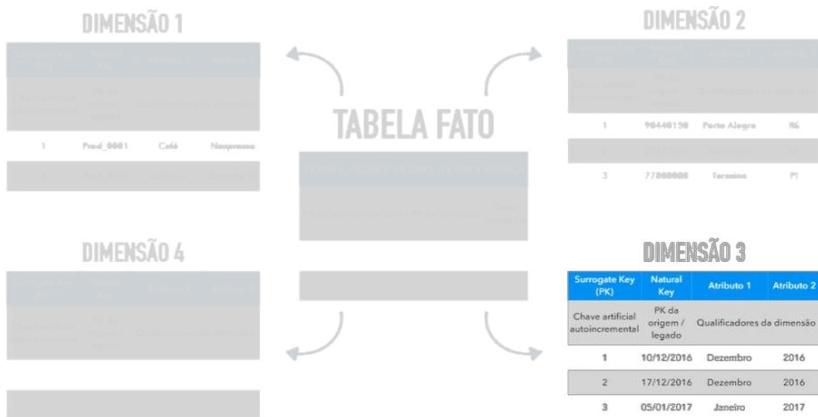
A natural key é o CEP, que é o código único usado para identificar esses dados na origem.

E os atributos são o nome da cidade e do estado.

A dimensão 3 é a dimensão de tempo.

A surrogate key da dimensão de tempo normalmente é a mesma natural key, a própria data, embora não tenha problema se usar sequência como nas outras.

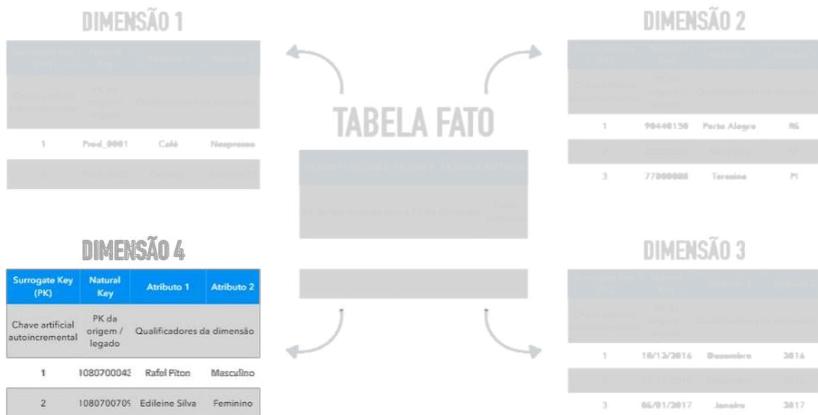
Aqui os atributos são o mês e o ano.



Depois vem a dimensão 4, que é a dimensão de cliente.

E qual a natural key? O CPF do cliente.

O atributo 1 é o nome do cliente e o 2 é o sexo.



Agora que as dimensões estão carregadas, o próximo passo é carregar a fato.

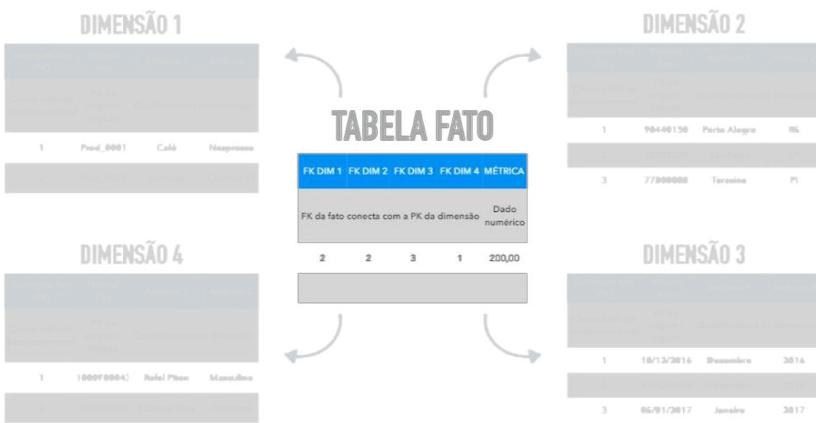
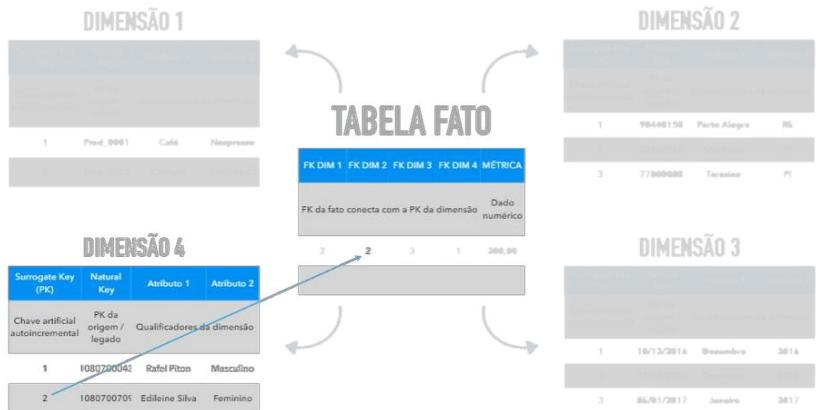
A fato só vai ser carregada depois de todas as dimensões porque ela trabalha com chaves estrangeiras, ou seja, não são chaves dela. E para isso você vai precisar ter as chaves das dimensões.



Na hora de inserir na fato, a gente vai lá na dimensão, pega a surrogate key e coloca na fato como foreign key.

Ou seja, esse 2 que está na dimensão, simplesmente vai ser inserido na linha da fato.

Depois, na dimensão de cliente a gente pega a chave e insere a surrogate key na fato.



O procedimento é o mesmo para todas as dimensões.

No fim, depois de fazer os cruzamentos, a gente vai na fato e insere a métrica.

Agora vamos fazer uma leitura dessa linha que foi registrada na fato para você entender o que isso aqui quer dizer.

Olha só:

No dia 10/12/16, a Edileine Silva comprou uma cachaça da marca Caninha 51, em Teresina no Piauí, no valor de R\$200,00.

Agora, quando você criar suas análises e relatórios, é dentro da fato que você vai buscar essas informações, para depois identificar os atributos daquela métrica nas dimensões através das foreign keys.

Entendendo isso, você está pronto para abrir uma ferramenta de modelagem e tornar isso real. E é exatamente isso que a gente vai fazer no próximo capítulo.

CRIANDO MEU PRIMEIRO PROJETO NO SQL POWER ARCHITECT

Nessa parte, é importante que você tenha entendido os fundamentos e os processos. É normal que apareçam algumas dúvidas enquanto você estiver fazendo a modelagem, mas aí você vai naquele ponto do conteúdo e revisa o conceito.

Eu separei a modelagem do Data Warehouse em passos pequenos e bem detalhados, para quando você tiver alguma dúvida no futuro, ficar mais fácil de encontrar aquele conteúdo específico.

Nesse ponto, você precisa ter o banco de dados criado e conectado com o SQL Power Architect. Se ainda não fez isso, volte no capítulo de Ferramentas de Data Warehouse e siga os passos.

Para fazer a modelagem do Data Warehouse, eu vou usar o SQL Power Architect. Aqui o banco de dados é coadjuvante, porque depois de ter o desenho pronto, você pode subir ele em um Oracle, PostgreSQL, SQL Server, tanto faz, desde que seja um banco de dados relacional. O importante nessa parte é você entender o negócio e desenhar o que faz sentido para ele.

PASSO 1: SALVAR O PROJETO

Depois de abrir o SQL Power Architect, você precisa salvar o projeto, mesmo que ainda não tenha criado nada, porque você pode sair sem salvar, pode faltar luz, o programa fechar, essas coisas, e se você não salvou nenhum projeto, a ferramenta não vai salvar o progresso.

Para isso, vá em *File > Save Project*.

Agora na área do lado esquerdo, vai aparecer o nome do projeto que você acabou de salvar.

PASSO 2: CRIAR DIMENSÕES

Aqui a gente vai criar na ferramenta o Star Schema que você viu no capítulo do Star Schema Cookbook.

Basicamente, vou ensinar como que você coloca aquela modelagem na ferramenta e deixa ele pronto esperando a carga de dados do ETL.

Então vamos criar as dimensões desse modelo que a gente fez, onde nós representamos:

- cliente;
- produto;
- região;
- tempo.

Para criar sua primeira dimensão, clique na opção *Nova Tabela* no menu da direita e preencha os dados na janela que vai abrir.



Nome da Tabela Lógica - é só no nome lógico, pode usar o que for mais fácil de você entender;

Nome da Tabela Física - o nome físico da dimensão. Aqui é aconselhado você usar design patterns, que são padrões de nomenclatura. Mostro os que eu utilizo mais para frente;

Nome da chave primária- o nome do campo que será a chave primary key da dimensão (a surrogate key);

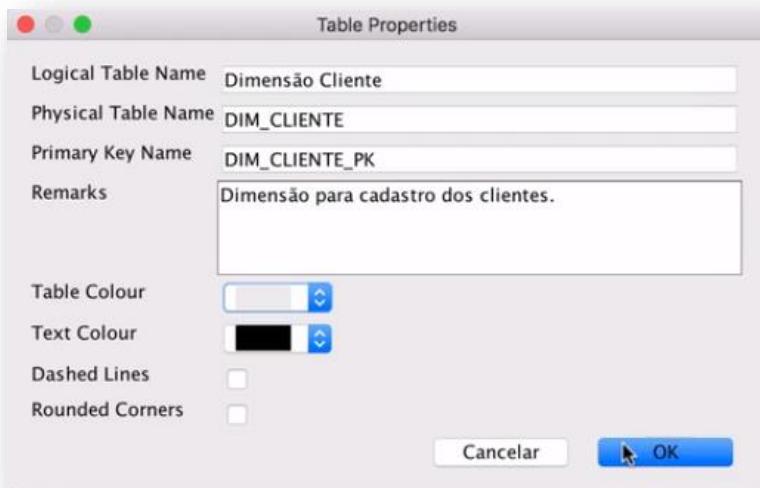
Comentário - campo para você fazer anotações se necessário;

Cor da tabela - você pode escolher uma cor para a tabela se precisar diferenciar elas;

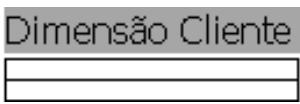
Cor do texto - e pode mudar a cor do texto também;

Linhas tracejadas - para deixar as linhas da tabela tracejadas;

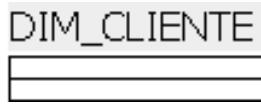
Cantos arredondados - deixa os cantos da tabela arredondados.



Com a dimensão criada, você vai ver que ele está mostrando o nome lógico nas tabelas.



Nome lógico



Nome físico

Se você quiser que mostre o nome físico, vai em *Arquivo > Configurações do Projeto* e lá vai ter uma configuração chamada *Mostrar Tabelas e Colunas Com*, aí é só selecionar a opção *Nomes Físicos*.

Caso você queira mostrar para algum cliente e acha que o nome físico fica muito técnico, pode ir lá e trocar para o lógico de novo.

Feito isso, é hora de criar os atributos, as chaves e tudo mais.

Para isso, você vai precisar de uma coluna, então seleciona a dimensão que você acabou de criar e no menu da direita, clica em *Inserir Coluna*, aí é só preencher os dados na janela que vai abrir.

Fonte para mapeamento ETL - não marca nada, isso é somente para quando a gente desenha o processo de ETL junto na ferramenta. Eu não costumo fazer isso porque o ETL é bem complexo e precisa de um controle bem maior, que a ferramenta aqui não dá.

Nome Lógico - aqui segue a mesma ideia da nomenclatura das tabelas.

Nome Físico - normalmente se usa os padrões de nomenclatura.



Chave primária - marque esse campo quando a coluna for uma primary key (que é o caso das surrogate keys).

Tipo - o tipo do dado (no caso da surrogate key, vai ser integer).

Precisão - tamanho do campo, a quantidade de caracteres que ele vai armazenar.

Escala - esse é para os números, para definir a quantidade de casas decimais.

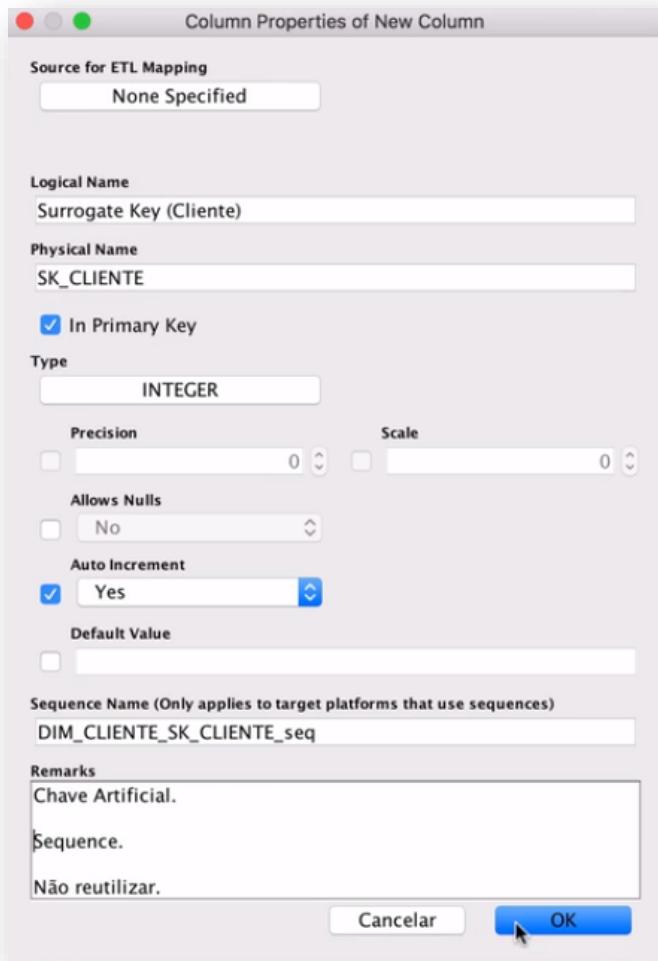
Permite nulos - se a coluna aceita valor nulo.

Auto Incremento - para quando o campo for autoincremental, que é o caso das surrogate keys. No momento que você marca sim nesse campo, ele habilita o nome da sequência, que é a sequence da primary key.

Valor Padrão - para você definir um valor padrão naquela coluna.

Nome da Sequência - nome da sequence da primary key. Ele é preenchido automaticamente, então você não precisa se preocupar com isso.

Comentários - campo para você fazer anotações se necessário. É bom para ajudar a pessoa que for mexer nessa modelagem depois.



Agora a gente criou a nossa surrogate key na dimensão.

E ela vai aparecer assim na sua dimensão:

DIM_CLIENTE
SK_CLIENTE: INTEGER NOT NULL [PK]

No lado esquerdo do design, ele vai mostrar onde que cada coisa está sendo criada.



Depois disso, você cria uma nova coluna para ser a natural key.

Então seleciona a dimensão e no menu da direita, clica em *Inserir Coluna* de novo.

Agora todas as colunas vão ter os mesmos campos de antes, e a ideia é sempre a mesma também.

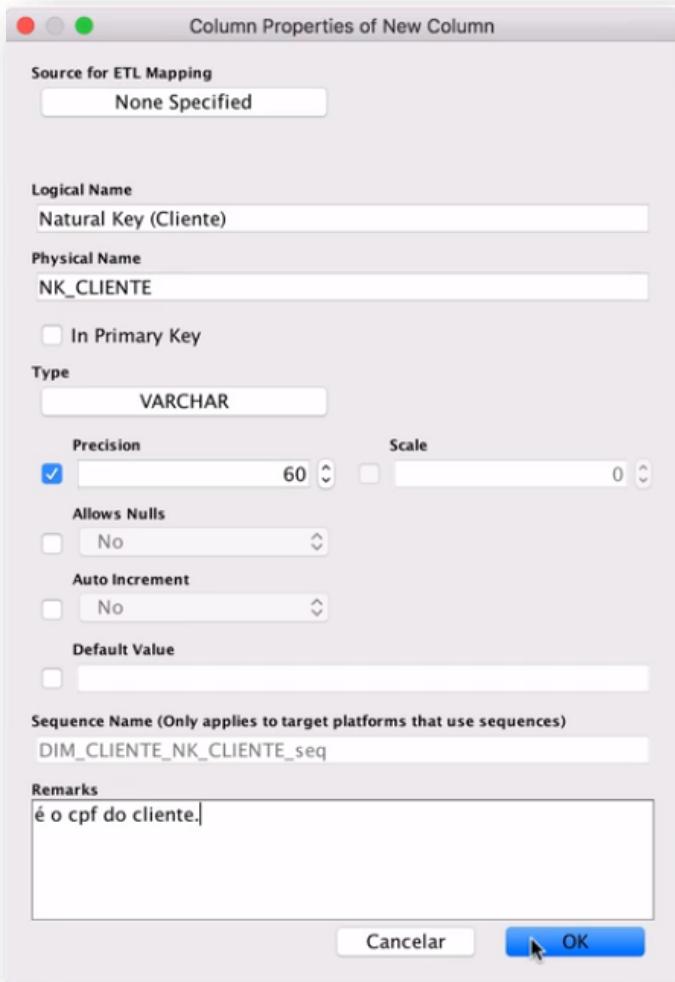
O que vai mudar aqui:

No *nome físico*, eu uso o *NK* para deixar padrão, mas você provavelmente vai encontrar em projetos por aí como *CD_CLIENTE*, de código de cliente. Isso varia de acordo o seu padrão de nomenclatura.

Dessa vez não vai marcar o campo de primary key.

No tipo, os dados vão depender de como o código vai estar cadastrado na origem. Então nem sempre vai ser só número, pode ter

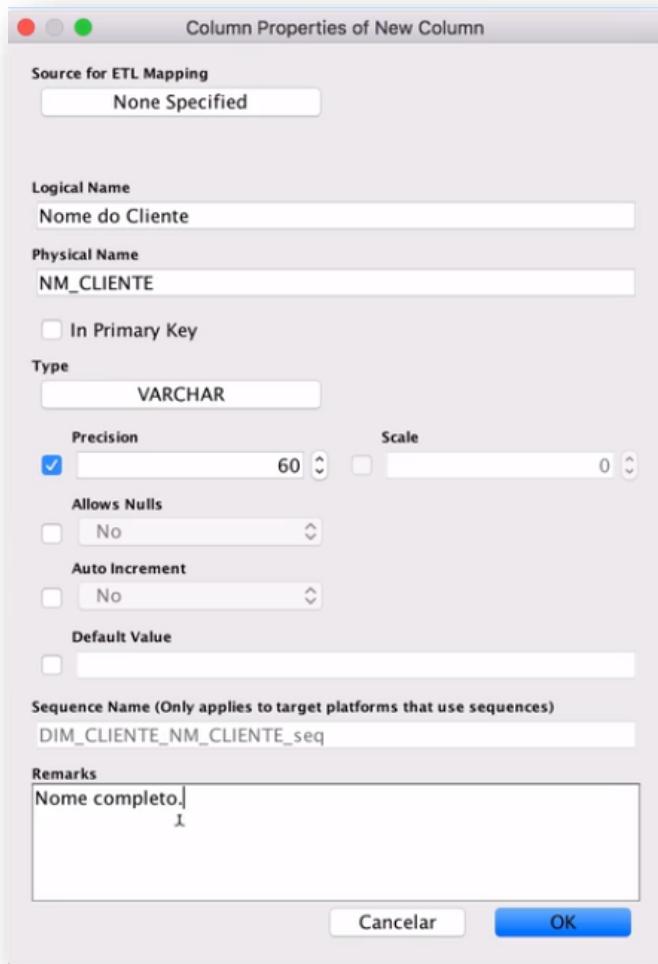
letras ou caracteres especiais também. Eu costumo deixar como *varchar* por causa disso. E na precisão, 60 é o suficiente.

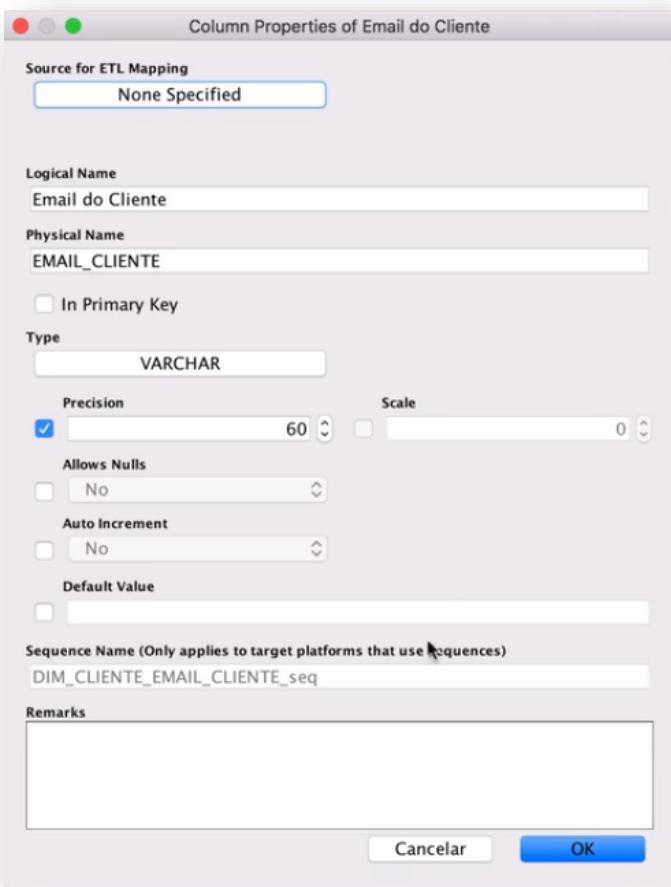


Depois, você cria as outras colunas da mesma forma.

A nossa dimensão de cliente tem os atributos:

- nome;
- email.





E agora a dimensão cliente está pronta.

DIM_CLIENTE
SK_CLIENTE: INTEGER NOT NULL [PK]
NK_CLIENTE: VARCHAR(60) NOT NULL
NM_CLIENTE: VARCHAR(60) NOT NULL
EMAIL_CLIENTE: VARCHAR(60) NOT NULL

Para criar as outras dimensões, o processo vai ser o mesmo.

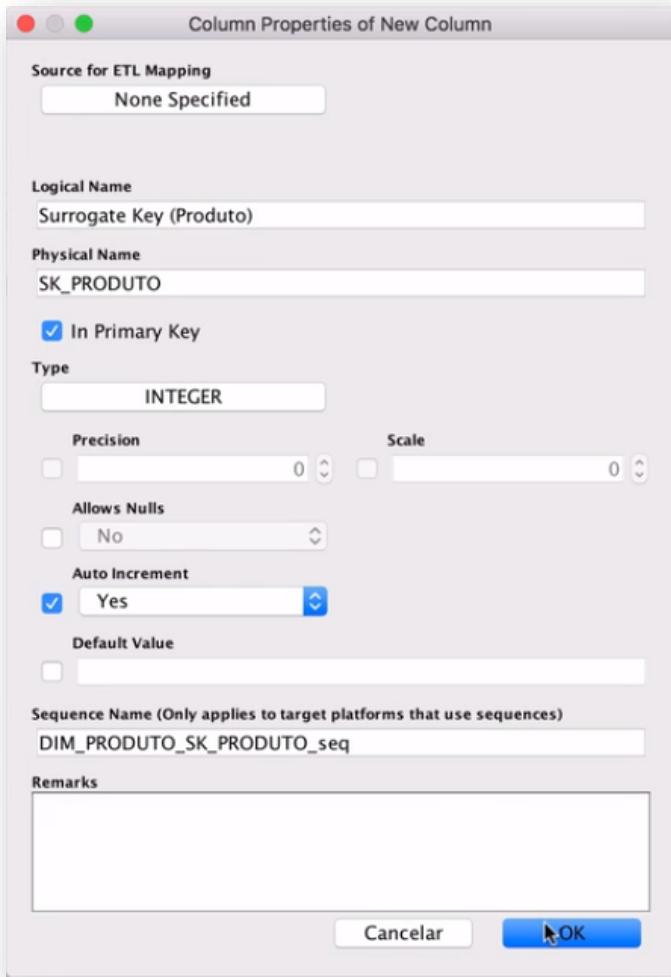
Vamos criar a dimensão de produto:

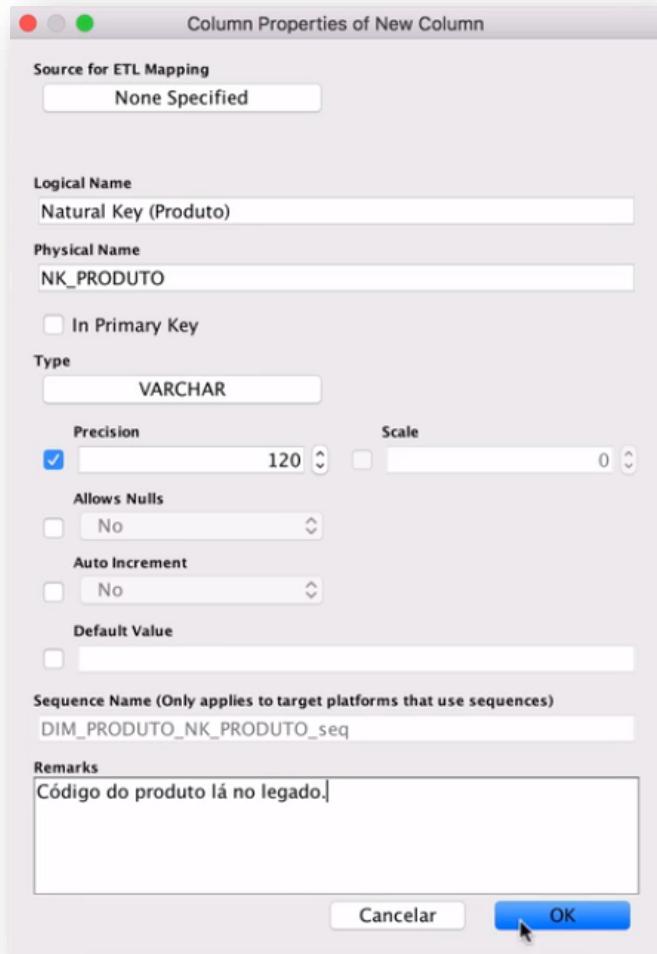


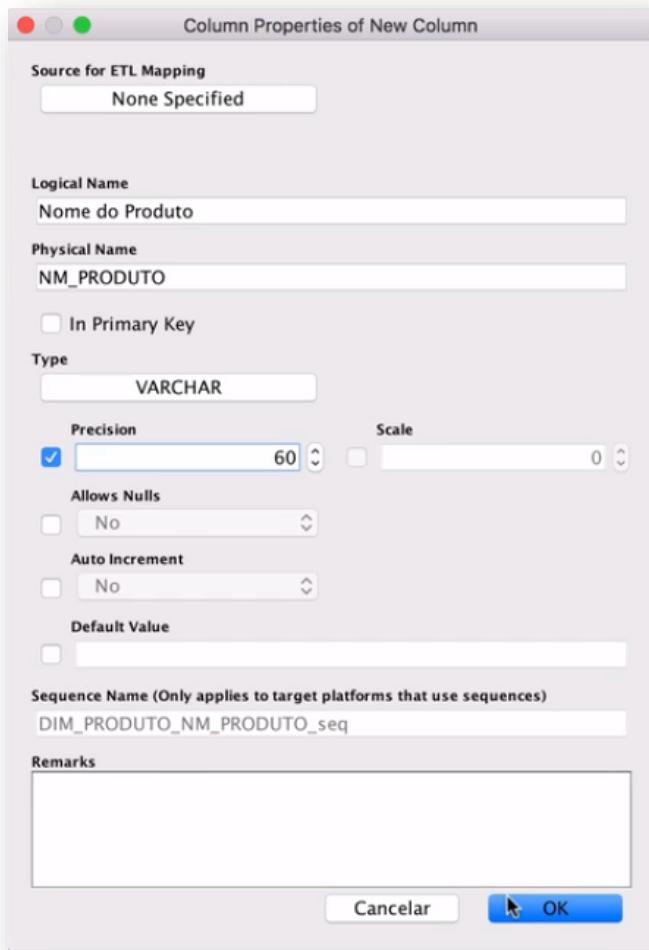
Essa dimensão tem 4 campos:

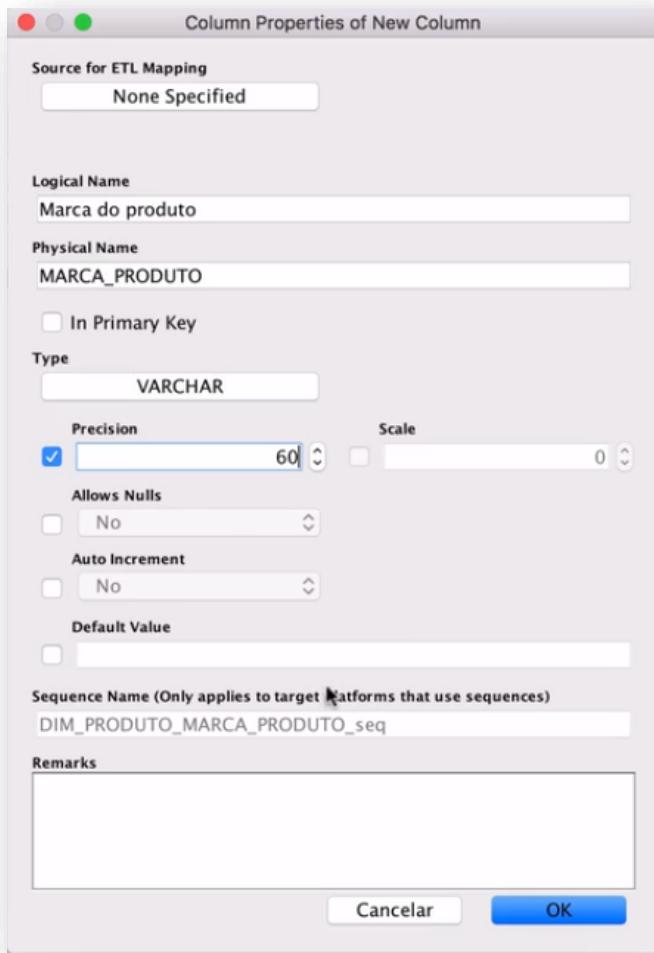
- surrogate key;
- natural key;
- nome;
- marca.

A criação das colunas vai ser igual à dimensão de cliente.









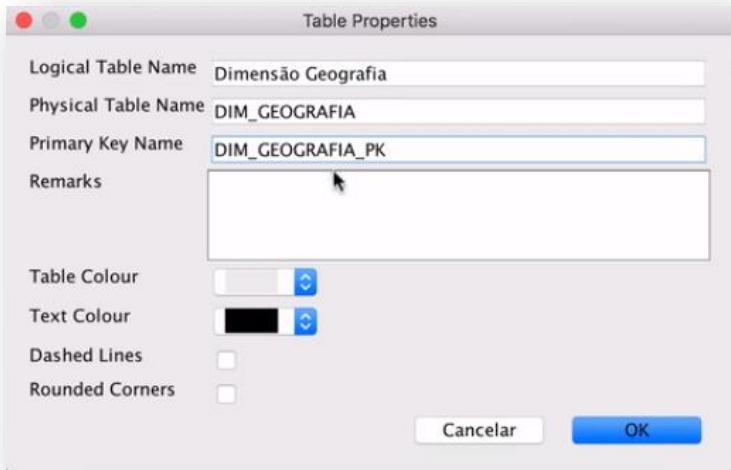
Pronto, dimensão de produto criada.

DIM_CLIENTE
SK_CLIENTE: INTEGER NOT NULL [PK]
NK_CLIENTE: VARCHAR(60) NOT NULL
NM_CLIENTE: VARCHAR(60) NOT NULL
EMAIL_CLIENTE: VARCHAR(60) NOT NULL

DIM_PRODUTO
SK_PRODUTO: INTEGER NOT NULL [PK]
NK_PRODUTO: VARCHAR(120) NOT NULL
NM_PRODUTO: VARCHAR(60) NOT NULL
MARCA_PRODUTO: VARCHAR(60) NOT NULL

Aqui no produto também tem uma hierarquia, que eu vou te explicar mais para frente como que funciona.

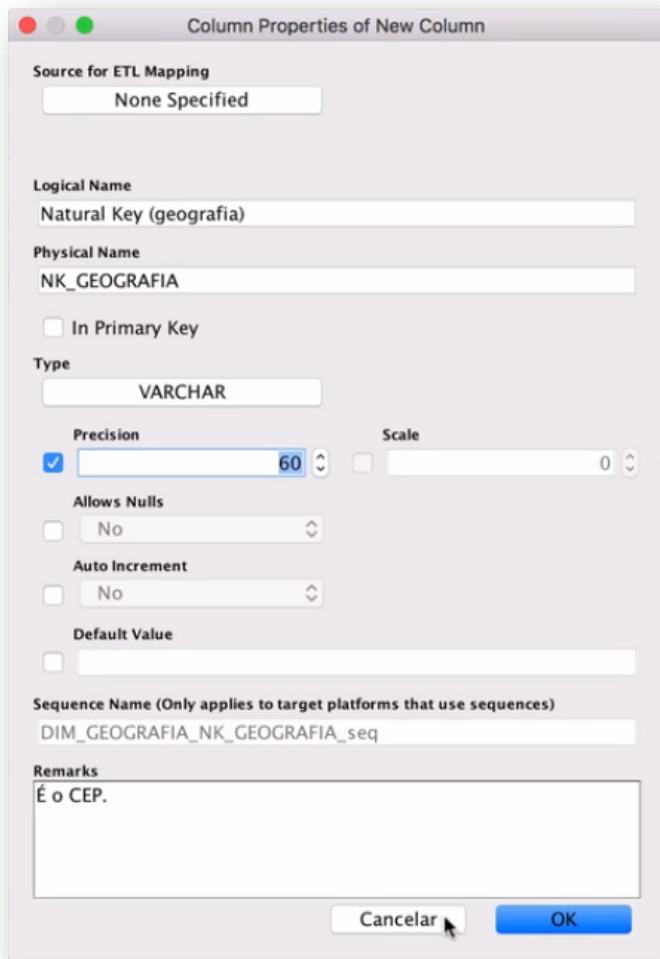
Agora a dimensão de geografia.

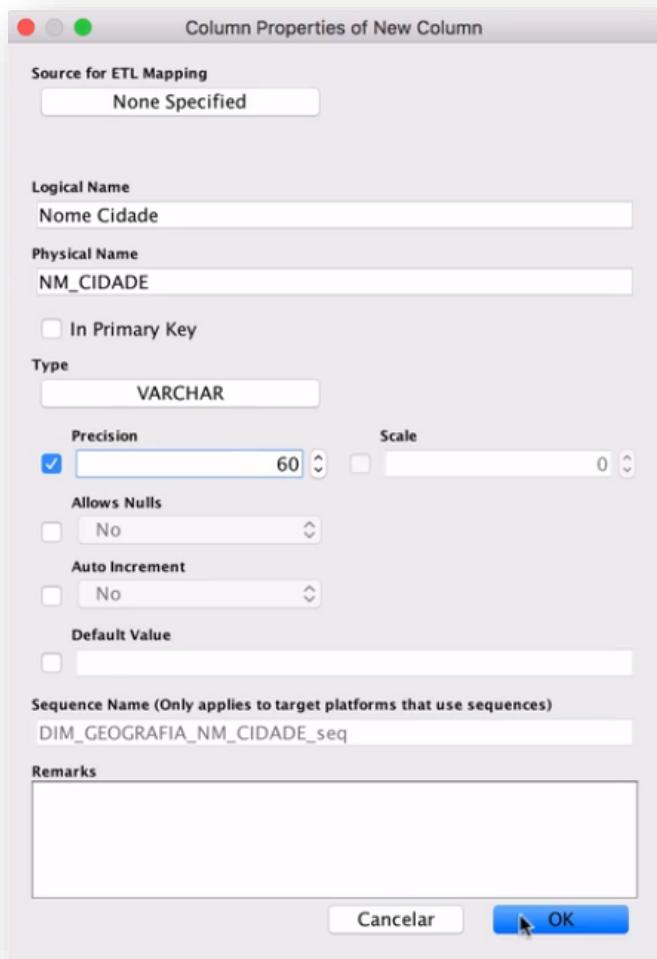


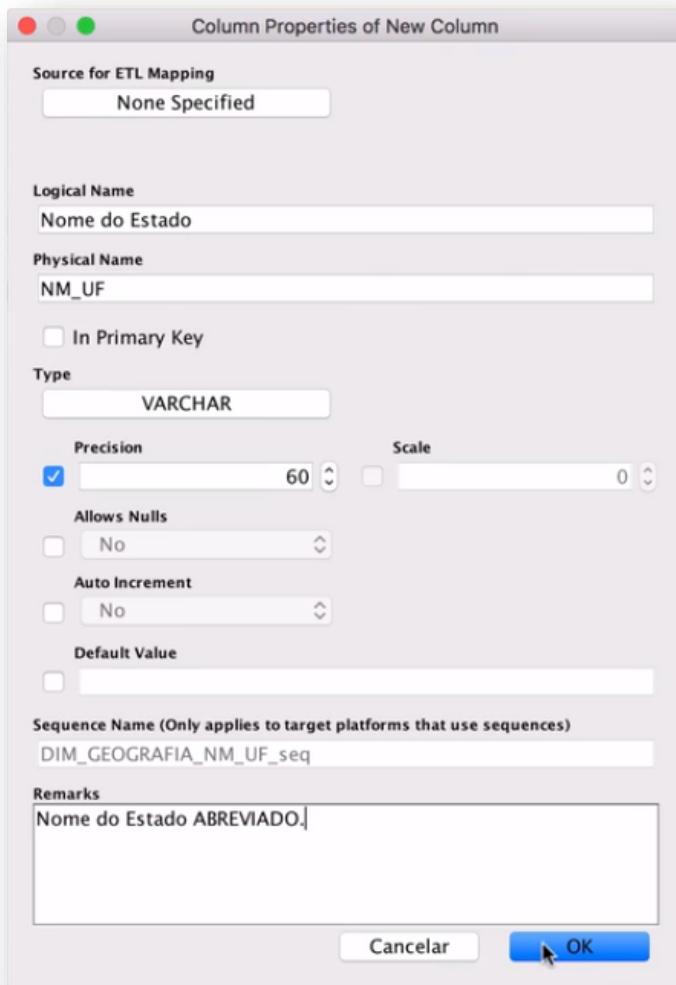
A dimensão de geografia tem os campos:

- surrogate key;
- natural key;
- cidade;
- estado.









E agora a dimensão geografia está criada.

DIM_CLIENTE
SK_CLIENTE: INTEGER NOT NULL [PK]
NK_CLIENTE: VARCHAR(60) NOT NULL
NM_CLIENTE: VARCHAR(60) NOT NULL
EMAIL_CLIENTE: VARCHAR(60) NOT NULL

DIM_PRODUTO
SK_PRODUTO: INTEGER NOT NULL [PK]
NK_PRODUTO: VARCHAR(120) NOT NULL
NM_PRODUTO: VARCHAR(60) NOT NULL
MARCA_PRODUTO: VARCHAR(60) NOT NULL

DIM_GEOGRAFIA
SK_GEOGRAFIA: INTEGER NOT NULL [PK]
NK_GEOGRAFIA: VARCHAR(60) NOT NULL
NM_CIDADE: VARCHAR(60) NOT NULL
NM_UF: VARCHAR(60) NOT NULL

Com isso, só fica faltando a dimensão de tempo. Mas essa dimensão eu vou explicar mais para frente, separei um capítulo inteiro só para mostrar a criação dela, porque quero que você veja em detalhes como que funciona.

Algumas observações sobre a criação dessas dimensões:

O nome da dimensão aparece em todas as colunas, como em *nome do cliente* ao invés de só *nome*, porque na hora que você for trabalhar isso em uma Analysis OLAP, na criação de um dashboard ou um relatório, pode ser que você queira trazer algumas colunas do banco de dados e vai ficar confuso vendo tudo junto, vai ter 10 colunas de nome e você não vai saber do que é cada uma delas.

No nome da natural key, você também pode utilizar o nome do campo da origem para ajudar a tomada de decisão. Essa opção eu vou deixar para você ver como o seu negócio vai se comportar melhor.

Você pode usar o nome *natural key* para deixar bem claro que campo é aquele e na hora de fazer os relatórios, dashboards e análises em cima desses dados, você renomeia para o que é realmente aquele campo. É uma boa prática que eu gosto.

Digamos que hoje você utilize o CEP para ser a sua natural key, e com o passar do tempo, inventaram um novo código porque mudaram a forma de identificar a localização. Se a sua coluna se chama CEP, você acabou de estragar a sua natural key, e se tentar renomear a coluna, todos os relatórios que foram construídos em cima dela vão se perder.

PASSO 3: CRIAR FATOS E MÉTRICAS

Agora que você já criou as dimensões, é hora de criar a fato. E o processo é exatamente igual ao que acabamos de fazer nas dimensões.

Você vai clicar em *Nova Tabela* para criar uma tabela.

Aqui você vai precisar colocar um nome no campo de *primary key*, mesmo não utilizando ela, para poder continuar o desenho, é uma coisa da ferramenta mesmo.



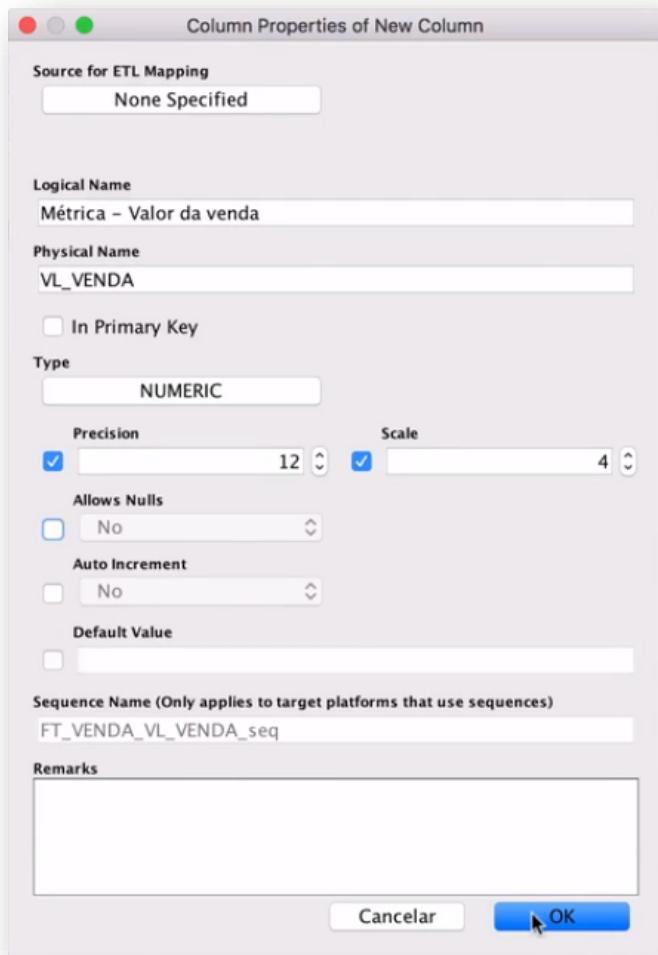
Com a tabela criada, é hora de fazer as colunas. A gente vai começar pelas métricas, que são:

- valor da venda;
- quantidade vendida.

Para criar as colunas das métricas, é só ir em *Inserir Coluna* e preencher a janelinha.

No valor da venda, na parte da escala, nós estamos acostumados a ver apenas 2 casas decimais, por exemplo: R\$200,00.

Mas tem alguns arquitetos que deixam a escala em 4 para melhorar a precisão e não ter nenhum tipo de erro na hora da carga, fazendo assim, na hora que for criar os reports, você precisa dizer que é para mostrar apenas 2 casas decimais, e a ferramenta se encarrega disso.





E agora você já está com todas as tabelas criadas, tanto a fato como as dimensões.

DIM_CLIENTE
SK_CLIENTE: INTEGER NOT NULL [PK]
NK_CLIENTE: VARCHAR(60) NOT NULL
NM_CLIENTE: VARCHAR(60) NOT NULL
EMAIL_CLIENTE: VARCHAR(60) NOT NULL

DIM_PRODUTO
SK_PRODUTO: INTEGER NOT NULL [PK]
NK_PRODUTO: VARCHAR(120) NOT NULL
NM_PRODUTO: VARCHAR(60) NOT NULL
MARCA_PRODUTO: VARCHAR(60) NOT NULL

FT_VENDA
VL_VENDA: NUMERIC(12, 4) NOT NULL
QTD_VENDIDA: INTEGER NOT NULL

DIM_GEOGRAFIA
SK_GEOGRAFIA: INTEGER NOT NULL [PK]
NK_GEOGRAFIA: VARCHAR(60) NOT NULL
NM_CIDADE: VARCHAR(60) NOT NULL
NM_UF: VARCHAR(60) NOT NULL

PASSO 4: FAZER JOINS ENTRE FATOS E DIMENSÕES

Com todas as tabelas criadas, falta só fazer os joins entre as tabelas e criar as foreign keys na fato.

Você vai precisar de uma foreign key de cada uma das dimensões, que são:

- cliente;
- produto;
- geografia.

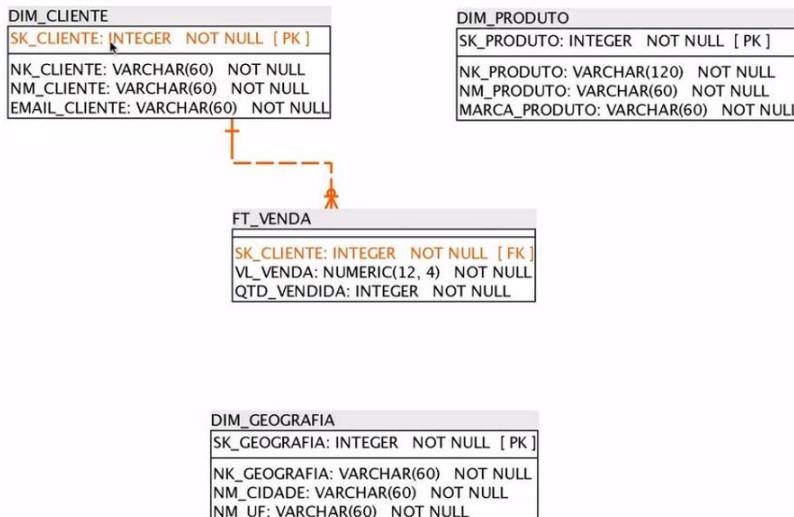
No lado direito, tem dois botões para fazer joins.

O primeiro é o *Novo relacionamento não identificado*, que faz com que a fato fique com uma foreign key. E o segundo é o *Novo relacionamento identificado*, que faz com que a fato fique com uma primary foreign key, ou seja, ele vai pegar todas as foreign keys que a fato gerar, e dessa combinação vai criar uma foreign key. Mas nesse exato momento você não precisa disso.

Então para fechar o seu modelo estrela, você vai clicar na primeira opção de join, que é o *Novo relacionamento não identificado*.

Para fazer o relacionamento, você vai clicar primeiro na primary key da dimensão, e depois na tabela fato.

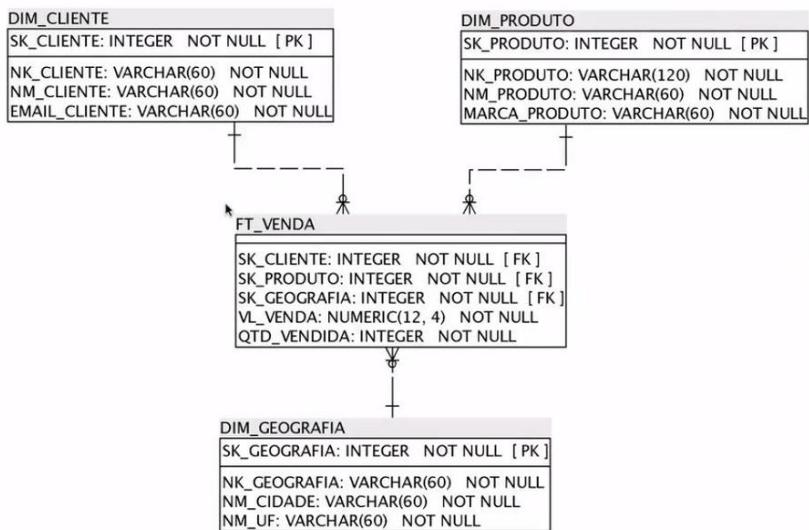
E está feito o join, a ferramenta já vai criar a foreign key na fato para você.



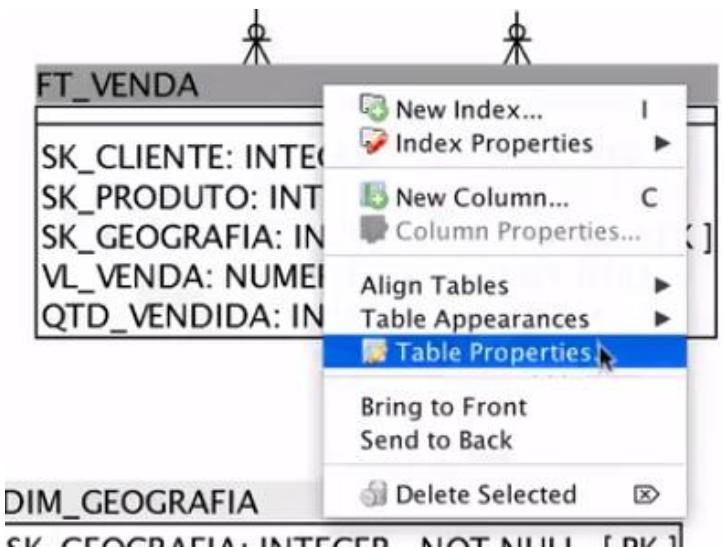
Esses tracinhos que ele cria significa que você tem vários registros na fato para uma linha da dimensão.

É só uma linha da dimensão porque a surrogate key é sempre única, e ela pode representar um ou mais registros na fato.

E agora é só fazer o mesmo com as outras dimensões e você está com a modelagem do Star Schema pronta.



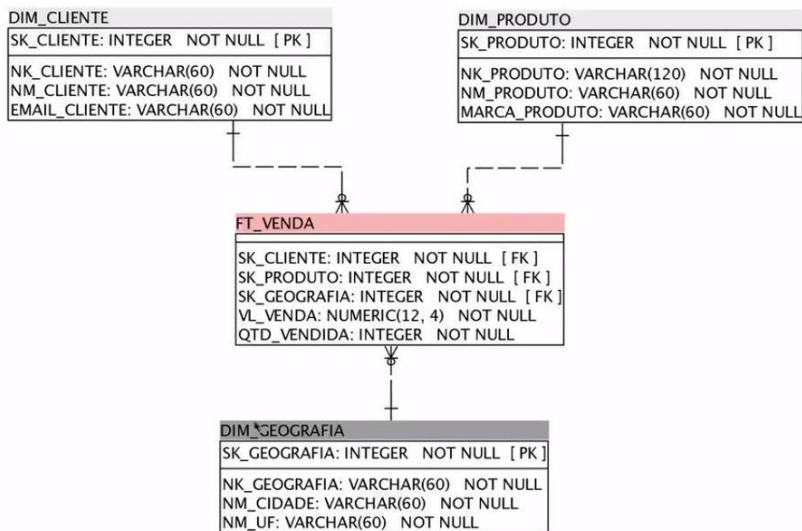
Para trocar a cor da tabela fato, que é uma coisa que eu gosto de fazer, você clica com o botão direito na tabela e vai em *Propriedades da Tabela...*



E aí você pode colocar a cor que achar melhor em *Cor da tabela*.

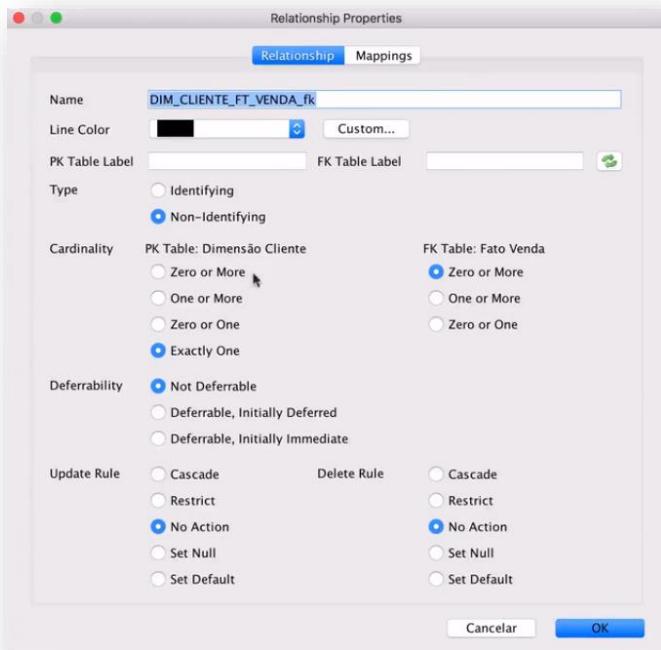


E pronto, a fato está com a cor diferente.



Se você quiser editar o join, pode apertar em cima da linha do relacionamento e clicar em *Alterar relacionamento*. Em *Cor da Linha* você também pode mudar a cor do relacionamento para deixar mais destacado.





Isso é muito importante para ficar mais fácil de bater o olho e entender o que está acontecendo.

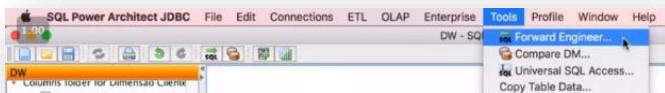
E esse é o nosso design, a gente acabou de fazer nosso modelo Star Schema. Aqui não tem mágica, é trabalho mesmo, então você tem que fazer, não espera ler todo o livro para só depois começar, porque você só vai aprender fazendo.

PASSO 5: CARREGAR A MODELAGEM NO BANCO DE DADOS

Se você já fez tudo que eu ensinei a fazer, então você já está com o seu primeiro modelo de Data Warehouse pronto.

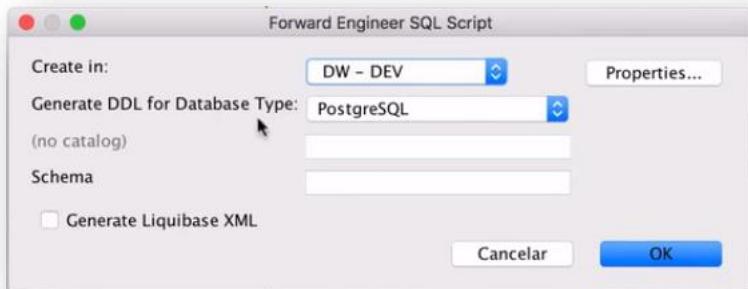
Tudo que você precisa agora é estar com o banco de dados conectado no SQL Power Architect, para a gente pegar esse modelo e inserir nele.

Para isso, você vai vir em *Ferramentas > Engenharia Reversa...*



Ele vai te dizer:

"Beleza, você quer criar isso. Mas onde?"



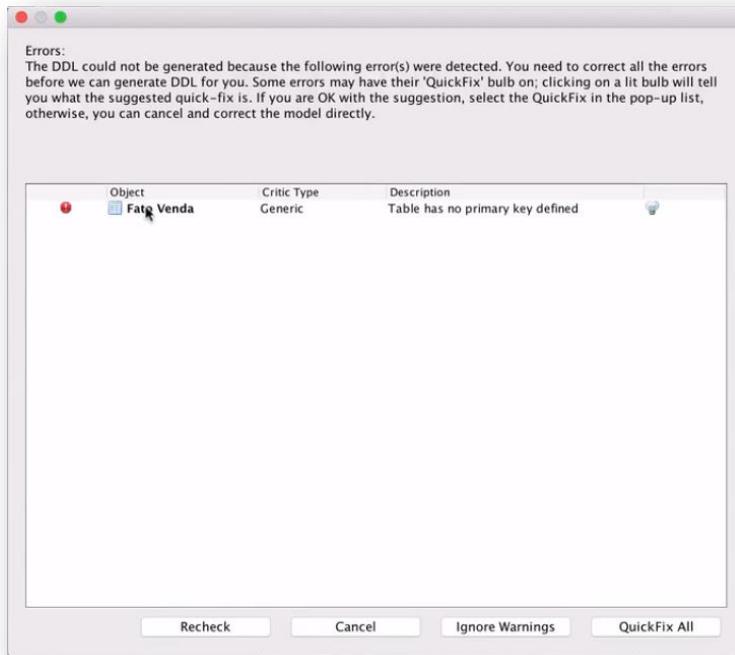
Aqui você preenche as configurações:

Criar em - você seleciona a conexão que você criou.

Gerar o DDL para o banco de dados - o DDL é um script para carregar no banco de dados, que vai criar as tabelas, as métricas, as dimensões e fatos, índices e tudo mais, você só seleciona o tipo do banco e ele já entende qual a sintaxe que deve ser usada.

Schema - você pode definir o schema de banco de dados que está usando.

Depois você dá *OK* e vai dar um erro, é normal.



Ele vai dizer que a fato está sem primary key, e realmente ela não tem esse campo. Você pode só ignorar esse erro clicando em *Ignorar Advertências*.

E agora ele vai criar todo o código do banco para você.

The screenshot shows a 'Preview SQL Script' window with the following content:

```
CREATE SEQUENCE dim_geografia_sk_geografia_seq;

CREATE TABLE DIM_GEOGRAFIA (
    SK_GEOGRAFIA INTEGER NOT NULL DEFAULT
nextval('dim_geografia_sk_geografia_seq'),
    NK_GEOGRAFIA VARCHAR(60) NOT NULL,
    NM_CIDADE VARCHAR(60) NOT NULL,
    NM_UF VARCHAR(60) NOT NULL,
    CONSTRAINT dim_geografia_pk PRIMARY KEY
(SK_GEOGRAFIA)
);
COMMENT ON COLUMN DIM_GEOGRAFIA.NK_GEOGRAFIA IS 'É o
CEP. ';
COMMENT ON COLUMN DIM_GEOGRAFIA.NM_UF IS 'Nome do
Estado ABREVIADO.';

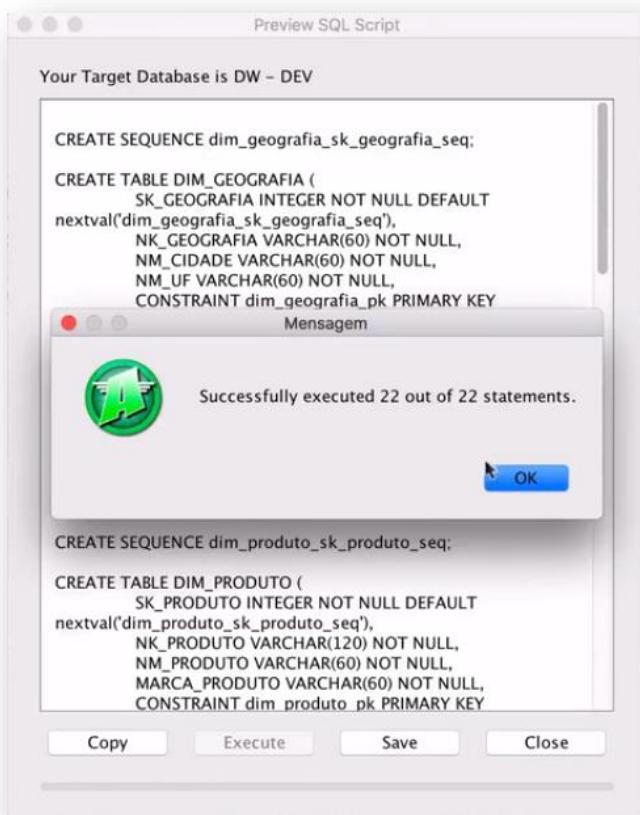
ALTER SEQUENCE dim_geografia_sk_geografia_seq OWNED BY
DIM_GEOGRAFIA.SK_GEOGRAFIA;

CREATE SEQUENCE dim_produto_sk_produto_seq;

CREATE TABLE DIM_PRODUTO (
    SK_PRODUTO INTEGER NOT NULL DEFAULT
nextval('dim_produto_sk_produto_seq'),
    NK_PRODUTO VARCHAR(120) NOT NULL,
    NM_PRODUTO VARCHAR(60) NOT NULL,
    MARCA_PRODUTO VARCHAR(60) NOT NULL,
    CONSTRAINT dim_produto_pk PRIMARY KEY
```

At the bottom of the window are four buttons: 'Copy', 'Execute' (highlighted in blue), 'Save', and 'Close'.

Você pode copiar esse código e executar no banco de dados direto, ou simplesmente clicar em *Execute* que ele já roda o código para você.



The screenshot shows the 'Preview SQL Script' window from SQL Power Architect. At the top, it says 'Your Target Database is DW - DEV'. Below that is a code editor containing two SQL scripts. The first script creates a sequence and a table for geographical dimensions:

```
CREATE SEQUENCE dim_geografia_sk_geografia_seq;
CREATE TABLE DIM_GEOGRAFIA (
    SK_GEOGRAFIA INTEGER NOT NULL DEFAULT nextval('dim_geografia_sk_geografia_seq'),
    NK_GEOGRAFIA VARCHAR(60) NOT NULL,
    NM_CIDADE VARCHAR(60) NOT NULL,
    NM_UF VARCHAR(60) NOT NULL,
    CONSTRAINT dim_geografia_pk PRIMARY KEY
```

The second script creates a sequence and a table for product dimensions:

```
CREATE SEQUENCE dim_produto_sk_produto_seq;
CREATE TABLE DIM_PRODUTO (
    SK_PRODUTO INTEGER NOT NULL DEFAULT nextval('dim_produto_sk_produto_seq'),
    NK_PRODUTO VARCHAR(120) NOT NULL,
    NM_PRODUTO VARCHAR(60) NOT NULL,
    MARCA_PRODUTO VARCHAR(60) NOT NULL,
    CONSTRAINT dim_produto_pk PRIMARY KEY
```

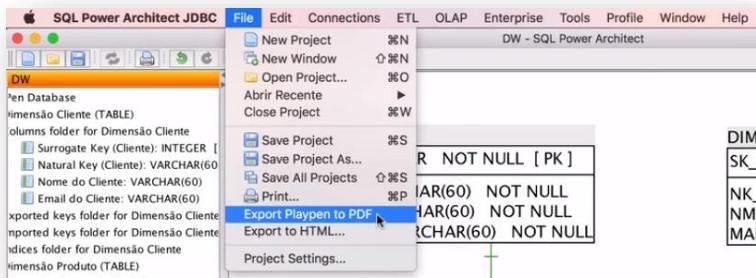
Below the code editor, there is a message box titled 'Mensagem' with a green icon of a person with a star. It says 'Successfully executed 22 out of 22 statements.' with an 'OK' button. At the bottom of the main window are buttons for 'Copy', 'Execute', 'Save', and 'Close'.

E se deu essa mensagem, é porque está tudo certo.

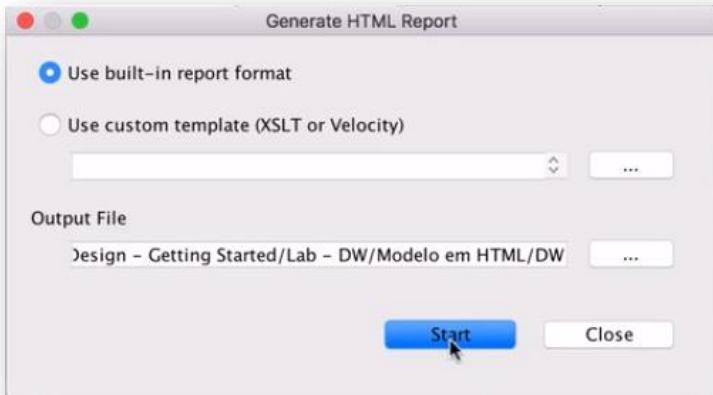
Agora, se você for olhar o seu banco, vai ver que ele já está com todas as tabelas que a gente criou.

E se você precisar de alguma documentação para apresentar, não precisa ficar criando PDF, se você sempre usar essa ferramenta para gerenciar o seu Data Warehouse, ele faz isso para você.

Clique em *Arquivo > Exportar para HTML...*



Depois você escolhe onde esse arquivo deve ser salvo e clica em *Iniciar*.



Agora você vai na pasta que escolheu para salvar o arquivo e abre ele no navegador.

The screenshot shows the SQL Power Architect application window. At the top, there's a toolbar with various icons. Below the toolbar, the title bar says "DW". The main area has a header "List of tables" followed by a bulleted list of tables:

- Dimensão Cliente
- Dimensão Geografia
- Dimensão Produto
- Fato Venda

Below this, a section titled "Dimensão Cliente (Physical Name: DIM_CLIENTE)" is shown. It contains a brief description: "Dimensão para cadastro dos clientes." and a note: "Natural Key (Cliente) é o cpf do cliente." There's also a note about the primary key being a sequence.

A table definition for the "Dimensão Cliente" table is displayed:

Logical Column Name	Physical Column Name	Type	PK	Nullable
Surrogate Key (Cliente) (PK) Chave Artificial.	SK_CLIENTE	INTEGER	PK	NOT NULL
Sequence.				
Não reutilizar.				
Natural Key (Cliente) é o cpf do cliente.	NK_CLIENTE	VARCHAR(60)		NOT NULL
Nome do Cliente Nome completo.	NM_CLIENTE	VARCHAR(60)		NOT NULL
Email do Cliente	EMAIL_CLIENTE	VARCHAR(60)		NOT NULL

Below the table, under "Referenced By", it lists "Fato Venda referencing (Surrogate Key (Cliente))".

Aqui está a documentação que você precisa.

E para compartilhar isso, você pode incorporar esse HTML em um Word ou colocar no Google Drive também.

DESIGN PATTERNS

Para os padrões de nomenclatura das tabelas e colunas, você pode usar algum modelo seu ou da empresa que você trabalha, não existe uma regra. Vou deixar aqui alguns que eu utilizo nos meus projetos de Data Warehouse.

Nomenclaturas

- **surrogate key** - SK_
- **natural key** - NK_
- **nomes** - NM_
- **descrições** - DS_
- **valores monetários** - VL_
- **quantidade** - QTD_
- **datas** - DT_
- **dimensão** - DIM_
- **fato** - FT_

- **index** - IDX_01
- **sequences** - _SEQ

Tipos de dados

- **SK** - integer
- **NK** - varchar (20)
- **texto** - varchar (60)
- **monetário** - numeric (12, 4)
- **alta densidade** - UF: char (2)

OBS: no tipo de dados de texto - varchar, acima de 60, aumentar de 10 em 10 até 100. De 50 em 50 até 500. E acima disso, de 100 em 100.

TIPOS DE MÉTRICAS

Vou explicar aqui quais são as diferenças entre os tipos de métricas, e depois nós voltamos para o nosso modelo Star Schema para fazer alterações conforme você for aprendendo a melhorar ele.

MÉTRICA ADITIVA

São as métricas que permitem operações matemáticas como soma e subtração por todas as dimensões. Por exemplo, eu posso dividir quantidade de vendas por tempo, por produto, por fornecedor.

Dentro da fato há diversas linhas, e as métricas aditivas devem poder somar todas elas. É óbvio, todas podem ser somadas se você quiser, mas ela precisa fazer sentido.

Um exemplo:

	A	B	C	D	E	F	G	H	I	J
	Dimensão Produto									
1	SK	NK	Nome	Preço unitário						
2	1	2323	Coca-Cola	3,56						
3	2	1111	Pizza	23,60						
4	3	4545	café	7,60						
5						
6						
7										
8										
9										
10										
11	Dimensão Data									
12	SK	NK	Ano	Mês						
13	10122016	10/12/2016	2016	Dezembro						
14	11122016	11/12/2016	2016	Dezembro						
15	12122016	12/12/2016	2016	Dezembro						
16						
17										
18										

Fato Venda			
PK	PK	Métrica aditiva	Métrica Derivada
DIM_DATA	DIM_PRODUTO	Quantidade da Venda	Valor da Venda
10122016	1	200	710,00
11122016	1	300	7080,00
12122016	1	200	1560,00
		700	9.350

Nessa tabela eu tenho a quantidade da venda, ou seja, a quantidade de coisas que foram vendidas.

Olhando a primeira linha da fato, pelas chaves você pode ver nas dimensões que aquele 200 é do produto Coca-Cola e da data 10/12/2016.

Nas métricas aditivas, o valor da métrica é representativo em todos os cruzamentos:

Quanto foi vendido no dia 10/12/2016? 200.

Quanto foi vendido de Coca-Cola? 200.

Nesse caso, se você somar as 3 linhas da fato, vai ter 700. Então se quiser saber quanto foi vendido no total em que há registros, você sabe, foi 700. E se quiser saber quantas Coca-Colas foram vendidas neste mês, 700 também.

Alguns exemplos de métricas aditivas:

- quantidade de vendas;

- quantidade de colaboradores;
- quantidade de demissões;
- quantidade de admissões.

Todas essas métricas podem ser somadas independentemente de onde estiverem cruzando. Ela tem que fazer um cruzamento completo e perfeito na linha da fato, então a métrica precisa fazer sentido com cada uma das dimensões sozinhas.

É a métrica mais simples que conhecemos.

MÉTRICA DERIVADA

É uma métrica calculada, uma métrica que você calcula para ter um segundo número. Esse cálculo é sempre em cima de métricas que já estão na fato, não no que está no legado.

São métricas que já estavam na fato e são calculadas, criando uma nova métrica, que chamamos de derivada.

Elas podem ser armazenadas diretamente na fato ou calculadas em tempo de execução nos cubos.

A diferença básica entre armazenar no Data Warehouse ou deixar para ser calculada em tempo de execução, é o seguinte:

No Data Warehouse quem fará esse cálculo é o time de ETL, ou seja, na hora da integração, se aquela métrica for derivada, já vai ser calculada antes de ser armazenada.

Quando você salva as métricas bases, na hora de fazer as análises, o cálculo é feito em tempo de execução.

Qual a diferença na prática?

O cálculo vai utilizar recursos computacionais, de servidores e tudo mais. O cubo vai ter que calcular isso para você. Quando está armazenada no Data Warehouse, esse procedimento já foi realizado.

E qual o melhor?

Não tem resposta certa, porque vai depender da estratégia de como você quer ver a informação e de como chegar nela.

Por exemplo, eu já peguei casos que estava muito difícil para a ferramenta calcular em tempo de execução, demorava muito porque era um cálculo bem complexo.

O que eu fiz? Fiz o cálculo no ETL. Antes de dar a carga final da fato, peguei a métrica que já tinha sido carregada, que era uma aditiva, fiz o cálculo que tinha que ser feito e inseri de novo, aí ela virou uma métrica derivada e aumentou muito a velocidade das consultas.

Uma coisa é você esperar 5 minutos no ETL e outra coisa é o usuário clicar no relatório e ele levar 5 minutos para abrir. Esse tipo de coisa você não pode deixar acontecer.

Então coloco tudo no Data Warehouse? Não.

Depende da estratégia, por isso que eu explico sempre o processo, para você saber o que fazer. Nesse caso que eu citei foi muito bom ter colocado no ETL, mas houveram casos que precisei deixar as métricas para serem calculadas direto na ferramenta, porque eram diversas métricas de negócio.

As métricas derivadas que o negócio demandava eram mais de 30, então dei que elas fossem calculadas na ferramenta, porque a forma como o cliente queria analisar mudava, uma hora ele queria analisar por tempo, outra hora por produto e outra por vendedor.

Quando você chumba uma métrica no Data Warehouse, também está chumbando aquele cruzamento. Quando precisar de um novo valor em cima daquela métrica para um novo cruzamento, vai ter que recalcular toda a fato no ETL.

Então quando são projetos muito complexos, a gente opta sempre por deixar as métricas derivadas para as ferramentas do BI fazer.

Uma coisa muito importante que sempre faz a gente optar por colocar as métricas derivadas na ferramenta de BI é que quando todas as métricas estão disponíveis limpinhas lá no cubo, o usuário de negócio pode fazer um ad-hoc (self-service BI) e o gráfico em que ele está trabalhando vai se movendo conforme ele faz as análises.

Exemplo de métrica derivada:

	A	B	C	D	E	F	G	H	I	J
1		Dimensão Produto								
2	SK	NK	Nome	Preço unitário						
3	1	2323	Coca-Cola	3,55						
4	2	1111	Pizza	23,60						
5	3	4545	café	7,80						
6						
7										
8										
9										
10										
11		Dimensão Data								
12	SK	NK	Ano	Mês						
13	10122016	10/12/2016	2016	Dezembro						
14	11122016	11/12/2016	2016	Dezembro						
15	12122016	12/12/2016	2016	Dezembro						
16						
17										
18										

Valor da Venda = Quantidade da Venda * Preço unitário

			Fato Venda	
FK	FK		Métrica ativa	Métrica Demanda
DIM_DATA	DIM_PRODUTO		Quantidade da Venda	Valor da Venda
10122016	1		200	710,00
11122016	1		300	7080,00
12122016	1		200	1560,00
...	...		700	9 350

Nesse exemplo, a métrica derivada é a multiplicação da quantidade da venda pelo preço unitário que está na dimensão de produto.

MÉTRICA SEMIADITIVA

Para exemplificar, fiz essa tabela de uma fato com saldo de conta corrente.

A	B	C	D	E	F	G	H
1							
2	FATO SALDO CONTA CORRENTE						
3	FK	FK		Métrica aditiva	Métrica aditiva	Métrica Semi-Aditiva	Métrica Não-Aditiva
4	DATA	CLIENTE	AGÊNCIA	ENTRADA	SAÍDA	SALDO	diff %
5	01/03/2017	PITON	ITAU-SP01	1.500,00	300,00	1.200,00	500,00%
6	01/03/2017	PITON	ITAU-SP01		400,00	800,00	0,00%
7	01/03/2017	PITON	ITAU-SP01		200,00	600,00	0,00%
8	01/03/2017	ROBSON	ITAU-SP01	2.500,00	700,00	1.800,00	357,14%
9	01/03/2017	MARCOS	ITAU-SP01	3.120,00	300,00	2.820,00	1040,00%
10	01/03/2017	DORIVAL	ITAU-SP01	7.500,00	300,00	7.200,00	2500,00%
11				14.620,00	2.200,00	14.420,00	4397,14%
12							
13							
14							
15							
16							
17							
18							
19							
20	Saldo de estoque e saldo bancário são métricas semi-aditivas bem comuns, porque são aditivas em todas as dimensões, exceto no tempo.						

Uma visão geral sobre essa fato: temos a dimensão de tempo, de cliente e de agência. Também temos algumas métricas: entrada, saída e o saldo.

A entrada é uma métrica aditiva, ela pode somar todas as linhas, totalizando 14.620,00 de entradas. E podemos cruzar esse dado com todas as dimensões. A saída é a mesma coisa, ela pode ser somada e cruzada com todas as dimensões. E a métrica de saldo, que nesse caso é a entrada menos a saída.

Por exemplo:

No dia 1º de março de 2017, na conta do Piton, na agência ITAU-SP01, entrou R\$1.500,00 e saiu R\$300,00, ficando com um saldo de R\$1.200,00.

Nesse mesmo dia, na mesma conta, não entrou nada e saiu R\$400,00. O saldo ficou R\$800,00. Porque é o saldo anterior menos a saída.

De novo, no mesmo dia, saiu mais R\$200,00 sem entrar nada, ficando com o saldo de R\$600,00.

Nesse mesmo dia também aconteceram transações de outras pessoas. Na conta do Robson, entrou R\$2.500,00 e saiu R\$700,00, ficando com o saldo de R\$1.800,00.

Se somarmos todas as linhas de saldo, vamos ter um resultado de 14.420,00, mas isso não faz sentido, porque no mesmo dia, tivemos várias transações em uma mesma conta, e o saldo do Piton não é a soma de todos os valores. Se somarmos $1.200 + 800 + 600$, vamos ter 2.600, enquanto o saldo real é 600.

Então o que faz sentido somar? A última movimentação de cada conta naquele dia, aí vou ter 12.420,00.

Resumidamente, o que isso quer dizer? Que métrica semiaditiva pode ser somada por todas as dimensões exceto a tempo.

Você só vai conseguir somar pela tempo se colocar um filtro que diga que ele só pegue o último registro.

Saldo de estoque e saldo bancário, quando representado de forma monetária, são métricas semiaditivas bem comuns, porque são aditivas em todas as dimensões, exceto na tempo.

MÉTRICA NÃO ADITIVA

São métricas tipo percentual, algum cálculo feito em tempo de execução, que não podem ser somadas por nenhuma dimensão. O ideal é salvar as métricas que levam àquela não aditiva e deixar que ela seja calculada na ferramenta de BI.

Um exemplo:

A	B	C	D	E	F	G	H
1	FATO SALDO CONTA CORRENTE						2
2	FK	FK		Métrica aditiva	Métrica aditiva	Métrica Semi-Aditiva	Métrica não-Aditiva
3	DATA	CLIENTE	AGÊNCIA	ENTRADA	SAÍDA	SALDO	
4	01/03/2017	PITON	ITAU-SP01	1.500,00	300,00	1.200,00	500,00%
5	01/03/2017	PITON	ITAU-SP01		400,00	800,00	0,00%
6	01/03/2017	PITON	ITAU-SP01		200,00	600,00	0,00%
7	01/03/2017	ROBSON	ITAU-SP01	2.500,00	700,00	1.800,00	357,14%
8	01/03/2017	MARCOS	ITAU-SP01	3.120,00	300,00	2.820,00	1040,00%
9	01/03/2017	DORIVAL	ITAU-SP01	7.500,00	300,00	7.200,00	2500,00%
10					14.620,00	2.200,00	14.420,00
11							4397,14%
12							
13							
14							
15							
16							
17							
18							
19							
20	Saldo de estoque e saldo bancário são métricas semi-aditivas bem comuns, porque são aditivas em todas as dimensões, exceto no tempo.						

No saldo temos a diferença numérica, enquanto na métrica não aditiva, temos a diferença percentual. E qual a conta? A entrada dividida pela saída.

Que leitura fazemos aqui? A entrada passou 500% da saída.

E eu posso somar percentuais? Não. Se eu somar as linhas vou ter 4397,14%, e isso não faz sentido, é 4397,14% de quê?

E é por causa dessas somas e cruzamentos que você pode ou não fazer que é importante entender cada um dos tipos, saber diferenciar eles e principalmente, como usar.

Pegando o exemplo do saldo, é uma métrica muito fácil de ser erroneamente tratada como uma aditiva, o que vai alterar os resultados e afetar a tomada de decisão.

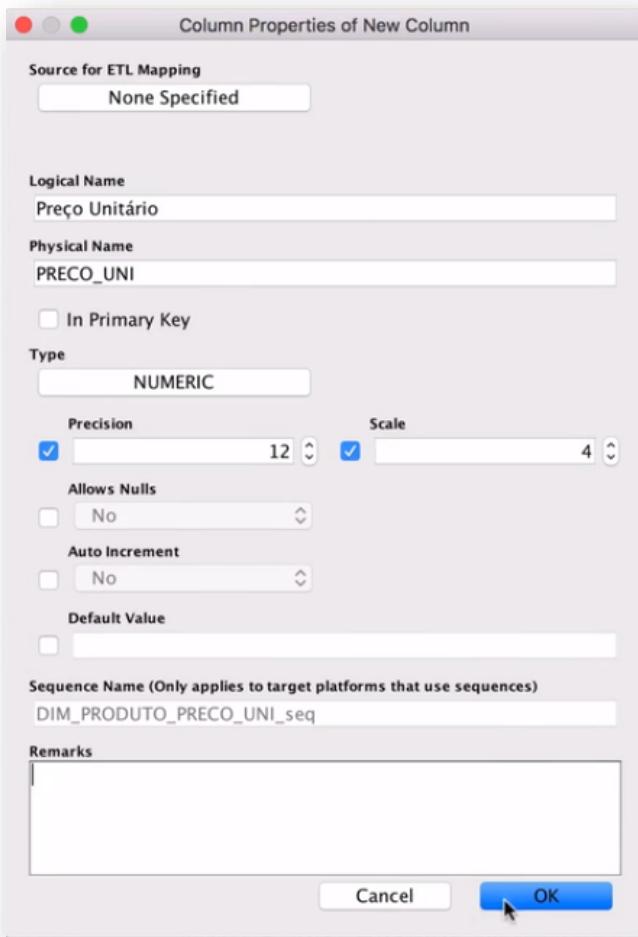
PASSO A PASSO: COMO FAZER MÉTRICAS ADITIVAS E DERIVADAS

Agora quero fazer com você alterações no nosso Data Warehouse para aplicar os tipos de métricas que acabamos de ver.

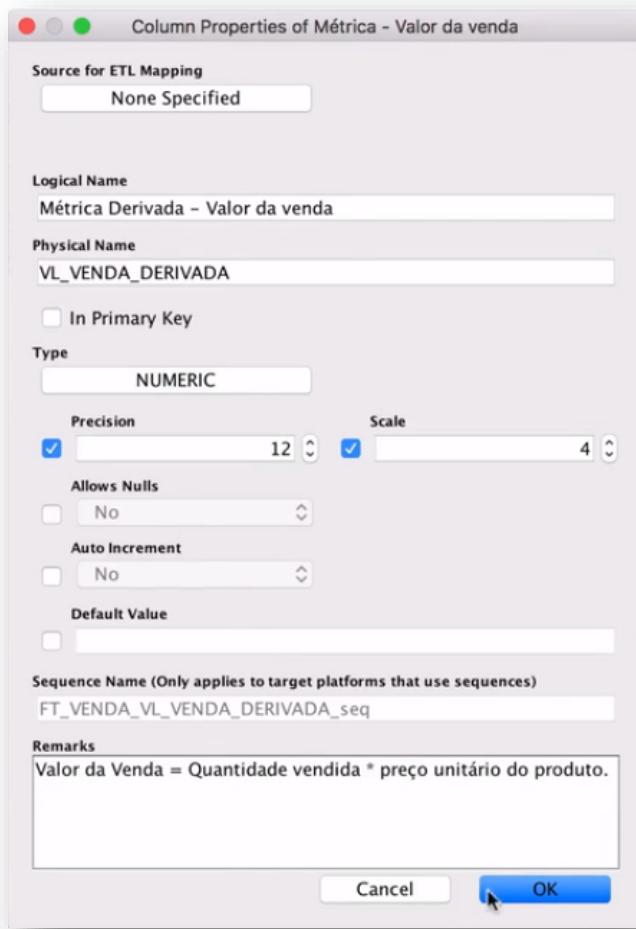
O que você vai fazer:

- adicionar uma coluna na dimensão de produto que vai se chamar *preço unitário* para armazenar o preço de cada produto;
- adicionar uma anotação na métrica *valor de venda* para saber como que chegamos naquele valor.

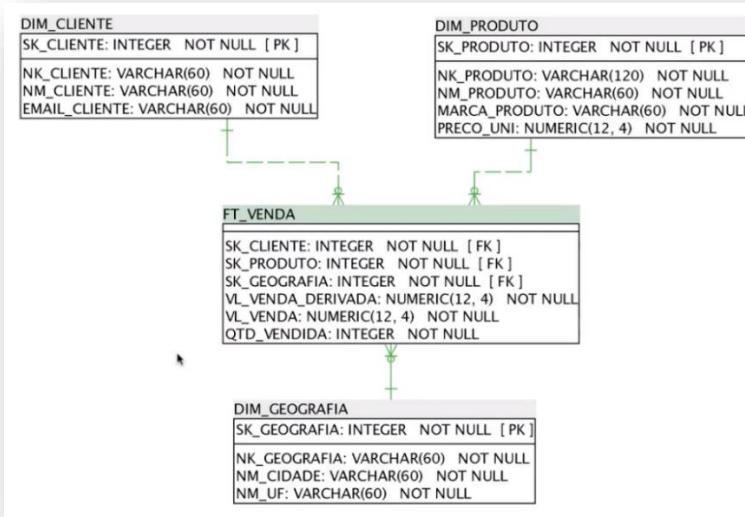
No Power Architect, você vai na dimensão de produto e adiciona uma nova coluna *preço unitário*.



Na fato, a gente já tem o *valor da venda*, que é a métrica derivada. Mas eu gosto de colocar no *logical name* que é uma métrica derivada.



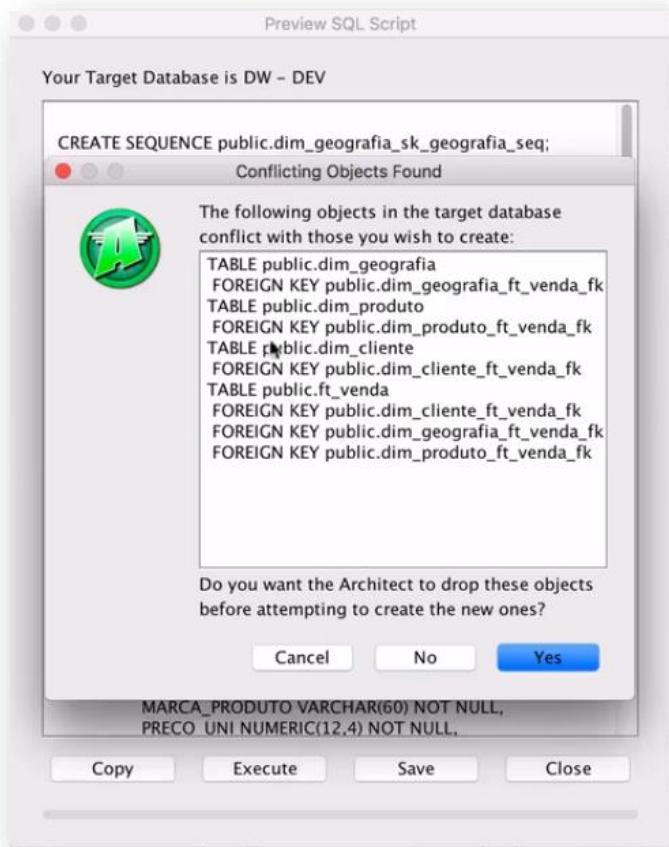
E agora as alterações estão prontas.



Agora você precisa salvar o projeto e persistir no banco de dados. Para fazer isso, você clica em *SQL*, porque ele já está com a sua conexão configurada.



O que vai acontecer agora é que já existe esses objetos no banco de dados, então ele vai te avisar isso.



Ele vai perguntar se você quer fazer um drop para criar esses objetos novamente. E a gente diz que sim, é só apertar no Yes.

E pronto, ele vai dropar o que tinha duplicado no banco e criar de novo.

Se você entrar no banco agora, as alterações já vão estar lá também.

TIPOS DE DIMENSÕES

Aqui eu vou cobrir com você os 3 principais tipos de dimensões que você precisa saber nesse exato momento para que consiga entregar seus projetos de Data Warehouse:

- dimensão hierárquica pai-filho;
- dimensão de tempo;
- dimensão degenerada.

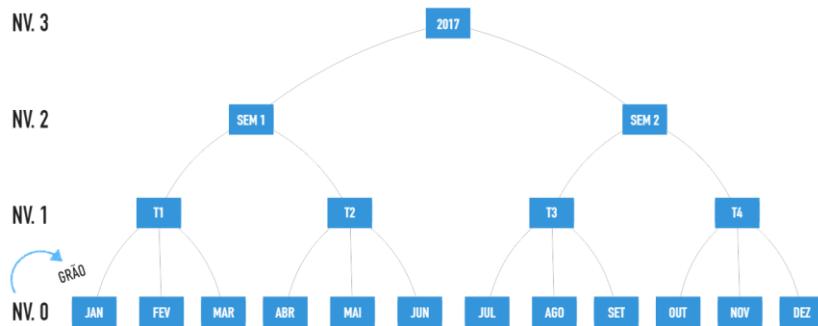
Vamos passar primeiro pelos conceitos e depois entrar no passo a passo para você criar elas no Power Architect.

DIMENSÃO HIERÁRQUICA: PAI-FILHO

Basicamente, é uma forma hierárquica de organizar os dados nas dimensões.

Existem vários tipos de hierarquia nas dimensões, mas a mais conhecida no mercado é a hierarquia chamada pai-filho.

Qual a ideia? Como o próprio nome já diz, é uma hierarquia que tem um pai e um filho. Ela não precisa ter só dois níveis, é claro.



Nessa hierarquia de exemplo, tem 4 níveis, então o cara que está lá em cima não pode acessar o que está aqui embaixo, ele não pode pular o pai.

Alguns exemplos de hierarquia para você entender:

- ano, mês e dia;
- categoria, subcategoria e produto;
- capitão, sargento, cabo e soldado;
- avô, pai e filho.

É comum existir apenas uma hierarquia por dimensão, mas não é erro ter duas na mesma. Trabalhei em um projeto da Sodexo onde o time de logística olhava o produto de uma forma e o de planejamento de outra, e foi preciso criar duas hierarquias para cada um analisar da forma que precisava.

O menor nível da hierarquia é o grão, onde é armazenado o dado. Então na imagem, os meses são chamados de grão ou de nível zero. A contagem de níveis começa no zero e vai subindo.

Ele é o nível mais atômico, o mais detalhado, e é no nível mais atômico em que será armazenada a informação, sem a oportunidade de “descer” mais um nível.

Se o mês é meu grão, só posso contemplar de ano a mês. Se depois lá na frente perceber que preciso descer para dia, não vai dar. Para fazer isso, vai precisar mexer em tudo, refazer as cargas e rever todo o modelo, porque o grão parou no mês.

Você precisa identificar qual o menor dado necessário, porque quanto menor for o nível do grão, mais ETL terá para fazer e mais demorado será, então não adianta colocar sempre o menor possível para tentar evitar o erro.

Tenho aqui o exemplo de uma dimensão de produto e uma dimensão data:

	A	B	C	D	E	F	G	H	I	J	K
	Dimensão Produto (Hierárquica)										
1	SK	cd_cat	Categoria	cd_sub-cat	subCategoria	cd_produto (grão)	produto	Preço unitário		Nível	dim_produto
2	1	120	Bebidas	30	Refrigerante	2323	Coca-Cola	3,55		2	Categoria
3	2	120	Bebidas	30	Refrigerante	1111	guarana	23,60		1	subCategoria
4	3	120	Bebidas	20	Café	4545	Nespresso	7,80		0 (grão)	produto
5			
6			
7											
8											
9											
10											
11	Dimensão Data (Hierárquica)										
12	SK	Ano	Semestre	Trimestre	Mês	data (grão)	dia	Feriado		Nível	dim_data
13	10122016	2016	2	4	Dezembro	10/12/2016	Terça-Feira	sim		4	ano
14	11122016	2016	2	4	Dezembro	11/12/2016	Quinta-Feira	não		3	semestre
15	01012017	2017	1	1	Janerio	12/01/2017	Segunda-Feira	não		2	trimestre
16		1	mês
17										0 (grão)	data
18											
19	Fato Venda										
20	FK	FK	Métrica ativa	Métrica Derivada							
21	DIM_DATA	DIM_PRODUTO	Quantidade de Venda	Valor da Venda							
22	10122016	1	200	710,00							
23	10122016	2	300	7080,00							
24	10122016	3	200	1560,00							
25	700	9.350							
26											

Como essa hierarquia é construída?

Basicamente, na dimensão de produto, a gente tem o código e nome da categoria, o código e nome da subcategoria, e então o código e nome do produto.

No canto da imagem tem um exemplo explicando a hierarquia:

J	K
Nível	dim_produto
2	Categoria
1	sub-categoria
0 (grão)	produto

Mas como eu cheguei nesses dados?

Em todo sistema transacional que eu já vi, quando existe a necessidade de uma hierarquia, ela está lá. O que nós estamos fazendo aqui é a desnormalização dos dados.

Nesse caso, a categoria era uma tabela no transacional, a subcategoria era outra e o produto também.

Mas e quando tem um cliente que quer hierarquia e não tem ela no legado? Pode deixar para ele arrumar isso no transacional ou você cria uma nova hierarquia no Data Warehouse, lá na parte de ETL. De qualquer forma, não vai interferir em nada aqui.

A gente não insere dados na subcategoria porque o nosso grão é o produto, então quando precisar analisar a quantidade de venda da

subcategoria, é só somar a quantidade de vendas de todos os produtos pertencentes àquela subcategoria.

É importante você entender essa técnica porque é muito comum o cliente querer esse tipo de hierarquia para poder ir descendo ou subindo o nível da informação, que é o que chamamos de drill down e drill up. Assim você pode ter uma visualização por ano e fazer um drill down, tendo a visualização por mês e depois por dia.

Vamos fazer uma leitura da fato para você entender como isso vai funcionar.

Na fato nós temos dimensão de data e de produto, quantidade da venda e valor da venda.

Na primeira linha, eu olho a surrogate key de data e encontro ela na dimensão data.

A gente está falando então que é o ano de 2016, 2º semestre, 4º trimestre, no mês de dezembro, em uma terça-feira que era feriado.

Logo, essa leitura fica o seguinte:

No dia 10/12/2016 eu vendi 200 itens de alguma coisa.

Que coisa é essa? Surrogate key 1. Então vamos na dimensão de produto e vemos o que foi vendido. Nesse caso, foi uma Coca-Cola.

No dia 10/12/2016 eu vendi 200 Coca-Colas.

O que acontece com a hierarquia?

É no nível 0 que é inserido o dado, a gente não insere dados nos outros níveis.

Então se você quiser os valores da subcategoria, vai somar os valores dos produtos daquela subcategoria.

J	K	L	M
Nível	dim_produto		
2	Categoria	Bebidas	200
1	sub-categoria	Refrigerante	200
0 (grão)	produto	Coca-Cola	200

Na segunda linha da fato, eu vendi 300 itens do produto da surrogate key 2, que é uma bebida, refrigerante e Guaraná.

J	K	L	M
Nível	dim_produto		
2	Categoria	Bebidas	500
1	sub-categoria	Refrigerante	500
0 (grão)	produto	Coca-Cola	200
		guarana	300

Na terceira linha da fato, eu vendi 200 itens do produto da surrogate key 3, que é um café Nespresso.

J	K	L	M	N	O
Nível	dim_produto				
2	Categoria	Bebidas	700		
1	sub-categoria	Café	200	Refrigerante	500
0 (grão)	produto	Nespresso	200	Coca-Cola	200
				guarana	300

É muito importante a gente ter essa opção de drill, de descer um nível.

Todo lugar que vende coisas no varejo tem a necessidade de saber a sazonalidade do negócio, com isso você pode dizer para o cara quando que a demanda de bebida dele aumenta.

Mas vamos supor que o café fosse só 15 unidades, vou ver que eu vendi 515 unidades de bebidas, o que eu faria? Repetiria o pedido de bebidas. Mas quando a gente fizesse o drill, poderíamos ver que no café foram só 15 unidades, e o restante é só refrigerante.

Por isso que é importante você ter essa hierarquia nas dimensões.

DIMENSÃO ESPECIAL: DIMENSÃO DE TEMPO

A dimensão de tempo é uma dimensão especial. Ela tem algumas funções em um modelo de negócio. E é uma dimensão especial porque sempre está nos modelos dimensionais.

Ela responde o que acredito ser a pergunta mais importante de todo o Data Warehouse: quando que ocorreu isso?

Se você tiver uma dimensão de tempo gabarito funcionando como um reloginho, vai ter muita informação sobre essa pergunta.

Por exemplo:

Na dimensão de tempo temos ano, mês e dia, e também podemos ver quais são os feriados, se é dia de semana ou fim de semana, qual o dia da semana e se tem uma sazonalidade naquele período.

Se a gente souber fazer a modelagem de uma dimensão de tempo e colocar todas essas informações nela, não é incomum você ver dimensões de tempo gigantes, porque você pode colocar, por

exemplo, calendário fiscal, calendário lunar e outros calendários que precisar.

Eu participei de um projeto na Oxiteno que a gente precisou calcular o período da lua, porque a produção de algumas coisas que eles faziam dependia de qual lua estava.

Você abria o dashboard e era impressionante, uma análise com a data normal e outra com a data lunar. E eles produziam muito mais seguindo o calendário lunar.

A dimensão de tempo vai se modificando de acordo com a demanda do negócio, mas quando a pessoa que está criando já é experiente, sabe o que o cliente vai pedir na dimensão de tempo, então ela já tem debaixo da manga uma dimensão de tempo gabarito com um monte de coisa já pronta. E é essa dimensão gabarito que eu vou te ensinar a criar nos próximos capítulos.

Outra característica dessa dimensão é que ela geralmente é carregada no Data Warehouse. Ela é carregada só uma vez, ou seja, nós criamos a estrutura dela, aí pegamos geralmente um script de banco de dados e inserimos todas as linhas da dimensão e acabou.

A dimensão de tempo não vai ser mexida. Ela não vai nem para o time de ETL para eles darem a carga, porque já vai estar pronta, o próprio arquiteto já deixa a dimensão como tem que ser, de tão importante que ela é.

Tem gente que insere esses dados na mão, mas aí é loucura total, porque se você tem uma dimensão de tempo que é diária, tem que inserir todos os dias, ou seja, 365 dias de todos os anos, imagina que você vai fazer de 20 anos para trás, é muita coisa.

Tem gente que faz um script de banco de dados, tem gente que faz com macro no Excel e tem gente que faz na ferramenta de ETL.

Eu recomendo que você veja com o seu cliente qual é o histórico que ele quer trabalhar. Às vezes a empresa tem dados de 20 anos, mas o BI normalmente não é feito de tudo isso, se pega os dados dos últimos 4 anos ou algo assim.

Para fechar esse capítulo: a dimensão de tempo geralmente é o arquiteto que está desenhandando o Data Warehouse que faz e já insere os dados, o time de ETL não mexe nisso. Ela não vai mudar, é carregada uma vez só e acabou, não fica atualizando.

DIMENSÃO DEGENERADA

É a dimensão que não mereceu ser uma dimensão e foi inserida como coluna na fato. Quando a gente vai definir uma dimensão, existem algumas perguntas que fazemos, que você vai ver no capítulo de modelagem de Data Warehouse.

Imagina uma fato venda em que a pessoa quer ver o código da transação da venda.

Às vezes você precisa ter aquela informação ali para fazer um filtro. Quando é algo desse tipo, que não dá para criar uma dimensão, você faz uma degenerada.

Pensa, você vai criar uma dimensão de transação. O que vai ter nela? O código da transação, talvez um nome da transação, se for ser a transação de uma vendedora ou de um produto, já são outras dimensões.

Nesse caso você só quer pôr aquele número porque o usuário precisa dele por algum motivo, para fazer um filtro ou criar uma análise.

E se não vai criar uma dimensão para isso, vai colocar onde? Muitas vezes isso entra na fato transacional, que você vai ver do que se trata na parte de tipos de fatos. A fato venda, com a qual você está trabalhando, é transacional, você coloca todas as transações linha por linha.

SLOWLY CHANGING DIMENSION

De uma forma resumida e bem direto ao ponto, Slowly Changing Dimension é uma técnica para atualizar a dimensão. Tem um nome todo pomposo, mas é como você vai atualizar a dimensão.

Todas as dimensões são SCD, porque elas vão precisar ser atualizadas para se manterem sincronizadas com o transacional.

A única exceção é a dimensão de tempo, que a gente chama de tipo 0, porque depois que os dados foram inseridos, não precisa mais atualizar.

Vou mostrar os 2 tipos mais utilizados da vida real. Se você for olhar na literatura, tem até 6 tipos, mas ninguém usa.

Isso é feito na modelagem do Data Warehouse, mas o trabalho pesado mesmo fica para o ETL.

Tipo 1

Nesse exemplo, tem a tabela de clientes da origem e a dimensão de cliente, com os mesmos dados nas duas.



Depois, a cidade foi alterada na tabela da origem.

Na dimensão, quando é SCD tipo 1, ela atualiza também, mantendo igual ao original e com a mesma surrogate key.

ORIGEM_CLIENTE

Cd_cliente	Nome	Email	Cidade
1070800006	Piton	suporte@raizzer.com	Florianópolis

DIM_CLIENTE

SK	Cd_cliente (NK)	Nome	Email	Cidade
1	1070800006	Piton	suporte@raizzer.com	Florianópolis

Tipo 2

Esta é a mais conhecida e mais utilizada. Além de ter todas as colunas igual a anterior, tem 3 a mais, que é a *data de início*, *data de fim* e uma flag de *ativo*, para controlar o registro.

A função disso é poder armazenar histórico.

Então toda alteração que acontece é inserida em uma nova linha e essas 3 colunas são usadas para controlar as alterações.

Por exemplo:

ORIGEM_CLIENTE

Cd_cliente	Nome	Email	Cidade
1070800006	Piton	suporte@raizzer.com	Porto Alegre

DIM_CLIENTE

SK	Cd_cliente (NK)	Nome	Email	Cidade	Data inicio	Data fim	Ativo
1	1070800006	Piton	suporte@raizzer.com	Porto Alegre	01/12/2016	01/01/1900	S

Na *data de início*, é colocada a data em que foi feita a inserção daquele dado, na *data de fim* você coloca *01/01/1990* porque não existe ainda, e no ativo indica que sim.

Na mesma situação de antes, mudou o endereço na tabela da origem, e na hora de atualizar a dimensão, como ela era tipo 2, insere uma nova linha com uma nova surrogate key e os novos dados.



Depois disso, ela atualiza a linha anterior, colocando a *data de fim* e desativando a flag *ativo*.

Ou seja, a *data de início* dessa nova linha vira a *data de fim* da anterior.

Agora você consegue filtrar a cidade atual onde o cliente mora, mas também tem o histórico de todas as atualizações que foram feitas enquanto ele morava em Porto Alegre.

Para que serve isso?

Imagina que enquanto eu morava em Porto Alegre, gastava R\$1.000,00 de café por mês. Aí eu vou morar em Florianópolis. Se a dimensão for do tipo 1, vai atualizar com a mesma surrogate key e todas as compras que eu fiz vão ser transferidas para a loja de Florianópolis, vai aparecer para eles que eu gastava R\$1.000,00 lá todo mês, e isso não é verdade, eu gastei na loja de Porto Alegre. Isso afeta muito quando tem metas de vendas, por exemplo, a loja de Porto Alegre está quase batendo a meta e quando eu me mudo, atualiza e eles ficam com R\$1.000,00 a menos porque passou para a loja de Florianópolis.

Uma dúvida comum é se o preço do produto vai ficar na fato ou na dimensão. Você pode fazer os dois, você pode colocar na fato para facilitar o cálculo, e pode colocar na dimensão para guardar histórico daquele produto, que é exatamente o que eu estou explicando para você, se mudar o preço do produto e ele está na fato, você não vai ter esse histórico. Se o preço de um produto muda e as vendas caem,

ninguém vai saber que foi porque o preço mudou, porque você não vai ter esse registro.

Mas é importante você saber que essas alterações impactam na carga. No momento que você fez todo esse trabalho, aumentou a complexidade do ETL.

Tem ferramentas, como PDI e ODI, que controlam essa questão da SCD, mas algumas não fazem isso, e o próprio ETL vai ter que controlar de alguma forma, mas geralmente ferramentas que trabalham com esse tipo de técnica já preparam para que isso fique fácil.

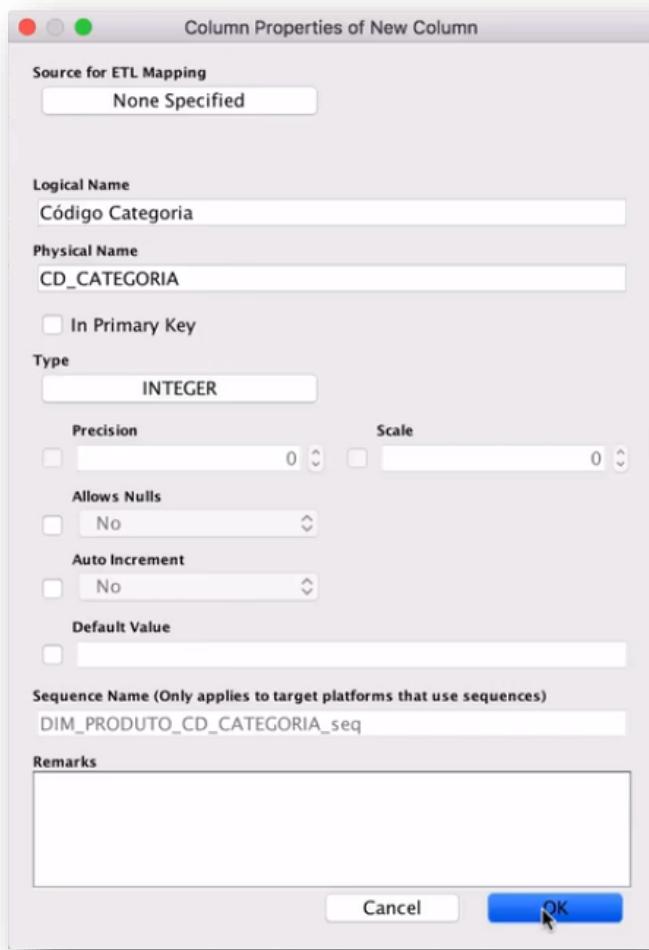
PASSO A PASSO: COMO CRIAR HIERARQUIA PAI-FILHO

Aqui a gente vai fazer alterações no nosso modelo Star Schema para aplicar a técnica de dimensão hierárquica.

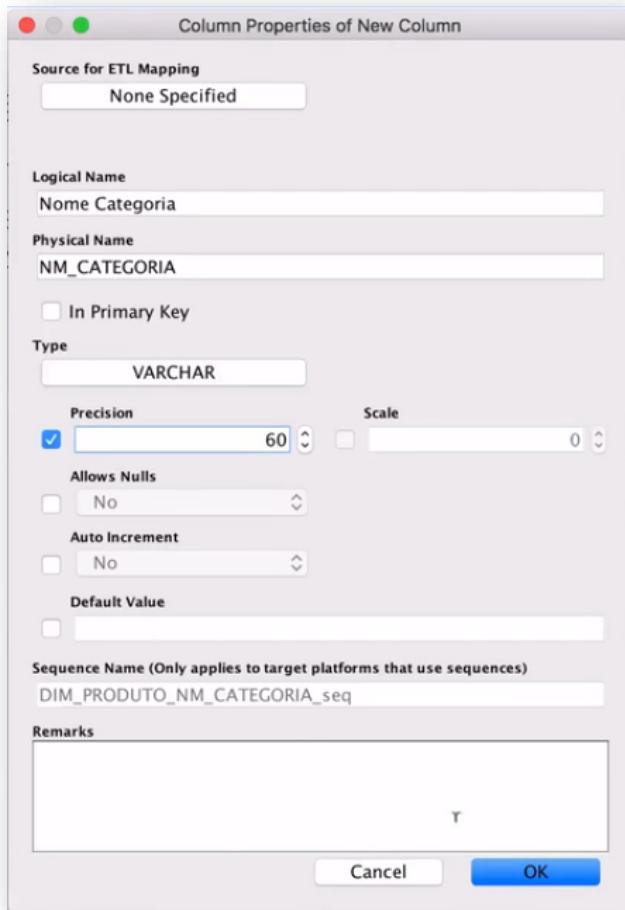
O que nós precisamos fazer:

- colocar a hierarquia da dimensão de produto;
- renomear a natural key para código do produto.

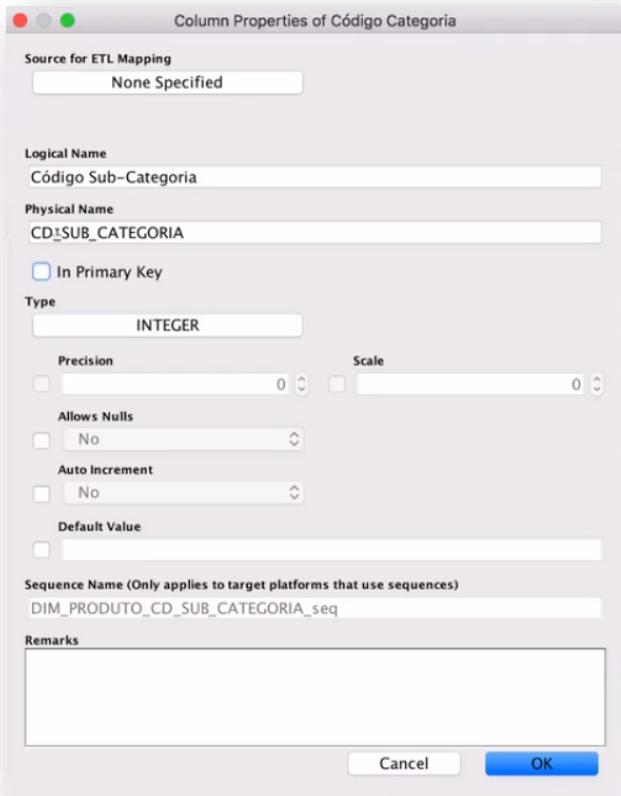
No Power Architect, você vai clicar na dimensão de produto e adicionar uma coluna que vai se chamar *Código Categoria*, o nome físico dela vai ser *CD_CATEGORIA* e ela é do tipo *integer*.



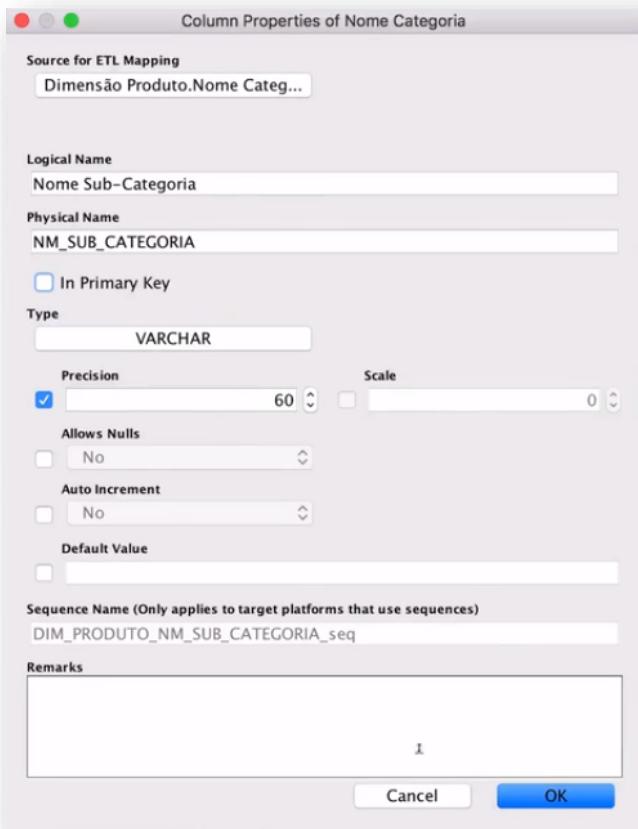
Agora você vai criar outra coluna chamada *Nome Categoria*, com nome físico *NM_CATEGORIA*, do tipo *varchar* com uma precisão de 60.



Agora você vai criar a coluna que vai se chamar *Código Subcategoria*, o nome físico dela vai ser *CD_SUBCATEGORIA* e ela é do tipo *integer*.



E depois você cria outra coluna chamada *Nome Subcategoria*, com nome físico *NM_SUBCATEGORIA*, do tipo *varchar* com uma precisão de 60.



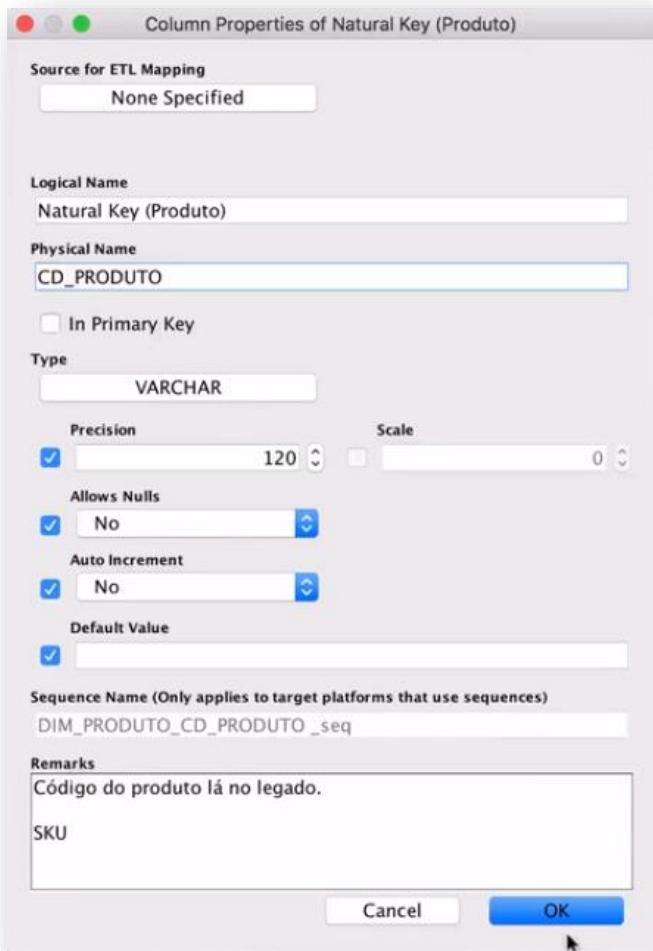
Eu gosto de colocar underline no nome físico porque a maioria das ferramentas de análise tem a opção de substituir o underline por espaço, assim você já consegue renomear de uma forma muito rápida, vai ficar fácil de entender.

Eu gosto de deixar a hierarquia ordenada na dimensão, então reposiciono as colunas.

DIM_PRODUTO
SK_PRODUTO: INTEGER NOT NULL [PK]
CD_CATEGORIA: INTEGER NOT NULL
NM_CATEGORIA: VARCHAR(60) NOT NULL
CD_SUB_CATEGORIA: INTEGER NOT NULL
NM_SUB_CATEGORIA: VARCHAR(60) NOT NULL
NK_PRODUTO: VARCHAR(120) NOT NULL
NM_PRODUTO: VARCHAR(60) NOT NULL
MARCA_PRODUTO: VARCHAR(60) NOT NULL
PRECO_UNI: NUMERIC(12, 4) NOT NULL

Na natural key do produto, você vai renomear o nome lógico para *Natural Key (Produto)*, para a gente saber do que está falando, e no nome físico vai colocar *CD_PRODUTO*.

Em alguns lugares, você vai ouvir falar de SKU, que é o código do produto em varejo e coisas assim, vou deixar aqui uma anotação de lembrete.



Agora sim ela está com uma hierarquia.

DIM_PRODUTO

SK_PRODUTO: INTEGER NOT NULL [PK]
CD_CATEGORIA: INTEGER NOT NULL
NM_CATEGORIA: VARCHAR(60) NOT NULL
CD_SUB_CATEGORIA: INTEGER NOT NULL
NM_SUB_CATEGORIA: VARCHAR(60) NOT NULL
CD_PRODUTO : VARCHAR(120) NOT NULL
NM_PRODUTO: VARCHAR(60) NOT NULL
MARCA_PRODUTO: VARCHAR(60) NOT NULL
PRECO_UNI: NUMERIC(12, 4) NOT NULL

Você acabou de deixar a sua dimensão pronta para receber os dados.

PASSO A PASSO: COMO CRIAR DIMENSÃO DE TEMPO

A única coisa que falta agora para o seu modelo estar pronto é a dimensão de tempo. E é isso que a gente vai fazer aqui. Eu quero te mostrar a forma que eu utilizo na vida real para criar essa dimensão.

Você poderia criar a dimensão como já vem aprendendo aqui, no Power Architect.

Mas a dimensão de tempo, como eu falei, é mais estática, então é possível fazer diferente.

Aqui você vai utilizar o DBeaver para fazer isso.

Você vai utilizar um código em SQL para criar essa dimensão, então você vai precisar abrir um editor de SQL em *SQL Editor > New SQL Editor*.



Aqui eu vou te passar o script que eu uso para criar a dimensão de tempo. Com ele, você não precisa ir no Power Architect toda vez para criar a mesma dimensão, porque ela geralmente é igual, só vamos aumentando o script com o tempo.

O script é esse:

```
create table dim_data (
    sk_data integer not null,
    data date not null,
    desc_data_completa varchar(60) not null

    nr_ano integer not null,
    nm_trimestre varchar(20) not null,
    nr_ano_trimestre varchar(20) not null,
    nr_mes integer not null,
    nm_mes varchar(20) not null,
    ano_mes varchar(20) not null,
    nr_semana integer not null,
    ano_semana varchar(20) not null,
```

```
nr_dia integer not null,  
nr_dia_ano integer not null,  
nm_dia_semana varchar(20) not null,  
flag_final_semana char(3) not null,  
flag_feriado char(3) not null,  
nm_feriado varchar(60) not null,  
dt_carga timestamp not null,  
constraint sk_data_pk primary key  
(sk_data)  
) ;
```

Mas antes de a gente criar a dimensão, vou te explicar o que cada um desse campos faz e mostrar um exemplo de valor de cada um deles.

- **sk_data** - essa é a surrogate key da dimensão.
 - Exemplo: 20160101
- **data** - aqui vai ser a data em si. O formato dela vai depender da configuração do banco, se está brasileira, americana, etc.
 - Exemplo: 01/01/2017
- **desc_data_completa** - descrição da data completa, com o mês escrito por extenso.
 - Exemplo: 01 janeiro de 2017
- **nr_ano** - número do ano

- Exemplo: 2017
- **nm_trimestre** - número do trimestre. Em que trimestre do ano aquela data está.
 - Exemplo: 1t
- **nr_ano_trimestre** - ano e número do trimestre, é como uma junção dos dois campos anteriores.
 - Exemplo: 2017/1t
- **nr_mes** - número do mês
 - Exemplo: 01
- **nm_mes** - nome do mês
 - Exemplo: janeiro
- **ano_mes** - ano e número do mês juntos.
 - Exemplo: 2017/01
- **nr_semana** - número daquela semana no ano.
 - Exemplo: 34
- **ano_semana** - ano e número da semana juntos.
 - Exemplo: 2017/34
- **nr_dia** - o número do dia no mês.
 - Exemplo: 31
- **nr_dia_ano** - ano e número do dia no mês juntos.
 - Exemplo: 2017/31

- **nm_dia_semana** - nome do dia da semana.
 - Exemplo: quinta-feira
- **flag_final_semana** - flag para indicar se é final de semana ou não.
 - Exemplo: sim
- **flag_feriado** - flag para indicar se é feriado ou não.
 - Exemplo: sim
- **nm_feriado** - se for feriado, o nome do feriado.
 - Exemplo: natal, ano novo, pascoa, etc....
- **dt_carga** - data que os dados foram inseridos na dimensão.
 - Exemplo: 01/01/2017 23:59:00

As flags são como indicadores, é só uma bandeirinha para dizer se sim ou se não. A flag de final de semana é muito importante para fazer comparações de resultados em dia de semana e final de semana.

Eu já fiz projetos bem grandes, onde eu precisava mexer na dimensão de tempo, e no final, o que o pessoal estava pedindo era só essa flag de final de semana, mas não sabiam como colocar o dado correto nela para todos os anos. Mas eu vou mostrar para você como faz isso também.

O campo de data da carga é importante para caso o key user esteja fazendo alguma análise, assim ele vai saber se os dados já foram atualizados naquele dia ou não.

Agora é só você colocar o script no DBeaver e rodar.

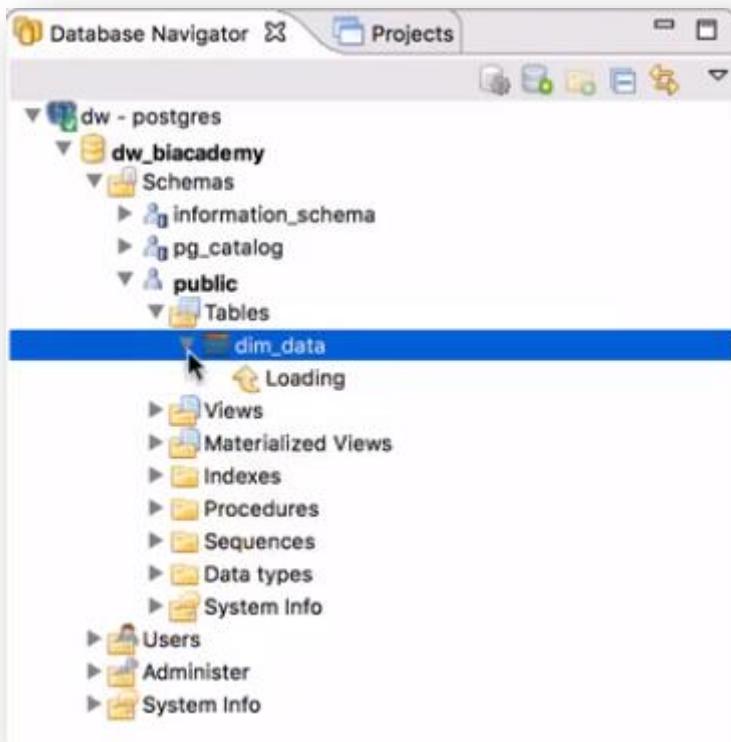
```
*<dw - postgres> Script
CREATE TABLE DIM_DATA (
    SK_DATA INTEGER NOT NULL, --20160101
    DATA DATE NOT NULL, --01/01/2017
    DESCRITIVO_COLETIVA VARCHAR(60) NOT NULL, --01 janeiro de 2017
    NR_ANO INTEGER NOT NULL, --2017
    NM_ANO_TRIMESTRE VARCHAR(20) NOT NULL, -- 1T
    NR_ANO_TRIMESTRE VARCHAR(20) NOT NULL, -- 2017/1T
    NR_MES INTEGER NOT NULL, -- 01
    NM_MES VARCHAR(20) NOT NULL, -- Janeiro
    ANO_MES VARCHAR(20) NOT NULL, -- 2017/01
    NR_SEMANA INTEGER NOT NULL, -- 34
    ANO_SEMANA VARCHAR(20) NOT NULL, --2017/34
    NR_DIA INTEGER NOT NULL, --31
    NR_SEMANA_DIA INTEGER NOT NULL, -- 2017/31
    NM_DIA_SEMANA VARCHAR(20) NOT NULL, -- Quinta-feira
    FLAG_FINAL_SEMANA CHAR(3) NOT NULL, -- sim
    FLAG_FERIADO CHAR(3) NOT NULL, -- sim
    NM_FERIADO VARCHAR(60) not null, --Natal, ano novo, pascoa, etc....
    DT_CARGA TIMESTAMP NOT null, -- 01/01/2017 23:59:00
    CONSTRAINT sk_data_pk PRIMARY KEY (sk_data)
);
```

E ele vai dar o resultado depois, sem nenhum erro.

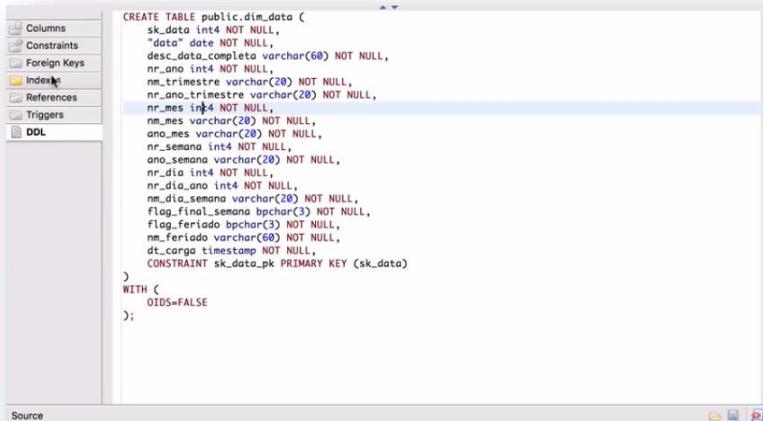
Name	Value
Query	CREATE TABLE DIM_DATA (
Updated Rows	0
Finish time	Thu Dec 29 13:56:19 BRST 2016

1 row(s) fetched - 6ms

Para saber se funcionou mesmo, você vai na coluna da esquerda e faz o *refresh*. Vai aparecer a dimensão de data dentro de *Tables*.



Se você clicar nela, vai ver que a ferramenta já faz uma conversão para DDL também, para caso você queira rodar isso direto no banco.

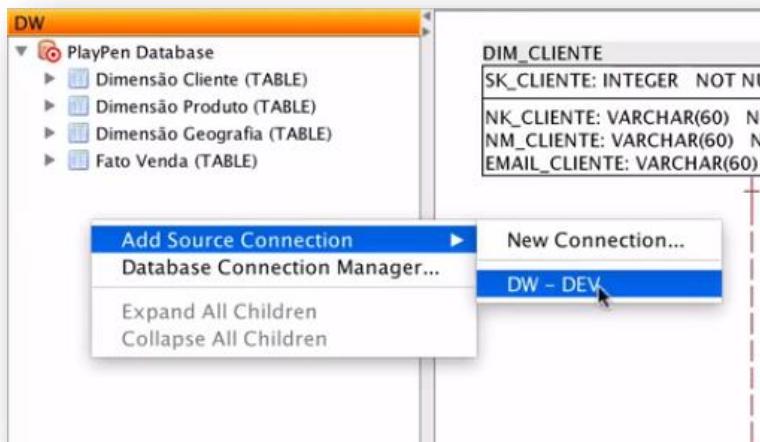


The screenshot shows a software interface for database management, specifically Power Architect. On the left, there's a sidebar with icons for Columns, Constraints, Foreign Keys, Indexes, References, Triggers, and DDL. The main area is titled 'Source' and contains the following SQL code:

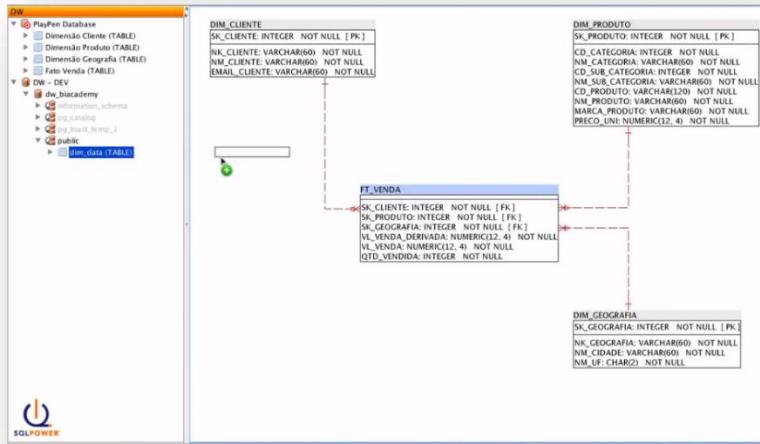
```
CREATE TABLE public.dim_data (
    sk_data int4 NOT NULL,
    "desc_data" varchar(60) NOT NULL,
    desc_data_completa varchar(60) NOT NULL,
    nr_ano int4 NOT NULL,
    nm_trimestre varchar(20) NOT NULL,
    nr_ano_trimestre varchar(20) NOT NULL,
    nr_mes int4 NOT NULL,
    nm_mes varchar(20) NOT NULL,
    ano_mes varchar(20) NOT NULL,
    nr_semana int4 NOT NULL,
    ano_semana varchar(20) NOT NULL,
    nr_dia int4 NOT NULL,
    nr_dia_ano int4 NOT NULL,
    nm_dia_semana varchar(20) NOT NULL,
    flag_feriado_semana bpchar(3) NOT NULL,
    flag_feriado bpchar(3) NOT NULL,
    nm_feriado varchar(60) NOT NULL,
    dt_carga timestamp NOT NULL,
    CONSTRAINT sk_data_pk PRIMARY KEY (sk_data)
)
WITH (
    OIDS=FALSE
);
```

E agora, como que a gente coloca isso lá no Power Architect?

No Power Architect, na coluna da esquerda, você pede para ele adicionar a conexão com o banco. Para fazer isso, é só clicar com o botão direito > *Add Source Connection* > dw (esse último vai ser o nome da sua conexão).

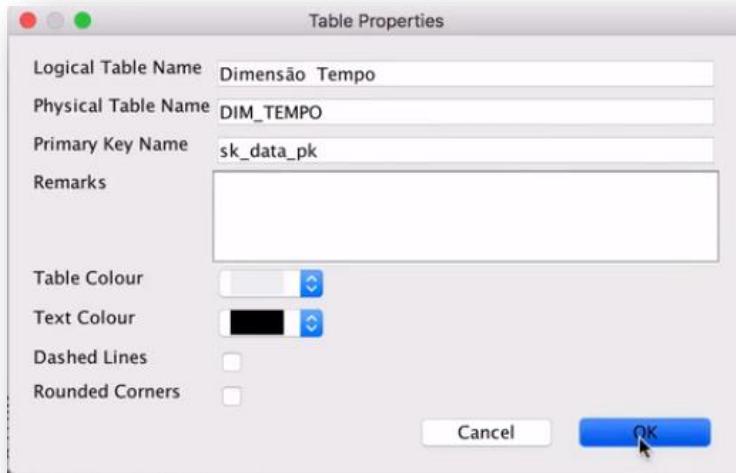


Depois de conectado, se eu entrar no schema *public*, a dimensão de data vai estar lá. Para levar ela para o Playpen (o campo da direita onde está a nossa modelagem), é só arrastar a tabela.



E depois fazer o join da dimensão com a fato.

Agora você também pode entrar na dimensão e arrumar as propriedades da tabela.

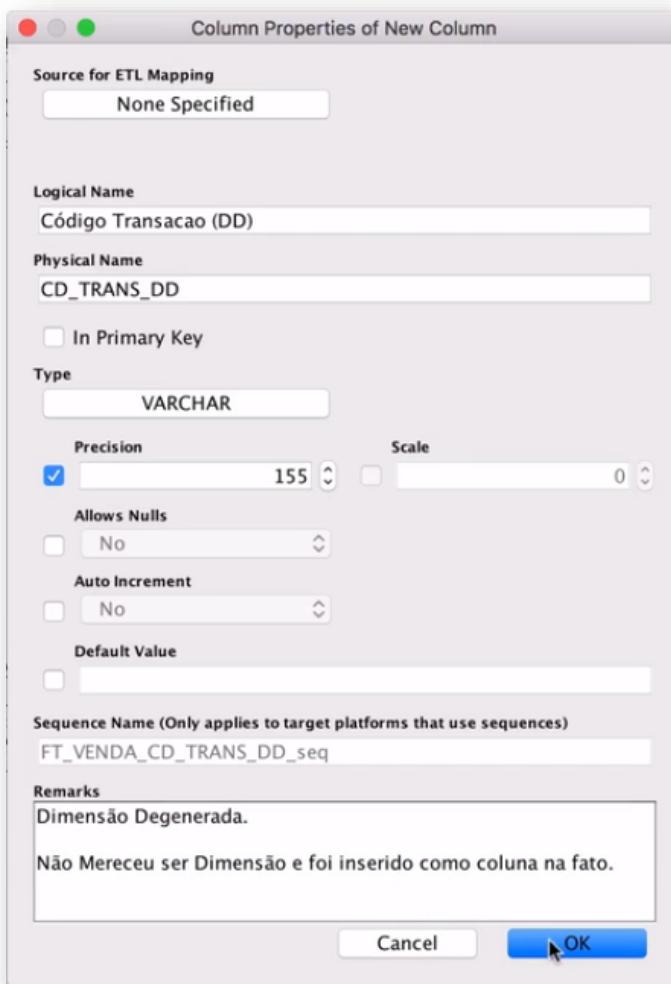


E na surrogate key dessa dimensão, você edita para tirar o mapeamento do ETL. Uma observação: a surrogate key dessa tabela não é *auto-increment*, porque apesar de ser uma surrogate key, é a gente que vai inserir esses dados, inclusive os da surrogate key.

PASSO A PASSO: COMO CRIAR DIMENSÃO DEGENERADA

A criação da dimensão degenerada é bem simples.

No Power Architect, você vai na fato e cria uma coluna nova. Eu costumo marcar quando é dimensão degenerada, por exemplo: *código transação (DD)*.



E agora você tem uma dimensão degenerada na fato.

FT_VENDA
sk_data: INTEGER NOT NULL [FK]
SK_CLIENTE: INTEGER NOT NULL [FK]
SK_PRODUTO: INTEGER NOT NULL [FK]
SK_GEOGRAFIA: INTEGER NOT NULL [FK]
CD_TRANS_DD: VARCHAR(155) NOT NULL
VL_VENDA_DERIVADA: NUMERIC(12, 4) NOT NULL
VL_VENDA: NUMERIC(12, 4) NOT NULL
QTD_VENDIDA: INTEGER NOT NULL

Apesar de ter um nome diferente, a criação dela não tem muito mistério.

PASSO A PASSO: COMO CRIAR SLOWLY CHANGING DIMENSION

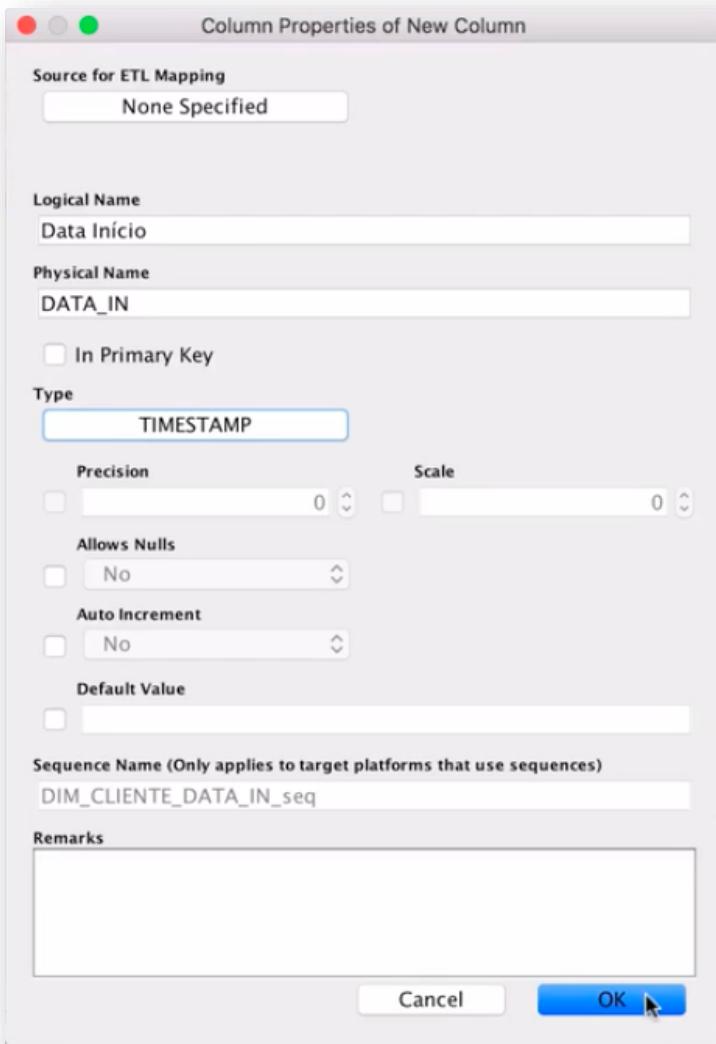
Para criar a Slowly Changing Dimension na sua modelagem, você vai no Power Architect.

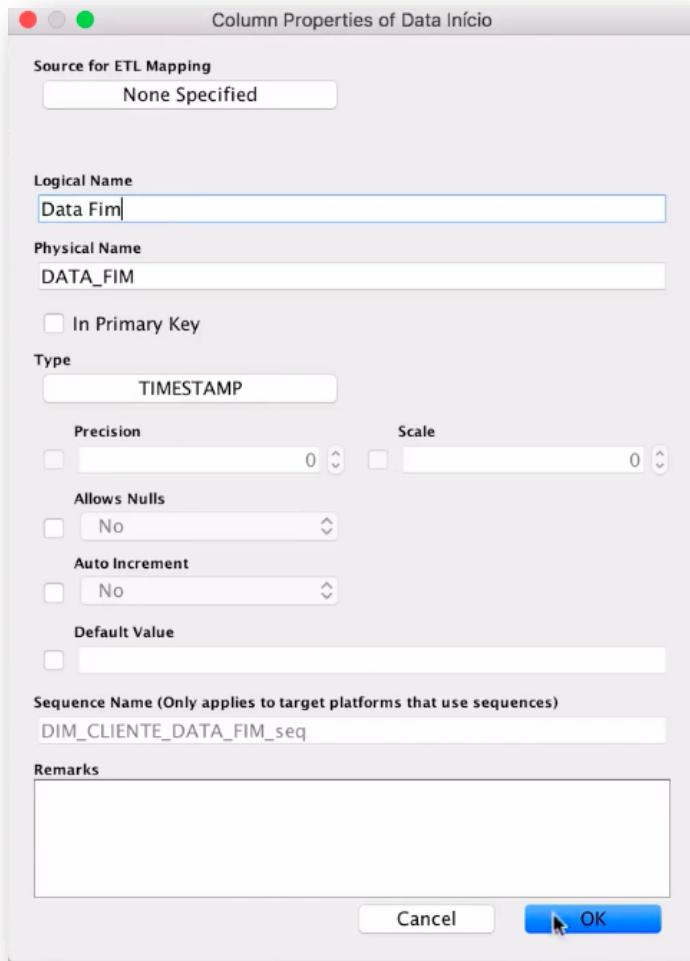
Na dimensão de cliente, você vai criar 3 colunas:

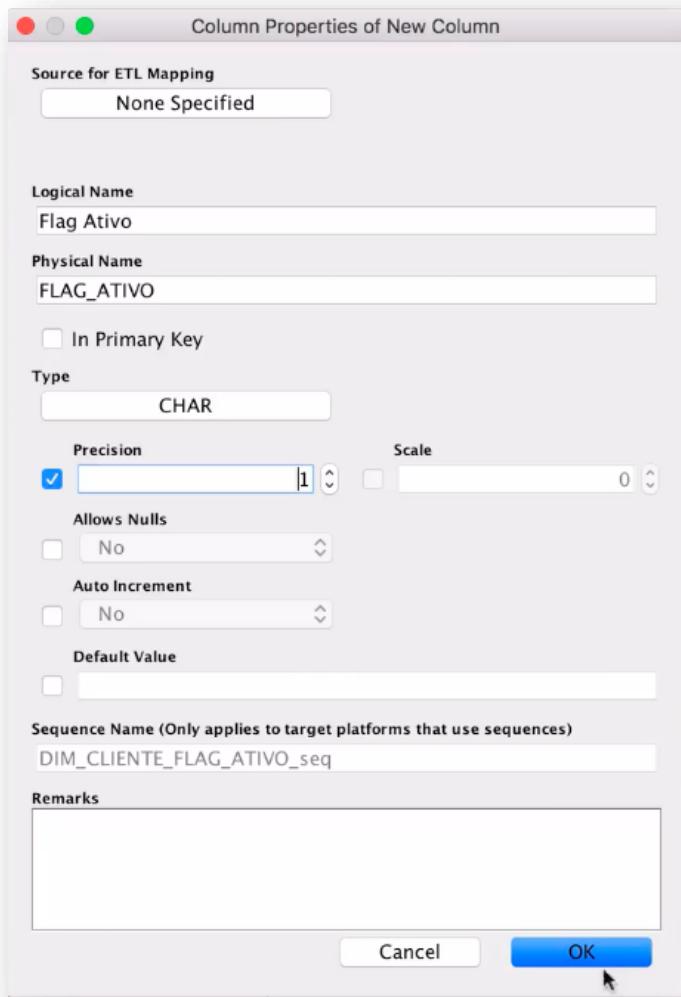
- *Data Início*;
- *Data Fim*;
- *Flag Ativo*.

Eu gosto de colocar o tipo da *Data Início* e *Data Fim* como *timestamp*, que é data e hora, embora você possa utilizar o tipo *date* também.

A *Flag Ativo* vai ser um *char* com precisão de 1, porque é só um caractere que vai ser inserido nela.

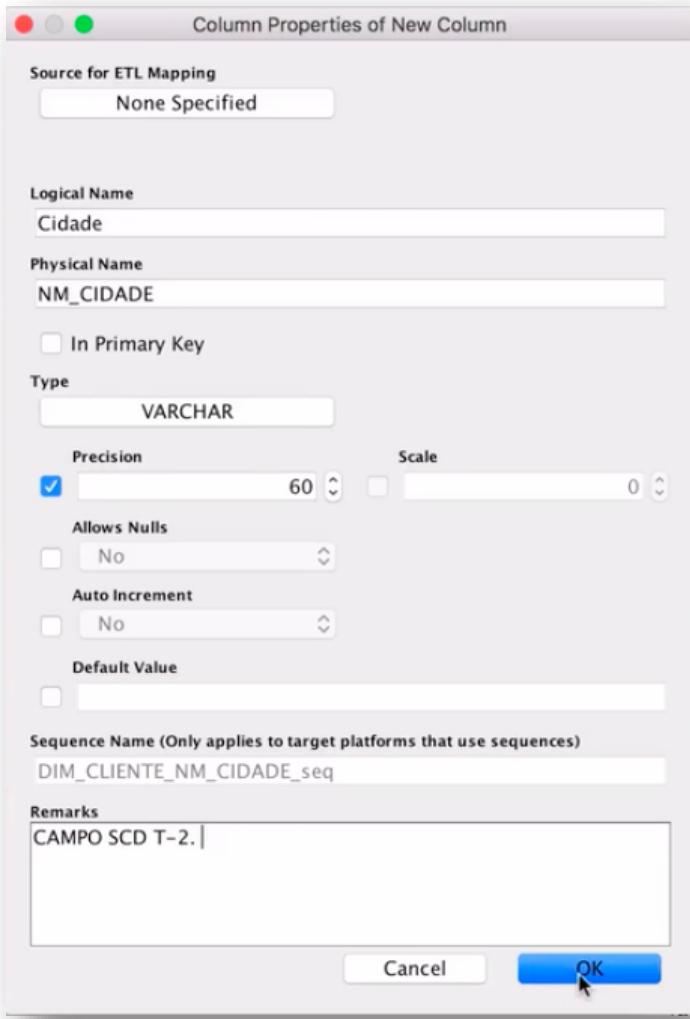






Agora você deixou a sua dimensão preparada para trabalhar com Slowly Changing Dimension.

Para representar o nosso exemplo do capítulo que mostrei a técnica, vou adicionar aqui também a cidade do cliente.



Importante: se você quiser guardar histórico de outro campo, vai ter que criar outras colunas de data para controlar esse campo.

As outras dimensões vão ser SCD tipo 1, e isso é controlado com o campo de atualização da carga, como a gente tinha feito na dimensão

de tempo. Então é só você criar campos iguais àquele na fato e em todas as outras dimensões, exceto a de cliente, que é SCD tipo 2.

FATO TRANSACIONAL

Fatos transacionais são as mais comuns. A maioria dos bilhões de linhas que temos no Data Warehouse são de tabelas fato transacionais. Elas geralmente utilizam métricas aditivas, aquelas métricas que podem ser somadas por todas as dimensões.

Existem duas formas de você armazenar os dados em uma fato transacional:

- transação por linha;
- linha por transação.

Em forma de transação por linha

A cada transação que ocorre, você insere uma nova linha.

Neste exemplo a fato é de vendas. Então cada item de uma venda será salvo em uma linha da fato.

A	B	C	D	E	F	G	H	I	J	K
1										
2										
3										
4										
5	Linha	Transação								
6										
7										
8										

É claro que você já sabe que nas colunas de dimensão não vai estar realmente o nome do item, mas uma surrogate key. O exemplo foi feito assim apenas para facilitar a visualização.

Agora, imagina que eu vou no mercado comprar 3 Coca-Colas.

Foi registrado que no dia 10/12/2016, foi vendido uma Coca-Cola para o cliente Piton em Porto Alegre, e quem fez essa venda foi a Bia. Ali também tem o código da transação, o valor unitário desse produto, quantos itens foram vendidos, se tinha desconto, e o valor final da compra.

Mas foram 3 Coca-Colas que eu comprei, então como é uma transação por linha, vou inserir uma nova linha para cada item, mantendo o mesmo código da transação, porque foi a mesma compra.

A	B	C	D	E	F	G	H	I	J	K
1										
2										
3										
4										
5	Linha	Transação								
6										
7										
8										

Se eu tivesse comprado um café também, ele entraria em uma nova linha com o mesmo código da transação.

A	B	C	D	E	F	G	H	I	J	K
Fato Transacional (Vendas)										
Cruzamento/Dimensionalidade é sempre feito no menor nível da transação (Grão: Nível 0.)										
DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA	Métrica adtiva
5 10/12/2016	6 Coca-Cola	7 Piton	8 Porto Alegre	9 Bia	10 #4474343	11 3,8	12 1	13 0,00	14 3,80	Métrica adtiva
5 Uma 6 TRANSAÇÃO 7 por 8 Lema	10/12/2015	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	café	Piton	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90

Aí eu saí da loja e fui para casa, mas chegou visita e tomaram minhas Coca-Colas. Então de noite eu fui lá buscar mais Coca-Cola, e gerou uma nova transação, porque agora é uma nova compra.

A	B	C	D	E	F	G	H	I	J	K
Fato Transacional (Vendas)										
Cruzamento/Dimensionalidade é sempre feito no menor nível da transação (Grão: Nível 0.)										
DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA	Métrica adtiva
5 Uma 6 TRANSAÇÃO 7 por 8 Lema	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	café	Piton	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#9988574	3,8	1	0,00	3,80

Então basicamente é isso, quando a gente registrar na fato uma transação por linha, é dessa forma que vai ficar. Uma coisa que você vai ver é que a quantidade do item vai ser sempre 1 e o valor da venda vai ser quase sempre igual, exceto se o desconto mudar.

É interessante trabalhar dessa forma quando você tem desconto, por exemplo: na compra de 3 itens, o último tem um desconto.

Aí já muda, eu vou ter um desconto em um dos itens.

A	B	C	D	E	F	G	H	I	J	K
Fato Transacional (Vendas)										
Cruzamento/Dimensionalidade é sempre feito no menor nível da transação (Grão: Nível 0.)										
DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA	Métrica adtiva
5 Uma 6 reivindicação	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	1,00	2,80
	10/12/2016	café	Piton	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#9988574	3,8	1	0,00	3,80

Em forma de linha por transação

Essa é outra forma que a gente utiliza para trabalhar uma fato transacional.

Vou fazer a mesma leitura aqui, no dia 10/12/2016, o cliente Piton comprou três Coca-Colas em Porto Alegre e quem fez a venda foi a Bia. O código da transação é o mesmo lá de cima porque é a mesma compra, só que está sendo representada de um jeito diferente.

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
Fato Transacional (Vendas)											
3											
4	Uma	DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA
5	TRANSAÇÃO	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
6	por	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
7	Linha	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
8		10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90
9		10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#5988574	3,8	1	1,99	2,89
10											
11	Uma Linha	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	3	1,00	10,40
12	por transação										

Mas aqui é uma linha por transação. Ou seja, essa transação, que é a mesma que repetiu três vezes no primeiro exemplo, vai ser um único registro nessa fato. O preço unitário aqui segue o mesmo, mas a quantidade vai ser 3. Aí no valor da venda já vai ter o total da compra, que vai ser R\$10,40.

E o café vai ficar em uma linha separada, porque mesmo sendo a mesma transação, o produto é diferente.

A	B	C	D	E	F	G	H	I	J	K
Fato Transacional (Vendas)										
Cruzamento/Dimensionalidade é sempre feito no menor nível da transação (Grão: Nível 0.)										
DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA	Métrica ativa
10/12/2016	Coca-Cola	Pilon	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80	Métrica ativa
10/12/2015	Coca-Cola	Pilon	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80	Métrica ativa
10/12/2015	Coca-Cola	Pilon	Porto Alegre	Bia	#4474343	3,8	1	1,00	2,80	Métrica ativa
10/12/2016	café	Pilon	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90	Métrica ativa
10/12/2016	Coca-Cola	Pilon	Porto Alegre	Bia	#5988074	3,8	1	1,00	2,80	Métrica ativa
Uma LINHA por transação										
10/12/2015	Coca-Cola	Pilon	Porto Alegre	Bia	#4474343	3,8	*	3	1,00	10,40
10/12/2016	café	Pilon	Porto Alegre	Bia	#4474343	5,9	1	0,00	5,90	Métrica ativa

A forma de uma linha por transação é a mais utilizada. Mas porque eu estou mostrando os dois? Para você saber como que isso pode ser feito.

E como nesse exemplo, às vezes você quer ver uma variação por item, como foi o caso do desconto, e precisa ter uma transação por linha. Mas em via de regra, o mercado trabalha com uma linha por transação.

PASSO A PASSO: COMO CRIAR FATO TRANSACIONAL

Agora eu vou mostrar aqui as alterações que você tem que fazer no seu modelo para aplicar os conceitos da fato transacional.

As alterações que você vai fazer são:

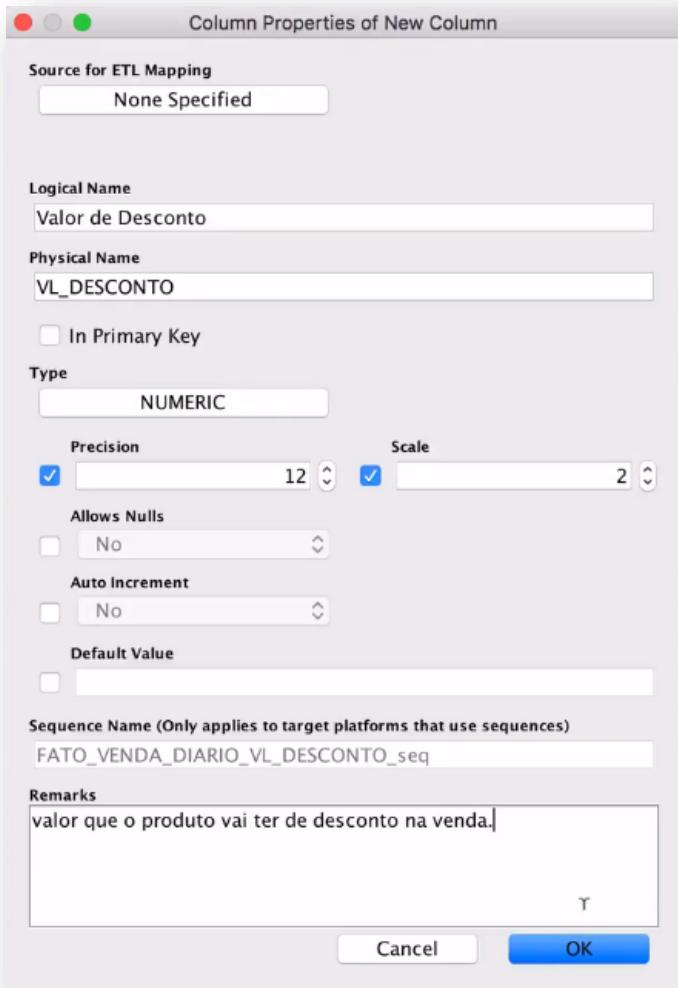
- renomear a *Fato Venda* para *Fato Venda Diário*;
- adicionar a métrica *valor de desconto*;
- ajustar as métricas de *valor da venda*.

No Power Architect, você vai clicar na sua fato e vai alterar as propriedades dela, no nome físico você coloca *FATO_VENDA_DIARIO* e no nome lógico *Fato Venda Diário*.



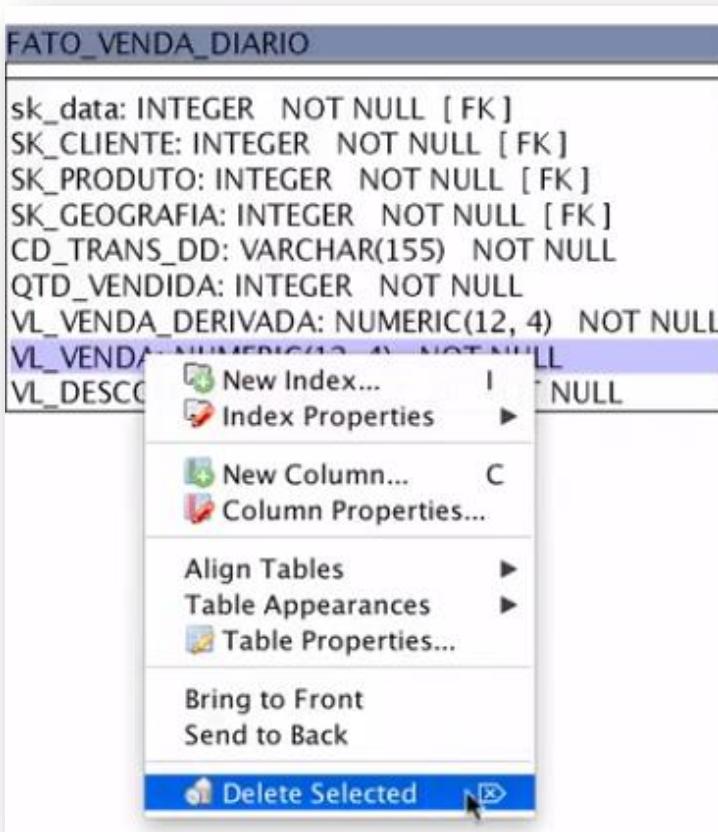
Depois, você adiciona uma nova coluna na fato para ser a sua métrica de *valor de desconto*.

O nome físico dela vai ser *VL_DESCONTO* e o lógico *Valor de Desconto*. Ela vai ser do tipo *numeric*, com a precisão de 12 e a escala de 2.

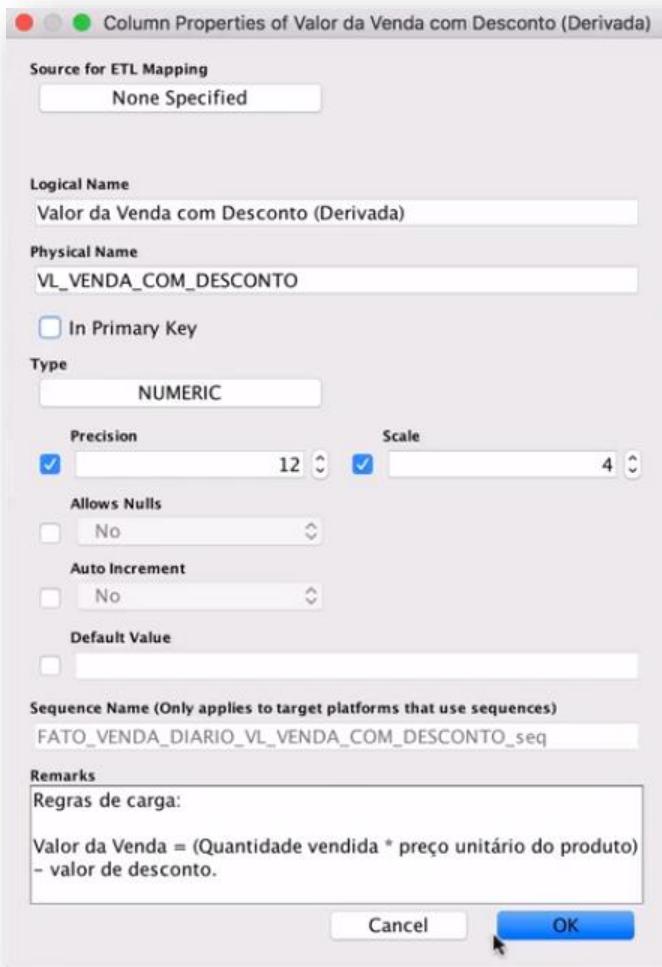


Depois você precisa arrumar as métricas de valor da venda. A métrica VL_VENDA que você já tem criada, pode deletar.

Você clica com o botão direito na métrica e em *Excluir selecionado*.

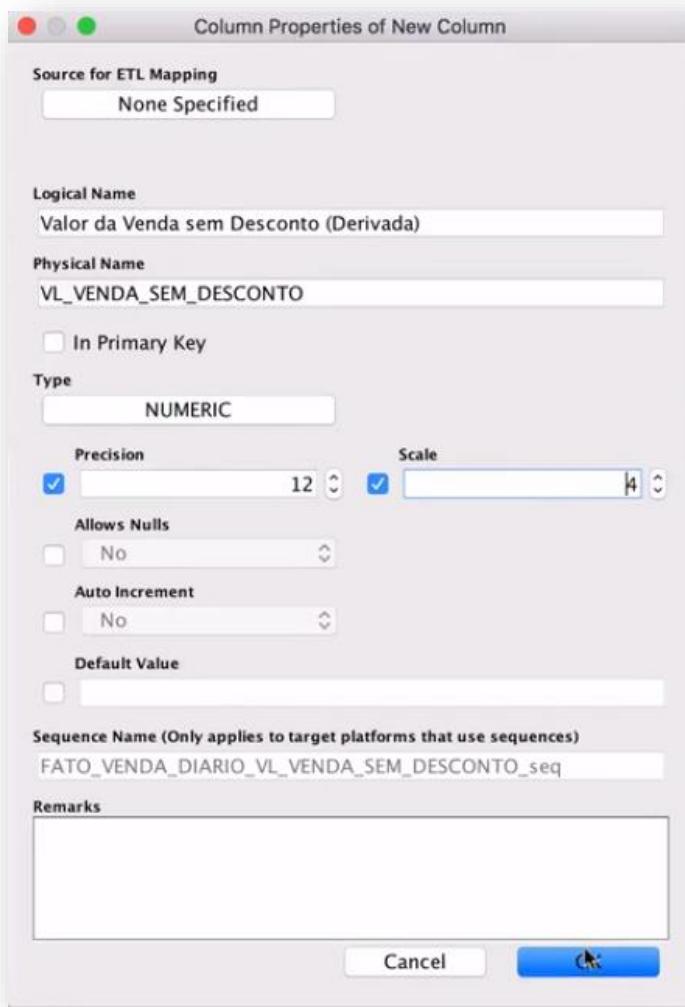


A métrica *VL_VENDA_DERIVADA* que você também já tem criada, você edita. No nome lógico você coloca *Valor da Venda com Desconto (Derivada)* e no nome físico *VL_VENDA_DESCONTO*. No comentário, você precisa atualizar a fórmula para ele subtrair o valor do desconto.



Agora você adiciona uma nova coluna na fato para ser a métrica de *valor da venda sem desconto*.

O nome lógico dela vai ser *Valor da Venda sem Desconto (Derivada)* e o físico *VL_VENDA_SEM_DESCONTO*. Como a outra, ela vai ser do tipo *numeric*, com precisão 12 e escala 4.



A métrica de quantidade vendida, que você já tem, troca o nome lógico para *QTD_VENDA*, para ficar mais fácil de a gente analisar isso no futuro.



E estão prontas as alterações.

FATO_VENDA_DIARIO

```
SK_DATA: INTEGER NOT NULL [ FK ]
SK_CLIENTE: INTEGER NOT NULL [ FK ]
SK_PRODUTO: INTEGER NOT NULL [ FK ]
SK_GEOGRAFIA: INTEGER NOT NULL [ FK ]
#DD_TRANSACAO: VARCHAR(155) NOT NULL
QTD_VENDA: INTEGER NOT NULL
VL_DESCONTO: NUMERIC(12, 2) NOT NULL
VL_VENDA_COM_DESCONTO: NUMERIC(12, 4) NOT NULL
VL_VENDA_SEM_DESCONTO: NUMERIC(12, 4) NOT NULL
```

Agora a fato já está com cara de um projeto da vida real. Você está passando por um processo de aprendizagem, por isso a gente começa com o mais simples e vai melhorando aos poucos conforme você vai aprendendo mais.

O Bolt não nasceu correndo daquele jeito, ele primeiro aprendeu a engatinhar, para depois caminhar e só aí começar a correr. Por isso é importante você ir fazendo tudo o que eu estou mostrando.

Agora você pode persistir as alterações no banco de dados como fez no [capítulo de Métricas](#).

AGILE DATA WAREHOUSE DESIGN

Aqui eu quero te ensinar a forma que eu utilizo para fazer o levantamento de requisitos com os meus clientes.

Enquanto tem empresas levando 6 meses para entregar um Data Warehouse, quero te mostrar como eu faço para entregar isso em 1 mês ou menos.

A ideia é criar um Data Warehouse de forma ágil. Ninguém mais tem tempo para perder, então ao invés de levar meses montando um Data Warehouse gigante, entregue ele por partes e comece o quanto antes a apresentar os dados que os key users precisam ver.

Eu uso um método que é o 5W3H, que é uma adaptação que eu fiz do 5W2H.

Com ele, você vai fazer o levantamento de todo seu Star Schema em 6 passos:

- passo 1 - Entrevistar os key users;
- passo 2 - Identificar a fato;
- passo 3 - Identificar as métricas;

- passo 4 - Identificar as dimensões;
- passo 5 - Definir a hierarquia;
- passo 6 - Identificar os atributos.

PASSO 1 - ENTREVISTAR OS KEY USERS

Nesse primeiro passo, tem algumas coisas que é importante você saber.

A obrigação da tomada de decisão correta é sua. Embora o cliente deva saber o que ele quer, como um profissional de BI, você tem que saber extrair as melhores respostas, então:

- use perguntas que os key users estão acostumados a ver no dia a dia;
- não use termos técnicos se não tiver necessidade;
- o usuário de negócio não sabe o que são fatos ou dimensões, então você vai precisar induzir ele a dar as informações que você precisa.

O 5W3H usa 9 perguntas simples para você:

- levantar as informações necessárias para montar seu Star Schema;
- falar a mesma língua dos usuários de negócio.

As perguntas são:

Qual fato aconteceu?

#1 – What (do quê?)

#2 – When (quando?)

#3 – Where (onde?)

#4 – Who (quem?)

#5 – How (como?)

#6 – Why (por quê?)

#7 – How often (com que frequência?)

#8 – How many (quantas vezes?)

Você pode colocar essas perguntas em um documento ou como preferir, eu gosto de criar um mapa mental com elas, e deixar isso na nuvem, assim todo mundo pode editar no tempo que tiver.

Ao invés de fazer mil reuniões, você manda essas perguntas para os envolvidos e eles respondem no tempo deles. Quando você for fazer a reunião com todo mundo, é só alinhar o que já foi respondido, não precisa começar do zero.

Eu vou passar com você por cada uma delas para explicar que tipo de resposta você vai identificar com cada pergunta.

Qual fato aconteceu?

Como seus key users podem não estar habituados com as terminologias de BI, invista algum tempo explicando o que você espera dessa pergunta.

Dê exemplo de fatos, como:

- venda;

- transação;
- pedido;
- faturamento.

Você vai ter respostas como:

Qual o fato que aconteceu? Uma venda.

Qual o fato que aconteceu? Uma demissão.

Juntando as informações, você vai montando a história de como aquele fato aconteceu:

ACONTECEU UMA VENDA

As próximas perguntas fazem referência à resposta desta. Nelas você também vai ter um template.

#1 – What – o quê / do quê?

Template: Aconteceu _____. Do quê?

Aconteceu uma venda. Do quê? De Coca-Cola.

Aconteceu uma demissão. Do quê? De um funcionário.

UMA COCA-COLA FOI VENDIDA

#2 When – quando?

Template: Aconteceu _____. Quando?

Aconteceu uma venda. Quando? 01/02/2017.

Aconteceu uma demissão. Quando? 01/01/2017.

**UMA COCA-COLA FOI VENDIDA NO
DIA 01/02/2017**

#3 Where – onde?

Template: Aconteceu _____. Onde?

Aconteceu uma venda. Onde? Na loja de São Paulo.

Aconteceu uma venda. Onde? Em Porto Alegre.

Aconteceu uma demissão. Onde? No departamento de marketing.

**UMA COCA-COLA FOI VENDIDA, NO
DIA 01/02/2017, EM SÃO PAULO**

#4 Who – quem?

“Quem” são as pessoas envolvidas com o fato, e o papel delas varia de acordo com o negócio.

Template: Aconteceu _____. Para quem?

Template: Aconteceu _____. Quem entregou?

Aconteceu uma venda. Para quem? Pedro

Aconteceu uma venda. Quem fez a venda? Joana

Aconteceu uma venda. Quem entregou? Transportadora Mercúrio.

Aconteceu uma demissão. De quem? Pedro

Aconteceu uma demissão. Quem fez a demissão? Joana

**JOANA VENDEU UMA COCA-COLA
PARA O PEDRO, NO DIA
01/02/2017, EM SÃO PAULO E A
ENTREGA FOI REALIZADA PELA
TRANSPORTADORA MERCÚRIO**

#5 How – como?

Template: Aconteceu _____. Como?

Aconteceu uma venda. Como? Com pagamento em cartão.

Aconteceu uma venda. Como? Em dinheiro.

Aconteceu uma venda. Como? Visitando o cliente.

**JOANA VENDEU UMA COCA-COLA
PARA O PEDRO, QUE PAGOU COM
CARTÃO, NO DIA 01/02/2017, EM
SÃO PAULO E A ENTREGA FOI
REALIZADA PELA
TRANSPORTADORA MERCÚRIO**

#6 Why – por quê?

Template: Aconteceu _____. Por quê?

Aconteceu uma venda. Por quê? Por causa de uma promoção de natal.

Aconteceu uma venda. Por quê? Por causa de uma promoção de Black Friday.

Aconteceu uma venda. Por quê? Porque lançamos um produto novo.

**JOANA VENDEU UMA COCA-COLA
PARA O PEDRO, QUE PAGOU COM
CARTÃO, EM UMA PROMOÇÃO DE
NATAL, NO DIA 01/02/2017 E A
ENTREGA FOI REALIZADA PELA
TRANSPORTADORA MERCÚRIO EM
SÃO PAULO**

As 2 últimas perguntas, ao invés de focar no fato em si, analisam a ocorrência dele.

#7 How often – com que frequência?

Essa pergunta é diferente de “quando”, porque ela não analisa só a data, mas a frequência com que o fato se repete. Essa pergunta vai nos ajudar a entender a periodicidade que a carga vai precisar ser feita.

Template: Com que frequência acontece _____?

Com que frequência acontece uma venda? A cada 3h.

Com que frequência acontece uma demissão? Uma vez por mês.

#8 How many – quanto / quantas?

Template: Quantas _____ aconteceram?

Template: Quanto foi _____?

Quantas vendas aconteceram? 2.

Quanto foi vendido? R\$200,00.

Quantas demissões aconteceram? 1.

PASSO 2 - IDENTIFICAR A FATO

Com a entrevista feita, é hora de organizar essas informações.

A tabela fato possui 2 elementos:

- foreign keys, que conectam a fato nas dimensões;
- métricas, que são sempre dados numéricos.

Essa parte é fácil. É só fazer a primeira pergunta corretamente que você já tem a sua fato definida.

FATO VENDA

FK DIM 1	FK DIM 2	FK DIM 3	FK DIM 4	MÉTRICAS

PASSO 3 - IDENTIFICAR AS MÉTRICAS

Depois de ter a fato definida, você precisa identificar as métricas dela.

E para isso, você pode usar as respostas da pergunta #8 - How many (quanto / quantas), onde você tem a quantidade de vendas e o valor delas.

FATO VENDA

FK DIM 1	FK DIM 2	FK DIM 3	FK DIM 4	MÉTRICAS
				Quantidade de venda Valor da venda

PASSO 4 - IDENTIFICAR AS DIMENSÕES

As dimensões possuem 3 elementos:

- surrogate key, que é a primary key da dimensão;
- natural key, que é a primary key da origem;
- atributos, que vão qualificar as métricas da fato.

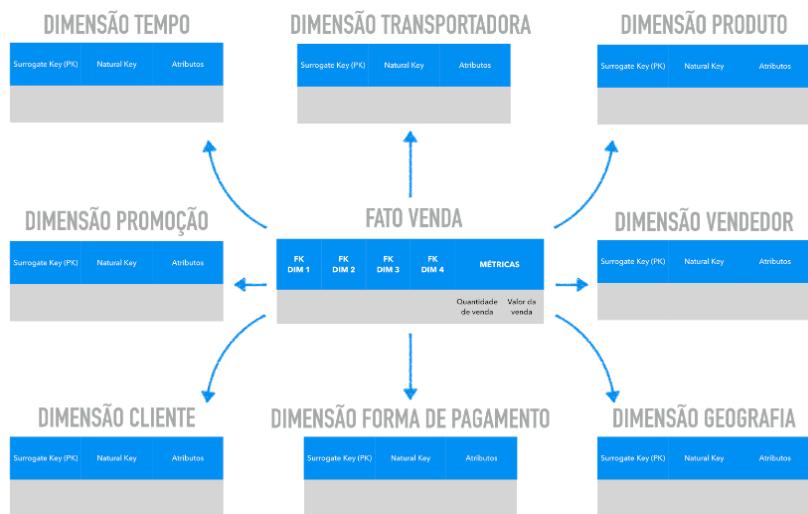
As outras perguntas da entrevista vão definir suas dimensões.

Na frase final, você já consegue identificar as dimensões:

Joana vendeu uma **Coca-Cola** para o **Pedro**, que **pagou com cartão**, em uma **promoção de natal**, no dia **01/02/2017** e a entrega foi realizada pela **Transportadora Mercúrio** em **São Paulo**.

- Joana = vendedor;
- Coca-Cola = produto;
- Pedro = cliente;
- pagou com cartão = forma de pagamento;
- promoção de natal = promoção;
- 01/02/2017 = tempo;
- Transportadora Mercúrio = transportadora;
- São Paulo = geografia.

E com esse levantamento, você já foi capaz de identificar 8 possíveis dimensões do seu Star Schema.



PASSO 5 - DEFINIR A HIERARQUIA

Com as dimensões identificadas, você precisa definir a hierarquia e o grão de cada uma delas.

Aqui é onde você deve entender exatamente o que o key user quer ver com aquela informação e se certificar de qual é o menor nível que ele vai precisar ver.

Para essa parte, é fundamental que você entenda como funciona a hierarquia de dimensões.

Na dimensão produto, você teria a seguinte hierarquia:

- categoria
- subcategoria
- produto

E isso você identifica conversando com os key users.

DIMENSÃO PRODUTO

Surrogate Key (PK)	Natural Key	Categoria	Subcategoria	Produto	Código	Peso	Unidade de medida
	Bebidas	Refrigerante	Coca-Cola	P0001	600	ml	

PASSO 6 - IDENTIFICAR OS ATRIBUTOS

Com as dimensões definidas, é hora de identificar os atributos de cada uma delas. Aqui você começa a fazer as perguntas para saber o que os key users precisam nas dimensões.

Pergunta assim: “Nessa primeira pergunta você respondeu Coca-Cola, que é um produto. O que você gostaria de analisar do produto?”

E entrevistando as pessoas envolvidas com o fato, você vai identificar os atributos necessários.

Você vai pegar informações como:

- Coca-Cola é o nome do produto;
- todo produto tem um código;
- o time de marketing vai dizer que precisa saber a marca do produto;
- o cara da logística vai precisar saber o peso do produto e a unidade de medida para planejar o caminhão.

DIMENSÃO PRODUTO

Surrogate Key (PK)	Natural Key	Atributo 1	Atributo 2	Atributo 3	Atributo 4	Atributo 5	Atributo 6
		Nome	Código	Marca	Peso	Unidade de medida	Preço unitário

Nessa parte de levantamento de requisitos, tem 2 coisas que acontecem com frequência que podem estragar o seu projeto:

1. Às vezes algum diretor ou gerente não vai poder estar na reunião ou participar do levantamento de requisitos e vai mandar alguém para representar ele. Esse diretor ou gerente que não foi é um key user, e é muito importante que a pessoa que estiver no lugar dele tenha conhecimento o suficiente do

negócio para responder essas perguntas. Se não, você vai acabar com informações capengas.

2. Projeto de Data Warehouse não é projeto de TI. Projeto de Data Warehouse é projeto de negócio. Já estou careca de ver projeto de Data Warehouse/BI em que a equipe fica socada junto com a TI dentro de um porão achando que é só pegar dados e jogar em um banco de dados. Cagada total.

Esse planejamento é extremamente importante. Empresas grandes e mais maduras já entendem o poder disso. Várias das empresas grandes pelas quais eu passei, como SBT, Sodexo e Renner, têm essa ideia de business first, então é sempre o negócio que está em foco, e não a TI. O importante agora não é saber se o cara já tem o dado coletado, se o banco aceita select ou se é MDX, isso é para outra hora.

Se você entender a estratégia que estou explicando aqui, vai poder aplicar ela da forma que for, porque é muito simples, basta seguir esse passo a passo e você vai conseguir entregar um Data Warehouse muito mais rápido e menos suscetível a falhas do que a maioria das pessoas.

BUS MATRIX

Matrix vem de matriz, e bus é porque por muito tempo surgiu um conceito chamado Enterprise Bus Architecture, que seria uma arquitetura empresarial, ou seja, de toda a organização, para que você possa controlar a empresa como um todo.

Isso acontece quando você tem muitas fatos, dimensões ou áreas de negócio. É como você vai controlar as expectativas de cada departamento. Assim, quando for alterar alguma dimensão, sabe se isso vai afetar algum lugar.

Ele também pode ser chamado de MID, que é o Mapa de Indicadores e Dimensões.

O BUS Matrix é uma ferramenta simples, mas muito poderosa se você utilizar ela com sabedoria.

Trouxe um exemplo para explicar como funciona o BUS Matrix.

BUS MATRIX POR PROCESSO DE NEGÓCIO

Isso aqui é uma etapa depois que você já levantou as dimensões.

	A	B	C	D	E	F	G	H	I
1	busmatrix por processo de negócio	Tempo	Produto	Cliente	Forma Pgto	Campanha	Fornecedor	Loja	CD
2	Vendas	X	X	X	X	X	X	X	

Aqui é um BUS Matrix por processo de negócio, ou seja, é aqui que você faz aquelas perguntas: qual o assunto, qual o processo, qual o business que vai ser analisado.

E esses X na tabela é simplesmente para marcar quais dimensões que você vai utilizar para aquele processo de negócio.

Por exemplo, a dimensão de centro de distribuição (CD), é utilizada para controle de estoque, então na venda não vai marcar ela.

Depois, vai projetar outro processo de negócio para a empresa, como o contas a pagar, que você quer analisar por tempo, forma de pagamento e fornecedor. E o contas a receber, que vai ser analisado por tempo, produto, cliente e forma de pagamento.

	A	B	C	D	E	F	G	H	I
1	busmatrix por processo de negócio	Tempo	Produto	Cliente	Forma Pgto	Campanha	Fornecedor	Loja	CD
2	Vendas	X	X	X	X	X	X	X	
3	Contas a Pagar	X			X		X		
4	Contas a Receber	X	X	X	X				
5	Inventário de Estoque	X	X					X	X

Isso serve para:

- você ter uma visão completa do Data Warehouse de forma bem simples, uma visão funcional, de negócio;

- você consegue, olhando assim, gerenciar as expectativas das pessoas que utilizam as dimensões, os filtros das dimensões e quem quer ver aquela informação.

Então, por exemplo, o time de vendas entende o produto de um jeito e o time de estoque entende de outro. Ou seja, para galera de vendas, uma certa característica é importante, enquanto para o pessoal de estoque é outra.

Vendas precisa saber do produto qual o valor unitário, a marca, e a descrição. Agora o pessoal do estoque precisa ver qual o peso do produto e o tipo de unidade de medida, porque o time de estoque tem que controlar o espaço no CD ou no caminhão.

O objetivo do BUS Matrix é permitir que você veja todos os setores que estão envolvidos para conseguir projetar algo que seja útil para os dois. Se não, quando você mexer na dimensão, vai ter que criar outra dimensão de produto para o outro setor.

O estoque quer ver peso e as vendas querem ver a marca, beleza, coloca as duas informações na mesma dimensão. E se cada um precisa ver a hierarquia de um jeito, duas hierarquias na mesma dimensão.

Você nunca vai ter duas dimensões iguais, como duas de produto, por exemplo.

É muito importante você ter essa visão quando seu Data Warehouse está crescendo. Então se você adicionar mais uma área de assunto ou mais uma fato, é extremamente recomendado que utilize esse tipo de ferramenta. Você fica com o mapa completo e é bem simples, faz em uma planilha mesmo.

BUS MATRIX POR MÉTRICAS E DIMENSÕES

Outra forma de analisar é descer um pouquinho mais nessa linha de raciocínio do BUS Matrix. Você vai ver muitos consultores fazendo também um BUS Matrix por métricas e dimensões. Basicamente é para ver o cruzamento das métricas com as dimensões. Por exemplo, aqui no processo de vendas, você faz um desdobramento desse BUS Matrix para métricas e dimensões.

	A	B	C	D	E	F	G	H	I	J	K	L
1	busmatrix por processo de negócio	Tempo	Produto	Cliente	Forma Pgto	Campanha	Fornecedor	Loja	CD			
2	Vendas	X	X	X	X	X	X	X				
3	Contas a Pagar	X			X		X					
4	Contas a Receber	X	X	X	X							
5	Inventário de Estoque	X	X					X	X			
6	Campanhas de Ads	X	X	X		X						
7												
8	busmatrix por métricas e dimensões	Tempo				Produto			Cliente	Forma Pgto	Campanha	
9	Hierarquia/grão	Ano	Semestre	Trimestre	Mês	Dia(grão)	Categoria	SubCategoria	produto(grão)			
10	Quantidade Vendida				X	X			X	X	X	X
11	Quantidade Vendida				X				X	X	X	X
12	Valor Unitário do produto				X				X	X	X	X
13	Valor da Venda				X				X	X	X	X

Aqui nas métricas você tem a quantidade vendida, valor unitário do produto e valor da venda, que são as métricas que foram levantadas no processo de vendas. Seguindo a mesma ideia, abre também as dimensões.

A diferença é que ao invés de marcar o X na dimensão de tempo, vai marcar o X no grão que aquela dimensão vai usar. Por exemplo, na quantidade vendida, precisa ver por dia. E também quer ver por cliente, forma de pagamento e campanha, que são dimensões sem hierarquia, então não precisa marcar qual o grão.

Depois você vai falar com a Joana do marketing, que vai dizer que precisa ver a quantidade vendida e o valor da venda só por categoria, e com o Pedro do estoque, que vai dizer que precisa saber a quantidade vendida pelo produto. E você vai marcar isso na planilha.

O que isso quer dizer? Mesmo que a Joana tenha pedido para ver a quantidade de vendas só pela categoria, o Pedro do estoque quer saber pelo produto, então o grão precisa ser o produto.

E isso serve para as outras métricas também. Assim você consegue ter uma visão mais detalhada do que as pessoas estão pedindo.

BUS MATRIX POR FATO E DIMENSÕES

Outro modelo que você pode ter aqui é por fato e dimensões. Por exemplo, você pegou o processo de negócio e fez um BUS Matrix por métrica. Agora, dentro desse processo de vendas, depois que você entender as métricas, vai mapear as fatos, que são:

- venda diária;
- venda agregada mensal;
- venda projetada;
- venda realizada.

	A	B	C	D	E	F	G	H	I	J	K	L
1	businatriz por processo de negocio	Tempo	Produto	Cliente	Forma Pgto	Campanha	Fornecedor	Loja	CD			
2	Vendas	X	X	X	X	X	X	X				
3	Contas a Pagar	X	X			X		X				
4	Contas a Receber	X	X	X	X							
5	Inventário de Estoque	X	X						X	X		
6	Campanhas de Ads	X	X			X						
7												
8	businatriz por métrica e dimensões											
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

	A	B	C	D	E	F	G	H	I	J	K	L
1	businatriz por fato e dimensões	Tempo	Produto	Cliente	Forma Pgto	Campanha	Fornecedor	Loja	CD	cenário		
2	Venda_diana	X	X	X	X	X	X	X		X		
3	venda_agg_mensal	X	X	X								
4	venda_projetada	X	X	X	X	X	X	X				
5	venda_realizada	X	X	X	X	X	X	X				

E você vai marcando com X para saber o que cruza com o que.

O importante aqui é que você entenda os fundamentos para ver qual é o melhor cenário para cada momento. Você não precisa criar as 3 BUS Matrix sempre, depende muito do projeto. Se você vai pegar um projetinho que vai ter só uma fato, não faz sentido mapear isso porque já vai estar no modelo. Agora, se você precisa passar isso para o key user, que quer uma visão mais macro, você faz um BUS Matrix por processo de negócio.

Eu recomendo sempre usar é quando tem mais de um processo de negócio ou mais de uma fato, para você saber que não vai dar problema quando fizer alterações.

FUNDAMENTOS DO ETL

Antes de entrar na parte avançada da modelagem dimensional, quero passar com você pelo ETL.

Na verdade, eu já venho falando algumas coisas sobre ETL ao longo do livro, mas este Passo é para os detalhes, as entradas do ETL. Vou mostrar como a gente faz para carregar o Data Warehouse independente da ferramenta.

Esta forma que eu vou ensinar para você é direto no banco de dados. A gente vai abrir o DBeaver e eu vou ensinar você a fazer as tabelas e inserir no banco, tudo na unha para que você entenda o processo como um todo, como funciona cada etapa do ETL.

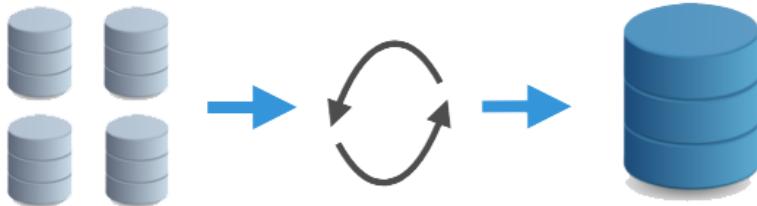
Quando você estiver no seu projeto de Data Warehouse, independente do banco de dados ou da ferramenta, esses serão os passos que precisarão ser feitos para o Data Warehouse funcionar como um reloginho.

Aqui vou cobrir com você o que é o ETL, quais seus objetivos, os componentes e as variações que existem em um processo de integração de dados.

O QUE É ETL?

Só para a gente conceitualizar, muita gente confunde ETL como somente integração de dados. Agora, integração de dados, no bem popular da palavra, é tirar o dado de um lado e jogar para outro. O ETL não é só isso.

O processo de ETL para Data Warehouse segue 3 passos:



Passo 1 – Extract / Extrair

Responsável por extrair os dados do sistema de origem, do próprio Data Warehouse ou do cubo, depende do contexto.

Por exemplo:

Quando eu estava trabalhando no projeto da Renner, tinha uma ferramenta que se chamava Hyperion, uma ferramenta de

planejamento orçamentário. E dentro dessa ferramenta existia um banco de dados multidimensional chamado Essbase. Alguns dos processos tinham que ler esse banco de dados multidimensional e carregar no Data Warehouse. Outras vezes, você tem o Data Warehouse e precisa fazer uma fato agregada, mas para isso acontecer, precisa tirar os dados da fato diária e colocar na fato mensal. Ou seja, vai extrair da fato.

Outra coisa que o “E” faz é extrair os dados dos sistemas de origem como ERP, CRM e outros sistemas transacionais.

Em um processo clássico de ETL, o principal objetivo desta etapa é liberar o sistema de origem do processamento do ETL. Ou seja, no momento em que a gente vai lá no legado e tira esses dados para só depois começar a trabalhar neles, a gente libera o sistema de produção desse processamento.

Passo 2 – Transform / Transformar

É aqui no “T” que a gente fala em transformação. Mas que transformação é essa? Vai transformar em um Power Ranger?

É neste momento que a gente:

- calcula métricas derivadas;
 - por exemplo: multiplica valor unitário com quantidade vendida para identificar o valor da venda;
- faz a limpeza dos dados;
 - por exemplo: troca campos nulos nas métricas para 0;
- unifica os dados;

- por exemplo: em um lugar está escrito “m”, em outro está “masculino” e em outro está “masc”. É aqui que faz a limpeza e coloca todo mundo como “m”;
- seleciona apenas as colunas que precisa;
 - por exemplo: digamos que liberaram uma tabela com duzentas colunas no legado, e você só precisa de três. É aqui que vai selecionar só essas três;
- faz o split dos dados;
 - por exemplo: pode transformar linha em coluna, coluna em linha e por aí vai;
- faz a mescla dos dados;
 - por exemplo: tem uma tabela de cliente no sistema A e outra no sistema B. É nesse momento que se faz a mescla das informações das duas tabelas de clientes;
- valida os dados;
 - por exemplo: aplica algumas regras para ter certeza de que não vai entrar sujeira. Se o registro está nulo, se a carga deu problema, se tem que abortar por algum motivo ou descartar tudo e não fazer mais nada. É tudo aqui;
- aplica o controle dos dados;
 - por exemplo: gera as surrogate keys, os controles de geração da carga, atualiza os campos de data da carga, data de inserção, data inicial e data final.

Passo 3 – Load / Carregar

O “L” carrega os dados tratados em um Data Warehouse, em uma staging area ou em uma base de dados. O load é o simples fato de carregar do outro lado tudo aquilo que você fez.

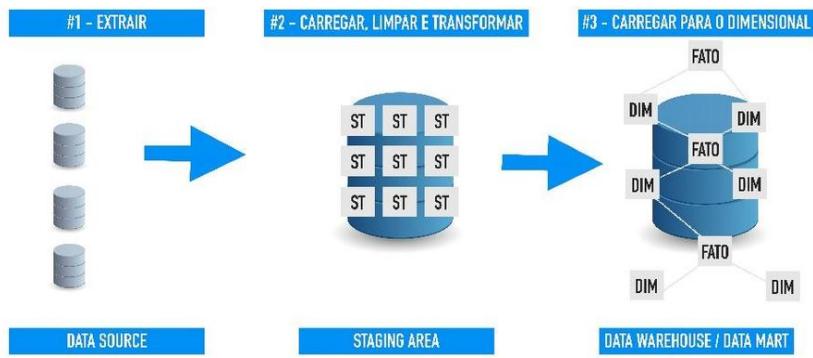
No Data Warehouse a gente lê de uma staging area e carrega em um modelo dimensional o que foi tratado na fase anterior.

Tem algumas coisas que você pode fazer na parte do load, como:

- fazer cargas em lotes;
 - por exemplo: você carrega 1.000 registros, depois mais 1.000 registros e depois mais 1.000 registros. E se falhar, porque faltou luz ou a tiazinha que limpa chutou o servidor, você já garantiu que 3.000 registros foram inseridos. Agora, através da ferramenta, do processo de ETL, você pode controlar qual é a próxima carga;
- carregar de forma distribuída;
 - por exemplo: eu posso ter 5 servidores carregando a mesma informação e cada um carregar um pedaço. É em cluster. Assim você consegue carregar muito mais rápido. Lógico, vai precisar de muito mais máquinas e a carga vai ficar mais complexa.

O QUE É STAGING AREA?

Neste ponto você provavelmente já sabe o que é a staging area, mas não custa a gente repassar antes de entrar nos detalhes de todo o processo.



Esse desenho você já viu, que é a arquitetura do Data Warehouse. Agora, antes de eu mostrar para você o processo de ETL, eu preciso garantir que você saiba o que é staging area.

A staging area é um banco de dados relacional sem nenhum relacionamento, são todas tabelas burras. E ela não está conectada na origem. São tabelas que recebem os dados da origem.

Na staging area é o momento que vamos fazer a bagunça. Ou, para quem é mais antigo, é a cozinha. É aqui que vamos fazer a bagunça para carregar os dados.

No processo do ETL, o “E” vai extrair as informações do data source e colocar na staging area. O “T” fica na staging area, ou seja, é nela que nós vamos transformar os dados.

A staging area é um banco de dados separado da origem, e você tem duas opções para fazer ela:

- servidores separados;
- mesmo servidores, mas em schemas separados.

Eu já vi pessoas colocando no mesmo schema, mas aí vira uma zona só.

E por que a gente precisa usar a staging area?

- para não usar os bancos de dados da origem;
 - quando você faz o trabalho direto no sistema de origem, o sistema de produção vai ficar lento;
- os sistemas de origem só estão disponíveis para extração durante um intervalo de tempo específico;
 - na Renner, por exemplo, eu tinha um projeto onde a janela de tempo para extrair os dados era da 0h às 6hs. Isso parece bastante tempo, mas todo o processo de ETL demorava 8 horas;
- você vai precisar executar uma consulta SQL unindo duas tabelas de dois bancos de dados fisicamente diferentes;

- ou seja, mesmo que você estivesse processado dados de uma base de dados, você pode querer dados de outra base. E você não vai poder fazer um dblink que vai conectar no outro banco. O cliente não vai te largar isso, porque são dados de produção;
- geralmente, cada fonte de dados tem sua própria janela para a extração de dados;
 - no caso do projeto da Sodexo, que é uma empresa francesa de cartão de alimentação, a janela era diferente da do Brasil. Cada momento liberava um horário;
- a frequência de carregamento de dados do Data Warehouse não corresponde às frequências de atualizações dos sistemas de origem;
 - muitas vezes você vai pegar legados com períodos de carregamentos diferentes. Um libera às 21hs, outro libera às 15hs e outro às 21hs, ou podem liberar em dias diferentes também. Então, no momento em que eles liberam, eu jogo todos os dados na staging area;
- o processo de ETL envolve transformações de dados complexas e precisa de tempo e recurso computacional para o processamento;
 - imagina que você precisa cruzar uma tabela de 3GB com outra tabela de 2GB. Esse processo demora e precisa de máquina para processar, e as maquininhas que estão no sistema de origem não estão preparados para isso.

Então, é por isso que a gente precisa da staging area. Ela é muito importante apesar de muita gente não dar a devida relevância para ela.

VARIACÕES DO ETL

Agora que você já entende o que é ETL e quais são as suas etapas, é importante eu explicar as variações dele.

Existem várias nomenclaturas que deram para esse processo de tirar o dado de um lado e jogar para outro, então existem diversas variações.

Eu vou apresentar 2 aqui para você, que são as mais utilizadas:

- ET-L;
- EL-T.

Você vai ver algumas pessoas defenderem uma forma ou outra de fazer. Eu trabalhei muito tempo com uma ferramenta que diz ser ET-L e com outra que diz ser EL-T, e ambas eram muito boas e faziam exatamente a mesma coisa. O que acontece é que dependendo da arquitetura que você quer implementar, uma é melhor que a outra, por isso é importante você entender como cada uma funciona, para poder escolher a ferramenta que vai se adequar melhor para o seu projeto.

ET-L

Aqui você na verdade vai realizar o ET-L duas vezes, por isso são 6 passos que você precisa seguir para levar os dados do data source para o Data Warehouse:

- passo 1 (E) - extrair os dados do data source;
- passo 2 (T) - transformar os dados;
- passo 3 (L) - inserir os dados na staging area;
- passo 4 (E) - extrair os dados da staging area;
- passo 5 (T) - transformar os dados;
- passo 6 (L) - inserir os dados no Data Warehouse.

Como eu expliquei no capítulo sobre staging area, uma das funções dela é liberar o legado do processamento das transformações. Então, a transformação do passo 2 já é feita no servidor da staging area, mas como a conexão com o data source ainda está ativa, você ainda não liberou os bancos de dados do legado. Por isso, você só vai fazer limpeza de coisinhas básicas. Por exemplo: você já quer transformar tudo que está vindo com a letra “m” no sexo em “masculino”.

Na transformação do passo 5, você já liberou o data source, então pode fazer toda aquela transformação pesada, a limpeza dos dados e preparar a surrogate key.

Quando você extrai os dados, basicamente, você vai tirar os dados do banco onde eles estão e jogar para uma área temporária, para depois carregar os dados em um novo banco de dados. Nessa área temporária, tudo fica na memória e as transformações são feitas em tempo de execução. Os únicos dados que realmente existem é quando

extraí e quando carrega. Os dados intermediários não existem, não estão de fato armazenados.

EL-T

O EL-T vai extraír, carregar e depois transformar.

Assim como no anterior, você vai fazer o processo do EL-T duas vezes, seguindo 6 passos:

- passo 1 (E) - extraír os dados do data source;
- passo 2 (L) - inserir os dados na staging area;
- passo 3 (T) - transformar os dados;
- passo 4 (E) - extraír os dados da staging area;
- passo 5 (L) - inserir os dados no Data Warehouse;
- passo 6 (T) - transformar os dados.

Lembrando novamente que não pode fazer nenhuma transformação no data source. Depois que carregar os dados na staging area, já pode fazer alguma transformação se quiser.

Toda essa diferença de arquitetura é muito pesada de fazer na mão. Então quando você utiliza uma ferramenta, tudo isso fica bem mais simples. Porque a ferramenta entende onde está o banco de dados e em que momento ela precisa extraír e carregar na memória. Enquanto você vai dizendo para a ferramenta o que ela pode ou não fazer.

Como eu falo, não tem uma variação melhor que a outra. O importante é que você entenda a arquitetura que você vai

implementar para depois poder ver qual a ferramenta que se adequa mais ao que você está precisando. Mas lógico, isso não quer dizer que uma não faça o trabalho da outra.

Se você der um Google e pesquisar as variações do ETL, vão aparecer umas 10 nomenclaturas: ETL, ELT, ELTT, ETTL, etc., mas todas elas explicam como os dados vão ser levados do data storage para o Data Warehouse.

No início parece um pouco complexo, um pouco complicado, mas o que eu quero que você frise bem nesse capítulo aqui é:

- você vai tirar os dados do data source sem processar nada no servidor dele;
- você vai carregar os dados na staging area e lá você pode fazer o diabo, vai processar, deletar, trabalhar o que precisa ser trabalhado;
- quando estiver pronto, você vai carregar os dados para o modelo dimensional.

PROCESSO DE ETL

Agora eu quero mostrar para vocês como que funciona o processo de ETL. As ferramentas escondem muita coisa do processo porque são feitas para facilitar o nosso trabalho, então se você aprender direto na ferramenta, vai perder muita coisa, não vai entender como que tudo funciona.

Qual o problema que eu vi em diversos projetos? Vi a consultoria pagar um curso para a pessoa aprender a usar uma ferramenta.

Por exemplo, na Intellego eles tinham um time pesado de pessoas que faziam integração de dados. No último projeto que eu participei com eles, a pessoa que foi enviada entendia de Oracle Data Integrator.

Eu falei para o cara: “aqui a gente tem integração no nosso Data Warehouse e depois a gente vai colocar os dados aqui e tal. Não tem nada muito complexo, é bem simples: só vai carregar duas ou três dimensões e a fato.”

E o cara falou: “beleza, isso é tabela, né? ”

O que acontece, ele estava em um projeto de integração de um outro cliente que usava ODI para fazer a integração dos dados de um

sistema legado para outro. Simplesmente era a cópia de uma tabela e uma carga na outra. Acabou, é isso que era feito.

O projeto começou a rodar, e eu comentei com os gestores da consultoria que a pessoa precisa entender dos processos, por que se não ia dar problema. A consultoria disse que já tinha passado alguma coisa para ele ler sobre o que eram fatos e dimensões.

Chegou no projeto e não deu outra, começou a dar pau. Os pessoal não consegui fazer o simples, que é carregar uma dimensão, porque não entendiam dos processos que acontece no passo a passo do ETL. O responsável pelo ETL estava com pensamento de carga de legado e tentava fazer isso igual no Data Warehouse e não funcionava. A ferramenta até colocava registos na tabela... mas uma coisa que eu gosto de falar sempre: se você pegou a ferramenta, leu o dado de um lado, colocou o dado na outra ponta e apareceu o verdinho lá com *check* dizendo que tudo deu certo, não garante que a integração funcionou, você precisa validar esse processo, garantir que se havia 50 laranjas de um lado, tem 50 laranjas do outro lado.

Como eu sempre falo, a gente precisa entender os fundamentos. Não se aprende a correr sem antes aprender a engatinhar. Você não nasceu correndo. Você aprendeu a engatinhar, a caminhar e tudo mais.

É isso que quero fazer com você aqui. Você vai ver como é simples. Eu já vou te passar os scripts prontos para te mostrar como se faz. E você replica isso no seu banco, porque o objetivo é você entender como cada coisinha funciona.

Você não vai precisar fazer isso sempre, é só para você aprender o processo. Existe uma grande diferença entre você ler e você fazer, colocar a mão na massa.

Em projetos da vida real, o ETL vai ser muito mais complexo do que eu vou mostrar aqui, e você não vai ficar escrevendo código na mão, vai usar uma ferramenta que facilite o trabalho. Aqui vós vamos passar por um processo simples de ETL para você entender o funcionamento.

O processo de ETL tem 6 passos:

- passo 0 – criar a origem;
- passo 1 – criar a staging area;
- passo 2 – carregar a staging area;
- passo 3 – criar as dimensões;
- passo 4 – carregar as dimensões;
- passo 5 – criar a fato;
- passo 6 – carregar a fato.

Eu vou passar com você por cada um deles.

Mas antes tenho algumas lembranças para você:

1. carregue somente o que você precisa. Por exemplo, eu tenho uma dimensão de cliente que eu quero nome, telefone, CPF e email. Você não vai trazer uma dimensão com 50 campos, você vai trazer só o que precisa, por três motivos, se você fizer isso:
 - vai afundar o seu processo de ETL;
 - vai aumentar a complexidade do ETL e a demora da carga. Aquela carga que poderia levar 1 minuto, vai levar 20 horas porque você quis trazer tudo;

- vai aumentar a complexidade da tomada de decisão do usuário. O que precisa ser analisado já foi definido antes.

Então, traz somente o que precisa, do jeito que precisa e põe na stage;

2. copie do legado com pouca ou nenhuma transformação. Simplesmente tire os dados de um lugar e coloque no outro, isso está ótimo. Às vezes, por exemplo, o legado possui o código do cliente com o tipo *varchar* e a staging area está esperando um *integer*. No momento dessa primeira carga, você já pode converter o tipo de dado. Não tem problema nenhum;
3. sem relacionamento (joins) entre as tabelas da stage. Por exemplo, eu tenho dois sistemas de origem, e os dois tem cadastros de clientes. A gente vai fazer uma junção desses dois cadastros, unificar e deixar bonitinho lá no Data Warehouse.

O que você vai ter na stage? A tabela de clientes do sistema A e a tabela de clientes do sistema B. Essas duas tabelas vão estar na staging area e depois você vai fazer toda essa mescla.

Algumas pessoas preferem fazer essa junção já na carga. Eu não gosto, porque digamos que o sistema de origem A libera para você extrair as informações às 21h e o sistema B só libera às 0h. Você não vai poder rodar aquela carga até os dois estarem liberados. Então traz tudo pra stage e só depois começa a fazer a brincadeira;

4. existem 3 formas de você criar a staging area:
 - na mão (direto no código);
 - através da ferramenta de ETL;

- algumas ferramentas já tem uma opção para quando você aponta onde é o sistema de origem, elas te perguntam onde você quer carregar esses dados, e quando você aponta no banco da stage, ele pergunta se você quer criar essa tabela com a estrutura que existe na origem e já cria para você;
- através da ferramenta de design;
 - você abre o Power Architect ou uma outra ferramenta de design e cria as suas tabelas visualmente. Lembra que não tem conexão nenhuma, você só vai jogando as tabelas. No projeto do SBT, lembro que fiquei, sem mentira nenhuma, duas semanas criando tabela de stage na ferramenta visual. Tem alguns clientes que querem toda a documentação do projeto, e você precisa estar com a stage documentada antes de passar para a parte de ETL.

Se alguém tivesse me explicado esse processo logo quando eu comecei a minha carreira, teria batido muito menos a cabeça. Mas eu bati a cabeça, aprendi e agora eu consigo passar para você com mais firmeza e tranquilidade porque eu sei que é isso que acontece.

Agora para você fazer o ETL, precisa abrir o DBeaver.

Você vai ver que é muito simples, não tem muito mistério, é um passo a passo. Para esse exemplo, eu estou conectado no meu banco de dados lá do PostgreSQL.

Na vida real, a staging area fica em um banco de dados separado do Data Warehouse. Eles podem estar no mesmo servidor ou em servidores distintos, aí vai da arquitetura.

Como aqui é só para exemplo, eu vou criar o Data Warehouse, a staging area e as tabelas do legado no mesmo lugar, para ficar mais fácil de você ver o processo acontecendo. E vou utilizar o mesmo banco de dados que a gente já vem usando.

PASSO 0 - CRIAR A ORIGEM

Na vida real, você nunca vai precisar criar a origem, isso é óbvio. Mas para o nosso exemplo, você vai precisar criar algumas tabelas que vão servir de legado nos nossos exemplos.

Eu vou colocar todos os comandos SQL aqui para você, e é muito importante para o seu processo de aprendizado que você escreva o código na mão. Se você está entrando nesta área de mexer em banco de dados, é fundamental dar uma dedilhada em um pouco de código para pegar os macetes. É importante.

Então o que você vai fazer nesse passo é:

1. criar 4 tabelas de exemplo do legado;
2. inserir os dados nas 4 tabelas.

1. Criar 4 tabelas de exemplo do legado

A primeira tabela vai se chamar *cadastro_cliente*, é um exemplo bem clássico de uma tabela de cliente que fica na origem.

Este é o script que você vai usar para criar essa tabela:

```
create table if not exists cadastro_cliente (
    cdcli int4 not null,
    nomecli varchar(100) not null,
```

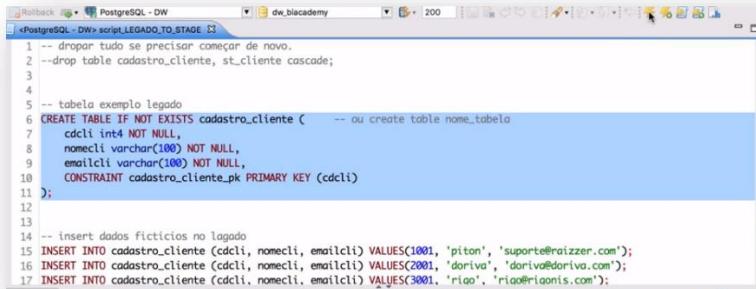
```
emailcli varchar(100) not null,  
constraint cadastro_cliente_pk primary  
key (cdcli)  
);
```

Os campos que você está criando aqui são:

- **cdcli** - o código do cliente;
 - Exemplo: 1001;
- **nomecli** - o nome do cliente;
 - Exemplo: piton;
- **emailcli** - o email do cliente;
 - Exemplo: suporte@raizzer.com.

OBS: o *if not exists* serve apenas para verificar se a tabela já não existe. Se não existir, ela vai ser criada.

Então você vai colocar esse código no DBeaver e vai executar clicando em *Execute SQL statement*, que é um raiozinho que fica na barra superior.



```

1 -- dropar tudo se precisar comegar de novo.
2 --drop table cadastro_cliente, st_cliente cascade;
3
4
5 -- tabela exemplo legado
6 CREATE TABLE IF NOT EXISTS cadastro_cliente ( -- ou create table nome_tabela
7     cdcli int4 NOT NULL,
8     nomecli varchar(100) NOT NULL,
9     emailcli varchar(100) NOT NULL,
10    CONSTRAINT cadastro_cliente_pk PRIMARY KEY (cdcli)
11 );
12
13
14 -- insert dados ficticios no lagado
15 INSERT INTO cadastro_cliente (cdcli, nomecli, emailcli) VALUES(1001, 'piton', 'suporte@raizzer.com');
16 INSERT INTO cadastro_cliente (cdcli, nomecli, emailcli) VALUES(2001, 'doriva', 'dorival@doriva.com');
17 INSERT INTO cadastro_cliente (cdcli, nomecli, emailcli) VALUES(3001, 'rlao', 'rlao@rlaonis.com');

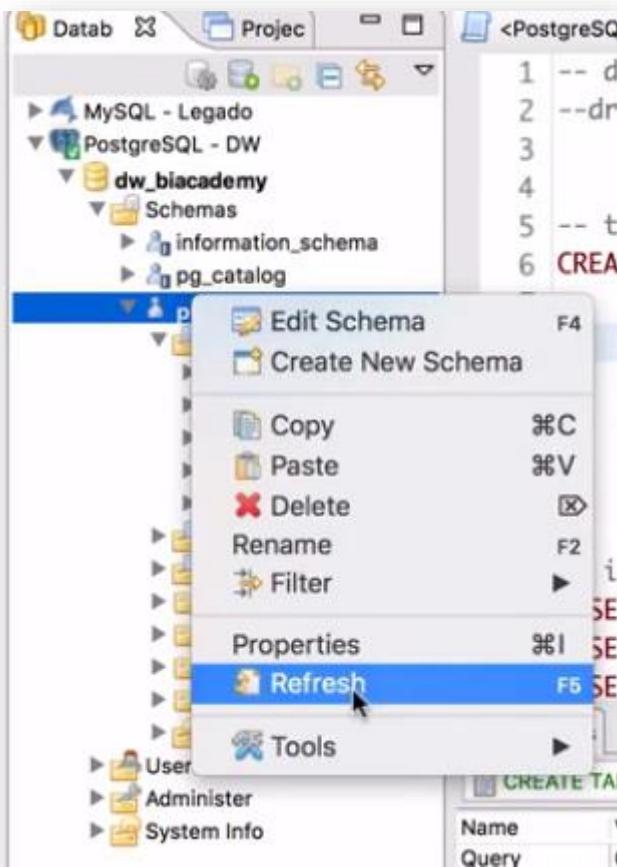
```

E ele vai mostrar o resultado dizendo que foi executado.

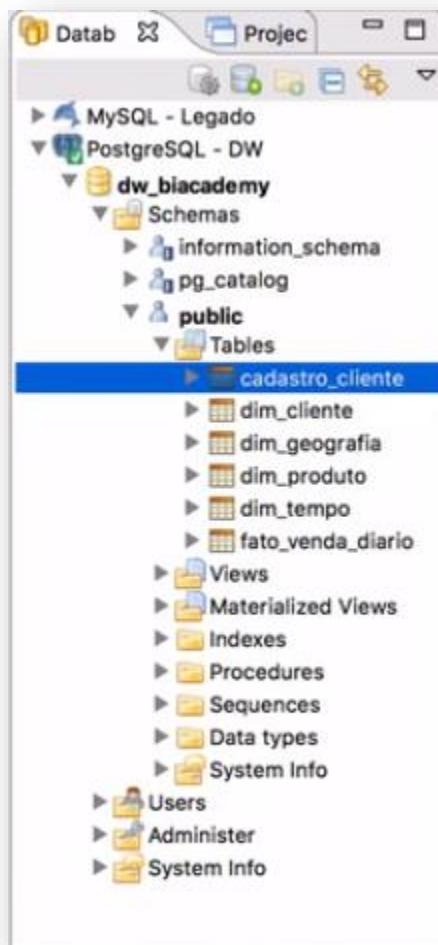


Name	Value
Query	CREATE TABLE IF NOT EXISTS cadastro_cliente (-- ou create table nome_tabela
Updated Rows	0
Finish time	Tue Jan 10 17:01:56 BRST 2017

Para validar se funcionou, na área da esquerda, você clica com o botão direito em *public*, dentro de *Schemas*, e clica em *Refresh*, para atualizar.



E o seu *cadastro_cliente* vai aparecer ali.



As outras três tabelas vão ser pedidos, itenspedido e produto, que é um modelo muito comum no mercado. O processo para criar elas é o mesmo que na anterior.

O código da pedidos é:

```
create table pedidos (
```

```
numeropedido integer not null,  
datapedido timestamp not null,  
datapagamentopedido timestamp default  
null,  
dataentregapedido timestamp default null,  
statuspedido varchar(100) not null,  
comentarios text,  
numerocliente integer not null,  
primary key (numeropedido)  
) ;
```

Os campos que você está criando aqui são:

- **numeropedido** - o código do pedido;
 - exemplo: 799898989;
- **datapedido** - a data que o pedido foi realizado;
 - exemplo: 16/01/2017;
- **datapagamentopedido** - a data do pagamento do pedido;
 - exemplo: 16/01/2017;
- **dataentregapedido** - a data que o pedido foi entregue;
 - exemplo: 25/01/2017;
- **statuspedido** - o status do pedido;
 - exemplo: pago;

- **comentários** - algum comentário sobre a venda;
 - exemplo: última peça;
- **numerocliente** - o código do cliente;
 - exemplo: 666666.

O código da *itenspedidos* é:

```
create table itenspedido (
    numeropedido integer not null,
    codigoproduto varchar(50) not null,
    quantidadepedido integer not null,
    precounitario double precision not null,
    numerolinhapedido smallint not null,
    primary key (numeropedido, codigoproduto)
);
```

Os campos que você está criando aqui são:

- **numeropedido** - o código do pedido;
 - exemplo: 799898989;
- **codigoproduto** - o código do produto;
 - exemplo: bm-120;
- **quantidadepedido** - a quantidade de itens do pedido;
 - exemplo: 03;
- **precounitario** - o preço unitário do produto;

- exemplo: 80.000,00;
- numerolinhapedido - nos sistemas transacionais, normalmente eles numeram as linhas do pedido, então se tiver mais de um produto no mesmo pedido, fica um em cada linha;
 - exemplo: 1.

O código da *produtos* é:

```
create table leg_produtos (
    cdproduto varchar(50) not null,
    nomeproduto varchar(70) not null,
    categoriaproduto varchar(50) not null,
    medidaproduto varchar(10) not null,
    fornecedorproduto varchar(50) not null,
    descricaoproduto text not null,
    qtdestoqueproduto integer not null,
    precoproduto numeric(10, 4) not null,
    statusprod varchar(100) not null,
    constraint leg_produtos_pk primary key
        (cdproduto)
);
```

Os campos que você está criando aqui são:

- cdproduto - código do produto;
 - Exemplo: s24_4278;
- nomeproduto - nome do produto;

- Exemplo: pizza 4 queijos;
- categoriaproduto - categoria do produto;
 - Exemplo: Pizza Salgada;
- medidaproduto - o tamanho do produto;
 - Exemplo: 45cm;
- fornecedorproduto - fornecedor do produto;
 - Exemplo: Casa das massas;
- descricaoproduto - descrição do produto;
 - Exemplo: queijo gorgonzola, cheddar, provolone, brie;
- qtdestoqueproduto - quantidade em estoque;
 - Exemplo: 45;
- precoproduto - preço do produto;
 - Exemplo: 48.81;
- statusProd - status do produto;
 - Exemplo: Ativo.

Agora você precisa rodar esses três códigos para as tabelas do legado.

2. Inserir os dados nas 4 tabelas

Aqui vai ser a mesma coisa, é só você rodar um script.

Os dados da tabela *cadastro_cliente* vão ser:

```
insert into cadastro_cliente (cdcli, nomecli,
emailcli) values(128, 'piton',
'suporte@raizzer.com');

insert into cadastro_cliente (cdcli, nomecli,
emailcli) values(2001, 'simone',
'simone@doriva.com');

insert into cadastro_cliente (cdcli, nomecli,
emailcli) values(3001, 'vinivius',
'vinicius@raizzer.com');

insert into cadastro_cliente (cdcli, nomecli,
emailcli) values(4001, 'adriana',
'adriana@raizzer.com');
```

Os dados da tabela *pedido* vão ser:

```
insert into pedidos
(numeropedido, datapedido, datapagamentopedido,
dataentregapedido, statuspedido, comentarios,
numerocliente)

values(10100, '2017-01-06 00:00:00.000', '2017-
01-06 00:00:00.000', null, 'aguardando
pagamento', null, 363);
```

```
insert into pedidos
(numeropedido, datapedido, datapagamentopedido,
dataentregapedido, statuspedido, comentarios,
numerocliente)
values (10101, '2017-01-09 00:00:00.000', '2017-
01-18 00:00:00.000', '2017-01-11 00:00:00.000',
'pago', 'passou em 2 cartoes', 128);

insert into pedidos
(numeropedido, datapedido, datapagamentopedido,
dataentregapedido, statuspedido, comentarios,
numerocliente)
values (10102, '2017-01-10 00:00:00.000', '2017-
01-18 00:00:00.000', '2017-01-20 00:00:00.000',
'pago', null, 181);
```

Os dados da tabela *itenspedido* vão ser:

```
insert into itenspedido
(numeropedido, codigoproduto, quantidadedepedido,
precounitario, numerolinhapedido)
values (10100, 's18_1749', 30, 171.7, 3);
```

```
insert into itenspedido
(numeropedido, codigoproduto, quantidadedepedido,
precounitario, numerolinhapedido)
values (10100, 's18_2248', 50, 67.8, 2);
```

```
insert into itenspedido
(numeropedido, codigoproduto, quantidadadepedido,
precounitario,numerolinhapedido)
values(10101, 's18_2325', 25, 151.28, 4);

insert into itenspedido
(numeropedido, codigoproduto, quantidadadepedido,
precounitario,numerolinhapedido)
values(10102, 's18_1342', 39, 123.29, 2);

insert into itenspedido
(numeropedido, codigoproduto, quantidadadepedido,
precounitario,numerolinhapedido)
values(10102, 's18_1367', 41, 50.14, 1);
```

Os dados da tabela *produto* vão ser:

```
insert into leg_produtos
(cdproduto,nomeproduto,categoriaproduto,medidap
roduto,fornecedorproduto,
descricaoproduto,qtdestoqueproduto,precoproduto
,statusprod)
values('s24_1937', 'pizza de milho', 'pizza
salgada', '45cm','casa das massas', 'queijo e
milho', 320, 24.50, 'ativo');

insert into leg_produtos
```

```
(cdproduto,nomeproduto,categoriaproduto,medidap  
roduto,fornecedorproduto,descricaoproduto,qtdes  
toqueproduto,precoproduto,statusprod)  
values('s24_4278', 'pizza 4 queijos', 'pizza  
salgada', '45cm','casa das massas', 'queijo  
gorgonzola, cheddar, provolone, brie', 45,  
48.81, 'ativo');  
  
insert into leg_produtos  
(cdproduto,nomeproduto,categoriaproduto,medidap  
roduto,fornecedorproduto,descricaoproduto,qtdes  
toqueproduto,precoproduto,statusprod)  
values('s18_4409', 'pizza calabresa', 'pizza  
salgada', '45cm','casa das massas', 'calabresa  
italiana', 22, 56.50, 'ativo');
```

Feito isso, os dados estão na sua origem.

Para validar se os inserts funcionaram, você só precisa dar um select nas tabelas para ver o conteúdo delas.

Para isso, é só rodar esses códigos, um de cada vez:

```
select * from cadastro_cliente;  
select * from pedidos;  
select * from itenspedido;  
select * from produtos;
```

E esse é o resultado da tabela de cliente:

	cdcli	nomedcli	emailcli
1	1.001	pilton	suporte@ralzzer.com
2	2.001	doriva	doriva@doriva.com
3	3.001	rgo	rgo@rgo.com
4	4.001	rob	rob@robhud.com

Da tabela de pedidos:

	numeropedido	datapedido	datapagamentopedido	dataentregapedido	statuspedido	comentarios	numerocliente
1	10100	2017-01-09 00:00:00	2017-01-08 00:00:00	[NULL]	Aguardando Pagamento	[NULL]	363
2	10101	2017-01-09 00:00:00	2017-01-11 00:00:00	[NULL]	Pago	passou em 2 cartões	128
3	10102	2017-01-10 00:00:00	2017-01-11 00:00:00	2017-01-11 00:00:00	Pago	[NULL]	188
4	10103	2017-01-29 00:00:00	2017-02-07 00:00:00	2017-02-02 00:00:00	Pedido Enviado	[NULL]	121
5	10104	2017-01-31 00:00:00	2017-02-08 00:00:00	2017-02-01 00:00:00	Pedido Enviado	[NULL]	141
6	10105	2017-02-11 00:00:00	2017-02-21 00:00:00	2017-02-12 00:00:00	Entregue	[NULL]	145
7	10106	2017-02-17 00:00:00	2017-02-24 00:00:00	2017-02-21 00:00:00	Entregue	[NULL]	278
8	10107	2017-02-24 00:00:00	2017-03-03 00:00:00	2017-02-26 00:00:00	Entregue	[NULL]	131
9	10108	2017-03-03 00:00:00	2017-03-12 00:00:00	2017-03-08 00:00:00	Entregue	[NULL]	385

Da tabela de itens do pedido:

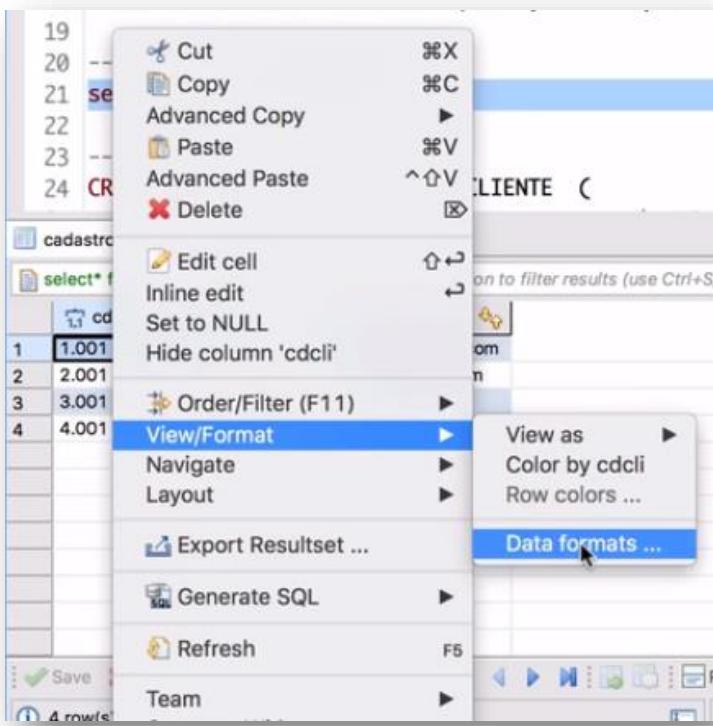
	numeropedido	codigoproduto	quantidadedepedido	precounitario	numerolinhapedido
1	10100	S18_1749	30	171,7	3
2	10100	S18_1048	50	67,8	2
3	10100	S18_4409	22	86,51	4
4	10100	S24_3969	49	34,47	1
5	10101	S18_2325	25	151,28	4
6	10101	S18_2795	26	145,13	1
7	10101	S24_1937	45	31,2	3
8	10101	S24_2022	46	53,76	2
9	10102	S18_1342	39	123,29	2
10	10102	S18_1367	41	50,14	1
11	10103	S10_1949	26	207,87	11
12	10103	S10_4962	42	128,53	4

E da tabela de produtos:

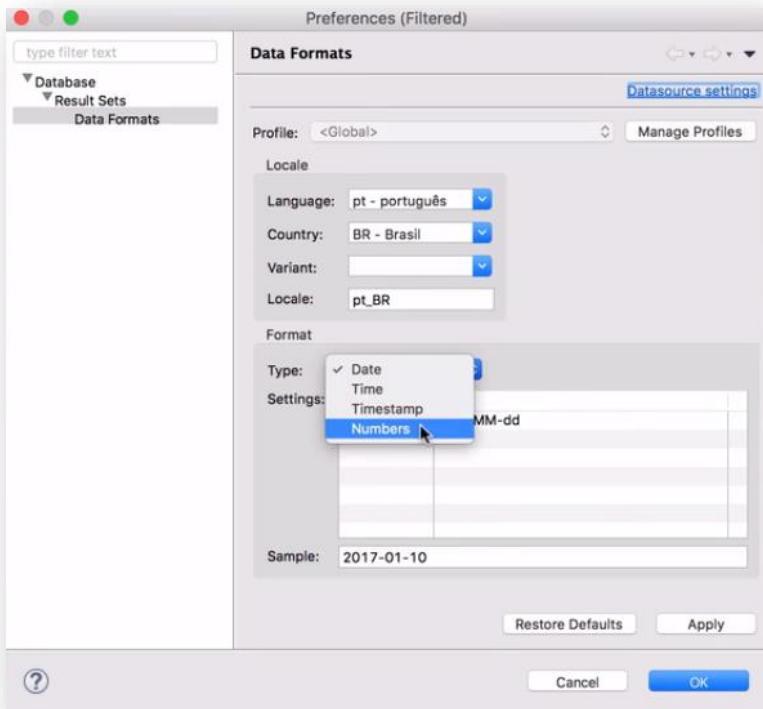
	numeropedido	codigoproduto	quantidadepedido	precounitario	numerolinhapedido
1	10100	S18_1749	30	171,7	3
2	10100	S18_2448	50	67,8	2
3	10100	S18_4409	22	86,51	4
4	10100	S24_3969	49	34,47	1
5	10101	S18_2325	25	151,28	4
6	10101	S18_2795	26	145,13	1
7	10101	S24_1937	45	31,2	3
8	10101	S24_2022	46	53,76	2
9	10102	S18_1342	39	123,29	2
10	10102	S18_1367	41	50,14	1
11	10103	S10_1949	26	207,87	11
12	10103	S10_4962	42	128,53	4

Detalhe: a ferramenta DBeaver e mesmo outras ferramentas, nos números inteiros, colocam ponto de milhar. Por isso, na coluna *cdcli* na tabela de cliente, está com aqueles pontos nos números.

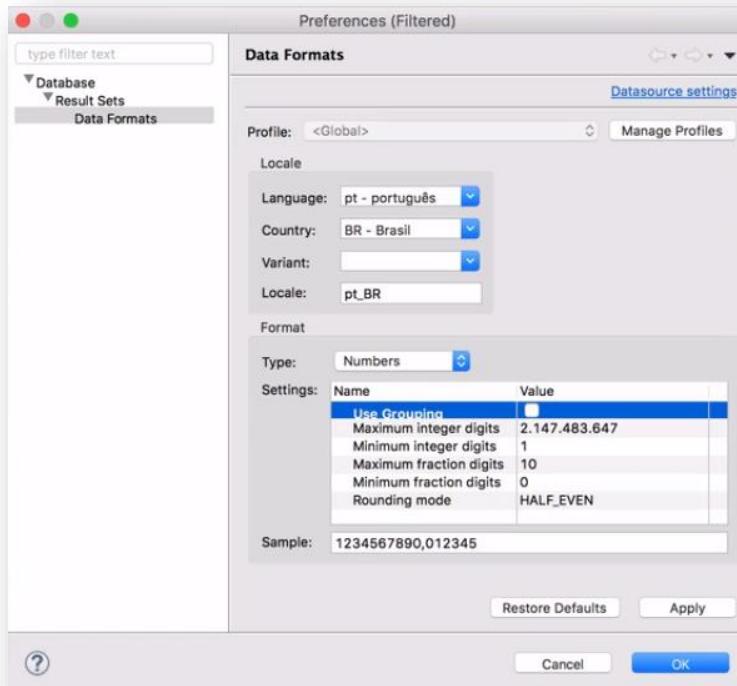
E como a gente resolve essa questão? Você pode apertar com o botão direito em cima da cédula, ir em *View/Format* e escolher a opção *Data Formats*.



Na janela que vai abrir, na opção *Type*, você escolhe *Numbers*.



Aí em *Settings*, desmarca a opção *Use Grouping* que isso vai tirar o ponto.



Depois clica em *Apply* e em *OK*.

Agora se você rodar de novo o select, já vai aparecer sem o ponto.

PASSO 1 - CRIAR A STAGING AREA

Agora você vai criar a staging area.

Aqui já vai 2 dicas:

1. sempre utilize o prefixo *ST_* no nome da tabela da stage. Isso serve para identificar no nosso processo de carga que estamos falando de uma tabela de stage;

2. nós nem sempre vamos saber o que vai vir do legado. Se você souber o tipo de dado do legado, é bom já criar na stage com o mesmo tipo. Aqui a gente sabe porque criamos o legado, mas imagina se você pergunta para o DBA e ele diz que é um campo de 100 caracteres, você prepara sua stage com 100 também e quando chegam os dados, não é. Isso vai estragar a carga.

Podem também vir outras coisas, pode vir código inteiro e você ter que converter. Então, uma boa prática para você não perder tempo com esse tipo de coisa é colocar em `varchar 255`. Então sempre reserva um espaço maior e na dúvida usa `varchar` para que você consiga manipular isso depois.

Um bom exemplo é o código do cliente, que geralmente é um CPF, isso é clássico. Daí a gente pensa: “bom, CPF é numérico”. Não se for armazenado com pontinho e com hífen, aí ele vira um `varchar`. Você não sabe como isso vai vir.

Se você tiver o código padrão das tabelas da staging area, você salva o script nas suas coisas e na hora de criar, simplesmente roda ele no banco, tirando o que você não precisa. É assim que eu faço.

Então para criar a stage de cliente, vou rodar esse script aqui:

```
create table if not exists st_cliente (
    cd_cliente integer,
    nome_cliente varchar(255),
    email_cliente varchar(255)
);
```

Depois é só atualizar o `schema public` de novo que a stage vai aparecer ali.



Para criar as outras tabelas, o processo vai ser o mesmo. O código para criar a stage de produto é:

```
create table st_produtos (
    cd_produto varchar(255) ,
    nome_produto varchar(255) ,
```

```

cat_produto varchar(255) ,
preco_produto numeric (12,4) ,
status_produto varchar(255) ,
dt_carga timestamp not null

);

```

Depois, você vai criar a stage de venda. Aqui, a gente poderia fazer uma stage de pedido e outra de itens do pedido, mas como é uma transformação simples que precisa ser feito para já colocar as duas tabelas juntas na stage, você já pode fazer aqui mesmo.

O código da stage de vendas é este:

```

create table st_venda (
    dt_venda date,
    cd_venda integer,
    cd_produto varchar(255),
    qtd_vendida integer,
    vl_vendido numeric(10,4),
    cd_cliente varchar(255),
    status_venda varchar(60)
);

```

Aqui, você já cria a stage com os campos da forma que vamos precisar. Cada um deles vai ser equivalente a algum campo da origem, que é de onde você vai buscar os dados:

- **dt_venda** = *datapedido* em *pedidos*;
- **cd_venda** = *numeropedido* em *pedidos*;

- **cd_produto** = *codigoproduto* em *itenspedido*;
- **qtd_vendida** = *quantidadedepedido* em *itenspedido*;
- **vl_vendido** = *precounitario* em *itenspedido*;
- **cd_cliente** = *numerocliente* em *pedidos*;
- **status_venda** = *statuspedido* em *pedidos*.

Essa relação, em projetos, você vai fazer em um documento, para deixar claro de onde vem cada dado da stage. E aqui você pode ver que nem foi preciso todos os campos que estão na origem, porque nem todos eles vão ser relevantes para a tomada de decisão.

PASSO 2 – CARREGAR A STAGING AREA

O objetivo desse passo é carregar os dados do legado para a staging area. Ou seja, a gente tem um dado no legado e vamos jogar ele para a stage.

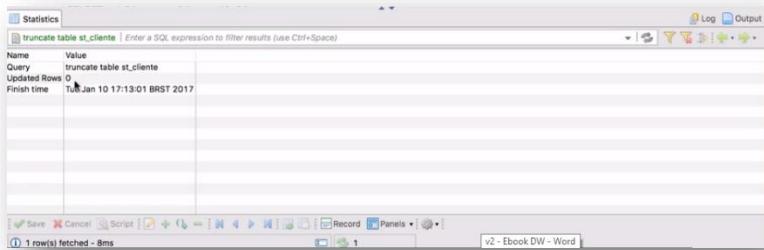
A primeira coisa que você tem que fazer é truncar a tabela da stage. Antes de inserir na stage, a tabela tem que estar vazia, então a gente sempre limpa a tabela antes de carregar.

Não sei se você viu, mas não colocamos primary key na stage, não coloquei integridade referencial para impedir a duplicidade dos dados. Então você precisa garantir que não vá ter problemas. Logo, você precisa truncar os dados da stage.

Este é o comando:

```
truncate table st_cliente;
```

Como você não tem dados na tabela ainda, ele vai mostrar que foram zero linhas atualizadas.



Agora sim você vai inserir na stage. O exemplo que eu trouxe aqui para você é um insert da stage na unha, só para mostrar como que faz.

Para inserir os dados na stage, você vai fazer um select que vai buscar os dados da origem. Esse select você vai colocar dentro de um insert, assim, ele vai inserir o resultado dessa consulta na stage.

O select que você vai usar é esse aqui:

```
select cdcli, nomecli, emailcli
from cadastro_cliente;
```

Você pode rodar ele para ver quais os dados que vão ser inseridos na stage. O resultado vai ser os dados que você inseriu na sua origem de cliente.

(print)

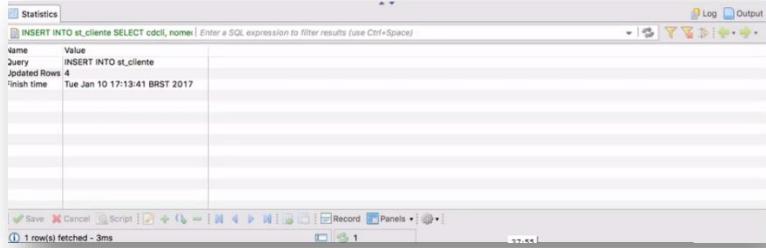
Agora é só você colocar esse select dentro de um insert para inserir os dados na stage. O código vai ficar assim:

```
insert into st_cliente
```

```
select cdcli, nomecli, emailcli  
from cadastro_cliente;
```

A ordem dos campos aqui é importante, porque se você colocar o email do cliente antes do nome no select, ele vai inserir na posição errada na stage.

Agora é só rodar isso e você vai ver que atualizou 4 registros.



The screenshot shows a database interface with a statistics panel on the left and a query editor on the right. The query editor contains the SQL command: `INSERT INTO st_cliente SELECT cdcli, nome`. The statistics panel displays the following information:

Name	Value
Query	INSERT INTO st_cliente
Updated Rows	4
Finish time	Tue Jan 10 17:13:41 BRST 2017

At the bottom of the interface, it says "1 row(s) fetched - 3ms".

Para validar se inseriu tudo corretamente, você vai dar um select na stage.

O código para isso é:

```
select* from st_cliente;
```

E agora você pode ver que os dados dos clientes estão na stage.

	cd_cliente	nome_cliente	email_cliente
1	1001	piton	suporte@raizer.com
2	2001	doriva	dorive@oriva.com
3	3001	rigo	rigo@ligenis.com
4	4001	rob	rob@robhud.com

Para inserir na stage de produtos, você vai usar um select dentro de um insert também.

O código vai ser:

```
insert into st_produtos
select
    cdproduto,
    nomeproduto,
    categoriaproduto,
    precoproduto,
    statusprod,
    current_date
from
    leg_produtos;
```

Para validar se tudo funcionou, você usa este código:

```
select* from st_produtos;
```

Para inserir na stage de venda, você vai usar este código:

```
insert into st_venda
select
    pedidos.datapedido,
    pedidos.numeropedido,
    itenspedido.codigoproduto,
    itenspedido.quantidadepedido,
    itenspedido.precounitario,
    pedidos.numerocliente,
    pedidos.statuspedido
from
    pedidos, itenspedido;
```

Eu coloquei todos os campos que você precisa para a stage venda ordenados como eles devem entrar.

Aqui você já pode aplicar algum filtro se precisar, por exemplo, se o cliente quiser só os dados a partir de 2017, você já coloca essa condição na consulta, ou se ele quer ver apenas os pedidos finalizados, essas coisas já são feitas aqui. Não tem necessidade de você inserir tudo na stage para depois deletar porque não precisa de algumas coisas.

PASSO 3 – CRIAR AS DIMENSÕES

A gente vai fazer todo o processo na mão de novo, direto no código, para quando você chegar em um projeto, independente da ferramenta, você saber o que tem que ser feito. E se a ferramenta der algum problema, você vai saber como resolver.

Vamos considerar que a gente não tem a dimensão ainda, como a gente fez na stage.

O que você vai fazer aqui é:

1. criar as sequences das surrogate keys;
2. criar as dimensões.

1. Criar as sequences das surrogate keys

Eu não precisaria explicar tudo isso aqui para você, porque essa parte já não é mais Data Warehouse, é conhecimento de banco de dados. Mas eu me comprometi com você de explicar tudo passo a passo e detalhado, e para garantir que você não vai ter nenhuma dúvida e vai entender todo o processo, vai entender porque as coisas acontecem, eu preciso te mostrar todos esses detalhes.

O código para criar a sequence para a surrogate key de cliente é este aqui:

```
create sequence dim_cliente_id_seq
increment by 1
minvalue 1
maxvalue 999999
start 200;
```

Se você rodar só a primeira linha do código, ele já vai criar a sequence para você com as configurações padrões do banco de dados que você está utilizando, mas eu vou mostrar algumas configurações que a gente pode fazer na criação da sequence:

- **increment by** - aqui você vai definir de quanto em quanto quer incrementar. Por exemplo, se eu colocar 10, as surrogate keys vão ser 10, 20, 30, 40, etc.;
- **minvalue** - esse é o valor mínimo que ele vai aceitar para a surrogate key;
- **maxvalue** - esse é o valor máximo que ele vai aceitar. Várias pessoas já me perguntaram: “e se um dia estourar esse espaço?”, você tem que dar muita carga nessa dimensão e ter muitos registros para isso acontecer, mas se caso algum dia precisar, você altera o tamanho da sequence;
- **start** - aqui você vai definir em que número ela vai começar a contar.

Depois é só você rodar esse comando.

Para você ver o número que ele está inserindo na tabela, você usa este comando aqui:

```
select nextval('dim_cliente_id_seq');
```

E ele vai mostrar 200, que foi o número em que a gente definiu para iniciar a sequence.

The screenshot shows a SQL developer interface with a results window titled "Result". Inside the results window, there is a single row of data from the query:

	nextval
1	200

Para criar a sequence da dimensão de produto, o processo vai ser o mesmo:

```
create sequence dim_produto_id_seq
increment by 1
minvalue 1
maxvalue 999999
start 200;
```

2. Criar as dimensões

Agora você vai criar uma nova tabela, que vai ser a sua dimensão de cliente.

O comando é este aqui:

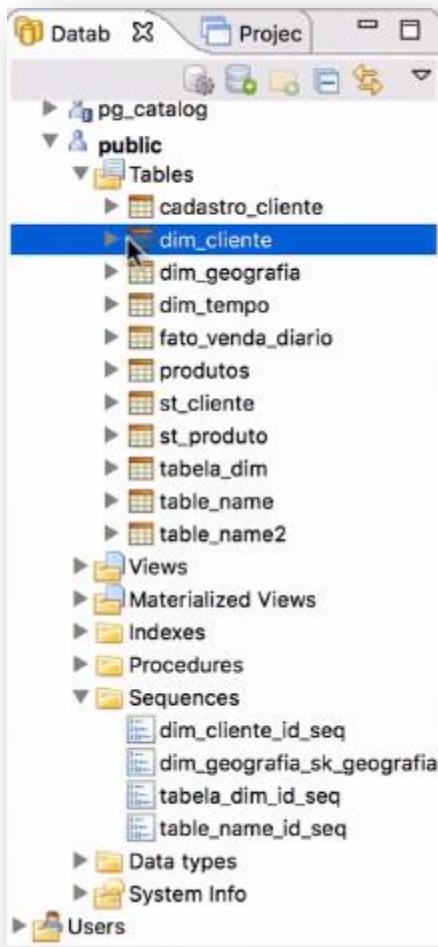
```
create table dim_cliente (
    sk_cliente integer primary key not null
    default nextval('dim_cliente_id_seq'),
    nk_cliente varchar (60) not null,
    nm_cliente varchar (100) not null,
    nm_email    varchar (100) not null
);
```

Aqui você está criando uma nova tabela chamada *dim_cliente*. Na segunda linha, você está definindo a surrogate key, e dizendo para ela de onde ela deve pegar o valor, que vai ser da *dim_cliente_id_seq*, a sequence que você criou.

Os atributos vão ser nome e email, que é o que colocamos na nossa origem.

Agora é só você rodar o comando para criar a sua tabela.

Para validar se funcionou, você atualiza o *Schema public* que a dimensão vai aparecer em *Tables*.



Para a dimensão se produto, você vai fazer a mesma coisa, o código vai ser este:

```
create table dim_produto (
    sk_produto integer primary key not null
        default nextval('dim_produto_id_seq'),
    nk_produto varchar(100),
    nm_produto varchar(100),
    cat_produto varchar(100),
    vl_produto numeric (12,4),
    status_produto varchar(50),
    dt_carga timestamp not null
);
```

Aqui você vai precisar também criar a dimensão de tempo, que vai ser a mesma que eu mostrei anteriormente, então você pode utilizar a mesma dimensão que você já tem criada.

PASSO 4 – CARREGAR AS DIMENSÕES

Você já viu como a gente carrega a stage, agora eu vou mostrar a mesma coisa para as dimensões.

O objetivo neste passo é pegar os dados que estão na stage e colocar nas dimensões.

Importante: você precisa carregar todas as dimensões antes de carregar a fato. Você não carrega uma dimensão, depois a fato, depois outra dimensão e depois a fato de novo, você carrega todas as dimensões e só depois a fato. Isso é o que a gente mais utiliza no mercado para não dar problema na carga. Porque se por exemplo, você tem quatro dimensões para carregar, três carregaram e uma deu problema, o que acontece é que você vai entregar a fato com uma

informação esquisita. Assim, se der algum problema durante esse processo, a gente aborta a carga.

Neste passo é onde você cria alguma transformação se precisar, cria os controles da dimensão como a surrogate key e também é onde decide se vai ser um update ou um insert.

Por exemplo, se você tem uma dimensão slowly changing dimension tipo 1, a dimensão vai atualizar sempre que você rodar a carga. Quase todas as dimensões são SCD tipo 1, porque ela precisa atualizar, uma hora você vai ter mais fornecedores, outra hora vai ter menos. Então às vezes você vai precisar só inserir um campo novo e às vezes vai atualizar algum que já existe.

Neste passo, o truncate só vai acontecer se houver uma carga full, que é uma carga completa. Quando alguma coisa está errada, os dados não estão batendo, você deleta tudo e reprocessa tudo de novo, aí vai precisar limpar todas as dimensões.

Aqui você vai:

1. passar os dados da staging area para as dimensões;
2. validar as cargas;
3. fazer as transformações.

1. Passar os dados da staging area para as dimensões

O processo de carregar as dimensões é muito similar ao de carregar a staging area. Você vai usar um select para buscar os dados da stage e inserir eles na dimensão.

O select que você vai usar para a dimensão de cliente é este aqui:

```
select
    nextval('dim_cliente_id_seq') ,
    cd_cliente,
    nome_cliente,
    email_cliente
from
    st_cliente;
```

Você pode rodar ele para ver os dados que vão ser inseridos na dimensão. O resultado vai ser os dados que você inseriu na stage de cliente.

	nextval	cd_cliente	nome_cliente	email_cliente
1	204	1001	piton	suporte@raizzer.com
2	205	2001	doriva	doriva@doriva.com
3	206	3001	rigo	rigo@rigonis.com
4	207	4001	rob	rob@rohud.com

Agora é só você colocar esse select dentro de um insert para inserir os dados na dimensão. O código vai ficar assim:

```
insert into dim_cliente
select
    nextval('dim_cliente_id_seq') ,
    cd_cliente,
    nome_cliente,
    email_cliente
```

```
from  
      st_cliente;
```

Agora é só rodar e a sua dimensão está carregada.

Para a dimensão de produto, o processo vai ser o mesmo. O código vai ser este:

```
insert into dim_produto  
select  
      nextval('dim_produto_id_seq'),  
      cd_produto,  
      nome_produto,  
      cat_produto,  
      preco_produto,  
      status_produto,  
      current_date  
from  
      st_produtos;
```

Já a dimensão de tempo é um pouco diferente.

Você vai dar um select sobre o banco de dados que você está operando. Eu fiz o código para o PostgreSQL que é o que a gente está utilizando, mas você pode converter para outros bancos.

A gente utiliza aqui algumas funções de extração das datas do banco de dados, para extrair o ano, o mês, a semana, e as outras informações que a gente precisa, mas são funções bem simples. Também criamos alguns cases para marcar as flags de final de semana

e feriado. E no final, definimos o período tempo que vai ser inserido na dimensão. Eu gosto de inserir de 7 a 10 para frente, se não logo você já vai precisar mexer na dimensão de tempo de novo.

O código completo é este aqui:

```
insert into dim_tempo
select
    to_number(to_char(datum, 'yyyymmdd'),
    '99999999') as sk_tempo,
    datum as data,
    to_char(datum, 'dd/mm/yyyy') as
    data_completa_formatada,
    extract (year from datum) as nr_ano,
    'T' || to_char(datum, 'q') as
    nm_trimestre,
    to_char(datum, '"T"q/yyyy') as
    nr_ano_trimenstre,
    extract(month from datum) as nr_mes,
    to_char(datum, 'tmMonth') as nm_mes,
    to_char(datum, 'yyyy/mm') as
    nr_ano_nr_mes,
    extract(week from datum) as
    nr_semana,
    to_char(datum, 'iyyy/iw') as
    nr_ano_nr_semana,
    extract(day from datum) as nr_dia,
    extract(doy from datum) as
    nr_dia_ano,
```

```

    to_char(datum, 'tmDay') as
        nm_dia_semana,
    case when extract(isodow from datum)
        in (6, 7) then 'Sim' else 'Não'
    end as flag_final_semana,
    case when to_char(datum, 'mdd') in
        ('0101', '1225') then 'Sim' else
        'Não'
    end as flag_feriado,
    case when to_char(datum, 'mdd') =
        '0101' then 'Ano Novo' when
        to_char(datum, 'mdd') = '1225' then
        'Natal' else 'Não é Fériado'
    end as nm_feriado,
    current_date as data_carga
from (
    select '2010-01-01'::date +
        sequence.day as datum
    from generate_series(0, 5479) as
        sequence(day)
    group by sequence.day
) dq
order by 1 desc;

```

2. Validar a carga

Existem várias formas para você validar a carga, vai depender do nível de garantia que você precisa.

Você pode só comparar a quantidade de registros da stage e da dimensão, ou se por exemplo, o campo de nome não pode estar errado, você pode validar um por um.

Para você ver os registros da dimensão, é só rodar este comando aqui:

```
select * from dim_cliente;
```

E ele vai mostrar o resultado.

	sk_cliente	nk_cliente	nm_cliente	nm_email
1	220	1001	piton	suporte@raizzer.com
2	221	2001	doriva	doriva@doriva.com
3	222	3001	rigo	rigo@rigonis.com
4	223	4001	rob	rob@robxhud.com

Para você comparar a quantidade de registros entre a stage e a dimensão, você faz assim:

- roda este comando para ver a quantidade de registros da dimensão:
 - **select count(1) from dim_cliente;**
- roda este comando para ver a quantidade de registros da stage:

- o `select count(1) from st_cliente;`

E os dois devem dar o mesmo resultado.

Para validar as outras dimensões, você vai utilizar os mesmos códigos, trocando onde diz `dt_cliente` pelo nome da dimensão que você quer validar.

3. Fazer as transformações

Deixei esta etapa para o final porque para entender ela, você primeiro precisava ver como funciona a inserção dos dados na dimensão. Essa parte da transformação vai depender do que você precisa transformar.

Por exemplo: você precisa deixar o nome de todos os clientes em maiúsculo.

Este seria o comando:

```
insert into dim_cliente
select
    nextval('dim_cliente_id_seq'),
    cd_cliente,
    upper(nome_cliente),
    email_cliente
from
    st_cliente;
```

Eu só adicionei um `upper` na parte do nome do cliente, o resto do código é exatamente igual ao que usamos para inserir na dimensão.

E ficaria assim:

	sk_cliente	cd_cliente	upper	email_cliente
1	228	1001	PITON	suporte@raizzer.com
2	229	2001	DORIVA	doriva@doriva.com
3	230	3001	RIGO	rigo@rigonis.com
4	231	4001	ROB	rob@robhud.com

Aqui você pode usar qualquer função de banco de dados, para deixar maiúsculo, minúsculo, remover espaços em branco ou colocar a primeira letra maiúscula, vai depender do que você quer fazer.

E agora a sua missão é rodar tudo isso, criar a sua dimensão, carregar ela e deixar tudo funcionando.

PASSO 5 – CRIAR A FATO

A fato é mais uma tabela, e a criação dela vai ser igual às que a gente já vem fazendo.

O código é este aqui:

```
create table fato_venda (
    sk_tempo integer not null,
    sk_cliente integer not null,
    sk_produto integer not null,
    qtd_venda integer not null,
    vlr_venda_uni numeric(12, 4) not null,
```

```

vl_venda_total numeric(12,2) not null,
dt_carga timestamp not null,
constraint fato_venda_dim_cliente_fk
foreign key (sk_cliente) references
dim_cliente(sk_cliente),
constraint fato_venda_dim_produto_fk
foreign key (sk_produto) references
dim_produto(sk_produto),
constraint fato_venda_dim_tempo_fk
foreign key (sk_tempo) references
dim_tempo(sk_tempo)
);

```

Nela, a gente tem 3 métricas:

- **qtd_venda** - a quantidade vendida;
- **vl_venda_uni** - o valor unitário da venda;
- **vl_venda_total** - o valor total da venda, que é uma métrica calculada. Nessa métrica, você tem duas opções, você pode calcular na hora de jogar os dados para a stage e você pode calcular na hora de jogar para a dimensão, que é a forma mais utilizada, porque assim você libera o legado mais rápido, fica menos tempo trabalhando nele.

E 3 surrogate keys, para você conectar nas dimensões:

- **sk_tempo** - a dimensão de data você já criou no capítulo que expliquei sobre essa dimensão;

- **sk_cliente** - a dimensão de cliente você criou no passo 3, de criar as dimensões;
- **sk_produto** - a dimensão de produto que você também criou no passo 3.

PASSO 6 – CARREGAR A FATO

Antes de mostrar o select completo, quero te explicar algumas coisas:

1. de onde vem os dados;
2. como você identifica a surrogate key;
3. o que é o left join;
4. como converter campos de data;
5. como carregar a fato.

1. De onde vem os dados

Antes de conseguir inserir na fato, você precisa identificar de onde vem os dados para cada um dos campos:

- **sk_tempo** - a sk_tempo vem da stage de vendas, porque é lá que está a data que a venda ocorreu.
- **sk_cliente** - para identificar a surrogate key do cliente, você vai precisar ver o código dele na stage de vendas e depois encontrar a surrogate key do cliente com aquele código na dimensão de cliente.
- **sk_produto** - aqui vai ser igual no campo anterior, você vai olhar na stage de vendas o código do produto e depois vai

pegar na dimensão de produto a surrogate key do produto com aquele código.

- **qtd_venda** - vem da stage de vendas
- **vl_venda_uni** - vem da stage de vendas
- **vl_venda_total** - essa métrica é calculada, então você vai pegar a quantidade de vendas e o valor da venda da stage de vendas e multiplicar os dois.
- **dt_carga** - essa você vai gerar na hora, porque é a data que está sendo feita a carga, o próprio banco de dados vai te dar esse valor.

2. como você identifica a surrogate key

Para carregar a `sk_cliente`, você vai precisar identificar qual é o cliente daquela venda na dimensão de cliente.

Faça uma consulta para ver os registo da `st_venda`:

```
select * from st_venda;
```

	dt_venda	cd_venda	cd_produto	qtd_vendida	vl_vendido	cd_cliente	status_venda
1	2017-01-06	10100	s18_1749	30	171,7	353	aguardando pagamento
2	2017-01-09	10101	s18_1749	30	171,7	128	pago
3	2017-01-10	10102	s18_1749	30	171,7	181	pago
4	2017-01-29	10103	s18_1749	30	171,7	121	pedido enviado
5	2017-01-31	10104	s18_1749	30	171,7	141	pedido enviado
6	2017-02-11	10105	s18_1749	30	171,7	145	entregue
7	2017-02-12	10106	s18_1749	30	171,7	278	entreque
8	2017-02-24	10107	s18_1749	30	171,7	311	entreque
9	2017-03-03	10108	s18_1749	30	171,7	385	entreque
10	2017-01-06	10109	s18_2248	60	67,8	353	aguardando pagamento
11	2017-01-09	10101	s18_2248	50	67,8	128	pago
12	2017-01-10	10102	s18_2248	50	67,8	181	pago

Uma das colunas é a `cd_cliente`, que tem o código do cliente.

E agora faça uma consulta para ver os registros da *dim_cliente*:

```
select * from dim_cliente;
```

E nela você vai ter a *nk_cliente*, que também tem o código do cliente.

	sk_cliente	nk_cliente	nm_cliente	nm_email
1	202	2001	DORIVA	DORIVA@DORIVA.COM
2	203	3001	RIGO	RIGO@RIGONIS.COM
3	204	4001	ROB	ROB@ROBHUD.COM
4	201	1001	PITON	SUPORTE@RAIZZER.COM

E para encontrar a surrogate key na dimensão de cliente, a gente vai fazer uma comparação entre esses dois campos.

Então você vai adicionar uma condição no select para pegar os registros só quando esses dois campos forem iguais.

A consulta vai ficar assim:

```
select
    st_vendas.cd_venda,
    dim_cliente.sk_cliente
from
    st_vendas, dim_cliente
where
    st_vendas.cd_cliente =
    dim_cliente.nk_cliente;
```

Se você for olhar nos códigos dos clientes, vai ver que dos clientes que aparecem na stage de venda, só um está na dimensão de cliente, que é o cliente da natural key 128. Por isso, essa consulta só vai retornar as vendas de um cliente.

	st_venda(+)	Enter a SQL expression to filter results (use Ctrl+Space)				
	dt_venda	sk_cliente	qtd_vendida	vl_vendido	vl_venda_total	
1	2017-01-09	201	30	171,7	5151	
2	2017-01-09	201	50	67,8	3390	
3	2017-01-09	201	22	86,51	1903,22	
4	2017-01-09	201	49	34,47	1689,03	
5	2017-01-09	201	25	151,28	3782	
6	2017-01-09	201	26	145,13	3773,38	
7	2017-01-09	201	45	31,2	1404	
8	2017-01-09	201	46	53,76	2472,96	
9	2017-01-09	201	39	123,29	4808,31	
10	2017-01-09	201	41	50,14	2055,74	
11	2017-01-09	201	26	207,87	5404,62	
12	2017-01-09	201	42	128,53	5398,26	
13	2017-01-09	201	27	105,74	3004,60	

E ele vai retornar os dados que a gente pediu no select. E é assim que a gente descobre qual é a surrogate key daquele cliente. Pode ver que na consulta ele trouxe o campo da surrogate key, que nesse caso é 201.

3. o que é o left join

Agora quero mostrar para você o que a gente faz quando tem mais coisas na stage que tem na dimensão.

Na consulta anterior que você fez, vai perceber que nem todas as vendas cadastradas apareceram, isso acontece porque a gente tem lançamentos na stage de vendas com código de clientes que não existem na dimensão de cliente, mas elas precisam ser inseridas na

fato mesmo assim, e tem algumas técnicas que a gente usa para classificar esses carinhos como não identificados.

Para isso, você vai alterar a linha do from e do where na consulta.

Da forma que nós fizemos, estávamos usando um inner join, que nos trazia apenas os registros quando existiam nas duas tabelas. O left join, vai nos trazer todos os registros da primeira tabela, e se ele não se relacionar com nada na segunda tabela, vai aparecer como null no campo vindo da segunda tabela. Assim, a gente garante que vamos pegar todos os registros da primeira tabela.

Aqui a consulta vai ficar assim:

Select

```
st_vendas.cd_venda,  
dim_cliente.sk_cliente  
from  
    st_vendas left join dim_cliente on  
        st_vendas.cd_cliente =  
            dim_cliente.nk_cliente;
```

Agora, todos os registros da st_venda vão aparecer no resultado, mesmo que o cliente daquele registro não exista.

Agora se você rodar esse select de novo, vai ter este resultado:

	123 cd_venda	123 sk_cliente
1	10,100	[NULL]
2	10,100	[NULL]
3	10,100	[NULL]
4	10,100	[NULL]
5	10,101	200
6	10,101	200
7	10,101	200
8	10,101	200
9	10,102	[NULL]
10	10,102	[NULL]
11	10,103	[NULL]

Agora ele colocou null nas surrogate keys dos clientes que ele não encontrou na dimensão de clientes.

Agora a gente vai dizer para ele colocar -1 em todos os campos que estiverem como null.

E na dimensão de cliente, a gente vai adicionar um campo, para informar que o -1 quer dizer que o cliente não é identificado.

Para fazer isso, você primeiro insere um novo campo na dimensão de cliente:

```
insert into dim_cliente
values (-1, '*não identificado*', '*não
identificado*', '*não identificado*',
current_date);
```

Agora no insert da fato, você vai fazer uma alteração na linha do dim_cliente.sk_cliente. Aqui você vai usar uma função do banco de dados, que faz uma substituição de valores null, ele identifica todos os campos null daquela coluna e insere algum valor que você quiser, que nesse caso vai ser -1.

No PostgreSQL, você faz isso assim:

```
select
    st_vendas.cd_venda,
    coalesce (dim_cliente.sk_cliente, -1)
from
    st_vendas left join dim_cliente on
    st_vendas.cd_cliente =
    dim_cliente.nk_cliente;
```

E de você rodar o seu select agora, vai ver que ele vai trocar todos os nulls por -1:

	123 cd_venda	123 coalesce
1	10,100	-1
2	10,100	-1
3	10,100	-1
4	10,100	-1
5	10,101	200
6	10,101	200
7	10,101	200
8	10,101	200
9	10,102	-1
10	10,102	-1
11	10,103	-1

Agora você também cria o campo de não identificado na dimensão de produto, para quando aparecer produtos que não estão cadastrados também:

```
insert into dim_produto
values(-1, '*não identificado*', '*não
identificado*', '*não identificado*', -1, '*não
identificado*', current_date);
```

4. Como converter campos de data

A coluna *sk_tempo* da fato é do tipo *integer*, e a coluna *dt_venda* da stage de vendas é do tipo *date*, então se você tentar inserir da stage direto na fato, vai dar erro.

Agora você tem que trocar o formato do resultado que a gente está pegando para poder inserir na fato.

Para você pegar o resultado da dt_venda, você usa este código:

```
select  
    dt_venda  
from  
    st_vendas;
```

No caso do PostgreSQL e alguns outros bancos de dados, ele não consegue fazer a conversão de date direto para integer, então a gente precisa primeiro converter para char e depois para integer.

Para converter para char, você vai usar:

```
select  
    to_char(dt_venda , 'YYYYMMDD' )  
from  
    st_vendas;
```

O to_char vai converter aquele resultado para char, enquanto o 'YYYYMMDD' é para dizer o formato em que a gente quer que ele converta. Assim, a gente vai ter a data no mesmo formato que já temos mas sem aqueles tracinhos que ela tinha.

Se você rodar o select assim, ele vai retornar assim:

	ABC to_char ↕↑?
1	20170106
2	20170106
3	20170106
4	20170106
5	20170109
6	20170109
7	20170109
8	20170109
9	20170110
10	20170110
11	20170129

Para transformar esse char agora em integer, você vai usar:

```
select
    to_number(to_char(dt_venda , 'YYYYMMDD'
    ), '99999999')
from
    st_vendas;
```

O to_number vai fazer essa conversão e o '99999999' vai definir o formato que ele vai retornar o dado. Aqui você coloca a mesma quantidade de números que tem na data, é esse o formato que a gente precisa.

Se você rodar essa consulta, o resultado vai ser igual ao anterior, mas agora o tipo do campo vai estar diferente.

5. Como carregar a fato

O código para carregar a fato, como nos outros, você vai fazer um select para pegar os dados da stage e das dimensões e colocar isso em um insert para inserir na fato. O select vai ser:

```
insert into fato_venda (
    select
        to_number(to_char(st_vendas.dt_venda
            , 'yyyymmdd'), '99999999'),
        coalesce(dim_cliente.sk_cliente, -
            1),
        coalesce(dim_produto.sk_produto, -
            1),
        st_vendas.qtd_venda,
        st_vendas.vl_venda,
        st_vendas.qtd_venda *
        st_vendas.vl_venda,
        current_date
    from st_vendas
        left join dim_cliente on
            st_vendas.cd_cliente =
            dim_cliente.nk_cliente
        left join dim_produto on
            st_vendas.cd_produto =
            dim_produto.nk_produto
) ;
```

Depois de rodar e inserir os registros na fato, você pode validar fazendo uma consulta nela:

```
select * from fato_venda;
```

PASSO 7 – VALIDAR A CARGA

Para validar a carga, você vai fazer um select cruzando a fato e as dimensões, para ver se os dados que estão nela, são iguais aos dados do legado.

O comando é:

```
select  
    dim_tempo.nr_ano,  
    dim_tempo.nr_mes,  
    dim_tempo.nm_mes,  
    sum(fato_venda.qtd_venda) as qtd_venda,  
    sum(fato_venda.vl_venda_total) as  
    vl_venda_total,  
    dim_produto.nm_produto  
  
from  
    fato_venda  
    inner join dim_tempo on  
        fato_venda.sk_tempo = dim_tempo.sk_tempo  
    inner join dim_cliente on  
        fato_venda.sk_cliente =  
        dim_cliente.sk_cliente
```

```
inner join dim_produto on
fato_venda.sk_produto =
dim_produto.sk_produto

group by
    dim_produto.nm_produto, dim_tempo.nr_ano, d
    im_tempo.nm_mes, dim_tempo.nr_mes

order by
    dim_tempo.nr_mes;
```

Você pode ver que na parte da quantidade de vendas e valor da venda, a gente usa o sum, para somar os resultados, assim você ve o valor total, que deve ser igual ao valor total do legado, porque seria inviável comparar linha por linha.

E com isso, você carregou todo o seu Data Warehouse e aprendeu como funciona o processo do ETL.

Em projetos da vida real, você vai utilizar uma ferramenta para fazer isso, e não vai precisar criar todos os códigos na mão, mas é importante você ter esse primeiro contato direto no código para entender como que as coisas acontecem, porque você tem que saber o que precisa que a ferramenta faça.

MODELAGEM DIMENSIONAL AVANÇADA

Agora que você já aprendeu a criar um Data Warehouse e a carregar ele, quero mostrar alguns tipos de fatos e dimensões um pouco mais complexas e que servem para resolver alguns problemas que você vai encontrar quando estiver criando Data Warehouse mais complexos.

Deixei essa parte para o final porque para entender como essas fatos e dimensões funcionam, você já precisava ter aprendido a construir e carregar o Data Warehouse, porque são tipos mais complexos, essa já é uma parte bem mais avançada da modelagem dimensional.

DIMENSÃO NÃO QUALIFICADA

Existem dois tipos de dimensões não qualificadas:

- não identificado;
- não se aplica.

Não identificado

Essa acontece quando você não encontra na dimensão a informação que vem lá do legado. Você já viu essa situação quando estava fazendo o ETL. Quando você não encontrava um dado nas dimensões, mas não podia deixar ele fora da fato, classificava como -1.

É uma boa prática fazer isso para não ter divergência de valores na fato. Se no legado o fechamento do dia estiver em 3.000 unidades vendidas, na fato também tem que estar, e se você não cadastrar alguma venda porque o produto não foi encontrado, os números não vão bater.

A ideia do não identificado é que você possa qualificar esses dados para no futuro corrigir o problema.

Às vezes pode ter uma demora na atualização do legado, o operador pode demorar para carregar um produto novo ou algo assim, e quando o ETL rodar, não vai encontrar esse dado, mas isso vai ser atualizado depois.

Um exemplo:

	A	B	C	D	E	F	G	H	I	J
Dimensão Produto										
1	SK	NK	Nome	Preço unitário						
2	1	2323	Coca-Cola	3,55						
3	2	1111	Pizza	23,60						
4	3	4545	café	7,80						
5						
6						
7						
8						
9						
10						
Dimensão Tempo										
11	SK	NK	Ano	Mês						
12	10122016	10/12/2016	2016	Dezembro						
13	11122016	11/12/2016	2016	Dezembro						
14	12122016	12/12/2016	2016	Dezembro						
15						
16						
17						
18						
Dimensão Vendedor										
	SK	NK	Nome	Conta						
	23	101	Rafael Piton	Grupo Coca-cola						
	24	102	Rigoni	Grupo Walmart						
	25	103	Caroline	Rede Globo						
						
						
						

É uma fato venda com dimensões de produto, tempo e vendedor.

Na hora de carregar a primeira linha, ele encontra todas as surrogate keys nas dimensões, e faz um cruzamento completo.

	A	B	C	D	E	F	G	H	I	J	K
Dimensão Produto											
1	SK	NK	Nome	Preço unitário							
2	1	2323	Coca-Cola	3,55							
3	2	1111	Pizza	23,60							
4	3	4545	café	7,80							
5							
6							
7							
8							
9							
10							
Dimensão Tempo											
11	SK	NK	Ano	Mês							
12	10122016	10/12/2016	2016	Dezembro							
13	11122016	11/12/2016	2016	Dezembro							
14	12122016	12/12/2016	2016	Dezembro							
15							
16							
17							
18							
Dimensão Vendedor											
	SK	NK	Nome	Conta							
	23	101	Rafael Piton	Grupo Coca-cola							
	24	102	Rigoni	Grupo Walmart							
	25	103	Caroline	Rede Globo							
							
							
							

Na hora de inserir a segunda linha, o código do vendedor não foi encontrado na dimensão. A gente sabe o que foi vendido, mas não recebeu a informação de quem vendeu. Nesse caso, coloca um -1.

	A	B	C	D	E	F	G	H	I	J	K	
Dimensão Produto												
1	SK	NK	Nome	Preço unitário								
2	1	2323	Coca-Cola	3,55								
3	2	1111	Pizza	23,60								
4	3	4545	café	7,00								
5								
6								
7								
8								
9								
10	Dimensão Tempo											
11	SK	NK	Ano	Mês								
12	10122016	10/12/2016	2016	Dezembro								
13	11122016	11/12/2016	2016	Dezembro								
14	12122016	12/12/2016	2016	Dezembro								
15								
16								
17								
18								

	A	B	C	D	E	F	G	H	I	J	K	
Dimensão Vendedor												
1	SK	NK	Nome	Conta								
2	23	101	Rafael Piton	Grupo Coca-cola								
3	24	102	Rigoni	Grupo Wallmart								
4	25	103	Caroline	Rede Globo								
5								
6								
7								
8								
9								
10	Dimensão Fato											
11	FK	FK	PK	Métrica aditiva	Métrica Demanda							
12	DIM_DATA	DIM_PRODUTO	DIM_VENDEDOR	Quantidade da Venda	Valor da Venda							
13	10122016	1	23	200	710,00							
14	11122016	2	-1	300	7080,00							
15	12122016			200	1569,00							
16							
17							
18							

Aí lá na dimensão de cliente, você precisa ter uma linha com a surrogate key -1.

	A	B	C	D	E	F	G	H	I	J	K	
Dimensão Produto												
1	SK	NK	Nome	Preço unitário								
2	1	2323	Coca-Cola	3,55								
3	2	1111	Pizza	23,60								
4	3	4545	café	7,00								
5								
6								
7								
8								
9								
10	Dimensão Tempo											
11	SK	NK	Ano	Mês								
12	10122016	10/12/2016	2016	Dezembro								
13	11122016	11/12/2016	2016	Dezembro								
14	12122016	12/12/2016	2016	Dezembro								
15								
16								
17								
18								

	A	B	C	D	E	F	G	H	I	J	K	
Dimensão Vendedor												
1	SK	NK	Nome	Conta								
2	23	101	Rafael Piton	Grupo Coca-cola								
3	24	102	Rigoni	Grupo Wallmart								
4	25	103	Caroline	Rede Globo								
5								
6								
7								
8								
9								
10	Dimensão Fato											
11	FK	FK	PK	Métrica aditiva	Métrica Demanda							
12	DIM_DATA	DIM_PRODUTO	DIM_VENDEDOR	Quantidade da Venda	Valor da Venda							
13	10122016	1	23	200	710,00							
14	11122016	2	-1	300	7080,00							
15	12122016			200	1569,00							
16							
17							
18							

Assim você insere o registro na fato e sabe que tem algo de errado com aquela venda.

Não se aplica

Na hora de inserir a terceira linha da fato, era um caso em que o vendedor não precisava ser monitorado, então foi inserido como -2.

	A	B	C	D	E	F	G	H	I	J	K
1	Dimensão Produto				Fato Venda						
2	SK	NK	Nome	Preço unitário	FK	FK	FK	Métrica ativa	Métrica Derivada		
3	1	✓ 2323	Coca-Cola	3,55		1	23	200	710,00	Cruzamento completo	
4	2	✓ 1111	Pizza	23,60		2	-1	300	7080,00	** Não Identificado **	
5	3	✓ 4545	café	7,80		3	-2	200	1560,00	** Não se Aplica **	
6							
7											
8											
9											
10											
11	Dimensão Tempo				Dimensão Vendedor						
12	SK	NK	Ano	Mês	SK	NK	Nome	Conta			
13	10122016	✓ 10/12/2016	2016	Dezembro	23	✓ 101	Rafael Piton	Grupo Coca-cola			
14	11122016	✓ 11/12/2016	2016	Dezembro	24	✓ 102	Rigoni	Grupo Walmart			
15	12122016	✓ 12/12/2016	2016	Dezembro	25	✓ 103	Caroline	Rede Globo			
16			
17											
18											

A diferença entre não identificado e não se aplica, é que quando você vai fazer a inserção e não consegue encontrar o vendedor na dimensão, ele é -1, mas quando você não precisa encontrar, é -2, porque aquela informação não é monitorada nesse caso.

É importante você sempre evitar deixar campos nulos na fato, ela tem que ter a dimensionalidade completa. Quando não for possível completar, marca como não identificado ou como não se aplica.

Não é sempre que você vai pegar um projeto do zero, e você vai encontrar diversas situações. Por exemplo: imagina que antes estavam trabalhando sem o campo de vendedor, e agora querem monitorar vendedor também, mas o cliente não tem o registro dos vendedores do histórico, uma forma de resolver isso é colocar o -2 para os registros antigos.

Você vai encontrar todo o tipo de situação, mas o importante é não deixar a fato com dado nulo ou deixar de inserir algum registro na fato por causa de algum dado faltando.

ROLE-PLAYING DIMENSION

No Data Warehouse, as dimensões muitas vezes são utilizadas para múltiplos objetivos.

Por exemplo:

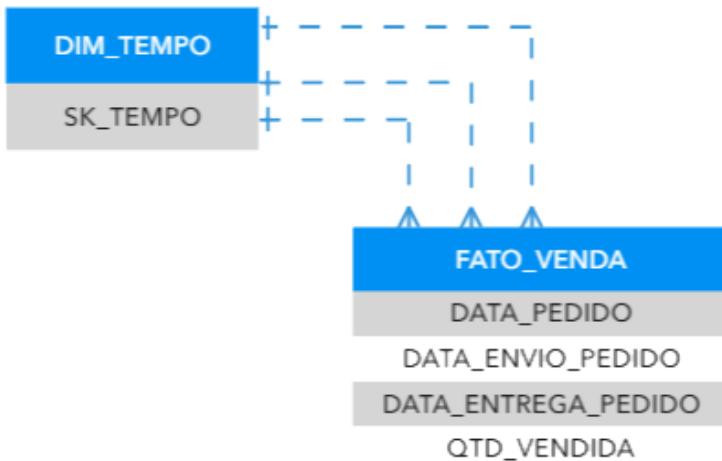
Imagina você está fazendo uma análise de vendas e nessa análise precisa mensurar a quantidade vendida e precisa ver essa quantidade pela data do pedido, data do envio do pedido e data do recebimento do pedido.

Quando acontece essa situação, nós precisamos ter as surrogate keys na fato para analisar cada data de forma separada.

Em muitos casos, quando as pessoas não entendem de modelagem dimensional, copiam a dimensão de tempo 3x.

Você até pode ter essa visão lógica do Data Warehouse, mas fisicamente, isso não precisa existir, porque você vai deixar seu Data Warehouse 3x maior e é uma péssima prática.

Nesse caso, você só precisa referenciar a mesma dimensão 3x na fato, assim:



Assim, você consegue analisar a quantidade vendida pela ótica das 3 datas.

CONFORMED DIMENSION

Essa é uma dimensão que tem o mesmo significado para todas as fatos com que se relaciona. Uma boa prática é deixar todas as dimensões conformadas. Quem bate muito nessa tecla é o próprio Ralph Kimball.

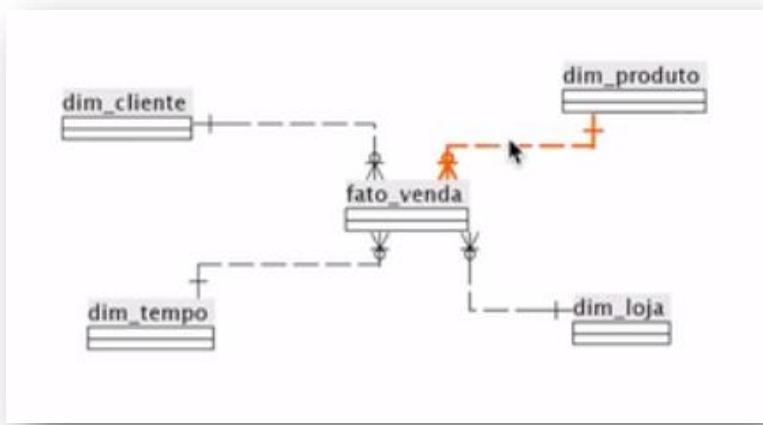
Por quê?

Porque assim todo mundo fala a mesma língua. Uma dimensão conformada nada mais é que as fatos e métricas classificadas e descritas da mesma forma em toda a empresa, ou seja, a verdade única.

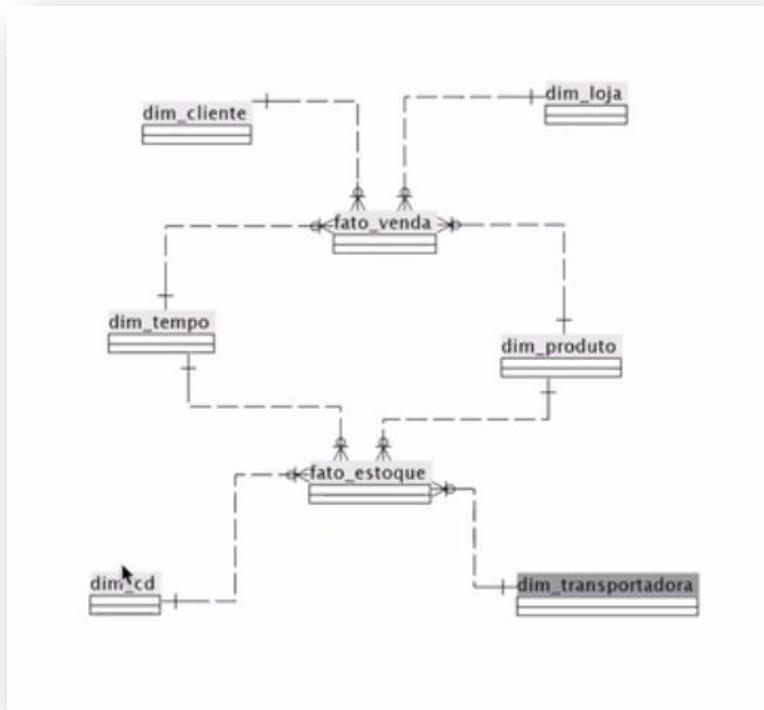
Quando falamos de um Data Warehouse corporativo, você tem uma dimensão relacionando com mais de uma fato, e dimensão conformada é quando essa dimensão tem todos os dados que as duas precisam.

Por exemplo:

Aqui eu tenho um desenho de um star schema com uma fato venda e quatro dimensões.



Depois, quando você começar a trabalhar em uma outra área de negócio, algumas das dimensões que ela vai precisar, já vão existir, e você não vai duplicar a dimensão, mas utilizar as mesmas.



Nesse caso, as dimensões de tempo e produto são conformadas, elas vão ter todas as informações que as duas fatos precisam na mesma dimensão, inclusive a hierarquia.

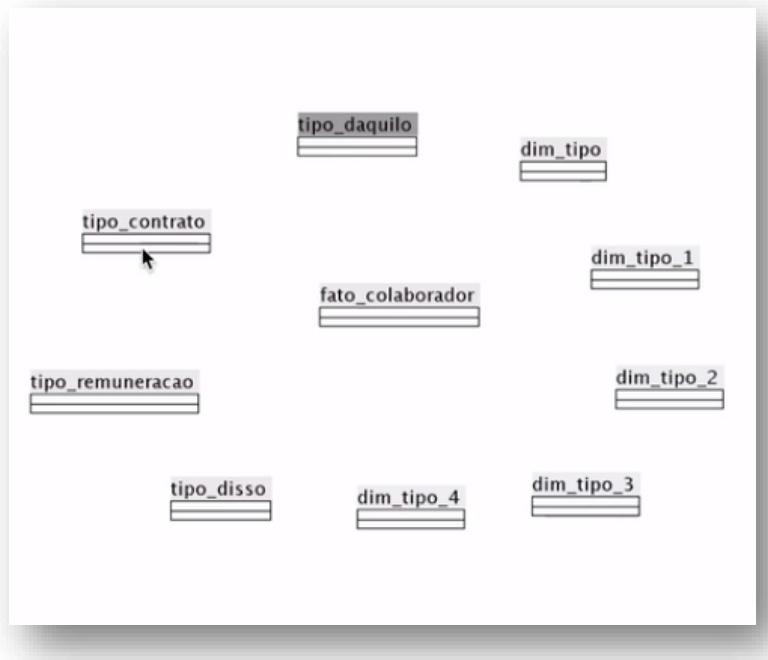
No projeto da Sodexo, tinha uma dimensão que cada área de negócio queria ver com uma hierarquia diferente. Nesse caso, eu coloquei as duas hierarquias na mesma dimensão, e assim ela seguiu sendo uma dimensão conformada, sem precisar criar uma nova dimensão de produto.

JUNK DIMENSION

A junk dimension normalmente é esquecida e não é ensinada em quase nenhum curso. Eu e alguns outros especialistas de Data Warehouse dizemos que o nome dessa dimensão ficou um pouco errado, porque quando você fala em dimensão de lixo, tem a impressão de que são coisas que não prestam mais.

Vou usar como exemplo um dos projetos que participei, onde a consultoria que tinha passado anteriormente pelo cliente, construiu um star Schema com uma fato colaborador e várias dimensões de tipos, como tipo de remuneração, tipo de contrato, tipo disso, tipo daquilo, e mais dezenas (sim, mais de 40) de outras dimensões de tipos.

Era algo mais ou menos assim:



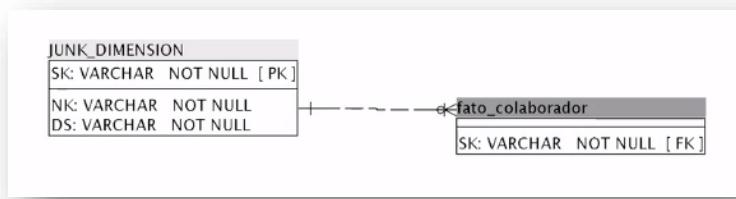
O problema disso é que essas dimensões de tipo, pensando fisicamente em um banco de dados, poderiam ser junk dimensions.

Não é a única forma de fazer, mas é bom para quando você tem uma característica de dimensões com a cardinalidade muito baixa ou com a densidade muito esparsa.

O que isso quer dizer? Que essas dimensões têm basicamente 3 colunas: a surrogate key, a natural key e uma descrição.

Se usassem a técnica de junk dimension, teriam somente uma dimensão com essas três colunas, e colocariam todas as dezenas de tipos nela, eliminando aquele monte de dimensão que tinha antes.

Ficaria assim:

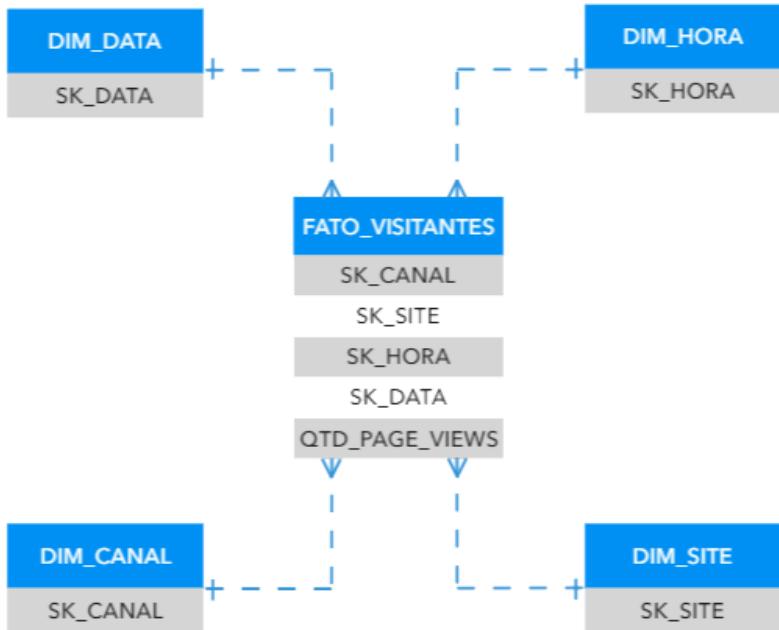


Mas é importante você entender que essa solução aqui é para quando não tem onde colocar as coisas.

FATO AGREGADA

Esta fato tem a função de acelerar o desempenho das consultas.

Tenho aqui um exemplo com uma fato visitantes e dimensões de data e hora. Não coloquei os atributos porque a ideia é que você entenda a técnica.



A fato vai monitorar as pageviews, ou seja, a quantidade de acessos do site.

Mas eu não quero ver os visitantes por minuto, quero eles consolidados.

Você vai pensar: mas isso as ferramentas de BI já fazem em tempo de execução.

Sim. E a maioria dos projetos deixa dessa forma, deixa a ferramenta consolidar em tempo real. Mas pensa em um caso que é um site que tem milhões de pageviews por segundo. Imagina pedir para a ferramenta de BI conectar no Data Warehouse e consolidar elas em tempo de execução, pensa no tempo que isso vai levar.

Então quando tem esse tipo de situação, você provavelmente vai ter problema de performance porque o dashboard vai demorar para carregar.

E como se resolve isso? Existe uma técnica para isso, que é a fato agregada.

A	B	C	D	E
DIM_DATA	DIM_HORA	DIM_CANAL	DIM_SITE	QTD_PAGE_VIEWS
1	10/08/2017	13:01	facebook	rafaelpiton.com/home
2	10/08/2017	13:02	facebook	rafaelpiton.com/home
3	10/08/2017	13:03	facebook	rafaelpiton.com/home
4	10/08/2017	13:04	facebook	rafaelpiton.com/home
5	10/08/2017	13:05	facebook	rafaelpiton.com/home
6	10/08/2017	13:06	facebook	rafaelpiton.com/home
7	10/08/2017	13:07	facebook	rafaelpiton.com/home
8	10/08/2017	13:08	facebook	rafaelpiton.com/home
9	10/08/2017	13:09	facebook	rafaelpiton.com/home
10	10/08/2017	13:10	facebook	rafaelpiton.com/home
11	10/08/2017	13:11	facebook	rafaelpiton.com/home
12	10/08/2017	13:12	facebook	rafaelpiton.com/home
13	10/08/2017	13:13	facebook	rafaelpiton.com/home
14	10/08/2017	13:14	facebook	rafaelpiton.com/home
15	10/08/2017	13:15	facebook	rafaelpiton.com/home
16	10/08/2017	13:16	facebook	rafaelpiton.com/home
17	10/08/2017	13:17	facebook	rafaelpiton.com/home
18	10/08/2017	13:18	facebook	rafaelpiton.com/home
19	10/08/2017	13:19	facebook	rafaelpiton.com/home
20	10/08/2017	13:20	facebook	rafaelpiton.com/home
21	10/08/2017	13:21	facebook	rafaelpiton.com/home

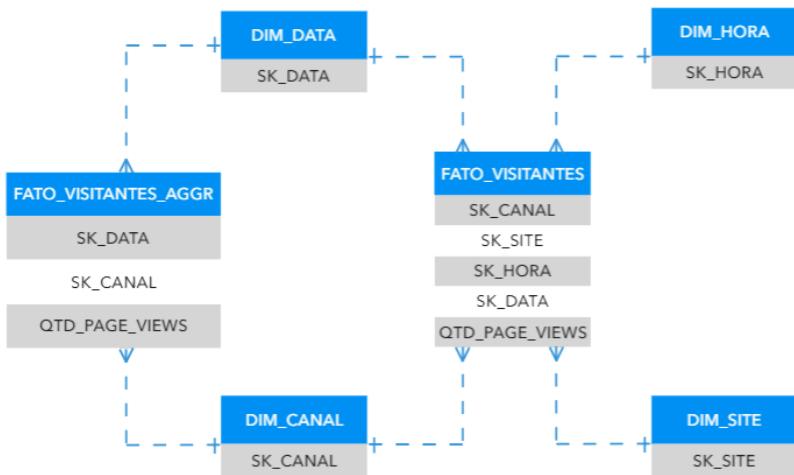
Aqui eu fiz um exemplo com os minutos da hora 13, do dia 10, do canal Facebook e da página home, mas uma fato completa teria todos os dias, todos os minutos de todas as horas, de todos os canais e de todas as páginas, ou seja, seria um monstro.

Mas o que o cliente precisa ver aqui é: o total de pageviews por dia e por canal.

É aí que entra a fato agregada, ela serve para juntar um bolo de dados quando eu não quero analisar no nível do grão, dessa forma:

A	B	C
1 SUM of QTD_PAGE_VIEWS	Rótulos de Coluna	
2 Rótulos de Linha	facebook	Total Geral
3 10/08/2017		3540
4 Total Geral		3540

E como que fica isso na modelagem então? Você cria uma fato agregada com os dados que precisa e mantém também a fato normal.



FATO CONSOLIDADA

Esta fato é bem parecida com a agregada, mas serve para combinar duas áreas de negócio.

Diferente da fato agregada, que serve para consolidar os dados. A fato consolidada serve para consolidar duas fatos.

Mas calma, você não vai fazer nenhum join com as fatos. O que acontece é que no processamento do ETL, na hora de carregar a fato, você vai carregar uma, carregar a outra, e misturar as duas. Evidente que o grão precisa ser o mesmo.

Um exemplo clássico:

Você tem uma fato venda realizada, e depois precisa juntar ela com uma fato venda orçada.

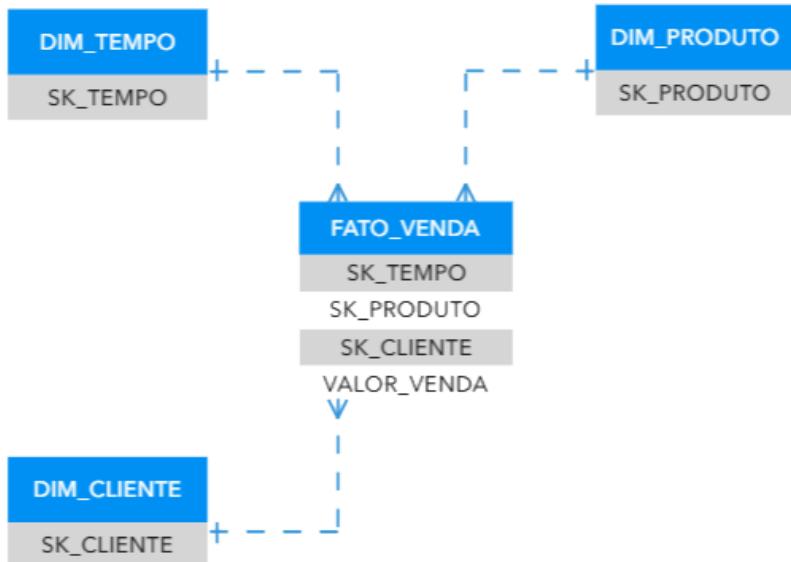
É claro que você pode fazer uma dimensão de cenário e já colocar lá se ela é orçada. Mas isso é em um mundo ideal onde você começa seu Data Warehouse do zero sem nenhum problema. E muitas vezes, mesmo que não tenha sido um desleixo, na hora de fazer o levantamento de requisitos, ninguém nunca falou que seria necessário trabalhar com orçamento de venda, então ao invés de criar uma

dimensão de cenário e reprocessar tudo, foi feito uma nova fato. São várias situações em que isso pode acontecer.

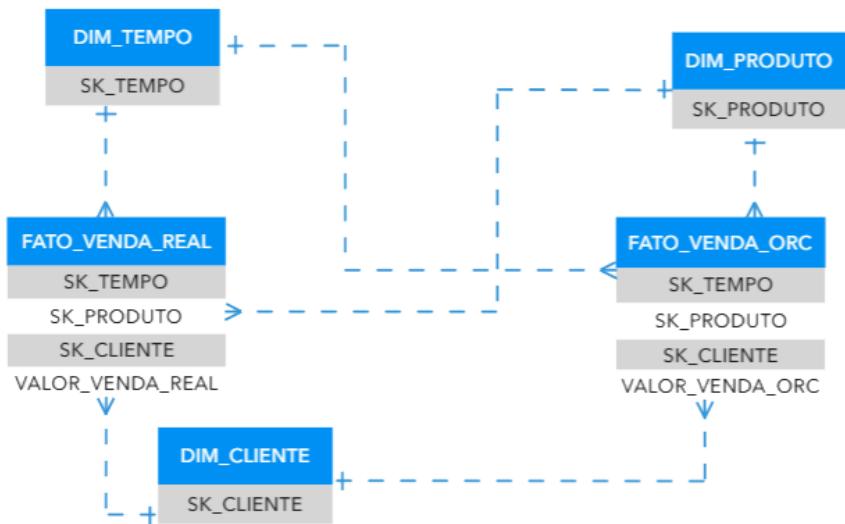
As fatos consolidadas adicionam uma complexidade extra no processamento do ETL. Você vai ter que reconfigurar um monte de coisa ou simplesmente fazer um novo, como a maioria das pessoas faz.

É sempre importante você entender onde isso aqui se encaixa e analisar pontualmente, vai ter que olhar o problema e ver como resolver.

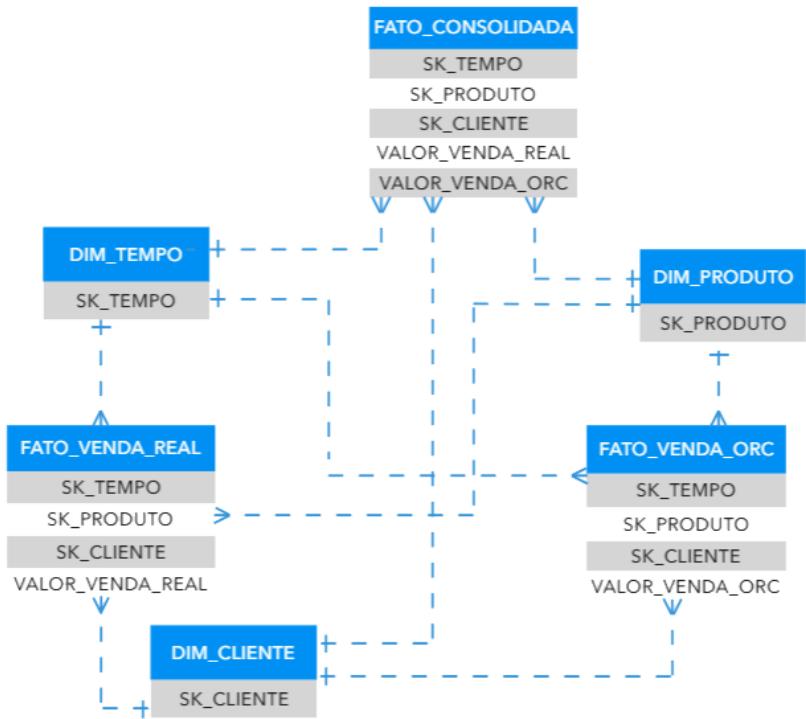
Essa era a modelagem que você tinha.



Então apareceu o problema do valor orçado que ninguém tinha falado e foi criada uma nova fato.



Depois disso, você cria uma fato consolidada e coloca nela o valor real e o orçado, que antes estavam em fatos diferentes.



E os dados dela você pode pegar tanto das fatos como direto da origem.

FATO SNAPSHOT PERIÓDICO

Aqui vou mostrar 3 opções de como uma determinada situação pode se comportar.

A opção #1 é uma fato transacional, é um exemplo clássico de uma fato estoque. Ela controla o produto, quantidade de entrada, quantidade de saída e o saldo do produto.

	A	B	C	D	E
1	OPÇÃO #1				
3 Fato_estoque_hora_transacional					
4	data	produto	qtd_entrada	qtd_saida	saldo
5	23/10/2017 06:00:00	Café Ristretto	2.000	500	1.500
6	23/10/2017 12:00:00	Café Ristretto	400	1.500	400
7	23/10/2017 16:00:00	Café Ristretto	700	300	800
8

No dia 23/10/2017, às 6h, eu tive a entrada de 2.000 cafés no estoque, e nesse mesmo horário, uma saída de 500, ficando com um saldo de 1.500.

Depois, ao meio dia, foi feito um novo pedido para o fornecedor, então entrou mais 400 cafés e depois saiu 1.500, ficando com o saldo de 400. E assim por diante.

O que é o snapshot periódico, então? Ele é baseado no tempo, seja data, dia, semana ou hora. Nesse exemplo a gente vai trabalhar no dia.

Na opção #2, temos uma snapshot periódica, que é a famosa foto. Eu tiro uma foto do momento atual e salvo ali.

	A	B	C	D	E	F	G	H	I	J	K
OPÇÃO #1						OPÇÃO #2					
Fato estoque hora transacional											
(opcional)											
data	produto	qtd_entrada	qtd_saida	saldo							
23/10/2017 06:00:00	Café Ristretto	2.000	500	1.500							
23/10/2017 12:00:00	Café Ristretto	400	1.500	400							
23/10/2017 16:00:00	Café Ristretto	700	300	800							
...											

Ela tem as colunas de entrada e saída, que são opcionais, mas é bom colocar, porque o pessoal de negócio sempre pede esse tipo de métrica adicional.

Essa opção salva só o saldo final do dia, então se fosse fazer uma análise em cima dessa fato, teria o saldo do estoque dia a dia.

Isso é bem comum, pega um Walmart da vida e olha a quantidade de entra e sai de estoque, então você vai precisar usar esse tipo de tabela, onde vai ter somente o consolidado daquele dia.

Então por que já não deixa só assim, sem a fato anterior?

Como eu falei, às vezes nós precisamos ter uma visão como na opção #1, mas em outros casos, vamos precisar de uma visão como na opção #2.

Nesse caso do exemplo você tem as duas opções, mas se não tiver a fato transacional, pode pegar direto do ETL e jogar para a snapshot.

Mas quando tem a transacional, é mais fácil ler ela e jogar na outra fato.

Na opção #3, tenho uma fato transacional diária, que é uma abstração da opção #2, então tenho a entrada e saída de produtos no dia. E também uma fato de snapshot mensal.

	A	B	C	D	E	F	G	H	I	J	K
10	OPÇÃO #3					OPÇÃO #3					
11		Fato estoque diaria transacional					Fato estoque diaria snapshot periodica				
12			produto	qtd_entrada	qtd_saida	saldo					
13	data		23/10/2017	Café Ristretto	2.000	500 1.500					
14			24/10/2017	Café Ristretto	400	1.500 400					
15			25/10/2017	Café Ristretto	700	300 800					
16			26/10/2017	Café Ristretto	2.000	500 2.300					
17			27/10/2017	Café Ristretto	400	1.500 1.200					
18			28/10/2017	Café Ristretto	700	300 1.600					
19			29/10/2017	Café Ristretto	300	500 1.400					
20			30/10/2017	Café Ristretto	300	1.500 200					
21			31/10/2017	Café Ristretto	250	400 150					
22			1/11/2017	Café Ristretto	2000	400 1.750					
23			2/11/2017	Café Ristretto	200	1100 850					
24			3/11/2017	Café Ristretto	300	500 650					
25			4/11/2017	Café Ristretto	200	400 450					
26											
27											

Então quando chega no último dia do mês, você pega o saldo e joga lá na snapshot mensal.

A forma de aplicar vai depender da sua necessidade, mas a ideia dessa fato é fazer um consolidado em algum período de tempo quando você tem muitos registros e isso pode comprometer o desempenho das consultas.

FATO SNAPSHOT ACUMULADO

Esta fato vai ser dividida em três etapas de atualização, mas entenda que etapa de atualização não é update na fato.

Então qual a diferença de um snapshot acumulado para o periódico? O periódico pega o momento no período, tira uma fotografia e insere na fato. O acumulado também é uma fotografia, mas em mais de um momento.

Como exemplo, peguei um caso que eu trabalhei em um centro de distribuição.

Nele, temos uma fato de estoque que controla a quantidade recebida do produto, a quantidade inspecionada, a quantidade armazenada e a data que cada uma dessas coisas aconteceu.

E como funciona essa fato?

Passo 1

Isso é um processo demorado, então em um primeiro momento, se recebe um lote de produtos.

A	B	C	D	E	F	G	H	I
1	ETAPAS DO FATO SNAPSHOT ACUMULADO							
2	Case: Recebimento de um lote de um produto no CD/Armazém.							
3								
4	PASSO #1							
5	Linha da fato inserida quando lote foi recebido pelo CD / Armazém:							
6								
7	Número Lote	Data	Data	Data	Produto(SK)	Quantidade	Quantidade	Quantidade
8	Recebimento	Recebimento(SK)	Inspeção(SK)	armazenado(SK)	Recebida	Inspecionada	Armazenada	
9								
10								
								Linha ÚNICA

Então no dia 10/04/2018 foi recebido 100 itens do produto 1. Aí no sistema de negócio deles, vão entrar com o processo de inspeção.

Passo 2

No outro dia aconteceu a inspeção, e é preciso armazenar isso na fato. Mas não precisa deletar tudo, você simplesmente revisita a fato.

A	B	C	D	E	F	G	H
12	PASSO #2						
13	fato revisitada quando o lote foi inspecionado.						
14							
15	Número Lote	Data	Data	Data	Produto(SK)	Quantidade	Quantidade
16	Recebimento	Recebimento(SK)	Inspeção(SK)	armazenado(SK)	Recebida	Inspecionada	Armazenada
17							
18							

Ou seja, não atualiza ou dá um update, mas volta na fato, porque a carga dela ainda não foi concluída.

Passo 3

A	B	C	D	E	F	G	H	
20	PASSO #3							
21	fato atualizada quando o lote foi armazenado							
22	Número Lote Recebimento	Data Recebimento(SK)	Data Inspeção(SK)	Data armazenado(SK)	Produto(SK)	Quantidade Recebida	Quantidade Inspecionada	Quantidade Armazenada
23	210	10/04/2018	11/04/2018	12/04/2018	1	100	20	98
24								
25								
26								

Depois, no dia 12/04/2018, os produtos foram armazenados, aí novamente, você revisita a fato e a data de armazenamento e quantidade são inseridas nela.

Algumas questões sobre esse tipo de fato:

Muitas vezes as pessoas acham que isso aqui vai aumentar a complexidade do ETL e vai ficar mais difícil de controlar, e é verdade, vai mesmo.

Você está fazendo paradas no processo por dias, então o processo não vai terminar nesse tempo. Algumas pessoas preferem dar uma carga só depois que já foi tudo carregado, o que é possível também.

Esse caso aqui é para quando você precisa monitorar o avanço, por exemplo, quando eu quero entender quanto tempo levou entre receber e inspecionar, ou inspecionar e armazenar.

É importante você entender aqui: quando você precisa revisitar a fato para inserir mais dados nela sem deletar o registro que já tinha, isso se chama snapshot acumulado.

Quem geralmente usa isso? Logística ou times de venda/entrega.

No geral, para fazer isso acontecer, o ETL vai lá e carrega o primeiro campo, depois deixa a carga, ou seja, o job, voltar depois de

tantos dias, então carrega um outro pedaço, e depois que está completo, ele dá OK naquele processinho ali.

FATO SEM FATO

Uma tradução para o nosso dia a dia seria: fato sem métricas.

Ela também é chamada de fato de associação ou de intersecção, mas o termo técnico é fato sem fato.

Ela serve para fazer uma intersecção de dimensões. Às vezes a gente quer comparar ou cruzar algo somente entre duas dimensões e não tem uma métrica para fazer essas comparações.

Essa fato é a exceção, só é usada quando se precisa fazer uma intersecção entre as dimensões.

Exemplos de fato sem fato:

- frequência de aluno;
- venda com promoção.

Imagina que tem as dimensões de aluno, universidade, curso, professor e a tempo. E tem a fato é frequência. O objetivo dessa fato é fazer a associação entre as dimensões.



Ou seja, quero comparar os alunos com os cursos, os alunos com os professores, o tempo com o curso, o curso com o professor, e por aí vai, mas não tem nenhuma métrica envolvida.

E normalmente quando precisa desse tipo de fato é neste caso que eu vou mostrar ou uma análise do que não vendeu em uma promoção.

Por que isso acontece? Lembra que para entrar na fato precisa ter um fato ocorrido, e que se não tem fato não tem dado?

Por exemplo: No dia 10 de dezembro entrou um aluno para o curso de Design de Data Warehouse, esse é um fato que ocorreu e ponto, é uma fato transacional. Agora, eu quero saber o que não aconteceu, ou seja, quando tem nulo, de não aconteceu nada.

Você pode ver que nesse exemplo a fato só tem foreign keys, eu uso ela só para fazer os cruzamentos e apresentar os dados, porque se não eu não teria como ver. É isso.

Alguns desses tipos não serão utilizados no seu dia a dia, mas é importante você saber que todos existem para poder voltar neles quando alguma situação atípica acontecer. Então mesmo que não aplique todos, tenha essa lista à mão para quando precisar.