Business Intelligence

Implementar do jeito certo
e a custo zero

Casa do Código

RONALDO BRAGHITTONI

ISBN

Impresso e PDF: 978-85-5519-252-4

EPUB: 978-85-5519-253-1

MOBI: 978-85-5519-254-8

Você pode discutir sobre este livro no Fórum da Casa do Código: http://forum.casadocodigo.com.br/.

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

A QUEM ESTE LIVRO SE DESTINA

Os dois primeiros capítulos deste livro são puramente conceituais. Não tratam de tecnologia, nem demandam conhecimentos em programação, quaisquer que sejam. Eles se destinam aos que precisam entender o que é (de verdade) *Business Intelligence* (BI), do que ele é composto e o que esperar dele, por que implementá-lo e os desafios de um projeto de BI.

Os demais capítulos discorrem objetivamente sobre o passo a passo para implementar uma plataforma de BI na íntegra, com exemplos, códigos, explicações e conceitos que lhe permitirão solucionar os cenários reais de sua empresa. Organizado dessa forma, este livro se destina a:

Todos aqueles que estão com o clássico problema: "preciso melhorar as ferramentas de tomada de decisão da minha empresa! O que fazer?"

Neste livro, definimos o que é Business Intelligence e para o que ele serve. Com esses subsídios, entender qual o momento de sua empresa e avançar nesse caminho se tornarão tarefas bem mais claras.

Todos aqueles que já sabem o que fazer, mas ainda não têm ao certo o "como".

Cobrimos a criação de uma solução de BI desde a sua concepção até a disponibilização final das análises aos usuários em seus diferentes níveis! Trilhamos o caminho item a item, a fim de determinar o que fazer, independentemente de você seguir as tecnologias aqui empregadas ou outras similares.

Todos aqueles que, conhecendo bem ou não os conceitos de BI, acreditam que é uma solução cara demais para sua empresa ou para seu momento.

Sem o gasto com ferramentas e usando os recursos disponíveis em sua empresa, pode-se progredir **muito** em termos de soluções de tomada de decisão. Este livro guia os passos que lhe levarão do zero ao melhor uso de uma ferramenta gratuita e, ainda, com total possibilidade de evolução para as soluções pagas mais conceituadas do mercado!

Todos os interessados em se aprofundar conceitualmente e tecnicamente em Business Intelligence.

Se sua empresa investiu em um projeto de BI que se inicia, ou se você simplesmente está estudando o assunto, não deixe de ler este livro. Uma enorme quantidade de informação errada ou simplesmente tendenciosa bombardeia quem se embrenha nessa ciência. Este livro fornece fundamentos claros e isentos sobre o tema!

SOBRE O AUTOR

Ronaldo Braghittoni, executivo de Tecnologia da Informação com mais de 15 anos de experiência em implantações de Business Intelligence e Gestão de Projetos. Co-fundador e Diretor de operações da consultoria SENNIT (http://www.sennit.com.br). Evangelista convicto dos processos de Gestão pragmática e Transformação digital.

Sumário

1 Introdução **1**

1.1 Definindo Business Intelligence 1

1.2 O projeto Business Intelligence 11

1.3 Por que implementar BI a custo zero 17

1.4 Conclusão 20

2 Arquitetura e ambiente **22**

2.1 Arquitetura de uma plataforma de Business Intelligence 22

2.2 A tecnologia escolhida e a montagem do ambiente 37

2.3 Conclusão 52

3 O desenho do Data Warehouse **53**

3.1 O cenário 53

3.2 O Data Warehouse 62

3.3 Conclusão 87

4 O processo de ETL **88**

4.1 Considerações iniciais 88

4.2 Data Stage 89

4.3 Manipulação de arquivos 92

4.4 Carregando a dimensão Data 99

4.5 Carregando a dimensão Cliente 103

4.6 Carregando a dimensão Geografia 107
4.7 Carregando a dimensão Funcionário 113
4.8 Carregando a dimensão Produto 116
4.9 Carregando a Fato de Venda 120
4.10 Carregando a Fato de Detalhe de Venda 125
4.11 Agendamento e carga no dia a dia 129
4.12 Revisão e testes 137
4.13 Conclusão 139

5 Expondo as informações 140
5.1 Opções de consumo do Business Intelligence 140
5.2 Obtendo as ferramentas e add-in 141
5.3 Criação e publicação do dashboard 150
5.4 Criação e publicação do relatório de Detalhe 174
5.5 Criação e disponibilização da Planilha de Consulta Dinâmica
5.6 Conclusão 193186

6 Estendendo a sua plataforma gratuita 194
6.1 Alternativa para a limitação de tamanho do Data Warehouse
6.2 Reporting Builder 195194
6.3 Power BI Desktop 198
6.4 Conclusão 200

CAPÍTULO 1

INTRODUÇÃO

1.1 DEFININDO BUSINESS INTELLIGENCE

Business Intelligence (ou BI) é um termo cunhado por Howard Dresner do Gartner Group, em 1989, para descrever um conjunto de conceitos e métodos para melhorar o processo de tomada de decisão das empresas, utilizando-se de sistemas fundamentados em fatos e dimensões. O BI baseia-se em agrupar informações de diversas fontes e apresentá-las de forma unificada e sob uma métrica comum, a fim de que indicadores aparentemente distantes possam fazer sentido entre si.

Ou seja, BI é:

*"Uma **metodologia** pela qual se estabelecem ferramentas para obter, organizar, analisar e prover acesso às informações necessárias aos tomadores de decisão das empresas para analisarem os fenômenos acerca de seus negócios".* — Howard Dresner

É interessante observar dessa definição que:

- **BI é uma metodologia, não uma ferramenta.** Isso significa que se pode implementar BI com praticamente *qualquer* ferramenta de controle de dados, ou com o conjunto de quaisquer ferramentas próprias de BI, bastando conhecer a metodologia.

  Curiosamente, o que se vê no mercado é uma disputa

dos "players" alegando que suas ferramentas são melhores, que podem fazer isso ou aquilo e que os concorrentes deles não fazem. O interessante é que a gigantesca maioria das funcionalidades de uma ferramenta de BI são **idênticas** entre as opções de mercado. E mais curioso ainda é que, para uma implantação nova em uma corporação, nem metade dessas funcionalidades serão necessárias.

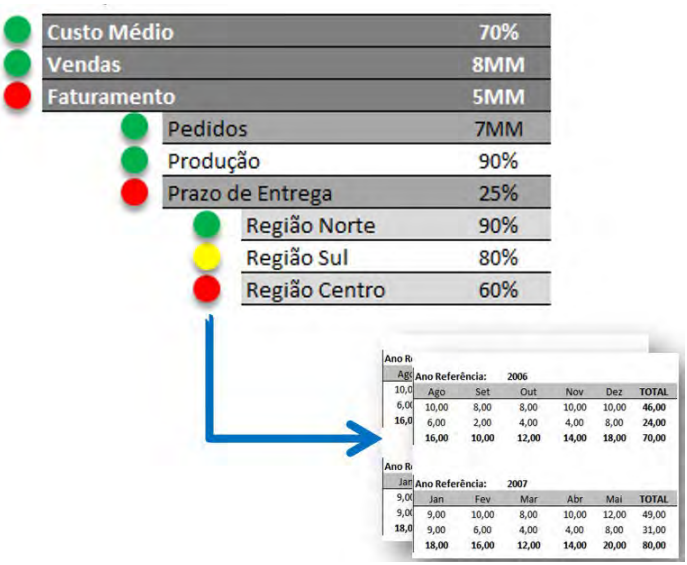
Vamos discorrer um pouco mais sobre esse tema logo adiante, mas o diferencial entre as ferramentas pode ser algo necessário apenas quando um elevado nível de maturidade dos usuários é alcançado. Antes disso, o que se tem é preciosismo e argumentação vazia de venda.

- **BI serve para analisar os fenômenos acerca do negócio.** Isso significa que Business Intelligence precisa ser uma plataforma capaz não só de aglutinar as informações transacionais, mas também de exibi-las de forma contextual, fazendo com que fenômenos escondidos se tornem visíveis. Um exemplo bem acadêmico desse conceito é o indicador de faturamento.

  Imagine que o CEO de uma empresa receba a informação de que o faturamento do mês está abaixo do esperado. Essa informação por si só não indica a causa desse problema.

  Sem mais dados, esse CEO ligaria para o VP Comercial e descobriria que as vendas estão acima do previsto. Se estamos vendendo bem, por que não estamos faturando bem? Ele então ligaria para o VP de Produção e descobriria que a produção está exatamente na meta, tendo manufaturado todos os pedidos. Se vendemos e

produzimos, o que impede o cliente de aceitar o faturamento? Ligando para o CFO, ele seria informado de que as NFs foram emitidas, mas os clientes de uma determinada região estão negando o pagamento! Falando com o COO, ele é informado de que as entregas de uma região não estão ocorrendo, porque uma transportadora está em greve!



Nesse exemplo, nota-se a importância não só de ter a informação, mas de tê-la de forma contextual, em conjunto com outras informações relevantes. O "fenômeno" nesse caso seria o impacto na meta de faturamento por conta de um fornecedor de transporte.

Já dizia a máxima de que:

"O conjunto de dados gera um registro, o conjunto de registros gera uma informação e o conjunto de informações gera o conhecimento!" — Autor desconhecido

Um CEO de posse desse conhecimento poderia facilmente promover uma multa ao fornecedor, a contratação de um backup de entrega etc. Enfim, o BI bem implementado deve buscar tornar essas relações facilmente visíveis e, mais do que isso, integradas aos processos da empresa e não de um indivíduo. Quando os fenômenos são mapeados e expostos pelo BI, eles deixam de ser de um único funcionário ou departamento, e passam a ser de propriedade da corporação.

Mas, para se falar de Business Intelligence, precisamos conceituar alguns termos com os quais conviveremos daqui por diante. Vamos ter em mente que implementar uma solução de BI é criar uma arquitetura que poderá evoluir ao longo do tempo, de algo simples para uma solução extremamente "parruda" que permeará toda a organização e, por que não, muito mais além dela!

O primeiro grande conceito é de que as informações do BI são cópias dos dados dos sistemas da sua empresa e, se necessário, de fontes externas. Mas o Business Intelligence, por definição, não deve gerar informações que não as de estatísticas sobre os dados importados das fontes chamadas transacionais.

Ter "uma telinha" que insere dados diretamente na base do seu BI (que se chama *Data Warehouse*, como veremos) é algo academicamente bastante questionável e que eu recomendo fortemente que evitemos! Assim sendo, temos que:

- **Sistemas transacionais:** sistemas em que as transações do dia a dia são geradas e atualizadas. São o ERP, o CRM, Sistema de pedidos, de chamados etc. Esses sistemas são chamados de OLTP (*on-line transaction process*).

- **Sistemas analíticos:** é o seu Business Intelligence, que importa os dados dos transacionais e disponibiliza as

informações de forma que elas sejam analisáveis pelos usuários. Esses sistemas são chamados de OLAP (*on-line analytical process*).

Observação

Usualmente, usa-se o termo OLAP para definir apenas os bancos de dados multidimensionais (os chamados "CUBOS", que comentaremos adiante) e não a plataforma toda de BI. Não vou entrar no mérito de certo ou errado de definições, mas não só de cubo vive a análise.

Mas por que copiar os dados dos Sistemas transacionais? Eu não posso consultar diretamente deles?

Essas perguntas são frequentes! Tão frequentes que alguns fornecedores de ferramentas de BI simplesmente pulam a etapa de copiar os dados, e passam a apresentar informações com leitura direta dos sistemas transacionais. Mas responder essas perguntas mostrará a fragilidade da abordagem desses "players".

Copiam-se os dados para uma base centralizada por 4 motivos principais:

1. **Os dados podem ser consultados sem atrapalhar o processamento diário dos sistemas transacionais:** fazer consultas que somam históricos semestrais, anuais ou intervalos ainda maiores pode impactar na performance do sistema, ou encarecer sua infraestrutura para que seu hardware seja capaz de responder às consultas sem impactar o dia a dia.

Imagine parar o faturamento de uma empresa no meio do último dia do mês porque se quer saber como foi o movimento do ano passado! A carga dos transacionais para o analítico é feita em horário controlado e sempre do "delta", ou seja, somente do que ainda não foi carregado. Geralmente, essas cargas se encerram durante a madrugada e não impactam nos processos produtivos.

2. **Os dados, ao serem copiados para uma base unificada, são validados:** por passar por processos de carga periódicos (o chamado ETL que veremos adiante), os dados são previamente validados quando centralizados pela plataforma do BI. Se houver algum erro, uma lista de exceção é criada e o dado não é computado. Análises de erro são periodicamente executadas. Sem esse processo, erros de sistemas podem passar anos desapercebidos!

3. **Os dados colocados no BI passam a ser "eternos":** sistemas transacionais muitas vezes possuem rotinas de expurgo de dados a fim de garantir a performance. Se os dados estiverem na sua plataforma de BI, eles podem ser apagados dos transacionais sem problemas! Consultas comparando ano a ano desde a última década, por exemplo, podem ser executadas sem a necessidade de se retornar backup de arquivos mortos etc.

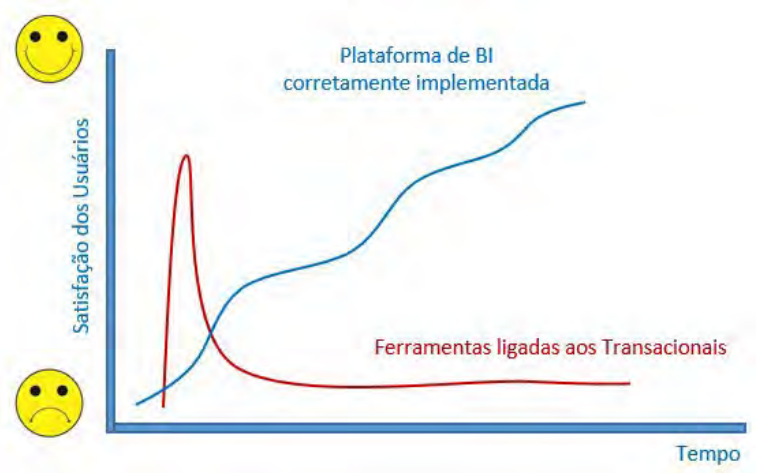
4. **Os dados de sistemas diferentes se tornam "próximos":** não é incomum você encontrar a mesma informação guardada de forma diferente em sistemas distintos. No sistema de Faturamento, você tem o `Id_cliente` como numérico e, no CRM, você tem o `CD_CLI` como texto. Ambos têm a mesma informação e tratam do mesmo cliente.

   Eventualmente, essas informações estão gravadas de forma tão distinta que, por exemplo, saber qual o faturamento do cliente

que abre mais reclamações pode ser algo bastante complexo! Ao unificar as informações em um local, essas consultas de dados de vários sistemas se tornam extremamente simples.

Isto é, ferramentas que efetuam consultas diretamente contra as bases transacionais são interessantes pela velocidade com que são implementadas. Porém, por definição, não podem ser chamadas de ferramentas de BI. Quando muito, são ferramentas de relatórios.

É interessante notar que essas ferramentas possuem um apelo de vendas muito forte e que, em demonstrações e reviews, são sempre bem cotadas. Isso porque elas têm a capacidade de exibir as informações oriundas dos transacionais de forma extremamente avançada e com apenas alguns cliques. Contudo, quando os usuários passam a estágios mais avançados do consumo de informações (conforme falaremos adiante, ainda neste capítulo), elas passam a gerar frustração.



A satisfação é gerada no momento em que se pergunta **"Quais as informações?"**. Essa resposta é dada rapidamente. Mas isso é um subset do que uma plataforma de BI pode fornecer, e a frustração

vem quando surge a pergunta **"Qual a causa?"**.

Uma plataforma de Business Intelligence é implantada de forma mais lenta, e demora para atender a todos os usuários e gerar todas as informações. Mas, tendo sido implementada em sua totalidade, o BI sedimenta a satisfação dos usuários a cada nova implantação de origens de dados, porque atende aos requisitos de sua definição de analisar os fenômenos, às consequências (informações) e às causas (relações entre informações de toda a corporação).

Uma vez que vamos centralizar os dados dos sistemas transacionais em uma única base de dados, por que não deixá-la preparada para responder às consultas da forma mais performática possível?

E é exatamente isso que faremos! Ao criar uma base centralizadora, que receberá todos os dados da empresa, criaremos o chamado *Data Warehouse*, ou apenas DW! Teremos um capítulo adiante só sobre como ele é criado. Na minha opinião, este tema é a chave para o sucesso ou fracasso de uma implementação de BI. Desenhar um bom DW ou não será a diferença entre facilitar ou impossibilitar o dia a dia de quem usa e opera a plataforma de Business Intelligence.

Uma boa referência bibliográfica nesse tema é o livro *The Datawarehouse Toolkit*, do professor Ralph Kimball. O que temos de saber por hora é que um *Data Warehouse* é formado por dois tipos de entidades e que elas permearão toda a relação que teremos com nosso BI:

- **O fato:** fato, ou medida (*measure*), é toda informação que será matematicamente analisada. São as quantidades, valores, médias etc. Por exemplo: o total de vendas, a quantidade de chamados, a média de dias etc.

- **A dimensão:** é a informação pela qual analisamos os fatos, ou seja, as visões pelas quais veremos os números. Por exemplo, se formos analisar o total de vendas, faremos pelo quê?
    - Total de vendas por Mês;
    - Total de vendas por Mês; por Produto;
    - Total de vendas por Mês; por Produto; por Loja;
    - Total de vendas por Mês; por Produto; por Loja; por Vendedor;
    - E assim sucessivamente.

      Cada "por" que adicionamos à consulta é uma nova Dimensão. Quanto mais estratificarmos a informação, mais dimensões estamos adicionando à análise.

Outro termo bastante importante é **atência**, ou seja, é o tempo de defasagem da informação. Dado que o BI precisa de um processo de carga que ocorre de tempos em tempos, a latência é o intervalo entre cada atualização.

Para a boa parte das situações, a latência é diária. Isto é, o processo de atualização ocorre toda noite, tornando disponíveis as informações atualizadas até o dia anterior, o que se chama **Latência D-1**. Pode-se ter uma latência de algumas horas, com processamentos em intervalos menores do que um dia. Se formos efetuar cargas a cada 12 horas, por exemplo, a latência seria **H-12**.

Disponibilizar os dados na plataforma de BI em tempo real, o **"Real-Time BI"**, é possível e, de alguma forma, até comum. O que se deve levar em consideração é que o processamento custa. Se formos fazer cargas em tempo real, o processamento — tanto dos transacionais quanto do BI — precisará suportar uma carga de trabalho considerável. E isso representa investimento, seja em hardware, ou em contratação de processamento para ambientes em

nuvem.

Se haverá custo, temos de entender a necessidade de se ter sempre o último dado contabilizado em uma análise história! Será que, por exemplo, a venda feita nesse último minuto impactará a análise de um ano inteiro? Se não, os processos com maior latência são mais recomendados!

Uma vez implementado, o BI será consumido pelos seus usuários. Algo óbvio, mas que engloba a que talvez seja a maior armadilha de todas! Consumir informações requer maturidade, treinamento. Isso mesmo, seu usuário precisa **aprender** a ter informações disponíveis. O entendimento do que deve ser entregue para cada usuário é de suma importância.

Existem basicamente 3 níveis de usuários que devem ser atendidos e, para cada um deles, um modelo estrutural de informação:

- **Corporativo:** são as visões mais aglutinadas, os KPIs e *cockpits* de dados agrupados. Geralmente, são os diretores e gestores que consomem essas informações.
- **Departamental:** são os relatórios detalhados e as visões com pesquisas. São utilizadas pelos coordenadores e analistas que operam o dia a dia.
- **Pessoal:** são as planilhas dinâmicas que geram análises precisas de dados. São usados pelos especialistas (independente da hierarquia) que buscam novas análises, novos fenômenos e relações.

Falaremos ainda bastante sobre essas formas de apresentação. Mas o foco é entender que entregar uma planilha dinâmica para quem espera visualizar um cockpit, e vice-versa, é o caminho para fracassar a implantação do BI!

1.2 O PROJETO BUSINESS INTELLIGENCE

Um projeto de BI pode ter três tipos de fracasso e, infelizmente para nosso mercado, esses fracassos são encontrados com mais frequência do que os sucessos.

- **O projeto todo foi abandonado** — Não é incomum encontrar empresas que já tiveram iniciativas de BI no passado, mas abandonaram o projeto antes dele apresentar sequer alguma informação. As causas para esse tipo de fracasso são inúmeras, a começar por um time desqualificado para a implantação do BI até um cliente totalmente despreparado.

- **O projeto foi abandonado, mas existem usuários ativos** — Esse problema é um pouco menos comum porque, tendo usuários ativos, geralmente opta-se por sofrer mais um pouco e terminar o que se começou. Mas já presenciei empresas que possuem BI atendendo a uma área e somente a ela, porque outras implantações foram proibidas. Pode-se imaginar o sofrimento e o custo dessa implantação para que se ter uma empresa inteira avessa à continuidade do que se começou.

- **A solução está em uso, mas gasta-se *muito* mais do que o necessário** — Na minha opinião, esse é o *pior* tipo de fracasso. Isso porque é um fracasso disfarçado de sucesso. Os usuários estão felizes, e o fornecedor coloca orgulhoso o logo do cliente em suas apresentações institucionais. Porém, esse tipo de projeto fomenta as máximas de que "o BI é caro", "eu não tenho tamanho para ter uma plataforma de BI" etc.

  Gasta-se milhões para se implementar o que milhares

resolveriam. Atualmente, são diversos "players" cobrando vultuosas somas para implantar aquilo que um ou outros softwares combinados fazem de graça e com mais poder!

Mas ao se decidir por implantar uma solução de BI, independente de usando este livro como guia ou tendo comprado uma plataforma de muitos milhões, é preciso ter em mente os benefícios que ele trará. Parece mais uma afirmação óbvia, mas vamos mencionar algumas boas práticas nesse sentido mesmo assim.

Quando uma empresa decide por investir em uma implantação de BI, geralmente algum problema está sendo apontado como *"sponsor"* da iniciativa. Por exemplo, pode-se implantar BI para que "um book informativo seja gerado instantaneamente, dado que atualmente se leva uma semana".

Nesse contexto, é interessante ter bem mapeado os *benchmarks* de como o processo era, quanto tempo ele levava e quanto custava e, se possível, qual a sensibilidade dele a erros de dados. Ao final da implantação, demonstrar os ganhos será fundamental para que a iniciativa cresça e alcance novas áreas e funções.

Informação e conhecimento têm um efeito viral dentro das empresas. Quando um departamento tiver acesso a informações confiáveis e de forma instantânea, mais e mais demandas aparecerão. Manter o controle de como era o processo antes do BI e depois de cada uma dessas iniciativas dele é algo que historicamente se mostrou extremamente importante. A memória de quem paga a conta é, geralmente, bastante volátil.

Nr	Processo	Demandante	Objetivo	Sensibilidade
1	Geração do Book Mensal	CFO	Criar o Book financeiro mensal	EXTREMA
2	Informativo de Romaneio	Logística	KPIs de Logística para acompanhamento diário	MÉDIA
3	Plano de MKT	MKT	Mapa de eficiência de Campanhas	MÉDIA

	Antes da Implantação			Pós Implantação				
Nr	Tempo Consumido (Mês)	Recursos envolvidos	Custo Mensal	Tempo Consumido (Mês)	Recursos envolvidos	Custo Mensal	Versão Pacote	ROI
1	4d	2	R$ 952,00	0	0	0	v_01	R$ 952,00
2	80h	1	R$ 2.380,00	0	0	0	v_02	R$ 2.380,00
3	168h	1	R$ 5.600,00	80h	1	R$ 2.660,00	v_03	R$ 2.940,00

REALIZADO			
Versão	TCO (12 meses)	ROI (12 meses)	SALDO
v_01	R$ 20.000,00	R$ 11.424,00	-R$ 8.576,00
v_02	R$ 5.000,00	R$ 28.560,00	R$ 23.560,00
		ROI FINAL:	**R$ 14.984,00**

PREVISTO			
Versão	TCO (12 meses)	ROI (12 meses)	SALDO
v_03	R$ 36.920,00	R$ 35.280,00	-R$ 1.640,00
		ROI FINAL:	**R$ 13.344,00**

Note no exemplo anterior (extraído de uma implantação para uma empresa de logística), que uma ou outra implantação sairia mais cara no primeiro ano do que o próprio projeto de BI. Isso é bastante normal principalmente para a primeira implantação que paga todo o "setup" do projeto. Contudo, mais e mais implantações e o decorrer dos anos provarão a viabilidade financeira do investimento, principalmente quando feito sem custo de licenciamento.

Outro ponto é quanto à sensibilidade das informações: vimos uma vez que uma empresa de energia foi multada em alguns

milhões por fornecer dados incorretos ao ministério. Só essa multa pagaria toda a implantação de uma plataforma de BI que geraria as informações automaticamente e sem erros! Isso sem contar nos danos à marca.

Uma vez feitos os levantamentos de demandas e feita a priorização de qual iniciativa será a primeira, e depois de se levantar o *benchmark*, inicia-se o projeto propriamente dito. Não vou me delongar em processos de gestão, mas uma das questões mais importantes de um projeto é o seu **escopo**. E definir o escopo de um projeto de Business Intelligence pode ser algo complicado.

Por sua natureza, uma plataforma de BI deve permitir ao usuário a criação de suas próprias visões de dados, de seus próprios relatórios. Isso gera infinitas possibilidades de relatórios, de tabelas dinâmicas e gráficos.

Então como contemplar tudo isso em um escopo e ainda evitar que se torne um "projeto infinito"?

Existe um artefato que vem em nossa salvação!

A **Matriz de Fatos e Dimensões** (ou MFD) é um documento que rege o que cada implantação do BI vai conter e, de quebra, nos apresenta dados técnicos como a granularidade, fórmulas e origens. Todos esses termos serão cobertos em detalhes nos capítulos técnicos a seguir, mas para a boa gestão de um projeto de BI, deve-se, antes de iniciar o desenvolvimento, criar o mapa do que será entregue. Esse mapa é a MFD!

Dimensão	Data				Empresa	
Hierarquia Default	Data Composta	Ano	Mês	Dia	Empresa	Nome
Type	Time	Years	MonthOfYear	DayOfMonth	Regular	DB_STR
Relation(SF)	-	-	-	-	-	-
MEASURES						
Qtd_Solicitação		X	X	X		X
Aging (Dias)		X	X	X		X

Tomando por base esse exemplo, temos que a coluna `Measure` serão os Fatos que serão entregues e que as colunas ao lado (Data, Empresa, Diretoria, Gerência e Colaborador) serão as Dimensões.

Os `X` são os cruzamentos oferecidos. No exemplo, mostraríamos quantas solicitações foram feitas e qual a idade em dias de cada uma delas, por data e para cada colaborador de cada empresa. E se seguíssemos a matriz, poderia haver Diretoria, Gerência, Departamento etc.

Ao lado esquerdo de `Measures`, podemos ter informações técnicas (que cobriremos nos tópicos de ETL e de visualização, adiante). Junto a `Dimensão`, também temos algumas informações técnicas que servem para guiar o desenvolvimento.

A princípio, se tivermos uma MFD apenas com o nome das Dimensões e das Measures, e os `X` representando o cruzamento entre elas, já temos o suficiente para a gestão de Escopo!

Uma vez definidas quais informações estarão disponíveis, deve-se ater a **como** exibi-las informações e para quem. Conforme comentamos anteriormente nas definições sobre as visualizações, a entrega das informações da forma correta para o público correto é uma das maiores possibilidades de problemas que permeará a vida da plataforma de BI.

Não obstante a esses entregáveis, um projeto de BI, como veremos a seguir, é feito pela implantação técnica de cargas, persistência e exibição, e também pela **capacitação dos usuários**. Principalmente em uma nova implantação, não espere que seus usuários saiam usando a plataforma sem nenhum problema!

A nomeação dos *Key users* é extremamente importante. Eles que serão treinados e responsáveis por multiplicar o conhecimento em suas respectivas áreas. Seguindo as melhores práticas de gestão

pragmática, é bom manter os *key users* informados do andamento de todos os passos do projeto e, se possível, colocá-los junto do processo de desenvolvimento da solução. Essa prática agiliza o ajuste de detalhes e coloca-os juntos do time do projeto, tornando todos corresponsáveis pelas entregas.

Uma vez que o BI esteja de fato entregando informações, é natural que os dados apresentados sejam confrontados com as formas anteriores de informações, ou seja, com as planilhas, books, e-mails etc. Por vezes as informações do BI serão postas à prova por conta de exibirem dados diferentes dos oriundos dessas fontes.

Será natural no processo de estabilização da plataforma que, durante o desenvolvimento e a homologação, encontre-se erros no BI (no processo de carga, na exibição etc.). Contudo, à medida que esses problemas são ajustados, os erros estarão frequentemente nas fontes de dados originais. E não se engane: será missão dos implementadores do BI provar que o Business Intelligence está certo e que essas fontes estão erradas. E isso vai dar trabalho!

Por esse motivo, um período bem "generoso" do cronograma do projeto deve ser direcionado para o **Data Quality**, processo no qual os dados do BI são confrontados com as antigas origens e um veredicto de qual fonte está correta deverá surgir.

Por vezes, passei horas e horas analisando planilhas à procura de erros em fórmulas, dados copiados errados, e afins. Se formos criar um cronograma padrão (MS Project), recomendo que os seguintes itens — mas não só eles — constem em seu WBS:

- **Kickoff do projeto**
    - Definir benchmark para ROI
    - Definir Sponsors e Power Users
- **Análise**
    - Identificação dos requisitos de negócio e riscos

    - Definir origens, qualidade e latência das informações
    - Definir regras de negócio corporativas e departamentais
- **Desenvolvimento**
    - *Data Warehouse (ou Datamart)*
        - Desenho
        - Testes de Mesa
        - Implantação
    - *Processos de Carga*
        - Desenho e regras
        - Implantação
    - *Relatórios e dashboards*
        - Dashboard comercial
        - Relatório de apoio ao vendedor
        - ...
    - *Data Quality*
        - Validação dos dados aprensentados vs. Fontes anteriores

1.3 POR QUE IMPLEMENTAR BI A CUSTO ZERO

A resposta "para não gastar, oras" não cabe aqui. A máxima de que o barato sai caro geralmente é verdadeira para o mundo da Tecnologia da Informação. A resposta mais correta seria "para viabilizar, oras".

O **Business Intelligence Maturity Model**, concebido pelo **The Datawarehouse Institute**, é uma classificação referente a como uma corporação lida com a informação e com o conhecimento:

Desavisado	Tático	Focado	Estratégico	Universal
Anarquia de informação: Dados espalhados pela organização em diversos sistemas e em diversos arquivos (excel, word, txt etc.)	**Distribuição de informação:** Sistemas espalhados sem centralização das regras e das informações. Diversas formas de entender uma mesma informação.	**Foco em algumas informações:** Informações centralizadas para determinados focos. Áreas com bastante maturidade de consumo de informações e outras nem tanto.	**Demanda estratégica de informação:** A obtenção e análise de informações é direcionada por estratégias da empresa, com Governança, "frameworks", e padrões.	**Informação para todos:** Toda a informação gerada na empresa é ubíqua e permeia todos os níveis, inclusive parceiros, clientes e fornecedores.

Se sua empresa não possui uma solução de Business Intelligence, é bem provável que ela se encontre entre os níveis Desavisado ou Tático. Ou seja, os dados estão espalhados em sistemas e em planilhas descentralizadas. Não é incomum dois departamentos chegarem a números diferentes para o que deveria ser a mesma informação.

Quando se opta por implementar uma plataforma de BI, passa-se para o nível Focado, ou seja, a informação e o conhecimento passam a ter a mesma relevância que a operação do dia a dia. Só que isso ocorre geralmente em uma determinada área. Um sponsor que entende essa necessidade e que solicita a implantação do BI. Ele, inicialmente, estará sozinho nessa empreitada.

Por isso que uma implantação a custo zero é importante

Já presenciei iniciativas que nunca passaram do estágio de proposta comercial, porque a área demandante não possuía o budget necessário para, sozinha, pagar pelos servidores, licenciamentos e consultoria. Como outros departamentos ainda estavam em estágios iniciais, a iniciativa do BI não saia do mundo dos desejos de poucos usuários.

Contudo, imaginando que o BI simplesmente "aparecesse funcionando" para essa primeira iniciativa, ele rapidamente passaria a ser desejado por outros departamentos da empresa e novas implantações de dados e visões seriam solicitadas. E esse seria o caminho para que a empresa atingisse os níveis Estratégico e, quem sabe, Universal.

O objetivo dessa implantação sem custos é o de, justamente, fazer a empresa dar os passos para o estágio Focado e, em certo nível, atingir o Estratégico. E esse é o segredo:

> Implantar uma solução gratuita que vai se pagar enquanto os usuários entendem o que é o Business Intelligence e o que ele pode oferecer, saindo assim do estágio Desavisado ou Tático e entrando no Focado. Mas também, implementar uma solução que seja facilmente transportada para uma arquitetura "parruda", com infindáveis recursos quando a corporação amadurecer para o estágio Estratégico!

Antes de haver necessidades reais que não são atendidas por uma plataforma gratuita, investir milhões em ferramentas pagas é um caminho perigoso. E se seu usuário nunca amadurecer a ponto de requisitar aquilo que só aquela ferramenta entrega? E se o budget for consumido no licenciamento e não no desenvolvimento? Que valor vamos entregar?

E nesse ponto entra outra afirmação bastante recorrente quando falamos em implementar uma solução que não envolverá custo de licenciamento:

"É preferível uma solução mais profissional para a nossa necessidade..."

Curiosamente, nada é mais amador do que usar o preço como fator decisório.

Recentemente, li um artigo de um sommelier que listava excelentes vinhos a um baixo custo. Para conseguir fazer essa lista, ele precisou ter um grande conhecimento sobre o que compõe um excelente vinho e ter habilidade suficiente para identificar essas características. Eu, por outro lado, já presenteei alguns amigos com péssimos vinhos mesmo tendo optado pelas garrafas mais caras da adega. A diferença é que ele entende **realmente** de vinhos, e eu sou um grande amador no assunto.

De forma análoga, ao optar por uma solução paga simplesmente por ela ser paga ou ser famosa, denota falta de conhecimento sobre o que cada ferramenta oferece e sobre quais são as reais necessidades de cada demandante.

É claro que optar por soluções caras e conhecidas trará um conforto a quem está implementando o BI. Se tudo falhar, pelo menos a escolha foi a menos controversa possível. Já ouvi justificativas como "o que mais eu poderia ter feito? Implementamos a ferramenta mais cara do mercado!".

E o mais interessante é que os insucessos não ocorreram **apesar** da opção da ferramenta, mas **por causa** dela.

1.4 CONCLUSÃO

O que vimos até agora são as definições básicas de Business

Intelligence e um pouco dos motivos pelos quais devemos implementá-lo paulatinamente, bem como os desafios e soluções dos projetos de implantação de uma plataforma de BI.

Nos próximos capítulos, entraremos no *hands on*. Vamos montar todo o ambiente necessário, e implementar a solução por completo!

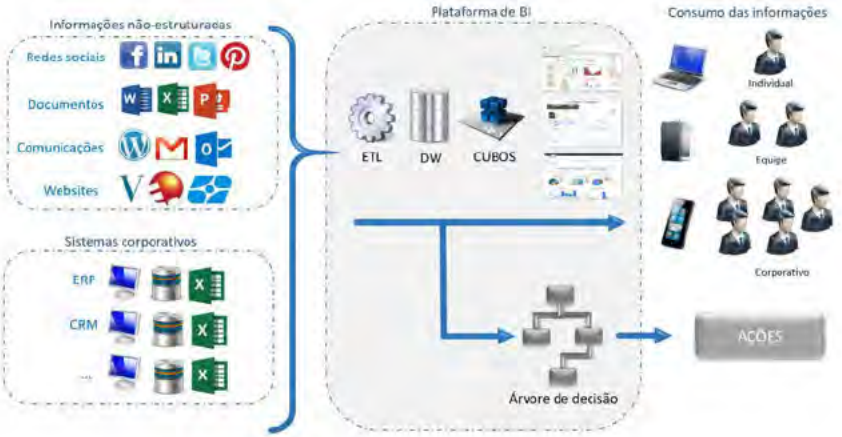
Capítulo 2

ARQUITETURA E AMBIENTE

2.1 ARQUITETURA DE UMA PLATAFORMA DE BUSINESS INTELLIGENCE

Como vimos no capítulo anterior, uma plataforma de BI deve ser implementada paulatinamente. Entretanto, desde seu início, ela deve ter por definição a capacidade de crescer, de agregar mais ferramentas, mais dados, mais usuários e atender a mais e mais necessidades.

Antes de entrarmos na tecnologia que suportará essa plataforma, vamos conhecê-la em mais detalhes afim de entender quais as preocupações que precisaremos ter em mente quando formos escolher essa tecnologia. No esquema a seguir, temos a plataforma de BI e seu encaixe dentro de uma corporação.



Note nos grupos mais à esquerda do esquema que temos uma série de possibilidades de origem de dados:

- **Sistemas corporativos:** como vimos no capítulo anterior, são os Sistemas Transacionais, o ERP, o CRM etc. A captura dessas informações pode ser feita por acesso direto ao banco de dados dela, por meio de serviços ou APIs, ou ainda pela extração de dados em planilhas ou arquivos de texto. Ainda assim, são dados estruturados e, geralmente, de obtenção mais simples.

- **Informações não estruturadas:** são informações não tabuladas. Podem ser e-mails, documentos ou dados espalhados por fontes externas. Por tratar-se de dados não estruturados, ou seja, com informações textuais sem um posicionamento predefinido, o uso dessas origens requer uma técnica específica.

NÃO CONFUNDA

O fato de dados serem obtidos por fontes externas à sua empresa, como redes sociais ou sites de informações financeiras, entre outras, não configura um Big Data.

ETL

Para que os dados sejam levados da origem até a plataforma de BI, tem-se um processo de carga, chamado de **ETL** (*Extract, Transform and Load*, ou Extração, Transformação e Carga). Houve um tempo em que o termo era **ETLM**, em que o M remetia à Manutenção (*Maintenance*).

Todavia, o termo extenso caiu em desuso e, hoje em dia, o ETL permeia o cotidiano das organizações quando o assunto é transportar dados de um ponto a outro. Tem-se uma série de ferramentas específicas para o ETL, como PowerCenter, Pentaho Data Integration, SQL Integration Services, e tantos outros. Contudo, veremos neste livro como fazer um processo de carga baseado em arquivos e em bases de dados bastante eficiente, sem o uso de nenhuma ferramenta específica.

Desse modo, precisamos ter a metodologia bem sedimentada, a fim de aplicá-la corretamente:

- **Extract:** é o processo de extração periódica dos dados das origens por meio da leitura de uma ou mais fontes de informação. Deve-se atentar para o fato de que essa periodicidade implica na latência da informação! Além disso, porque o processo de carga roda repetidas vezes, tem-se a necessidade de tratamento de erros e avaliação

de eventuais indisponibilidades.

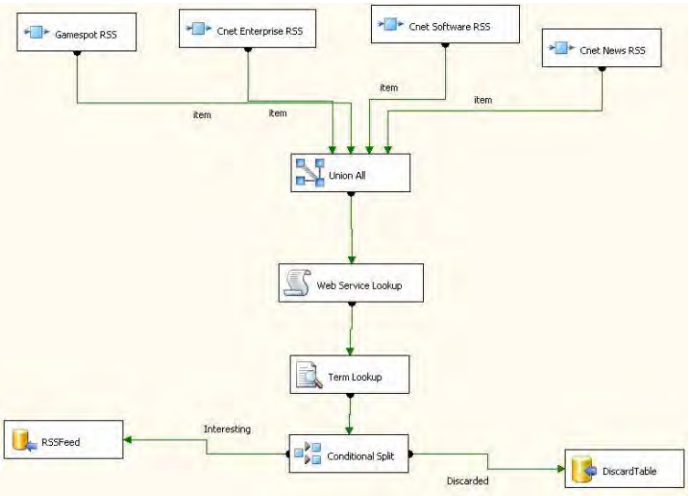
Em ferramentas específicas, todos os passos possuem logs e existem monitores de ocorrências. No nosso caso, nós mesmos implementaremos todos esses controles — e de uma maneira bastante simples.

- **Transform:** é o processo pelo qual os dados são trabalhados, colocados sob um formato específico, validados mediante as regras de negócio, calculados etc. Novamente, as ferramentas de ETL possuem diversos componentes de transformação pré-programados, que são configurados para cada carga. No nosso caso, trataremos os casos mais frequentes em conjuntos de comandos que farão o mesmo trabalho de forma relativamente simples.

  Claro que, conforme a complexidade de cada necessidade, o trabalho de implementar essas cargas via codificação aumenta e, eventualmente, pode-se gerar custos com a necessidade de contratação de consultorias especializadas. Se a solução chegar a esse ponto, uma reavaliação deve ser feita a fim de se determinar se já não seria o momento para um upgrade da versão gratuita para a paga. Veremos mais sobre esse processo decisório.

- **Load:** é a inclusão propriamente dita dos dados na plataforma de BI. A inclusão deve ser feita de forma incremental, pois, como veremos adiante, uma informação inserida não poderá sofrer alterações (existem exceções, mas por hora vamos nos ater à regra; mapearemos algumas exceções mais adiante). Isso implica em ter sempre a carga do chamado "delta", que será a diferença dos dados entre a última carga e o

momento atual.



Veja o exemplo de um package do SQL Integration Services, que é um fluxo de carga dos dados com agregações, consumo de serviços, obtenção de dados de diversas fontes e a persistência na base desejada. Um package geralmente atende a todo o processo, desde a extração, a transformação até a carga.

Data Warehouse (DW)

Como mencionamos anteriormente, o Data Warehouse é o coração de sua plataforma de BI! Ele é um banco de dados relacional como qualquer outro dos sistemas transacionais (servido pelos SGDBRs de mercado, tal qual o SQL Server, Oracle, DB2 etc.). Contudo, ele é desenhado para responder às pesquisas da forma mais performática possível.

Em bancos de dados Transacionais, temos de nos ater à díade clássica de que: se aumentarmos a performance para `SELECT` ,

estaremos denegrindo a performance para `INSERT`, `DELETE`, `UPDATE`, e vice-versa. Nesse ponto, as chamadas *formas normais* nos dão um guia de melhores práticas de modelagem de dados a fim de atingirmos um bom equilíbrio dessa equação. No DW, também chamado de banco de dados dimensional (por ser desenhado baseado em dimensões, como vimos anteriormente), as formas normais são justamente o que temos de evitar. Devemos desenhá-lo de maneira "desnormalizada".

Além disso, nossa base de dados centralizadora deve ser capaz de receber novas informações sem prejudicar as informações existentes. Ou seja, se já temos nela os dados do sistema de faturamento, devemos ser capazes de incluir os dados do CRM sem alterar o que já foi criado e, mesmo assim, deve-se poder relacionar as informações de ambos os sistemas!

O que, à primeira vista, pode ser extremamente complexo de se atingir, foi postulado em 1993 por W. H. Inmon, definindo assim como deve ser um Data Warehouse:

> *"Data Warehouse é um conjunto de dados orientado por assunto, conciso e integrado, variável com o tempo e não volátil."*

Entendendo cada uma dessas 4 propriedades:

- **Orientado por assunto:** isso significa que teremos de desenhar nosso banco de dados dividindo os assuntos em entidades que serão as chamadas dimensões. Por exemplo, se houver uma informação relacionada a Empresa, precisaremos ter uma tabela — e somente uma — que responderá por todos os atributos da

dimensão Empresa. Esta deverá ser vinculada a todas as tabelas de Fatos (como vimos, de dados que podem ser matematicamente trabalhados) que possuem alguma relação com essa dimensão.

Dessa forma, quando novos fatos forem acrescentados, bastará inserir a informação de Empresa — e das dimensões existentes — a fim de se possibilitar o cruzamento desses novos fatos com os já existentes. O DW poderá crescer indefinidamente.

- **Conciso e integrado:** significa que todo o dado inserido no DW deverá estar correto. Parece óbvio, mas nesse ponto reside a maior complexidade: definir as regras corporativas de cálculo, de origens e de validações. Em alguns casos, departamentos diferentes possuem regras diferentes para o que seria a mesma informação e, academicamente falando, deve-se chegar em um consenso para a inclusão do dado no DW.

  Já vivenciei situações em que esse consenso se mostrou impossível e tivemos de criar um Fato para cada departamento: Idade Estoque Produção e Idade Estoque Expedição. Os dois números deveriam ser iguais, mas por diferenças entre os entendimentos de cada departamento, geravam ligeiras diferenças entre eles. Como não se chegou em um acordo, precisamos optar por essa solução não muito ortodoxa. De qualquer forma, o modelo de dados se manteve correto, dado que cada uma dessas medidas era concisa com sua regra específica.

  Além disso, os dados são uniformizados. Informações idênticas podem ter formatos e máscaras distintas em diversos sistema. Ao inserir esses dados no DW, eles

devem assumir um formato padrão único, e assim permanecer.

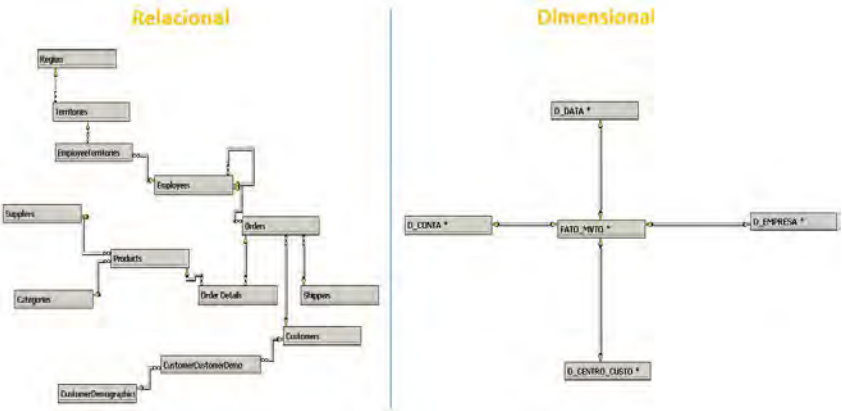
- **Variável com o tempo:** talvez seja esse o ponto-chave do DW. Sempre, invariavelmente e de qualquer forma, as informações devem ser posicionadas no tempo. Por mais que não exista nenhuma outra dimensão no seu DW, a dimensão temporal deve estar lá! Quando se faz a carga de valores de venda, por exemplo, precisa-se ter o dado de **quando** essa venda ocorreu mesmo que nenhuma outra informação seja carregada. E se por algum motivo estivermos carregando um dado que não possui informação sobre temporalidade, a data em que ele está sendo carregado no DW será a pontuação temporal dele!

  Isso garante a universalidade dos cruzamentos de dados. Se toda e qualquer informação for relacionada pela dimensão temporal, todas elas terão um ponto de cruzamento. Por exemplo, ao implantarmos uma plataforma de BI em uma grande indústria do setor de peças automobilísticas, cruzamos os dados do Sistema de Facilities (controle de manutenções prediais etc.) com o de Enfermaria (exames admissionais, atendimentos de funcionários etc.) e notou-se que, nos períodos em que mais chamados de manutenção apareciam, também mais notificações de afastamento por acidente de trabalho ocorriam. Resultado: criou-se uma política de manutenções preventivas para baixar a ocorrência de afastamentos. Um "fenômeno" entre duas áreas completamente distintas que foram aproximados pela dimensão temporal.

- **Não volátil:** conceitualmente simples, esse ponto roga

que, quando um dado é inserido em seu DW, ele se torna imutável! Não pode ser apagado, não pode ser atualizado! Isso porque, se for diferente, as informações do passado podem apresentar diferenças quando consultadas em momentos diferentes. Tudo bem que nosso futuro é incerto, mas ter incertezas no passado é algo que denigre a confiabilidade de qualquer informação. Contudo, existem leves exceções.

Vamos ver no modelo do nosso DW adiante que pode haver a necessidade de alterarmos "flags" em determinadas dimensões, de alterarmos um valor totalizador etc. Ter alterações extremamente controladas e com motivos bastante claros é permitido, mas temos de ter em mente que essas alterações não podem, de forma alguma, mudar as informações que o BI apresentará! Por essas especificidades que esse ponto é conceitualmente simples, mas de implementação que pode se tornar ligeiramente complexa.



Nessa figura, vemos a diferença entre o desenho de um banco de dados relacional, com suas tabelas normalizadas e de um banco de

dados dimensional, dividido entre fatos e dimensões.

Cubos

Se o DW é uma base de dados comum, um SGDBR como o seu transacional, já os cubos são outra coisa. Apesar de não tratarmos deles na implementação deste livro, vamos sim conceituá-los por se tratarem de um ponto que pode ser o maior incentivo para mudarmos da plataforma gratuita para a paga.

Os cubos são alternativas para os bancos de dados relacionais, ou ainda dimensionais, como chamamos o DW. São banco de dados multidimensionais que suportam análises dinâmicas e consultas extremamente complexas com baixo tempo de resposta.

Os SGBDM (Sistemas de Gerenciamento de Banco de Dados Multidimensionais) são ferramentas que pré-processam as agregações de dados, deixando informações prontas de todos os cruzamentos possíveis, a fim de que seja dada resposta instantânea à consulta do usuário. Da mesma forma que temos diversas tecnologias de SGDBRs, temos diversos SGDBMs como o SQL Analysis Services, IBM Cognos, SAS etc.

Os cubos são formados por um mapa de acesso, informações detalhadas e valores agregados, sendo possível criá-los em principalmente três tipos, conforme o esquema seguinte:

- **MOLAP (Multidimensional OLAP)**
  - Dados e agregações armazenados no cubo
  - Boa performance de processamento
  - Maior consumo de espaço
- **ROLAP (Relational OLAP)**
  - Dados permanecem no relacional
  - Agregações são armazenadas no relacional
  - Pior performance de consulta
  - Pior performance de processamento
  - Maior carga de processamento para o DW
  - Dados real-time
- **HOLAP (Hybrid OLAP)**
  - Dados permanecem no relacional
  - Agregações no cubo
  - Melhor tempo de processamento
  - Performance média de consultas
  - Menor consumo de espaço

Como não vamos implementá-los, não vamos entrar em extremos detalhes sobre os cubos. Porém, vale mencionar que, se um DW estiver com um tempo de resposta muito alto (mesmo utilizando as ferramentas de exibição que usarão processamento local, como veremos adiante), os cubos se farão necessários.

Vale lembrar de que, além do custo de licenciamento da plataforma paga a fim de se obter essa tecnologia, a implantação de um SGBDM por si só implicará na aquisição de hardware (eles consumirão um enorme processamento e uma grande quantia de disco). Não só isso, mas costumeiramente são mais caros os profissionais qualificados para trabalhar com a criação dos cubos, MDX (**M**ulti**D**imensional **E**xpression, que é a linguagem de consulta aos cubos), manutenção de acessos e demais necessidades para o bom uso dessa tecnologia. Ao optar pela implementação de soluções multidimensionais, tenha em mente que seu TCO vai subir, e que a necessidade real deles deverá estar bastante bem mapeada!

Exibição

Uma vez que tenhamos os dados corretamente disponíveis em nosso DW (e em nossos Cubos, se for o caso), podemos exibi-los para os usuários. Existem cada vez mais formas de apresentar informações, mas vamos cobrir as mais comuns (que resolvem 90% das necessidades) e que serão as cobertas na implantação deste livro:

- **Relatórios:** sem dúvida, a forma de apresentação mais conhecida, o relatório é uma formatação predefinida de colunas que apresenta registros (linhas) de acordo com os filtros de uma seleção. Não é incomum esses relatórios apresentarem centenas e até milhares de linhas. São usados para acessar informações mais detalhadas e, em geral, não se relacionam com

informações de outras fontes. Ou seja, um relatório chamado Vendas apresentará todas as vendas (registro a registro) de um determinado intervalo de tempo pela loja, pelo produto etc.

Sales Order Detail 2008

ADVENTURE WORKS cycles

Sales Order
Order #: SO50750

Bill To:	Central Discount Store 259826 Russell Rd. South Kent, Washington 98031 United States	Ship To:	Central Discount Store 259826 Russell Rd. South Kent, Washington 98031 United States

Contact:	Jean Handley	Purchasing Manager	582-555-0113
	Carina Stanev	Purchasing Agent	597-555-0100

Date	Order Date	Sales Person	Purchase Order	Shipment Method
7/31/2009	6/1/2003	David Campbell, Sales Representative 740-555-0182	PO7192170677	CARGO TRANSPORT 5

Line	Qty	Item Number	Description	Tracking #	Unit Price	Subtotal	Discount	Item Total
1	2	CA-1098	AWC Logo Cap	373D-417C-AE	$5.19	$10.37	$0.00	$10.37
2	4	GL-F110-M	Full-Finger Gloves, M	373D-417C-AE	$22.79	$91.18	$0.00	$91.18
3	4	LJ-0192-L	Long-Sleeve Logo Jersey, L	373D-417C-AE	$28.84	$115.36	$0.00	$115.36
4	3	GL-F110-L	Full-Finger Gloves, L	373D-417C-AE	$22.79	$68.38	$0.00	$68.38
5	1	BK-M68B-42	Mountain-200 Black, 42	373D-417C-AE	$1,229.46	$1,229.46	$0.00	$1,229.46
6	1	TG-W091-L	Women's Tights, L	373D-417C-AE	$44.99	$44.99	$0.00	$44.99
7	1	BK-M68S-38	Mountain-200 Silver, 38	373D-417C-AE	$1,242.85	$1,242.85	$0.00	$1,242.85
			Total Discount: $0.00				Total:	$2,802.60

Contact us:
- Phone: 800-ADVENTURE
- Email: help@adventureworks.com
- Fax: 800-555-2424

Nesse exemplo (Adventure Works é uma empresa exemplo da Microsoft para estudo das ferramentas), vemos um relatório bem detalhado sobre uma ordem de venda específica.

- **KPI e Dashboard:** é bem provável que, mesmo sem nunca ter trabalhado com BI, você já tenha tido contato com o termo KPI. O *Key Performance Indicator* (ou Indicador Chave de Performance) é um conceito bastante difundido e muito útil.

  Ele parte do princípio de analisar uma informação sob a comparação de outra informação. Ou seja, estabelece-se um objetivo e se compara o realizado a esse objetivo,

indicando assim se a performance foi boa, média ou ruim.

Por exemplo, podemos dizer que nosso alvo é vender nesse ano 10% a mais do que vendemos ano passado. Temos então uma **meta**. Se eu avaliar as vendas desse ano, poderei dizer se estou na Meta, acima ou abaixo dela. Com isso, posso criar indicadores visuais para determinar a minha situação, os chamados "faróis".

Contudo, como vimos no exemplo do "Indicador de Faturamento", nada melhor do que apresentarmos uma informação em conjunto com outras informações a fim de obtermos o conhecimento! Quando temos diversos KPIs em conjunto, ditando uma correlação significativa entre eles, temos então o *Dashboard* ou *Cockpit*. Não é incomum em um Dashboard, unirmos os KPIs com gráficos e pequenas tabelas de detalhamento. Tudo o que permitir melhor entendimento dos fenômenos é bem-vindo.

Territory	Total Sales	MTD Sales	vsLastYear	Order Qty	MTD Qty	vsLastYear Sales
Australia	$12,668.63	$439,772.12	2600 %	27	1,030	518 %
North Sydney	$2,393.06	$10,686.52		2	22	
Silverwater	$2,322.28	$12,591.86		2	22	
Melbourne	$1,754.98	$15,814.70		2	38	
Findon	$1,735.98	$8,991.62		2	15	
Hervey Bay	$1,700.99	$16,210.03		1	30	
Goulburn	$1,230.46	$22,954.75		1	39	
Melton	$1,120.49	$10,870.30		1	26	
Geelong	$142.97	$9,957.85		4	25	
Sunbury	$128.97	$13,098.81		3	27	
Rockhampton	$74.48	$5,773.11		2	18	
Coffs Harbour	$63.97	$8,722.48		3	51	

- **Tabela dinâmica:** focada na possibilidade do usuário criar suas próprias análises, a tabela dinâmica permite que o usuário escolha quais fatos e quais dimensões serão exibidos e em que ordem, com que regra de

cálculo e com quais filtros. Pode-se facilmente incluir uma dimensão arrastando-a para a tabela, bem como mudá-la de linha para coluna com o mesmo processo de arrastar e soltar (o chamado *slice and dice*).

As informações também podem ser exibidas de forma sumarizada ou detalhada, apenas clicando-se sobre o dado. Essa possibilidade de ir do mais sumarizado para o mais detalhado é o chamado *drill down*. O caminho inverso, do mais detalhado para o mais sumarizado, é o *drill up*. O interessante dessas visões é que, com alguns "cliques", pode-se gerar comparações de dados completamente novas e infinitas combinações de visualizações!

Tipo	Despesa
Valor	Ano

Rótulos de Linha	2011	2012	2013	2014	2015	2016
Supermercado	R$ 5.609,51	R$ 10.843,36	R$ 7.556,10	R$ 74,06	R$ 11.406,48	R$ 1.737,17
Viagem	R$ 4.059,43	R$ 18.808,00	R$ 2.876,53	R$ 125,28	R$ 5.891,34	R$ 2.480,36
Condomínio	R$ 3.996,78	R$ 8.321,73	R$ 10.909,83		R$ 8.831,55	R$ 1.877,87
Previdência-Seguro de Vida		R$ 4.268,88	R$ 13.979,49		R$ 11.060,96	R$ 3.334,73
Refeição	R$ 5.513,65	R$ 8.255,77	R$ 7.567,59	R$ 101,03	R$ 8.022,82	R$ 1.431,16
Vestuário	R$ 4.448,75	R$ 7.732,83	R$ 3.936,10		R$ 5.059,87	R$ 268,00
Estudo	R$ 1.959,89	R$ 1.969,32	R$ 8.699,13		R$ 4.763,54	R$ 520,61
Telefone-TV-Internet	R$ 2.473,99	R$ 2.781,63	R$ 2.286,32		R$ 2.431,88	R$ 388,68
Celular	R$ 2.854,49	R$ 2.232,50	R$ 3.657,13		R$ 628,90	R$ 206,92
Presentes	R$ 3.361,60	R$ 2.198,55	R$ 1.515,52	R$ 850,00	R$ 1.481,02	R$ 40,00
Combustível	R$ 3.008,96	R$ 1.241,02	R$ 1.684,22		R$ 1.995,20	R$ 450,00
Lazer	R$ 1.328,85	R$ 1.955,68	R$ 1.124,99		R$ 1.542,30	R$ 158,13
Assinatura de Periódicos	R$ 593,46	R$ 486,20	R$ 421,89		R$ 1.539,87	
Diversas	R$ 154,13					
Total Geral	R$ 41.370,32	R$ 70.552,82	R$ 66.204,85	R$ 700,47	R$ 64.655,19	R$ 12.917,63

Conforme a figura, vemos uma tabela dinâmica do Excel sendo apresentada em conjunto com um gráfico dinâmico. Na implementação que faremos adiante, veremos como criar essas estruturas de forma bastante simples.

Decision Tree e ações

Academicamente, o BI não responde por efetivamente tomar ações. Ele entrega a informação certa para que melhores ações sejam tomadas.

Contudo, temos apresentações de dados contextualizados. Temos os KPIs, por exemplo, que indicam se uma situação é boa ou ruim por meio de comparações entre dados realizados e metas estabelecidas. Se a plataforma é capaz de definir se algo é bom ou ruim mediante parâmetros estabelecidos, podemos criar ações baseadas nessas situações a fim de agilizar procedimentos padronizados.

Por exemplo, tem-se em SRM (*Social Relationship Management*), a coleta de informações de comentários de clientes nas redes sociais acerca de sua marca. Pode-se contextualizar esses comentários como positivos, negativos, entusiastas etc. Uma vez categorizado, pode-se ter rotinas que executam a inclusão de respostas automáticas a comentários negativos afim de se diminuir o impacto deles nas redes sociais. Ou ainda, em medições geotécnicas, pode-se estabelecer que o BI envie alertas quando uma determinada medição indicar uma movimentação de terreno maior do que a prevista, e assim por diante.

2.2 A TECNOLOGIA ESCOLHIDA E A MONTAGEM DO AMBIENTE

Se avaliarmos o mercado, veremos soluções gratuitas de Business Intelligence de diversos fornecedores, para diversos fins. Desde soluções nativas gratuitas, até BI como serviço, passando por versões gratuitas de ferramentas pagas.

Atualmente, o mercado é bem farto de opções, o que, para quem vai iniciar uma empreitada de escolha, pode ocasionar um grande trabalho! Trabalho bom, claro, pois ter opções é sempre melhor do que não as ter.

Mas o fato de iniciar a implantação da plataforma de Business Intelligence em sua empresa com uma solução não licenciada torna

tudo mais fácil! O ponto é determinar qual ferramenta. Para avaliar com quais soluções poderíamos trabalhar, levei em consideração os seguintes pontos:

- **Se são gratuitas**, ou seja, não requerem licenciamento. Todo o budget que você tiver para essa iniciativa será revertido em entregáveis, em valor agregado.
- **Se são padrão de mercado**, isto é, havendo a necessidade de ajuda, os profissionais de mercado aptos a trabalharem com elas são facilmente localizados e, por isso mesmo, não são muito caros.
- **Se são escaláveis para soluções pagas**, ou seja, na ocasião do amadurecimento dos demandantes, a ferramenta gratuita pode ser facilmente migrada para sua versão paga e completa, possibilitando o crescimento sem a necessidade de "refatoração". Além disso, os componentes expostos aos usuários continuam sendo os mesmos, isto é, a versão paga da ferramenta não gera necessidade de novos treinamentos.

Dessa forma, optei pelo **Microsoft SQL Server 2014 Express**. Dentre as opções gratuitas, com certeza não é a que possui mais recursos. O Pentaho, por exemplo, é bem mais completo. Mas, se sua plataforma for evoluir em tamanho, funcionalidades e quantidade de usuários, as soluções de BI da Microsoft estão hoje entre as mais interessantes do mercado.

Então, nada melhor do que começar com a versão gratuita de algo que pode ser migrado para uma versão completa e, assim sendo, fará frente a qualquer outra ferramenta como IBM, SAS, SAP etc. Além disso, e muito importante, é o fato de que SQL Server é padrão de mercado. Mais do que qualquer outro SGDBR, inclusive

Oracle, o SQL está difundido nas empresas, e isso cria uma vasta oferta de mão de obra.

Notamos pela dispersão (fornecida pelo Gartner), que a Microsoft, com seu SQL Server, se posiciona como o extremo do chamado "quadrante mágico". Ou seja, é a solução mais completa e difundida no mercado até agora.

Partindo da escolha do SQL Server, temos de avaliar seus pré-requisitos para a montagem do ambiente:

- **Sistema operacional compatível:** Windows 7 Service Pack 1; Windows 8; Windows 8.1; Windows 10;

Windows Server 2008 R2; Windows Server 2008 R2 SP1; Windows Server 2012; Windows Server 2012 R2.

- **Processador:** processador compatível com Intel, com velocidade mínima de 1 GHz ou um processador mais rápido.
- **RAM:** mínimo de 512 MB para o SQL Server Express com Ferramentas, e o SQL Server Express com Advanced Services e 4 GB para o Reporting Services instalado com o SQL Server Express, com Advanced Services.
- **Espaço em disco rígido:** 4,2 GB de espaço em disco.

Parto do princípio que, pelo menos, um servidor Windows está rodando em sua empresa e que poderemos usá-lo para essa empreitada — lembrando de que esse servidor não pode ser um controlador de domínio. SQL Express não pode ser instalado no seu AD Server. Se não houver nenhum, avalie a possibilidade de aquisição.

Se não for possível comprar um *server*, veja que o SQL Express pode ser instalado em versões desktop do Windows que, obviamente, não são nada recomendadas para fazer de um servidor às vezes. Mas, claro, que se nada mais for possível, melhor implementar sua plataforma de BI em um desktop, e depois solicitar o budget do servidor quando os usuários começarem a reclamar de lentidão.

Disclaimer

Se sua empresa não possuir nenhum servidor, avalie também a aquisição de soluções em *cloud*, tanto para Servers quanto para seu BI propriamente dito. Claro que essas opções são pagas, mas o TCO de ambientes e soluções em Cloud tem se mostrado menores do que seus similares físicos.

No nosso exemplo, usaremos uma Máquina Virtual com Windows Server 2012 R2. Mesmo que você use versões anteriores do SO para o SQL Server 2014, tenha certeza de ter instalado o .Net framework 3.5 SP1. Ele é pré-requisito para a instalação que faremos a seguir.

Resolvido onde instalaremos o SQL, temos agora de obtê-lo! Para isso, basta entrar no site do produto (https://www.microsoft.com/pt-br/download/details.aspx?id=42299) e efetuar o download, precisando de um breve cadastro para isso.

Microsoft

Sign out

Profile Center

Home My profile Manage communications Help

SQL Server 2014 Express

Thank you for taking the time to fill out the following online form. If you do not want to submit your information, click **Cancel**.

* indicates a required field

*** My name (personal information)**

* First name Ronaldo

* Last name Braghittoni

*** My email address**

braghittoni@outlook.com

*** Please select the version of SQL Server 2014 Express you'd like to download**

- ○ SQL Server 2014 LocalDB Express 32bit
- ○ SQL Server 2014 LocalDB Express 64 Bit
- ○ SQL Server 2014 Express 32 Bit
- ○ SQL Server 2014 Express 64 Bit
- ○ SQL Server 2014 Express with Tools 32 Bit
- ○ SQL Server 2014 Express with Tools 64 Bit
- ○ SQL Server 2014 Management Studio Express 32 Bit
- ○ SQL Server 2014 Management Studio Express 64 Bit
- ○ SQL Server 2014 Express with Advanced Services 32 Bit
- ◉ SQL Server 2014 Express with Advanced Services 64 Bit

* **Please select the language of your SQL Server 2014 Express w/Advanced Servies 64 Bit download**

- ○ Chinese (Simplified)
- ○ Chinese (Traditional)
- ○ English
- ○ French
- ○ German
- ○ Italian
- ○ Japanese
- ○ Korean
- ◉ Portuguese (Brazil)
- ○ Russian
- ○ Spanish

* **Country/Location**

Brazil

To subscribe, select the communication(s) below and your preferred delivery format (HTML or Text).

Subscribe	Format HTML/Text	Communication Description
☐	◉ ○	⊞ **TechNet Flash (US, XX)**

Communication preferences

Choose how Microsoft may use your contact information. Please note: these settings may not reflect your current permission settings in our systems. Review your settings.

I would like to hear from Microsoft about products, services, and events, including the latest solutions, tips, and exclusive offers.

- ☑ My email address
- ☐ Business phone number

I would like to hear from Microsoft Partners, or Microsoft on their behalf, about their products, services, and events. Share or use my details with Microsoft Partners.

- ☑ My email address
- ☐ Business phone number

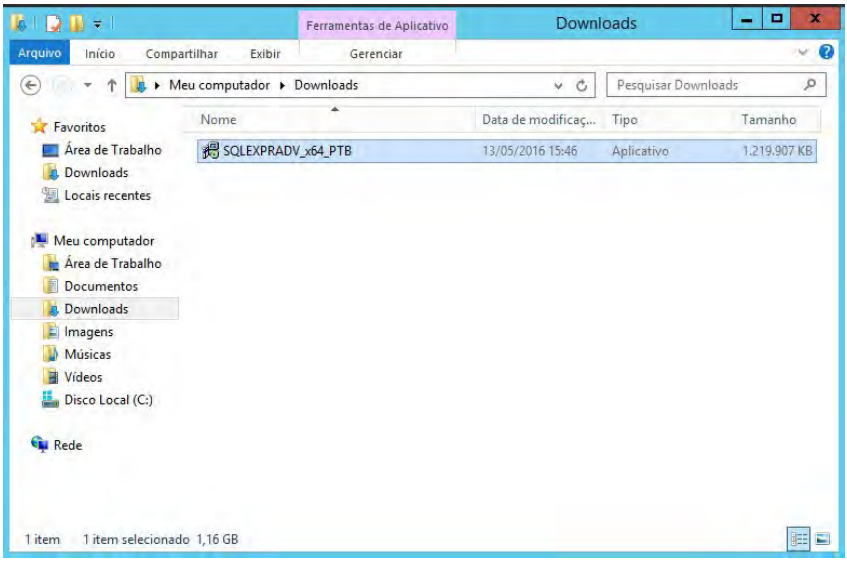
Note: These settings will not affect other newsletters or mandatory service communications from Microsoft. To learn how to set your contact preferences for other Microsoft sites, read the privacy statement.

By Downloading SQL Server 2014 Express software, you may receive emails from Microsoft with SQL Server 2014 Express resources.

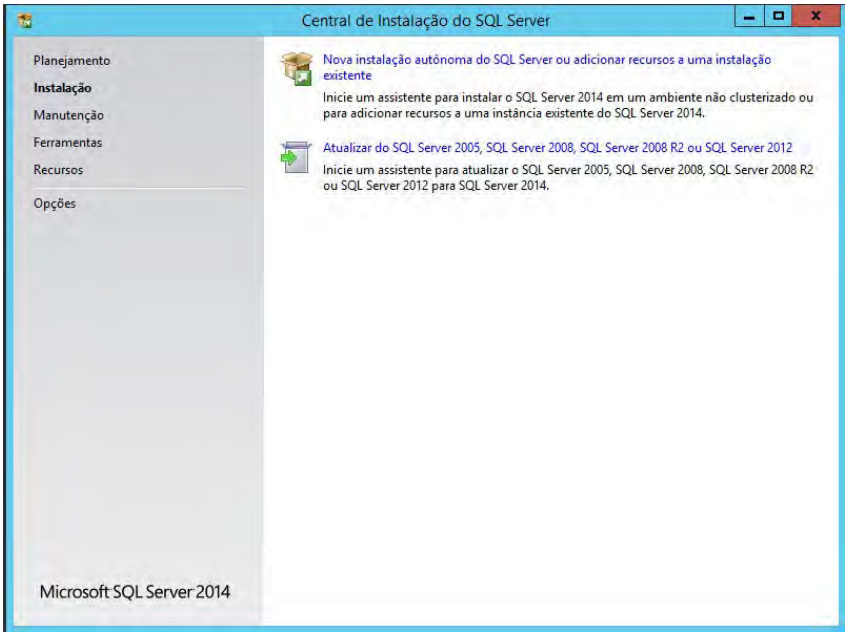
Continue Cancel

Selecione a última versão disponível, e opte por `...With Advanced Services`. A opção por 32 ou 64 bits dependerá do

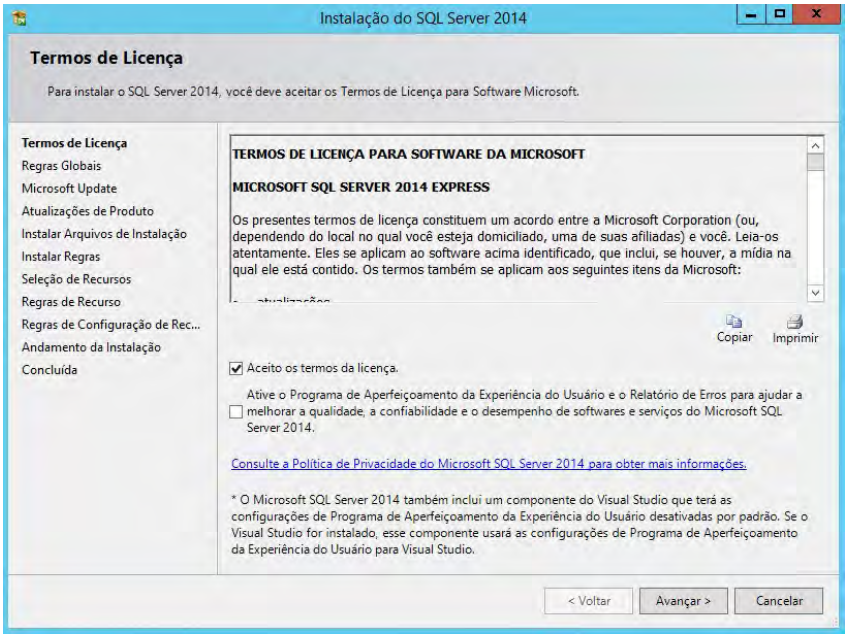
Sistema Operacional do seu Servidor. Quanto à língua, recomendo usar a versão de sua língua local, a menos que sua empresa trabalhe com escritórios em outros países — quando então a língua padrão da corporação pode ser a melhor escolha.



A versão 2014 do SQL Server Express With Advanced Tools vem em um arquivo compactado de 1GB de tamanho. Clicando duas vezes no arquivo, ele iniciará a descompactação e o instalador:

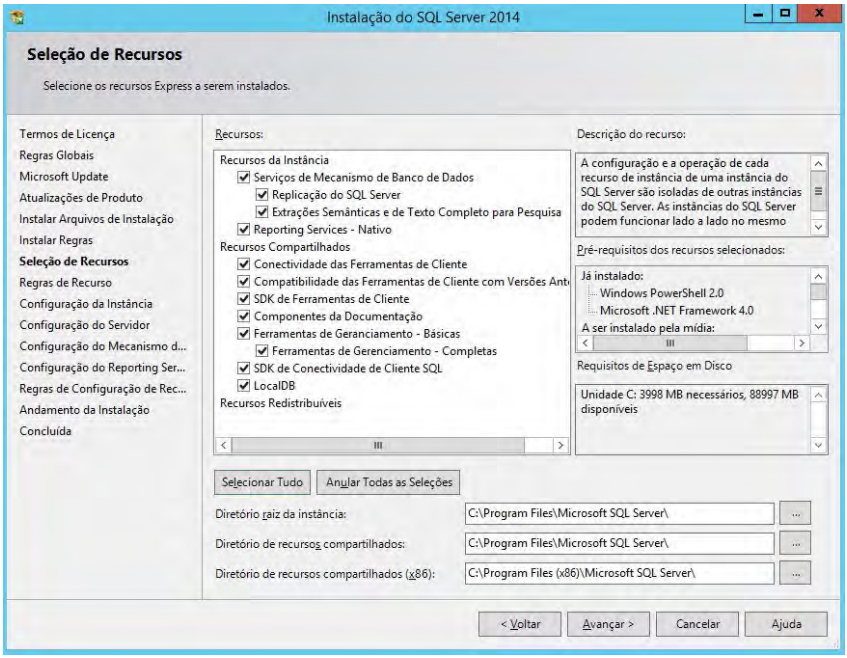


Selecione a opção de *Nova instalação autônoma*, e depois aceite os termos da licença e clique em `Avançar` :

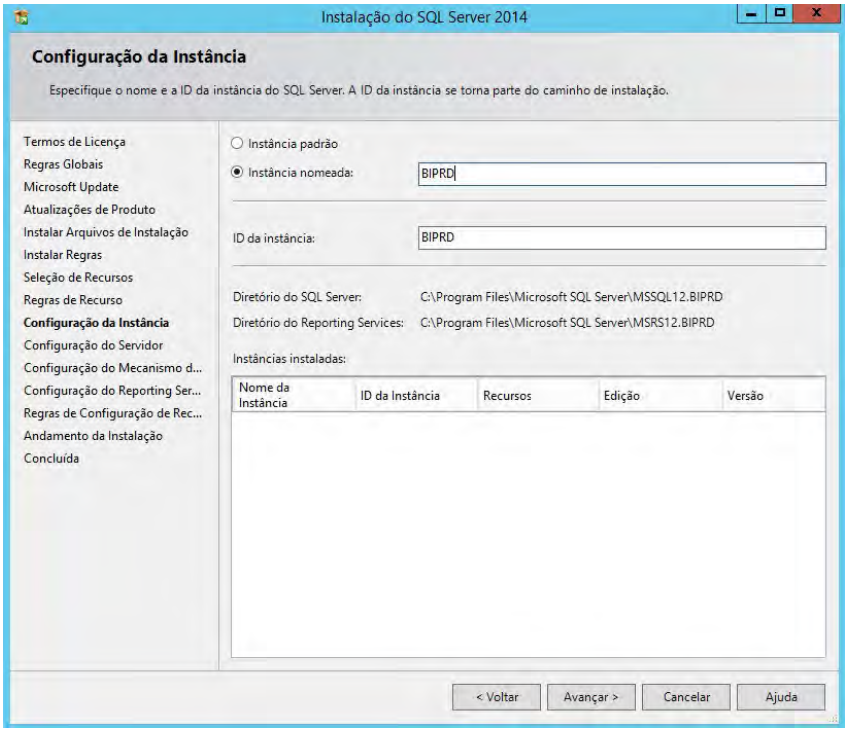


Depois o instalador abrirá a `Seleção de Recursos`. Selecione todos os recursos, inclusive as ferramentas administrativas. Em um ambiente de alta performance, é recomendado instalar somente os serviços, sem as ferramentas administrativas e somente os serviços estritamente necessários (no nosso caso, a replicação não seria necessária, por exemplo).

Contudo, vamos simplificar a instalação, selecionando as opções padrão. Para mais detalhes da instalação do SQL Server, procure a documentação do fabricante, em https://www.microsoft.com/pt-br/server-cloud/products/sql-server/features.aspx/Overview.aspx/?WT.srch=1&WT.mc_ID=N1ZzdQIx.

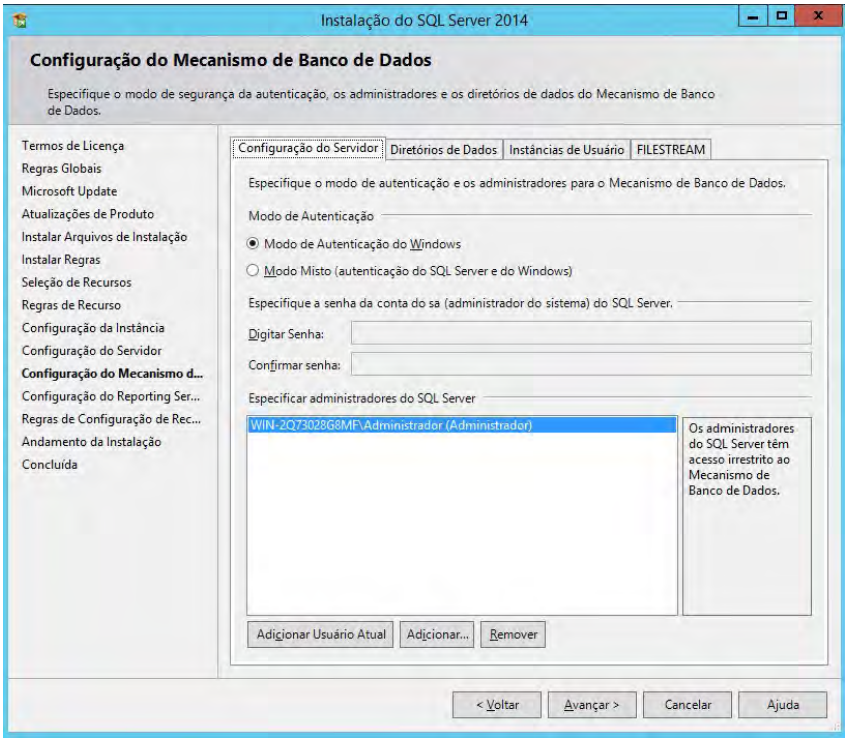


Estando todos os pré-requisitos instalados, o SQL vai solicitar que se dê um nome à instância. Por padrão, ele sugere `SQLEXRESS`. Mas como se trata de uma instância que hospedará um ambiente de BI, um nome mais significativo pode ser atribuído, como `BIPRD`, por exemplo (remetendo a ser o ambiente de produção do nosso BI). Se sua empresa já possuir um padrão de nomenclatura de instâncias, não haverá nenhum problema em segui-lo.

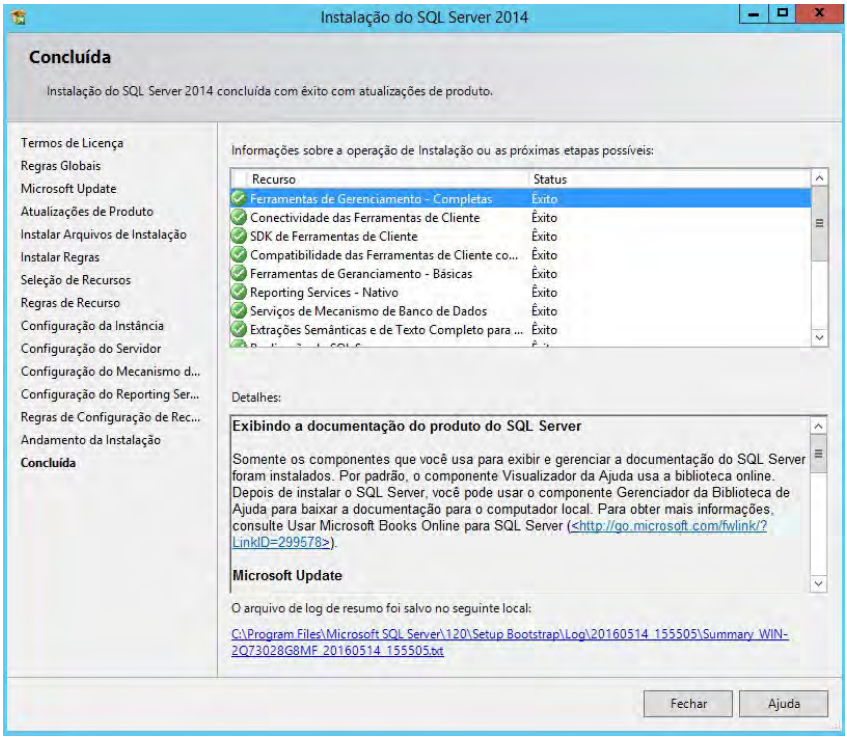


Mantenha as opções padrão para `Configuração do Servidor` e clique em `Avançar`. Depois, o instalador apresentará a `Configuração de Mecanismo de Banco de Dados`. Na aba de `Autenticação`, garanta ter adicionado o usuário atual.

A aba de `Diretório de Dados` é onde podemos alterar o padrão para criação dos arquivos de dados do SQL (`.mdf` , `.lfd`) em discos que não o `C:\`. Existe um conjunto de melhores práticas que rogam que, a grosso modo, é interessante que os arquivos de dados fiquem em uma unidade diferente da unidade de sistema, e que os arquivos de log fiquem em uma terceira unidade. Em nosso exemplo, vou efetuar a instalação padrão, com todos os arquivos em `C:\`.

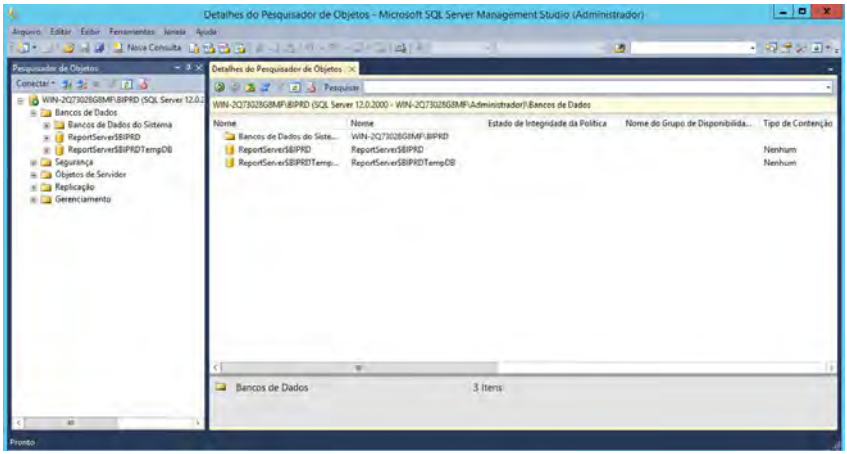


Clique em `Avançar` e deixe marcada a opção `instalar` e `configurar` do `Reporting Services`. Clique em `Avançar` novamente e o processo de instalação será iniciado! Estando tudo certo, com a instalação finalizada, deverá apresentar um checklist de todos os pontos com `Êxito`:



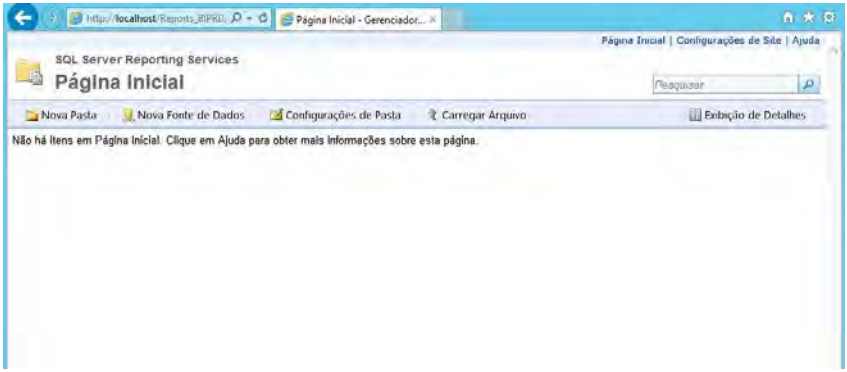
E os componentes do SQL passam a ser exibidos na lista de aplicativos instalados (nas versões anteriores, busque no `Menu Iniciar`):

Inicie o SQL Server Management Studio:



Não vamos nos delongar no uso do Management Studio, mas para avaliar se a instalação ocorreu com sucesso, basta acessar a instância que criamos (o `BIPRD`) e expandir os bancos de dados. Os bancos de dados do Reporting Services devem estar criados.

Abra o navegador na máquina onde foi feita a instalação e digite `http://localhost/Reports_BIPRD/Pages/Folder.aspx` . Se estiver tudo em ordem, a página a seguir deve ser exibida:



Caso haja algum problema em alguma parte da instalação, procure soluções no site da Microsoft e com a comunidade pelo Technet (http://social.technet.microsoft.com/wiki/contents/articles/23878.installing-sql-server-2014-step-by-step-tutorial.aspx).

2.3 CONCLUSÃO

Neste capítulo, entendemos a arquitetura do Business Intelligence e seus componentes. Esse entendimento é fundamental para prosseguirmos na criação de cada uma das partes, o que faremos nos próximos capítulos. Configuramos todo o ambiente necessário e podemos partir para a implementação!

CAPÍTULO 3

O DESENHO DO DATA WAREHOUSE

3.1 O CENÁRIO

Para criarmos uma plataforma de Business Intelligence com fins didáticos, vamos estabelecer um cenário, ou seja, uma situação na qual implantaremos a nossa plataforma de BI.

Uma solução nacional de *Business Intelligence as a Service*, a gratuita OpenBI (http://openbi.com.br), tem como premissa que só são necessários os dados de notas fiscais (NFs) emitidas para se conseguir uma gigantesca gama de informações e indicadores. Além disso, dado a obrigatoriedade das notas fiscais eletrônicas, praticamente todas as empresas são capazes de gerar arquivos contendo os dados oriundos de suas emissões.

Utilizemos essas mesmas premissas e vamos implementar uma plataforma de BI, tendo como informações iniciais os dados de notas fiscais emitidas. O interessante dessa abordagem é que, com certeza, será possível usar esse exemplo para a implantação real do BI em seu ambiente que, posteriormente, poderá crescer para ter cada vez mais e mais origens, e poder fornecer mais e mais informações.

Entendendo a estrutura de informações

Quando mencionamos dados de notas fiscais, estamos nos referindo a documentos que comprovam a compra de um serviço ou produto e que podem variar de formato de acordo com a região. Contudo, todas possuem algumas informações padrão e será com essas informações que trabalharemos:

- **Cabeçalho:** número e série da Nota Fiscal, data de emissão e dados da empresa que emitiu (importante no caso do BI trabalhar com múltiplas empresas).
- **Dados do Cliente:** nome/razão social, endereço, dados de contato, como telefone, e-mail, CPF/CNPJ.
- **Detalhamento:** nome e código dos produtos ou serviços, quantidades, preço unitário (às vezes) e preço total.
- **Rodapé:** valor total da nota, valor de impostos, valor de frete, dados de transportadora, forma de pagamento (às vezes), data de vencimento.

Pode não parecer muita informação para se fazer análises de fenômenos, mas acredite: quando você disponibiliza essas informações de forma a possibilitar diversos cruzamentos de dados e comparativos históricos, passa a contar com uma poderosa ferramenta. Na referida OpenBI, por exemplo, esses dados já possibilitam análises como:

- Evolução da receita mensal com comparativo ao período anterior e a uma meta;
- Análise de venda por produtos, com tendências de demanda e sazonalidade;
- Participação percentual dos produtos na receita total do período;
- Vendas por clientes com gráficos de evolução histórica;
- Consulta detalhada de NF com filtros de data, cliente, NrNF, produto etc.;

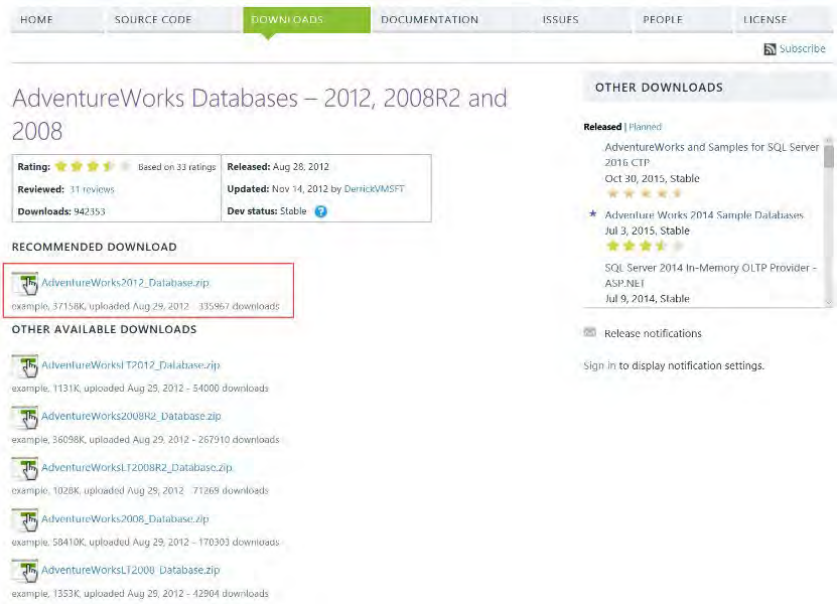
- Distribuição geográfica das vendas;
- Dashboard do cliente (para força de vendas) com dados de compras mês a mês, representatividade no semestre, histórico de relação com os produtos e detalhamento de NFs emitidas.

Então, para atender ao nosso cenário, vamos trabalhar com as informações mínimas. Quando você for efetuar a sua implantação, avalie sempre a possibilidade de incluir mais dados a partir do que estiver disponível na sua origem. Como veremos no desenho do nosso Data Warehouse, pode-se adicionar informações às dimensões e aos fatos conforme elas estiverem disponíveis.

Obtendo e preparando os dados de exemplo

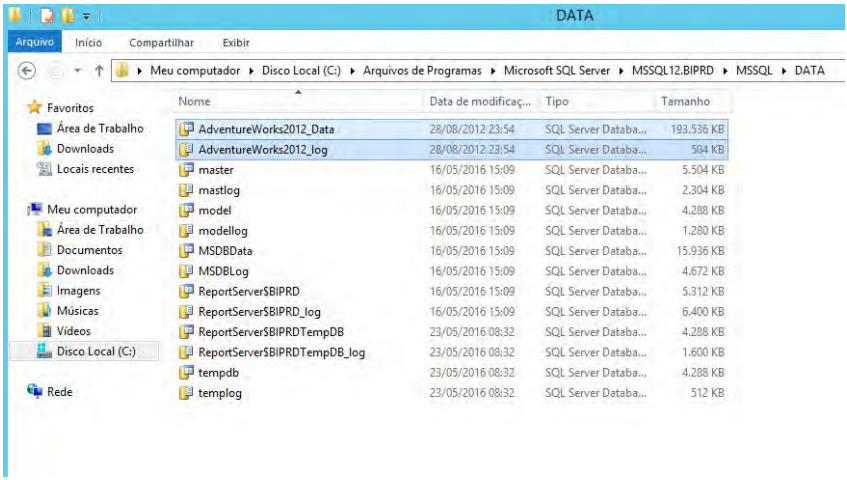
Se você tem acesso a dados reais, pode avançar para o próximo tópico! Caso não, podemos obter facilmente com a Codeplex, um hub de soluções open source, que mantém uma área na qual é possível encontrar o AdventureWorks para download de diversas versões de SQL (2008, 200 R2, 2012 etc.).

Qualquer versão igual ou anterior ao de nosso ambiente (SQL 2014) seria compatível. Faremos então o download da versão 2012. Basta clicar no link http://msftdbprodsamples.codeplex.com/releases/view/93587, e selecionar a base para SQL 2012:

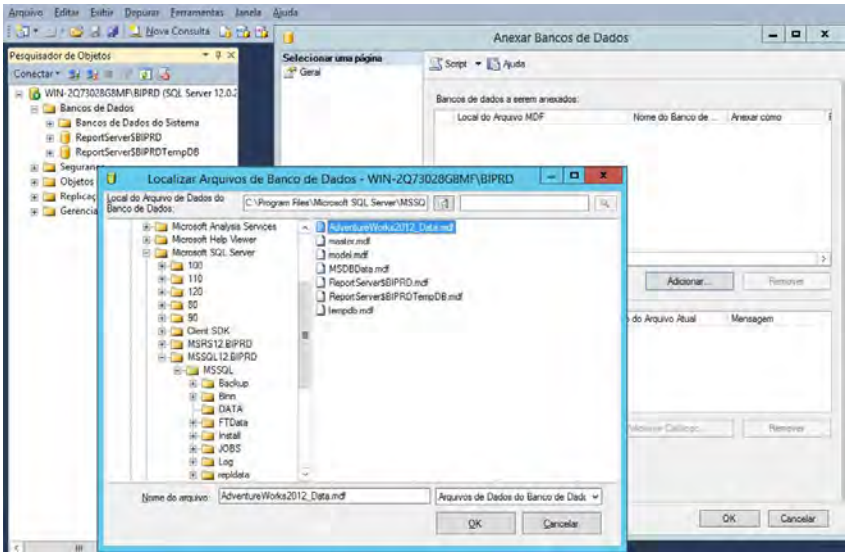


O download será de um ZIP com dois arquivos: `AdventureWorks2012_Data.mdf` e `AdventureWorks2012_log.ldf` . Esse é o padrão de arquivos do SQL, um para dados (`.mdf`) e outro para log de transações (`.ldf`).

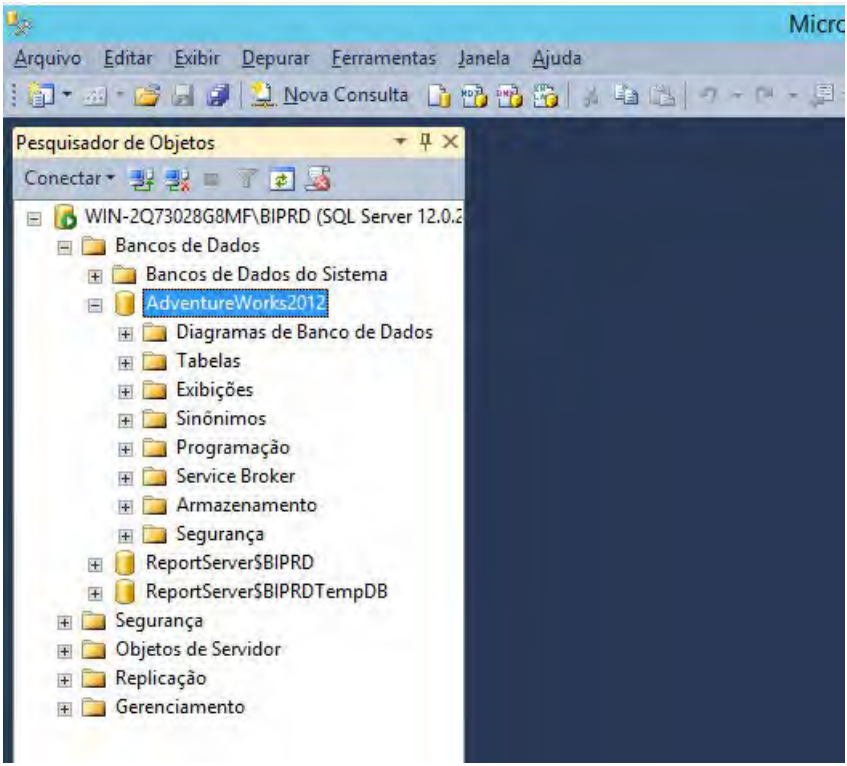
Copie esses arquivos para o caminho de dados do SQL. No nosso exemplo, foi o diretório padrão `C:\Program Files\Microsoft SQL Server\MSSQL12.BIPRD\MSSQL\DATA\` :



Uma vez copiados, abra o Management Studio, conecte na instância `PRDBI`, clique com o botão direito sobre `Banco de Dados` e comande `Anexar...` (*Attach Database*). Na janela que se abrirá, selecione `Adicionar...` e, em seguida, marque o arquivo do AdventureWorks (apenas os arquivos de dados serão exibidos):



Clicando em `ok`, o SQL procederá a inclusão dessa base ao gerenciamento da instância e ela aparecerá no seu explorador da instância:



Agora vamos executar um comando para salvar os dados como se fossem oriundos de um sistema de faturamento (ou ERP) que nos seria disponibilizado para a importação na nossa plataforma de BI:

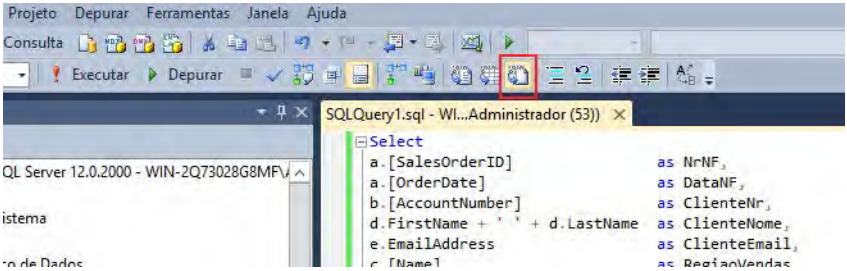
No Management Studio, logo acima do `Pesquisador de Objetos`, existe o botão `Nova Consulta`. Clique nele e copie o comando a seguir:

```
Select
cast(a.[SalesOrderID] as varchar(100))                +'|'+
convert(varchar,a.[OrderDate],102)                    +'|'+
cast(b.[AccountNumber] as varchar(100))                +'|'+
```

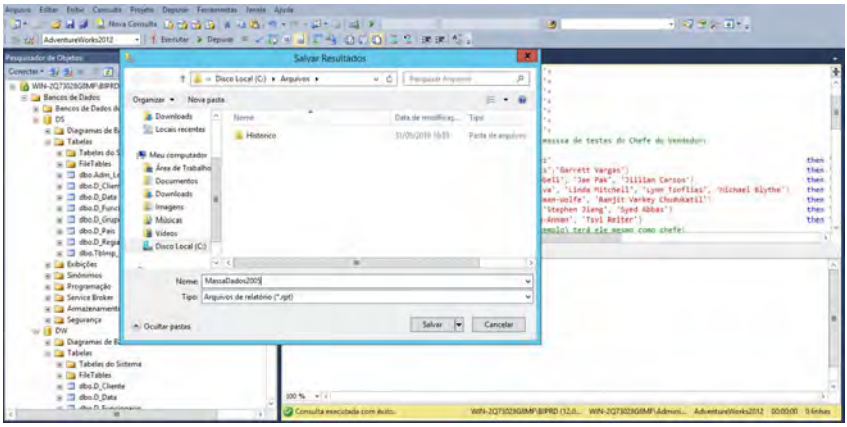
```
cast(d.FirstName + ' ' + d.LastName as varchar(100))+'|'+
cast(e.EmailAddress as varchar(100))                +'|'+
cast(c.[Name] as varchar(100))                        +'|'+
cast(c.[CountryRegionCode] as varchar(100))            +'|'+
cast(c.[Group] as varchar(100))                        +'|'+
Isnull(f.[LoginID], 'Sistema')                        +'|'+
Isnull(g.FirstName + ' ' + g.LastName, 'Sistema')    +'|'+
-- os dados a seguir são inventados para criar a nossa masssa de t
estes do Chefe do Vendedor:
case
    when g.FirstName + ' ' + g.LastName = 'Rachel Valdez'
                                                then '' -- Ela
é a Chefe!
    when g.FirstName + ' ' + g.LastName in ('Amy Alberts','Garrett
 Vargas')                                        then 'Rachel
Valdez' -- Eles são o Segundo nivel.
    when g.FirstName + ' ' + g.LastName in ('David Campbell', 'Jae
 Pak', 'Jillian Carson')                         then 'Amy Albe
rts' -- Esses são funcionarios da Amy.
    when g.FirstName + ' ' + g.LastName in ('José Saraiva', 'Linda
 Mitchell', 'Lynn Tsoflias', 'Michael Blythe')    then 'Garrett Va
rgas' -- Esses são funcionarios do Garrett.
    when g.FirstName + ' ' + g.LastName in ('Pamela Ansman-Wolfe',
'Ranjit Varkey Chudukatil')                     then 'Jillian C
arson' -- Esses são funcionarios da Jillian.
    when g.FirstName + ' ' + g.LastName in ('Shu Ito', 'Stephen Ji
ang', 'Syed Abbas')                             then 'Michael B
lythe' -- Esses são funcionarios do Michael.
    when g.FirstName + ' ' + g.LastName in ('Tete Mensa-Annan', 'T
svi Reiter')                                     then 'Stephen
Jiang' -- Esses são funcionarios do Stephen.
    Else 'Sistema'    -- O Sistema (venda on-line, por exemplo) te
rá ele mesmo como chefe!
    End                                                +'|'+
isnull(i.ProductNumber,'00000')                        +'|'+
isnull(i.Name,'-')                                    +'|'+
isnull(i.[Size],'-')                                +'|'+
isnull(i.[ProductLine],'-')                            +'|'+
isnull(i.[Color],'-')                                +'|'+
cast(a.[SubTotal] as varchar(100))                    +'|'+
cast(a.[TaxAmt] as varchar(100))                    +'|'+
cast(a.[Freight] as varchar(100))                    +'|'+
cast(h.UnitPrice     as varchar(100))                +'|'+
cast(h.OrderQty     as varchar(100))                    as Texto

from [Sales].[SalesOrderHeader] as a inner join [Sales].[Customer]
as b
on a.CustomerID = b.CustomerID
```

```
inner join [Sales].[SalesTerritory] as c
on a.TerritoryID = c.TerritoryID
inner join [Person].[Person] as d
on b.PersonID = d.BusinessEntityID
inner join [Person].[EmailAddress]as e
on d.BusinessEntityID = e.BusinessEntityID
left join [HumanResources].[Employee] as f
on a.SalesPersonID = f.BusinessEntityID
left join [Person].[Person] as g
on f.BusinessEntityID = g.BusinessEntityID
inner join [Sales].[SalesOrderDetail] as h
on a.SalesOrderID = h.SalesOrderID
inner join [Production].[Product] as i
on h.ProductID = i.ProductID
where year([OrderDate]) = 2005
```

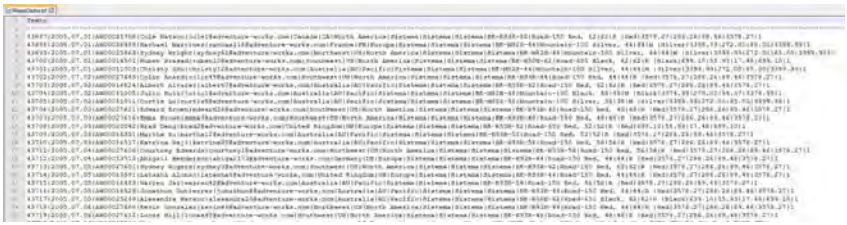
Para a extração dos dados em arquivo, clique em `Resultados em Arquivo`, destacado em vermelho na imagem seguinte, e comande `Executar` (a exclamação em vermelho, logo ao lado esquerdo do botão que acabamos de clicar):



Você será perguntado quanto a um endereço de destino para esse arquivo. Clique em `C:\` na janela e, com o botão direito, crie uma nova pasta chamada `Arquivos`:



Dê o nome de `MassaDados2005` ao arquivo e clique em `Abrir`. O SQL executará a consulta, criando um arquivo de resultados com pouco mais de 1 MB de dados (afinal, são 5.151 linhas geradas nessa nossa massa de testes) que deve estar com o seguinte layout:



Para visualizar o arquivo `.rpt`, talvez você tenha de selecionar o programa padrão. O bloco de notas serve tranquilamente. Eu prefiro o Notepad++, um editor de texto gratuito e com diversos recursos! Lembrando: para nosso processo, a visualização do arquivo não é necessária, mas é sempre bom ter em mãos uma ferramenta para entendermos as fontes de dados se algum problema aparecer.

Até agora temos trabalhado para preparar o nosso ambiente transacional para enviar dados para a Plataforma de BI que, por

enquanto, nem começamos a criar.

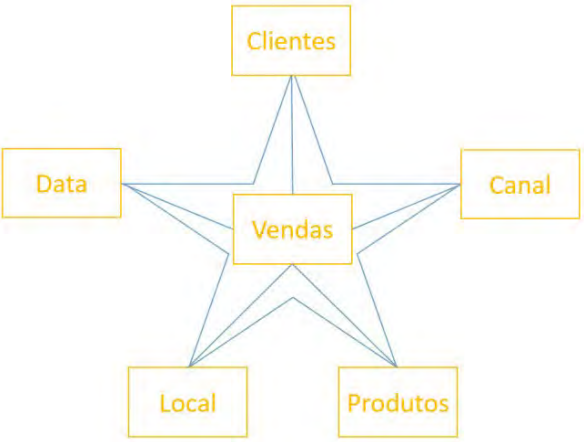
Mas tendo finalizado essa frente e entendido como está estruturada a nossa origem de dados, vamos desenhar o DW que irá conter as informações das Notas Fiscais e, assim, possibilitar que iniciemos a nossa plataforma!

Um detalhe é que limitamos nossa massa de dados a um único ano, dos 04 anos disponíveis nessa massa. Isso porque vamos criar todo nosso ambiente usando essa porção dos dados e, depois, iremos criar o arquivo ano a ano, simulando cargas diárias a fim de validarmos se os processos incrementais estarão funcionando corretamente.

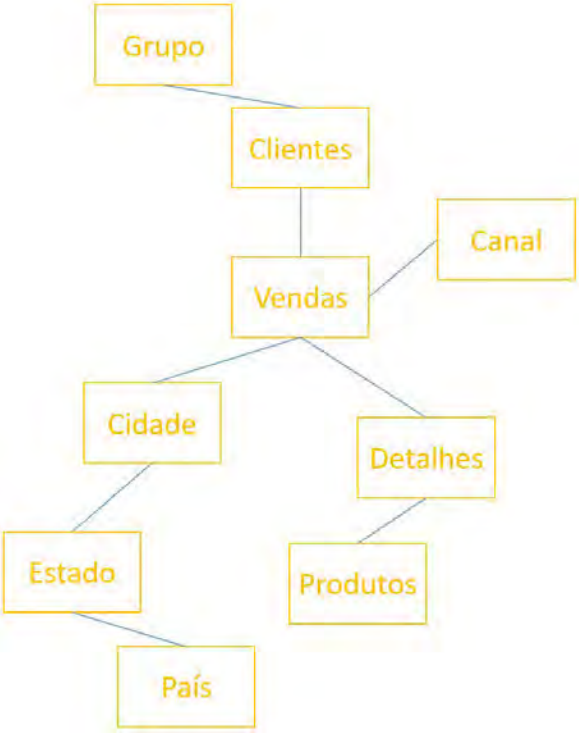
3.2 O DATA WAREHOUSE

Para implantar o Data Warehouse, existem algumas regras de design e algumas boas práticas. Entre as regras, encontram-se os postulados de Imnon, que vimos anteriormente, e também o chamado *esquema*. Já vimos que o DW será baseado em um desenho dimensional (fatos e dimensões), mas esse desenho pode ter dois esquemas: Esquema Estrela ou esquema Floco de Neve.

No esquema Estrela (*Star Schema*), as tabelas `Dimensão` são diretamente relacionadas com a tabela `Fato`. Não existe normalização das informações e as consultas são respondidas da maneira mais rápida possível:



Já no esquema Floco de Neve (*Snow Flake Schema*), as dimensões podem possuir algum nível de normalização, gerando relacionamentos entre as tabelas das dimensões:



A escolha por um esquema ou por outro estará relacionada ao trabalho (e, consequentemente, custo) de implantação. O modelo mais performático e que devemos buscar sempre é o Estrela. Mas às vezes a informação é conhecidamente estruturada de forma que a desnormalização causará um grande trabalho e, efetivamente, trará um ganho de performance bastante baixo. Um exemplo é a Geografia. A relação de País, Estado e Cidade é sempre bastante conhecida e pouquíssimo mutável.

Outro ponto nessa escolha é o limite de colunas para a chave primária. No caso do SQL (https://msdn.microsoft.com/en-us/library/ms143432(v=sql.120).aspx), temos um total de 16 colunas para usarmos na chave da Fato. Assim, se tivermos muitas dimensões ligadas a Fato, usar o esquema Floco de Neve "economizará" colunas para a chave primária.

Exemplificando, no caso de País, Estado e Cidade, se tivermos um esquema Estrela, cada uma das tabelas estaria ligada a Fato, consumindo três colunas da chave primaria. Se a tabela de País estiver ligada à de Estado, essa à de Cidade e somente essa última à Fato, teremos apenas uma coluna usada na chave para mantermos a mesma informação.

Quanto às boas práticas, elas podem variar de profissional para profissional, de escola para escola e, eventualmente, podem levantar discussões calorosas. Não é meu objetivo postular certo e errado, então apenas mencionarei o que tenho feito ao longo dos anos e que, na minha experiência profissional, tem se mostrado bastante adequado.

Uma boa prática é a de diferenciar as tabelas do seu DW pela especialização dela. Ou seja, se é uma tabela de `Dimensão`, vamos nomeá-la como `D_NomeDaTabela`. Se ela for uma `Fato`, que seja `F_NomeDaTabela`.

Um outro ponto é fugir da chamada "notação Húngara", apelido dado à notação de Charles Simonyi para as variáveis de sistema criadas a partir de abreviações. Nela, você lê o nome do objeto e não faz a menor ideia do que ele faz ou para que ele serve! Então dê nomes significativos às tabelas e às colunas. Evite dar nomes abreviados, formados por siglas e afins.

É normal encontrar regras mirabolantes de nomenclaturas nas empresas. Mas na prática, cria-se um padrão que não serve efetivamente para nada, e que mais atrapalham do que ajudam. Então, vamos criar nomes simples e significativos.

A origem do dado, muitas vezes, deve ser informada mesmo que somente para uso no processo de desenvolvimento ou de auditoria, sem ser exibido para o usuário final. O processo de se rastrear de onde veio e quando um dado foi carregado no DW possui o nome de "data lineage" (que também é uma metodologia com seus gráficos e conceitos sobre acompanhamento de dados através de um processo).

Na prática do BI, consiste em manter uma coluna de data da carga e de origem em cada uma das tabelas do DW. Assim, ao visitar um registro, sabe-se quando ele entrou e de onde ele veio.

Evite criar seu DW antes de tê-lo desenhado primeiramente em alguma ferramenta. A visão do todo que um desenho lhe proporciona com certeza reduz a chance de erros. Eu costumo usar o Enterprise Architect, mas outras soluções existem, como a Model Right e a DeZign.

Feitas as considerações, vamos à implantação.

Dimensões

Entre as boas práticas que mencionei, está também a de iniciar o

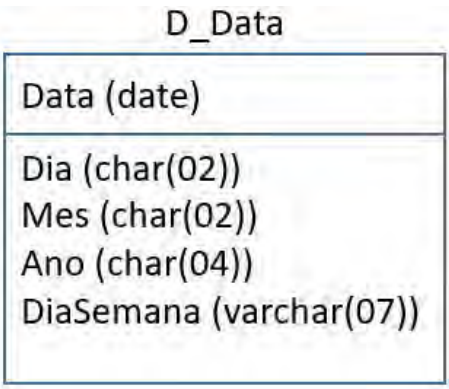
desenho do DW (ou da expansão dele) sempre pelas novas dimensões. Existe um motivo prático para isso: como as dimensões são as tabelas da Chave Primária, elas precisam existir antes de criarmos as Fatos que hospedarão as Chaves Estrangeiras. Se essa frase soou como grego, basta entender que para um registro existir dentro de uma tabela Fato, ele precisa antes (e obrigatoriamente antes) ter sido carregado na tabela Dimensão. Assim sendo, começar pelo desenho da Dimensão, que não precisa de uma outra entidade para existir, é sempre mais simples.

Tempo (ou Data)

Como postulado por Inmon, o DW é sempre variável com o tempo, ou seja, a dimensão `DATA` deve invariavelmente existir. Se é essa a regra, então nada mais lógico do que começar nosso DW por ela. E ela será composta por uma Chave Primária que pode ser a própria Data e colunas de especialização, como Dia, Mês e Ano separados, Dia da Semana, Ano Fiscal, Trimestre etc.

Pode-se ter as informações de especialização que forem necessárias. Essas informações poderiam muito bem ser calculadas em tempo de exibição, mas procura-se armazená-las justamente para que o processo de seleção dos dados e respectiva apresentação sejam mais rápidos. Além disso, quando formos acoplar uma solução de banco de dados multidimensional (os cubos), a existência das informações na Dimensão facilitará bastante a criação dos objetos.

Dessa forma, criei a tabela chamada `D_Data` com a seguinte estrutura:



Por prática, não costumo inserir o "data lineage" na dimensão temporal. Mostrou-se pouco útil. Mas, fique à vontade caso queira ser mais puritano do que eu!

Cliente

Uma dimensão que encontramos na maioria dos DW é a Cliente. Se vamos vender algo, provavelmente vamos vender para alguém! E, como na Tempo, a ideia é criar uma chave que vai identificar esse cliente nas Fatos e ir acrescentando demais informações.

Essa chave é geralmente um "indicador artificial", ou seja, um código que vamos estabelecer arbitrariamente para cada registro a fim de, principalmente, garantir que as buscas por ele sejam as mais rápidas possíveis. A essa chave artificial damos um nome: *Surrogate Key*. Todas as dimensões devem ter essa surrogate.

A exceção é, como vimos, a Tempo, que pode ela mesma ser usada como sua própria chave. Entretanto, se encontrar um DW com surrogate na dimensão Tempo, não ache que está errado, pelo contrário. Essa é uma das boas práticas que variam bastante!

Então, além da surrogate, teremos para a Dimensão Cliente e as

especializações como o nome ou razão social, CPF ou CNPJ, endereço etc. Dessa forma, criei a tabela chamada `D_Cliente` com a seguinte estrutura:

Lembrando de que a coluna `Id_Cliente` será preenchida em nosso processo de carga, dado que é uma chave artificial. Uma boa prática nesse sentido é padronizar as surrogate como `Id` e as colunas chave oriundas dos sistemas Transacionais sempre chamadas de `Cod`. Dessa forma, fica simples identificar a origem da informação. Também inseri duas colunas para nosso `data lineage`, a `LinData` e a `LinOrig`. Ambas serão preenchidas também no processo do ETL.

Produto

Se vendemos para alguém, vendemos algo. A dimensão de Produto vai possuir a surrogate e as especializações do produto, como SKU, Nome, Grupo, Subgrupo, e demais propriedades como cor, peso, tamanho etc.

Essa dimensão varia muito de caso para caso, e as informações contidas nela dependem totalmente de como a sua empresa gerencia seus produtos. Mas uma questão interessante é que essa dimensão,

na maioria das vezes, possui a propriedade de ter pequenas alterações em seus registros de tempos em tempos. Essas dimensões que sofrem variações ao longo do tempo são chamadas de *Slowly Changing Dimension*.

Por exemplo, imagine que em 2015 uma empresa vendia o produto "Curso de BI". Em 2016, a empresa muda o nome do produto para "Curso de BI a Custo Zero". Basicamente, as três formas mais comuns de se lidar com essas alterações são:

- **1SCD:** simplesmente atualiza-se a dimensão para contemplar a nova informação. Por ferir o "não volátil" que Inmon prega, essa opção faz com que percamos o passado. Se atualizarmos a dimensão Produto, o que vai ocorrer é que, quando exibirmos um relatório de vendas de 2015 e de 2016, veremos apenas o produto com o novo nome "Curso de BI a Custo Zero" sendo vendido nos dois anos.
- **2SCD:** cria-se uma nova entrada na tabela de Dimensão, deixando uma notação de "inativo" no registro original e de "ativo" no novo registro. Nesse caso, quando exibirmos um relatório de vendas de 2015 e 2016, veremos duas linhas, uma do produto "Curso de BI" com dados para 2015 e sem dados para 2016, e outra linha do produto "Curso de BI a Custo Zero" sem dados para 2015 e com dados para 2016.
- **3SCD:** para as dimensões que sofrerão alterações, coloca-se uma coluna de valor histórico, e atualiza-se o valor da coluna de valor corrente e de valor histórico quando há uma alteração. Nesse caso, quando exibirmos um relatório de vendas de 2015 e 2016, veremos uma única linha com vendas para 2015 e 2016 com o nome novo de "Curso de BI a Custo Zero", mas com a possibilidade de consultar o nome anterior.

Baseado na minha experiência e sem entrar no mérito de certo e errado, a 2SCD se mostra a melhor por alguns motivos:

- Ela mantém uma representação fiel do passado (a 1SCD não);
- Ela suporta infinitas alterações (a 3SCD só suporta o histórico de uma alteração);
- Ela mantém as ligações pelas surrogates, mantendo as buscas o mais rápido possível.

Assim sendo, no caso da Dimensão Produto, a trataremos como *Slowly Changing Dimension* resolvida pela segunda forma 2SCD. Claro que, em uma implantação em sua empresa, deve-se discutir com as áreas clientes para determinar como o usuário deseja visualizar os dados. Em alguns casos, pode-se, por exemplo, criar a dimensão "Produto Imediato" (resolvida como 1SCD) e a "Produto Histórico" (resolvida como 2SCD), mantendo duas visões distintas da mesma informação.

Dessa forma, criei a tabela chamada `D_Produto` com a seguinte estrutura:

D_Produto

Id_Produto (int)
Cod_Produto (varchar(20)) Nome (varchar(50)) Tamanho (varchar(05) Linha (varchar(05)) Cor (varchar(20)) Ativo (char(01)) LinData (date) LinOrig (varchar(50))

Note que temos a surrogate, as qualificações oriundas do sistema origem e um flag para determinar se o registro está ou não ativo (para atender à 2SCD). Há quem questionará onde estão as datas de início e fim de vigência do registro. Na verdade, eu, também por experiência do dia a dia mais do que por seguir um conceito acadêmico, resolvo essas datas na ligação com a Fato.

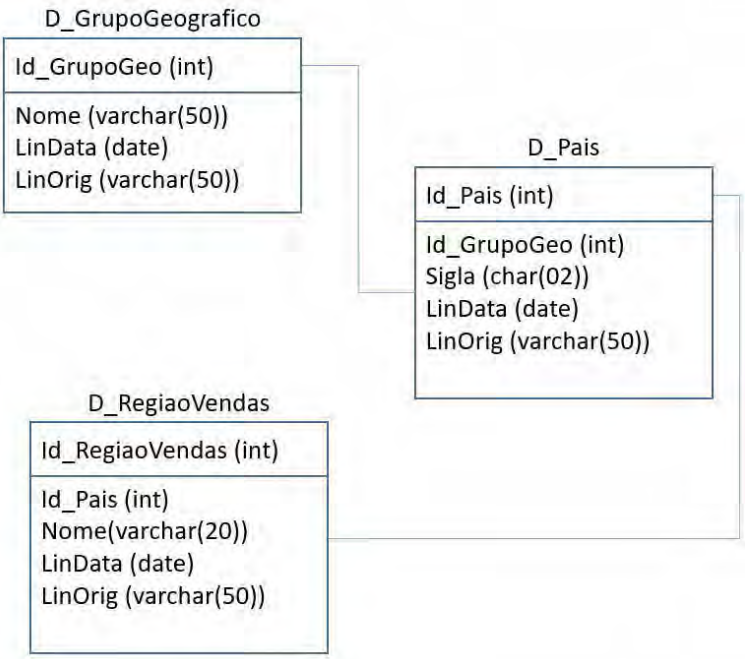
Enquanto o registro estiver ativo, haverá lançamentos na Fato para ele. Quando ele não estiver mais ativo, as datas dos novos registros estarão vinculadas ao produto ativo. E se não houver venda? Não havendo um Fato, a informação na Dimensão é muito pouco útil! Dessa forma, simplificamos a estrutura e o processo de carga sem comprometer a informação exibida. Os dados de lineage constam como as duas últimas colunas.

Geografia

Geralmente, uma das informações relativas aos Clientes é de onde eles são, e isso é bastante importante. Saber onde nossos produtos fazem sucesso e onde eles precisam ser mais incentivados pode ser de grande valia. Dessa forma, a Dimensão de Geografia pode ser formada por País, Estado, Cidade e o que mais de informação estiver disponível, como Bairro, Logradouro, CEP, Região (Norte, Sudeste, Sul etc.).

Não é incomum as empresas terem divisão territorial do processo de vendas. Se for o caso, pode-se colocar na Dimensão Geografia a Região de Venda, que podem ser definidas por range de CEP, por coordenadas cardinais etc.

Dessa forma, criei as tabelas chamadas `D_GrupoGeografico`, `D_Pais` e `D_RegiaoVendas` com as seguintes estruturas:



A estrutura segue a primeira Forma Normal no esquema Floco de Neve, no qual colocamos a Primary Key da tabela pai como Foreign Key na tabela filho, evitando assim os chamados grupos de repetição. Dessa forma, ligaremos apenas a tabela mais granular (ou seja, a `D_RegiaoVendas` com a `Fato`), e poderemos ter todos os dados das tabelas pai.

E, claro, temos as colunas de `data lineage` em todas as três tabelas.

Vendedor

Nem sempre esse dado está disponível em arquivos de notas fiscais, mas vou inserir em nosso exemplo para essa dimensão cobrir um exemplo bastante importante: a *Parent-Child Dimension*, ou dimensão de Pais e Filhos. Esta permite criar uma hierarquia dinamicamente, com infinitos níveis e diferentes estruturas. E a

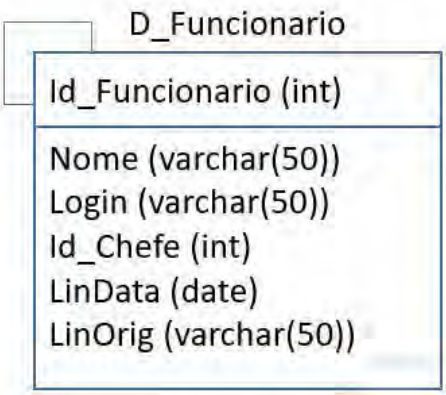
flexibilidade toda está simplesmente na forma como ela é construída!

Vamos imaginar que nossa empresa tenha uma estrutura de Vendedor definida por um Diretor que possui alguns Gerentes. Estes podem ou não ter Coordenadores, dependendo do tamanho do time. Se não houver Coordenador, os Vendedores estarão logo abaixo do Gerente. Se houver Coordenador, ele estará abaixo do Gerente e os Vendedores abaixo do Coordenador. Implementar essa estrutura de forma flexível basicamente se resolve com uma coluna na tabela de Dimensão!

Na dimensão `Vendedor` , teremos a coluna chave, a surrogate, e as colunas de especializações: Nome, CPF, matrícula, e-mail etc. Além delas, vamos incluir uma coluna para `Chefe` . Nela, vamos inserir o valor que está na surrogate da pessoa que será o chefe desse Vendedor. Pronto! Assim, dinamicamente, podemos ter sempre como montar a árvore de hierarquia.

Vale lembrar de que esse conceito serve para qualquer situação em que se tenha a relação de um registro "pai" e que se pode ter **n** registros "filhos". Desse jeito, monta-se a estrutura dinamicamente de relações entre esses registros, como uma árvore de diretórios no seu Windows Explorer, por exemplo!

Dessa forma, criei a tabela chamada `D_Funcionario` com as seguintes estruturas:



Vale mencionar que não batizamos essa dimensão de "Vendedor", porque, via de regra, essas pessoas são funcionárias da empresa. Quando nosso DW for crescendo, eventualmente dados de Recursos Humanos serão importados e todos os funcionários estarão carregados. No caso de termos uma dimensão "Vendedor", teríamos esses registros duplicados com a dimensão "Funcionário", ou teríamos de alterar o nome da Dimensão, o que geraria um belo impacto em retrabalho com o que já estivesse em produção!

A coluna `Id_Chefe` será o que garantirá da relação de *Parent-Child*. Note que, para tanto, a tabela é autorreferenciada, ou seja, a Primary Key está ligada a uma Foreign Key e ambas estão na mesma tabela.

No nosso cenário, cobrimos a dimensão Tempo, uma Dimensão Simples (a Cliente), uma SCD (a Produto), uma implantação em esquema Snow Flake (a Geografia) e uma Parent-Child (a Vendedor). Com isso, cobrimos os principais tipos de dimensões, e vamos parar por aqui para não nos delongarmos muito!

Existem dimensões calculadas, dimensões que se comportam como 2SCD e Parent-Child ao mesmo tempo etc. Mas com os casos que cobrimos já temos base para resolver a esmagadora maioria do

que se encontra em uma implantação inicial de BI.

Em sua implantação, aproveite para carregar e disponibilizar todas as informações disponíveis mesmo que elas não tenham sido solicitadas. Se sua origem de dados tiver, por exemplo, a informação de Loja ou Canal de Venda, crie as respectivas Dimensões! Sempre que a informação existir, avalie seriamente a possibilidade de incluí-la no seu DW.

Fatos

Uma vez que tenhamos desenhado todas as dimensões conhecidas (e com certeza novas dimensões aparecerão com a vida do DW), é hora de desenhar a tabela Fato. No nosso caso, temos dois níveis distintos de granularidade: o cabeçalho e o detalhe.

Recordando da nossa matriz de Fatos e Dimensões, a MFD, veremos que temos cruzamentos que percorrem todas as possibilidades e cruzamentos específicos. Note que, no cruzamento das "Measures" com a Data, poderemos ver todas as medidas por todas as informações temporais:

	Dimensão	Data			
	Hierarquia Default	Data Composta	Ano	Mês	Dia
	Type	Time	Years	MonthOfYear	DayOfMonth
Origem	MEASURES				
Sales Header	VlrImpostos		X	X	X
	VlrFrete		X	X	X
Sales Details	PrecoUnitario		X	X	X
	Quantidade		X	X	X

Isso significa que, por termos a Data da Venda, saberemos os valores de Impostos, por exemplo, num determinado mês. Saberemos também quanto vendemos de um determinado produto em um ano específico, e assim por diante.

As dimensões de cliente, funcionários e de região de vendas

seguem o mesmo padrão. O detalhe para essa última é que, na MFD, colocamos o nome da dimensão sempre com o termo mais granular, ou seja, Região Venda! E para cada nível, o nome significativo (aquele que o usuário poderá ver e será compreensível para ele):

	Dimensão	Regiao Vendas			
	Hierarquia Default	Regiao Vendas	Grupo	Pais	RegiaoVendas
	Type	SnowFlake	DB_STR	DB_STR	DB_STR
Origem	MEASURES				
Sales Header	VlrImpostos		X	X	X
	VlrFrete		X	X	X
Sales Details	PrecoUnitario		X	X	X
	Quantidade		X	X	X

Mas ao verificarmos o cruzamento da dimensão Produto com as nossas "Measures", entendemos que nem sempre os cruzamentos valerão para todos:

	Dimensão	Produto	
	Hierarquia Default	Produto	Nome
	Type	2SCD	DB_STR
Origem	MEASURES		
Sales Header	VlrImpostos		-
	VlrFrete		-
Sales Details	PrecoUnitario		X
	Quantidade		X

Note que não teremos os dados referentes ao cabeçalho da Nota para a dimensão Produto, o que faz todo sentido, dado que o produto é mais granular do que os dados do cabeçalho!

Explicando: se uma Nota tem diversos produtos e apenas um valor de frete, qual seria o valor de frete por produto? Não é possível saber! O valor do Frete não está disponível para a dimensão Produto!

Vale mencionar que existiria, sim, uma forma de calcular esse

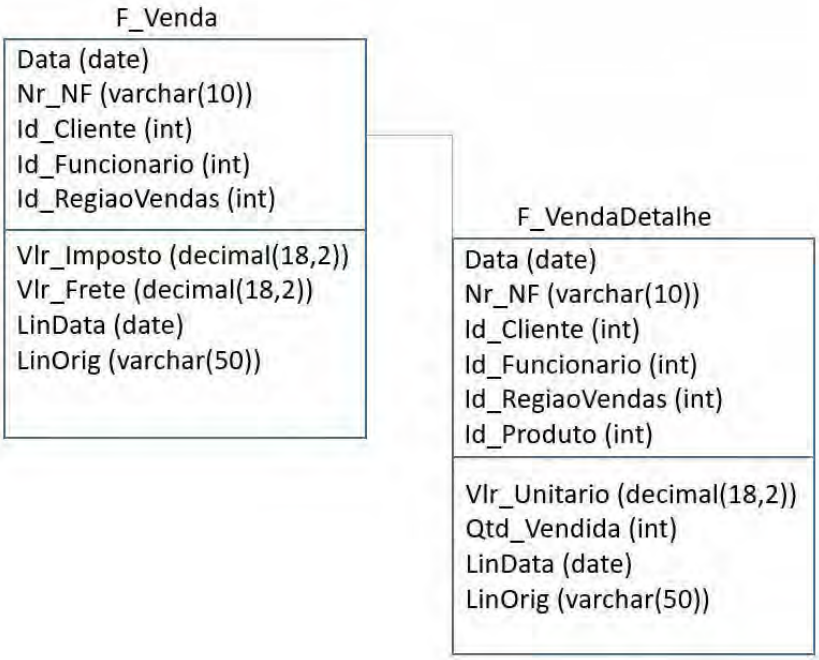
frete por produto. Poderíamos fazer o que se chama de "rateio", ou seja, cada produto recebe o valor da média ponderada do valor do frete. Se seu usuário assim quiser, muito bem! Calcule para ele, e torne os totais "mais granulares" a ponto de serem exibidos para os detalhes. Mas por ora é interessante entender que nem sempre todas as "Measures" estarão obrigatoriamente disponíveis para todas as Dimensões.

Fato Vendas

A nossa Fato será sobre Vendas! Teremos os dados de quantas vendas fizemos, qual o valor, impostos e taxas totais de cada uma, o que foi vendido e quanto de cada produto!

Um ponto interessante é entender o modelo de dados que usaremos para a Fato de Vendas. Na verdade, para esse caso, teremos mais de uma tabela física respondendo por essa "Fato Lógica".

Isso quer dizer que: quando houver um evento (um Fato que no nosso caso é a venda) em que há diferentes granularidades (nesse caso, um cabeçalho e um detalhamento), poderemos ter mais de uma tabela física para responder a essa situação. Dessa forma, criei as tabelas chamadas `F_Venda` e `F_VendaDetalhe` com as seguintes estruturas:



Note que as tabelas de Fato possuem uma Primary Key composta com todas as demais chaves das Dimensões que se relacionam com ela. Ela mesma não possui nenhuma surrogate própria.

Outro ponto é que a Fato também possui as informações de data lineage e o número da nota fiscal (`Nr_NF`). Existe uma prática de colocar dados descritivos nas Fatos quando eles estão na mesma granularidade dela. Contudo, precisamos usar esse recurso com parcimônia!

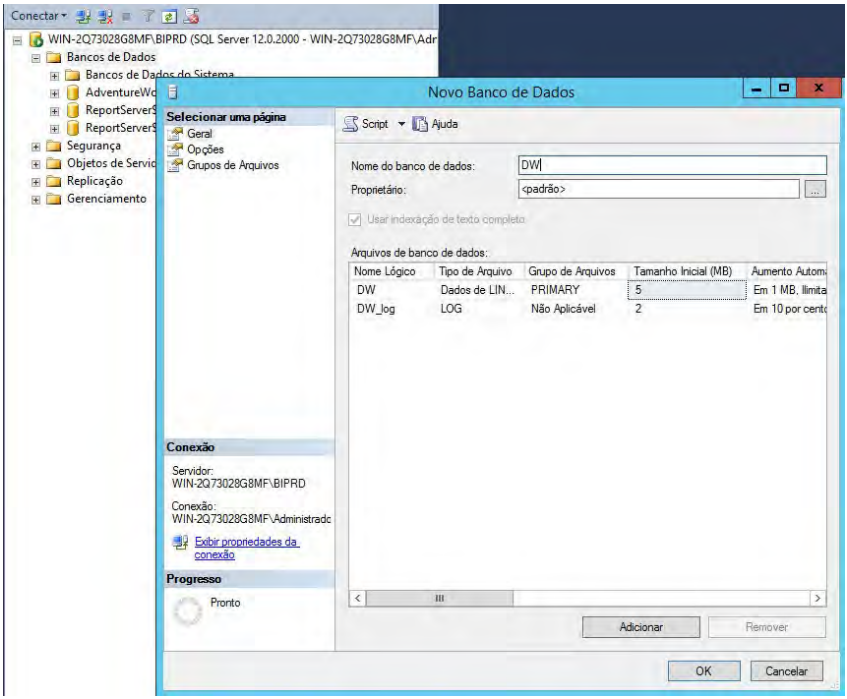
No nosso modelo, colocamos um dado descritivo "pequeno" (com poucos caracteres). Se tivéssemos textos descritivos, como o campo de Observação da NF, por exemplo, ou diversos campos descritivos, o recomendado seria criar uma dimensão para a nota fiscal e atribuir a ela a responsabilidade de conter os textos todos.

Em ambientes de DW gigantescos, a inclusão de campos grandes nas Fatos faz com que haja maior fragmentação e consequente perda de performance. Se nos lembrarmos de que o DW é essencialmente desenhado para manter a performance das consultas, evitar práticas que afetem essa performance é, no mínimo, coerente.

Outro ponto a notar é que o sumarizador na nota fiscal não consta na `F_Venda`. Teremos esse dado calculado pela somatória da `F_VendaDetalhe`. Com esse conjunto de entidades desenhadas, temos o nosso DW conceituado! É hora de implantá-lo no SQL e torná-lo algo real.

Para iniciarmos, vamos criar a Base de Dados que será nosso DW. Eu geralmente uso o nome da empresa e a notação `DW` ao final. Para nosso exemplo, vamos usar apenas `DW`.

No Management Studio, clique com o botão direito do mouse sobre `Banco de Dados` e selecione `Novo Banco de Dados`:



Vamos manter todas as opções padrão para simplificar! Clique em `OK` e a base de dados chamada `DW` aparecerá na árvore de objetos. Uma vez com sua Base de Dados criada, vamos à criação das Dimensões e das Fatos. Para tanto, basta executar o seguinte comando:

```
USE [DW]
GO


CREATE TABLE [dbo].[D_Cliente](
    [Id_Cliente] [int] identity(1,1) NOT NULL,
    [Cod_Cliente] [varchar](10) NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [Email] [varchar](50) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_Cliente] PRIMARY KEY CLUSTERED
(
    [Id_Cliente] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_Data](
    [Data] [date] NOT NULL,
    [Dia] [char](2) NOT NULL,
    [Mes] [char](2) NOT NULL,
    [Ano] [char](4) NOT NULL,
 CONSTRAINT [PK_D_Data] PRIMARY KEY CLUSTERED
(
    [Data] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_Funcionario](
    [Id_Funcionario] [int] identity(1,1) NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [Login] [varchar](50) NOT NULL,
    [Id_Chefe] [int] NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_Funcionario] PRIMARY KEY CLUSTERED
(
    [Id_Funcionario] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_GrupoGeografico](
    [Id_GrupoGeo] [int]  identity(1,1)NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_GrupoGeografico] PRIMARY KEY CLUSTERED
(
    [Id_GrupoGeo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
```

```
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_Pais](
    [Id_Pais] [int] identity(1,1) NOT NULL,
    [Id_GrupoGeo] [int] NOT NULL,
    [Sigla] [char](2) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_Pais] PRIMARY KEY CLUSTERED
(
    [Id_Pais] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_Produto](
    [Id_Produto] [int] identity(1,1) NOT NULL,
    [Cod_Produto] [varchar](20) NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [Tamanho] [varchar](5) NOT NULL,
    [Cor] [varchar](20) NOT NULL,
    [Ativo] [char](1) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_Produto] PRIMARY KEY CLUSTERED
(
    [Id_Produto] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[D_RegiaoVendas](
    [Id_RegiaoVendas] [int] identity(1,1) NOT NULL,
    [Id_Pais] [int] NOT NULL,
    [Nome] [varchar](20) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_D_RegiaoVendas] PRIMARY KEY CLUSTERED
```

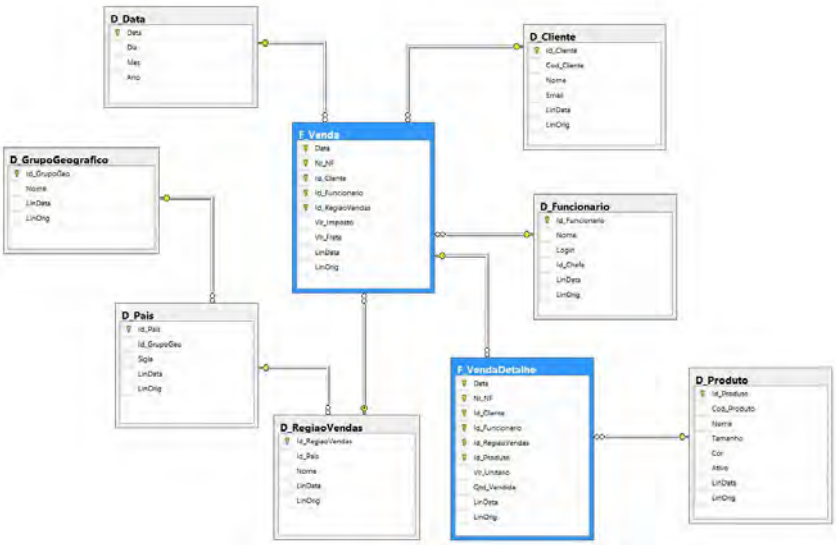
```
(
    [Id_RegiaoVendas] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[F_Venda](
    [Data] [date] NOT NULL,
    [Nr_NF][varchar] (10) NOT NULL,
    [Id_Cliente] [int] NOT NULL,
    [Id_Funcionario] [int] NOT NULL,
    [Id_RegiaoVendas] [int] NOT NULL,
    [Vlr_Imposto] [decimal](18, 2) NOT NULL,
    [Vlr_Frete] [decimal](18, 2) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_F_Venda] PRIMARY KEY CLUSTERED
(
    [Data] ASC,
    [Nr_NF] ASC,
    [Id_Cliente] ASC,
    [Id_Funcionario] ASC,
    [Id_RegiaoVendas] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[F_VendaDetalhe](
    [Data] [date] NOT NULL,
    [Nr_NF][varchar] (10) NOT NULL,
    [Id_Cliente] [int] NOT NULL,
    [Id_Funcionario] [int] NOT NULL,
    [Id_RegiaoVendas] [int] NOT NULL,
    [Id_Produto] [int] NOT NULL,
    [Vlr_Unitario] [decimal](18, 2) NOT NULL,
    [Qtd_Vendida] [int] NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_F_VendaDetalhe] PRIMARY KEY CLUSTERED
(
    [Data] ASC,
    [Nr_NF] ASC,
```

```
    [Id_Cliente] ASC,
    [Id_Funcionario] ASC,
    [Id_RegiaoVendas] ASC,
    [Id_Produto] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
EY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY
]
) ON [PRIMARY]
GO


CREATE NONCLUSTERED INDEX [IX_D_Pais] ON [dbo].[D_Pais]
(
    [Id_GrupoGeo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMP
DB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
 ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO




CREATE NONCLUSTERED INDEX [IX_D_RegiaoVendas] ON [dbo].[D_RegiaoVe
ndas]
(
    [Id_Pais] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMP
DB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
 ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO


ALTER TABLE [dbo].[D_Pais]  WITH CHECK ADD  CONSTRAINT [FK_D_Pais_
D_GrupoGeografico] FOREIGN KEY([Id_GrupoGeo])
REFERENCES [dbo].[D_GrupoGeografico] ([Id_GrupoGeo])
GO


ALTER TABLE [dbo].[D_Pais] CHECK CONSTRAINT [FK_D_Pais_D_GrupoGeog
rafico]
GO


ALTER TABLE [dbo].[D_RegiaoVendas]  WITH CHECK ADD  CONSTRAINT [FK
_D_RegiaoVendas_D_Pais] FOREIGN KEY([Id_Pais])
REFERENCES [dbo].[D_Pais] ([Id_Pais])
GO
```

```
ALTER TABLE [dbo].[D_RegiaoVendas] CHECK CONSTRAINT [FK_D_RegiaoVe
ndas_D_Pais]
GO


ALTER TABLE [dbo].[F_Venda]  WITH CHECK ADD  CONSTRAINT [FK_F_Vend
a_D_Cliente] FOREIGN KEY([Id_Cliente])
REFERENCES [dbo].[D_Cliente] ([Id_Cliente])
GO


ALTER TABLE [dbo].[F_Venda] CHECK CONSTRAINT [FK_F_Venda_D_Cliente
]
GO


ALTER TABLE [dbo].[F_Venda]  WITH CHECK ADD  CONSTRAINT [FK_F_Vend
a_D_Data] FOREIGN KEY([Data])
REFERENCES [dbo].[D_Data] ([Data])
GO
ALTER TABLE [dbo].[F_Venda] CHECK CONSTRAINT [FK_F_Venda_D_Data]
GO


ALTER TABLE [dbo].[F_Venda]  WITH CHECK ADD  CONSTRAINT [FK_F_Vend
a_D_Funcionario] FOREIGN KEY([Id_Funcionario])
REFERENCES [dbo].[D_Funcionario] ([Id_Funcionario])
GO


ALTER TABLE [dbo].[F_Venda] CHECK CONSTRAINT [FK_F_Venda_D_Funcion
ario]
GO


ALTER TABLE [dbo].[F_Venda]  WITH CHECK ADD  CONSTRAINT [FK_F_Vend
a_D_RegiaoVendas] FOREIGN KEY([Id_RegiaoVendas])
REFERENCES [dbo].[D_RegiaoVendas] ([Id_RegiaoVendas])
GO


ALTER TABLE [dbo].[F_Venda] CHECK CONSTRAINT [FK_F_Venda_D_RegiaoV
endas]
GO


ALTER TABLE [dbo].[F_VendaDetalhe]  WITH CHECK ADD  CONSTRAINT [FK
```

```
_F_VendaDetalhe_D_Produto] FOREIGN KEY([Id_Produto])
REFERENCES [dbo].[D_Produto] ([Id_Produto])
GO


ALTER TABLE [dbo].[F_VendaDetalhe] CHECK CONSTRAINT [FK_F_VendaDet
alhe_D_Produto]
GO


ALTER TABLE [dbo].[F_VendaDetalhe]  WITH CHECK ADD  CONSTRAINT [FK
_F_VendaDetalhe_F_Venda] FOREIGN KEY([Data], [Nr_NF],[Id_Cliente],
 [Id_Funcionario], [Id_RegiaoVendas])
REFERENCES [dbo].[F_Venda] ([Data], [Nr_NF],[Id_Cliente], [Id_Func
ionario], [Id_RegiaoVendas])
GO


ALTER TABLE [dbo].[F_VendaDetalhe] CHECK CONSTRAINT [FK_F_VendaDet
alhe_F_Venda]
GO
```

Executado o comando, ao expandir as tabelas do seu DW, todas as tabelas estarão criadas em um esquema conforme:



3.3 CONCLUSÃO

Neste capítulo, criamos o coração da nossa plataforma de BI! Nosso Data Warehouse ganhou corpo e está preparado para receber os dados, o que faremos no próximo capítulo!

Alguns conceitos de bancos de dados básicos ajudam na criação do seu DW:

- Nenhuma coluna pode aceitar `null`. Se um dado veio nulo, o processo de carga deve entender se é algo esperado ou um erro. Deve-se preencher o campo com uma informação genérica para que se saiba que aquele dado não foi carregado. Essa prática resolve muitos problemas de origens de dados incoerentes. Faremos alguns exemplos no nosso ETL!
- Toda coluna que for chave estrangeira deve ser indexada (conforme consta no comando que executamos anteriormente)! Se houver a necessidade de indexar outras colunas por conta das buscas, sem problemas. Mas as chaves estrangeiras serão usadas com certeza, então devemos indexá-las!
- Hoje em dia, não há uma gigantesca preocupação com espaço em disco, mas nossa versão de SQL é gratuita e tem suas limitações. Então vamos tentar trabalhar com os menores *data types* possíveis.

Com tudo criado no lado do Data Warehouse, é hora de carregarmos os dados!

CAPÍTULO 4

O PROCESSO DE ETL

4.1 CONSIDERAÇÕES INICIAIS

Já temos o nosso Data Warehouse montado e conhecemos a origem das informações (seja porque seguimos o passo a passo deste livro, ou porque pegamos dados reais do ambiente da nossa empresa). Nesse ponto, temos todas as informações necessárias para criarmos o processo que vai ler das origens, transformar e carregar os dados no nosso DW.

Se tivéssemos uma ferramenta de ETL, como o SSIS, por exemplo, nosso processo seria um pouco mais simples e talvez mais performático! Como nossa proposição é fazer todo o processo sem nenhuma ferramenta paga, vamos usar o que temos em mãos: Transact-SQL e Agendamentos do Windows!

Mas uma carga feita inteiramente em código tem uma enorme vantagem: vamos poder conceituar cada um dos passos, e assim teremos subsídios para implantar esses mesmos passos em outras ferramentas quando nosso budget nos permitir adquiri-las.

Como vamos fazer todos os processos com o T-SQL (ou *Transact SQL*, a linguagem de programação do SQL Server, tal qual o PL/SQL é a do Oracle), alguns conceitos dessa linguagem serão necessários. Passaremos pelos tópicos mais específicos dos processos que utilizaremos, mas se você ainda não tem familiaridade nenhuma com ela, recomendo dar uma pausa para

estudar um pouco antes de prosseguir.

Nosso maior uso será da instrução `SELECT` do T-SQL. Se quiser um tutorial inicial sobre o tema, o MSDN tem um conjunto de lições que usa justamente a base de dados AdventureWorks que nós temos em nosso servidor.

Então, clique em https://msdn.microsoft.com/en-us/library/cc546519.aspx, execute os comandos, e entenda os conceitos! Recomendo as duas primeiras lições (em inglês).

Havendo o interesse em se aprofundar (e eu sempre recomendo que esse aprofundamento ocorra!), recomendo o livro *Microsoft SQL Server 2012 T-SQL Fundamentals (Developer Reference)* e o *Microsoft SQL Server 2012 T-SQL Fundamentals (Developer Reference)*, ambos da editora Microsoft Press.

O SQL Server é um produto imenso, com centenas de assuntos para se aprofundar, desde instalação e administração, até as ferramentas de Business Intelligence, como o SSAS, SSRS e SSIS, passando por replicação, alta disponibilidade etc. Porém, tudo começa com um bom entendimento de como usar a sintaxe de comando dele. Por isso, seja qual seu caminho de interesse com esse produto, conhecer bem T-SQL lhe será sempre muito útil.

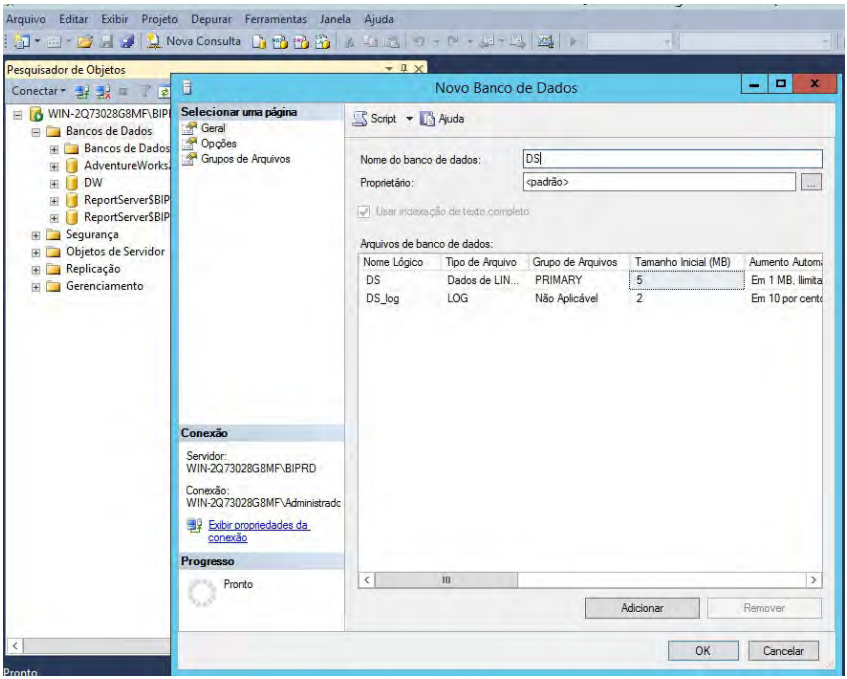
4.2 DATA STAGE

Um dos postulados do Data Warehouse é que ele não é volátil, ou seja, se um registro entrou nele, não pode ser alterado. Mas no processo de carga, teremos de avaliar se o dado está correto ou não, precisaremos eventualmente alterar a formatação dele, verificar se ele deve mesmo ser carregado ou se ele é uma duplicidade etc.

Se vamos proceder com todos esses trabalhos e se o DW não deve sofrer alterações, teremos de arrumar um outro lugar para

fazer tudo isso. Esse lugar é, na verdade, uma base de dados que se usa para trabalhar com os dados e que é populada no processo de carga e depois devidamente limpa para o próximo ciclo.

Essa base recebe o nome de Data Stage e, além de conter as tabelas temporárias, vai conter também todos os códigos que faremos para o processo de ETL, as *Stored Procedures* (procedimentos armazenados). Para criarmos nossa Data Stage, vamos abrir o Management Studio. Clique em `Banco de Dados` com o botão direito e selecione `Novo Banco de Dados...` :



Dê o nome de `DS` e clique em OK. Novamente vamos usar todas as opções padrão para facilitar a criação.

Uma vez que temos a base DS, precisaremos habilitar uma função do SQL que, por padrão, vem desabilitada. No Management Studio, rode o comando a seguir:

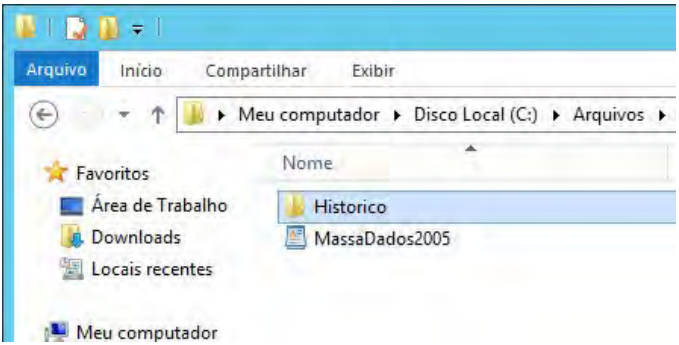
```
exec sp_configure "show advanced options", 1
go
reconfigure
go
EXEC sp_configure 'xp_cmdshell', 1
go
reconfigure
go
```

Esse comando vai ativar a `xp_cmdshell`, que será necessária para passarmos comandos para o Windows diretamente pelo SQL Server!

Além disso, vamos criar também uma tabela que receberá toda a informação dos passos do nosso ETL. Ela será consultada para sabermos se tudo rodou bem, ou se tivemos algum problema em algum passo. Vamos batizá-la de `Adm_Log` (dado que ela tem uma função administrativa do processo, que é guardar os logs dos processos de carga). Para isso, basta rodar o comando:

```
USE [DS]
GO


CREATE TABLE [dbo].[Adm_Log](
    [Id_Log] [uniqueidentifier] primary key NOT NULL,
    [Data] [datetime] NOT NULL,
    [Passo] [varchar](50) NOT NULL,
    [SucessoFalha] [char](1) NOT NULL,
    [mensagem] [varchar](255) NOT NULL
    )
```

Uma vez criado nosso DS e nossa tabela de Log, e habilitado o recurso que precisaremos, vamos criar uma estrutura no nosso diretório para receber os arquivos de importação de dados e para guardá-los assim que forem importados. No nosso drive `C:\` já havíamos criado a pasta `Arquivos` onde colocamos o arquivo chamado `MassaDados2005.rpt`. Basta criar uma pasta dentro de `Arquivos` com o nome de `Historico`:



Feito isso, nosso ambiente está pronto! Podemos começar!

4.3 MANIPULAÇÃO DE ARQUIVOS

A primeira coisa a se fazer é importar o arquivo `MassaDados.rpt` para o SQL. Uma vez que tenhamos todo ele dentro da estrutura de tabelas do banco de dados, poderemos trabalhar com seus registros de forma mais simples. Como tivemos o cuidado de gerar esse arquivo com separação de colunas por `|`, o que faremos será a criação de uma tabela temporária que receberá todas as colunas, uma a uma, do arquivo de origem.

Quando outras fontes de dados aparecerem, pode-se replicar esse processo de importação de arquivos para cada uma, tendo um processo para cada arquivo de origem de dados.

Para importar o arquivo texto, precisaremos então de uma tabela que possa comportar as linhas com suas respectivas colunas. Vamos batizar essa tabela de `TbImp_Vendas`, dado que é uma "**Tab**ela de **Imp**ortação dos dados de **Vendas**". Para criá-la, basta executar o código:

```
USE [DS]
GO
```

```
CREATE TABLE [dbo].[TbImp_Vendas](
    [NrNf] [varchar](50) NULL,
    [DataVenda] [varchar](50) NULL,
    [CodCliente] [varchar](50) NULL,
    [NomeCliente] [varchar](50) NULL,
    [EmailCliente] [varchar](50) NULL,
    [RegiaoVendas] [varchar](50) NULL,
    [Pais] [varchar](50) NULL,
    [GrupoGeografico] [varchar](50) NULL,
    [VendedorLogin] [varchar](50) NULL,
    [VendedorNome] [varchar](50) NULL,
    [VendedorChefeNome] [varchar](50) NULL,
    [Cod_Produto] [varchar](20) NULL,
    [Produto] [varchar](50) NULL,
    [Tamanho] [varchar](50) NULL,
    [Linha] [varchar](50) NULL,
    [Cor] [varchar](50) NULL,
    [SubTotal] [varchar](50) NULL,
    [ImpTotal] [varchar](50) NULL,
    [Frete] [varchar](50) NULL,
    [PrecoUnitario] [varchar](50) NULL,
    [Qtd] [varchar](50) NULL
) ON [PRIMARY]
```

Feita a criação da tabela, vamos criar um procedimento armazenado que fará os seguintes passos:

1. Apagar os dados da `TbImp_Vendas` caso existam de uma carga anterior;
2. Renomear os arquivos existentes para um nome esperado;
3. Importar os dados do arquivo para o SQL;
4. Fazer o log do processo;
5. Copiar o arquivo para a pasta `Historico`, renomeando-o com a data da importação.

Esse procedimento será batizado de `Importa_Vendas` e é criado com o comando a seguir:

```
USE [DS]
GO


Create procedure [dbo].[Importa_Vendas]
as
```

```
Declare @STR as nvarchar(1000)


-- Apaga os dados da TbImp_Vendas caso existam para uma nova carga;

    truncate table TbImp_Vendas


-- Renomeia arquivos existentes para um nome esperado:
    set @STR = 'Move c:\Arquivos\*.rpt c:\Arquivos\MassaDados.rpt'
    exec xp_cmdshell @STR




--Importa os dados do arquivo para o SQL:
    bulk insert [dbo].[TbImp_Vendas]
    from 'C:\Arquivos\MassaDados.rpt'
    with (
        FIRSTROW = 3,
        FIELDTERMINATOR  = '|',
        ROWTERMINATOR = '\n'
        )


-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta MassaDados.rpt','S', 'Arquivo importado com sucesso')


-- Copia o arquivo para a pasta “Historico”, renomeando ele com a
data da importação:

    declare @nomearquivo varchar(50)
        Set @nomearquivo = (Select cast(year(getdate())as char(4))
 + right('00'+ cast(month(getdate())as varchar(2)),2)+ right('00'+
cast(day(getdate())as varchar(2)),2) +'_Vendas.rpt')


        Set @STR = 'move c:\Arquivos\MassaDados.rpt c:\Arquivos\Hi
storico\' + @nomearquivo


        exec xp_cmdshell @STR
```

O resultado esperado é `Comando(s) concluído(s) com êxito .`

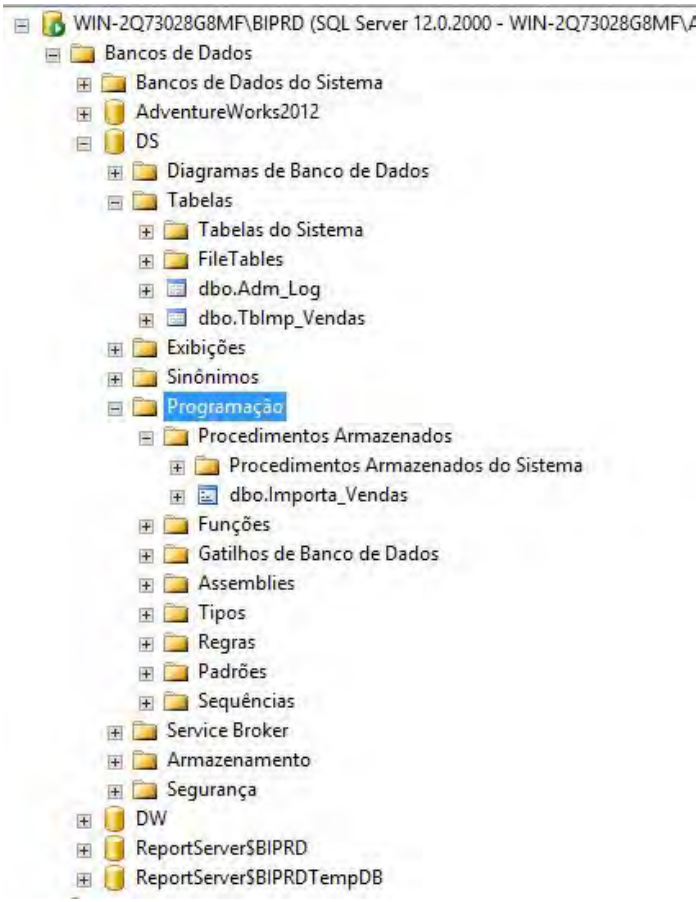
Alguns comentários sobre esse comando: inserimos os dados pelo `Bulk Insert`, o processo mais performático. Para tanto, iniciamos a carga na terceira linha, eliminando os nomes das colunas (`firstrow = 3`), e usamos o delineador de linha `Enter` (`ROWTERMINATOR = '\n'`) e o delineador de campo (`FIELDTERMINATOR = '|'`).

Note que a execução foi imediata e nada foi carregado. Nós apenas criamos a rotina que vai fazer esse trabalho, mas ainda não a executamos. Para tanto, basta comandar em uma nova janela do Management Studio:

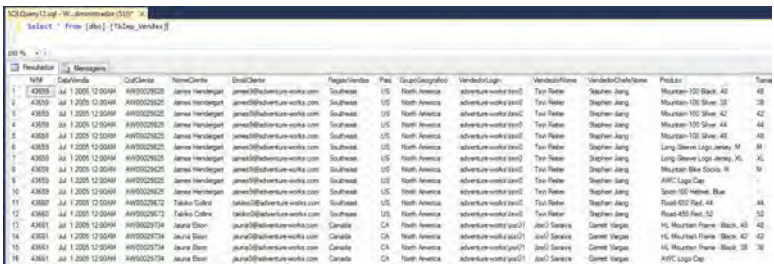
```
USE [DS]
GO
exec Importa_Vendas
```

Ao final da execução, devemos ter:

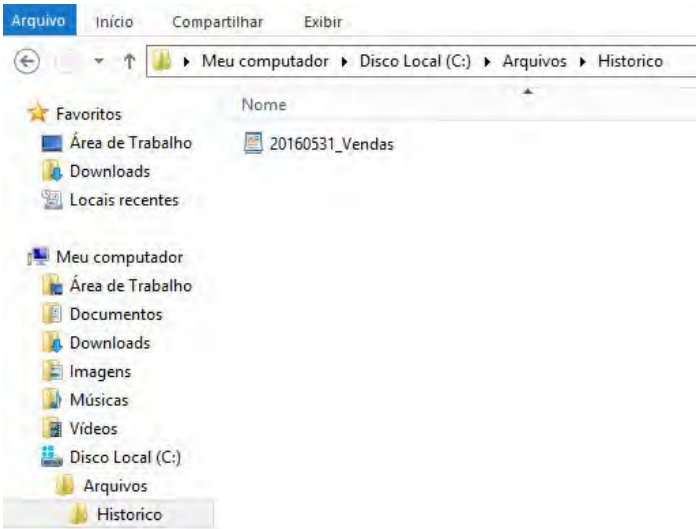
1. As tabelas `TbImp_Vendas` e `Adm_Log` criadas, bem como uma Stored Procedure chamada `Importa_vendas` :



2. Os dados na tabela `TbImp_Vendas` devem ter sido carregados. Usando nosso exemplo, teremos 5.149 linhas:



3. O arquivo `MassaDados2005.rpt` deve ter desaparecido da pasta `Arquivos`, e um arquivo com a data atual e a terminação `_Vendas.rpt` deve ter surgido na pasta `Historico`:



Dessa forma, poderemos comandar esse procedimento para ocorrer toda vez que um novo arquivo for gerado com os dados de um novo dia. Com esse agendamento, a plataforma de BI vai sendo carregada constantemente, mas falaremos disso adiante, ainda neste capítulo!

Observação Ao executar o procedimento `Importa_Vendas`, o seu retorno pode ser o erro:

```
USE [DS]
GO
exec Importa_Vendas
```

```
Mensagens
Mensagem 4832, Nível 16, Estado 1, Procedimento Importa_Vendas, Linha 11
Carregamento em massa: fim de arquivo inesperado no arquivo de dados.
Mensagem 7399, Nível 16, Estado 1, Procedimento Importa_Vendas, Linha 11
O provedor do OLE DB "BULK" para o servidor vinculado "(null)" reportou um erro. O provedor não forneceu informações sobre o erro.
Mensagem 7330, Nível 16, Estado 2, Procedimento Importa_Vendas, Linha 11
Não é possível buscar uma linha no provedor do OLE DB "BULK" para o servidor vinculado "(null)".
```

Não há motivo para pânico! Abra o arquivo `MassaDados.rpt` (com o NotePad++ ou com o bloco de notas mesmo, tanto faz) e digite `Crtl + End` para ir direto ao final do arquivo. É possível que você encontre o seguinte resultado (gerado por conta de como foi criada a massa de testes):

```
75123|Jul 31 2008 12:00AM|AW00018759|Devin Phillips|devin38@adventure-w
75123|Jul 31 2008 12:00AM|AW00018759|Devin Phillips|devin38@adventure-w
75123|Jul 31 2008 12:00AM|AW00018759|Devin Phillips|devin38@adventure-w

(121317 linha(s) afetadas)
```

Manualmente mesmo, apague essas três últimas linhas (as duas em branco e a que contém o contador de linhas afetadas). Feche e salve o arquivo, e execute o procedimento novamente com o comando:

```
USE [DS]
GO
exec Importa_Vendas
```

Dessa vez, o erro deve ter desaparecido e uma menção de que o arquivo foi movimentado será exibido como retorno. Com os dados na tabela `TbImp_Vendas` , o que vamos fazer é começar a carregar as Dimensões e depois as Fatos. Uma a uma!

Para isso, criaremos uma tabela de “Stage” para cada dimensão, com os mesmos atributos das tabelas finais do nosso DW.

Nela, colocaremos todos os dados fazendo todos os ajustes necessários. A ideia é que, se um dado foi posto na entidade do Data Stage, ele será incluído no Data Warehouse sem problemas.

4.4 CARREGANDO A DIMENSÃO DATA

Como nossa primeira dimensão é a Data, vamos começar por ela. A má notícia é que não vai ser tão simples como carregar a data "hoje" e pronto, dado que os registros estão sendo carregados agora. Note que as vendas ocorreram em datas diferentes e, no nosso exemplo, em anos anteriores. O que precisamos é inserir no nosso DW as datas em que as vendas ocorreram! A boa notícia é que, ainda assim, é um processo bastante simples.

Primeiro vamos criar a tabela Data na Base de Stage. Vamos batizá-la de `D_Data` mesmo, dado que estará em outra base de dados. Como teremos de usar o nome das bases de dados nas consultas, não haverá confusão. Mas, novamente, se você quiser ser mais puritano do que eu e batizá-la com a notação `DS_D_Data`, fique à vontade!

A criação da dimensão na Stage segue exatamente os conceitos do Data Warehouse. Então, podemos criar essa tabela usando o mesmo comando:

```
USE [DS]
GO


CREATE TABLE [dbo].[D_Data](
    [Data] [date] Primary Key NOT NULL,
    [Dia] [char](2) NOT NULL,
    [Mes] [char](2) NOT NULL,
    [Ano] [char](4) NOT NULL,
    )
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios:

- A Data é chave primária, ou seja, não pode repetir. Teremos de ter sempre um único registro para cada Data.
- Devemos ter a Data no formato de `Date` , e Ano, Mês e Dia separados.
- Será uma carga sempre incremental, ou seja, serão carregados somente dados que não existam no DW. Mesmo que um novo arquivo possua uma data existente no arquivo que estamos carregando hoje, os dados não podem se repetir.
- Se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

Quanto à criação da tabela Stage, note que é importante que a chave primária da tabela no DW seja também chave primária no DS. Isso, além de garantir unicidade, garantirá uma boa performance na hora de carregarmos o DW.

O comando seguinte criará um procedimento para a carga da `D_Data` , desde o Data Stage até o Data Warehouse:

```
USE DS
GO


Create procedure [dbo].[Carrega_D_Data]
as


begin try
    -- Apaga os dados da D_Data no Stage:
    truncate table [DS]..[D_Data]


    -- Insere os dados na D_Data no Stage:
    insert into [dbo].[D_Data]
```

```
    Select Distinct
    Cast(DataVenda as date) as Data,
    right('00'+ cast(Day(DataVenda)as varchar(2)),2) as Dia,
    right('00'+ cast(Month(DataVenda)as varchar(2)),2) as Mes,
    Year(DataVenda) as Ano
    from [DS]..[TbImp_Vendas]


    -- Carrega os dados no DW:
    insert into [DW]..[D_Data]
    Select
    ds.data,
    ds.Dia,
    ds.Mes,
    ds.Ano
    from [DS]..[D_Data] as ds left join [DW]..[D_Data] as dw
    on ds.Data = dw.Data
    where dw.Data is null


    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Data','S', 'Carga de D_Data com sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Data','F', 'Erro ao carregar D_Data.')
end catch
```

Algumas observações:

- Sempre apagamos a tabela do DS para iniciar uma carga sem resquícios de cargas anteriores.
- A transformação e (quando houver) validação dos dados ocorrem na inserção na tabela do DS. Quando os dados forem ser inseridos no DW, já deverão estar ok.
- Quando fazemos `left join` do DS com o DW e incluímos a condição `where` de que uma coluna do DW seja nula, estamos na verdade efetuando uma carga incremental! Ou seja, vamos buscar todos os dados do DS desde que a igualdade dele no DW seja nula.

Execute o procedimento `Carrega_D_Data` duas vezes seguidas! Note que, na primeira, seu retorno é:

```
SQLQuery8.sql - WI...Administrador (53))*
Use DS
GO

Exec [dbo].[Carrega_D_Data]

100 %
Mensagens

(181 linha(s) afetadas)

(181 linha(s) afetadas)

(1 linha(s) afetadas)
```

Ou seja, foram 181 linhas inseridas na tabela `D_Data` do Stage, depois 181 linhas inseridas no DW (afinal, a Dimensão `D_Data` do DW estava vazia) e mais uma linha inserida na tabela de Log! Na segunda vez, o retorno será:

```
SQLQuery8.sql - WI...Administrador (53))*
Use DS
GO

Exec [dbo].[Carrega_D_Data]

100 %
Mensagens

(181 linha(s) afetadas)

(0 linha(s) afetadas)

(1 linha(s) afetadas)
```

Ou seja, foram 181 linhas inseridas na tabela `D_Data` do Stage, depois **zero** linhas inseridas no DW (afinal, a Dimensão `D_Data` do DW agora contém todas as datas que temos na tabela `TbImp`) e mais uma linha inserida na tabela de Log! A carga da `D_Data` foi concluída com sucesso!

4.5 CARREGANDO A DIMENSÃO CLIENTE

Feita a carga dos dados temporais, vamos criar a carga da dimensão `Cliente` . Das dimensões, ela é a mais simples de carregar, por isso é um bom começo para avaliarmos o conceito da chave artificial que veremos à seguir.

De forma análoga à carga anterior, criaremos a tabela que conterá os dados de Cliente no DataStage, usando o comando de criação igual ao do DW:

```
USE DS
GO


CREATE TABLE [dbo].[D_Cliente](
    [Cod_Cliente] [varchar](10) NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [Email] [varchar](50) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
 )
 GO
 create index IX_Cod_Cliente on DS..D_Cliente (Cod_Cliente)
 GO
 create index IX_Cod_Cliente on DW..D_Cliente (Cod_Cliente)
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios:

- Teremos um `Id_Cliente` que não existe no arquivo original. É a chave artificial que, como vimos anteriormente, recebe o nome de Surrogate Key. Essa

chave será criada de forma incremental. Note que ela não existe no DS. Por ser incremental, ela apenas existirá no destino do DW (por isso as surrogate foram criadas como `Identity`).

- O código do Cliente, seu nome e seu e-mail serão carregados e atrelados a essa surrogate.
- Teremos de colocar a fonte do dado e a data em que ele entrou para nossa Base, como recurso de Data Lineage.
- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

Diferentemente da Data, é importante que o código do cliente seja indexado tanto no DataStage quanto no DataWarehouse, e não uma Primary Key (que não existirá na tabela de Stage). Isso garantirá a performance da carga, uma vez que precisamos verificar se trata-se ou não de um registro novo justamente por esse campo.

Quando temos uma chave primária, temos automaticamente um índice (na verdade, o tipo de índice mais performático, o chamado Índice Cluster). Quando não estamos usando uma chave primária para a pesquisa, como é o caso do código do cliente, temos de criar manualmente o índice para tornar a consulta mais rápida (e quando criamos um índice que não é a chave primária, temos o índice não-cluster, menos performático, mas ainda muito melhor do que não ter índice algum).

O comando seguinte vai criar um procedimento para a carga da `D_Cliente` , desde o Data Stage até o Data Warehouse:

```
USE [DS]
GO


Create procedure [dbo].[Carrega_D_Cliente]
as
```

```
begin try
    -- Apaga os dados da D_Cliente no Stage:
    truncate table [DS]..[D_Cliente]


    -- Insere os dados na D_Cliente no Stage:
    insert into [dbo].[D_Cliente]
    Select Distinct
    CodCliente     as [Cod_Cliente],
    NomeCliente     as [Nome],
    EmailCliente as [Email],
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas]


    -- Carrega os dados no DW:
    insert into [DW]..[D_Cliente]
    Select
    ds.Cod_Cliente,
    ds.Nome,
    ds.Email,
    ds.LinData,
    ds.LinOrig
    from [DS]..[D_Cliente] as ds left join [DW]..[D_Cliente] as dw
    on ds.Cod_Cliente = dw.Cod_Cliente
    where dw.ID_Cliente is null


    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Cliente','S', 'Carga de D_Cliente com sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Cliente','F', 'Erro ao carregar D_Cliente.')
end catch
```

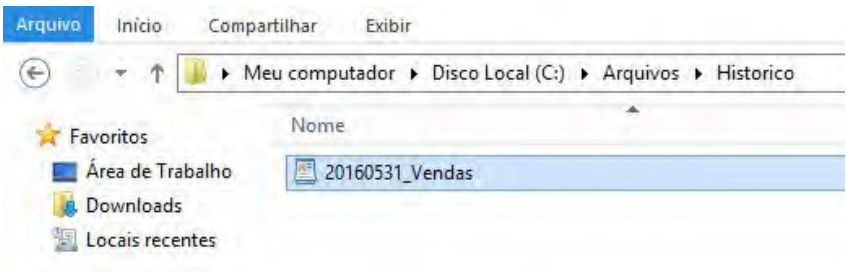
Mais uma vez, como na carga de Data, se executarmos essa procedure duas vezes seguidas, teremos os registros carregados na primeira execução e nenhum carregado na segunda (pois os dados não podem se repetir).

Verifique conforme a figura a seguir que o `Id_Cliente`, nossa surrogate, foi devidamente preenchida na base do DW, mesmo sem ter sido mencionada no nosso procedimento de carga. Isso se deve ao `Identity` que determinamos. Ele carrega automaticamente a coluna com um numerador sequencial.

Resultados | Mensagens

	Id_Cliente	Cod_Cliente	Nome	Email	LinData	LinOrig
1	1	AW00011000	Jon Yang	jon24@adventure-works.com	2016-05-31	Arquivo de Vendas
2	2	AW00011001	Eugene Huang	eugene10@adventure-works.com	2016-05-31	Arquivo de Vendas
3	3	AW00011002	Ruben Torres	ruben35@adventure-works.com	2016-05-31	Arquivo de Vendas
4	4	AW00011003	Christy Zhu	christy12@adventure-works.com	2016-05-31	Arquivo de Vendas
5	5	AW00011004	Elizabeth Johnson	elizabeth5@adventure-works.com	2016-05-31	Arquivo de Vendas
6	6	AW00011005	Julio Ruiz	julio1@adventure-works.com	2016-05-31	Arquivo de Vendas
7	7	AW00011006	Janet Alvarez	janet9@adventure-works.com	2016-05-31	Arquivo de Vendas
8	8	AW00011007	Marco Mehta	marco14@adventure-works.com	2016-05-31	Arquivo de Vendas
9	9	AW00011008	Rob Verhoff	rob4@adventure-works.com	2016-05-31	Arquivo de Vendas
10	10	AW00011009	Shannon Carlson	shannon38@adventure-works.com	2016-05-31	Arquivo de Vendas
11	11	AW00011010	Jacquelyn Suarez	jacquelyn20@adventure-works.com	2016-05-31	Arquivo de Vendas
12	12	AW00011011	Curtis Lu	curtis9@adventure-works.com	2016-05-31	Arquivo de Vendas
13	13	AW00011017	Shannon Wang	shannon1@adventure-works.com	2016-05-31	Arquivo de Vendas
14	14	AW00011018	Clarence Rai	clarence32@adventure-works.com	2016-05-31	Arquivo de Vendas
15	15	AW00011025	Alejandro Beck	alejandro45@adventure-works.com	2016-05-31	Arquivo de Vendas
16	16	AW00011026	Harold Sai	harold3@adventure-works.com	2016-05-31	Arquivo de Vendas

Também temos quando ele entrou no nosso DW e de onde ele veio para cada um dos valores carregados! Isso poderá ser de grande valia quando formos confrontar os dados do BI com os apresentados pelas demais fontes e em auditorias futuras. Como temos a origem e a data, podemos facilmente acessar a pasta `Historico` dos nossos arquivos e achar o arquivo que originou a informação pela data dele, ou seja, arquivo de Vendas do dia `31.05.2016`:



4.6 CARREGANDO A DIMENSÃO GEOGRAFIA

Dado que temos a Data e o Cliente, que são as cargas mais simples, vamos passar para um movimento mais "complexo". A ideia da dimensão Geografia é de carregarmos todas as tabelas em um único procedimento.

Novamente, sinta-se à vontade para discordar de mim e carregar uma em cada procedimento! A diferença será apenas a de termos mais ou menos procedimentos armazenados. Porém, os códigos em cada um e a ordem de execução serão exatamente iguais por uma imposição das chaves estrangeiras do DW. Eu opto por mantê-los todos juntos por uma simples questão de simplificar a execução (um único comando por dimensão).

Assim sendo, vamos construir as tabelas do nosso DS que receberão os dados geográficos bem como os índices para mantermos uma boa performance no processo de carga. Seguindo o nosso modelo, serão três tabelas:

```
USE [DS]
GO


CREATE TABLE [dbo].[D_GrupoGeografico]
    (
    [Nome] [varchar](50) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
    )
GO
create index IX_D_GrupoGeografico_nome on DS..D_GrupoGeografico ([
Nome])
create index IX_D_GrupoGeografico_nome on DW..D_GrupoGeografico ([
Nome])
GO


CREATE TABLE [dbo].[D_Pais]
    (
    [Id_GrupoGeo] [int] NOT NULL,
```

```
    [Sigla] [char](2) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
    )
GO
create index IX_D_PaisIdGrupo on DS..D_Pais ([Id_GrupoGeo])
create index IX_D_PaisSigla      on DS..D_Pais ([Sigla])
create index IX_D_PaisSigla      on DW..D_Pais ([Sigla])


GO


CREATE TABLE [dbo].[D_RegiaoVendas]
    (
    [Id_Pais] [int] NOT NULL,
    [Nome] [varchar](20) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
    )
GO
create index IX_D_RegiaoIdPais on DS..D_RegiaoVendas ([Id_Pais])
create index IX_D_RegiaoNome   on DS..D_RegiaoVendas ([Nome])
create index IX_D_RegiaoNome   on DW..D_RegiaoVendas ([Nome])


GO
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios:

- Teremos de carregar os dados na ordem do menos granular para o mais granular, pois temos dependência dos registros. Ou seja, para inserir uma região de vendas, ela deve pertencer a um País previamente carregado! Essa ordenação fará com que carreguemos o DS e o DW para cada uma das tabelas para depois seguir para a próxima, até finalizarmos!
- Da mesma forma que a `D_Cliente` , a chave de cada tabela será artificialmente criada por um autonumerador, a nossa Surrogate Key.
- Teremos de colocar a fonte do dado e a data em que ele

entrou para nossa base, como recurso de Lineage para cada uma das tabelas.

- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

Note no comando que indexamos os nomes (por onde as buscas ocorrerão) e as chaves das tabelas "Pai". Essas últimas não foram indexadas no DW, porque já fizemos isso quando criamos as tabelas dele. Uma das regras da boa performance é que as chaves estrangeiras sejam sempre indexadas!

O comando criará um procedimento para a carga das tabelas de Geografia, desde o Data Stage até o Data Warehouse:

```
USE [DS]
GO


Create  procedure [dbo].[Carrega_D_Geografia]
as


begin try
    -- Apaga os dados das tabelas no Stage:
    truncate table [DS]..[D_GrupoGeografico]
    truncate table [DS]..[D_Pais]
    truncate table [DS]..[D_RegiaoVendas]


    ------------------------------------------------------------
-------------------
    ------------------------------------------------------------
-------------------
    -- Insere os dados na D_GrupoGeografico no Stage:
    insert into [dbo].[D_GrupoGeografico]
    Select Distinct
    GrupoGeografico,
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas]

    -- Carrega os dados no DW:
```

```
    insert into [DW]..[D_GrupoGeografico]
    Select Distinct
    ds.Nome,
    ds.LinData,
    ds.LinOrig
    from [DS]..[D_GrupoGeografico] as ds left join DW..D_GrupoGeog
rafico as dw
    on ds.[Nome] =dw.[Nome]
    Where dw.[Id_GrupoGeo] is null


    ------------------------------------------------------------
-------------------
    ------------------------------------------------------------
-------------------
    -- Insere os dados na D_Pais do Stage:
    insert into [dbo].[D_Pais]
    Select Distinct
    dw.Id_GrupoGeo,
    ds.Pais,
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas] as ds inner join [DW]..[D_GrupoGeogr
afico] as dw
    on ds.GrupoGeografico = dw.Nome


    -- Carrega os dados no DW:
    insert into [DW]..[D_Pais]
    select
    ds.[Id_GrupoGeo],
    ds.[Sigla],
    ds.[LinData],
    ds.[LinOrig]
    from [DS]..[D_Pais] as ds left join [DW]..[D_Pais] as dw
     on ds.Id_GrupoGeo  = dw.Id_GrupoGeo
    and ds.Sigla        = dw.Sigla
    Where dw.Id_Pais is null


    ------------------------------------------------------------
-------------------
    ------------------------------------------------------------
-------------------
    -- Insere os dados na D_RegiaoVendas do Stage:
    insert into [dbo].[D_RegiaoVendas]
    select Distinct
    dw.Id_Pais,
```

```
    ds.RegiaoVendas,
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas] as ds inner join [DW]..[D_Pais] as d
w
    on ds.Pais = dw.[Sigla]


    -- Carrega os dados no DW
    insert into [DW]..[D_RegiaoVendas]
    Select
    ds.[Id_Pais],
    ds.[Nome],
    ds.[LinData],
    ds.[LinOrig]
    from [DS]..[D_RegiaoVendas] as ds left join [DW]..[D_RegiaoVen
das] as dw
     on ds.Id_Pais = dw.Id_Pais
    and ds.Nome    = dw.Nome
    where dw.Id_RegiaoVendas is null

    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Geografia','S', 'Carga das tabelas de Geografia com sucesso
.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Geografia','F', 'Erro ao carregar tabelas de Geografia.')
end catch
```

Esse processo de carga se mostra um pouco mais complexo apenas por termos de capturar o valor da surrogate no DW da tabela Pai antes de carregarmos os dados da tabela Filho. Esse passo que fazemos para preencher os dados ainda no Stage garante que teremos a tabela Filho com a devida surrogate quando a enviarmos para o DW. A indexação se faz necessária por conta das cargas do dia a dia que podem ter muitos e muitos registros, mesmo tratando-se de dimensões (que dificilmente superam as centenas de registros).

O fato de efetuarmos três cargas em um mesmo procedimento, mais uma vez, é apenas para centralizarmos a codificação de um mesmo "tema". Se você achar melhor dividi-la em três procedimentos distintos por achar que unificados o procedimento fica mais complexo, não há nenhum impedimento.

Para entender o resultado, execute a `Carrega_D_Geografia` e, em seguida, rode o comando:

```
select
a.[Nome] as RegGeo,
b.[Sigla] as Pais,
c.[Nome] as RegVendas
from [D_GrupoGeografico] as a inner join [D_Pais] as b
on a.[Id_GrupoGeo] = b.[Id_GrupoGeo]
inner join [dbo].[D_RegiaoVendas] as c
on b.[Id_Pais] = c.[Id_Pais]
```

Unindo as tabelas por suas chaves, tem-se a estrutura geográfica completa:

Resultados | Mensagens

	RegGeo	Pais	RegVendas
1	Europe	DE	Germany
2	Europe	FR	France
3	Europe	GB	United Kingdom
4	North America	CA	Canada
5	North America	US	Central
6	North America	US	Northeast
7	North America	US	Northwest
8	North America	US	Southeast
9	North America	US	Southwest
10	Pacific	AU	Australia

Quando usarmos a Região Vendas (informação mais granular) nas Fatos, poderemos saber também o País e a Região Geográfica (informações menos granulares)!

4.7 CARREGANDO A DIMENSÃO FUNCIONÁRIO

Como vimos no projeto do DW, os nossos vendedores serão postos em uma dimensão do tipo *parent-child*. Esse conceito pode ser usado em todas as dimensões que são organizadas de forma hierárquica, como pastas na árvore de diretórios de nossos computadores. Podem ser dimensões de Organograma, como a nossa Funcionário, Plano de Contas contábeis e gerenciais etc.

O interessante desse conceito é que ele pode formar uma árvore hierárquica infinita e complexa de forma extremamente simples: basta informar no registro, qual o `Id` do registro superior. E a carga é tão simples quanto o conceito!

Primeiramente, vamos criar a tabela `D_Funcionario` no nosso DS e os respectivos índices:

```
USE [DS]
GO


CREATE TABLE [dbo].[D_Funcionario](
    [Nome] [varchar](50) NOT NULL,
    [Login] [varchar](50) NOT NULL,
    [Id_Chefe] [int] NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
    )
create index IX_FuncionarioLogin on [DS]..[D_Funcionario]([Login])
create index IX_FuncionarioLogin on [DW]..[D_Funcionario]([Login])
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios:

- Teremos de carregar os dados dos funcionários, deixando a informação de Chefe em branco. Isso porque os "chefes" deverão já estar carregados e ter assim recebido sua surrogate, a fim de que possamos

atribuí-la a cada funcionário.

- Precisamos fazer um "update" na coluna `Id_Chefe` que está em branco para vincular com o respectivo chefe para cada funcionário, deixando `null` se ele não tiver chefe.
- Teremos de colocar a fonte do dado e a data em que ele entrou para nossa base, como recurso de Lineage para cada registro.
- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

O comando a seguir criará um procedimento para a carga da tabela de Funcionário, desde o Data Stage até o Data Warehouse:

```
USE [DS]
GO


Create procedure [dbo].[Carrega_D_Funcionario]
as


begin try
    -- Apaga os dados da D_Funcionario no Stage:
    truncate table [DS]..[D_Funcionario]


    -- Insere os dados na D_Funcionario no Stage:
    insert into [dbo].[D_Funcionario]
    ([Nome],[Login],[LinData],[LinOrig])
    Select Distinct
    VendedorNome,
    VendedorLogin,
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas]


    -- Carrega os dados no DW:
```

```
    insert into [DW]..[D_Funcionario]
    Select
    ds.[Nome],
    ds.[Login],
    ds.[Id_Chefe],
    ds.[LinData],
    ds.[LinOrig]
    from [DS]..[D_Funcionario] as ds left join [DW]..[D_Funcionari
o] as dw
    on ds.[Login] = dw.[Login]
    where dw.Id_Funcionario is null


    -- Atualiza o Id_Chefe no DW:
    declare @Id_funcionario int
    declare @LoginFuncionario varchar(50)
    declare @NomeFuncionario varchar(50)
    declare @NomeChefe varchar(50)
    declare @Id_Chefe int


    declare cur_AtualizaChefe cursor for
        Select Id_Funcionario, Login, Nome from DW..D_Funcionario
    Open cur_AtualizaChefe
    fetch next from cur_AtualizaChefe into @Id_funcionario, @Login
Funcionario, @NomeFuncionario
    while @@FETCH_STATUS = 0
        Begin
        -- determina o nome do Chefe:
            Set @NomeChefe = (Select Distinct VendedorChefeNome
                              from [DS]..[TbImp_Vendas] as b
                              where b.VendedorLogin = @LoginFuncion
ario
                              )
        -- Determina o ID do Chefe:
            Set @Id_Chefe = (Select Id_Funcionario
                              from [DW]..[D_Funcionario]as b
                              where b.Nome = @NomeChefe
                              )
        -- Determina o ID do Chefe:
            If @NomeChefe != @NomeFuncionario -- A pessoa não é o
prorpio chefe (quem não tem chefe)
                Begin
                    update [DW]..[D_Funcionario]
                    Set Id_Chefe = @Id_Chefe
                    where Id_Funcionario = @Id_funcionario
                End
        fetch next from cur_AtualizaChefe into @Id_funcionario,@Lo
```

```
ginFuncionario,@NomeFuncionario
        end
    close cur_AtualizaChefe
    deallocate cur_AtualizaChefe


    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Funcionario','S', 'Carga de D_Funcionario com sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Funcionario','F', 'Erro ao carregar D_Funcionario.')
end catch
```

O resultado dessa carga será conforme a consulta:

Resultados | Mensagens

	Id_Funcionario	Nome	Login	Id_Chefe	LinData	LinOrig
1	1	Michael Blythe	adventure-works\michael9	13	2016-05-31	Arquivo de Vendas
2	2	Linda Mitchell	adventure-works\linda3	13	2016-05-31	Arquivo de Vendas
3	3	Lynn Tsoflias	adventure-works\lynn0	13	2016-05-31	Arquivo de Vendas
4	4	Rachel Valdez	adventure-works\rachel0	NULL	2016-05-31	Arquivo de Vendas
5	5	JosÚ Saraiva	adventure-works\josÚ1	13	2016-05-31	Arquivo de Vendas
6	6	David Campbell	adventure-works\david8	18	2016-05-31	Arquivo de Vendas
7	7	Jillian Carson	adventure-works\jillian0	18	2016-05-31	Arquivo de Vendas
8	8	Ranjit Varkey Chudukatil	adventure-works\ranjit0	7	2016-05-31	Arquivo de Vendas
9	9	Tsvi Reiter	adventure-works\tsvi0	10	2016-05-31	Arquivo de Vendas
10	10	Stephen Jiang	adventure-works\stephen0	1	2016-05-31	Arquivo de Vendas
11	11	Sistema	Sistema	NULL	2016-05-31	Arquivo de Vendas
12	12	Jae Pak	adventure-works\jae0	18	2016-05-31	Arquivo de Vendas

Note que o `Id_Chefe` ficou nulo para a Rachel Valdez e para Sistema. Ambos os usuários que não possuem chefes. Os demais possuem o `Id` do seu respectivo chefe!

4.8 CARREGANDO A DIMENSÃO PRODUTO

Essa talvez seja o processo mais complexo. Ainda assim, não teremos nenhuma grande dificuldade a não ser a de seguir um passo a passo correto para a carga dessa que é uma dimensão que pode

sofrer alterações ao longo do tempo.

Primeiramente, precisamos entender **qual** alteração que levaremos em consideração. No nosso exemplo, o que queremos é que o nome do produto seja tratado como alteração. Isto é, se um mesmo produto receber um novo nome, teremos de estar preparados para capturar essa alteração. Como vimos no desenho do DW, tem-se mais de uma forma de se resolver essa questão, mas partimos para a chamada 2SCD.

Primeiramente, vamos criar a tabela no DataStage a fim de receber os dados de produto da nossa origem com os seus índices:

```
USE [DS]
GO


CREATE TABLE [dbo].[D_Produto](
    [Cod_Produto] [varchar](20) NOT NULL,
    [Nome] [varchar](50) NOT NULL,
    [Tamanho] [varchar](5) NOT NULL,
    [Cor] [varchar](20) NOT NULL,
    [Ativo] [char](1) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL
    )
GO
create index IX_ProdutoNM on DS..D_Produto (nome)
create index IX_ProdutoNM on DW..D_Produto (nome)
create index IX_ProdutoCod on DS..D_Produto (Cod_Produto)
create index IX_ProdutoCod on DW..D_Produto (Cod_Produto)
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios:

- Teremos de carregar os dados de novos produtos normalmente, fazendo com que a coluna de `ATIVO` permaneça como `SIM`.
- Precisaremos fazer um update na coluna `NOME` se ocorrer de um produto receber um novo nome para um mesmo `Cod_Produto`.

- Teremos de colocar a fonte do dado e a data em que ele entrou para nossa base, como recurso de Lineage para cada registro.
- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

O seguinte comando vai criar um procedimento para a carga da tabela de Produto, desde o Data Stage até o Data Warehouse, atualizando o produto se necessário:

```
USE [DS]
GO


Create procedure [dbo].[Carrega_D_Produto]
as


begin try
    -- Apaga os dados da D_Produto no Stage:
    truncate table [DS]..[D_Produto]


    -- Insere os dados na D_Produto no Stage:
    insert into [dbo].[D_Produto]
    Select Distinct
    Cod_Produto     as [Cod_Produto],
    Produto         as [Nome],
    Tamanho         as [Tamanho],
    Cor             as [Cor],
    Ativo        = 1,
    Getdate() as [LinData],
    'Arquivo de Vendas' as [LinOrig]
    from [DS]..[TbImp_Vendas]


    -- Carrega os dados no DW com os novos produtos (novo COD_PROD
UTO):
    insert into DW..D_Produto
    Select
    ds.Cod_Produto,
    ds.Nome,
    ds.Tamanho,
```

```
    ds.Cor,
    ds.Ativo,
    ds.LinData,
    ds.LinOrig
    from [DS]..[D_Produto] as ds left join [DW]..[D_Produto] as dw
    on ds.Cod_Produto = dw.Cod_Produto
    where dw.Cod_Produto is null


    -- Atualiza os Produtos que vieram com nome diferente (mas que
 possuem o mesmo COD_PRODUTO):
    declare @CodProduto varchar(20)
    declare @NomeProduto varchar(50)
    declare @Tamanho varchar(5)
    declare @Cor varchar(20)
    declare @LinData date
    declare @LinOrigem Varchar(50)
    declare cur_AtualizaProduto cursor for
        Select ds.Cod_Produto, ds.Nome, ds.tamanho, ds.cor, ds.Lin
Data, ds.LinOrig
        from DS..D_Produto as ds left join DW..D_Produto as dw
        on ds.Cod_Produto = dw.Cod_produto
        where ds.Nome != dw.Nome
          and dw.Ativo = 1
    Open cur_AtualizaProduto
    fetch next from cur_AtualizaProduto into @CodProduto, @NomePro
duto, @Tamanho, @Cor, @LinData, @LinOrigem
    while @@FETCH_STATUS = 0
        Begin
            -- Inativa o Produto com nome antigo:
            update dw..D_Produto
            set Ativo = 0
            where Cod_Produto = @CodProduto


            -- insere o registro com os novos dados:
            insert into DW..D_Produto values (@CodProduto,@NomePro
duto,@Tamanho,@Cor,1,@LinData, @LinOrigem)


        fetch next from cur_AtualizaProduto into @CodProduto, @Nom
eProduto, @Tamanho, @Cor, @LinData, @LinOrigem
        end
    close cur_AtualizaProduto
    deallocate cur_AtualizaProduto


    -- Faz o log do processo:
```

```
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Produto','S', 'Carga de D_Produto com sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Produto','F', 'Erro ao carregar D_Produto.')
end catch
```

Ao carregarmos os produtos — por ser a primeira carga —, não teremos nenhuma alteração de nome, então serão 60 produtos carregados e uma linha inserida no Log. Quando formos efetuar a carga dos próximos arquivos (dos demais anos que existem na nossa origem), faremos uma alteração no nome de um dos produtos a fim de testar a 2SCD.

Com isso, temos todas as nossas dimensões carregadas!

4.9 CARREGANDO A FATO DE VENDA

Finalizada a carga das dimensões, vamos carregar nossa Fato, a primeira de duas que temos em nosso DW. Teremos que começar pela Fato de Venda, dado que os registros dela serão "pais" da tabela detalhada de Vendas, o que nos impossibilitaria de carregarmos a Fato Venda Detalhes primeiramente.

Antes de iniciarmos a criação da nossa tabela no Stage, vamos recordar o primeiro dos conceitos do DW. Nenhuma coluna pode aceitar `null`. Se um dado veio nulo, o processo de carga deve entender se é algo esperado ou um erro. Deve-se preencher o campo com uma informação genérica para que se saiba que aquele dado não foi carregado. Essa prática resolve muitos problemas de origens de dados incoerentes. Isso quer dizer que temos de impedir que um dado da Fato venha com um *nulo* ou que, pior ainda, deixe de ser carregado (tornando os números incorretos).

Para darmos esse tratamento, vamos inserir um registro genérico em cada uma das dimensões. Ou seja, se tiver ocorrido qualquer problema na carga de uma dimensão e não houver uma relação válida entre o Fato e essa Dimensão, um registro genérico será atribuído, a fim de se manter os valores coerentes.

Explicando melhor: imagine que, por um erro, um Vendedor não foi carregado na Dimensão. Quando formos carregar a Fato, precisaríamos deixar de fora todas as vendas dele (já que temos uma relação de chave estrangeira entre a Fato e as dimensões). Com isso, o total de vendas da nossa empresa estaria errado no BI! Para evitar esse problema, vamos colocar um vendedor "Genérico". Mesmo que não se tenha a informação correta de quem fez a venda, ela não vai deixar de constar em nossos números!

Para tanto, o script a seguir fará essa inserção. E teremos de colocar esse registro em cada nova dimensão conforme o DW for crescendo:

```
Use DW
GO


Insert into D_Cliente          values ('999999','Não Aplica', 'Não
 Aplica', Getdate(), 'Registro padrão inserido manualmente');
Insert into D_Data              values ('1900/01/01', '01','01','1
900');
Insert into D_Funcionario      values ('Não Aplica','Não Aplica',n
ull, Getdate(), 'Registro padrão inserido manualmente');
Insert into D_GrupoGeografico values ('Não Aplica', Getdate(), 'Re
gistro padrão inserido manualmente');
Insert into D_Pais              values ((select Id_GrupoGeo from D
_GrupoGeografico where Nome = 'Não Aplica'), 'XX', Getdate(), 'Reg
istro padrão inserido manualmente');
Insert into D_RegiaoVendas    values ((select Id_Pais from D_Pais
where Sigla = 'XX'), 'Não Aplica', Getdate(), 'Registro padrão ins
erido manualmente');
Insert into D_Produto          values ('999999','Não Aplica','NA',
'NA','1', Getdate(), 'Registro padrão inserido manualmente');
```

Devemos ter em mãos as chaves geradas para cada um desses

registros, a fim de colocá-las no processo de carga das Fatos. Assim, usaremos s função `Isnull` que ditará: se o valor da surrogate da dimensão vier nulo, traga o valor do genérico.

Dessa forma, nunca deixaremos de contar com todos os valores e, de quebra, poderemos verificar nas sumarizações os registros com esses valores para atuarmos nos erros de carga (se houver).

> **Cuidado**
>
> Coloque como código dos dados genéricos apenas informações que, **com certeza**, não existam na realidade. Por exemplo, usar o dia 01/01/2020 como genérica para a `D_Data` é um erro, pois existirão vendas reais nesse dia, e elas seriam misturadas com as vendas cujas datas não foram estipuladas. Use sempre informações bastante diferentes das possíveis (por exemplo, 01/01/1900 para a Data!)

Vamos então criar a estrutura no nosso DS. E como a Fato não receberá uma surrogate própria, ela será **exatamente** igual à estrutura do DW. Para isso, vamos rodar o comando:

```
USE [DS]
GO
CREATE TABLE [dbo].[F_Venda](
    [Data] [date] NOT NULL,
    [Nr_NF] [varchar](10) NOT NULL,
    [Id_Cliente] [int] NOT NULL,
    [Id_Funcionario] [int] NOT NULL,
    [Id_RegiaoVendas] [int] NOT NULL,
    [Vlr_Imposto] [decimal](18, 2) NOT NULL,
    [Vlr_Frete] [decimal](18, 2) NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_F_Venda] PRIMARY KEY CLUSTERED
        (
        [Data] ASC,
```

```
        [Nr_NF] ASC,
        [Id_Cliente] ASC,
        [Id_Funcionario] ASC,
        [Id_RegiaoVendas] ASC
        )
)
```

Agora faremos o comando de carga em si, que deverá responder pelos seguintes critérios: Selecionar todos os dados de venda;

- Atribuir a cada dimensão, sua surrogate, mesmo que seja a dos registros genéricos.
- Avaliar se o dado já existe ou não na Fato do DW e carregar apenas os novos registros.
- Teremos de colocar a fonte do dado e a data em que ele entrou para nossa base, como recurso de Lineage para cada registro.
- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

O comando vai criar um procedimento para a carga da tabela de Fato Vendas, desde o Data Stage até o Data Warehouse:

```
USE [DS]
GO


Create Procedure [dbo].[Carrega_F_Venda]
as


begin try
    -- Apaga os dados da F_Vendas no Stage:
        truncate table [DS]..[F_Venda]


    -- Insere os dados na F_Vendas no Stage:
        -- Determina os "Ids" dos Genéricos:
        declare @idclienteGenerico int = (Select Id_Cliente from D
W..D_Cliente where Cod_Cliente = '999999')
        declare @idFuncionarioGenerico int = (Select ID_Funcionari
```

```
o from DW..D_Funcionario where Login = 'Não Aplica')
        declare @idRegiaoGenerico int = (Select Id_RegiaoVendas fr
om DW..D_RegiaoVendas where Nome = 'Não Aplica')

        insert into DS..F_Venda
        Select
        Data,
        Nr_NF,
        Id_Cliente,
        Id_Funcionario,
        Id_RegiaoVendas,
        sum(Vlr_Imposto),
        sum(Vlr_Frete),
        Getdate() as [LinData],
        'Arquivo de Vendas' as [LinOrig]
        from(
            Select Distinct
            isnull(a.DataVenda,'1900/01/01') as Data,
            isnull (a.NrNf, '000000') as Nr_NF,
            isnull(c.Id_Cliente, @idclienteGenerico) as Id_Cliente
,
            isnull(d.Id_Funcionario, @idFuncionarioGenerico) as Id
_Funcionario,
            isnull(e.Id_RegiaoVendas, @idRegiaoGenerico) as Id_Reg
iaoVendas,
            cast(ImpTotal as decimal(18,2)) as Vlr_Imposto,
            cast(Frete as decimal(18,2)) as [Vlr_Frete]
            from DS..TbImp_Vendas as a Left join DW..D_Data as b
            on a.DataVenda = b.Data
            left join DW..D_Cliente as c
            on a.CodCliente = c.Cod_Cliente
            left join DW..D_Funcionario as d
            on a.VendedorLogin = d.Login
            left join DW..D_RegiaoVendas as e
            on a.RegiaoVendas = e.Nome
            ) as x
        group by
        Data,
        Nr_NF,
        Id_Cliente,
        Id_Funcionario,
        Id_RegiaoVendas


        -- Carrega os dados no DW:
        insert into DW..F_Venda
        Select
        ds.[Data],
```

```
        ds.[Nr_NF],
        ds.[Id_Cliente],
        ds.[Id_Funcionario],
        ds.[Id_RegiaoVendas],
        ds.[Vlr_Imposto],
        ds.[Vlr_Frete],
        ds.[LinData],
        ds.[LinOrig]
        from DS..F_Venda as ds left join DW..F_Venda as dw
        on  ds.Data                 = dw .Data
        and ds.Nr_NF            = dw.Nr_NF
        and ds.Id_Cliente        = dw.Id_Cliente
        and ds.Id_Funcionario    = dw.Id_Funcionario
        and ds.Id_RegiaoVendas  = dw.Id_RegiaoVendas
        where dw.Data is null


    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Fatos de Venda','S', 'Carga de F_VENDA com sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Fatos de Venda','F', 'Erro ao carregar F_VENDA.')
end catch
```

Algumas observações quanto a este código:

- Utilizamos variáveis para receber o ID dos dados genéricos. Por favor, nada de "chumbar" informação variável em código!
- Como os dados do que seria o cabeçalho da NF se repetem para cada item da nota (para cada produto), basta colocarmos o comando `Distinct` no começo do `Select` para trazermos um único registro por nota fiscal, que é o que queremos!

4.10 CARREGANDO A FATO DE DETALHE DE VENDA

Essa carga é extremamente parecida com a anterior! Basta adicionarmos os dados do Produto na nossa seleção, mas basicamente teremos o mesmo processo.

Primeiramente vamos criar a `F_VendaDetalhe` no nosso DataStage, também igual à sua contraparte no DW:

```
USE [DS]
GO


CREATE TABLE [dbo].[F_VendaDetalhe](
    [Data] [date] NOT NULL,
    [Nr_NF] [varchar](10) NOT NULL,
    [Id_Cliente] [int] NOT NULL,
    [Id_Funcionario] [int] NOT NULL,
    [Id_RegiaoVendas] [int] NOT NULL,
    [Id_Produto] [int] NOT NULL,
    [Vlr_Unitario] [decimal](18, 2) NOT NULL,
    [Qtd_Vendida] [int] NOT NULL,
    [LinData] [date] NOT NULL,
    [LinOrig] [varchar](50) NOT NULL,
 CONSTRAINT [PK_F_VendaDetalhe] PRIMARY KEY CLUSTERED
    (
        [Data] ASC,
        [Nr_NF] ASC,
        [Id_Cliente] ASC,
        [Id_Funcionario] ASC,
        [Id_RegiaoVendas] ASC,
        [Id_Produto] ASC
    )
)
```

Tal qual fizemos anteriormente, nosso processo de carga deverá:

- Selecionar todos os dados de venda;
- Atribuir a cada dimensão, sua surrogate, mesmo que seja a dos registros genéricos;
- Avaliar se o dado já existe ou não na Fato do DW, e carregar apenas os novos registros;
- Teremos de colocar a fonte do dado e a data em que ele entrou para nossa Base, como recurso de Lineage para

cada registro;

- E, como sempre, se ocorrer um problema, um log de erro deve ser inserido. Se não, um log de sucesso deve ser inserido.

Este comando criará um procedimento para a carga da tabela de Fato Vendas Detalhe, desde o Data Stage até o Data Warehouse:

```
USE [DS]
GO


Create Procedure [dbo].[Carrega_F_VendaDetalhe]
as


begin try
    -- Apaga os dados da F_VendaDetalhe no Stage:
        truncate table [DS]..[F_VendaDetalhe]


    -- Insere os dados na F_Vendas no Stage:
        -- Determina os "Ids" dos Genéricos:
        declare @idclienteGenerico int = (Select Id_Cliente from D
W..D_Cliente where Cod_Cliente = '999999')
        declare @idFuncionarioGenerico int = (Select ID_Funcionari
o from DW..D_Funcionario where Login = 'Não Aplica')
        declare @idRegiaoGenerico int = (Select Id_RegiaoVendas fr
om DW..D_RegiaoVendas where Nome = 'Não Aplica')
        declare @idProdutoGenerico int = (Select Id_Produto from D
W..D_Produto where Cod_Produto = '999999' and [Ativo] = '1')

        insert into DS..F_VendaDetalhe
        Select
        Data,
        Nr_NF,
        Id_Cliente,
        Id_Funcionario,
        Id_RegiaoVendas,
        Id_Produto,
        sum(Vlr_Unitario),
        sum(Qtd_Vendida),
        Getdate() as [LinData],
        'Arquivo de Vendas' as [LinOrig]
        from(
            Select Distinct
```

```
            isnull(a.DataVenda,'1900/01/01') as Data,
            isnull (a.NrNf, '000000') as Nr_NF,
            isnull(c.Id_Cliente, @idclienteGenerico) as Id_Cliente
,
            isnull(d.Id_Funcionario, @idFuncionarioGenerico) as Id
_Funcionario,
            isnull(e.Id_RegiaoVendas, @idRegiaoGenerico) as Id_Reg
iaoVendas,
            isnull(f.Id_Produto, @idProdutoGenerico) as Id_Produto
,
            isnull(cast(a.PrecoUnitario as decimal(18,2)),0) as Vl
r_Unitario,
            isnull(cast(a.Qtd as int),1) as Qtd_Vendida,
            Getdate() as [LinData],
            'Arquivo de Vendas' as [LinOrig]
            from DS..TbImp_Vendas as a Left join DW..D_Data as b
            on a.DataVenda = b.Data
            left join DW..D_Cliente as c
            on a.CodCliente = c.Cod_Cliente
            left join DW..D_Funcionario as d
            on a.VendedorLogin = d.Login
            left join DW..D_RegiaoVendas as e
            on a.RegiaoVendas = e.Nome
            left join DW..D_Produto as f
            on a.Cod_Produto = f.Cod_Produto
            where f.Ativo = '1'
            ) as x
        Group by
        Data,
        Nr_NF,
        Id_Cliente,
        Id_Funcionario,
        Id_RegiaoVendas,
        Id_Produto


        -- Carrega os dados no DW:
        insert into DW..F_VendaDetalhe
        Select
        ds.[Data],
        ds.[Nr_NF],
        ds.[Id_Cliente],
        ds.[Id_Funcionario],
        ds.[Id_RegiaoVendas],
        ds.[Id_Produto],
        ds.[Vlr_Unitario],
        ds.[Qtd_Vendida],
        ds.[LinData],
```

```
        ds.[LinOrig]
        from DS..F_VendaDetalhe as ds left join DW..F_VendaDetalhe
as dw
        on  ds.Data             = dw .Data
        and ds.Nr_NF             = dw.Nr_NF
        and ds.Id_Cliente         = dw.Id_Cliente
        and ds.Id_Funcionario     = dw.Id_Funcionario
        and ds.Id_RegiaoVendas   = dw.Id_RegiaoVendas
        and ds.Id_Produto         = dw.Id_Produto
        where dw.Data is null


    -- Faz o log do processo:
        insert into [dbo].[Adm_Log] values (newid(), getdate(), 'I
mporta Fatos de Detalhe de Venda','S', 'Carga de F_VENDADETALHE co
m sucesso.')


end try
begin catch
-- Faz o log do processo:
    insert into [dbo].[Adm_Log] values (newid(), getdate(), 'Impor
ta Fatos de Venda Detalhe','F', 'Erro ao carregar F_VENDADETALHE.'
)
end catch
```

Note a condição `Where` ao final do processo de carga do DataStage. Ela garante que os produtos carregados serão somente os que possuem a versão ativa. Sem esse comando, haveria duplicidade dos registros que já tivessem algum produto inativado pela 2SCD.

4.11 AGENDAMENTO E CARGA NO DIA A DIA

Criamos todas as procedures que compõe o processo de carga. Diariamente, ou na periodicidade desejada (lembre-se da latência que comentamos anteriormente), vamos executar esse processamento a fim de carregar os arquivos que forem gerados à partir dos transacionais.

Uma forma de facilitar a chamada a cada um dos processos de carga é criar uma procedure que vai fazer a execução de todas as outras, na ordem certa!

O comando seguinte criará a procedure "Workflow" que chamará cada uma das cargas na ordem correta. Quando formos criar os processos de agendamento, bastará executar essa única procedure e todo o processo será chamado:

```
USE DS
GO


create procedure Workflow
as


-- Carrega os dados do arquivo de Vendas:
exec [dbo].[Importa_Vendas]


-- Carrega as Dimensões:
exec [dbo].[Carrega_D_Data]
exec [dbo].[Carrega_D_Cliente]
exec [dbo].[Carrega_D_Geografia]
exec [dbo].[Carrega_D_Funcionario]
exec [dbo].[Carrega_D_Produto]


-- Carrega as Fatos:
exec [dbo].[Carrega_F_Venda]
exec [dbo].[Carrega_F_VendaDetalhe]
```

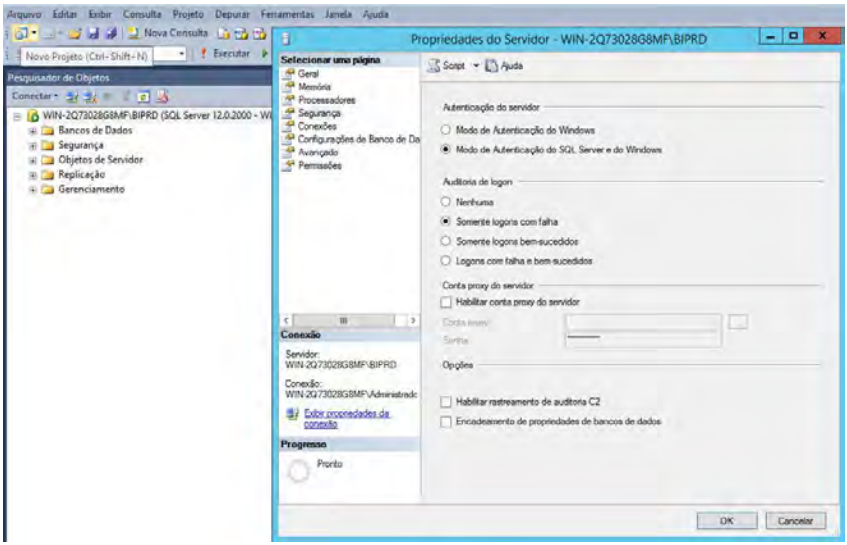
Se estivéssemos usando a versão paga do SQL Server, teríamos a nosso dispor a possibilidade de criar *Jobs*, que são rotinas agendadas. Eles seriam criados para executarem os processos de carga na hora correta. Como estamos usando a versão gratuita, não temos esse recurso disponível.

Mas essa falta é facilmente contornada com o uso do agendamento do próprio Windows! Com ele, poderemos criar a chamada para a execução da nossa procedure `Workflow` sem problemas! Só que, para isso, teremos de seguir alguns passos.

Primeiramente, vamos criar um usuário no nosso servidor de banco de dados que terá permissão para executar esse comando. Por

padrão (e fizemos a instalação padrão), o SQL só permite conexão de usuários autenticados pelo Windows. Como vamos usar um usuário do próprio SQL, precisaremos alterar essa configuração.

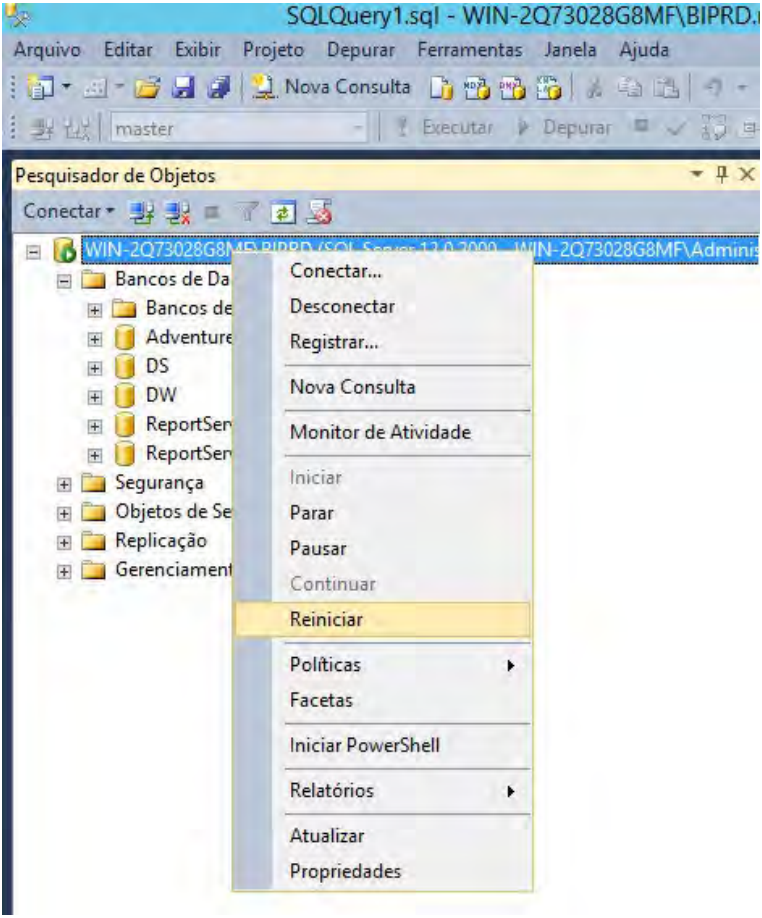
No Management Studio, clique com o botão direito na Instância e selecione `Propriedades`. Na aba `Segurança`, selecione `Modo de Autenticação do SQL Server e do Windows`, conforme:



Feito isso, o comando a seguir vai criar o usuário com permissão para acessar o DS e o DW, e para utilizar os comandos de `Bulk Insert`:

```
USE [master]
GO
CREATE LOGIN [BI_User]
WITH PASSWORD= 'P@ssw0rd',
     DEFAULT_LANGUAGE=[Português (Brasil)],
     CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
master..sp_addsrvrolemember @loginame = N'BI_User', @rolename = N'
sysadmin'
GO
```

Para que esse usuário de fato passe a ter acesso, a Instância do SQL Server Express deverá ser reiniciada para efetivar o modo de autenticação. Clique com o botão direito do mouse novamente na instância e selecione `Reiniciar`, dando `OK` para a mensagem de aviso.



Depois vamos criar um arquivo executável que disparará nossa procedure, o chamado `arquivo .bat`. Nele inseriremos uma linha de comando que fará a requisição para o SQL, a fim de executar nossa procedure.

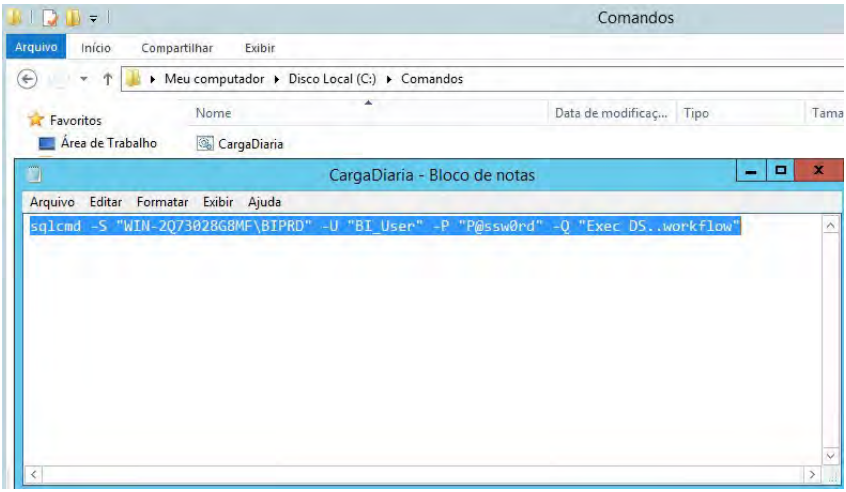
O SQL Server possui um command prompt chamado `sqlcmd` pelo qual podemos passar as credenciais de usuário (por isso criamos o `BI_User`) e os comandos que serão executados. No nosso caso, a procedure `Workflow` .

Para tanto, vamos criar uma pasta que conterá nossos `bats` . Podemos criar a `c:\Comandos` . Depois, abra um editor de texto (Notepad, Notepad++, Aton, qualquer um) e digite o comando:

```
sqlcmd -S "WIN-2Q73028G8MF\BIPRD" -U "BI_User" -P "P@ssw0rd" -Q "E
xec DS..workflow"
```

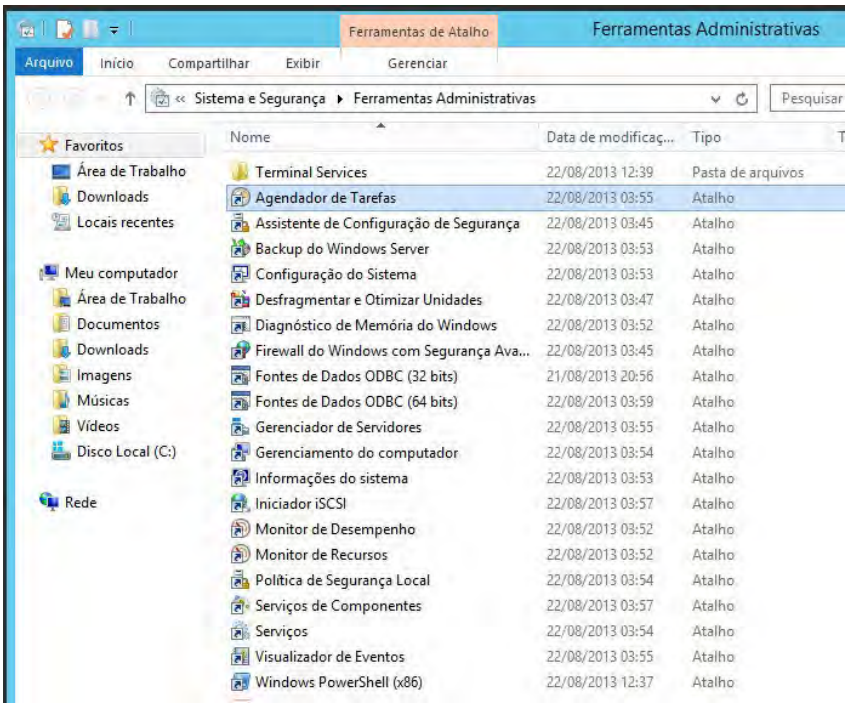
Detalhando o comando, temos que `-S` é o Servidor e instância (para obter corretamente essa informação, digite o comando `select @@SERVERNAME` no Management Studio), `-U"` é o comando para passar o usuário, `-P` passará a senha e `-Q` passará a query propriamente dita para o SQL.

Salve o arquivo com o nome `CargaDiaria.bat` dentro da pasta que criamos e pronto! Temos nosso comando finalizado:

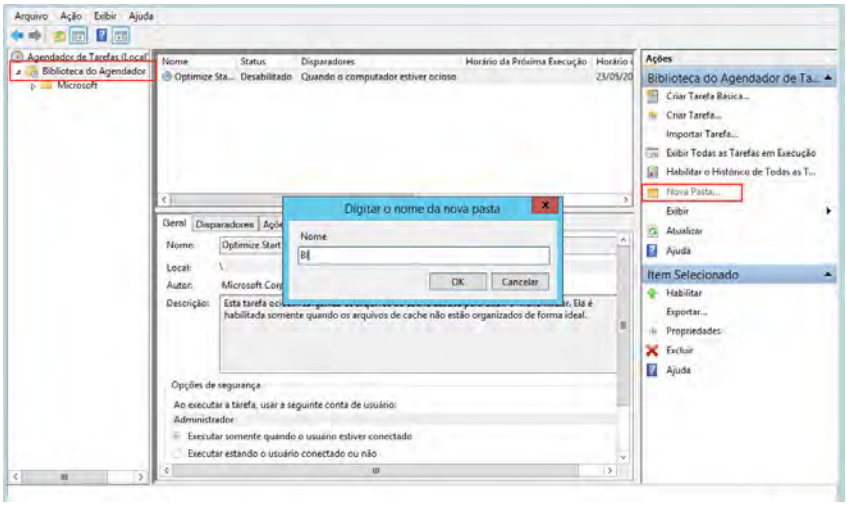


Para o agendamento propriamente dito, precisaremos entrar nas `Ferramentas Administrativas do Windows` e abrir `Agendador`

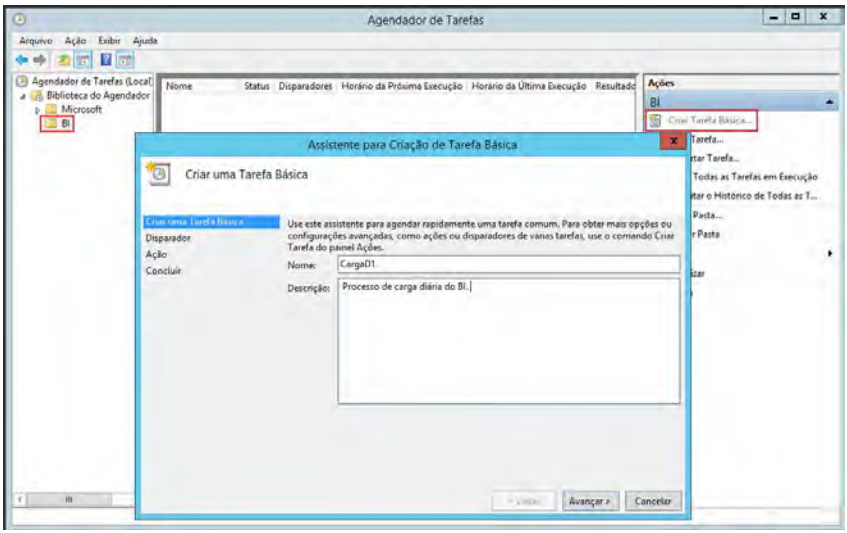
de Tarefas :



Abrindo o Agendador, clique em `Biblioteca do Agendador` e, ao lado direito, `Nova Pasta` . Nomeie a pasta como `BI` :



Clique na pasta `BI` recém-criada, e clique em `Criar Tarefa Básica`:

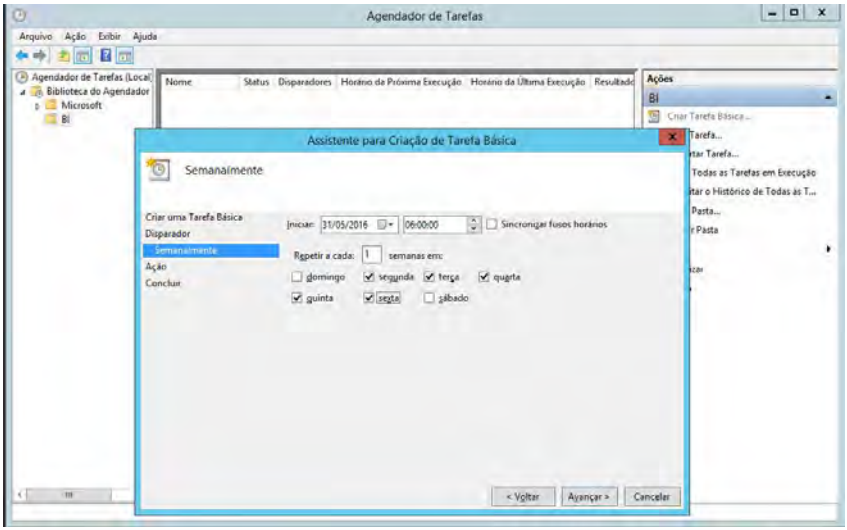


Podemos dar o nome de `CargaD1`, porque esse agendamento será responsável por executar as cargas que ocorrerão uma vez ao dia. Quando novos arquivos forem sendo disponibilizados, bastará

adicionar seus processos na procedure Workflow, sem necessidade de dar nenhuma manutenção nesse agendamento. Se precisarmos de cargas com latências diferentes, aí sim criaremos novos agendamentos com o nome referenciando a periodicidade, como `CargaH12` ou `CargaM1` (para dados carregados mensalmente), por exemplo.

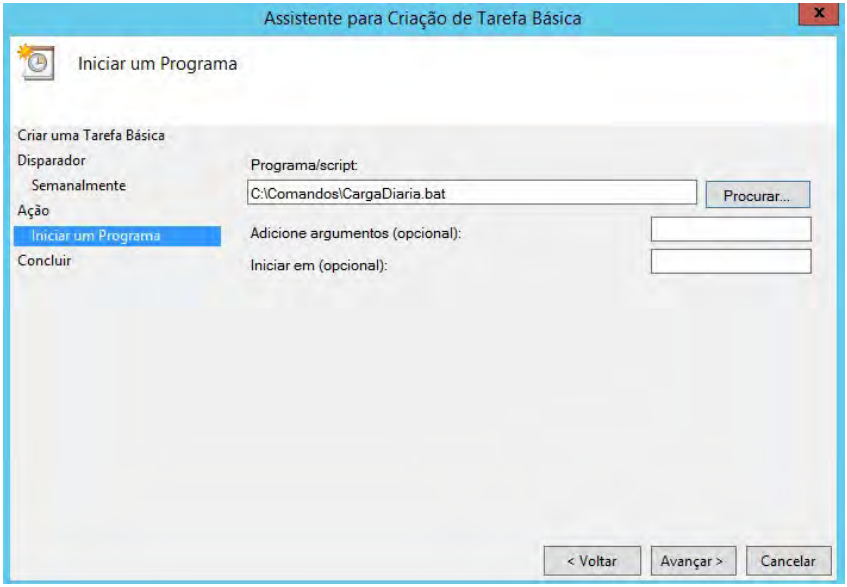
Clicando em `Avançar`, selecione o agendamento `Semanalmente` e vamos atribuir que esse procedimento ocorrerá às 6:00hs somente nos dias úteis. A periodicidade deverá atender ao seu negócio. Se a origem dos dados (no nosso exemplo, o arquivo `MassaDados.rpt`) for gerado todo dia, inclusive aos finais de semana, opte pela periodicidade `Diariamente`.

Mas para efeito do nosso cenário, vamos prosseguir com o agendamento nos dias úteis e, para tanto, configure selecionando os dias de segunda a sexta-feira:



Avançando, teremos as opções de `Iniciar um Programa`, `Enviar um Email` ou `Exibir uma Mensagem`. Opte por

`Iniciar um programa` e clique em `Avançar`. Na seleção `Procurar...`, selecione o `.bat` que fizemos:



Feito isso, basta avançar e concluir! Nosso agendamento estará pronto para ocorrer todos os dias úteis, ás 6:00hs.

4.12 REVISÃO E TESTES

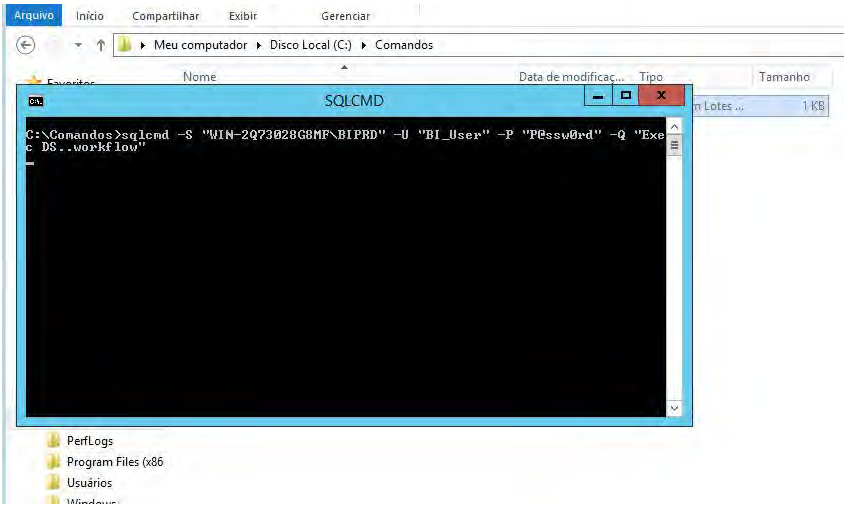
Para testarmos o processo todo, vamos criar mais um arquivo de dados. Se nos lembrarmos do capítulo do desenho de nosso DW, havíamos feito um script para a geração da massa de testes e colocado nele o filtro para obtermos apenas os dados de um determinado ano. Recuperando o referido script, altere a condição `where` para o ano de 2006 e repita o processo de geração do arquivo:

```
where year([OrderDate]) = 2006
```

Teremos então, novamente, um arquivo chamado

`MassaDados.rpt` no diretório `c:\Arquivos`.

Feito isso, vá até a pasta `c:\Comandos` onde deixamos nosso arquivo `.dat` que será executado todo dia às 6:00hs. Vamos, para efeito de testes, executá-lo manualmente. Clique duas vezes nele. Um prompt será exibido com o comando que havíamos declarado:



Assim que o processo terminar, o prompt fechará automaticamente. Nesse momento, teremos de ter o arquivo na pasta `c:\Arquivos\Historico` com a data de hoje e os dados carregados no BI! Execute o comando `select count(*) from F_VendaDetalhe`. Teremos passado de 5.149 registros (da primeira carga) para 24.496.

Finalmente, execute a geração de um arquivo com todos os dados disponíveis (apenas excluindo a cláusula `where` do comando de extração). Será um arquivo com 121.319 linhas (sendo os dois primeiros do cabeçalho). Não haverá problemas dos dados dos anos 2005 e 2006 estarem na massa. Nosso processo de carga é incremental, então os registros já carregados não serão duplicados.

Deixe o arquivo na pasta `c:\Arquivos` e espere até a manhã do próximo dia útil! O processo deve ter sido carregado, inclusive com os Logs de sucesso (ou falha!) na tabela `Adm_Log` :

Id_Log	Data	Passo	SucessoFalha	mensagem
0FB568B8-0D9F-4267-B575-3E72DCA0B65B	2016-06-01 06:00:01.613	Importa MassaDados.rpt	S	Arquivo importado com sucesso
3FBEF26D-56E8-41B6-BAB6-AE794C01C15C	2016-06-01 06:00:01.997	Importa Data	S	Carga de D_Data com sucesso.
6DE70362-9110-42BA-B108-07895B7C9A39	2016-06-01 06:00:03.217	Importa Cliente	S	Carga de D_Cliente com sucesso.
55C9C642-3B28-42B1-86F7-CB5058213D75	2016-06-01 06:00:03.797	Importa Geografia	S	Carga das tabelas de Geografia com sucesso.
BE9D9A1E-5ADF-40D9-A1A7-D2D7C9D4C60D	2016-06-01 06:00:04.767	Importa Funcionario	S	Carga de D_Funcionario com sucesso.
CAC0C8B9-1B64-4596-8DFF-9DCFAC84D570	2016-06-01 06:00:05.890	Importa Produto	S	Carga de D_Produto com sucesso.
636F10B4-E0D1-403C-B41B-DEAB74A5F2D7	2016-06-01 06:00:07.113	Importa Fatos de Venda	S	Carga de F_VENDA com sucesso.
EFB94EE2-DBF8-4DAE-B4A3-07993D12E254	2016-06-01 06:00:09.523	Importa Fatos de Detalhe de Venda	S	Carga de F_VENDADETALHE com sucesso.

4.13 CONCLUSÃO

Fizemos neste capítulo todo o processo de carga e agendamento dos dados de um arquivo texto para nosso DS, e posteriormente para nosso Data Warehouse. Criamos soluções para tratamento de erros e para dados nulos!

Nosso ETL está pronto para carregar dados todos os dias, alimentar nosso DW e possibilitar que consultemos as informações das mais variadas formas a fim de, finalmente, começarmos a analisar nossos fenômenos.

CAPÍTULO 5

EXPONDO AS INFORMAÇÕES

5.1 OPÇÕES DE CONSUMO DO BUSINESS INTELLIGENCE

Como vimos na descrição da plataforma de BI, existem diversas opções para consumirmos as informações existentes no nosso DW e, com o advento de novas tecnologias e de novas formas de análise, a tendência é que essas opções aumentem cada vez mais. O que temos de ter em mente é com qual forma de exibição atenderemos às necessidades de nossos usuários.

Se formos entregar as ferramentas que atendem aos níveis pessoais de Business Intelligence para os consumidores dos níveis corporativos (ou vice-versa), teremos um grande fracasso. Vale lembrar de que temos três tipos de consumo de BI:

- **Corporativo:** são as visões mais aglutinadas, os KPIs e cockpits de dados agrupados. Geralmente, são os diretores e gestores que consomem essas informações.
- **Departamental:** são os relatórios detalhados e as visões com pesquisas. São utilizadas pelos coordenadores e analistas que operam o dia a dia.
- **Pessoal:** são as planilhas dinâmicas que geram análises precisas de dados. São usados pelos especialistas (independente da hierarquia) que buscam novas

análises, novos fenômenos e relações.

O que cobriremos neste capítulo é a criação de soluções para cada tipo de consumo, nos atendo aos exemplos relativos aos dados de notas fiscais que importamos e às possibilidades com as ferramentas que escolhemos. Pela própria natureza de uma plataforma de BI, temos infinitas formas de apresentar e interpretar as informações, por isso mesmo o que pretendemos aqui é exemplificar o que se pode fazer e não gerar um guia sobre quais indicadores deveriam ser criados para uma análise de varejo.

Assim sendo, vamos criar um dashboard para nosso nível corporativo e um relatório detalhado para nosso nível departamental. Para o nível pessoal, criaremos uma visualização por meio de uma tabela dinâmica.

5.2 OBTENDO AS FERRAMENTAS E ADD-IN

Usaremos duas ferramentas para atendermos às demandas de uso da nossa plataforma de BI: o Reporting Service para atender às dashboards e aos relatórios, e o Excel com o SQL Server PowerPivot para atender às consultas dinâmicas e mesmo às dashboards.

O Reporting Service é de fato gratuito e foi instalado juntamente com o SQL Server. Ele é um servidor de relatórios (*server side*). Para criarmos os relatórios que ficarão disponíveis nele, precisaremos instalar uma ferramenta chamada SSDT (*SQL Server Data Tools*) que atualmente é componente do Visual Studio. Veremos a seguir como baixar e configurar tudo!

O Excel é um componente do Office e rodará nas estações dos usuários (*client side*). É certo que ele é um produto pago, mas entendo que já está presente e disponível em seu ambiente, e por isso o consideramos como parte integrante da solução gratuita. O

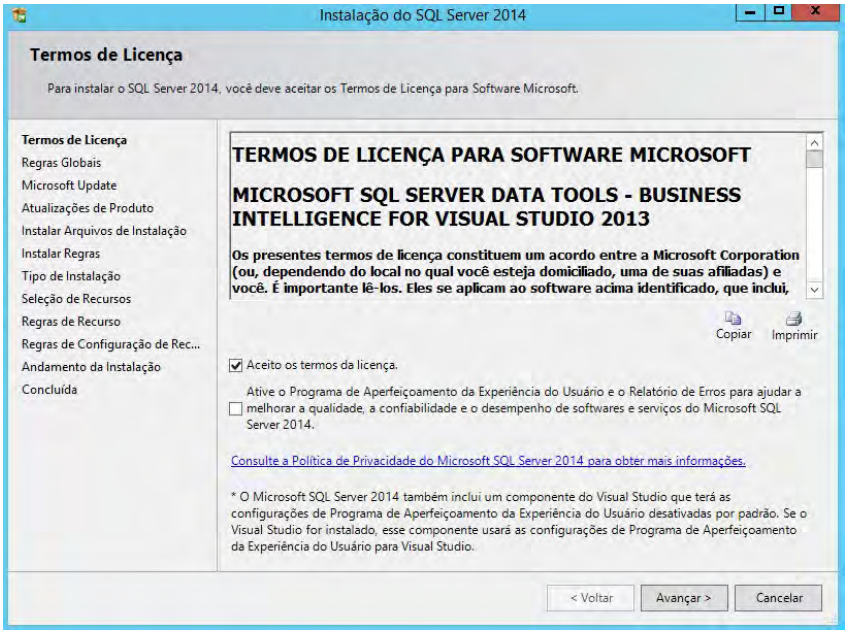
detalhe é para o *Add-in* PowerPivot que deve ser instalado ou habilitado em cada máquina cliente, como veremos logo a seguir.

Não vou entrar no mérito do uso de, por exemplo, o OpenOffice versus as ferramentas da Microsoft, porque não quero criar inimizades por aqui. Mas caso você não possua o Excel nas máquinas de seus usuários, avalie a possibilidade (estou apenas convidando, ok?) de adquirir uma assinatura do Office 365 (http://www.office365.com).

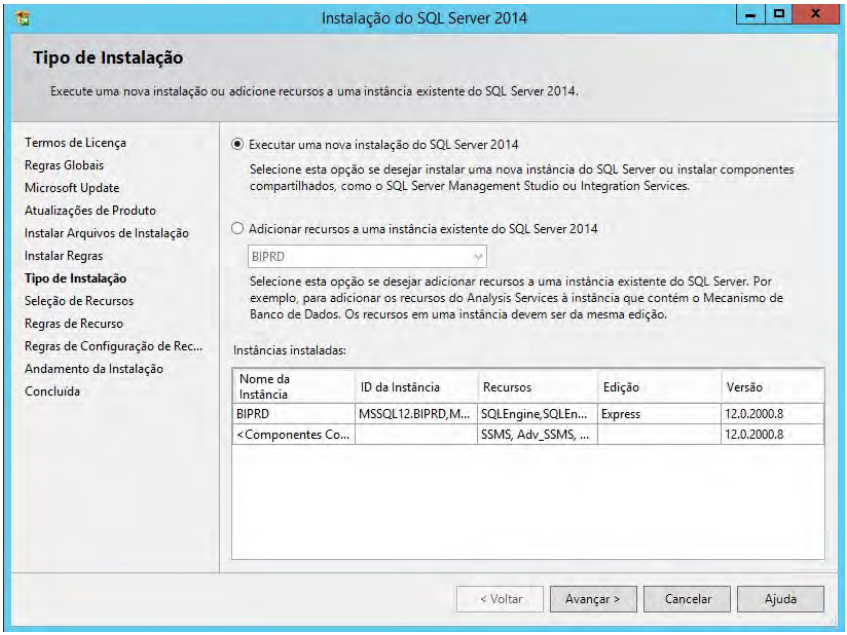
Para a criação dos relatórios que serão publicados no Reporting Service, precisaremos do SSDT. Nas versões anteriores, essa ferramenta já vinha integrada ao pacote de Ferramentas Administrativas da instalação do SQL Server, tal qual o Management Studio e o Profiler. Desde as últimas versões, contudo, a Microsoft decidiu por disponibilizá-la como um pacote à parte integrado ao Visual Studio.

O correto é instalar as ferramentas de desenvolvimento em outra máquina que não o servidor. Um PC com Windows 7 ou superior é o ideal. Cria-se os relatórios nas estações dos desenvolvedores para depois efetuar o upload para o servidor.

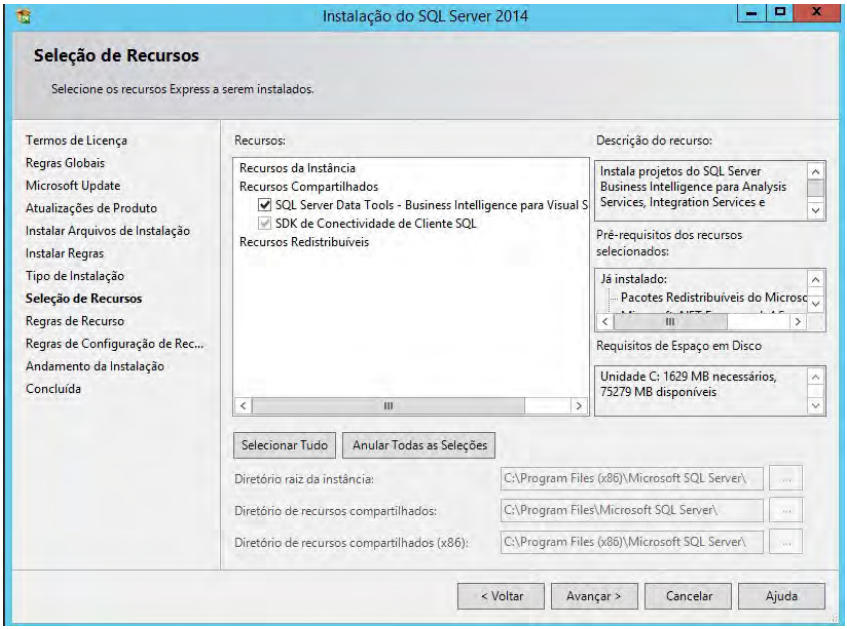
Primeiramente, temos de fazer o download dessa ferramenta que pode ser feito pelo link https://www.microsoft.com/pt-br/download/confirmation.aspx?id=42313. Depois de feito o download, clique duas vezes no arquivo `SSDTBI_x86_PTB.exe` (baixei a versão em português). Selecione um local para a extração dos arquivos, e aguarde a extração e execução automática do instalador:



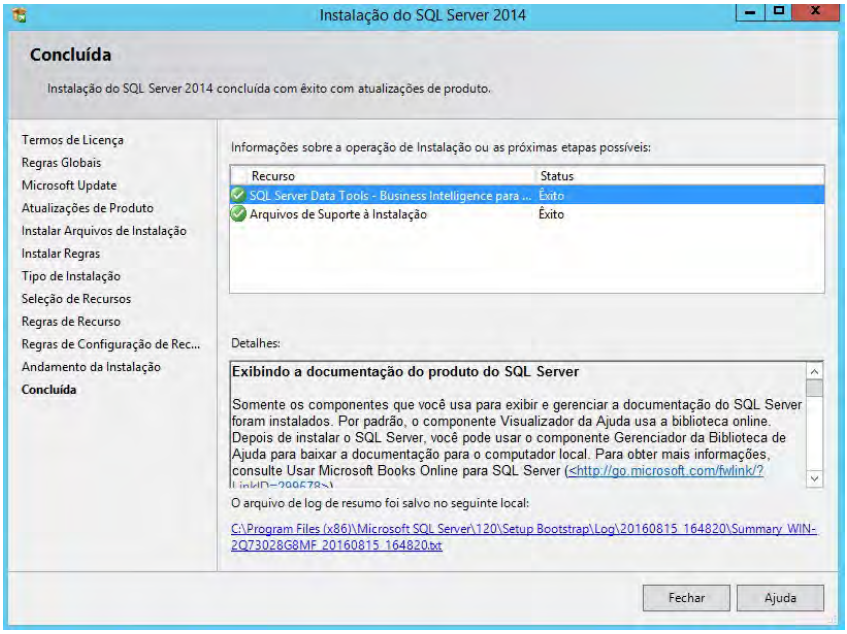
Aceite os termos de licença e clique em `Avançar`. Opte por usar o Microsoft Update e avance novamente. Opte por executar uma nova Instalação do SQL Server. Se optar por adicionar à instância existente, a instalação apresentará um erro.



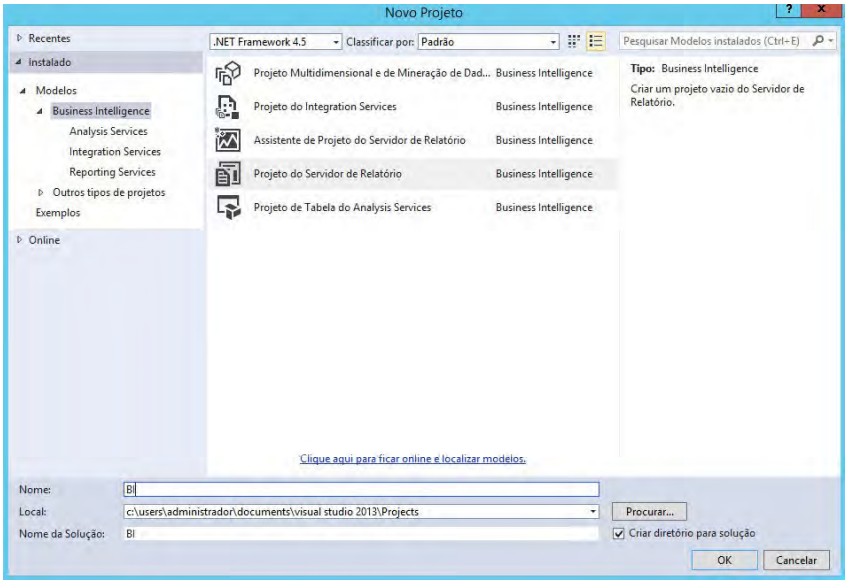
Selecione a opção do SQL Server Data Tools e clique em `Avançar` :



Se tudo correu bem, a mensagem de êxito deve ser exibida ao final do processo:



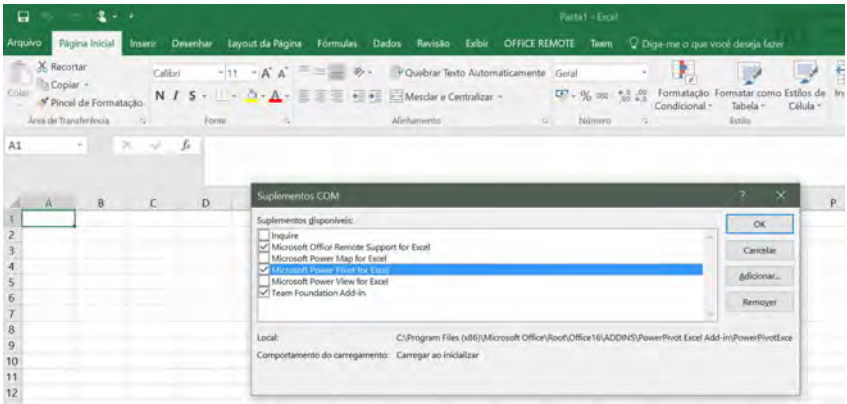
Em `Programas`, o Visual Studio 2013 poderá ser localizado. Ao ser iniciado, selecione `Novo Projeto` e note que abrirá a tela de opções de criação de projetos do Analysis Services, do Integration Services e do Reporting Services. Vamos iniciar um novo projeto de servidor de relatório. Podemos nomeá-lo de `BI` e deixar os arquivos no local padrão. Depois, basta clicar em `OK`:



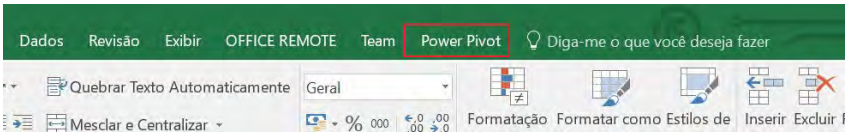
Conforme formos criando os relatórios e dashboards, comentaremos sobre os conceitos de uso do Visual Studio. Por hora, termos chegado até aqui indica que tudo está em ordem!

Feita a instalação do SSDT, vamos instalar ou habilitar o PowerPivot, que será usado nas estações dos usuários do BI. Esse add-in permitirá dar muito mais poder às consultas dinâmicas do Excel!

Para obter o PowerPivot para o Excel 2013 em diante, basta habilitá-lo. No Excel, vá em `Arquivo`, depois em `Opções` e depois em `Suplementos`. Na parte debaixo da janela, selecione `Suplementos COM` e clique em `ir`. Selecione o PowerPivot e clique em `OK`:



O Menu `PowerPivot` deve ter aparecido no seu Excel:



Havendo algum problema, para maiores auxílios, entre no site https://support.office.com/pt-br/article/Iniciar-o-suplemento-do-Power-Pivot-no-Microsoft-Excel-a891a66d-36e3-43fc-81e8-fc4798f39ea8?omkt=pt-BR&ui=pt-BR&rs=pt-BR&ad=BR.

Para o Excel 2010 ou 2007, deve-se efetuar o Download do Add-in, em https://www.microsoft.com/pt-BR/download/details.aspx?id=29074. Opte pela versão de 32 ou 64 bits (deve ser a mesma do Office instalado na máquina de quem vai utilizar a ferramenta):

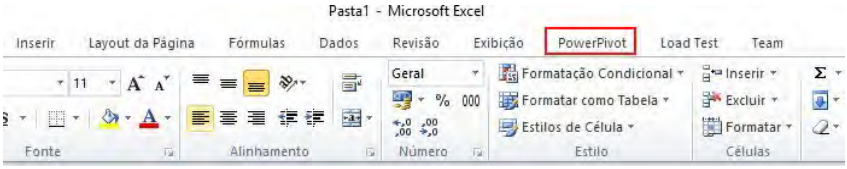
Escolha o download desejado

Nome do arquivo	Tamanho
1046\ReadMe_PowerPivot.htm	12 KB
1046\x64\PowerPivot_for_Excel_amd64.msi	129.9 MB
1046\x86\PowerPivot_for_Excel_x86.msi	98.5 MB

Vale lembrar de que o .Net Framework 4.0 ou superior é pré-requisito e deve estar também instalado em cada uma das máquinas dos usuários que farão uso do PowerPivot. Ele pode ser baixado (versão em português brasileiro) em https://www.microsoft.com/pt-br/download/details.aspx?id=30653.

Uma vez que todos os pré-requisitos tenham sido verificados e que se tenha o arquivo de instalação, basta executá-lo nas máquinas dos usuários. Clique duas vezes para executar o instalador, aceite os termos do contrato e clique em `Instalar` :

Feita a instalação, abra o Excel e nova aba deve estar visível. Caso contrário, podemos seguir os mesmos passos para habilitar a `PowerPivot` que fizemos para as versões mais novas. Fazendo isso, o menu aparecerá no seu Excel 2010:



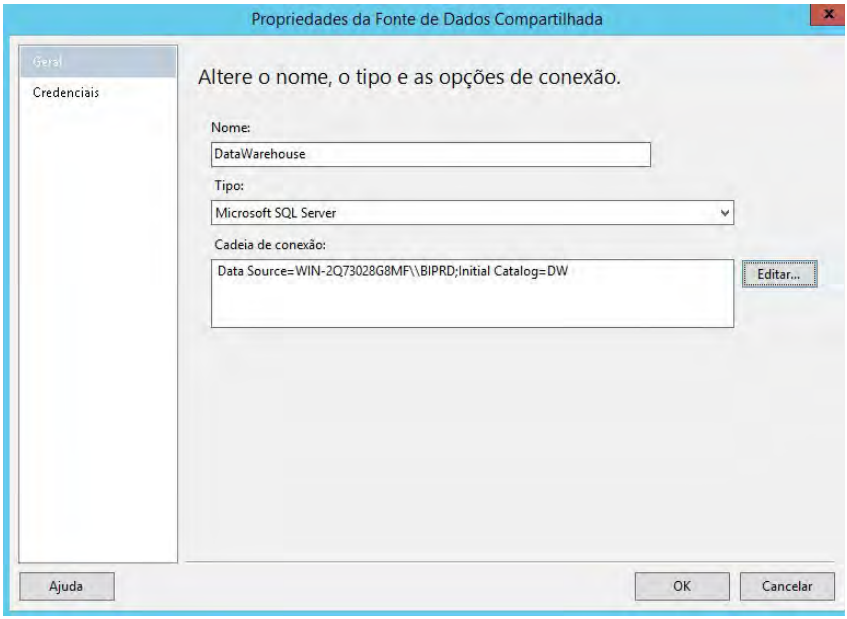
Quando formos criar nossas visões dinâmicas, explicarei um pouco sobre a PowerPivot. Novamente, se a aba do Excel está disponível, estamos com tudo certo!

5.3 CRIAÇÃO E PUBLICAÇÃO DO DASHBOARD

Vamos então desenvolver nosso dashboard que será disponibilizado para nossos consumidores do BI Corporativo. A ideia é a de criarmos alguns gráficos e indicadores que terão sentido entre si, dando aos nossos usuários um panorama geral de como as vendas estão indo. Como exemplo, vamos apresentar **Receita Mensal, Análise de Volume e Participação por Produto** e **Análise de Volume e Participação por Cliente** em uma mesma visualização.

Para tanto, vamos abrir novamente o Visual Studio e entrar na solução BI que havíamos criado. Ao lado direito, existe o `Gerenciador da Solução` (ou `Solution Explorer`) que é a árvore do que temos desenvolvido na solução que estamos criando. No caso do projeto de relatórios que estamos fazendo, existem 3 itens: fontes de dados compartilhadas, conjunto de dados compartilhados e relatórios.

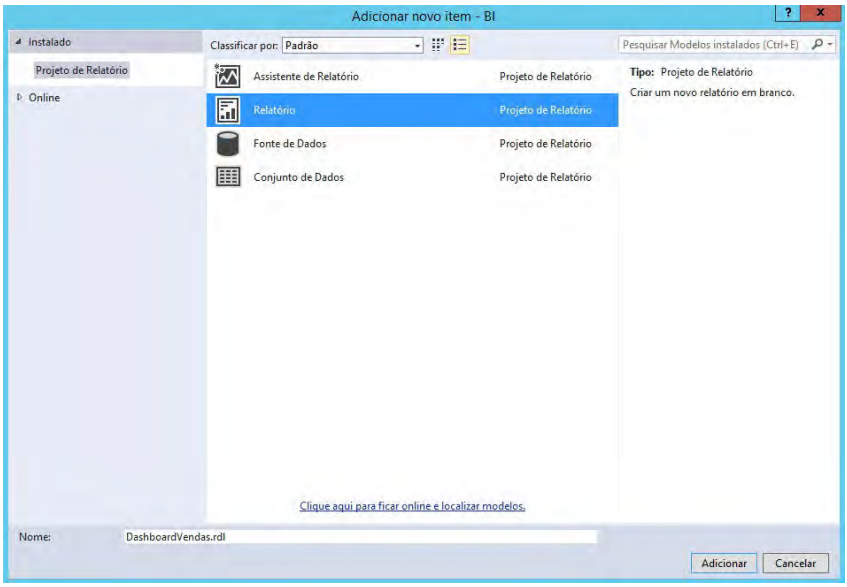
Clique com o botão direito em `Fonte de Dados Compartilhadas` e clique em `Adicionar Nova Fonte de Dados...` . Na janela que aparecerá, nomeie a fonte de dados como Datawarehouse (dado que é lá que vamos nos conectar) e em `Cadeia de Conexão` , clique em `Editar` . A nova janela solicitará o servidor, usuário, senha e qual o banco de dados. Preencha as informações com os dados de acesso ao nosso DW e clique em `OK` . Se estiver tudo certo, o resultado deverá ser:



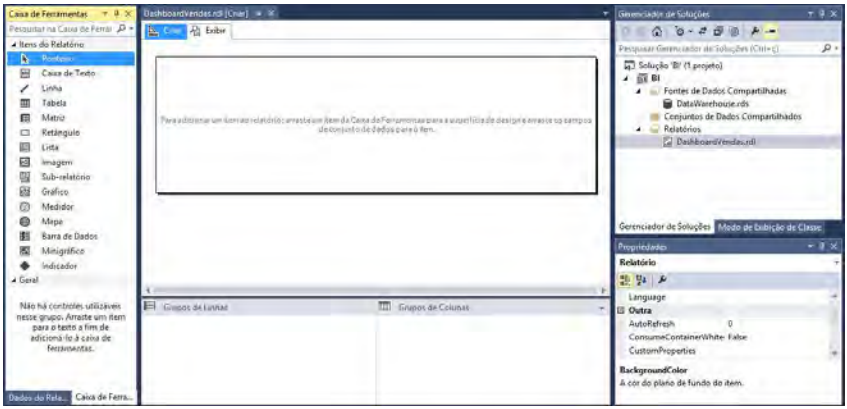
Se tiver dificuldade para lembrar do nome do servidor, basta ir ao Management Studio do SQL Server, abrir uma nova consulta e digitar `select @@SERVERNAME` . O resultado será o servidor a qual devemos conectar. O usuário e a senha são os mesmos que criamos para o acesso da nossa rotina diária em `.bat` , no capítulo anterior. Clicando em `OK` , a janela será fechada e a fonte de dados `DataWarehouse.rds` deverá ter aparecido.

Feito isso, temos agora uma fonte de dados ligada ao nosso DW

que poderá ser usada por todos os relatórios que desenvolveremos. Vamos então clicar com o botão direito em `Relatórios` e selecionar `Adicionar => Novo Item` (ou apenas o atalho `Crtl+Shift+A`). Na janela que aparecerá, selecione `Relatório` e nomeie de `DashboardVendas.rdl` :



Uma nova configuração de tela do Visual Studio aparecerá, com as ferramentas para o desenvolvimento de relatórios. Vamos discorrer um pouco sobre essa interface com a qual conviveremos bastante durante a vida de nosso BI (novos relatórios serão sempre solicitados!).



À direita, temos o já mencionado `Gerenciador de Soluções` que exibirá tudo o que está sendo construído. Por ele que criaremos novos itens (como acabamos de fazer), clicaremos duas vezes em itens existentes para editá-los etc.

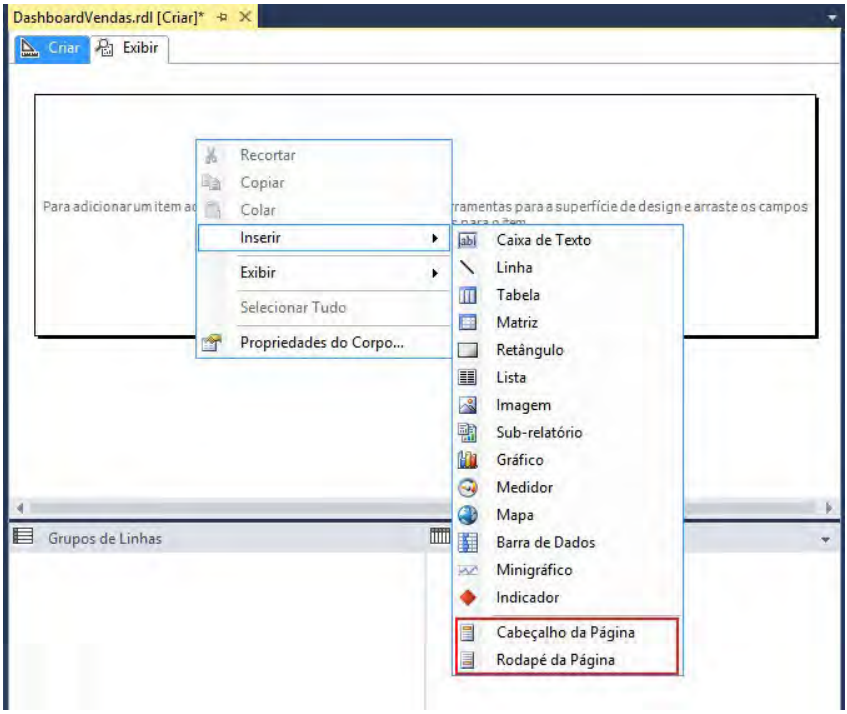
Logo abaixo dele, temos as `Propriedades`. Essa janela se modifica de acordo com o objeto que está selecionado, mostrando sempre as propriedades desse objeto.

À esquerda, temos a `Caixa de Ferramentas` que contém os componentes que podemos incluir em nosso relatório. Interessante mencionar que podemos instalar ferramentas novas (compradas ou gratuitas) caso necessário. Se notar bem abaixo, temos uma outra aba que é a `Dados do Relatório`, que fica alternada à `Caixa de Ferramentas`. Usaremos bastante essa aba para definir nosso conjunto de dados do relatório e seus respectivos campos!

Já ao centro, temos o `Canvas`, que é o nosso relatório propriamente dito. Ele se apresenta inicialmente no modo de edição. Mas se clicarmos em `Exibir`, teremos a execução dele tal qual o nosso usuário final terá. Vamos usar bastante esse recurso durante o desenvolvimento.

Primeiramente, vamos cuidar de deixar nosso dashboard

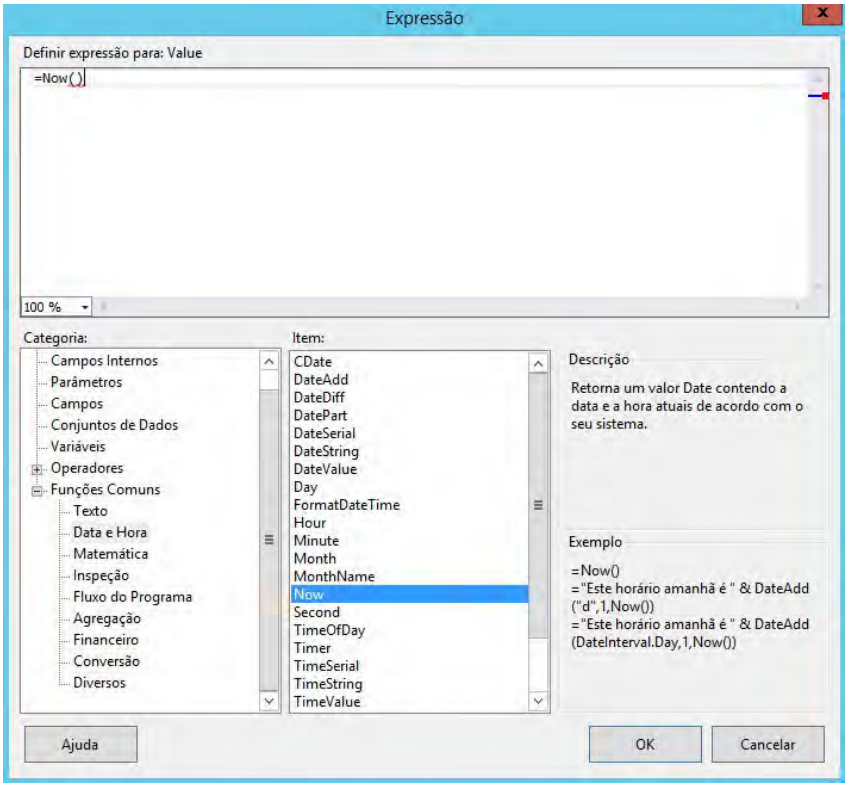
bonito! Vou colocar um cabeçalho com logotipo e um título e um rodapé com a data/hora. Para tanto, clique com o botão direito sobre o canvas e clique em `inserir`. As duas últimas opções são Cabeçalho e Rodapé. Insira os dois:



Clique com o botão direito no cabeçalho e clique novamente em `Inserir`. Escolha `Imagem` e busque pelo logotipo de sua empresa (que chamei de `Empresa` para nosso exemplo). Depois, na caixa de ferramentas, arraste uma `Caixa de Texto` para o cabeçalho e escreva nela o `Título Dashboard de Vendas`. Ajuste a fonte (tamanho, cor etc.) pelos botões do ribbon, ou pela caixa `Propriedades`.

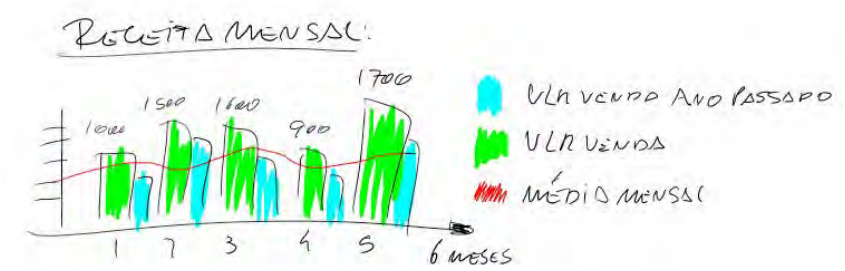
Para a data/hora do rodapé, usaremos uma fórmula! Arraste uma `Caixa de Texto` para o rodapé. Clique com o botão direito dentro da caixa e selecione `Expressão`. Na caixa que abrirá, na

parte esquerda, existe um grupo de funções chamado `Funções Comuns`. Expanda e localize `Data e Hora`. Na caixa do centro, selecione `Now`, clicando duas vezes nela. Na caixa superior, temos de ter a expressão `=Now()`. Clique em `OK`.



Cuide para centralizar os textos, colocar as caixas de texto nos lugares esteticamente apropriados (conforme os padrões de sua empresa, ou gosto dos seus usuários), usar a linha (da caixa de ferramenta) para separar as áreas etc. Se clicar em `Exibir`, poderemos ver nosso relatório com cabeçalho e rodapé, mas sem nenhuma informação ainda:

Feitas as firulas, vamos ao desenvolvimento! Antes de iniciar uma codificação, recomendo que se faça um desenho do que será o produto final, até para que seus usuários possam validar. Para esse primeiro gráfico, que é o de Receita Mensal, o esboço seria algo mais ou menos assim:



Será um gráfico de barras com o eixo X determinando os últimos seis meses, e o eixo Y será o de valor. Para cada mês, teremos duas barras: a do mês do ano atual, e a do mesmo período para o ano anterior (para comparação). Teremos também uma linha com a média para determinar uma visão da tendência.

A primeira coisa a ser desenvolvida deverá ser uma consulta no

SQL que traga as informações que precisamos:

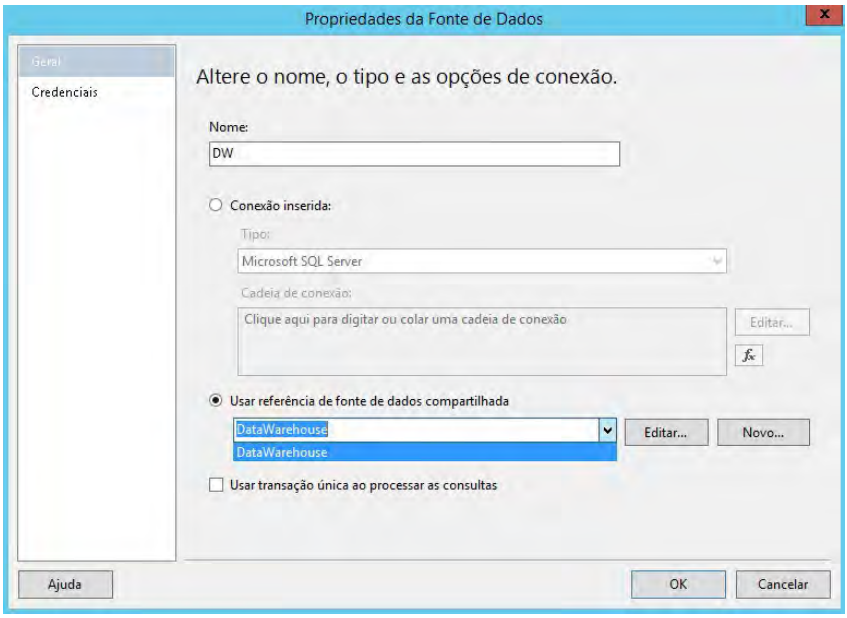
```
Declare @DataAtual Date = (Select max(data) from D_Data)
Declare @DataAnterior Date = (Select dateadd(year,-1,@DataAtual))


Select
Month(Data) as Mes,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAtual,
0 as VlrAnterior
from F_VendaDetalhe as a
where a.Data between dateadd(month,-5,@DataAtual) and @DataAtual
group by Month(Data)
union all
Select
Month(Data) as Mes,
0 as VlrAtual,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAnterior
from F_VendaDetalhe as a
where a.Data between dateadd(month,-5,@DataAnterior) and @DataAnte
rior
group by Month(Data)
```

Alguns pontos sobre esse comando:

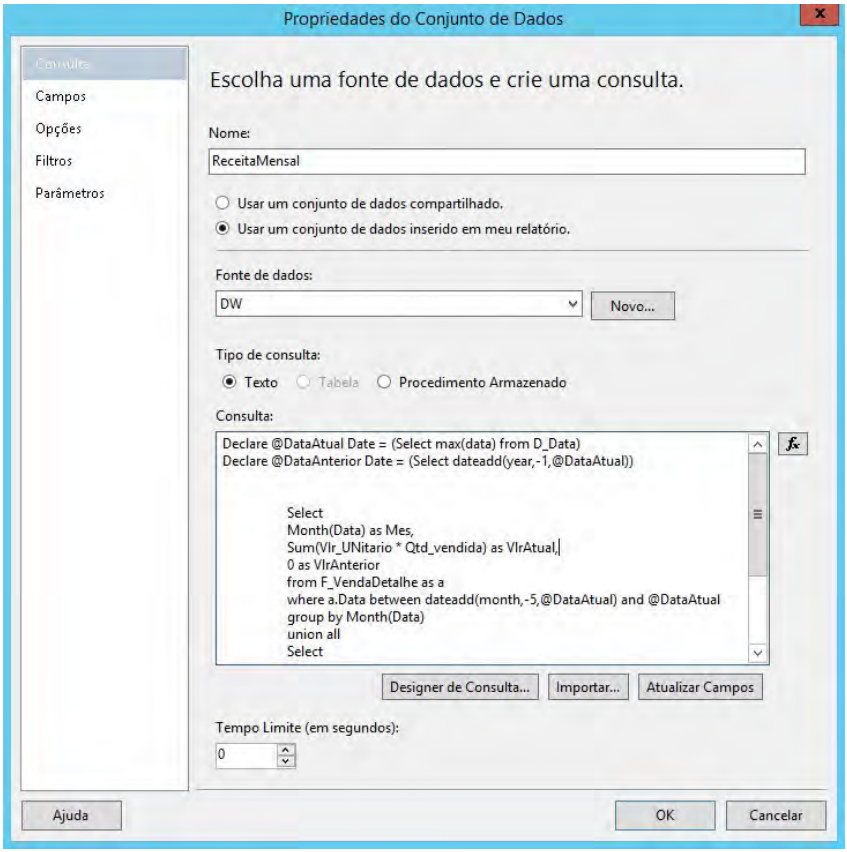
- Não usamos a data de referência como "hoje" e sim como a última data carregada no DW. Usualmente deverá corresponder à mesma coisa, mas essa prática evita termos colunas zeradas em viradas de ano ou de mês, se por acaso passarem alguns dias sem vendas.
- Note que esse comando não "junta" os dados de venda atual com venda do mês passado na mesma linha. Essa junção será feita pelo componente de gráfico do relatório. Se fossemos resolver pelo SQL, teríamos um `sub-select` e a performance poderia ser prejudicada.
- Não existe a média de cada mês! Essa coluna também será calculada pelo componente do gráfico do relatório. Da mesma forma que o ponto anterior, fazer o cálculo no relatório será uma operação bem mais rápida do que se fôssemos resolver em uma função AVG do SQL que

atuaria em todos os registros.

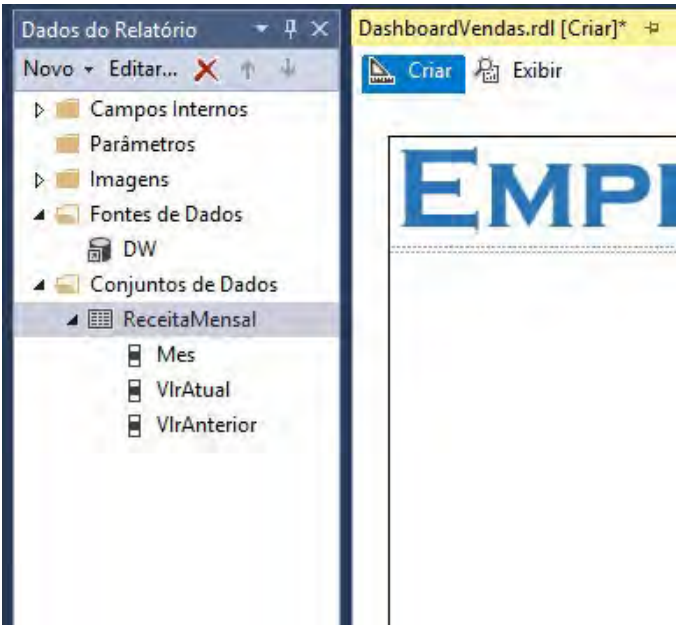
Voltando ao Visual Studio, selecione a aba `Dados do Relatório` e clique com o botão direito em `Fontes de Dados`. Em uma nova fonte, selecione aquela que havíamos criado, como fonte de dados compartilhada (chamada de Data Warehouse) e clique em `OK`:



Na mesma aba de `Dados do Relatório`, clique com o botão direito em `Conjunto de Dados` e vamos adicionar o comando que fizemos anteriormente. Dê o nome de `Receita Mensal` e cole nosso comando na área de texto:



Clicando em `OK`, teremos agora os campos disponíveis para serem inseridos em tabelas, gráficos etc.:



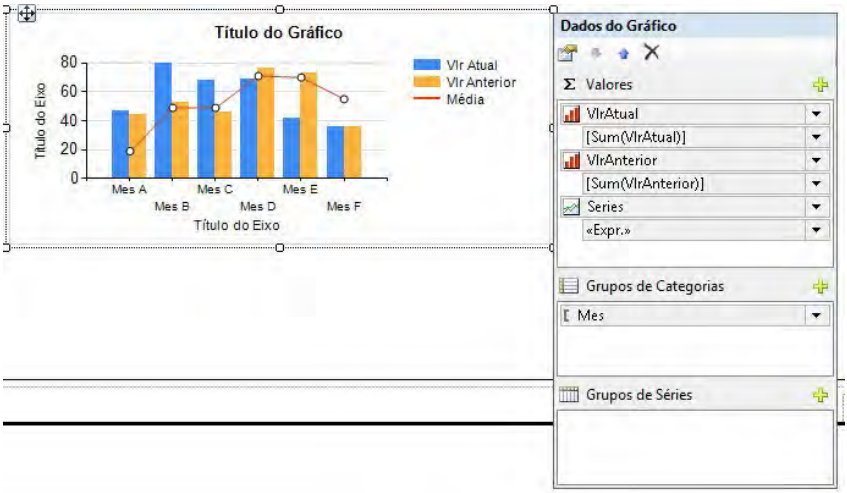
Já que temos os dados disponíveis, vamos colocá-los em um gráfico! Arraste um componente `Gráfico` da `Caixa de ferramentas` para o canvas, e selecione o primeiro tipo dos gráficos de coluna. Clicando sobre o gráfico, aparecerá um assistente de Dados do Gráfico. Arraste o campo `Mês` do nosso conjunto de dados para o campo `Grupo de Categorias` . Arraste os valores atual e anterior para o campo de `Valores` . Estaremos com algo parecido com o seguinte:

Ainda na área de Valores do assistente de Dados do Gráfico, clique no sinal de `+` (mais) em verde, e selecione `Expressão...`. Na caixa de expressões, digite:

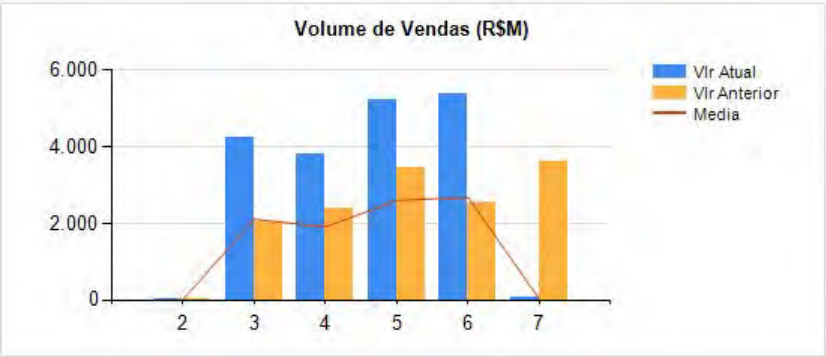
```
=(Fields!VlrAtual.Value+Fields!VlrAnterior.Value)/2
```

Aparecerá um conjunto de valor chamado `Series`, que será nossa média! Vamos alterar seu nome e o tipo de gráfico para transformá-lo em uma linha. Clique na seta que aponta para baixo ao lado da palavra `Series`, e selecione `Alterar o tipo de gráfico...`. Selecione o primeiro tipo de Linha.

Clique novamente na mesma seta e selecione `Propriedades da Serie`. Vá em `Legenda` e digite `Media` na caixa `Texto da Legenda Personalizada`. Teremos algo como:



Podemos excluir os títulos dos eixos e formatar o valor do eixo Y (clicando com o botão direito sobre ele e selecionando `Propriedades do eixo vertical`) para número sem casas decimais e com exibição em milhar (dividido por mil). Altere o título do gráfico para `Receita Mensal (R$M)` e pronto! Clique em `Exibir` para ver o resultado:



Temos nosso primeiro gráfico de dados que saíram de um sistema transacional, percorreram todo o ETL até o DW, foram selecionados em um conjunto de dados e apresentados em um

gráfico amigável e bastante significativo! Já de bate-pronto, notamos que:

- Temos um volume de venda maior nesse ano do que o ano anterior;
- O volume de vendas de fevereiro é bem baixo (foi assim no ano corrente e no anterior);
- Julho está ainda bem abaixo do ano passado, mas que tem uma tendência de passar o mês anterior.

Vamos criar agora o gráfico de Análise de Volume e Participação por Produto, que ficará junto com o Volume de Vendas no nosso dashboard. Novamente, faremos um esboço do que devemos criar para embasar nosso raciocínio:

Teremos então uma tabela com os dados do ano passado e do ano correte, tendo ao lado um gráfico de pizza com o percentual de participação de cada produto. Esses dados levarão em consideração os TOP 10 produtos pelo total de valor de vendas do ano anterior (para contemplar o efeito de vendas sazonais que ainda possam afetar os dados do ano corrente). Novamente, vamos começar criando o comando do SQL que trará as informações até o relatório. O comando será conforme a seguir:

```
Declare @AnoAtual char(4) = (Select year(max(data)) from D_Data)
```

```
Declare @AnoAnterior char(4) = (Select Year(dateadd(year,-1,@AnoAt
ual)))
declare @TotalVenda decimal(18,2) = (select Sum(Vlr_UNitario * Qtd
_vendida) from F_VendaDetalhe where year(Data) = @AnoAnterior)


Select top 10
b.Nome,
VlrAtual = (Select Sum(Vlr_UNitario * Qtd_vendida)
            from F_VendaDetalhe as x
            where x.Id_Produto = b.Id_Produto
            ),
Sum(Vlr_UNitario * Qtd_vendida)  as VlrAnterior,
Sum(Vlr_UNitario * Qtd_vendida)/@TotalVenda as percentual
from F_VendaDetalhe as a inner join D_Produto as b
on a.Id_Produto = b.Id_Produto
where year(a.Data) = @AnoAnterior
group by b.Id_produto, b.Nome
order by 3 desc
```

Vamos criar um novo conjunto de dados no relatório, tal qual fizemos para o primeiro gráfico, nomeando-o de `Participação Produto`, colocando nele o comando SQL anterior.

Como vamos querer a tabela e o gráfico de pizza alinhados, arraste a ferramenta `Retângulo` para dentro do canvas, e alinhe-o com o gráfico já existente. Expanda para que ele fique com a mesma área do gráfico de Volume de Vendas.

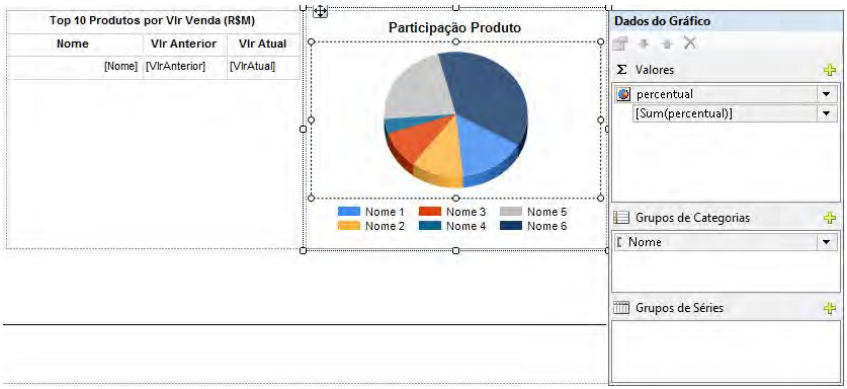
Depois, arraste a ferramenta `Tabela` para dentro do retângulo. Arraste os campos do `Participação Produto` para dentro dela, e formate os números e textos para que fiquem de acordo com a identidade visual que você quer dar.

Como apresentamos os números em milhar (divididos por mil) no primeiro gráfico, vamos manter o padrão na tabela também! A formatação é feita para cada célula, clicando-se com o botão direito sobre ela e selecionando `Propriedades da caixa de texto`.

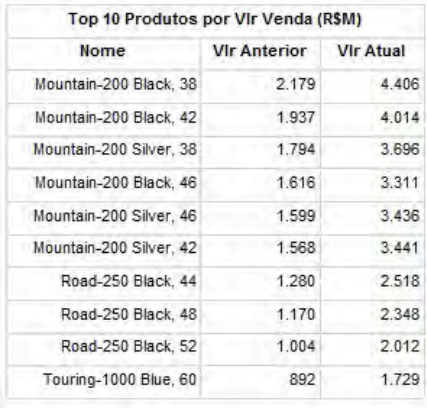
Para colocar um título na tabela, clique com o botão direito sobre a primeira linha e selecione `Inserir linha acima`.

Selecione as células dessa nova linha, clique com o botão direito e selecione `Mesclar células`. Nessa nova linha de uma única célula, digite `Top 10 Produtos por Vlr Venda`.

Feito o trabalho com a tabela, vamos inserir um gráfico, arrastando `Gráfico` para dentro do retângulo, ao lado da tabela. Tal qual fizemos no primeiro gráfico, insira o `Nome` na Série e o `Percentual` na área de Valores. Nomeie o gráfico e arraste a legenda para a parte de baixo. O resultado deve ser algo como:



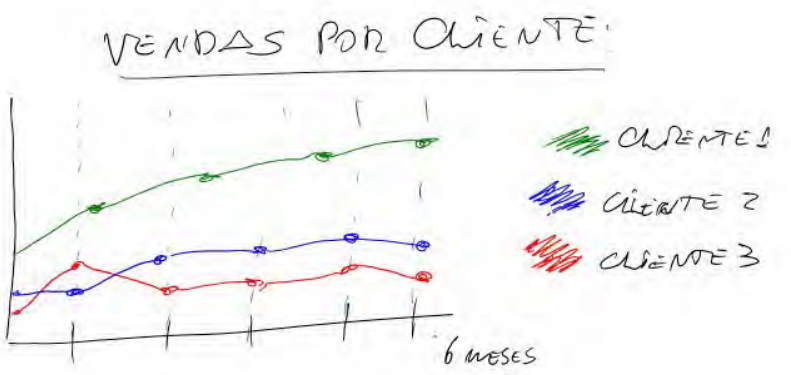
E executando o modo e `Exibir`, devemos ter algo como:



Top 10 Produtos por Vlr Venda (R$M)		
Nome	Vlr Anterior	Vlr Atual
Mountain-200 Black, 38	2.179	4.406
Mountain-200 Black, 42	1.937	4.014
Mountain-200 Silver, 38	1.794	3.696
Mountain-200 Black, 46	1.616	3.311
Mountain-200 Silver, 46	1.599	3.436
Mountain-200 Silver, 42	1.568	3.441
Road-250 Black, 44	1.280	2.518
Road-250 Black, 48	1.170	2.348
Road-250 Black, 52	1.004	2.012
Touring-1000 Blue, 60	892	1.729

Novamente, quando colocamos os dados em um dashboard, podemos entender quais são os produtos mais vendidos e as relações percentuais entre eles. Combinando com as vendas mensais do primeiro gráfico, podemos ter uma ideia de previsão de estoque, sazonalidade etc.

Para finalizar esse painel, vamos inserir a visão por cliente com a **Análise de Volume e Participação por Cliente**. A ideia é visualizar os principais clientes e suas participações mês a mês para definirmos as tendências (se um cliente está diminuindo suas compras, se outro está aumentando etc.). Temos a seguir um exemplo do que seria o esboço dessa análise:

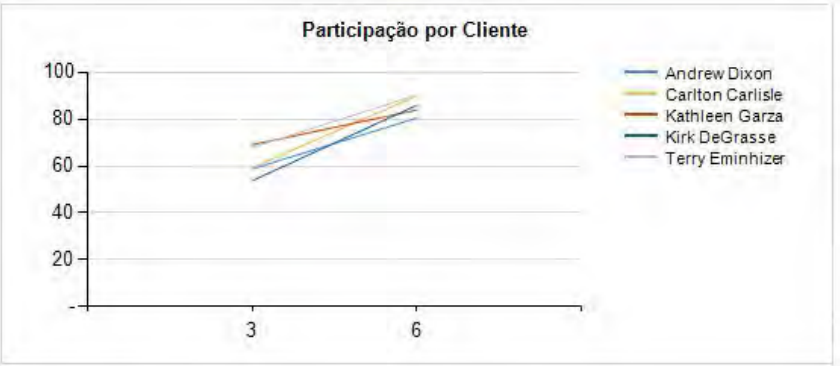


A ideia é mostrar os TOP 5 clientes e os volumes financeiros de cada um deles na evolução mês a mês. Para fazer essa busca, vamos precisar do seguinte comando no SQL:

```
Declare @DataAtual Date = (Select max(data) from D_Data)


Select
Month(a.Data) as Mes,
b.Nome,
Sum((a.Vlr_Unitario * a.Qtd_Vendida)) as Vlr
from F_VendaDetalhe as a inner join D_Cliente as b
on a.Id_Cliente = b.Id_Cliente
```

```
where a.Data between dateadd(month,-5,@DataAtual) and @DataAtual
  and a.Id_Cliente in (Select Id_cliente
                        from (
                            Select top 5
                            Id_cliente, Sum((Vlr_Unitario * Qtd_Ve
ndida)) as vlr
                            from F_VendaDetalhe
                            where  Data between dateadd(month,-5,@
DataAtual) and @DataAtual
                            Group by Id_cliente
                            order by 2 desc
                            ) as x)
group by Month(a.Data), b.Nome
order by 1,2,3
```

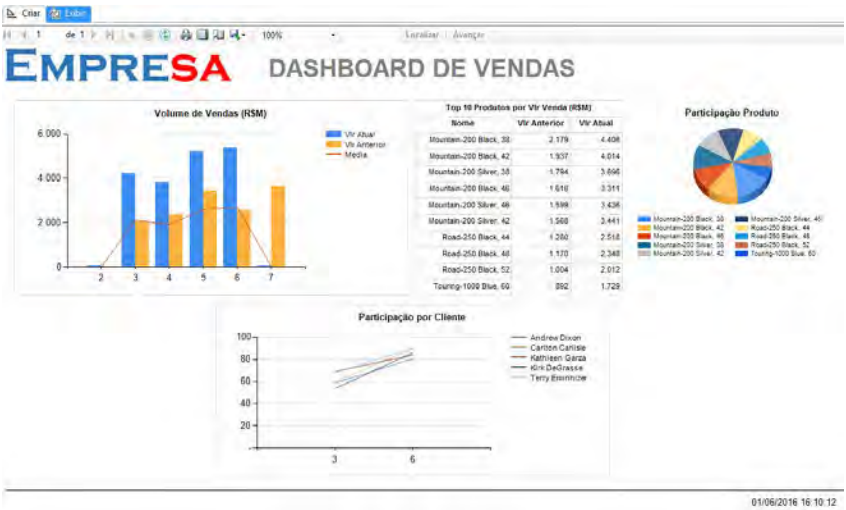
Fizemos então uma seleção de valor mensal de compra dos TOP 5 clientes dos últimos 5 meses. Se seguirmos os mesmos passos de criação de gráficos que já fizemos nos exemplos anteriores, teremos como resultado o seguinte gráfico de linhas:



Notamos que todos os clientes selecionados estão com uma tendência de alta, alguns mais agressivamente do que outros. Ainda assim, não tivemos compras deles nos meses 2, 4, 5 e 7! Essa situação pode denotar que um trabalho para aumentar a perenidade das compras pode ser uma estratégia comercial faltante, mesmo com os clientes que mais compram.

Executar nosso relatório em modo `Exibir` nos trará a visão de

como o produto final será visto por nossos usuários:



Uma boa prática nesse momento é apresentar esse relatório antes mesmo de publicá-lo a fim de ter o "aceite" dos usuários. Cores, títulos, disposições **sempre** sofrem alteração no momento em que se apresenta o produto final ao usuário. Estando tudo aprovado, vamos ao processo de publicação!

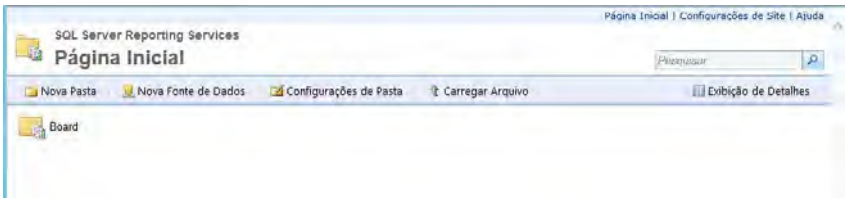
No segundo capítulo, quando instalamos o SQL e configuramos o Reporting Services, havíamos visitado o site de relatórios, que no nosso exemplo é `http://localhost/reports_BIPRD`. Se entrarmos novamente nesse endereço, veremos a página inicial do SSRS!

O que faremos é organizar nossos relatórios em pastas. Essa organização facilitará o processo de permissão. Uma boa prática é criar uma pasta para cada departamento, e nela colocar os relatórios que serão acessados pelos usuários daquela área. Uma pasta `Presidência` ou `Board` pode ser criada para um público que verá os dados mais sensíveis da empresa. Vamos usar esse exemplo e colocar o nosso dashboard comercial em uma pasta chamada

`Board`.

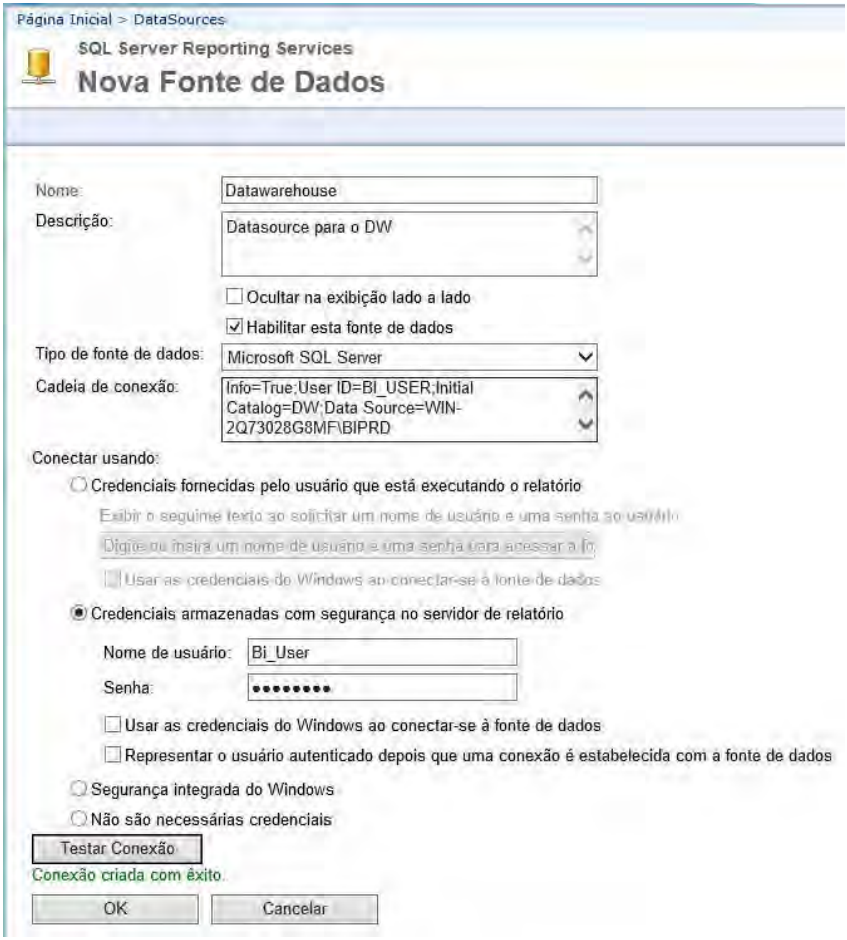
Antes de colocarmos nosso relatório no SSRS, temos de criar uma fonte de dados que será vinculada a ele. Como é um componente necessário apenas para o funcionamento dos relatórios, os usuários não precisarão visualizá-la. Clique em `Nova Pasta` e dê o nome de `DataSources`. Cheque a opção `Ocultar na visão lado a lado`. Isso fará com que ela seja ocultada para os usuários em geral.

Depois, crie outra pasta chamada `Board` e não selecione a opção de ocultar. Com as duas pastas criadas, nossa página inicial do SSRS estará assim:



Ao lado direito, existe o ícone `Exibição de Detalhes`. Clique nessa opção e veja que as duas pastas que criamos passaram a ser exibidas (uma abaixo da outra). Clique na pasta `Datasources`, e

depois em `Nova Fonte de Dados`. Vamos criar uma fonte de dados (tal qual fizemos no Visual Studio), preenchendo servidor, banco de dados, usuário e senha, conforme a seguir:

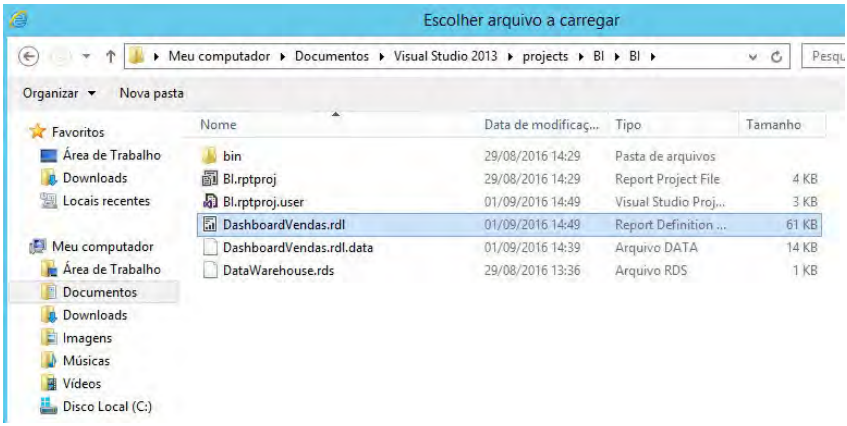


Clique em `Testar Conexão` e, tendo tido êxito, clique em `OK`. Volte para a página inicial do SSRS onde veremos a pasta `Board`. Clique nela e depois em `Carregar Arquivo`.

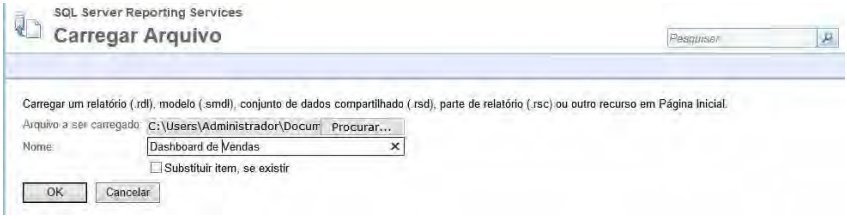
Clique em `Procurar...` para subir o arquivo do nosso dashboard. Para tanto, vá na pasta da solução do Visual Studio e

ache o arquivo `DashboardVendas.rdl`. Este é o que o Visual Studio cria com as definições de relatório e, carregando-o para o SSRS, teremos o relatório executado diretamente no browser!

Por padrão, o seus arquivos `.rdl` devem estar na pasta com o nome da solução dentro de `Documentos\Visual Studio 2013\projects\....`



Clicando em `abrir` ao selecionar o arquivo, retorna-se para a tela de upload. Você pode alterar o nome do arquivo para um nome mais amigável (colocando "de" entre Dashboard e Vendas, por exemplo):

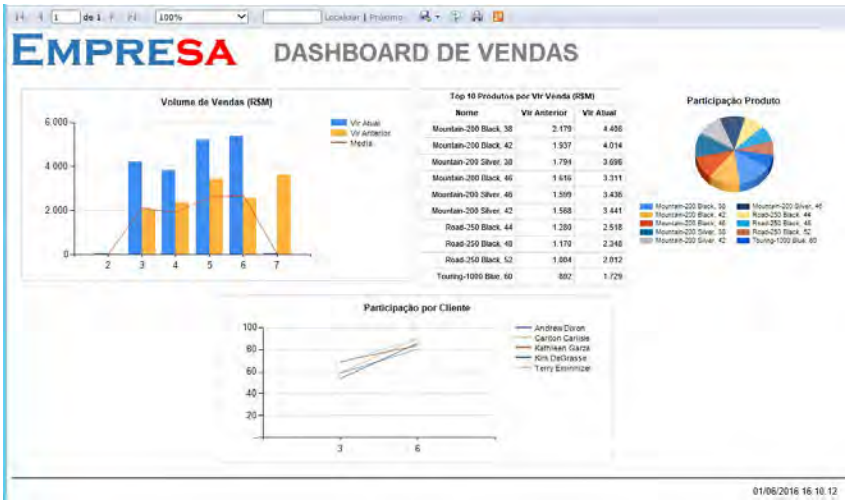


Clique em `OK` e o relatório estará publicado! Selecione a seta ao lado direito do relatório e clique em `Gerenciar`. Selecione a aba `Fonte de Dados` à esquerda e note a mensagem de erro: `A`

`referência da fonte de dados compartilhados não é mais válida` . Para corrigir esse problema, clique em `Procurar` e selecione a fonte de dados que criamos há pouco no SSRS:

Clique em `OK` e em `Aplicar` na tela anterior (não se esqueça de clicar no `Aplicar` !). Volte para a página inicial e clique novamente em `Board` para ver nosso relatório publicado:

Clique no relatório e ele será exibido tal qual foi construído no Visual Studio:



Bastará informar a URL do SSRS, ou colocá-la em um link na sua intranet, que seus usuários acessarão os relatórios que, por sua vez, estarão acessando os dados do DW que, novamente, serão carregados diariamente! Sem intervenção nenhuma, os dados passam das origens para os consumidores!

A dúvida que deve ter lhe acometido agora é quanto à segurança! Como garantir que somente um determinado público tenha acesso a cada uma das pastas? O SSRS funciona atribuindo-se papéis aos usuários (ou grupo de usuários) para cada pasta. Na aba de `Configuração do Site` (bem acima e à direita) e `Configuração da Página` (ao lado de `Subir Arquivo`), existem as opções de segurança nas quais se inserem os usuários e se define os papéis de cada um.

Um estudo detalhado do SSRS é muito recomendado nesse ponto não só para as questões de segurança e acesso, mas também para a criação de relatórios avançados, incorporação de ferramentas externas etc. Recomendo fortemente a leitura do livro *Pro SQL Server 2012 Reporting Services*. Ele trata da versão 2012 e Enterprise,

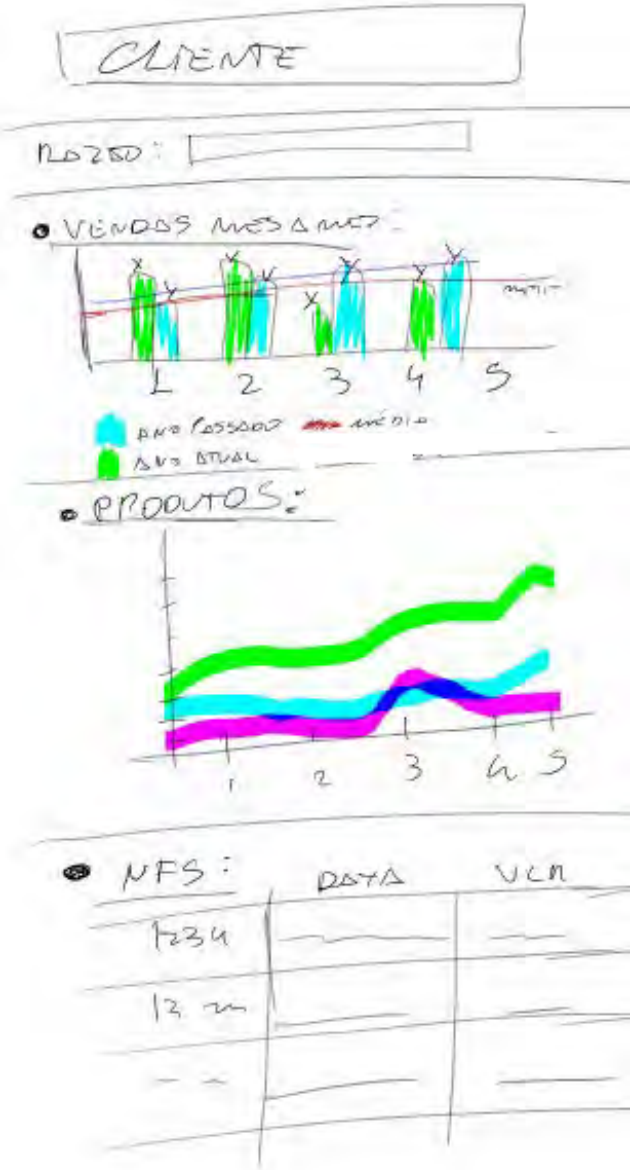
e nós estamos usando a 2014 Express. Mas não se preocupe. A diferença de ano não tornou o livro obsoleto. Ter em mãos um livro com as funcionalidades das versões completas do produto pode ser bastante útil! Claro que uma vasculhada nas lojas lhe renderá uma centena de outras opções de bons livros sobre o tema (quem sabe eu escreva um também ☺)!

Mas tal qual o T-SQL, não deixe de se aprofundar nos conhecimentos do SSRS!

5.4 CRIAÇÃO E PUBLICAÇÃO DO RELATÓRIO DE DETALHE

Fizemos um dashboard com o Reporting Services que será consumido pelo nível corporativo de consumo de dados, e agora vamos criar um relatório detalhado para atender ao nível departamental. Por definição, Business Intelligence serve à tomada de decisão e não seria focado em questões operacionais. Mas temos os dados todos no nosso DW, e dar suporte à operação se torna quase que um ganho colateral.

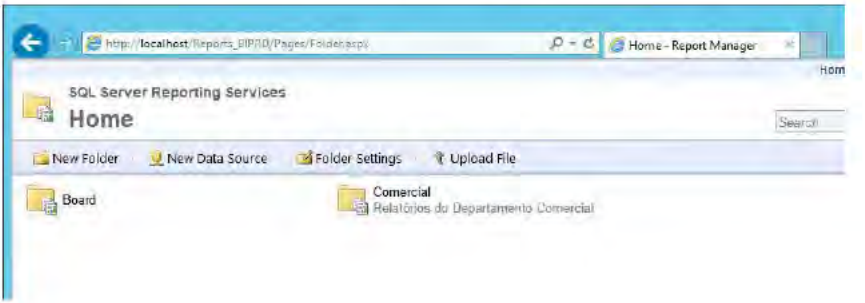
O exemplo seria um relatório de Posição do Cliente. Imagine que um vendedor vai efetuar uma visita comercial e quer saber o resumo do relacionamento desse cliente com a empresa. Nesse relatório, nosso vendedor escolherá o cliente e o relatório exibirá os dados conforme a seleção! Um esboço do nosso relatório seria algo assim:



O interessante desse exemplo é que se trata de um relatório para

o qual vamos passar parâmetros, ou seja, o resultado dele dependerá da seleção do cliente que queremos consultar. Extrapole esse exemplo para suas necessidades e quase qualquer consulta pode ser resolvida através da passagem de parâmetros!

Vamos iniciar esse desenvolvimento deixando nossa área de relatórios do Reporting Services preparada para receber o upload do `.rdl` que criaremos para o departamento comercial. Assim sendo, na página inicial do Reporting Services, crie uma pasta chamada `Comercial`. Ela ficará ao lado da pasta `Board`, já existente. Assim que tivermos o relatório pronto, será lá que o colocaremos:



Quanto ao desenvolvimento propriamente dito, começaremos clicando com o botão direito no item `Relatórios` do `Gerenciador de Soluções` e selecionar `Adicionar\Novo Item`. Selecione `Relatório` e nomeie como `PosicaoCliente.rdl` e, depois de criado, coloque o cabeçalho e o rodapé tal qual fizemos no dashboard.

Para criarmos esse relatório, vamos precisar dos seguintes conjuntos de dados distintos (e para cada um, um comando SQL):

- `ListaClientes`: esse conjunto de dados listará quais são os clientes disponíveis para nosso filtro! A seleção de dados será conforme a seguir (e deve ser inserida no campo de consulta do nosso conjunto de dados como

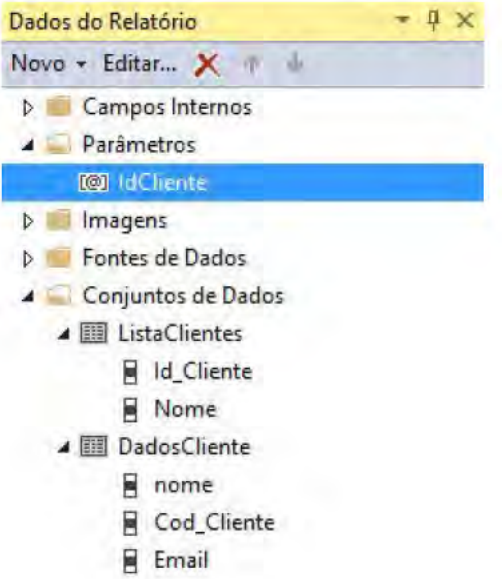
fizemos nos exemplos anteriores):

```
Select
Id_Cliente,
Nome
from D_Cliente
order by Nome
```

- `DadosCliente` : crie um novo conjunto de dados. Esse conjunto trará os dados do nosso cabeçalho. No exemplo, traremos nome, e-mail e o código pelo qual o cliente é conhecido. A seleção de dados será conforme:

```
Select
nome,
Cod_Cliente,
Email
from [dbo].[D_Cliente]
where Id_Cliente = @IdCliente
```

Note que, assim que o conjunto de dados for criado, um parâmetro chamado `idcliente` foi inserido automaticamente na aba de parâmetros!



Isso porque o Reporting Services entende que a variável `@IdCliente` deverá ser preenchida pelo usuário quando o relatório for solicitado e já cria o parâmetro. Vamos terminar de criar os conjuntos de dados e voltaremos para configurar esse nosso parâmetro!

- `VendasMensal` : esse novo conjunto responderá pelo gráfico de vendas mês a mês (e note que é um comando bastante similar ao que fizemos no nosso dashboard, apenas incluindo a separação pelo cliente em questão):

  ``` Declare @DataAtual Date = (Select max(data) from D_Data) Declare @DataAnterior Date = (Select dateadd(year,-1,@DataAtual))

```
Select
Month(Data) as Mes,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAtual,
0 as VlrAnterior
from [F_VendaDetalhe] as a inner join [D_Cliente] as b
on a.Id_Cliente = b.Id_Cliente
where a.Data between dateadd(month,-5,@DataAtual) and @DataAtual
and a.Id_Cliente = @IdCliente
group by Month(Data)
    union all
Select
Month(Data) as Mes,
0 as VlrAtual,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAnterior
from [F_VendaDetalhe] as a inner join [D_Cliente] as b
on a.Id_Cliente = b.Id_Cliente
where a.Data between dateadd(month,-5,@DataAnterior) and @DataAnte
rior
and a.Id_Cliente = @IdCliente
group by Month(Data)
```

- `ProdutosMensal` : esse novo conjunto de dados será responsável pelo gráfico de consumo dos produtos do nosso cliente em sua distribuição mensal:
```

```` Declare @AnoAtual char(4) = (Select year(max(data)) from D_Data)

```
Select
b.Nome,
month(a.Data) as Mes,
Sum(Vlr_UNitario * Qtd_vendida)  as VlrAtual
from F_VendaDetalhe as a inner join D_Produto as b
on a.Id_Produto = b.Id_Produto
where year(a.Data) = @AnoAtual
and a.Id_Cliente = @IdCliente
group by b.Nome,month(a.Data)
```
```

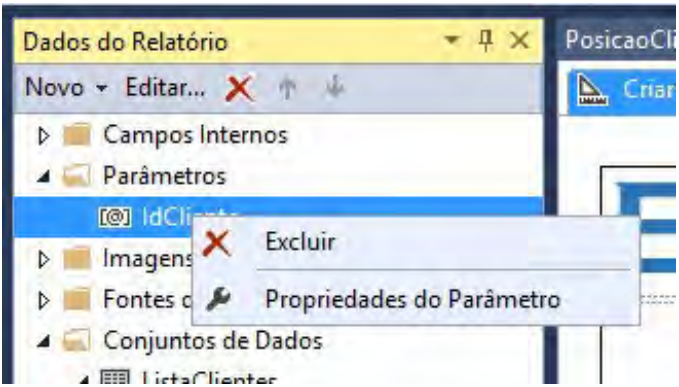
- `NotasFiscais` : esse novo conjunto de dados trará as últimas 10 compras feitas pelo cliente com os detalhes como número da nota, data da compra e valor de cada item, e preencherá os dados da tabela de NFs:

```
Select TOP 10
a.Nr_NF,
a.Data,
Sum(Vlr_UNitario * Qtd_vendida)  as VlrAtual
from F_Venda as a inner join F_VendaDetalhe as b
on a.Data              = b.Data              and
    a.Nr_NF              = b.Nr_NF              and
    a.Id_Cliente         = b.Id_Cliente         and
    a.Id_Funcionario = b.Id_Funcionario    and
    a.Id_RegiaoVendas = b.Id_RegiaoVendas
where a.Id_Cliente = @IdCliente
group by a.Nr_NF, a.Data
order by a.Data
```

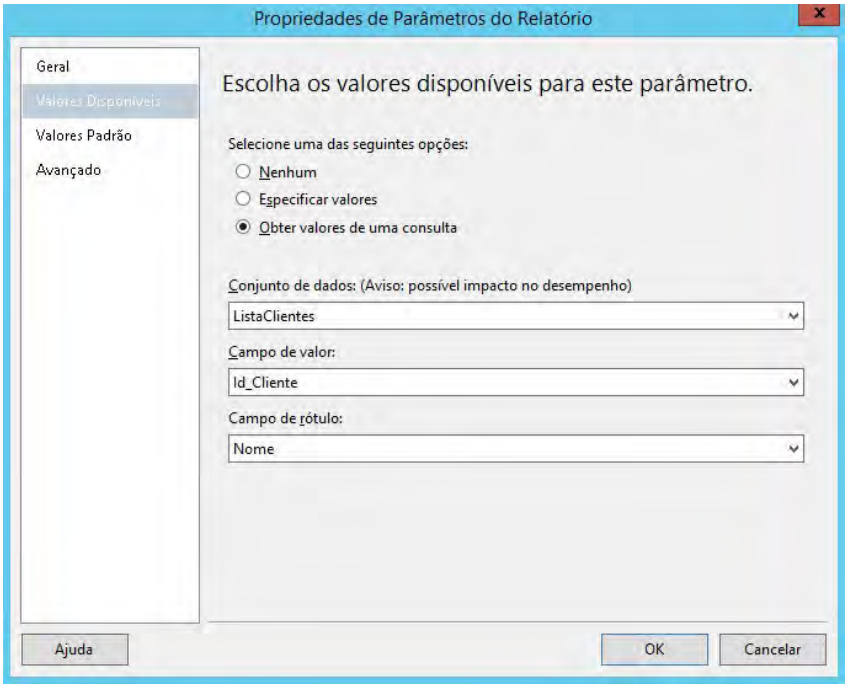
Feitos os conjuntos de dados, voltemos ao nosso parâmetro! Note que todos os conjuntos posteriores à lista de clientes possuem a mesma variável `@IdCliente` . Por isso mesmo (por termos mantido sempre o mesmo nome de variável), temos somente um parâmetro criado para nosso relatório. Se executarmos o relatório (clicando em `Exibir` ) sem nenhuma informação, já notaremos que esse parâmetro será solicitado antes de que o corpo do relatório seja exibido:

Vemos que o `Id Cliente` é solicitado e que tem-se o botão `Exibir Relatório` que fará a requisição. Como esse id é uma informação interna do nosso DW, o usuário não tem como preencher esse campo!

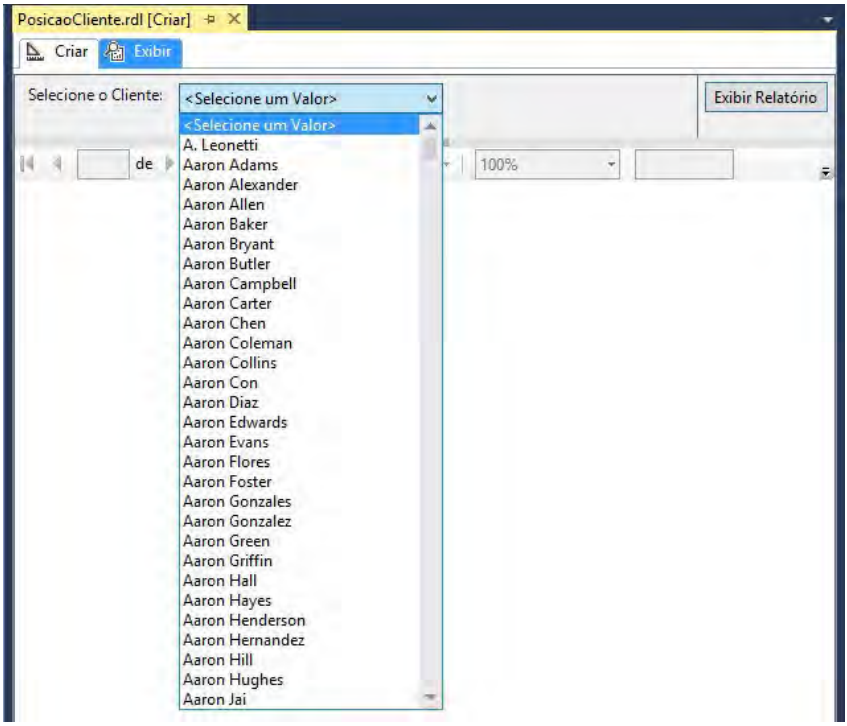
O que faremos agora será criar uma forma amigável de preenchimento desse dado! Volte ao modo de edição do relatório ( `Criar` ) e clique com o botão direito sobre o parâmetro e selecione `Propriedades do Parâmetro` :




Na aba `Geral` , vá para a caixa prompt (que é o texto que será exibido ao usuário), troque `Id Cliente` por `Selecione o Cliente:` . Selecione a aba `Valores Disponíveis` e preencha com os dados do conjunto de dados que criamos, chamado `ListaClientes` , conforme a figura demonstra:




Ou seja, teremos os valores obtidos de uma consulta que será a `ListaClientes`, cujo campo de valor (que será passado aos conjuntos de dados) será o `IdCliente` e o campo de rótulo (que será visualizado pelos usuários) será o `Nome`! Se executar novamente o relatório no modo de exibição, notará que o parâmetro já trará a lista dos clientes para serem selecionados.



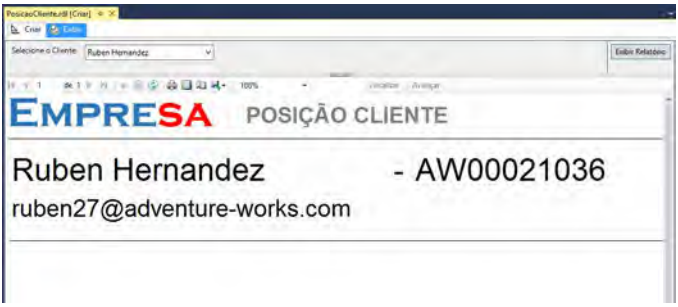

Ajustado o parâmetro, podemos criar nossos componentes gráficos para serem exibidos no relatório. Vamos iniciar colocando os dados do cliente como um cabeçalho! A ideia é deixar esses dados em destaque, com uma fonte um pouco maior para rápida identificação.

Para tanto, vamos arrastar o componente `Caixa de texto` três vezes para a área do relatório e atribuir a cada um deles os valores do conjunto de dados `DadosCliente`. Assim, basta clicar com o botão direito em cada caixa e selecionar `Expressão...`. Para cada uma, selecione os valores do conjunto de dados desejado com o comando:

```
=First(Fields!nome.Value, "DadosCliente")
```

Podemos ajustar a fonte, colocar uma linha acima e outra abaixo

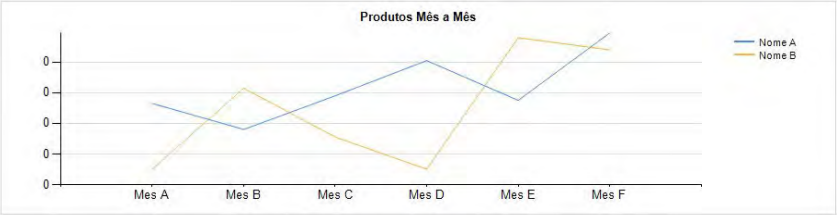
dos dados do cliente, enfim, deixar da forma que nossos usuários desejam! Feita essa parte simples, podemos executar o relatório pelo modo `Exibir` e selecionar um cliente qualquer no parâmetro. O relatório já trará os dados desse cliente:




Vale mencionar que qualquer informação relevante de Cliente que esteja disponível em sua empresa deve ser levada ao DW e, eventualmente, apresentada nesse relatório. É sempre útil incluir dados como endereço, telefones, cargo etc.

Avançando para o gráfico de vendas mês a mês, bastará recriarmos os passos do dashboard, arrastando o componente de gráfico para o relatório, arrastando os dados de Valores e Mês do conjunto de dados `VendasMensal` e criando o campo calculado de média. Se tiver alguma dúvida, retorne ao exemplo do dashborad. Nosso resultado ficará como o a seguir:

Arraste um outro componente de gráfico logo abaixo desse que fizemos e siga os mesmos passos de criação do gráfico anterior, mas agora com o conjunto de dados `Produtos Mensal` . Teremos um novo gráfico:




Vamos colocar uma tabela logo abaixo e vamos vinculá-la ao conjunto de dados de Notas Fiscais. Para tanto, arraste uma tabela para baixo do último gráfico. Por padrão, ela terá três colunas (exatamente o que necessitamos) e duas linhas, sendo que a primeira é um cabeçalho e a segunda conterá os detalhes, que no nosso caso será de 10 linhas.

Arraste os campos do nosso Conjunto de Dados de Notas Fiscais diretamente para a célula de Detalhe, primeiro o `NrNF` , depois a `Data` e, por último, o `VlrAtual` . Note que o cabeçalho recebeu esses nomes. Podemos editá-lo para nomes mais significativos, inclusive adicionando uma nova linha acima do cabeçalho (e fazendo o "merge" das colunas) para conter o nome da nossa tabela! O resultado ficará assim:

| Últimas 10 Vendas | | |
|---|---|---|
| Nota Fiscal | Data | Valor Total |
| 66392 | 18-03-2008 | R$4,99 |

Dessa forma, o relatório encontra-se finalizado e podemos executá-lo em modo de exibição para avaliar o resultado. Lembre-se de formatar os campos e alinhar os componentes todos, a fim de

tornar a exibição o mais agradável possível.

Uma vez estando tudo certo, vamos repetir o processo de upload do arquivo `.rdl` na pasta que criamos para o Comercial:

Executando esse relatório, o comportamento será ligeiramente diferente, porque, por ter a necessidade de parâmetros, teremos de selecionar o Cliente antes de termos a exibição de alguma informação:

Tendo selecionado um Cliente e clicando-se em `Exibir Relatório`, à direita, o relatório é exibido com os dados do Cliente selecionado:

## 5.5 CRIAÇÃO E DISPONIBILIZAÇÃO DA PLANILHA DE CONSULTA DINÂMICA

Fizemos a criação de alguns relatórios para disponibilizarmos para nossos diretores, gerentes comerciais e vendedores, atendendo às demandas corporativas e departamentais. Mas, muito provavelmente, esses relatórios não atenderão a todas as necessidades, e mais e mais solicitações de criação de novos relatórios serão recebidas pelo time do Business Intelligence.

Mesmo criando relatórios de forma muito rápida (como vimos, o Visual Studio nos permite criar e publicar relatórios no SSRS de

forma extremamente performática), teremos usuários que necessitarão de análises complexas, buscas de fenômenos de formas inesperadas e dinâmicas. Para esse público, o BI Pessoal, muitas vezes chamado de *self stracted BI* (ou ainda *self service BI*), é o que deve ser entregue.

Existem várias formas de distribuir o Business Intelligence Pessoal. Entretanto, vamos tratar por hora de uma das formas mais difundidas e versáteis: a consulta dinâmica, ou PowerPivot do Excel.

Para tanto, temos antes que disponibilizar uma fonte de dados para ser conectada à planilha e possibilitar o pivoteamento de dados. O comando a seguir cria uma `View` , ou seja, uma visão dos dados do nosso DW.

**Cuidado**

Faremos uma `View` com todos os dados do nosso DW, a fim de disponibilizar consultas sem nenhum tipo de limitação ao usuário final. Isso pode causar uma grande demora nas execuções e um tráfego de rede bastante pesado.

Se performance se tornar um problema, estará na hora de avaliar a implantação de estruturas multidimensionais (os cubos do SSAS, por exemplo) em seu ambiente, ou a inclusão de filtros que, por exemplo, disponibilizam apenas os dados do ano corrente para serem trabalhados, ou os dados já sumarizados.

Feita essa observação, vamos ao comando de criação da nossa `View` .

```
Create View Vw_AnaliseDW
```

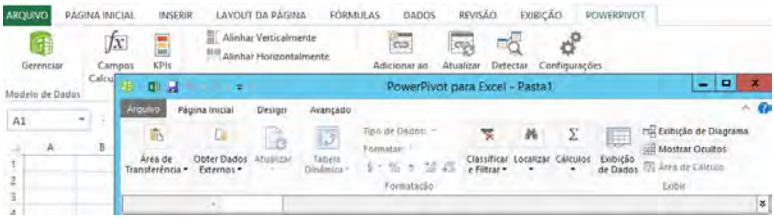
```
as
Select
a.Data,
c.Dia,
c.Mes,
c.Ano,
d.Cod_Cliente,
d.Nome as ClienteNome,
d.Email,
e.Nome as FuncNome,
e.Login,
f.Nome as RegiaoVendas,
g.Sigla as Pais,
h.Nome as GrupoGeo,
i.Nome as Produto,
i.Cod_Produto,
b.Vlr_Unitario,
b.Qtd_Vendida
From F_Venda as a inner join F_VendaDetalhe as b on
a.Data            = b.Data            and
a.Nr_NF            = b.Nr_NF            and
a.Id_Cliente    = b.Id_Cliente        and
a.Id_Funcionario =b.Id_Funcionario    and
a.Id_RegiaoVendas=b.Id_RegiaoVendas
inner join D_Data as c on
a.Data = c.Data
inner join D_Cliente as d on
a.Id_Cliente = d.Id_Cliente
inner join D_Funcionario as e on
a.Id_Funcionario = e.Id_Funcionario
inner join D_RegiaoVendas as f on
a.Id_RegiaoVendas = f.Id_RegiaoVendas
inner join D_Pais as g on
f.Id_Pais = g.Id_Pais
inner join D_GrupoGeografico as h
on g.Id_GrupoGeo = h.Id_GrupoGeo
inner join D_Produto as i on
b.Id_Produto = i.Id_Produto
Where i.Ativo = 1 -- Apenas para Produtos ativos
```

Criada a `View`, pode-se dar permissão pelo SQL Server por usuário (ou a grupos de usuários do AD) para que somente um determinado público tenha acesso a essa estrutura, bem como criar diversas views com filtros de dados na cláusula `where`, tendo uma para cada Região de Vendas, por exemplo, e para cada uma tem-se a permissão para um grupo do AD formado por vendedores apenas

daquela região. Dessa forma, implementa-se uma segurança ao nível dos dados que são apresentados!

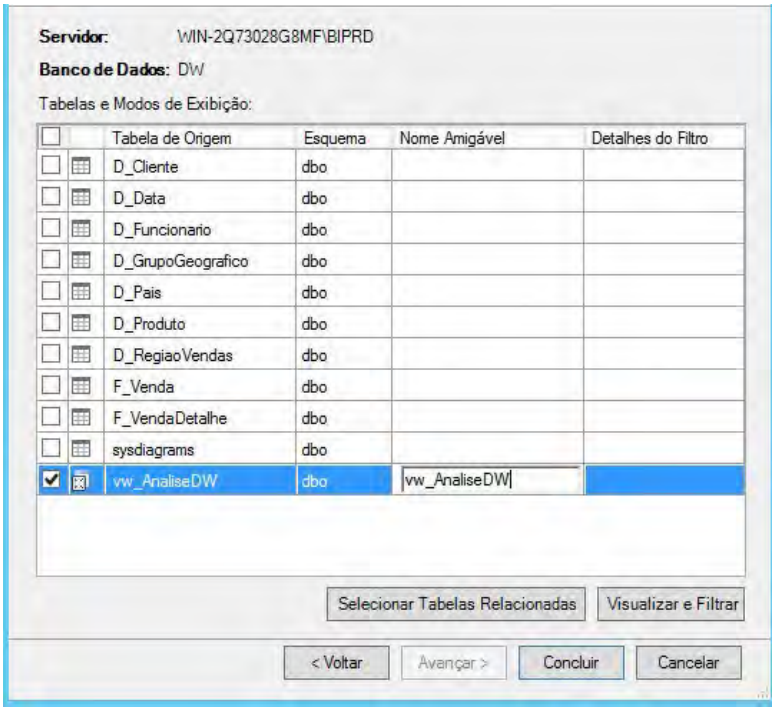
Tendo a nossa `View`, vamos vinculá-la a uma PowerPivot do Excel para podermos navegar pelos dados. Logo, seguiremos um passo a passo:

1. Abrindo o Excel, clique na aba PowerPivot (conforme configuração que fizemos anteriormente) e clique em `Gerenciar`. Será apresentada a janela de gerenciamento da PowerPivot:



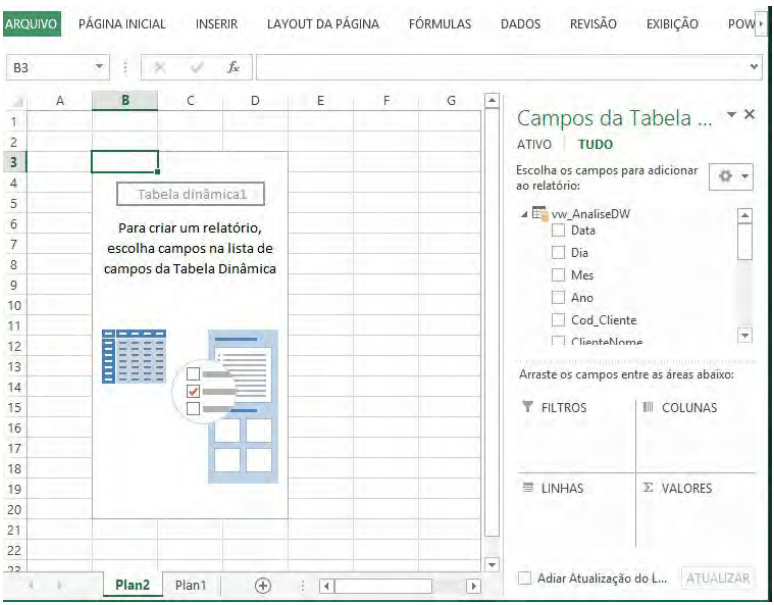

2. Clique em `Obter Dados Externos` (do Banco de Dados > do SQL Server) para criar uma conexão com o nosso DW. Renomeie a conexão para DW e aponte para nossa instância de SQL. Em segurança, use autenticação do Windows para garantir que somente usuários com as devidas permissões serão capazes de acessar os dados (se essa planilha for aberta por outro usuário, os dados não serão atualizados).

3. Clique em `Próximo` e mantenha checado o item `Selecionar itens em uma lista de tabelas e exibições para escolher os dados a serem importados`. Em `Próximo`, aparecerá a lista de todas as tabelas e visões disponíveis. Selecione a `vw_AnaliseDW` que fizemos anteriormente e selecione `Concluir`.

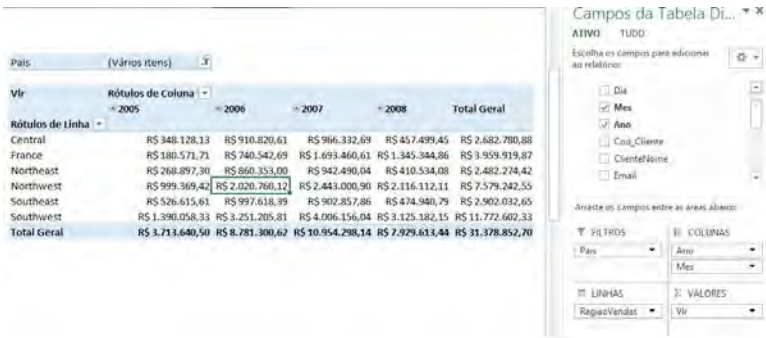



4. Assim que a importação terminar, clique em `Finalizar` e os dados serão exibidos em um preview. No Ribbon da Power Pivot, clique em `Tabela Dinâmica`. Você será automaticamente encaminhado novamente para a planilha do Excel, na qual poderá selecionar a célula na qual a tabela será incluída. Mantenha o padrão e clique em `OK`.

5. A área da tabela dinâmica será exibida à esquerda, e a lista de campos e áreas de soltura serão apresentados à direita:




6. Arrastando os campos disponíveis para Filtros, Colunas, Linhas e Valores, pode-se montar uma infinidade de combinações de visualizações:

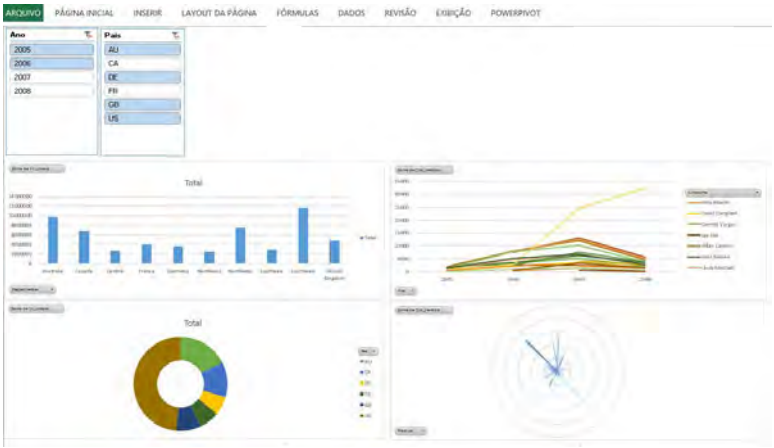


| Pais | (Vários itens) | | | | |
|---|---|---|---|---|---|
| Vlr | Rótulos de Coluna | | | | |
| Rótulos de Linha | 2005 | 2006 | 2007 | 2008 | Total Geral |
| Central | R$ 348.128,13 | R$ 910.820,61 | R$ 966.332,69 | R$ 457.499,45 | R$ 2.682.780,88 |
| France | R$ 180.571,71 | R$ 740.542,69 | R$ 1.693.460,61 | R$ 1.345.344,86 | R$ 3.959.919,87 |
| Northeast | R$ 268.897,30 | R$ 860.353,00 | R$ 942.490,04 | R$ 410.534,08 | R$ 2.482.274,42 |
| Northwest | R$ 999.369,42 | R$ 2.020.760,12 | R$ 2.443.000,90 | R$ 2.116.112,11 | R$ 7.579.242,55 |
| Southeast | R$ 526.615,61 | R$ 997.618,39 | R$ 902.857,86 | R$ 474.940,79 | R$ 2.902.032,65 |
| Southwest | R$ 1.390.058,33 | R$ 3.251.205,81 | R$ 4.006.156,04 | R$ 3.125.182,15 | R$ 11.772.602,33 |
| Total Geral | R$ 3.713.640,50 | R$ 8.781.300,62 | R$ 10.954.298,14 | R$ 7.929.613,44 | R$ 31.378.852,70 |



Nesse exemplo, selecionamos a comparação do Total de vendas das Regiões de Vendas da França e Estados Unidos, ano a ano. Poderíamos fazer qualquer cruzamento necessário

entre os dados disponíveis. Com um pouco mais de atuação com as opções da aba `Analisar` e `Design` da PowerPivot, um dashboard bem mais significativo pode ser feito com alguns cliques:




Utilizando-se dos comandos da PowerPivot, diversos dashboards podem ser montados, inclusive com `Segmentação de Dados` que são os filtros por seleção (no exemplo, usei Ano e País) que, ao serem selecionados, afetam todos os gráficos e tabelas da planilha.

Salvando a planilha, você poderá enviá-la para seu usuário, e este poderá trabalhar em suas próprias análises sem a necessidade de novos desenvolvimentos por parte da equipe de BI. Se ele receber uma mensagem de erro do SQL por falta de autorização, bastará criar um acesso para ele no Servidor (criação de um novo logon na aba de segurança do SQL Server) com permissão de leitura à view que criamos para esse fim!

Para maiores detalhes, tanto do processo de criação dos dahsboards e uso dos recursos da PowerPivot quanto do processo

de acesso ao SQL Server mediante logons de usuário e demais questões como o Power Query, o livro *Power Pivot and Power BI: The Excel User's Guide to DAX, Power Query, Power BI & Power Pivot in Excel 2010-2016*, de Rob Collie, é uma excelente fonte de conhecimentos!

## 5.6 CONCLUSÃO

Vimos nossos dados ganhando vida! De um sistema transacional, os dados passaram por um processo de carga e foram armazenados em um Data Warehouse. De lá, foram postos em perspectiva, mensurados, avaliados e comparados!

No próximo capítulo, veremos como estender um pouco mais as funcionalidades do nosso Business Intelligence sob a plataforma gratuita do SQL Server Express, a fim de garantir que seus investimentos em licenciamento só ocorram quando, de fato, forem imprescindíveis!

Capítulo 6

# ESTENDENDO A SUA PLATAFORMA GRATUITA

## 6.1 ALTERNATIVA PARA A LIMITAÇÃO DE TAMANHO DO DATA WAREHOUSE

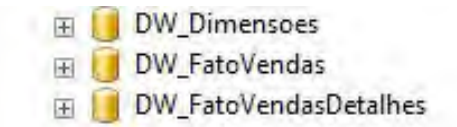
Seguindo a filosofia de conter ao máximo a necessidade de investimento em licenças, podemos dar uma sobrevida à nossa plataforma gratuita quando ela estiver crescendo em tamanho do DW e aproximando-se do limite do SQL Server 2014 Express, que é de 10GB por base de dados. Fato é que, quanto maior for nosso DW, maior será o consumo de memória para a execução das consultas e maiores serão as requisições de consultas feitas contra ele. Ou seja, o tamanho do DW será somente um dos problemas que a plataforma enfrentará em seu crescimento.

Os demais problemas não temos como contornar. Porém, este pode ser transposto com uma alternativa pouco ortodoxa, mas que surte o efeito necessário,

Crie suas tabelas de Dimensão em uma base de dados e as Fato em bases diferentes! Essa abordagem simples causará um impacto nos processos administrativos e nas consultas (os nomes das tabelas terão que ser trabalhados para incluir o banco de dados), mas que pode ajudar a manter a plataforma utilizável por mais tempo.

No exemplo do nosso Data Warehouse, utilizando essa

abordagem, teríamos algo como:




Os processos de carga fariam o insert do DS diretamente para a base DW correta, e as consultas teriam os nomes das tabelas completamente qualificados, ou seja, contendo o nome do banco, do owner, da tabela e da coluna. Algo como consulta de Volume de Vendas que fizemos para nosso dashboard:

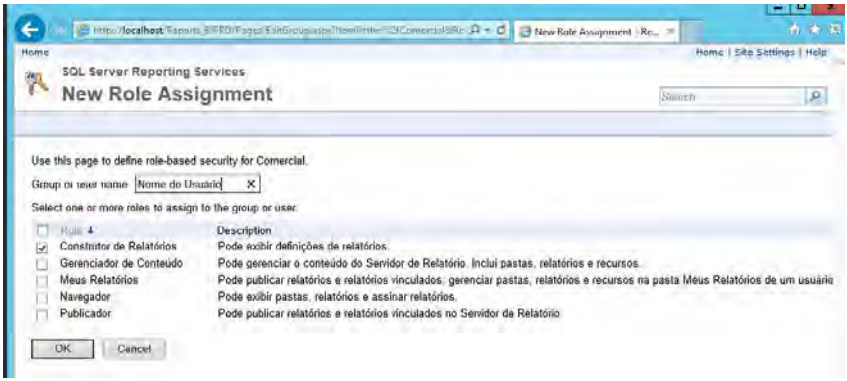
```
Select
Month(Data) as Mes,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAtual,
0 as VlrAnterior
from DW_FatoVendasDetalhes..F_VendaDetalhe as a
where a.Data between dateadd(month,-5,@DataAtual) and @DataAtual
group by Month(Data)
union all
Select
Month(Data) as Mes,
0 as VlrAtual,
Sum(Vlr_UNitario * Qtd_vendida) as VlrAnterior
from DW_FatoVendasDetalhes..F_VendaDetalhe as a
where a.Data between dateadd(month,-5,@DataAnterior) and @DataAnte
rior
group by Month(Data)
```

Note que, em vermelho, o nome da tabela passa a contar com o nome do banco e o "ponto, ponto" que indicam o owner padrão. Dessa forma, cada banco (ou seja, cada Fato) poderá ter 10GB de tamanho. Sua solução de BI ganhará uma sobrevida até que um licenciamento seja feito e um SQL Server full seja instalado, ou um SQL Azure contratado, o que permitirá o uso de bases de dados virtualmente ilimitadas.

## 6.2 REPORTING BUILDER

O Reporting Builder é uma ferramenta com a qual o usuário final desenvolve e publica seus próprios relatórios que ficarão disponíveis no SSRS. Deve-se dar a devida permissão para os usuários que farão a publicação por meio do Reporting Builder. Mas uma vez dada e de posse da ferramenta, pode-se dar aos *power users* a possibilidade de criar seus relatórios!

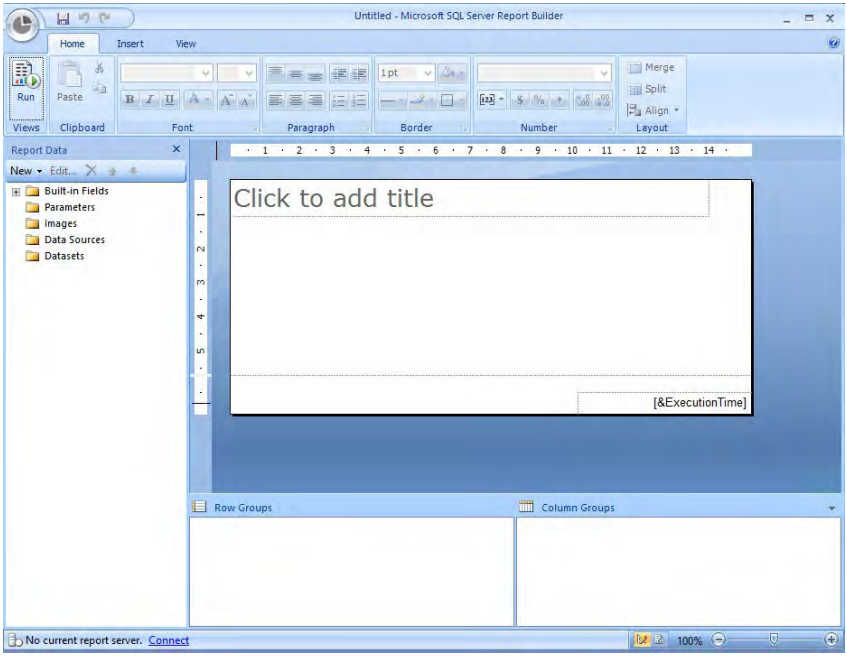
Para tanto, vamos criar uma pasta no servidor que contemplará os relatórios feitos pelos usuários dentro de cada uma das pastas departamentais. Na página do Reporting Services, abriremos a pasta `Comercial` e, dentro dela, criaremos uma nova pasta chamada `Relatórios de Usuários`. Todos os relatórios publicados pelos usuários ficarão ali. Será nessa pasta que daremos acesso aos usuários para a `role` chamada `Construtor de Relatórios`:




Para adquirir o Reporting Builder, basta fazer o download (gratuito) em https://www.microsoft.com/pt-br/download/details.aspx?id=42301. Arquivo baixado, basta fazer a instalação nas máquinas de quem usará o recurso. Essa instalação é feita de forma simples, bastando clicar duas vezes sobre o instalador, clicar em `Seguinte` e aceitar o contrato de licenciamento.

Depois disso, o instalador questionará sobre a URL de publicação, mas que podemos deixar em branco por hora. Clique

em `Seguinte` até o instalador iniciar a instalação e, ao final, abra o Reporting Builder:




O conceito é bastante parecido com o do Visual Studio. Possuímos `DataSources` que são as origens de dados, os `Datasets` que são os conjuntos de dados, e os parâmetros e as ferramentas que agora estão no ribbon acima. A interface é mais simplificada, mas pode-se criar relatórios bastante completos!

Contudo, antes de iniciar, note que, no rodapé esquerdo do Reporting Builder, existe a notação de que ele não está conectado a nenhum servidor de relatórios. Para conectá-lo, basta colar a URL do nosso Reporting Services, mas alterando a palavra `Reports` para `ReportServer`, dessa forma: `http://NomeDoSeuServidor/Reportserver_BIPRD`.

Se a conexão foi realizada, a informação terá alterado para:

Com essa interface, é possível criar os relatórios tal qual fizemos no Visual Studio. Contudo, o processo de publicação e as ferramentas de composição do relatório são mais simples e podem ser aprendidas por *power users* com muito menos esforço de treinamento que a versão completa do Visual Studio.

## 6.3 POWER BI DESKTOP

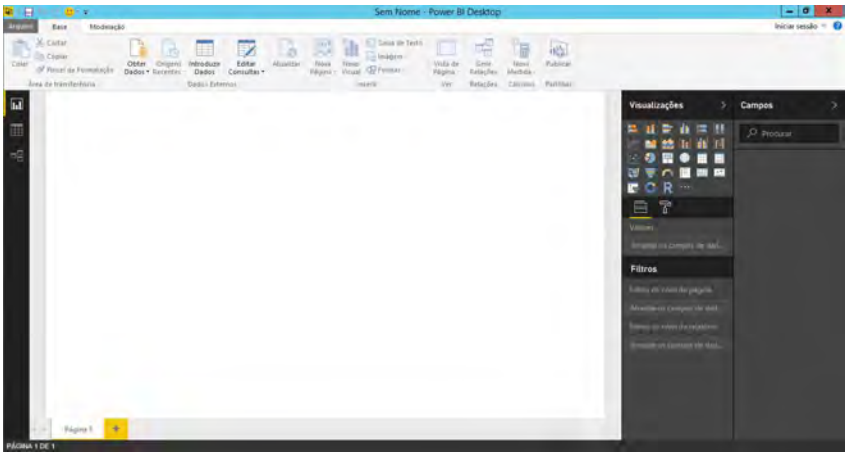
Dar mais e mais opções para que os usuários possam criar seus indicadores e suas análises é sempre uma boa forma de extrair ao máximo o que uma plataforma de BI pode oferecer. É bem verdade que os relatórios do Reporting Service não são simples para o usuário final, e que o Excel nem sempre oferece a melhor formatação de um dashboard!

Mas, pensando nesse gap, em 2016 a Microsoft lançou o Power BI com diversos recursos, diversas ferramentas e "espalhado" em diversas plataformas. Tem-se o Power BI como ferramenta em Cloud, como um app para celular e, como veremos, um aplicativo desktop de criação de Dashboard bastante interessante.

Pelo link https://powerbi.microsoft.com/pt-br/desktop/, você faz o download da ferramenta, bastando para isso preencher um cadastro com seu nome, e-mail e país. Feito o download, execute o instalador, aceitando o contrato de uso e selecionando o local de instalação.

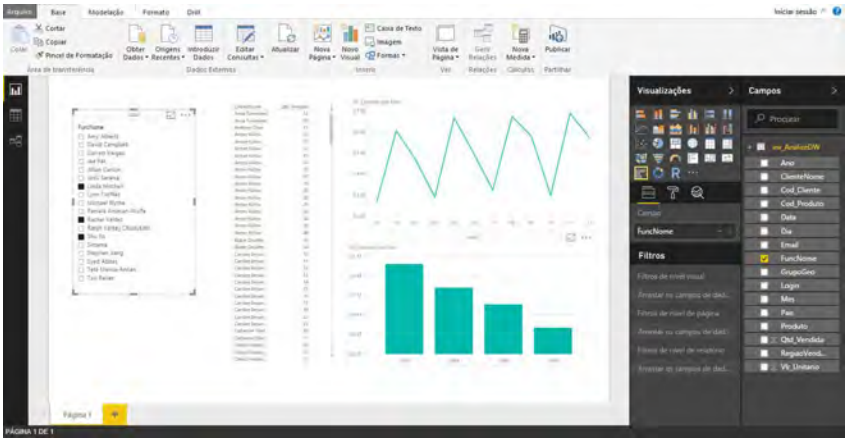
A partir daí, o Power BI Desktop estará instalado e pronto para o uso. Ah, como você notou, ele é 100% gratuito e poderá ser mantido em seu ambiente mesmo com outra plataforma que não a SQL Server (ele possui conector para dezenas de origens de dados, inclusive Facebook, SalesForce, SAP BW, e diversos outros)!

Executando o Power BI, a interface será a de um canvas em branco para a criação do seu dashboard, a caixa de ferramentas à direita e o ribbon acima com as opções contextualizadas.




Clicando no botão `Obter Dados` do ribbon, pode-se optar pelo SQL Server e fazer a conexão com o nosso DW (inserindo as informações de servidor e base de dados tal qual fizemos na Power Pivot).

Novamente, selecione a nossa view `vw_AnaliseDW` na lista de tabelas e visualizações que aparecerá e estaremos com o Power BI pronto para o uso! Usando as ferramentas de Visualizações e de Campos, é possível criar os gráficos e tabelas de forma simples e posicioná-los no canvas juntamente com filtros de página:




O Power BI pode ser usado offline, pois, ao criar um dashboard nele, é gerado um arquivo de extensão `.pbix` que pode ser reaberto, atualizado, enviado para outros usuários etc., bem como pode-se publicar no Portal do Power BI do Office 365, para acesso diretamente pelo browser. Entretanto, esse serviço é disponibilizado mediante a assinatura do 365.

## 6.4 CONCLUSÃO

Nossos dados saíram dos sistemas transacionais e foram postos no Data Warehouse! Deixaram de ser apenas dados e viraram **informação**.

Informação que pode guiar as decisões de uma empresa, definir as ações de uma equipe, ou revelar comportamentos inesperados que implicam em erros ou até mesmo em fraudes! A utilização dessa plataforma, aliada à experiência de negócio de cada usuário, passa a gerar **conhecimento**! Esse conhecimento, por sua vez, torna a empresa mais focada, mais competitiva, mais integrada, com menos perdas. O conhecimento vira **valor**!

E agregamos valor ao nosso negócio com um investimento

baixíssimo! Depois de discorrer sobre o passo a passo para implementar uma plataforma inteira de BI, sem a necessidade de licenciamentos milionários ou de "software assurance" anual, espero ter desfeito a visão de que Business Intelligence é para poucos, de que é caro ou complicado!

É certo que o seu BI passa agora a ser um organismo vivo que receberá mais e mais importância, e mais e mais usuários estarão dependendo dele para as tarefas do dia a dia. Mais dimensões, mais fatos, mais visões. As limitações do que apresentamos neste livro em breve se farão sentir!

O Data Warehouse começará a ficar grande e chegará ao limite máximo do SQL Server Express. O tempo de resposta dos relatórios poderá começar a crescer, funcionalidades adjacentes poderão se tornar relevantes, como exibições em dispositivos móveis, acessos por clientes que não estão na sua rede interna etc.

Em algum momento no futuro, sua plataforma poderá necessitar de um "upgrade" para contemplar funcionalidades ou capacidades que apenas softwares pagos poderão fornecer. Antes de decidir por embarcar nessa nova fase da jornada, estude (mesmo!) as necessidades dos seus usuários, as funcionalidades de cada ferramenta e os conceitos acadêmicos que suportam cada tecnologia!

Como quase todo o projeto de implantação de sistemas que presenciei ou tive conhecimento, o sucesso ou o fracasso da implantação, ou evolução de uma plataforma de um BI (e de outras disciplinas), está muito mais atrelado ao **time**, na competência e no comprometimento de cada um, do que nos softwares envolvidos.